Welcome!

eZ Components

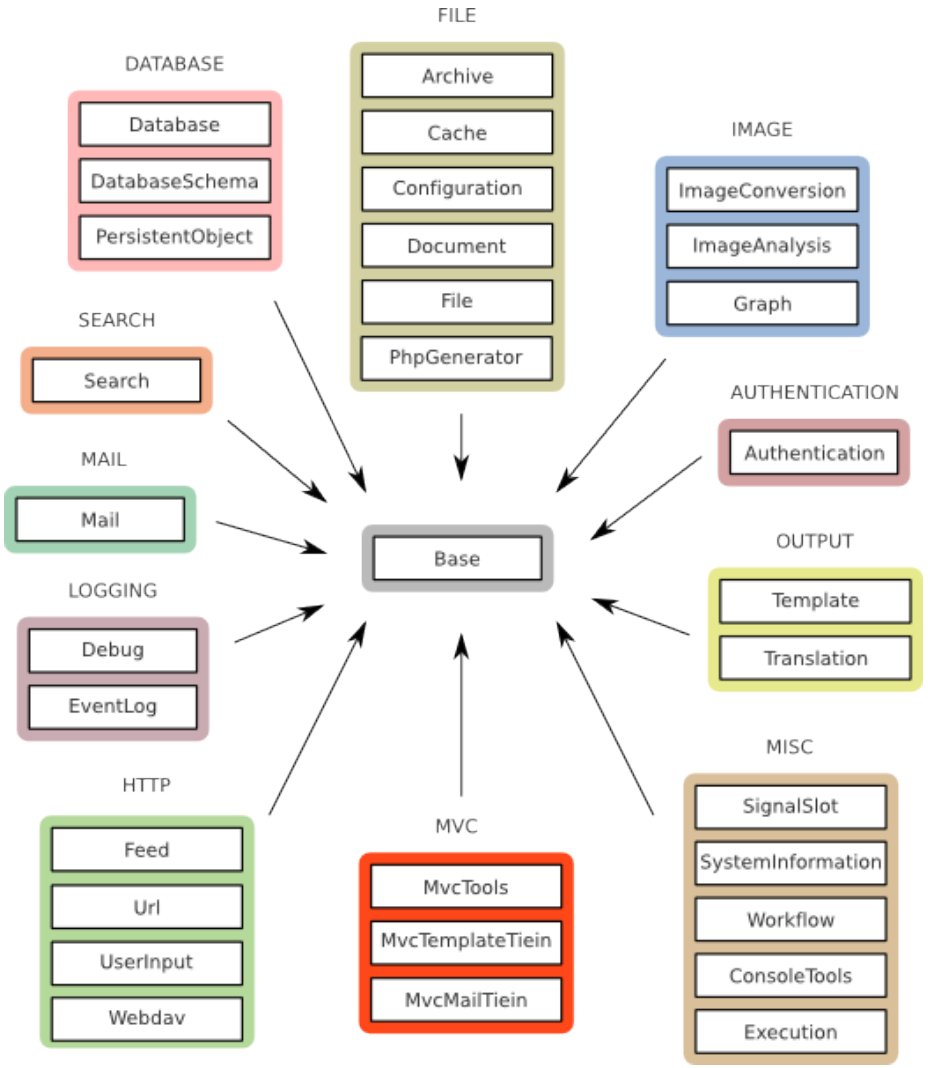
eZ Components Perspective

eZ Conference 2009 - Paris, France

Derick Rethans - dr@ez.no

http://derickrethans.nl/talks.php

Overview


FILE
DATABASE
Archive
Cache
Configuration
Document
File
PhpGenerator
Database
DatabaseSchema
PersistentObject
IMAGE
ImageConversion
ImageAnalysis
Graph
SEARCH
Search
AUTHENTICATION
Authentication
MAIL
Mail
Base
OUTPUT
Template
Translation
LOGGING
Debug
EventLog
MISC
SignalSlot
SystemInformation
Workflow
ConsoleTools
Execution
HTTP
Feed
Url
UserInput
Webdav
MVC
MvcTools
MvcTemplateTiein
MvcMailTiein

