Zend_Loader_Autoloader - Ralph Schindler

<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_Loader_Autoloader Component Proposal

Proposed Component Name	Zend_Loader_Autoloader
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Loader_Autoloader
Proposers	Ralph Schindler Matthew Weier O'Phinney
Zend Liaison	Matthew Weier O'Phinney
Revision	1.0 - 1 January 2008: Initial Draft. 1.1 - 17 December 2008: Initial Draft. (wiki revision: 10)

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_Loader_Autoloader is a component for autoloading registered namespaces & managing arbitrary autoload callbacks.

2. References

- SPL autoloading functions
- Matthew's pastebin autoloader implementation

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** register itself with spl_autoload
- This component **will** autoload the Zend and ZendX namespaces by default
- This component **will** allow registration of arbitrary namespaces that follow ZF/PEAR/Horde coding standards
- This component **will** allow itself to optionally act as a fallback autoloader for arbitrary namespaces
- This component **will** allow registering arbitrary autoload callbacks

    - This component **will** maintain an internal registry of these callbacks
    - This component **will** call these autoloaders itself
- This component **will** provide a flag to toggle suppression of class loading errors
    - The class loading error suppression flag **will** default to off (i.e., no suppression)

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_Loader

5. Theory of Operation

The component is available for use by applications that might use multiple autoloaders (for example modules that might require the use of a custom autoloader).

Problems this autoloader attempts to solve:

- Ability to autoload only **specific** namespaces
- Ability to handle arbitrary autoload callbacks. spl_autoload is problematic to manipulate when using object instance methods
- Ability to manage the ordering of the autoloader stack

6. Milestones / Tasks

- Milestone 1: [DONE] Requirements written and published as a proposal
- Milestone 2: [DONE] Working prototype checked into a public repository supporting the use cases
- Milestone 3: Community review of proposal
- Milestone 4: Acceptance of proposal
- Milestone 5: Unit tests exist, work, and are checked into icubator
- Milestone 6: Initial documentation exists
- Milestone 7: Approval for trunk

7. Class Index

- Zend_Loader_Autoloader
- Zend_Loader_Autoloader_Interface

8. Use Cases

UC-01

Standard usage; no extra libraries:

```
// Instantiation registers with spl_autoload_register;
// nothing else to do
Zend_Loader_Autoloader::getInstance();
```

UC-02

Register additional namespaces:

```
$autoloader = Zend_Loader_Autoloader::getInstance();
$autoloader->registerNamespace('Phly');
```

UC-03

Register an arbitrary autoloader callback:

```
Zend_Loader_Autoloader::getInstance();
Zend_Loader_Autoloader::pushAutoloader(array('My_Loader', 'autoload'));
```

UC-04

Use the autoloader interface to define an autoloader, and register an instance with the global autoloader.

```
class My_Autoloader implements Zend_Loader_Autoloader_Interface
{
    protected $_baseDir;
    protected $_prefix;
    protected $_prefixLen;

    public function __construct($baseDir, $prefix)
    {
        $this->_baseDir   = $baseDir;
        $this->_prefix    = $prefix;
        $this->_prefixLen = strlen($prefix);
    }

    public function autoload($class)
    {
        if (strlen($class) <= $this->_prefixLen) {
            return false;
        }
        if (0 !== strpos($class, $this->_prefix)) {
            return false;
        }

        $fragment = substr($class, $this->_prefixLen);
        $fragment = str_replace('_', '/', trim($fragment, '_'));
        return include($this->_baseDir . '/' . $fragment . '.php');
    }
}

$myAutoloader = new My_Autoloader('/path/to/classes', 'My_Stuff');
Zend_loader_Autoloader::pushAutoloader($myAutoloader);
```

UC-05

Suppress warnings.

```
Zend_Loader_Autoloader::getInstance()->suppressNotFoundWarnings(true);
```

UC-06

Use autoloader as a fallback autoloader (i.e., will load any namespace):

```
Zend_Loader_Autoloader::getInstance()->setFallbackAutoloader(true);
```

9. Class Skeletons

Zend_Loader_Autoloader

```
class Zend_Loader_Autoloader
{
    protected function __construct()
    {
        spl_autoload_register(array(__CLASS__, 'autoload'));
    }

    public function setAutoloaders(array $autoloaders);

    public function getAutoloaders();

    public function registerNamespace($namespace);

    public function unregisterNamespace($namespace);

    public function getRegisteredNamespaces();

    public function suppressNotFoundWarnings($flag = null);

    public function setFallbackAutoloader($flag);

    public function isFallbackAutoloader();

    public static function autoload($class);

    public static function getInstance();

    public static function pushAutoloader($callback);

    public static function unshiftAutoloader($callback);

    public static function removeAutoloader($callback);
}
```

Zend_Loader_Autoloader_Interface

```
interface Zend_Loader_Autoloader_Interface
{
    public function autoload($class);
}
```

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

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