MORE CAKE

A deeper overview of the NIGRAHA architecture

★ ★ ★ ★ ★ ★ ★ ★

Anant Narayanan
Malaviya National Institute of Technology

January 12, 2008

What do we know so far?

- Web applications are the best way to deploy institute-wide systems.
- Conventional PHP development is good for small teams, but makes the code inextensible and unmanageable after a point.
- Model-View-Controller based design is especially suited for projects involving several people, each working on different modules.

Revision

- No more hand-crafted SQL (almost!)
- Clean separation of web designing and development.
- Everyone can work on their own portion without worrying about how it will affect others.
- MVC is self-documenting, it is easier to figure out what to do. Especially useful for code that is “handed-over” from one group to another.

Thinking in MVC

- Separate three aspects of web development, which can be developed simultaneously by different teams:
  - Data Management and interface with the Data
  - Application logic, processing and flow control
  - Presenting to the data to the user, and collecting it

Models

- Classes that represent tables in your database
- Exports methods to query the data, extract, modify and delete
- Models perform the exclusive function of providing a consistent interface to the database
- No other portion should directly query the database, they should use **only** data structures provided by the model

Controllers

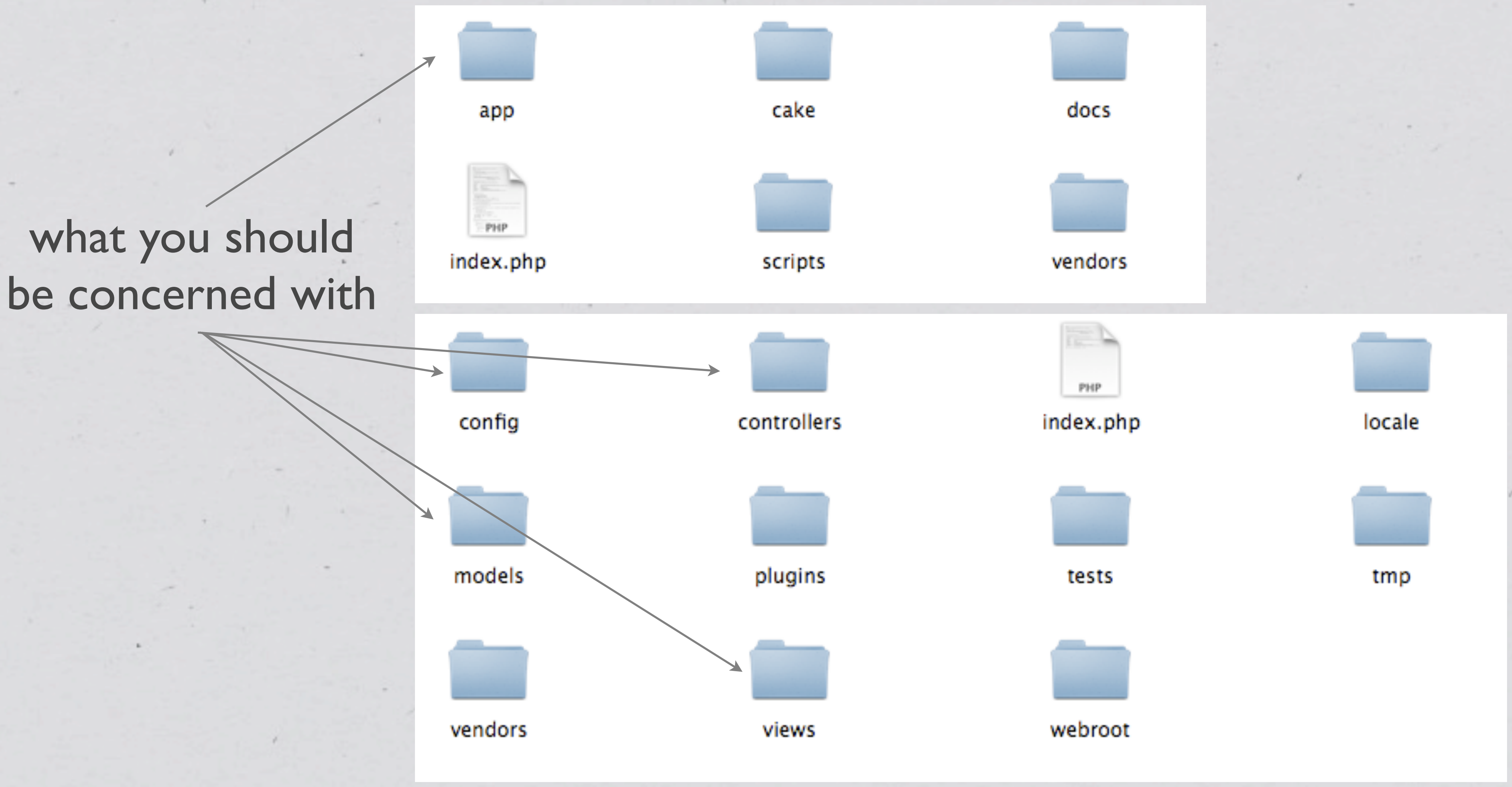
- All your application logic, flow control, and processing go here
- Calls methods from the Model to extract data structures, compare, modify and possibly write back to the database
- Exports a set of variables to the next component: ‘View’ to display the user
- Imports a set of variables from ‘Views’ after a form has been submitted to be processed

