Zend_Deprecation - Tobias Gies

<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_Deprecation Component Proposal

Proposed Component Name	Zend_Deprecation
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Deprecation
Proposers	Tobias Gies
Zend Liaison	TBD
Revision	1.0 - 3 June 2008: Initial Draft. (wiki revision: 5)

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_Deprecation is a component that is meant to offer developers both an easy way to control the behaviour of deprecation errors and a unified interface and nomenclature to provide the user with deprecation information.

The need for such a component arises as we approach version 2.0 and need to find a way to give developers information about soon-to-be-removed components while still maintaining a sensible amount of backwards compatibility.

The component is meant to be designed in a similar fashion to PHP's own error handling functionality.

2. References

- Naming Conventions for 2.0 - Matthew Ratzloff
- PHP error handling functionality

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** offer a completely static interface.

- This component **will** display error messages in the same way PHP does, respecting the setting of the ini variable `html_errors`
- This component **will** log error messages to an error stack which can be called later to retrieve the errors and, for example, log them to a file, output them to a command line, etc.
- This component **will** offer a way to configure which kinds of deprecation errors will be displayed, including an option to turn off the displaying of error messages completely.
- This component **will** offer a way to configure which kinds of deprecation errors will be logged, including an option to turn off the logging of error messages completely.

4. Dependencies on Other Framework Components

- Zend_Exception

5. Theory of Operation

In her bootstrap, the developer will be able to configure the behaviour of ALL deprecation errors that could ever be thrown during the lifetime of the script. During the script lifetime, deprecation errors thrown by the application will be handled with respect to the settings made in the bootstrap.

6. Milestones / Tasks

- Milestone 1: design notes will be published here
- Milestone 2: Working prototype checked into the incubator supporting use cases #1, #2
- Milestone 3: Unit tests exist, work, and are checked into SVN.
- Milestone 4: Initial documentation exists.

7. Class Index

- Zend_Deprecation

8. Use Cases

UC-01

Developer configures behaviour of deprecation error handling

```
Zend_Deprecation::setErrorReporting(Zend_Deprecation::D_CLASS_NAMING &
Zend_Deprecation::D_METHOD_NAMING);
Zend_Deprecation::setErrorLogging(Zend_Deprecation::D_ALL);

// The developer might also opt to turn the whole thing off
Zend_Deprecation::setErrorReporting(Zend_Deprecation::D_NONE);
Zend_Deprecation::setErrorLogging(Zend_Deprecation::D_NONE);
```

UC-02

Example of throwing a deprecation error in a deprecated component

```
class DeprecatedClass extends NewClass {
    public function __construct() {
        Zend_Deprecation::invokeError(Zend_Deprecation::D_CLASS_NAMING, "DeprecatedClass's new name is
NewClass!", __FILE__, __LINE__);
        parent::__construct();
    }
}
```

UC-03

Example of pulling deprecation errors from the stack later on

```
// Simply pull all data from the stack
$fullStack = Zend_Deprecation::getStackData();

// Or pull a specific error type...
$classNamingErrors = Zend_Deprecation::getStackData(Zend_Deprecation::D_CLASS_NAMING);
```

9. Class Skeletons

```
class Zend_Deprecation {
    const D_NONE = 0;
    const D_CLASS_NAMING = 1;
    const D_METHOD_NAMING = 2;
    const D_PARAMS = 4;
    const D_ALL = 7;
    protected static $_stack = array();
    private __construct() {}
    public static setErrorReporting($level) {}
    public static setErrorLogging($level) {}
    public static invokeError($level, $message, $file, $line) {}
    public static getStackData() {}
}

/**
 * An array returned by getStackData() is either empty or contains one or more arrays
 * of the following structure:
 */
array (
  'level' => Zend_Deprecation::[D_CLASS_NAMING|D_METHOD_NAMING|D_PARAMS],
  'message' => 'Stuff the component developer wants to tell you',
  'file' => '/path/to/file/in/which/error/occured.php',
  'line' => int // self-explanatory
)
```

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