



PostgreSQL PHP Generator

User's guide

Table of Contents

Foreword	0
I Welcome to PostgreSQL PHP Generator	1
1 System Requirements	2
2 Installation	3
3 How can I purchase PostgreSQL PHP Generator?	4
4 License Agreement	5
5 About SQL Maestro Group	7
6 What's new	10
II Getting started	11
1 How to connect to PostgreSQL	13
2 Projects	17
3 Command line options	18
4 Report sending	19
5 Shortcut keys	20
6 Deployment	21
III Configuring datasources	22
1 Tables and views	24
2 Custom SQL queries	25
IV Page Management	27
1 Root level pages	28
2 Detail pages	31
V Page Editor	33
1 Columns	35
Lookup settings	37
View controls	44
Text	47
DateTime	49
Checkbox	50
Toggle	51
File download	52
Image	53
External File	55
External Image	56
External Audio	57
External Video	58
Embedded Video	59
Barcode	60

QR Code	61
Edit controls	62
Text	66
AutoComplete	67
Suggestions	68
Radio group	69
Combobox.....	70
Dynamic Combobox.....	73
Cascading Combobox.....	75
Dynamic Cascading Combobox	79
Check box.....	85
Toggle	86
Check box group	87
Multiple select	89
DateTime	90
Time	91
Spin edit	92
Range edit.....	93
Color edit	94
Mask edit	95
Text area	96
Html Wysiwyg.....	97
Password.....	100
File upload.....	100
Image upload.....	101
Upload file to folder.....	103
Upload image to folder	104
Signature	106
Calculated Columns	107
2 String templates	110
3 Master-Detail Presentations	111
4 Events	117
Application-level (global) Events	117
OnBeforePageExecute	119
OnAddEnvironmentVariables	119
OnPreparePage.....	119
OnCustomHTMLHeader	120
OnGetCustomTemplate.....	121
OnBeforeInsertRecord.....	128
OnBeforeUpdateRecord	129
OnBeforeDeleteRecord.....	129
OnAfterInsertRecord.....	130
OnAfterUpdateRecord.....	131
OnAfterDeleteRecord.....	133
OnGetFieldValue.....	134
OnGetCustomExportOptions.....	134
OnCustomizePageList.....	136
OnGetCustomPagePermissions	136
Client Side Page Events	137
OnBeforePageLoad.....	138
OnAfterPageLoad.....	138
OnInsertFormValidate	139
OnEditFormValidate	139

OnInsertFormEditorValueChanged	140
OnEditFormEditorValueChanged	142
OnInsertFormLoaded	144
OnEditFormLoaded	145
OnCalculateControlValues	146
Server Side Page Events	147
OnBeforePageExecute	148
OnAddEnvironmentVariables	148
OnPreparePage	149
OnPageLoaded	150
OnCustomRenderColumn	151
OnCustomRenderPrintColumn	154
OnCustomRenderExportColumn	154
OnCustomHTMLHeader	154
OnExtendedCustomDrawRow	155
OnCustomRenderTotals	158
OnGetCustomTemplate	159
Common templates	167
Data Grid	168
Single Record View	169
Edit and Insert forms	170
Print	172
OnCustomDrawRow	174
OnAfterInsertRecord	175
OnAfterUpdateRecord	176
OnAfterDeleteRecord	177
OnBeforeInsertRecord	178
OnBeforeUpdateRecord	179
OnBeforeDeleteRecord	180
OnGetFieldValue	181
OnGetCustomExportOptions	181
OnPrepareFilterBuilder	183
OnPrepareColumnFilter	183
OnGetCustomFormLayout	184
OnGetCustomColumnGroup	189
OnCustomCompareValues	190
OnFileUpload	190
OnGetCustomPagePermissions	191
OnGetCustomRecordPermissions	192
OnCustomDefaultValues	193
OnCalculateFields	194
OnGetSelectionFilters	194
Using Variables	195
5 Filter	198
6 Charts	199
Data Query	199
Common options	201
Advanced options	201
7 Page Properties	203
Common properties	204
RSS options	208
Multi Upload	208
Abilities	212

Export and Print	213
Options	217
8 Data Partitioning	220
9 Data Validation	224
VI Project Options	226
1 Page	228
Abilities	230
Export and Print	232
2 Shared options	235
VII More customization options	238
1 Color schemes	241
2 Header and Footer	242
3 User-defined styles	243
4 User JavaScript	245
5 Custom templates	246
VIII Security settings	248
1 Hard-coded authorization	250
2 Table-based authorization	252
Custom Password Encryption	253
Email-based features	256
3 Database server authorization	258
4 User-defined authorization	259
5 Security Events	261
OnAfterLogin	262
OnAfterFailedLoginAttempt	262
OnBeforeLogout	263
OnVerifyPasswordStrength	264
OnBeforeUserRegistration event	265
OnAfterUserRegistration event	265
OnPasswordResetRequest event	265
OnPasswordResetComplete event	266
6 Google reCAPTCHA	267
7 Record-level security	268
8 Permission manager	269
IX Interface language	273
X Common generation options	275
XI Developer Reference	277
1 Client Side API	278
All editors	278
getValue	278

setValue	280
getEnabled	282
setEnabled	282
getReadonly	284
setReadonly	284
getVisible	284
setVisible	285
getRequired	285
setRequired	285
getState	285
setState	286
setHint	286
getFieldName	286
Text editor	288
getPlaceholder	288
setPlaceholder	288
Text area	289
getPlaceholder	289
setPlaceholder	289
Combobox	290
addItem	290
removeItem	292
getItemCount	294
getCaption	294
clear	294
Radio group	295
addItem	295
removeItem	296
getItemCount	298
getCaption	298
clear	298
Checkbox group and Multiple select	299
addItem	299
removeItem	300
getItemCount	301
clear	302
2 Server Side API	304
class Page	304
GetEnvVar	304
GetConnection	305
GetGrid	305
GetMasterGrid	305
GetCurrentUserId	305
GetCurrentUserName	306
IsLoggedInAsAdmin	306
IsCurrentUserLoggedIn	306
setShowGrid	306
setDescription	307
setDetailedDescription	308
SetTitle	309
SetInsertFromTitle	310
SetEditFromTitle	311
SetViewFromTitle	312
Print & Export enableity methods	313

class Application	314
IsGETValueSet.....	315
GetGETValue.....	315
IsPOSTValueSet.....	315
GetPOSTValue.....	316
IsLoggedInAsAdmin.....	316
IsCurrentUserLoggedIn.....	316
Session methods.....	316
GetEnvVar	317
class EngConnection	318
ExecScalarSQL	318
ExecSQL.....	318
ExecQueryToArray	318
GetLastInsertId	319
class Grid	319
Columns related methods	319
class PermissionSet	320
class PageList	321
addGroup.....	321
addGroupAt.....	321
addPage	321
Global functions	322
GetApplication	322
sendMailMessage.....	322
createConnection.....	323
3 Style sheets internals	325
4 Setting up common SMTP servers	326

XII Options 327

1 Application	328
Page	328
Export and Print	331
Abilities	334
Generation rules	335
Display formats	336
Output	337
Confirmations	338
2 Editors & Viewers	340
General	340
Display	341
SQL highlight	342
XML highlight	343
PHP highlight	344
Code Insight	345
Code Folding	346
3 Appearance	348
Bars and menus	348
Trees and lists	349
Edit controls	350
Check boxes	351
Buttons	352
Page controls	353

Group boxes	354
Splitters	355

Index	357
--------------	------------

1 Welcome to PostgreSQL PHP Generator

PostgreSQL PHP Generator is a tool for creating database-driven web applications visually. It allows you to generate professional quality PHP-based frontends for your PostgreSQL databases in a few minutes. You needn't have any programming background to use it.

Key features:

- 100% responsive design based on professional-quality page templates
- Support for [updatable SQL queries](#)^[25]
- [Master-detail presentations](#)^[111]
- [Out-of-the-box Charts](#)^[199]
- 25 color themes
- [Tabbed forms](#)^[184]
- [Event-driven content management](#)^[117]
- Client-side data validation
- [Custom pagination \(display partitioning\)](#)^[220]
- Data export to PDF, XML, CVS, Excel and Word
- Support for hard-coded, table-based, database server and custom [user authentication](#)^[248]
- Support for [Google reCAPTCHA](#)^[267]
- [Record-level security](#)^[268]
- [Flexible appearance customization](#)^[238]
- [Multi-language support](#)^[273]
- and much more.

Almost all features provided by PostgreSQL PHP Generator can be seen in action in our [Feature Demo](#), [Security Demo](#), and [other demo applications](#). We would highly recommend you to take a look at them as this can save you hours of work when developing your own websites.

1.1 System Requirements

Client environment

- Pentium PC or higher;
- Windows NT4/2000/XP/Vista/Windows 7/Windows 8/Windows 10/Windows 11;
- 512 MB RAM (1 GB recommended);
- 25 MB of free hard disk space;
- SVGA-compatible video adapter;
- Internet Explorer 9 or higher (to display the Preview page correctly);
- Microsoft .NET Framework 4.0 or higher (to compile styles).

Server environment

- PostgreSQL from 7.3 to 17;
- Linux/Unix or Windows Web Server;
- PHP 5.1 - 8.4.

1.2 Installation

To install **PostgreSQL PHP Generator** on your PC:

- download the PostgreSQL PHP Generator distribution package from the [download page](#) at our site;
- run setup.exe from the local folder and follow the instructions of the installation wizard;
- find the PostgreSQL PHP Generator shortcut in the corresponding program group of the Windows Start menu after the installation is completed.

1.3 How can I purchase PostgreSQL PHP Generator?

Thank you for your interest in purchasing **PostgreSQL PHP Generator Professional!**

You can select licensing options and register PostgreSQL PHP Generator at its [on-line order page](#). It is possible to purchase on-line, by fax, mail, toll-free phone call, or place a purchase order. We send the software activation key by email within 24 hours after completion of the order process. If you have not received the activation key within this period, please contact our [sales department](#).

All our products and bundles are shipped with 12 months of free upgrades (minor and major ones) or with 36 months of free upgrades for a quite small additional fee. After this period you may renew your license for the next 12(36) months with a 50% discount.

PostgreSQL PHP Generator has a free 30-day trial. Upon purchasing the product you confirm that you have tested it and you are completely satisfied with its current version.

To obtain technical support, please visit the [appropriate section](#) on our website or contact us by email to support@sqlmaestro.com.

1.4 License Agreement

Notice to users: carefully read the following legal agreement. The use of the software provided with this agreement (the "SOFTWARE") constitutes your acceptance of these terms. If you do not agree to the terms of this agreement, do not install and/or use this software. The use of this software is conditioned upon the user's compliance with the terms of this agreement.

- **License grant.** SQL Maestro Group grants you a license to use one copy of the version of this SOFTWARE on any single hardware product for as many licenses as you purchase. "You" means a company, an entity or an individual. "Use" means storing, loading, installing, executing or displaying the SOFTWARE. You may not modify the SOFTWARE or disable any licensing or control features of the SOFTWARE except as an intended part of the SOFTWARE's programming features. This license is not transferable to any other company, entity or individual. You may not publish any registration information (serial numbers, registration keys, etc.) or pass it to any other company, entity or individual.
- **Ownership.** The SOFTWARE is owned and copyrighted by [SQL Maestro Group](#). Your license confers no title or ownership of the SOFTWARE and should not be construed as a sale of any rights for the SOFTWARE.
- **Copyright.** The SOFTWARE is protected by the United States copyright law and international treaty provisions. You acknowledge that no title to the intellectual property in the SOFTWARE is transferred to you. You further acknowledge that title and full ownership rights to the SOFTWARE will remain the exclusive property of [SQL Maestro Group](#) and you will not acquire any rights to the SOFTWARE except as expressly set forth in this license. You agree that any copies of the SOFTWARE will contain the same proprietary notices which appear on and in the SOFTWARE.
- **License and distribution.** An unregistered copy of the SOFTWARE ("UNREGISTERED SOFTWARE") may be used for evaluation purposes. The UNREGISTERED SOFTWARE may be freely copied and distributed to other users for their evaluation. If you offer this UNREGISTERED SOFTWARE installation package for download, then you agree to:
 - replace existing version of the UNREGISTERED SOFTWARE installation package with the new package immediately after a new version of the SOFTWARE is released by SQL Maestro Group, or
 - delete an obsolete version of the UNREGISTERED SOFTWARE installation package immediately upon written email notice by SQL Maestro Group.

A registered copy of the SOFTWARE ("REGISTERED SOFTWARE") allows you to use the SOFTWARE only on a single computer and only by a single user at a time. If you wish to use the SOFTWARE for more than one user, you will need a separate license for each individual user. You are allowed to make one copy of the REGISTERED SOFTWARE for back-up purposes.

- **Reverse engineering.** You affirm that you will not attempt to reverse compile, modify, translate, or disassemble the SOFTWARE in whole or in part.
- **Unauthorized use.** You may not use, copy, rent, lease, sell, modify, decompile, disassemble, otherwise reverse engineer, or transfer the SOFTWARE except as provided in this agreement. Any such unauthorized use shall result in immediate and

automatic termination of this license.

- **No other warranties.** [SQL Maestro Group](#) does not warrant that the SOFTWARE is error-free. SQL Maestro Group disclaims all other warranties with respect to the SOFTWARE, either express or implied, including but not limited to implied warranties of merchantability, fitness for a particular purpose and noninfringement of third party rights. Some jurisdictions do not allow the exclusion of implied warranties or limitations on how long an implied warranty may last, or the exclusion or limitation of incidental or consequential damages, so the above given limitations or exclusions may not apply to you. This warranty gives you specific legal rights and you may also have other rights which vary from jurisdiction to jurisdiction.
- **Limited warranty.** This SOFTWARE is provided on an "AS IS" basis. [SQL Maestro Group](#) disclaims all warranties relating to this SOFTWARE, whether expressed or implied, including but not limited to any implied warranties of merchantability or fitness for a particular purpose. Neither SQL Maestro Group nor anyone else who has been involved in the creation, production, or delivery of this SOFTWARE shall be liable for any indirect, consequential, or incidental damages arising out of the use or inability to use such SOFTWARE, even if SQL Maestro Group has been advised of the possibility of such damages or claims. The person using the SOFTWARE bears all risk as to the quality and performance of the SOFTWARE.

Some jurisdictions do not allow limitation or exclusion of incidental or consequential damages, so the above given limitations or exclusion may not apply to you to the extent that liability is by law incapable of exclusion or restriction.

- **Severability.** In the event of invalidity of any provision of this license, the parties agree that such invalidity shall not affect the validity of the remaining portions of this license.
- **No liability for consequential damages.** In no event shall [SQL Maestro Group](#) or its suppliers be liable to you for any consequential, special, incidental or indirect damages of any kind arising out of the delivery, performance or use of the SOFTWARE, even if SQL Maestro Group has been advised of the possibility of such damages. In no event will SQL Maestro Group's liability for any claim, whether in contract, tort or any other theory of liability, exceed the license fee paid by you, if any.
- **Entire agreement.** This is the entire agreement between you and [SQL Maestro Group](#) which supersedes any prior agreement or understanding, whether written or oral, relating to the subject matter of this license.
- **Reserved rights.** All rights not expressly granted here are reserved to [SQL Maestro Group](#).

1.5 About SQL Maestro Group

SQL Maestro Group is a privately-held company producing high-quality software for database administrators and developers. The united team of eminently qualified developers is pleased to create new software products for commercial, academic and government customers worldwide. We do our best to design and develop products that remove complexity, improve productivity, compress time frames, and increase database performance and availability. We are glad to realize that our products take usual chores upon themselves, so that our customers could have more time left for their creative work.

The company was founded in 2002 as an essential partner for every business that is trying to harness the explosive growth in corporate data. SQL Maestro Group employs an international team concentrating their efforts on cutting-edge DBA tools development.

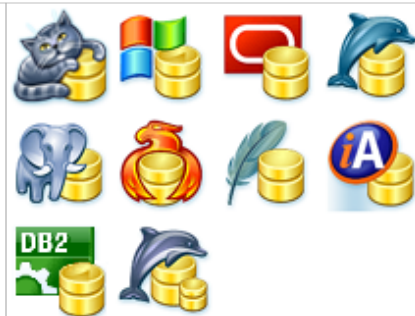
The slogan of our company is **The Shortest Path to SQL**. It is aimed to denote that we set to create easy-to-use products meant for those who appreciate comfort, friendly program interface and support when working with SQL servers.

- We are pleased to facilitate your job.
- We aim at being of considerable assistance to our clients.
- We feel contented doing our beloved work.

At present, our company offers a series of Windows GUI admin tools for SQL management, control and development of the following servers: **MySQL, Microsoft SQL Server, PostgreSQL, Oracle, SQL Anywhere, DB2, SQLite, Firebird, and MaxDB**. We also produce universal tools to be used for administering any database engine accessible via ODBC driver or OLE DB provider. Such products may be the clear-cut decision for those who constantly work with several database servers.

SQL Maestro is the premier Windows GUI admin tool for database development, management, and control.

It provides you with the ability to perform all the necessary database operations such as creating, editing, copying, extracting and dropping database objects; moreover, you can build queries visually, execute queries and SQL scripts, view and edit data including BLOBs, represent data as diagrams, export and import data to/from most popular file formats, manage users and their privileges (if possible), and use a lot of other tools designed for making your work with your server comfortable and efficient.



SQL PHP Generator is a powerful tool for creating database-driven web applications visually. It allows you to generate high-quality PHP scripts for working with tables, views and queries through the web. You needn't have any programming background to use it.



SQL Data Wizard is a high-capacity Windows GUI utility for managing your data.

It provides you with a number of easy-to-use wizards for performing the required data manipulation easily and quickly. The tool allows you to export data from PostgreSQL tables and queries to most popular formats, import data into the tables, generate SQL dump of selected tables, and export/import BLOB fields from/to files.



SQL Code Factory is a premier GUI tool aimed at the SQL queries and scripts development.

It allows you to manage SQL queries and scripts using such useful features as code folding, code completion and syntax highlighting, build query visually, execute several queries at a time, execute scripts from files, view and edit result data with filtering, sorting and grouping abilities, export data to as many as 14 file formats including Excel, RTF and HTML, import data from Excel, CSV, XML and text files, view and edit BLOBs in various way, build diagrams based on Oracle data, and much more.



Database Converter is a user friendly tool to migrate any local or remote ADO-compatible database to PostgreSQL .

Such tools transfer database schema and data and are equipped with native support for the most popular database servers.



Data Sync is a powerful and easy-to-use tool for database contents comparison and synchronization.

Such tools can be useful for database administrators, developers and testers that need a quick, easy and reliable way to compare and synchronize their data.



The software products are constantly optimized for the latest server versions support.

You can use the following contact information if necessary:

Our web-site www.sqlmaestro.com

Postal address: **SQL Maestro Group**
140 Broadway, Suite 706
New York City, New York 10005
United States

Thank you for your interest to our company!

1.6 What's new

Please find out the latest PostgreSQL PHP Generator news at <http://www.sqlmaestro.com/products/postgresql/phpgenerator/news/>

2 Getting started

In general, all you need to create your own feature rich data-driven web application is to complete the following simple steps (in fact, only first two steps are mandatory):

- [Connect to the database](#)^[13] you want to work with through the web;
- [Specify datasources](#)^[22] for web pages;
- [Customize web pages](#)^[27] to be created;
- Set [additional_generation_options](#)^[275] such as [webpages_appearance](#)^[238] and [interface language](#)^[273];
- Specify [security settings](#)^[248] to protect your data from an unauthorized access.

The screenshot shows a 'Connection properties' dialog box with two main sections: 'Connection properties' and 'Script connection properties'.

Connection properties:

- Radio button: ☒ I can connect to the server directly or via SSH tunneling. Below it is a link: [Configure SSH options](#).
- Radio button: ☐ I have to use HTTP tunneling. Below it is a text field for 'URL'.
- Links: [Configure tunneling options](#) and [Test script using default browser](#).
- Fields: 'Host' (sun), 'Port number' (5432), 'User name' (postgres), 'Password' (empty), 'Database name' (pagila).

Script connection properties:

- Checkbox: ☒ Use the same connection options.
- Fields: 'Host' (sun), 'Port number' (5432), 'User name' (postgres), 'Password' (empty), 'Database name' (pagila).

PostgreSQL PHP Generator allows you to save and restore all the options set during a session. All the session parameters may be saved and loaded to/from a [project](#)^[17]. Loaded settings may be edited if necessary. To open an existing project, click [More... > Load Project](#) on the first wizard step and select the project file or choose one of recently used projects.

See also: [Working with projects](#)¹⁷.

2.1 How to connect to PostgreSQL

Connection properties

Specify the connection parameters for a database you want PostgreSQL PHP Generator to work with.

Script connection properties

These parameters will be used by the generated web application. By default they are the same as parameters used by PostgreSQL PHP Generator but you can change them if necessary. For example, if you are working with a remote database located at your web hosting and your database server and web server are installed on the same computer, you have to specify the value of the Host parameter as [localhost](#).

PostgreSQL PHP Generator allows you to connect to PostgreSQL directly, or via Secure SHell (SSH) tunnel, or HTTP tunnel.

- **Direct connection**

It is the most natural and the most preferable connection mode. Use it each time it is possible.

- **SSH tunnel connection**

If your PostgreSQL server does not allow direct connections from your remote workstations, you can establish connection to an allowed intermediate SSH server and forward all PostgreSQL commands through the [Secure SHell \(SSH\) tunnel](#).

- **HTTP tunnel connection**

[HTTP tunneling](#) is a technique used in conditions of restricted network connectivity including firewalled networks, networks behind proxy servers, and NATs. It is the slowest way and is recommended to use if the others are impossible.

Irrespectively of a connection mode you should specify common credentials as follows:

Host

The host name of the PostgreSQL server.

Port number

The TCP/IP port to use. Default PostgreSQL port is 5432.

User name

The username used to connect to PostgreSQL.

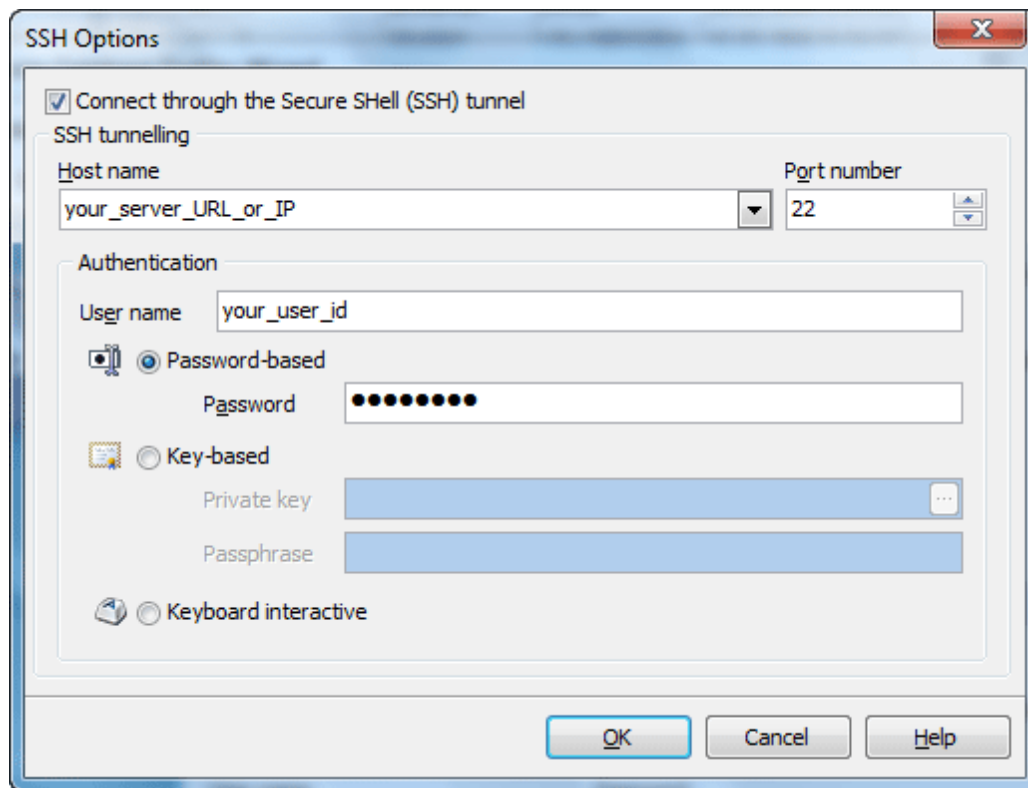
Password

The password for the user account on server.

More about SSH tunnel connection

To establish connection to intermediate SSH server and forward all PostgreSQL commands through the secure tunnel, you need to:

1. Check [I can connect to the server directly or via SSH tunneling](#).
2. Follow the [Configure SSH options](#) link to open the [SSH Options](#) window.



3. Check [Connect through the Secure Shell \(SSH\) tunnel](#) and complete the following fields:

Host name

Specify the host name or IP of your site. Note, that PostgreSQL host name always should be set relatively to the SSH server. For example, if both of PostgreSQL and SSH servers are located on the same computer, you should specify localhost as Host name instead of server's external host name or IP address.

Port number

Enter the port number for the SSH server.

4. Enter valid [User name](#) for the remote server and select the [Authentication](#) method and set corresponding credentials.

Password-based

Set the [password](#) corresponding to the specified user.

Key-based

Specify the path to the [Private key](#) file with the corresponding [Passphrase](#) to log in to the remote server. PostgreSQL PHP Generator accepts keys in **ssh.com** or **OpenSSH** formats. To convert a private key from PuTTY's format to one of the formats supported by our software, [use the PuTTYgen utility](#) that can be freely downloaded from the [PuTTY website](#).

Keyboard interactive

Keyboard authentication is the advanced form of password authentication, aimed specifically at the human operator as a client. During keyboard authentication zero or more prompts (questions) is presented to the user. The user should give the answer to each prompt (question). The number and contents of the questions are virtually not limited, so certain types of automated logins are also possible.

More about connection via HTTP tunnel

To connect to a remote server using an HTTP tunnel, you need to:

1. Upload the connection PHP script to your website. The script is named *pgsql_tunnel.php* and can be found under the installation folder, usually *C:\Program Files\SQL Maestro Group\PostgreSQL PHP Generator*.
2. Select the [I have to use HTTP tunneling](#) radio button.
3. Enter the connection PHP script URL, e.g. *www.yoursite.com/files/pgsql_tunnel.php*. You can test the connection before the profile is created. Just use [Test script using default browser](#) to open connection script in your browser, enter all the required connection parameters and click the [Test connection](#) button.

Connection Script

Fields marked by * are required.

Host/Server name (or IP) *:	<input type="text" value="neptun"/>
User *:	<input type="text" value="postgres"/>
Password:	<input type="password" value="••••••••"/>
Port (if not 5432):	<input type="text" value="5433"/>
Database *:	<input type="text" value="adventure"/> <input type="button" value="v"/>
	<input type="button" value="Get Database List"/>
	<input type="button" value="Test Connection"/>
	<input type="button" value="ShowTables"/>

© 2002-2008 SQL Maestro Group

4. In case using of a proxy server use [Configure tunnelling options](#) to open the [HTTP tunnelling options](#) window and specify your [proxy server](#) connection parameters and [HTTP authentication](#).

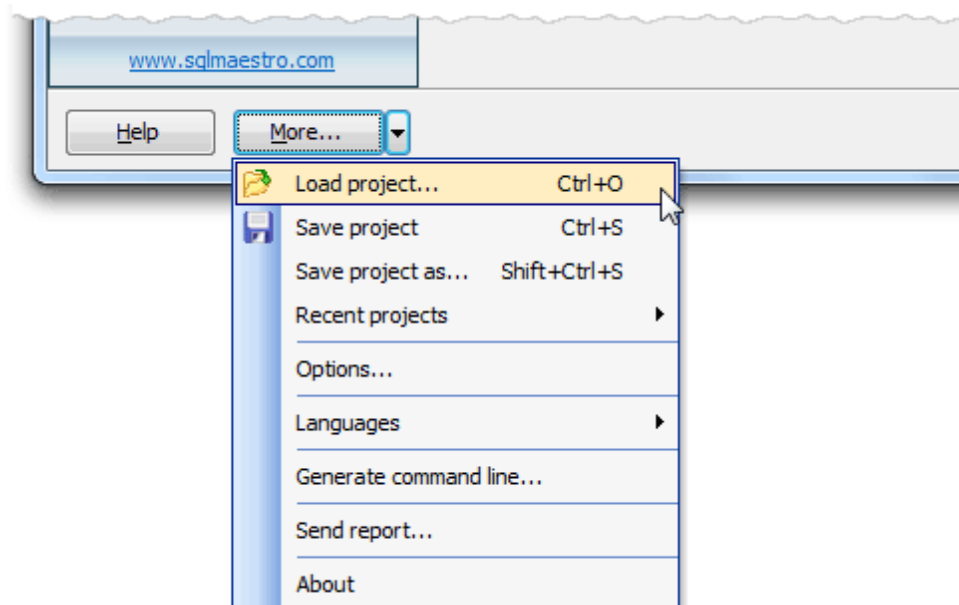
Note: You are actually connecting to your database through the PHP script on the server, so in most cases the host/server name is "localhost" unless the target database server is not installed on the same computer as the Web server.

2.2 Projects

PostgreSQL PHP Generator allows you to save and restore all the options set during a session, so you don't need to specify all options each time you work with the application anew; instead you can load all settings from a project and change them if necessary.

To create a project, configure [data sources](#)^[2] and click **More > Save Project** at any next step (**Ctrl+S**) or **More > Save Project as...** (**Ctrl+Shift+S**). All the settings you have made will be saved to a file.

To restore previously saved settings from a project, click **More > Load Project** at the first wizard step. Recently used projects are available from the **More > Recent Projects** popup menu.



2.3 Command line options

PostgreSQL PHP Generator supports a number of command line options that allow you to fully automate PHP scripts creation. To generate the command line automatically, load the project to be used or specify the generation options manually and click [More > Generate command line](#). The Command Line Builder allows you to save the prepared line to clipboard or to a batch file.

The PostgreSQL PHP Generator command line syntax is as follows:

PHPGenerator[.exe] [<project_file_name>] [-o|output <output_directory>] [-g|generate] [-h|help]

PHPGenerator[.exe]	The PostgreSQL PHP Generator program file.
<project_file_name>	The project ¹⁷ with all the task's settings.
<output_directory>	A directory where file are generated to.
-g generate	Generate without running the GUI application.
-h help	Show help information.

Examples

The examples below assume that you are entering the command lines in the PostgreSQL PHP Generator program directory. Don't forget to enclose all paths and filenames containing spaces in quotes.

To open the NBA_db.pgtm project in PHP Generator GUI:

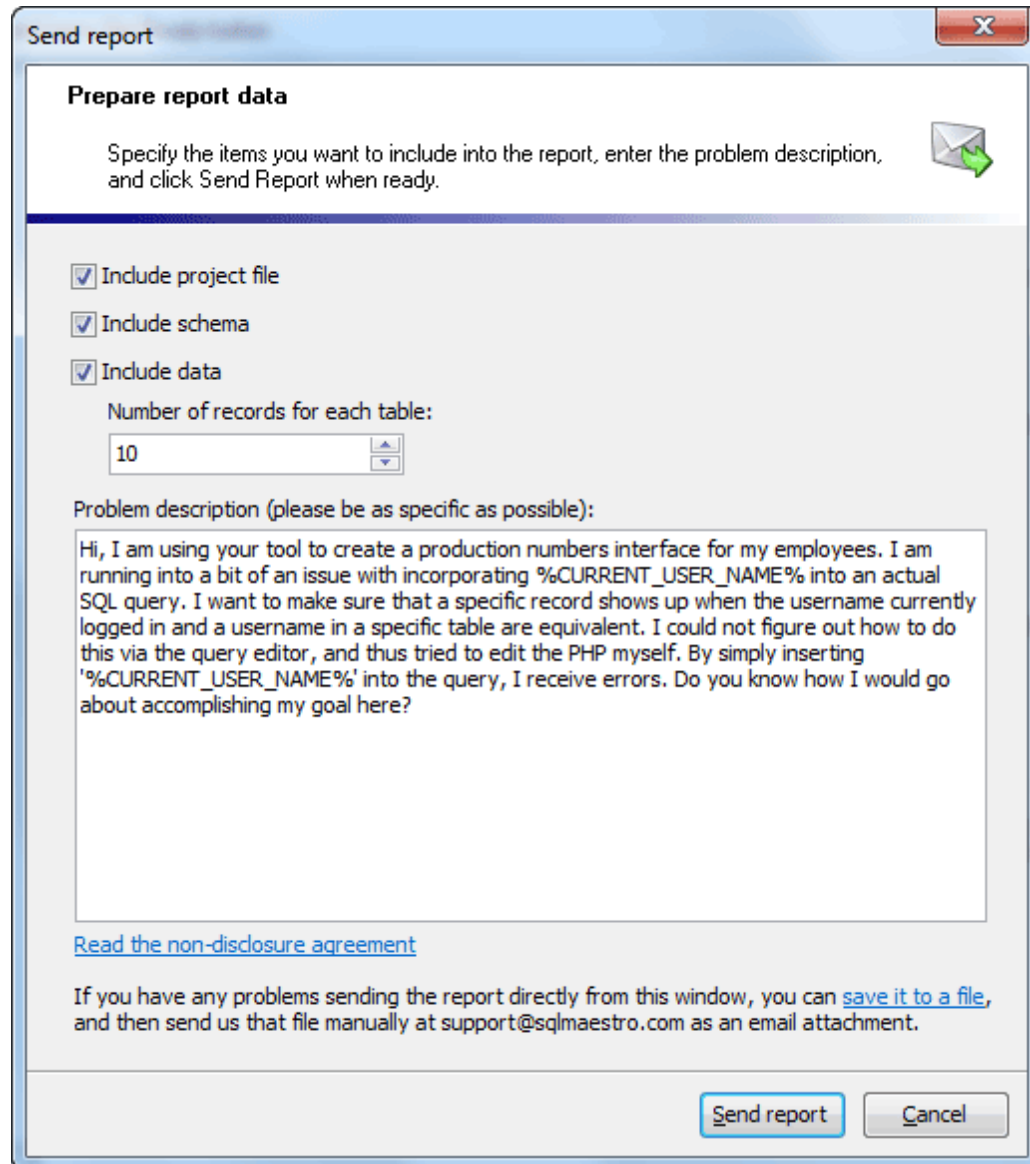
```
myphpgenerator C:\PHPGen\MySQL\NBA_db.pgtm
```

To generate files without opening PHP Generator GUI:

```
fbphpgenerator C:\PHPGen\Firebird\online_store.pgtf -g -o C:\xampp\htdocs\myapp
```


2.4 Report sending

To send a report to SQL Maestro support team, use the corresponding PostgreSQL PHP Generator feature. To invoke the window, click [More > Send report...](#) .



Send report

Prepare report data

Specify the items you want to include into the report, enter the problem description, and click Send Report when ready.

☒ Include project file

☒ Include schema

☒ Include data

Number of records for each table:

10

Problem description (please be as specific as possible):

Hi, I am using your tool to create a production numbers interface for my employees. I am running into a bit of an issue with incorporating %CURRENT_USER_NAME% into an actual SQL query. I want to make sure that a specific record shows up when the username currently logged in and a username in a specific table are equivalent. I could not figure out how to do this via the query editor, and thus tried to edit the PHP myself. By simply inserting '%CURRENT_USER_NAME%' into the query, I receive errors. Do you know how I would go about accomplishing my goal here?

[Read the non-disclosure agreement](#)

If you have any problems sending the report directly from this window, you can [save it to a file](#), and then send us that file manually at support@sqlmaestro.com as an email attachment.

[Send report](#) [Cancel](#)

Check the corresponding options to include project file, schema, and specified number of the table records, add a description and click [Send record](#) to get the prepared report in your default email client. In case you have no desktop email client installed, save the prepared report to a file with the corresponding option and send it manually to support@sqlmaestro.com as email attachment.

2.5 Shortcut keys

The following table describes the default shortcut keys in applications created with PostgreSQL PHP Generator.

Add new record	Alt+Ctrl+I, Alt+Insert
Save (in input forms)	Ctrl+Return
Save and add another record	Ctrl+Shift+Return
Previous page	Ctrl+Left
Next page	Ctrl+Right
Open all details	Ctrl+Shift+/
Open Filter Builder	Ctrl+Shift+F
Add a new condition in Filter Builder	Alt+Ctrl+I, Alt+Insert

All or any shortcuts may be customized in *components/js/pgui.shortcuts.js*

2.6 Deployment

Applications created by PHP Generator use the [Smarty](#) library. Smarty is a template engine for PHP, facilitating the separation of presentation (HTML/CSS) from the application logic. Smarty compiles copies of the templates as PHP scripts. Compilation happens once when each template is first invoked, and then the compiled versions are used from that point forward. The compiled templates are stored under the *templates_c* directory which must be accessible for writing by Smarty.

To upload a ready web application to your web hosting, you need to have the following conditions.

1. Allow write access to *templates_c*

The web server user must have write access to the *templates_c* directory. The most secure method is to make this directory owned by this user. The change of ownership is the easiest way to accomplish this. Only an administrator can execute this operation so if you can't do that then ask your hosting provider to do it for you. If you can do this, then it is possible (and recommended) to disallow "other" users read/write access for optimum security.

2. Deploy generated files

You can upload generated files to your web server manually or using a file synchronization software. In case of manual deployment there are two possible scenarios:

- If the new and the live web applications were created **by the same version** of PostgreSQL PHP Generator, just copy to the remote server all *.php* files from the root directory of the generated application and the *custom_templates* folder (if you [customized one or more templates](#)¹⁵⁹ in this project).
- If the new and the live web applications were created **by different versions** of PostgreSQL PHP Generator, copy all files from the output directory to the corresponding folder of the remote server and then empty the *templates_c* directory.

We recommend you to use a [file synchronization software](#). Any such tool compares and synchronizes the output directory and the corresponding folder on your web hosting quickly and can significantly simplify the deployment process. One thing you have to do is to exclude the *template_c* directory from the synchronization process.

3 Configuring datasources

PostgreSQL PHP Generator creates webpages aimed at interaction with PostgreSQL tables, views, and queries throw the web. This chapter explains how to manage data sources your pages are based. [Live Demos](#).

List of data sources

This list displays all data sources pages of your website are based on. The [Status](#) column indicates if a data source is OK or provides a brief description of the problem that must be solved before the data source can be involved into the project (a more detailed description is displayed in bottom part of the window). The [Create top-level page](#) column indicates whether a [top_level_page](#)^[28] for this data source should be created (this option usually turned OFF for queries and views to be used only as lookup data sources).

Adding tables and views

To add a table or a view as a page data source, open the [Select object](#)^[24] window with the corresponding button.

Adding queries

To add a query, type the query text in the [Query editor](#)^[28] or load it from an `.sql` file with [More > Load query from file...](#) Moreover you can use queries stored a single `.qrp` file (query repository). This feature may be extremely useful if you need to share a set of the same queries between several different projects. To save/load queries to/from a single file, use [More > Save all queries as repository](#) / [Load query repository](#) items from the [More](#) button menu accordingly.

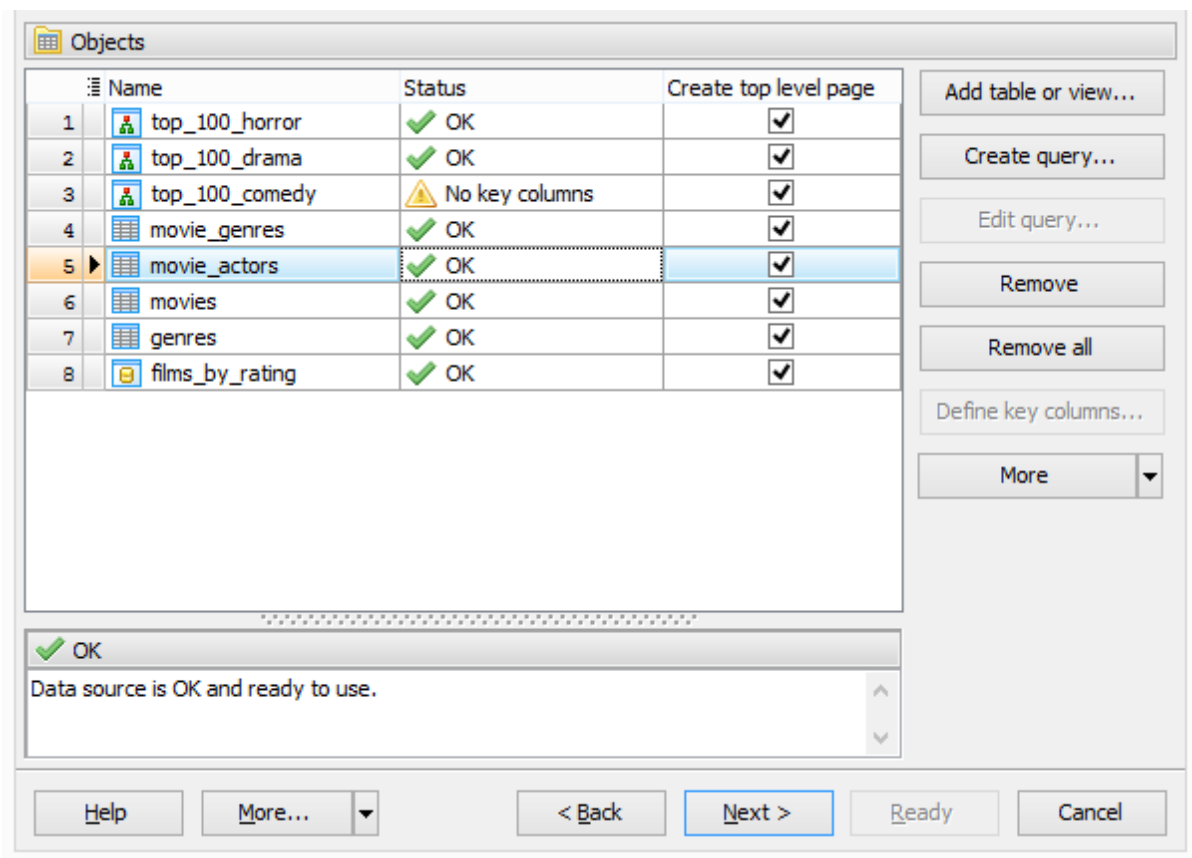
Invalid queries

PHP Generator automatically validates data sources and displays names of invalid queries in **red**; in this case the reason of the problem is displayed in the Status column. Please [follow these simple rules](#)^[28] to get your query valid.

Key columns

PHP Generator automatically inspects all data objects for key columns that are required for single-record operations like view, edit, and delete. For tables you have to define primary key constraints at the database level, for views and queries it is necessary to define key columns at this step of the wizard. If key columns are not defined for a data source, it is marked with [No key columns](#) label and a [confirmation](#)^[338] is displayed on moving to the next step. If one of the primary key columns in your updatable query or a view is autoincrement, set the appropriate flag to allow correct locating the newly record added via inline or modal forms.

Please note that you must define key columns for all data sources involved into the project including lookup data sources; otherwise duplicate records can be displayed in the data grid.



Set SQL statements

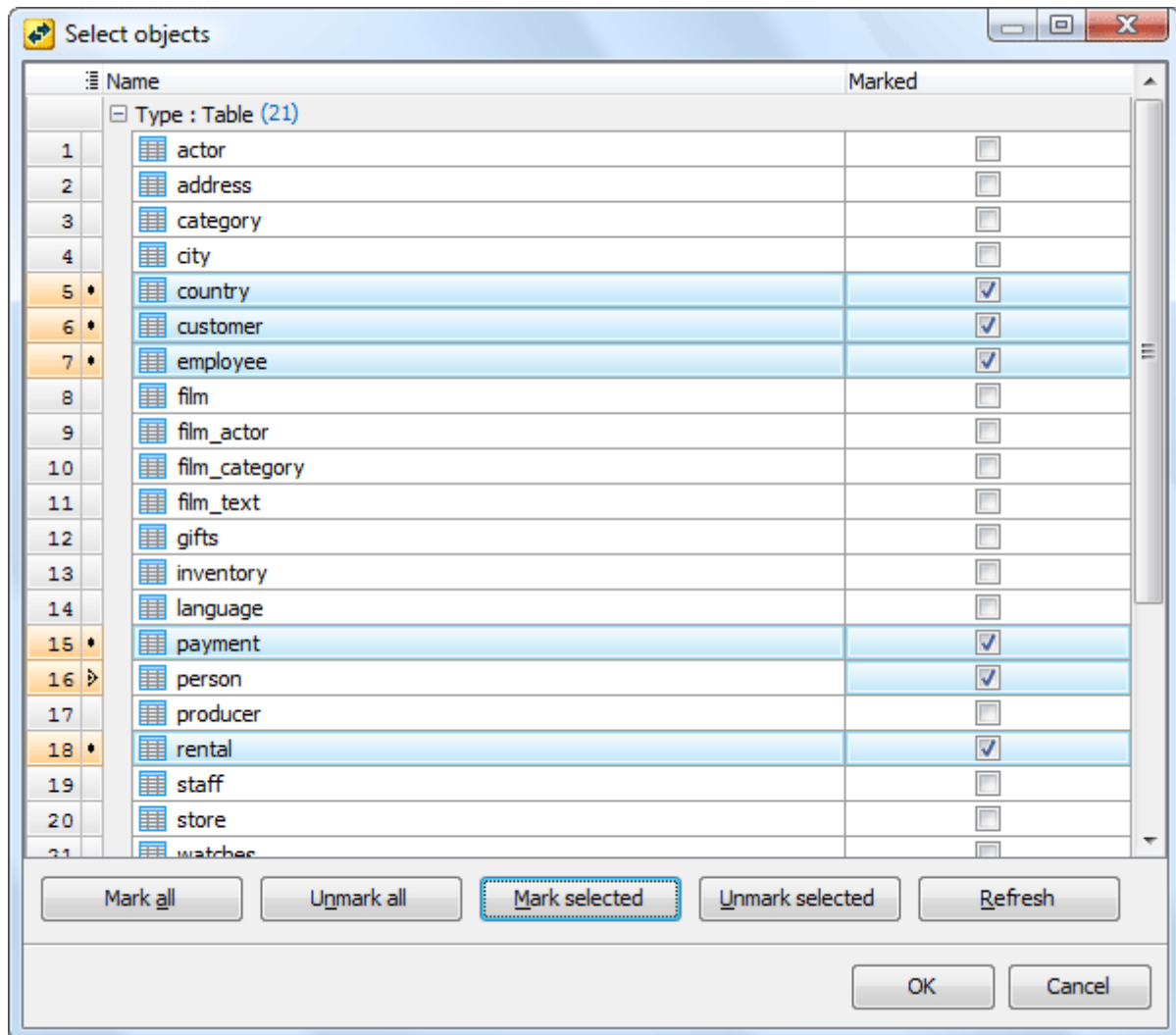
This button allows you to provide Insert, Update, and Delete statements for [views](#)^[24] and [queries](#)^[25]. These statements can be omitted if you don't plan to support insert, update, and delete operations for the corresponding page. If a view is updatable at the database level, these statements can be omitted as well: in this case they will be generated automatically (as for tables).

Views vs Queries

What is preferable: create a view in the database or a query directly in PHP Generator? We would recommend you to use views always when possible as in this case all business logic is concentrated in a single place (database). The only suitable exception is the case when you for some reasons cannot modify the database schema.

3.1 Tables and views

To specify tables and views to be used as page data sources, pick the corresponding objects at the [Select objects](#) tab. To choose multiple objects at a time, select them using **Ctrl** and **Shift** keys, then click [Mark selected](#).



The [Refresh](#) button allows you to refresh the database schema (useful if you just added a new object to the database: in this case you can involve it into the project even without closing this dialog window).

3.2 Custom SQL queries

To use a SELECT query as a page data source, click the [Create Query](#) button or select the corresponding popup menu item, specify the query name and text, and click OK.

Rules for creating queries used by PHP Generator

All queries must satisfy the simple requirement: the following query must be correct.

```
select * from
(
    QUERY_TEXT_YOU_ENTERED
) an_alias
```

This happens because the software uses similar queries for internal needs. In case such SQL expression is not valid, the wizard marks the query as invalid and displays its name in **red**.

To meet this requirement, make sure that **all the columns** in the result dataset have **unique aliases**. For example, the following query works fine itself, but returns a dataset with two columns named *id*:

```
SELECT
    table1.*,
    table2.*
FROM table1, table2
WHERE table1.id = table2.id;
```

This is the reason the wizard marks this query as invalid. To solve the problem, provide these columns with unique aliases:

```
SELECT
    table1.id as table1_id,
    table2.id as table2_id
FROM table1, table2
WHERE table1.id = table2.id;
```

Creating updatable datasets (For Professional Edition Only)

To get an updatable dataset based on an SQL query, you have to provide up to three SQL queries: UPDATE, INSERT, and DELETE to be able to modify, add and remove records accordingly. The first query provides an UPDATE statement for modifying existing records; the second query provides an INSERT statement to add a new record to the tables; and the third one provides a DELETE statement to remove the records. Each of these queries can contain several parameterized statements that use parameters like : *field_name*.

Example

Assume that we have the following SELECT statement:

```
SELECT
    id,
    first_name,
    last_name
FROM customer
WHERE last_name LIKE 'A%'
```


To create an updatable dataset based on this query, INSERT, UPDATE and DELETE statements can be specified as follows:

```
INSERT INTO
  customer
VALUES (:id, :first_name, :last_name);
```

```
UPDATE customer
SET   id = :id,
      first_name = :first_name,
      last_name = :last_name
WHERE id = :OLD_id;
```

```
DELETE FROM customer
WHERE id = :id;
```

The screenshot shows the 'Query' editor window. The 'Query name' field is set to 'Customer_list'. The 'Select' tab is active, displaying the following SQL query:

```
SELECT
  id,
  first_name,
  last_name
FROM customer
WHERE
  last_name LIKE 'A%';
```

At the bottom of the window, there is a checkbox labeled 'Updatable query' which is checked, and a button labeled 'Edit update statements...'.

The screenshot shows the 'Insert' tab of the statement editor. The following SQL statement is displayed:

```
INSERT INTO customer
VALUES (
  :id,
  :first_name,
  :last_name);
```

The screenshot shows the 'Update' tab of the statement editor. The following SQL statement is displayed:

```
UPDATE customer SET
  id = :id,
  first_name = :first_name,
  last_name = :last_name
WHERE
  id = :old_id;
```

The screenshot shows the 'Delete' tab of the statement editor. The following SQL statement is displayed:

```
DELETE FROM customer
WHERE id = :id;
```


4 Page Management

This step of the wizard displays the list of the webpages to be created. It allows you to customize the pages as well as manage master/detail views and define the structure of the application menu.

The working area of this step is divided into two main sections: [root level pages](#)^[28] (i.e. pages to be included into the site menu) and [detail pages](#)^[31] that are covered in the appropriate topics.

The [Setup project options to configure default page settings](#) link at the bottom of the step working area opens the [Project Options](#)^[26] dialog where you can setup application-level settings as well as specify default options for generated pages.

The screenshot shows the 'Page Management' wizard interface. It is divided into two main sections: 'Pages' and 'Details'.

Pages Section: This section displays a table of root-level pages. A red arrow points to the first row, 'Regular season games', with the label 'Root-level Pages'.

	Title	Menu label	File name	Include into menu	Group caption
1	Regular season games	Regular season games	regular_season_games	<input checked="" type="checkbox"/>	Games
2	Players	Players	players	<input checked="" type="checkbox"/>	Participants
3	Teams	Teams	teams	<input checked="" type="checkbox"/>	Participants
4	Arenas	Arenas	arenas	<input checked="" type="checkbox"/>	Participants
5	Playoff	Playoff bracket	playoff	<input checked="" type="checkbox"/>	Games

Details Section: This section displays a table of detail pages. A red arrow points to the first row, '%away_team_caption% at %home_team_caption Quarters', with the label 'Detail Pages'.

Title	Menu label	Detail data source	Link condition
%away_team_caption% at %home_team_caption Quarters	Quarters	game_quarter	game_list.id = game_quarter.g

At the bottom of the 'Details' section, there is a link: [Setup project options to configure default page settings](#). A red arrow points to this link with the label 'Project Options'.

The interface includes various action buttons on the right side of each section, such as 'Edit...', 'Properties...', 'Reorder...', 'Refresh', 'Groups...', 'Add...', 'Copy...', 'Remove', and 'More...'. At the bottom of the wizard, there are navigation buttons: 'Help', 'More...', '< Back', 'Next >', 'Ready', and 'Cancel'.

4.1 Root level pages

This list contains root-level pages i.e. pages to be included into the website menu. You can manage and customize these pages using the buttons on the right of the page list as described below.

All cells in the list are editable i.e. it is possible to edit the value of any cell directly in the list. To do so, just select the cell and click the left mouse button or the F2 key. All these values can be also set in the [Page Properties](#)^[203] dialog.

Edit

Opens [Page Editor](#)^[33] for the selected page.

Properties

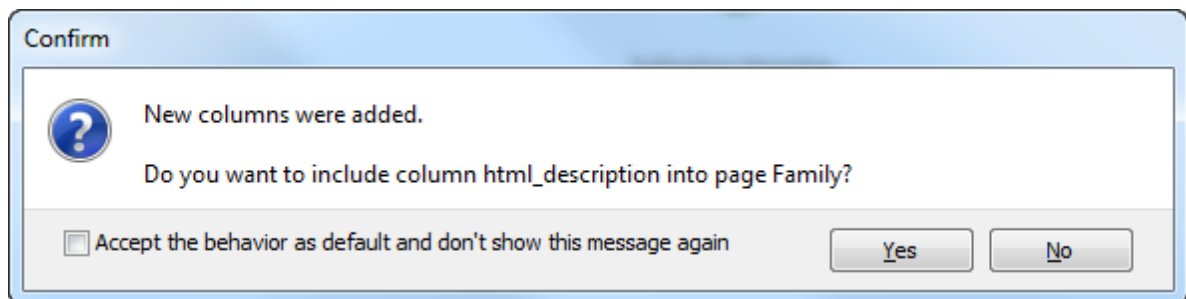
Provides quick access to the [Page Properties](#)^[203] dialog which is also can be invoked from [Page Editor](#)^[33].

Reorder

Allows you to reorder the pages in the site menu.

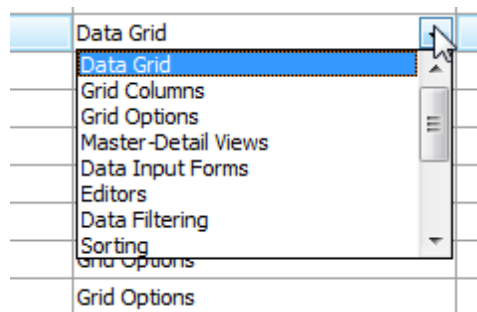
Refresh

Refreshes the database schema. If a new column detected in a page data source, PostgreSQL PHP Generator asks you whether you want to include that column into the page. This behavior can be specified in the [Application Options](#)^[335] dialog.



Groups

Invokes a modal dialog that allows you to manage top-level site menu groups. To include a page into a group, click a cell in the Group column and select the desired group in the drop-down list.

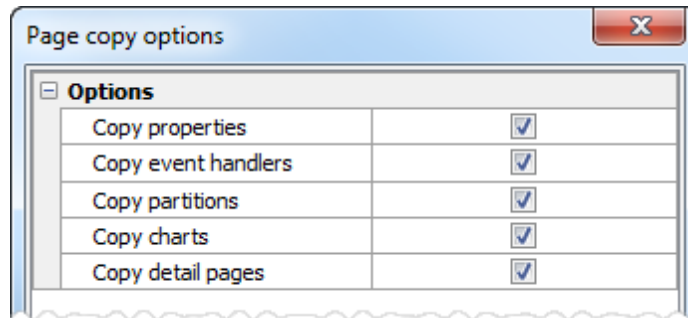


Add

Helps you to quickly add a new page to the website without necessity to return to the previous step. Commands in this button's drop down list allows you to add two or more pages based on the same data source.

Copy

Clones the selected page. Settings you want to copy can be customized in the dialog window.

**Remove**

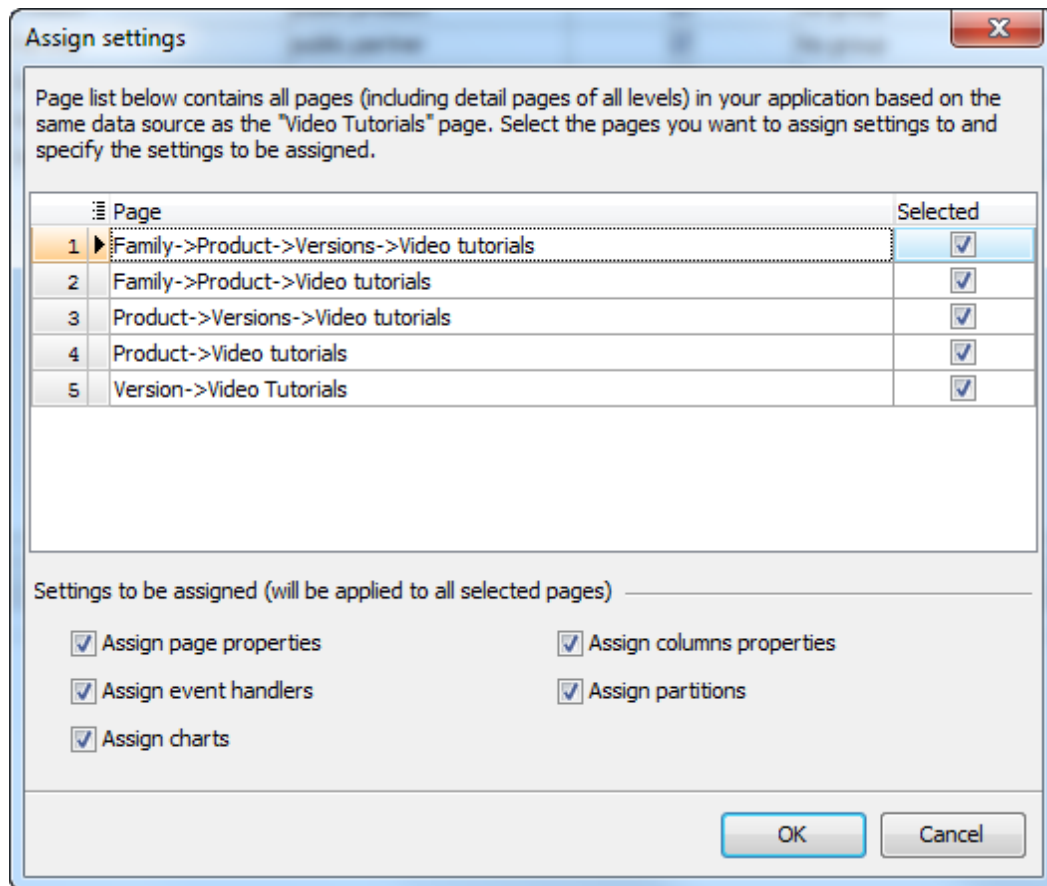
Removes the selected page from the list and website.

More

This button provides access to additional page management commands as described below.

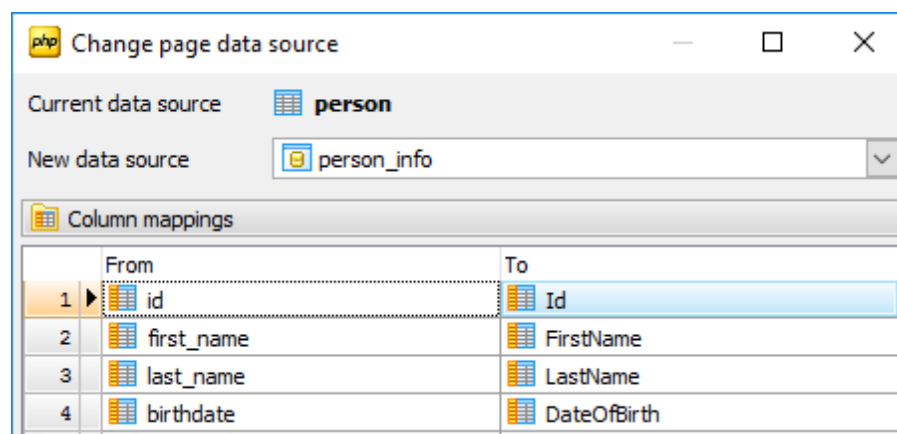
Assign settings

Copies all or certain page settings from the current page to selected pages based on the same data source.



Change page data source

To change a data source of an existing page, select a new data source and specify the accordance between columns to transfer View/Edit column properties. Columns used for master-detail linking are marked by asterisks and must be mapped.



Reset, Reset All

Resets all the settings for the selected page or for all the website pages to their default values specified in the [Project Options](#) ²²⁶ dialog.

4.2 Detail pages

This list contains [detail pages](#)^[11] i.e. child views for [root level pages](#)^[28]. If the [Setup details by foreign key](#)^[335] option is turned ON, such pages are created automatically according to foreign key constraints detected in the database schema (only for the first-level details). You can manage and customize these pages using the buttons on the right of the page list as described below.

Add

Adds a new detail page located at the same level as the selected page.

Add Child

Adds a child page for the selected detail page. This command allows you to create a detail page at any level of nesting.

Edit

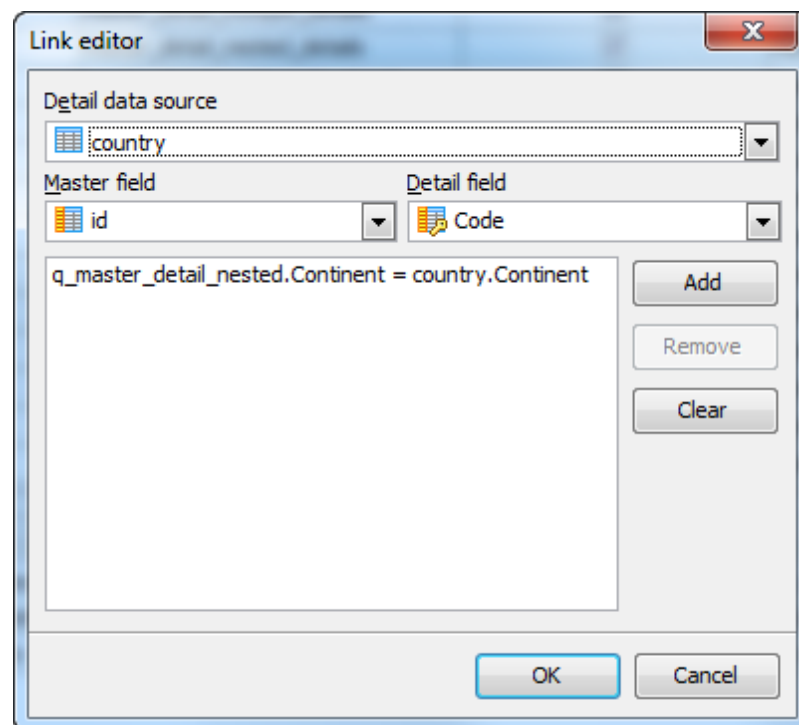
Opens [Page Editor](#)^[33] for the selected page.

Properties

Provides quick access to the [Page Properties](#)^[203] dialog which is also can be invoked from [Page Editor](#)^[33].

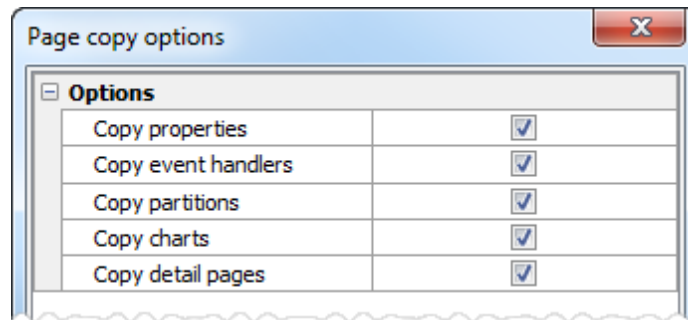
Link Editor

Opens a dialog window that allows you to edit link conditions between master and detail pages. This dialog is also invoked when you add a new detail page with **Add** or **Add Child** commands.



Copy

Clones the selected page. Settings you want to copy can be customized in the dialog window.



Remove

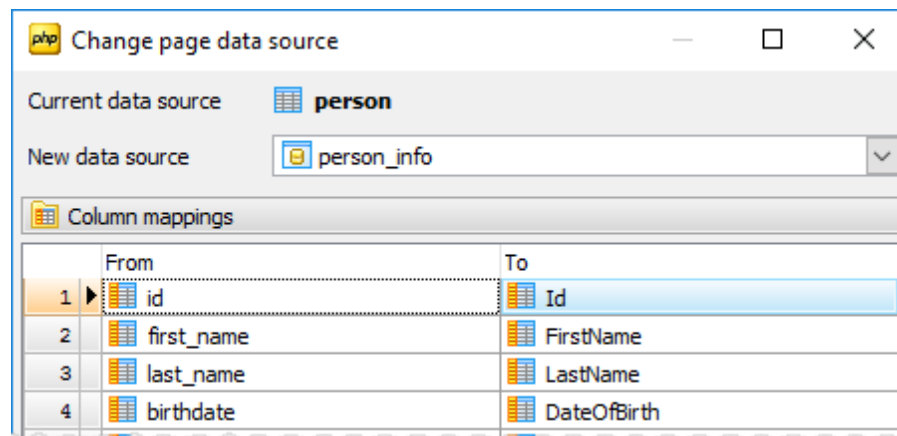
Removes the selected page from the list and website.

More

This button provides access to additional page management commands as described below.

Change page data source

To change a data source of an existing page, select a new data source and specify the accordance between columns to transfer View/Edit column properties. Columns used for master-detail linking are marked by asterisks and must be mapped.



Reorder

Allows you to change the order of detail pages.

Reset, Reset All

Resets all the settings for the selected page or for all the pages to their default values specified in the [Project Options](#) ²²⁶ dialog.

5 Page Editor

Page Editor is a feature-rich tool that allows you to customize website pages. Its working area consists of several tabs as described below.

[Columns](#) ³⁵

Use this tab to specify columns to be displayed on the result web page and corresponding forms, controls to be used for these columns in Insert and Edit forms, lookup options and much more.

[Details](#) ¹¹¹

Use this tab to adjust the page detail presentations and setup detail pages properties.

[Events](#) ¹¹⁷

Set here the fragments of PHP and JavaScript code to be executed before or after a record was added, edited, deleted, etc.

[Filter](#) ¹⁹⁸

Use this tab to reduce the number of records available at the generated page.

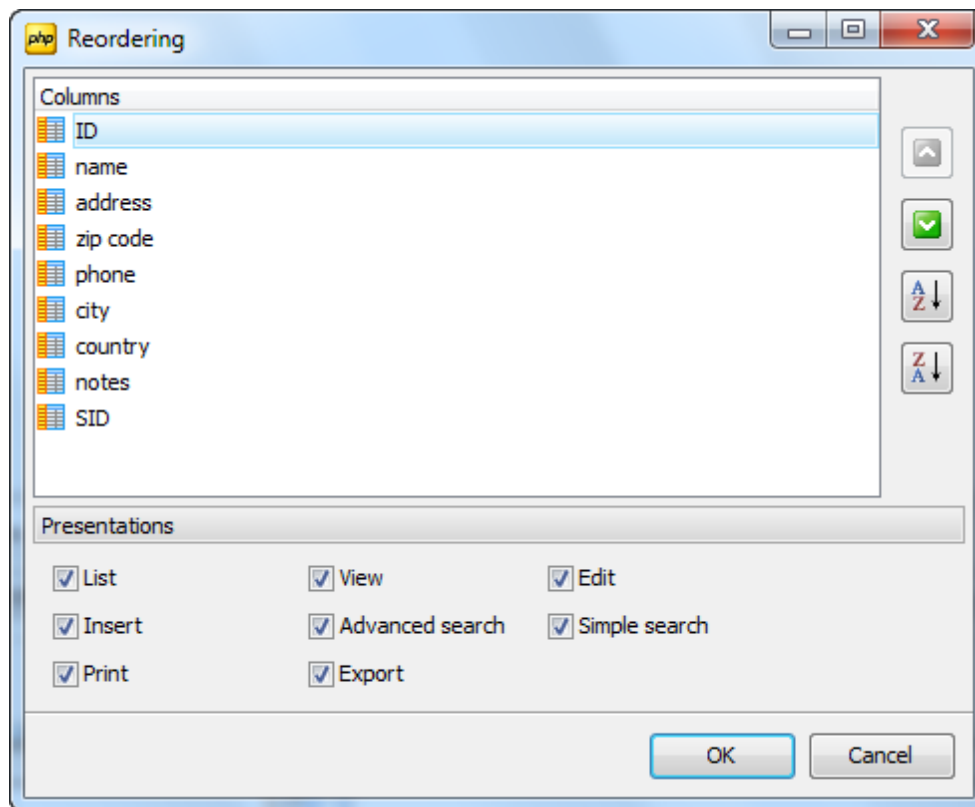
[Charts](#) ¹⁹⁹

Use this tab to equip the web page with interactive charts.

Use the [Page Properties](#) ²⁰³ window to setup common page properties such as view/edit/delete/filter/export abilities, pagination options, page header, and more.

The [Data Partitioning](#) ²²⁰ wizard allows you to create a custom pagination i.e. split the records on the generated page by a specified criteria.

To setup order of columns, click the [Reorder columns...](#) button and set the new order in the opened window. Check the corresponding boxes to apply this order to necessary presentations. The order of columns within Page Editor is changed accordingly if the new order is applied to all presentations and remains unchanged if a different column order is specified for two or more presentations.



5.1 Columns

The Columns tab of Page Editor allows you to customize the way column data is displayed for all the available presentations (like List, Edit, Insert, etc). Details are covered below.

Common

Properties specified in this section are applied to all presentations.

Caption

Allows you to specify the caption for the column. To provide different captions for different presentations (for example, you might want to specify a shorter caption for the List view and a more longer caption for Insert/Edit form), click the ellipsis button and specify the appropriate values.

Use lookup

Turn this checkbox ON and set the appropriate options to enable displaying of [lookup values](#)^[37] instead of the "real" values stored in this column. For example, if a column stores country IDs, you can use this option to display country names instead of their IDs (assuming that there is a table that stores the ID/name combination for each country).

Common	
Caption	Team
Use lookup	<input checked="" type="checkbox"/>
Data source	team
Link field	id
Display field	caption
Sorting	(none)
Filter condition	

By default, PostgreSQL PHP Generator enables lookups for columns linked by a foreign key with a single column from another table (if the [Setup lookups by foreign key option](#)^[335] option is enabled).

View^[44]

Use this section to specify the way column values are displayed on List, View, Print, Export, and Compare pages.

View	
Display Properties	Text
Display as hyperlink	<input checked="" type="checkbox"/>
HREF Template	http://www.nba.com/%website%/
Target	_blank
Header hint	
Minimal visibility	Phone
Fixed width	<input type="checkbox"/>
Width	0
Units	px
Totals	None

List

Here you can specify some options that are applied only for the List view.

Show column filter

Turn ON the checkbox to enable the column filtering for the appropriate column. Use *Sorting* and *Number of values to display* controls to specify the sort order and number of values to be displayed in the column filter dropdown accordingly. [Live Demo](#).

Minimal visibility

PostgreSQL PHP Generator provides you with an ability to generate responsive pages that look beautiful on any device from a mobile phone up to an extra-large desktop. This option allows you to select the minimum resolution of devices the column will be visible at. [Live Demo](#).

Available values correspond to the device widths as follows:

Phone	The column is displayed on all devices.
Tablet	The column is displayed on all devices with width greater than 768px .
Desktop	The column is displayed on all devices with width greater than 992px .
Large desktop	The column is displayed only on devices with width greater than 1200px .

Fixed column width

To set a fixed width for a column, check the *Fixed width*, specify the *Width* and select the *Units*. You can use both relative-length (like *px* or *mm*) and absolute-length (*em*, *rem*, etc) units. [Live Demo](#). More about [CSS units](#).

Totals

This option allows you to enable a grid footer to display summaries (*Sum*, *Average*, *Count*, etc) for the selected column. [Live Demo](#).

Edit/Insert

Allows you to select a control to be used for the selected column in Edit and Insert forms and customize the control's properties.

Edit/Insert	
Edit properties	Autocomplete editor
Read only	<input type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Enabled	<input checked="" type="checkbox"/>
Required	<input checked="" type="checkbox"/>
Client validators	...
Default value	▼

Filter Builder

Here you can specify operators to be allowed for the selected column in the [Filter Builder](#) dialog. To select the operators, open the drop-down list and select/unselect the

corresponding options.

5.1.1 Lookup settings

Introduction

Lookups are used in data grids and in data input forms. For read-only views (List, Single Record View, Print, etc) lookups are used to display records of another dataset corresponding to and instead of values stored in the webpage base data source. For Insert and Edit forms lookup editors are used to simplify input by selecting a value storing in another dataset and corresponding to a pre-defined value from the base one.

In data input forms the following editors can be used for lookup controls: [Dynamic Combobox](#)^[73] (default option), [Radio Group](#)^[89], [Combobox](#)^[70], [Cascading Combobox](#)^[75], and [Dynamic Cascading Combobox](#)^[79]. Here are some recommendations:

- for lookup data sources containing a few (usually no more than 5-7) values, all editors can be used.
- for lookup data sources containing no more than 15-20 values, both [Dynamic Combobox](#)^[73] and [Combobox](#)^[70] editors can be used.
- for lookup data sources containing 20+ values, the [Dynamic Combobox](#)^[73] editor is recommended.
- to allow users select the desired value step-by-step (for example, user selects a country and then a city in this country), [Cascading Combobox](#)^[75] or [Dynamic Cascading Combobox](#)^[79] can be used.

Setting a lookup editor bound to a data source (table, view, query)

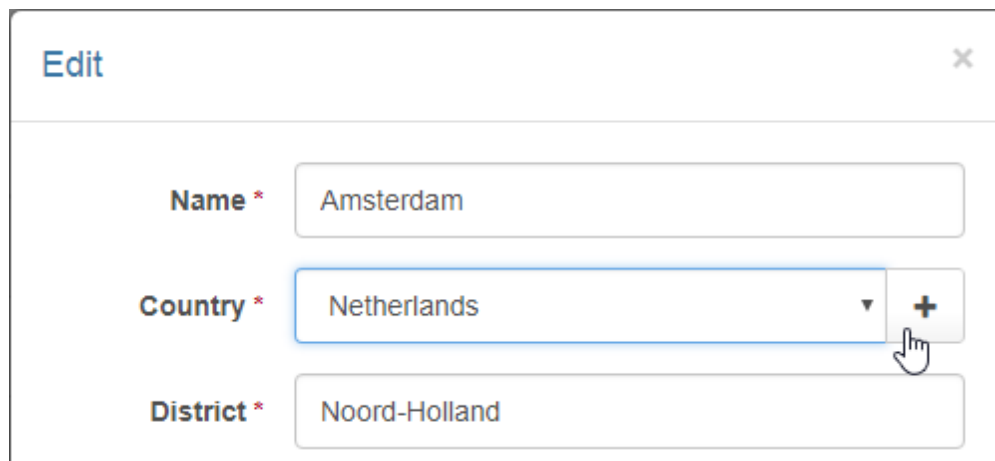
If you create a PHP script for taking orders, the 'Orders' table generally includes a field hosting a number indicating the customer who made the order. Working directly with the customer numbers is not the most natural way; most users will prefer to work with customer names. However, in the database the customers' names are stored in a different table to avoid duplicating the customer data for each order by the same customer. To get around such a situation, you can enable a lookup editor:

- check the [Use lookup](#) box;
- select the foreign table/view/query as [Data Source](#);
- specify the field with the same data as [Link field](#);
- set the field with the data to be appeared in the lookup editor as [Display field](#);

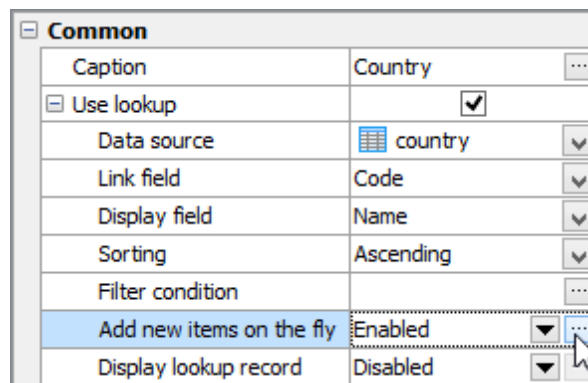
By default, PostgreSQL PHP Generator uses the [Dynamic Combobox](#)^[73] editor for a column linked by a foreign key with a single column from another table (if [Setup lookups by foreign key option](#)^[335] is enabled).

Add new item on the fly

Turn this option ON in order to allow adding new items directly in Insert and Edit forms. When this option is checked, a plus button is displayed on the right of the corresponding control ([Live demo](#)).



Click the ellipsis button to customize the modal dialog to be displayed when adding a new item. For example, you might want to hide some fields and/or customize the form layout.



Common	
Caption	Country ...
<input checked="" type="checkbox"/> Use lookup	<input checked="" type="checkbox"/>
Data source	country
Link field	Code
Display field	Name
Sorting	Ascending
Filter condition	...
Add new items on the fly	Enabled ...
Display lookup record	Disabled

Display lookup record

Turn this option ON in order to allow users to see the whole lookup record clicking on the corresponding cell in the data grid ([live demo](#)).

Country Lookup View



Common info

Country Name	Continent	Region
United States	North America	North America

Additional info

Surface Area	Population	Life Expectancy
9,363,520.00 km ²	278,357,000	77.10

Flag map & Emblem

Click the ellipsis button to customize the modal dialog to be displayed when viewing the record. For example, you might want to hide some fields and/or customize the form layout.

Common	
Caption	Country ...
Use lookup	<input checked="" type="checkbox"/>
Data source	country
Link field	Code
Display field	Name
Sorting	Ascending
Filter condition	...
Add new items on the fly	Disabled
Display lookup record	Enabled
View	

Using Filter condition

Filter condition allows you to reduce the list of values represented in the lookup editor with a specified criteria. This condition corresponds to the WHERE clause applied to the data source (you must not add the WHERE keyword to beginning of the condition). The

following operators can be used in this clause: =, <> (!=), >, <, >=, <=, BETWEEN, LIKE, IN. It is also possible to use [predefined variables](#)^[195] like %CURRENT_USER_NAME%.

Example 1

To enable a lookup editor with a list of USA cities (the corresponding value of "country_id" is 103) named like Da*, specify the following condition: `country_id = 103 AND city LIKE 'Da%'`

Example 2

Suppose we have a table that contains a column 'owner' with owner information. To set a lookup editor with the list of values owned by the current user, specify the filter condition as follows: `owner = '%CURRENT_USER_NAME%'`

NB. The Filter Condition property cannot be used to implement [dependent lookups](#) or another similar logic. Only [predefined variables](#)^[195] and constant expressions are allowed.

Setting a lookup editor represented data of multiple columns

To create such lookup editor, create a query with all the necessary data concatenated into a single column and specify the query as [Data Source](#). A complete example can be found below.

Example

Suppose we have three tables: 'employee' with a list of office employees, 'job' with employees' job titles and salaries, and 'department' with a list of office departments.

See definitions here

```
CREATE TABLE employee (
  EMP_NO      integer NOT NULL PRIMARY KEY,
  FIRST_NAME  varchar(15) NOT NULL,
  LAST_NAME   varchar(20) NOT NULL,
  JOB_CODE    integer NOT NULL,
  DEPT_NO     integer NOT NULL
);
```

```
CREATE TABLE job (
  JOB_CODE    integer NOT NULL PRIMARY KEY,
  JOB_TITLE   varchar(25) NOT NULL,
  SALARY      real NOT NULL
);
```

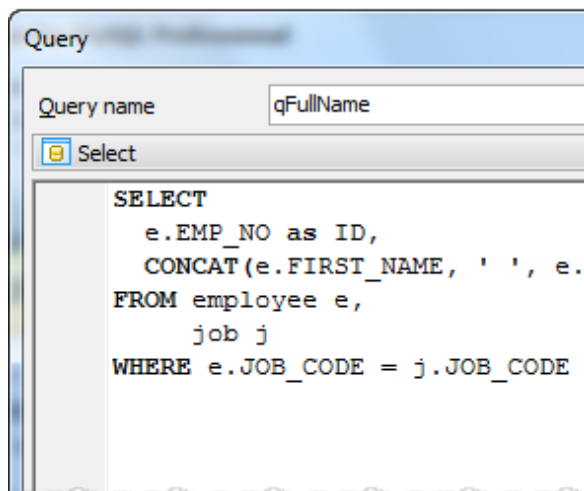
```
CREATE TABLE department (
  DEPT_NO     integer NOT NULL PRIMARY KEY,
  DEPARTMENT  varchar(20) NOT NULL,
  HEAD_DEPT   integer,
  MNGR_NO     integer,
  BUDGET      real,
  LOCATION    integer,
  PHONE_NO    char(20)
);
```

To enable a lookup editor for the 'HEAD_DEPT' field of the 'department' table

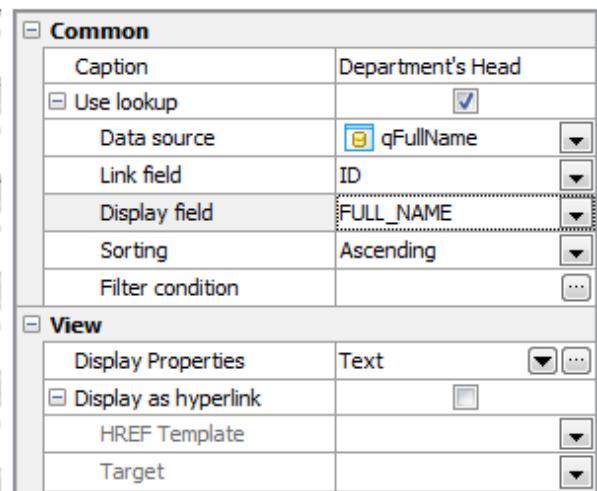
representing first name, last name, and job title of the employee, follow the steps above with the following query text:

```
SELECT
    e.EMP_NO as ID,
    CONCAT(e.FIRST_NAME, ' ', e.LAST_NAME, ' ', j.JOB_TITLE) as FULL_NAME
FROM employee e,
     job j
WHERE e.JOB_CODE = j.JOB_CODE
```

Adding a query

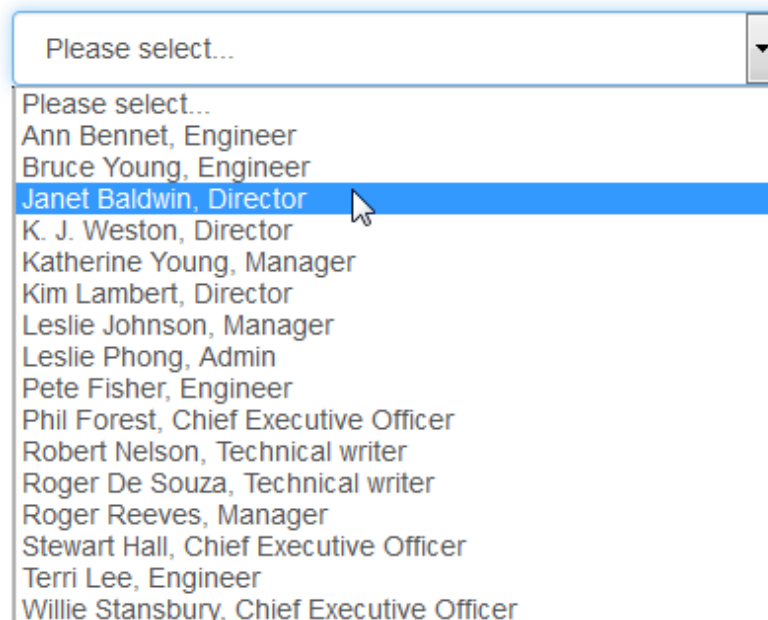


Setting lookup options



Lookup editor

Department's Head



Setting a lookup editor bound to a custom value list

To create a lookup editor bound to a custom value list i.e. to a list of values that are not stored in a database table and cannot be retrieved by a query, process as follows:

- Select [Combobox](#) or [Radio_group](#) in the Edit Properties line in the Insert/Edit group;
- Click the ellipsis button next to the "Edit properties";
- Enter the list of the allowed values in pairs *an_allowed_value=value_to_be_represented* separated by a comma (Example: 1=One,2=Two).

