
Subject: MVC alternative for simple projects.
Posted by [madpoet](#) on Wed, 30 Jul 2008 00:07:16 GMT
[View Forum Message](#) <> [Reply to Message](#)

MVC (Model-View-Controller) is a common pattern for software applications. In simple projects I prefer a different organization which have the MVC characteristics.

What you'll see in this article are:

- Alternative PHP coding style to use within HTML.
- Using both GET and POST methods together within the same HTML form
- HTTP redirection method to avoid reposting data when the page is refreshed after insert/update/delete.
- Flash messages using sessions for HTTP redirection method.
- An example of call_user_func()

Let's take a simple CRUD operation as an example. (CRUD: Create - Read - Update - Delete)
Think about a simple e-mail addressbook. We have the list of friends. Friends can be edited, deleted, or a new friend can be added.

The Files

lib/common.php:

All common tasks like authentication, db connection, session, configuration etc. are defined in this file.

lib/functions.php:

Global functions in here. Required in common.php.

index.php:

This is the page where people are listed. This file is organized as a template.

form.php:

The form for the update and insert action. This file is organized as a template.

header.php and footer.php:

Obvious.

post.php:

insert, update, delete actions are in this file.

This file is only accessed with POST. (In this example deletion is sent via GET method for

simplicity)

After every action, the page is redirected to index.php by HTTP header. So if the user refreshes the page after inserting or updating or deleting, they would never see the annoying "do you want to resend the data" confirmation message.

File Contents

header.php and footer.php are obvious I think, so I skip them.

index.php:

```
<?php
require("lib/common.php");

/**
 * All bussiness logic is at the beginning of the file.
 * Think it as the Model, but not in a seperate file.
 */
$users = $db->fetchAll('select * from people');

/**
 * The rest is formatted as a template. Think it as the View...
 * For easier reading alternative formatting of php is used so that indentation
 * of HTML and php code are not messed up.
 * No extra PHP codes below, only loops, if/else statements and formatting functions
 * are allowed as in common template systems.
 */
?>
<? include('header.php');?>

<? if($_SESSION['flashMessage']):?>

    <div id="flash">
        <?=$_SESSION['flashMessage'];?>
        <? unset($_SESSION['flashMessage']);?>
    </div>

<? endif;?>
```

```

<? if($users):?>
  <ul>
  <? foreach($users as $user):?>
    <li>
      <?=$user['name'];?>
      <span class="action">
        <a href="form.php?id=<?=$user['id'];?>">Update</a> |
        <a href="post.php?cmd=delete&id=<?=$user['id'];?>" onclick="return confirm('Are you
sure?');">Delete</a>
      </span>
    </li>
  <? endforeach;?>
  </ul>
<? endif;?>

<p><a href="form.php">New User</a></p>

<? include('footer.php');?>

```

form.php

```

<?
require("lib/common.php");

/**
 * If id is sent then update the data. Else it's a new user
 */
if(is_numeric($_GET['id'])) {
  $user = $db->fetchRow('select * from people where id = '.$_GET['id']);
  $cmd = 'update';
} else {
  unset($_GET['id']); // Avoid XSS
  $cmd = 'insert';
}

// Here comes the View!

```

?>

```
<? include('header.php');?>
```

<!-- We may use GET and POST methods together. Only POST necessary values for update.

Send other values via GET -->

```
<form action="post.php?cmd=<?=$cmd;?>" method="post">
```

```
<input type="hidden" name="id" value="<?=$_GET['id'];?>">
```

```
<input type="text" name="name" value="<?=$user['name'];?>">
```

```
<input type="submit">
```

```
</form>
```

```
<? include('footer.php');?>
```

post.php

```
<?
```

```
require("lib/common.php");
```

```
/**
```

```
 * Smt. like a controller
```

```
 * You may easily extend this file by simply adding a 'case' and
```

```
 * the function with the same name.
```

```
*/
```

```
switch($_GET['cmd']) {
```

```
  case 'insert':
```

```
  case 'update':
```

```
  case 'delete':
```

```
    call_user_func($_GET['cmd']);
```

```
    break;
```

```
}
```

```
/**
```

```
 * Start actions
```

```
*/
```

```
function insert() {
```

```
  // Do the insert
```

```

...

redirect_to('index.php', 'User added');
}

function update() {
    // Do the update
    ...

    redirect_to('index.php', 'User edited');
}

function delete() {
    // Do the delete
    ...

    redirect_to('index.php', 'User deleted');
}

```

functions.php

```

/**
 * This function makes an HTTP redirect. The message is stored in session
 * so that it can be shown in the redirected page.
 * After it's shown, the message is cleared. (See: index.php)
 */
function redirect_to($url, $msg) {
    $_SESSION['flashMessage'] = $msg;

    header("Location: $url");
    exit;
}

```

One step forward of this structure is adding an external template system like smarty or savant. In that case index.php and form.php would act like model, template files will be the views.

And if you want to get closer to MVC even further, try Zend Framework!

Enjoy...

Subject: Re: MVC alternative for simple projects.
Posted by [coche](#) on Thu, 31 Jul 2008 18:31:14 GMT
[View Forum Message](#) <> [Reply to Message](#)

madpoet, I think is a good approach of yours. Thanks

I prefer Rasmus MVC approach rather than Zend's Framework.
