Network and Security Research Center
Department of Computer Science and Engineering
Pennsylvania State University, University Park PA

CSE598i - Web 2.0 Security

Zend Framework Tutorial

Thomas Moyer
Spring 2010

Tutorial Source Code

The tutorial source code is available at

http://statecollege.cse.psu.edu/files/cse598i-zend-tutorial.tbz

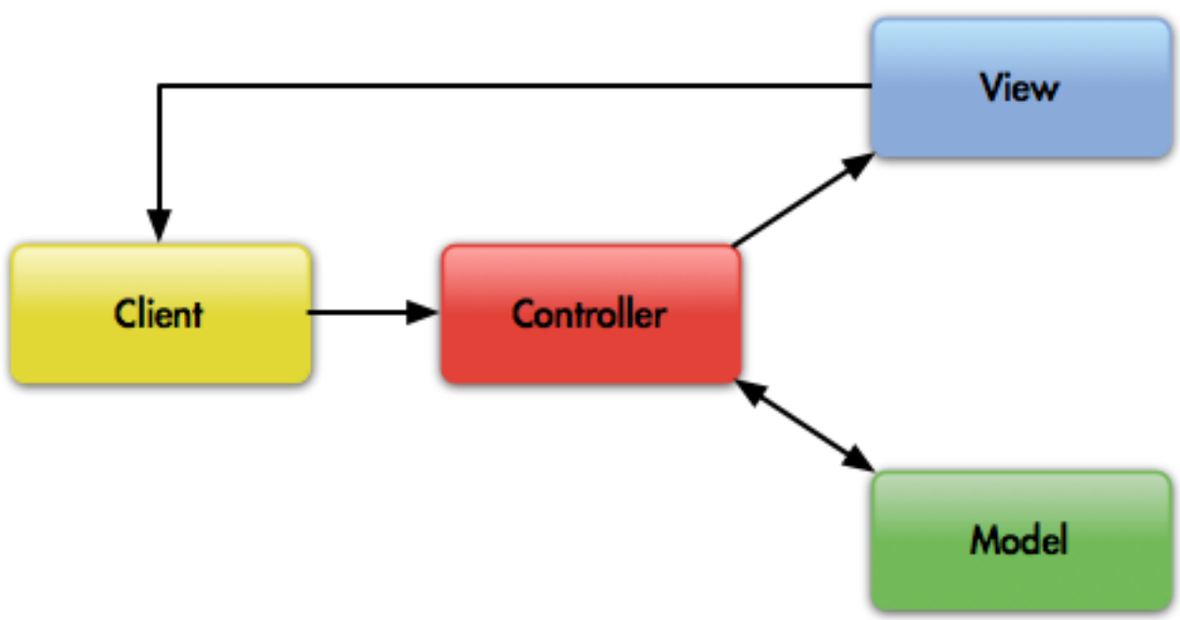
What is the Zend Framework

PENNSTATE 1855

- A *web application framework* written in PHP5
- *Loosely coupled* set of modules that perform various tasks
  - Database access (`Zend_DB`)
  - Google Data API's (`Zend_Gdata`)
  - OpenID (`Zend_OpenId`)
  - many, many others...
- Easy to implement MVC model

What is MVC?

- MVC stands for *Model-View-Controller*
  - Code is divided into three distinct groups
    - Model -- Internal representation of data, interface to backend storage (i.e. database), and "business logic"
    - View -- Code that represents the application's UI
    - Controller -- Code that generates output to populate the view using the model



Installing in your VM

PENNSTATE 1855

- Login in to your VM
- As root run

```
sudo apt-get install zend-framework
```

- This will install the PHP files for the framework
- Configure PHP to access the Zend Framework files
  - Modify */etc/php5/apache2/php.ini* (be sure to use *sudo* to edit the file)
  - Change line

    ‘`; include_path = “.:/usr/share/php”`’

    to

    ‘`include_path = “/usr/share/php”`’

Configuring PHP...

PENNSTATE 1855

- Now modify the file

  `/etc/php5/conf.d/zend-framework.ini`

- Uncomment line regarding include_path
- Restart Apache

```
sudo /etc/init.d/apache2 restart
```

Your first project...

