

eZ Components MVC Tools

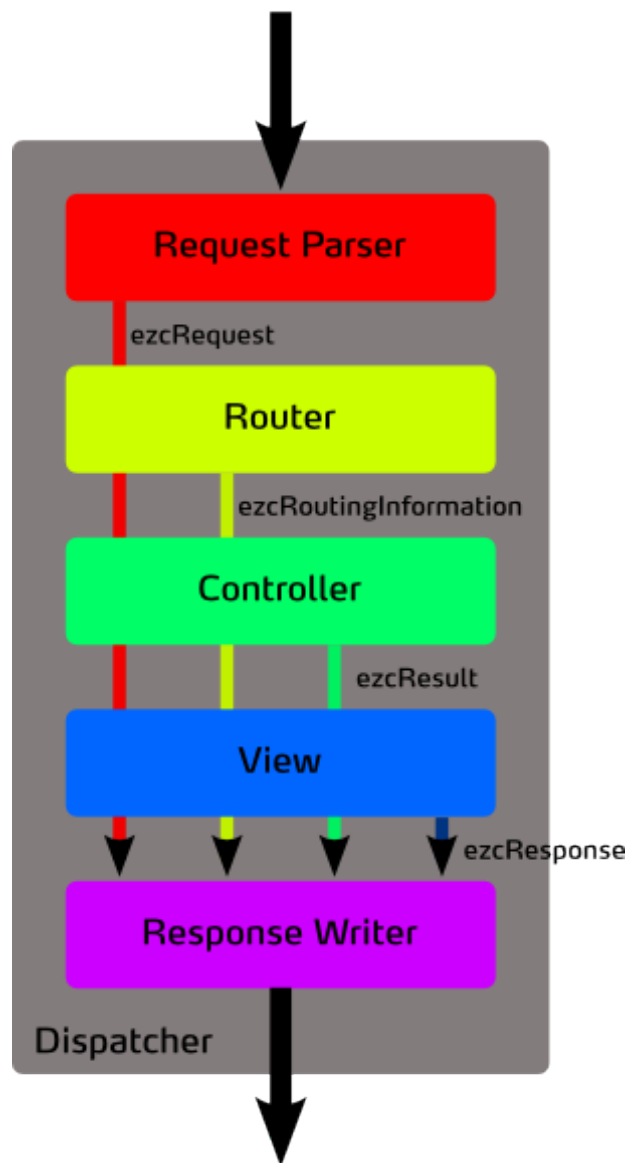
PHP Québec 2009 - Montréal, Canada

Derick Rethans - dr@ez.no

<http://derickrethans.nl/talks.php>

- Everything is decoupled
- All your own code can be reused if developed according to guidelines
- It's easy to override and overload functionality
- The MvcTools package does not dictate any kind of structure
- Full flexibility

MvcTools Overview



- Is responsible for the processing of the request
- Implements the `ezcMvcDispatcher` interface
- The current (and only) dispatcher implementation right now is configured through a configuration object
- Configuration objects implement the `ezcMvcDispatcherConfiguration` interface

In the next release:

- An dispatcher using configuration files will be added

ezcMvcConfigurableDispatcher

The ezcMvcDispatcherConfiguration interface

```
createRequestParser();

createRouter(
    ezcMvcRequest $request
);

createView(
    ezcMvcRoutingInformation $routeInfo,
    ezcMvcRequest $request,
    ezcMvcResult $result
);

createResponseWriter(
    ezcMvcRoutingInformation $routeInfo,
    ezcMvcRequest $request,
    ezcMvcResult $result,
    ezcMvcResponse $response
);

createFatalRedirectRequest(
    ezcMvcRequest $request,
    ezcMvcResult $result,
    Exception $e
);
```

- Converts raw request data from a source to an abstracted `ezcMvcRequest` object.
- `ezcMvcHttpRequestParser` and `ezcMvcMailRequestParser`.

```
class ezcMvcRequest {
    public $date; // DateTime
    public $protocol; // string
    public $host; // string
    public $uri; // string
    public $requestId; // string
    public $referrer; // string
    public $variables; // array
    public $body; // string
    public $files; // array(ezcMvcRequestFile)
    public $accept; // ezcMvcRequestAccept
    public $agent; // ezcMvcRequestUserAgent
    public $authentication; // ezcMvcRequestAuthentication
    public $raw; // ezcMvcRawRequest
    public $cookies; // array(ezcMvcRequestCookie)
}
```

Request:

```
http://kossu/test/test.php
GET /test/test.php HTTP/1.1
Host: kossu
User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.0.6) Gecko/2009020407 Icedweasel/3.0.6
(Debian-3.0.6-1) FirePHP/0.2.4
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en,en-us;q=0.8,nb;q=0.5,nl;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: UTF-8,*
Keep-Alive: 3000
Connection: keep-alive
Cache-Control: max-age=0
```