Views

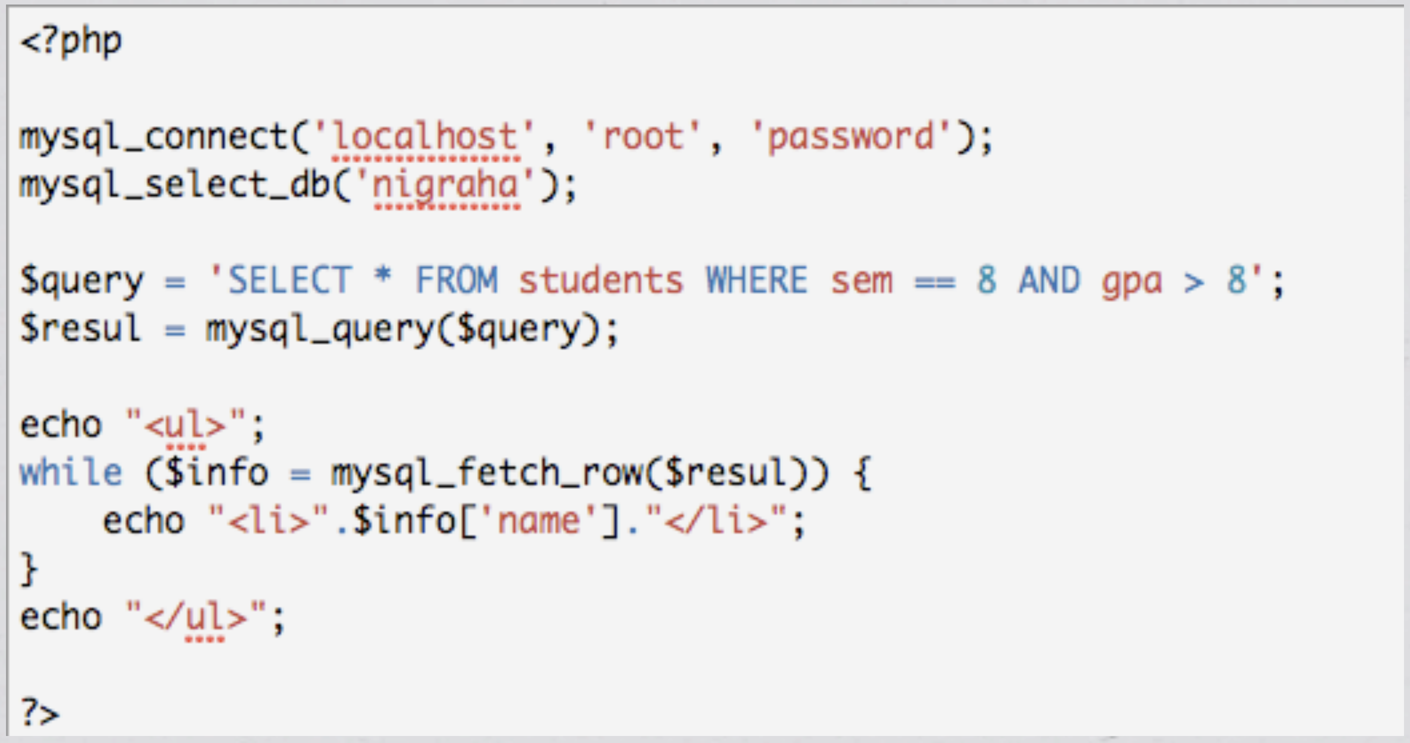
- Imports a set of variables from a controller to display
- Mostly HTML with ‘echo $variable’ whereever required
- NO programming is to be done here
- EXCEPT when using AJAX to provide a more interactive user interface
- Export form values to a controller for processing

How the pieces fit

- We'll take a look at the MVC architecture as it pertains to CakePHP, the web application framework used by NIGRAHA



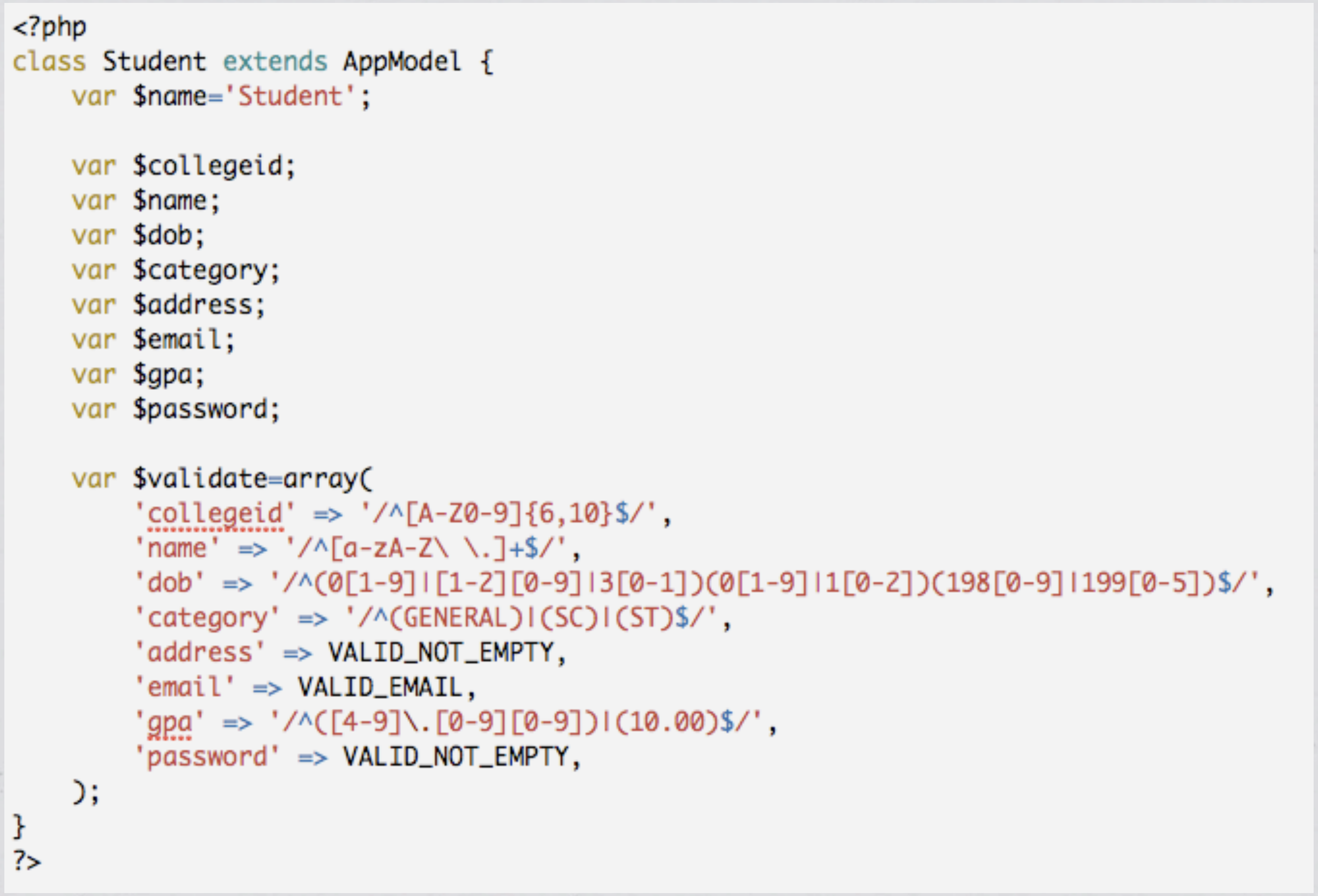
What you would usually do



```php
<?php

mysql_connect('localhost', 'root', 'password');
mysql_select_db('nigraha');

$query = 'SELECT * FROM students WHERE sem == 8 AND gpa > 8';
$resul = mysql_query($query);

echo "<ul>";
while ($info = mysql_fetch_row($resul)) {
    echo "<li>".$info['name']."</li>";
}
echo "</ul>";

?>
```

