Zend_Db_Table - Fabien Marty, Eugene Panaitov, Simon Mundy

<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_Db_Table Enhancements Component Proposal

Proposed Component Name	Zend_Db_Table Enhancements
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Db_Table Enhancements
Proposers	Fabien MARTY Eugene Panaitov Simon Mundy Darby Felton, Zend liaison
Revision	0.1 - 14 October 2006: Summary of community feedback on ZF Mailing list. (wiki revision: 8)

Table of Contents

1. Overview

This proposal summarises several requests on the mailing list and the issue tracker to provide enhancements to the Zend_Db_Table component. It is a response to Gavin Vess's request to formalise these enhancements and not meant to override or overlap any existing Db work already being performed. It's been posted here so that more focussed feedback can be given and to ensure these enhancements follow the established procedures for accepting ZF components.

2. References

- ZF General Mailing List
- ZF Issue Tracker

3. Component Requirements, Constraints, and Acceptance Criteria

- This component will allow developers to extend the capability of Zend_Db_Table rows and rowsets.
- It will allow developers to specify a custom class for rowsets / rows in a Zend_Db_Table instance.
- It will provide better ORM capabilities by allowing the possibility of chained queries and domain logic outside of the Db itself.
- It will allow Zend_Db_Table_Row objects to be serialized (removing dependency on stored DB connections)
- It will provide methods to allow custom pre-filtering of data for insertion/updates to the database.
- It will allow arrays for primary keys (for composite keys on tables)

4. Dependencies on Other Framework Components

- Zend_Db_Table
- Zend_Db_Exception

5. Theory of Operation

The component allows developers to override the default class used for rowsets and rows, meaning that result sets for single rows or collections of rows may be instantiated with user-defined classes.

Custom rows can also have hooks to perform user-transformations to data before rows are inserted or updated. The empty _insert() and _update() methods are called before data is committed and operate directly on row data using transformed columns (e.g. if you had camelCaps enabled, you would set $this->clientListId = 3 rather than $this->client_list_id = 3).

To allow rows and rowsets to be serialized (as they can contain domain logic rather than simply hold database values), the database is not stored within the row or rowset instance and is instead accessed by querying an instance of the parent table. A reference to the parent table is provided as a classname so that a row can take advantage of lazy-loading.

Because tables can also have composite keys, the primary key will allow arrays, and this will be enforced during a 'find()' operation - if the primary key is an associative array, all keys must be assigned values in the find() method or an exception will be thrown.

The enforcement of 'camelCaps' and other case transformations will not be set by default, however the option to add these will remain. The method setColumnCase() will create a new column filter for a Zend_Db_Table and all rows/rowsets will inherit this transformation.

6. Milestones / Tasks

- Milestone 1: Community discussion and initial acceptance by Zend Db team.
- Milestone 2: Working prototype checked into the incubator.
- Milestone 3: Unit tests exist, work, and are checked into SVN.
- Milestone 4: Update existing documentation and use-cases.

7. Class Index

- Zend_Db_Table
- Zend_Db_Table_Rowset
- Zend_Db_Table_Rowset_Abstract
- Zend_Db_Table_Row
- Zend_Db_Table_Row_Abstract

8. Use Cases

9. Class Skeletons

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