Source table

		first_name	last_name	sex
1	1	John	Doe	1
2	2	Betty	Fisher	2
3	3	Forest	Gump	1
4	4	Sara	Connor	2

Page Editor

Common	
Caption	Sex
<input checked="" type="checkbox"/> Use lookup	<input type="checkbox"/>
Data source	
Link field	
Display field	
Sorting	Ascending
Filter condition	
View	
Edit/Insert	
Edit properties	Combo box
Read only	<input type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Enabled	<input checked="" type="checkbox"/>
Required	<input checked="" type="checkbox"/>
Client validators	
Default value	

Setting Combo box's options

Edit properties

Properties

Additional

Max width

Values

Most frequently used values

Item caption template

Advanced

Custom attributes

Inline styles

OK

Cancel

Lookup editor

First Name

Betty

Last Name

Fisher

Sex *

Female

Please select...

Male

Female

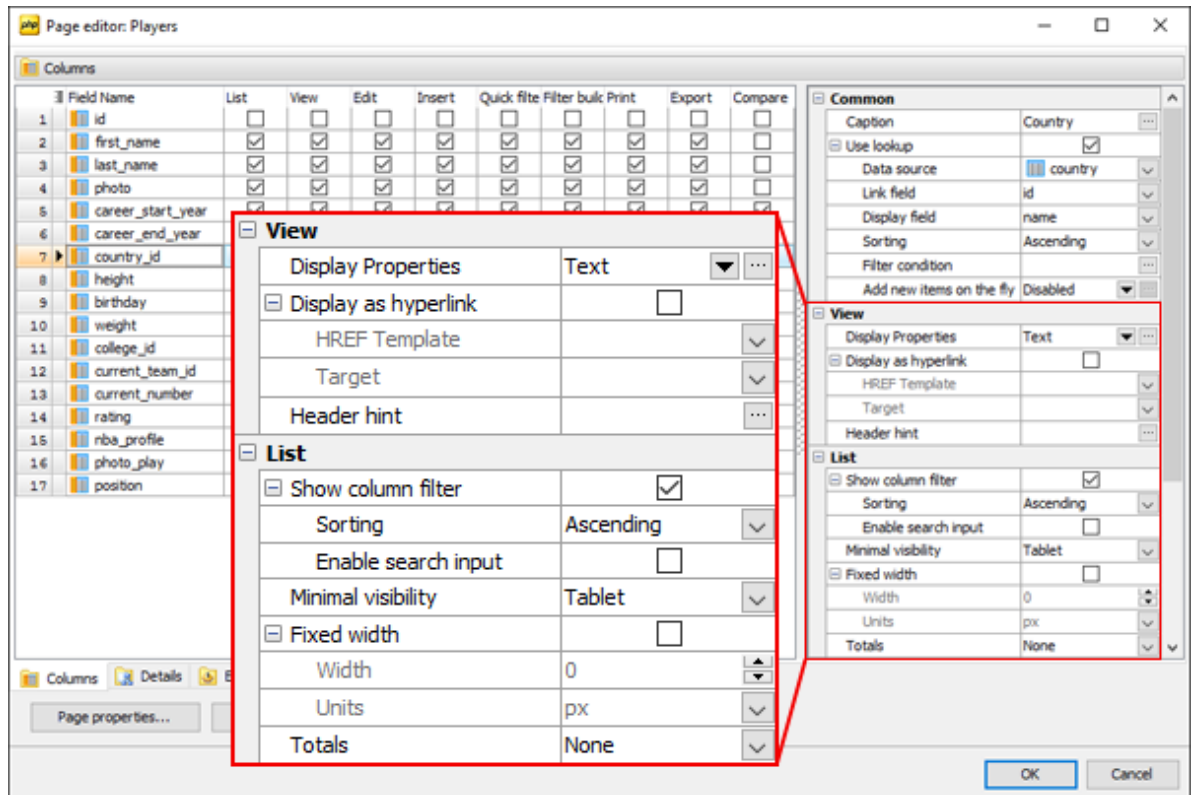
Save

Cancel

By default, PostgreSQL PHP Generator creates lookup editors described above for columns based on enumeration data types.

5.1.2 View controls

The View options define the way the column data is displayed on [List](#), [View](#), [Print](#), [Export](#), and [Compare](#) pages.



Display properties

Select the way to be used to represent column data from the drop-down list. Available controls are as follows:

- [Text](#) ⁴⁷
- [DateTime](#) ⁴⁹
- [Checkbox](#) ⁵⁰
- [Toggle](#) ⁵¹
- [File download](#) ⁵²
- [Image](#) ⁵³
- [External File](#) ⁵⁵
- [External Image](#) ⁵⁶

- [External Audio](#)^[57]
- [External Video](#)^[58]
- [Embedded video](#)^[59]
- [Barcode](#)^[60]
- [QR Code](#)^[61]

To set the format to be applied to the column data such as text alignment, image size, and so on, use the dialog opened by the ellipsis button. By default, the format is the same as it is set at the [Project options](#)^[336].

Along with available formatting options you can specify any property you want using the [Custom attributes](#) option. This option allows you to specify the content of standard HTML style tag applied to the column data.

The **Null Label** property allows you to customize the representation of NULL values at the column level.

To implement a conditional data formatting, use the [OnCustomDrawRow](#)^[174] or [OnExtendedCustomDrawRow](#)^[155] events.

Display as hyperlink

To represent the column's data as hyperlink, check the [Display as hyperlink](#) option and specify the [HREF Template](#)^[110]. **Target** controls where the new document is displayed when a user follows the link.

<code>_blank</code>	opens the new document in a new window.
<code>_parent</code>	is used in the situation where a frameset file is nested inside another frameset file. A link in one of the inner frameset documents which uses " <code>_parent</code> " will load the new document where the inner frameset file had been.
<code>_self</code>	puts the new document in the same window and frame as the current document. " <code>_self</code> " works the same as if you had not used <code>TARGET</code> at all.
<code>_top</code>	loads the linked document in the topmost frame... that is, the new page fills the entire window.

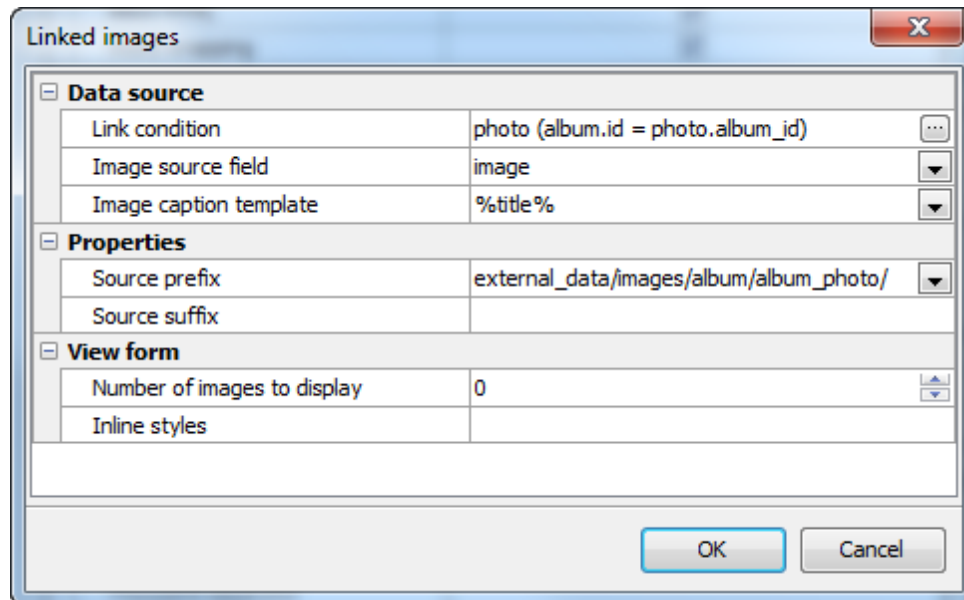
Header hint

Use this field to specify the the column caption's hint. By default, it is a comment to the table/view column.

Linked Images

This feature is an alternative way for displaying [master/detail](#)^[111] data when the detail data source stores filenames for external images (for example, the master table stores information about photo albums while the detail table stores information about album photos). These images are displayed in the popup window when you click a cell in a grid column linked images are associated with. [Live Demo](#).

To activate linked images for a column, set the corresponding option to "Enabled", then click the ellipsis button and provide values for the controls in the modal dialog as described below.



Link condition

Click the ellipsis button to invoke a modal dialog that allows you to select a data source containing information about external images and the columns to be linked.

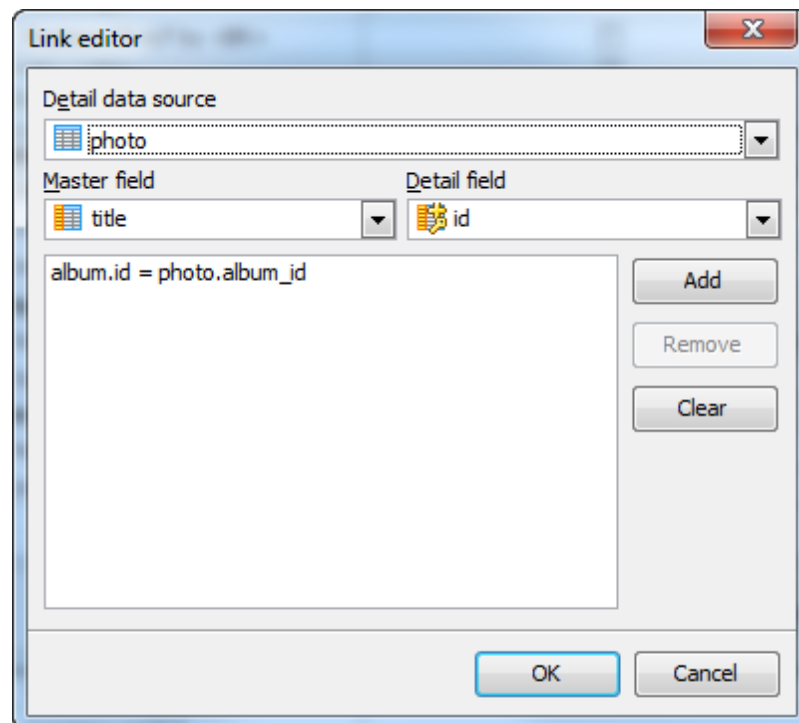


Image source field

A column in the image data source where image file names are stored.

Image caption template

The value of this property (if specified) is displayed below the left bottom corner of each image. You can use here column names from the image data source wrapped by

percents, for example, %title% (%id%).

Source prefix and Source suffix

These properties allow you to improve the flexibility of the application. For example, if files are stored in the media directory, you can specify the value of the Prefix property as 'media/' and store in the database only short file names. If one day you decide to rename the directory, the only thing you will need to change is the value of this property. In short, the actual filename is calculated as follows: Source prefix + value_stored_in_the_database + Source suffix.

Enter topic text here.

Number of images to display

The value of this property defines number of images to be [previewed in the View form](#). Default value is 0 (no images are previewed).

Inline styles

Allows you to specify the value of the style attribute of each tag in the View form. For example, the following value can be used to limit the width of previewed images:

```
max-width: 150px;
```

5.1.2.1 Text

The most popular type of data representation when data is represented "as is" with minimum modifications. All fields represented on the picture below are displayed as text.

Boston Celtics	1947	The Boston Celtics is owned by Wycliffe Grousbeck and coached by Doc Rivers, with Danny Ainge as the President of Basketball Operations. Founded in 1946, their 17 NBA Championships are the most for any NBA franchise. The Celtics' gre... more
----------------	------	--

Max length

Use this option to specify the maximum number of symbols to be displayed on the generated webpage. In case the length of stored text is longer, this text will be cropped and the more link will be added after the allowed number of symbols. The full text will be displayed in a popup window on putting mouse over the [more](#) link or in a separate window on clicking this link. Use [application options](#)^[338] to specify the default max length of text fields.

The **Boston Celtics** is owned by Wycliffe Grousbeck and coached by Doc Rivers, with Danny Ainge as the President of Basketball Operations. Founded in 1946, their 17 NBA Championships are the most for any NBA franchise. The Celtics' gre... [more](#)

The **Boston Celtics** is owned by Wycliffe Grousbeck and coached by Doc Rivers, with Danny Ainge as the President of Basketball Operations. Founded in 1946, their 17 NBA Championships are the most for any NBA franchise. The Celtics' greatest domination came from 1957 to 1969, with 11 championships in 13 years, and eight in a row, the longest consecutive championship winning streak of any North American professional sports team. They currently play their home games at [TD Garden](#).

Replace CR+LF by

This option is useful for correct representation of line feeds in the text in which line breaks are marked with the Carriage Return (CR) and Line Feed (LF).

Use the **Allow HTML** and **Word wrapping** checkboxes to set whether HTML tags will be applied to the stored data and whether word wrapping will be allowed.

Align

Allows you to specify the alignment of the control. Possible values are Default, Left, Right, and Center.

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. Foreexample, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

Null label

This property allows you to define how NULL values are represented for this column. By default the value of this property corresponds to the one defined at the project level. You can specify a custom value for a certain column (for example, "Not selected", "Not available", "Not supported", etc.) if necessary.

Set the **Format** options to be applied to the column data depending of data type (Number, Percent, String, Currency).

5.1.2.2 DateTime

Use this type of representation to display date and time in convenient format.

	id	game_date	home_team_id
1	1	28.10.2008	1
2	2	28.10.2008	3
3	3	28.10.2008	5

28 Oct 2008	Boston Celtics
28 Oct 2008	Chicago Bulls
28 Oct 2008	Los Angeles Lakers

Date time format

Use the drop-down list to select a format to be applied to the column data or specify your own format manually. Find out more information about supported date and time formats at [PHP manual](#).

Format	Example
Y-m-d	2014-11-18
d.m.Y	18.11.2014
d M Y	18 Nov 2014
Y-m-d H:i	2014-11-18 12:30:00
s	
d.m.Y H:i:s	18.11.2014 12:30:00
d M Y H:i:s	18 Nov 2014 12:30:00

Align

Allows you to specify the alignment of the control. Possible values are Default, Left, Right, and Center.

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document

compatible with the HTML5 requirements.

Null label

This property allows you to define how NULL values are represented for this column. By default the value of this property corresponds to the one defined at the project level. You can specify a custom value for a certain column (for example, "Not selected", "Not available" "Not supported", etc.) if necessary.

5.1.2.3 Checkbox

Use this control to display column values as checkboxes. This type of representation is used for columns storing *BOOLEAN*, or *INTEGER* data. PostgreSQL PHP Generator provides you with the following types of the control appearance.

- Standard check box control

Additional	
Display type	Check box control
"True" caption	
"False" caption	
Style	
Custom attributes	
Align	Default

First Name	Last Name	Active
MARY	SMITH	<input checked="" type="checkbox"/>
PATRICIA	JOHNSON	<input type="checkbox"/>

- Images

Additional	
Display type	Images
"True" caption	
"False" caption	
Style	
Custom attributes	
Align	Default

First Name	Last Name	Active
MARY	SMITH	✓
PATRICIA	JOHNSON	✗

- Text values

Additional	
Display type	Text values
"True" caption	Active
"False" caption	Not active
Style	
Custom attributes	
Align	Default

First Name	Last Name	Active
MARY	SMITH	Active
PATRICIA	JOHNSON	Not active

Align

Allows you to specify the alignment of the control. Possible values are Default, Left, Right, and Center.

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the

element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

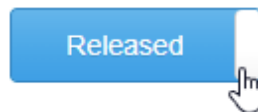
It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

Null label

This property allows you to define how NULL values are represented for this column. By default the value of this property corresponds to the one defined at the project level. You can specify a custom value for a certain column (for example, "Not selected", "Not available" "Not supported", etc.) if necessary.

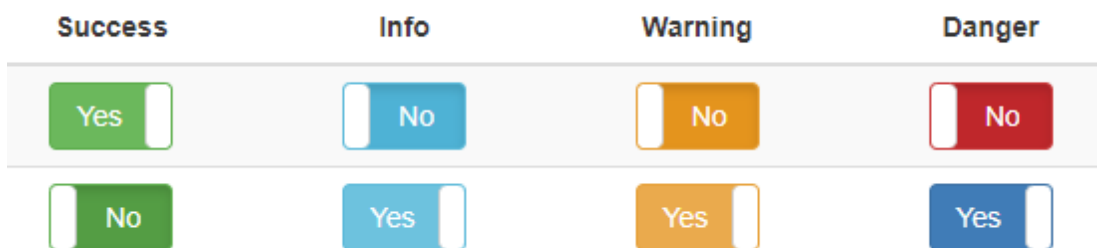
5.1.2.4 Toggle

This view control allows to change a logical column value in a data grid or in a card in one mouse click.

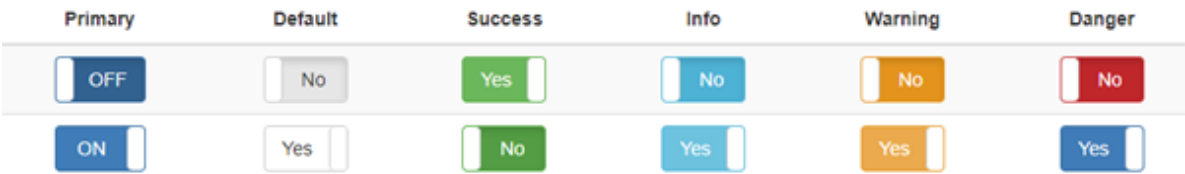


To make this control more informative, specify the ON and OFF toggle [Captions](#), select the [Size](#) and [Style](#) of these editors. The control appearance depends of the selected color scheme.

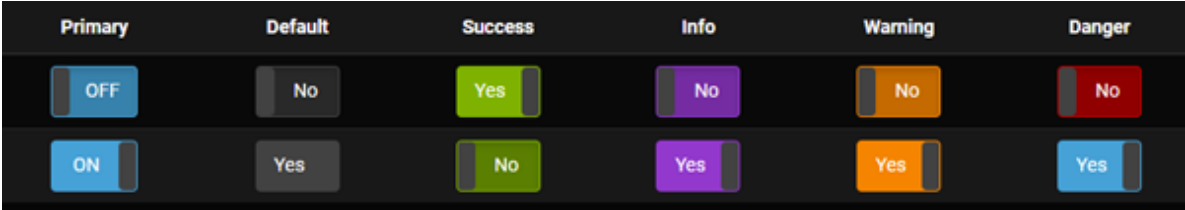
There are *Large*, *Medium*, *Small* and *Extra Small* sizes of toggles. The screen below demonstrates toggles of available sizes in the Default color scheme.



Available styles are: *Primary*, *Default*, *Success*, *Info*, *Warning*, *Danger*. The screen below demonstrates available toggle styles in the Default color scheme.



The same styles in a dark scheme:



Align

Allows you to specify the alignment of the control. Possible values are Default, Left, Right, and Center.

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. For example, to set the font color and the background color for a control, place the following string into Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

Null label

This property allows you to define how NULL values are represented for this column. By default the value of this property corresponds to the one defined at the project level. You can specify a custom value for a certain column (for example, "Not selected", "Not available" "Not supported", etc.) if necessary.

5.1.2.5 File download

This control allows you to represent BLOB data stored in the database as download links. For files stored outside of the database the [External File](#)^[55] control should be used. If the [Setup binary fields as image](#)^[335] option is turned ON, the [Image](#)^[53] control will be automatically selected for all the BLOB columns.



Use the **Additional** options to specify [templates](#)^[110] of file names and content type

(optional) to be used for binary files on download and enable the [Force downloading](#) option to force browser to download files, instead of open them in the browser.

Align

Allows you to specify the alignment of the control. Possible values are Default, Left, Right, and Center.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

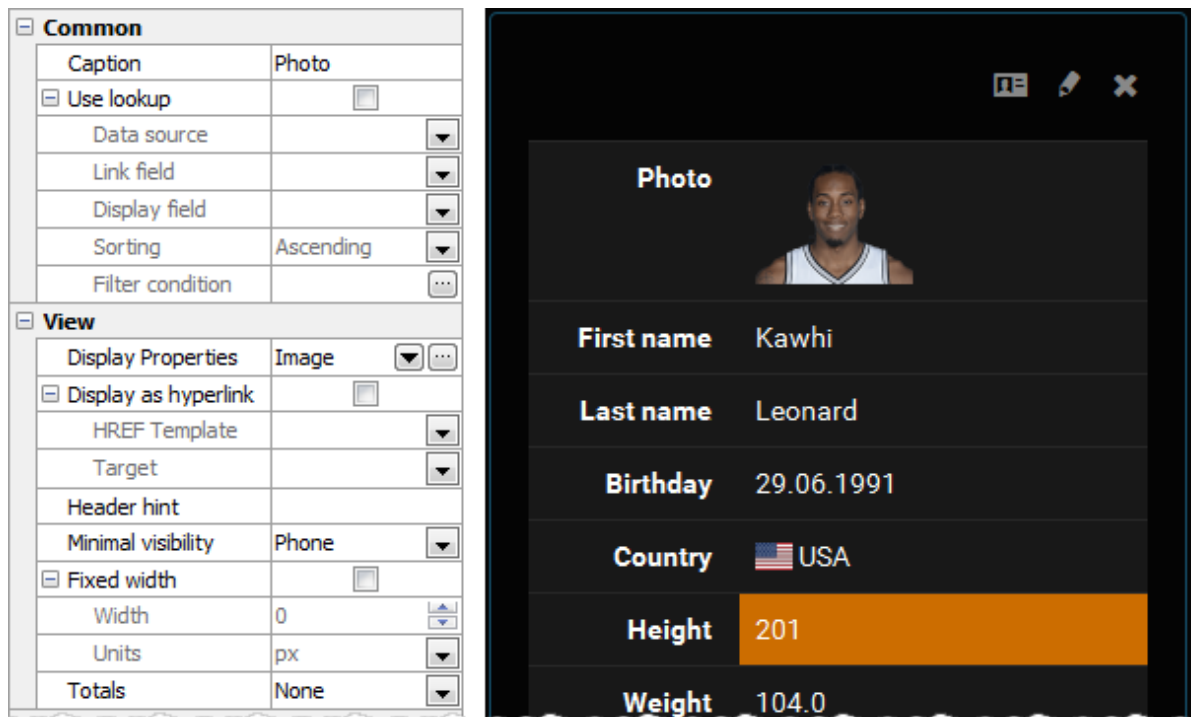
Null label

This property allows you to define how NULL values are represented for this column. By default the value of this property corresponds to the one defined at the project level. You can specify a custom value for a certain column (for example, "Not selected", "Not available" "Not supported", etc.) if necessary.

See also: [File upload](#)^[100], [Image upload](#)^[101]

5.1.2.6 Image

This control allows you to represent BLOB data stored in the database as images. For images stored outside of the database the [External Image](#)^[56] control should be used. If the [Setup binary fields as image](#)^[335] option is turned ON, this control will be automatically selected for all the BLOB columns.



The **Additional** attributes allows you to manipulate the size of represented images: to resize them to a specified height or width. To change image sizes, the appropriate [PHP extension](#) is required. You can also specify a [template](#)⁽¹¹⁰⁾ to be used for image hints to be shown when the mouse passes on an image.

Align

Allows you to specify the alignment of the control. Possible values are Default, Left, Right, and Center.

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

Null label

This property allows you to define how NULL values are represented for this column. By default the value of this property corresponds to the one defined at the project level.

You can specify a custom value for a certain column (for example, "Not selected", "Not available" "Not supported", etc.) if necessary.

See also: [Image upload](#)^[107], [File upload](#)^[100]

5.1.2.7 External File

This control is intended for representing links to external files. For files stored in the database the [File download](#)^[52] control should be used.



Source prefix and Source suffix

These properties allow you to improve the flexibility of the application. For example, if files are stored in the media directory, you can specify the value of the Prefix property as 'media/' and store in the database only short file names. If one day you decide to rename the directory, the only thing you will need to change is the value of this property. In short, the actual filename is calculated as follows: Source prefix + value_stored_in_the_database + Source suffix.

Enter topic text here.

Hint template

The value of this property is displayed as the value of the [title_attribute](#) of the corresponding web element at the generated page. You can use column names enclosed in percents (like %column_name%) to include the value(s) of the corresponding fields into the hint.

Align

Allows you to specify the alignment of the control. Possible values are Default, Left, Right, and Center.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

Null label

This property allows you to define how NULL values are represented for this column. By default the value of this property corresponds to the one defined at the project level.

You can specify a custom value for a certain column (for example, "Not selected", "Not available" "Not supported", etc.) if necessary.

5.1.2.8 External Image

This control is intended for displaying images stored in external files. For images stored in the database the [Image](#) ⁵³ control should be used.



Live examples

You can see this control in action at [Column Types](#) and [Image Galleries](#) pages in the [Feature Showcase](#) demo. Other examples are available at [Arenas](#) and [Roster](#) pages in the [NBA demo](#) (the last page also learns how to display images stored at an external website).

Source prefix and Source suffix

These properties allow you to improve the flexibility of the application. For example, if files are stored in the media directory, you can specify the value of the Prefix property as 'media/' and store in the database only short file names. If one day you decide to rename the directory, the only thing you will need to change is the value of this property. In short, the actual filename is calculated as follows: Source prefix + value_stored_in_the_database + Source suffix.

Enter topic text here.

Hint template

The value of this property is displayed as the value of the [title attribute](#) of the corresponding web element at the generated page. You can use column names enclosed in percents (like %column_name%) to include the value(s) of the corresponding fields into the hint.

Align

Allows you to specify the alignment of the control. Possible values are Default, Left, Right, and Center.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes

to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

Null label

This property allows you to define how NULL values are represented for this column. By default the value of this property corresponds to the one defined at the project level. You can specify a custom value for a certain column (for example, "Not selected", "Not available" "Not supported", etc.) if necessary.

Height and Width

Specifies the height and width for the image in pixels. You can specify any of these dimensions as well as both dimensions. If both height and width are set, the space required for the image is reserved when the page is loaded. However, without these attributes, the browser does not know the size of the image, and cannot reserve the appropriate space to it. The effect will be that the page layout will change during loading (while the images load). Please note that downsizing a large image with these properties forces a user to download the large image (even if it looks small on the page). To avoid this, use thumbnails (see below).

Is thumbnail

Defines whether this image is a thumbnail for another image. If this property is checked, you should specify the original image field name and, optionally, its prefix and suffix that are handled exactly as Source prefix and Source suffix properties described above.

5.1.2.9 External Audio

This control is intended for displaying and playing audio files with the standard [html5 audio player](#).



Source prefix and Source suffix

These properties allow you to improve the flexibility of the application. For example, if files are stored in the media directory, you can specify the value of the Prefix property as 'media/' and store in the database only short file names. If one day you decide to rename the directory, the only thing you will need to change is the value of this property. In short, the actual filename is calculated as follows: Source prefix + value_stored_in_the_database + Source suffix.

Enter topic text here.

Hint template

The value of this property is displayed as the value of the [title attribute](#) of the corresponding web element at the generated page. You can use column names enclosed in percents (like %column_name%) to include the value(s) of the corresponding fields into the hint.

Align

Allows you to specify the alignment of the control. Possible values are Default, Left, Right, and Center.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string into Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

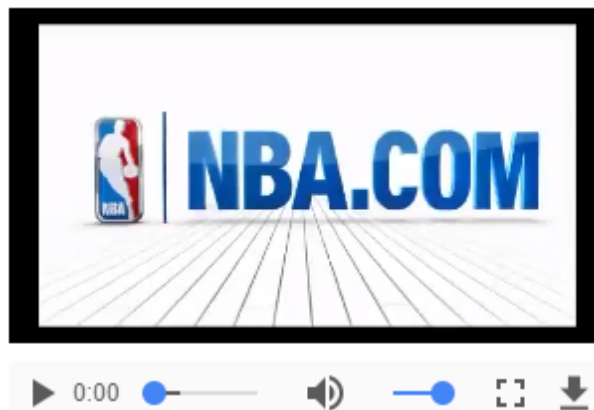
It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

Null label

This property allows you to define how NULL values are represented for this column. By default the value of this property corresponds to the one defined at the project level. You can specify a custom value for a certain column (for example, "Not selected", "Not available" "Not supported", etc.) if necessary.

5.1.2.10 External Video

This control is intended for displaying and playing raw (for example, locally stored) video files. For videos stored on a video hosting service like YouTube or Vimeo the [Embedded Video](#) control should be used.



Source prefix and Source suffix

These properties allow you to improve the flexibility of the application. For example, if files are stored in the media directory, you can specify the value of the Prefix property as 'media/' and store in the database only short file names. If one day you decide to rename the directory, the only thing you will need to change is the value of this property. In short, the actual filename is calculated as follows: Source prefix + value_stored_in_the_database + Source suffix.

Enter topic text here.

Hint template

The value of this property is displayed as the value of the [title_attribute](#) of the corresponding web element at the generated page. You can use column names enclosed in percents (like %column_name%) to include the value(s) of the corresponding fields into the hint.

Video player width and Video player height

Width and height of the [HTML5 video player control](#) used to display and play the video.

Align

Allows you to specify the alignment of the control. Possible values are Default, Left, Right, and Center.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string into Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:


```
data-city="Boston" data-lang="js" data-food="Bacon"
```

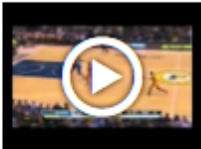
It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

Null label

This property allows you to define how NULL values are represented for this column. By default the value of this property corresponds to the one defined at the project level. You can specify a custom value for a certain column (for example, "Not selected", "Not available" "Not supported", etc.) if necessary.

5.1.2.11 Embedded Video

This control allows you to embed a video stored on a video hosting service like YouTube or Vimeo into the page (for videos stored in external files the [External Video](#)  control should be used). It is represented as a thumbnail of the video or as a play button (for low screen resolutions). Clicking the thumbnail or the button opens a modal window and starts playing the video.

Home team 1 ½	Home Score	Away Score	Away team	Recap
Indiana Pacers	97	87	Orlando Magic	

Align

Allows you to specify the alignment of the control. Possible values are Default, Left, Right, and Center.

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

Null label

This property allows you to define how NULL values are represented for this column. By default the value of this property corresponds to the one defined at the project level. You can specify a custom value for a certain column (for example, "Not selected", "Not available" "Not supported", etc.) if necessary.

5.1.2.12 Barcode

PostgreSQL PHP Generator allows to represent string (CHAR, VARCHAR, BINARY, VARBINARY, BLOB, TEXT, ENUM, and SET) columns as barcodes, symbols that can be scanned electronically using laser or image-based technology. To equip you application with barcodes, you need installed PHP 7.0 or higher.



To generate barcodes, select the **type of Barcode** and specify **Single bar width** and **Bar height** in pixels.

This version PostgreSQL PHP Generator supports Barcode types:

Code 32

Code 128

MSI

<i>Code 39</i>	<i>Code 128A</i>	<i>MSI + Checksum</i>
<i>Code 39 + Checksum</i>	<i>Code 128B</i>	<i>POSTNET</i>
<i>Code 39 Extended</i>	<i>Code 128C</i>	<i>PLANET</i>
<i>Code 39 Extended + Checksum</i>	<i>EAN-2</i>	<i>IMB</i>
<i>Code 93</i>	<i>EAN-5</i>	<i>Codabar</i>
<i>Standard 2 of 5</i>	<i>EAN-8</i>	<i>Code 11</i>
<i>Standard 2 of 5 + Checksum</i>	<i>EAN-13</i>	<i>One-Track</i>
<i>Interleaved 2 of 5</i>	<i>UPC-A</i>	<i>Pharmacode</i>
<i>Interleaved 2 of 5 + Checksum</i>	<i>UPC-E</i>	<i>Two-Track</i>
		<i>Pharmacode</i>

Bar color

Specify the color of barcodes foreground to make them visible in dark color schemes. The default bar color is black, but you can select white or specify the color in RGB format.

Align

Allows you to specify the alignment of the control. Possible values are Default, Left, Right, and Center.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

Null label

This property allows you to define how NULL values are represented for this column. By default the value of this property corresponds to the one defined at the project level. You can specify a custom value for a certain column (for example, "Not selected", "Not available" "Not supported", etc.) if necessary.

5.1.2.13 QR Code

PostgreSQL PHP Generator allows to represent string (CHAR, VARCHAR, BINARY, VARBINARY, BLOB, TEXT, ENUM, and SET) columns as QR codes, two-dimensional barcodes.



The size of generated QR-code depends of column values. Use the [Size factor](#) property to tune up QR codes size.

Make it possible to use QR codes as hyperlinks with the [view control options](#)⁴⁴.

Align

Allows you to specify the alignment of the control. Possible values are Default, Left, Right, and Center.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

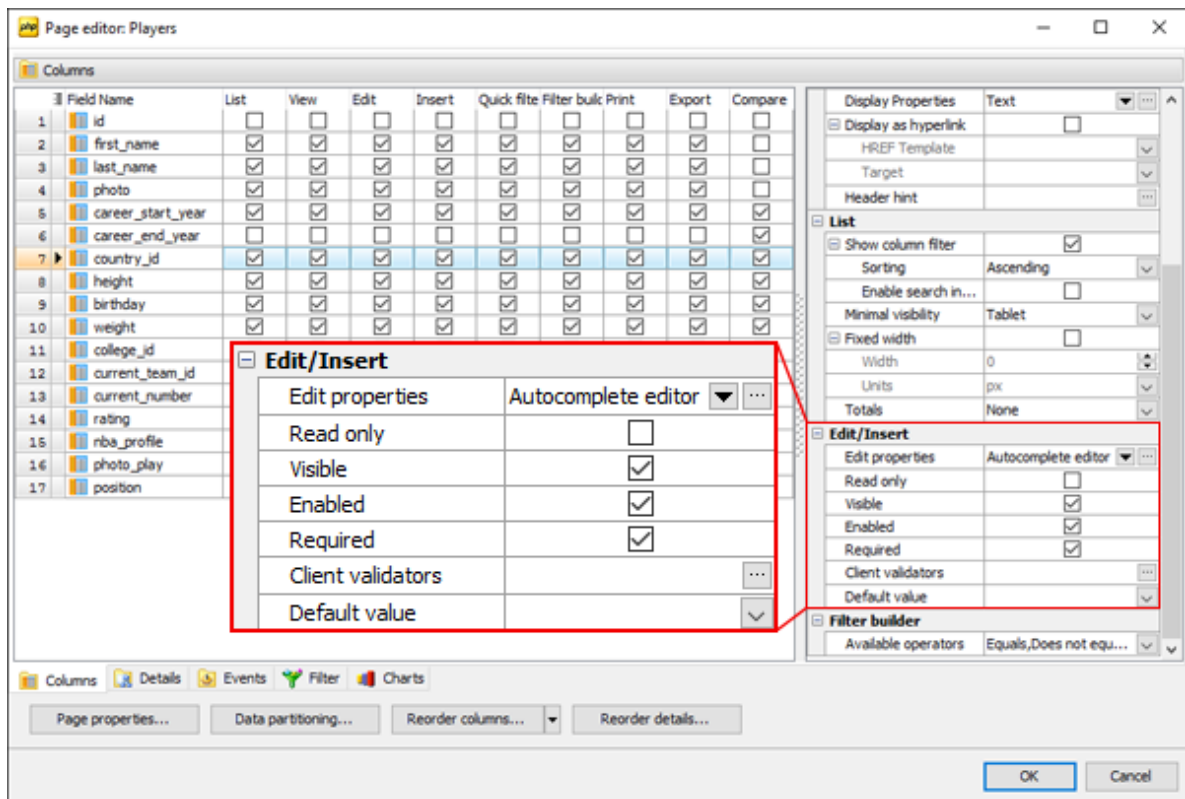
It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

Null label

This property allows you to define how NULL values are represented for this column. By default the value of this property corresponds to the one defined at the project level. You can specify a custom value for a certain column (for example, "Not selected", "Not available" "Not supported", etc.) if necessary.

5.1.3 Edit controls

The [Edit options](#) define the way the column data is represented in [data input forms](#) i.e. on [Edit](#) and [Insert](#) pages.



Edit properties

Use this drop-down list to select a control to be used for this column on **Edit** and **Insert** pages. Available controls are:

- [Text](#)^[66]
- [AutoComplete](#)^[67]
- [Radio group](#)^[68]
- [Combo box](#)^[70]
- [Dynamic Combobox](#)^[73]
- [Cascading Combobox](#)^[75]
- [Dynamic Cascading Combobox](#)^[79]
- [Check box](#)^[86]
- [Check box group](#)^[87]
- [Multiple Select](#)^[88]
- [DateTime](#)^[90]
- [Time](#)^[91]
- [Spin edit](#)^[92]
- [Range edit](#)^[93]
- [Color edit](#)^[94]

- [Mask edit](#) 
- [Text area](#) 
- [Html Wysiwyg](#) 
- [Password](#) 
- [File Upload](#) , [Image Upload](#) , [Upload File to Folder](#) , [Upload Image to Folder](#) 
- [Signature](#) 

To set additional control properties such as captions, formatting options, element attributes, and so on, use the dialog opened by the ellipsis button.

Common properties (applied to all editors)

Properties explained below are applied to all editors.

Read only

Use this option to make the control readonly. A readonly input field cannot be modified (however, a user can tab to it, highlight it, and copy the text from it). Readonly form elements will get passed to the form processor.

Visible

This option defines whether the control will be visible on the generated page. An invisible element stays in its original position and size.

Enabled

Use this option to specify whether the control will be enabled on the generated page. Turn it OFF to disable the control. Disabled input elements in a form will not be submitted.

Required

Turn this option ON if the field is mandatory (this is the default value for columns marked as NOT NULL in the database). In data input forms required columns are marked by the red asterisk. When a user tries to submit a form with an empty required field, data will not be submitted and an error message will be shown.

Hint

Use this field to provide controls with handy clues. These hints are displayed when a user hover over editor captions. Use HTML tags to make hints more readable.

Email	PATRICIA.JOHNSON@sakilacustomer.org
Address Id *	1121 Loja Avenue
Create Date *	2006-02-14 22:04:36

* - Required field

Default value

This option allows you to set the expected value of an input field with [string templates](#) ^[110]. You can use such environment variables as %CURRENT_DATETIME%, %CURRENT_DATE%, %CURRENT_TIME%, %CURRENT_USER_ID%, and %CURRENT_USER_NAME%. To specify a non-trivial default value for a column, use the [OnCustomDefaultValues](#) ^[193] event and [OnAddEnvironmentVariables](#) ^[148].

Client validation

PostgreSQL PHP Generator allows you to check for correctness of input data on the client side on two scopes:

1. The input value is validated **when a user leaves the control**. For this purpose, specify the suitable **Client validator**.

Range	The generated script validates if number is between the largest and smallest values.
Length range	The script validates if the length of text is between the min length and the max length.
Email	The script makes the element require a valid email.
Credit card	The script makes the element require a credit card number.
Number	The script makes the element require a decimal number.
URL	The script makes the element require a valid URL.
Digits	The script makes the element require digits only.
Regular expression	You can also specify your regular expression for data validation. Such expression is a pattern and every character entered in a form field is matched against that pattern – the form can only be submitted if the pattern and the user-input matches.

2. The whole data input form is validated when a user clicks the Save button. This may be useful to check the compatibility of input data. For this purpose, use the [OnInsertFormValidate](#) ^[139] and [OnEditFormValidate](#) ^[139] client side events. These events

occur before submitting of insert and edit forms accordingly and allow you to detect errors on the client side before the form is submitted to the server to avoid the round trip of information necessary for server-side validation.

5.1.3.1 Text

Select this control to create a single-line input field for entering text. [Live Demo](#).

First *

Last *

Career start *

Related editors

To create a multi-line input, use the [TextArea](#) editor. To allow user to enter only valid characters, use the [MaskEdit](#) editor. To provide user with a list of pre-defined values, use the [AutoComplete](#) editor. To restrict user to select only a pre-defined value, use [Radio_group](#), [Combobox](#), [Dynamic_Combobox](#), [Cascading_Combobox](#), or [Dynamic Cascading Combobox](#) editor.

Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Max length

Use this field to restrict the number of symbols of the value that can be entered.

Placeholder

Use this field to set a short hint that describes the expected value of an input field (e.g. a sample value or a short description of the expected format). The placeholder is displayed in the input field before the user enters a value.

Prefix/Suffix

Use these properties to create prepended and appended inputs allowing for simple punctuation or units to be paired with an input. For example, if you need to indicate a field is asking for money, use the prepend with a \$ sign. Other examples include @ with a username (a la Twitter.com's settings pages) and "+1" for phone number inputs.

Appended input with a placeholder:

Email

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```


Custom attributes

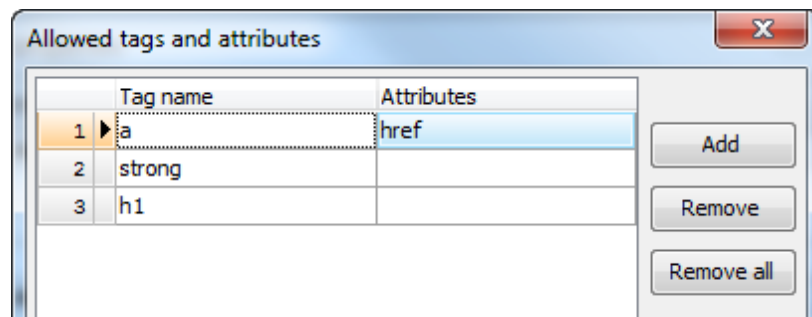
This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

HTML filter

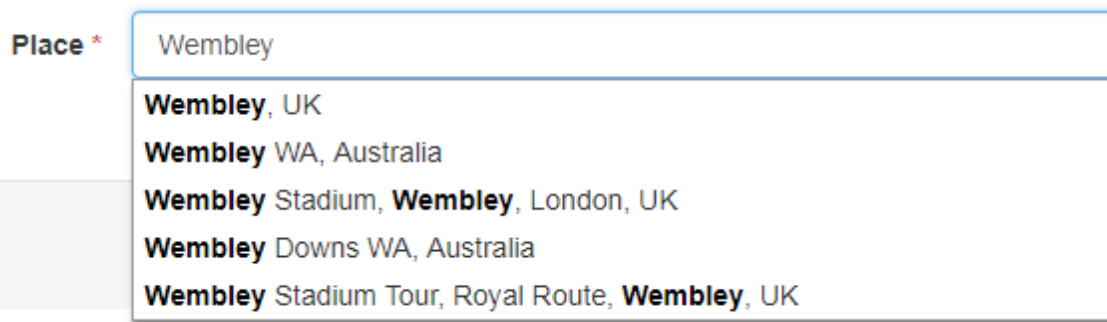
Use it to strip unwanted HTML tags and attributes from user input. By default all tags and attributes are stripped out, so you have to define allowed items explicitly.



Settings on the screenshot above are to allow the <a> tag (optionally with the href attribute as well as the and <h1> tags. All other tags and attributes will be removed from the user input.

5.1.3.2 AutoComplete

Select this control to create a single-line input field for entering text with a possibility of choosing a pre-defined value from a list. [Live Demo](#).



Related editors

To force user to select only from predefined values, use [Combobox](#) or [Dynamic Combobox](#) editors.

Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be

specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Max length

Use this field to restrict the number of symbols of the value that can be entered.

Placeholder

Use this field to set a short hint that describes the expected value of an input field (e.g. a sample value or a short description of the expected format). The placeholder is displayed in the input field before the user enters a value.

Prefix/Suffix

Use these properties to create prepended and appended inputs allowing for simple punctuation or units to be paired with an input. For example, if you need to indicate a field is asking for money, use the prepend with a \$ sign. Other examples include @ with a username (a la Twitter.com's settings pages) and "+1" for phone number inputs.

Appended input with a placeholder:

Email

Suggestions

This property group allows you to define the list of items to be displayed when a user enters a symbol in the control. [More information](#).

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

5.1.3.2.1 Suggestions

Suggestions for the [AutoComplete editor](#) can be based either on a [data source](#) (in this case you need to specify the data source name, field name, and the sort order) or on a PHP function with the following signature:

```
function OnGetSuggestions($term, &$suggestions);
```

Parameters:

\$term	Current input
\$suggestions	Array of suggestions for the current input

Example

The following example from our [Feature Demo](#) demonstrates how it is possible to use the Google API to get the list of suggestions. Don't forget to replace XXX to your [Google API Key](#).

```
$url = 'https://maps.googleapis.com/maps/api/place/queryautocomplete/json?key=%s&input=%s';
$key = 'XXX';
$sourceUrl = sprintf($url, $key, urlencode($term));

$contextOptions = array(
    "ssl" => array(
        "verify_peer" => false,
        "verify_peer_name" => false
    )
);

$json = file_get_contents($sourceUrl, false, stream_context_create($contextOptions));
$result = json_decode($json, true);
if ($result['status'] == 'OK') {
    foreach ($result['predictions'] as $prediction) {
        $suggestions[] = $prediction['description'];
    }
} elseif ($result['status'] == 'OVER_QUERY_LIMIT') {
    $suggestions[] = 'Query limit is exceeded';
}
```

Minimum input length

Use this option to specify minimal amount of symbols to start retrieving suggestions.

Number of values to display

Defines the maximum number of suggestions to be displayed in the drop-down list.

5.1.3.3 Radio group

Select this control to let the visitor select one option from a (small) set of alternatives. [Live Demo](#).

Rating ☐ G ☐ PG ☒ PG-13 ☐ R ☐ NC-17

Related editors

To allow user to select multiple options at the same time, use [Check box group](#) or [Multiple Select](#) editors instead. For medium or large sets of options use [Combobox](#) or [Dynamic Combobox](#) editors.

To allow user to select the value step-by-step (for example, first select a country, then a city in the selected country), use [Cascading Combobox](#) or [Dynamic Cascading Combobox](#) editors.

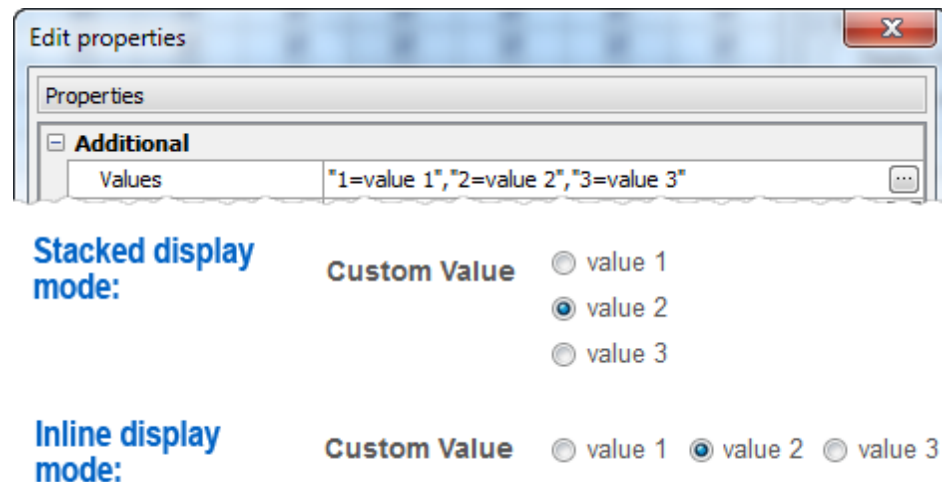
Max width

Use this property to restrict the maximum width of the editor (this means that in any

screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Values

You can specify values to be represented as radio group values instead of stored ones. For this purpose use the dialog opened by the ellipsis button, or add them manually as pairs *allowed_value=value_to_be_represented* separated by commas (Example: *1=One, 2=Two*).



Display mode

Use the drop-down list to select whether the radio group will be represented on the same column (*Stacked*) or on the same line (*Inline*).

Inline styles

Use this field to set formatting options to be used inside the *style* attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

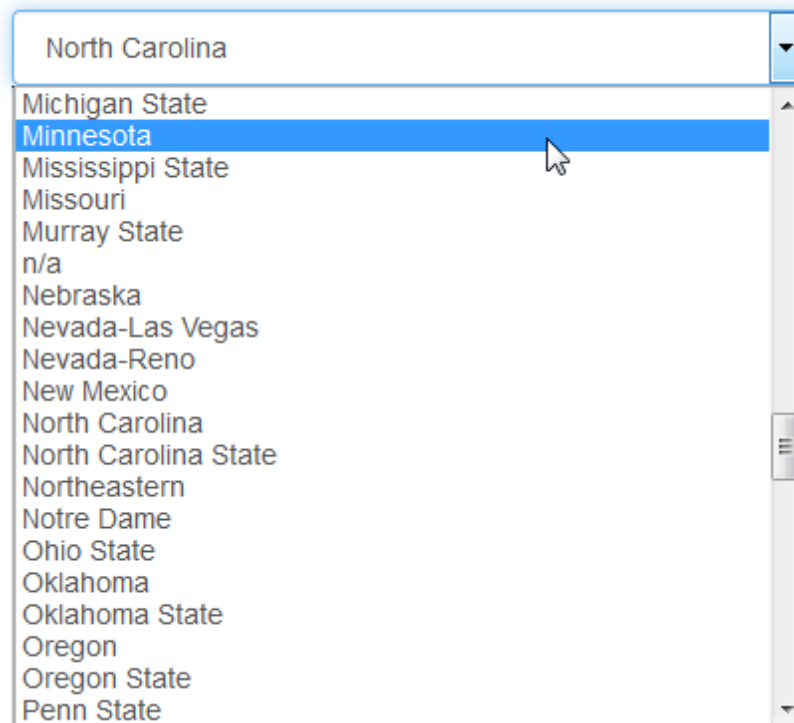
This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with *data-* to keep the result document compatible with the HTML5 requirements.

5.1.3.4 Combobox

Select this control to let the visitor select one option from a (medium) set of alternatives. [Live Demo](#).

College**Related editors**

To allow user to select multiple options at the same time, use [Check_box_group](#) or [Multiple_Select](#) editors instead. For large sets of options use the [Dynamic_Combobox](#) editor.

To allow user to select the value step-by-step (for example, first select a country, then a city in the selected country), use [Cascading_Combobox](#) or [Dynamic_Cascading_Combobox](#) editors.

Max width

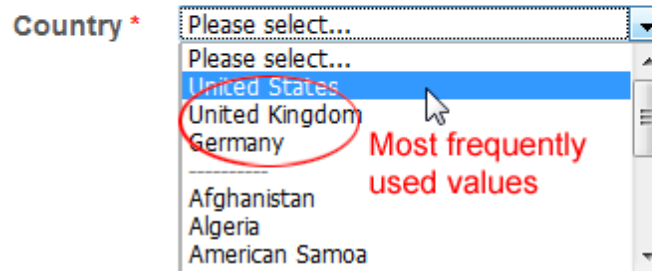
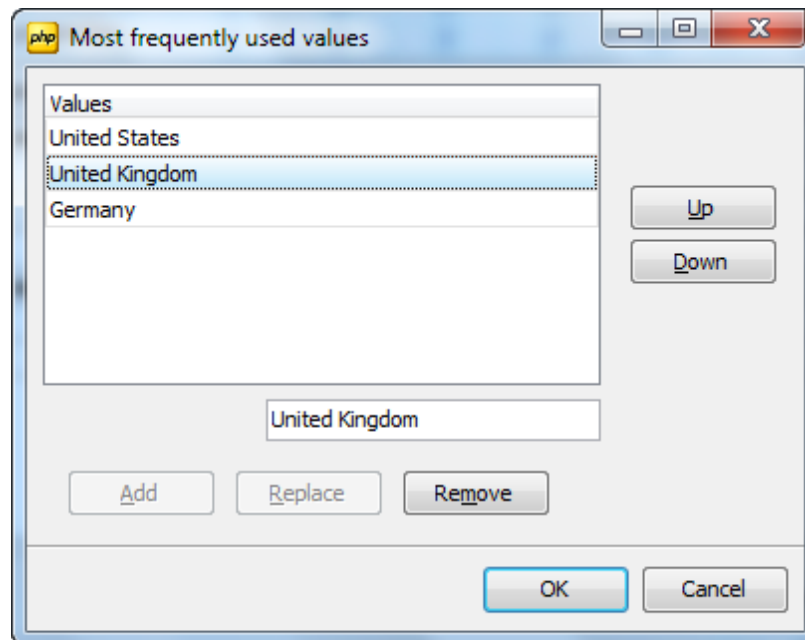
Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Values

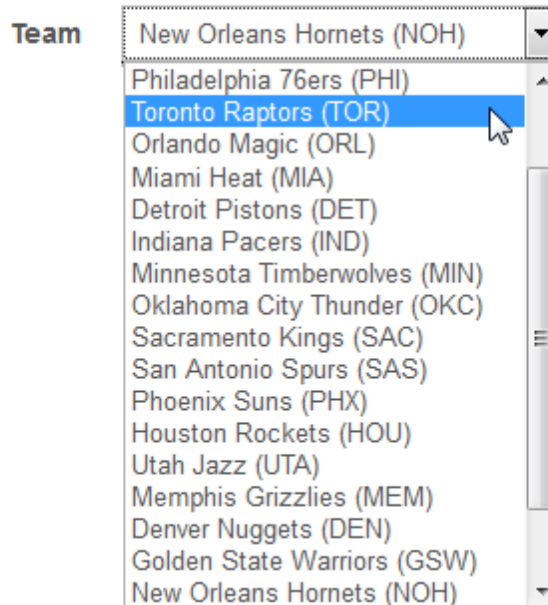
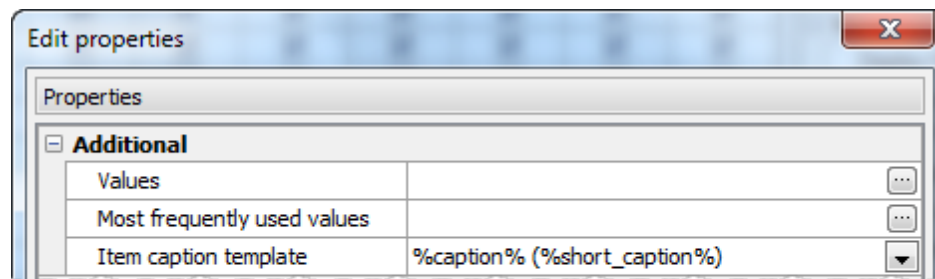
Use this field to fill the combobox with more convenient values instead of stored ones. For this purpose use the Value list window or enter them manually as pairs *allowed_value=value_to_be_represented* separated by commas (Example: *1=One, 2=Two*).

Most frequently used values

Use the ellipsis button next to this field to specify combobox values to be always displayed at the top of the drop-down list.

**Item caption template**

This option allows you to populate the combobox with values of several columns. For this purpose, specify a [template](#)^[110] to be used for each item in the list.



Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

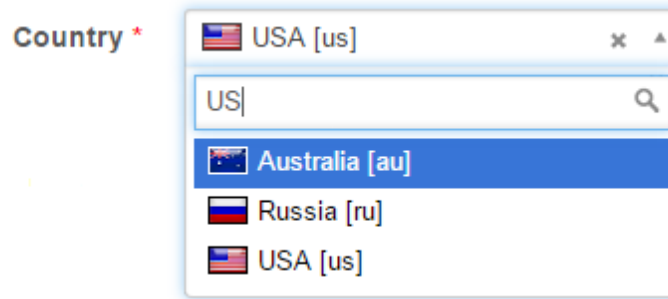
This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

5.1.3.5 Dynamic Combobox

Select this control to create a combination of a single-line editable textbox and a drop-down list filtered according to values entered in the textbox. It is recommended to use this editor type as lookup editor to work with a big number of records. [Live Demo](#).



Related editors

In Dynamic Combobox items are filled on-the-fly, so this control is the best choice when you have a sufficiently large number of items. If number of items is not exceed 10-15, consider to use "classic" [Combobox](#)^[70] as it works slightly faster.

To allow user to select the value step-by-step (for example, first select a country, then a city in the selected country), use [Cascading Combobox](#)^[71] or [Dynamic Cascading Combobox](#)^[72] editors.

Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Allow clear

Turn it ON to allow end users to clear the selected value using special button.

Minimum input length

Use this option to specify minimal amount of symbols to start the search (useful for large lookup datasets where short search terms are not very useful).

Number of values to display

Defines the number of lookup values to be displayed in the drop-down list.

Item caption template

This option allows you to populate the combobox with values of several columns. For this purpose, specify a [template](#)^[110] to be used for each item in the list.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```


It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

Formatting functions

Use these functions to format **search results** and the **selection**. For example, it is possible to use fonts, colors, images, and so on. All you need is to specify two functions in JavaScript each of which accepts the current item as a parameter and returns the HTML code to represent search results and the selected item accordingly. In the example below search results are displayed in italic and the selected item is displayed in bold.

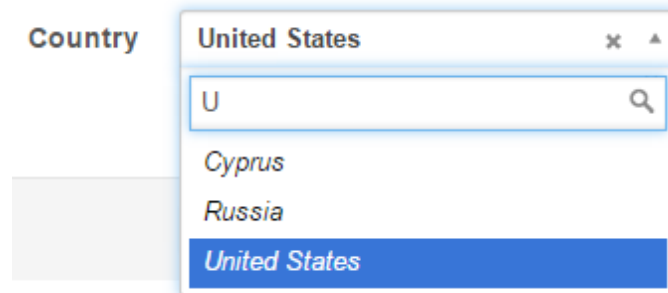
formatResult function:

```
return '<i>' + item.text + '</i>';
```

formatSelection function:

```
return '<b>' + item.text + '</b>';
```

The editor now looks as follows:



5.1.3.6 Cascading Combobox

Cascading Combobox is a series of 2 or more [comboboxes](#) in which each combobox is filtered according to the selection in the upper combobox. Select this editor to get greater control over data input, and to make things easier for the user. The number of levels is not limited. [Live Demo](#).

Address	Country	Netherlands	▼	+
	City	Amersfoort	▼	+
	Address	992 Klerksdorp Loop	▼	+

Note: regardless of the number of levels this control is intended to select a single value.

Related editors

In Cascading Combobox items in all levels are filled statically. If all or any of the levels can contain a sufficiently large number of items, consider to use [Dynamic Cascading Combobox](#).

Max width

Use this property to restrict the maximum width of the editor (this means that in any

screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

Levels

Here you can select and setup data sources to be used for additional controls. Levels should be added in the reverse order of logical priority, i.e. if our goal is to create a 3-level control that allows user to select a country, then a city in the selected country, and then an address in the selected city, we need to add the City level first and the Country level after that.

To add a new level, click [Add level](#) and setup its properties as described below. Click [Edit level](#) to edit the properties of an existing level. To delete a level, click [Remove level](#) (only a level from the top of hierarchy can be removed).

Level properties

The following properties can be set for each level:

Caption

Caption of the corresponding control.

Parent data source

Data source for the level. The value of this property can be edited only for the top level (Country in our example).

Key field

A column from the data source above to be used to filter values in the low level control. The value of this property can be edited only for the top level (Country in our example).

Display field

A column, which values are displayed at this level.

Sorting

Defines the sort order for the level's items. Possible values are Ascending, Descending, and None. The last one allows to use a native sorting order of their data sources.

Filter condition

This property allows you to restrict number of displayed values. [Variables](#)^[195] like %CURRENT_USER_ID% are allowed.

Add new items on the fly

Defines whether a user is allowed to add items at this level. If enabled, a plus button is displayed on the right of each level. Pressing this button opens a modal window where user can enter a value that does not exist in the item list.

Child data source

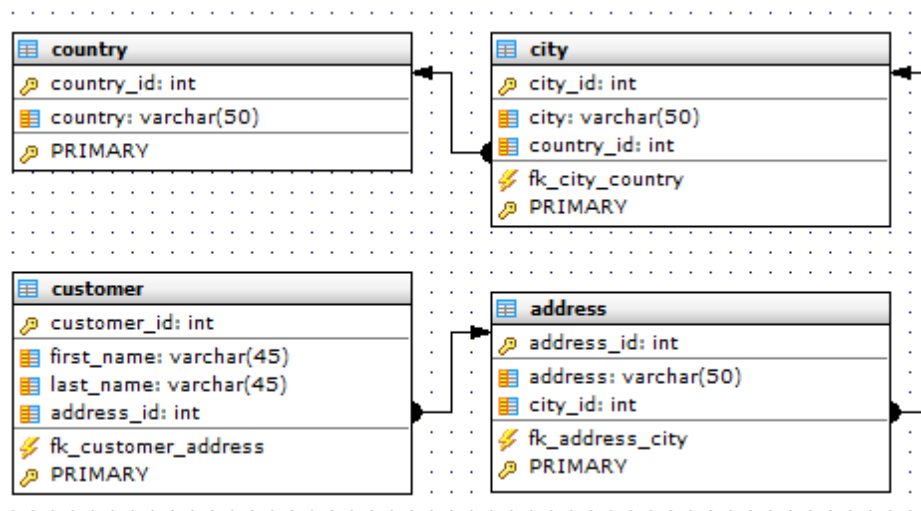
The parent data source for the previous level. For the first level it is the lookup data source defined for the "initial" column, i.e. for the column for which we want to use this editor. This property cannot be edited.

Child field to filter

Field from the child data source, which values will be filtered by the values of the Key field property (see above). This property can be edited only for the top level (Country).

Example

Let's consider step-by-step building of Country -> City -> Address control. Assume our database schema is as follows:



Our goal is to create a three-level editor like this one:

Address	Country	Netherlands	▼	+
	City	Amersfoort	▼	+
	Address	992 Klerksdorp Loop	▼	+

1. Open [Page Editor](#)^[33] for a table containing the `address_id` column, select this column and specify its [lookup properties](#)^[37]. If a foreign key constraint is defined and the [Setup lookup by foreign key](#)^[335] option is enabled, these properties are set automatically.

The 'Common' properties dialog shows the following settings:

Property	Value	Action
Caption	Address	...
Use lookup	<input checked="" type="checkbox"/>	
Data source	address	▼
Link field	id	▼
Display field	caption	▼
Sorting	Ascending	▼
Filter condition		...
Add new items on the fly	Enabled	▼ ...

2. Select *Cascading Combobox* as edit control and click the ellipsis button to open the [Edit properties](#) dialog.

The 'Edit/Insert' properties dialog shows the following settings:

Property	Value	Action
Edit properties	Cascading Combobox	▼ ...
Read only	<input type="checkbox"/>	
Visible	<input checked="" type="checkbox"/>	
Enabled	<input checked="" type="checkbox"/>	
Required	<input type="checkbox"/>	
Client validators		...
Default value		▼

3. As described above, data sources for additional levels are placed in the reverse order of logical priority. The data entry order would be Country, City, and then Address, so the first additional level is intended for selecting a city. To add a new level for the control, click [Add level](#) and fill fields in the dialog window as follows:

The 'Edit level' dialog box is shown with the following settings:

- Caption: City
- Parent data source: city
- Key field: ID
- Display field: Name
- Sorting: Ascending
- Filter condition: (empty)
- Add new items on the fly: Enabled
- Child data source: address
- Child field to filter: city_id

Buttons: OK, Cancel

4. Add the next level in the same way:

The 'Edit level' dialog box is shown with the following settings:

- Caption: Country
- Parent data source: country
- Key field: Code
- Display field: Name
- Sorting: Ascending
- Filter condition: (empty)
- Add new items on the fly: Enabled
- Child data source: city
- Child field to filter: CountryCode

Buttons: OK, Cancel

That's all. Click OK to save all settings and return to the Page Editor.

5.1.3.7 Dynamic Cascading Combobox

Dynamic Cascading Combobox is a series of 2 or more [dynamic comboboxes](#) in which each combobox is filtered according to the selection in the upper combobox. Select this editor to get greater control over data input, and to make things easier for the user.

The number of levels is not limited. [Live Demo](#).

The screenshot shows a form with three cascading dropdown menus. The first dropdown, labeled 'Country', has 'United States' selected. The second dropdown, labeled 'City', has 'Please select...' selected. The third dropdown, labeled 'Address', has 'ne' entered and is open, displaying a list of cities. 'New York' is highlighted in the list. Each dropdown menu has a small plus sign button to its right.

Note: regardless of the number of levels this control is intended to select a single value.

Related editors

In Dynamic Cascading Combobox items in all levels are filled on-the-fly, so this control is the best choice when all or any of the levels can contain a sufficiently large number of items. If number of items is not exceed 10 for all levels, consider to use the [Cascading Combobox](#) as it works slightly faster.

Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Allow clear

Turn it ON to allow end users to clear the selected value using special button.

Minimum input length

Use this option to specify minimal amount of symbols to start the search (useful for large lookup datasets where short search terms are not very useful).


Number of values to display


Defines the number of lookup values to be displayed in the drop-down list for each level.


Formatting functions


Use these functions to format search results and the selection. For example, it is possible to use fonts, colors, images, and so on. All you need is to specify two functions in JavaScript each of which accepts the current item as a parameter and returns the HTML code to represent search results and the selected item accordingly. Formatting functions can be specified at the level basis. The screenshot below is from our [NBA demo application](#):


Home team *


Conference  Eastern x ▾

Division  Southeast x ▾

Away team  Atlanta Hawks x ▲

M 

 Miami Heat

 Orlando Magic

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

Levels

Here you can select and setup data sources to be used for additional controls. Levels should be added in the reverse order of logical priority, i.e. if our goal is to create a 3-level control that allows user to select a country, then a city in the selected country, and then an address in the selected city, we need to add the City level first and the Country level after that.

To add a new level, click [Add level](#) and setup its properties as described below. Click [Edit level](#) to edit the properties of an existing level. To delete a level, click [Remove level](#) (only a level from the top of hierarchy can be removed).

Level properties

The following properties can be set for each level:

Caption

Caption of the corresponding control.

Parent data source

Data source for the level. The value of this property can be edited only for the top level (Country in our example).

Key field

A column from the data source above to be used to filter values in the low level control. The value of this property can be edited only for the top level (Country in our example).

Display field

A column, which values are displayed at this level.

Sorting

Defines the sort order for the level's items. Possible values are Ascending, Descending, and None. The last one allows to use a native sorting order of their data sources.

Filter condition

This property allows you to restrict number of displayed values. [Variables](#)¹⁹⁵ like %CURRENT_USER_ID% are allowed.

Add new items on the fly

Defines whether a user is allowed to add items at this level. If enabled, a plus button is displayed on the right of each level. Pressing this button opens a modal window where user can enter a value that does not exist in the item list.

Child data source

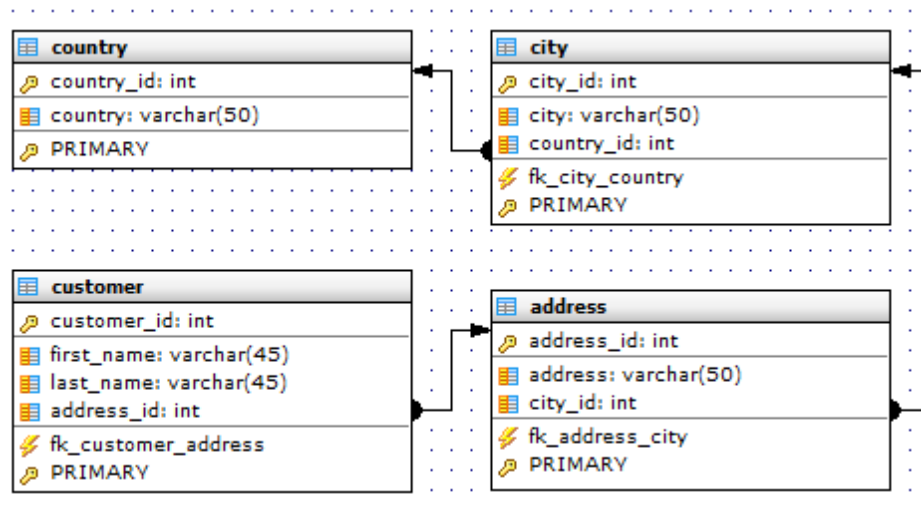
The parent data source for the previous level. For the first level it is the lookup data source defined for the "initial" column, i.e. for the column for which we want to use this editor. This property cannot be edited.

Child field to filter

Field from the child data source, which values will be filtered by the values of the Key field property (see above). This property can be edited only for the top level (Country).

Example

Let's consider step-by-step building of Country -> City -> Address control. Assume our database schema is as follows:



Our goal is to create a three-level editor like this one:

Country	United States	+
City	Please select...	+
Address	<input type="text" value="ne"/> <div> <div>New Haven</div> <div>New Orleans</div> <div>New York</div> <div>Newark</div> <div>Newport News</div> <div>Pembroke Pines</div> <div>Spokane</div> </div>	+

1. Open [Page Editor](#) for a table containing the `address_id` column, select this column and specify its [lookup properties](#). If a foreign key constraint is defined and the [Setup lookup by foreign key](#) option is enabled, these properties are set automatically.

Common	
Caption	Address
Use lookup	<input checked="" type="checkbox"/>
Data source	address
Link field	id
Display field	caption
Sorting	Ascending
Filter condition	
Add new items on the fly	Enabled

2. Select *Dynamic Cascading Combobox* as edit control and click the ellipsis button to open the [Edit properties](#) dialog.

Edit/Insert	
Edit properties	Dynamic Cascading Combobox
Read only	<input type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Enabled	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Client validators	
Default value	

3. As described above, data sources for additional levels are placed in the reverse order of logical priority. The data entry order would be Country, City, and then Address, so the first additional level is intended for selecting a city. To add a new level for the control, click [Add level](#) and fill fields in the dialog window as follows:

Edit level	
Caption	City
Parent data source	city
Key field	ID
Display field	Name
Sorting	Ascending
Filter condition	
Add new items on the fly	Enabled
Child data source	address
Child field to filter	city_id

OK Cancel

4. Add the next level in the same way:

That's all. Click OK to save all settings and return to the Page Editor.

5.1.3.8 Check box

Select this control to let the user to make a binary choice. This type of editor is automatically selected for boolean and columns.

Checkbox ☒

Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```


It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

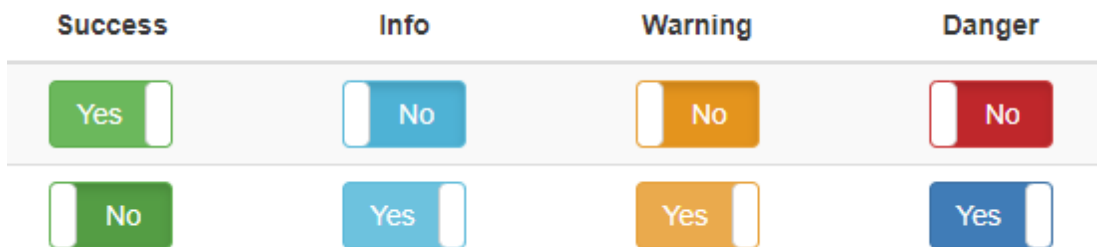
5.1.3.9 Toggle

Use this control allow webpage users edit logical data with one-click editors.

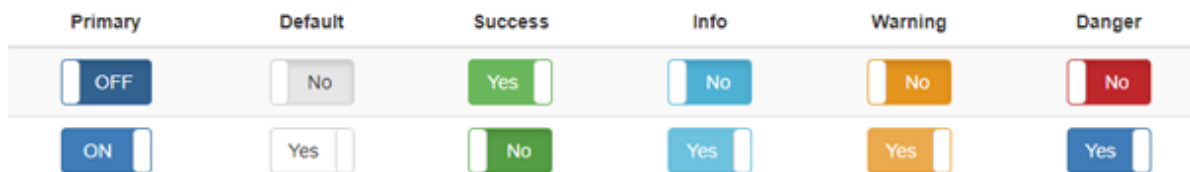


To make this control more informative, specify the ON and OFF toggle [Captions](#), select the [Size](#) and [Style](#) of these editors. The control appearance depends of the selected color scheme.

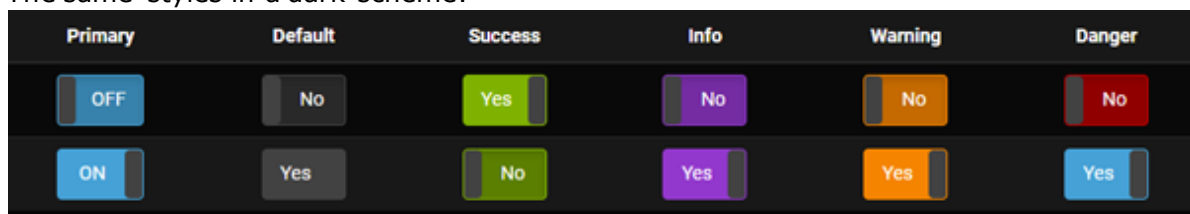
There are *Large*, *Medium*, *Small* and *Extra Small* sizes of toggles. The screen below demonstrates toggles of available sizes in the Default color scheme.



Available styles are: *Primary*, *Default*, *Success*, *Info*, *Warning*, *Danger*. The screen below demonstrates available toggle styles in the Default color scheme.



The same styles in a dark scheme:



Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the

element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```


Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

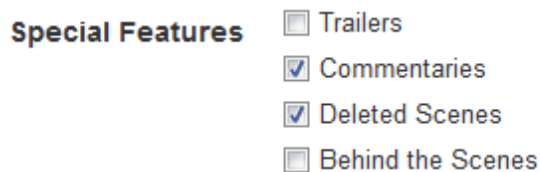
5.1.3.10 Check box group

Select this control to provide users with the ability to select multiple items. You can also use the [Multiple select](#)  control for this purpose. The difference between these editors are:

- [Check box group](#) demonstrates all values available for selection,
- [Multiple select](#) provides adding new items using drop-down combobox and requires less space in the form. [Live Demo](#).

Changing the control type to [Multiple select](#) creates this multi-choice control with the same properties.

In case you use [Check box group](#) to work with a table column, the corresponding [Filter Builder](#) uses [Multiple select](#) for this column.



Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Values


You can separate values to be represented in the control and values stored in the database. For this purpose use the dialog opened by the ellipsis button and enter the values you need, or add them manually as pairs *stored_value=value_to_be_represented* separated by commas (Example: *1=One, 2=Two*).

Display mode

Use the drop-down list to select whether the check box group will be represented on the same column ([Stacked](#)) or on the same line ([Inline](#)).

Retrieve values from database

Turn this option ON to fill in the control with values from a database.

<input checked="" type="checkbox"/> Retrieve values from database	<input checked="" type="checkbox"/>
Data source	 iphone_color
Stored field	id
Displayed field	color
Sorting	Ascending
Filter condition	...
Add new items on the fly	Enabled

Data source

Data source name to retrieve values from.

Stored field

Data source field name to retrieve stored values from.

Displayed field

Data source field name to retrieve displayed values from.

Sorting

Sorting order of displayed values in the editor.

Filter condition

Allows to reduce the list of values represented in the editor with a specified criteria. This condition corresponds to the WHERE clause applied to the data source (you must not add the WHERE keyword to beginning of the condition). The following operators can be used in this clause: =, <, > (!=), >, <, >=, <=, BETWEEN, LIKE, IN. It is also possible to use [predefined variables](#) like %CURRENT_USER_NAME%.

Add new item on the fly

Turn this option ON to allow adding new items directly in the editor. When this option is checked, an 'Insert item' link is displayed on the bottom of the editor.

Click the ellipsis button at the right to customize the modal dialog to be displayed when adding a new item. For example, you might want to hide some fields and/or customize the form layout.

Note: If values are retrieved from a database, hard-coded values described above are ignored and NOT included into the list of available choices.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

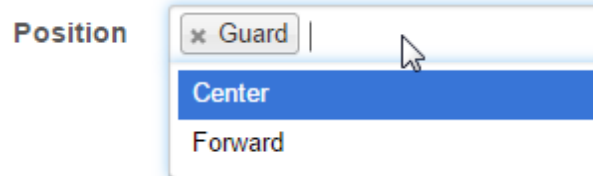

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

5.1.3.11 Multiple select

Select this control to provide the user with the ability to select multiple items. It provides the same functionality as [Checkbox Group](#) (i.e. allows you to select multiple options for a single field) but looks much different and requires less space in the forms. [Live Demo](#).

Changing the control type to [Checkbox Group](#) creates this multi-choice control with the same properties.



Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Values

You can separate values to be represented in the control and values stored in the database. For this purpose use the dialog opened by the ellipsis button and enter the values you need, or add them manually as pairs *stored_value=value_to_be_represented* separated by commas (Example: *1=One, 2=Two*).

Maximum selection size

This property allows you to restrict number of items that can be selected simultaneously.

Retrieve values from database

Turn this option ON to fill in the control with values from a database.

<input checked="" type="checkbox"/> Retrieve values from database	<input checked="" type="checkbox"/>
Data source	iphone_color
Stored field	id
Displayed field	color
Sorting	Ascending
Filter condition	...
Number of values to display	20
Add new items on the fly	Enabled

Data source

Data source name to retrieve values from.

Stored field

Data source field name to retrieve stored values from.

Displayed field

Data source field name to retrieve displayed values from.

Sorting

Sorting order of displayed values in the editor.

Filter condition

Allows to reduce the list of values represented in the editor with a specified criteria. This condition corresponds to the WHERE clause applied to the data source (you must not add the WHERE keyword to beginning of the condition). The following operators can be used in this clause: =, <> (!=), >, <, >=, <=, BETWEEN, LIKE, IN. It is also possible to use [predefined variables](#)^[195] like %CURRENT_USER_NAME%.

Number of values to display

Number of values to be displayed in the dropdown list of the editor.

Add new item on the fly

Turn this option ON to allow adding new items directly in the editor. When this option is checked, a plus button is displayed on the right of the editor.

Click the ellipsis button at the right to customize the modal dialog to be displayed when adding a new item. For example, you might want to hide some fields and/or customize the form layout.

Note: If values are retrieved from a database, hard-coded values described above are ignored and NOT included into the list of available choices.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

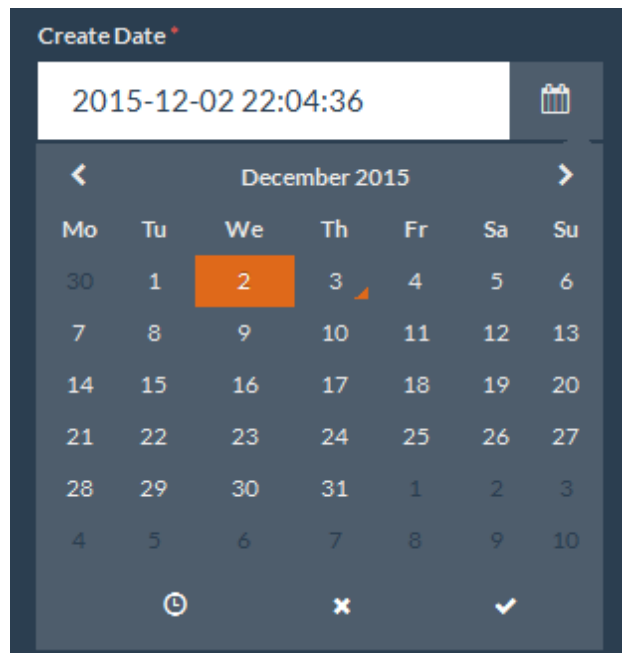
This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. Foreexample, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

5.1.3.12 DateTime

Select this control to let the user to enter date time values with datepicker, a combination of an interactive calendar and a single-line editable textbox.



Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

By default, the format of the column's data is the same as it is set at the [Project options](#). To use another Date time format, uncheck the [Default format](#) option and select it from the corresponding field to setup a format to be applied to the input value.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

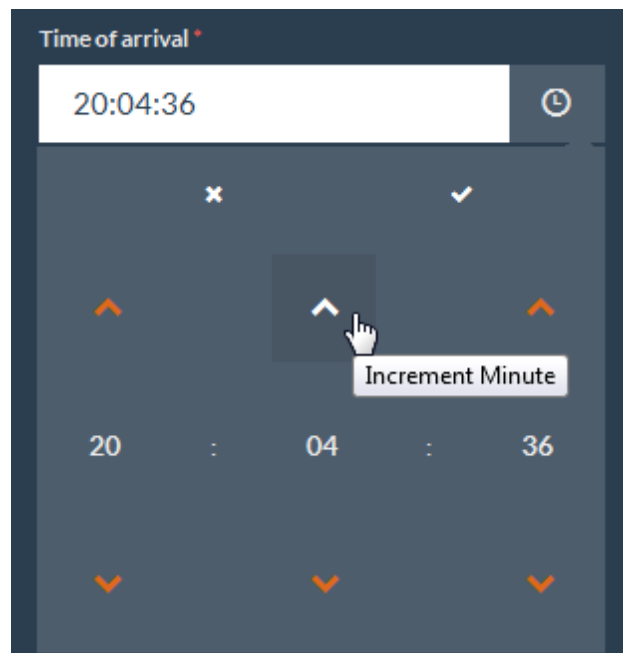
This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

5.1.3.13 Time

Select this control to let the user to enter time values in a single-line editable textbox provided by a spinner helping to quickly change the editor's value as well as to reset its value to the current time.



Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

5.1.3.14 Spin edit

Select this control to create a single-line input field allowing users to edit numeric values with ease. The editor consists of an edit region and one pairs of spin buttons which can be used to adjust the numerical value. [Live demo](#).



Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Step specifies the legal number intervals for the input field. When an end-user presses a spin button, the value is incremented or decremented by this amount. The default value is 1.

To limit end-user input to a specified range, turn ON the **Use constraints** option and define the editor's minimum and maximum allowed values.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

5.1.3.15 Range edit

The range input (so-called slider) is useful for imprecise number input. It is possible to indicate the allowed range of values in the appropriate dialog window (default values are 0 and 100 for bottom and top limits accordingly). [Live demo](#).

**Max width**

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Step specifies the legal number intervals for the input field. When an end-user presses a spin button, the value is incremented or decremented by this amount. The default value is 1.

To limit end-user input to a specified range, turn ON the **Use constraints** option and define the editor's minimum and maximum allowed values.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

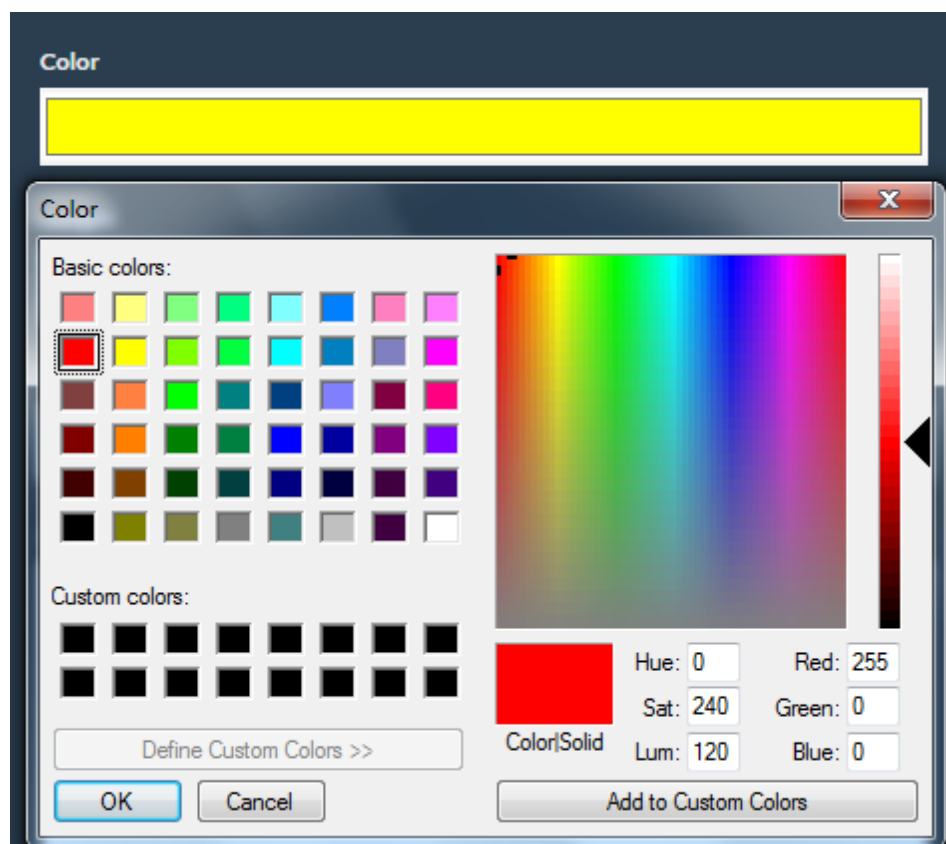
This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

5.1.3.16 Color edit

Use this control to edit values of columns storing HTML color codes with a native-browser Color Picker (it is assumed that a string data type is used for storing such values in the database).



Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

5.1.3.17 Mask edit

Use this control to define an edit box that limits the user to a specific format (dates, phone numbers, etc) and accepts only valid characters.

On input	<input type="text"/>
Result	<input type="text" value="564-743-5372"/>

Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Mask

A mask is defined by a format made up of mask literals and mask definitions. Any character not in the definitions list below is considered a mask literal. Mask literals will be automatically entered for the user as they type and will not be able to be removed by the user. The following mask definitions are predefined:

- a** - Represents an alpha character (A-Z,a-z)
- 9** - Represents a numeric character (0-9)
- *** - Represents an alphanumeric character (A-Z,a-z,0-9)

Example

To define an edit box for telephone numbers, specify the following string as the mask of the editor.

```
999-999-9999
```

In this case the created data entry field accepts only numeric input and if a user then tries to enter a letter in this edit box, the application will not accept it.

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the

element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

5.1.3.18 Text area

Use this control to provide the Edit form with a multi-line text input control. A text area can hold an unlimited number of characters, and the text renders in a fixed-width font (usually Courier). [Live Demo](#).

About

The Boston Celtics is owned by Wycliffe Grousbeck and coached by Doc Rivers, with Danny Ainge as the President of Basketball Operations. Founded in 1946, their 17 NBA Championships are the most for any NBA franchise. The Celtics' greatest domination came from 1957 to 1969, with 11 championships in 13 years, and eight in a row, the longest consecutive championship winning streak of any North American professional sports team. They currently play their home games at TD Garden.

Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

The size of a text area can be specified by the **Column count** and **Row count** values.

Placeholder

Use this field to set a short hint that describes the expected value of an input field (e.g. a sample value or a short description of the expected format). The placeholder is displayed in the input field before the user enters a value.

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```


Custom attributes

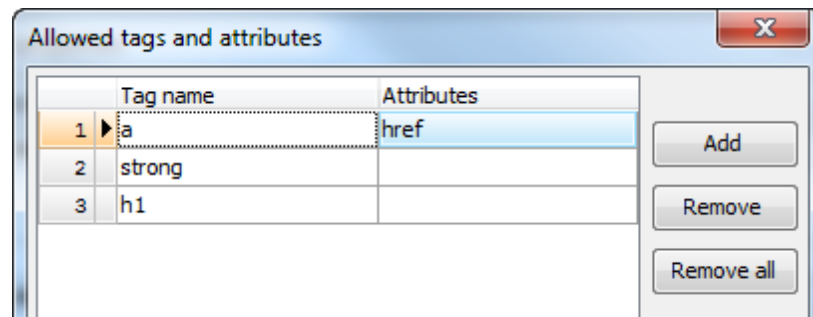
This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

HTML filter

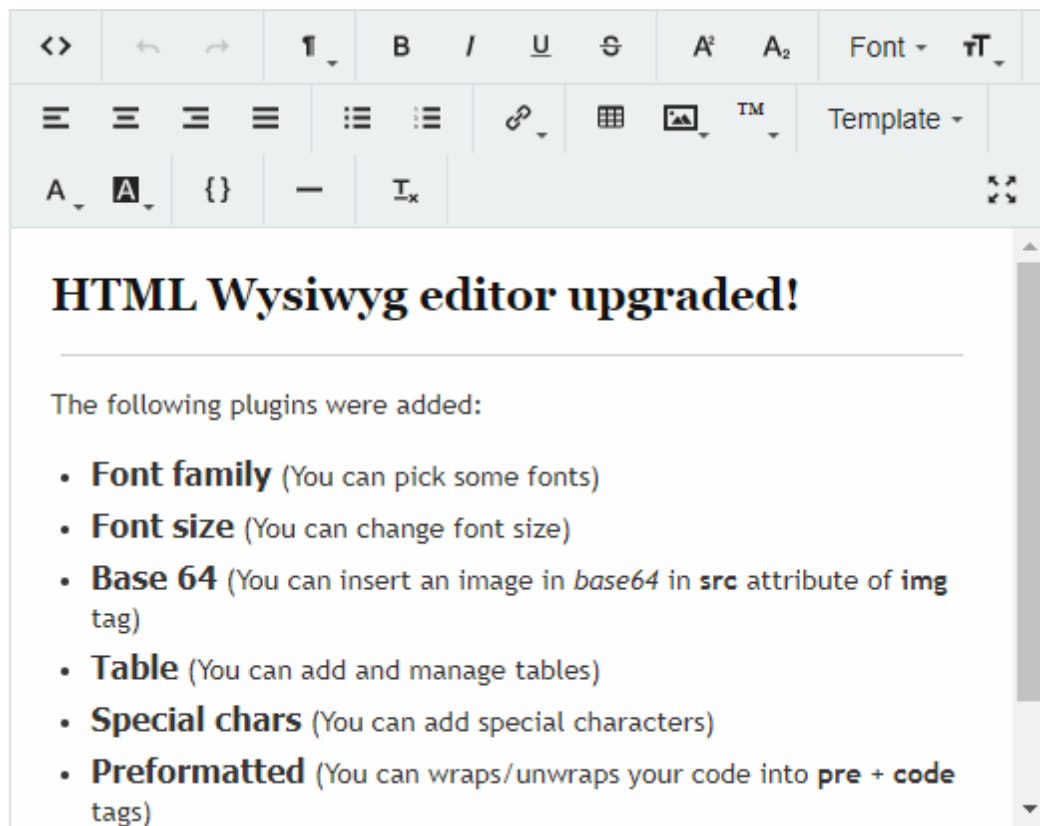
Use it to strip unwanted HTML tags and attributes from user input. By default all tags and attributes are stripped out, so you have to define allowed items explicitly.



Settings on the screenshot above are to allow the <a> tag (optionally with the href attribute as well as the and <h1> tags. All other tags and attributes will be removed from the user input.

5.1.3.19 Html Wysiwyg

Select WYSIWYG (What You See Is What You Get) editor to provide users with an ability to directly manipulate the layout of a HTML data stored in the column, without having to type or remember names of layout commands.

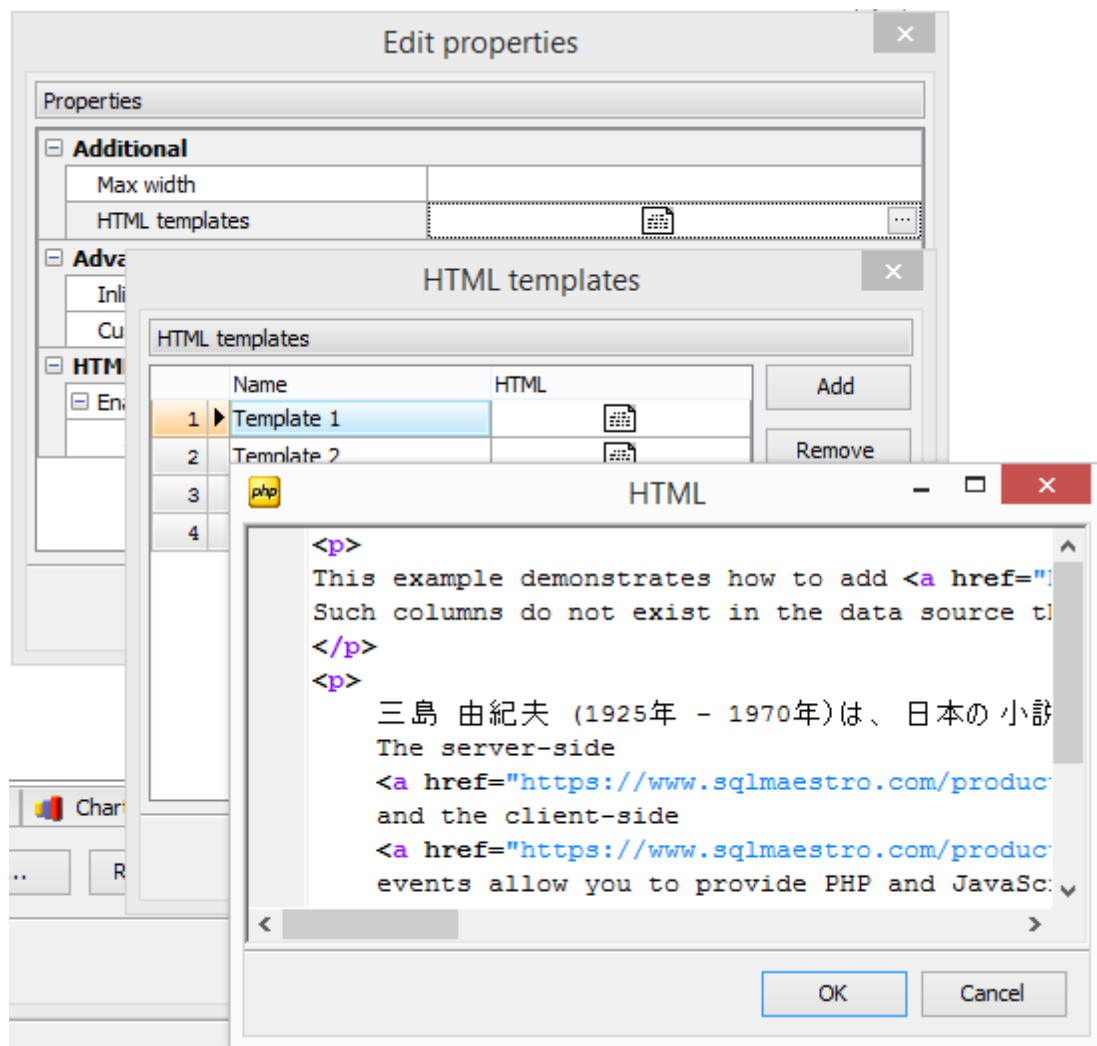


Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

HTML Templates

Use this dialog to set predefined HTML templates to be available in the editor. To add a template, open HTML templates dialog with the ellipsis button, set the template name and HTML code.



Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. For example, to set the font color and the background color for a control, place the following string in Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

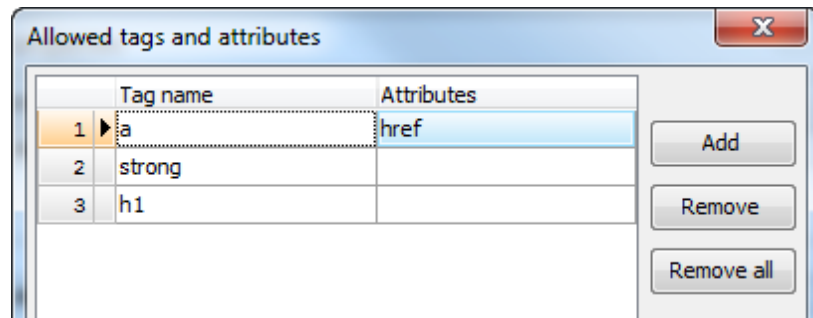
```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

HTML filter

Use it to strip unwanted HTML tags and attributes from user input. By default all tags

and attributes are stripped out, so you have to define allowed items explicitly.



Settings on the screenshot above are to allow the <a> tag (optionally with the href attribute as well as the and <h1> tags. All other tags and attributes will be removed from the user input.

5.1.3.20 Password

Select this control to add a masked textbox.

Password

Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

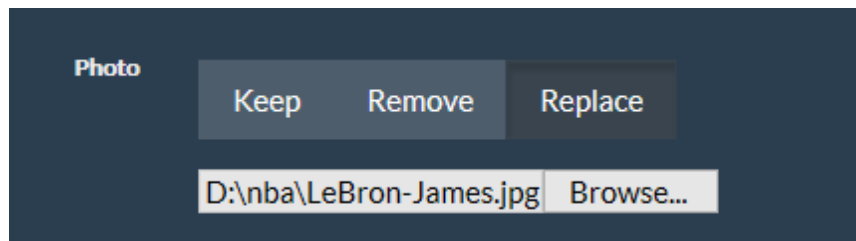
This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

5.1.3.21 File upload

Use this control to allow users to upload binary files to the database. To store files externally, use the [Upload File to Folder](#) ¹⁰³ control.



Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Acceptable file types

The value of this property defines which file types are displayed in the dialog invoked by pressing the Choose File button. The drop down list contains pre-defined values for some common cases.

Use file size limitation

To restrict the size of uploaded files, check this option and specify the maximum allowed file size in kilobytes.

You can save the **type**, **name** and **size** of uploaded files to table columns (optional).

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

See also: [Upload file to folder](#)^[103], [File download](#)^[102]

5.1.3.22 Image upload

Use this control to allow users to upload images to the database. To store images externally, use the [Upload Image to Folder](#)^[104] control.

Photo



Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Acceptable file types

The value of this property defines which file types are displayed in the dialog invoked by pressing the Choose File button. The drop down list contains pre-defined values for some common cases.

Use file size limitation

To restrict the size of uploaded files, check this option and specify the maximum allowed file size in kilobytes.

You can save the **type**, **name** and **size** of uploaded files to table columns (optional).

Use image size limitation

To restrict the size of uploaded images, check this option and specify the maximum allowed image width and height.

Uploaded images resizing

To upload resized images, check the [Resize image](#) option, select [Resize type \(Fit by width, Fit by height\)](#) and specify the height or width correspondingly.

Example

To allow uploading images which size is not greater than 200Kb, width is not greater than 600px and height is not greater than 400px and to save the original file names to the `file_name` column, specify the [Edit properties](#) as follows:

Style	
Custom attributes	
Additional	
<input type="checkbox"/> Use file size limitation	<input checked="" type="checkbox"/>
Max file size (Kb)	200
File type field	
File name field	file_name
File size field	
<input type="checkbox"/> Use image size limitation	<input checked="" type="checkbox"/>
Max width	600
Max height	400

Inline styles

Use this field to set formatting options to be used inside the `style` attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

See also: [Upload Image to Folder](#)^[104], [Image \(view column\)](#)^[53]

5.1.3.23 Upload file to folder

Use this control to allow users to upload external files. To files stored in the database, use the [File Upload](#)^[100] control.

Photo

Keep Remove Replace

D:\nba\LeBron-James.jpg Browse...

Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Acceptable file types

The value of this property defines which file types are displayed in the dialog invoked by pressing the Choose File button. The drop down list contains pre-defined values for some common cases.

Use file size limitation

To restrict the size of uploaded files, check this option and specify the maximum allowed file size in kilobytes.

Folder to upload

Use this field to specify the folder to be used to store the uploaded files.

Check the **Generate random file name** option to save uploaded files with random names or specify the **File name** [template](#)^[110] to be used for file name generation.

Store file name only

Defines whether the full file path (e.g. external_data/uploaded_files/filename.ext) or only file name (e.g. filename.ext) is stored in the database after the uploading.

Replace file if exists

This option allows you to set whether the uploaded file will be saved or ignored in case a file with the same name already exists in the folder to upload.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

See also: [File Upload](#)^[100], [File download](#)^[52]

5.1.3.24 Upload image to folder

This control allows users to upload [external images](#)^[55]. For images stored in the database, use the [Image Upload](#)^[101] control.

Photo



Max width

Use this property to restrict the maximum width of the editor (this means that in any screen resolution editor's width will be less or equal then the property value). Can be specified in any units supported by web browsers e.g. 300px, 25em, 50%, etc.

Acceptable file types

The value of this property defines which file types are displayed in the dialog invoked by pressing the Choose File button. The drop down list contains pre-defined values for some common cases.

Use file size limitation

To restrict the size of uploaded files, check this option and specify the maximum allowed file size in kilobytes.

Folder to upload

Use this field to specify the folder to be used to store the uploaded files.

Check the **Generate random file name** option to save uploaded files with random names or specify the **File name [template](#)** ^[110] to be used for file name generation.

Store file name only

Defines whether the full file path (e.g. external_data/uploaded_files/filename.ext) or only file name (e.g. filename.ext) is stored in the database after the uploading.

Replace file if exists

This option allows you to set whether the uploaded file will be saved or ignored in case a file with the same name already exists in the folder to upload.

Enable the **Generate thumbnail** option to save reduced-size versions of the uploaded files along with these files. By default thumbnails are saved in the same folder where full size images are saved. The names of thumbnails are 'field name'_'uploaded_file_name'. You can also specify another Folder to upload and File name template.

To set the size of thumbnails, select **Resize type** ([Fit by width](#), [Fit by height](#)) and specify the height or width correspondingly.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string into Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

It is recommended to prefix all custom attributes with data- to keep the result document compatible with the HTML5 requirements.

See also: [Image Upload](#)^{10f}, [Image \(view column\)](#)⁵³

5.1.3.25 Signature

Use this control to allow users to draw smooth signatures, save and store them as external files.



Format for saving (PNG, JPG)

Specify the file format the drawing will be saved.

Folder to save

Use this field to specify the folder to be used to store signatures.

Store file name only

Defines whether the full file path (e.g. signatures/John_Smith.jpg) or only file name (e.g.

John_Smith.jpg) is stored in the database after the saving.

Check the **Generate random file name** option to save signatures with random names or specify the **File name** [template](#)^[110] to be used for file name generation.

Replace file if exists

This option allows you to set whether the signature will be saved or ignored in case a file with the same name already exists in this folder.

Set the size of **Draw area (width and height)** in pixels.

Pen color

Define the color to be used to draw the lines. The color be any color format accepted by context (i.e. *transparent*, *black*, *red*, *#0fe128*). By default, it is black.

Background color

Specify here the color to be used to clear the background. As the pen color use any color format accepted by context (i.e. *transparent*, *black*, *red*, *#0fe128*). Use a non-transparent color e.g. "rgb(255,255,255)" (opaque white) if you'd like to save signatures as JPEG images.

Inline styles

Use this field to set formatting options to be used inside the [style](#) attribute of the element. For example, to set the font color and the background color for a control, place the following string to Inline styles:

```
color: red; background-color: yellow;
```

Custom attributes

This property allows you to add simple metadata to individual elements, largely for the purpose of providing information to make JavaScript functions easier. Such attributes can be later handled in client-side events. For example, to add several custom attributes to an editor, enter the following string into the Custom attributes edit box:

```
data-city="Boston" data-lang="js" data-food="Bacon"
```

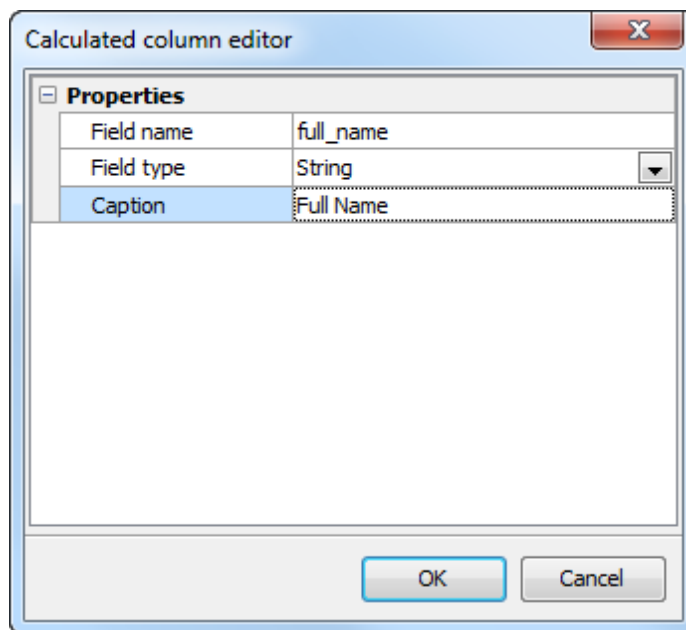
It is recommended to prefix all custom attributes with `data-` to keep the result document compatible with the HTML5 requirements.

5.1.4 Calculated Columns

Calculated (AKA computed, generated, or virtual) columns are columns that do not exist in the data source the page is based on and their values are automatically computed using other column values or another deterministic expression. For example, you might create a string column that displays concatenated values from other fields, create an integer column to calculate the person age based on his birthday, and so on.

Creating calculated columns

To create a new calculated column, open Page Editor, right-click the grid at the Columns tab and choose [Add calculated column...](#) from the popup menu. The following window will be opened:



Provide the name, the data type and the caption for the new column and press OK. A new calculated column will be added to the column list. To quickly distinguish between ordinal and calculated fields, the latter are marked with a green calculator icon.

	Field Name	List	View	Edit	Multi-edit	Insert	Quick filt	Filter buil	Print	Export	Compare
1	id	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	first_name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	last_name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4	birthday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5	full_name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6	age	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

It is possible to add multiple calculated columns for a single page.

Dropping calculated columns

To drop an existing calculated column, select the column in the column list, right-click and choose **Drop calculated column** from the popup menu.

Compute column values

To compute column values in read-only views (List, Print, Export, etc) provide the [OnCalculateFields](#)^[194] server-side event handler. For Edit and Insert forms provide the [OnCalculateControlValues](#)^[146] client-side event handler.

Use totals

Check this option to aggregate values this calculated column, specify the [Aggregate expression](#) (like *SUM(quantity*price)* where *quantity* and *price* are the table column names) and the [Aggregate caption](#). Total content of calculated columns can be customized via [OnCustomRenderTotals](#)^[158] event.

Restrictions

Calculated columns cannot be used with [Data_Filtering](#) and [Multi-Edit](#) tools, so the corresponding options in the column list are always disabled.

5.2 String templates

String templates are used by PostgreSQL PHP Generator on setting HREF templates, hints, etc. The strings may contain column names enclosed by %.

Examples:

1. Suppose a table *'team'* stores various info about NBA teams. The table has *'caption'* column with such data:

maverics
cavaliers

To create links to the team home pages:

<http://www.nba.com/maverics/>
<http://www.nba.com/cavaliers/>

set the HREF template

http://www.nba.com/%caption%/

2. Suppose there is a table *'employee'* storing employee info. The table has *'first_name'* and *'last_name'* columns with such data:

Forest Gump
Sara Connor

To add such hints to their photos on the generated webpage:

Forest Gump Photo
Sara Connor Photo

Specify the hint template:

%first_name% %last_name% Photo

5.3 Master-Detail Presentations

Master-detail presentations allow you to view and edit records from master and detail data sources (like country/city) on a single page. So, browsing the page representing product information, you can, for example, see all the orders related to a certain product, add new orders, edit/delete existing orders, and so on; working with list of continents - to inspect countries of the specific continent as on the picture below.

<input type="checkbox"/>		Actions	Id	Continent
<input type="checkbox"/>			1	Asia
<input type="checkbox"/>			2	Europe
<input type="checkbox"/>			3	North America

Country

Showing first 20 of 37 records (full view)

<input type="checkbox"/>		Actions	Code	Name	SurfaceArea	IndepYear	Population	LifeExpectancy
<input type="checkbox"/>			AIA	Anguilla	96.00	NULL	8,000	76.10
<input type="checkbox"/>			ATG	Antigua and Barbuda	442.00	1981	68,000	70.50
<input type="checkbox"/>			ABW	Aruba	193.00	NULL	103,000	78.40

You can also create nested master-detail presentations when you can see the "details of details". On the picture below the webpage with a list of continents is represented. For any continent all the corresponding countries may be browsed with a single click and at the same time it is possible to see cities of a certain country.

	+	Actions	Id	Continent
<input type="checkbox"/>	+		1	Asia
<input type="checkbox"/>	+		2	Europe
<input type="checkbox"/>	-		3	North America

Country ×

Showing first 20 of 37 records ([full view](#))

+

	+	Actions	Code	Name	SurfaceArea	IndepYear	Population	LifeExpectancy
<input type="checkbox"/>	+		AIA	Anguilla	96.00	NULL	8,000	76.10
<input type="checkbox"/>	-		ATG	Antigua and Barbuda	442.00	1981	68,000	70.50

City ×

Showing first 1 of 1 records ([full view](#))

+

Actions	ID	Name	District	Population
	63	Saint John's	St John	24,000

First 20 detail records can be previewed without opening a separate page. To see all the detail records, click the Full view link above the detail data grid. Alternatively you can access a detail page using the drop-down list on the right of the plus button.

<input type="checkbox"/>	+	Actions	Code	Name
<input type="checkbox"/>	- ▾		CHN	China
<div> <div>Language</div> <div>City </div> </div>				
Showing first 20 of 363 records (full view)				
<div> <div>+ Add new</div> <div> Export ▾</div> <div> Print ▾</div> </div>				
<input type="checkbox"/>		Actions	ID	Name
<input type="checkbox"/>			1890	Shanghai
<input type="checkbox"/>			1891	Peking

Database schema requirements

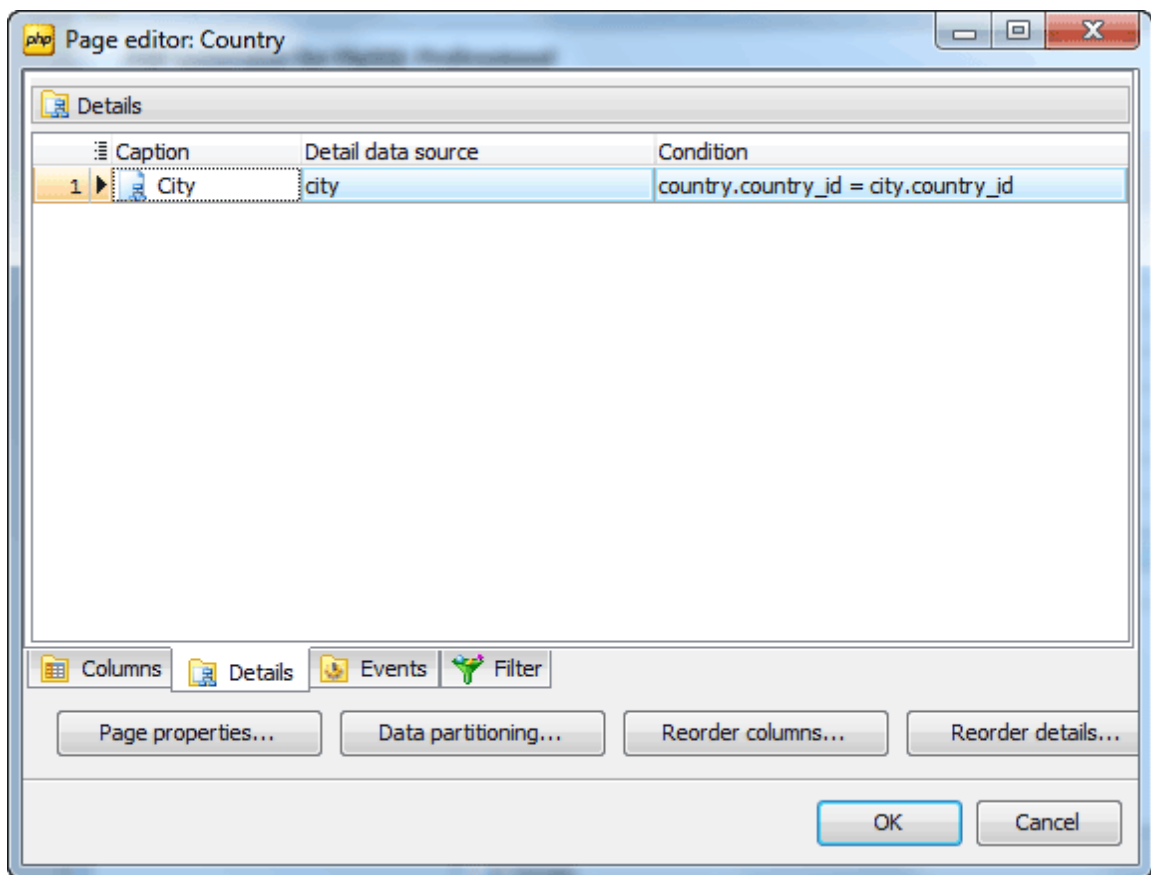
To create a two-level master-detail presentation, you need two tables with a parent-child relationship. The foreign key constraints are not required but highly recommended to enforce the referential integrity at the database level.

Creating master-detail views

There are two ways to provide the result webpage with a master-detail presentation:

Automatic creation

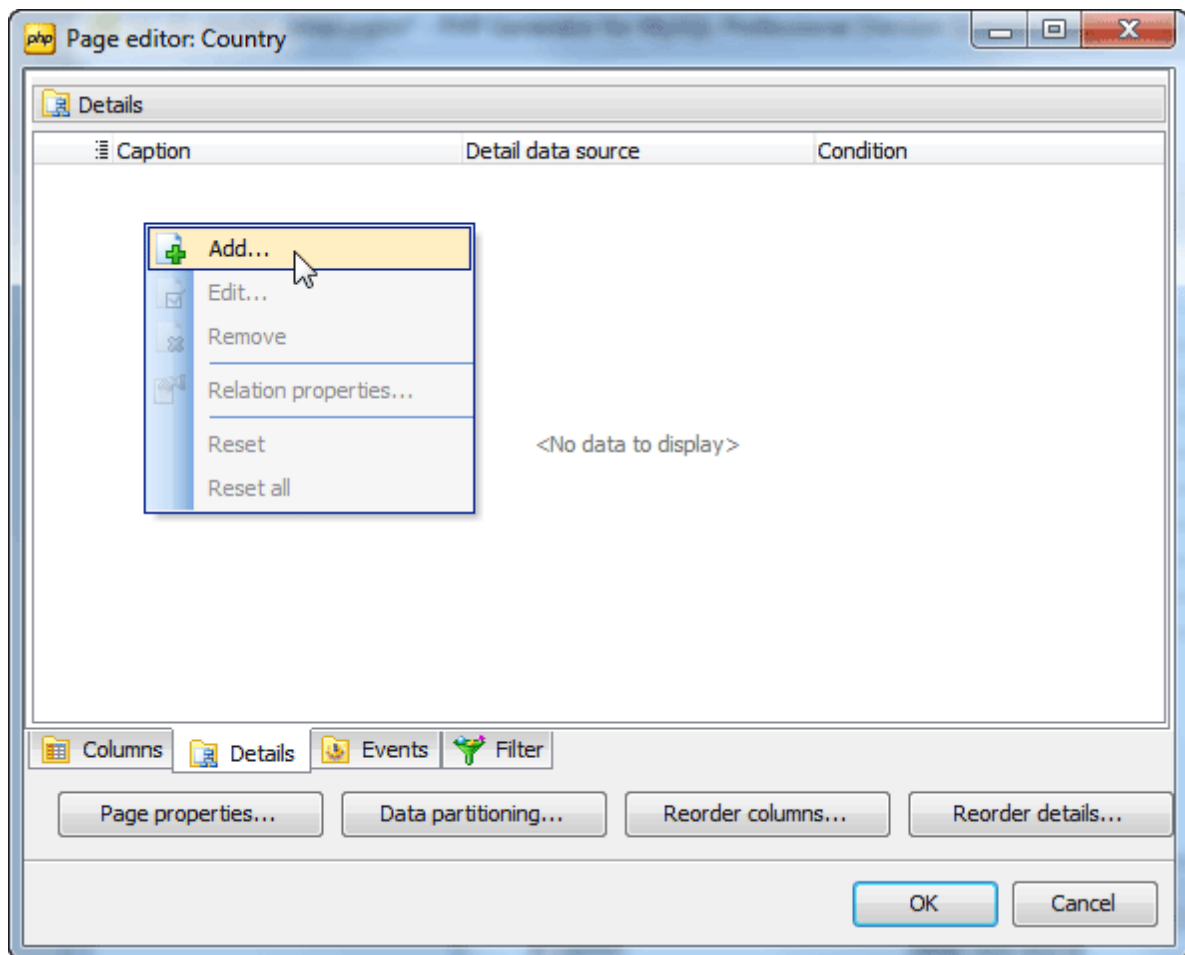
If tables are linked with a foreign key and the [Setup details by foreign key](#)^[335] option is enabled, the first-level detail pages are created automatically.



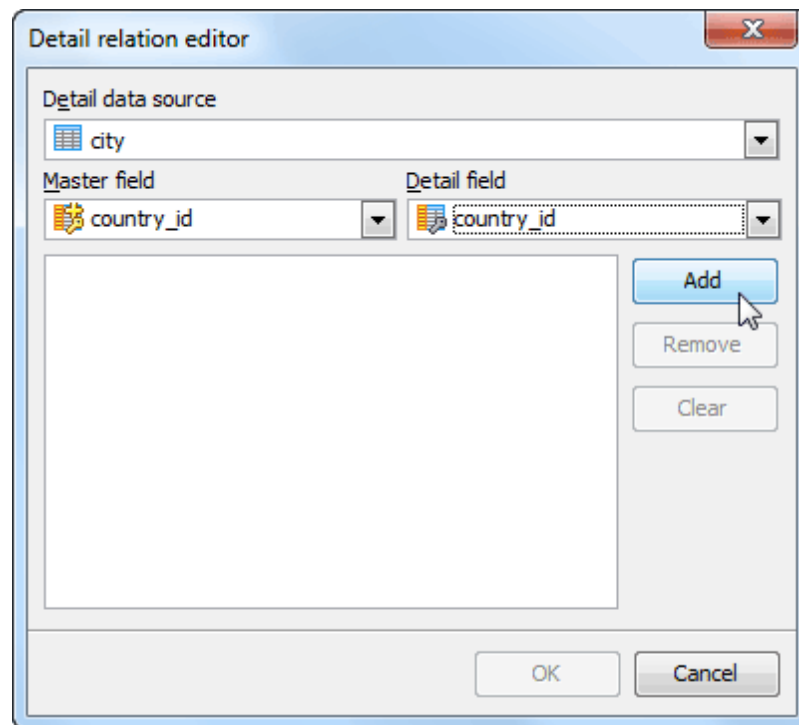
Manual creation

If tables are NOT linked with a foreign key or the [Setup details by foreign key](#)^[335] option is disabled, you can create detail pages manually as described below. The same technique is applied for creating nested detail pages.

1. At the Page Management step of the wizard select a [root-level page](#)^[28] you want to create a detail page for or (for creating a nested detail page) select a [detail page](#)^[31] in the corresponding list.
2. Click the [Add...](#) button on the right of the detail page list to open Link Editor.



3. Select a detail data source (table, view, or query) and columns from the [Master field](#) and [Detail field](#) lists. Click Add to add a link condition (you can specify multiple columns from each list for a single link). Click Remove to delete a field pair.



4. Click [OK](#) to save changes and close Link Editor.

5.4 Events

PostgreSQL PHP Generator allows you to supply the generated applications with an additional functionality with the help of event handlers. Event handlers are fragments of PHP/Javascript code executed at the appointed time. This feature provides you with wide opportunities: for example, you can equip webpages with tracking of user activity, completely redesign any of generated pages, supply the application with the power of third-part libraries, etc.

The most of events supported by PostgreSQL PHP Generator may be divided by the scope the events are fired on (possible options are all generated webpages (global events) or a certain webpage (page-specific events)); and by the side the event handlers are executed on: the webserver (written on **PHP**) or browsers (written on **Javascript**).

- [Server-side application-level \(global\) events](#)^[117]
- [Server-side page-level events](#)^[147]
- [Client-side page-level events](#)^[137]

To add or modify an event handler:

- open [Page Editor](#)^[33] of a webpage the event to be fired on (for page-level events) or the [Project Options](#)^[226] dialog (for global events);
- go to the [Events](#) tab;
- select an event from the list and double-click the appropriate line in the grid (it is also possible to use popup menu or **Ctrl+Enter**);
- enter/modify your PHP/Javascript code in the text area.

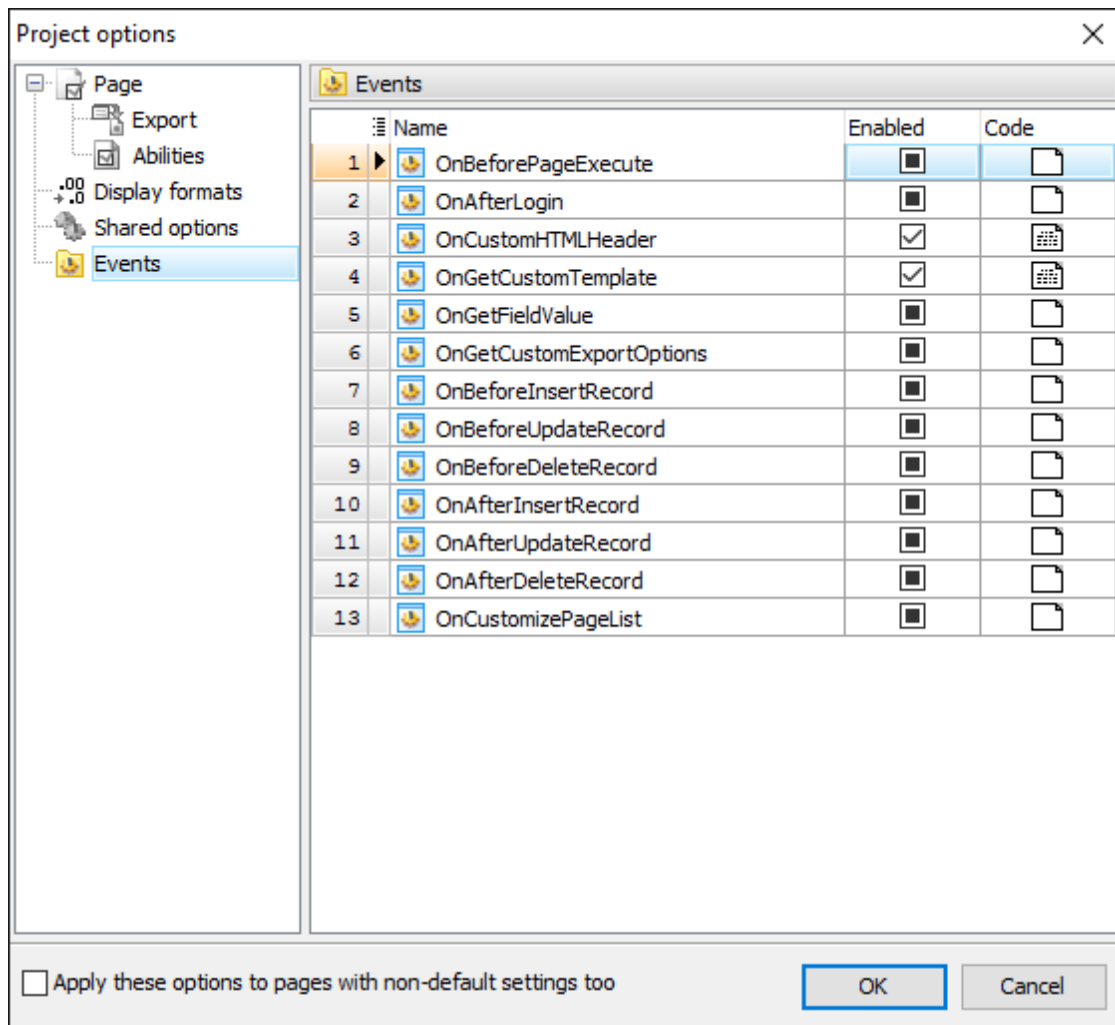
To temporarily disable an event, use the corresponding checkbox at the [Events](#) tab of the according [Page Editor](#)^[33] (for Page events) or the [Project Options](#)^[226] (for Global events).

You can also adjust the behaviour of your application on user login/logout with [Security Events](#)^[261]. These events are activated with the application [Security Options](#)^[248] are enabled.

5.4.1 Application-level (global) Events

Handlers of global events are pieces of **PHP** code that are not specific for any page i.e. these events are fired for all the generated webpages. It is possible to use [Server Side API](#)^[304] and [environment variables](#)^[196] within these event handlers.

To specify a handler for a global event, switch to the Events tab of the [Project options](#)^[226] dialog and double click the appropriate line in the grid.



The complete list of currently available application-level events is as follows.

- [OnBeforePageExecute](#) ¹¹⁹
- [OnPreparePage](#) ¹¹⁹
- [OnGetCustomTemplate](#) ¹²¹
- [OnCustomHTMLHeader](#) ¹²⁰
- [OnBeforeUpdateRecord](#) ¹²⁹
- [OnBeforeInsertRecord](#) ¹²⁸
- [OnBeforeDeleteRecord](#) ¹²⁹
- [OnAfterUpdateRecord](#) ¹³¹
- [OnAfterInsertRecord](#) ¹³⁰
- [OnAfterDeleteRecord](#) ¹³³
- [OnGetFieldValue](#) ¹³⁴
- [OnGetCustomExportOptions](#) ¹³⁴

- [OnCustomizePageList](#)^[136]
- [OnGetCustomPagePermissions](#)^[136]

See also: [Server side page-level events](#)^[147], [Client side page-level events](#)^[137],

5.4.1.1 OnBeforePageExecute

This event occurs before other events are declared and allow to create global objects, declare functions, and include third-party libraries. This helps you to define a snapshot of PHP code that will be included into all the pages.

Signature:

```
function OnBeforePageExecute()
```

Example:

In our [Schema Browser demo project](#) all the code implementing database filter, formatting and highlighting SQL code we entered in the *schema_browser_utils.php* file. The following code is used to include it into all application web pages.

```
include_once 'schema_browser_utils.php';
```

5.4.1.2 OnAddEnvironmentVariables

This event allows you to add environment variables for your application. You can access the values of these variables in any server-side event of your application via [GetEnvVar\(\)](#)^[317] method of the Application class, and also use them in lookup filter criteria, on data filtering, and as default values.

Signature:

```
function OnAddEnvironmentVariables(&$variables)
```

Parameters:

<code>\$variables</code>	An associative array of environment variables. Use it to add your own environment variables for your application.
--------------------------	---

Example 1:

The following code defines a new variable, which can be used in lookup filter criteria, on data filtering, as a default value.

```
$variables['DATE_TWO_MONTHS_AFTER_NOW'] = date('Y-m-d', strtotime('+2 months'));
```

5.4.1.3 OnPreparePage

This piece of code is a method of the Page class that is called at the end of the constructor. It allows you to customize all members of the class.

Signature:

```
function OnPreparePage()
```

Parameters:

This method has no parameters.

Example

The following code hides the *update_date_time* column from the data grid and Single Record View form as well as the *status* column from Edit and Insert forms from all users except ones who have the Admin privilege for the current page.

```
if (!GetApplication()->IsLoggedInAsAdmin()) {
    $updateDateTimeColumnName = 'update_date_time';
    $column = $this->GetGrid()->getSingleRecordViewColumn($updateDateTimeColumnName);
    if ($column) {
        $column->setVisible(false);
    }
    $column = $this->GetGrid()->getViewColumn($updateDateTimeColumnName);
    if ($column) {
        $column->setVisible(false);
    }

    $statusColumnName = 'status';
    $column = $this->GetGrid()->getEditColumn($statusColumnName);
    if ($column) {
        $column->setVisible(false);
    }
    $column = $this->GetGrid()->getInsertColumn($statusColumnName);
    if ($column) {
        $column->setVisible(false);
    }
}
```

See also: [OnPageLoaded](#)^[150].

5.4.1.4 OnCustomHTMLHeader

This event occurs when generating the HEAD section of the page. It allows you to define the contents of the HEAD section (like meta tags or favicon) for all pages of the generated website.

Signature:

```
function OnCustomHTMLHeader ($page, &$customHTMLHeaderText)
```

Parameters:

\$page	An instance of the Page class ^[304] declared in <i>components/page.php</i> .
\$customHtmlHeaderText	The text block to be added to the head section of all created pages.

Example 1:

To add a new title and a favicon to all webpages of the generated application, place the following code to the event body.

```
$customHtmlHeaderText = '<meta name="author" content="SQL Maestro Group">';
$customHtmlHeaderText .= "\n";
$customHtmlHeaderText = '<meta name="copyright" content="SQL Maestro Group" />';
$customHtmlHeaderText .= "\n";
$customHtmlHeaderText = '<meta name="keywords" content="nba,basketball,game">';
$customHtmlHeaderText .= "\n";
$customHtmlHeaderText = '<link rel="icon" href="favicon.ico" type="image/x-icon" />';
```

Example 2:

The following code is used to include external style file and javascript library to the generated application.

```
$customHtmlHeaderText =  
'<link rel="stylesheet" href="external_data/pg_styles.css">' .  
'<script src="external_data/pg_utils.js"></script>';
```

5.4.1.5 OnGetCustomTemplate

This event allows you to customize almost any part of the generated application using an own template instead of the default one.

Signature:

```
function OnGetCustomTemplate ($part, $mode, &$amp;result, &$amp;params)
```

Parameters:

\$part	The part of the page to be affected by the template.
\$mode	The current state of the webpage the template to be used.
\$result	The path to the file to be used as template according to the <i>components/templates/custom_templates</i> folder.
\$params	An associative array of additional parameters you want to assign to the template.

PHP Generator uses [Smarty 2.0](#) as template language. If you are not familiar with web template engines, you might want to [start here](#).

Our Feature Demo provides [a number of live examples](#) of customized templates. We would recommend you to study them carefully before trying to customize your application.

To customize a webpage, you need to:

- create a new template to be used for this webpage;
- instruct PHP Generator to use this template for the selected webpage.

The simplest way to create a new template is to modify an existing one (it is much easier than create a new template "from scratch"). Use the table below to select the template corresponding to the necessary state of page and page part. Custom template files must be uploaded to the **'components/templates/custom_templates'** directory. This folder should be created by the user of PHP Generator and its content is not changing during the PHP Generator sessions.

To instruct PHP Generator to use a customized template file for a certain webpage, specify the following code in the page's OnGetCustomTemplate event handler (of course you should replace YourPart and YourMode to the appropriate values from the table below).

```
if ($part == YourPart && $mode == YourMode) {  
    $result = 'your_template_file.tpl';  
}
```


The following table shows how to customize various pages used in the generated applications.

State of the webpage	Page Part	Default template	Parameters
All Pages ¹⁶⁷	webpage layout	common/layout.tpl	PagePart::Layout Any PageMode
	page list	page_list_menu.tpl (for top-side menu) page_list_sidebar.tpl (for sidebar menu)	PagePart::PageList Any PageMode
List page ¹⁶⁸	table grid	list/grid_table.tpl (for grid view mode) list/grid_card.tpl (for card view mode)	PagePart::Grid PageMode::ViewAll
	grid toolbar	list/grid_toolbar.tpl	PagePart::GridToolbar
	single row	list/single_row.tpl (for grid view mode) list/sinlge_row_card.tpl (for card view mode)	PagePart::GridRow PageMode::ViewAll
Single Record View ¹⁶⁹	separate page	view/grid.tpl	PagePart::RecordCard PageMode::View
	modal dialog	view/record_card_view.tpl	PagePart::VerticalGrid PageMode::ModalView
	inline form	view/ record_card_inline_view.tpl	PagePart::VerticalGrid PageMode::InlineView
Edit form ¹⁷⁰	editors area	forms/form.tpl	PagePart::VerticalGrid PageMode::FormEdit
	separate page form	forms/page_form.tpl	PagePart::VerticalGrid PageMode::Edit
	modal dialog	forms/modal_form.tpl	PagePart::VerticalGrid PageMode::ModalEdit
	inline form	forms/inline_form.tpl	PagePart::VerticalGrid PageMode::InlineEdit

Insert form <small>170</small>	editors area	forms/form.tpl	PagePart::VerticalGrid PageMode::FormInsert
	separate page form	forms/page_form.tpl	PagePart::VerticalGrid PageMode::Insert
	modal dialog	forms/modal_form.tpl	PagePart::VerticalGrid PageMode::ModalInsert
	inline form	forms/inline_form.tpl	PagePart::VerticalGrid PageMode::InlineInsert
Print <small>172</small>	webpage layout (list page)	print/page.tpl	PagePart::PrintLayout PageMode::PrintAll
	data grid (list page)	print/grid.tpl	PagePart::Grid PageMode::PrintAll
	detail page	print/detail_page.tpl	PagePart::PrintLayout PageMode::PrintDetailPage
	webpage layout (single record)	print/page.tpl	PagePart::PrintLayout PageMode::PrintOneRecord
	data grid (single record)	view/print_grid.tpl	PagePart::Grid PageMode::PrintOneRecord
Export	data grid	export/pdf_grid.tpl, export/excel_grid.tpl, export/csv_grid.tpl, etc	PagePart::Grid PageMode::ExportPdf, PageMode::ExportExcel, PageMode::ExportCsv, etc
	single record (PDF only)	export/pdf_record.tpl	PagePart::RecordCard PageMode::ExportPdf
Record comparison	data grid	compare/grid.tpl	PagePart::Grid PageMode::Compare
Login page		login_page.tpl	PagePart::LoginPage
Login control		login_control.tpl	PagePart::LoginControl

Home page	home_page.tpl	PagePart::HomePage
Navigation	navigation.tpl	PagePart::Navigation

The following table shows how to customize user registration and password recovering pages. These parameters can be used only for the project-level version of this event.

Registration Page	registration_page.tpl	PagePart::RegistrationPage
Registration Form	registration_form.tpl	PagePart::RegistrationForm
Password Recovery Page	recovering_password_page.tpl	PagePart::PasswordRecovery
Reset Password Page	reset_password_page.tpl	PagePart::ResetPassword
Resend Verification Email Page	resend_verification_page.tpl	PagePart::ResendVerification

The following table shows how to customize [emails sent to users](#)^[256] on registration and password recovering. These parameters can be used only for the project-level version of this event.

Verification email subject	user_verification_subject.tpl	PagePart::Mail PageMode::MailVerificationSubject
Verification email body	user_verification_body.tpl	PagePart::Mail PageMode::MailVerificationBody
Password reset email subject	recovering_password_subject.tpl	PagePart::Mail PageMode::MailRecoveringPasswordSubject
Password reset email body	recovering_password_body.tpl	PagePart::Mail PageMode::MailRecoveringPasswordBody

Example 1:

Suppose we need to use *components/templates/custom_templates/staff_edit.tpl* file as a template for the edit form of a webpage.

```
if ($part == PagePart::VerticalGrid && $mode == PageMode::Edit) {
    $result = 'staff_edit.tpl';
}
```

Now we can compare the result webpages without using this event and with the enabled one:

Standard edit form:

Custom edit form:

Example 2:

A single event handler can be used to customize multiple templates:

```
if ($part == PagePart::Grid && $mode == PageMode::ViewAll)
    $result = 'games_grid.tpl';
if ($part == PagePart::VerticalGrid && $mode == PageMode::Edit)
    $result = 'games_edit.tpl';
if ($part == PagePart::RecordCard && $mode == PageMode::View)
    $result = 'games_view.tpl';
```

Example 3

This example shows how to fill a SELECT input with names of all MySQL databases available at the server.

1. Specify the OnGetCustomTemplate event handler as follows:

```
// set the template file
if ($part == PagePart::Grid && $mode == PageMode::ViewAll) {
    $result = 'custom_grid.tpl';

    // get the list of available databases
    $queryResult = array();
    $this->GetConnection()->ExecQueryToArray('SHOW DATABASES', $queryResult);

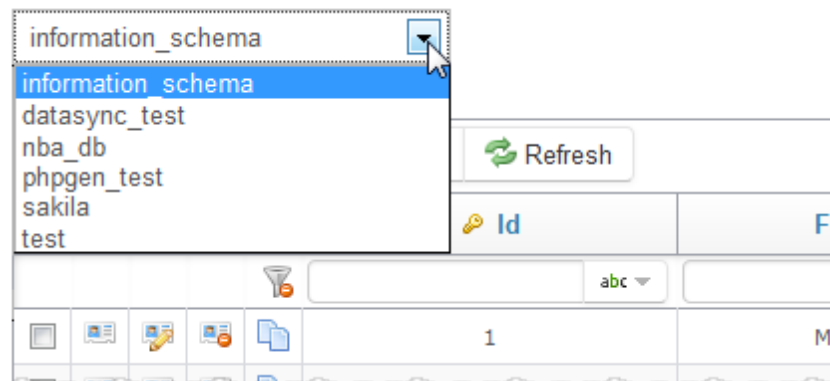
    // fill a one-dimensional array with database names
    $databaseNames = array();
    foreach($queryResult as $row)
        $databaseNames[] = $row[0];

    // assign the parameter in the template
    $params['databaseNames'] = $databaseNames;
}
```

2. Add the following code to the template file:

```
<select name="database">
    {foreach from=$databaseNames item=database}
        <option value="{ $database }">{ $database }</option>
    {/foreach}
</select>
```

The screen below demonstrates the result output.



Example 4

This example shows the use of the superglobal \$_GET array values in a custom template.

1. Specify the OnGetCustomTemplate event handler as follows:

```
if ($part == PagePart::Grid && $mode == PageMode::ViewAll) {
    $result = 'custom_grid.tpl';

    // check if the parameter custom_var available
    if (GetApplication()->IsGETValueSet('custom_var'))
        // found
```



```

        $params['custom_var'] = GetApplication()->GetGETValue('custom_var');
    else
        // not found. Assign a default value.
        $params['custom_var'] = '5 (default)';
}

```

2. Add the following code to the template file:

```

<div>
<h4>I am a custom template with a custom variable $custom_var. Its value
    {if !isset($custom_var)}
        still undefined
    {else}
        = {$custom_var}
    {/if}.
</h4>
</div>

```

Example 4

Let's see the login form customization used in [our NBA online demo](#).

1. Specify the OnGetCustomTemplate event handler as follows:

```

if ($part == PagePart::LoginPage) {
    $result = 'login_page.tpl';
}

```

2. Add the following code to the template file:

```

<h2>Logins</h2>
<table class="table table-bordered">
    <thead>
        <tr>
            <th>Username</th>
            <th>Password</th>
            <th>Description</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>admin</td>
            <td>admin</td>
            <td>Can modify any record at any page and manage other users.</td>
        </tr>
        <tr>
            <td>lakers</td>
            <td>lakers</td>
            <td>Can modify team info, home games and players for LA Lakers.</td>
        </tr>
        <tr>
            <td>boston</td>
            <td>boston</td>
            <td>The same for the Boston Celtics team.</td>
        </tr>
        <tr>
            <td>game_manager</td>
            <td>game</td>
            <td>Can access only game list. Can modify any game.</td>
        </tr>
    </tbody>
</table>

```



```

    </tbody>
</table>

```

Example 5

The example below is used in our NBA demo application to customize the template used to export a single record to PDF.

```

if ($part == PagePart::RecordCard && $mode == PageMode::ExportPdf) {
    $result = 'game_pdf.tpl';
}

```

Live example can be found [here](#).

5.4.1.6 OnBeforeInsertRecord

This event occurs when the Insert command is executed, and before the actual insertion.

Signature:

```

function OnBeforeInsertRecord ($page, &$rowData, &$cancel, &$message,
    &$messageDisplayTime, $tableName)

```

Parameters:

\$page	An instance of the Page ^[304] class declared in <i>components/page.php</i> .
\$rowData	The associative array of values that corresponds to the currently processed row.
\$cancel	The value indicating whether the operation should be canceled.
\$message	The message string that is displayed after the operation is completed (or canceled).
\$messageDisplayTime	A time interval (in seconds) after which the message will disappear automatically. Default value is 0 (the message will not disappear).
\$tableName	The name of processed table.

Example 1:

```

if (!(allowDataEditing())) {
    $cancel = true;
    $message = 'The application is running in read-only mode.';
}

```

Example 2:

The following code allows to assign values to some fields (usually these fields are not included in an insert form) before inserting of a record:

```

$rowData['line_total'] = $rowData['quantity'] * $rowData['unit_price'];
$rowData['create_datetime'] = SMDatetime::Now();
$rowData['insert_by'] = $page->GetCurrentUserId();

```

See also: [OnBeforeUpdateRecord](#) ^[179], [OnBeforeDeleteRecord](#) ^[180], [OnAfterInsertRecord](#) ^[178].

5.4.1.7 OnBeforeUpdateRecord

This event occurs when the Update command is executed, and before the actual update.

Signature:

```
function OnBeforeUpdateRecord ($page, $oldRowData, &$rowData, &$cancel,    &$message,
                              &$messageDisplayTime,    $tableName)
```

Parameters:

\$page	An instance of the Page ^[304] class declared in <i>components/page.php</i> .
\$oldRowData	The associative array of old (previous) values of the currently processed row.
\$rowData	The associative array of values that corresponds to the currently processed row.
\$cancel	The value indicating whether the operation should be canceled.
\$message	The message string that is displayed after the operation is completed (or canceled).
\$messageDisplayTime	A time interval (in seconds) after which the message will disappear automatically. Default value is 0 (the message will not disappear).
\$tableName	The name of processed table.

Example 1:

```
if (!(allowDataEditing())) {
    $cancel = true;
    $message = 'The application is running in read-only mode.';
}
```

Example 2:

The following code allows to assign values to some fields (usually these fields are not included in an edit form) before updating of a record:

```
$rowData['line_total'] = $rowData['quantity'] * $rowData['unit_price'];
$rowData['update_datetime'] = SMDatetime::Now();
$rowData['last_updated_by'] = $page->GetCurrentUserId();
```

See also: [OnBeforeDeleteRecord](#)^[180], [OnBeforeInsertRecord](#)^[178], [OnAfterUpdateRecord](#)^[176].

5.4.1.8 OnBeforeDeleteRecord

This event occurs when the Delete command is executed, and before the actual deletion.

Signature:

```
function OnBeforeDeleteRecord ($page, &$rowData, &$cancel,    &$message,
                              &$messageDisplayTime,    $tableName)
```

Parameters:

\$page	An instance of the Page ^[304] class declared in <i>components/page.php</i> .
\$rowData	The associative array of values that corresponds to the currently processed row.
\$cancel	The value indicating whether the operation should be canceled.
\$message	The message string that is displayed after the operation is completed (or canceled).
\$messageDisplayTime	A time interval (in seconds) after which the message will disappear automatically. Default value is 0 (the message will not disappear).
\$tableName	The name of processed table.

Example 1:

```
if (!(allowDataEditing())) {
    $cancel = true;
    $message = 'The application is running in read-only mode.';
}
```

See also: [OnBeforeUpdateRecord](#)^[179], [OnBeforeInsertRecord](#)^[178], [OnAfterDeleteRecord](#)^[177].

5.4.1.9 OnAfterInsertRecord

This event occurs when the Insert command is executed, and after the actual insertion.

Signature:

```
function OnAfterInsertRecord ($page, $rowData, $tableName,
    &$success, &$message, &$messageDisplayTime)
```

Parameters:

\$page	An instance of the Page ^[304] class.
\$rowData	The associative array of values that corresponds to the currently processed row.
\$tableName	The name of processed table.
\$success	Indicates whether the last data manipulation statement was successful or not.
\$message	A message to be displayed to a user. If the statement completed successfully, the value of this parameter is equal to empty string. If the statement failed, the value of this parameter contains the error message that came from the database server.
\$messageDisplayTime	A time interval (in seconds) after which the message will disappear automatically. Default value is 0 (the message will not disappear).

Success messages are displayed in green while error messages are displayed in red.

Example 1:

The following code shows the message about a success of the operation. The message

will be hidden in 5 seconds:

```
if ($success) {
    $message = 'Record processed successfully.';
}
else {
    $message = '<p>Something wrong happened. ' .
        '<a class="alert-link" href="mailto:admin@example.com">' .
        'Contact developers</a> for more info.</p>';
}
$messageDisplayTime = 5;
```

Example 2:

The following code logs information about added records into a separate table:

```
if ($success) {
    $userId = $page->GetCurrentUserId();
    $currentDateTime = SMDatetime::Now();
    $sql =
        "INSERT INTO activity_log (table_name, action, user_id, log_time) " .
        "VALUES ('$tableName', 'INSERT', $userId, '$currentDateTime');";
    $page->GetConnection()->ExecSQL(sprintf($sql, $userId, $action, $currentDateTime));
}
```

Example 3:

The following code is used in our [Feature Demo](#) to save genres of a newly added movie to a separate table:

```
if ($success && GetApplication()->IsPOSTValueSet('genres')) {
    $genres = GetApplication()->GetPOSTValue('genres');
    $lastInsertId = $page->GetConnection()->GetLastInsertId();
    foreach ($genres as $genre) {
        $sql = sprintf('INSERT INTO movie_genres VALUES(%d, %d);', $lastInsertId, $genre);
        $page->GetConnection()->ExecSQL($sql);
    }
}
```

See also: [OnAfterUpdateRecord](#)^[176], [OnAfterDeleteRecord](#)^[177], [OnBeforeInsertRecord](#)^[178].

5.4.1.10 OnAfterUpdateRecord

This event occurs when the Update command is executed, and after the actual update.

Signature:

```
function OnAfterUpdateRecord ($page, $oldRowData, $rowData, $tableName,
    &$success, &$message, &$messageDisplayTime)
```

Parameters:

\$page	An instance of the Page ^[304] class.
\$oldRowData	The associative array of old (previous) values of the currently processed row.
\$rowData	The associative array of values that corresponds to the currently processed row.
\$tableName	The name of processed table.
\$success	Indicates whether the last data manipulation statement was

	successful or not.
<code>\$message</code>	A message to be displayed to a user. If the statement completed successfully, the value of this parameter is equal to empty string. If the statement failed, the value of this parameter contains the error message that came from the database server.
<code>\$messageDisplayTime</code>	A time interval (in seconds) after which the message will disappear automatically. Default value is 0 (the message will not disappear).

Success messages are displayed in green while error messages are displayed in red.

Example 1:

```
if ($success) {
    $message = 'Record processed successfully.';
}
else {
    $message = '<p>Something wrong happened. ' .
        '<a class="alert-link" href="mailto:admin@example.com">' .
        'Contact developers</a> for more info.</p>';
}
```

Example 2:

The following code logs information about modified records into a separate table:

```
if ($success) {
    // Check if record data was modified
    $dataModified =
        $oldRowData['first_name'] != $rowData['first_name'] ||
        $oldRowData['last_name'] != $rowData['last_name'] ||
        $oldRowData['email'] != $rowData['email'];

    if ($dataModified) {
        $userId = $page->GetCurrentUserId();
        $currentDateTime = SMDatetime::Now();
        $sql =
            "INSERT INTO activity_log (table_name, action, user_id, log_time) " .
            "VALUES ('$tableName', 'UPDATE', $userId, '$currentDateTime');";
        $page->GetConnection()->ExecSQL($sql);
    }
}
```

Example 3:

The following code is used in our [Feature Demo](#) to modify genres of the updated movie to a separate table:

```
if ($success && GetApplication()->IsPOSTValueSet('genres')) {
    $sql = sprintf('DELETE FROM movie_genres WHERE movie_id = %d;', $rowData['id']);
    $page->GetConnection()->ExecSQL($sql);
    $genres = GetApplication()->GetPOSTValue('genres');
    foreach ($genres as $genre) {
        $sql = sprintf('INSERT INTO movie_genres VALUES(%d, %d);', $rowData['id'], $genre);
        $page->GetConnection()->ExecSQL($sql);
    }
}
```


See also: [OnAfterDeleteRecord](#)^[177], [OnAfterInsertRecord](#)^[175], [OnBeforeUpdateRecord](#)^[179].

5.4.1.11 OnAfterDeleteRecord

This event occurs when the Delete command is executed, and after the actual deletion.

Signature:

```
function OnAfterDeleteRecord ($page, $rowData, $tableName,
    &$success, &$message, &$messageDisplayTime)
```

Parameters:

\$page	An instance of the Page ^[304] class.
\$rowData	The associative array of values that corresponds to the currently processed row.
\$tableName	The name of processed table.
\$success	Indicates whether the last data manipulation statement was successful or not.
\$message	A message to be displayed to a user. If the statement completed successfully, the value of this parameter is equal to empty string. If the statement failed, the value of this parameter contains the error message that came from the database server.
\$messageDisplayTime	A time interval (in seconds) after which the message will disappear automatically. Default value is 0 (the message will not disappear).

Success messages are displayed in green while error messages are displayed in red.

Example 1:

The following code shows the message about a success of the operation. The message will be hidden in 5 seconds:

```
if ($success) {
    $message = 'Record processed successfully.';
}
else {
    $message = '<p>Something wrong happened. ' .
        '<a class="alert-link" href="mailto:admin@example.com">' .
        'Contact developers</a> for more info.</p>';
}
$messageDisplayTime = 5;
```

Example 2:

The following code logs information about deleted records in a separate table:

```
if ($success) {
    $userId = $page->GetCurrentUserId();
    $currentDateTime = SMDatetime::Now();
    $sql =
        "INSERT INTO activity_log (table_name, action, user_id, log_time) " .
        "VALUES ('$tableName', 'DELETE', $userId, '$currentDateTime');";
    $page->GetConnection()->ExecSQL(sprintf($sql, $userId, $action, $currentDateTime));
}
```


See also: [OnAfterUpdateRecord](#)^[176], [OnAfterInsertRecord](#)^[175], [OnBeforeDeleteRecord](#)^[180].

5.4.1.12 OnGetFieldValue

This event occurs on displaying a value of a field. It allows you to replace the actual value of a field to your own.

Signature:

```
function OnGetFieldValue ($fieldName, &$value, $tableName)
```

Parameters:

\$fieldName	The name of currently processed field.
\$value	The actual value of the field. Can be changed to another value.
\$tableName	The name of currently processed table.

Example

This example shows how this event can be used for implementing a very simple transparent encryption based on the [str_rot13](#) PHP function.

```
if ($fieldName == 'private_data') {
    $value = str_rot13($value);
}
```

Since the call of this function for an encrypted expression returns the original string, we can define [OnBeforeInsertRecord](#)^[178] and [OnBeforeUpdateRecord](#)^[179] event handlers as follows:

```
$rowData['private_data'] = str_rot13($rowData['private_data']);
```

This is just an example. We would recommend that you use more cryptographically strong algorithms to encrypt your important data.

5.4.1.13 OnGetCustomExportOptions

This event fires on exporting data from a database to a file. It allows you to customize some advanced export settings.

Signature:

```
function OnGetCustomExportOptions ($Page page, $exportType, $rowData, &$options)
```

Parameters:

\$page	An instance of the Page ^[304] class.
\$exportType	The type of the result file. Possible values are "csv", "xls", "doc", "pdf", "xml".
\$rowData	An associative array of values that corresponds to the currently processed row (for exporting a single row). NULL if a record list is exported.
\$options	An associative array of options passed to the export engine (see below).

The list of supported options is as follows:

All export types

filename	The name of the output file.
----------	------------------------------

Export to PDF

orientation	Page orientation. Possible values are "P" (portrait orientation) or "L" (landscape orientation). Default value is "P".
size	Size of the page. Possible values are A0 - A10, B0 - B10, C0 - C10, 4A0, 2A0, RA0 - RA4, SRA0 - SRA4, Letter, Legal, Executive, Folio, Demy, Royal, A (type A paperback 111x178mm), B (type B paperback 128x198mm). Default value is A4.
margin-left	Distance in mm from the left side of page to the left side of text.
margin-right	Distance in mm from the right side of page to the right side of text.
margin-top	Distance in mm from top of page to start of text (ignoring any headers).
margin-bottom	Distance in mm from bottom of page to bottom of text (ignoring any footers).
margin-header	Distance in mm from top of page to start of header.
margin-footer	Distance in mm from bottom of page to bottom of footer.
header	<p>In any page header, '{PAGENO}' or '{DATE j-m-Y}' can be used - which will be replaced by the page number or current date. j-m-Y can be replaced by any of the valid formats used in the php date() function.</p> <p>The page number is the calculated number, taking regard of any resetting of numbering during the document.</p>
html-header	Standard HTML code to define the header of pdf file. It can only be defined outside HTML block tags (except <body>).
footer	<p>In any page footer, '{PAGENO}' or '{DATE j-m-Y}' can be used - which will be replaced by the page number or current date. j-m-Y can be replaced by any of the valid formats used in the php date() function.</p> <p>The page number is the calculated number, taking regard of any resetting of numbering during the document.</p>
html-footer	Standard HTML code to specify the footer of pdf file. It can only be defined outside HTML block tags (except <body>).

Export to Excel

file-format	The format of the output file. Possible values are "xlsx" and "xls". Default value is "xlsx".
engine	The engine to be used on data export. Possible values are "template" (template-based export) and "phpexcel" (to create native .xls files). Default value is "phpexcel". If you customize templates for export to Excel then you should use the "template"

	engine.
--	---------

Example

This example shows how to customize the result file name according to row values. If a record list is exported, the file name is set to list.pdf(list.xml, list.xls, etc).

```
$options['filename'] = ($rowData ? $rowData['id'] : 'list') . '.' . $exportType;
```

Example

This example shows how to customize header, footer and margins of pdf file.

```
if ($exportType == 'pdf') {

    $options['header'] = '{DATE Y-m-d}';

    $options['footer'] = '{PAGENO}';

    $options['margin-top'] = 12;

}
```

5.4.1.14 OnCustomizePageList

This event allows you to add new groups and items to the application menu (both top-side and sidebar).

Signature:

```
function OnCustomizePageList($page, $pageList)
```

Parameters:

\$page	An instance of the CommonPage class that corresponds currently processed page.
\$pageList	An instance of the PageList class.

Example

This example shows how to add a new group with a couple of links.

```
$pageList->addGroup('External links');
$pageList->addPage(new PageLink('Home Site', 'http://www.mysite.com',
    'Vist my site', false, false, 'External links'));
$pageList->addPage(new PageLink('Get Support', 'http://www.mysite.com/support/',
    'Get support for this application', false, false, 'External links'));
```

5.4.1.15 OnGetCustomPagePermissions

This event allows you to customize page-level permissions.

Signature:

```
function OnGetCustomPagePermissions($pageName, $userId, $userName, $connection, &$permissions)
```

Parameters:

\$pageName	The name of the page
------------	----------------------

\$userId	The id of the current user
\$userName	The name of the current user
\$connection	An instance of EngConnection ^[318] class
\$permissions	Permissions to be applied to the page. An instance of the PermissionSet ^[320] class.

Example

This example shows how to implement a simple role-based permission model. Our goal is to ensure that only site admins and members of the Sales role can add, remove and edit records displayed on this page.

```
// do not apply these rules for site admins
if (!GetApplication()->HasAdminGrantForCurrentUser()) {
    // retrieving all user roles
    $sql =
        "SELECT r.role_name " .
        "FROM phpgen_user_roles ur " .
        "INNER JOIN phpgen_roles r ON r.id = ur.role_id " .
        "WHERE ur.user_id = %d";
    $result = $connection->fetchAll(sprintf($sql, $userId));

    // iterating through retrieved roles
    if (!empty($result)) {
        foreach ($result as $row) {
            // is current user a member of the Sales role?
            if ($row['role_name'] === 'Sales') {
                // if yes, allow all actions.
                // otherwise default permissions for this page will be applied
                $permissions->setGrants(true, true, true, true);
                break;
            }
        }
    }
};
}
```

See also: [OnGetCustomRecordPermissions](#)^[192].

5.4.2 Client Side Page Events

Client-side events are fragments of **JavaScript** code executed at the appointed time. They allow you to manipulate with DOM elements directly, using [jQuery](#), and/or via the [Client Side API](#)^[278] provided by PostgreSQL PHP Generator.

To specify a handler for a client-side event, switch to the Events tab of [Page Editor](#)^[33] and double click the appropriate line in the grid. The complete list of currently available client-side events is as follows.

- [OnBeforePageLoad](#)^[138]
- [OnAfterPageLoad](#)^[138]
- [OnInsertFormValidate](#)^[139]
- [OnEditFormValidate](#)^[139]
- [OnInsertFormEditorValueChanged](#)^[140]

- [OnEditFormEditorValueChanged](#)^[142]
- [OnInsertFormLoaded](#)^[144]
- [OnEditFormLoaded](#)^[145]
- [OnCalculateControlValues](#)^[146]

See also: [Server Side Events](#)^[147]

5.4.2.1 OnBeforePageLoad

This event occurs before page loading. It allows you to declare functions and global variables to be used in other client-side events.

Example:

To get the details of the first master record expanded on opening of a certain page of the generated application, define the *expandFirstDetail* function within the OnBeforePageLoad event handler as follows:

```
window.expandFirstDetail = function() {
    var $firstExpandDetailsButton = $('a.js-expand-details').first();
    if (($firstExpandDetailsButton.length > 0) &&
        (eventData = $.data($firstExpandDetailsButton.get(0), "events")) &&
        (eventData.click))
    {
        $firstExpandDetailsButton.click();
        $(window).scrollTop(0);
    }
    else {
        setTimeout(expandFirstDetail, 100);
    }
}
```

This function can be called then within the [OnAfterPageLoad](#)^[138] event handler.

5.4.2.2 OnAfterPageLoad

This event occurs after page has been fully rendered. It does not get triggered until all assets such as images have been completely received and DOM hierarchy has been fully constructed.

Example:

To get details of the first record expanded on the webpage opening, specify the event handler as follows:

```
function expandFirstDetail() {
    var $firstExpandDetailsButton = $('a.js-expand-details').first();
    if (($firstExpandDetailsButton.length > 0) &&
        (eventData = $.data($firstExpandDetailsButton.get(0), "events")) &&
        (eventData.click))
    {
        $firstExpandDetailsButton.click();
        $(window).scrollTop(0);
    }
    else {
        setTimeout(expandFirstDetail, 100);
    }
}
```



```
expandFirstDetail();
```

It is also possible to specify the function above within the [OnBeforePageLoad](#)¹³⁸ event handler. In this case the OnAfterPageload event handler should contain only the function call:

```
window.expandFirstDetail();
```

5.4.2.3 OnInsertFormValidate

This event occurs before submitting of an insert form. It allows you to detect errors on the client side before the form is submitted to the server to avoid the round trip of information necessary for server-side validation. [Live example](#).

Signature:

```
function OnInsertFormValidate (fieldValues, errorInfo)
```

Parameters:

fieldValues	An associative array of values containing user input. To access the value of the editor of the <i>column_name</i> column, use the field_values ['column_name'] syntax.
errorInfo	The object that provides interface (the SetMessage method) to set a validation error message.

Example 1:

The code below demonstrates the verification of the percent number on the record inserting.

```
if (fieldValues['percents'] < 0 || fieldValues['percents'] > 100)
{
    errorInfo.SetMessage('Percent value should be between 0 and 100.');
```

```
    return false;
}
```

Example 2:

The following code is used in the [NBA_demo_project](#) to ensure that home and away teams are different for an inserted game:

```
if (fieldValues['home_team_id'] == fieldValues['away_team_id'])
{
    errorInfo.SetMessage('Home and away teams must be different');
```

```
    return false;
}
```

See also: [OnEditFormValidate](#)¹³⁹

5.4.2.4 OnEditFormValidate

This event occurs before submitting an edit form. It allows you to detect errors on the client before the form is submitted to the server to avoid the round trip of information necessary for server-side validation. [Live example](#).

Signature:

```
function OnEditFormValidate (fieldValues, errorInfo)
```


Parameters:

fieldValues	An associative array of values containing user input. To access the value of the editor of the <i>column_name</i> column, use the field_values ['column_name'] syntax.
errorInfo	The object that provides interface (the SetMessage method) to set a validation error message.

Example 1:

The code below demonstrates the verification of the percent number on the record inserting.

```
if (fieldValues['percents'] < 0 || fieldValues['percents'] > 100)
{
    errorInfo.SetMessage('Percent value should be between 0 and 100.');
```

```
    return false;
}
```

Example 2:

The following code is used in the [NBA_demo_project](#) to ensure that home and away teams are different for an inserted game:

```
if (fieldValues['home_team_id'] == fieldValues['away_team_id'])
{
    errorInfo.SetMessage('Home and away teams must be different');
```

```
    return false;
}
```

See also: [OnInsertFormValidate](#)^[139]

5.4.2.5 OnInsertFormEditorValueChanged

This event occurs on changing a value in the Insert form. [Live example](#).

Signature:

```
function OnInsertFormEditorValueChanged (sender, editors)
```

Parameters:

sender	A control ^[278] whose value has been changed.
editors	An associative array of form controls ^[278] . To access the value of the editor of the <i>column_name</i> column, use the editors['column_name'] syntax.

The examples below show how this method can be used in the [OnInsertFormEditorValueChanged](#)^[140] and [OnEditFormEditorValueChanged](#)^[142] event handlers.

Example 1:

The code below allows you to select a college only for players from U.S. ([see this in action](#)). See how it looks on the webpage at the screen below.

```
console.log(sender);
if (sender.getFieldName() == 'country_id')
{
    console.log(sender.getValue());
}
```



```

    editors['college_id'].enabled(sender.getValue() == 1);
    if (sender.getValue() != 1) {
        editors['college_id'].setValue(null);
        $('#college_id_edit').next().show();
    }
    else
        $('#college_id_edit').next().hide();

```

The image shows two side-by-side screenshots of a player edit form. The left screenshot shows a player from the USA with a dropdown menu open for selecting a college. The right screenshot shows a player from France, where the college dropdown is disabled and a message states "College can be selected only for players from USA." The form fields include Country, NBA Debut, Position, College, Team, and Number.

Example 2:

Another example from the insert and edit forms of the Players page ([see this in action](#)). This piece of code allows you to select player number only if a player's team is already selected (in other words, players who do not belong to a team cannot have numbers).

```

if (sender.getFieldName() == 'current_team_id')
{
    if (sender.getValue() == '')
    {
        editors['current_number'].setValue('');
        editors['current_number'].enabled(false);
        $('#current_number_edit').next().show();
    }
    else

```



```
{
    editors['current_number'].enabled(true);
    $('#current_number_edit').next().hide();
}
}
```

Team

Los Angeles Lakers

Number

24

Team

Please select...

Number

See also: [OnEditFormEditorValueChanged](#)¹⁴²

5.4.2.6 OnEditFormEditorValueChanged

This event occurs on changing a value in the Edit form. [Live example](#).

Signature:

```
function OnEditFormEditorValueChanged(sender, editors)
```

Parameters:

sender	A control whose value has been changed.
editors	An associative array of form controls. To access the value of the editor of the <i>column_name</i> column, use the editors['column_name'] syntax.


The examples below show how this method can be used in the [OnInsertFormEditorValueChanged](#)¹⁴⁰ and [OnEditFormEditorValueChanged](#)¹⁴² event handlers.

Example 1:

The code below allows you to select a college only for players from U.S. ([see this in action](#)). See how it looks on the webpage at the screen below.

```
console.log(sender);
if (sender.getFieldName() == 'country_id')
{
    console.log(sender.getValue());
    editors['college_id'].enabled(sender.getValue() == 1);
    if (sender.getValue() != 1) {
        editors['college_id'].setValue(null);
        $('#college_id_edit').next().show();
    }
    else
        $('#college_id_edit').next().hide();
}
```


Country *

 USA [us]

✕ ▼

NBA Debut *

1996

Position

✕ Guard


College

Please select...

▼

Please select...
Alabama
Alabama-Birmingham
Arizona
Arizona State
Arkansas
Arkansas-Little Rock
Auburn
Augsburg
Austin Peay
Baylor
Blinn Coll. TX (JC)
Boston College
Bowling Green
Bradley
Cal State Fullerton
Cal State-Fullerton
California
Central Michigan
Charleston (SC)

Country *

 France [fr]

✕ ▼

NBA Debut *

1996

Position

✕ Guard


College

Please select...

⛔ ▼

College can be selected only for players from USA.

Team

 Los Angeles Lakers

✕ ▼

Number

24

* - Required field

Save ▼

Cancel

Example 2:

Another example from the insert and edit forms of the Players page ([see this in action](#)). This piece of code allows you to select player number only if a player's team is already selected (in other words, players who do not belong to a team cannot have numbers).

```

if (sender.getFieldName() == 'current_team_id')
{
    if (sender.getValue() == '')
    {
        editors['current_number'].setValue('');
        editors['current_number'].enabled(false);
        $('#current_number_edit').next().show();
    }
    else
    {
        editors['current_number'].enabled(true);
        $('#current_number_edit').next().hide();
    }
}

```


Team	<input type="text" value="Los Angeles Lakers"/>	Team	<input type="text" value="Please select..."/>
Number	<input type="text" value="24"/>	Number	<input type="text"/>

See also: [OnInsertFormEditorValueChanged](#)¹⁴⁰

5.4.2.7 OnInsertFormLoaded

This event occurs after an Insert form is loaded. [Live example](#).

Signature:

```
function OnInsertFormLoaded(editors)
```

Parameters:

editors	An associative array of form controls ²⁷⁸ . To access the value of the editor of the <i>column_name</i> column, use the editors['column_name'] syntax.
---------	---

Example:

This piece of code targets two things:

1) Player number is cleared and the appropriate control becomes disabled for players that do not belong to a team.

```
if (editors['current_team_id'].getValue() == '')
{
    editors['current_number'].setValue('');
    editors['current_number'].enabled(false);
}
else
    editors['current_number'].enabled(true);
```

2) Player's college is cleared and the appropriate control becomes disabled if player's country is not USA (country code == 1). The inline hint describes the reason of the disabling.

```
editors['college_id'].enabled(editors['country_id'].getValue() == 1);
addInlineHint('college_id', 'College can be selected only for players from USA.')
if (editors['country_id'].getValue() != 1)
    editors['college_id'].setValue(null);
```


The insert form of the Players
webpage with disabled event

Country *

Please select...

set default

Height

Birthday

Weight

College

Please select...

Team

Please select...

Number

* - Required field

The insert form of the Players
webpage with enabled event

Country *

Please select...

set default

Height

Birthday

Weight

College

Please select...

Team

Please select...

Number

* - Required field

See also: [OnEditFormLoaded](#)¹⁴⁵

5.4.2.8 OnEditFormLoaded

This event occurs after the Edit form is loaded. [Live example](#).

Signature:

function OnEditFormLoaded(editors)

Parameters:

editors	An associative array of form controls. To access the value of the editor of the <code>column_name</code> column, use the <code>editors['column_name']</code> syntax.
---------	--

Example:

This piece of code targets two things:

- 1) Player number is cleared and the appropriate control becomes disabled for players that do not belong to a team.

```
if (editors['current_team_id'].getValue() == '') {
    editors['current_number'].setValue('');
    editors['current_number'].setEnabled(false);
}
else {
    editors['current_number'].setEnabled(true);
}
```

- 2) Player's college is cleared and the appropriate control becomes disabled if player's country is not USA (country code == 1).


```

editors['college_id'].setEnabled(editors['country_id'].getValue() == 1);
if (editors['country_id'].getValue() != 1)
    editors['college_id'].setValue(null);

```

The edit form of the Players
webpage with disabled event

Country *	<input type="text" value="Argentina [ar]"/>
Height	<input type="text" value="198"/>
Birthday	<input type="text" value="1977-07-28"/>
Weight	<input type="text" value="93.00"/>
College	<input type="text" value="Please select..."/>
Team	<input type="text" value="San Antonio Spurs"/>
Number	<input type="text" value="20"/>

The edit form of the Players
webpage with enabled event

Country *	<input type="text" value="Argentina [ar]"/>
Height	<input type="text" value="198"/>
Birthday	<input type="text" value="1977-07-28"/>
Weight	<input type="text" value="93.00"/>
College	<input type="text" value="Please select..."/>
Team	<input type="text" value="San Antonio Spurs"/>
Number	<input type="text" value="20"/>

See also: [OnInsertFormLoaded](#)^[144]

5.4.2.9 OnCalculateControlValues

This event occurs whenever the values of [calculated_columns](#)^[107] are (re)computed in Edit and Insert forms. [Live examples](#).

Signature:

```
function OnCalculateControlValues(editors)
```

Parameters:

editors	An associative array of form controls. To access the value of the editor of the <i>column_name</i> column, use the editors['column_name'] syntax.
---------	---

Example 1:

This code snippet sets the value of the control displaying the full name according to values entered in the controls displaying first and last names.

