Introduction to Doctrine ORM Integration with Zend Framework

Kevin Gustavson – CellTrak Technologies, Inc.

CellTrak.com

January 14, 2011

Introduction

What is an ORM and why should we use one?

What is an ORM?

Definition

Object Relational Mapping is a method of abstracting object-based data models from the underlying database structure.

Definition

Doctrine provides transparent persistence for PHP objects.

What else does Doctrine do?

- Supports Data Models
- Supports tree-structured data
- Caching (memcached, sqlite, APC, etc.)
- Model behaviors (sluggable, timestampable, nested set, i18n, search, etc.)
- Model Inheritance
  - Mapped Superclasses
  - Single Table Inheritance
  - Class Table Inheritance
- Hooks, Listeners, and Events
- Database migrations

Doctrine 1 vs Doctrine 2

Doctrine 1.x is based on the Active Record pattern:

```
$person = new Person;
$person->firstname = 'Jane';
$person->lastname = 'Doe';
$person->save();
```

Doctrine 1 vs Doctrine 2

The Active Record Pattern:

Strengths:

- It's tightly coupled to the database schema.
- It's easy to understand and implement.

Weaknesses:

- It's tightly coupled to the database schema.
- Business logic must sit directly in the Model class.
- It's harder to refactor.

Doctrine 1 vs Doctrine 2

Doctrine 2.0 uses the Data Mapper pattern:

```
$employee = new \Entities\Employee;
$employee->setName('Jane Doe');
$employee->setDepartment('development');

$em->persist($employee);
$em->flush();
```

Doctrine 1 vs Doctrine 2

Why use separate persist() and flush() calls?
Doctrine 2 uses the UnitOfWork pattern to keep a list of all writes using persist(). Then calling flush() uses a transaction to write them to the database.

Table of Contents
Initialize the Entity Manager
Schemas, Entities, and Models
DQL

Lifecycle Example

- Connect to the database
- Create a schema
- Doctrine Entities
- Simple SQL queries
- Doctrine Query Language (DQL)
- Joins
- Transactions
- Hydration
- Migrations

Create a connection

Introducing the Entity Manager

- Central access point to all ORM functionality
- Employs a transactional write-behind strategy
- Internally uses UnitOfWork to track objects

Create a connection

Autoloading the classes using Doctrine ClassLoader

```
$classLoader = new \Doctrine\Common\ClassLoader(
        'Doctrine\Common',
        'lib/vendor/doctrine-common/lib'
);
$classLoader->register();

$classLoader = new \Doctrine\Common\ClassLoader(
        'Doctrine\DBAL',
        'lib/vendor/doctrine-dbal/lib'
);
$classLoader->register();

$classLoader = new \Doctrine\Common\ClassLoader('Doctrine\ORM', 'lib/');
$classLoader->register();

// Required if you don't use Doctrine2 in combination with Symphony2
$classLoader = new \Doctrine\Common\ClassLoader('Symfony', 'lib/vendor/');
$classLoader->register();
```

Create a connection

Create caches

```
if ($applicationMode == "development") {
    $cache = new \Doctrine\Common\Cache\ArrayCache;
} else {
    $cache = new \Doctrine\Common\Cache\ApcCache;
}

$config = new Configuration;
$config->setMetadataCacheImpl($cache);   // Metadata cache
$config->setQueryCacheImpl($cache);      // Query cache
```

Attach entities

```
$driverImpl = $config->newDefaultAnnotationDriver('Entities');
$config->setMetadataDriverImpl($driverImpl);
```

Create a connection

Proxy Configuration

```
$config->setProxyDir('Proxies');
$config->setProxyNamespace('Proxies');

if ($applicationMode == "development") {
    $config->setAutoGenerateProxyClasses(true);
} else {
    $config->setAutoGenerateProxyClasses(false);
}
```

Create a connection

Connection Options

```
$connectionOptions = array(
    'driver'   => 'pdo_mysql',
    'host'     => 'localhost',
    'dbname'   => 'visits_174',
    'user'     => 'dbadmin',
    'password' => 'password',
);
```

Create a connection

With Metadata, Cache, Proxies, and connectionOptions set-up, we can now create the Entity Manager instance.

```
$em = EntityManager::create(
        connectionOptions,
        $config
);
```

