Naming conventions for 2.0 - Matthew Ratzloff

<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: Naming conventions for 2.0 - Matthew Ratzloff Component Proposal

Proposed Component Name	Naming conventions for 2.0 - Matthew Ratzloff
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Naming conventions for 2.0 - Matthew Ratzloff
Proposers	Matthew Ratzloff
Zend Liaison	TBD
Revision	1.0 - May 25, 2008: Incomplete initial draft (wiki revision: 12)

Table of Contents

1. Overview

To do: Zend_Search, methods, Boolean accessors

Since the initial preview in March 2006, there hasn't really been much in the way of direction given on naming standards for classes in Zend Framework. This lack of consistency is what has yielded two nomenclatures for CLI components ("Zend_Console" vs. "Zend_Controller_Response_Cli"), differing terms for similar concepts ("Zend_Json_Decoder" vs. "Zend_Mime_Decode"), both nouns and verbs in equal measure ("Zend_Loader" but not "Zend_Translator"; alternately, "Zend_Translate" but not "Zend_Load"), and so on.

I first brought this issue up in February 2007 and concluded, "Consistency means predictability, which means being able to recall names without having to check the manual every time. It's why most people can't use PHP's date or string functions without looking at the documentation, for example." Beyond this there are also intangible benefits, such as presenting a more professional appearance. Because many skilled PHP developers contribute to this project, the Zend Framework community also has an opportunity to lead the general PHP community by example.

There was a lot of agreement from the community (and a couple Zenders) but no movement because the framework was preparing for 1.0. At the time I was told, "There is life after 0.9.0," and so as we prepare for 2.0, I think it's the perfect time to revisit this.

Note: Because this is a non-traditional proposal, the bulk of the proposal will be in the first three sections.

2. References

- Java Naming Conventions
- .NET Guidelines for Names (versions 1.1, 2.0, 3.0)
- ActionScript 2.0 Naming Conventions (3.0 apparently uses 2.0 naming conventions)

3. Component Requirements, Constraints, and Acceptance Criteria

Class Names

There are currently 1074 classes that comprise Zend Framework. With few exceptions, these all follow similar standards found in other languages and frameworks. For example, that Java Naming Conventions state,

> *Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).*

The .NET Guidelines for Names state the following:

> *In general, type names should be noun phrases, where the noun is the entity represented by the type. For example, Button, Stack, and File each have names that identify the entity represented by the type. Choose names that identify the entity from the developer's perspective; names should reflect usage scenarios.*

ActionScript Naming Conventions are similar:

> *Class names are usually nouns or qualified nouns. (...) Don't use nouns that also might be interpreted as verbs. For example, Running, or Gardening. Using these nouns might lead to confusion with methods, states, or other application activities.*

Therefore, I propose the Zend Framework Coding Standard section B.3 be modified to include language similar to the following:

> *Class names must be nouns, noun phrases, or proper nouns (i.e., protocols, formats, algorithms, PHP extensions, or products), with two exceptions.*
>
> *In cases of classes that represent implementation strategies, an unambiguous adjective is permitted. For example, Zend\Controller\Router\Rewrite is acceptable (it should be interpreted as "RewriteRouter"), but Zend\Loader\Plugin is not, as it is unclear whether the class represents a Zend::Loader plugin or a generic plugin loader. (In this case it is the latter; the class is actually named Zend\Loader\PluginLoader.)*
>
> *Gerunds (verbs in noun form ending in "-ing") are permitted when a suitable noun cannot be found. The Zend\Measure\Cooking namespace is an example of this.*
>
> *Verbs, adverbs, and prepositions are not permitted.*

Abstract Classes and Interfaces

As Matthew Weier O'Phinney notes, there are issues with PHP 5.3's namespace implementation and the framework's pre-2.0 standard of interface names in the form of `Zend_Example_Interface` and quasi-standard of abstract class names in the form of `Zend_Example_Abstract`. As such, I propose the following:

> *Abstract classes and interfaces are special cases and therefore are treated as such. Abstract classes must be named in the form of `Zend\Example\AbstractExample`, while interfaces are named in the form of `Zend\Example\ExampleInterface`.*

Abbreviations

> *Abbreviations are acceptable, so long as they are universally understood and unambiguous, and the unabbreviated word is sufficiently long. Examples include "db", "config", and "info". "Auth" is acceptable so long a disambiguation note is included in the*

> *documentation. Examples of out-of-conformance words include "str".*

I'd like some community input here. Which abbreviations are unacceptable? What are some examples of common abbreviations that aren't acceptable?

Our Lexicon

> *A consistent set of names for common concepts is important for the framework to feel like a cohesive whole. To that end, we have standardized on certain words: Alphabetic not Alpha or Text, Alphanumeric not Alnum or TextNum, Directory not Dir, Integer not Int, and Utilities not Util or Utils.*

These specific word choices are of course debatable and community input is welcome. Naturally, this is also related to the above section.

Specific class name change recommendations

These are specific name change recommendations.

Old class name	New class name	Notes
Zend_Acl_Assert	Zend\Acl\Assertion	
Zend_Filter_Alnum	Zend\Filter\Alphanumeric	1
Zend_Filter_Alpha	Zend\Filter\Alphabetic	1
Zend_Filter_Dir	Zend\Filter\Directory	1
Zend_Filter_Int	Zend\Filter\Integer	1
Zend_Controller_Action	Zend\Controller\AbstractAction	
Zend_Controller_Response_Cli	Zend\Controller\Response\Console	
Zend_Db_Statement_Oracle*	Zend\Db\Statement\Oci*	
Zend_Db_Table_Rowset	Zend\Db\Table\RowSet	2
Zend_Gdata_App_Util	Zend\Gdata\App\Utilities	1
Zend_Gdata_Calendar_Extension_Timezone	Zend\Gdata\Calendar\Extension\TimeZone	2
Zend_Gdata_Exif_Extension_FStop	Zend\Gdata\Exif\Extension\Fstop	2
Zend_Log_Filter_Suppress	Zend\Log\Filter???	
Zend_Mime_Decode	Zend\Mime\Decoder	
Zend_Service_Amazon*	Zend\Service\Amazon\Associates*	4
Zend_Service_StrikeIron_USAddressVerification	Zend\Service\StrikeIron\UsAddressVerification	2
Zend_Service_Technorati_Utils	Zend\Service\Technorati\Utilities	1
Zend_Translate	Zend\Translator	
Zend_Validate*	Zend\Validator*	
Zend_Validate_Alnum	Zend\Validator\Alphanumeric	1
Zend_Validate_Alpha	Zend\Validator\Alphabetic	1
Zend_Validate_Ccnum	Zend\Validator\CreditCard	3
Zend_Validate_Int	Zend\Validator\Integer	1

Zend_Validate_Ip	Zend\Validator\IpAddress	3

1 Subject to community input about abbreviations and/or lexicon
2 Does not follow existing naming standard
3 For reasons of clarity
4 This must be namespaced to allow for EC2, S3, etc.

4. Dependencies on Other Framework Components

5. Theory of Operation

6. Milestones / Tasks

7. Class Index

8. Use Cases

9. Class Skeletons

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