

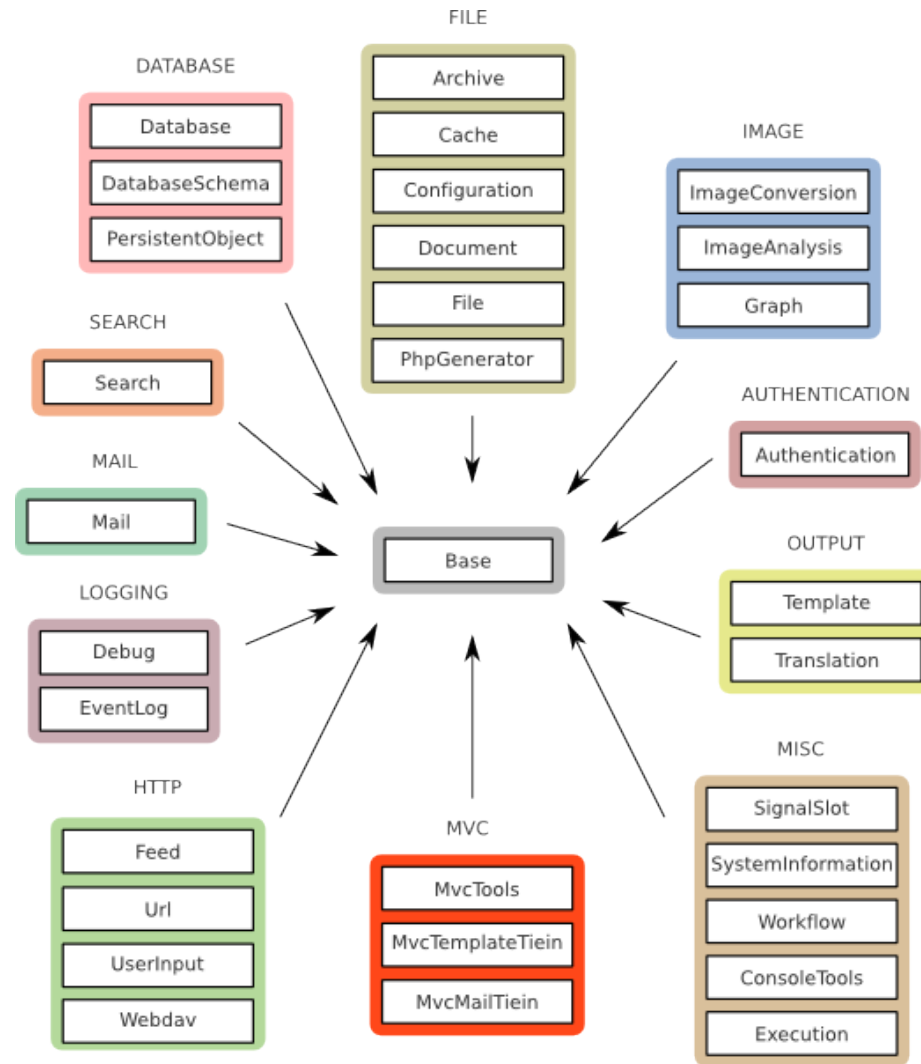
eZ Components eZ Components Perspective

eZ Conference 2009 - Paris, France

Derick Rethans - dr@ez.no

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

Overview

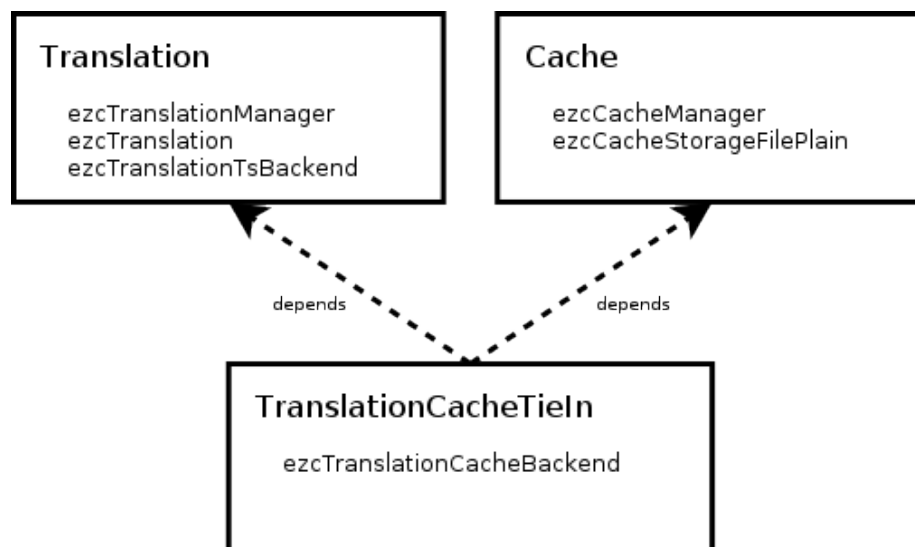


- Provide a solid platform for PHP application development
- Don't force a structure: no "framework"
- Clean and simple API
- Excellent documentation
- Keep backward compatibility for longer periods of time
- Stable and few regressions
- Clean IP, Open Source friendly

- New BSD license: Open Source, very permissive, compatible with GPL
- CLA: To accept contributions to the components we need a signed "Contributor Licensing Agreement"

- The less the better
- Only if really necessary
- Dependency-only packages

Tie-Ins:



Through the PEAR installer

- Easy to upgrade one (or more) specific component(s)
- Depends on external tool

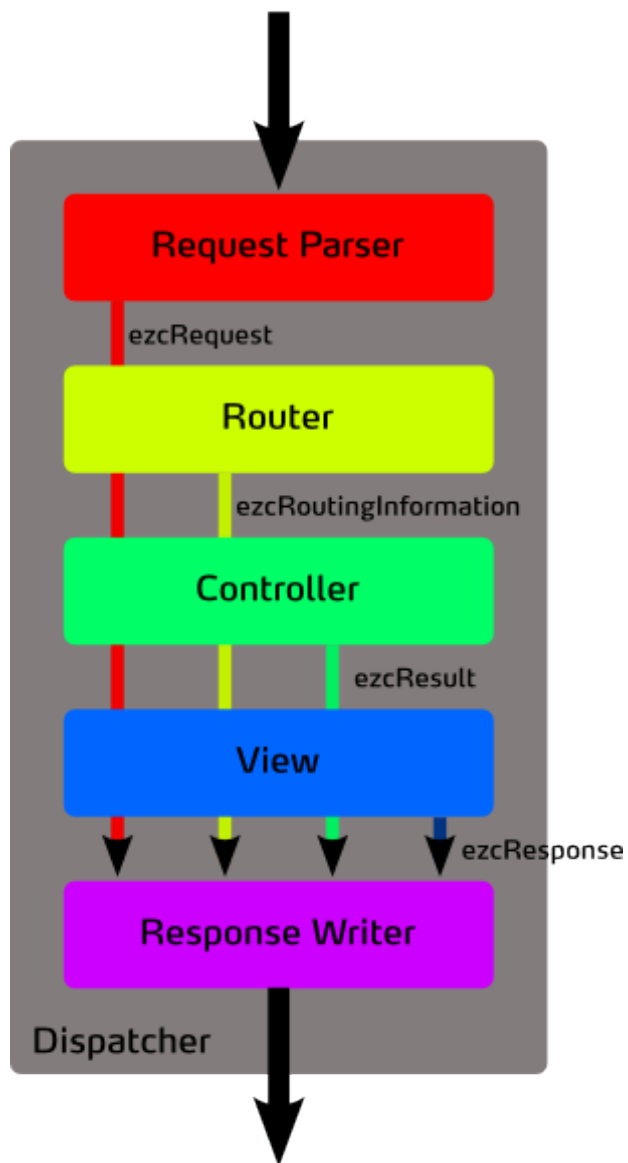
From a download bundle

- Everything comes in one package
- Hard to impossible to upgrade one specific component

Through eZ Publish download

- 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
- eZ Components is not full stack or a glue framework

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

- 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)
}
```

- The router is responsible for the selection of controller and action method.
- Each router contains an array of routes.
- Routes are objects of classes that implement the `ezcMvcRoute` interface: `ezcMvcRailsRoute`, `ezcMvcRegexRoute` and `ezcMvcCatchAllRoute`.
- 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( '/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' => '' ),
    new ezcMvcCatchAllRoute() );
?>
```

```
<?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 )
    {
    }
}
?>
```

- 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;
?>
```

- 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.

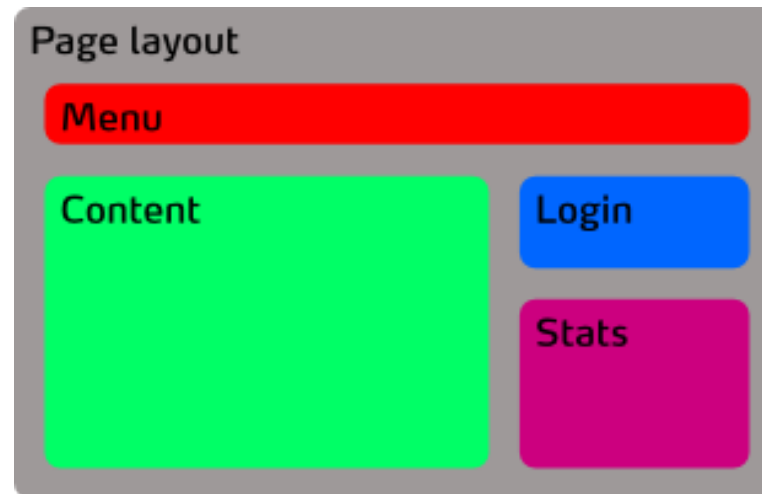
```
<?php
class helloController extends ezcMvcController
{
    public function doGreet()
    {
        $ret = new ezcMvcResult;
        $ret->variables['greeting'] = 'Hello World!';
        return $ret;
    }
}
?>
```

- 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;
    }
}
?>
```

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



```
<?php class helloNameView extends ezcMvcView
{
    function createZones( $layout )
    {
        $zones = array();
        $zones[] = new ezcMvcTemplateViewHandler( 'menu', 'menu.ezt' );
        $zones[] = new ezcMvcPhpViewHandler( 'content', 'greeting.php' );
        $zones[] = new ezcMvcTemplateViewHandler( 'page_layout', 'layout.ezt' );
        return $zones;
    }
}
```

- 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 result objects

- 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: Open ID 2
- Base: meta-data about eZ Components
- Document: PDF writing support
- EventLog: Memory writer
- MvcTools: Reversed routes, improved support for reimplementations
- MvcAuthenticationTiein: Authentication support for MvcTools
- MvcFeedTiein: RSS feeds for MvcTools
- PersistentObject: Identity management
- Template: whitespace mangling functionality
- Translation: script for extracting strings from a template

When...

- you want personal assistance
- you want extra functionality
- you require a bug fixed

We...

- provide paid support to help you using the components
- can implement the functionality you request

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>