Zend Framework 2 module preDispatch and redirect

Posted on JANUARY 20, 2013 *Written by* MARSHALL 3 COMMENTS

This tutorial shows you how to run some scripting before a controller action method is processed and also how to redirect the user from there if needed.

This is ideal when you need to authenticate a user before processing any database CRUD – **Synopsis**: Check for a value in the session. If found, continue to controller action. Otherwise, redirect user to the log in form.

Lets Begin

Lets say you have a module called "Admin" – and everything inside "Admin" is off limits to the public.

Before a controller action is dispatched, an event is triggered by ZF2's MVC framework. We can attach a handler method to this event like this:

/module/Admin/Module.php

```
<?php
namespace Admin;

use Zend\Mvc\MvcEvent;

class Module
{
    public function onBootstrap(MvcEvent $e) {
        $eventManager = $e->getApplication()->getEventManager();
        $eventManager->attach(MvcEvent::EVENT_DISPATCH, array($this, 'authPreDispatch'),1);
    }

    /**
    * Authenticate user or redirect to log in
    */
    public function authPreDispatch($event) {
        //... Authentication logic ...
    }
}
```

The "attach()" method attaches a method to a given event handle. In this case the event handle is:

```
MvcEvent::EVENT_DISPATCH - which is equivalent to (string) "dispatch".
```

The method we are attaching is given as an array:

```
[0] => (Object) containing method, [1] => (string) 'methodNameOfObject'
```

The 3rd parameter of "attach" is the priority in which this method will be handled in the scenario where there are more methods attached to this same event. ("1" being called immediately).

Redirecting from Module.php

So lets say the authentication failed and the user needs to be redirected to the log in page. This is how we do it from the "authPreDispatch" method we just made:

```
public function authPreDispatch($event) {
    ... authentication didn't work {
        //- assemble your own URL - this is just an example
        $url = $event->getRouter()->assemble(array('action' => 'login'), array('name' => 'fronte

        $response = $event->getResponse();
        $response->getHeaders()->addHeaderLine('Location', $url);
        $response->setStatusCode(302);
        $response->sendHeaders();
        exit;
    }
}
```

You are simply adding a "location" line to the headers with the assembled URL and sending them NOW, killing the application before any processing has taken place.