Designing for Reusability

Dutch PHP Conference - Amsterdam, Netherlands
Derick Rethans - derick@derickrethans.nl - twitter: @derickr
http://derickrethans.nl/talks.php
http://joind.in/1534
About Me

Derick Rethans

• Dutchman living in London
• PHP development
• Author of the mcrypt, input_filter, dbus, translit and date/time extensions
• Author of Xdebug
• Contributor to the Apache Zeta Components Incubator project (formerly eZ Components)
• Freelancer doing PHP (internals) development
Any larger application will consist of multiple components
<?php
class Wrappy
{
    private $lineLength;

    function __construct()
    {
        $configObj = new Config();
        $this->lineLength = $configObj->getLineLength();
    }
}
?>
<?php

class Wrappy
{
    private $lineLength;

    function __construct()
    {
        $configObj = Config::get();
        $this->lineLength = $configObj->getLineLength();
    }
}

The Singleton Disaster

<?php

class Wrappy
{
    private $lineLength;

    function __construct()
    {
        global $configObj;
        $this->lineLength = $configObj->getLineLength();
    }
}

The Globals Catastrophe
<?php

class Wrappy
{
    private $lineLength;

    function __construct( Config $configObj )
    {
        $this->lineLength = $configObj->getLineLength();
    }
}

$configObj = new Config();
$wrap = new Wrappy( $configObj );
?>

<?php

class Wrappy
{
    private $lineLength;

    function __construct( Config $configObj = null )
    {
        $this->lineLength = $configObj ? $configObj->getLineLength() : 78;
    }
}

$wrap = new Wrappy();
?>
Building stuff
<?php

class Wrappy
{
    private $lineLength;

    function __construct( Config $configObj )
    {
        $this->lineLength = $configObj->getLineLength();
    }
}

class TextManipulationFactory
{
    function getWrappy()
    {
        $config = new Config();
        return new Wrappy($config);
    }
}
?>
Factory
<?php

class Wrappy
{
    private $lineLength;

    function __construct( Config $configObj )
    {
        $this->lineLength = $configObj->getLineLength();
    }
}

/*
<service>
    <wrappy>
        <config>
        ...
        </config>
    </wrappy>
</service>
*/

$wrappy = ServiceDepmanager::createWrappy();
?>
"Why I Hate Frameworks"
Dependencies

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

Tie-Ins:

```
Translation
  ezcTranslationManager
  ezcTranslation
  ezcTranslationTsBackend

Cache
  ezcCacheManager
  ezcCacheStorageFilePlain

TranslationCacheTieIn
  ezcTranslationCacheBackend
```

- depends
- depends
public function createRequest()
{
    $this->request = new ezcMvcRequest();
    $this->processStandardHeaders();
    $this->processAcceptHeaders();
    $this->processUserAgentHeaders();
    $this->processFiles();
    $this->processAuthVars();
    $this->processCookies();

    $this->request->raw = &$_SERVER;

    return $this->request;
}

public function createRequest()
{
    $this->request = $this->createRequestObject();
    $this->processStandardHeaders();
    $this->processAcceptHeaders();
    $this->processUserAgentHeaders();
    $this->processFiles();
    $this->processAuthVars();
    $this->processCookies();

    $this->request->raw = &$_SERVER;

    return $this->request;
}
$text = "<ul{$idPart}>
";
foreach ( $children as $child )
{
    $path = $child[2]->nodes;
    if ( !this->options->displayRootNode )
    {
        array_shift( $path );
    }
    if ( $this->options->selectedNodeLink )
    {
        $slice = array_slice( $path, -1 );
        $path = htmlspecialchars( $this->options->basePath . '/' . array_pop( $slice ) );
    } else
    {
        $path = htmlspecialchars( $this->options->basePath . '/' . join( '/', $path ) );
    }
    $text .= str_repeat( ' ', $level + 2 );
    $data = $this->formatData( $child[1], in_array( $child[0], $this->options->highlightNodeIds ) );
    foreach ( $children as $child )
    {
        $text .= str_repeat( ' ', $level + 2 );
        $path = $this->formatPath( $child );

        $data = $this->formatData( $child[1], in_array( $child[0], $this->options->highlightNodeIds ) );
        foreach ( $children as $child )
        {
            $text .= str_repeat( ' ', $level + 2 );
            $path = $this->formatPath( $child );
Visibility

- public
- protected
- private
• Is a mechanism for mapping objects to a database
• Makes classes inherit from "ActiveRecord"
• Hibernate is alternative
• Traits

```php
<?php
trait ActiveRecord
{
    public function load()
    {
        var_dump(get_class($this));
    }
}

class Car {
    use ActiveRecord;
}
$c = new Car;
$c->load();
?>
string(3) "Car"
Refactoring

Code refactoring is the process of changing a computer program's source code without modifying its external functional behavior in order to improve some of the nonfunctional attributes of the software. Advantages include improved code readability and reduced complexity to improve the maintainability of the source code, as well as a more expressive internal architecture or object model to improve extensibility. (Wikipedia)
Refactoring

- Readability
- Maintainability
- Extensibility
- Testability
Testability

- Test driven development
- Code is written in small blocks
- Makes the API better
Un-testable Problems

- Testing singletons more than one time
- System/Operating System dependent tests
- Private methods
- Code that depends on the state of an external resource
- Things that simply should "never" happen
Mocking

Hiding an implementation by a fake implementation
MVC/Testing

- MVC generally uses HTTP
- Is that necessary?
- How do we test that?

```php
interface ezcMvcDispatcherConfiguration
{
    public function createRequestParser();
    public function createRouter( ezcMvcRequest $request );
    public function createView( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request, ezcMvcResult $result );
    public function createResponseWriter( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request, ezcMvcResult $result, ezcMvcResponse $response );
    public function createFatalRedirectRequest( ezcMvcRequest $request, ezcMvcResult $result, Exception $e );
    public function runPreRoutingFilters( ezcMvcRequest $request );
    public function runRequestFilters( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request );
    public function runResultFilters( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request, ezcMvcResult $result );
    public function runResponseFilters( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request, ezcMvcResult $result, ezcMvcResponse $response );
}
```
Derick Rethans - derick@derickrethans.nl - twitter: @derickr
http://derickrethans.nl/talks.php
http://joind.in/1534