Generates:

```
object (ezcMvcRequest)#2 (14) {
  ["date"]=>
  object (DateTime)#3 (3) {
    ["date"]=>
    string(19) "2009-03-03 08:55:50"
    ["timezone_type"]=>
    int(1)
    ["timezone"]=>
    string(6) "+00:00"
  }
  ["protocol"]=>
  string(8) "http-get"
  ["host"]=>
  string(5) "kossu"
  ["uri"]=>
  string(14) "/test/test.php"
  ["requestId"]=>
  string(19) "kossu/test/test.php"
  ["referrer"]=>
  NULL
  ["variables"]=>
  &array(0) {
  }
  ["body"]=>
```

Request:

```
Return-path: <SRS0=DbNu=XN=example.no=ts@example.net>
Message-ID: <4869F84F.2060803@example.no>
Date: Tue, 01 Jul 2008 11:26:39 +0200
From: Tobias Schlitt <ts@example.no>
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.5) Gecko/20041124 Thunderbird/0.9
Mnenhy/0.6.0.104
MIME-Version: 1.0
To: Derick Rethans <dr@example.no>
Cc: Components <components@lists.example.no>
Subject: Re: [Components] [Sdk-public] MmvTools requirements and design
References: <alpine.DEB.0.98.0806251707220.16594@kossu.example.no> <48678AF4.7010908@example.no>
<alpine.DEB.0.98.0806300846100.16594@kossu.example.no> <4868A272.2080800@example.no>
<alpine.DEB.0.98.0806301153050.16594@kossu.example.no> <4868DFE3.5090001@example.no>
<alpine.DEB.0.98.0807011003550.16594@kossu.example.no>
In-Reply-To: <alpine.DEB.0.98.0807011003550.16594@kossu.example.no>
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit
```

Ze body

Generates:

```
ezMvcRequest::__set_state(array(
  'date' =>
DateTime::__set_state(array(
  'date' => '2008-07-01 09:26:39',
  'timezone_type' => 1,
  'timezone' => '+00:00',
)),
  'protocol' => 'mail',
  'host' => 'example.no',
  'uri' => 'dr',
  'requestId' => 'example.no/dr',
  'referrer' => 'alpine.DEB.0.98.0807011003550.16594@kossu.example.no',
  'variables' =>
array (
  'fromAddress' => 'ts@example.no',
  'fromName' => 'Tobias Schlitt',
  'subject' => 'Re: [Components] [Sdk-public] MmvTools requirements and design',
```

- Is configured by inheriting and reimplementing the `createRoutes()` method.
- The `createRoutes()` method returns an array of routes.
- Routes are objects of classes that implement the `ezcMvcRoute` interface: `ezcMvcRailsRoute` and `ezcMvcRegexRoute`.
- Each route tests itself if it matches the request data, and returns an `ezcMvcRoutingInformation` object if it matched.

```
<?php
return array(
    new ezcMvcRailsRoute( '/', 'shareHomeController', 'list' ),
    new ezcMvcRailsRoute( '/updates', 'shareHomeController', 'list' ),
    new ezcMvcRailsRoute( '/update/:id', 'shareHomeController', 'show' ),

    new ezcMvcRailsRoute( '/view/:id',
        'shareHomeController', 'show', array( 'id' => 1 ) ),

    new ezcMvcRegexRoute( '@^people(/((?P<nr>[0-9]+)|(?P<name>.+)))?$@',
        'testController', 'action', array( 'nr' => '', 'name' => '' ) );
?>
```

- Is returned by the router if a route is found in the form of an `ezcMvcRoutingInformation` object.

```
<?php
class ezcMvcRoutingInformation extends ezcBaseStruct
{
    public $matchedRoute;
    public $controllerClass;
    public $action;
}
?>
```

```
<?php
class slugRoute implements ezcMvcRoute
{
    public function __construct( $controller, $action, $paramName = 'slug' )
    {
        $this->controller = $controller;
        $this->action = $action;
        $this->paramName = $paramName;
    }

    public function matches( ezcMvcRequest $request )
    {
        $q = ezcDbInstance::get()->createSelectQuery();
        $q->select( 'id' )->from( 'alias' )
        ->where( $q->expr->eq( 'slug', $q->bindValue( $request->uri ) ) );
        ....
        $params = array( $this->paramName => $resultRow['id'] );
        $request->variables = array_merge( $request->variables, $params );
        return new ezcMvcRoutingInformation(
            $request->uri,
            $this->controller,
            $this->action
        );
    }

    public function prefix( $prefix )
    {
    }
}
?>
```