PENNSTATE 1855

- Part of the Zend Framework is a *project management tool*
  - This tool 'zf', can handle creating new projects as well as creating the various files for your application
- Create a basic project

```
zf create project <path>
```

- This will create the basic project in '<path>' which should be someplace you can easily edit
  - I put mine in /home/tmmoyer/tutorial
- Make <path>/public readable by Apache

```
sudo chgrp www-data <path>/public
```

Zend Framework

PENNSTATE 1855

- What this creates
  - <path>/application
    - Core application code
  - <path>/library
    - Auxillary code
  - <path>/public
    - Code that is directly accessible to the web server (index.php)
  - <path>/tests
    - Directory for test code

Zend Project Skeleton

PENNSTATE 1855

- application/Bootstrap.php
  - Application bootstrap code
- application/configs
  - Configuration files
- application/controllers
  - Backend controller code
- application/models
  - Code mapping from domain data to storage data (PHP interface to DB for example)
- application/views/scripts
  - User interface code

Zend Project Skeleton

PENNSTATE 1855

- application/configs/application.ini
  - Main configuration file
- application/controllers/
  - ErrorController.php
    - Default controller called when an error occurs
  - IndexController.php
    - Default controller when no controller is specified

Setting up Apache

PENNSTATE 1855

- Apache's configuration must be tweaked to host your Zend Framework project
- Modify the file '*/etc/apache2/sites-available/default*'
- Change */var/www* to *<path>/public*
- Set *AllowOverides* to *All*
- Set *Options* to *All*
- Enable `mod_rewrite`

```
sudo a2enmod rewrite
```

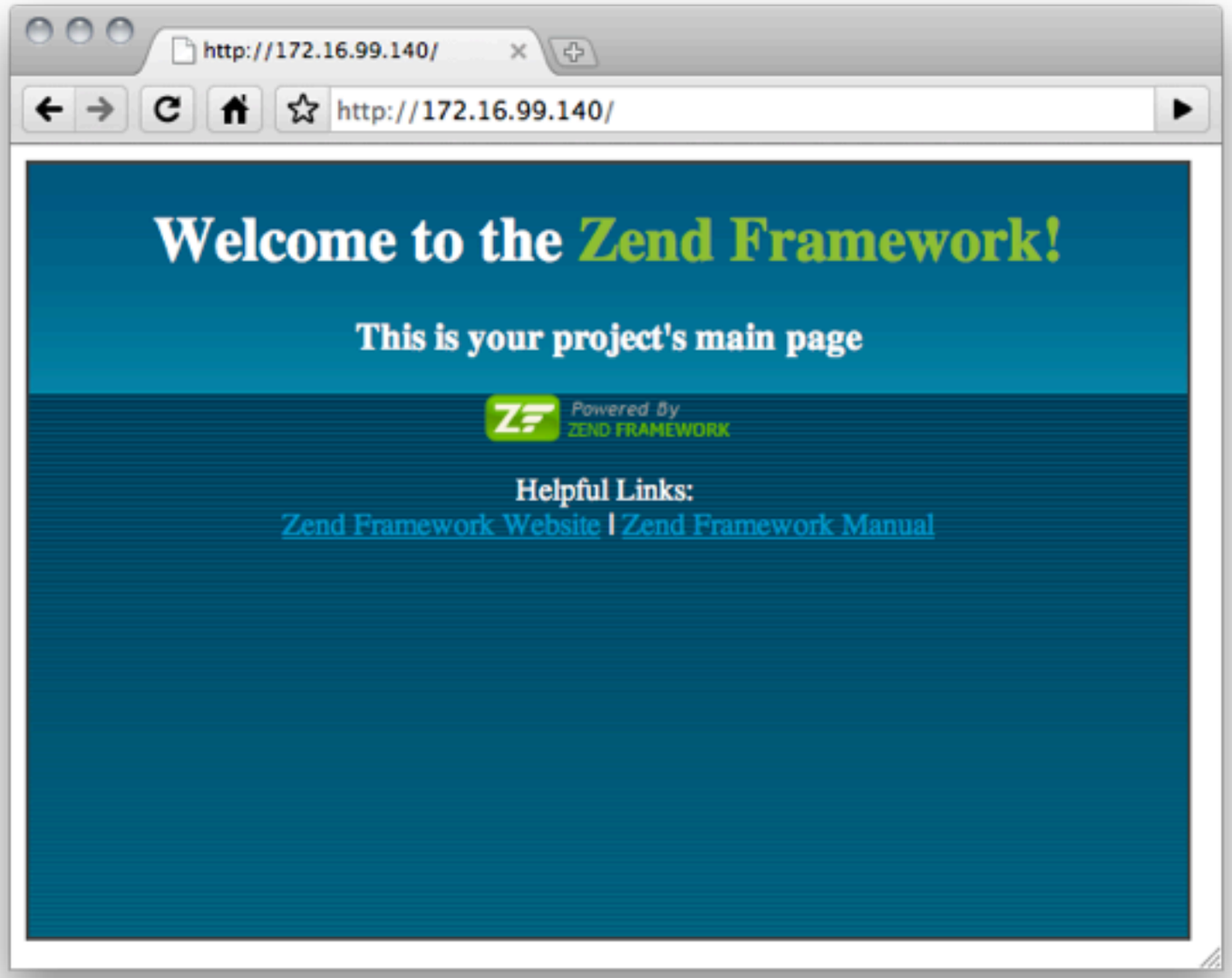
- Restart Apache

```
sudo /etc/init.d/apache2 restart
```