What you should do

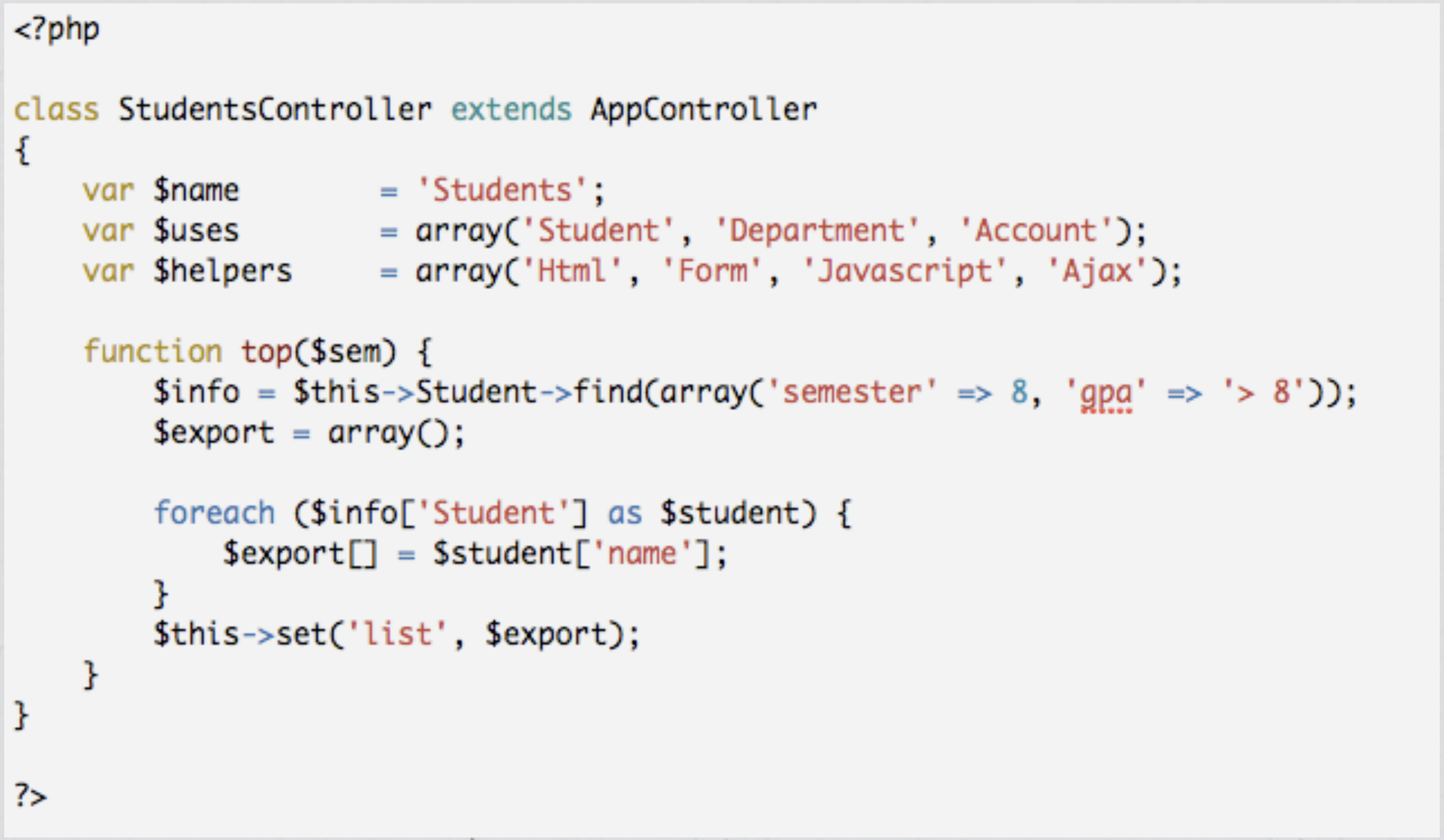
* Create a model: app/models/student.php



```php
<?php
class Student extends AppModel {
    var $name='Student';

    var $collegeid;
    var $name;
    var $dob;
    var $category;
    var $address;
    var $email;
    var $gpa;
    var $password;

    var $validate=array(
        'collegeid' => '/^[A-Z0-9]{6,10}$/',
        'name' => '/^[a-zA-Z\ \.]+$/',
        'dob' => '/^(0[1-9]|[1-2][0-9]|3[0-1])(0[1-9]|1[0-2])(198[0-9]|199[0-5])$/',
        'category' => '/^(GENERAL)|(SC)|(ST)$/',
        'address' => VALID_NOT_EMPTY,
        'email' => VALID_EMAIL,
        'gpa' => '/^([4-9]\.[0-9][0-9])|(10.00)$/',
        'password' => VALID_NOT_EMPTY,
    );
}
?>
```

What you should do

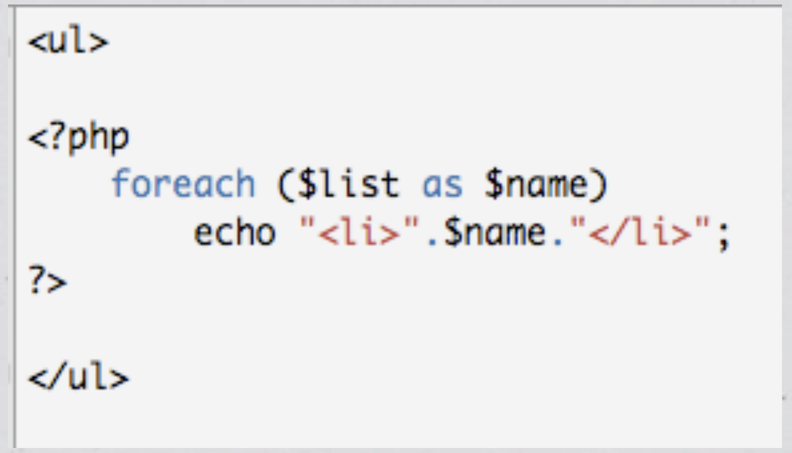
- Create a controller: app/controllers/students_controller.php



```php
<?php

class StudentsController extends AppController
{
    var $name       = 'Students';
    var $uses       = array('Student', 'Department', 'Account');
    var $helpers    = array('Html', 'Form', 'Javascript', 'Ajax');

    function top($sem) {
        $info = $this->Student->find(array('semester' => 8, 'gpa' => '> 8'));
        $export = array();

        foreach ($info['Student'] as $student) {
            $export[] = $student['name'];
        }
        $this->set('list', $export);
    }
}

?>
```