- The controller is created by the dispatcher from the routing information returned by the router.
- The controller is responsible for business logic and returns its results in an `ezcMvcResult` object.
- Users should implement inherited classes to provide the actions.
- The "default" controller uses the actions name to map to a method, but this algorithm can be easily overridden.

```
<?php
class helloController extends ezcMvcController
{
    public function doGreet()
    {
        $ret = new ezcMvcResult;
        $ret->variables['greeting'] = 'Hello World!';
        return $ret;
    }
}
?>
```

- The "default" controller uses the actions name to map to a method, but this algorithm can be easily overridden.

```
<?php
class helloController extends ezcMvcController
{
    public function action_greet()
    {
        $ret = new ezcMvcResult;
        $ret->variables['greeting'] = 'Hello World!';
        return $ret;
    }

    public function createActionMethodName( $actionName )
    {
        return 'action_' . $actionName;
    }
}
?>
```

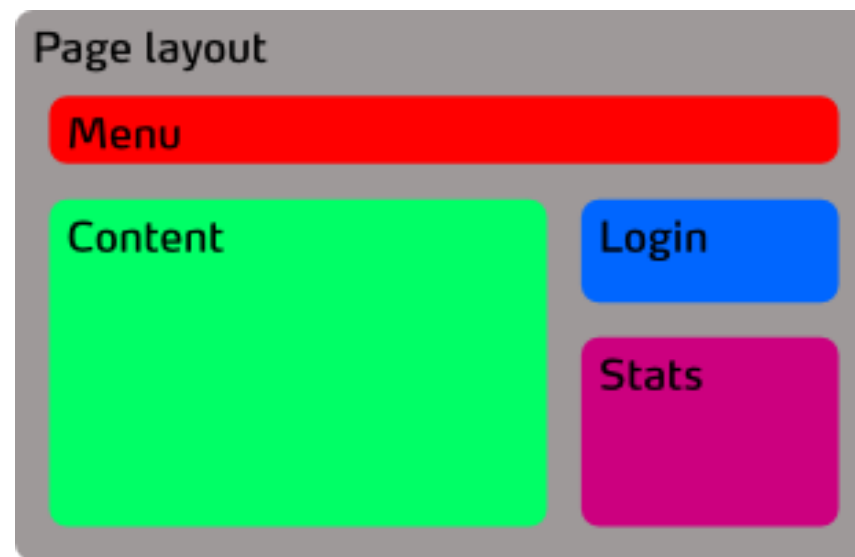
Internal redirects are replacements for ezcMvcResult objects:

```
class redirectController extends ezcMvcController
{
    public function doLogout()
    {
        $request = clone $this->request;
        $request->variables['redirUrl'] = '/new-url';
        $request->variables['reasons'] = 'something was wrong';
        $request->uri = '/login-required';
        return new ezcMvcInternalRedirect( $request );
    }
}
```

External redirects are done through the status property:

```
class redirectController extends ezcMvcController
{
    public function doLogout()
    {
        $res = new ezcMvcResult;
        $res->status = new ezcMvcExternalRedirect( '/' );
        return $res;
    }
}
```

- Render the abstract output into an `ezcMvcResponse` object.
- Are implemented by inheriting from `ezcMvcView`.
- Define zones with a view handler.



- Renders a zone with variables from the abstract output into an `ezcMvcResponse` object.
- Are implemented by inheriting from `ezcMvcViewHandler`.
- The result can be re-used in following zones.
- The last view handler's `process()` method is special.

Available view handlers:

- `ezcMvcTemplateViewHandler`: uses the template component to render result objects
- `ezcMvcPhpViewHandler`: uses PHP files to render result objects
- `ezcMvcJsonViewHandler`: renders result objects as JSON
- `ezcMvcFeedViewHandler`: uses XML feeds to render

- Takes the abstract `ezcMvcResponse` object.
- Is transport-mechanism specific.
- Matches with request parsers.
- Currently only implement for HTTP as the `ezcMvcHttpResponseWriter`.
- Uses the abstract data to set HTTP headers/cookies.
- Supports a "status" object for HTTP redirections.

Can be run on four stages from the dispatcher:

- Before the router with `runPreRoutingFilters()` to modify parsed request data to be able to select different routers.
- After the router has run, but before the controller has been selected with `runRequestFilters()` (example: password protecting parts of the site).
- After the controller/action has run, but before the view is applied with `runResultFilters()` (example: adding common variables to be used in views, such as INI settings).
- After the view has run, but before the request writer writes output with `runResponseFilters()` (example: gzipping HTTP content on the fly).

