Zend_Message - Matthew Weier o'Phinney

<ac:macro ac:name="info"><ac:parameter ac:name="title">Zend_Message</ac:parameter></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_Message Component Proposal

Proposed Component Name	Zend_Message
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Message
Proposers	Matthew Weier O'Phinney Alexander Veremyev (Zend Liaison)
Revision	1.5 - 16 June 2008: Renamed, added usecase from comments 1.4 - 11 March 2008: updated class skeletons and text 1.3 - 20 Februari 2008: Updated link and added another usecase 1.2 - 3 January 2008: Added some use cases, operation description and updated skeletons 1.1 - 30 December 2007: Refactoring and renaming 1.0 - 29 December 2007: Setting up first draft (wiki revision: 32)

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_Message provides an infrastructure to relay messages between objects that don't necessarily know eachother.
This provides a loose coupling between components and makes your app more flexible.

Also, this might be of interest for the Zend_Tool CLI tooling?

An implementation of this proposal is available in Zym.

2. References

- Zym_Notification
- Zym_Notification docs
- Cocoa NSNotification documentation

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **must** provide a clean, non-intrusive way to handle event notification
- **Must not** require existing classes to implement interfaces or extend classes for it to work

4. Dependencies on Other Framework Components

- Zend_Exception

5. Theory of Operation

The operation is very simple, but also very powerful. You can attach an object to an event by calling the notification center's attach() method. This method takes the observer and the name of the event as argument. More than one object can register to the same event.
When a message is sent, all objects that subscribed to that event will be notified in order of subscription.
By default the observer's notify() method will be called. You can also register another method when registering the observering object at the message dispatcher.

The message constructor takes three arguments (of which the last is optional). The first is the name of the message. The second is (usually) the object that sent the message.
The last argument is an array that allows for extra data to be sent along with the message.

When attaching an observer to an event you can either use a string to attach it to just one event, or provide an array with multiple event names. (See usecase UC-02)
When you attach an observer to an event with an asterisk in the name, the asterisk will be used as wildcard. See usecase UC-03 for an example.

6. Milestones / Tasks

- Milestone 1: [DONE] finish this proposal
- Milestone 2: [DONE] write documentation
- Milestone 3: [DONE] write unit tests
- Milestone 4: [IN PROGRESS] process comments
- Milestone 5: get approval and move the component to incubator
- Milestone 6: finish component and move it to core

7. Class Index

- Zend_Message
- Zend_Message_Dispatcher
- Zend_Message_Registration
- Zend_Message_Exception

8. Use Cases

9. Class Skeletons