What you should do

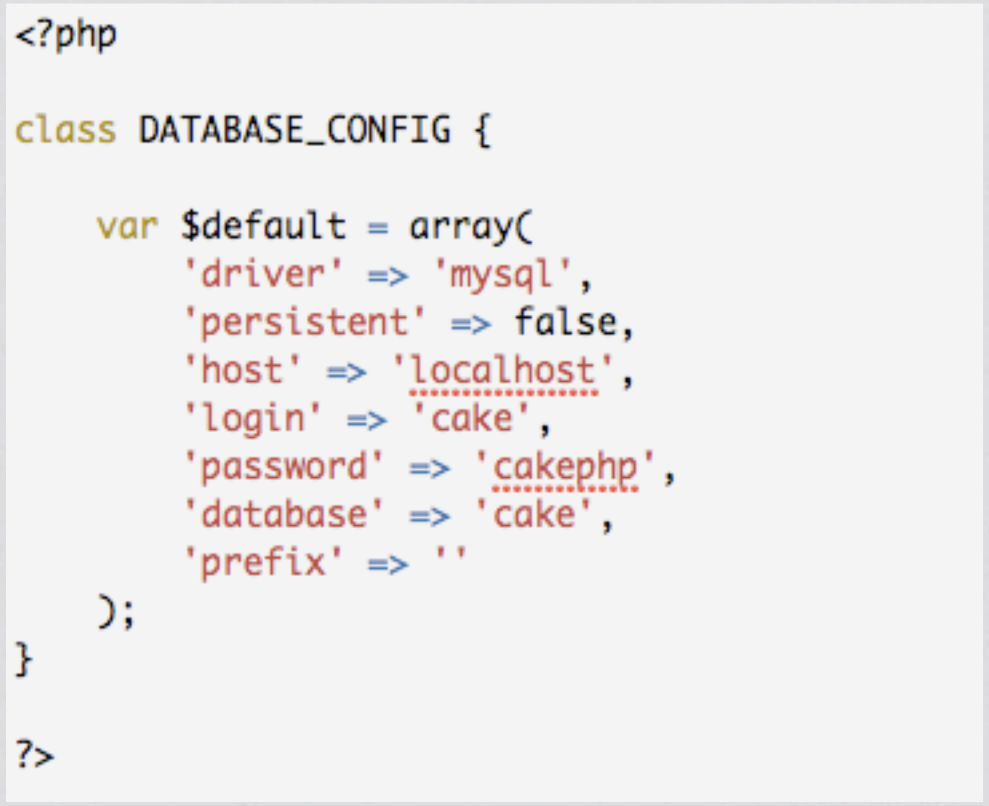
- And finally a view: app/views/students/top.ctp



```
<ul>

<?php
    foreach ($list as $name)
        echo "<li>".$name."</li>";
?>

</ul>
```

The final step

- Database configuration

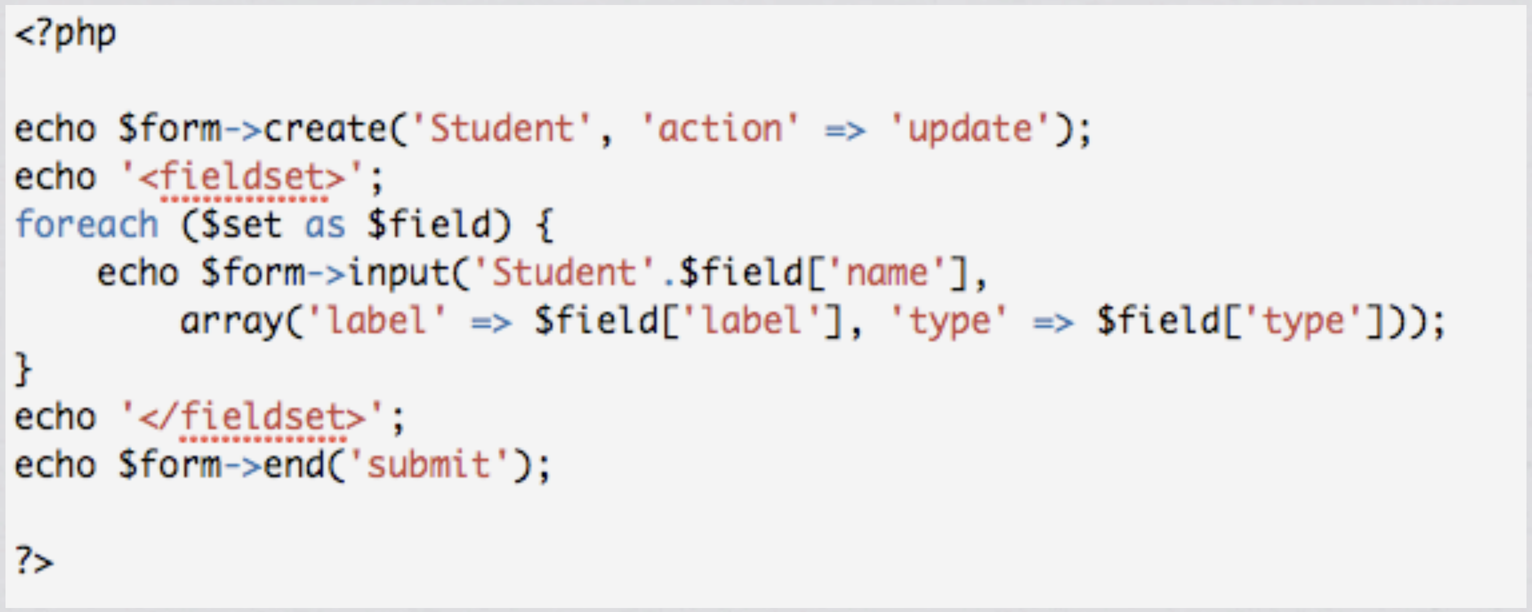


```
<?php

class DATABASE_CONFIG {

    var $default = array(
        'driver' => 'mysql',
        'persistent' => false,
        'host' => 'localhost',
        'login' => 'cake',
        'password' => 'cakephp',
        'database' => 'cake',
        'prefix' => ''
    );
}

?>
```

- Now access **http://localhost/cake/students/top/8**

Helpers

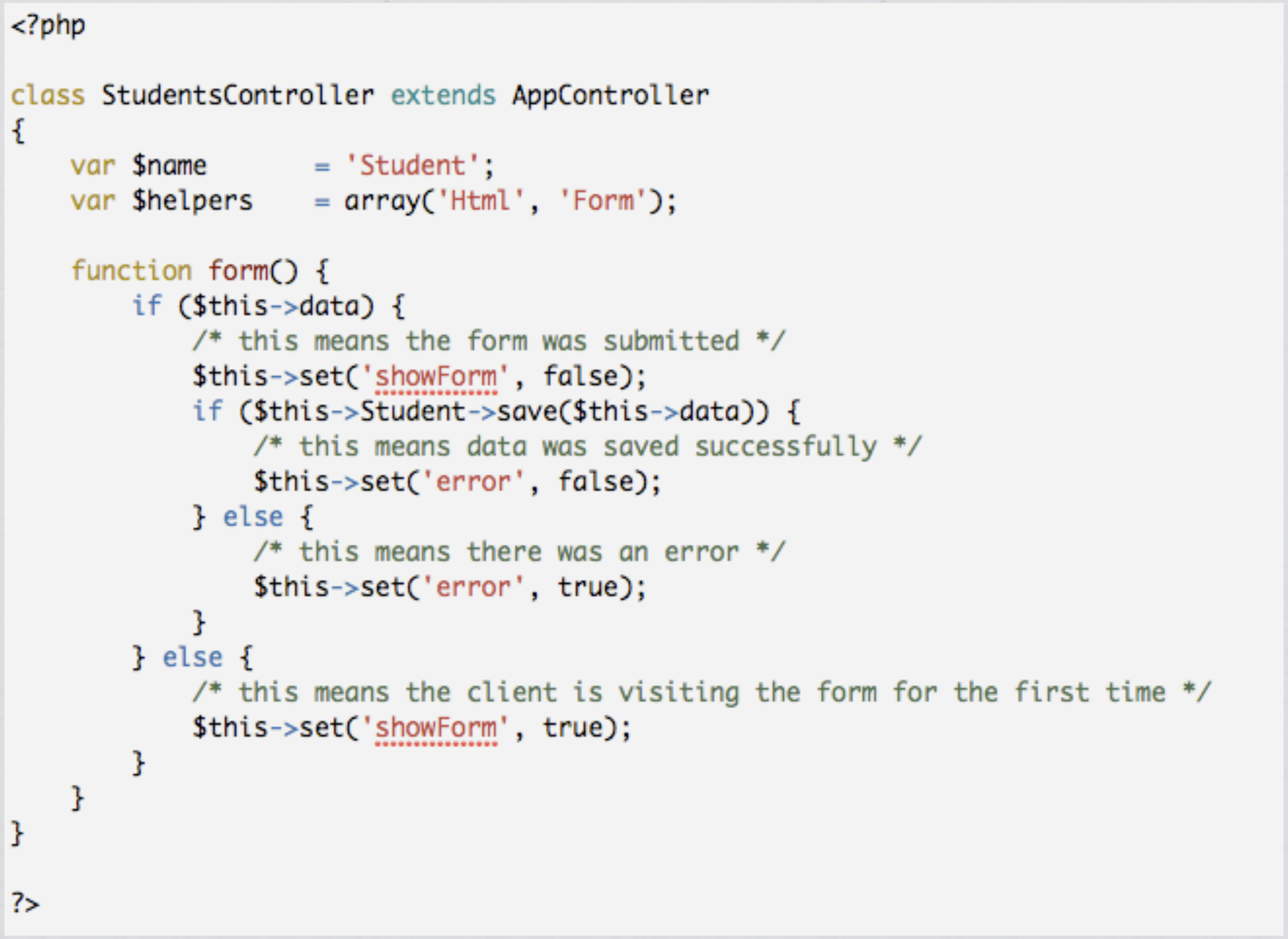
* As discussed earlier, view creation can be made more programmatic with the use of helpers



```
<?php

echo $form->create('Student', 'action' => 'update');
echo '<fieldset>';
foreach ($set as $field) {
    echo $form->input('Student'.$field['name'],
        array('label' => $field['label'], 'type' => $field['type']));
}
echo '</fieldset>';
echo $form->end('submit');

?>
```