- Authentication: User registration
- Database: Storing the recipes
- Document: Displayed the wiki-formatted recipes as XHTML
- Feed: New recipes
- MvcTools: Application layout
- Template: Application templates

Setting-Up Directory Structure

```
# directories:
mkdir OmNomR
cd OmNomR
mkdir cache
mkdir cache/compiled_templates
mkdir lib
mkdir lib/autoload
mkdir lib/controllers
mkdir lib/views
mkdir templates
mkdir www
mkdir www/stylesheets

# permissions:
chgrp nogroup cache/compiled_templates
chmod g+w cache/compiled_templates
```

config.php:

```
<?php
ini_set( 'include_path', '/home/derick/dev/ezcomponents/trunk:.' );
require 'Base/src/ezc_bootstrap.php';

ezcDbInstance::set( ezcDbFactory::create( 'sqlite://' . dirname( __FILE__ ) . '/omnomr.sqlite' ) );

ezcBase::addClassRepository( dirname( __FILE__ ) . '/lib', null, 'onr' );

$tc = ezcTemplateConfiguration::getInstance();
$tc->templatePath = dirname( __FILE__ ) . '/templates';
$tc->compilePath = dirname( __FILE__ ) . '/cache';
?>
```

www/index.php:

```
<?php
include '../config.php';

$config = new onrMvcConfiguration();
$dispatcher = new ezcMvcConfigurableDispatcher( $config );
$dispatcher->run();
?>
```

www/.htaccess:

```
RewriteEngine On

RewriteCond %{REQUEST_FILENAME} -s [OR]
RewriteCond %{REQUEST_FILENAME} -l
RewriteRule ^.*$ - [NC,L]
RewriteRule ^.*$ index.php [NC,L]

ErrorDocument 404 /error/404

php_value session.gc_maxlifetime 864000
```

Step 1: The home controller and view

Dispatcher configuration

config.php:

```
<?php
class onrMvcConfiguration implements ezcMvcDispatcherConfiguration
{
    function createRequestParser()
    {
        return new ezcMvcHttpRequestParser;
    }

    function createRouter( ezcMvcRequest $request )
    {
        return new onrRouter( $request );
    }

    function runPreRoutingFilters( ezcMvcRequest $request ) { }
    function runRequestFilters( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request ) { }
    function runResultFilters( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request,
    ezcMvcResult $result ) { }
    function runResponseFilters( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request,
    ezcMvcResult $result, ezcMvcResponse $response ) { }

    function createView( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request, ezcMvcResult
$result )
    {
        switch ( $routeInfo->matchedRoute )
        {
            case '/':
                $view->contentTemplate = 'home.ezt'; break;
        }
        return $view;
    }

    function createResponseWriter( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request,
    ezcMvcResult $result, ezcMvcResponse $response )
    {
        return new ezcMvcHttpResponseWriter( $response );
    }

    function createFatalRedirectRequest( ezcMvcRequest $request, ezcMvcResult $result, Exception
```

lib/autoload/autoload.php:

```
<?php
return array(
    'onrMvcConfiguration' => 'config.php',
    'onrRouter'           => 'router.php',
    'onrHomeController' => 'controllers/home.php',
    'onrView'            => 'views/main.php',
);
?>
```

lib/router.php:

```
<?php
class onrRouter extends ezcMvcRouter
{
    public function createRoutes()
    {
        return array(
            new ezcMvcRailsRoute( '/', 'onrHomeController', 'list' ),
        );
    }
}
?>
```

lib/controllers/home.php:

```
<?php
class onrHomeController extends ezcMvcController
{
    public function doList()
    {
        $res = new ezcMvcResult();
        return $res;
    }
}
?>
```

lib/views/main.php:

```
<?php
class onrView extends ezcMvcView
{
    public $contentTemplate;

    function createZones( $layout )
    {
        $zones = array();

        $zones[] = new ezcMvcTemplateViewHandler(
            'content', $this->contentTemplate
        );

        $zones[] = new ezcMvcTemplateViewHandler(
            'page_layout', 'main.ezt'
        );

        return $zones;
    }
}
?>
```

templates/main.ezt:

```
{use $installDomain = 'dev.omnomr', $user = 'Anonymous', $content}
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>OmNomR</title>
<link type="text/css" rel="stylesheet" href="http://{$installDomain}/stylesheets/core.css"/>
</head>
<body>
<div id="everything">
<ul id="menu">
<if $user != 'Anonymous'>
<li><a href='http://{$installDomain}/'>Home</a></li>
<li><a href='http://{$installDomain}/add'>Add Recipe</a></li>
<else>
<li><a href='http://{$installDomain}/'>Login</a></li>
<li><a href='http://{$installDomain}/register'>Register</a></li>
</if>
</ul>
<br/>
<div id="main">
{raw $content}
</div>

<div style="float: right">
<div id="user">
<div class="username">{$user}</div>
<if $user != 'Anonymous'>
<div><a href='http://{$installDomain}/prefs'>Preferences</a></div>
</if>
</div>
</div>
</body>
</html>
```

templates/home.ezt:

```
<h1>OmNomR</h1>  
<p>  
Welcome!  
</p>
```

Current result:

