Практическое применение паттерна ActiveRecord для PHP. Краткий обзор ORM решений для PHP

Молтянинов Илья
ООО БИТ

конференция “Инженерия ПО”
апрель 2007, г. Пенза

Цели:

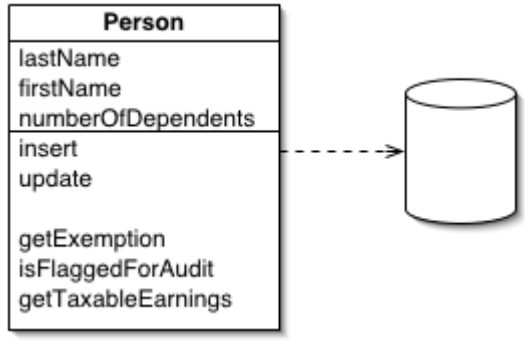
- Сделать обзор основных решений ORM для языка PHP:
  - Propel
  - Doctrine
  - EZPDO
- Показать собственный опыт в разработке и использовании ORM решения для RAD (Rapid Application Development) - lmbActiveRecord

Общая информация

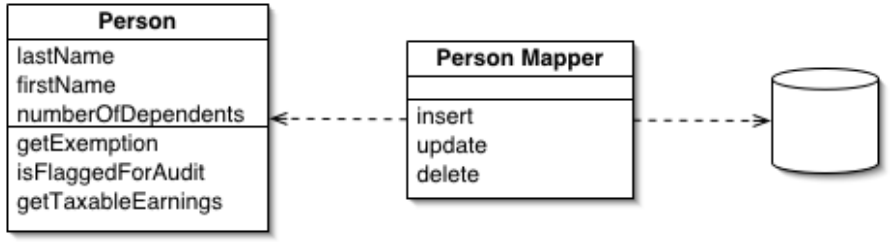
- ORM – Object-Relational Mapping
- Решает задачи:
  - преобразования объектов в форму, для сохранения в persistence layer
  - извлечения данных в последующем, с сохранением свойств объектов и отношений между ними

Паттерны, относящиеся к ORM

- **ActiveRecord** – объект представляет одну запись для таблицы в БД. Инкапсулирует данные и логику

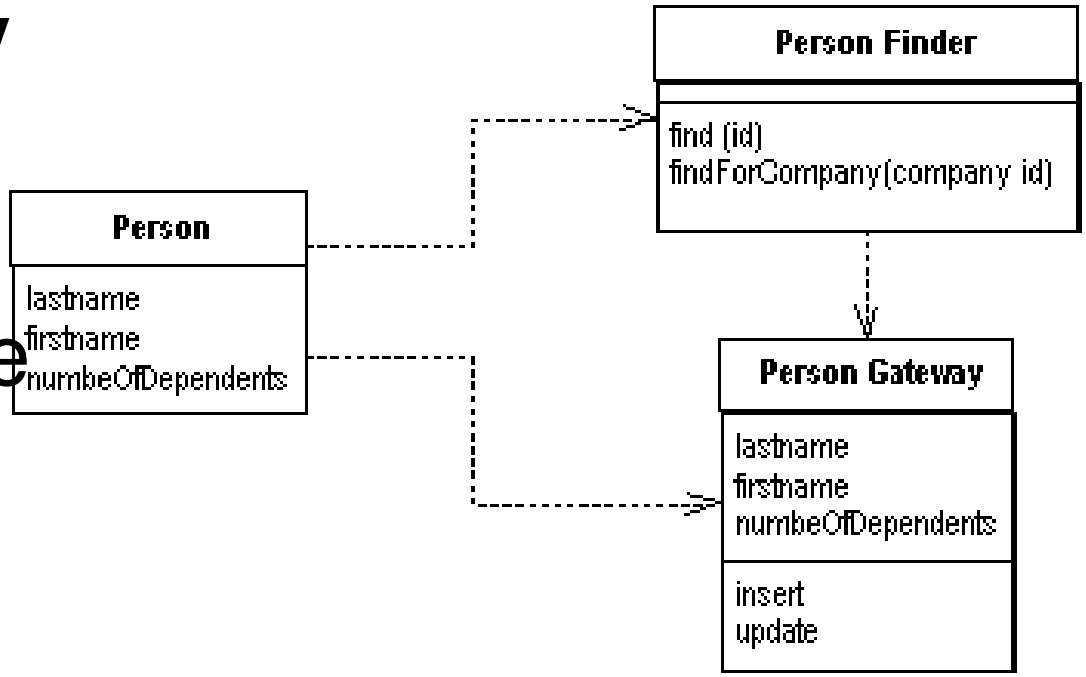


- **Data Mapper** – отвечает за передачу данных между



Паттерны...