Checking Site

- Once you have created the basic site, you should be able to see it by going to:

  `http://<Your VM>/`



Hosting Static Content

PENNSTATE 1855

- The Zend project has a public folder
  - This folder has an .htaccess file that contains some Apache URL rewriting rules
  - These rules ensure that requests for static content will be served before redirecting to the Zend application
- Example: Paper summaries page
  - In public directory I place my summaries.html file
  - When I surf to http://statecollege.cse.psu.edu/summaries.html, that static summaries.html file will be served

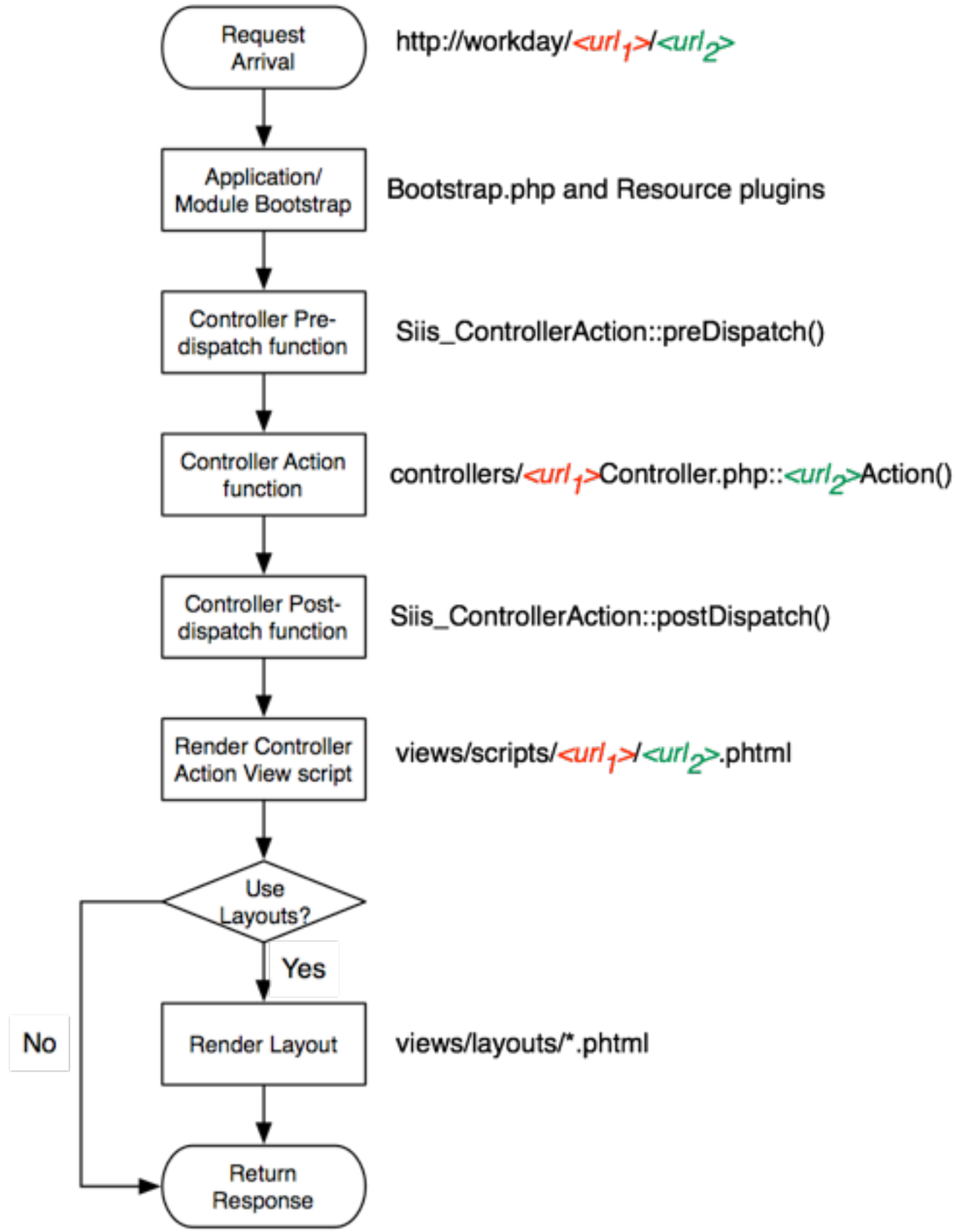
Mapping URLs to Code

PENNSTATE 1855

- Zend maps URLs to specific files