```

editors['full_name'].setValue(editors['first_name'].getValue() + ' ' +
    editors['last_name'].getValue());

```

Example 2:

This code snippet sets the value of the control displaying the age according to the entered birthday. To simplify the code, the [Moment.js](#) library is used.

```

if (editors['birthday'].getValue()) {

```



```

        require(['moment'], function(moment) {
            editors['age'].setValue(
                moment().diff(editors['birthday'].getValue(), 'years')
            );
        });
    }

```

See also:

[OnCalculateFields](#)^[194].

5.4.3 Server Side Page Events

Server-side events are fragments of **PHP** code executed at the appointed time. It is possible to use [Server_Side_API](#)^[304] and [environment_variables](#)^[195] within these event handlers.

To specify a handler for a client-side event, switch to the Events tab of [Page Editor](#)^[33] and double click the appropriate line in the grid. The complete list of currently available server-side events is as follows.

- [OnBeforePageExecute](#)^[148]
- [OnPreparePage](#)^[149]
- [OnPageLoaded](#)^[150]
- [OnCustomRenderColumn](#)^[151]
- [OnCustomRenderPrintColumn](#)^[154]
- [OnCustomRenderExportColumn](#)^[154]
- [OnCustomHTMLHeader](#)^[154]
- [OnExtendedCustomDrawRow](#)^[155]
- [OnCustomRenderTotals](#)^[158]
- [OnGetCustomTemplate](#)^[159]
- [OnCustomDrawRow](#)^[174]
- [OnAfterUpdateRecord](#)^[176]
- [OnAfterDeleteRecord](#)^[177]
- [OnAfterInsertRecord](#)^[175]
- [OnBeforeUpdateRecord](#)^[179]
- [OnBeforeDeleteRecord](#)^[180]
- [OnBeforeInsertRecord](#)^[178]
- [OnGetFieldValue](#)^[181]
- [OnGetCustomExportOptions](#)^[181]
- [OnPrepareFilterBuilder](#)^[183]
- [OnPrepareColumnFilter](#)^[183]
- [OnGetCustomFormLayout](#)^[184]
- [OnGetCustomColumnGroup](#)^[189]

- [OnCustomCompareValues](#)^[190]
- [OnFileUpload](#)^[190]
- [OnGetCustomPagePermissions](#)^[191]
- [OnGetCustomRecordPermissions](#)^[192]
- [OnCustomDefaultValues](#)^[193]
- [OnCalculateFields](#)^[194]

See also: [Client side events](#)^[137], [Application-level events](#)^[117].

5.4.3.1 OnBeforePageExecute

This event handler contains PHP code to be included into the page. This code is executed only once and intended to create global objects, declare functions, include third-party libraries, and so on.

Example

Suppose we decided to implement a syntax highlighting using the [GeSHi library](#) somewhere in the generated application. To implement such a feature, we need (among other things) to include the main library file into the generated scripts, so the event handler should be specified as follows:

```
include_once    '../geshi/geshi.php';
```

5.4.3.2 OnAddEnvironmentVariables

This event allows you to add environment variables for a page. You can access the values of these variables in any server-side event of the page via [GetEnvVar\(\)](#)^[304] method of the Page class. To set custom variables to be used in any server-side event of any application page, use the global [OnAddEnvironmentVariables](#)^[119] event handler.

Signature:

```
function OnAddEnvironmentVariables($page,    &$variables)
```

Parameters:

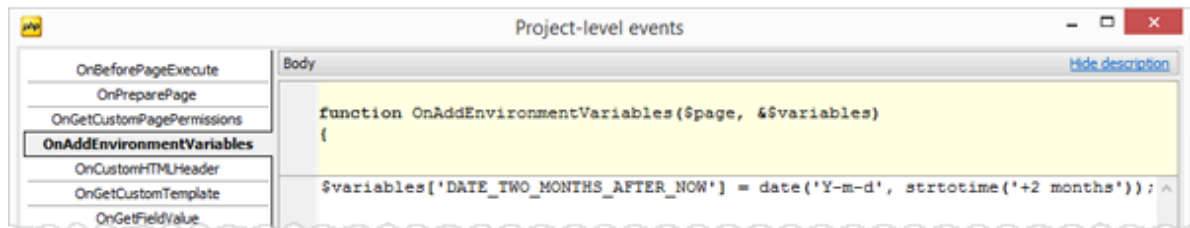
\$page	An instance of the Page class ^[304] declared in <i>components/page.php</i> .
\$variables	An associative array of environment variables. Use it to add your own variables for a page.

Example 1:

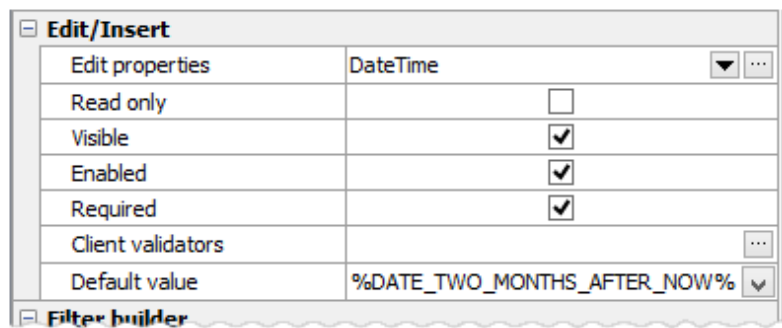
The following code defines a new variable, which can be used in lookup filter criteria, on data filtering, as a default value.

```
$variables['DATE_TWO_MONTHS_AFTER_NOW'] = date('Y-m-d',    strtotime('+2    months'));
```


Adding a custom variable



Using this variable as a default value



5.4.3.3 OnPreparePage

This piece of code is a method of the Page class that is called at the end of the constructor. It allows you to customize all members of the class.

Signature:

```
function OnPreparePage()
```

Parameters:

This method has no parameters.

Example

The following code hides the *update_date_time* column from the data grid and Single Record View form as well as the *status* column from Edit and Insert forms from all users except ones who have the Admin privilege for the current page.

```
if (!GetApplication()->IsLoggedInAsAdmin()) {
    $updateDateTimeColumnName = 'update_date_time';
    $column = $this->GetGrid()->getSingleRecordViewColumn($updateDateTimeColumnName);
    if ($column) {
        $column->setVisible(false);
    }
    $column = $this->GetGrid()->getViewColumn($updateDateTimeColumnName);
    if ($column) {
        $column->setVisible(false);
    }

    $statusColumnName = 'status';
    $column = $this->GetGrid()->getEditColumn($statusColumnName);
    if ($column) {
        $column->setVisible(false);
    }
}
```



```

    }
    $column = $this->GetGrid()->getInsertColumn($statusColumnName);
    if ($column) {
        $column->setVisible(false);
    }
}

```

See also: [OnPageLoaded](#)^[150].

5.4.3.4 OnPageLoaded

This event occurs after the page is completely loaded from the server. It allows you, for example, to add additional filters to the dataset.

Signature:

```
function OnPageLoaded()
```

Parameters:

This method has no parameters.

Example 1

The following code is used in our [Schema Browser Demo](#) to filter database objects belonging to the selected database:

```
applyDatasetFilter($this->dataset);
```

Here `applyDatasetFilter()` is a function defined in the `schema_browser_utils.php` file as follows:

```

function applyDatasetFilter(Dataset $dataset) {
    if (GetApplication()->IsGetValueSet('database'))
        $value = GetApplication()->GetGetValue('database');
    else
        $value = ArrayUtils::GetArrayValueDef($_COOKIE, 'database');
    if (StringUtils::IsNullOrEmpty($value)) {
        $globalConnectionOptions = GetGlobalConnectionOptions();
        $value = $globalConnectionOptions['database'];
    }
    if (!StringUtils::IsNullOrEmpty($value)) {
        $dataset->AddFieldFilter('Database_name', new FieldFilter($value, '='));
        setcookie('database', $value);
    }
}

```

Example 2

The following code is used to filter countries by life expectancy:

```

// default values
$minLifeExpectancy = 40;
$maxLifeExpectancy = 90;

// checking values in the browser address line
if (GetApplication()->isGetValueSet('minLifeExpectancy') &&
    GetApplication()->isGetValueSet('maxLifeExpectancy')) {
    $minLifeExpectancy = GetApplication()->GetGetValue('minLifeExpectancy');
    $maxLifeExpectancy = GetApplication()->GetGetValue('maxLifeExpectancy');
}

```



```

} // or (if not found) in the session
elseif (GetApplication()->IsSessionVariableSet('minLifeExpectancy') &&
        GetApplication()->IsSessionVariableSet('maxLifeExpectancy')) {
    $minLifeExpectancy = GetApplication()->GetSessionVariable('minLifeExpectancy');
    $maxLifeExpectancy = GetApplication()->GetSessionVariable('maxLifeExpectancy');
}

// applying filter to the dataset
$this->dataset->AddCustomCondition(
    "life_expectancy >= {$minLifeExpectancy} ".
    "AND life_expectancy <= {$maxLifeExpectancy}");

// saving calculated values to the session
GetApplication()->SetSessionVariable('minLifeExpectancy', $minLifeExpectancy);
GetApplication()->SetSessionVariable('maxLifeExpectancy', $maxLifeExpectancy);

```

See also: [OnPreparePage](#)¹⁴⁹.

5.4.3.5 OnCustomRenderColumn

This event occurs before column rendering and allows you to completely replace the cell content. It is an extremely useful event for conditional rendering or embedding third-party components to extend standard functionality.

Signature:

```
function OnCustomRenderColumn ($fieldName, $fieldData, $rowData,
    &$customText, &$handled)
```

Parameters:

\$fieldName	The field name for the currently processed cell.
\$fieldData	The data of currently processed cell.
\$rowData	The associative array of values that corresponds to the currently processed row.
\$customText	A string to replace the original cell content.
\$handled	A parameter to indicate whether the event handler takes effect. Set \$handled to true to apply the new content.

Example 1:

Suppose a table 'employee' has a column storing data about the employee's sex in that way that '1' corresponds to male and '2' to female. Our goal is to represent the employees sex as 'M' and 'F' for men and women accordingly. To do so, you could specify the OnCustomRenderColumn event handler as follows:

```

if ($fieldName == 'sex') {
    $customText = $rowData['sex'] == 1 ? 'M' : 'F';
    $handled = true;
}

```

Example 2:

The code below is used in [our demo project](#) to show the number of overtime periods played in the game.

```

if ($fieldName == 'overtime_count') {
    if ($fieldData > 0)
        $customText = $fieldData . ' OT';
}

```



```

else
    $customText = '-';
    $handled = true;
}

```

The screenshot demonstrates the result of this event fired on the Games webpage.

31 Oct 2008	Philadelphia 76ers	116 : 87	New York Knicks	-
31 Oct 2008	Toronto Raptors	112 : 108	Golden State Warriors	1 OT
31 Oct 2008	Miami Heat	103 : 77	Sacramento Kings	-
31 Oct 2008	Boston Celtics	96 : 80	Chicago Bulls	-











Example 3:

The code below is used in [our demo project](#) to add flag images to the "Country" column.

```

if ($fieldName == 'current_team_id')
{
    if (!isset($fieldData))
    {
        $customText = 'Free agent';
        $handled = true;
    }
}
elseif ($fieldName == 'country_id')
{
    $countriesPics = array (
        1 => 'us',
        2 => 'fr',
        3 => 'br',
        4 => 'it',
        6 => 'sp',
        8 => 'ar',
        11 => 'li',
        15 => 'ru',
        18 => 'gr',
        20 => 'ge'
    );
    $customText =
        '<div style="width: 80px; text-align: left;">' .
            '' .
            ' ' . $fieldData .
        '</div>';
    $handled = true;
}

```


	Kevin	Garnett	211	114.8	1995	Forward	 USA
	Marc	Gasol	216	120.2	2008	Center	 Spain
	Pau	Gasol	213	113.4	2001	Forward - Center	 Spain
	Rudy	Gay	203	100.7	2006	Forward	 USA
	Manu	Ginobili	198	93.0	2002	Guard	 Argentina

Example 4

This example demonstrates the applying of the syntax highlighting provided by the Geshi library. We use this library to highlight SQL syntax in strings stored in the 'Body' column.

```
if ($fieldName == 'Body') {
    $source = $rowData['Body'];
    $language = 'sql';
    $formatted_sql = SqlFormatter::format($source, false);
    $geshi = new GeSHi($formatted_sql, $language);
    $customText = '<div align="left">'.$geshi->parse_code().'</div>';
    $handled = true;
}
```

Plain text

```
SELECT A.Employee_Name, B.Spouse_Name
FROM Employee A LEFT OUTER JOIN Spouse B
ON A.Employee_ID = B.Employee_ID
```

Highlighted text

```
SELECT A.Employee_Name, B.Spouse_Name
FROM Employee A LEFT OUTER JOIN Spouse B
ON A.Employee_ID = B.Employee_ID
```


See also: [OnCustomRenderPrintColumn](#)^[154], [OnCustomRenderExportColumn](#)^[154]

5.4.3.6 OnCustomRenderPrintColumn

This event occurs before column rendering on the print page and allows you to replace cell content. It is an extremely useful for conditional rendering or embedding third-party components to extend standard functionality.

Signature:

```
function OnCustomRenderPrintColumn ($fieldName, $fieldData, $rowData,
    &$customText, &$handled)
```

Parameters:

\$fieldName	The field name of currently processed cell.
\$fieldData	The data of currently processed cell.
\$rowData	The associative array of values that corresponds to the currently processed row.
\$customText	A string to replace the original cell content.
\$handled	A parameter to indicate whether the event handler executed. Set \$handled to true in the event handler to apply new content.

See also: [OnCustomRenderColumn](#)^[151], [OnCustomRenderExportColumn](#)^[154]

5.4.3.7 OnCustomRenderExportColumn

This event occurs before column rendering into Word, Excel, etc. and allows you to replace cell content. It is an extremely useful event for conditional rendering.

Signature:

```
function OnCustomRenderExportColumn ($fieldName, $fieldData, $rowData,
    &$customText, &$handled)
```

Parameters:

\$fieldName	The field name of currently processed cell.
\$fieldData	The data of currently processed cell.
\$rowData	The associative array of values that corresponds to the currently processed row.
\$customText	A string to replace the original cell content.
\$handled	A parameter to indicate whether the event handler executed. Set \$handled to true in the event handler to apply new content.

See also: [OnCustomRenderColumn](#)^[151], [OnCustomRenderPrintColumn](#)^[154]

5.4.3.8 OnCustomHTMLHeader

This event occurs when the head section of the page is generating. Use it to provide additional information for the HEAD section of the page (such as keywords, author, or description).

Signature:

```
function OnCustomHTMLHeader ($page, &$customHtmlHeaderText)
```


Parameters:

\$page	An instance of the Page class declared in <i>components/page.php</i>
\$customHtmlHeaderText	A string to place into the head section

Example 1:

The following code demonstrates how to use this event to add metadata to the generated webpage.

```
$customHtmlHeaderText = '<meta name="copyright" content="SQL Maestro Group"/>';
```

Example 2:

The following code includes external style file and javascript library to the page.

```
$customHtmlHeaderText =
    '<link rel="stylesheet" href="external_data/pg_styles.css">' .
    '<script src="external_data/pg_utils.js"></script>';
```

5.4.3.9 OnExtendedCustomDrawRow

This event occurs when rendering a grid row and allows you to change the row and/or cell styles directly. It is an extremely useful event for conditional formatting such as changing font color, font styles, row background color, cell background color, etc. The specified styles and classes are applied to <tr> and <td> tags accordingly.

Signature:

```
function OnExtendedCustomDrawRow ($rowData, &$rowCellStyles, &$rowStyles,
    &$rowClasses, &$cellClasses)
```

Parameters:

\$rowData	The associative array of values that corresponds to the currently processed row.
\$rowCellStyles	The associative array of styles. Each field name is associated with its style string.
\$rowStyles	Use this string to modify styles of a whole row.
\$rowClasses	Use this string to add a class to a whole row.
\$cellClasses	The associative array of cell classes.

This event (as well as the [OnCustomDrawRow](#)^[174] one) is used for conditional formatting. The only difference between these two events is that OnCustomDrawRow has a more understandable parameter list while OnExtendedCustomDrawRow provides more flexible abilities.

Example 1:

The code below is used in [our online demo](#) to display the winning team and the losing team scores according to the current theme.

```
if ($rowData['home_team_score'] > $rowData['away_team_score']) {
    $cellClasses['home_team_score'] = 'win-score';
    $cellClasses['away_team_score'] = 'loss-score';
}
```



```

}
else {
    $cellClasses['home_team_score'] = 'loss-score';
    $cellClasses['away_team_score'] = 'win-score';
}

```

The 'win-score' and 'loss-score' classes are defined in User-defined styles as follows:

```

.base-score {
    font-size: 1.4em;
    font-weight: bold;
}

.win-score {
    &:extend(.base-score);
    color: @brand-danger;
}

.loss-score {
    &:extend(.base-score);
}

```

The screenshot below demonstrates the result of the event fired on the Game list webpage in Cyborg Bootswatch theme for Bootstrap.

Home team	Home Score	Away Score	Away team
Los Angeles Lakers	116	103	Los Angeles Clippers
Miami Heat	107	95	Chicago Bulls
Indiana Pacers	97	87	Orlando Magic

The next one shows the result of the event fired on this webpage in Facebook Bootswatch theme.

Home team	Home Score	Away Score	Away team
Los Angeles Lakers	116	103	Los Angeles Clippers
Miami Heat	107	95	Chicago Bulls
Indiana Pacers	97	87	Orlando Magic
Cleveland Cavaliers	98	94	Brooklyn Nets

Example 2

The code below is used in [our online demo](#) to highlight fields representing player's height depending on the value.

```
$height = $rowData['height'];
if ($height > 200)
    $cellClass = 'tall';
elseif ($height > 185)
    $cellClass = 'medium-height';
else
    $cellClass = 'undersized';
$cellClasses ['height'] = $cellClass;
```

The 'undersized', 'medium-height', and 'tall' classes are defined in User Defined Styles as follows:







```
@height-color: @state-warning-bg;

.undersized {
    background-color: lighten(@height-color, 10%);
}







.medium-height {
    background-color: @height-color;
}

.tall {
    background-color: darken(@height-color, 10%);
}
```

The screenshot below demonstrates the result of the event fired on the Game list webpage in Journal Bootswatch theme for Bootstrap.

Photo	First name	Last name	Country	Height
	Chris	Paul	 USA	183
	Paul	Pierce	 USA	201
	Rodney	Stuckey	 USA	196

The next one shows the result of the event fired on this webpage in Superhero Bootswatch theme.

Photo	First name	Last name	Country	Height
	Chris	Paul	 USA	183
	Paul	Pierce	 USA	201
	Rodney	Stuckey	 USA	196

5.4.3.10 OnCustomRenderTotals

This event occurs before total values rendering and allows you to replace the default total footer content. To use it, enable the grid footer in the [View properties](#) ⁴⁴ of the corresponding column.

Signature:

```
function OnCustomRenderTotals ($totalValue, $aggregate, $columnName,
    &$customText, &$handled)
```

Parameters:

\$totalValue	The value of currently processed total.
\$aggregate	The string representation of total aggregate ('AVG', 'SUM', 'MIN', 'MAX', 'COUNT').
\$columnName	The column name of currently processed total.
\$customText	A string to replace the total cell content.
\$handled	A parameter to indicate whether the event handler executed. Set \$handled to true in the event handler to apply new content.

Example 1:

The code below changes the caption of the 'amount' column footer and gives it strong importance.

```
if ($columnName == 'amount')
{
    $customText = '<strong>Total: $' . $totalValue . '</strong>';
    $handled = true;
}
```

This picture demonstrates the difference between a webpage generated with default footer and a webpage with the OnCuStomRenderTotals event specified in this way:

Rental Id	Amount	Payment Date
3021	2.9900	2005-05-25 11:30:37
4020	0.9900	2005-05-28 10:35:23
2785	5.9900	2005-06-15 00:54:12
726	4.9900	2005-06-16 15:18:57
197	4.9900	2005-06-18 08:41:48
SUM = 19.95		

Rental Id	Amount	Payment Date
3021	2.9900	2005-05-25 11:30:37
4020	0.9900	2005-05-28 10:35:23
2785	5.9900	2005-06-15 00:54:12
726	4.9900	2005-06-16 15:18:57
197	4.9900	2005-06-18 08:41:48
Total: \$19.95		

Example 2:
The code below is used in [our online demo](#) on the 'Teams' page to display the year of foundation of the oldest team on the page in more convenient form.

```
if ($columnName == 'year_founded') {  
    $customText = 'MIN: ' . intval(str_replace(',', ' ', $totalValue));  
    $handled = true;  
}
```

Dallas Mavericks	1980	The Dallas are a profes Texas, US won tw championsh
MIN = 1,945.00		

Dallas Mavericks	1980	The Dallas are a profes Texas, USA won tw championsh
MIN: 1945		

5.4.3.11 OnGetCustomTemplate

This event allows you to customize almost any part of the generated application using an own template instead of the default one.

Signature:

```
function OnGetCustomTemplate ($part, $mode, &$result, &$params)
```

Parameters:

\$part	The part of the page to be affected by the template.
\$mode	The current state of the webpage the template to be used.
\$result	The path to the file to be used as template according to the <i>components/templates/custom_templates</i> folder.
\$params	An associative array of additional parameters you want to assign to the template.

PHP Generator uses [Smarty 2.0](#) as template language. If you are not familiar with web template engines, you might want to [start here](#).

Our Feature Demo provides [a number of live examples](#) of customized templates. We would recommend you to study them carefully before trying to customize your

application.

To customize a webpage, you need to:

- create a new template to be used for this webpage;
- instruct PHP Generator to use this template for the selected webpage.

The simplest way to create a new template is to modify an existing one (it is much easier than create a new template "from scratch"). Use the table below to select the template corresponding to the necessary state of page and page part. Custom template files must be uploaded to the '**components/templates/custom_templates**' directory. This folder should be created by the user of PHP Generator and its content is not changing during the PHP Generator sessions.

To instruct PHP Generator to use a customized template file for a certain webpage, specify the following code in the page's OnGetCustomTemplate event handler (of course you should replace YourPart and YourMode to the appropriate values from the table below).

```
if ($part == YourPart && $mode == YourMode) {
    $result = 'your_template_file.tpl';
}
```

The following table shows how to customize various pages used in the generated applications.

State of the webpage	Page Part	Default template	Parameters
All Pages ¹⁶⁷	webpage layout	common/layout.tpl	PagePart::Layout Any PageMode
	page list	page_list_menu.tpl (for top-side menu) page_list_sidebar.tpl (for sidebar menu)	PagePart::PageList Any PageMode
List page ¹⁶⁸	table grid	list/grid_table.tpl (for grid view mode) list/grid_card.tpl (for card view mode)	PagePart::Grid PageMode::ViewAll
	grid toolbar	list/grid_toolbar.tpl	PagePart::GridToolbar
	single row	list/single_row.tpl (for grid view mode) list/single_row_card.tpl	PagePart::GridRow PageMode::ViewAll

		(for card view mode)	
Single Record View 169	separate page	view/grid.tpl	PagePart::RecordCard PageMode::View
	modal dialog	view/record_card_view.tpl	PagePart::VerticalGrid PageMode::ModalView
	inline form	view/ record_card_inline_view.tpl	PagePart::VerticalGrid PageMode::InlineView
Edit form 170	editors area	forms/form.tpl	PagePart::VerticalGrid PageMode::FormEdit
	separate page form	forms/page_form.tpl	PagePart::VerticalGrid PageMode::Edit
	modal dialog	forms/modal_form.tpl	PagePart::VerticalGrid PageMode::ModalEdit
	inline form	forms/inline_form.tpl	PagePart::VerticalGrid PageMode::InlineEdit
Insert form 170	editors area	forms/form.tpl	PagePart::VerticalGrid PageMode::FormInsert
	separate page form	forms/page_form.tpl	PagePart::VerticalGrid PageMode::Insert
	modal dialog	forms/modal_form.tpl	PagePart::VerticalGrid PageMode::ModalInsert
	inline form	forms/inline_form.tpl	PagePart::VerticalGrid PageMode::InlineInsert
Print 172	webpage layout (list page)	print/page.tpl	PagePart::PrintLayout PageMode::PrintAll
	data grid (list page)	print/grid.tpl	PagePart::Grid PageMode::PrintAll
	detail page	print/detail_page.tpl	PagePart::PrintLayout PageMode::PrintDetailPage
	webpage layout (single record)	print/page.tpl	PagePart::PrintLayout PageMode::PrintOneRecord

	data grid (single record)	view/print_grid.tpl	PagePart::Grid PageMode::PrintOneRecord
Export	data grid	export/pdf_grid.tpl, export/excel_grid.tpl, export/csv_grid.tpl, etc	PagePart::Grid PageMode::ExportPdf, PageMode::ExportExcel, PageMode::ExportCsv, etc
	single record (PDF only)	export/pdf_record.tpl	PagePart::RecordCard PageMode::ExportPdf
Record comparison	data grid	compare/grid.tpl	PagePart::Grid PageMode::Compare
Login page		login_page.tpl	PagePart::LoginPage
Login control		login_control.tpl	PagePart::LoginControl
Home page		home_page.tpl	PagePart::HomePage
Navigation		navigation.tpl	PagePart::Navigation

The following table shows how to customize user registration and password recovering pages. These parameters can be used only for the project-level version of this event.

Registration Page	registration_page.tpl	PagePart::RegistrationPage
Registration Form	registration_form.tpl	PagePart::RegistrationForm
Password Recovery Page	recovering_password_page.tpl	PagePart::PasswordRecovery
Reset Password Page	reset_password_page.tpl	PagePart::ResetPassword
Resend Verification Email Page	resend_verification_page.tpl	PagePart::ResendVerification

The following table shows how to customize [emails sent to users](#)^[256] on registration and password recovering. These parameters can be used only for the project-level version of this event.

Verification email subject	user_verification_subject.tpl	PagePart::Mail PageMode:: MailVerificationSubject
Verification email body	user_verification_body.tpl	PagePart::Mail PageMode:: MailVerificationBody
Password reset email subject	recovering_password_subject.tpl	PagePart::Mail PageMode:: MailRecoveringPasswordSubject
Password reset email body	recovering_password_body.tpl	PagePart::Mail PageMode:: MailRecoveringPasswordBody

Example 1:

Suppose we need to use *components/templates/custom_templates/staff_edit.tpl* file as a template for the edit form of a webpage.

```
if ($part == PagePart::VerticalGrid && $mode == PageMode::Edit) {
    $result = 'staff_edit.tpl';
}
```

Now we can compare the result webpages without using this event and with the enabled one:

Standard edit form:

The 'Staff' form contains the following fields and controls:

- Save / Cancel buttons:** Located at the top and bottom of the form.
- First Name ***: Text input with 'Mike'.
- Last Name ***: Text input with 'Hillyer'.
- Picture**: A portrait of Mike Hillyer. Below it are 'Keep', 'Remove', and 'Replace' buttons, and a 'Browse...' button.
- Email**: Text input with 'Mike.Hillyer@sakilastaff.com'.
- Password**: Password input field with masked characters '*****'.
- Legend**: '* - Required field'.

Custom edit form:

The 'Mike Hillyer Info' form is divided into two sections:

- Personal Data**:
 - Full name ***: Split into a text input 'Hillyer' and a dropdown menu showing 'Mike'.
 - Picture**: A portrait of Mike Hillyer. Below it are 'Keep', 'Remove', and 'Replace' buttons, and a 'Browse...' button.
- Login Information**:
 - Email**: Text input with 'Mike.Hillyer@sakilastaff.com'.
 - Password**: Password input field.
 - Legend**: '* - Required field'.
- Save Changes / Cancel buttons**: Located at the bottom of the form.

Example 2:

A single event handler can be used to customize multiple templates:

```
if ($part == PagePart::Grid && $mode == PageMode::ViewAll)
    $result = 'games_grid.tpl';
if ($part == PagePart::VerticalGrid && $mode == PageMode::Edit)
    $result = 'games_edit.tpl';
if ($part == PagePart::RecordCard && $mode == PageMode::View)
    $result = 'games_view.tpl';
```

Example 3

This example shows how to fill a SELECT input with names of all MySQL databases available at the server.

1. Specify the OnGetCustomTemplate event handler as follows:

```
// set the template file
if ($part == PagePart::Grid && $mode == PageMode::ViewAll) {
    $result = 'custom_grid.tpl';
}

// get the list of available databases
```



```

$queryResult = array();
$this->GetConnection()->ExecQueryToArray('SHOW DATABASES', $queryResult);

// fill a one-dimensional array with database names
$databasesNames = array();
foreach($queryResult as $row)
    $databasesNames[] = $row[0];

// assign the parameter in the template
$params['databaseNames'] = $databasesNames;
}

```

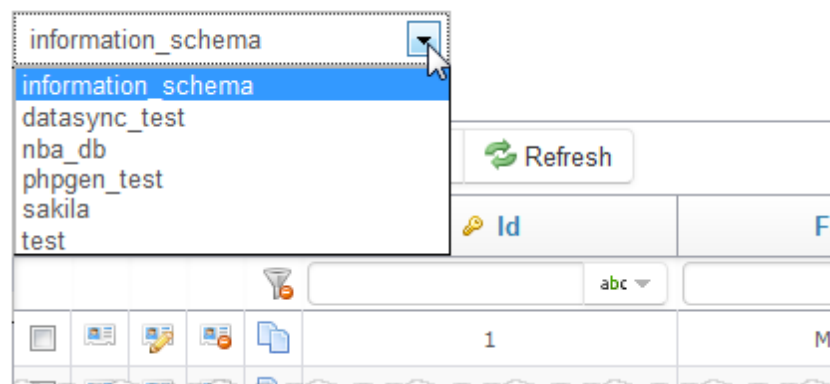
2. Add the following code to the template file:

```

<select name="database">
    {foreach from=$databasesNames item=database}
        <option value="{ $database}">{ $database}</option>
    {/foreach}
</select>

```

The screen below demonstrates the result output.



Example 4

This example shows the use of the superglobal \$_GET array values in a custom template.

1. Specify the OnGetCustomTemplate event handler as follows:

```

if ($part == PagePart::Grid && $mode == PageMode::ViewAll) {
    $result = 'custom_grid.tpl';

    // check if the parameter custom_var available
    if (GetApplication()->IsGETValueSet('custom_var'))
        // found
        $params['custom_var'] = GetApplication()->GetGETValue('custom_var');
    else
        // not found. Assign a default value.
        $params['custom_var'] = '5 (default)';
}

```

2. Add the following code to the template file:

```

<div>
<h4>I am a custom template with a custom variable $custom_var. Its value

```



```

        {if !isset($custom_var)}
            still undefined
        {else}
            = {$custom_var}
        {/if}.
    </h4>
</div>

```

Example 4

Let's see the login form customization used in [our NBA online demo](#).

1. Specify the OnGetCustomTemplate event handler as follows:

```

if ($part == PagePart::LoginPage) {
    $result = 'login_page.tpl';
}

```

2. Add the following code to the template file:

```

<h2>Logins</h2>
<table class="table table-bordered">
    <thead>
        <tr>
            <th>Username</th>
            <th>Password</th>
            <th>Description</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>admin</td>
            <td>admin</td>
            <td>Can modify any record at any page and manage other users.</td>
        </tr>
        <tr>
            <td>lakers</td>
            <td>lakers</td>
            <td>Can modify team info, home games and players for LA Lakers.</td>
        </tr>
        <tr>
            <td>boston</td>
            <td>boston</td>
            <td>The same for the Boston Celtics team.</td>
        </tr>
        <tr>
            <td>game_manager</td>
            <td>game</td>
            <td>Can access only game list. Can modify any game.</td>
        </tr>
    </tbody>
</table>

```

Example 5

The example below is used in our NBA demo application to customize the template used to export a single record to PDF.

```

if ($part == PagePart::RecordCard && $mode == PageMode::ExportPdf) {
    $result = 'game_pdf.tpl';
}

```


Live example can be found [here](#).

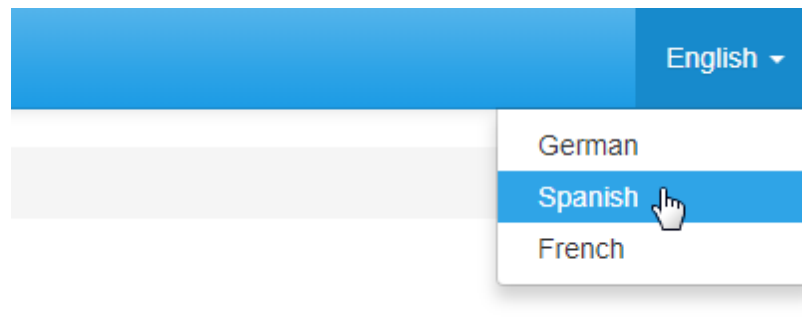
5.4.3.11.1 Common templates

Common templates are applied for all pages (List, View, Edit, Insert, etc). The table below shows how to customize these templates.

State of the webpage	Page Part	Default template	Parameters
All Pages	webpage layout	common/layout.tpl	PagePart::Layout Any PageMode
	page list	page_list_menu.tpl (for top-side menu) page_list_sidebar.tpl (for sidebar menu)	PagePart::PageList Any PageMode

Example.

This example learns how to implement runtime language selection.



You can see it in action in our [Feature Demo](#). To implement this feature, we have to create a template containing the language menu and instruct PHP Generator to use it.

Template code is as follows:

```
{include file='page_list_menu.tpl'}
<ul class="nav navbar-nav navbar-right">
  <li class="dropdown">
    <a href="#" class="dropdown-toggle" data-toggle="dropdown"
      role="button" aria-haspopup="true" aria-expanded="false">
      {availableLangs[$currentLang]}
      <span class="caret"></span>
    </a>
    <ul class="dropdown-menu" id="langs">
      {foreach item=lang key=key from=$availableLangs}
        {if $currentLang != $key}
          <li><a href="#" data-lang="{ $key }">{$lang}</a></li>
        {/if}
      {/foreach}
    </ul>
  </li>
</ul>
```



```

        </ul>
    </li>
</ul>

```

Event handler is as follows:

```

if ($part == PagePart::PageList)
{
    $langs = array(
        'en' => 'English',
        'de' => 'German',
        'es' => 'Spanish',
        'fr' => 'French',
    );

    $lang = 'en'; // default language

    // trying to retrieve language from cookies
    if (isset($_COOKIE['lang']) && $_COOKIE['lang'] && isset($langs[$_COOKIE['lang']])) {
        $lang = $_COOKIE['lang'];
    }

    $params['availableLangs'] = $langs;
    $params['currentLang'] = $lang;

    $result = 'custom_menu.tpl';
}

```

Almost done. Now we need to handle the language selection. This can be done with [user-defined JavaScript](#)^[245]:

```

function handleLanguageSelection() {
    $("#langs").on("click", "a", function (e) {
        var query = jQuery.query;
        query = query.set('lang', $(this).data('lang'));
        window.location = query;
        e.preventDefault();
    });
}

require(['jquery'], function () {
    $(function () {
        handleLanguageSelection();
    });
});

```

That's all. Don't forget to place [language_files](#)^[273] to the components/languages directory.

5.4.3.11.2 Data Grid

The table below show parameters for customizing the list page (data grid).

State of the webpage	Page Part	Default template	Parameters

List page	table grid	list/grid_table.tpl (for grid view mode)	PagePart::Grid PageMode::ViewAll
	single row	list/grid_card.tpl (for card view mode)	
		list/single_row.tpl (for grid view mode)	PagePart::GridRow PageMode::ViewAll
		list/single_row_card.tpl (for card view mode)	

As you can see from the table above, it is possible to customize whole data grid or just a template for displaying a single row.

Examples

Examples for both [table](#) and [card](#) view modes can be found in the Feature Demo application. Each example contains links that allow you to view the complete code of overwritten templates as well as the code of the event handler. You can also [download the demo project file](#) for a deeper investigation.

5.4.3.11.3 Single Record View

The table below show parameters for customizing the single record view form.

State of the webpage	Page Part	Default template	Parameters
Single Record View	separate page	view/grid.tpl	PagePart::RecordCard PageMode::View
	modal dialog	view/record_card_view.tpl	PagePart::VerticalGrid PageMode::ModalView
	inline form	view/record_card_inline_view.tpl	PagePart::VerticalGrid PageMode::InlineView

Examples

A [quite simple example](#) can be found in our Feature Demo. A [more complex example](#) can be found in the NBA Demo. You can study these examples online or [download project files](#) for a deeper investigation.

Common

Dimensions & Display

Hardware & Software

Camera & Battery

Model Name Apple iPhone X**Release Year** 2018**Release Month** October**Colors** Silver,Space Gray**Photo front**

The screenshot above demonstrates how it is possible to use tabs in the single record view forms.

5.4.3.11.4 Edit and Insert forms

The table below show parameters for customizing Edit and Insert forms.

State of the webpage	Page Part	Default template	Parameters
Edit form	editors area	forms/form.tpl	PagePart::VerticalGrid PageMode::FormEdit
	separate page form	forms/page_form.tpl	PagePart::VerticalGrid PageMode::Edit
	modal dialog	forms/modal_form.tpl	PagePart::VerticalGrid PageMode::ModalEdit
	inline form	forms/inline_form.tpl	PagePart::VerticalGrid PageMode::InlineEdit
Insert form	editors area	forms/form.tpl	PagePart::VerticalGrid PageMode::FormInsert
	separate page form	forms/page_form.tpl	PagePart::VerticalGrid PageMode::Insert

	modal dialog	forms/modal_form.tpl	PagePart::VerticalGrid PageMode::ModalInsert
	inline form	forms/inline_form.tpl	PagePart::VerticalGrid PageMode::InlineInsert

You can customize the whole form or only the area containing the controls. The picture below shows the difference between these concepts.

Edit Form

Edit The Matrix movie [X]

[About](#) [Additional info](#) [Media](#)

Release Date * 1999-03-30 [Calendar icon]

Original language * English [X] [v]

Genre Adventure [X] [v]

Runtime * 136 min

Rating * 7.72

Cancel Save and continue editing Save

Editors Area

Examples

Examples for all page modes ([Separate Page](#), [Modal Dialog](#), and [Inline Form](#)) can be found in the Feature Demo application. Additionally the demo contains an example of a

[wizard form](#). Each example contains links that allow you to view the complete code of overwritten templates as well as the code of the event handler. You can also [download the demo project file](#) for a deeper investigation.

5.4.3.11.5 Print

The table below show parameters for customizing printer-friendly version of the page.

State of the webpage	Page Part	Default template	Parameters
Print	webpage layout (list page)	print/page.tpl	PagePart::PrintLayout PageMode::PrintAll
	data grid (list page)	print/grid.tpl	PagePart::Grid PageMode::PrintAll
	detail page	print/detail_page.tpl	PagePart::PrintLayout PageMode::PrintDetailPage
	webpage layout (single record)	print/page.tpl	PagePart::PrintLayout PageMode::PrintOneRecord
	data grid (single record)	view/print_grid.tpl	PagePart::Grid PageMode::PrintOneRecord

Example

This example learns how to implement a two-column version of the printer-friendly page. It [can be seen live](#) in our Feature Demo that also contains an example of customization templates for [printing a single record](#). You can study these examples online or [download the demo project file](#) for a deeper investigation.

Exporting & Printing.Custom Grid

Afghanistan (Asia)

Become independent (year)	1919
Population	22,720,000
Life Expectancy	45.90

Angola (Africa)

Become independent (year)	1975
Population	12,878,000
Life Expectancy	38.30

Albania (Europe)

Become independent (year)	1912
Population	3,401,200
Life Expectancy	71.60

Andorra (Europe)

Become independent (year)	1278
Population	78,000
Life Expectancy	83.50

The template code is as follows:

```
<div style="max-width: 800px">
{foreach item=Row from=$Rows name=RowsGrid}
    <div style="width: 50%; float: left; padding-bottom: 2em; border-bottom: 1px solid #ddd">
        <h3>{$Row.1} <small>({$Row.2})</small></h3>
        <table>
            <tr>
                <td style="text-align:right"><strong>Become independent (year)</strong></td>
                <td style="text-align:left">{$Row.3}</td>
            </tr>
            <tr>
                <td style="text-align:right"><strong>Population</strong></td>
                <td style="text-align:left">{$Row.4}</td>
            </tr>
            <tr>
                <td style="text-align:right"><strong>Life Expectancy</strong></td>
                <td style="text-align:left">{$Row.5}</td>
            </tr>
        </table>
    </div>
{/foreach}
</div>
```

The event handler code is as follows:

```
if (($part == PagePart::Grid) && ($mode == PageMode::PrintAll)) {
    $result = 'custom_print_list.tpl';
}
```


5.4.3.12 OnCustomDrawRow

This event occurs on rendering a grid row. It is an extremely useful event for conditional formatting such as changing font color, font styles, row background color, cell background color, etc. This event (as well as the [OnExtendedCustomDrawRow](#)^[155] one) is used for conditional formatting. The only difference between these two events is that OnCustomDrawRow has a more understandable parameter list while OnExtendedCustomDrawRow provides more flexible abilities.

Signature:

```
function OnCustomDrawRow ($rowData, &$cellFontColor, &$cellFontSize, &$cellBgColor,
    &$cellItalicAttr, &$cellBoldAttr)
```

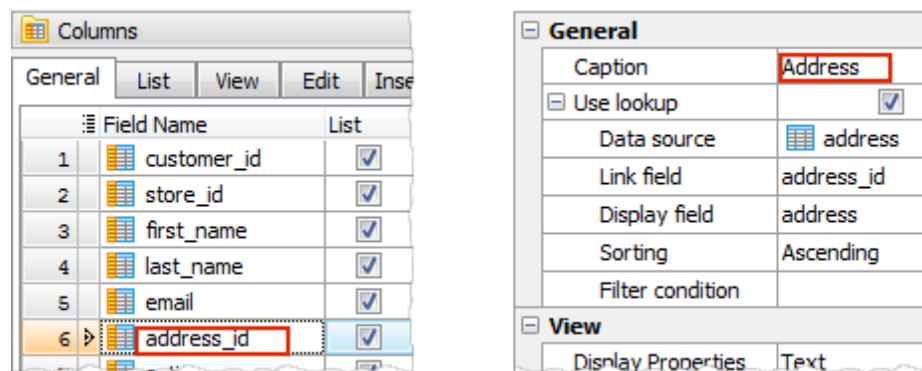
Parameters:

\$rowData	The associative array of values that corresponds to the currently processed row.
\$cellFontColor	The parameter allows to set a font color for selected fields.
\$cellFontSize	The parameter defines a font size of selected data.
\$cellBgColor	The parameter allows to set a background color of fields.
\$cellItalicAttr	Set the parameter to true to represent text in cursive font.
\$cellBoldAttr	Set the parameter to true to use bold font.

NB. \$rowData array keys should contain the real column names in the data source (table, view, or query). Don't use column captions instead!

Example 1:

Suppose we need to create a webpage with list of customers with addresses represented in cursive. This column in the data source is named 'address_id' and the column's caption is "Address".



To define the font attribute, use the following code:

```
$cellItalicAttr['address_id'] = true;
```

Example 2:

In the example below we need to display winning team score in red and losing team score in black; moreover, both scores should be in bold and displayed by a 16pt font.

```
$cellFontSize['home_team_score'] = '16pt';
$cellBoldAttr['home_team_score'] = true;
```



```

$cellFontSize['away_team_score'] = '16pt';
$cellBoldAttr['away_team_score'] = true;

if ($rowData['home_team_score'] > $rowData['away_team_score'])
    $cellFontColor['home_team_score'] = '#F65317';
else
    $cellFontColor['away_team_score'] = '#000000';

```

5.4.3.13 OnAfterInsertRecord

This event occurs when the Insert command is executed, and after the actual insertion.

Signature:

```

function OnAfterInsertRecord ($page, $rowData, $tableName,
    &$success, &$message, &$messageDisplayTime)

```

Parameters:

\$page	An instance of the Page class.
\$rowData	The associative array of values that corresponds to the currently processed row.
\$tableName	The name of processed table.
\$success	Indicates whether the last data manipulation statement was successful or not.
\$message	A message to be displayed to a user. If the statement completed successfully, the value of this parameter is equal to empty string. If the statement failed, the value of this parameter contains the error message that came from the database server.
\$messageDisplayTime	A time interval (in seconds) after which the message will disappear automatically. Default value is 0 (the message will not disappear).

Success messages are displayed in green while error messages are displayed in red.

Example 1:

The following code shows the message about a success of the operation. The message will be hidden in 5 seconds:

```

if ($success) {
    $message = 'Record processed successfully.';
}
else {
    $message = '<p>Something wrong happened. ' .
        '<a class="alert-link" href="mailto:admin@example.com">' .
        'Contact developers</a> for more info.</p>';
}
$messageDisplayTime = 5;

```

Example 2:

The following code logs information about added records into a separate table:

```

if ($success) {
    $userId = $page->GetCurrentUserId();
}

```



```

$currentDateTime = SMDatetime::Now();
$sql =
    "INSERT INTO activity_log (table_name, action, user_id, log_time) " .
    "VALUES ('$tableName', 'INSERT', $userId, '$currentDateTime');";
$page->GetConnection()->ExecSQL(sprintf($sql, $userId, $action, $currentDateTime));
}

```

Example 3:

The following code is used in our [Feature Demo](#) to save genres of a newly added movie to a separate table:

```

if ($success && GetApplication()->IsPOSTValueSet('genres')) {
    $genres = GetApplication()->GetPOSTValue('genres');
    $lastInsertId = $page->GetConnection()->GetLastInsertId();
    foreach ($genres as $genre) {
        $sql = sprintf('INSERT INTO movie_genres VALUES(%d, %d);', $lastInsertId, $genre);
        $page->GetConnection()->ExecSQL($sql);
    }
}

```

See also: [OnAfterUpdateRecord](#)^[176], [OnAfterDeleteRecord](#)^[177], [OnBeforeInsertRecord](#)^[178].

5.4.3.14 OnAfterUpdateRecord

This event occurs when the Update command is executed, and after the actual update.

Signature:

```
function OnAfterUpdateRecord ($page, $oldRowData, $rowData, $tableName,
    &$success, &$message, &$messageDisplayTime)
```

Parameters:

\$page	An instance of the Page ^[304] class.
\$oldRowData	The associative array of old (previous) values of the currently processed row.
\$rowData	The associative array of values that corresponds to the currently processed row.
\$tableName	The name of processed table.
\$success	Indicates whether the last data manipulation statement was successful or not.
\$message	A message to be displayed to a user. If the statement completed successfully, the value of this parameter is equal to empty string. If the statement failed, the value of this parameter contains the error message that came from the database server.
\$messageDisplayTime	A time interval (in seconds) after which the message will disappear automatically. Default value is 0 (the message will not disappear).

Success messages are displayed in green while error messages are displayed in red.

Example 1:

```

if ($success) {
    $message = 'Record processed successfully.';
}

```



```

}
else {
    $message = '<p>Something wrong happened. ' .
        '<a class="alert-link" href="mailto:admin@example.com">' .
        'Contact developers</a> for more info.</p>';
}

```

Example 2:

The following code logs information about modified records into a separate table:

```

if ($success) {
    // Check if record data was modified
    $dataModified =
        $oldRowData['first_name'] != $rowData['first_name'] ||
        $oldRowData['last_name'] != $rowData['last_name'] ||
        $oldRowData['email'] != $rowData['email'];

    if ($dataModified) {
        $userId = $page->GetCurrentUserId();
        $currentDateTime = SMDatetime::Now();
        $sql =
            "INSERT INTO activity_log (table_name, action, user_id, log_time) " .
            "VALUES ('$tableName', 'UPDATE', $userId, '$currentDateTime');";
        $page->GetConnection()->ExecSQL($sql);
    }
}

```

Example 3:

The following code is used in our [Feature Demo](#) to modify genres of the updated movie to a separate table:

```

if ($success && GetApplication()->IsPOSTValueSet('genres')) {
    $sql = sprintf('DELETE FROM movie_genres WHERE movie_id = %d;', $rowData['id']);
    $page->GetConnection()->ExecSQL($sql);
    $genres = GetApplication()->GetPOSTValue('genres');
    foreach ($genres as $genre) {
        $sql = sprintf('INSERT INTO movie_genres VALUES(%d, %d);', $rowData['id'], $genre);
        $page->GetConnection()->ExecSQL($sql);
    }
}

```

See also: [OnAfterDeleteRecord](#)^[177], [OnAfterInsertRecord](#)^[178], [OnBeforeUpdateRecord](#)^[179].

5.4.3.15 OnAfterDeleteRecord

This event occurs when the Delete command is executed, and after the actual deletion.

Signature:

```

function OnAfterDeleteRecord ($page, $rowData, $tableName,
    &$success, &$message, &$messageDisplayTime)

```

Parameters:

\$page	An instance of the Page ^[304] class.
\$rowData	The associative array of values that corresponds to the currently processed row.
\$tableName	The name of processed table.

\$success	Indicates whether the last data manipulation statement was successful or not.
\$message	A message to be displayed to a user. If the statement completed successfully, the value of this parameter is equal to empty string. If the statement failed, the value of this parameter contains the error message that came from the database server.
\$messageDisplayTime	A time interval (in seconds) after which the message will disappear automatically. Default value is 0 (the message will not disappear).

Success messages are displayed in green while error messages are displayed in red.

Example 1:

The following code shows the message about a success of the operation. The message will be hidden in 5 seconds:

```
if ($success) {
    $message = 'Record processed successfully.';
}
else {
    $message = '<p>Something wrong happened. ' .
        '<a class="alert-link" href="mailto:admin@example.com">' .
        'Contact developers</a> for more info.</p>';
}
$messageDisplayTime = 5;
```

Example 2:

The following code logs information about deleted records in a separate table:

```
if ($success) {
    $userId = $page->GetCurrentUserId();
    $currentDateTime = SMDatetime::Now();
    $sql =
        "INSERT INTO activity_log (table_name, action, user_id, log_time) " .
        "VALUES ('$tableName', 'DELETE', $userId, '$currentDateTime');";
    $page->GetConnection()->ExecSQL(sprintf($sql, $userId, $action, $currentDateTime));
}
```

See also: [OnAfterUpdateRecord](#)^[176], [OnAfterInsertRecord](#)^[175], [OnBeforeDeleteRecord](#)^[180].

5.4.3.16 OnBeforeInsertRecord

This event occurs when the Insert command is executed, and before the actual insertion.

Signature:

```
function OnBeforeInsertRecord ($page, &$amp;rowData, &$amp;cancel, &$amp;message,
    &$amp;messageDisplayTime, $tableName)
```

Parameters:

\$page	An instance of the Page ^[304] class declared in <i>components/page.php</i> .
\$rowData	The associative array of values that corresponds to the

	currently processed row.
\$cancel	The value indicating whether the operation should be canceled.
\$message	The message string that is displayed after the operation is completed (or canceled).
\$messageDisplayTime	A time interval (in seconds) after which the message will disappear automatically. Default value is 0 (the message will not disappear).
\$tableName	The name of processed table.

Example 1:

```
if (!(allowDataEditing())) {
    $cancel = true;
    $message = 'The application is running in read-only mode.';
}
```

Example 2:

The following code allows to assign values to some fields (usually these fields are not included in an insert form) before inserting of a record:

```
$rowData['line_total'] = $rowData['quantity'] * $rowData['unit_price'];
$rowData['create_datetime'] = SMDatetime::Now();
$rowData['insert_by'] = $page->GetCurrentUserId();
```

See also: [OnBeforeUpdateRecord](#)^[179], [OnBeforeDeleteRecord](#)^[180], [OnAfterInsertRecord](#)^[178].

5.4.3.17 OnBeforeUpdateRecord

This event occurs when the Update command is executed, and before the actual update.

Signature:

```
function OnBeforeUpdateRecord ($page, $oldRowData, &$rowData, &$cancel, $message,
    &$messageDisplayTime, $tableName)
```

Parameters:

\$page	An instance of the Page ^[304] class declared in <i>components/page.php</i> .
\$oldRowData	The associative array of old (previous) values of the currently processed row.
\$rowData	The associative array of values that corresponds to the currently processed row.
\$cancel	The value indicating whether the operation should be canceled.
\$message	The message string that is displayed after the operation is completed (or canceled).
\$messageDisplayTime	A time interval (in seconds) after which the message will disappear automatically. Default value is 0 (the message will not disappear).

\$tableName	The name of processed table.
-------------	------------------------------

Example 1:

```
if (!(allowDataEditing())) {
    $cancel = true;
    $message = 'The application is running in read-only mode.';
}
```

Example 2:

The following code allows to assign values to some fields (usually these fields are not included in an edit form) before updating of a record:

```
$rowData['line_total'] = $rowData['quantity'] * $rowData['unit_price'];
$rowData['update_datetime'] = SMDatetime::Now();
$rowData['last_updated_by'] = $page->GetCurrentUserId();
```

See also: [OnBeforeDeleteRecord](#)^[180], [OnBeforeInsertRecord](#)^[178], [OnAfterUpdateRecord](#)^[176].

5.4.3.18 OnBeforeDeleteRecord

This event occurs when the Delete command is executed, and before the actual deletion.

Signature:

```
function OnBeforeDeleteRecord ($page, &$rowData, &$cancel, $message,
    &$messageDisplayTime, $tableName)
```

Parameters:

\$page	An instance of the Page ^[304] class declared in <i>components/page.php</i> .
\$rowData	The associative array of values that corresponds to the currently processed row.
\$cancel	The value indicating whether the operation should be canceled.
\$message	The message string that is displayed after the operation is completed (or canceled).
\$messageDisplayTime	A time interval (in seconds) after which the message will disappear automatically. Default value is 0 (the message will not disappear).
\$tableName	The name of processed table.

Example 1:

```
if (!(allowDataEditing())) {
    $cancel = true;
    $message = 'The application is running in read-only mode.';
}
```

See also: [OnBeforeUpdateRecord](#)^[179], [OnBeforeInsertRecord](#)^[178], [OnAfterDeleteRecord](#)^[177].

5.4.3.19 OnGetFieldValue

This event occurs on displaying a value of a field. It allows you to replace the actual value of a field to your own.

Signature:

```
function OnGetFieldValue ($fieldName, &$amp;value, $tableName)
```

Parameters:

\$fieldName	The name of currently processed field.
\$value	The actual value of the field. Can be changed to another value.
\$tableName	The name of currently processed table.

Example

This example shows how this event can be used for implementing a very simple transparent encryption based on the [str_rot13](#) PHP function.

```
if ($fieldName == 'private_data') {
    $value = str_rot13($value);
}
```

Since the call of this function for an encrypted expression returns the original string, we can define [OnBeforeInsertRecord](#)^[178] and [OnBeforeUpdateRecord](#)^[179] event handlers as follows:

```
$rowData['private_data'] = str_rot13($rowData['private_data']);
```

This is just an example. We would recommend that you use more cryptographically strong algorithms to encrypt your important data.

5.4.3.20 OnGetCustomExportOptions

This event fires on exporting data from a database to a file. It allows you to customize some advanced export settings.

Signature:

```
function OnGetCustomExportOptions ($Page page, $exportType, $rowData, &$amp;options)
```

Parameters:

\$page	An instance of the Page ^[304] class.
\$exportType	The type of the result file. Possible values are "csv", "xls", "doc", "pdf", "xml".
\$rowData	An associative array of values that corresponds to the currently processed row (for exporting a single row). NULL if a record list is exported.
\$options	An associative array of options passed to the export engine (see below).

The list of supported options is as follows:

All export types

filename	The name of the output file.
----------	------------------------------

Export to PDF

orientation	Page orientation. Possible values are "P" (portrait orientation) or "L" (landscape orientation). Default value is "P".
size	Size of the page. Possible values are A0 - A10, B0 - B10, C0 - C10, 4A0, 2A0, RA0 - RA4, SRA0 - SRA4, Letter, Legal, Executive, Folio, Demy, Royal, A (type A paperback 111x178mm), B (type B paperback 128x198mm). Default value is A4.
margin-left	Distance in mm from the left side of page to the left side of text.
margin-right	Distance in mm from the right side of page to the right side of text.
margin-top	Distance in mm from top of page to start of text (ignoring any headers).
margin-bottom	Distance in mm from bottom of page to bottom of text (ignoring any footers).
margin-header	Distance in mm from top of page to start of header.
margin-footer	Distance in mm from bottom of page to bottom of footer.
header	<p>In any page header, '{PAGENO}' or '{DATE j-m-Y}' can be used - which will be replaced by the page number or current date. j-m-Y can be replaced by any of the valid formats used in the php date() function.</p> <p>The page number is the calculated number, taking regard of any resetting of numbering during the document.</p>
html-header	Standard HTML code to define the header of pdf file. It can only be defined outside HTML block tags (except <body>).
footer	<p>In any page footer, '{PAGENO}' or '{DATE j-m-Y}' can be used - which will be replaced by the page number or current date. j-m-Y can be replaced by any of the valid formats used in the php date() function.</p> <p>The page number is the calculated number, taking regard of any resetting of numbering during the document.</p>
html-footer	Standard HTML code to specify the footer of pdf file. It can only be defined outside HTML block tags (except <body>).

Export to Excel

file-format	The format of the output file. Possible values are "xlsx" and "xls". Default value is "xlsx".
engine	The engine to be used on data export. Possible values are "template" (template-based export) and "phpexcel" (to create native .xls files). Default value is "phpexcel". If you customize templates for export to Excel then you should use the "template" engine.

Example

This example shows how to customize the result file name according to row values. If a record list is exported, the file name is set to list.pdf(list.xml, list.xls, etc).

```
$options['filename'] = ($rowData ? $rowData['id'] : 'list') . '.' . $exportType;
```

Example

This example shows how to customize header, footer and margins of pdf file.

```
if ($exportType == 'pdf') {
    $options['header'] = '{DATE Y-m-d}';
    $options['footer'] = '{PAGE NO}';
    $options['margin-top'] = 12;
}
```

5.4.3.21 OnPrepareFilterBuilder

This event allows you to setup initial condition for the Filter Builder tool.

Signature:

```
function OnPrepareFilterBuilder(FilterBuilder $filterBuilder,
    FixedKeysArray $columns)
```

Parameters:

\$filterBuilder	An instance of the FilterBuilder class.
\$columns	An associative array of columns available in the tool.

Example

This example shows how to setup initial value for Filter Builder (available live in the [Feature Demo](#)).

```
$rootGroup = new FilterGroup(FilterGroupOperator::OPERATOR_AND,
    array(
        new FilterCondition($columns['rating'],
            FilterConditionOperator::IS_BETWEEN, array(5, 7)),
        new FilterCondition($columns['genre_id'],
            FilterConditionOperator::EQUALS, array(14), array('Fantasy'))
    )
);
$filterBuilder->setFilterComponent($rootGroup);
```

5.4.3.22 OnPrepareColumnFilter

This event allows you to customize column-specific filters.

Signature:

```
function OnPrepareColumnFilter ($columnFilter)
```

Parameters:

\$columnFilter	An instance of the ColumnFilter class.
----------------	--

All examples below can be seen in action in the [Feature Demo](#).

Example 1

This example shows how to setup a column filter as a list of custom values.

```
$columnFilter->setOptionsFor(
    'rating',
    array(
        "Less than 3" => FilterCondition::lessThan(3),
        "4-5" => FilterCondition::between(4, 5),
        "6-7" => FilterCondition::between(6, 7),
        "7-8" => FilterCondition::between(7, 8),
        "Higher than 8" => FilterCondition::greaterThan(8)
    ),
    false // no default values
);
```

Example 2

This example shows how to group custom values.

```
$columnFilter->setOptionsFor(
    'runtime',
    array(
        'Shorter than 90 min' => FilterCondition::lessThan(90),
        '90 - 180 min' => FilterGroup::orX(
            array(
                '90-120 min' => FilterCondition::between(90, 120),
                '120-150 min' => FilterCondition::between(120, 150),
                '150-180 min' => FilterCondition::between(150, 180)
            )
        ),
        'Longer than 180 min' => FilterCondition::greaterThan(180)
    ),
    false // no default values
);
```

Example 3

This example shows how to add custom values to default ones.

```
$columnFilter->setOptionsFor('release_date',
    array(
        '2010s' => FilterCondition::between('2010-01-01', '2019-12-31'),
        '2000s' => FilterCondition::between('2000-01-01', '2009-12-31')
    ),
    true // add default values (this parameter can be omitted)
);
```

5.4.3.23 OnGetCustomFormLayout

This event allows you to customize the layout for View, Edit, and Insert forms ([Live Demo](#)). [Video Tutorial](#) demonstrates an example of View, Edit and Insert forms adjustment.

Signature:

```
function OnGetCustomFormLayout($mode, FixedKeysArray $columns,
                               FormLayout $layout)
```


Parameters

\$mode	The form mode. Possible values are "insert", "inline_insert", "edit", "inline_edit", "multi_edit", "view" and "inline_view".
\$columns	The associative array of columns displayed in the form.
\$layout	An instance of the FormLayout class.

Layout structure

The layout has the following hierarchical structure: layout -> tabs -> groups -> rows -> columns. This means you can add tabs to the layout, groups to tabs and layout, rows to groups, and columns to rows as shown below.

The screenshot displays a form editor interface for a 'Customer' record. The form is titled 'Edit' and has two tabs: 'Personal Info' and 'Activity'. The 'Personal Info' tab is active and contains two groups of fields. The first group contains 'Store' (a dropdown menu with '2' selected) and 'Address Id' (a dropdown menu with '1519 Santiago de los Caballeros Loop' selected). The second group contains 'Active' (a checkbox), 'Create Date' (a date field with '2006-02-14' and a calendar icon), and 'Last Update' (a date field with '2020-03-24' and a calendar icon). The form is annotated with colored lines and labels to illustrate its hierarchical structure: a blue line labeled 'Layout' points to the entire form; red lines labeled 'Tabs' point to the 'Personal Info' and 'Activity' tabs; magenta lines labeled 'Groups' point to the two groups of fields; green lines labeled 'Rows' point to the individual fields within each group; and red lines labeled 'Columns' point to the individual fields within each row. A legend at the bottom left indicates that a red line represents a 'Required field'.

```
if ($mode=='insert' or $mode=='edit') {
\\ this customization is used for Insert and Edit forms
$layout->setMode(FormLayoutMode::VERTICAL);
\\ labels are placed on the top of the editors
$layout->enableTabs(FormTabsStyle::TABS);
\\ these forms consist of tabs

$personalInfoTab = $layout->addTab('Personal Info');
```



```

\\ the first tab is 'Personal Info'
$personalInfoTab->setMode(FormLayoutMode::HORIZONTAL);
\\ this tab's control labels are placed on the left of the editors
$commonInfoGroup = $personalInfoTab->addGroup('Common Information');
\\ there is a 'Common Information' group in this tab
$commonInfoGroup->addRow()
\\ the first row of this group contains two columns:
    ->addCol($columns['first_name'], 4, 2)
    \\ 'first_name' and 'last_name'
    ->addCol($columns['last_name'], 4, 2);
$commonInfoGroup->addRow()->addCol($columns['email'], 10, 2);
\\ the second row contains 'email'

$activityTab = $layout->addTab('Activity');
\\ the second tab is 'Activity'
$addressGroup = $activityTab->addGroup(null, 6);
\\ 50% (6 of 12) of this tab's space is $addressGroup, a group without caption
$addressGroup->addRow()->addCol($columns['store_id'], 12);
\\ this group contains of two rows: 'store_id'
$addressGroup->addRow()->addCol($columns['address_id'], 12);
\\ and 'address_id' with width 100% (12 of 12)
$activityGroup = $activityTab->addGroup(null, 6);
\\ the next 50% of this tab's space is $activityGroup, a group without caption
$activityGroup->addRow()->addCol($columns['active'], 12);
\\ the first row of this group is 'active'
$activityGroup->addRow()
\\ the second contains two columns:
    ->addCol($columns['create_date'], 6)
    \\ 'create_date' and 'last_update'
    ->addCol($columns['last_update'], 6);
}

```

Forms to be customized

By default the form customization will be used in the Edit, Multi Edit, Insert, View, and also on inline inserting and editing. To adjust a concrete form (i.e Edit), use the following condition:

```

if ($mode=='edit') {
    //Your customization code
}

```

Layout mode (VERTICAL, HORIZONTAL)

By default all forms are horizontal i.e. the control label is placed on the left of the editor (for vertical forms the label is placed on the top of the editor). You can switch the form mode with the *setMode* method. This method works for layout, a tab, a group and a row.

```

$layout->setMode(FormLayoutMode::VERTICAL);
\\ the label is placed on the top of the editor
$layout->enableTabs(FormTabsStyle::TABS);
$personalInfoTab = $layout->addTab('Personal Info');
    $personalInfoTab->setMode(FormLayoutMode::HORIZONTAL);
    \\ the layout consists of tabs and control labels of 'Personal Info'
    \\ tab are placed on the left of the editors

```

Adding a tab

To add a new tab to the layout, first enable tabs on a form with the *enableTabs* command. The picture below illustrates the tabs appearance in the [default_color_scheme](#)

²⁴¹ depending of the `%TABS_STYLE%` value.

```
$layout->enableTabs(%TABS_STYLE%);
```

Available values of `%TABS_STYLE%`:

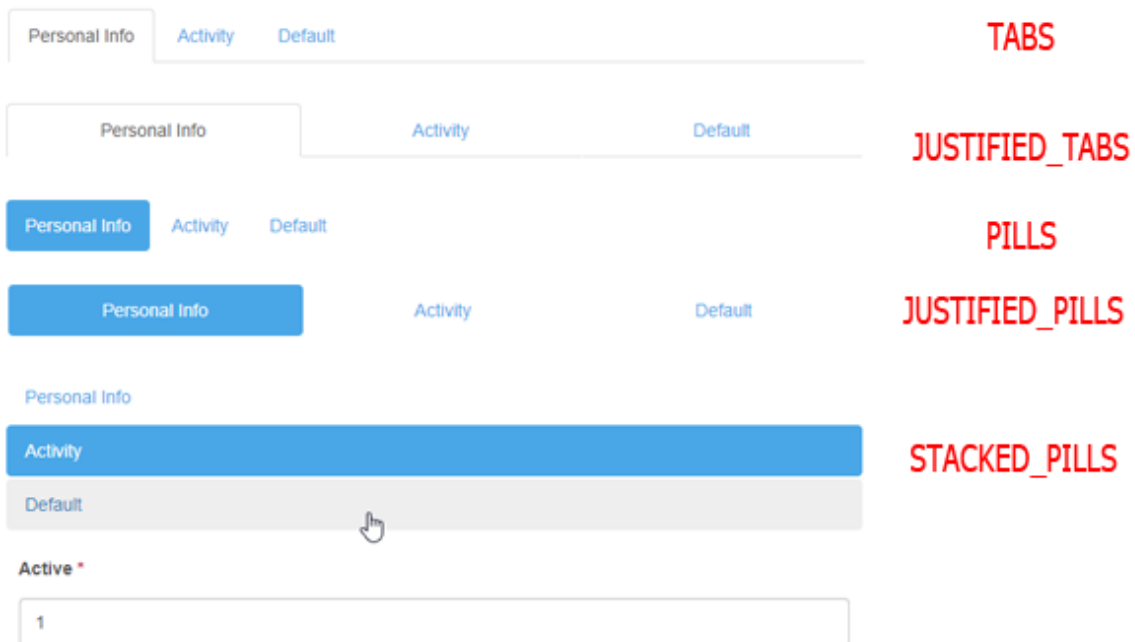
`FormTabsStyle::TABS`

`FormTabsStyle::JUSTIFIED_TABS`

`FormTabsStyle::PILLS`

`FormTabsStyle::JUSTIFIED_PILLS`

`FormTabsStyle::STACKED_PILLS`



To add a new tab use the following command:

```
$personalInfoTab = $layout->addTab('Personal Info');
```

Adding a group

To add a new group to the layout, use the *addGroup* method of the *FormLayout* class:

```
public function addGroup($name = null, $width = 12, $customAttributes = '', $inlineStyles = '');
```

This function returns an instance of the *FormLayoutGroup* class that can be later used to add rows to the new group (see below). Width is provided in relative units, possible values are integers from 1 to 12.

Example

The following code adds an 'addressGroup' group to the layout. This group has no caption and attributes, takes 50% of the tab's space, and its text color is green.

```
$addressGroup = $activityTab->addGroup(null, 6, '', 'color: green;');
```

The following methods can be used for groups:

<code>public function setVisible(\$value);</code>	Use it to hide/show groups. Example: <code>\$group->setVisible(false);</code>
<code>public function setCustomAttributes(\$customAttributes);</code>	Specifies HTML attributes of the group. Example: <code>\$group->setCustomAttributes('data-group="test"');</code>
<code>public function setInlineStyles(\$inlineStyles);</code>	Use it to set the group style. Example: <code>\$group->setInlineStyles('color: green;');</code>
<code>public function setMode(\$mode);</code>	Defines where the control label is placed: on the left of the editor (for vertical mode) or on the top of the editor (for horizontal). Example <code>\$addressGroup->setMode(FormLayoutMode::HORIZONTAL);</code>

Adding rows to a group

To add a new row to a group, use the `addRow` method of the `FormLayoutGroup` class (see above):

```
function addRow();
```

This function has no parameters and returns an instance of the `FormLayoutRow` class that can be later used to add controls to the new row (see below).

Adding controls to a row

To add a new control to a row, use the `addCol` method of the `FormLayoutRow` class (see above):

```
function addCol($column, $inputWidth = null, $labelWidth = null);
```

This function also returns an instance of the `FormLayoutRow` class to allow method chaining. Here `$column` is an element of the `$columns` array and `$inputWidth` and `$labelWidth` are relative widths of the editor and its label accordingly (`$labelWidth` can be used only for horizontal forms).

Examples

All the examples below can be seen live in our [Feature Demo](#).

Example 1

This example shows how to create a simple group with 2 rows and 2 controls in each row (all controls have equal width):

```
$displayGroup = $layout->addGroup('Display');
$displayGroup->addRow()
    ->addCol($columns['display_type'], 6)
    ->addCol($columns['display_size'], 6);
$displayGroup->addRow()
    ->addCol($columns['display_resolution_x'], 6)
```



```
->addCol($columns['display_resolution_y'], 6);
```

Example 2

This example shows how to create a group and place 3 editors in the same row:

```
$storageGroup = $layout->addGroup('Storage', 12);
$storageGroup->addRow()
    ->addCol($columns['storage_min'], 4)
    ->addCol($columns['storage_max'], 4)
    ->addCol($columns['storage_external'], 4);
```

Example 3

This example shows how it is possible to place 2 groups in a row:

```
$hardwareGroup = $layout->addGroup('Hardware', 6);
$hardwareGroup->addRow()->addCol($columns['chipset'], 12);
$hardwareGroup->addRow()->addCol($columns['cpu'], 12);
$hardwareGroup->addRow()->addCol($columns['gpu'], 12);

$softwareGroup = $layout->addGroup('Software', 6);
$softwareGroup->addRow()->addCol($columns['os_basic'], 12);
$softwareGroup->addRow()->addCol($columns['os_upgradable'], 12);
$softwareGroup->addRow()->addCol($columns['web_browser'], 12);
```

5.4.3.24 OnGetCustomColumnGroup

This event allows you to setup multi-row grid header. [Live example](#).

Signature:

```
function OnGetCustomColumnGroup(FixedKeysArray $columns,
                                ViewColumnGroup $columnGroup)
```

Parameters:

\$columns	The associative array of column headers.
\$columnGroup	The default (root) group. By default all column headers are placed in this group.

Description

You can organize columns in logical groups and display them using multi-row header representation. A column group is visually represented by a header displayed above the headers of the columns it combines. Each group can contain data columns as well as other groups.

Examples

All the examples below can be seen live in our [Feature Demo](#).

Example 1

This example shows how to define an additional header for 3 columns:

```
$columnGroup->add(new ViewColumnGroup('Common info',
    array(
        $columns['id'],
        $columns['title'],
        $columns['release_date']
    )
));
```


Example 2

This example shows how to define nested column groups:

```
$columnGroup->add(new ViewColumnGroup('Additional info',
    array(
        new ViewColumnGroup('Texts',
            array(
                $columns['genre_id'],
                $columns['original_title'],
                $columns['original_language_id']
            )
        ),
        new ViewColumnGroup('Numbers',
            array(
                $columns['runtime'],
                $columns['rating']
            )
        )
    )
);
```

5.4.3.25 OnCustomCompareValues

This event allows you to define your own function for value comparison. [Live example](#) and [video tutorial](#).

Signature:

```
function OnCustomCompareValues($columnName, $valueA, $valueB, &$amp;result)
```

Parameters:

\$columnName	The name of currently processed column.
\$valueA	The first value to compare.
\$valueB	The second value to compare.
\$result	The result of the comparison. Should be set to 0 if values are equal and to a non-zero value otherwise.

Example

This example shows how to define an algorithm that considers two values to be equal if their difference is less than or equal to 0.5.

```
if (in_array($columnName, array('height', 'length', 'width'))) {
    $result = abs($valueA - $valueB) <= 0.5;
}
```

The example can be seen in action in our [Feature Demo](#).

5.4.3.26 OnFileUpload

This event allows you to control file uploading. It fires on uploading a file with [Upload file to folder](#)^[103] and [Upload image to folder](#)^[104] editors.

Signature:

```
function OnFileUpload($fieldName, &$amp;result, &$amp;accept, $originalFileName,
    $originalFileExtension, $fileSize, $tempFileName)
```


Parameters:

\$fieldName	The name of currently processed column.
\$rowData	The associative array of values that corresponds to the currently processed row.
\$result	The actual file name (with full path). Change the value of this parameter to customize the filename.
\$accept	Indicates whether the file uploading is allowed. Default value is true. Set \$accept to false to cancel file uploading.
\$originalFileName	The original file name (including extension)
\$originalFileExtension	The original file extension
\$fileSize	The file size
\$tempFileName	The temporary filename of the file in which the uploaded file was stored on the server.

Example

This example shows how to add current date and time to the uploaded filename.

```
if ($fieldName === 'filename') {
    $newFileName = date('Y-m-d_H-i-s') . "_" . $originalFileName;
    $result = str_replace($originalFileName, $newFileName, $result);
}
```

5.4.3.27 OnGetCustomPagePermissions

This event allows you to customize page-level permissions.

Signature:

```
function OnGetCustomPagePermissions($pageName, $userId, $userName, $connection, &$permissions)
```

Parameters:

\$pageName	The name of the page
\$userId	The id of the current user
\$userName	The name of the current user
\$connection	An instance of EngConnection ^[318] class
\$permissions	Permissions to be applied to the page. An instance of the PermissionSet ^[320] class.

Example

This example shows how to implement a simple role-based permission model. Our goal is to ensure that only site admins and members of the Sales role can add, remove and edit records displayed on this page.

```
// do not apply these rules for site admins
if (!GetApplication()->HasAdminGrantForCurrentUser()) {
    // retrieving all user roles
    $sql =
        "SELECT r.role_name " .
        "FROM phpgen_user_roles ur " .
```



```

        "INNER JOIN phpgen_roles r ON r.id = ur.role_id " .
        "WHERE ur.user_id = %d";
$result = $connection->fetchAll(sprintf($sql, $userId));

// iterating through retrieved roles
if (!empty($result)) {
    foreach ($result as $row) {
        // is current user a member of the Sales role?
        if ($row['role_name'] === 'Sales') {
            // if yes, allow all actions.
            // otherwise default permissions for this page will be applied
            $permissions->setGrants(true, true, true, true);
            break;
        }
    }
};
}

```

See also: [OnGetCustomRecordPermissions](#)^[192].

5.4.3.28 OnGetCustomRecordPermissions

This event allows you to customize [record-level permissions](#)^[268].

Signature:

```
function OnGetCustomRecordPermissions($page, &$usingCondition, $rowData,
    &$allowEdit, &$allowDelete, &$mergeWithDefault, &$handled)
```

Parameters:

\$page	An instance of the Page ^[304] class.
\$usingCondition	Any logical SQL expression. Rows for which the expression returns true will be visible
\$rowData	The associative array of values that corresponds currently processed row
\$allowEdit	If true, the user can edit values of the currently processed row.
\$allowDelete	If true, the user can delete the currently processed row.
\$mergeWithDefault	Indicates whether custom permissions should be merged with default ones ^[268] (if any). Default value is true.
\$handled	A parameter to indicate whether the new permissions should be applied. Set \$handled to true to apply the changes.

Example

Assume we have a small company with several sales departments. All users of our application are sales managers, which work in one of these departments. Each such user can work as an ordinary manager or as a head manager of the department. Our challenge is to grant privileges in the following way:

- Ordinary managers must have full access to their own sales records except completed ones. They should have no access to the sales made by other managers.
- Head managers must have full access to all sales records of the department. They should have no access to sales of other departments.

To implement the scenario above, the following code can be used:

```
// do not apply these rules for site admins
if (GetApplication()->IsLoggedInAsAdmin()) {
    return;
}

// retrieving the ID of the current user
$userId = $page->GetCurrentUserId();

// retrieving the ID of sales department and the status of the current user
$sql = "SELECT sales_department_id, is_head_manager " .
        "FROM phpgen_users WHERE user_id = $userId";
$result = $page->GetConnection()->fetchAll($sql);

if (empty($result))
    return;

$salesDepartmentId = $result[0]['sales_department_id'];
$isHeadManager = (boolean) $result[0]['is_head_manager'];

// Granting permissions according to the scenario
$allowEdit = $isHeadManager || !$rowData['completed'];
$allowDelete = $isHeadManager || !$rowData['completed'];

// Specifying the condition to show only necessary records
if ($isHeadManager) {
    $sql = 'manager_id IN '.
            '(SELECT user_id FROM phpgen_users WHERE sales_department_id = %d)';
    $usingCondition = sprintf($sql, $salesDepartmentId);
} else {
    $usingCondition = sprintf('manager_id = %d', $userId);
}

// apply granted permissions
$handled = true;

// Do not merge the new record permissions with default ones (true by default).
// We have to add this line, otherwise head managers will not be able to see
// sales made by other managers of the department.
$mergeWithDefault = false;
```

See also: [OnGetCustomPagePermissions](#)¹⁹¹.

5.4.3.29 OnCustomDefaultValues

This event allows you to provide a custom default value for a column in the Insert form. Use this event if you need to provide a non-trivial default value for a column (for example, retrieve something from the database or calculate something with PHP code). [Live examples](#).

Signature:

```
function OnCustomDefaultValues($page, &$values, &$handled)
```

Parameters:

\$page	An instance of the Page ³⁰⁴ class.
\$values	An associative array of default values

<code>\$handled</code>	A parameter to indicate whether custom default values should be applied. Set <code>\$handled</code> to true to apply the changes.
------------------------	---

Example

This example shows how to provide the `release_date` column with the default value "a week later".

```
$release_date = SMDatetime::now();
$release_date->addDays(7);
$values['release_date'] = $release_date;
$handled = true;
```

5.4.3.30 OnCalculateFields

This event occurs whenever the values of [calculated columns](#)¹⁰⁷ are (re)computed in read-only views (List, Print, Export, etc). [Live examples](#).

Signature:

```
function OnCalculateFields($rowData, &fieldName, &$value)
```

Parameters:

<code>\$rowData</code>	An associative array of values of the currently processed row
<code>\$fieldName</code>	The name of the currently processed calculated column
<code>\$value</code>	The value to be assigned to the column

Example 1:

This code snippet sets the value of the *full_name* column according to the values stored in the *first_name* and *last_name* columns.

```
if ($fieldName == 'full_name') {
    $value = $rowData['first_name'] . ' ' . $rowData['last_name'];
}
```

Example 2:

This code snippet sets the value of the *age* column according to the birthday.

```
if ($fieldName == 'age') {
    $dateOfBirth = new DateTime($rowData['birthday']);
    $dateInterval = $dateOfBirth->diff(new DateTime());
    $value = $dateInterval->format('%y');
}
```

See also:

[OnCalculateControlValues](#)¹⁴⁶.

5.4.3.31 OnGetSelectionFilters

This event allows you to specify a set of pre-defined options for record selection.

Signature:

```
function OnGetSelectionFilters(FixedKeysArray $columns, &$result)
```

Parameters:

<code>\$columns</code>	An associative array of columns available for filtering.
------------------------	--

\$result	An associative array of selection options (caption => filter)
----------	---

Example

This example adds "Americas", "Asia+Africa", and "Europe" selection options. Clicking an option selects the corresponding countries.