eZ Components Goals

- 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

License and Contributing

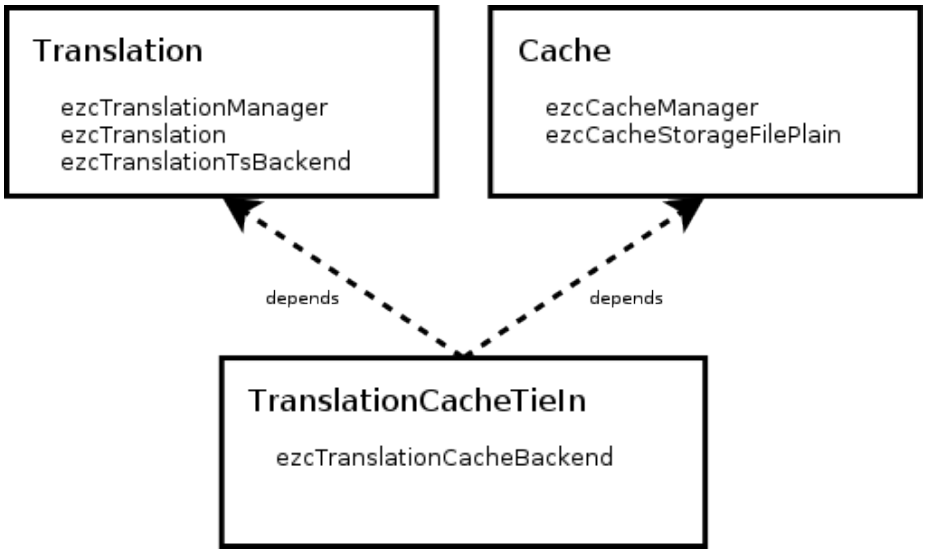
Clean IP, Open Source friendly

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

Dependencies

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

Tie-Ins:



Installation and Setup

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

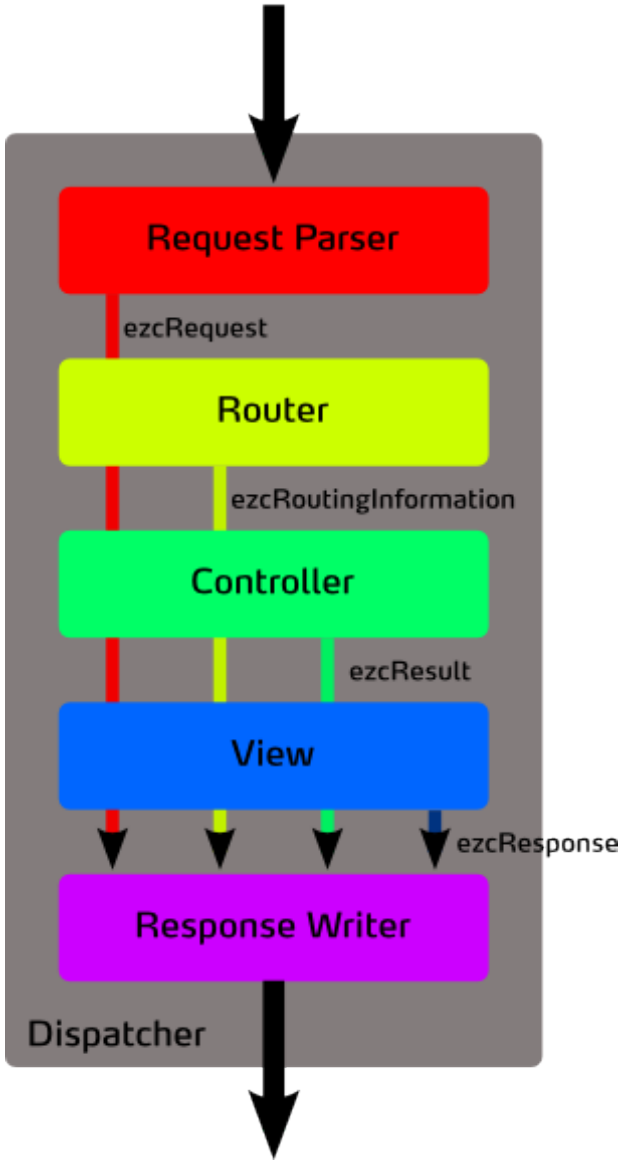
Through eZ Publish download

eZ Components' MVC Tools

Core Ideas

- 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



Dispatcher

- 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

Request Parser

- 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)
}
```

Router

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

Router

User Implementation

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

Routing Information

- 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;
?>
```

Controller

- 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 inheritted 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;
    }
?>
```

Controller

Changing action mapping algorithm

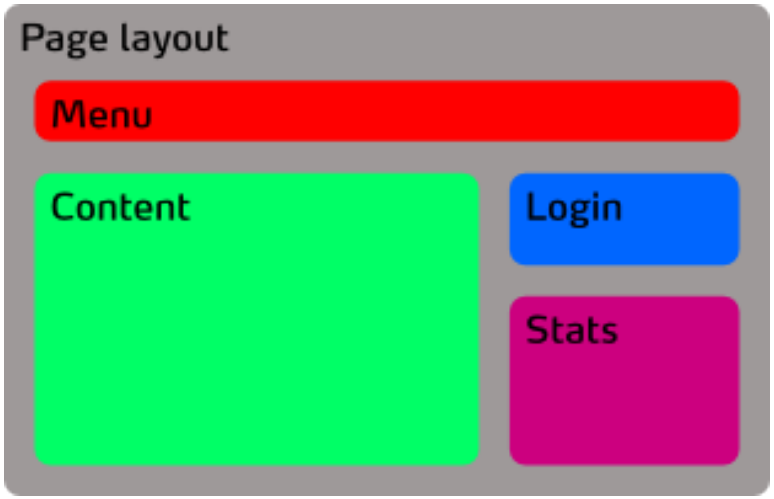
- The "default" controller uses the actions name to map to a method, but this algorithm can be easily overridden.

```
<?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;
    }
?>
```

Views

- 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;
```

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 result objects

Response Writer

- 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.

Filters

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).

New in 2009.1

- 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

Support
Commercial

When...

- you want personal assistence
- 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

Support
Community

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 and Resources

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