  - http://myexampleapp.com/news/viewall
    - First directory in URL indicates the controller to use (news in this example)
      - Zend will (by default) look for `application/controllers/NewsController.php`
    - Zend then calls the correct action (viewall in this example) to handle the request inside the correct controller
      - The action corresponds to a function in the controller `public function viewallAction()`

Zend Request Processing

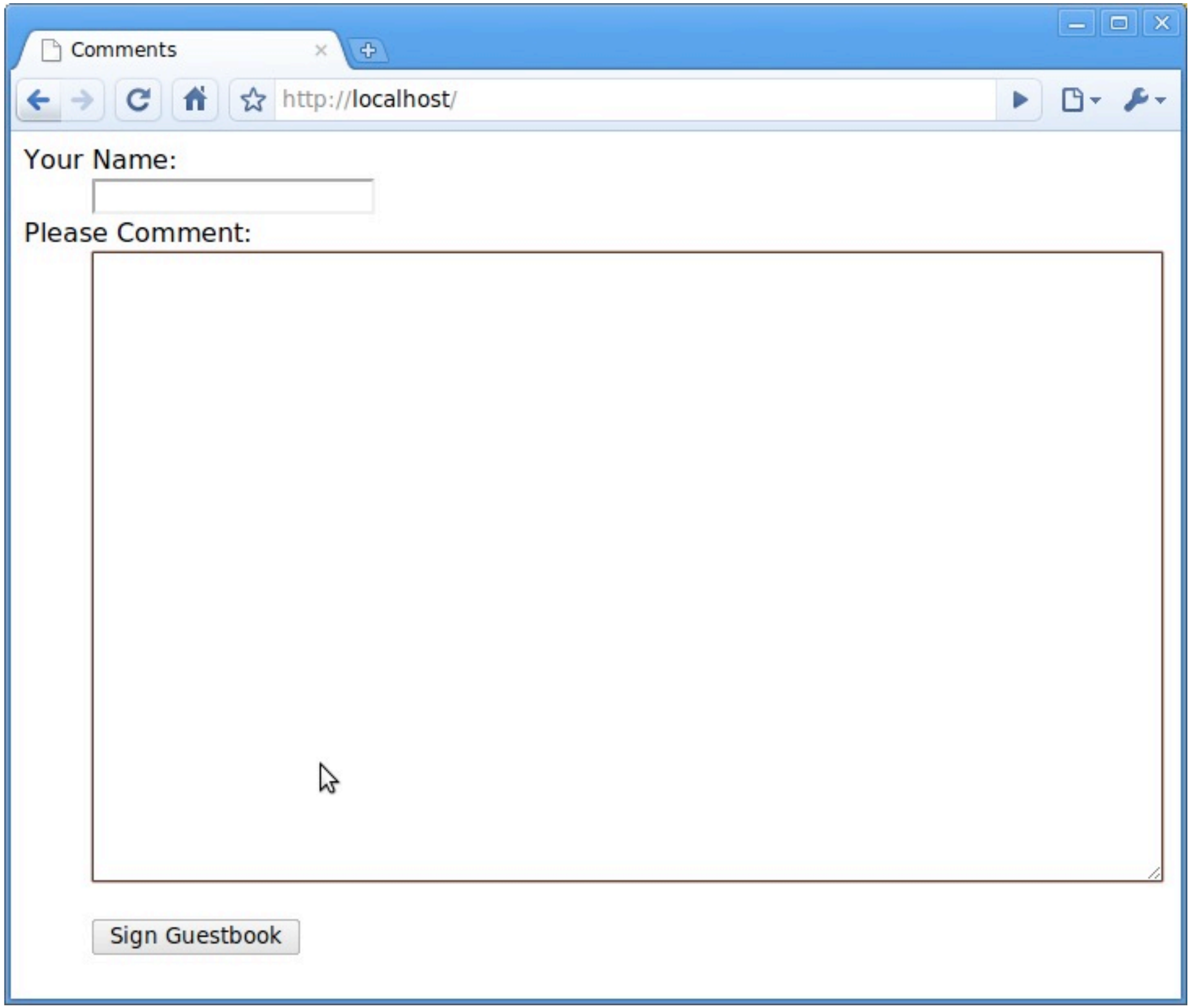
- Zend maps URLs to application code
  - First part of URL maps to the specific controller
  - Second part maps to the action function within the chosen controller
- All requests start in `public/index.php`