Capturing Form Data

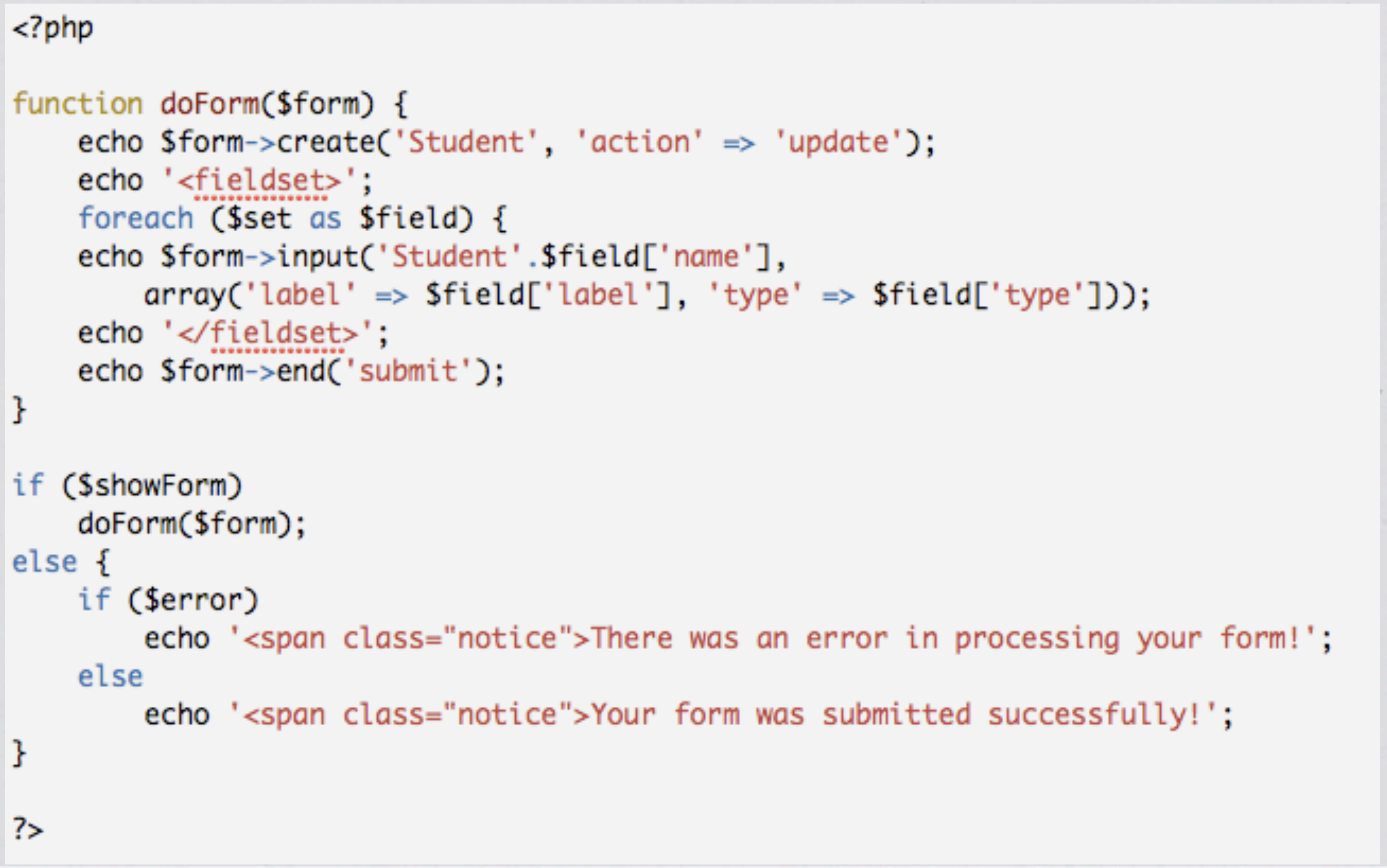
- If you used the form helper, saving the data is as simple as:



```php
<?php

class StudentsController extends AppController
{
    var $name       = 'Student';
    var $helpers    = array('Html', 'Form');

    function form() {
        if ($this->data) {
            /* this means the form was submitted */
            $this->set('showForm', false);
            if ($this->Student->save($this->data)) {
                /* this means data was saved successfully */
                $this->set('error', false);
            } else {
                /* this means there was an error */
                $this->set('error', true);
            }
        } else {
            /* this means the client is visiting the form for the first time */
            $this->set('showForm', true);
        }
    }
}

?>
```