Schemas, Entities, and Models

Traditional Schema:

```
CREATE TABLE `visits` (
  `VisitID` int(11) NOT NULL auto_increment,
  `sch_auto` varchar(64) NOT NULL,
  `cp_key` varchar(32) NOT NULL default '0',
  `NurseID` varchar(32) NOT NULL default '0',
  `PatientID` varchar(32) NOT NULL default '0',
  `start` int(11) NOT NULL default '0',
  `stop` int(11) NOT NULL default '0',
  `total_time` int(11) NOT NULL default '0',
  `PatientFirstName` varchar(32) default NULL,
  `PatientLastName` varchar(32) default NULL,
  `NurseFirstName` varchar(32) default NULL,
  `NurseLastName` varchar(32) default NULL,
  `status` varchar(32) default NULL,
  `exporterror` varchar(64) default NULL,
  `loc_id` int(11) NOT NULL,
  `Miles` varchar(64) default 'PENDING',
  `srmileage` float(11,2) default NULL,
  `odmileage` float(11,2) default NULL,
  `adjmileage` float(11,2) default NULL,
  `mileagetoexport` tinyint(2) NOT NULL default '1',
  `traveltime` varchar(64) default 'PENDING',
  `srtraveltime` int(5) default NULL,
  `adjtraveltime` int(5) default NULL,
  `odtraveltime` int(5) default NULL,
  `exporttime` tinyint(1) default '1',
```

Schemas, Entities, and Models

YAML Schema:

```
Visits:
  type: entity
  table: visits
  fields:
    visitid:
      id: true
      type: integer
      unsigned: false
      nullable: false
      column: VisitID
      generator:
        strategy: IDENTITY
    schAuto:
      type: string
      length: 64
      fixed: false
      nullable: false
      column: sch_auto
    cpKey:
      type: string
      length: 32
      fixed: false
      nullable: false
      column: cp_key
    nurseid:
      type: string
```

Schemas, Entities, and Models

XML Schema:

```
<?xml version="1.0" encoding="utf-8"?>
<doctrine-mapping xmlns="http://doctrine-project.org/schemas/orm/doctrine-mapping" xmln
  <entity name="Visits" table="visits">
    <change-tracking-policy>DEFERRED_IMPLICIT</change-tracking-policy>
    <id name="visitid" type="integer" column="VisitID">
      <generator strategy="IDENTITY" />
    </id>
    <field name="schAuto" type="string" column="sch_auto" length="64" />
    <field name="cpKey" type="string" column="cp_key" length="32" />
    <field name="nurseid" type="string" column="NurseID" length="32" />
    <field name="patientid" type="string" column="PatientID" length="32" />
    <field name="start" type="integer" column="start" />
    <field name="stop" type="integer" column="stop" />
    <field name="totalTime" type="integer" column="total_time" />
    <field name="patientfirstname" type="string" column="PatientFirstName" length="32" />
    <field name="patientlastname" type="string" column="PatientLastName" length="32" />
    <field name="nursefirstname" type="string" column="NurseFirstName" length="32" />
    <field name="nurselastname" type="string" column="NurseLastName" length="32" />
    <field name="status" type="string" column="status" length="32" />
    <field name="exporterror" type="string" column="exporterror" length="64" />
    <field name="locId" type="integer" column="loc_id" />
    <field name="miles" type="string" column="Miles" length="64" />
    <field name="srmileage" type="float" column="srmileage" />
    <field name="odmileage" type="float" column="odmileage" />
    <field name="adjmileage" type="float" column="adjmileage" />
    <field name="mileagetoexport" type="boolean" column="mileagetoexport" />
```

Schemas, Entities, and Models

Annotated PHP Schema:

```
/**
 * Visits
 *
 * @Table(name="visits")
 * @Entity
 */
class Visits
{
    /**
     * @var integer $visitid
     *
     * @Column(name="VisitID", type="integer", nullable=false)
     * @Id
     * @GeneratedValue(strategy="IDENTITY")
     */
    private $visitid;

    /**
     * @var string $schAuto
     *
     * @Column(name="sch_auto", type="string", length=64, nullable=false)
     */
    private $schAuto;

    /**
     * @var string $cpKey
```

Schemas, Entities, and Models

In Doctrine 2.0, Models are called Entities. Entities are originally generated from the the mapping files but once created can be modified.
Business logic can either go into a Superclass, the Entity itself, or into the Entity's proxy file.
We'll need to decide where best to put our business logic.

Doctrine Query Language

DQL is an object querying language that supports SQL constructs.

- GROUP BY
- HAVING
- Subselects
- Fetch-Joins
- COUNT(), MAX(), etc.
- and many more...