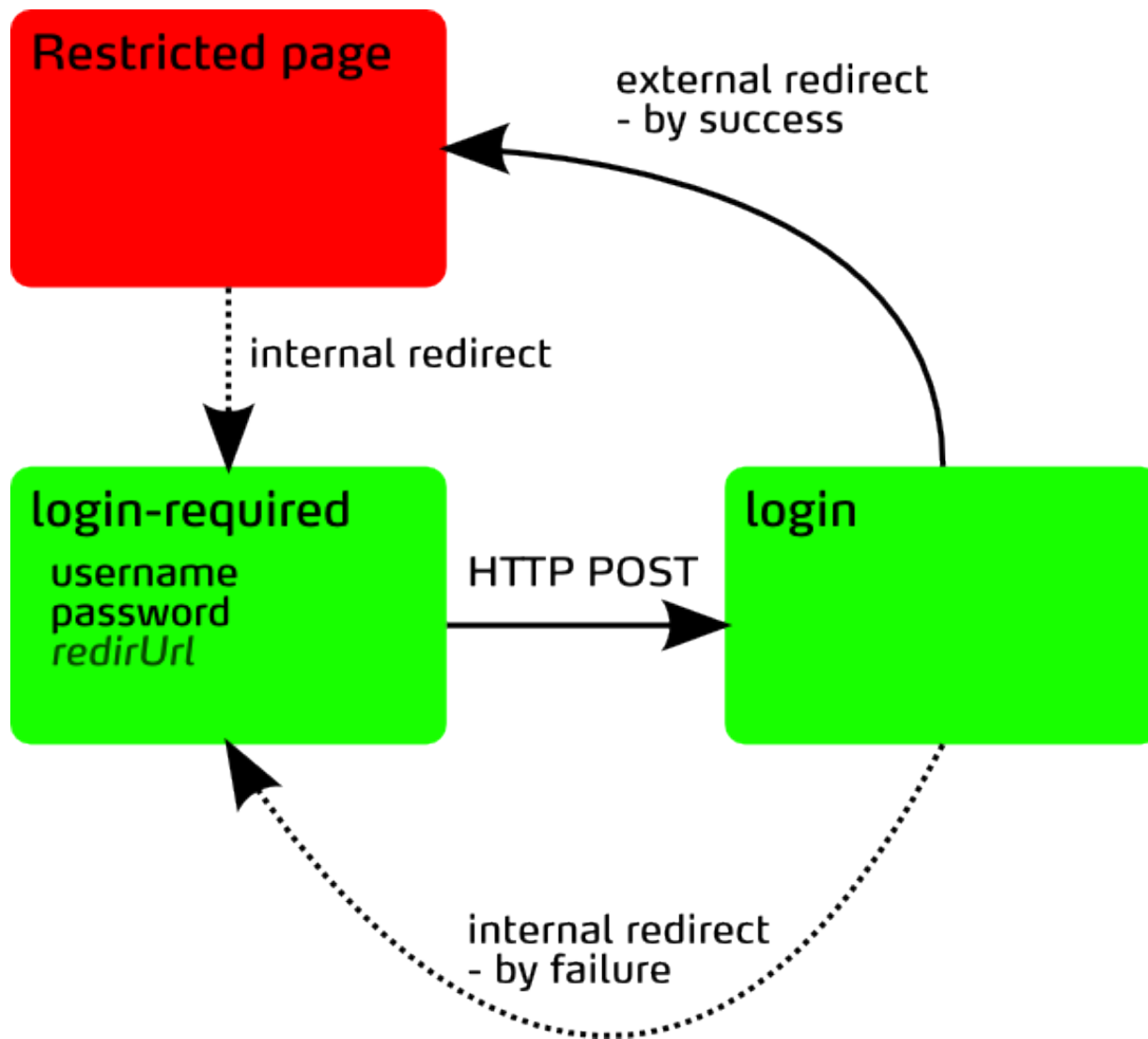
OmNomR

Welcome!

[Login](#) [Register](#)

Anonymous

Step 2: Adding authentication Diagram



- Prevent access to content with a request filter.
- Request filter runs after the router has selected a router with controller and action.
- Make sure that we do not limit access to the authentication pages and register pages.

Implement:

- The new routes to the router.
- The onrAuthController class to the autoload file.
- The filter logic in the configuration object.
- Template views to configuration object.
- Templates
- A "login required" action which a user gets redirected too.
- A "login" and "logout" action.

lib/router.php:

```
<?php
class onrRouter extends ezcMvcRouter
{
    public function createRoutes()
    {
        return array(
            // Authentication
            new ezcMvcRailsRoute(
                '/register/submit', 'onrAuthController', 'register-sub' ),
            new ezcMvcRailsRoute( '/register', 'onrAuthController', 'register' ),
            new ezcMvcRailsRoute(
                '/login-required', 'onrAuthController', 'login-required' ),
            new ezcMvcRailsRoute( '/login', 'onrAuthController', 'login' ),
            new ezcMvcRailsRoute( '/logout', 'onrAuthController', 'logout' ),
        );
    }
}
```

lib/autoload/autoload.php:

```
<?php
return array(
    'onrAuthController' => 'controllers/auth.php',
);
?>
```