- **Row Data Gateway** – работает как gateway для одной записи в источнике данных. Отделяет логику по сохранению от бизнес-логики объекта.



patterns

- .... и многое другое на http://www.martinfowler.com/eaaCatalog/

Требования к ORM для RAD

- Простота и удобство API
  - Краткость и выразительность описания объектов и их отношений
- Возможность оптимизации (например, написания конкретных SQL запросов для увеличения быстродействия)

Propel

- Object Relational Mapping (ORM) framework for PHP5 (http://propel.phpdb.org/trac/)
- Порт на php от Apache Torque - object persistence for Java
- Использует PHING - build system для php основанная на Apache Ant
- Использует Creole(DB Abstraction for PHP5) для общения с БД
- Лицензия: GNU LGPL

Propel

- Реализует паттерн Table Row Gateway
- Схема данных первична
- Схема описывается в виде xml
- XML можно сгенерировать уже из готовой схемы данных в БД

Propel

- Расширение функционала класса с доменной логикой решается наследованием от базового (генерируемого) класса
- На основе XML генерируются – creation-SQL; классы gateway (BaseClass), finder (BasePeerClass), DB Map (ClassMap).

Пример XML Propel

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<database name="bookstore">

  <table name="book">

    <column name="book_id" type="INTEGER"
          required="true" primaryKey="true"/>

    <column name="title" type="VARCHAR" size="50"
          required="true" />

  </table>

</database>
```

Пример кода Propel

```
$b = new Book();

$b->setTitle("War & Peace");

$b->save();


$c = new Criteria();

$c->add(BookPeer::TITLE, "War%", Criteria::LIKE);

$c->setLimit(10);


$books = BookPeer::doSelect($c);

foreach($books as $book)
    print "<br/>" . $book->getTitle();
```

Возможности PROPEL

- Поддержка типов данных: boolean, numeric, string, BLOB, Date
- Поддержка отношений между объектами:
  - one-to-one
  - many-to-many
- Поддержка каскадного удаления объектов по отношениям
- Abstract Entities – как механизм

Пример XML many-to-many

```
<table name="book_reader_ref">
  <column name="book_id" type="INTEGER" required="true" primaryKey="true"/>
  <column name="reader_id" type="INTEGER" required="true" primaryKey="true"/>
    <foreign-key foreignTable="book">
      <reference local="book_id"   foreign="book_id"/>
    </foreign-key>
    <foreign-key foreignTable="reader">
      <reference local="reader_id" foreign="reader_id"/>
    </foreign-key>
</table>
```

Пример кода many-to-many

```
$books = BookPeer::doSelect(new Criteria());


foreach($books as $book)
 $readers =
   $book->getBookReaderRefsJoinReader();
```

Достоинства Propel

- Хорошее и понятное API
- Минимально достаточный функционал
- Поддержка основных вендоров БД (за счет Creole)

Недостатки PROPEL

- В основном связаны с “неудобством” конфигурирования схемы в виде XML и необходимостью перегенерации классов при изменении схемы данных

DOCTRINE

- PHP Data Persistence and ORM Tool
- PDO в качестве DBAL
- LGPL
- http://www.phpdoctrine.org/

Пример кода

```
class Forum_Category extends Doctrine_Record {
    public function setTableDefinition() {
        $this->hasColumn("root_category_id", "integer", 10);
        $this->hasColumn("parent_category_id", "integer", 10);
        $this->hasColumn("name", "string", 50);
        $this->hasColumn("description", "string", 99999);
    }
    public function setUp() {
        $this->hasMany("Forum_Category as Subcategory", "Subcategory.parent_category_id");
        $this->hasOne("Forum_Category as Rootcategory", "Forum_Category.root_category_id");
    }
}
```

Doctrine

- Поддержка многих типов данных
- Поддержка псевдо-enum
- Конфигурируемость на уровнях
  - Global level
  - Connection level
  - Table level

Doctrine

- Поддержка отношений объектов
  - One-to-One
  - One-to-Many, Many-to-One
  - Many-to-Many
  - Self-referencing
- Поддержка наследования объектов
- Поддержка иерархических данных (Adjacency list, Nested set, Materialized path)

Doctrine

- Eventlisteners
- Validators
- TableView – аналог Query Alias
- Cache
- Locking Manager
- Lazy DB Connection
- Прозрачные настраиваемые multiple DB connections

Doctrine

- connection binding

```
$conn = $manager->openConnection(new
   PDO('dsn','username','password'), 'connection 1');

  $conn2 = $manager->openConnection(new
   PDO('dsn2','username2','password2'), 'connection
   2');

  $manager->bindComponent('User', 'connection 1');

  $manager->bindComponent('Group', 'connection 2');
```

Doctrine

- UnitOfWork – обеспечивает атомарность, целостность, изоляцию данных и отказоустойчивость
- Вложенность транзакций UoW
- SavePoints – откат на конкретную позицию цепочки транзакций
- Isolation levels для транзакций

Doctrine

- Кэширование запросов – несколько драйверов (Memcache, APC)
- Export – создание creation-sql
- Поддержка raw sql
- Свой язык запросов Doctrine QL:

```
$q = new Doctrine_Query();

