TRT GFX 3

user manual 0.1

Claudio Desideri

1 Introduction

This library is written not for fun nor for any other purpose other than quick development. So, also the user manual tries to follow the main philosophy behind this framework.

One of the most important tips you should never forget while coding with this lib is *be lazy*. You shouldn't find yourself doing so much work for doing a small (or) repetitive job. Maybe you're doing some mistakes somewhere. You should looking over standard classes and maybe consider deriving some of them.

One other thing you should keep in mind is: this lib is built with many *conventions* in mind. If you keep them in mind, you should have no problems in fast coding. In the other case, you'll better change framework.

For now, that's all folks! Let's so exploring the standard website structure of a GFX installation.

2 Structure overview

Here we go. If you unzip your downloaded gfx source, you'll find a very top "index.php" file and some subfolders. You'll preferably place your sources here (and down here if needed). The structure of your root folder will be:

/ root folder of your website. Here goes your source,

/gfx3 contains all the files for this lib,

/gfx3/src contains the actual code,

/gfx3/cache contains various cache files,
(erase regularly in case of database changes),

/gfx3/docs contains this manual,

/template contains all your template files for this web site
(see Templates chapter),

/utils some utility tools you can use to develop faster.

So, if you're ready, let's build a simple Hello World! Here is a small but complete list of all core modules:

EMain used as a total root manager of the application

EDatabase used to handle every db request directly

EData a wrapper around db tables that aims to autogenerate queries

ELog a module that take care about errors, warnings

EUser a module that take care of user management

3 Hello World

In this example we will use every part of the library in its very basic use. Let's starting creating our first file: `index.php` under the root of our website.

```
<?php
include('gfx3/lib.php'); //including gfx3 library
$main = new Emain(); //creating $main singleton
?>
```

File: `/index.php`

Notice that the basic file using GFX3 should *always* start with a declaration of `$main`. I'm not speaking generically: `$main` *must* be declared as an `EMain` object. This is needed by the engine in order to optimize some actions like many database queries on the same db object. The next step is understanding how `EStructure` works.

As its first step `EStructure` takes a template stored under the `/template` directory, in html format and exposes it's structure to our modifications. Let's define an example template:

```
<html>
  <head>
    <title><!--^--title--^--></title>
  </head>
  <body>
    Welcome to our website! Today is a shiny day!
    <!--^--content--^-->
    </hr>
    <!--^--footer--^-->
  </body>
</html>
```

File: `/template/home.html`

This is the starting point of your website: a working template. Now we can modify this one in order to build our webpage. We can observe that inside other html elements are present strange comments like:

Those are called *zones*. Every zone is a... zone in which you can put whatever you want. There are mainly 2 methods to load content: inline and module.

```
<!--^--title--^-->
<!--^--content--^-->
<!--^--footer--^-->
```

Example zones

With modules: If in the same folder the template resides in there is a file with the same name as the zone and with a php extension, then the module will automatically be loaded into the template.

So let's define 2 more files:

```
<?php
echo "Trololo Inc.";
?>
```

File: `/template/title.php`

```
<?php
echo "Today is the ".date("d").".";
?>
```

File: `/template/footer.php`

And modify index.php like this, also adding some dynamic inline code insertion:

```
<?php
include('gfx3/lib.php');
$main = EMain();
$temp = EStructure(`home');
/* this will load home.html and automatically
   load title.php and footer.php */

$temp->code();
echo 'Gfx is the new way of eating!';
$temp->insert('content');
/* inline code insertion into a zone */
?>
```

File: `/template/index.php`

This will generate the following webpage:

So, now that you have basically understood how the template engine works, it's time to see how other core functionality works.

4 Users

Of course GFX3 provides a handy way to manage users. The user module is called EUser and most of the time it's sufficient to instantiate it like this, in

```
<html>
  <head>
    <title>Trololo Inc.</title>
  </head>
  <body>
    Welcome to our website! Today is a shiny day!
    Gfx is the new way of eating!
    <!--^--content--^-->
    </hr>
    Today is the 21.
  </body>
</html>
```

Result

pages we want to have a full user management:

```
$user = new EUser();
```

This will automatically checks if there are cookies, if those are valid data, and in a positive case, it will perform login, checks the user group and prepare the `$user` object with all the data we need to use.

```
$user->login(string $nick, string $pass);
```

This performs a correct login using `$nick` and `$pass` as nickname and password.

```
$user->logout();
```

Execute a complete logout.

```
$user->gdeny(string $group);
```

Deny access to the webpage to a user belonging to a `$group`.

```
$user->gallow(string $group);
```

Allow access to the webpage to a user belonging to a `$group`.

```
$user->belongs_to_group(string $group);
```

Returns a true if the user belongs to `$group`. False otherwise.

```
$user->register(string $nick, string $pass, string $group);
```

Tries to register a new user.

Various getters are also available.

In order to work, `EUser` needs a database table with this structure:

```
CREATE TABLE ‘utenti‘ (
  ‘id‘ int(7) unsigned NOT NULL AUTO_INCREMENT,
  ‘nick‘ text NOT NULL,
  ‘pass‘ varchar(255) NOT NULL,
  ‘tgroup‘ text NOT NULL,
  ‘mail‘ text NOT NULL,
  PRIMARY KEY (‘id‘)
) ENGINE=InnoDB;
```

Table structure