`CodeGen_MySQL_Plugin` - the MySQL Plugin code generator

`CodeGen_MySQL_Plugin` - the MySQL Plugin code generator

Table of Contents

List of Examples

Chapter 1. Introduction

1.1. What is it?

`CodeGen_MySQL_Plugin` is a tool that can automatically create the basic framework for MySQL Plugin from a rather simple XML specification file. The actual functionality is provided by the script `mysql-plugin-gen` that is installed by the `CodeGen_MySQL_Plugin` package.

The code generated by `mysql-plugin-gen` is designed to work with MySQL versions 5.1 and above without modifications (although it is usually necessary to compile it for a specific server version).

1.2. Features

`mysql-plugin-gen` tries to support as many plugin writing aspects as possible. This currently includes code generation for fulltext parser plugins and preparation of `configure` and `Makefile` related build files.

Support for pluggable storage engines is currently being worked on and user defined functions and other plugin types will be added when these plugin types are supported by the MySQL server.

1.3. Installation

`CodeGen_MySQL_Plugin` is available in PEAR, the PHP Extension and Application Repository, on `http://pear.php.net/`.

1.3.1. Online installation

Online installation using the PEAR installer is the easiest way to install `CodeGen_MySQL_Plugin`, just issue the following command:

```
pear install -o CodeGen_MySQL_Plugin
```

The PEAR installer will download and install the package itself and all packages that it depends on.

1.3.2. Installing from package files

When installing from package files downloaded from `pear.php.net` you have to resolve dependencies yourself. Currently `CodeGen_PECL` depends on two other PEAR packages: `Console_Getopt`, which is part of the PEAR base installation, `CodeGen`, the code generator base package, and `CodeGen_MySQL`, the base package for MySQL specific code generators. You need to download all three packages for installation. The actual installation is once again performed by the PEAR installer:

```
pear install CogeGen-0.9.0.tgz
pear install CogeGen_MySQL-0.9.0.tgz
pear install CogeGen_MySQL-Plugin-0.9.0.tgz
```

1.3.3. Installing from PEAR CVS

You can also install CodeGen_MySQL_Plugin snapshots from PEAR CVS. CVS snapshots may include features not yet available in any release package, but the code in CVS may not be as well tested as the release packages (or even broken at times). Be warned, your milage may vary. Use the following sequence of commands in your PEAR CVS checkout to install the latest `CodeGen_MySQL_Plugin` snapshot:

```
cd pear
cd CodeGen
cvs update
pear install -f package.xml
cd ..
cd CodeGen_MySQL
cvs update
pear install -f package.xml
cd ..
cd CodeGen_MySQL_Plugin
cvs update
pear install -f package.xml
cd ..
```

1.4. How to use it

Given that you already have written your XML specs file invoking **mysql-plugin-gen** is as simple as:

```
mysql-plugin-gen specfile.xml
```

mysql-plugin-gen will parse the specs file, create a new subdirectory and puts all generated files in there. The generated code is ready to be compiled using the usual

```
configure [--with-mysql=...]; make
```

sequence.

Below you find a hardcopy of `udf-gen --help` output:

```
mysql-plugin-gen 0.1.0dev, Copyright (c) 2006, 2007 Hartmut Holzgraefe

Usage:

/usr/local/bin/mysql-plugin-gen [-hxfl] [-d dir] [--version] specfile.xml

  -h|--help          this message
  -x|--experimental  enable experimental features
  -d|--dir           output directory
  -f|--force         overwrite existing files/directories
  -l|--lint          check syntax only, don't create output
  --version          show version info
```

Chapter 2. The XML description

2.1. Basics

The top level container tag describing a plugin library is the `<plugin>` tag. The name of the plugin is given in the `name=...` attribute. The plugin library name has to be a valid file name as it is used the extensions directory name.

You can specify which CodeGen_MySQL_Plugin version your specification file was build for using the `version=...` attribute. The **mysql-plugin-gen** command will not accept specifications written for a newer version of CodeGen_MySQL_Plugin than the one installed. If the requested version is older then the current one then **mysql-plugin-gen** will try to fall back to the older versions behavior for features that have changed in incompatible ways.

> **Note:** So far no such changes have happened.

The tags `<summary>` and `<description>` should be added at the very top of your plugins. The summary should be a short one-line description of the plugin library while the actually description can be as detailed as you like.

Example 2-1. Plugin basics

```
<plugin name="sample" version="0.1.0">
 <summary>A sample plugin library</summary>
 <description>
  This is a sample plugin specification
  showing how to use CodeGen_MySQL_Plugin for
  plugin library generation.
 </description>
 ...
```

2.2. Release information

The release information for your plugin library should include the plugin authors and maintainers, the version number, state and release date, the chosen license and maybe a change log describing previous releases.

The `<maintainers>`, `<release>` and `<changelog>` tags specifications are identical to those in the PEAR `package.xml` specification so please refer to the PEAR documentation here.

Example 2-2. Release information

```
...
  <maintainers>
    <maintainer>
      <user>hholzgra</user>
      <name>Hartmut Holzgraefe</name>
      <email>hartmut@php.net</email>
      <role>lead</role>
    </maintainer>
  </maintainers>

  <release>
    <version>1.0</version>
    <date>2002-07-09</date>
    <state>stable</state>
    <notes>
     The sample plugin library is now stable
    </notes>
  </release>

  <changelog>
    <release>
      <version>0.5</version>
      <date>2002-07-05</date>
      <state>beta</state>
      <notes>First beta version</notes>
    <release>
    <release>
      <version>0.1</version>
      <date>2002-07-01</date>
      <state>alpha</state>
      <notes>First alpha version</notes>
    <release>
  </changelog>
...
```

The `<license>` tag is a little more restrictive as its `package.xml` counterpart as it is used to decide which license text should actually be written to the `LICENSE`. For now you have to specify either `GPL`, `LGPL` or `BSD`, any other value is taken as '*proprietary*'.

Example 2-3. License

```
...
  <license>GPL</license>
...
```