Step 2: Adding authentication

Setting up the filter logic

lib/config.php:

```
<?php
function runRequestFilters( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request )
{
    switch ( $routeInfo->matchedRoute )
    {
        case '/':
            return $this->runAuthCheckLoggedIn( $request );
            break;
        case '/register/submit':
        case '/register':
        case '/login-required':
        case '/login':
        case '/logout':
        case '/fatal':
            break;
        default:
            return $this->runAuthRequiredFilter( $request );
    }
}

private function doAuthCheck( $request )
{
    $database = new ezcAuthenticationDatabaseInfo( ezcDbInstance::get(), 'user', array( 'email',
'password' ) );
    $databaseFilter = new ezcAuthenticationDatabaseFilter( $database );

    // use the options object when creating a new Session object
    $options = new ezcAuthenticationSessionOptions();
    $options->validity = 86400;

    $session = new ezcAuthenticationSession( $options );
    $session->start();

    $user = $session->load();
    $password = null;
    $loginWithForm = true;

    $credentials = new ezcAuthenticationPasswordCredentials( $user, md5( $password ) );
    $authentication = new ezcAuthentication( $credentials );
}
```

Step 2: Adding authentication

Adding view templates

lib/config.php:

```
<?php
function createView( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request, ezcMvcResult
$result )
{
    switch ( $routeInfo->matchedRoute )
    {
        case '/':
            $view->contentTemplate = 'home.ezt'; break;
        case '/register/submit':
            $view->contentTemplate = 'register-submit.ezt'; break;
        case '/register':
            $view->contentTemplate = 'register.ezt'; break;
        case '/login-required':
            $view->contentTemplate = 'login.ezt'; break;
    }
}
?>
```

templates/login.ezt:

```
{use $redirectUrl, $reasons}
<h1>Login</h1>

<div class="login">
<p>
You are not logged in, because:
</p>
<ul>
{foreach $reasons as $reason}
  <li>{$reason}</li>
{/foreach}
</ul>

<form class="login" action="/login" method="post">
<div>
Email: <input name="email" size="32"/><br/>
Password: <input type="password" name="password"/><br/>
<br/>
<input type="hidden" name="redirectUrl" value="{ $redirectUrl }"/>
<input type="submit" value="Login"/>
</div>
</form>
</div>
```

templates/register.ezt:

```
{use $mailDomain = 'dev.omnomr'}  
<h1>Register</h1>  
<div class="pres">  
<form class="prefs" action='register/submit' method='post'>  
E-mail: <input name='email' size='32' /><br/>  
Full name: <input name='fullname' size='32' /> (Like "Derick Rethans")<br/><br/>  
<br/>  
<input type='submit' name='reg' value="Register" />  
</form>  
</div>
```

templates/register-submit.ezt:

```
{use $message = '', $success = false}  
<h1>Registration Result</h1>  
<div class="pres">  
<div class='regmessage'>  
{ $message }  
</div>  
{if !$success}  
<a href='/register'>try again</a>  
{/if}  
</div>  
</div>
```

Step 2: Adding authentication

Adding actions: login-required

lib/controllers/auth.php:

```
<?php
class onrAuthController extends ezcMvcController
{
    public function doLoginRequired()
    {
        $res = new ezcMvcResult;
        $res->variables['redirUrl'] = $this->redirUrl;

        $err = array(
            'ezcAuthenticationDatabaseFilter' => array(
                ezcAuthenticationHtpasswdFilter::STATUS_USERNAME_INCORRECT => 'Incorrect or no
credentials provided.',
                ezcAuthenticationHtpasswdFilter::STATUS_PASSWORD_INCORRECT => 'Incorrect or no
credentials provided.'
            ),
            'ezcAuthenticationSession' => array(
                ezcAuthenticationSession::STATUS_EMPTY => 'No session',
                ezcAuthenticationSession::STATUS_EXPIRED => 'Session expired'
            ),
            'logout' => array(
                1 => 'You logged out',
            )
        );

        $reasonText = array();

        $reasons = $this->reasons;
        foreach ( $reasons as $line )
        {
            list( $key, $value ) = each( $line );
            $reasonText[] = $err[$key][$value];
        }
        $res->variables['reasons'] = $reasonText;
        $res->variables['redirUrl'] = $this->redirUrl;
        return $res;
    }
}
?>
```

