Zend_Defaults - Gavin Vess

<ac:macro ac:name="note"><ac:parameter ac:name="title">In Progress</ac:parameter><ac:rich-text-body>
This proposal is not completely done, but please review anyway <ac:emoticon ac:name="smile" />
I am very open to suggestions, and even welcome others to dive in and edit this page! </ac:rich-text-body></ac:macro>

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Defaults Component Proposal

Proposed Component Name	Zend_Defaults
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Defaults
Proposers	Gavin
Revision	0.2 - 1 February 2007: Initial Notes Only (wiki revision: 13)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

Zend_Defaults provides a consistent mechanism both for sharing defaults between components, and for publishing defaults for consumption by specific components. Numerous past discussions on this topic resulted in conflicting goals, requirements, implementation designs, and dead-locked progress. This component aims to bypass these issues by instead purposely trying to be "stupidly simple", and offloading some of the decisions onto application developers, such as how to bring into scope the arrays of configuration data.

2. References

- Zend_Registry
- Zend_Session's setOptions() method

3. Component Requirements, Constraints, and Acceptance Criteria

- This component will establish mechanisms for developers to automate the process of providing default values to components.
- This component will not place restrictions on where and how developers choose to persist these default values.

- This component will not place restrictions the format used to persist these default values.
- This component will require the default values to be provided to this API in the form of a simple array of <key,value> pairs.
- This component will enable individual components to reliably query for preset defaults chosen by the application developer.
- This component will not force other components to use the defaults made accessible by this component.
- You are encouraged to suggest requirements here! .. yes you!

Requirement guidelines:

> *Most requirements take the form of "foo will do ...." or "foo will not support ...", although different words and sentence structure might be used. Adding functionality to your proposal is requirements creep (bad), unless listed below. Discuss major changes with your team first, and then open a "feature improvement" issue against this component.*
>
> - *This component **will** correctly reads a developers mind for intent and generate the right configuration file.*
> - *The generated config file **will not** support XML, but will provide an extension point in the API.*
> - *This component **will** use no more memory than twice the size of all data it contains.*
> - *This component **will** include a factory method.*
> - *This component **will not** allow subclassing. (i.e. when reviewed, we expect to see "final" keyword in code)*
> - *This component **will** only generate data exports strictly complying with RFC 12345.*
> - *This component **will** validate input data against formats supported by ZF component Foo.*
> - *This component **will not** save any data using Zend_Cache or the filesystem. All transient data **will be** saved using Zend_Session.*

4. Dependencies on Other Framework Components

- Zend_Registry
- ArrayObject

5. Theory of Operation

This component would exist at runtime as object(s) containing developer-determined application defaults. These object(s) would be placed inside Zend_Registry at well known (reserved) keys. These application-specific default values themselves would never be nakedly placed directly into the Zend_Registry to avoid mixing namespaced data with other application-specific entries in the registry.

6. Milestones / Tasks

Describe some intermediate state of this component in terms of design notes, additional material added to this page, and / code. Note any significant dependencies here, such as, "Milestone #3 can not be completed until feature Foo has been added to ZF component XYZ." Milestones will be required for acceptance of future proposals. They are not hard, and many times you will only need to think of the first three below.

- Milestone 0: Requirements obtain consensus from community and ZF devteam.
- Milestone 1: design notes will be published here
- Milestone 2: Working prototype checked into the incubator supporting use cases #1, #2, ...
- Milestone 3: Working prototype checked into the incubator supporting use cases #3 and #4.
- Milestone 4: Unit tests exist, work, and are checked into SVN.
- Milestone 5: Initial documentation exists.

If a milestone is already done, begin the description with "[DONE]", like this:

- Milestone #: [DONE] Unit tests ...

7. Class Index

- Zend_Defaults
- Zend_Defaults_Exception

8. Use Cases

UC-01

OLD way:

```
$config = new Zend_Config_Ini('myapp.ini', 'sandbox');
require_once 'Zend/Session.php';
Zend_Session_Core::setOptions($config->asArray());
```

New way:

```
$config = new Zend_Config_Ini('myapp.ini', 'sandbox');
Zend::setComponentOptions('Zend_Session', $config->asArray());


// inside Zend_Session:
public function setOptions($options = null)
{
   $componentOptions = Zend::getComponentOptions(__CLASS__);
   // now overlay $options on top of $componentOptions
   $options = array_merge($componentOptions, $options);
   // continue as usual by filling in any missing defaults with hard-coded values
below
   .
   .
}
```

We impose by convention the requirement for existence of setOptions(), and the necessary check for defaults using Zend::getComponentOptions().

The getComponentOptions() and array_merge() steps could be combined into a single function call.

UC-02

Sharing defaults between components relies on extremely simple use of "well-known" keys.
All of these keys and values are stored in an $instance of Zend_Registry, which is then stored via Zend::register('Zend', $instance) in the bootstrap.
This allows great flexibility within persistence mechanisms for the defaults, and a chance for userland code to tweak the loaded defaults.

OLD way: none

New way:

```
// general preferences and defaults shared by all ZF components
$config = array('Zend_Db_Adapter' => 'pdo_mysqli', 'optimize_for' => 'memory');
// the array could come from anywhere, including a DB, Zend_Config*, etc.
Zend::setOptions($config);


// inside Zend_Image_Converter:
public function setOptions($options = null)
{
   $componentOptions = Zend::getComponentOptions(__CLASS__);
   // now overlay $options on top of $componentOptions
   $options = array_merge(Zend::getOptions('optimize_for'), $componentOptions,
$options);
   // continue as usual by filling in any missing defaults with hard-coded values
below
   .
   .
}
```

UC-03

Please edit this page, and add more use cases! If you don't see an edit folder tab link at the top of the page, then submit the ZF CLA.

9. Class Skeletons

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>