BAKING WEB APPLICATIONS

An overview of the NIGRAHA architecture

Anant Narayanan
Malaviya National Institute of Technology

January 8, 2008

What do we know so far?

* Web applications are the best way to deploy an institute-wide systems

* With the new campus-wide network, all operations will be available to anyone in the campus, with due authentication

* To make a web application, you need three things

  * A web server. We use **Apache**.

  * A database (SQL) server. We use **mySQL**.

  * Scripting language(s). We use **PHP** on the server side and **Javascript** on the client side.

Traditional PHP Development

* Version 1 of Nigraha was built this way.

* A set of PHP scripts loosely connected to provide functionality.

* All SQL was hand-crafted.

* The PHP scripts themselves were responsible for output, and were hence filled with **echo** statements with HTML arguments.

Balance Sheet

* Pros:
    * It worked!
        * Easy to manage when only 2 people are involved in the project.
* Cons:
    * Difficult to look back and re-factor the code. New features may be added but with significant effort.
    * Will definitely not work in a group effort of more than 3.

Rethinking the Method

* Need a clean interface across all modules of NIGRAHA.
* The system must withstand the test of time, as new people will need to maintain, understand and be able to add new features for years to come.
* Web development and designing are two distinct elements: you may have programmers who are not aesthetic and designers who don’t know programming.

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

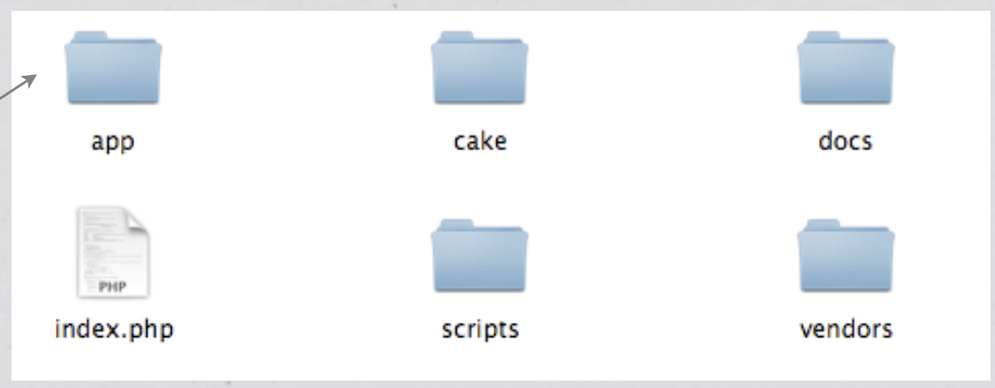
* All your application logic, flow control, and processing go here
* Calls methods from the Model to extract data structures, compare, modify and possibly write back to the database
* Exports a set of variables to the next component: ‘View’ to display the user
* Imports a set of variables from ‘Views’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

Views

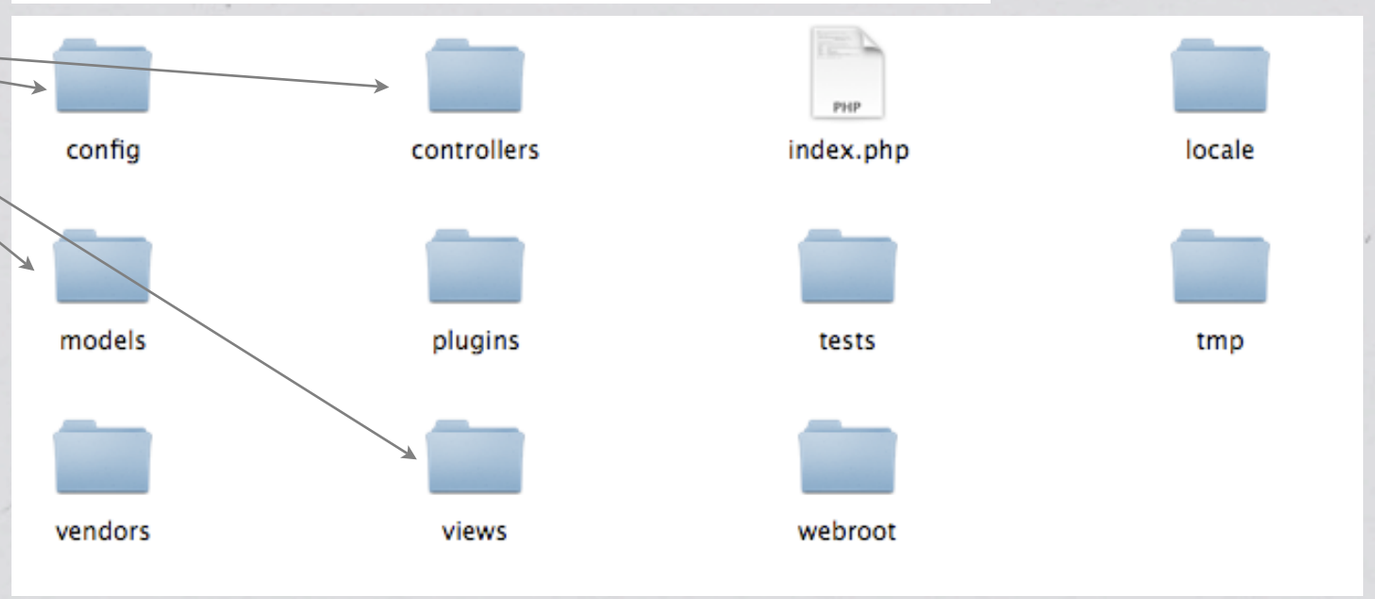
* Imports a set of variables from a controller to display
* Mostly HTML with ‘echo $variable’ whereever required
* NO programming is to be done here
* EXCEPT when using AJAX to provide a more interactive user interface
* Export form values to a controller for processing

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 should be concerned with



What you would usually do

```php
<?php

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

$query = 'SELECT * FROM sem9 WHERE 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
<?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**

You're done

* That covered all the basics, everything else is built on these concepts.
* Let's look at how the registration module fits in with what we've learnt...

Thank You!