```
$result = array(
    'Americas' => new FilterCondition(
        $columns['continent'],
        FilterConditionOperator::ENDS_WITH,
        array('America')
    ),
    'Asia+Africa' => new FilterGroup(FilterGroupOperator::OPERATOR_OR,
        array(
            new FilterCondition($columns['continent'],
                FilterConditionOperator::EQUALS, array('Asia')),
            new FilterCondition($columns['continent'],
                FilterConditionOperator::EQUALS, array('Africa'))
        )
    ),
    'Europe' => new FilterCondition(
        $columns['continent'],
        FilterConditionOperator::EQUALS,
        array('Europe')
    ),
);
```

5.4.4 Using Variables

PostgreSQL PHP Generator supports:

- some environment variables (such as `CURRENT_USER_ID`, `CURRENT_USER_NAME`, `UNIQUE_ID`) in both [page](#)^[147] and [application](#)^[117] level server-side events.
- custom environment variables to be used in any server-side event of the concrete page (specified within the [OnAddEnvironmentVariables](#)^[148] event handler) and in any server-side event of any application page (specified within the global [OnAddEnvironmentVariables](#)^[119] event handler). Such variables can also be used in lookup filter criteria, on data filtering, and as default values.

Retrieving the list of available variables

To obtain a complete list of supported variables, turn ON the "Show environment variables" option in the [Project Options](#)^[226] dialog, re-generate the code and then open any of produced web pages in the browser.

Variable name	Value
PAGE_SHORT_CAPTION	Family
PAGE_CAPTION	Family
PAGE_CSV_EXPORT_LINK	public.family.php?operation=ecsv
PAGE_XLS_EXPORT_LINK	public.family.php?operation=eexcel
PAGE_PDF_EXPORT_LINK	public.family.php?operation=epdf
PAGE_XML_EXPORT_LINK	public.family.php?operation=exml
PAGE_WORD_EXPORT_LINK	public.family.php?operation=eword
CURRENT_USER_ID	100
CURRENT_USER_NAME	admin

Accessing variable value

To access the value of a variable, use the [GetEnvVar](#)^[304] method of the [Page](#)^[304] class.

Example 1

The following example demonstrates how to use variables within the [OnBeforeInsertRecord](#)^[178] event handler.

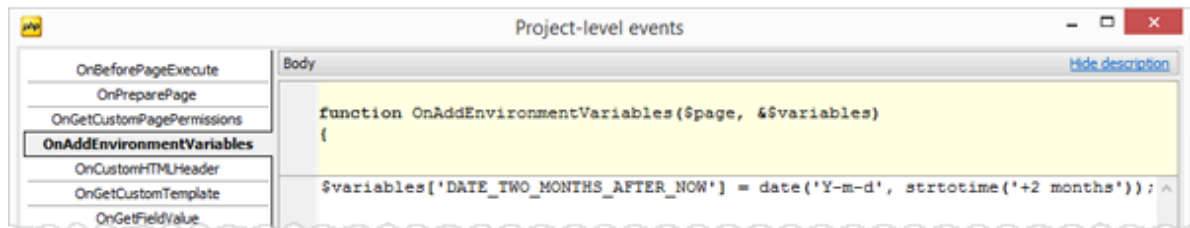
```
$rowData['ip_address'] = $page->GetEnvVar('REMOTE_ADDR');
$username = $page->GetEnvVar('CURRENT_USER_NAME');
if ($username != 'admin')
    $rowData['changed_by'] = $username;
```

Within server-side page-level events it is also possible to use `$this` instead of `$page`.

Example 2

The following code used in the [OnAddEnvironmentVariables](#)^[148] event handler defines a new variable, which can be used in lookup filter criteria, on data filtering, as a default value, and also in any server-side event.

Adding a custom variable



Using this variable as a default value

Edit/Insert	
Edit properties	DateTime ▼ ...
Read only	<input type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Enabled	<input checked="" type="checkbox"/>
Required	<input checked="" type="checkbox"/>
Client validators	...
Default value	%DATE_TWO_MONTHS_AFTER_NOW% ▼

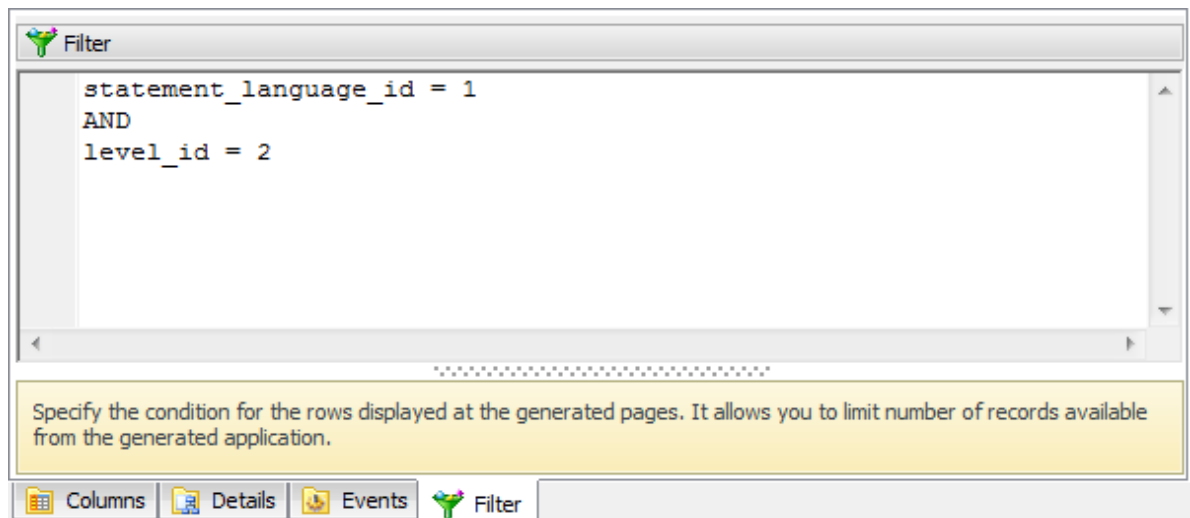
Filter builder

5.5 Filter

To reduce the number of records available at the page, specify the [Filter](#) condition. This condition corresponds to the WHERE clause of the query returned page data (you must not add the WHERE keyword to beginning of the condition).

Example 1

There is a table with test questions of a Web programming course. To display only questions of the 'Beginner' level (`level_id = 2`) concerning to PHP (`statement_language_id = 1`), set the filter condition as follows:



Example 2

You can use subqueries in the filter expression. For example, to display the list of channels that hosted even if a single game, specify the Filter condition as follows:

```
(channel.id > 0) AND channel.id IN
(
  SELECT game.channel_id
  FROM game
)
```


5.6 Charts

PostgreSQL PHP Generator allows you to visualize data on your website with feature-rich, responsive, customizable, and interactive charts.

Live examples

You can find a number of examples in the [Charts](#) group of the Feature Demo. Some more examples can be found at the [Games](#), [Players](#), and [Teams](#) page of the NBA Demo.

Key facts

1. Charts are based on the [Google chart library](#).
2. [Column](#), [Bar](#), [Pie](#), [Line](#), [Area](#), [Geo](#), [Candlestick](#), [Histogram](#), [Bubble](#), [Stepped Area](#), [Timeline](#), [Gantt](#), [Tree Map](#) and [Scatter](#) charts are currently supported.
3. Charts can be placed in rows above the data grid or below the grid.
4. Each row can contain any number of charts.
5. Common properties for each chart can be set up [directly in the software UI](#)^[201].
6. All other properties provided by the library can be customized with the [OnPrepareChart](#)^[201] event.
7. Charts are sensitive to the data grid i.e. if you apply a filter to the grid, charts will change accordingly.

Creating and editing charts

To add a new chart to a webpage or edit an existing one:

- open the webpage [editor](#)^[33];
- go to the Charts tab;
- Click Add... or Edit... to create a new chart or modify an existing chart accordingly;
- define [an SQL query](#)^[199] to retrieve chart data;
- setup [common](#)^[201] and [advanced](#)^[201] options of the chart.

You can hide/show a chart with the [Visible](#) checkbox.

5.6.1 Data Query

This query returns data to be used by the chart. The retrieved data must contain exactly one column to specify labels along the major axis of the chart (**domain column**) and one or more columns to specify series data to render in the chart (**data columns**).

To allow a chart to be sensitive to grid data, refer to the chart's parent page query using the **%source%** placeholder.

Example 1

Assume we have the following table:


```
CREATE TABLE usd_exchange_rate (
    id            int NOT NULL,
    exchange_date date,
    eur_rate      decimal(10,4),
    gbp_rate      decimal(10,4),
    chf_rate      decimal(10,4),
    cad_rate      decimal(10,4),
    aud_rate      decimal(10,4),
    /* Keys */
    PRIMARY KEY (id)
);
```

Our goal is to create a chart that displays the USD course. As data are already grouped, our query is very simple:

```
%source%
```

Then we should specify the *exchange_rate* column as domain column and all **_rate* columns as data columns.

Example 2

Suppose we have two tables *department* and *employee* defined as follows:

```
CREATE TABLE department (
    id      int NOT NULL,
    name    varchar(100) NOT NULL,
    /* Keys */
    PRIMARY KEY (id)
);

CREATE TABLE employee (
    id            int NOT NULL,
    full_name     varchar(100) NOT NULL,
    department_id int NOT NULL,
    salary        int NOT NULL,
    /* Keys */
    PRIMARY KEY (id),
    /* Foreign keys */
    CONSTRAINT fk_emp_department
        FOREIGN KEY (department_id)
        REFERENCES department(id)
);
```

Our goal is to create a chart representing the total salary by departments on the *Department* webpage. The following query returns the appropriate data:

```
SELECT
    d.name as dep_name,
    SUM(e.salary) AS total_salary
FROM
    (%source%) d
    INNER JOIN employee e ON (e.department_id = d.id)
GROUP BY
    d.name
order by 2 desc
```

Then we should specify *dep_name* as the domain column and *total_salary* as a data

column.

5.6.2 Common options

The second step of the chart editor allows you to specify common options of the chart. Other settings can be specified in the [OnPrepareChart](#)^[201] event handler.

Type

The type for your chart. Allowed values are [Column](#), [Bar](#), [Pie](#), [Line](#), [Area](#), [Geo](#), [Candlestick](#), [Histogram](#), [Bubble](#), [Stepped Area](#), [Timeline](#), [Gantt](#), [Tree Map](#) and [Scatter](#).

Id

The unique identifier of the chart. Corresponds to the value of the id attribute of the web element. Can contain letters, numbers, underscore, and hyphens.

Title

The chart title (to be displayed above the chart).

Position

Select whether the chart is placed above or below the data grid.

Height

The height of the chart in pixels.

Width

The width of the chart in columns (from 1 to 12).

Domain column

The column that specifies labels along the major axis of the chart. The Format property is suitable for [date](#) and [number](#) columns and should be specified as a pattern (for example, #,###,### or MMM d, yyyy).

Data columns

One or more columns to specify series data to render in the chart.

5.6.3 Advanced options

The [previous topic](#)^[201] learns how to setup common options of a chart. This topic explains how to customize the chart according to your needs. [Live Demo](#).

The key idea is that you can setup absolutely all the options that are available in the Google chart library. All you need is to study the [reference from Google](#) and write a few lines of PHP code to set property values.

To specify the event, use the [Customize](#) button of the Charts tab. The function that is invoked for each chart on the page is defined as follows:

Signature:

```
function OnPrepareChart(Chart $chart)
```

Parameters:

\$chart	An instance of the Chart class
getId()	Returns the identifier of the chart (unique among all charts on a

	page).
setOptions(\$array)	Sets chart options. Here \$array is an associative array that stores all the options you want to apply to the chart.

Example 1:

To set the value of the property propName, use the following syntax:

```
$chart->setOptions(array(
    'propName' => 'Value',
));
```

To set the value of the property objectName.propName, use the following syntax:

```
$chart->setOptions(array(
    'objectName' => array('propName' => 'value'),
));
```

Operating in the same way you can set any property provided by the library.

Example 2:

The following code is used in the NBA demo application to customize the chart on the [Players](#) page:

```
if ($chart->getId() === 'height-by-avg-age') {
    $chart->setOptions(array(
        'height' => 400,
        'theme' => 'maximized',
        'legend' => array('position' => 'bottom'),
        'hAxis' => array('format' => '0'),
    ));
}
```

Example 3:

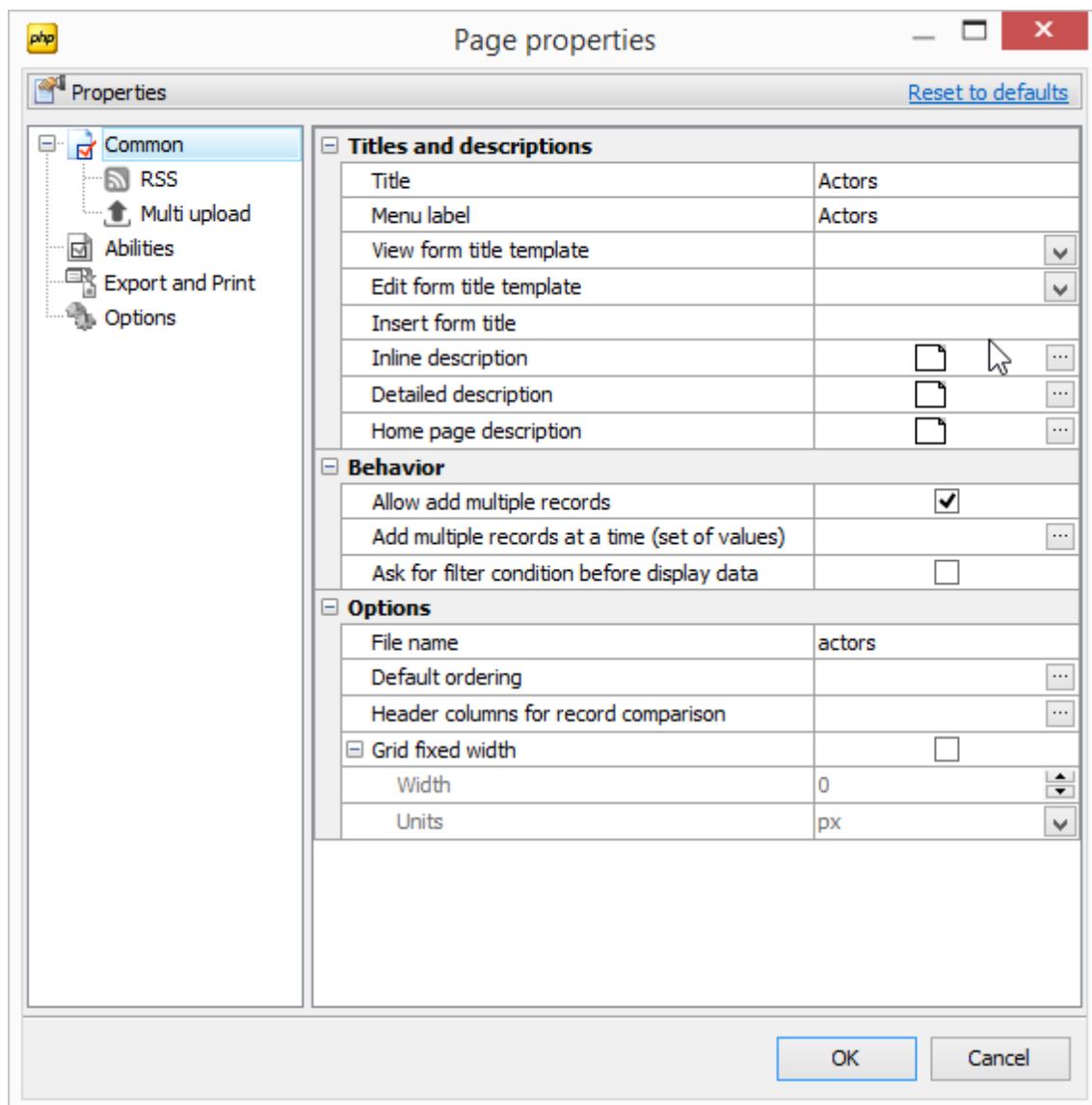
The following code is used in the NBA demo application to customize charts on the [Teams->Home games](#) and [Teams->Away games](#) pages:

```
$options = array(
    'colors' => array('#9ECC3C', '#CC3C3C'),
    'titleTextStyle' => array('fontSize' => 14),
);
if ($chart->getId() === 'overall-win-loss') {
    $chart->setOptions(array_merge($options, array(
        'chartArea' => array('width' => '70%', 'height' => '70%', 'top' => '18%'),
        'legend' => 'none',
    )));
} else if ($chart->getId() === 'win-loss-by-month') {
    $chart->setOptions(array_merge($options, array(
        'vAxis' => array('format' => '0'),
    )));
}
```


5.7 Page Properties

The [Page Properties](#) window allows you to define general properties of the generated page. To invoke this window, use the corresponding button of [Page Editor](#)^[33] or select the necessary page in the [list of pages](#)^[27] to be generated and click the Properties button on the right side.

All properties are grouped in several tabs. By default, all property values defined at the [Abilities](#)^[212], [Export and Print](#)^[213], and [Options](#)^[217] tabs are set in accordance with the corresponding values of [project options](#)^[226]. To change this, uncheck the [Use default options](#) box at the top of the tab and change the values.



- [Common](#)^[204] tab represents page-specific properties i.e. properties that are unique for each page.
- [RSS](#)^[208] allows you to create an RSS feed based on the table.

- [Multi Upload](#)^[208] allows you to upload multiple external files or images at a time.
- [Abilities](#)^[212] tab allows you to specify operations (like View, Edit, Delete, etc) to be available for this page.
- [Export and Print](#)^[213] tab allows you to specify exporting and printing options to be available for this page.
- [Options](#)^[217] tab is to specify page appearance and behavior settings for this page.

5.7.1 Common properties

Here you can setup page-specific settings (i.e. settings that are not inherited from the [Project settings](#)^[226]).

Titles and descriptions

Title

The title of the page.

Menu label

The text to be displayed in menus.

[View form title template](#), [Edit form title template](#), [Insert form title](#)

The titles of View, Edit, and Insert forms accordingly. For View and Edit forms it is possible to use field name tags in the title template, so you can easily change default "Players" to a more user-friendly "Edit Tim Duncan profile".

[Inline description](#)

A text to be displayed at the top of the page. HTML tags are allowed.

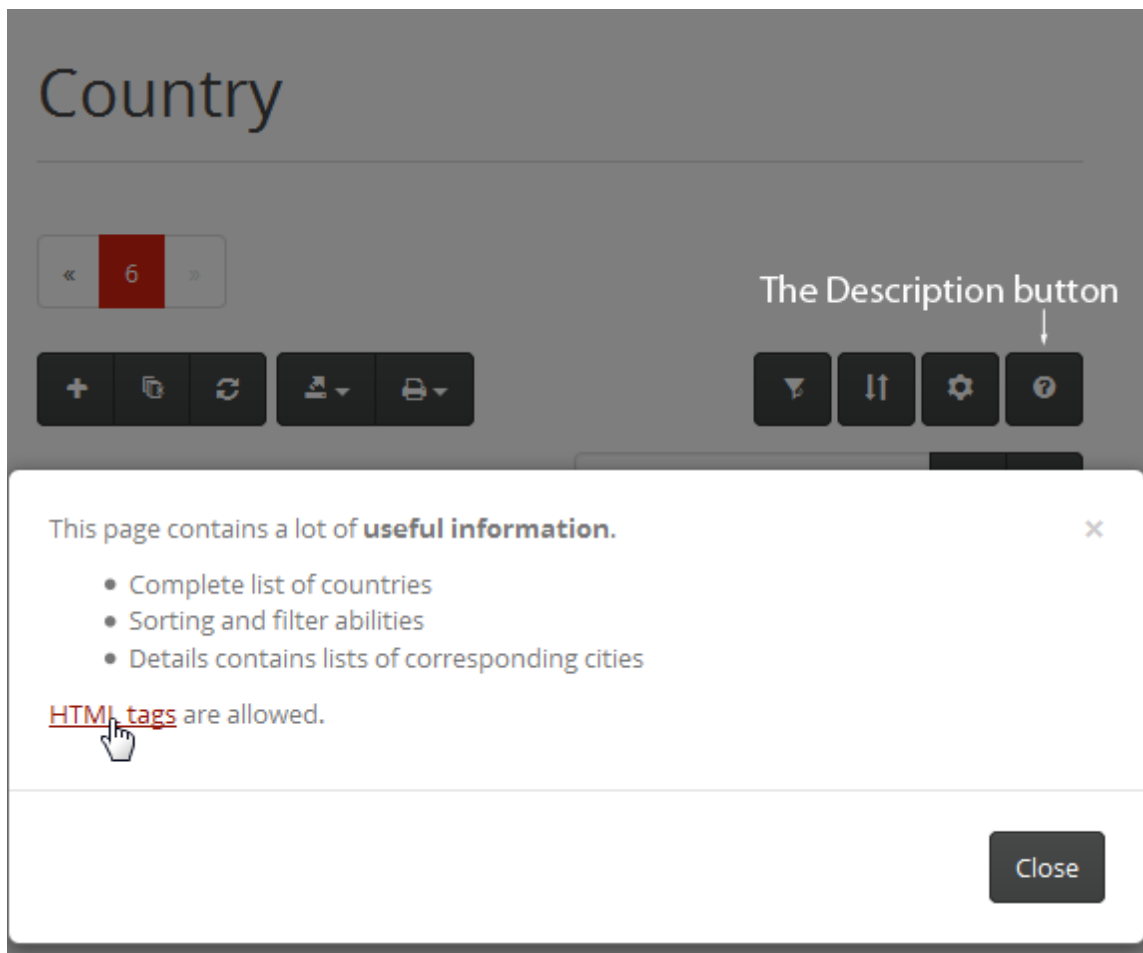
Data Grid.Autohiding Columns

To make a table view responsive, define column visibilities for each display resolution (mobile, tablet, desktop, and large desktop). If you are on a desktop, reduce the size of the browser window to see this feature in action.



[Detailed description](#)

If specified, the generated webpage is equipped with a button with a question mark on the right side of the title bar of the page. This button opens a modal window with the specified description. HTML tags are allowed.



Home page description

A text to be displayed at the home page ([live example](#)).

Partitioning

Range

Range partitioning means that each partition contains rows for which the partitioning expression value lies within a given range.

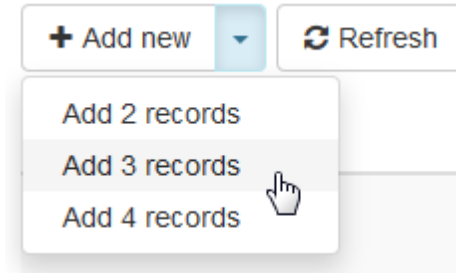
Behavior

Allow new records on Insert form

Defines whether the "Add another record" link is displayed on the Insert form.

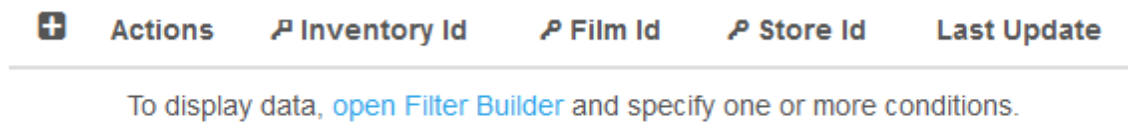
Add multiple records at a time (set of values)

Defines additional items (if any) for the "Add new" button's drop-down menu ([live example](#)).



Ask for filter condition before display data

If checked, no data is displayed in the grid until a user specifies a filter condition (can be useful for pages containing a lot of data). A page with this option applied looks as shown below.



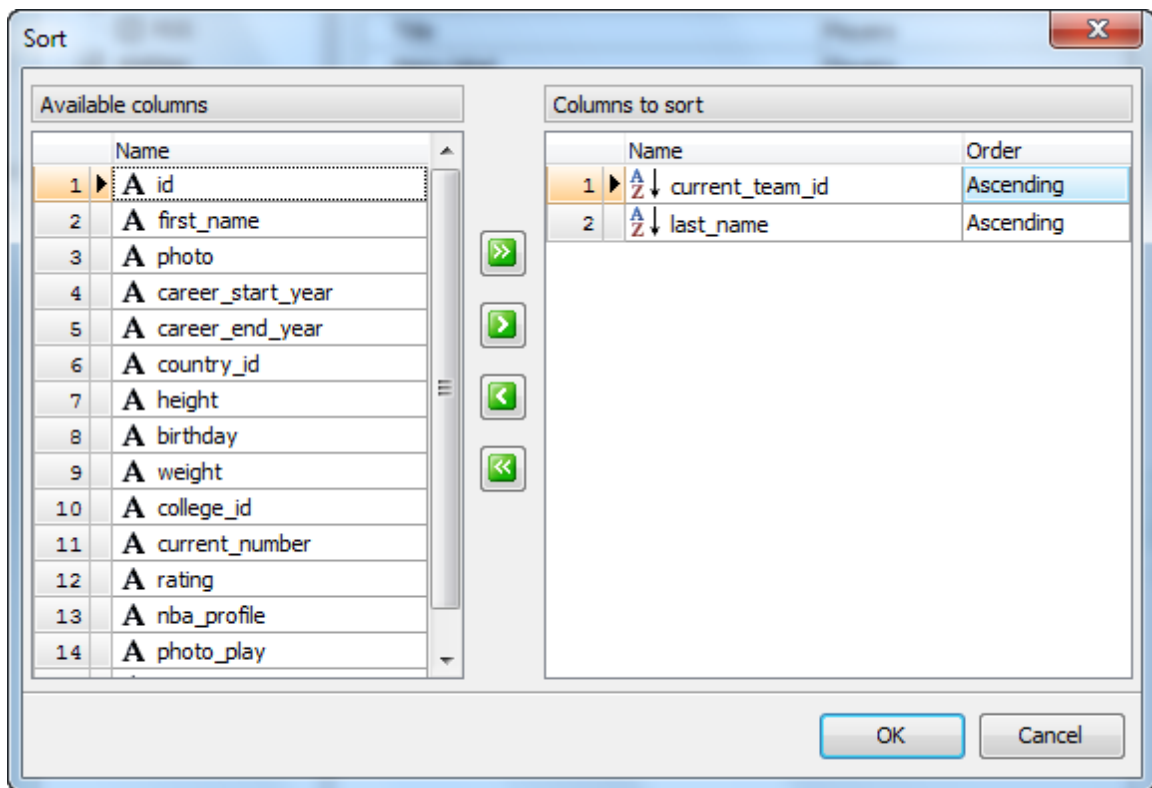
Options

File name

The name of generated *.php* file. If the file name extension is not specified, it will be defined according to application [output options](#)³³⁷.



Default ordering

To set the default sort order of page data, use the [Sort](#) dialog opened by the ellipsis button next to the [Default ordering](#) option. Specify the columns to sort by moving them from the [Available columns](#) list to the [Columns to sort](#) list and specify which order for each column you prefer.



Header columns for record comparison

Columns which values will be displayed in the header row of the data comparison grid ([live example](#)).

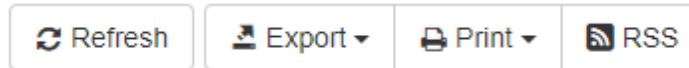
Show differences		Show all
	Vince Carter 	Tyson Chandler 
NBA Debut	1998	2001

Grid fixed width

To set a fixed width of the grid, check the [Grid fixed width](#), specify the [Width](#), and select the [Units](#). You can use both relative-length (like *px* or *mm*) and absolute-length (*em*, *rem*, etc) units. [Live Demo](#). More about [CSS units](#).

5.7.1.1 RSS options

PostgreSQL PHP Generator provides support for full-fledged RSS feeds based on a [page data source](#)^[22]. To retrieve a generated RSS feed, click the appropriate button as shown below.



The URL that provides the RSS feed for a page stored in the mypage.php file is as follows:

<http://www.mysite.com/mypage.php?operation=rss>

[Live example](#) is available in the NBA demo.

Adding an RSS feed to a page

To add an RSS feed to a page, open the [Page Properties](#) dialog, switch to the Common - > RSS tab, turn ON the [Enable](#) checkbox above the grid and provide values for the edit fields as described below.

Channel title

Defines the title of the channel.

Channel link

Defines the hyperlink to the channel.

Channel description

Describes the channel.

Item title [template](#)^[110]

Defines the title of the item.

Item link [template](#)^[110]

Defines the hyperlink to the item.

Item description [template](#)^[110]

Describes the item.

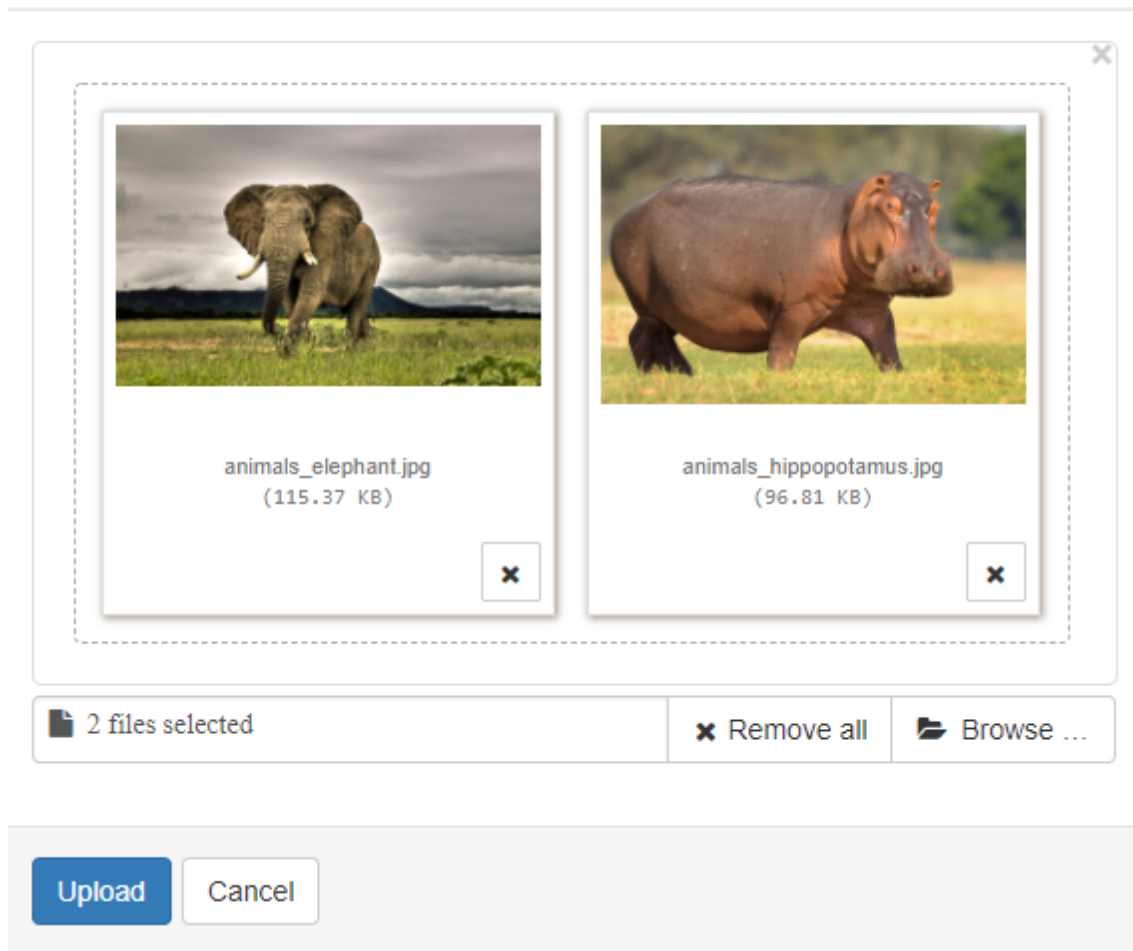
Item publication date field

This field is optional. Defines the last publication date for the content of the feed.

For more details see the [RSS specification](#).

5.7.1.2 Multi Upload

This feature allows you to upload multiple external files or images at a time. A new record is added to the database for each uploaded file.



To activate multi uploading for a page, turn ON the Enable checkbox above the grid, then setup storage properties and (optionally) upload area options as described below.

Storage	
Field name	image
Folder to upload	external_data/images/album/albu...
Store file name only	<input checked="" type="checkbox"/>
Generate random file name	<input type="checkbox"/>
File name template	%original_file_name%
Replace file if exists	<input checked="" type="checkbox"/>
Generate thumbnail	
Field name	
Folder to upload	
Store file name only	<input type="checkbox"/>
Generate random file name	<input type="checkbox"/>
File name template	small_%original_file_name%
Resize type	Fit by height
Resize height	100
Resize width	0
Upload area	
Caption	Images
Minimum file size (KB)	0
Maximum file size (KB)	0
Allowed file types	All
Allowed file extensions	
Custom attributes	

Storage

Field name

Column name to store names of uploaded files

Folder to upload

The folder to be used to store the uploaded files.

Generate random file name

If checked, uploaded files are saved with random names.

File name template

A [template](#) to be used for the file name generation.

Store file name only

Defines whether the full file path (e.g. external_data/uploaded_files/filename.ext) or only file name (e.g. filename.ext) is stored in the database after the uploading.

Replace file if exists

This option allows you to set whether the uploaded file will be saved or ignored in case a file with the same name already exists in the folder to upload.

Thumbnail options

Generate thumbnails

If checked, a thumbnail is generated for each uploaded image.

Field name

Column name to store names of thumbnails.

Folder to upload

The folder to be used to store thumbnails.

Store file name only

Defines whether the full file path (e.g. external_data/uploaded_files/filename.ext) or only file name (e.g. filename.ext) is stored in the database for each thumbnail.

Generate random file name

If checked, thumbnails are saved with random file names.

File name template

A [template](#)^[110] to be used for the thumbnail file name generation.

Resize type, Resize height, Resize width

These properties allow you to specify an algorithm of thumbnail generation.

Upload area

Caption

A label to be displayed on the left of the upload area

Minimum file size (KB)

A minimum file size allowed to be uploaded. A value of 0 means this restriction is not applied.

Maximum file size(KB)

A maximum file size allowed to be uploaded. A value of 0 means this restriction is not applied.

Allowed file types

The list of allowed file types for upload. By default all file types are allowed.

Allowed file extensions

The list of allowed file extensions for upload. By default all file extensions are allowed.

Custom attributes

This property allows you to setup any of options supported by the control via data attributes. Full list of available options can be found [here](#). For example, to hide file preview thumbnails, you can set the value of this property as follows:

```
data-show-preview="false"
```

Related events

The [OnFileUpload](#)^[190] and [OnBeforeInsertRecord](#)^[178] events fire on uploading each file to help you to accept only suitable files and/or provide values for other columns in the dataset.

5.7.2 Abilities

This tab allows you to specify operations available for a certain page. By default values of these options are set according to the corresponding values specified in the [Project Options](#) ^[226] dialog. To change the abilities for a certain page, uncheck the *Use default options* checkbox first. Click *Reset to defaults* to reset these options to their default values defined at the project level.

Actions

View

Defines whether browsing a single record is available. Possible values are Disabled, Separated Page (default value), and Modal window.

Edit

Defines whether data editing is available. Possible values are Disabled, [Separated Page](#) (default value), [Inline mode](#), and [Modal window](#).

Quick Edit

Defines whether the [quick edit](#) mode is available. Possible values are List and view (quick editing is available in both data grid and single record view window), List only, View only, and None.

Multi-edit

Defines whether [mass data editing](#) is available. Possible values are Disabled, Separated Page (default value), and Modal window.

Fields to be updated by default

Defines whether all or none fields will be selected in the appropriate control when the multi-edit command is executed.

Insert

Defines whether data insertion is available. Possible values are Disabled, [Separated Page](#) (default value), [Inline mode](#), and [Modal window](#).

Copying

Defines whether data copying is available. Possible values are Disabled, [Separated Page](#) (default value), [Inline mode](#), and [Modal window](#).

Delete

Defines whether data deletion is available. Possible values are Enabled (default) and Disabled.

Multi-delete

Defines whether multi deletion is available (user can delete multiple records at a time). Possible values are Enabled (default) and Disabled.

Filtering

Quick search

Defines whether the [Quick Search](#) box is displayed. Possible values are Enabled (default) and Disabled.

Filter Builder

Defines whether the [Filter Builder](#) tool is available. Possible values are Enabled (default) and Disabled.

Column Filter

Defines whether the [Column Filter](#) tool is available. Possible values are Enabled (default) and Disabled. It is also possible to enable/disable this feature at the column level.

Selection Filter

Defines whether [Selection Filters](#) is available. Possible values are Enabled (default) and Disabled.

Sorting

Sorting by click

Defines whether data can be [sorted by click on a column header](#). Possible values are Enabled (default) and Disabled.

Sorting by dialog

Defines whether the [data sort dialog](#) is available. Possible values are Enabled (default) and Disabled.

Additional

Runtime Customization

Defines whether end user is able to customize such page properties as number of records displayed on a page and a number of cards for each screen resolution. Possible values are Enabled (default) and Disabled.

Records comparison

Defines whether the [record comparison](#) tool is available. Possible values are Enabled (default) and Disabled.

Refresh

Defines whether the Refresh button is displayed. Possible values are Enabled (default) and Disabled.

5.7.3 Export and Print

This tab allows you to specify exporting and printing options available for a certain page. By default values of these options are set according to the corresponding values specified in the [Project Options](#) ²²⁶ dialog. To change settings for a certain page, uncheck the *Use default options* checkbox first. Click *Reset to defaults* to reset these options to their default values defined at the project level.

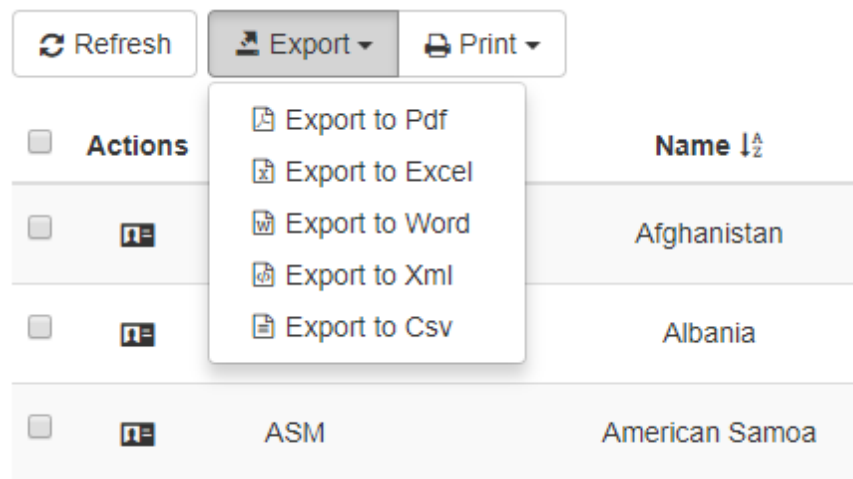
The screenshot shows the 'Properties' dialog box for the PostgreSQL PHP Generator, specifically the 'Export and Print' tab. The left sidebar contains a tree view with the following items: Common, RSS, Multi upload, Abilities, Export and Print (selected), and Options. The main area of the dialog is titled 'Export and Print' and contains several sections for configuring export and print options. At the top right of the main area is a link 'Reset to defaults'. The sections are as follows:

- Use default options:** A checked checkbox.
- All records from grid:** A section with two rows: 'Export' with a dropdown menu showing 'Pdf,Excel,Word,Xml,Csv' and 'Print' with a checked checkbox.
- Selected records from grid:** A section with two rows: 'Export' with a dropdown menu showing 'Pdf,Excel,Word,Xml,Csv' and 'Print' with a checked checkbox.
- Single record from grid:** A section with two rows: 'Export' with a dropdown menu showing 'None selected' and 'Print' with an unchecked checkbox.
- Record from view form:** A section with two rows: 'Export' with a dropdown menu showing 'Pdf,Excel,Word,Xml,Csv' and 'Print' with a checked checkbox.
- Options:** A section with two rows: 'Open print form in new tab' with a checked checkbox and 'Open exported Pdf in new tab' with a checked checkbox.

At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

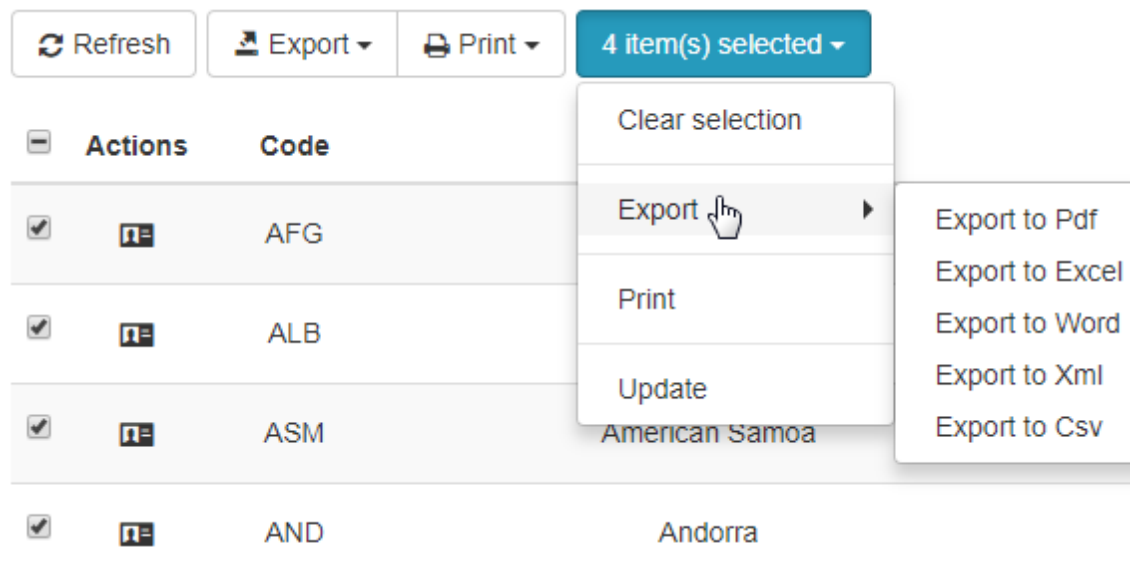
All records from grid

Defines exporting and printing options to be available for the data grid.


















Selected records from grid

Defines exporting and printing options to be available for the selected records in the data grid.



Single record from grid

Defines exporting and printing options to be available for a single record in the data grid.

<input type="checkbox"/>	Actions	Code
<input type="checkbox"/>	  	ABW
<input type="checkbox"/>	  	AFG
<input type="checkbox"/>	  	AGO
<input type="checkbox"/>	  	AIA
<input type="checkbox"/>	  	ALB

[Record from view form](#)

Defines exporting and printing options to be available for a single record view form.

View

[← Back to list](#) [Export](#) [Print](#)

Code AFG

Name Afghanistan

Continent Asia

Indepyear 1,919

Population 22,720,000

Lifeexpectancy 45.90

Export to Pdf

Export to Excel

Export to Word

Export to Xml

Export to Csv

Options

The [Open print form in new tab](#) option allows you to open print forms in a new browser tab. The [Open exported PDF in new tab](#) option allows you to open exported PDFs in a new browser tab. The both options are enabled by default.

5.7.4 Options

This tab allows you to specify page options. By default values of these options are set according to the corresponding values specified in the [Project Options](#)^[226] dialog. To change settings for a certain page, uncheck the *Use default options* checkbox first. Click *Reset to defaults* to reset these options to their default values defined at the project level.

Common

These options are applied for all generated pages.

Content encoding

The default option value is UTF-8 and you must have really good reasons to change it.

Page direction

Allows you to select between Left-to-right and Right-to-left page directions.

Modal form size

A preferred size (default, small, or large) for View, Edit, and Insert modal windows.

Hide sidebar menu by default

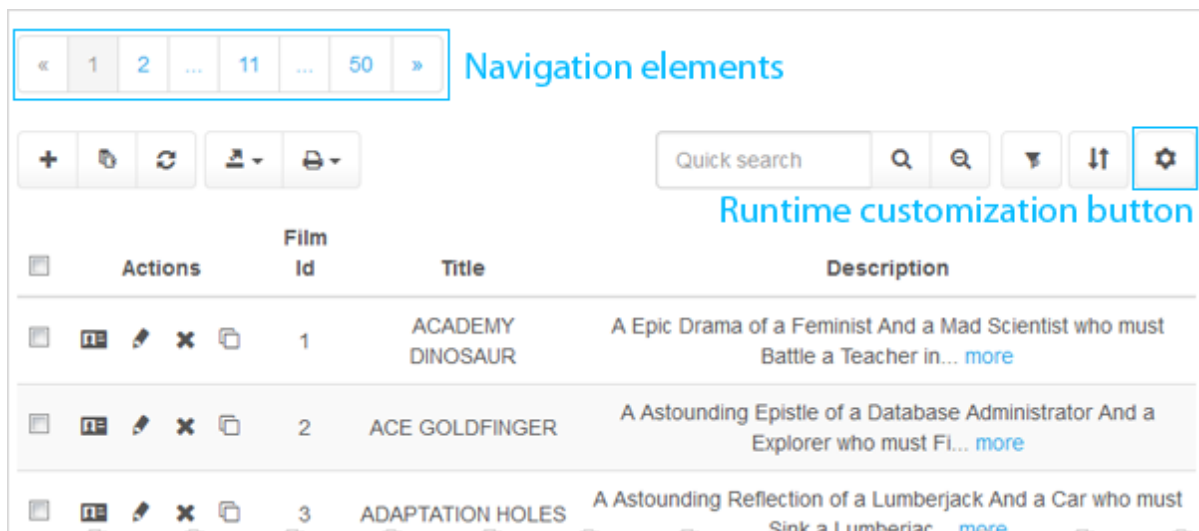
If checked, the sidebar menu will be closed until a user opens it manually.

List

These options are applied for List views (both grid and card).

Page navigator

Allows you to specify the position of the navigational control(s) and the default number of records displayed on the page. This value may be changed by application users in runtime if the Runtime Customization option is enabled.



To create a custom pagination, use [Data Partitioning](#)^[220].

View mode

Select whether page data are displayed in a grid or as info cards. The end-user can easily switch from the grid mode to the card one and vice versa using the Page Settings dialog (if the Runtime Customization option is enabled).

Grid view mode

Photo	First name	Last name	Country	Team	Actions
	Kobe	Bryant	USA	Los Angeles Lakers	
	Vince	Carter	USA	Memphis Grizzlies	
	Tyson	Chandler	USA	Dallas Mavericks	
	Jamal	Crawford	USA	Los Angeles Clippers	
	Tim	Duncan	USA	San Antonio Spurs	
	Kevin	Durant	USA	Oklahoma City Thunder	
	Steven	Galtieri	Italy	Denver Nuggets	

Cards view mode

Photo	First name	Last name	Birthday	Country	Height	Weight	College	NBA Debut	Position	Team	Number
	Kobe	Bryant	23.08.1978	USA	198	96.2	n/a	1996	Guard	Los Angeles Lakers	24
	Vince	Carter	26.01.1977	USA	196	99.5	North Carolina	1996	Guard/Forward	Memphis Grizzlies	15

Number of cards in a row

Allows you to specify maximum number of cards displayed in a row for each screen resolution. These values also can be changed by end-users if the Runtime Customization option is enabled. [Live Example](#).

Control buttons position

Select whether the Edit, View, Delete, and Copy buttons to be displayed on the left side of the generated page or on the right side. [Live Example](#).

Bordered table

Check this option to add borders on all sides of the table and cells. [Live Example](#).

Condensed table

Turn this option 'ON' to make tables more compact by cutting cell padding in half. [Live Example](#).

Highlight row at mouse over

Allows you to enable/disable the highlighting of rows at mouse over.

Use images for actions

Defines whether images or text captions are used for control buttons (like Edit, View, or Delete).

Use fixed grid header

Allows you to make the grid header are non-scrollable. [Live Example](#).

Show key columns images

Turn is 'ON' to mark key columns with the 'key' images.

<input type="checkbox"/> City Id	City	<input type="checkbox"/> Country Id	Last Update
1	A Corua (La Corua)	Spain	2006-02-14 16:45:25
2	Abha	Saudi Arabia	2006-02-14 16:45:25
3	Abu Dhabi	United Arab Emirates	2006-02-14 16:45:25

Show line numbers

Check this option to add row numbering to the grid. [Live Example](#).

Edit/Insert

These options are applied to data input forms.

Display "Set NULL" checkboxes

Defines whether "Set NULL" checkboxes are displayed for each control in Edit and Insert forms.

Display "Set Default" checkboxes

Defines whether "Set Default" checkboxes are displayed for each control in Edit and Insert forms.

Error message placement

Allows you to select the placement for error messages. Possible values are "Below the form", "Above the form", and "Below and above the form". Default value is "Below the form".

Reload page after Ajax operations

Allows you to reload the page after [Inline](#) or [Modal](#) editing/inserting.

5.8 Data Partitioning

The Data Partitioning wizard allows you to create a custom pagination i.e. split the records on the generated page by a specified criteria. [Live demos](#).

Partitioning types

Supported partitioning types are as follows:

Range partitioning

Selects a partition by determining if the partitioning expression value is inside a certain range. [Live demo](#).

List partitioning

A partition is assigned a list of values. If the partitioning expression value has one of these values, the partition is chosen. For example, all rows where the column 'Country' is either Iceland, Norway, Sweden, Finland or Denmark could build a partition for the 'Nordic countries'. [Live demo](#).

Custom partitioning

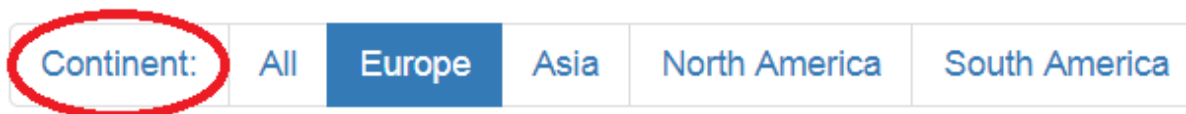
Create your own partitioning with fill partitions and build conditions functions. [Live demo](#).

To disable data partitioning, select None in the drop-down list, then press OK.

Partition navigator options

Navigator caption

Defines the caption of the label to be displayed



Navigation style

Defines whether the partition navigator is displayed as a list of hyperlinks or as a combobox.

Allow to view record from all partitions

Enable/disable this option to allow/deny viewing data from all partitions.

Example 1 (Range partitioning)

There is a table storing information about films such as title, release year, and length. See definition here


```
CREATE TABLE film (
  film_id          integer NOT NULL,
  title            varchar(255) NOT NULL,
  release_year     integer,
  length           integer UNSIGNED,
  /* Keys */
  PRIMARY KEY (film_id)
```






);

To create a pagination by the film length, select [Range partitioning](#) at the first wizard step, choose 'length' as partition expression and set the partitioning ranges as follows:

Partition expression (e.g. unit_price, MONTH(payment_date))

length

 Partitions

	⋮	Caption	Right bound
1		<<1h.	60
2		[1h.,2h.]	120
3		[2h.,3h.]	180
4		3h.>>	Max Value

Here you can see the result PHP script:

Film

Film length:
 <<1h.
 [1h.,2h.]
 [2h.,3h.]
 3h.>>

«
 1
 »

Title	Description	Release Year	Length
ACE GOLDFINGER	A Astounding Epistle of a Database Administrator And a Explo... more	2006	48

Example 2 (Custom partitioning)

Assume that a table 'customer' contains the 'last_name' column. Our goal is to group customers by the first letter of their last name as displayed below.

Customer

Last name starts with:	A	B	C	D	E	F	G	H	I	J	K	L	M
------------------------	---	---	---	---	---	---	---	---	---	---	---	---	---

«	1	2	3	»
---	---	---	---	---

+ Add new	↻ Refresh	📄 Export ▾	🖨 Print ▾
-----------	-----------	------------	-----------

+	Customer Id	Store Id	First Name	Last Name	Email
+ ▾	5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilac
+ ▾	37	1	PAMELA	BAKER	PAMELA.BAKER@sakilacus

To get around such a situation, select **Custom partitioning** and define functions as follows:

```
function GetPartitions($partitions)
{
    $tmp = array();
    $this->GetConnection()->ExecQueryToArray("
        SELECT DISTINCT
            left(c.last_name, 1) as first_letter
        FROM customer c
        ORDER BY first_letter", $tmp
    );
    foreach($tmp as $letter) {
        $partitions[$letter['first_letter']] = $letter['first_letter'];
    }
}

function GetPartitionCondition($partitionKey, &$condition)
{
    $condition = "left(last_name, 1) = '$partitionKey'";
}
```


5.9 Data Validation

Data validation is intended to provide certain guarantees for accuracy and consistency for any of various kinds of user input. PostgreSQL PHP Generator allows you to check for correctness of input data on the client side and on the server side as well.

Client-side data validation

This type of data validation refers to validation performed on the client side. That is, it's performed on the client's, also known as the user's, computer. This type of validation checks the input information before it gets to the server. It has following advantages:

- Reducing the amount of traffic to and from your web server since the data doesn't have to travel to the server before it gets checked (server-side data validation). If an error is found in the data in the server-side validation, the error must travel back to the user, who then corrects the problem and resubmits the data. This back-and-forth traveling can eat up precious bandwidth.
- Saving processing time on the server by ensuring that data is clean and clear before it gets to your database.
- Giving immediate feedback to your users since the processing is performed in their browsers. The user doesn't have to wait for the data to travel to the server to be checked and a response to be sent. They know immediately whether they need to tweak the information they have entered.

PostgreSQL PHP Generator allows you to implement the client-side data validation on two levels: when a user leaves a certain control and when a user submits a form.

To enable **validation of data entered in a control**, specify the appropriate [Client validator](#)^[62].

Range	The generated script validates if number is between the largest and smallest values.
Length range	The script validates if the length of text is between the min length and the max length.
Email	The script makes the element require a valid email.
Credit card	The script makes the element require a credit card number.
Number	The script makes the element require a decimal number.
URL	The script makes the element require a valid URL.
Digits	The script makes the element require digits only.
Regular expression	You can also specify your regular expression for data validation. Such expression is a pattern and every character entered in a form field is matched against that pattern – the form can only be submitted if the pattern and the user-input matches.

To implement **data validation on submitting a form**, use the [OnInsertFormValidate](#)^[139] and [OnEditFormValidate](#)^[139] client side events. It may be helpful to check the compatibility of input data. When a user clicks the Save button, Javascript ensures that the data conforms to specifications before moving on the server.

Server-side data validation

This type of data validation consists of validating user input on the server.

This type of data validation means using PHP to verify that good values have been sent to the script. Using server-side validation has pretty much the exact opposite pros and cons of client-side development: it is more secure, works seamlessly with all browsers, and allows to use session variables, but it does so at the cost of slightly higher server load and slower feedback for users.

[OnBeforeUpdateRecord](#)¹⁷⁹

[OnBeforeInsertRecord](#)¹⁷⁸

6 Project Options

PostgreSQL PHP Generator allows you to specify default settings to be applied to each generated webpage. By default, most of these options have the same values as the [application options](#)^[328]. To open this dialog window, follow the [Setup project options to configure default page settings](#) link at the [Page Management](#)^[27] wizard step

All settings are grouped into several tabs:

[Page](#)^[228]

This tab group allows you to define default settings for all pages of the website including page appearance, operations to be available, exporting and printing options, and so on. By default these settings are set in accordance with the [Application options](#)^[328].

Display formats

These tab fields allow to adjust the display data formats in the way you need. By default these settings are set in accordance with the [Application options](#)^[328].

[Shared options](#)^[235]

This tab allows you to specify options that are applied for a whole website (i.e. shared between all pages).

Events

This tab allows you to specify [application-level events](#)^[117].

Project options

Page

- Export
- Abilities
- Display formats
- Shared options
- Events

Page

Common

Content encoding	UTF-8	▼
Page direction	Left-to-right	▼
Modal form size	Default	▼
Hide sidebar menu by default	<input type="checkbox"/>	

List

Page navigator	Both	▼
Records per page	20	▲▼
View mode	Grid	▼
Number of cards in a row	Phone=1,Tablet=1,Desktop=...	...
Control buttons position	Left	▼
Bordered table	<input type="checkbox"/>	
Condensed table	<input type="checkbox"/>	
Highlight row at mouse over	<input type="checkbox"/>	
Use images for actions	<input checked="" type="checkbox"/>	
Use fixed grid header	<input type="checkbox"/>	
Show key columns images	<input type="checkbox"/>	
Show line numbers	<input type="checkbox"/>	
Disable custom ordering	<input type="checkbox"/>	

Edit/Insert

Display "Set NULL" checkboxes	<input type="checkbox"/>	
Display "Set Default" checkboxes	<input type="checkbox"/>	

☐ Apply these options to pages with non-default settings too

OK Cancel

6.1 Page

This tab allows you to specify default page options those can be later changed [for a certain page](#)^[217].

Common

These options are applied for all generated pages.

Content encoding

The default option value is UTF-8 and you must have really good reasons to change it.

Page direction

Allows you to select between Left-to-right and Right-to-left page directions.

Modal form size

A preferred size (default, small, or large) for View, Edit, and Insert modal windows.

Hide sidebar menu by default

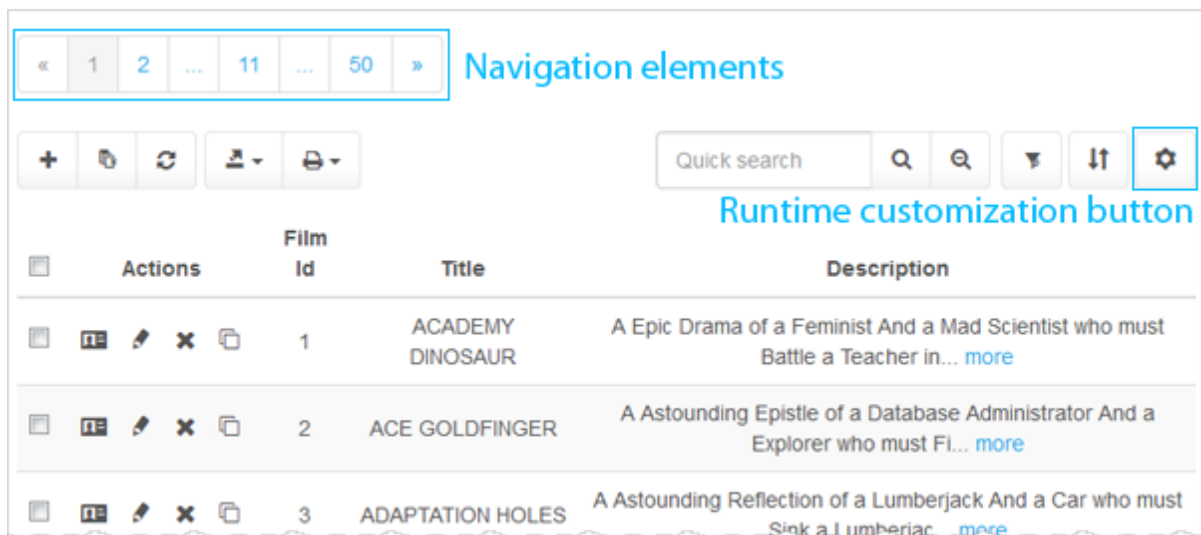
If checked, the sidebar menu will be closed until a user opens it manually.

List

These options are applied for List views (both grid and card).

Page navigator

Allows you to specify the position of the navigational control(s) and the default number of records displayed on the page. This value may be changed by application users in runtime if the Runtime Customization option is enabled.

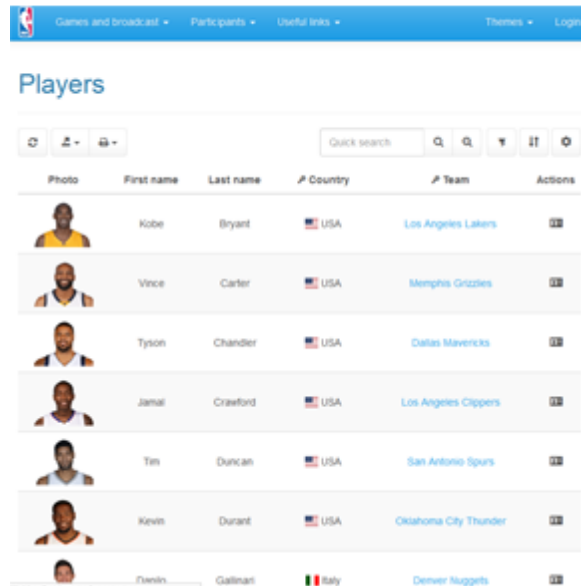


To create a custom pagination, use [Data Partitioning](#)^[220].

View mode

Select whether page data are displayed in a grid or as info cards. The end-user can easily switch from the grid mode to the card one and vice versa using the Page Settings dialog (if the Runtime Customization option is enabled).

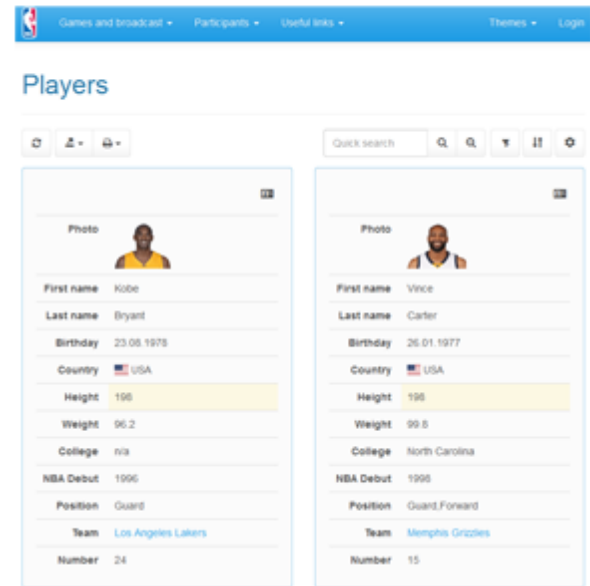
Grid view mode



The screenshot shows the NBA website's 'Players' section in grid view mode. It features a table with columns for Photo, First name, Last name, Country, Team, and Actions. The table lists several players, including Kobe Bryant, Vince Carter, Tyson Chandler, Jamal Crawford, Tim Duncan, Kevin Durant, and Steven Galtieri. Each row includes a small photo of the player, their name, country, team, and a set of control buttons (Edit, View, Delete, Copy).

Photo	First name	Last name	Country	Team	Actions
	Kobe	Bryant	USA	Los Angeles Lakers	[Edit] [View] [Delete] [Copy]
	Vince	Carter	USA	Memphis Grizzlies	[Edit] [View] [Delete] [Copy]
	Tyson	Chandler	USA	Dallas Mavericks	[Edit] [View] [Delete] [Copy]
	Jamal	Crawford	USA	Los Angeles Clippers	[Edit] [View] [Delete] [Copy]
	Tim	Duncan	USA	San Antonio Spurs	[Edit] [View] [Delete] [Copy]
	Kevin	Durant	USA	Oklahoma City Thunder	[Edit] [View] [Delete] [Copy]
	Steven	Galtieri	Italy	Denver Nuggets	[Edit] [View] [Delete] [Copy]

Cards view mode



The screenshot shows the NBA website's 'Players' section in cards view mode. It displays two player cards side-by-side. Each card contains a photo, first name, last name, birthday, country, height, weight, college, NBA debut, position, team, and number. The cards are designed to be visually appealing and easy to read.

Photo	First name	Last name	Birthday	Country	Height	Weight	College	NBA Debut	Position	Team	Number
	Kobe	Bryant	23.08.1978	USA	198	96.2	n/a	1996	Guard	Los Angeles Lakers	24
	Vince	Carter	26.01.1977	USA	198	99.5	North Carolina	1996	Guard/Forward	Memphis Grizzlies	15

Number of cards in a row

Allows you to specify maximum number of cards displayed in a row for each screen resolution. These values also can be changed by end-users if the Runtime Customization option is enabled. [Live Example](#).

Control buttons position

Select whether the Edit, View, Delete, and Copy buttons to be displayed on the left side of the generated page or on the right side. [Live Example](#).

Bordered table

Check this option to add borders on all sides of the table and cells. [Live Example](#).

Condensed table

Turn this option 'ON' to make tables more compact by cutting cell padding in half. [Live Example](#).

Highlight row at mouse over

Allows you to enable/disable the highlighting of rows at mouse over.

Use images for actions

Defines whether images or text captions are used for control buttons (like Edit, View, or Delete).

Use fixed grid header

Allows you to make the grid header are non-scrollable. [Live Example](#).

Show key columns images

Turn is 'ON' to mark key columns with the 'key' images.

<input type="checkbox"/> City Id	City	<input type="checkbox"/> Country Id	Last Update
1	A Corua (La Corua)	Spain	2006-02-14 16:45:25
2	Abha	Saudi Arabia	2006-02-14 16:45:25
3	Abu Dhabi	United Arab Emirates	2006-02-14 16:45:25

Show line numbers

Check this option to add row numbering to the grid. [Live Example](#).

Edit/Insert

These options are applied to data input forms.

Display "Set NULL" checkboxes

Defines whether "Set NULL" checkboxes are displayed for each control in Edit and Insert forms.

Display "Set Default" checkboxes

Defines whether "Set Default" checkboxes are displayed for each control in Edit and Insert forms.

Error message placement

Allows you to select the placement for error messages. Possible values are "Below the form", "Above the form", and "Below and above the form". Default value is "Below the form".

Reload page after Ajax operations

Allows you to reload the page after [Inline](#) or [Modal](#) editing/inserting.

6.1.1 Abilities

This tab allows you to specify default operation to be available for all pages. They can be later changed [for a certain page](#)^[212].

Actions

View

Defines whether browsing a single record is available. Possible values are Disabled, Separated Page (default value), and Modal window.

Edit

Defines whether data editing is available. Possible values are Disabled, [Separated Page](#) (default value), [Inline mode](#), and [Modal window](#).

Quick Edit

Defines whether the [quick edit](#) mode is available. Possible values are List and view (quick editing is available in both data grid and single record view window), List only, View only, and None.

Multi-edit

Defines whether [mass data editing](#) is available. Possible values are Disabled, Separated Page (default value), and Modal window.

Fields to be updated by default

Defines whether all or none fields will be selected in the appropriate control when the multi-edit command is executed.

Insert

Defines whether data insertion is available. Possible values are Disabled, [Separated Page](#) (default value), [Inline mode](#), and [Modal window](#).

Copying

Defines whether data copying is available. Possible values are Disabled, [Separated Page](#) (default value), [Inline mode](#), and [Modal window](#).

Delete

Defines whether data deletion is available. Possible values are Enabled (default) and Disabled.

Multi-delete

Defines whether multi deletion is available (user can delete multiple records at a time). Possible values are Enabled (default) and Disabled.

Filtering

Quick search

Defines whether the [Quick Search](#) box is displayed. Possible values are Enabled (default) and Disabled.

Filter Builder

Defines whether the [Filter Builder](#) tool is available. Possible values are Enabled (default) and Disabled.

Column Filter

Defines whether the [Column Filter](#) tool is available. Possible values are Enabled (default) and Disabled. It is also possible to enable/disable this feature at the column level.

Selection Filter

Defines whether [Selection Filters](#) is available. Possible values are Enabled (default) and Disabled.

Sorting

Sorting by click

Defines whether data can be [sorted by click on a column header](#). Possible values are Enabled (default) and Disabled.

Sorting by dialog

Defines whether the [data sort dialog](#) is available. Possible values are Enabled (default) and Disabled.

Additional

Runtime Customization

Defines whether end user is able to customize such page properties as number of records displayed on a page and a number of cards for each screen resolution. Possible values are Enabled (default) and Disabled.

Records comparison

Defines whether the [record_comparison](#) tool is available. Possible values are Enabled (default) and Disabled.

Refresh

Defines whether the Refresh button is displayed. Possible values are Enabled (default) and Disabled.

6.1.2 Export and Print

This tab allows you to specify default exporting and printing options to be available for all pages. They can be later changed [for a certain page](#)²¹³.

Project options

Page: Export and Print

- All records from grid**

Export	Pdf,Excel,Word,Xml,Csv
Print	<input checked="" type="checkbox"/>
- Selected records from grid**

Export	Pdf,Excel,Word,Xml,Csv
Print	<input checked="" type="checkbox"/>
- Single record from grid**

Export	None selected
Print	<input type="checkbox"/>
- Record from view form**

Export	Pdf,Excel,Word,Xml,Csv
Print	<input checked="" type="checkbox"/>
- Options**

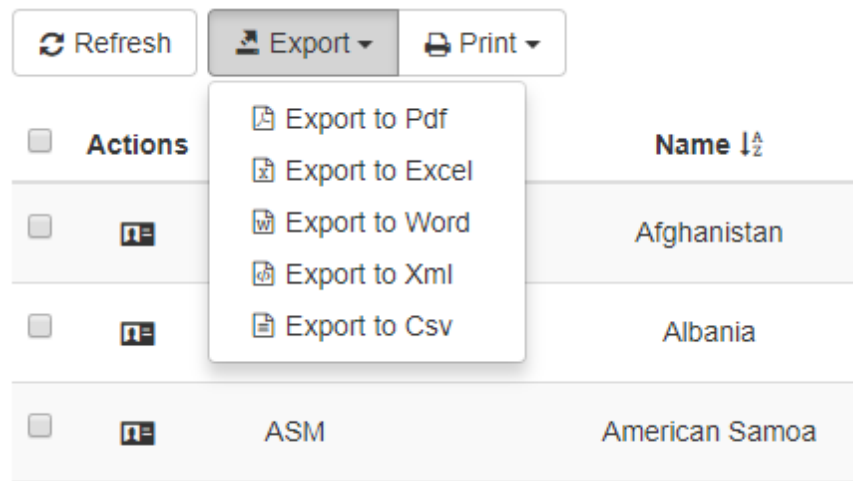
Open print form in new tab	<input checked="" type="checkbox"/>
Open exported Pdf in new tab	<input checked="" type="checkbox"/>

☐ Apply these options to pages with non-default settings too

OK Cancel

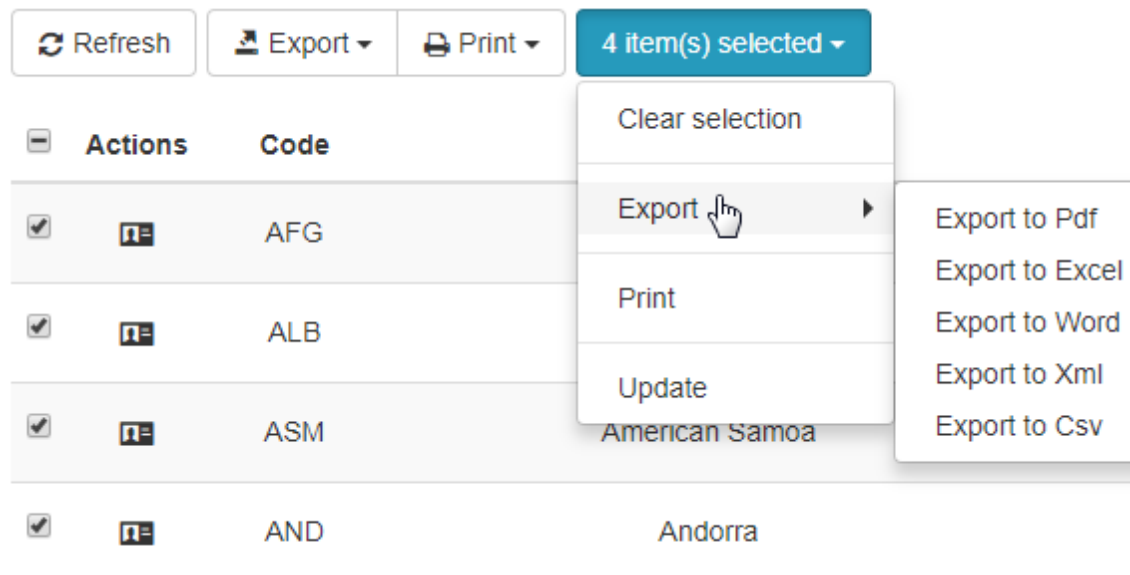
All records from grid

Defines exporting and printing options to be available for the data grid.


















Selected records from grid

Defines exporting and printing options to be available for the selected records in the data grid.



Single record from grid

Defines exporting and printing options to be available for a single record in the data grid.

<input type="checkbox"/>	Actions	Code
<input type="checkbox"/>	  	ABW
<input type="checkbox"/>	  	AFG
<input type="checkbox"/>	  	AGO
<input type="checkbox"/>	  	AIA
<input type="checkbox"/>	  	ALB

[Record from view form](#)

Defines exporting and printing options to be available for a single record view form.

View

[← Back to list](#) [Export](#) [Print](#)

Code AFG

Name Afghanistan

Continent Asia

Indepyear 1,919

Population 22,720,000

Lifeexpectancy 45.90

Export to Pdf

Export to Excel

Export to Word

Export to Xml

Export to Csv

Options

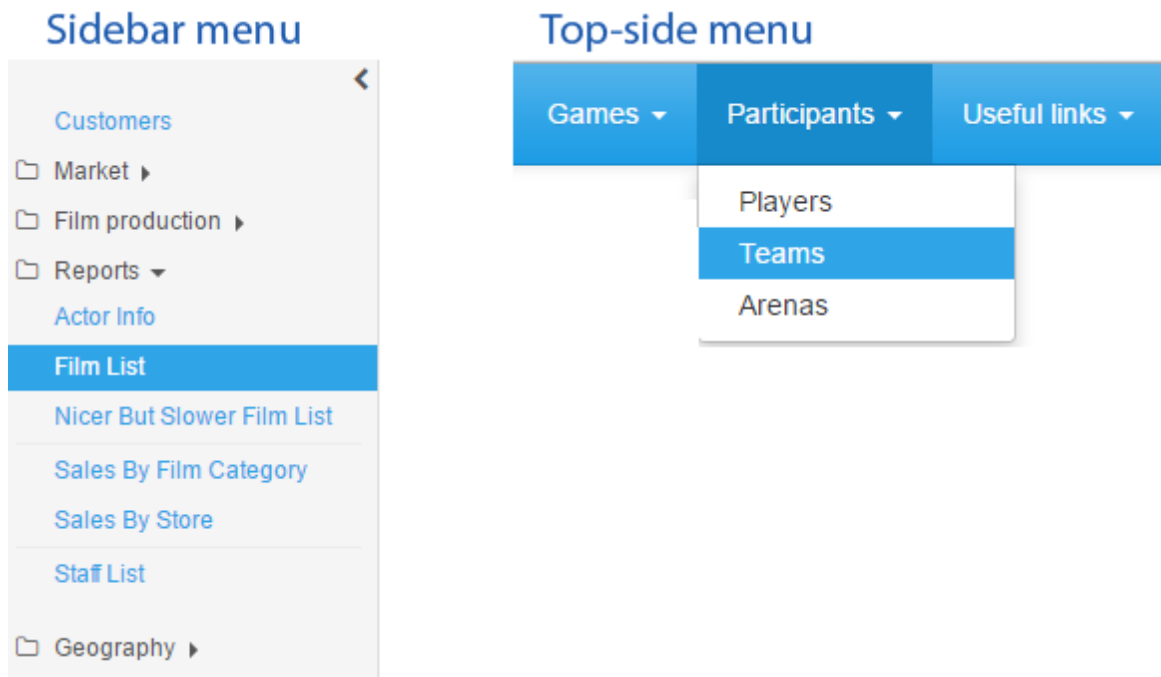
The [Open print form in new tab](#) option allows you to open print forms in a new browser tab. The [Open exported PDF in new tab](#) option allows you to open exported PDFs in a new browser tab. The both options are enabled by default.

6.2 Shared options

This tab allows you to specify options that are applied for a whole website (i.e. shared between all pages).

Menu mode

Defines the location of the application menu. Possible values are as **Top-side drop-down menu** (the menu will be displayed on the top of each page), **Sidebar menu** (the menu will be displayed on the left of each page), and **None** (no menu will be displayed at all).

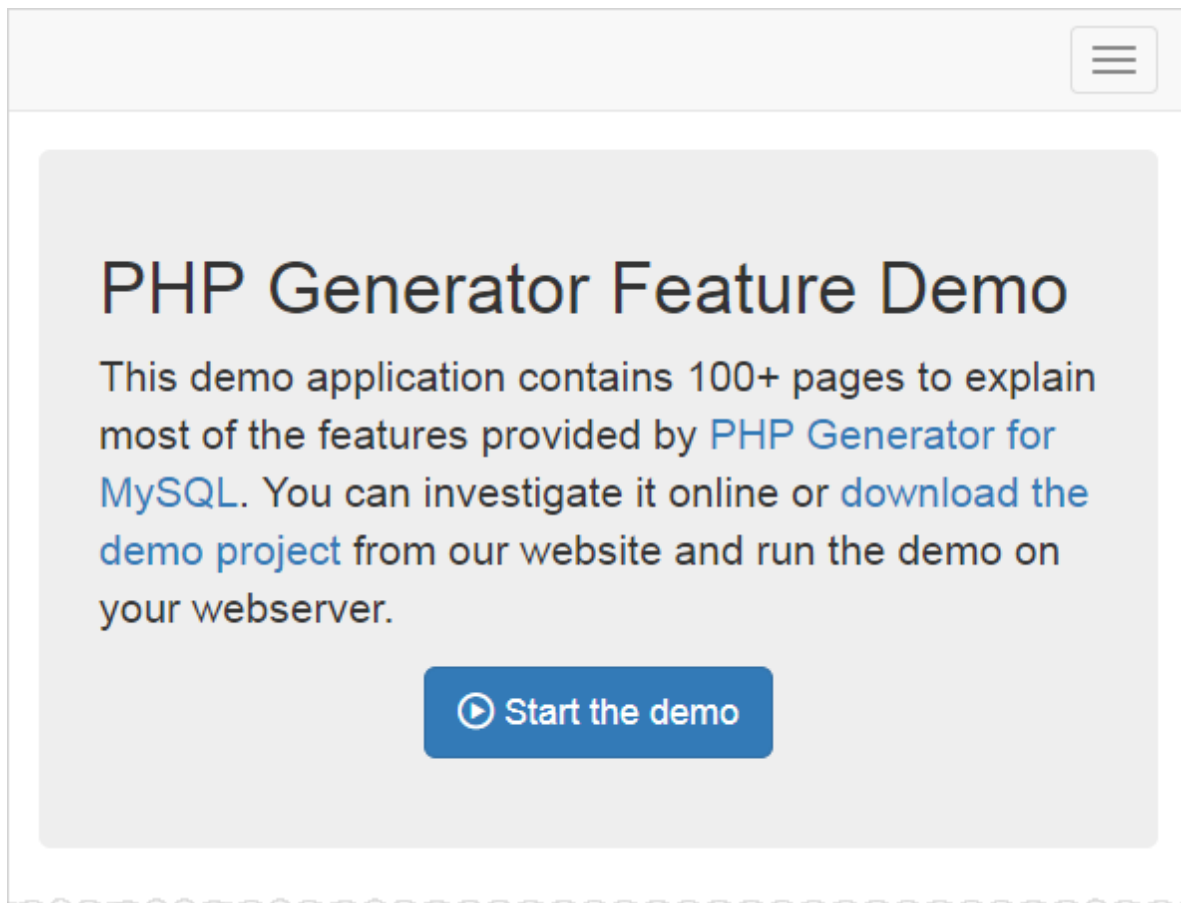


Generate home page

Defines whether a home page for the application is generated ([live example](#)). If checked, you can customize a filename for this page (default value is index.php).

Home page banner

A piece of the HTML code to be displayed at the top of the application home page. You can use this property for calling extra attention to some special content or information.



Show breadcrumb navigation

Defines whether a breadcrumb navigation is available in the application.



Outgoing mail settings

A set of settings to be used to send emails from the generated website. By default the [SendMail](#) mail transfer agent is used. These settings are used in all [email-based features](#)^[256] and when an email is sent with the [sendMailMessage](#)^[322] function. See also [Setting up common SMTP servers](#)^[326].

Common	
Mailer	Sendmail
From mail	admin@mysite.com
From name	Site Admin
SMTP options	
Host	
Port	25
<input type="checkbox"/> Use authentication	<input type="checkbox"/>
Username	
Password	
Encryption	None

Force show PHP errors and warnings

If checked, the following PHP directives will be added to all pages:

```
error_reporting(E_ALL ^ E_NOTICE);
ini_set('display_errors', 'On');
```

We would recommend you to turn this option ON in the development environment and turn it OFF in the production environment.

Show environment variables

If checked, the list of the available [environment variables](#)^[195] will be displayed below the data grid. These variables can be accessible in server-side event handlers via the [GetEnvVar](#)^[304] method of the [Page](#)^[304] class.

Disable magic quotes_runtime option

Defines whether the [magic quotes_runtime](#) PHP option will be enabled or disabled. Please note that this options was removed as of PHP 5.4.

Minify JavaScript code

Defines whether the generated JavaScript code should be minified (compressed). Minification reduces the amount of data that needs to be transferred on loading a page, so we would recommend you to keep this option checked.

Offline mode

Turn this option ON for websites running on web servers without an Internet connection. Note that when this option is checked, you will not be able to use [charts](#)^[199] and [embedded videos](#)^[99] in your application.

PHP options (ini_set)

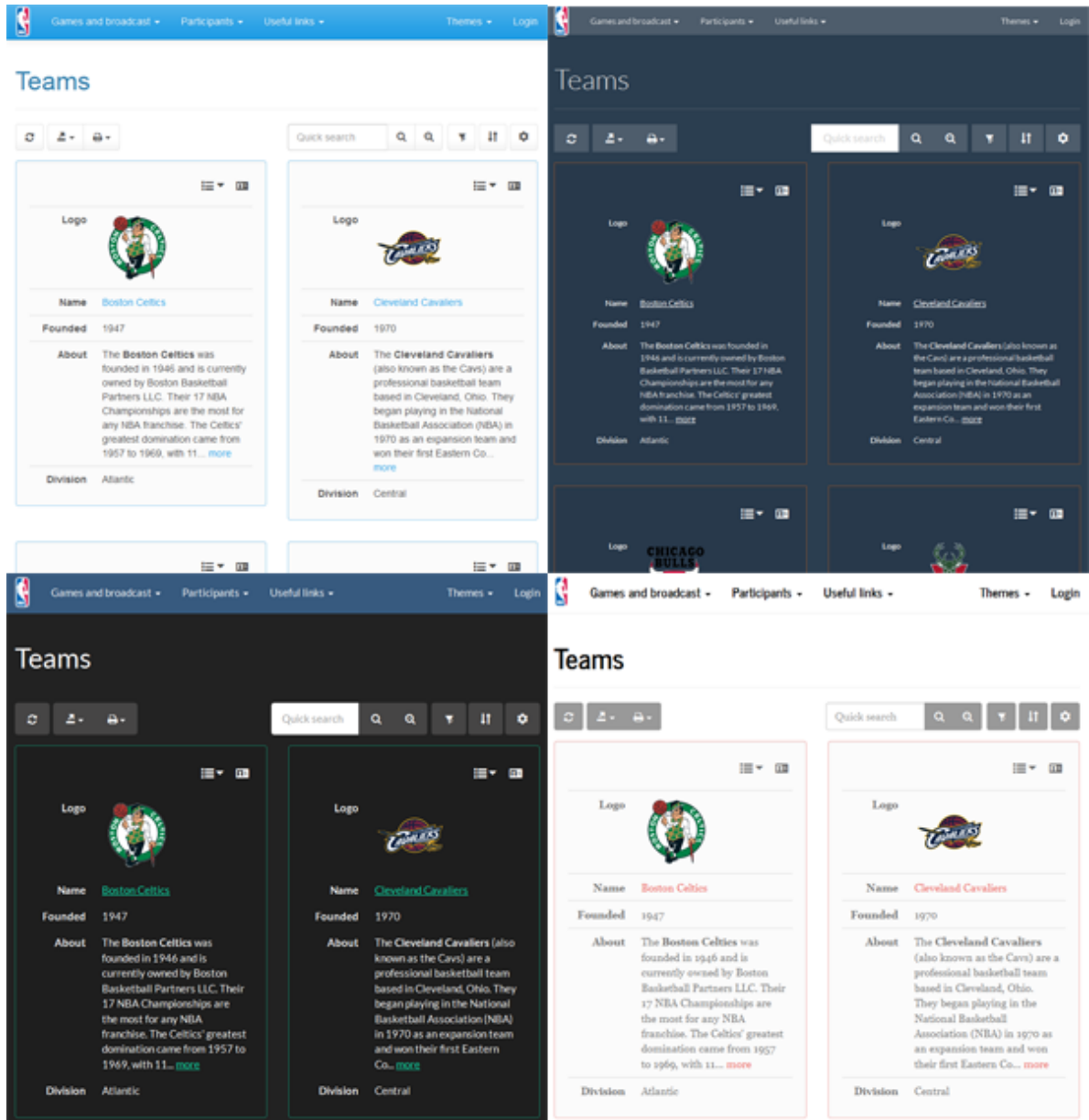
This allows you to customizing PHP configuration options during the application runtime. These options are represented as calls of the [ini_set](#) function in the phpgen_settings.php file.

7 More customization options

Besides settings available in the [Page Editor](#)^[33] tool, PostgreSQL PHP Generator allows you to customize created webpages in the following ways:

[Built-in color schemes](#)^[24]

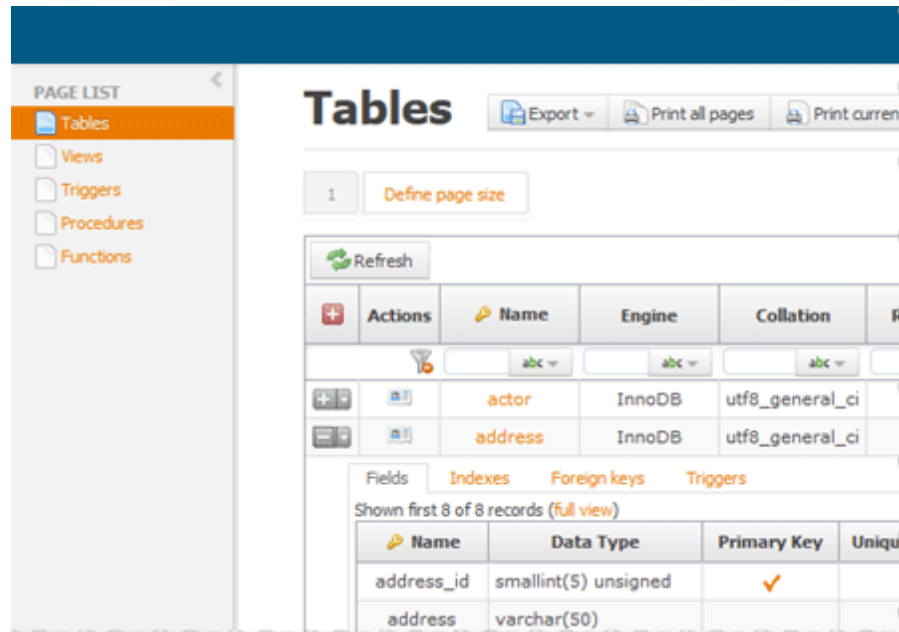
The simplest way to adjust the look and feel of the result application is to select one of color schemes the software comes with.



A sample webpage generated using different color schemes.

[Custom color scheme](#)^[24]

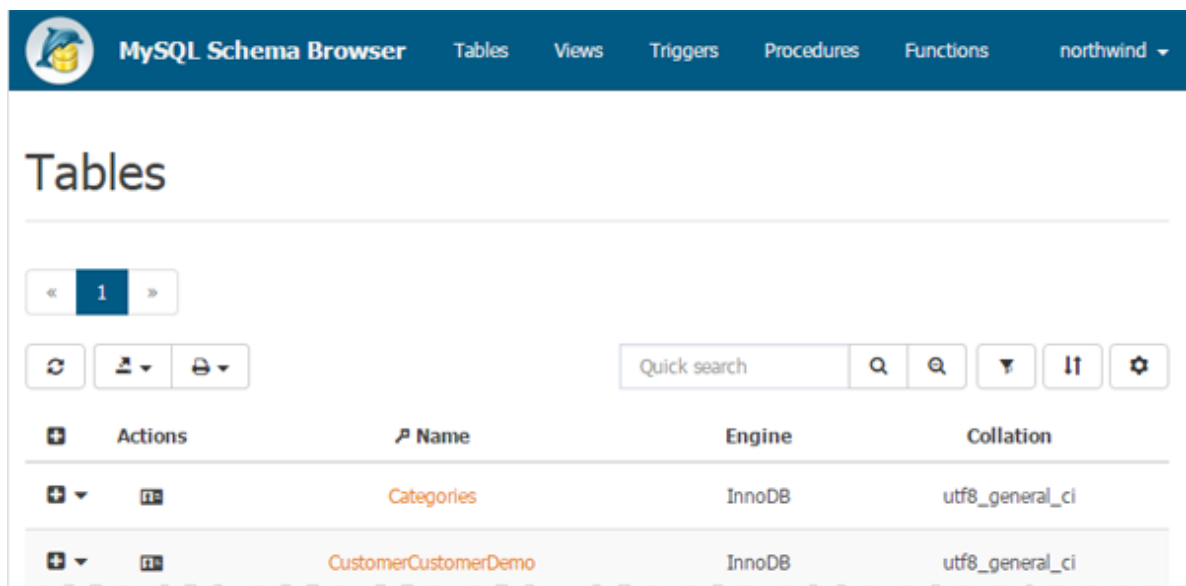
This allows you to provide the application with a specific look and feel defining your own style sheet (or changing an existing one).



A sample webpage generated with custom color scheme.

Header and Footer ²⁴²

These are fragments of HTML code that will be included into all the generated pages. They can be customized within the appropriate window at the [last step of the wizard](#) ²⁷⁵.



A sample webpage with specified header.

Custom Style Sheets ²⁴³

It is also possible to define custom styles to be applied after all pre-defined CSS rules. This can be useful to define such things as size of controls, text alignments, background images, and more. [More about user-defined styles](#) ³²⁵.

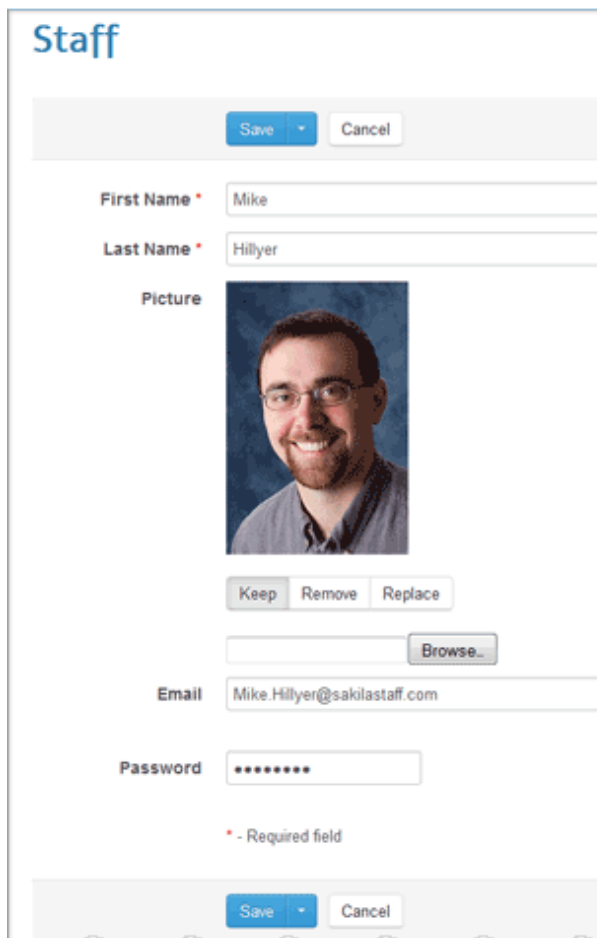
Events^[117]

Events are pieces of code that are run when an action is taken by the user. Such events as [OnBeforePageExecute](#)^[148], [OnCustomRenderColumn](#)^[151], and [OnCustomDrawRow](#)^[174] can be used to change the webpage appearance. Please find some examples of using these and other events in [our demo application](#) and in the **Resources** section at the product home page. Open the [User Javascript](#)^[245] window to define functions to be called from event handlers and other client-side code for your website.

Custom templates^[246]

While all the previously described methods allow you to define colors, fonts, text alignments, and other display attributes of generated webpages, this one helps you to redesign the webpages entirely i.e. define what controls are displayed on a specific form, where they are displayed, and what they display. [Live examples](#).

Standard edit form:

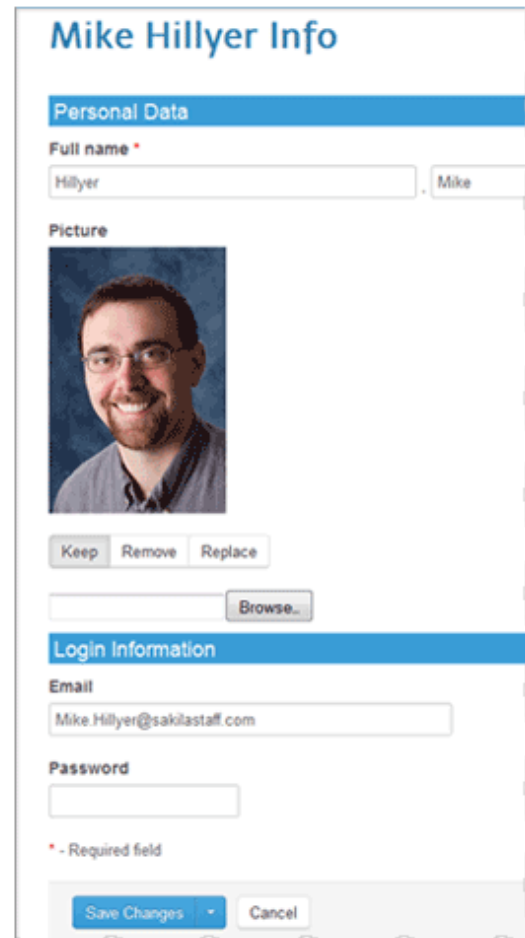


The 'Standard edit form' is titled 'Staff'. It features a 'Save' button with a dropdown arrow and a 'Cancel' button at the top. The form contains the following fields:

- First Name ***: Text input with 'Mike'.
- Last Name ***: Text input with 'Hillyer'.
- Picture**: A placeholder image of a man with glasses and a beard. Below it are 'Keep', 'Remove', and 'Replace' buttons, and a 'Browse...' button.
- Email**: Text input with 'Mike.Hillyer@sakilastaff.com'.
- Password**: Password input field with masked characters '*****'.

A legend at the bottom indicates '* - Required field'. At the bottom of the form are 'Save' and 'Cancel' buttons.

Custom edit form:



The 'Custom edit form' is titled 'Mike Hillyer Info'. It has a blue header bar with the title. The form is divided into two sections:

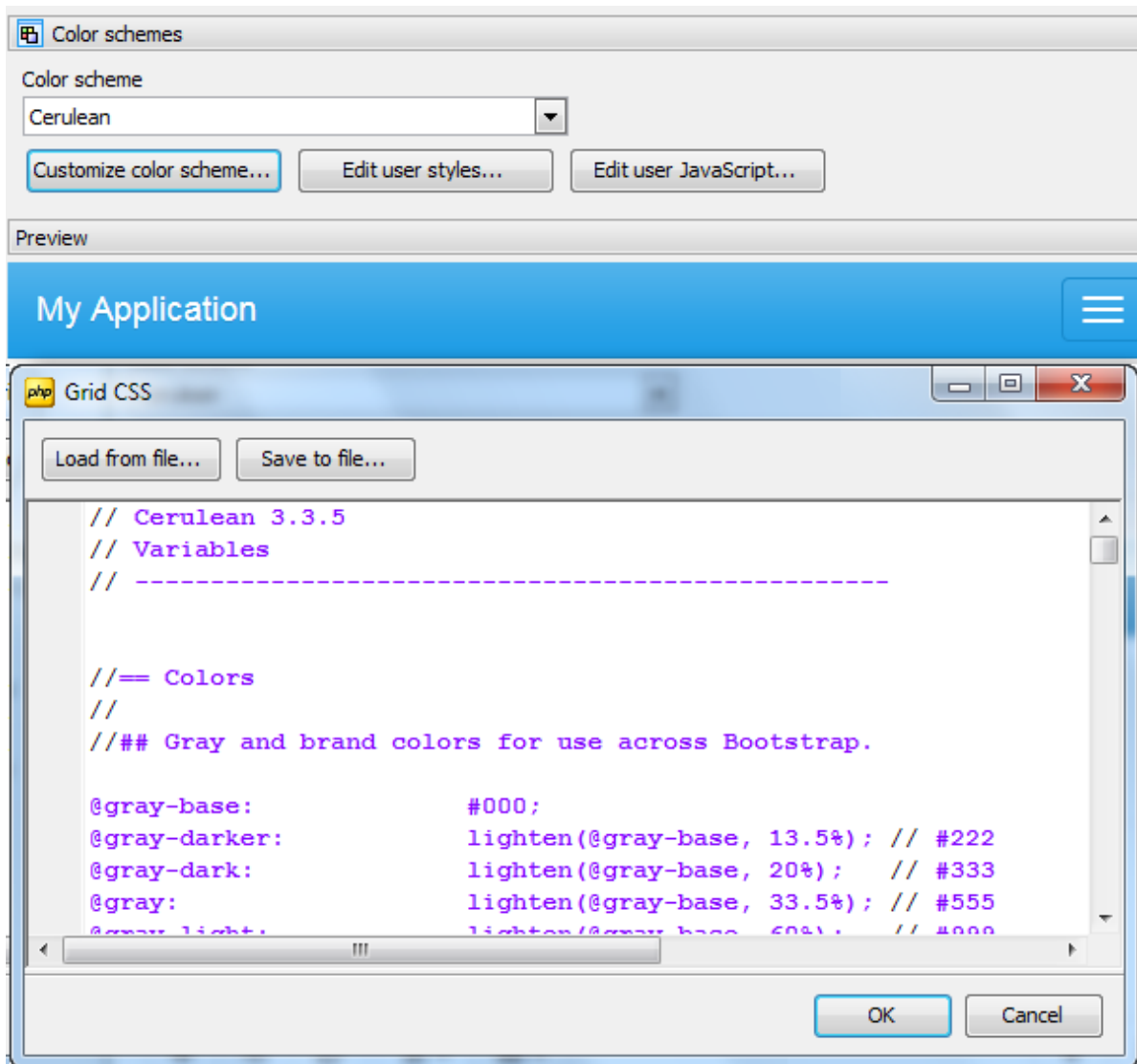
- Personal Data**: Contains a 'Full name *' field with 'Hillyer' and 'Mike' (separated by a dot), a 'Picture' field with the same man's image and 'Keep', 'Remove', 'Replace', and 'Browse...' buttons.
- Login Information**: Contains an 'Email' field with 'Mike.Hillyer@sakilastaff.com' and a 'Password' field with masked characters '*****'.

A legend at the bottom indicates '* - Required field'. At the bottom of the form are 'Save Changes' and 'Cancel' buttons.

7.1 Color schemes

The color scheme of created application contains the information on basic style definitions for all key HTML components: fonts, colors, backgrounds, borders, and others. PostgreSQL PHP Generator comes with a number of built-in color schemes. To use one of them, select the appropriate item from the "Color scheme" drop-down list.

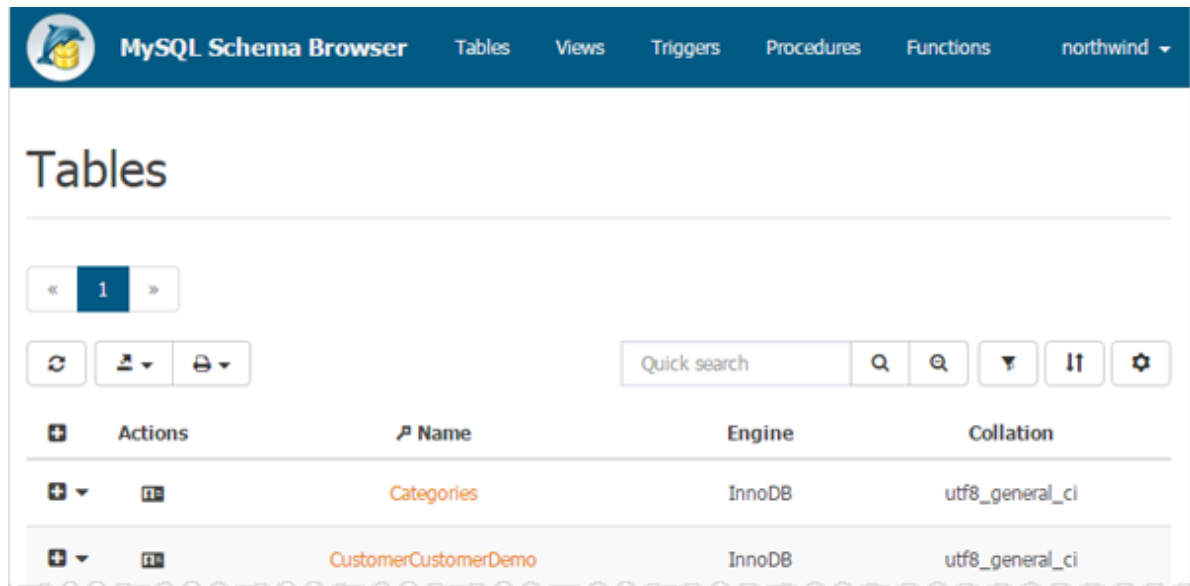
To change default scheme settings according to your needs, click the [Customize color scheme](#) button and modify the necessary variables within the opened window. To learn more about the LESS syntax used in this window, please refer to the [LESS manual](#).



The [Preview](#) window displays current appearance settings applied to a sample webpage.

7.2 Header and Footer

To add an HTML markup to be displayed at the top and at the bottom of each page of the generated web application, enter the appropriate code in the [Header](#) and [Footer](#) text boxes accordingly.



Example: To add the header like one on the screen above, paste the following text in the Header edit box:

```
<span class="navbar-brand">
  <span>
    
  </span>
</span>
<span class="navbar-brand">
  <span class="hidden-xs"><strong>MySQL Schema Browser</strong></span>
</span>
```


7.3 User-defined styles

The ability to use [custom styles](#) allows you to set up such parameters of web pages as size of controls, text alignment, and more. To define a custom style sheet that will be applied after all the pre-defined rules, click the "Edit user styles" button and enter the rules you need manually or load them from a file. [Less syntax](#)^[325] is allowed.

You can also define custom style sheets for export to PDF as well as for the print version of generated pages. These styles are saved to `user_print.css` and `user_pdf.css` files accordingly.

To determine a selector to apply the styles for, use any available inspect HTML tools such as [default Firefox developer tools](#), [Firebug](#), [Chrome developer tools](#), [Internet Explorer developer tools](#), and so on. Among other things these tools allow you to view the current element's attributes and applied styles and experiment with new ones.

Example 1

The code below is used in [our online demo](#) to display the winning team and the losing team scores according to the current theme. The 'win-score' and 'loss-score' classes are defined as follows:

```
.base-score {
    font-size: 1.4em;
    font-weight: bold;
}

.win-score {
    &:extend(.base-score);
    color: @brand-danger;
}

.loss-score {
    &:extend(.base-score);
}
```

The conditional formatting is specified in the `OnExtendedCustomDrawRow` event as follows:

```
if ($rowData['home_team_score'] > $rowData['away_team_score']) {
    $cellClasses['home_team_score'] = 'win-score';
    $cellClasses['away_team_score'] = 'loss-score';
}
else {
    $cellClasses['home_team_score'] = 'loss-score';
    $cellClasses['away_team_score'] = 'win-score';
}
```

The screenshot below displays the Game list page in the Cyborg theme:

Home team	Home Score	Away Score	Away team
Los Angeles Lakers	116	103	Los Angeles Clippers
Miami Heat	107	95	Chicago Bulls
Indiana Pacers	97	87	Orlando Magic

The next one shows the same page in the Facebook theme:

Home team	Home Score	Away Score	Away team
Los Angeles Lakers	116	103	Los Angeles Clippers
Miami Heat	107	95	Chicago Bulls
Indiana Pacers	97	87	Orlando Magic
Cleveland Cavaliers	98	94	Brooklyn Nets

Example 2

To limit maximum width of Date and DateTime pickers in all Edit and Insert forms by 300 pixels, use the following rule:

```
div.pgui-date-time-edit {
    max-width: 300px;
}
```


7.4 User JavaScript

The ability to use custom JavaScript allows you to specify the client-side code that will be included into all the generated pages. This is the optimal place to define, for example, JavaScript functions that are called from multiple event handlers (in order to avoid duplicate code).

Example

Suppose we want to fill the content of 'title' control depending of the value selected as 'sex' on Insert and Edit forms. Of course we can define the [OnInsertFormEditorValueChanged](#)^[140] and [OnEditFormEditorValueChanged](#)^[142] with the same code, but it's easier to define a function in the User Javascript window and call it in these event handlers.

User Javascript:

```
function fillRelatedTitles(sender, editors)
{
    if (sender.getValue() == 'Male') {
        editors['title'].removeItem('Ms', 'Ms');
        editors['title'].removeItem('Mrs', 'Mrs');
        editors['title'].addItem('Mr', 'Mr');
    }
    else {
        editors['title'].removeItem('Mr');
        editors['title'].addItem('Ms', 'Ms');
        editors['title'].addItem('Mrs', 'Mrs');
    }
}
```

OnInsertFormEditorValueChanged/OnEditFormEditorValueChanged event handlers:

```
if (sender.getFieldName() == 'sex')
{
    fillRelatedTitles(sender, editors);
}
```


7.5 Custom templates

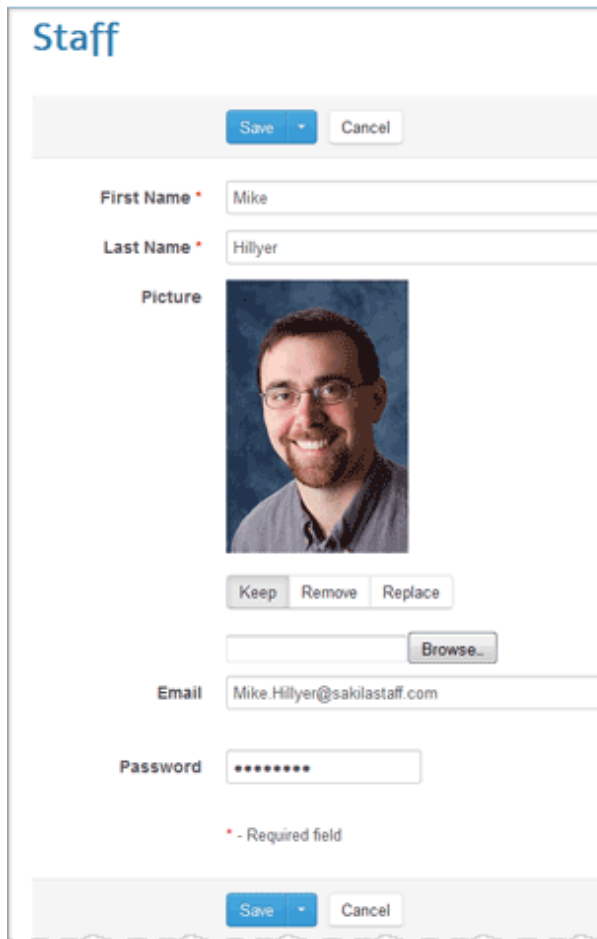
Applications created by PostgreSQL PHP Generator use [Smarty](#), a typical representative of the web template engines family. So in order to redesign a webpage, you need to change the appropriate template. All templates used by the software are stored in the *components/templates* subfolder of the output directory. Each folder in this catalog contains template files to be used by the application in different cases: on editing, printing, inserting, etc.

Using custom templates

To change the webpage appearance with custom templates, follow these steps:

- Create a new template. Custom templates must be stored under the *components/templates/custom_templates* folder which contents is not changing on re-generation of the scripts. You must create this folder manually.
- Instruct PHP Generator to use this template for the selected webpage with the *OnGetCustomTemplate* event.
- Customize the template according to your needs and preferences.

The detailed description of such customization is given in the [appropriate article](#). [Another article](#) learns how to add a chart component to a webpage using the way described above. [The demo application](#) also shows some additional examples of using custom templates.

Standard edit form:



Staff

Save Cancel

First Name * Mike

Last Name * Hillyer

Picture



Keep Remove Replace

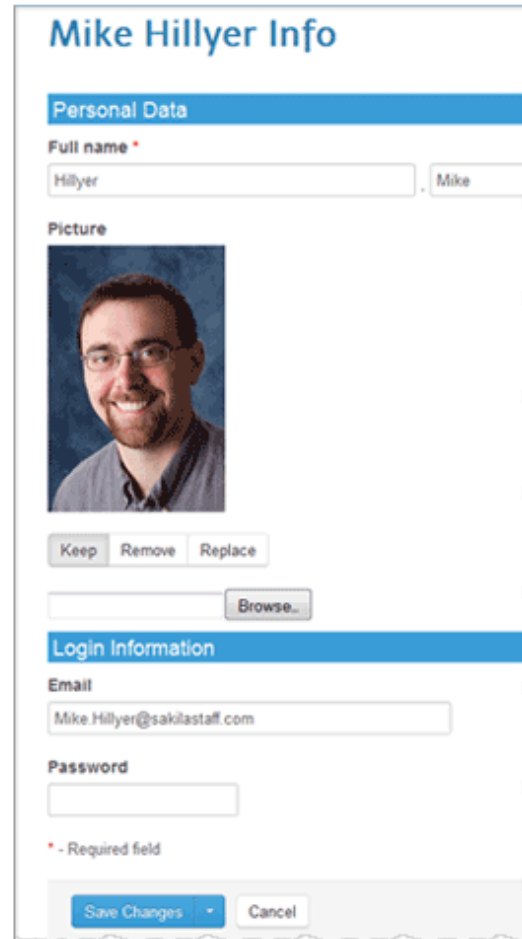
Browse...

Email Mike.Hillyer@sakilastaff.com

Password *****

* - Required field

Save Cancel


Custom edit form:


Mike Hillyer Info

Personal Data

Full name * Hillyer Mike

Picture



Keep Remove Replace

Browse...

Login Information

Email Mike.Hillyer@sakilastaff.com

Password

* - Required field

Save Changes Cancel

Changing existing templates

You can modify generated template files directly. The example of supplying of generated pages with look and feel of an existing website illustrates this method in the [appropriate article](#). Since all the templates are rewritten each time you generate an application, you have to add modified templates to the [Excluded_files](#)^[275] list to prevent them from overwriting.

8 Security settings

Check the [Enable security](#) option to supply the generated application with a login page to request user name and password when someone attempts to access your website. PostgreSQL PHP Generator provides you with the following authorization types:

- [Hard-coded authorization](#)^[250]
Select this option to incorporate the authorization procedure directly into the PHP code of the generated application.
- [Table-based authorization](#)^[252]
Select this option to authorize a user of the web application by a login/password combination stored in a database table. This kind of authorization also provides support for user self-registration and other [email-based features](#)^[256].
- [Database server authorization](#)^[258]
Select this option to accomplish the authorization by PostgreSQL server.
- [User-defined authorization](#)^[259]
Select this option to use a custom function to check user identity.

See also: [Security_events](#)^[261], [Google_reCAPTCHA](#)^[267], [Record-level_security](#)^[268], [User_permissions](#)^[269]

[Set the home page as a startup page](#)

Use this option to allow a user to be redirected to the home page after logging in, otherwise he will be redirected to the first page of an application he has access to.


[Enable guest access](#)

Turn this options ON to allow unauthorized users to access all or certain pages of your website.

[Enable inactivity timeout](#) and its period

You can specify the maximum amount of time (in seconds) after that a user will be automatically logged out from the application if he/she does not perform any action during this period. The appropriate message will be displayed to the user and he/she will have to log in again to continue his/her work with the website. Inactivity timeout is supported for all kinds of authorization. When the timeout is reached, the [OnBeforeLogout](#)^[263] event fires.

Security options

 Security options

☒ Enable security

Authentication type
Table-based authentication

Users table
Table name
phpgen_users
Create new table...

Mandatory fields
User id
user_id
User name
user_name
Password
user_password

Email-based features
☒ Enable self-registration
☒ Enable password recovering
Outgoing mail settings...

Additional users table fields
Email
user_email
Token
user_token
User status
user_status

Advanced options
Password encryption
SHA256
☒ Enable guest access
☒ Allow users to change their passwords
☒ Enable inactivity timeout
Log out after
5 minutes of inactivity

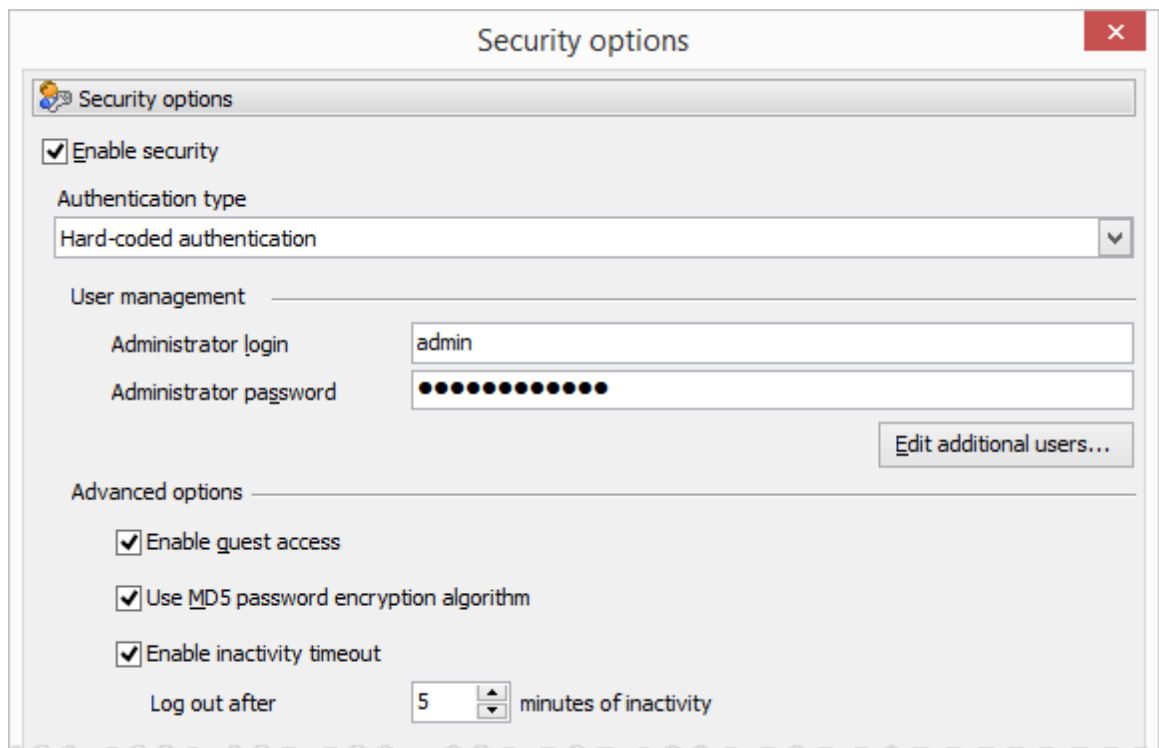
Security events...
Google reCAPTCHA...
Record-level security...
Manage permissions...

OK
Cancel

8.1 Hard-coded authorization

On using this kind of authorization all user accounts are stored in generated .php files. As any operation related to the user management requires re-generation and re-uploading of the application, it is not recommended to use this kind of authorization for real websites except the simplest ones.

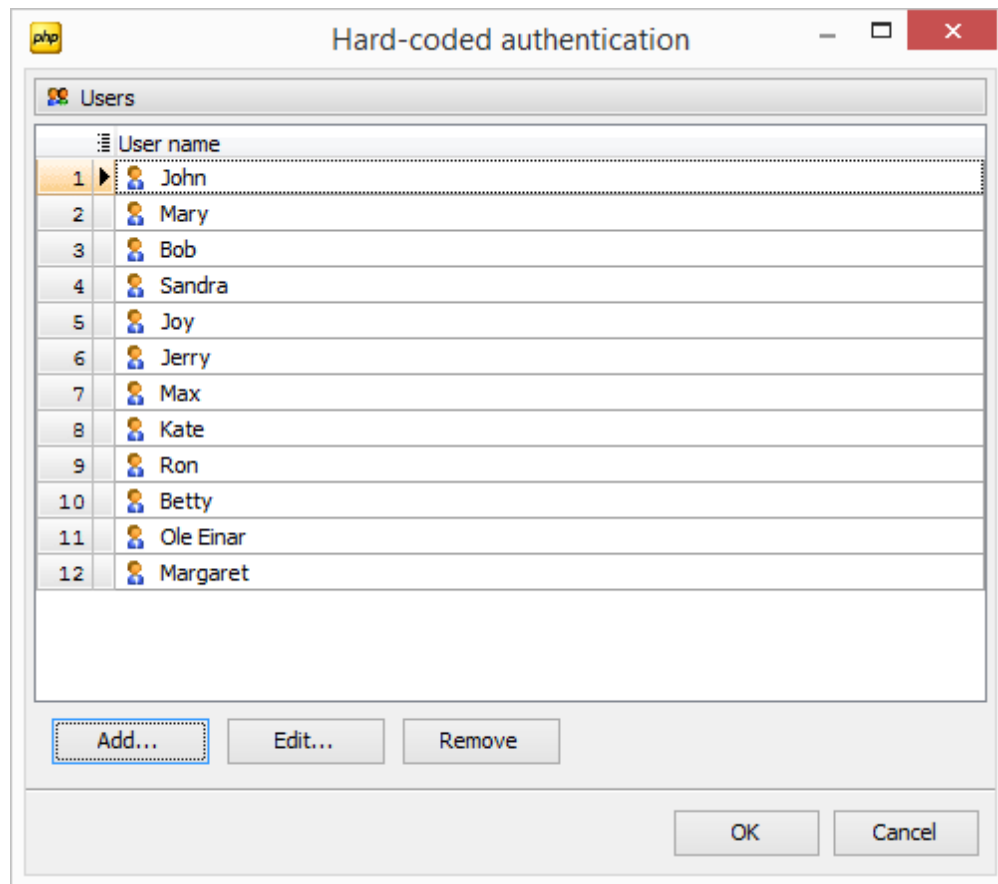
To create the administrator user, fill in the [Administrator login](#) and [Administrator password](#) fields. Check the [Enable guest access](#) option to allow an anonymous user to access the generated app without completing the authentication procedure. Turn ON the [Enable MD5 password encryption](#) checkbox to store user passwords encrypted with the MD5 algorithm (recommended).



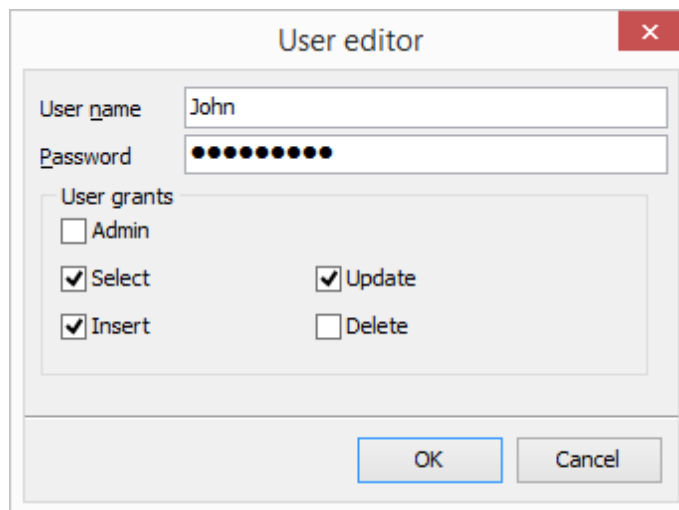
The screenshot shows a 'Security options' dialog box with a red close button in the top right corner. The dialog has a title bar with a small icon and the text 'Security options'. Inside, there are several sections:

- Enable security:** A checkbox that is checked.
- Authentication type:** A dropdown menu currently set to 'Hard-coded authentication'.
- User management:** A section with two input fields: 'Administrator login' containing the text 'admin', and 'Administrator password' containing a series of black dots. To the right of these fields is a button labeled 'Edit additional users...'.
- Advanced options:** A section with three checked checkboxes: 'Enable guest access', 'Use MD5 password encryption algorithm', and 'Enable inactivity timeout'. Below these is a 'Log out after' field with a spinner set to '5' and the text 'minutes of inactivity'.

To manage additional users, click the [Edit additional users](#) button, then use Add/Edit/Remove buttons or the user grid's popup menu.



When creating/editing a user, provide its user name and password and check the necessary options to grant application-level permissions to this user. To learn how to specify page-level permissions, see the [User permissions](#) ²⁶⁹ section.



8.2 Table-based authorization

This kind of authorization means that user accounts are stored a database table. The table may be prepared earlier or created directly within PHP Generator.

The screenshot shows a configuration window titled "Table-based user authorization". It contains several dropdown menus and a checkbox. The "Users table" dropdown is set to "phpgen_users". The "Login field" dropdown is set to "user_name". The "Password field" dropdown is set to "user_password". The "User ID field" dropdown is set to "user_id". The "Password encryption" dropdown is set to "None". There is a checkbox labeled "Enable guest access" which is checked. A button labeled "Create new table..." is located at the bottom right.

Table structure

The basic table structure is as follows. Some additional columns should be added for support [email-based features](#)^[256].

Default name	Data type	Description
user_id	integer	ID of the user. This column must be defined as autoincrement for support user self-registration ^[256] .
user_name	character varying(255)	The login of the user
user_password	character varying(255)	The password of the user

Using an existing table

Specify the table name in the [Users table](#) fields, then select columns where user names, passwords, and IDs are stored. If passwords are encrypted, select the corresponding algorithm from the [Password Encryption](#) drop down list or enter the algorithm manually (if passwords are encrypted with a [hash algorithm](#) missing in the list like SHA512). It is also possible to [provide custom functions](#)^[253] for encrypting and verifying the passwords.

Using a new table

Click the [Create new table...](#) button, enter desired table and column names, then click OK. A new table will be created in the database.

Create users table	
Users table	
Table name	phpgen_users
Mandatory fields	
User id	user_id
User name	user_name
User password	user_password
Fields to support email-based features	
Enable	<input checked="" type="checkbox"/>
User email	user_email
User token	user_token
User status	user_status
<div>OK Cancel</div>	

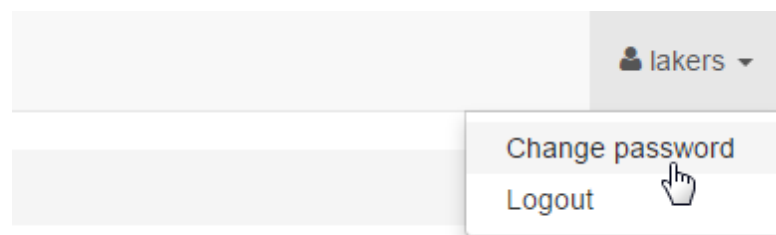
To be able to log in to your website, you should create the first user (i.e. the first record in this table) with the Admin privilege manually (for example, you can do it with our PostgreSQL Maestro). Don't forget that the password should be encrypted with the algorithm you selected. Other users can be created online via the [Administration Panel](#) that is generated automatically.

Customizing permissions

By default all users have the SELECT privilege granted at the application level via the PUBLIC group. Open the [Permission Manager](#)^[269] window to setup permission storage and/or customize the permissions.

Allow users to change their passwords

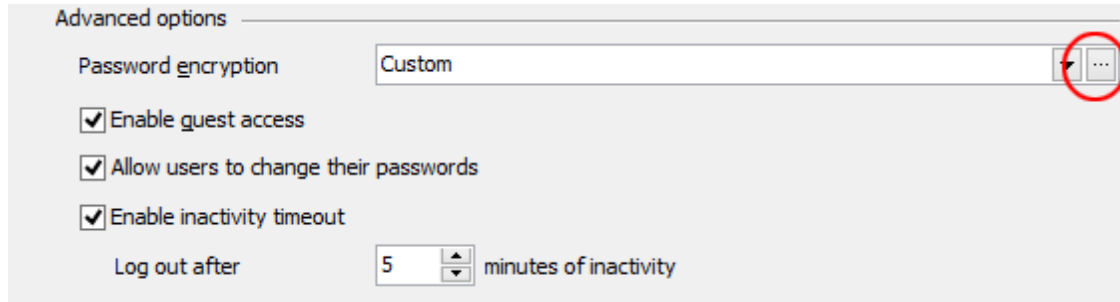
Turn this option ON to allow your end-users change their passwords themselves (the appropriate button is located in the top right corner of the window). Please note that this functionality required storing user permissions in a database table as described [here](#)^[269].



8.2.1 Custom Password Encryption

PostgreSQL PHP Generator comes with built-in support of a number of reliable and strong algorithms for password encryption; however, you can define your own functions for encrypting and verifying the passwords. This allows you to use any encryption algorithm API or library you like.

To define a custom password encryption algorithm, select "Custom" in the Password Encryption field, then press the ellipsis button and provide two PHP functions for encrypting and verifying the password accordingly.

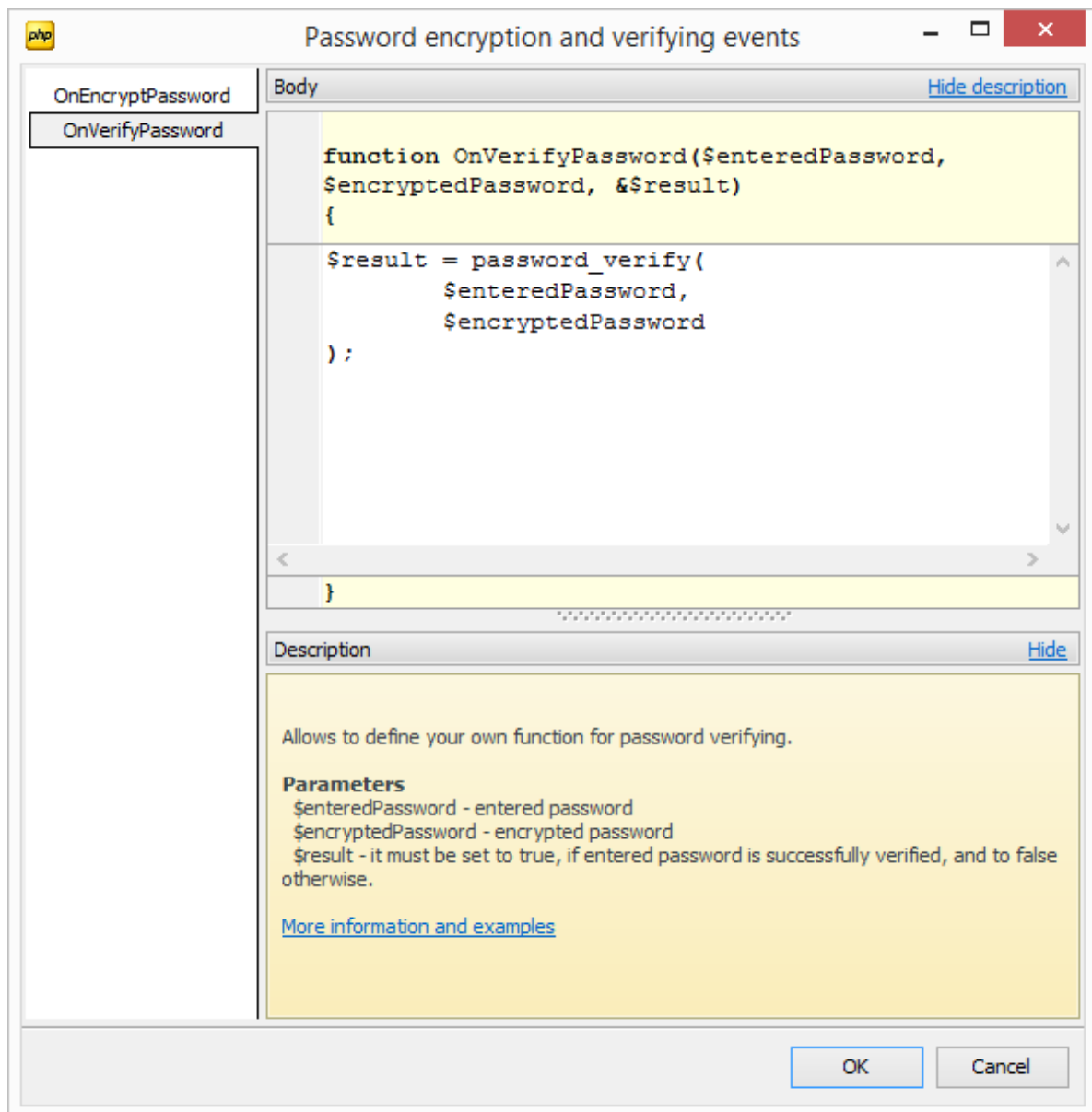


```
function OnEncryptPassword($password, &$result)
```

This function accepts an unencrypted password in the `$password` parameter and returns the encrypted password in the `$result` parameter that is passed by reference.

```
function OnVerifyPassword($enteredPassword, $encryptedPassword, &$result)
```

This function accepts an unencrypted password entered by the user and an encrypted password stored in the database. The `$result` parameter must be set to true if the entered password has been verified successfully and to false otherwise. By default this function encrypts the entered password with the `OnEncryptPassword` function call (see above) and compares the result with the encrypted password, so you should define the `OnVerifyPassword` function explicitly only if the default behavior is not suitable for your needs.



Example

The example below shows how it is possible to use the [PHP Native password hashing API](#) for encrypting and verifying the password.

The OnEncryptPassword function should be defined as follows:

```
$result = password_hash($password, PASSWORD_DEFAULT);
```

As we need to call a separate function for the password verification, the OnVerifyPassword function should be provided as well:

```
$result = password_verify($enteredPassword, $encryptedPassword);
```


8.2.2 Email-based features

Some security-related features provided by PostgreSQL PHP Generator like self-registration and password recovering are based on sending email messages to users of generated sites. This chapter explains how these features work and what is required from the site developer for their support.

Users table extension

To support email-based features, user table must contain the following columns (besides columns for storing user names, passwords, and IDs that are required to support table-based authentication basically):

Default name	Data type	Description
user_email	character varying(255)	Email address of the user
user_token	character varying(255)	Internal data for verification and password recovering
user_status	integer	Status of the user. Possible values are as follows: 0 - registered user 1 - user awaiting verification 2 - user requested password reset

Enable self-registration

If this option is turned ON, users can register in the generated application.

Enable password recovering

If this option is turned ON, users are allowed to reset their passwords.

Customizing email messages

Email messages to be sent to the user on the registration and password recovering can be customized with the [OnGetCustomTemplate](#)^[159] event.

Outgoing mail settings

A set of settings to be used to send emails from the generated website. These settings can be also specified in the [Project Options](#)^[235] dialog. See also [Setting up common SMTP servers](#)^[326].

[-] Common	
Mailer	Sendmail
From mail	admin@mysite.com
From name	Site Admin
[-] SMTP options	
Host	
Port	25
<input type="checkbox"/> Use authentication	<input type="checkbox"/>
Username	
Password	
Encryption	None

Events

A number of events help you to control user registration and password recovering processes in the most flexible way possible. The following events are currently available:

- [OnBeforeUserRegistration](#)^[265]
- [OnAfterUserRegistration](#)^[265]
- [OnPasswordResetRequest](#)^[265]
- [OnPasswordResetComplete](#)^[266]

8.3 Database server authorization

This kind of authorization means that each user in your application should correspond to a user at the database server. To allow access without entering a password, you need to create an additional user account at the database server and specify its username/password in the corresponding fields.

The screenshot shows a configuration window for database security. At the top, there is a checked checkbox labeled "Enable security". Below it, the "Authentication type" is set to "Database server authentication" in a dropdown menu. A section titled "Advanced options" contains three more checked checkboxes: "Enable guest access", "Enable inactivity timeout", and "Log out after". The "Enable guest access" checkbox is followed by two input fields: "Guest login" with the value "guest" and "Guest password" with four masked dots. The "Log out after" checkbox is followed by a spinner box set to "5" and the text "minutes of inactivity".

User permissions should also be managed with the server facilities (GRANT and REVOKE statements). You can align user interface to real permissions granted at the server side with the [OnGetCustomPagePermissions](#)^[19] event.

8.4 User-defined authorization

This kind of authorization allows you to define your own custom authentication. This feature requires you to define a function that takes entered username and password and returns true or false depending on their validity. This means you can use any possible storage for user credentials and any encryption algorithm you like.

The screenshot shows two overlapping windows from a software application. The background window is titled "Security options" and has a close button (X) in the top right corner. It contains the following elements:

- A tab labeled "Security options" with a small icon.
- A checked checkbox labeled "Enable security".
- A dropdown menu for "Authentication type" currently set to "User-defined authentication".
- A section titled "Advanced options" with a horizontal line separator.
- Text explaining: "This kind of authentication requires you to define your own function that takes entered username and password and returns true or false depending on their validity."
- A button labeled "Edit function code..."
- A checked checkbox labeled "Enable inactivity timeout".
- A label "Log out after" followed by a spinner box set to "5" and the text "minutes of inactivity".

The foreground window is titled "CheckUserIdentity" and has a PHP icon in the top left corner. It contains:

- A tab labeled "Body" with a "Hide description" link.
- A code editor with the following PHP code:

```
function CheckUserIdentity($username, $password, &$result)
{
    if ($username == $password)
        $result = strlen($password) >= 5;
    else
        $result = false;
}
```
- A section titled "Description" with a "Hide" link.
- Text: "Define a custom function to check user identity."
- A section titled "Parameters" with the following text:

\$username, \$password - username/password combination to check
\$result - it must be set to true, if username/password combination is valid, and to false otherwise.
- Buttons for "OK" and "Cancel" at the bottom right.

Example:

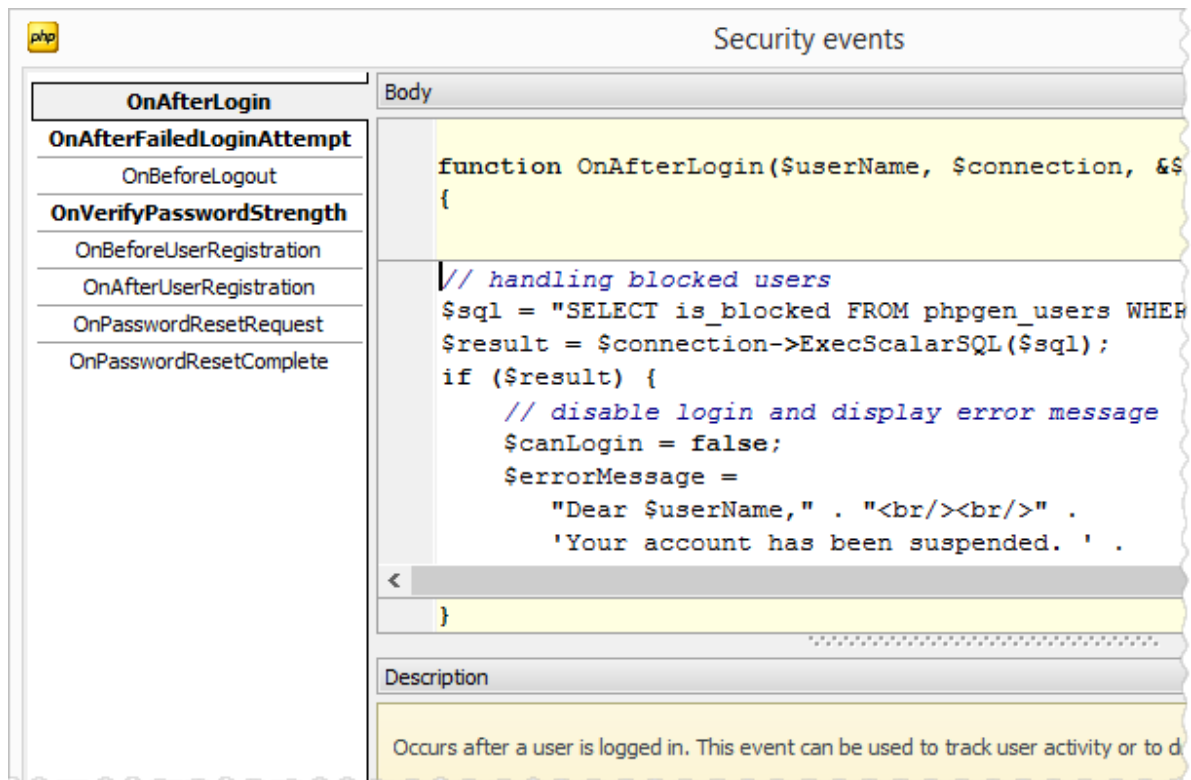
To check that the password is equal to the user name and password contains 5 or more symbols, place the following code to the body of the *CheckUserIdentity* function. This condition means that the 'John/John' or 'John/Michael' pair will be declined while the 'Michael/Michael' or 'Gloria/Gloria' login/password combination will be accepted.

```
if ($username == $password)
    $result = strlen($password) >= 5;
else
    $result = false;
```

See also: [OnGetCustomPagePermissions](#)^[191], [OnGetCustomRecordPermissions](#)^[192].

8.5 Security Events

Server-side events are fragments of **PHP** code executed at the appointed time. It is possible to use [Server Side API](#)^[304] and [environment variables](#)^[195] within these event handlers.



To specify a handler for a security event, open the [Security options](#)^[248] window on the last wizard step and use the [Security events](#) button. The complete list of currently available security events is as follows.

- [OnAfterLogin](#)^[262]
- [OnAfterFailedLoginAttempt](#)^[262]
- [OnBeforeLogout](#)^[263]
- [OnVerifyPasswordStrength](#)^[264]
- [OnBeforeUserRegistration event](#)^[265]
- [OnAfterUserRegistration event](#)^[265]
- [OnPasswordResetRequest event](#)^[265]
- [OnPasswordResetComplete event](#)^[266]

See also: [Client side events](#)^[137], [Server Side Page Events](#)^[147], [Application-level events](#)^[117]

8.5.1 OnAfterLogin

This event occurs after a user is successfully logged in into the application.

Signature:

```
function OnAfterLogin ($userName, $connection, &$canLogin, &$errorMessage)
```

Parameters:

\$userName	The name of the logged user.
\$connection	An instance of the EngConnection ³¹⁸ class.
\$canLogin	Specifies whether a user can login (true by default). You can set the value of this parameter to false to disable login even for a user who provided valid credentials.
\$errorMessage	A message to be displayed when valid credentials are provided, but \$canLogin == false.

Example 1:

Among other things this event can be used for tracking the user activity. The code snippet below inserts a new record containing user name and the login time into the 'log' table.

```
$connection->ExecSQL("INSERT INTO log(user_name, log_date)
VALUES ('$userName', CURRENT_TIMESTAMP)");
```

Example 2:

The following code allows to redirect users to home page after login.

```
header('Location: index.php');
exit;
```

Example 3:

The following code locks user account if number of failed login attempts exceeded otherwise the counter of failed login attempts is reset.

```
$sql = "SELECT failed_login_attempts FROM phpgen_users WHERE user_name='$userName'";
$failedLoginAttempts = $connection->ExecScalarSQL($sql);
if ($failedLoginAttempts >= 3) {
    $canLogin = false;
    $errorMessage =
        "Dear $userName, your account is locked due to too many failed login attempts.";
} else {
    $sql = "UPDATE phpgen_users SET failed_login_attempts = 0 WHERE user_name='$userName'";
    $connection->ExecSQL($sql);
}
```

See also: OnAfterFailedLoginAttempt and [OnBeforeLogout](#)²⁶³ events

8.5.2 OnAfterFailedLoginAttempt

This event occurs after a failed login attempt. It allows you to trace failed login attempts. The event is usually used in conjunction with [OnAfterLogin](#)²⁶² event. For example, you can limit the number of failed login attempts per user and to lock user account after a number of failed login attempts.

Signature:

```
function OnAfterFailedLoginAttempt ($userName, $connection, $errorMessage)
```

Parameters:

\$userName	The name of the user.
\$connection	An instance of the EngConnection ^[318] class.
\$errorMessage	A message to be displayed when valid credentials are provided, but \$canLogin == false.

Example:

The following code locks user accounts after three failed login attempts.

```
// Check if user exists
$sql = "SELECT count(*) FROM phpgen_users WHERE user_name='$userName'";
$userExists = $connection->ExecScalarSQL($sql);
if ($userExists == 0) {
    exit;
}

// Retrieve a number of previous failed login attempts
$sql = "SELECT failed_login_attempts FROM phpgen_users WHERE user_name='$userName'";
$failedLoginAttempts = $connection->ExecScalarSQL($sql);

// Add a current failed login attempt
$failedLoginAttempts++;

// Display message based on a number of failed login attempts
if ($failedLoginAttempts == 2) {
    $errorMessage = 'You have one attempt left before your account will be locked.';
} elseif ($failedLoginAttempts == 3) {
    $errorMessage = 'Too many failed login attempts. Your account has been locked.';
} elseif ($failedLoginAttempts > 3) {
    $errorMessage =
        "Dear $userName, your account is locked due to too many failed login attempts. " .
        'Please contact our support team.';
}

// Update a number of failed login attempts in users table
if ($failedLoginAttempts <= 3) {
    $sql =
        "UPDATE phpgen_users " .
        "SET failed_login_attempts = $failedLoginAttempts " .
        "WHERE user_name='$userName'";
    $connection->ExecSQL($sql);
}
```

8.5.3 OnBeforeLogout

This event occurs just before a user logged out from the application by pressing the Logout button or when the [inactivity timeout](#)^[248] is reached.

Signature:

```
function OnBeforeLogout ($userName, $connection)
```


Parameters:

\$userName	The name of the logged user.
\$connection	An instance of the EngConnection class ^[318] .

Example

Among other things this event can be used for tracking the user activity. The code snippet below inserts a new record containing user name and the action into the 'activity_log' table.

```
$connection->ExecSQL(
    "INSERT INTO activity_log (user_name, action) values ('$userName', 'logout')"
);
```

See also: [OnAfterLogin](#)^[262] and OnAfterFailedLoginAttempt events

8.5.4 OnVerifyPasswordStrength

This event allows you to verify an entered password. The password is accepted only when it meets specified complexity requirements.

Signature:

```
function OnVerifyPasswordStrength ($password, &$amp;result, &$amp;passwordRuleMessage)
```

Parameters:

\$password	The entered password.
\$result	It must be set to true if entered password meets complexity requirements or to false otherwise.
\$passwordRuleMessage	The message to be displayed if entered password does not meet complexity requirements

Example:

The following code accepts only passwords with 8 and more characters in length, including at least one upper case letter, one lower case letter, one number and one special character.

```
$atleastOneUppercaseRule    = preg_match('@[A-Z]@', $password);
$atleastOneLowercaseRule   = preg_match('@[a-z]@', $password);
$atleastOneNumberRule      = preg_match('@[0-9]@', $password);
$atleastOneSpecialCharRule = preg_match('@^[^w]@', $password);

$result =
    strlen($password) >= 8 &&
    $atleastOneUppercaseRule &&
    $atleastOneLowercaseRule &&
    $atleastOneNumberRule &&
    $atleastOneSpecialCharRule;

$passwordRuleMessage =
    'Password must be at least 8 characters in length ' .
    'and must include at least one upper case letter, ' .
    'one lower case letter, one number, and one special character.';
```


8.5.5 OnBeforeUserRegistration event

This event occurs before a user is registered.

Signature:

```
function OnBeforeUserRegistration($userName, $email, $password,  
                                &$allowRegistration, &$errorMessage)
```

Parameters:

\$userName	The name of the user.
\$email	The email of the user.
\$password	The password of the user.
\$allowRegistration	A parameter to indicate whether user registration is allowed. Set \$allowRegistration to false to reject the new user. Default value is true.
\$errorMessage	A message to be displayed if \$allowRegistration == false.

Example

The code snippet below demonstrates how to deny registrations for users with a mailinator.com email address.

```
$emailDomainName = substr($email, strrpos($email, '@') + 1);  
if ($emailDomainName == 'mailinator.com') {  
    $allowRegistration = false;  
    $errorMessage = "It is forbidden to register with mailinator's emails";  
}
```

8.5.6 OnAfterUserRegistration event

This event occurs after a user is registered.

Signature:

```
function OnAfterUserRegistration ($userName, $email)
```

Parameters:

\$userName	The name of the user.
\$email	The email of the user.

Example

The code snippet below [sends an email](#) ³²² to the site administrator when a new user is registered.

```
$messageBody = "A new user $userName ($email) is registered";  
sendMessage('admin@example.com', 'A new user just registered', $messageBody);
```

8.5.7 OnPasswordResetRequest event

This event occurs after a user requested to reset his password.

Signature:

```
function OnPasswordResetRequest ($userName, $email)
```


Parameters:

\$userName	The name of the user.
\$email	The email of the user.

Example

The code snippet below shows how to add a record to a log table when a user requested to reset his password.

```
$connection = createConnection323();  
  
$sqlTemplate =  
    "INSERT INTO user_log (user_name, action_type, action_time) " .  
    "VALUES ('%s', '%s', '%s')";  
  
$sql = sprintf($sqlTemplate, $userName, 'password_reset_request',  
    date('Y-m-d H:i:s'));  
  
$connection->Connect();  
$connection->ExecSQL($sql);
```

8.5.8 OnPasswordResetComplete event

This event occurs after a user resets his password.

Signature:

```
function OnPasswordResetComplete ($userName, $email)
```

Parameters:

\$userName	The name of the user.
\$email	The email of the user.

Example

The code snippet below shows how to add a record to a log table when a user resets his password.

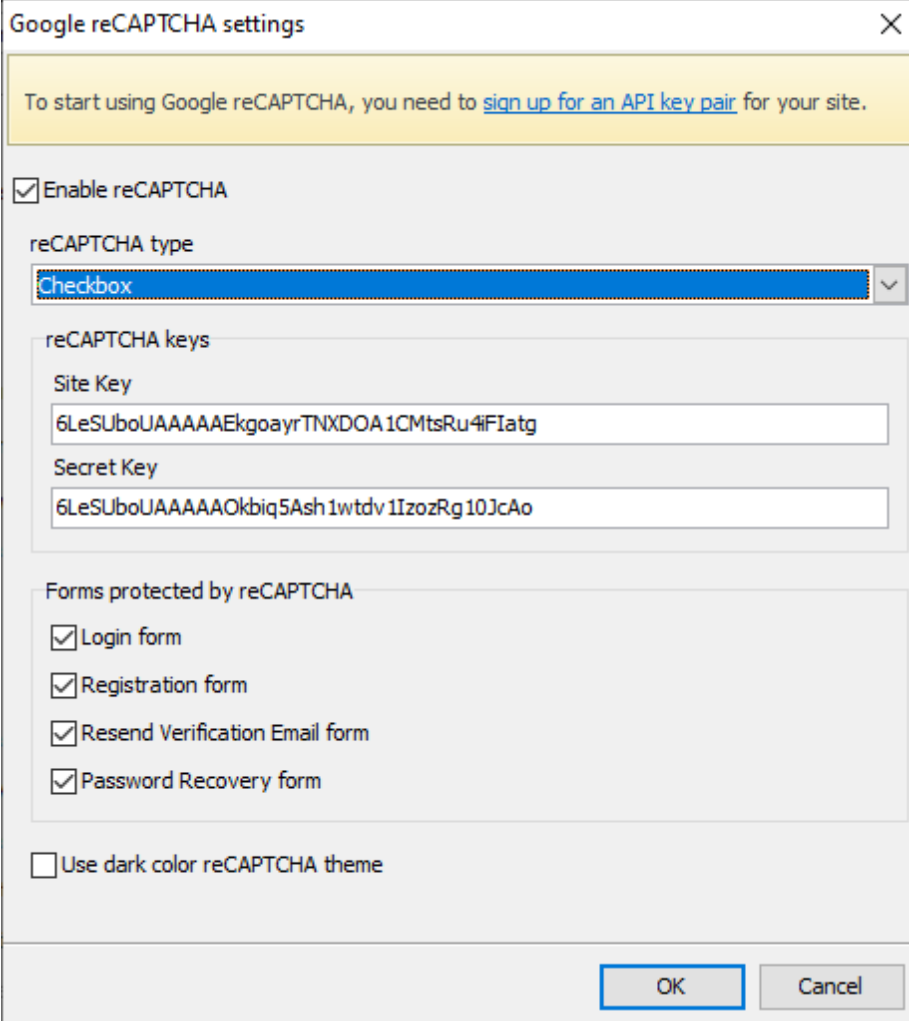
```
$connection = createConnection323();  
  
$sqlTemplate =  
    "INSERT INTO user_log (user_name, action_type, action_time) " .  
    "VALUES ('%s', '%s', '%s')";  
  
$sql = sprintf($sqlTemplate, $userName, 'password_reset_complete',  
    date('Y-m-d H:i:s'));  
  
$connection->Connect();  
$connection->ExecSQL($sql);
```


8.6 Google reCAPTCHA

PostgreSQL PHP Generator allows you to protect your site from spam and abuse with [Google reCAPTCHA service](#). To start using it, you need to [sign up](#) for an API key pair (site key and secret key) for your site.

To enable reCAPTCHA switch ON the appropriate option, choose the type of reCAPTCHA you want to use (Checkbox (I'm not a robot) and Invisible reCAPTCHAs are supported for now),

then provide reCAPTCHA keys in the appropriate edit boxes, and select the forms you want to protect. It is possible also to use dark color reCAPTCHA's theme.



The screenshot shows a 'Google reCAPTCHA settings' dialog box. At the top, a yellow banner states: 'To start using Google reCAPTCHA, you need to [sign up for an API key pair](#) for your site.' Below this, the 'Enable reCAPTCHA' checkbox is checked. The 'reCAPTCHA type' dropdown menu is set to 'Checkbox'. Under 'reCAPTCHA keys', the 'Site Key' is '6LeSuboUAAAAAEkgoayrTNXDOA1CMtsRu4FIatg' and the 'Secret Key' is '6LeSuboUAAAAAOkbiq5Ash1wtdv1IzozRg10JcAo'. In the 'Forms protected by reCAPTCHA' section, four checkboxes are checked: 'Login form', 'Registration form', 'Resend Verification Email form', and 'Password Recovery form'. The 'Use dark color reCAPTCHA theme' checkbox is unchecked. At the bottom right are 'OK' and 'Cancel' buttons.

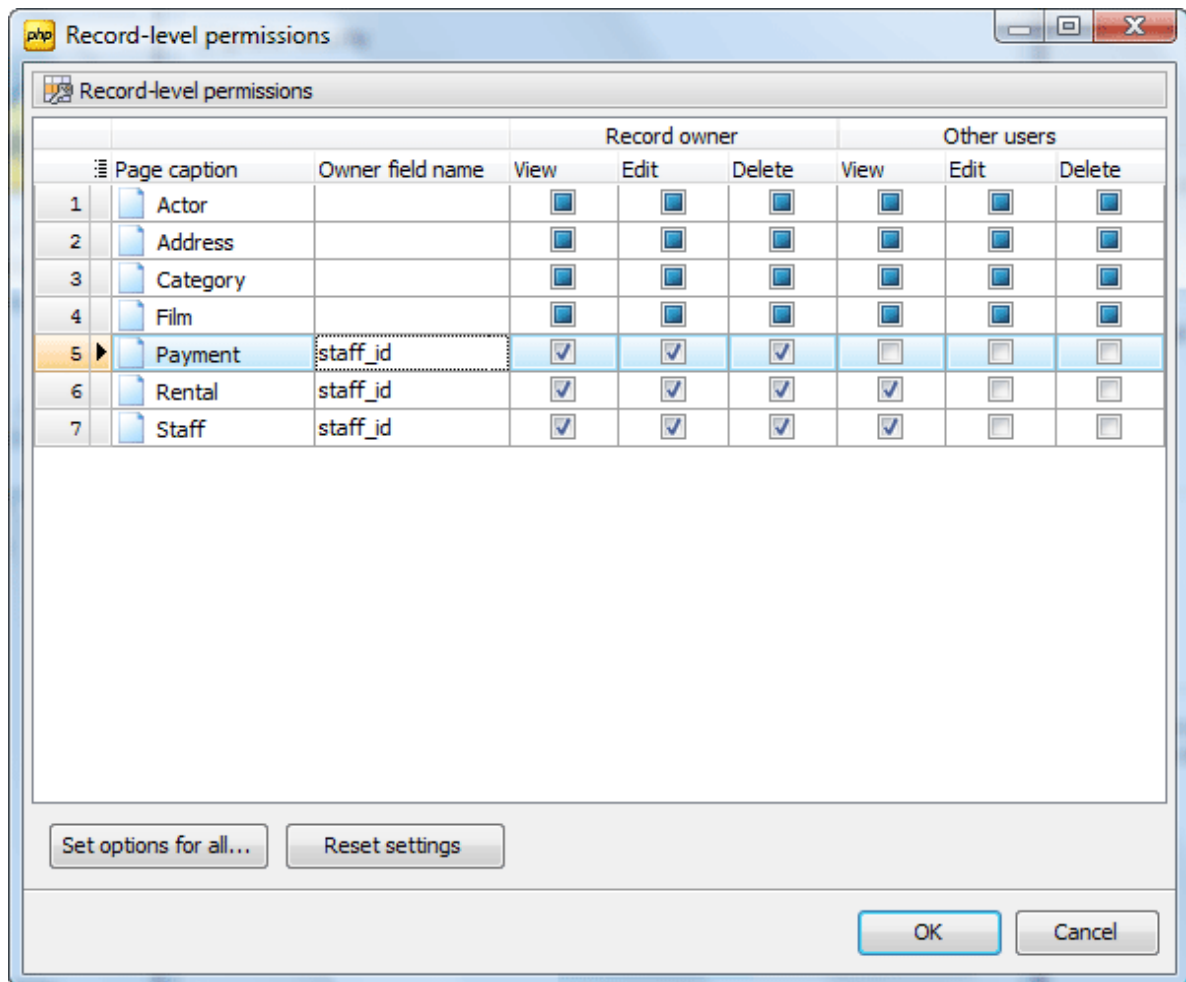
Note: The feature requires cURL php extension installed.

8.7 Record-level security

Record-level security allows you to specify records to be accessible to a certain user. This feature is available for [table-based](#)^[252] and [hard-coded](#)^[250] authorization types.

To enable record-level security for a page, define a column that identifies the record owner selecting the appropriate value from the drop down list in the **Owner field name** column. Now all users are divided into two groups: the record owners, and the others. Then grant privileges for each of these groups separately. Note the application and page admins bypass record-level security, so they can access (and change) all the records.

Click the **Set options for all** button to specify owner field name and/or permissions for all pages at once. Use the **Reset settings** button to clear all the settings made in this window.

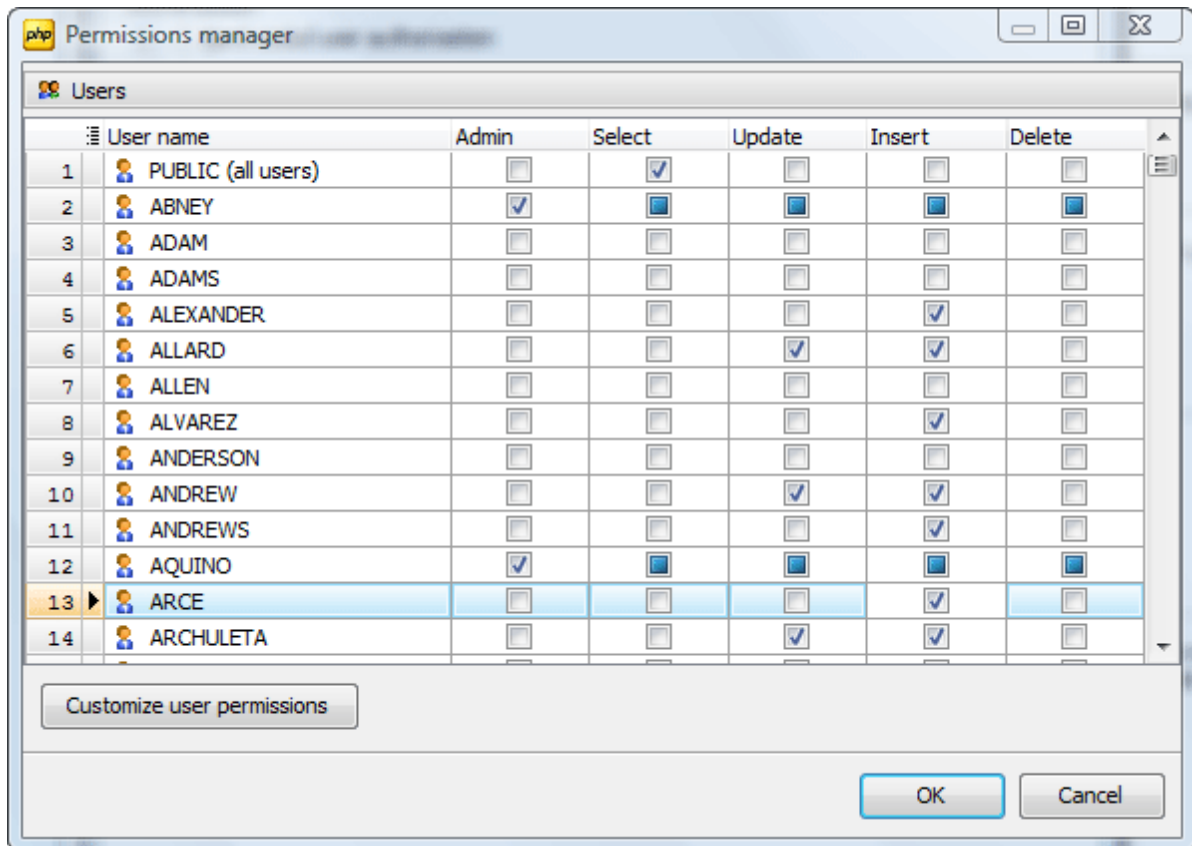


Live examples of using this feature can be found in the [Security Demo](#).

You can customize record-level security for a certain page with the [OnGetCustomRecordPermissions](#)^[192] event.

8.8 Permission manager

Permission manager allows you to restrict user access to the pages of the generated application. To open this window, press the [Manage permissions...](#) button. This is not applicable for the database server authorization, in that case the permissions should be set with the server facilities (GRANT and REVOKE statements).



How permissions are granted and applied

1. Permissions can be granted at the application and page levels.
2. Permissions granted at the application level are automatically applied to all pages in the application.
3. Page-level permissions are granted for each page individually. Effective user permissions for a page are calculated as sum of permissions granted to the user at the application and page levels i.e. permissions are cumulative.
4. PUBLIC permissions are permissions to be granted to all authorized users of your application, including those that might be created later. Any user in the application will have the sum of privileges granted directly to him and PUBLIC privileges. PUBLIC privileges also can be granted at the application and page levels.
5. You can enable the guest account in your application to allow any unauthorized user access to your website. The privileges can be granted to guest just like any user in

the application. Note that PUBLIC privileges do not affect the guest privileges and vice versa.

6. The following permissions can be granted: Select, Insert, Update, Delete, and Admin. If a user has the Admin permission for a page, he/she can read, change, and delete all the records of the page as well as add the new records and manage the page access permissions via Admin Panel (if the [table-based authorization](#)^[252] is used and permissions are stored in a database table).
7. Exporting, Printing, and Comparison operations are available for everyone who has the Select privilege.
8. The Copy operation is available for everyone who has the Insert privilege.

Live examples can be found in the [Security Demo](#).

Hard-coded authorization

On using this kind of authorization, all users and their permissions are stored in generated .php files, so it is necessary to re-generate the application to modify this data. Use the checkboxes at the Permissions tab to grant/revoke the corresponding application-level permissions to users specified at the [Hard-coded authorization window](#)^[250]. To grant or revoke page-level permissions, use the [Customize page level permission...](#) button.

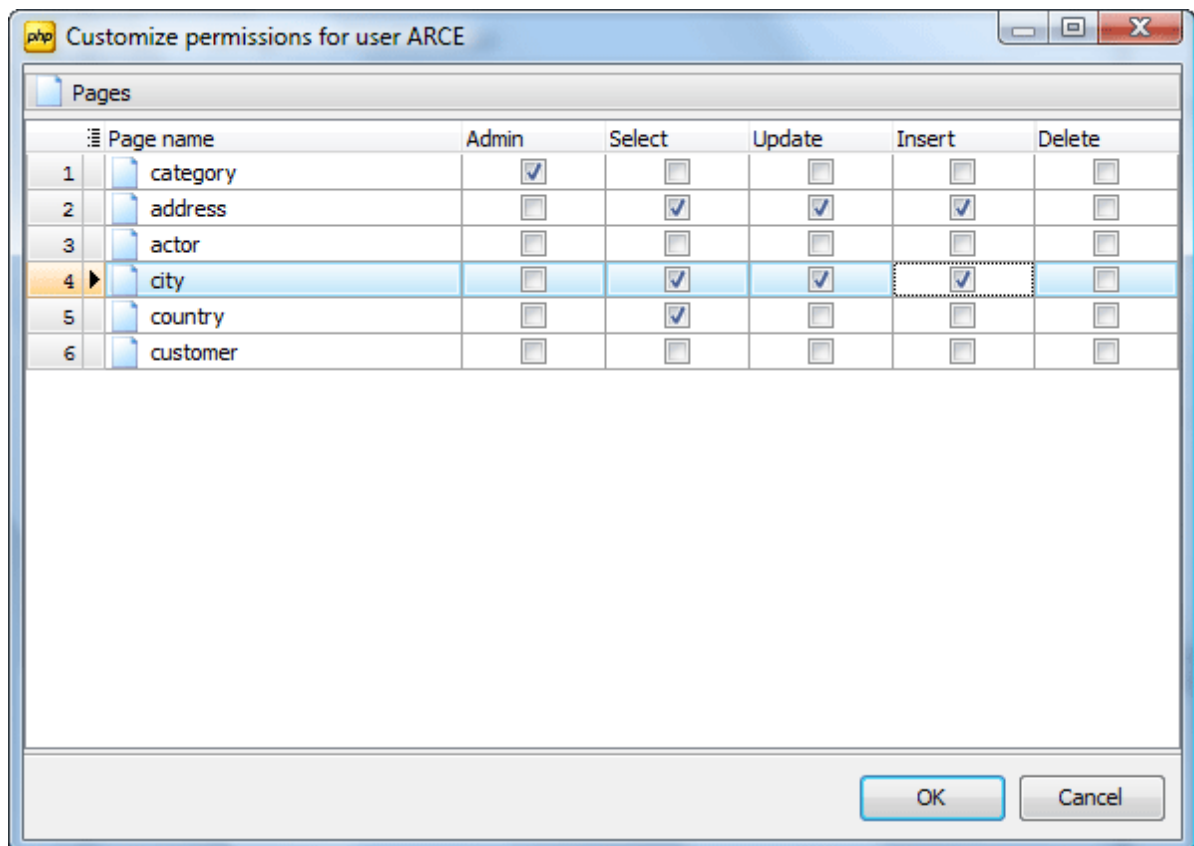
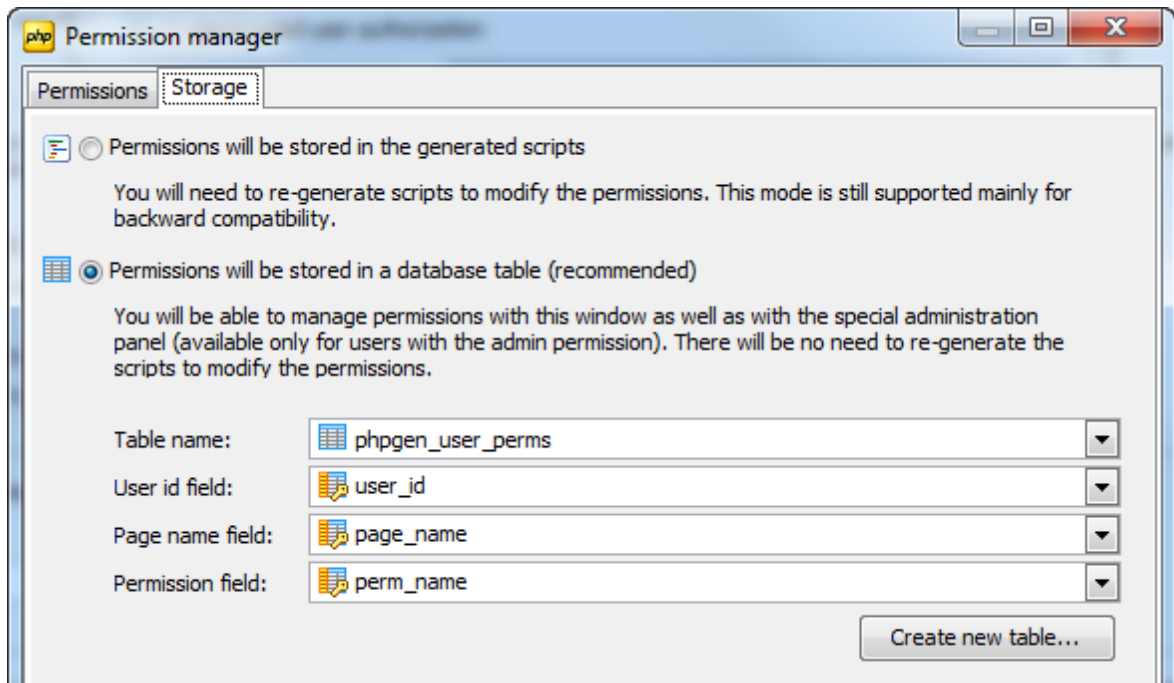


Table-based authorization

On using this kind of authorization, user accounts are always stored in a database table while permissions can be stored either in the generated files (as on using the hard-coded authorization) or in a database table like user accounts (recommended). To choose between these options, switch to the Storage tab and click the appropriate radio button.



The screenshot shows a web-based application window titled "Permission manager". It has two tabs: "Permissions" and "Storage", with "Storage" being the active tab. There are two radio buttons for selecting the storage method. The first option, "Permissions will be stored in the generated scripts", is unselected. The second option, "Permissions will be stored in a database table (recommended)", is selected. Below the radio buttons, there are four text input fields with dropdown menus: "Table name:" (containing "phpgen_user_perms"), "User id field:" (containing "user_id"), "Page name field:" (containing "page_name"), and "Permission field:" (containing "perm_name"). A "Create new table..." button is located at the bottom right of the form.

Permissions will be stored in the generated scripts

You will need to re-generate scripts to modify the permissions. This mode is still supported mainly for backward compatibility.

Permissions will be stored in a database table (recommended)

You will be able to manage permissions with this window as well as with the special administration panel (available only for users with the admin permission). There will be no need to re-generate the scripts to modify the permissions.

Table name:


User id field:

Page name field:


Permission field:



















Create new table...

To store permissions in the database, you can use an existing table or create a new one. On using this storage option, you can manage users and their permissions (both application and page levels) without necessity of re-building your site via PostgreSQL PHP Generator GUI as well as via the web-based [Administration Panel](#).

 Games and broadcast ▾ Participants ▾ Useful links ▾ Themes ▾ admin ▾

Administration panel

 Add user...

Actions	Username
 Permissions	PUBLIC (All users)
 Permissions	guest
 Permissions  Rename  Change password  Delete	admin
 Permissions  Rename  Change password  Delete	boston
 Permissions  Rename  Change password  Delete	game_manager
 Permissions  Rename  Change password  Delete	lakers

9 Interface language

By default a web application created with PostgreSQL PHP Generator has English interface but it can be localized (i.e. translated) to any language. The text strings to be used in the application's interface are stored in a localization file. To use a specific language, select the corresponding file on the last wizard step. In case you use the same file on a regular basis, it is recommended to specify it in the application [output options](#)³³⁷ to enable it by default.

Using prepared localization files

PostgreSQL PHP Generator comes with a number of localization files that can be found under the installation folder, usually *C:\Program Files\SQL Maestro Group\PostgreSQL PHP Generator* or *C:\Program Files (x86)\SQL Maestro Group\PostgreSQL PHP Generator* if you have a 64-bit operation system installed on your computer. The table below contains currently available files and the corresponding interface languages:

lang.ar.php	Arabic
lang.br.php	Brazilian
lang.cs.php	Czech
lang.de.php	German
lang.dk.php	Danish
lang.en.php	English
lang.es.php	Spanish
lang.fi.php	Finnish
lang.fr.php	French
lang.hu.php	Hungarian
lang.it.php	Italian
lang.nl.php	Dutch
lang.pl.php	Polish
lang.ru.php	Russian
lang.se.php	Swedish
lang.sk.php	Slovak
lang.sl.php	Slovenian
lang.sr.php	Serbian
lang.tr.php	Turkish

These files were sent to us by our customers and we are not responsible for their quality. If none of these files is right for you, it is possible to customize the localization as described below. We will be very appreciated if you send us your custom file to be enjoyed by all the PostgreSQL PHP Generator users.

Using a custom localization file

If none of supplied files is appropriate for your application, you can create the localization yourself. For this purpose get a localization file the clearest for you, enter/correct necessary captions with any text editor, and save it in the **UTF-8 encoding** (this is very important for the correct displaying of captions in a web browser).

Example

Suppose we need to create a web application in Esperanto. First we need to define the localization file that is the clearest for us. In our case it is *lang.en.php*. Let's open the file with a text editor (we prefer Notepad++), translate all captions to Esperanto, select the UTF-8 encoding and save it as *lang.eo.php* (eo is the two-char language code for Esperanto). All we need to do is select this file at the last wizard step and generate the application. The result is shown below.

Localization files

lang.en.php

```
$cAddNewRecord='Add new';
$cDeleteSelected='Delete selected';
$cRefresh='Refresh';
```

lang.eo.php

```
$cAddNewRecord='Aldoni novan';
$cDeleteSelected='Forigu elektatajn';
$cRefresh='Ĝisdatigi';
```

Generated web applications

Add new Delete selected Refresh						
		Actions				Actor Id
	+					1
	+ -					2

Aldoni novan Forigu elektatajn Ĝisdatigi						
		Actions				Actor Id
	+					1
	+ -					2

10 Common generation options

At the last wizard step you can specify [header and footer](#)^[242], set a [localization file](#)^[273], and define [security settings](#)^[248] of the created application.

Access driver

There are several PHP drivers to communicate with database servers and PostgreSQL. PHP Generator allows you to select a driver to be used by the generated web application. Note that the corresponding set of PHP functions should be available on your web server. By default the driver is set in accordance with the application [output options](#)^[337].

Output directory

Generated files will be created in the output directory stored on your computer. To see the result of the process, open any prepared .php file with any browser. Note, that to run PHP code you need to have PHP and Apache installed or to run any portable web server. To deploy ready web application to the Internet, [copy the created directory to your hosting](#)^[217].

Generation options

To make the repeated generating process more fast, it's recommended to avoid repeated copying of non-changeable system files stored in *components*, *database_engine*, *images*, *libs* folders. These files must be rewritten only in case they were generated by the former version of PostgreSQL PHP Generator.

Header and footer



Set default header and footer for the generated pages.

Security options



Our software allows you to use either hard-coded or table-based authorization. It is also possible to setup roles as well as define separate privileges for individual users.

PHP data access implementation



Select a driver for the data access level. The corresponding set of PHP functions should be available on your

PHP data access driver

Classic PHP PostgreSQL Driver (pgsql)

Localization



Select the localization file. This allows you to change the interface language of the generated app.

Localization file

S:\Projects\PHPGen\PgSQL\lang.en.php

Output options



Select the directory where files are generated to and choose which files you want to copy. Use the "Do not copy" option to avoid repeated copying of non-changeable files such as images, third-party components, libraries and

Output directory

P:\PostgreSQL\Test

Generation options

☒ Copy all files

☐ Do not copy system files

11 Developer Reference

The following features and technologies allow you to develop applications with PostgreSQL PHP Generator.

[Client Side API](#) ²⁷⁸

Extends the functionality of Insert and Edit forms of generated web applications.

[Server Side API](#) ³⁰⁴

Extends the functionality of the generated application by the possibility to execute additional queries and use environment variables.

[Style files compilation](#) ³²⁵

Allows to create websites in several color schemas.

11.1 Client Side API

The programming interface allows you to extend the functionality of [Insert and Edit forms](#) of generated web applications by the manipulating of their controls programmatically. Use these functions to operate with editors' attributes and values on the air.

PostgreSQL PHP Generator provides you with a number of operations executed on the client side. These functions may be used in [client_side_events](#)^[137]. Find out the description of available operations appropriate to each editor type:

- [All editors](#)^[278]
- [Text editor](#)^[288]
- [Text area](#)^[289]
- [Combobox](#)^[290]
- [Radio group](#)^[295]
- [Checkbox group and Multiple select](#)^[299]

11.1.1 All editors

The following operations may be applied to all editors:

- [getValue](#)^[278]
- [setValue](#)^[280]
- [getEnabled](#)^[282]
- [setEnabled](#)^[282]
- [getReadonly](#)^[284]
- [setReadonly](#)^[284]
- [getVisible](#)^[284]
- [setVisible](#)^[285]
- [getRequired](#)^[285]
- [setRequired](#)^[285]
- [getState](#)^[285]
- [setState](#)^[286]
- [setHint](#)^[286]
- [getFieldName](#)^[286]

11.1.1.1 getValue

Returns a value of a certain control (input, textarea, select, radio button, etc). To set a control value, use the [appropriate function](#)^[280].

Signature:

```
function getValue()
```


The examples below show how this method can be used in the [OnInsertFormEditorValueChanged](#)^[140] and [OnEditFormEditorValueChanged](#)^[142] event handlers.

Example 1:

The code below allows you to select a college only for players from U.S. ([see this in action](#)). See how it looks on the webpage at the screen below.