Single view, Multiple roles

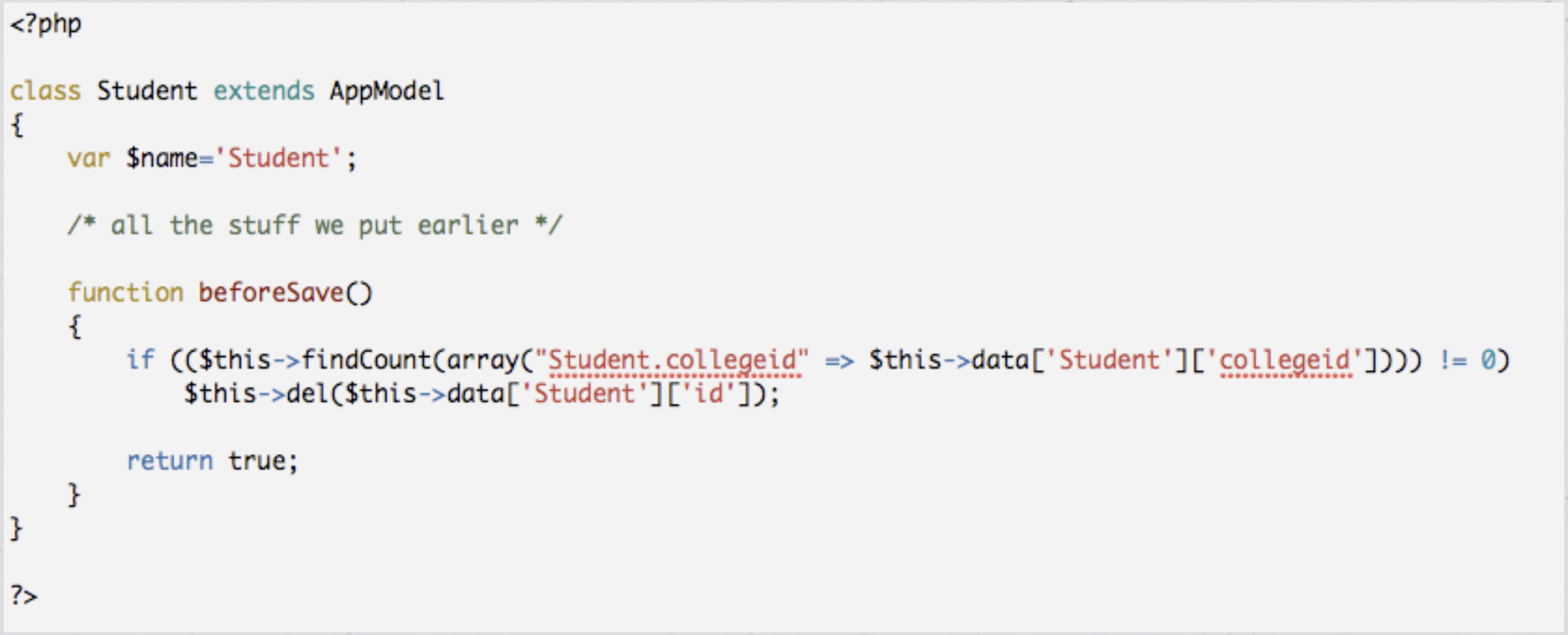
- Notice in the controller we set some variables defining state. Let’s go back to the view:



```
<?php

function doForm($form) {
    echo $form->create('Student', 'action' => 'update');
    echo '<fieldset>';
    foreach ($set as $field) {
    echo $form->input('Student'.$field['name'],
        array('label' => $field['label'], 'type' => $field['type']));
    echo '</fieldset>';
    echo $form->end('submit');
}

if ($showForm)
    doForm($form);
else {
    if ($error)
        echo '<span class="notice">There was an error in processing your form!';
    else
        echo '<span class="notice">Your form was submitted successfully!';
}

?>
```

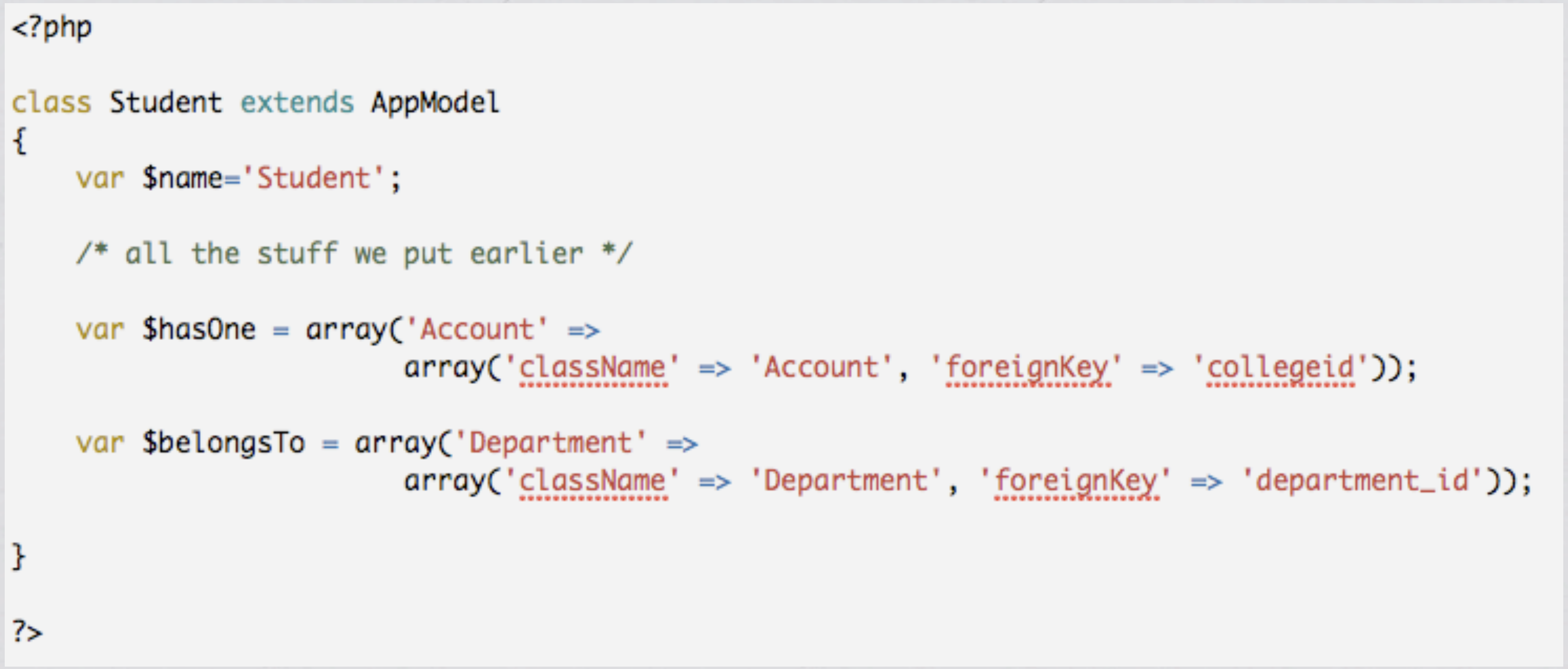
Model Hooks

- We want to call save(), but it must update the record, not create a new one if the student already exists:



```php
<?php

class Student extends AppModel
{
    var $name='Student';

    /* all the stuff we put earlier */

    function beforeSave()
    {
        if (($this->findCount(array("Student.collegeid" => $this->data['Student']['collegeid'])) != 0)
            $this->del($this->data['Student']['id']);

        return true;
    }
}

?>
```

