

Object-Relational Mapping

Simplifying Relational Databases Using Object Orientation



An Overview

- In this class we handled queries by hand
 - Generally simple interactions with database
 - Did not do much "work" on data outside of database
- This type of database interaction is not always the way it works
- What if the database held large objects?
- What if we did a lot of work on those objects?



What is it?

- Unsurprisingly it maps relational databases to objects
- A table or group of tables might represent an entity
- A class is made that also represents the entity
- Info from the DB is loaded into an instance of the class
- Changes made to the class are then reflected in the database



When to Use It?

- If we have big objects that we work on a lot
 - And the database schema is 'simple'
 - And we have the resources to handle some overhead
 - ORM might be a good solution
- Not always a good option
 - Takes away some flexibility
 - Can add some additional complexity



A Simple ORM

- RedBeanPHP is a very simple ORM library for PHP
- No configuration required
 - But this means little customizability is possible
- When new objects are requested
 - If a table exists it adds a new row
 - If no table exists it makes a new table
- When an object property is written to
 - If the attribute exists it is modified
 - If not the attribute is added to the table



What Does This Look Like

 All ORMs are going to look very different in the way the function, this is just one example



Making Some Entities

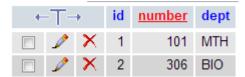
```
2 //Gets "user", "video" and "time" in post
 3 ini set('display errors', 'On');
 4 require ('rb.php');
 5 R::setup('mysgl:host=niddb.cws.oregonstate.edu;dbname=wolfordj-db', 'wolford
   j-db', 'mypassword');
 6 //Make a student
 7 $example student = R::dispense('student');
 8 $example student->name = 'John';
 9 $example student->age = '19';
10 //Make another student
11 $example student2 = R::dispense('student');
12 $example student2->name = 'Anna';
13 $example student2->age = '20';
14 //Make some classes
15 $class1 = R::dispense('class');
16 $class1->number = 101;
17 $class1->dept = "MTH";
18 $class2 = R::dispense('class');
19 $class2->number = 306:
20 $class2->dept = "BIO";
21 R::store($example student);
22 R::store($example student2);
23 R::store($class1);
24 R::store($class2);
25 R::close();
26 ?>
```



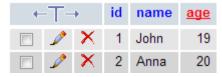
Added Tables and Entries

Table			Act	ion			Records	Туре	Collation	Size	Overhead
class			1	3-6		×	2	InnoDB	utf8_unicode_ci	16.0 KiB	-
student			1	3-6		×	2	InnoDB	utf8_unicode_ci	16.0 KiB	-
2 table(s)	Sum						4	MyISAM	latin1 swedish ci	32.0 KiB	0 B

class



student



And Relationships

Many-to-one

```
8 $classes = R::find("class","dept=?",array('BIO'));
9 foreach($classes as &$c){
10    echo $c->number;
11    $c->ownEnrolled = R::find("student","Name=?",array("Anna"));
12    R::store($c);
13 }
14 R::close();
```

Updated Schema

←T→			id	name	age	class_id	$\leftarrow T \rightarrow$		+	id	number	dept
		×	1	John	19	NULL		1	×	1	101	MTH
		X	2	Anna	20	2		<i></i>	×	2	306	BIO



More relationships

Many-To-Many

```
8 $classes = R::findAll("class");
9 foreach($classes as &$c){
10    echo $c->number;
11    $c->sharedEnrolled = R::find("student", "Name=?", array("Anna"));
12    R::store($c);
13 }
14 R::close();
```

class

student

class_student

←T→		id	number	dept	←T→		id	name	age	+	←T→		id	student_id	class_id		
	<i>></i>	×	1	101	MTH		1	×	1	John	19		1	×	1	2	1
	1	×	2	306	BIO		1	×	2	Anna	20		1	×	2	2	2



This Is Super Great! Why Not Use it for All The Things!?

- Some limitations of this ORM library
 - What if you want to store attributes with a relationship? (e.g. Enrollment date for a class)
 - You can't. You would need to make enrollment a entity and handle relationships manually
 - Multiple instances of the same object confuse it
 - A department may have two people objects
 - \$dept->manager;\$dept->pointOfContact;
 - Will save them as people, but wont know they are people when they are loaded
 - //Code to load as a person \$dept->fetchAs('person')->manager;



ORMs In Review

- They make some applications much simpler
- Work well with object oriented languages
- If you play by their rules, things go well
- If you go outside their rules things get complicated, quickly
- Often times more limited than creating schemas by hand