Zend_Db_ActiveRecord

<ac:macro ac:name="note"><ac:parameter ac:name="title">Under Construction</ac:parameter><ac:rich-text-body>
This proposal is under construction and is not ready for review. </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_Db_ActiveRecord Component Proposal

Proposed Component Name	Zend_Db_ActiveRecord
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Db_ActiveRecord
Proposers	Ken Chau
Revision	1.0 - 03/05/2008: Initial draft (wiki revision: 25)

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_Db_ActiveRecord is an implementation of the ActiveRecord pattern based on the Patterns of Enterprise Application Architecture book by Martin Fowler. It also has similar functionality as the Ruby on Rails ActiveRecord implementation. This implementation relies on PHP 5.3's late static binding capability (currently not in stable yet).

2. References

- Martin Fowler's Patterns of Enterprise Application Architecture Book (PoEAA)
- Ruby on Rails implementation
- What's new in PHP 5.3?

3. Component Requirements, Constraints, and Acceptance Criteria

An implementation of ActiveRecord according to PoEAA (Fowler, 161) should have the following characteristics:

- Construct an instance of the Active Record from a SQL result set row

- Construct a new instance for later insertion into table
- Static finder methods to wrap commonly used SQL queries and return Active Record objects
- Update the database and insert into it the data in the Active Record
- Get and set the fields
- Implement some pieces of business logic

In the Zend Framework context, here are the requirements:

- This component **will** perform CRUD operations on a table mapped to an ActiveRecord object.
- This component **will** use existing Zend_Db for all actual database operations.
- This component **will** use static find methods and treat instantiations to be an object representation of a table row.
- This component **will** allow for global default configuration much like Zend_Db_Table_Abstract.
- This component **will** work with validators and filters as "some pieces of business logic"

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_Db
- Zend_Db_Select

5. Theory of Operation

```
// Create a Zend_Db adapter
$db = Zend_Db::factory('Pdo_Mysql', array(
    'host'     => '127.0.0.1',
    'username' => 'root',
    'password' => '',
    'dbname'   => 'test'
));

// Pass it on as the default adapter for all Zend_Db_ActiveRecord classes
Zend_Db_ActiveRecord_Abstract::setDefaultAdapter($db);

// Declare a class 'Person' to extend an (abstract) Zend_Db_ActiveRecord class
class Person extends Zend_Db_ActiveRecord_Abstract { }

// Simple find by id
// TODO more finder examples to come!
$p = Person::find(1);
$p->name = 'Test Person 1';
$p->save();

// Creating new Person
$p2 = new Person;
$p2->name = 'Test Person 2';
$p2->save();

// Delete
$p3 = Person::find(3);
$p3->delete();
```

6. Milestones / Tasks

- 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.

7. Class Index

- Zend_Db_ActiveRecord_Abstract

8. Use Cases

UC-01

Declaring an ActiveRecord Model, defining Relationships

```
class Person extends Zend_Db_ActiveRecord_Abstract
{
    public static function initRelationships()
    {
        self::hasOne('Group', $groupOptions);
        self::hasOne('Person as Nemesis', $nemesisOptions);
        self::hasMany('Gadget as Gadgets', $gadgetOptions);
        /* ...more definitions... */
    }
}
```

UC-02

Create, Update, Delete

```
$bob = new Person;
$bob->name = "Bob";
$bob->save();

$bob->name = "Robert";
$bob->save();

$bob->delete();
```

UC-03

Find a record in the table by ID and by where clause

```
$bob = Person::find(5);
$voters = Person::find("all", "age >= 18");
$firstLady = Person::find("first", "gender = 'female'");
```

UC-04

Find records in the table by a custom SQL

```
$sql = Person::select()->where(array('age < ?', 2));
$babies = Person::findBySql($sql);
```

UC-05

One-to-one relationship

In this example use case, a Driver owns a Car and only one Car. Note that the foreign key field exists on the owned item.

```
class Driver extends Zend_Db_ActiveRecord_Abstract
{
 public static function initRelationships()
 {
  // Driver.car_id points to Car.id
  self::hasOne('Car', array('local'=>'id', 'foreign'=>'driver_id'));
 }
}

class Car extends Zend_Db_ActiveRecord_Abstract
{
 public static function initRelationships()
 {
  self::hasOne('Driver', array('local'=>'driver_id', 'foreign'=>'id');
 }
}

$car = new Car;
$car->make = "Honda"
$car->model = "Civic LX"
$car->save();

$driver = new Driver;
$driver->Car = $car;
```

UC-06

One-to-many relationship

```
class Person extends Zend_Db_ActiveRecord_Abstract
{
 public static function initRelationships()
 {
  // Person.group_id is a foreign key to Group.id
  self::hasOne('Group', array('local'=>'school_id', 'foreign'=>'id'));
 }
}

class School extends Zend_Db_ActiveRecord_Abstract
{
 public static function initRelationships()
 {
  self::hasMany('Person as Students', array('local'=>'id', 'foreign'=>'school_id');
 }
}


// Creating a new member on the fly
$college = new School;
$college->name = "College";
$college->Students[0]->name = "Bob";

$college->save();
```

UC-07

Many-to-many relationship

```
class Person extends Zend_Db_ActiveRecord_Abstract
{
 public static function initRelationships()
 {
  // Person.group_id is a foreign key to Group.id
  self::hasMany('Group as Groups', array('local'=>'person_id', 'foreign'=>'group_id', 'refClass' =>
'Membership'));
 }
}

class Group extends Zend_Db_ActiveRecord_Abstract
{
 public static function initRelationships()
 {
  self::hasMany('Person as Members', array('local'=>'group_id', 'foreign'=>'person_id', 'refClass' =>
'Membership');
 }
}


// Creating a new member on the fly
$person = new Person;
$person->Groups[0]->name = 'First Group';
$person->Groups[1]->name = 'Second Group';

$person->save();

// deleting the associations
$person->Membership->delete();
```

9. Class Skeletons

```
...
```

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