lib/controllers/auth.php:

```
<?php
class onrAuthController extends ezcMvcController
{
    public function doLogin()
    {
        // obtain credentials from POST
        $email = isset( $_POST['email'] ) ? $_POST['email'] : null;
        $password = isset( $_POST['password'] ) ? $_POST['password'] : null;
        $redirectUrl = isset( $_POST['redirectUrl'] ) ? $_POST['redirectUrl'] : '/';

        $database = new ezcAuthenticationDatabaseInfo( ezcDbInstance::get(), 'user', array( 'email',
'password' ) );
        $databaseFilter = new ezcAuthenticationDatabaseFilter( $database );

        $options = new ezcAuthenticationSessionOptions();
        $options->validity = 86400;

        $session = new ezcAuthenticationSession( $options );
        $session->start();

        // use the options object when creating a new Session object
        $credentials = new ezcAuthenticationPasswordCredentials( $email, md5( $password ) );
        $authentication = new ezcAuthentication( $credentials );
        $authentication->session = $session;
        $authentication->addFilter( $databaseFilter );

        if ( !$authentication->run() )
        {
            $request = clone $this->request;
            $status = $authentication->getStatus();
            $request->variables['redirectUrl'] = $redirectUrl;
            $request->variables['reasons'] = $status;

            $request->uri = '/login-required';
            return new ezcMvcInternalRedirect( $request );
        }

        $res = new ezcMvcResult;
        $res->status = new ezcMvcExternalRedirect( $redirectUrl );
    }
}
```

lib/controllers/auth.php:

```
<?php
class onrAuthController extends ezcMvcController
{
    public function doRegister()
    {
        $res = new ezcMvcResult;
        return $res;
    }

    public function doRegisterSubmit()
    {
        $res = new ezcMvcResult;
        if ( isset( $this->reg ) )
        {
            $res->variables['success'] = self::register( $message );
            $res->variables['message'] = $message;
        }
        return $res;
    }

    private static function register( &$message )
    {
        $db = ezcDbInstance::get();
        $email = substr( preg_replace( '/[^a-z0-9.@]/', '', $_POST['email'] ), 0, 32 );
        $fullname= ucwords( strtolower( preg_replace( '/[^a-zæøå0-9 ]/ui', '',
        $_POST['fullname'] ) ) );

        // check if the user already exists.
        $q = $db->createSelectQuery();
        $q->select( 'email' )->from( 'user' )->where( $q->expr->eq( 'email', $q-
        >bindValue( $email ) ) );
        $s = $q->prepare();
        $s->execute();
        $r = $s->fetchAll();

        if ( count( $r ) > 0 )
        {
            $message = "A user with e-mail address '{$email}' already exists.";
            return false;
        }
    }
}
```

Step 2: Adding authentication

Viewing recipes

[Login](#) [Register](#)

Register

E-mail:

Full name: (Like "Derick Rethans")

Anonymous

- Add the new routes (`/recipe/*`) to the router.
- Add the `onrRecipeController` class to the autoload file.
- Update configuration.
- Create templates
- Add the actions.

lib/router.php:

```
<?php
class onrRouter extends ezcMvcRouter
{
public function createRoutes()
{
    return array(
        // Recipes
        new ezcMvcRailsRoute( '/recipes', 'onrRecipeController', 'list' ),
        new ezcMvcRailsRoute( '/recipe/list', 'onrRecipeController', 'list' ),
        new ezcMvcRailsRoute( '/recipe/add', 'onrRecipeController', 'add' ),
        new ezcMvcRailsRoute( '/recipe/submit',
            'onrRecipeController', 'submit' ),
        new ezcMvcRailsRoute( '/recipe/:id', 'onrRecipeController', 'view' ),
    );
}
}
```

lib/autoload/autoload.php:

```
<?php
return array(
    'onrAuthController' => 'controllers/auth.php',
    'onrRecipeController' => 'controllers/recipe.php',
);
?>
```

lib/config.php:

```
<?php
function runRequestFilters( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request )
{
    switch ( $routeInfo->matchedRoute )
    {
        case '/recipe/add':
        case '/recipe/submit':
            return $this->runAuthRequiredFilter( $request );
            break;
        case '/register/submit':
        case '/register':
        case '/login-required':
        case '/login':
        case '/logout':
        case '/fatal':
            break;
        default:
            return $this->runAuthCheckLoggedIn( $request );
    }
}
```