Compound Models

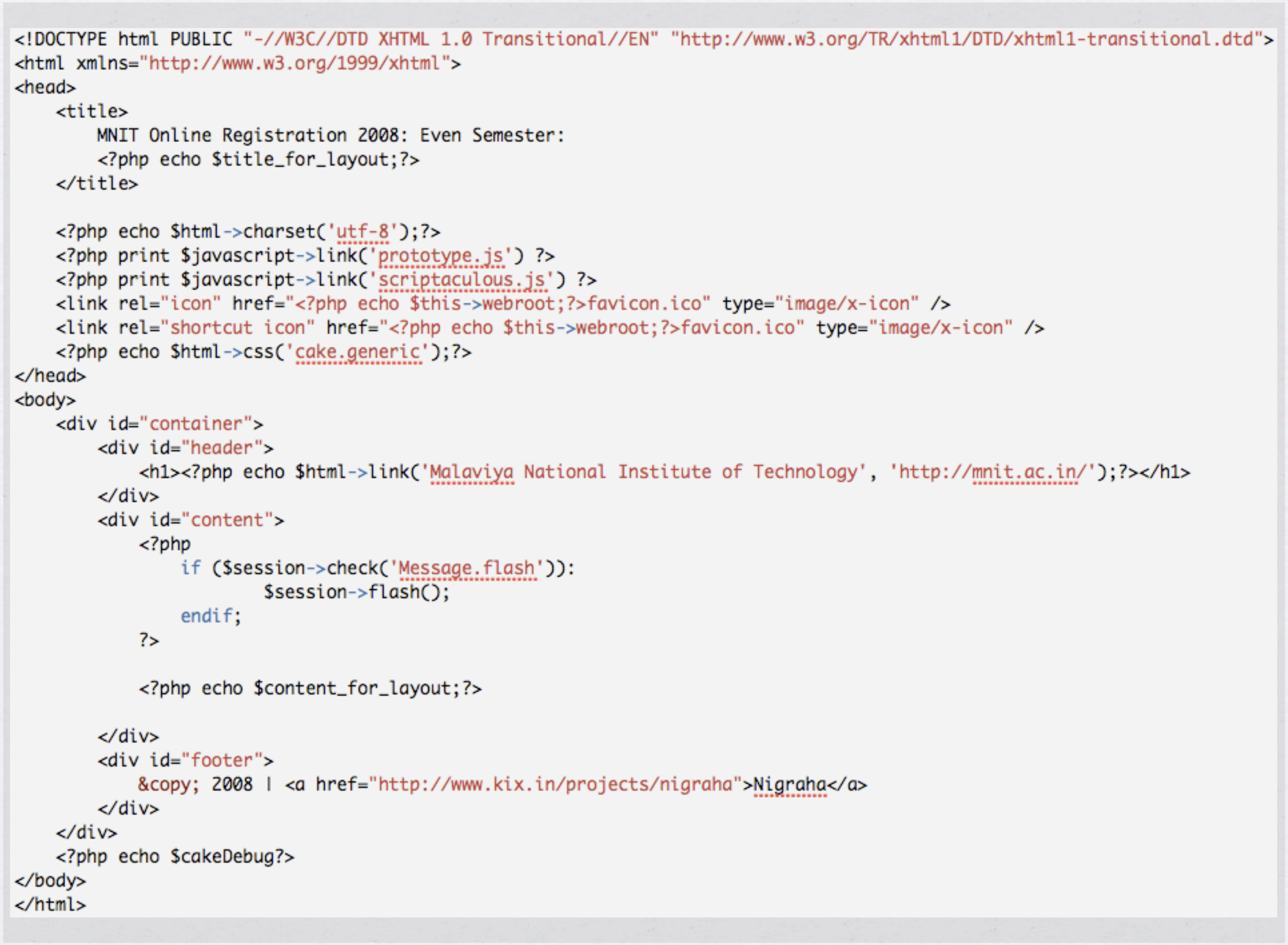


```
<?php

class Student extends AppModel
{
    var $name='Student';

    /* all the stuff we put earlier */

    var $hasOne = array('Account' =>
                        array('className' => 'Account', 'foreignKey' => 'collegeid'));

    var $belongsTo = array('Department' =>
                        array('className' => 'Department', 'foreignKey' => 'department_id'));

}

?>
```

- Compound models simply make save() and read() calls simpler
- We don’t use much of $hasMany or $hasAndBelongsToMany