Doctrine Query Language

DQL can be written in a number of ways. Here's a straight-forward example.

```
$query = $em->createQuery(
            'SELECT u '
          . 'FROM MyProject\Model\User u '
          . 'WHERE u.age > 20'
);
$users = $query->execute();
```

Doctrine Query Language

Consider our earlier code sample:

```
$employee = new \Entities\Employee;
$employee->setName('Jane Doe');
$employee->setDepartment('development');

$em->persist($employee);
$em->flush();
```

Using Query Builder

```
$qb = new QueryBuilder;
$qb->select('u')
    ->from('User', 'u')
    ->where('u.id = ?1')
    ->orderBy('u.name', 'ASC');

/* JOIN */
$qb->select('u')
    ->from('User', 'u')
    ->innerJoin('u.Group', 'g',
        'ON', 'u.group_id = g.id AND g.name = ?1'
        );

$q = $qb->getQuery();
$users = $q->execute();
```

Transactions

Why transactions?

- Performance
- Failure Recovery
- Database Integrity

Doctrine 1 Transactions

```
$conn = Doctrine_Manager::connection();

try {
    $conn->beginTransaction();

    $this->_performCreatesOrUpdates($conn);
    $this->_performDeletes($conn);

    $conn->commit();
} catch(Doctrine_Exception $e) {
    $conn->rollback();
}

$this->clearAll();
```

Doctrine 2 Transactions

Yes, Doctrine 2.0 transactions are implict. Remember this?

```
$employee = new \Entities\Employee;
$employee->setName('Jane Doe');
$employee->setDepartment('development');

$em->persist($employee);
$em->flush();
```

The flush() call performs "COMMIT TRANSACTION".

Hydration

```
// Prepare Query
$q = $em->createQuery('SELECT u from User u');

// Return an object DEFAULT
$users = $q->getResult(Query::HYDRATE_OBJECT);

// Return an array of values
$users = $q->getResult(Query::HYDRATE_ARRAY);
```

Other Hydration methods are:

- HYDRATE_SCALAR
- HYDRATE_SINGLE_SCALAR
- custom (define your own)

Migrations

Sample Doctrine 2 Migration:

```
namespace DoctrineMigrations;

use Doctrine\DBAL\Migrations\AbstractMigration,
    Doctrine\DBAL\Schema\Schema;

class Version20100416130401 extends AbstractMigration
{
    public function up(Schema $schema)
    {
        $table = $schema->createTable('users');
        $table->addColumn('username', 'string');
    }

    public function down(Schema $schema)
    {
        $schema->dropTable('users');
    }
}
```

Zend Framework Integration

Ways to integration Zend Framework 1.x and Doctrine 2.0

- Minimal (Create a bootstrap and call from init.)
- Bisna Integration library
- Custom integration

Minimal Zend Framework - Doctrine Integration

We already covered most of what you need to know to do a minimal bootstrap. Create a bootstrap file with the following steps:

- Load your classes or set-up autoloading
- Set-up a cache
- Set-up a proxy
- Set-up a repository (optional)
- Set-up your connection options
- Set-up a SQL logger (optional)
- Create an Entity Manager instance

You can repeat these steps for any additional connections you need. Call the bootstrap file from application/Bootstrap.php.

Zend Framework - Doctrine Integration using Bisna

Included with Bisna integration:

- Doctrine configured using Zend_Config
- Doctrine commandline added to the zf command
- It puts everything in one place.
- Bisna was created by Guilherme Blanco, a Doctrine core developer.
- A good sample of Zend Framework integration with Bisna is a project called NOLAsnowball.
  https://github.com/ralphschindler/NOLASnowball/

Custom Zend Framework - Doctrine Integration

Roll your own:
Lot's of people are talking about integrating Doctrine 2 with Zend Framework. If we really want to go this route, we need to be reading the forums and understanding the details.

Any Questions?

- Doctrine Website
  - Download, Documentation, API, and Cookbook
  - http://www.doctrine-project.org/
- Zend Framework 1 + Doctrine 2 webinar
  - http://www.zend.com/en/resources/webinars/
  - http://www.slideshare.net/ralphschindler/zend-framework-1-doctrine-2-6177485
- Zend Framework 2.0 and Doctrine2 wiki
  http://framework.zend.com/wiki/display/ZFDEV2/Doctrine2

More resources:

Books:

- "Patterns of Enterprise Application Architecture" by Martin Fowler
- "Refactoring" by Martin Fowler

Blogs:

- Doctrine Blog – http://www.doctrine-project.org/blog/
- Matthew Weier O'Phinney – http://weierophinney.net/
- Guilherme Blanco – http://blog.bisna.com/
- Jonathan Wage – http://www.jwage.com/ (Appears to be down at the moment)