$rows = $q->update('Account')->set('amount', 'amount + 200')
          ->where('id > 200')->execute();
```

Недостатки Doctrine

- неполная документация
- Ранняя версия
- Возможность забрать только из SVN

Достоинства DOCTRINE

- Мощный ORM инструмент
- Гибко конфигурируется
- Огромный функционал
- Очень хочется попробовать библиотеку в деле :)

EZPDO

- ORM для php5 (http://ezpdo.org)
- LGPL
- DB persistence - adodb, peardb
- DB vendors – MySQL, PgSQL, Sqlite

EZPDO features

- One-to-one, One-to-many, many-to-many
- Unit Of Work
- EZQL
- Array access
  - $author['name'] == $author->name

EZPDO

```
class Author extends Base {
    /**
    * Name of the author
    * @var string
    * @orm char(64)
    */
    public $name;
}
```

EZPDO

```
/**
* Books written by the author
* @var array of Book
* @orm has many Book
*/
public $books = array();
```

Достоинства EZPDO

- Простота
- Необходимый базовый функционал, достаточный для реализации простых проектов

Недостатки EZPDO

- Слабо конфигурируемый
  - Некоторые сложности с подключением к нескольким источникам данных – persistance manager, т.к. является синглтоном
  - Жесткая завязка на имена полей в таблице

lmbActiveRecord

- Пакет для LIMB - PHP Web Application Framework (http://limb-project.com)
- Попытка взять лучшее от ActiveRecord RubyOnRails
- LGPL

Почему своя разработка

- Исторические причины:
  - Готовая подсистема DBAL (DBDrivers, TableGateway, QueryObject)
  - Наличие базовой реализации ActiveRecord без поддержки взаимоотношений объектов
- Желание сделать из этого удобный и легко воспринимаемый инструмент

Функциональность ActiveRecord

- Отношения между классами
  - Один-к-одному
  - Один-ко-многим
  - Много-ко-многим
  - Достаточно удобный механизм для работы с коллекциями связаный объектов (итерирование, поиск, сортировка)

Один к одному

```
class Person extends lmbActiveRecord

{
  protected $_has_one = array('social_security' =>
  array('field' => 'social_security_id',
        'class' => 'SocialSecurity'));

}

class SocialSecurity extends lmbActiveRecord

{
  protected $_belongs_to = array('person' =>
      array('field' => 'social_security_id',
            'class' => 'Person'));

}
```

Один ко многим

```
class Course extends lmbActiveRecord

{
  protected $_has_many = array('lectures' =>
    array('field' => 'course_id',
          'class' => 'Lecture'));

}

class Lecture extends lmbActiveRecord

{
  protected $_many_belongs_to = array('course' =>
    array('field' => 'course_id',
          'class' => 'Course'));

}
```

Много ко многим

```
class Group extends lmbActiveRecord

{

  protected $_db_table_name = 'user_group';
  protected $_has_many_to_many = array('users' =>
   array('field' => 'group_id',
         'foreign_field' => 'user_id',
         'table' => 'user2group',
         'class' => 'User'));

}
```

Пример работы с объектами

```
$course = new Course();
$course->setTitle('Super course');

$l1 = new Lecture();
$l1->setTitle('Physics');
$l2 = new Lecture();
$l2->setTitle('Math');

$course->addToLectures($l1);
$course->addToLectures($l2);

//или

//$course->setLectures(array($l1, $l2));

$course->save();
```

Работа с коллекциями

```
$course = lmbActiveRecord :: findById('Course',
$course_id);

$lectures = $course->getLectures();

echo $lectures->getCount();

$lectures->sort(array('title' => 'asc'));

$lectures->paginate(10, 20);

foreach($lectures as $lecture)

  echo $lecture->getTitle() . "\n";
```

Дополнительные возможности

- Поддержка наследования.
- Отложенная загрузка некоторых атрибутов. Lazy Attributes
- Поддержка ValueObjects
- Прозрачная работа с объектами в шаблонной подсистеме
- Валидация данных

Value objects

```
class Lesson extends lmbActiveRecord

{

  protected $_composed_of = array(

    'date_start' => array('field' => 'date_start',
                          'class' => 'lmbDate',
                          'getter' => 'getStamp'),

    'date_end' =>   array('field' => 'date_start',
                          'class' => 'lmbDate',
                          'getter' => 'getStamp'));

}
```

Валидация при сохранении

```
class Lesson extends lmbActiveRecord

{
  function _createValidator()
  {
    $validator = new lmbValidator();
    $validator->addRequiredRule('title');
    $validator->addRequiredObjectRule('topic',
  'Topic');

    $validator->addRule(
       new ValidLessonPeriodRule('date_start',
                                 'date_end', $this));

    return $validator;
  }

}
```

Базовая поддержка событийной модели

- ON_BEFORE_SAVE, ON_AFTER_SAVE
- ON_BEFORE_UPDATE, ON_UPDATE
- ON_AFTER_UPDATE
- ON_BEFORE_CREATE
- ON_CREATE
- ON_AFTER_CREATE
- ON_BEFORE_DESTROY, ON_AFTER_DESTROY

контакты

- Команда разработчиков LIMB всегда рада пообщаться на интересные темы
- http://limb-project.com
- http://agiledev.ru

вопросы?