Zend_Model - Jurriën Stutterheim

<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_Model Component Proposal

Proposed Component Name	Zend_Model
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Model
Proposers	Jurriën Stutterheim
Revision	0.1 - 4 January 2008: Initial setup 0.9 - 8 July 2008: Major rewrite 0.10 - 9 July 2008: Added a few interfaces 0.11 - 27 July 2008: Major updates (wiki revision: 27)

Table of Contents

1. Overview

Zend Framework provides a powerful Model View Controller (MVC) implementation. The View and the Controller are very well featured in this implementation. However, the Model part is still left largely up to the developer. Unfortunately it is not clear for a lot of developers how the Model should be implemented. What exactly is its purpose? What are its responsibilities? What is a decent Model design? How can my Model make use of Zend_Db or a web service? How do I make my Model implementation flexible?

One (if not the most) important aspect of the Model implementation in Zend Framework should be documentation. The documentation should answer those (and probably more) questions for developers. It should give them a head start to explore a good application design for their models. This could be done by providing common use cases and possible implementations of those use cases.

To support this, Zend_Model provides some interfaces to identify a class as being a model. Additional interfaces could be used to identify common methods for models.

This proposal is also setup to provide a place for Model discussions. For the creation of the suggested documentation I would want to setup a focus group to come up with common use cases (and their implementations) for web applications.

There are a few topics related to models that aren't covered (fully) in this proposal yet. For these topics I would like some feedback from you all.

Relations
First off are relations between models. For example, a blog can have multiple posts, which in turn can contain multiple comments. Each post also

has one author. A post belongs to a blog. You get the point  These are relations that are actually quite common. For a blog post the following relations would be relevant: hasOne (author), hasMany (comments), belongsTo (blog). These relations are already supported by Zend_Db. Should they be supported by Zend_Model as well?

Multiple data sources
In some cases you might want multiple data sources for your model (e.g. multiple web services). The Data_Interface I'm currently suggesting doesn't cover this. Would an interface that supports multiple data sources be wanted? How would this look?

To/From array
Throughout the framework methods like toArray() and setFromArray() are used. It would be nice if there were an interface to indicate the presence of these methods. Would this have a place in Zend_Model?

Singleton
The same question goes for the Singleton interface. Does this have a place in Zend_Model? What would be a use-case where you want your Model to be a singleton? (the idea for this came from Agavi). Also, this interface could be used by other components with singleton classes. The name Model would probably cause confusion. What could be a better namespace for an interface that defines a singleton?

2. References

- Why ActiveRecord sucks
- ActiveRecord sucks, but Kore Nordmann is wrong
- ActiveRecord does not suck
- Fat Models and the Data Access Layer
- Zym implementation
- Agavi
- Mailing list discussion #1
- Mailing list discussion #2
- Mailing list discussion #3

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **must** provide a comprehensive documentation explaining the implementation of the M in MVC.
- This component **must** provide a generic interface for models
- This component **must** be independent of data source (e.g. database, web service, file system etc.)
- This component **must not** discard functionality that current components provide

4. Dependencies on Other Framework Components

5. Theory of Operation

For this section the PHP 5.3 naming scheme will be used. (See Class Index)
Zend_Model consists of a few interfaces that can be implemented by your models.

Zend_Model_Interface
This is an empty interfaces, which serves only to indicate that the class is in fact a model. This serves no other reason other than making the Model aspect of ZF more tangible.

Zend_Model_Data_Interface
This provides the getDataSource() method, which would serve to fetch the data source from the model. This would only be useful for simple scenarios where the model does not have multiple data sources.

Zend_Model_Form_Interface
This provides a getForm() method. As the name indicates, this fetches the Zend_Form instance from the model, should it have one. Again, this might not be usable in every situation.

Zend_Model_Singleton_Interface
This provides a public static getInstance() method, which is the name which is commonly used to get singleton instances in the framework.

6. Milestones / Tasks

- Milestone 1: [DONE] Working prototype
- Milestone 2: [DONE] Finish first draft of this proposal
- Milestone 3: [IN PROGRESS...] Collect & process feedback
- Milestone 4: Component is incubated
- Milestone 5: Documentation & Unit Tests
- Milestone 6: Move to core

7. Class Index

Naming suggestions are welcome!

PHP < 5.3

- Zend_Model_Interface
- Zend_Model_Data_Interface
- Zend_Model_Form_Interface
- Zend_Model_Singleton_Interface

PHP >= 5.3

- Zend::Model::IModel
- Zend::Model::IData
- Zend::Model::IForm
- Zend::Model::ISingleton

8. Use Cases

9. Class Skeletons

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