Step 3: Recipes

Adding view configuration

lib/config.php:

```
<?php
function createView( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request, ezcMvcResult
$result )
{
    switch ( $routeInfo->matchedRoute )
    {
        case '/':
            $view->contentTemplate = 'home.ezt'; break;

        // auth:
        case '/register/submit':
            $view->contentTemplate = 'register-submit.ezt'; break;
        case '/register':
            $view->contentTemplate = 'register.ezt'; break;
        case '/login-required':
            $view->contentTemplate = 'login.ezt'; break;

        // recipes:
        case '/recipes':
        case '/recipe/list':
            $view->contentTemplate = 'recipes/list.ezt'; break;
        case '/recipe/:id':
            $view->contentTemplate = 'recipes/view.ezt'; break;
        case '/recipe/add':
            $view->contentTemplate = 'recipes/add.ezt'; break;
    }
}
?>
```

templates/recipes/list.ezt:

```
{use $recipes}
<h1>Recipes</h1>
<div class="list">
  {foreach $recipes as $recipe}
    <div class="recipe-short">
      <h1><a href="/recipe/{$recipe['id']}">{$recipe['name']}</a></h1>
      <hr />
    </div>
  {/foreach}
</div>
```

templates/recipes/view.ezt:

```
{use $data, $ingredients}
<h1>{$data['name']}</h1>
<div class="description">
  {raw $data['description']}
</div>
<div class="ingredients">
  {foreach $ingredients as $ingredient}
    <div class="ingredient">
      {$ingredient['id']}
    </div>
  {/foreach}
</div>
```

templates/recipes/add.ezt:

```
<h1>Adding a recipe</h1>

<div class="pres">
<form class="prefs" action="/recipe/submit" method="post">
<div>
Recipe name: <input type="text" name="name" value=""/><br/>
Portions: <input type="text" size="2" maxlength="2" name="portions" value="4"/><br/>
Description:<br/>
<textarea cols="60" rows="10" name="description">
</textarea>
<br/>
<br/>
<input type="submit" name="add" value="Add"/>
</div>
</form>
</div>
```

lib/controllers/recipe.php:

```
<?php
class onrRecipeController extends ezcMvcController
{
    public function doAdd()
    {
        return new ezcMvcResult;
    }

    public function doSubmit()
    {
        $d = ezcDbInstance::get();
        $q = $d->createInsertQuery();
        $q->insertInto( 'recipe' )
        ->set( 'user_id', $q->bindValue( $this->user_id ) )
        ->set( 'portions', $q->bindValue( (int) $this->portions ) )
        ->set( 'description', $q->bindValue( $this->description ) )
        ->set( 'name', $q->bindValue( $this->name ) );
        $s = $q->prepare();
        $s->execute();
        $newId = $d->lastInsertId();
        $res = new ezcMvcResult;
        $res->status = new ezcMvcExternalRedirect( '/recipe/' . $newId );
        return $res;
    }
}
?>
```

lib/controllers/recipe.php:

```
<?php
class onrRecipeController extends ezcMvcController
{
    public function doList()
    {
        $d = ezcDbInstance::get();
        $q = $d->createSelectQuery();
        $q->select( 'id', 'name' )->from( 'recipe' )->orderBy( 'name' );
        $s = $q->prepare();
        $s->execute();

        $res = new ezcMvcResult;
        $res->variables['recipes'] = $s;
        return $res;
    }
}
?>
```

lib/controllers/recipe.php:

```
<?php
class onrRecipeController extends ezcMvcController
{
    public function doView()
    {
        $id = (int) $this->id;
        $res = new ezcMvcResult;

        $d = ezcDbInstance::get();
        $q = $d->createSelectQuery();
        $q->select( '*' )
        ->from( 'recipe' )
        ->where( $q->expr->eq( 'id', $q->bindValue( $id ) ) );
        $s = $q->prepare();
        $s->execute();

        $data = $s->fetchAll();
        if ( empty ( $data ) )
        {
            $res->status = new ezcMvcExternalRedirect( '/' );
            return $res;
        }

        // render rest recipe as HTML
        $desc = $data[0]['description'];
        $document = new ezcDocumentRst();
        $document->loadString( $desc );
        $document->options->xhtmlVisitor = 'ezcDocumentRstXhtmlBodyVisitor';
        $xhtml = $document->getAsXhtml();
        $xml = $xhtml->save();
        $data[0]['description'] = $xml;

        $res->variables['data'] = $data[0];

        return $res;
    }
}
?>
```

Butter Chicken

Portions: 4

Do the following things:

1. heat oil, once heated add butter
2. add onions on low heat until they are caramalized (not browned)
3. add the crushed tomatoes
4. lay the marinated diced chicken over the tomatoes and leave for 5 minutes - do not stir too much
5. add 0.5 cup hot water, and add salt to taste
6. simmer the chicken for about 30 minutes
7. add cream
8. add water to taste
9. leave on stove until it just boils

Derick Rethans
[Preferences](#)

When...

- documentation isn't clear enough
- you simply have questions and or suggestions

Feel free to...

- use our mailinglist
- post to the eZ components forum
- visit us on IRC

Questions anybody?

Resources:

- Download: <http://ezcomponents.org/download>
- Documentation: <http://ezcomponents.org/docs>
- Mailinglist:
<http://lists.ez.no/mailman/listinfo/components>
- IRC: #ezcomponents @ Freenode
- These Slides: <http://derickrethans.nl/talks.php>