Zend_Loader Improvements - Court Ewing

<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 Improvements Component Proposal

Proposed Component Name	Zend_Loader Improvements
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Loader Improvements
Proposers	Court Ewing
Zend Liaison	TBD
Revision	1.0 - 1 January 2008: Initial Draft. (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

The plugin loader and autoloader resources in the Zend_Loader package are currently designed to only work with non-namespaced class names (pre 5.3). These improvements will modify these components to work for both the existing class naming format as well as actual PHP 5.3 namespaces.

2. References

- PSR-0 Final Proposal

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** be backwards compatible
- This component **will not** change the current workflow when working with the existing package
- This component **will** allow traditional and modern namespaces to be used simultaneously for autoloading and pluginloading

4. Dependencies on Other Framework Components

No additional dependencies

5. Theory of Operation

The use of underscores and namespace separators should be fairly interchangeable; at the very least, they can both map directly to a directory structure. The improvements on Zend_Loader_PluginLoader and Zend_Loader_Autoloader_Resource will make it so plugins and resources utilizing namespaces from PHP 5.3 can be used in conjunction with existing infrastructure.

6. Milestones / Tasks

- [DONE] Milestone 1: Write proposal
- [DONE] Milestone 2: Develop use cases
- [DONE] Milestone 3: Write outline of changes as class skeletons
- [DONE] Milestone 4: Submit for community review
- Milestone 5: Gather feedback and apply changes
- Milestone 6: Write unit tests
- Milestone 7: Refactor existing classes

7. Class Index

- Zend_PluginLoader
- Zend_Autoloader_Resource

8. Use Cases

UC-1: Register PHP 5.3 namespace in autoloader resource

```
// Create a new resource pointing to a specified directory with the PHP 5.3 namespace 'Fully'
$resourceLoader = new Zend_Loader_Autoloader_Resource(array(
'basePath'  => 'path/to/some/directory',
'namespace' => 'Fully\\',
));
```

UC-2: Adding a prefix path to pluginloader that maps to a PHP 5.3 namespace

```
// Register the path 'Fully/Qualified/View/Helper' with the plugin loader
// All classes mapped to this path will have the PHP 5.3 namespace (prefix)
'\Fully\Qualified\View\Helper'
$loader = new Zend_Loader_PluginLoader();
$loader->addPrefixPath('Fully\Qualified\View\Helper\\', 'Fully/Qualified/View/Helper/');
```

9. Class Skeletons

```
class Zend_Loader_Autoloader_Resource implements Zend_Loader_Autoloader_Interface
{
    /**
     * To preserve complete backwards compatibility, this property will still
     * not contain a trailing namespace separator of any kind.
     */
    protected $_namespace;

    /**
     * ADDED - The type of namespace separator.
     *
     * @var string
     */
    protected $_namespaceSeparator = '_';

    /**
     * The constructor will now append the defined namespace separator rather
     * than assuming an underscore should be appended before adding it the main
     * autoloader.
     */
    public function __construct($options);

    /**
     * No changes would need to be done in this method.
     */
    public function __call($method, $args);

    /**
     * This can no longer assume that the class is divided by underscores. To be
     * consistent with the linked autoloading standard, it should replace both
     * underscores and backslashes with value of DIRECTORY_SEPARATOR.
     *
     * To accomplish this, backslashes in $class will be first converted to
     * underscores before the value is exploded into $segments.  $class is no
     * longer used after it is exploded, so this should not introduce any
     * conflicts.
     *
     * This can not assume that the component name is separated by underscores
     * either.  Instead, it should use the current namespace separator.
     */
    public function getClassPath($class);

    /**
     * No changes would need to be done in this method.
     */
    public function autoload($class);

    /**
     * No changes would need to be done in this method.
     */
    public function setOptions(array $options);

    /**
     * This method should now determine and set the namespace separator based on
     * the last character in the namespace.  If the last character is not a
     * backslash, default the separator to an underscore.
     *
     * To preserve backwards compatibility, remove all trailing underscores
     * and backslashes before setting the namespace.
     */
    public function setNamespace($namespace);

    /**
```

```
     * No changes would need to be done in this method.
     */
    public function getNamespace();

    /**
     * ADDED - Set the separator to be used for the current namespace
     *
     * @param  string $separator
     * @return Zend_Loader_Autoloader_Resource
     */
    protected function _setNamespaceSeparator($separator);

    /**
     * ADDED - Get the separator to be used for the current namespace
     *
     * @return string
     */
    public function getNamespaceSeparator();

    /**
     * No changes would need to be done in this method.
     */
    public function setBasePath($path);

    /**
     * No changes would need to be done in this method.
     */
    public function getBasePath();

    /**
     * This method will no longer assume underscores need to be trimmed from
     * $namespace and will instead rely on the currently defined namespace
     * separator.  The value associated with the namespace key of each element
     * in the resource types array will utilize the defined namespace separator
     * as well.  This change would also affect the keys of the component array.
     */
    public function addResourceType($type, $path, $namespace = null);

    /**
     * No changes would need to be done in this method.
     */
    public function addResourceTypes(array $types);

    /**
     * No changes would need to be done in this method.
     */
    public function setResourceTypes(array $types);

    /**
     * No changes would need to be done in this method.
     */
    public function getResourceTypes();

    /**
     * No changes would need to be done in this method.
     */
    public function hasResourceType($type);

    /**
     * No changes would need to be done in this method.
     */
    public function removeResourceType($type);

    /**
     * No changes would need to be done in this method.
     */
    public function clearResourceTypes();

    /**
     * No changes would need to be done in this method.
```

```
     */
    public function setDefaultResourceType($type);

    /**
     * No changes would need to be done in this method.
     */
    public function getDefaultResourceType();

    /**
     * This method should no longer assume an underscore should be used to
     * separate the resource type namespace from the provided resource name.  It
     * should instead use the currently defined namespace separator.
```

```
     */
    public function load($resource, $type = null);
}
```

```
<?php

class Zend_Loader_PluginLoader implements Zend_Loader_PluginLoader_Interface
{
    /**
     * No changes would need to be done in this method.
     */
    public function __construct(Array $prefixToPaths = array(), $staticRegistryName = null);

    /**
     * This method should only append an underscore if the last character of
     * $prefix is not an underscore or backslash.
     */
    protected function _formatPrefix($prefix);

    /**
     * No changes would need to be done in this method.
     */
    public function addPrefixPath($prefix, $path);

    /**
     * No changes would need to be done in this method.
     */
    public function getPaths($prefix = null);

    /**
     * No changes would need to be done in this method.
     */
    public function clearPaths($prefix = null);

    /**
     * No changes would need to be done in this method.
     */
    public function removePrefixPath($prefix, $path = null);

    /**
     * No changes would need to be done in this method.
     */
    protected function _formatName($name);

    /**
     * No changes would need to be done in this method.
     */
    public function isLoaded($name);

    /**
     * No changes would need to be done in this method.
     */
    public function getClassName($name);

    /**
     * No changes would need to be done in this method.
     */
    public function getClassPath($name);

    /**
     * This method will no longer assume that classes are namespaced with
     * underscores.  Instead, it will replace backslashes and underscores with
     * DIRECTORY_SEPARATOR.
     */
    public function load($name, $throwExceptions = true);
```

```
    /**
     * No changes would need to be done in this method.
     */
    public static function setIncludeFileCache($file);

    /**
     * No changes would need to be done in this method.
     */
    public static function getIncludeFileCache();

    /**
     * No changes would need to be done in this method.
```

```
     */
    protected static function _appendIncFile($incFile);
}
```

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