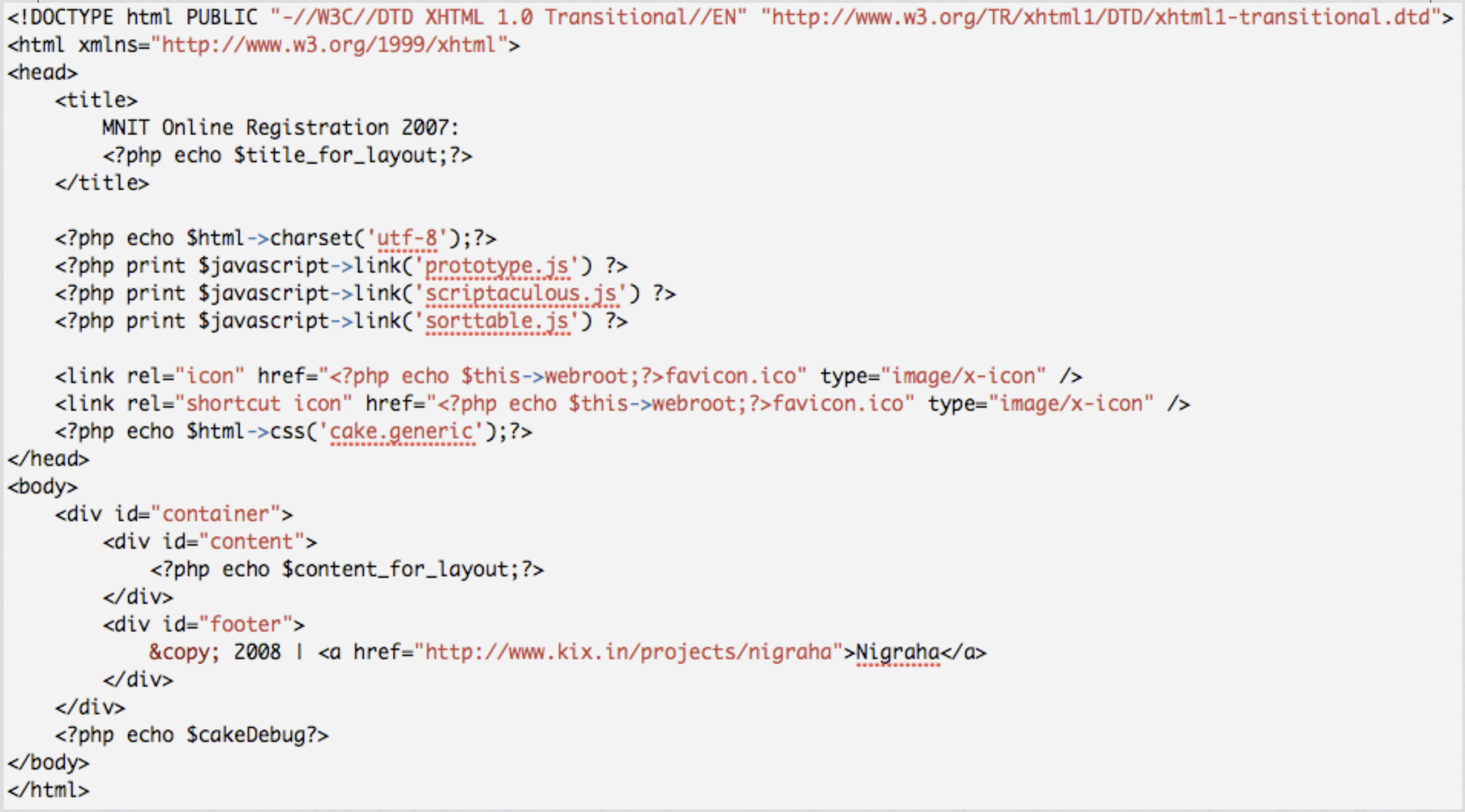
Templating

- You can specify different templates that are used to fill in the view.
- We use three templates:
  - Default: app/views/layout/default.thtml
  - For Print: app/views/layout/print.thtml
  - For AJAX: app/views/layout/plain.thtml



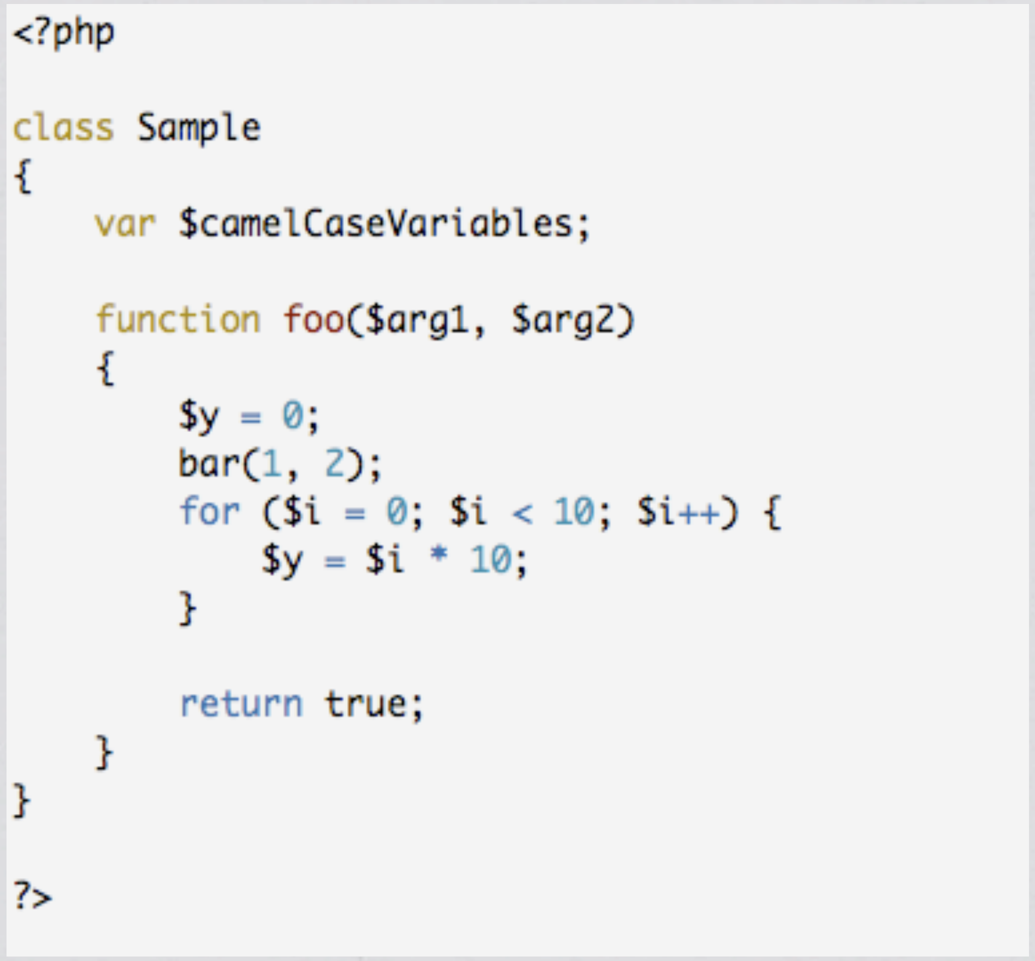
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>
        MNIT Online Registration 2008: Even Semester:
        <?php echo $title_for_layout;?>
    </title>

    <?php echo $html->charset('utf-8');?>
    <?php print $javascript->link('prototype.js') ?>
    <?php print $javascript->link('scriptaculous.js') ?>
    <link rel="icon" href="<?php echo $this->webroot;?>favicon.ico" type="image/x-icon" />
    <link rel="shortcut icon" href="<?php echo $this->webroot;?>favicon.ico" type="image/x-icon" />
    <?php echo $html->css('cake.generic');?>
</head>
<body>
    <div id="container">
        <div id="header">
            <h1><?php echo $html->link('Malaviya National Institute of Technology', 'http://mnit.ac.in/');?></h1>
        </div>
        <div id="content">
            <?php
                if ($session->check('Message.flash')):
                        $session->flash();
                endif;
            ?>

            <?php echo $content_for_layout;?>

        </div>
        <div id="footer">
            &copy; 2008 | <a href="http://www.kix.in/projects/nigraha">Nigraha</a>
        </div>
    </div>
    <?php echo $cakeDebug?>
</body>
</html>
```

Print Layout



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>
        MNIT Online Registration 2007:
        <?php echo $title_for_layout;?>
    </title>

    <?php echo $html->charset('utf-8');?>
    <?php print $javascript->link('prototype.js') ?>
    <?php print $javascript->link('scriptaculous.js') ?>
    <?php print $javascript->link('sorttable.js') ?>

    <link rel="icon" href="<?php echo $this->webroot;?>favicon.ico" type="image/x-icon" />
    <link rel="shortcut icon" href="<?php echo $this->webroot;?>favicon.ico" type="image/x-icon" />
    <?php echo $html->css('cake.generic');?>
</head>
<body>
    <div id="container">
        <div id="content">
            <?php echo $content_for_layout;?>
        </div>
        <div id="footer">
            &copy; 2008 | <a href="http://www.kix.in/projects/nigraha">Nigraha</a>
        </div>
    </div>
    <?php echo $cakeDebug?>
</body>
</html>
```

A word on Coding Standards



```php
<?php

class Sample
{
    var $camelCaseVariables;

    function foo($arg1, $arg2)
    {
        $y = 0;
        bar(1, 2);
        for ($i = 0; $i < 10; $i++) {
            $y = $i * 10;
        }

        return true;
    }
}

?>
```

Thank You!