Zend_Thrift - Theoziran Lima

<ac:macro ac:name="info"><ac:parameter ac:name="title">Zend_Thrift - Theoziran Lima</ac:parameter><ac:rich-text-body>
This page has been created from a template that uses "zones." To proceed:

Edit the page
Replace sample content within each zone-data tag with your own content
Remove this notice
Save the page
When you are ready for community review, move this page to the <ac:link><ri:page ri:content-title="Ready for Review" /></ac:link> section on the edit page.

<ac:macro ac:name="note"><ac:parameter ac:name="title">No placeholders allowed!</ac:parameter><ac:rich-text-body>
Please do not create placeholders. Wait until you have sufficient content to replace all sample data in the proposal template before creating your proposal document. </ac:rich-text-body></ac:macro></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_Thrift Component Proposal

Proposed Component Name	Zend_Thrift
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Thrift
Proposers	Theoziran Lima
Zend Liaison	TBD
Revision	0.1 - 13 May 2011: Initial Draft. (wiki revision: 3)

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

Thrift is a software framework for scalable cross-language services development. It combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, and OCaml.
Zend_Thrift is a componente that helps to create Thrift services.

2. References

- Thrift: Scalable Cross-Language Services Implementation

3. Component Requirements, Constraints, and Acceptance Criteria

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_Exception

5. Theory of Operation

The component is instantiated with a mind-link that ...

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 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_Thrift_Exception
- Zend_Thrift (factory class)

8. Use Cases

UC-01

... (see good use cases book)

9. Class Skeletons

```
class Zend_Thrift_Exception extends Zend_Exception {}

class Zend_Thrift {
...
```

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