```
console.log(sender);
if (sender.getFieldName() == 'country_id')
{
    console.log(sender.getValue());
    editors['college_id'].enabled(sender.getValue() == 1);
    if (sender.getValue() != 1) {
        editors['college_id'].setValue(null);
        $('#college_id_edit').next().show();
    }
    else
        $('#college_id_edit').next().hide();
}
```

Country *

USA [us] × ▾

NBA Debut *

1996

Position

× Guard

College

Please select... ▾

Please select... ▲

Alabama

Alabama-Birmingham

Arizona

Arizona State

Arkansas

Arkansas-Little Rock

Auburn

Augsburg

Austin Peay

Baylor

Blinn Coll. TX (JC)

Boston College

Bowling Green

Bradley

Cal State Fullerton

Cal State-Fullerton

California

Central Michigan

Charleston (SC)

Country *

France [fr] × ▾

NBA Debut *

1996

Position

× Guard

College

Please select... ▾

College can be selected only for players from USA.

Team

Los Angeles Lakers × ▾

Number

24

* - Required field

Save ▾ Cancel

Example 2:

Another example from the insert and edit forms of the Players page ([see this in action](#)). This piece of code allows you to select player number only if a player's team is already selected (in other words, players who do not belong to a team cannot have numbers).

```
if (sender.getFieldName() == 'current_team_id')
{
    if (sender.getValue() == '')
    {
        editors['current_number'].setValue('');
        editors['current_number'].enabled(false);
        $('#current_number_edit').next().show();
    }
    else
    {
        editors['current_number'].enabled(true);
        $('#current_number_edit').next().hide();
    }
}
```

See also: [setValue](#) ²⁸⁰

11.1.1.2 setValue

Sets the value of a certain control. To get a control value, use the [appropriate function](#) ²⁷⁸.

Signature:

```
function setValue(value);
```

The examples below show how this method can be used in the [OnInsertFormEditorValueChanged](#) ¹⁴⁰ and [OnEditFormEditorValueChanged](#) ¹⁴² event handlers.

Example 1:

The code below allows you to select a college only for players from U.S. ([see this in action](#)). See how it looks on the webpage at the screen below.

```
console.log(sender);
if (sender.getFieldName() == 'country_id')
{
    console.log(sender.getValue());
    editors['college_id'].enabled(sender.getValue() == 1);
    if (sender.getValue() != 1) {
        editors['college_id'].setValue(null);
        $('#college_id_edit').next().show();
    }
    else
        $('#college_id_edit').next().hide();
}
```


Country *
 USA [us] × ▾

NBA Debut *
1996

Position
× Guard

College
Please select... ▾
Please select...
Alabama
Alabama-Birmingham
Arizona
Arizona State
Arkansas
Arkansas-Little Rock
Auburn
Augsburg
Austin Peay
Baylor
Blinn Coll. TX (JC)
Boston College
Bowling Green
Bradley
Cal State Fullerton
Cal State-Fullerton
California
Central Michigan
Charleston (SC)

Country *
 France [fr] × ▾

NBA Debut *
1996

Position
× Guard

College
Please select... ▾
College can be selected only for players from USA.

Team
 Los Angeles Lakers × ▾

Number
24

* - Required field

Save ▾ Cancel

Example 2:

Another example from the insert and edit forms of the Players page ([see this in action](#)). This piece of code allows you to select player number only if a player's team is already selected (in other words, players who do not belong to a team cannot have numbers).

```

if (sender.getFieldName() == 'current_team_id')
{
    if (sender.getValue() == '')
    {
        editors['current_number'].setValue('');
        editors['current_number'].enabled(false);
        $('#current_number_edit').next().show();
    }
    else
    {
        editors['current_number'].enabled(true);
        $('#current_number_edit').next().hide();
    }
}

```


Team	<input type="text" value="Los Angeles Lakers"/>	Team	<input type="text" value="Please select..."/>
Number	<input type="text" value="24"/>	Number	<input type="text"/>

See also: [getValue](#)^[278]

11.1.1.3 getEnabled

Returns the value specifying whether the control is enabled.

Signature:

```
function getEnabled();
```

Example:

```
var editorState = editors['current_number'].getEnabled();
```

Disabled control	<input type="text" value="6"/>
Ordinal control	<input type="text" value="86"/>

See also: [setEnabled](#)^[282]

11.1.1.4 setEnabled

Sets a value specifying whether the control is enabled. True to enable the control; false to disable it. Disabled input elements in a form will not be submitted.

Signature:

```
function setEnabled([true|false])
```


The examples below show how this method can be used in the [OnInsertFormEditorValueChanged](#)^[140] and [OnEditFormEditorValueChanged](#)^[142] event handlers.

Example 1:

The code below allows you to select a college only for players from U.S. ([see this in action](#)). See how it looks on the webpage at the screen below.

```
console.log(sender);
if (sender.getFieldName() == 'country_id')
{
    console.log(sender.getValue());
    editors['college_id'].enabled(sender.getValue() == 1);
    if (sender.getValue() != 1) {
        editors['college_id'].setValue(null);
        $('#college_id_edit').next().show();
    }
    else
        $('#college_id_edit').next().hide();
}
```


Country *

 USA [us]

✕ ▼

NBA Debut *

1996

Position

✕ Guard


College

Please select...

▼

Please select...
Alabama
Alabama-Birmingham
Arizona
Arizona State
Arkansas
Arkansas-Little Rock
Auburn
Augsburg
Austin Peay
Baylor
Blinn Coll. TX (JC)
Boston College
Bowling Green
Bradley
Cal State Fullerton
Cal State-Fullerton
California
Central Michigan
Charleston (SC)

Country *

 France [fr]

✕ ▼

NBA Debut *

1996

Position

✕ Guard


College

Please select...

⛔ ▼

College can be selected only for players from USA.

Team

 Los Angeles Lakers

✕ ▼

Number

24

* - Required field

Save ▼

Cancel

Example 2:

Another example from the insert and edit forms of the Players page ([see this in action](#)). This piece of code allows you to select player number only if a player's team is already selected (in other words, players who do not belong to a team cannot have numbers).

```

if (sender.getFieldName() == 'current_team_id')
{
    if (sender.getValue() == '')
    {
        editors['current_number'].setValue('');
        editors['current_number'].enabled(false);
        $('#current_number_edit').next().show();
    }
    else
    {
        editors['current_number'].enabled(true);
        $('#current_number_edit').next().hide();
    }
}

```


Team	<input type="text" value="Los Angeles Lakers"/>	Team	<input type="text" value="Please select..."/>
Number	<input type="text" value="24"/>	Number	<input type="text"/>

See also: [getEnabled](#)²⁸²

11.1.1.5 getReadOnly

Returns the value specifying whether the control is readonly.

Signature:

```
function getReadOnly()
```

Example:

```
var editorState = editors['team'].getReadOnly();
```

Disabled control	<input type="text" value="6"/>
Readonly control	<input type="text" value="0.99"/>
Ordinal control	<input type="text" value="86"/>

See also: [setReadOnly](#)²⁸⁴

11.1.1.6 setReadOnly

Sets a value specifying whether the control is readonly. A read-only input field cannot be modified (however, a user can tab to it, highlight it, and copy the text from it). Form elements with the readonly attribute set will get passed to the form processor.

Signature:

```
function setReadOnly([true|false])
```

Example:

```
editors['team'].setReadOnly(true);
```

Disabled control	<input type="text" value="6"/>
Readonly control	<input type="text" value="0.99"/>
Ordinal control	<input type="text" value="86"/>

See also: [getReadOnly](#)²⁸⁴

11.1.1.7 getVisible

Returns the value specifying whether the control is visible.

Signature:

```
function getVisible()
```

Example:

```
var editorState = editors['team'].getVisible();
```

See also: [setVisible](#)²⁸⁵

11.1.1.8 setVisible

Sets a value specifying whether the control is visible. An invisible element stays in its original position and size.

Signature:

```
function setVisible([true|false])
```

Example:

```
editors['team'].setVisible(true);
```

See also: [getVisible](#)²⁸⁴

11.1.1.9 getRequired

Returns the value specifying whether the control is required.

Signature:

```
function getRequired()
```

Example:

```
var editorState = editors['team'].getRequired();
```

See also: [setRequired](#)²⁸⁵

11.1.1.10 setRequired

Sets or clears the required attribute for a control.

Signature:

```
function setRequired([true|false])
```

Example:

```
if (sender.getFieldName() === 'payment_method') {  
    var isCreditCard = editors.payment_method.getValue() === 'Credit Card';  
    editors.credit_card.setVisible(isCreditCard).setRequired(isCreditCard);  
}
```

The example can be seen live in our [Feature Demo](#).

See also: [getRequired](#)²⁸⁵

11.1.1.11 getState

Returns the current state of a control.

Signature:

```
function getRequired()
```


Returning value:

One of the following strings: 'normal', 'warning', 'error', or 'success'.

Example:

```
var editorState = editors['team'].getState();
```

See also: [setState](#)^[286]

11.1.1.12 setState

Sets the state for a control.

Signature:

```
function setState(['normal'|'warning'|'error'|'success'])
```

Example:

```
if (sender.getValue() < 10) {  
    editors['percentage'].setState('warning');  
}  
else {  
    editors['percentage'].setState('normal');  
}
```

See also: [getState](#)^[285]

11.1.1.13 setHint

Sets or clears an inline hint for a control.

Signature:

```
function setHint(['Hint text'|null])
```

Example:

To set a hint for a control:

```
editors['field_name'].setHint('Hint text');
```

To clear a hint:

```
editors['field_name'].setHint(null);
```

11.1.1.14 getFieldName

Returns the name of the field in the database the editor corresponds to.

Signature:

```
function getFieldName()
```

The examples below show how this method can be used in the [OnInsertFormEditorValueChanged](#)^[140] and [OnEditFormEditorValueChanged](#)^[142] event handlers.

Example 1:

The code below allows you to select a college only for players from U.S. ([see this in action](#)). See how it looks on the webpage at the screen below.

```
console.log(sender);
if (sender.getFieldName() == 'country_id')
{
    console.log(sender.getValue());
    editors['college_id'].enabled(sender.getValue() == 1);
    if (sender.getValue() != 1) {
        editors['college_id'].setValue(null);
        $('#college_id_edit').next().show();
    }
    else
        $('#college_id_edit').next().hide();
}
```

Country *

NBA Debut *

Position

College

Please select...

Please select...
Alabama
Alabama-Birmingham
Arizona
Arizona State
Arkansas
Arkansas-Little Rock
Auburn
Augsburg
Austin Peay
Baylor
Blinn Coll. TX (JC)
Boston College
Bowling Green
Bradley
Cal State Fullerton
Cal State-Fullerton
California
Central Michigan
Charleston (SC)

Country *

NBA Debut *

Position

College

Please select...

College can be selected only for players from USA.

Team

Number

* - Required field

Save
Cancel

Example 2:

Another example from the insert and edit forms of the Players page ([see this in action](#)). This piece of code allows you to select player number only if a player's team is already selected (in other words, players who do not belong to a team cannot have numbers).


```

if (sender.getFieldName() == 'current_team_id')
{
    if (sender.getValue() == '')
    {
        editors['current_number'].setValue('');
        editors['current_number'].enabled(false);
        $('#current_number_edit').next().show();
    }
    else
    {
        editors['current_number'].enabled(true);
        $('#current_number_edit').next().hide();
    }
}

```

11.1.2 Text editor

The following operations may be applied to one-line input fields:

- [getPlaceholder](#)²⁸⁸
- [setPlaceholder](#)²⁸⁸

11.1.2.1 getPlaceholder

Returns the value specifying whether the control has a placeholder.

Signature:

```
function getPlaceholder()
```

Example:

```
var placeholder = editors['user_name'].getPlaceholder();
```

See also: [setPlaceholder](#)²⁸⁸

11.1.2.2 setPlaceholder

Specifies a short hint that describes the expected value of an input field (e.g. a sample value or a short description of the expected format). The short hint is displayed in the input field before the user enters a value.

Signature:

```
function setPlaceholder(value)
```

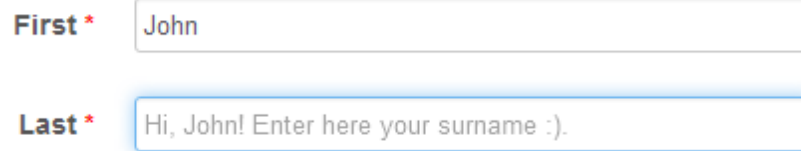
Example:

To add a placeholder to an Insert form depending of the entered first name, place the following strings to the OnInsertFormValueChanged.

```

if (editors['first_name'].getValue() == 'John')
{
    editors['last_name'].setPlaceholder('Hi, John! Enter here your surname :).');
}

```

See also: [getPlaceholder](#)²⁸⁸

11.1.3 Text area

The following operations may be applied to multi-line text input controls:

- [getPlaceholder](#)²⁸⁹
- [setPlaceholder](#)²⁸⁹

11.1.3.1 getPlaceholder

Returns the value specifying whether the control has a placeholder.

Signature:

```
function getPlaceholder()
```

Example:

```
var placeholder = editors['user_name'].getPlaceholder();
```

See also: [setPlaceholder](#)²⁸⁸

11.1.3.2 setPlaceholder

Specifies a short hint that describes the expected value of an input field (e.g. a sample value or a short description of the expected format). The short hint is displayed in the input field before the user enters a value.

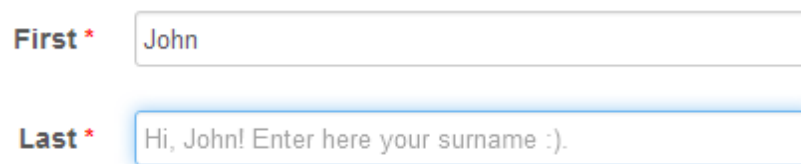
Signature:

```
function setPlaceholder(value)
```

Example:

To add a placeholder to an Insert form depending of the entered first name, place the following strings to the OnInsertFormValueChanged.

```
if (editors['first_name'].getValue() == 'John')
{
    editors['last_name'].setPlaceholder('Hi, John! Enter here your surname :).');
}
```



See also: [getPlaceholder](#)^[288]

11.1.4 Combobox

The following operations may be used to manipulate comboboxes on a form:

- [addItem](#)^[290]
- [removeItem](#)^[292]
- [getItemCount](#)^[294]
- [getCaption](#)^[294]
- [clear](#)^[294]

11.1.4.1 addItem

Adds items to a combobox dynamically.

Signature:

```
function addItem(value, caption);
```

Example:

This example shows how we can manipulate combobox and radio group items depending on the 'sex' value. Place the code in the [OnInsertFormEditorValueChanged](#)^[140] and [OnEditFormEditorValueChanged](#)^[142] event handlers to change the 'Title' content depending of the selected 'Sex' value.

```
function fillRelatedTitles(sender, editors)
{
    if (sender.getValue() == 'Male') {
        editors['title'].removeItem('Ms', 'Ms');
        editors['title'].removeItem('Mrs', 'Mrs');
        editors['title'].addItem('Mr', 'Mr');
    }
    else {
        editors['title'].removeItem('Mr');
        editors['title'].addItem('Ms', 'Ms');
        editors['title'].addItem('Mrs', 'Mrs');
    }
}
```

The screenshot demonstrates the result of the fired event.

Sex

First Name

Last Name

Title

Please select...
Mr
Ms
Mrs
Unknown

Save Cancel

Sex

First Name

Last Name

Title

Please select...
Mr
Unknown

Save Cancel

Sex

First Name

Last Name

Title

Please select...
Ms
Mrs
Unknown

Save Cancel

See also: [removeItem](#)^[292], [getItemCount](#)^[294], [clear](#)^[294]

11.1.4.2 removeItem

Removes an item from a combobox.

Signature:

```
function removeItem(value)
```

Example:

This example shows how we can manipulate combobox and radio group items depending on the 'sex' value. Place the code in the [OnInsertFormEditorValueChanged](#)^[140] and [OnEditFormEditorValueChanged](#)^[142] event handlers to change the 'Title' content depending of the selected 'Sex' value.

```
function fillRelatedTitles(sender, editors)
{
    if (sender.getValue() == 'Male') {
        editors['title'].removeItem('Ms', 'Ms');
        editors['title'].removeItem('Mrs', 'Mrs');
        editors['title'].addItem('Mr', 'Mr');
    }
    else {
        editors['title'].removeItem('Mr');
        editors['title'].addItem('Ms', 'Ms');
        editors['title'].addItem('Mrs', 'Mrs');
    }
}
```

The screenshot demonstrates the result of the fired event.

Sex

First Name

Last Name

Title

Please select...
Mr
Ms
Mrs
Unknown

Save Cancel

Sex

First Name

Last Name

Title

Please select...
Mr
Unknown

Save Cancel

Sex

First Name

Last Name

Title

Please select...
Ms
Mrs
Unknown

Save Cancel

See also: [addItem](#)^[290], [getItemCount](#)^[294], [clear](#)^[294]

11.1.4.3 getItemCount

Returns the number of combobox elements.

Signature:

```
function getItemCount()
```

Example:

The following code populates the *item_count* variable with the number of combobox elements.

```
var item_count = editors['title'].getItemCount();
```

See also: [addItem](#)^[290], [removeItem](#)^[292], [clear](#)^[294]

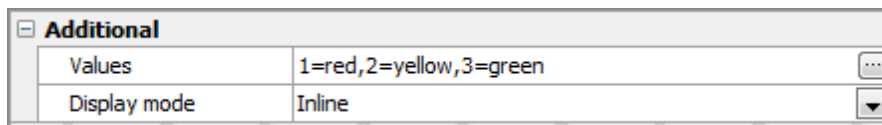
11.1.4.4 getCaption

Returns the caption of a certain control.

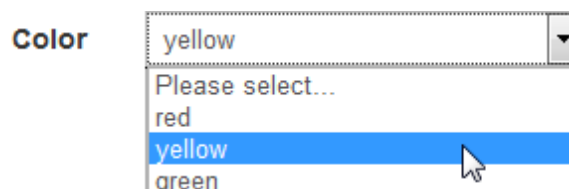
Signature:

```
function getCaption()
```

Let's see the difference between *getValue()* and *getCaption()* functions. Assume we have a Combobox editor with the following properties:



... and the second item is selected:



The following code shows the difference between these functions:

```
var value = editors['color'].getValue(); // value == 2
var caption = editors['color'].getCaption(); // caption == 'yellow'
```

See also: [addItem](#)^[290], [removeItem](#)^[292], [getItemCount](#)^[294], [clear](#)^[294]

11.1.4.5 clear

Deletes all elements in a combobox.

Signature:

```
function clear()
```


Example:

The following code empties the 'title' control.

```
editors['title'].clear();
```

See also: [addItem](#)^[290], [removeItem](#)^[292], [getItemCount](#)^[294]

11.1.5 Radio group

The following operations may be applied to groups of radio buttons:

- [addItem](#)^[295]
- [removeItem](#)^[296]
- [getItemCount](#)^[298]
- [getCaption](#)^[298]
- [clear](#)^[298]

11.1.5.1 addItem

Adds items to a radio group dynamically.

Signature:

```
function addItem(value, caption);
```

Example:

This example shows how we can manipulate combobox and radio group items depending on the 'sex' value. Place the code in the [OnInsertFormEditorValueChanged](#)^[140] and [OnEditFormEditorValueChanged](#)^[142] event handlers to change the 'Title' content depending of the selected 'Sex' value.

```
function fillRelatedTitles(sender, editors)
{
    if (sender.getValue() == 'Male') {
        editors['title'].removeItem('Ms', 'Ms');
        editors['title'].removeItem('Mrs', 'Mrs');
        editors['title'].addItem('Mr', 'Mr');
    }
    else {
        editors['title'].removeItem('Mr');
        editors['title'].addItem('Ms', 'Ms');
        editors['title'].addItem('Mrs', 'Mrs');
    }
}
```

The screenshot demonstrates the result of the fired event.

Sex	<input type="text" value="Please select..."/>
First Name	<input type="text"/>
Last Name	<input type="text"/>
Title	<input type="radio"/> Mr <input type="radio"/> Mrs <input type="radio"/> Ms <input type="radio"/> Unknown

Sex	<input type="text" value="Male"/>
First Name	<input type="text"/>
Last Name	<input type="text"/>
Title	<input type="radio"/> Mr <input type="radio"/> Unknown

Sex	<input type="text" value="Female"/>
First Name	<input type="text"/>
Last Name	<input type="text"/>
Title	<input type="radio"/> Ms <input type="radio"/> Mrs <input type="radio"/> Unknown

See also: [removeItem](#)^[296], [getItemCount](#)^[298], [clear](#)^[298]

11.1.5.2 removeItem

Removes an item from a radio group.

Signature:

```
function removeItem(value)
```

Example:

This example shows how we can manipulate combobox and radio group items depending on the 'sex' value. Place the code in the [OnInsertFormEditorValueChanged](#)^[140] and [OnEditFormEditorValueChanged](#)^[142] event handlers to change the 'Title' content depending of the selected 'Sex' value.


```
function fillRelatedTitles(sender, editors)
{
    if (sender.getValue() == 'Male') {
        editors['title'].removeItem('Ms','Ms');
        editors['title'].removeItem('Mrs','Mrs');
        editors['title'].addItem('Mr','Mr');
    }
    else {
        editors['title'].removeItem('Mr');
        editors['title'].addItem('Ms','Ms');
        editors['title'].addItem('Mrs','Mrs');
    }
}
```

The screenshot demonstrates the result of the fired event.

The image displays three sequential screenshots of a form, illustrating the state of the 'Sex' dropdown and 'Title' radio buttons after a selection change.

Top Screenshot: The 'Sex' dropdown is set to 'Please select...'. The 'Title' radio buttons are 'Mr', 'Mrs', 'Ms', and 'Unknown', all of which are unselected.

Middle Screenshot: The 'Sex' dropdown is set to 'Male'. The 'Title' radio buttons are 'Mr' and 'Unknown', both of which are unselected.

Bottom Screenshot: The 'Sex' dropdown is set to 'Female'. The 'Title' radio buttons are 'Ms', 'Mrs', and 'Unknown', all of which are unselected.

See also: [addItem](#)^[295], [getItemCount](#)^[298], [clear](#)^[298]

11.1.5.3 getItemCount

Returns the number of radio group elements.

Signature:

```
function getItemCount()
```

Example:

The following code populates the *item_count* variable with the number of combobox elements.

```
var item_count = editors['title'].getItemCount();
```

See also: [addItem](#)^[295], [removeItem](#)^[296], [clear](#)^[298]

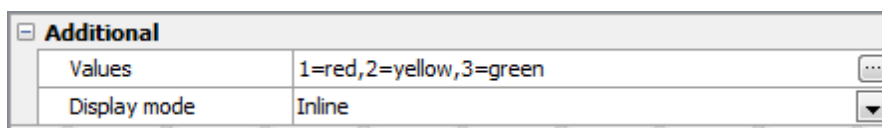
11.1.5.4 getCaption

Returns the caption of a certain control.

Signature:

```
function getCaption()
```

Let's see the difference between *getValue()* and *getCaption()* functions. Assume we have a Radio group editor with the following properties:



... and the second item is selected:

Color ☐ red ☒ yellow ☐ green

The following code shows the difference between these functions:

```
var value = editors['color'].getValue(); // value == 2
var caption = editors['color'].getCaption(); // caption == 'yellow'
```

See also: [addItem](#)^[295], [removeItem](#)^[296], [getItemCount](#)^[298], [clear](#)^[298]

11.1.5.5 clear

Deletes all elements in a radio group.

Signature:

```
function clear()
```

Example:

The following code empties the 'title' control.

```
editors['title'].clear();
```


See also: [addItem](#)^[295], [removeItem](#)^[296], [getItemCount](#)^[298]

11.1.6 Checkbox group and Multiple select

These function are helpful to define the behaviour of a checkbox group or a multiple select:

- [addItem](#)^[299]
- [removeItem](#)^[300]
- [getItemCount](#)^[301]
- [clear](#)^[302]

11.1.6.1 addItem

Adds items to a checkbox group or a multiple select dynamically.

Signature:

```
function addItem(value, caption);
```

Example:

The code below placed in the [OnInsertFormEditorValueChanged](#)^[140] and [OnEditFormEditorValueChanged](#)^[142] changes the content of the 'Clothes' checkbox group depending of the selected season.

```
if (sender.getFieldName() == 'season')
{
    var $item_count = editors['clothes'].getItemCount();
    if ($item_count == 6)
    {
        if (sender.getValue() == 'Winter')
        {
            editors['clothes'].removeItem('Shorts');
            editors['clothes'].removeItem('Sandals');
            editors['clothes'].removeItem('Swimsuit');
        }
        else
        {
            editors['clothes'].removeItem('Coat');
            editors['clothes'].removeItem('Boots');
            editors['clothes'].removeItem('Cap');
        }
    }
    else
    {
        editors['clothes'].clear();
        if (sender.getValue() == 'Winter')
        {
            editors['clothes'].addItem('Coat', 'Coat');
            editors['clothes'].addItem('Boots', 'Boots');
            editors['clothes'].addItem('Cap', 'Cap');
        }
        else
        {
            editors['clothes'].addItem('Shorts', 'Shorts');
            editors['clothes'].addItem('Sandals', 'Sandals');
            editors['clothes'].addItem('Swimsuit', 'Swimsuit');
        }
    }
}
```



```
}
}
```

The screenshot demonstrates the result of the fired event.

The image shows three sequential states of a web form. Each state has a 'Season' section with radio buttons for 'Summer' and 'Winter', and a 'Clothes' section with checkboxes for 'Swimsuit', 'Shorts', 'Coat', 'Boots', 'Sandals', and 'Cap'.
 - First state: 'Summer' is selected. No clothes are checked.
 - Second state: 'Summer' is selected. 'Swimsuit', 'Shorts', and 'Sandals' are checked.
 - Third state: 'Winter' is selected. 'Coat', 'Boots', and 'Cap' are checked.

See also: [removeItem](#)^[300], [getItemCount](#)^[301], [clear](#)^[302]

11.1.6.2 removeItem

Removes an item from a checkbox group or a multiple select.

Signature:

```
function removeItem(value)
```

Example:

The code below placed in the [OnInsertFormEditorValueChanged](#)^[140] and [OnEditFormEditorValueChanged](#)^[142] changes the content of the 'Clothes' checkbox group depending of the selected season.

```
if (sender.getFieldName() == 'season')
{
    var $item_count = editors['clothes'].getItemCount();
    if ($item_count == 6)
    {
        if (sender.getValue() == 'Winter')
        {
            editors['clothes'].removeItem('Shorts');
            editors['clothes'].removeItem('Sandals');
            editors['clothes'].removeItem('Swimsuit');
        }
        else
        {
            editors['clothes'].removeItem('Coat');
            editors['clothes'].removeItem('Boots');
            editors['clothes'].removeItem('Cap');
        }
    }
}
else
{
    editors['clothes'].clear();
    if (sender.getValue() == 'Winter')
    {
        editors['clothes'].addItem('Coat', 'Coat');
        editors['clothes'].addItem('Boots', 'Boots');
```



```

        editors['clothes'].addItem('Cap', 'Cap');
    }
    else
    {
        editors['clothes'].addItem('Shorts', 'Shorts');
        editors['clothes'].addItem('Sandals', 'Sandals');
        editors['clothes'].addItem('Swimsuit', 'Swimsuit');
    }
}
}

```

The screenshot demonstrates the result of the fired event.

The image shows three sequential states of a web form. Each state has a 'Season' section with radio buttons for 'Summer' and 'Winter', and a 'Clothes' section with checkboxes for 'Swimsuit', 'Shorts', 'Coat', 'Boots', 'Sandals', and 'Cap'.

- State 1:** 'Summer' is selected. 'Clothes' is empty.
- State 2:** 'Summer' is selected. 'Swimsuit', 'Shorts', and 'Sandals' are checked.
- State 3:** 'Winter' is selected. 'Coat', 'Boots', and 'Cap' are checked.

See also: [addItem](#)^[299], [getItemCount](#)^[301], [clear](#)^[302]

11.1.6.3 getItemCount

Returns the number of checkbox group or a multiple select elements.

Signature:

```
function getItemCount()
```

Example:

The code below placed in the [OnInsertFormEditorValueChanged](#)^[140] and [OnEditFormEditorValueChanged](#)^[142] changes the content of the 'Clothes' checkbox group depending of the selected season.

```

if (sender.getFieldName() == 'season')
{
    var $item_count = editors['clothes'].getItemCount();
    if ($item_count == 6)
    {
        if (sender.getValue() == 'Winter')
        {
            editors['clothes'].removeItem('Shorts');
            editors['clothes'].removeItem('Sandals');
            editors['clothes'].removeItem('Swimsuit');
        }
        else
        {
            editors['clothes'].removeItem('Coat');
            editors['clothes'].removeItem('Boots');
            editors['clothes'].removeItem('Cap');
        }
    }
}

```



```

    }
    else
    {
        editors['clothes'].clear();
        if (sender.getValue() == 'Winter')
        {
            editors['clothes'].addItem('Coat','Coat');
            editors['clothes'].addItem('Boots','Boots');
            editors['clothes'].addItem('Cap','Cap');
        }
        else
        {
            editors['clothes'].addItem('Shorts','Shorts');
            editors['clothes'].addItem('Sandals','Sandals');
            editors['clothes'].addItem('Swimsuit','Swimsuit');
        }
    }
}
}

```

The screenshot demonstrates the result of the fired event.

The image shows three sequential states of a web form. Each state has a 'Season' section with radio buttons for 'Summer' and 'Winter', and a 'Clothes' section with checkboxes for 'Swimsuit', 'Shorts', 'Coat', 'Boots', 'Sandals', and 'Cap'.
 - Left state: 'Summer' is selected. 'Clothes' is empty.
 - Middle state: 'Winter' is selected. 'Clothes' contains 'Swimsuit', 'Shorts', and 'Sandals'.
 - Right state: 'Winter' is selected. 'Clothes' contains 'Coat', 'Boots', and 'Cap'.

See also: [removeItem](#)^[300], [addItem](#)^[299], [clear](#)^[302]

11.1.6.4 clear

Deletes all elements in a checkbox group or a multiple select.

Signature:

```
function clear()
```

Example:

The code below placed in the [OnInsertFormEditorValueChanged](#)^[140] and [OnEditFormEditorValueChanged](#)^[142] changes the content of the 'Clothes' checkbox group depending of the selected season.

```

if (sender.getFieldName() == 'season')
{
    var $item_count = editors['clothes'].getItemCount();
    if ($item_count == 6)
    {
        if (sender.getValue() == 'Winter')
        {
            editors['clothes'].removeItem('Shorts');
            editors['clothes'].removeItem('Sandals');
        }
    }
}

```



```

        editors['clothes'].removeItem('Swimsuit');
    }
    else
    {
        editors['clothes'].removeItem('Coat');
        editors['clothes'].removeItem('Boots');
        editors['clothes'].removeItem('Cap');
    }
}
else
{
    editors['clothes'].clear();
    if (sender.getValue() == 'Winter')
    {
        editors['clothes'].addItem('Coat', 'Coat');
        editors['clothes'].addItem('Boots', 'Boots');
        editors['clothes'].addItem('Cap', 'Cap');
    }
    else
    {
        editors['clothes'].addItem('Shorts', 'Shorts');
        editors['clothes'].addItem('Sandals', 'Sandals');
        editors['clothes'].addItem('Swimsuit', 'Swimsuit');
    }
}
}
}

```

The screenshot demonstrates the result of the fired event.

Season <input type="radio"/> Summer <input type="radio"/> Winter	Season <input checked="" type="radio"/> Summer <input type="radio"/> Winter	Season <input type="radio"/> Summer <input checked="" type="radio"/> Winter
Clothes <input type="checkbox"/> Swimsuit <input type="checkbox"/> Shorts <input type="checkbox"/> Coat <input type="checkbox"/> Boots <input type="checkbox"/> Sandals <input type="checkbox"/> Cap	Clothes <input checked="" type="checkbox"/> Swimsuit <input type="checkbox"/> Shorts <input checked="" type="checkbox"/> Sandals	Clothes <input checked="" type="checkbox"/> Coat <input checked="" type="checkbox"/> Boots <input type="checkbox"/> Cap

See also: [addItem](#)^[299], [removeItem](#)^[300], [getItemCount](#)^[301]

11.2 Server Side API

The programmatic interface allows you to extend the functionality of the generated applications. Available methods are grouped by classes they belong to. Use them in the [sever side events](#)^[147]. In page-level events the current page is also accessible as *\$this*.

- [class Page](#)^[304]
- [class Application](#)^[314]
- [class EngConnection](#)^[318]
- [class Grid](#)^[319]
- [class PageList](#)^[321]
- [Global functions](#)^[322]

11.2.1 class Page

Class Page encapsulates the whole generated page. The following methods of this class might be useful for application developers:

- [GetEnvVar](#)^[304]
- [GetConnection](#)^[305]
- [GetGrid](#)^[305]
- [GetCurrentUserId](#)^[305]
- [GetCurrentUserName](#)^[306]
- [IsLoggedInAsAdmin](#)^[306]
- [IsCurrentlyUserLoggedIn](#)^[306]
- [setShowGrid](#)^[306]
- [setDescription](#)^[307]
- [setDetailedDescription](#)^[308]
- [SetTitle](#)^[309]
- [SetInsertFromTitle](#)^[310]
- [SetEditFromTitle](#)^[311]
- [SetViewFromTitle](#)^[312]
- [Print & Export enableity methods](#)^[313]

11.2.1.1 GetEnvVar

Returns the value of a specified environment variable.

Signature:

```
function GetEnvVar($name)
```


Example:

The following example demonstrates how to use variables within the [OnBeforeInsertRecord](#)^[178] event handler.

```
$rowData['ip_address'] = $this->GetEnvVar('REMOTE_ADDR');
$username = $this->GetEnvVar('CURRENT_USER_NAME');
if ($username != 'admin')
    $rowData['changed_by'] = $username;
```

11.2.1.2 GetConnection

Returns the connection object (an instance of the [EngConnection](#)^[318] class).

Signature:

```
function GetConnection()
```

Example:

To get the list of available databases, use the following code:

```
$queryResult = array();
$this->GetConnection()->ExecQueryToArray('SHOW DATABASES', $queryResult);
```

11.2.1.3 GetGrid

Returns the grid object (an instance of the [Grid](#)^[319] class).

Signature:

```
function GetGrid()
```

Example:

To disable the Multi Edit feature for a certain page, place the following code in the [OnPreparePage](#)^[149] event handler:

```
$this->GetGrid()->setMultiEditAllowed(false);
```

11.2.1.4 GetMasterGrid

Returns the master grid of a detail page (an instance of the [Grid](#)^[319] class).

Signature:

```
function GetMasterGrid()
```

Example:

To hide the 'owner' column in a master grid, use the following code in the [OnPreparePage](#)^[149] event handler:

```
$masterGrid = $this->GetMasterGrid();
if ($masterGrid) {
    $column = $masterGrid->GetViewColumn('owner');
    $column->setVisible(false);
}
```

11.2.1.5 GetCurrentUserid

Returns Id of currently logged in user.

Signature:

```
function GetCurrentUserId()
```

Example:

```
$rowData['user_id'] = $this->GetCurrentUserId();
```

11.2.1.6 GetCurrentUserName

Returns name of currently logged in user.

Signature:

```
function GetCurrentUserName()
```

Example:

```
if ($this->GetCurrentUserName() == 'john') {  
    // perform some actions for john  
}
```

11.2.1.7 IsLoggedInAsAdmin

Checks whether currently logged in user has admin permissions for a page.

Signature:

```
function IsLoggedInAsAdmin()
```

Example

```
if (!$this->IsLoggedInAsAdmin()) {  
    // perform some actions on a page for ordinal users  
}
```

11.2.1.8 IsCurrentUserLoggedIn

Checks if the current visitor is a logged in user. This is useful when the guest access is enabled in your application.

Signature:

```
function IsCurrentUserLoggedIn()
```

Example

```
if (!$this->IsCurrentUserLoggedIn()) {  
    // perform some actions for guest  
}
```


11.2.1.9 setShowGrid

Allows to show/hide a data grid on a page.

Signature:

```
function setShowGrid($value)
```

Example

To hide a data grid on a page, place the following code in the [OnPreparePage](#)  event handler:


```
$this->setShowGrid(false);
```

11.2.1.10 setDescription

Allows to provide a page with a text block located below the page header. HTML tags are allowed. The code in the examples below should be placed into the [OnPreparePage](#) event handler.

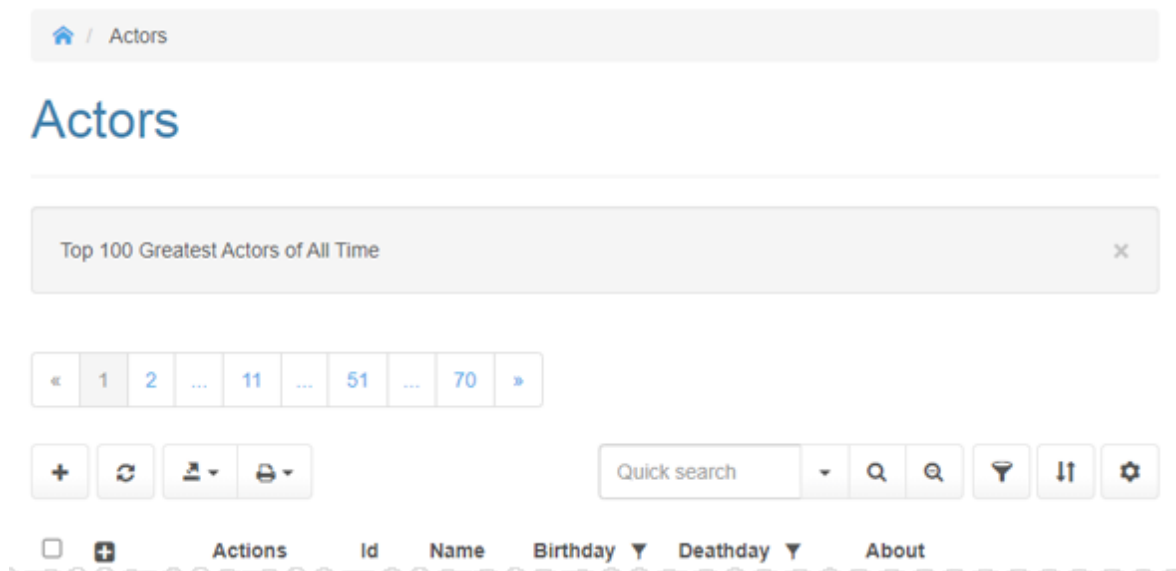
Signature:

```
function setDescription($value)
```

Example 1:

To provide a page with a description "Top 100 Greatest Actors of All Time", place the following code in the event body:

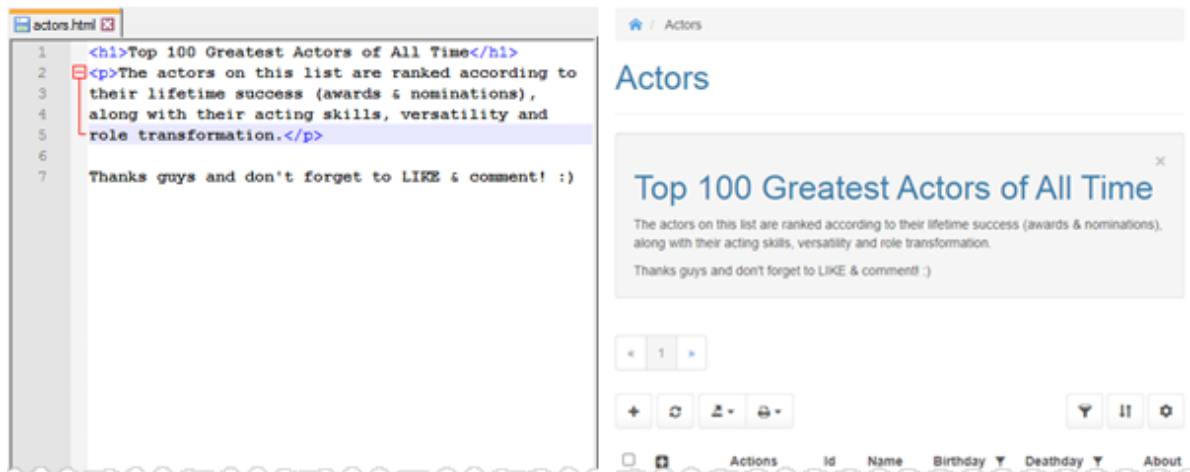
```
$this->setDescription("Top 100 Greatest Actors of All Time");
```



Example 2:

In this example the description is loaded from an external file. In this case you can modify the text directly in the file without regenerating the application.

```
$description = file_get_contents('external_data/descriptions/actors.html');  
$this->setDescription($description);
```

11.2.1.11 setDetailedDescription

Allows to provide a page with a text description displayed in a modal window invoked with the question mark button on the right side of the title bar of the page. HTML tags are allowed. The code in the examples below should be placed into the [OnPreparePage](#) event handler.

Signature:

```
function setDetailedDescription($value)
```

Example 1:

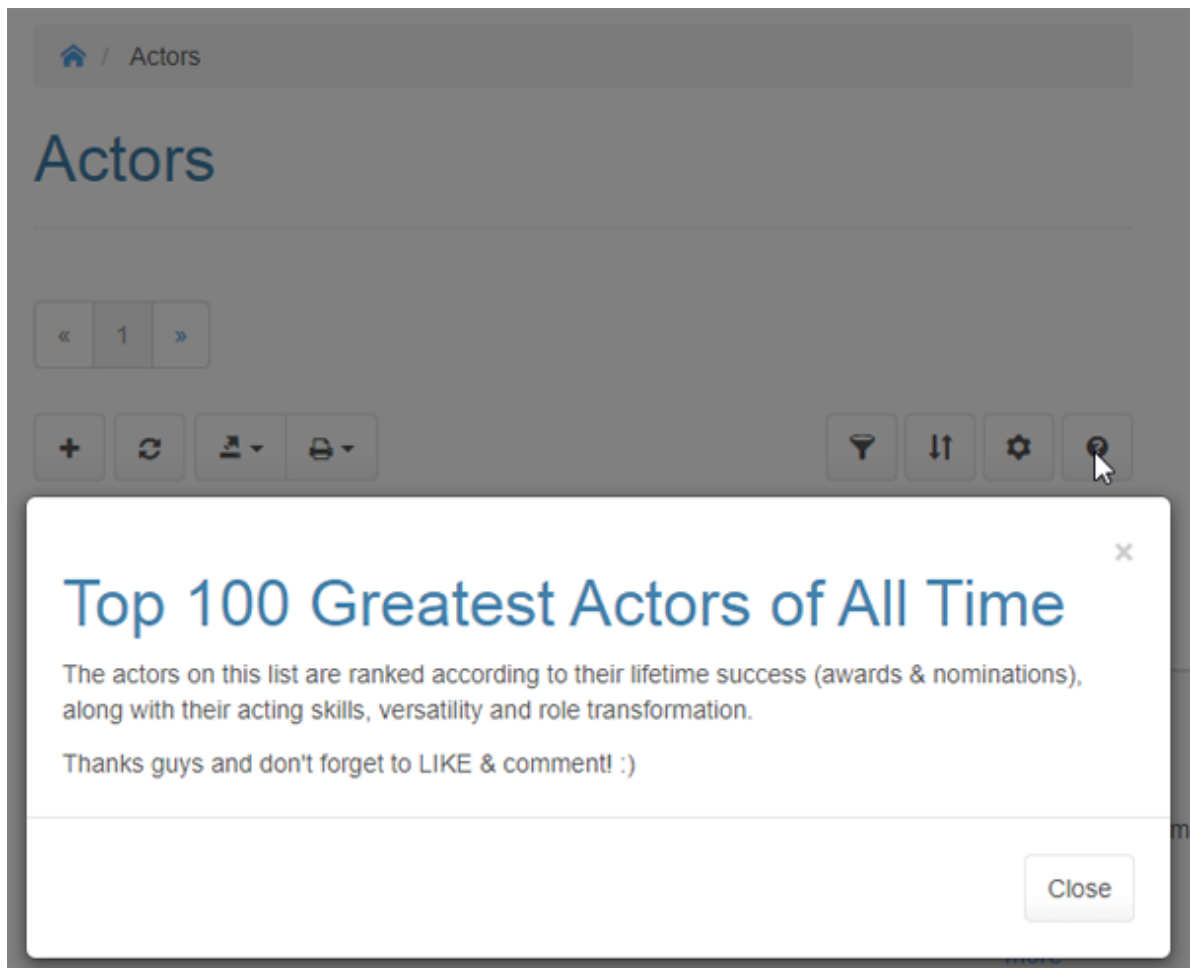
To specify a detailed description "Top 100 Greatest Actors of All Time", place the following code in the event body:

```
$this->setDetailedDescription("Top 100 Greatest Actors of All Time");
```

Example 2:

In this example the description is loaded from an external file. In this case you can modify the text directly in the file without regenerating the application.

```
$description = file_get_contents('external_data/descriptions/actors.html');
$this->setDetailedDescription($description);
```

11.2.1.12 SetTitle

Allows to specify a webpage title. The code in the example below should be placed into the [OnPreparePage](#)¹⁴⁹ event handler.


Signature:

```
function SetTitle($value)
```







Example 1:

To provide a page with a title "Top 100 Actors", place the following code in the event body:

```
$this->SetTitle("Top 100 Actors");
```


 / Top 100 Actors

Top 100 Actors

<input type="checkbox"/>		Actions	Id	Name	Birthday ▼	Deathday ▼
<input type="checkbox"/>	 ▼	   	2	Mark Hamill	1951-09-25	NULL

11.2.1.13 SetInsertFormTitle

Allows to specify a title of the webpage Insert form. The code in the example below should be placed into the [OnPreparePage](#)¹⁴⁹ event handler.

Signature:

```
function SetInsertFormTitle($value)
```

Example 1:

To provide the page Insert form with the "Insert your favorite actor" title, place the following code in the event body:

```
$this->SetInsertFormTitle("Insert your favorite actor");
```


[Home](#) / [Top 100 Actors](#) / Insert your favorite actor

Insert your favorite actor

Save

Cancel



Name *

Richard Gere

11.2.1.14 SetEditFromTitle

Allows to specify the title of the page Edit form. The code in the example below should be placed into the [OnPreparePage](#)¹⁴⁹ event handler. It is possible to use field name tags in the title template.

Signature:

```
function SetEditFormTitle($value)
```

Example 1:

To provide the page Edit form with such titles as "Editing Richard Gere profile", "Editing Harrison Ford profile" and so on, place the following code in the event body:

```
$this->SetEditFormTitle("Editing %name% profile");
```



[Home](#) / [Top 100 Actors](#) / Editing Harrison Ford profile

Editing Harrison Ford profile

Save

Cancel

Name *

Birthday
 

11.2.1.15 SetViewFromTitle

Allows to specify the title of the page View form. The code in the example below should be placed into the [OnPreparePage](#)¹⁴⁹ event handler. It is possible to use field name tags in the title template ([Live Demo](#)).

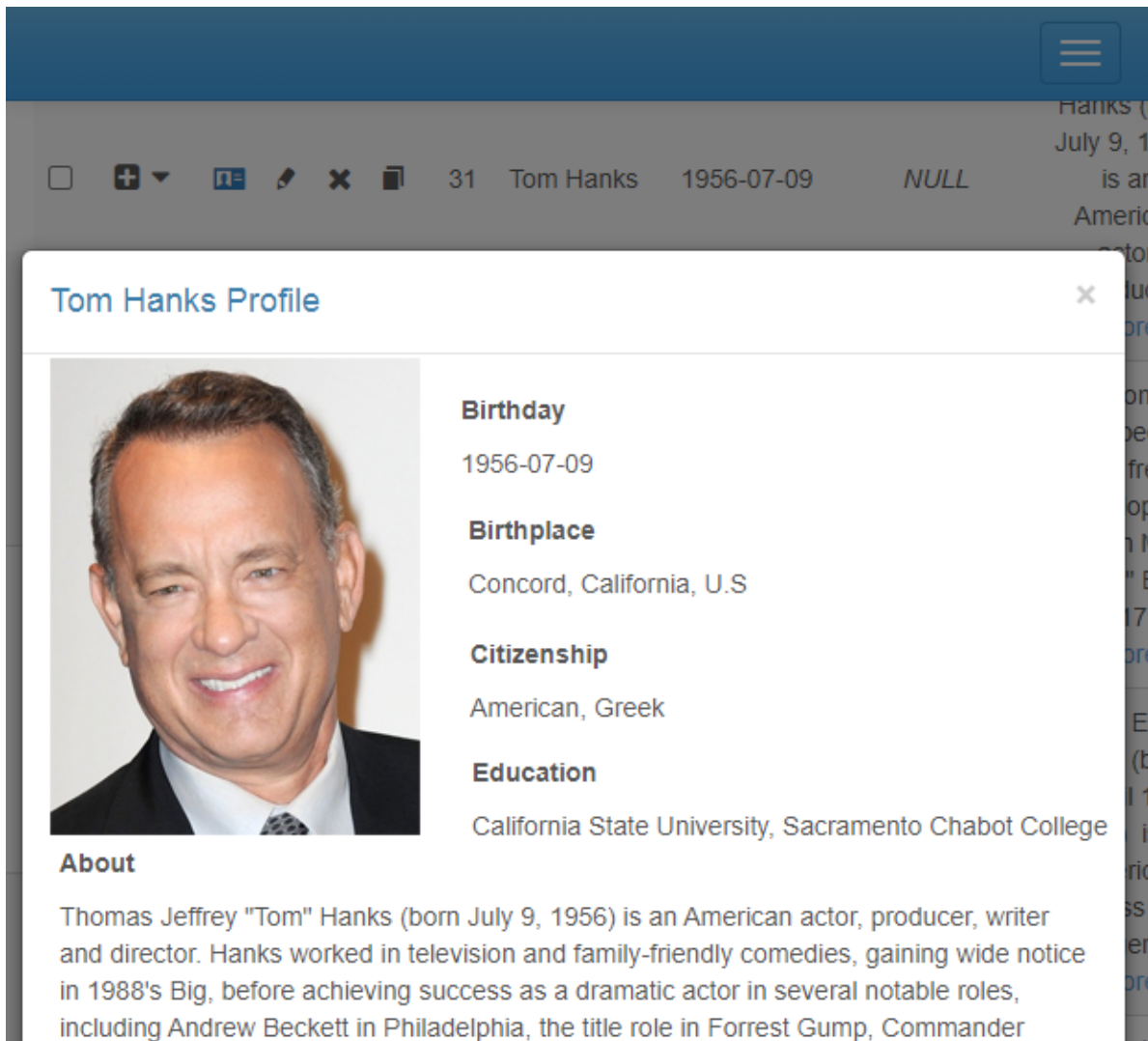
Signature:

```
function SetViewFormTitle($value)
```

Example 1:

To provide the View form with such title as "Tom Hanks Profile", "Bruce Willis Profile", and so on, place the following code in the event body:

```
$this->SetViewFormTitle("%name% Profile");
```

Tom Hanks Profile

Birthday
1956-07-09

Birthplace
Concord, California, U.S

Citizenship
American, Greek

Education
California State University, Sacramento Chabot College

About
Thomas Jeffrey "Tom" Hanks (born July 9, 1956) is an American actor, producer, writer and director. Hanks worked in television and family-friendly comedies, gaining wide notice in 1988's Big, before achieving success as a dramatic actor in several notable roles, including Andrew Beckett in Philadelphia, the title role in Forrest Gump, Commander

11.2.1.16 Print & Export enableility methods

The following methods allow to enable/disable various print and export abilities for a page.

Signature	Description
<code>setPrintListAvailable(\$value)</code>	Allows to print all records from a grid.
<code>setExportListAvailable(array \$exportFormats)</code>	Allows to export all records from a grid to the given formats.
<code>setAllowPrintSelectedRecords(\$value)</code>	Allows to print the selected records from a grid.
<code>setExportSelectedRecordsAvailable(array \$exportFormats)</code>	Allows to export the selected records from a grid to the given formats.
<code>setPrintListRecordAvailable(\$value)</code>	Allows to print a single record from a grid.
<code>setExportListRecordAvailable(array \$exportFormats)</code>	Allows to export a single record from a grid to the given formats.

<code>setPrintOneRecordAvailable(\$value)</code>	Allows to print a record from a view form.
<code>setExportOneRecordAvailable(array \$exportFormats)</code>	Allows to export a record from a view form to the given formats.

The `$exportFormats` array can include the following values: 'pdf', 'excel', 'word', 'xml', 'csv'.

The code in the examples below should be placed into the [OnPreparePage](#)^[149] event handler:

Example 1:

To allow to export the selected records in 'pdf' and 'word' formats only, use the following code:

```
$this->setExportSelectedRecordsAvailable(array('pdf', 'word'));
```

Example 2:

To allow to execute any print operations only for users with admin permissions, use the following code:

```
$isPageAdmin = $this->IsLoggedInAsAdmin();
$this->setPrintListAvailable($isPageAdmin);
$this->setAllowPrintSelectedRecords($isPageAdmin);
$this->setPrintListRecordAvailable($isPageAdmin);
$this->setPrintOneRecordAvailable($isPageAdmin);
```

Example 3:

To disable any export operations for ordinary users, use the following code:

```
$isOrdinaryUser = !$this->IsLoggedInAsAdmin();
if ($isOrdinaryUser) {
    $this->setExportListAvailable(array());
    $this->setExportSelectedRecordsAvailable(array());
    $this->setExportListRecordAvailable(array());
    $this->setExportOneRecordAvailable(array());
}
```

11.2.2 class Application

The following methods of the Application class might be useful for application developers:

- [IsGetValueSet](#)^[315]
- [GetGetValue](#)^[315]
- [IsPOSTValueSet](#)^[315]
- [GetPOSTValue](#)^[316]
- [IsLoggedInAsAdmin](#)^[316]
- [IsCurrentUserLoggedIn](#)^[316]
- [Session methods](#)^[316]

11.2.2.1 IsGETValueSet

Checks if \$_GET contains a parameter named paramName. Returns true if such parameter does exist and false otherwise.

Signature:

```
function IsGETValueSet($paramName)
```

Example:

The code below checks if a \$_GET parameter does exist and, if so, assigns its value to a variable.

```
// check if the parameter custom_var available
if (GetApplication()->IsGETValueSet('custom_var'))
    // found
    $params['custom_var'] = GetApplication()->GetGETValue('custom_var');
else
    // not found. Assign a default value.
    $params['custom_var'] = '5 (default)';
```

See also: [GetGETValue](#)^[315]

11.2.2.2 GetGETValue

Returns the value of a parameter named paramName from the superglobal \$_GET array.

Signature:

```
function GetGETValue($name)
```

Example:

The code below checks if a \$_GET parameter does exist and, if so, assigns its value to a variable.

```
// check if the parameter custom_var available
if (GetApplication()->IsGETValueSet('custom_var'))
    // found
    $params['custom_var'] = GetApplication()->GetGETValue('custom_var');
else
    // not found. Assign a default value.
    $params['custom_var'] = '5 (default)';
```

See also: [IsGETValueSet](#)^[315]

11.2.2.3 IsPOSTValueSet

Checks if \$_POST contains a parameter named paramName. Returns true if such parameter does exist and false otherwise.

Signature:

```
function IsPOSTValueSet($paramName)
```

The code below checks if a \$_POST parameter does exist and, if so, assigns its value to a variable.

```
// check if the parameter custom_var available
if (GetApplication()->IsPOSTValueSet('custom_var'))
    // found
```



```

        $params['custom_var'] = GetApplication()->GetPOSTValue('custom_var');
    else
        // not found. Assign a default value.
        $params['custom_var'] = '5 (default)';

```

See also: [GetPOSTValue](#)³¹⁶

11.2.2.4 GetPOSTValue

Returns the value of a parameter named paramName from the superglobal \$_POST array.

Signature:

```
function GetPOSTValue($name)
```

The code below checks if a \$_POST parameter does exist and, if so, assigns its value to a variable.

```

// check if the parameter custom_var available
if (GetApplication()->IsPOSTValueSet('custom_var'))
    // found
    $params['custom_var'] = GetApplication()->GetPOSTValue('custom_var');
else
    // not found. Assign a default value.
    $params['custom_var'] = '5 (default)';

```

See also: [IsPOSTValueSet](#)³¹⁵

11.2.2.5 IsLoggedInAsAdmin

Checks whether currently logged in user has admin permissions at the application level.

Signature:

```
function IsLoggedInAsAdmin()
```

Example

```

if (!GetApplication()->IsLoggedInAsAdmin()) {
    // perform some actions for ordinal users
}

```

11.2.2.6 IsCurrentUserLoggedIn

Checks if the current visitor is a logged in user. This is useful when the guest access is enabled in your application.

Signature:

```
function IsCurrentUserLoggedIn()
```

Example

```

if (!GetApplication()->IsCurrentUserLoggedIn()) {
    // perform some actions for guest
}

```

11.2.2.7 Session methods

The following methods can be used to handle session variables.

Signature	Description
IsSessionVariableSet(\$name)	Checks if the given variable exists in the session.
GetSessionVariable(\$name)	Returns the value of the given session variable.
SetSessionVariable(\$name, \$value)	Sets a value to a session variable.
UnSetSessionVariable(\$name)	Removes the given variable from the session.

The code in the examples below is used in the [OnPreparePage](#)^[149] event handler:

Example 1:

The following code is used to filter countries by life expectancy:

```
// default values
$minLifeExpectancy = 40;
$maxLifeExpectancy = 90;

// checking values in the browser address line
if (GetApplication()->isGetValueSet('minLifeExpectancy') &&
    GetApplication()->isGetValueSet('maxLifeExpectancy')) {
    $minLifeExpectancy = GetApplication()->GetGETValue('minLifeExpectancy');
    $maxLifeExpectancy = GetApplication()->GetGETValue('maxLifeExpectancy');
} // or (if not found) in the session
elseif (GetApplication()->IsSessionVariableSet('minLifeExpectancy') &&
    GetApplication()->IsSessionVariableSet('maxLifeExpectancy')) {
    $minLifeExpectancy = GetApplication()->GetSessionVariable('minLifeExpectancy');
    $maxLifeExpectancy = GetApplication()->GetSessionVariable('maxLifeExpectancy');
}

// applying filter to the dataset
$this->dataset->AddCustomCondition(
    "life_expectancy >= {$minLifeExpectancy} ".
    "AND life_expectancy <= {$maxLifeExpectancy}");

// saving calculated values to the session
GetApplication()->SetSessionVariable('minLifeExpectancy', $minLifeExpectancy);
GetApplication()->SetSessionVariable('maxLifeExpectancy', $maxLifeExpectancy);
```

Example 2:

The code below removes the 'minLifeExpectancy' and 'maxLifeExpectancy' variables from the session if the 'resetLifeExpectancyFilter' variable is specified in \$_GET.

```
if (GetApplication()->IsGetValueSet('resetLifeExpectancyFilter')) {
    GetApplication()->UnSetSessionVariable('minLifeExpectancy');
    GetApplication()->UnSetSessionVariable('maxLifeExpectancy');
}
```

11.2.2.8 GetEnvVar

Returns the value of a specified environment variable of an application.

Signature:

```
function GetEnvVar($name)
```


Example:

The following example demonstrates how to get IP address from which the user is viewing the current page.

```
$ipAddress = GetApplication()->GetEnvVar('REMOTE_ADDR');
```

11.2.3 class EngConnection

The following methods of class EngConnection might be useful for application developers:

- [ExecScalarSQL](#)^[318]
- [ExecSQL](#)^[318]
- [ExecQueryToArray](#)^[318]
- [GetLastInsertId](#)^[319]

11.2.3.1 ExecScalarSQL

Executes a query returning a single value.

Signature:

```
function ExecScalarSQL($sql)
```

Example:

```
$userId = $this->GetCurrentUserId();
$sql = "SELECT status FROM phpgen_users WHERE user_id = $userId";
$userStatus = $this->GetConnection()->ExecScalarSQL($sql);
```

11.2.3.2 ExecSQL

Executes a query that does not return results.

Signature:

```
function ExecSQL($sql)
```

Example:

```
$sql = "INSERT INTO customer (first_name, last_name, email) VALUES ('John', 'Doe', 'john_doe@example.com')";
$this->GetConnection()->ExecSQL($sql);
```

11.2.3.3 ExecQueryToArray

Execute a query that returns a result set.

Signature:

```
function ExecQueryToArray($sql, &$array)
```

Example:

```
$queryResult = array();
$sql = "SELECT first_name, last_name, email FROM customer WHERE active = 1";
$this->GetConnection()->ExecQueryToArray($sql, $queryResult);

foreach ($queryResult as $row) {
    $customerFirstName = $row['first_name']; // you can get the first name by key
    $customerFirstName = $row[0]; // or by index
}
```



```

$customerLastName = $row['last_name']; // you can get the last name by key
$customerLastName = $row[1]; // or by index

$customerEmail = $row['email']; // you can get the email by key
$customerEmail = $row[2]; // or by index
}

```

11.2.3.4 GetLastInsertId

Returns the auto generated id used in the latest query (usually INSERT) on a table with an autoincrement column.

Signature:

```
function GetLastInsertId()
```

Example:

```

$sql = "INSERT INTO customer (first_name, last_name) VALUES ('John', 'Doe')";
$this->GetConnection()->ExecSQL($sql);
$lastInsertedCustomerId = $this->GetConnection()->GetLastInsertId();

```

11.2.4 class Grid

The following methods of class Grid are available:

- [Columns related methods](#)³¹⁹

11.2.4.1 Columns related methods

The grid has eight main presentations. Specific set of columns can be defined for each of presentation.

The following methods can be used to get a column of each of presentation.

Signature	Description
getViewColumn(\$columnName)	Returns the column in a table grid with the specified name.
getSingleRecordViewColumn(\$columnName)	Returns the column in a view form with the specified name.
getExportColumn(\$columnName)	Returns the column for an export with the specified name.
getPrintColumn(\$columnName)	Returns the column for a print with the specified name.
getCompareColumn(\$columnName)	Returns the column in a comparison page with the specified name.
getInsertColumn(\$columnName)	Returns the column in an insert form with the specified name.
getEditColumn(\$columnName)	Returns the column in an edit form with the specified name.
getMultiEditColumn(\$columnName)	Returns the column in a multi-edit form with the specified name.

The code in the examples below is used in the [OnPreparePage](#)^[149] event handler:

Example 1:

To hide the 'owner' column in a table grid, use the following code:

```
$column = $this->GetGrid()->getViewColumn('owner');
$column->setVisible(false);
```

Example 2:

To set a caption for the 'status' column of a view form, use the following code:

```
$column = $this->GetGrid()->getSingleRecordViewColumn('status');
$column->setCaption('New caption');
```

Example 3:

To set the 'approved' editor to read only in an edit form, use the following code:

```
$column = $this->GetGrid()->getEditColumn('approved');
$column->SetReadOnly(true);
```

11.2.5 class PermissionSet

PermissionSet is a simple class that encapsulates object permissions and provide methods to query/change all or certain permissions. An instance of this class is passed to the [OnGetCustomPagePermissions](#)^[191] event to customize page-level permissions.

Querying permissions

The following methods can be used to query the current state of a class instance. All functions return Boolean.

Signature	Description
hasViewGrant()	Returns whether the current user can view records.
hasAddGrant()	Returns whether the current user can add records.
hasEditGrant()	Returns whether the current user can edit records.
hasDeleteGrant()	Returns whether the current user can delete records.
hasAdminGrant()	Returns whether the current user has admin privileges for the current page.

Changing permissions

The following methods can be used to change the current state of a class instance. All parameters are Boolean.

Signature	Description
setViewGrant(\$value)	Allows the current user to view records.
setAddGrant(\$value)	Allows the current user to add records.

setEditGrant(\$value)	Allows the current user to edit records.
setDeleteGrant(\$value)	Allows the current user to delete records.
setAdminGrant(\$value)	Grants the current user admin privileges for the current page.
setGrants(\$view, \$add, \$edit, \$delete)	Sets all page permissions at once.

11.2.6 class PageList

PageList is a class that encapsulates the application menu. An instance of this class is passed to the [OnCustomizePageList](#)^[136] event.

The following methods of the PageList class might be useful for application developers:

- [addGroup](#)^[321]
- [addGroupAt](#)^[321]
- [addPage](#)^[321]

11.2.6.1 addGroup

Adds a new group to the application menu.

Signature:

```
function addGroup($caption)
```

Parameters:

\$caption	A caption of the new group.
-----------	-----------------------------

Example:

```
$pageList->addGroup('External links');
```

11.2.6.2 addGroupAt

Inserts a new group into the application menu.

Signature:

```
function addGroup($caption, $index)
```

Parameters:

\$caption	A caption of the new group.
\$index	0-based position of the new group in the application menu.

Example:

```
$pageList->addGroup('External links', 1);
```

11.2.6.3 addPage

Adds a new page to the application menu.

Signature:

```
function addPage(PageLink $page)
```

Parameters:

\$page	An instance of the PageLink class.
--------	------------------------------------

Example:

```
$pageList->addPage(new PageLink('Home Site', 'http://www.mysite.com',
    'Vist my site', false, false, 'External links'));
```

Signature of the PageLink class constructor is as follows:

```
function __construct($caption, $link, $hint = '',
    $showAsText = false, $beginNewGroup = false, $groupName = '');
```

Parameters:

\$caption	A caption of the new page.
\$link	A webpage the new menu item is linked to.
\$hint	A popup hint to be displayed for the new menu item.
\$showAsText	Always false (used only for internal needs).
\$beginNewGroup	If true, a separator is added to the menu before this item.
\$groupName	A caption of the group the new menu item is added to.

11.2.7 Global functions

The following global functions might be useful for application developers:

- [GetApplication](#) ³²²
- [sendMailMessage](#) ³²²
- [createConnection](#) ³²³

11.2.7.1 GetApplication

Returns an instance of the [Application](#) ³¹⁴ class. Actually it is implemented via the Singleton pattern, so the same instance always will be returned.

Signature:

```
function GetApplication()
```

Example:

To check if the parameter *custom_var* is available, use the following code:

```
if (GetApplication()->IsGETValueSet('custom_var'))
    // found
    $params['custom_var'] = GetApplication()->GetGETValue('custom_var');
else
    // not found. Assign a default value.
    $params['custom_var'] = '5 (default)';
```

11.2.7.2 sendMailMessage

Sends an email message to specified recipients using [project-level outgoing mail settings](#) ²³⁵.

Signature:


```
function sendMailMessage($recipients, $messageSubject, $messageBody,
    $attachments = '', $cc = '', $bcc = '')
```

Parameters:

Name	Data type	Description	Mandatory
\$recipients	string or array	A recipient or a list of recipients for the email message	yes
\$messageSubject	string	A subject of the email message	yes
\$messageBody	string	A body of the email message. HTML is allowed.	yes
\$attachments	string or array	An attachment or a list of attachments for the email message	no
\$cc	string or array	A recipient or a list of recipients specified in the CC field	no
\$bcc	string or array	A recipient or a list of recipients specified in the BCC field	no

Example 1

This example demonstrates the simplest use of the function.

```
sendMailMessage('recipient@example.com', 'Message subject', 'Message body');
```

Example 2

This example demonstrates an advanced use of the function.

```
$messageSubject = 'Welcome!!!';
$messageBody = file_get_contents("external_data/mail_message_body.html");

$attachments =
    array(
        'external_data/files/example.pdf',
        'external_data/images/example.png'
    );
$cc = array('mark_lynch@example.com', 'david_frost@example.gov');
$bcc =
    array(
        'dougie_jones@example.edu' => 'Dougie Jones',
        'bob_cooper@example.com' => 'Bob Cooper',
        'laura_horne@example.en' => 'Laura Horne'
    );

sendMailMessage('admin@example.com', $messageSubject, $messageBody, $attachments, $cc, $bcc);
```

See also: [Setting up common SMTP servers](#)^[326].

11.2.7.3 createConnection

Creates a new connection to the database. Returns an instance of the [EngConnection](#)^[318] class.

Signature:


```
function createConnection()
```

Example

The code snippet below shows how to add a record to a log table when a user [resets his password](#)^[266].

```
$connection = createConnection();
```

```
$sqlTemplate =
```

```
    "INSERT INTO user_log (username, action_type, action_time) " .  
    "VALUES ('%s', '%s', '%s')";
```

```
$sql = sprintf($sqlTemplate, $username, 'password_reset_complete', date('Y-m-d H:i:s'));
```

```
$connection->Connect();
```

```
$connection->ExecSQL[318]($sql);
```


11.3 Style sheets internals

Style sheets structure

PHP Generator's style sheets are made with [LESS](#), a dynamic style sheet language that can be compiled into Cascading Style Sheets (CSS). The choice of Less is motivated by such advantages of this language as variables, mixins, functions and many other techniques that simplify the development of large style sheets.

All .less files are stored in the components/assets/less folder of the output directory. The main.less file references (directly or indirectly) all styles defined in other files and subdirectories of this folder as well as user-defined styles, which are stored in components/assets/less/user.less. Current color theme is defined in components/assets/less/theme.less. When you click "Generate" in PostgreSQL PHP Generator, the software automatically compiles main.less and saves the results to components/assets/css/main.css.

Manual compilation of LESS files

You might want to compile .less files manually. To do so, you will need to compile main.less using any .less compiler you like. For example, you can use dotless.Compiler (a command-line compiler that comes with PHP Generator and can be found in SQL Maestro Group\DotLess inside the Common Files directory) using the following command:

```
dotless.compiler.exe -r -m full/path/to/main.less
```

Visit [dotless.Compiler wiki](#) to find more info on using this tool.

11.4 Setting up common SMTP servers

If you plan to use Google Mail, Yahoo, or Hotmail/Outlook mail servers for sending outgoing emails, the following information could be useful for you.

Google Mail

To allow sending emails, go to My account->Sign-in & security->Connected apps & sites and turn on the 'Allow less secure apps' option.

Host: smtp.gmail.com

Port: 465

Use authentication: Yes

Username: Full Gmail email address(name@domain.com)

Password: Your account's password.

Encryption: ssl

Yahoo

To allow sending emails, go to Account->security and turn on the 'Allow apps that use less secure sign in' option.

Host: smtp.mail.yahoo.com

Port: 465

Use authentication: Yes

Username: Full Yahoo email address(name@domain.com)

Password: Your account's password.

Encryption: ssl

Hotmail/Outlook

No additional settings required.

From mail: Existing Outlook (Hotmail) email address

Host: smtp.live.com

Port: 587

Use authentication: Yes

Username: Full Outlook (Hotmail) email address(name@domain.com)

Password: Your account's password.

Encryption: tls

12 Options

PostgreSQL PHP Generator allows you to customize the way it works within the [Options](#) dialog. To open the dialog, use the More button and select Options at the drop-down list.

The window allows you to customize the options grouped by the following sections:

- [Application](#) ³²⁸
General PostgreSQL PHP Generator options: generation rules, default page options, and display data formats.
- [Editors & Viewers](#) ³⁴⁰
Customizing of all the SQL editors.
- [Appearance](#) ³⁴⁸
Customizing program interface - bars, trees, menus, etc.

It is a good idea to check through these settings before you start working with PostgreSQL PHP Generator. You may be surprised at all the things you can adjust and configure!

12.1 Application

The [Application](#) section allows you to customize common rules of PostgreSQL PHP Generator behavior. The section consists of several tab; follow the links to find out more about each of them.

- [Page](#)^[328]
- [Generation defaults](#)^[335]
- [Display formats](#)^[336]
- [Output](#)^[337]

12.1.1 Page

This tab allows you to specify default page options to be applied to all the generated pages in all applications. These settings can be overwritten for a project as well as for a certain page.

Common

These options are applied for all generated pages.

[Content encoding](#)

The default option value is UTF-8 and you must have really good reasons to change it.

[Page direction](#)

Allows you to select between Left-to-right and Right-to-left page directions.

[Modal form size](#)

A preferred size (default, small, or large) for View, Edit, and Insert modal windows.

[Hide sidebar menu by default](#)

If checked, the sidebar menu will be closed until a user opens it manually.

List

These options are applied for List views (both grid and card).

[Page navigator](#)

Allows you to specify the position of the navigational control(s) and the default number of records displayed on the page. This value may be changed by application users in runtime if the [Runtime Customization](#)^[334] option is enabled.

The screenshot shows a table with columns: Actions, Film Id, Title, and Description. The first three rows are visible, each with a set of action icons (edit, delete, copy) and a 'more' link. Above the table, there is a navigation bar with page numbers (1, 2, ..., 11, ..., 50) and a 'Runtime customization button' (gear icon) in the top right corner. The text 'Navigation elements' is placed above the navigation bar, and 'Runtime customization button' is placed above the gear icon.

To create a custom pagination, use [Data Partitioning](#)^[220].

View mode

Select whether page data are displayed in a grid or as info cards. The end-user can easily switch from the grid mode to the card one and vice versa using the Page Settings dialog (if the [Runtime Customization](#)^[334] option is enabled).

Grid view mode

The screenshot shows the NBA website's 'Players' section in grid view. It displays a table with columns: Photo, First name, Last name, Country, Team, and Actions. The first six rows are visible, each with a player's photo and details. The text 'Players' is displayed above the table.

Cards view mode

The screenshot shows the NBA website's 'Players' section in cards view. It displays two cards, each with a player's photo and details. The text 'Players' is displayed above the cards.

Number of cards in a row

Allows you to specify maximum number of cards displayed in a row for each screen resolution. These values also can be changed by end-users if the [Runtime Customization](#)^[334] option is enabled. [Live Example](#).

Control buttons position

Select whether the Edit, View, Delete, and Copy buttons to be displayed on the left side

of the generated page or on the right side. [Live Example](#).

Bordered table

Check this option to add borders on all sides of the table and cells. [Live Example](#).

Condensed table

Turn this option 'ON' to make tables more compact by cutting cell padding in half. [Live Example](#).

Highlight row at mouse over

Allows you to enable/disable the highlighting of rows at mouse over.

Use images for actions



Defines whether images or text captions are used for control buttons (like Edit, View, or Delete).

Use fixed grid header

Allows you to make the grid header are non-scrollable. [Live Example](#).

Show key columns images

Turn is 'ON' to mark key columns with the 'key' images.

 City Id	City	 Country Id	Last Update
1	A Corua (La Corua)	Spain	2006-02-14 16:45:25
2	Abha	Saudi Arabia	2006-02-14 16:45:25
3	Abu Dhabi	United Arab Emirates	2006-02-14 16:45:25

Show line numbers

Check this option to add row numbering to the grid. [Live Example](#).

Edit/Insert

These options are applied to data input forms.

Display "Set NULL" checkboxes

Defines whether "Set NULL" checkboxes are displayed for each control in Edit and Insert forms.

Display "Set Default" checkboxes

Defines whether "Set Default" checkboxes are displayed for each control in Edit and Insert forms.

Error message placement

Allows you to select the placement for error messages. Possible values are "Below the form", "Above the form", and "Below and above the form". Default value is "Below the form".

Reload page after Ajax operations

Allows you to reload the page after [Inline](#) or [Modal](#) editing/inserting.

12.1.1.1 Export and Print

This tab allows you to specify default exporting and printing options to be available for all the generated pages in all applications. These settings can be overwritten for a project as well as for a certain page.

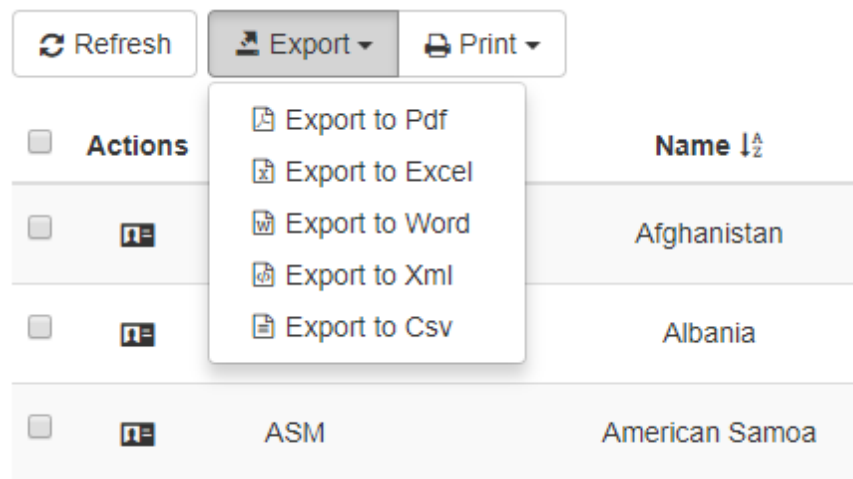
The screenshot shows the 'Page: Export and Print' dialog box. The 'Appearance' tab is selected. The left sidebar shows a tree view with 'Page', 'Abilities', 'Export and Print' (selected), 'Generation rules', 'Display formats', 'Output', and 'Confirmations'. The main area contains the following settings:

Section	Export	Print
All records from grid	Pdf,Excel,Word,Xml,Csv	<input checked="" type="checkbox"/>
Selected records from grid	Pdf,Excel,Word,Xml,Csv	<input type="checkbox"/>
Single record from grid	None selected	<input type="checkbox"/>
Record from view form	Pdf,Excel,Word,Xml,Csv	<input checked="" type="checkbox"/>
Options		
Open print form in new tab		<input checked="" type="checkbox"/>
Open exported Pdf in new tab		<input checked="" type="checkbox"/>

At the bottom of the dialog are 'OK', 'Cancel', and 'Help' buttons.

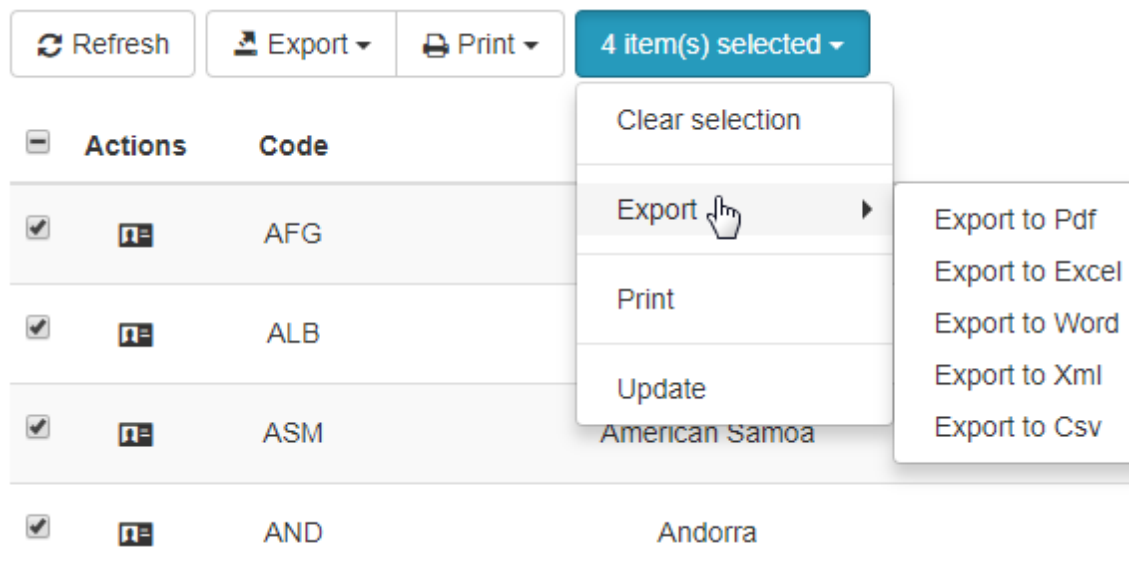
All records from grid

Defines exporting and printing options to be available for the data grid.


















Selected records from grid

Defines exporting and printing options to be available for the selected records in the data grid.



Single record from grid

Defines exporting and printing options to be available for a single record in the data grid.

<input type="checkbox"/>	Actions	Code
<input type="checkbox"/>	  	ABW
<input type="checkbox"/>	  	AFG
<input type="checkbox"/>	  	AGO
<input type="checkbox"/>	  	AIA
<input type="checkbox"/>	  	ALB

[Record from view form](#)
Defines exporting and printing options to be available for a single record view form.

View

← Back to list

Export ▾

Print

Code

AFG

Name

Afghanistan

Continent

Asia

Indepyear


1,919


Population


22,720,000


Lifeexpectancy


45.90

 Export to Pdf

 Export to Excel

 Export to Word

 Export to Xml

 Export to Csv

[Options](#)
The [Open print form in new tab](#) option allows you to open print forms in a new browser tab. The [Open exported PDF in new tab](#) option allows you to open exported PDFs in a new browser tab. The both options are enabled by default.

12.1.1.2 Abilities

This tab allows you to specify default operations to be available for all the generated pages in all applications. These settings can be overwritten for a project as well as for a certain page.

Actions

View

Defines whether browsing a single record is available. Possible values are Disabled, Separated Page (default value), and Modal window.

Edit

Defines whether data editing is available. Possible values are Disabled, [Separated Page](#) (default value), [Inline mode](#), and [Modal window](#).

Quick Edit

Defines whether the [quick edit](#) mode is available. Possible values are List and view (quick editing is available in both data grid and single record view window), List only, View only, and None.

Multi-edit

Defines whether [mass data editing](#) is available. Possible values are Disabled, Separated Page (default value), and Modal window.

Fields to be updated by default

Defines whether all or none fields will be selected in the appropriate control when the multi-edit command is executed.

Insert

Defines whether data insertion is available. Possible values are Disabled, [Separated Page](#) (default value), [Inline mode](#), and [Modal window](#).

Copying

Defines whether data copying is available. Possible values are Disabled, [Separated Page](#) (default value), [Inline mode](#), and [Modal window](#).

Delete

Defines whether data deletion is available. Possible values are Enabled (default) and Disabled.

Multi-delete

Defines whether multi deletion is available (user can delete multiple records at a time). Possible values are Enabled (default) and Disabled.

Filtering

Quick search

Defines whether the [Quick Search](#) box is displayed. Possible values are Enabled (default) and Disabled.

Filter Builder

Defines whether the [Filter Builder](#) tool is available. Possible values are Enabled (default)

and Disabled.

Column Filter

Defines whether the [Column Filter](#) tool is available. Possible values are Enabled (default) and Disabled. It is also possible to enable/disable this feature at the column level.

Selection Filter

Defines whether [Selection Filters](#) is available. Possible values are Enabled (default) and Disabled.

Sorting

Sorting by click

Defines whether data can be [sorted by click on a column header](#). Possible values are Enabled (default) and Disabled.

Sorting by dialog

Defines whether the [data sort dialog](#) is available. Possible values are Enabled (default) and Disabled.

Additional

Runtime Customization

Defines whether end user is able to customize such page properties as number of records displayed on a page and a number of cards for each screen resolution. Possible values are Enabled (default) and Disabled.

Records comparison

Defines whether the [record comparison](#) tool is available. Possible values are Enabled (default) and Disabled.

Refresh

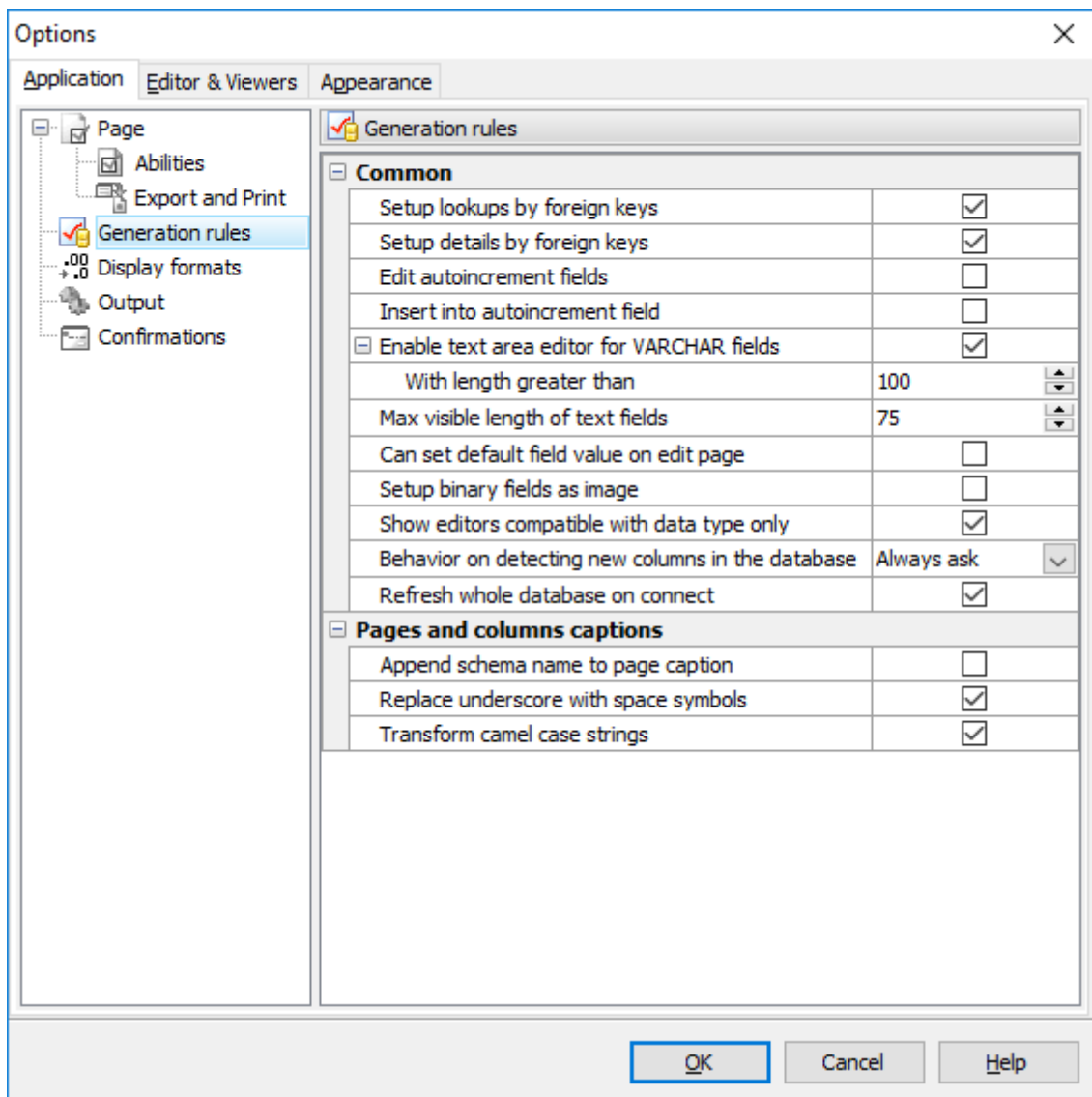
Defines whether the Refresh button is displayed. Possible values are Enabled (default) and Disabled.

12.1.2 Generation rules

Use this tab to define rules to be used for the page generation. These options allow you to enable/disable the automatic implementation of lookup editors and details by foreign keys, the editing of autoincrement fields and inserting data into them, automatic enabling of text area editors for columns storing varchar data with length greater than a specified value and automatic setting Image view control and Image upload edit control for binary fields.

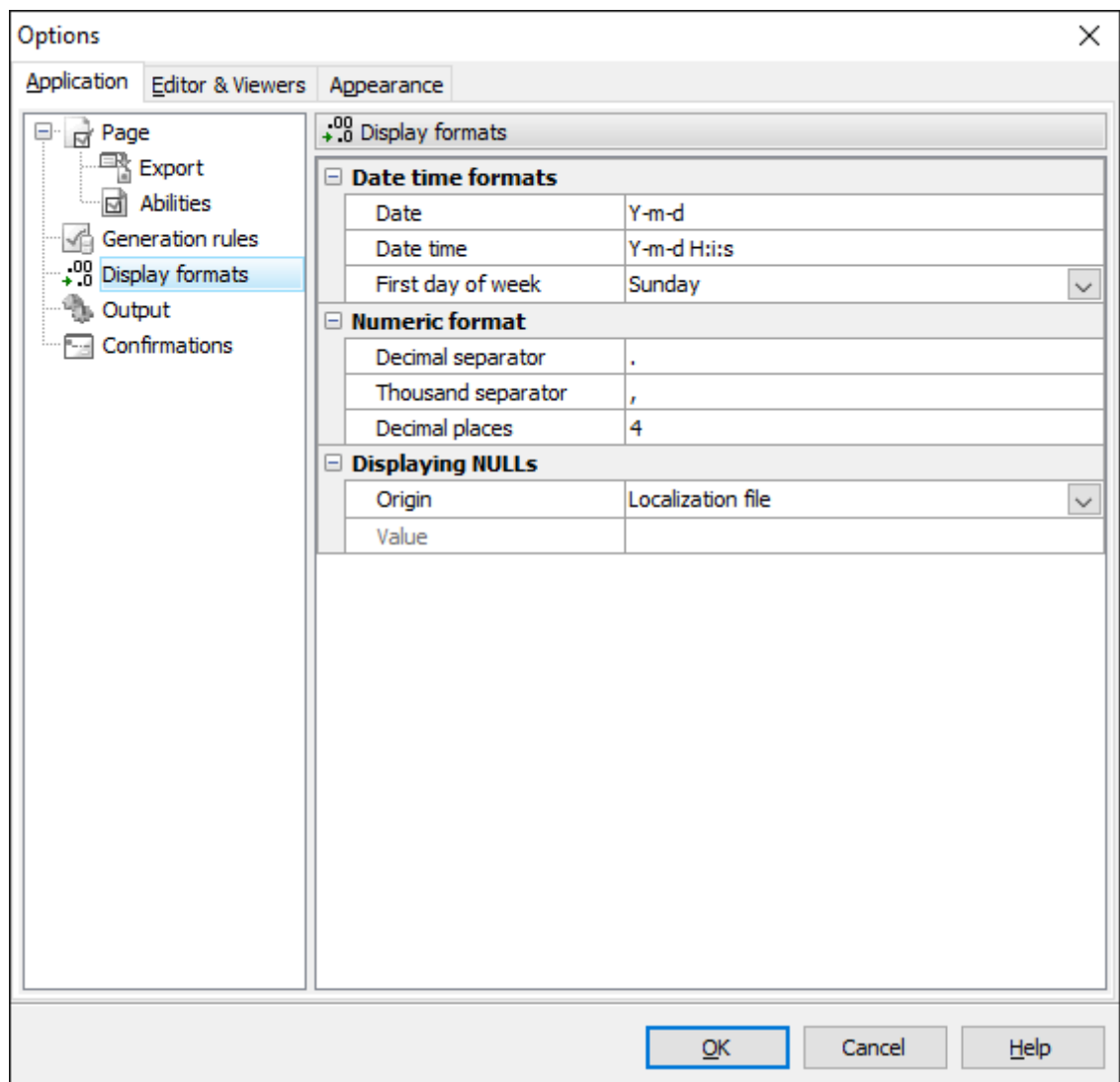
Specify here the [Max visible length of text fields](#) value. If the text length is greater than this value, the residual text will be hidden under the [More...](#) link on the generated page. Full text can be displayed in the special window in this case.

To optimize your work with big databases, uncheck the [Refresh whole database on connect](#) option.



12.1.3 Display formats

Use this tab's fields to adjust the display data formats in the way you need.



12.1.4 Output

Use this tab to specify default values for [project output options](#)^[275].

PHP Driver

There are several PHP drivers to communicate with database servers and PostgreSQL. PHP Generator allows you to select a driver to be used by the generated web application. Note that the corresponding set of PHP functions should be available on your web server.

Localization file

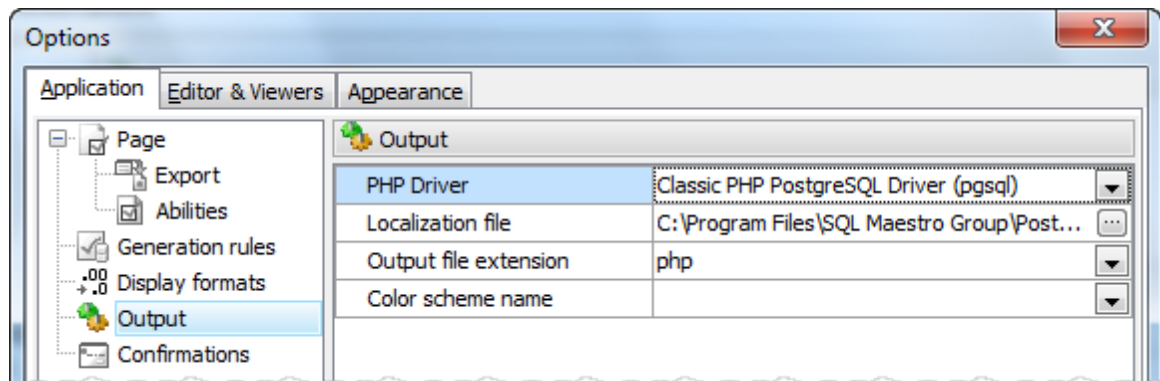
Use this field to define the interface language to be used for the generated application by default. Find out more on applications localization at [the corresponding page](#)^[273].

Default output directory

Use this field to define the default output directory for the generated files.

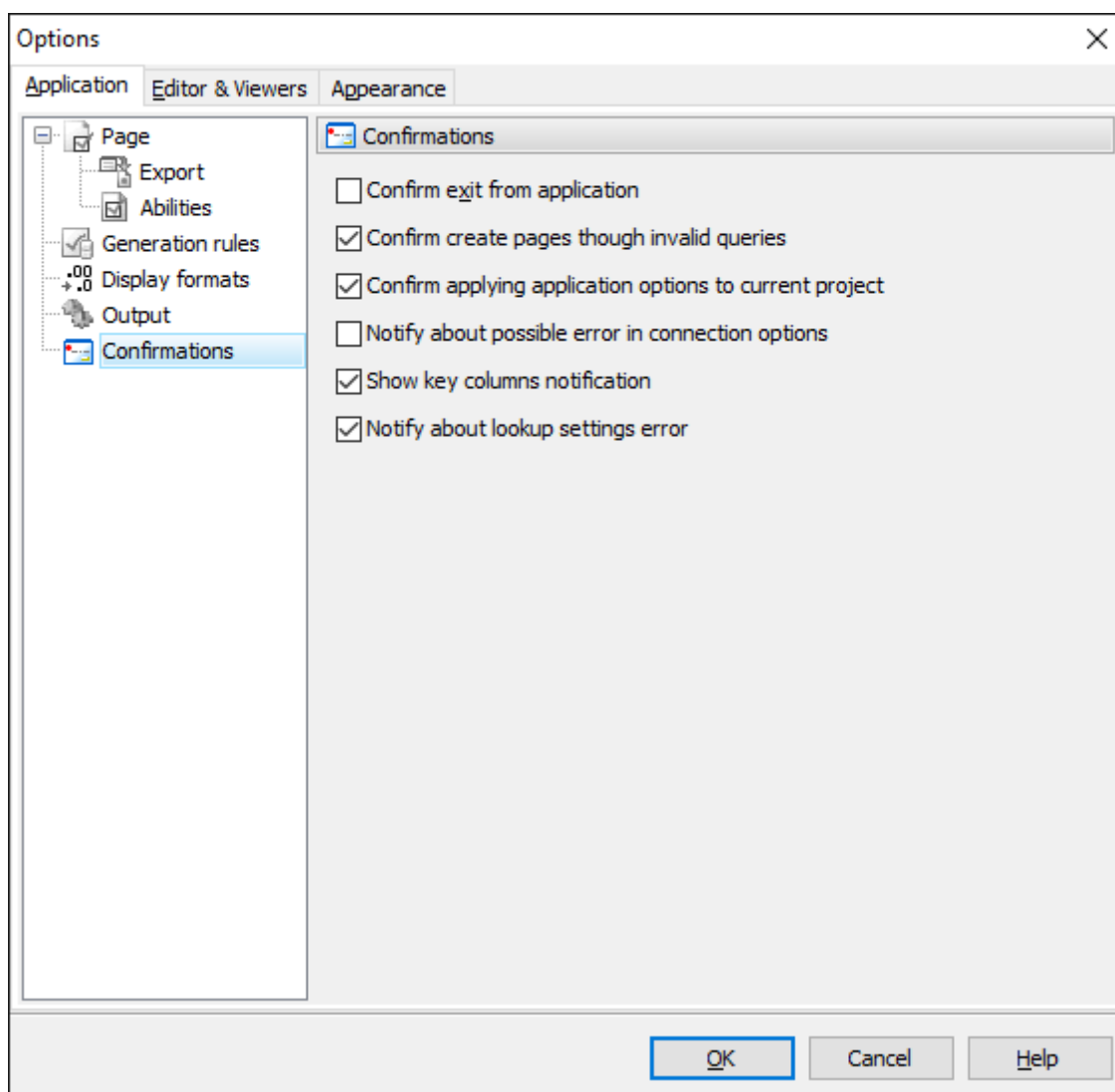
You can also set here the output file extension and the default color scheme for the

generated pages.



12.1.5 Confirmations

This tab allows you to disable/enable application confirmations. Most of these options can be also set directly from the corresponding confirmation dialogs.



12.2 Editors & Viewers

The [Editors & Viewers](#) section allows you to set the parameters of viewing and editing the SQL statements within PostgreSQL PHP Generator.

- [General](#) ³⁴⁰
- [Display](#) ³⁴¹
- [SQL highlight](#) ³⁴²
- [PHP highlight](#) ³⁴⁴
- [XML highlight](#) ³⁴³
- [Code Insight](#) ³⁴⁵
- [Code Folding](#) ³⁴⁶

12.2.1 General

If the [Auto indent](#) option is checked, each new indentation is the same as the previous when editing SQL text.

☒ [Insert mode](#)

If this option is checked, insert symbols mode is default on.

☒ [Use syntax highlight](#)

Enables syntax highlight in the object editor window.

☒ [Always show links](#)

If this option is checked, hyperlinks are displayed in the editor window. To open a link click it with the **Ctrl** button pressed.

☒ [Show line numbers](#)

If this option is checked, line numbers are displayed in the editor window.

☒ [Show special chars](#)

If this option is checked, special chars (like line breaks) are displayed in the editor window.

☒ [Use smart tabs](#)

With this option on the number of tab stops is calculated automatically, depending on the previous line tab.

☒ [Convert tabs to spaces](#)

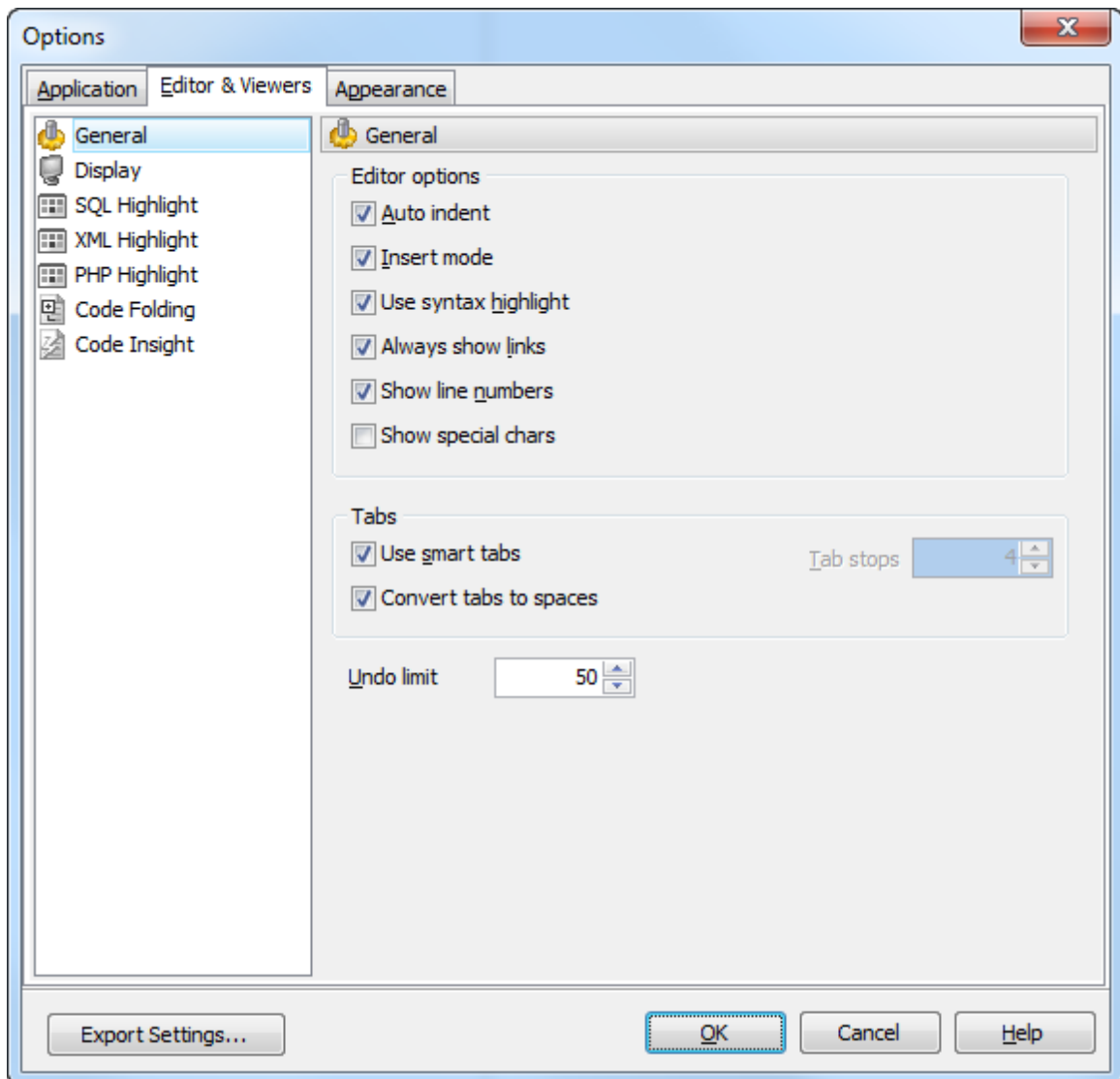
If this option is checked, each time you press the Tab key, the appropriate number of spaces will be added to the edited text.

[Tab Stops](#)

Defines the tab length, used when editing text.

[Undo Limit](#)

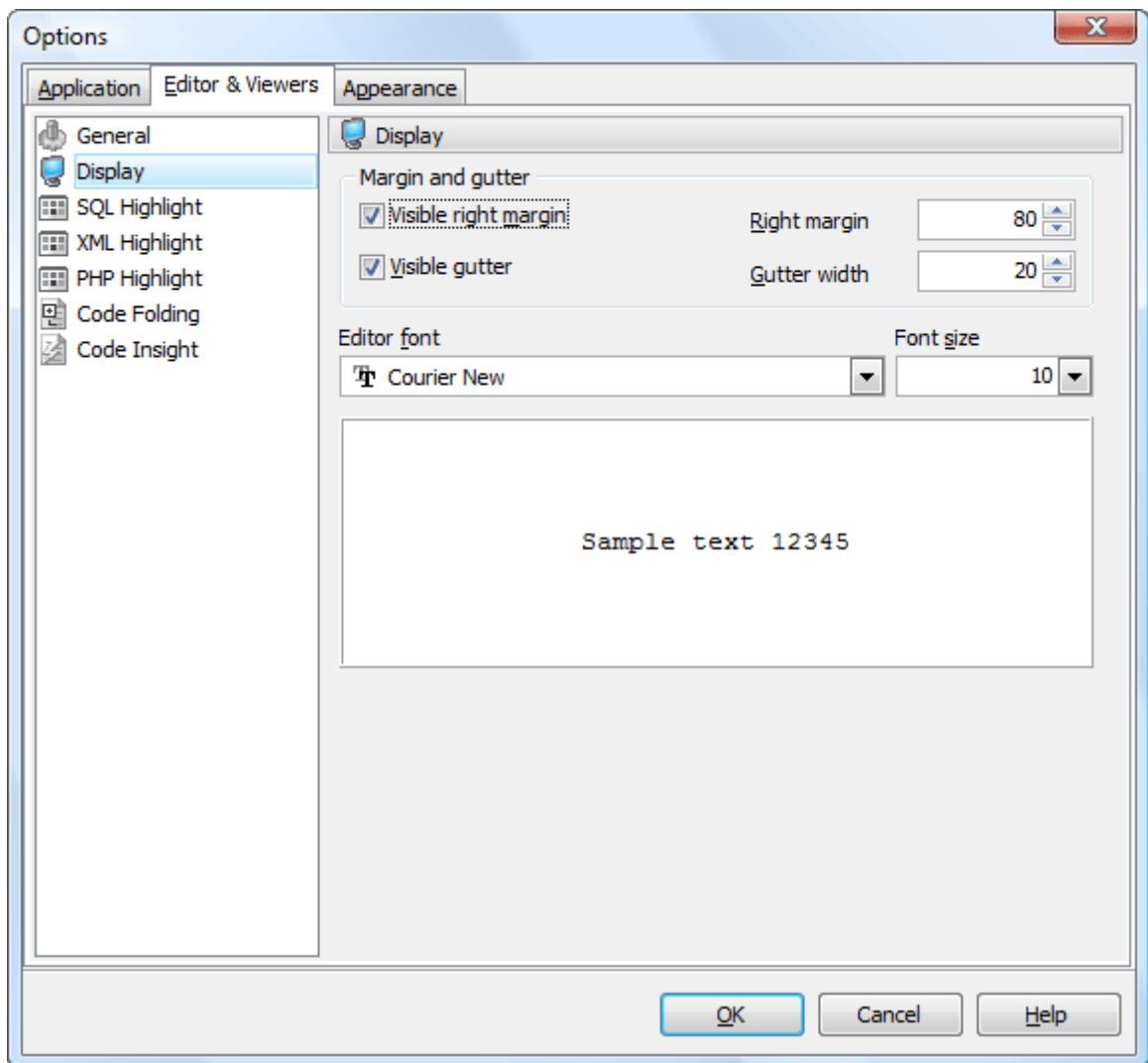
Defines the maximum number of changes possible to be undone.



12.2.2 Display

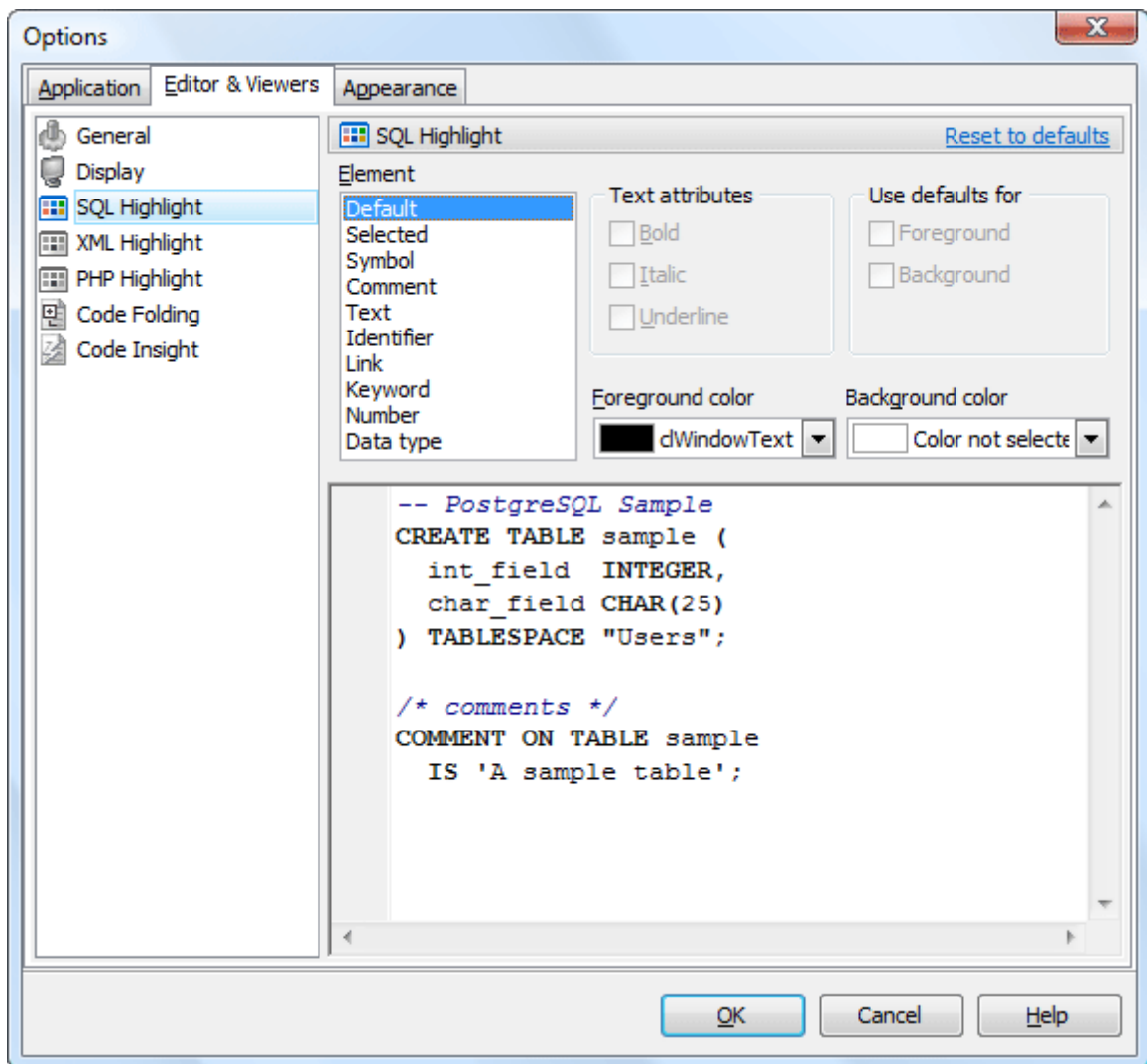
You can disable/enable the right text margin and the gutter of the editor area, set the position of the right text margin as [Right margin](#), and [the Gutter width](#).

Use the [Editor font](#) and [Font size](#) to define the font used in all program editors and viewers. The panel below displays the sample of the selected font.



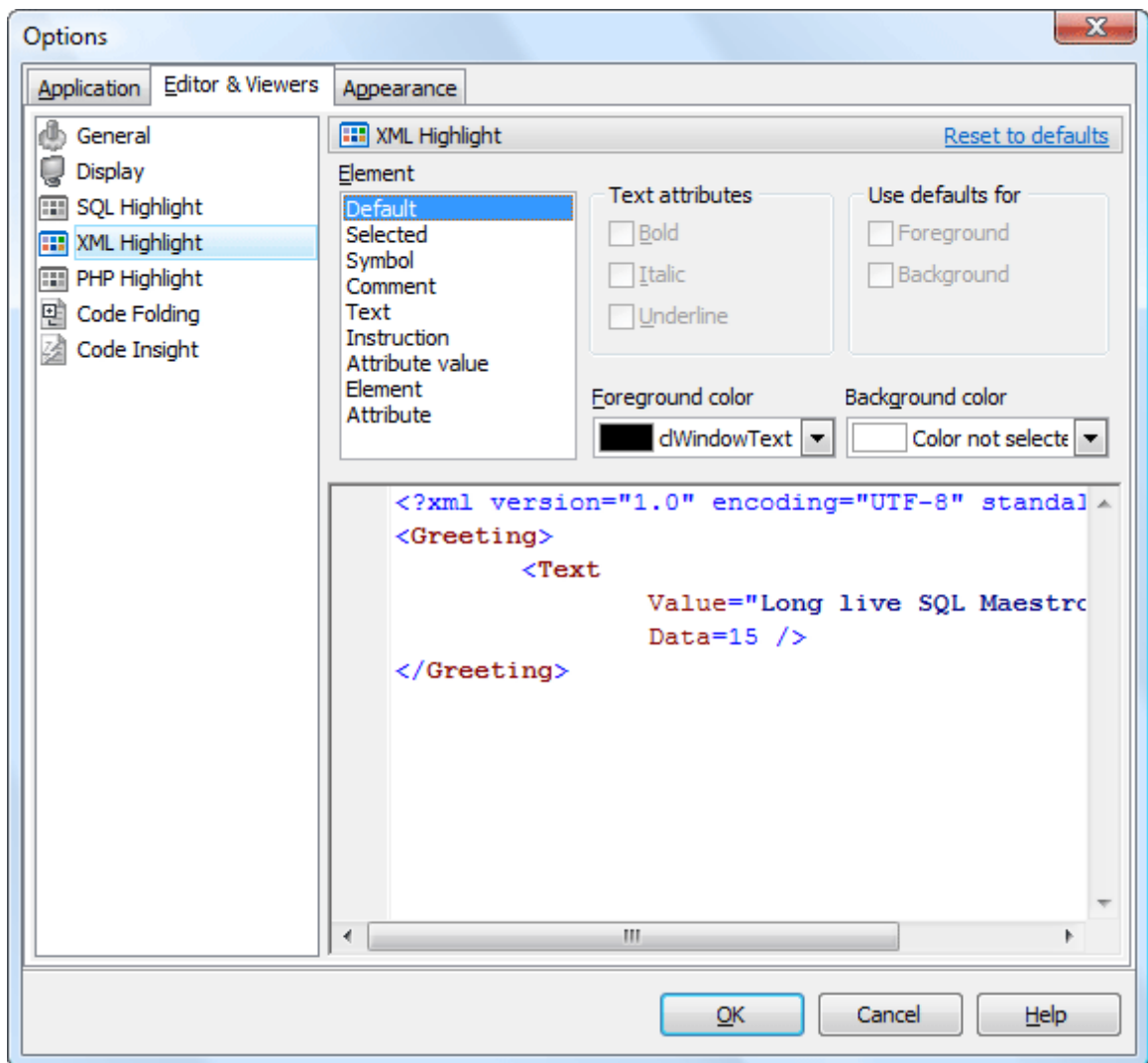
12.2.3 SQL highlight

Use the [SQL highlight](#) item to customize syntax highlight in all SQL editors and viewers. Select the text element from the list, e.g. *comment* or *SQL keyword* and adjust its foreground color, background color and text attributes according to your preferences.



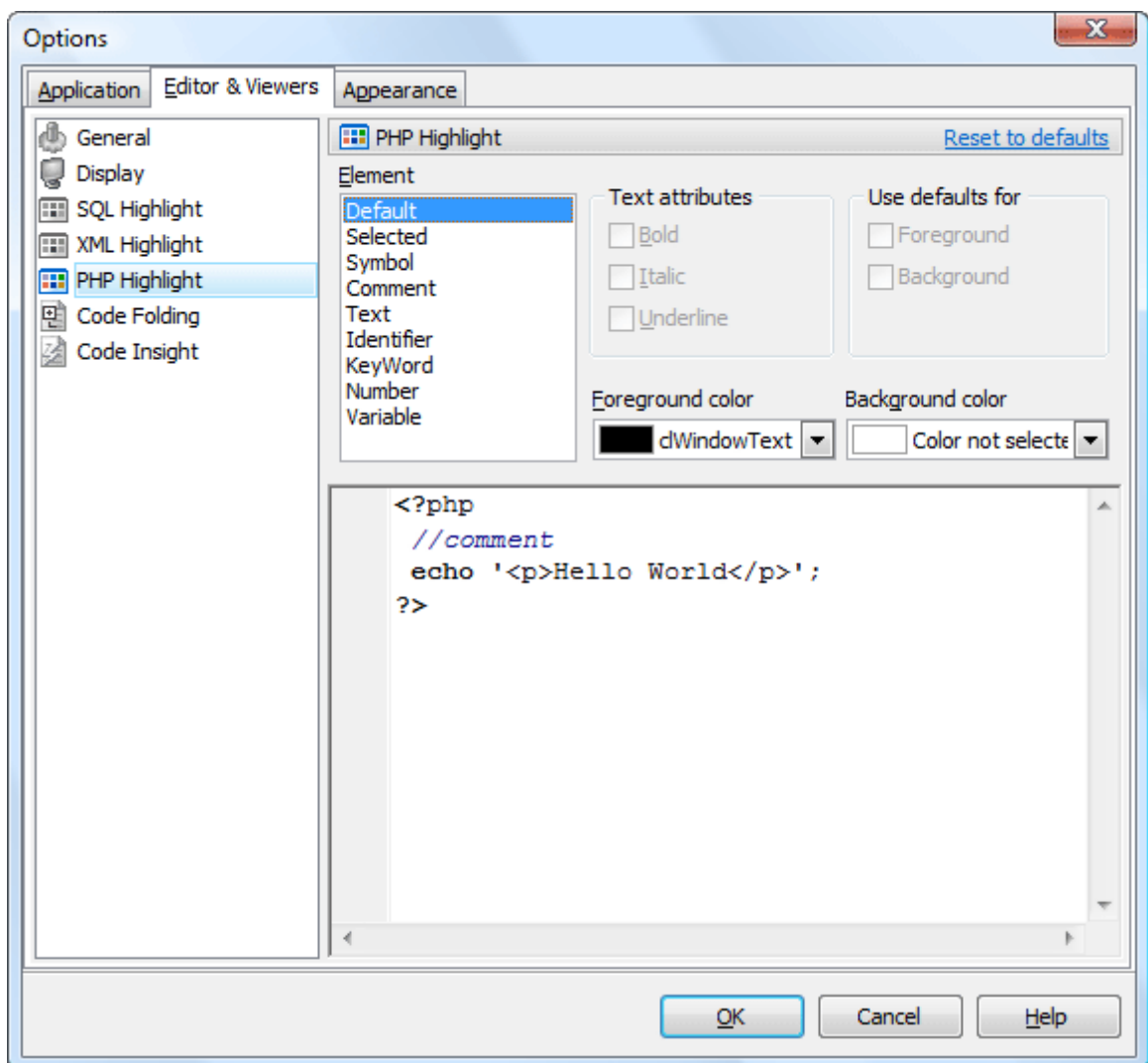
12.2.4 XML highlight

Use the [XML highlight](#) item to customize XML syntax highlight for the text representation of BLOBs. Select the text element from the list, e.g. attribute or attribute value and adjust its foreground color, background color and text attributes according to your wishes.



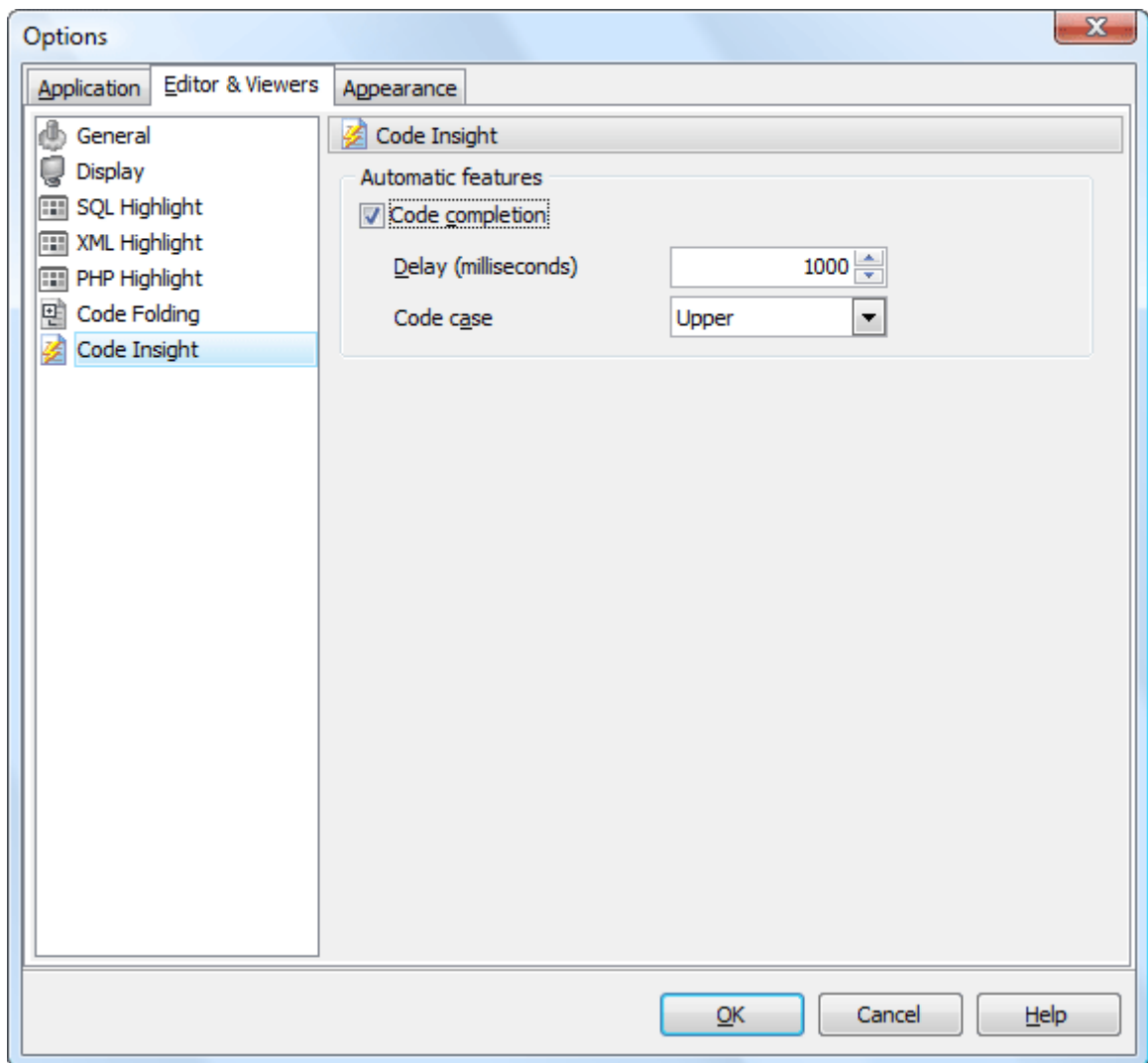
12.2.5 PHP highlight

Select the text element from the list (e.g. Keyword, Comment, Identifier), and adjust its foreground color, background color and text attributes according to your wishes.



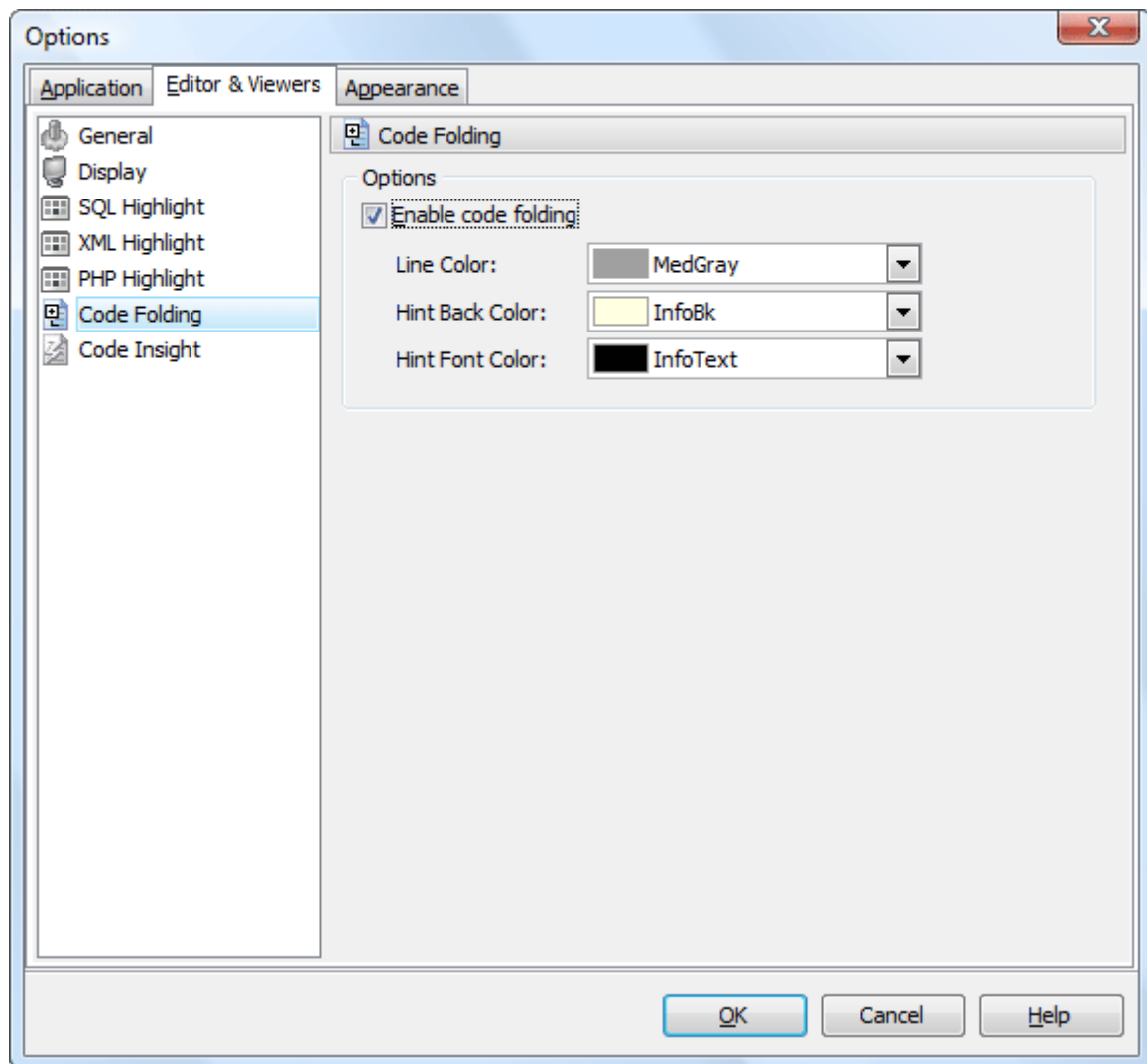
12.2.6 Code Insight

You can disable/enable the code completion with the corresponding option and also set the time it appears as *Delay*, and case of words inserted automatically.



12.2.7 Code Folding

The [Code Folding](#) item group makes it possible both to view the whole text and to divide it into logical parts (regions). Each part can be collapsed and extended. In extended mode the whole text is displayed (set by default), in collapsed mode the text is hidden behind one text line denoting the first line of the collapsed region.



You can enable/disable code folding in SQL editors and viewers and customize the colors of its items.

12.3 Appearance

The [Appearance](#) section allows you to customize the application interface style to your preferences.

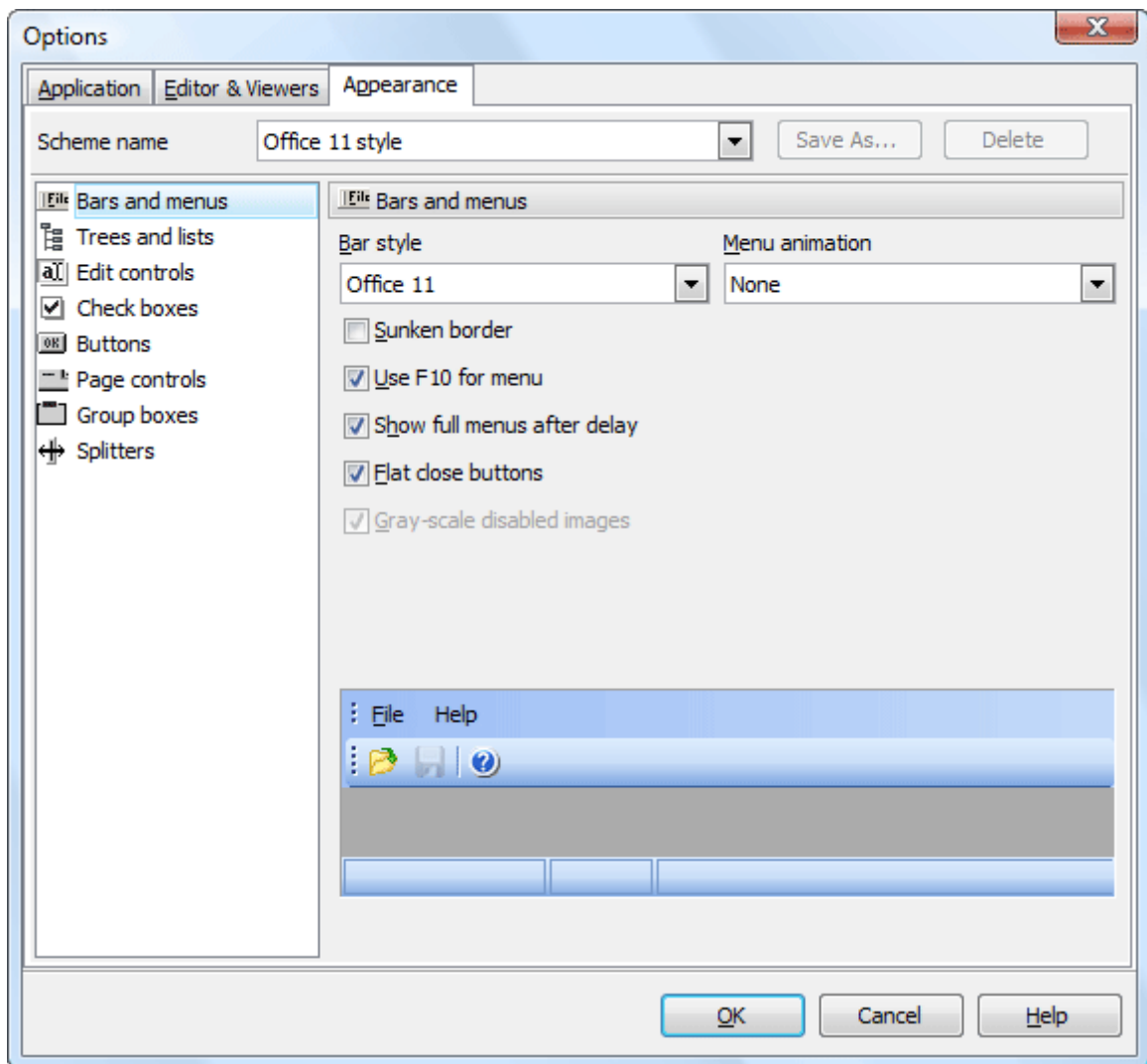
Use the [Scheme name](#) box to select the interface scheme you prefer: *Office XP style*, *Windows XP native style*, etc. You can create your own interface schemes by customizing any visual options ([Bars and menus](#), [Trees and lists](#), [Edit controls](#), [Check boxes](#), [Buttons](#), etc.) and clicking the [Save As](#) button. All the customized options are displayed on the sample panel.

- [Bars and menus](#) ³⁴⁸
- [Trees and lists](#) ³⁴⁹
- [Edit controls](#) ³⁵⁰
- [Check boxes](#) ³⁵¹
- [Buttons](#) ³⁵²
- [Page controls](#) ³⁵³
- [Group boxes](#) ³⁵⁴
- [Splitters](#) ³⁵⁵

12.3.1 Bars and menus

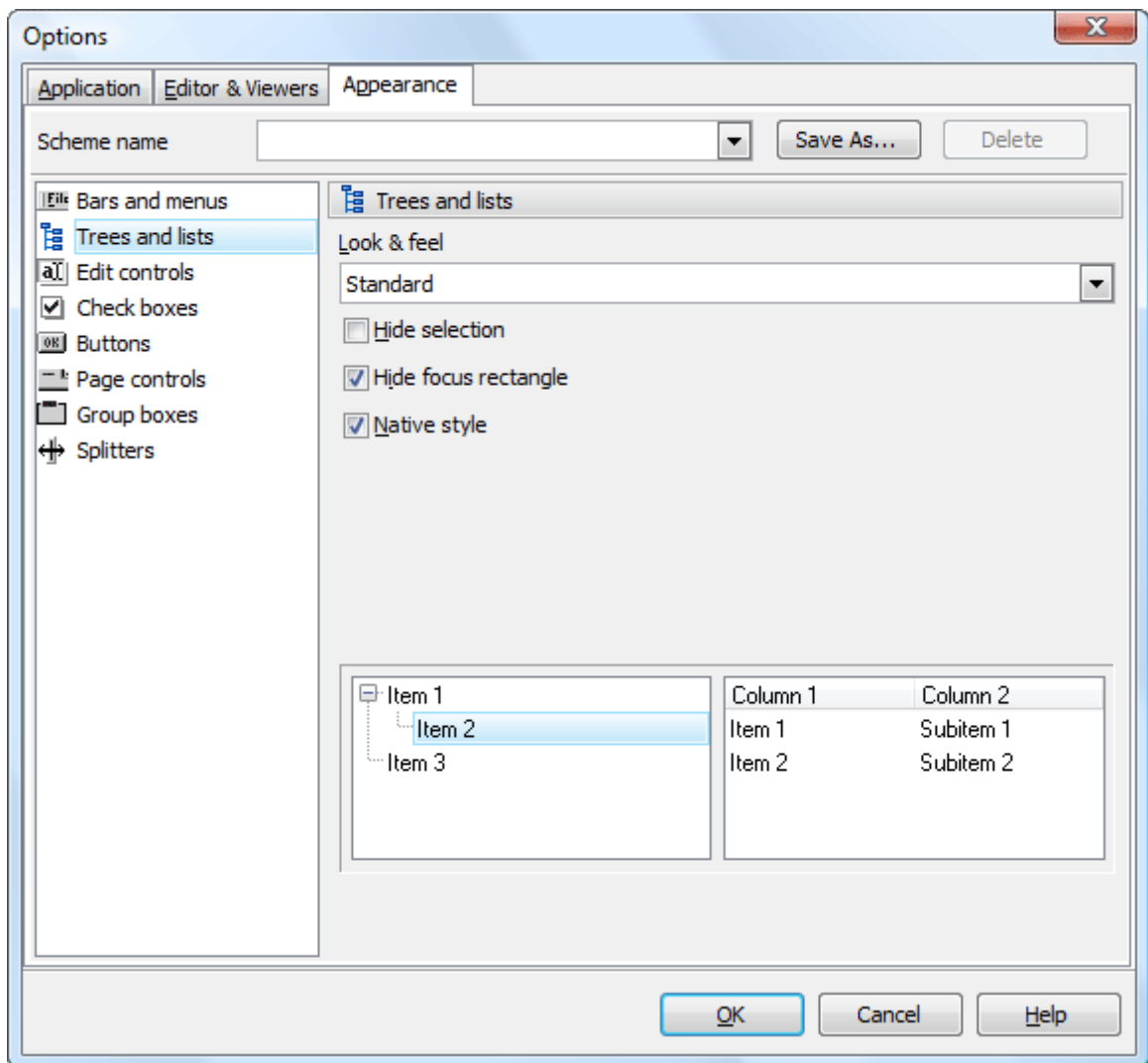
Use the [Bars and menus](#) item to customize PostgreSQL PHP Generator toolbars style and menus animation.

The item allows you to select Bar style and menu animation from the corresponding drop-down lists and to enable or disable such options as sunken border, F10 key for opening menu, viewing full menus after delay, flat close buttons, gray-scale images.



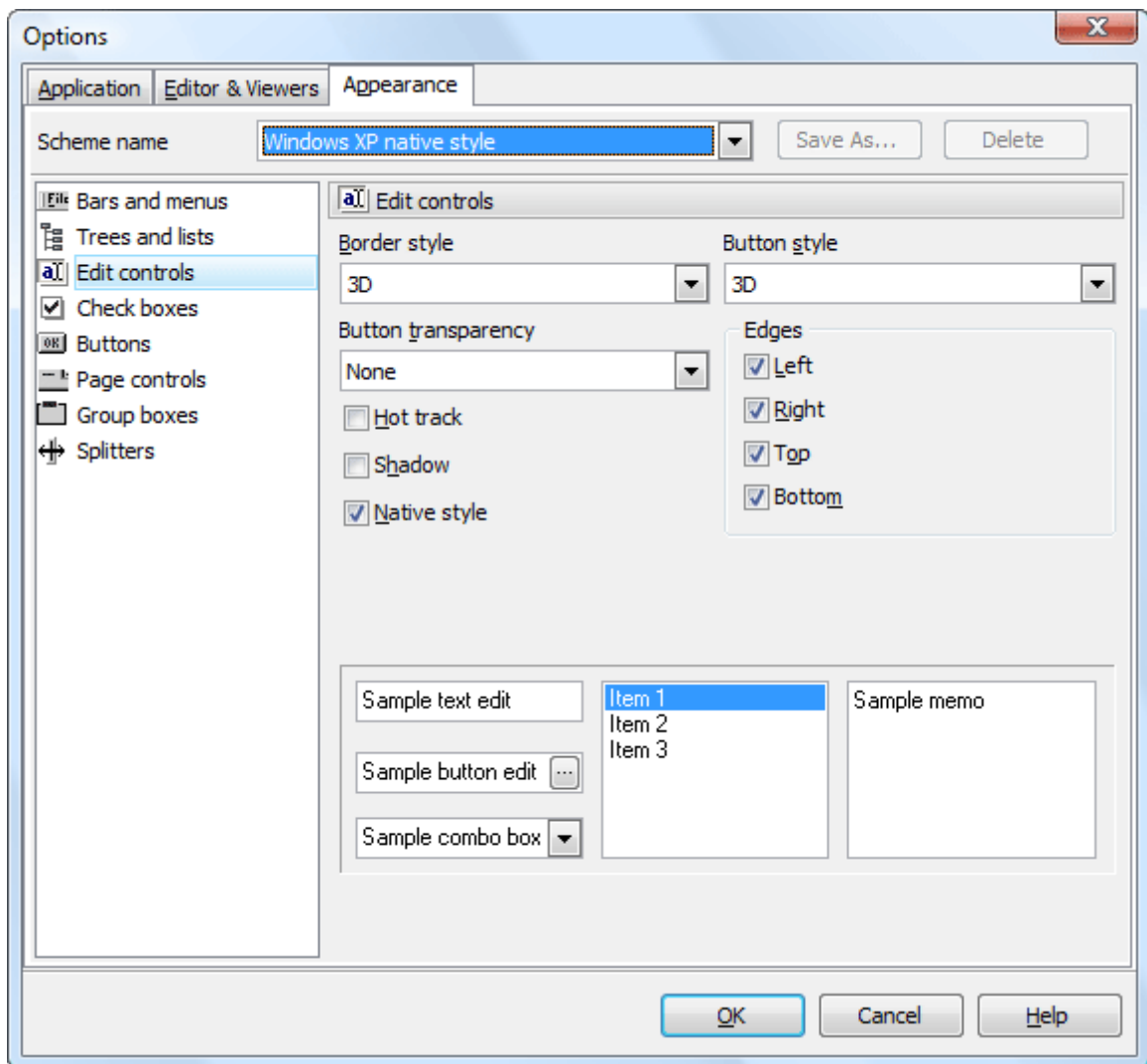
12.3.2 Trees and lists

Use the **Trees and lists** item to select various tree view options. Use the item to select *standard*, *flat* or *ultraflat* styles, check or uncheck the *hide selection*, *hide focus rectangle* and *native style* options.



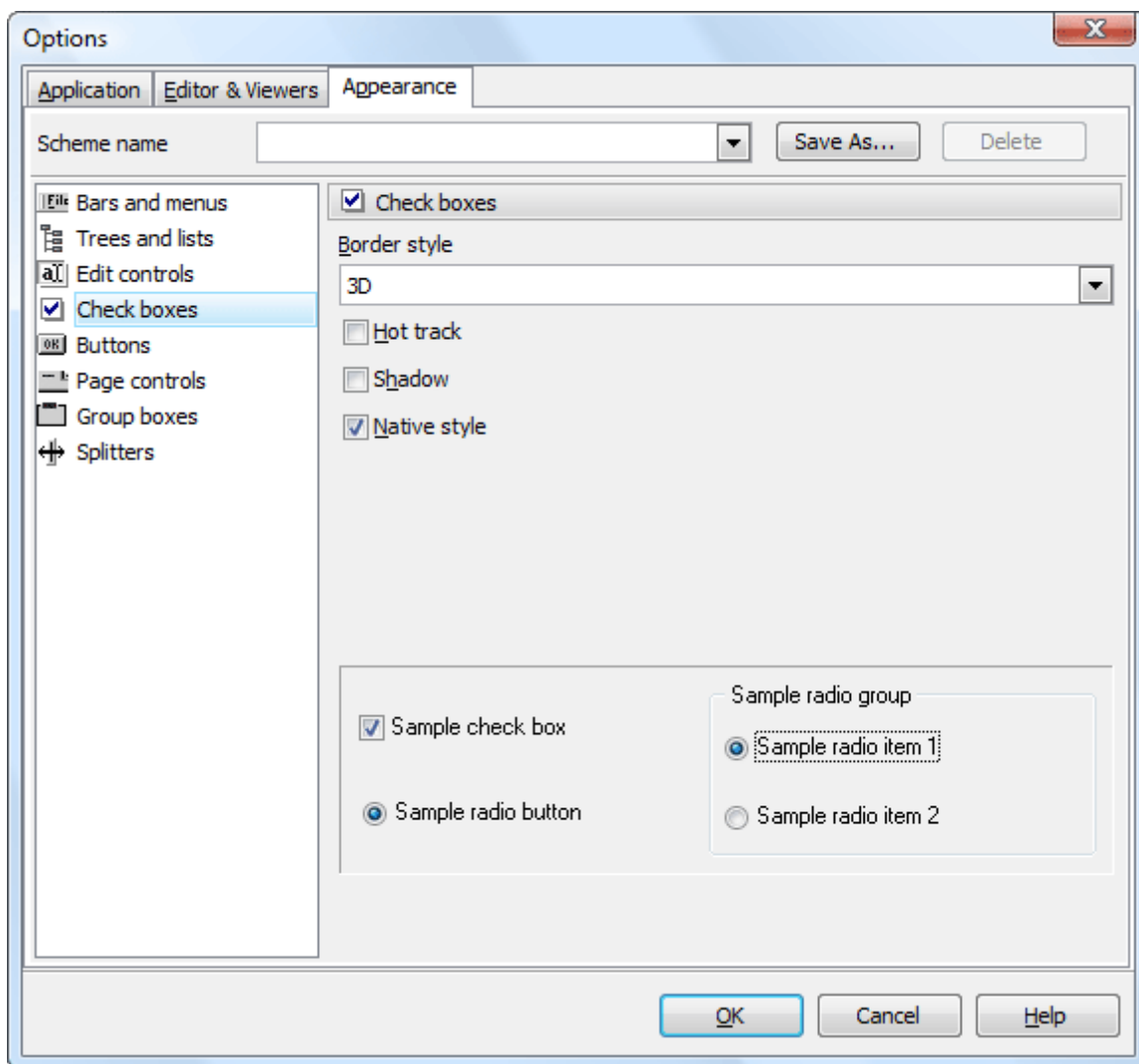
12.3.3 Edit controls

Use the [Edit controls](#) item to customize the appearance of different PostgreSQL PHP Generator edit controls. The tab allows you to select the edit controls border style, button style and transparency, enable/disable hot tracks, shadows, native style and customize edges. It is also possible to define samples for the text edit, button edit and combo box controls.



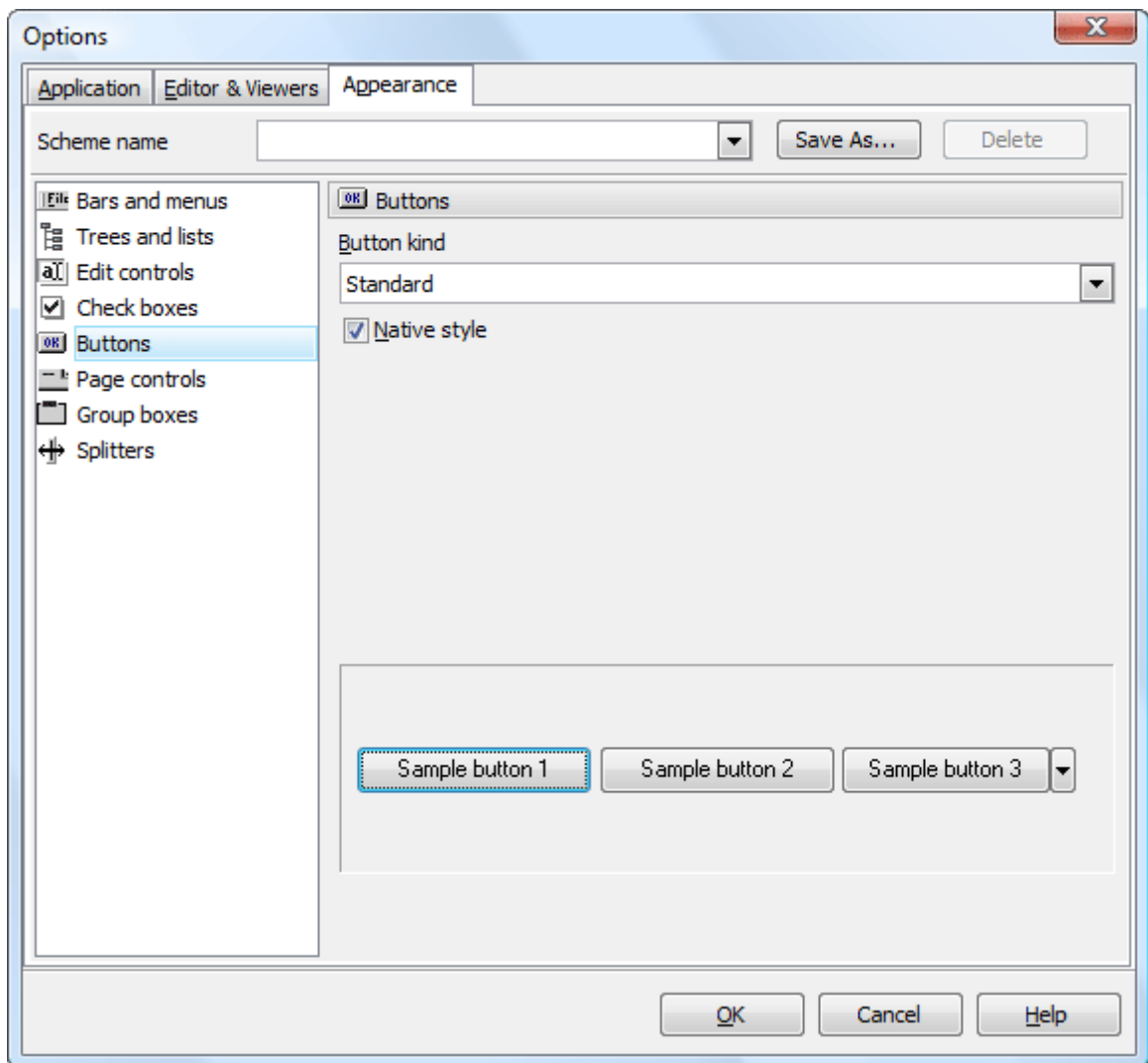
12.3.4 Check boxes

The [Check boxes](#) item allows you to customize the appearance of check boxes and radio buttons. The tab allows you to customize the appearance of check boxes: set border style, enable/disable hot tracks, shadows, native style. It is also possible to define samples for check boxes and radio buttons.



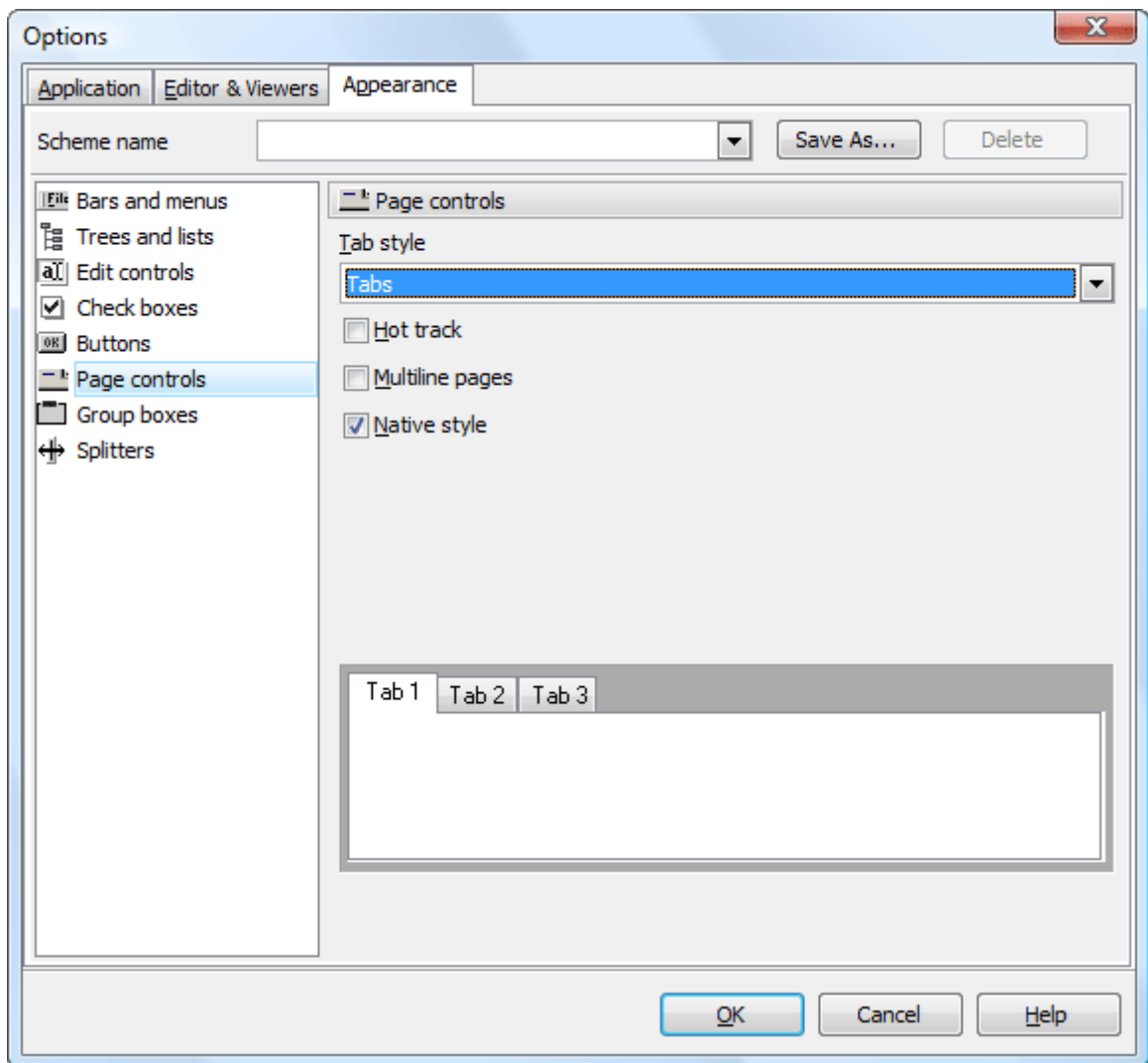
12.3.5 Buttons

Use the [Buttons](#) item to customize PostgreSQL PHP Generator buttons. The tab allows you to adjust the appearance of buttons and define sample buttons as well.



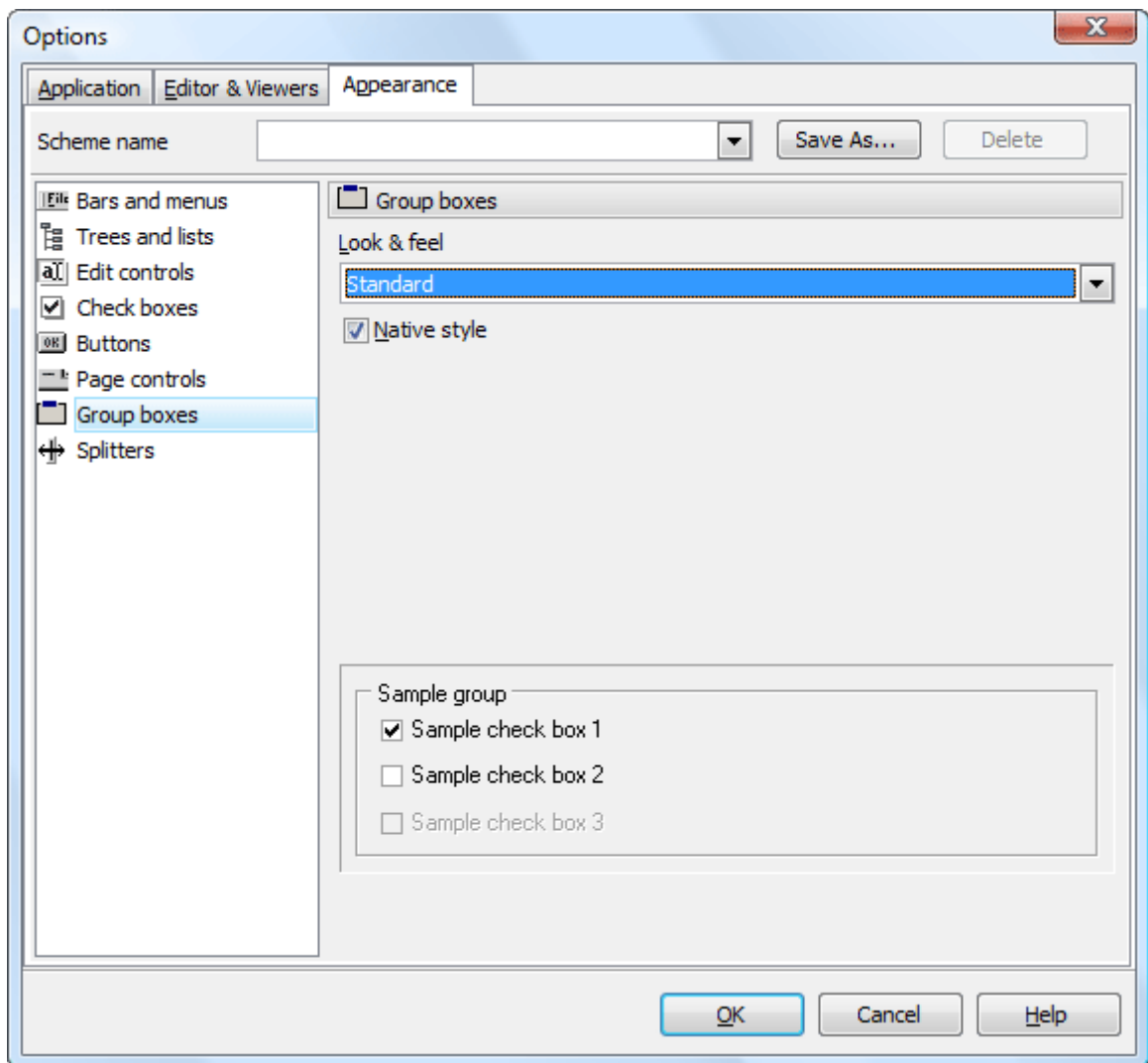
12.3.6 Page controls

The [Page controls](#) item allows you to customize the style of all PostgreSQL PHP Generator page controls. The tab allows you to select tab styles, enable/disable hot track, multi-line pages and native style.



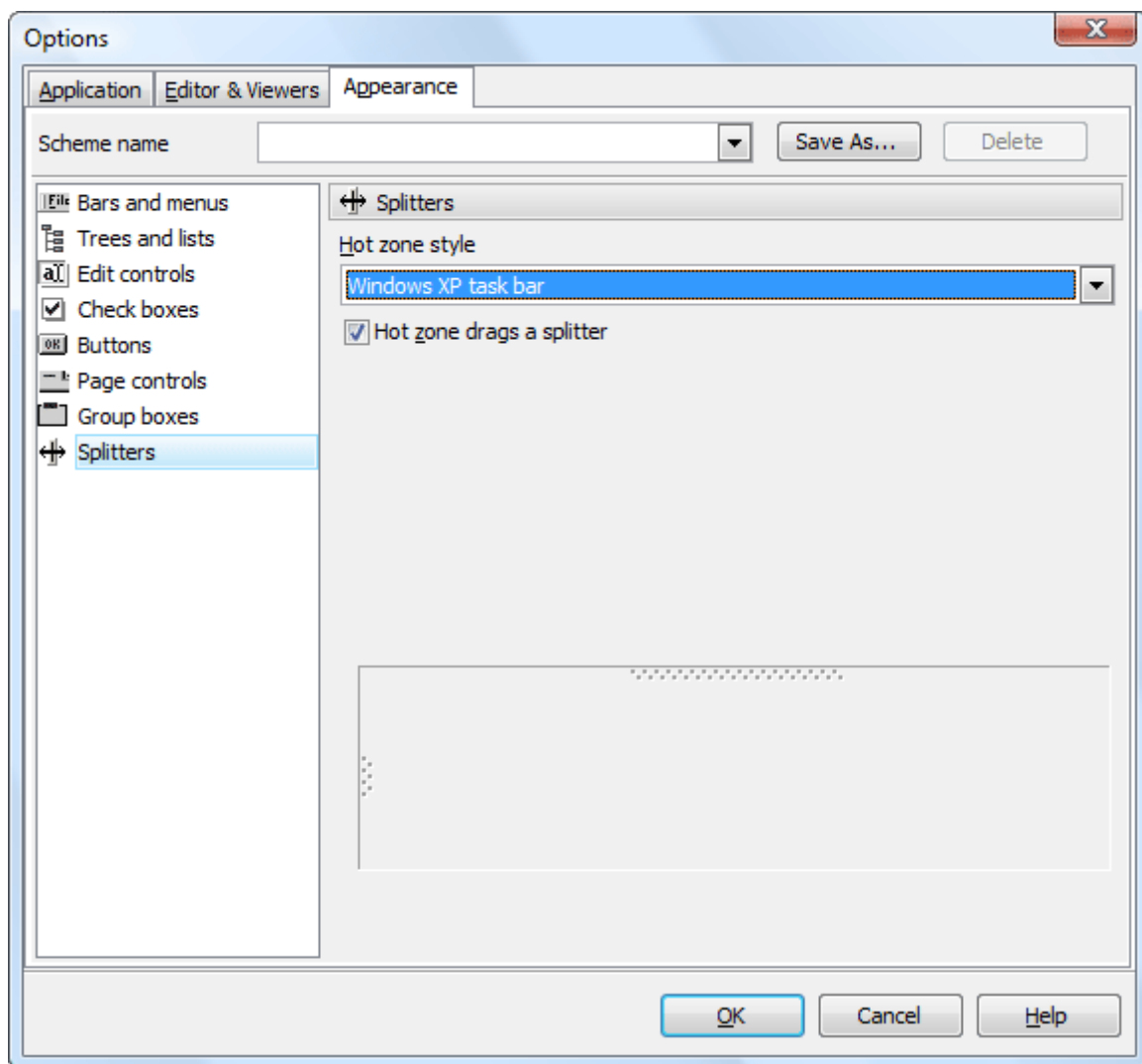
12.3.7 Group boxes

Use the [Group boxes](#) item to customize all PostgreSQL PHP Generator group boxes according to your preferences. Use tab to apply styles for group boxes, enable/disable native style and define samples.



12.3.8 Splitters

Use the [Splitters](#) item to customize all PostgreSQL PHP Generator splitters according to your preferences. Use the tab to select hot zone style (*Windows XP task bar*, *Media Player 8*, *Media Player 9*, *Simple* or *none*) and specify the [Hot zone drags a splitter](#) option.



Index

- A -

Appearance Options

- Bar and menus 348
- Buttons 352
- Check boxes 351
- Edit controls 350
- Group boxes 354
- Page controls 353
- Splitters 355
- Trees and lists 349

- C -

Charts

- General options 201
- OnPrepareChart 201
- Query 199

class Application

- GetGETValue 315
- GetPOSTValue 316
- IsGETValueSet 315
- IsLoggedInAsAdmin 316
- IsPOSTValueSet 315

Client Side API

- All editors API 278
- getEnabled 282
- getFieldName 286
- getPlaceholder 288, 289
- getReadOnly 284
- getRequired 285
- getState 285
- getValue 278, 280
- getVisible 284, 285
- setEnabled 282
- setHint 286
- setPlaceholder 288, 289
- setReadOnly 284
- setRequired 285
- setState 286
- Text editors 288

Client Side Events

- Client Side Events 146

- OnInsertFormLoaded 144
- Command line options 18
- Connect to database 13
- Connection options 13

- D -

Developer reference 277

- Client Side API 278
- Server Side API 304
- Style sheets internals 325

- E -

Edit Controls

- AutoComplete 67
- Cascading Combobox 75
- Check box 85
- Check box group 87
- Color Edit 94
- Combobox 70
- DateTime 90
- Dynamic Cascading Combobox 79
- Dynamic Combobox 73
- File Upload 100
- Html Wysiwyg 97
- Image upload 101
- Mask Edit 95
- Multiple select 89
- Password 100
- Radio group 69
- Range Edit 93
- Signature 106
- Spin Edit 92
- Text 66
- Text Area 96
- Time 91
- Toggle 86
- Upload file to folder 103
- Upload image to folder 104

Editor & Viewer Options

- Code Folding 346
- Code Insight 345
- Display 341
- General 340
- PHP highlight 344
- SQL highlight 342

Editor & Viewer Options

XML highlight 343

Email-based features

OnAfterUserRegistration 265

OnBeforeUserRegistration 265

OnPasswordResetComplete 266

OnPasswordResetRequest 265

EULA 5

Events 117

Global Events 117

Page Events executed at Client Side 117, 137

Page Events executed at Server Side 117, 147

Security Events 261

- G -

Getting started 11

Global Events

OnAfterDeleteRecord 133

OnAfterInsertRecord 130

OnAfterUpdateRecord 131

OnBeforeDeleteRecord 129

OnBeforeInsertRecord 128

OnBeforeUpdateRecord 129

OnCustomHTMLHeader 120

OnCustomizePageList 136

OnGetCustomExportOptions 134

OnGetCustomPagePermissions 136

OnGetFieldValue 134

OnPreparePage 119

- I -

Installation instructions 3

- L -

License Agreement 5

- O -

OnGetCustomTemplate

Common Templates 167

Edit and Insert forms 170

List Page 168

Options 327

Appearance 348

Application 328

Editor & Viewers 340

Page 328

- P -

Page Editor

Calculated Columns 107

Charts 199

Columns 35

Data Partitioning 220

Edit controls 62

Events 117

Filter 198

Lookup options 37

Master- Detail Presentations 111

Page properties 203

Templates 110

View properties 44

Page Events executed at Client Side 137

OnAfterPageLoad 138

OnBeforePageLoad 138

OnEditFormEditorValueChanged 142

OnEditFormLoaded 145

OnEditFormValidate 139

OnInsertFormEditorValueChanged 140

OnInsertFormValidate 139

Page Events executed at Server Side 147

OnAfterDeleteRecord 128, 129, 130, 133, 175, 177, 178, 179, 180

OnAfterUpdateRecord 131, 176

OnBeforePageExecute 148

OnCustomDrawRow 174

OnCustomHTMLHeader 154

OnCustomRenderColumn 151

OnCustomRenderExportColumn 154

OnCustomRenderPrintColumn 154

OnCustomRenderTotals 158

OnExtendedCustomDrawRow 155

OnGetCustomExportOptions 134, 181

OnGetCustomTemplate 121, 159

OnGetFieldValue 134, 181

Page properties

Abilities 212

Common options 204

RSS options 208

PostgreSQL PHP Generator

Installation 3

PostgreSQL PHP Generator

- License agreement 5
- Overview 1
- Registration 4
- System requirements 2
- Projects 17
- Purchase PostgreSQL PHP Generator 4

- R -

- Registration 4
- Report sending 19

- S -

Security Events

- OnAfterFailedLoginAttempt 262
- OnAfterLogin 262
- OnBeforeLogout 263
- OnVerifyPasswordStrength 264

Security options

- Custom Password Encryption 253
- Database server authorization 258
- Email-based features 256
- Google reCAPTCHA 267
- Hard-coded authorization 250
- Permission manager 269
- Record-level security 268
- Table-based authorization 252
- User-defined authorization 259

Server Side API

- class Page 304

Server Side Page Events

- OnCalculateFields 194
- OnCustomCompareValues 190
- OnCustomDefaultValues 193
- OnFileUpload 190
- OnGetCustomColumnGroup 189
- OnGetCustomFormLayout 184
- OnGetCustomPagePermissions 136, 191
- OnGetCustomRecordPermissions 192
- OnGetSelectionFilters 194
- OnPageLoaded 150
- OnPrepareColumnFilter 183
- OnPrepareFilterBuilder 183

- System requirements 2

- V -

View Controls

- Barcode 60
- Checkbox 50
- DateTime 49
- Embedded Video 59
- External Audio 57
- External File 55
- External Image 56
- External Video 58
- File download 52
- Image 53
- QR Code 61
- Text 47
- Toggle 51

- W -

Webpage appearance 238

- Color Scheme 241
- Color Scheme Customization 241
- Custom CSS 243
- Custom templates 246
- Header and Footer 242
- Using templates 246

Working with PostgreSQL PHP Generator

- Adding queries 25
- Advanced generation options 275
- Configuring datasources 22
- Deployment to the Internet 21
- Localization 273
- Output folder 275
- Page Editor 33
- Project Options 226
- Security options 248
- Selecting tables and views 24
- Shared Options 235
- Webpage appearance 238