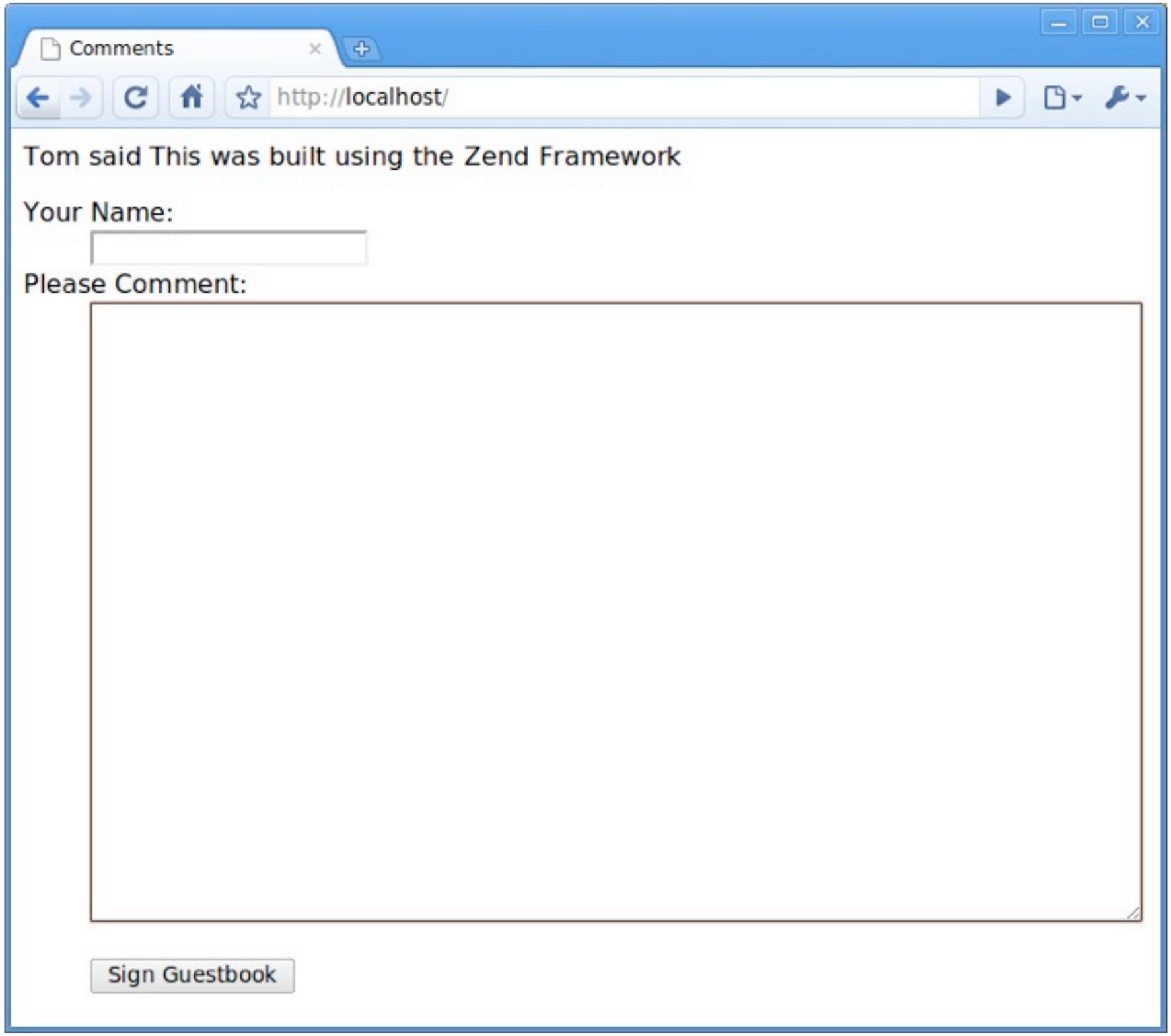
Basic application outline

- Build a simple comment system
  - Takes user's name and comment
- Displays all comments ever entered
  - Need to be careful with user input

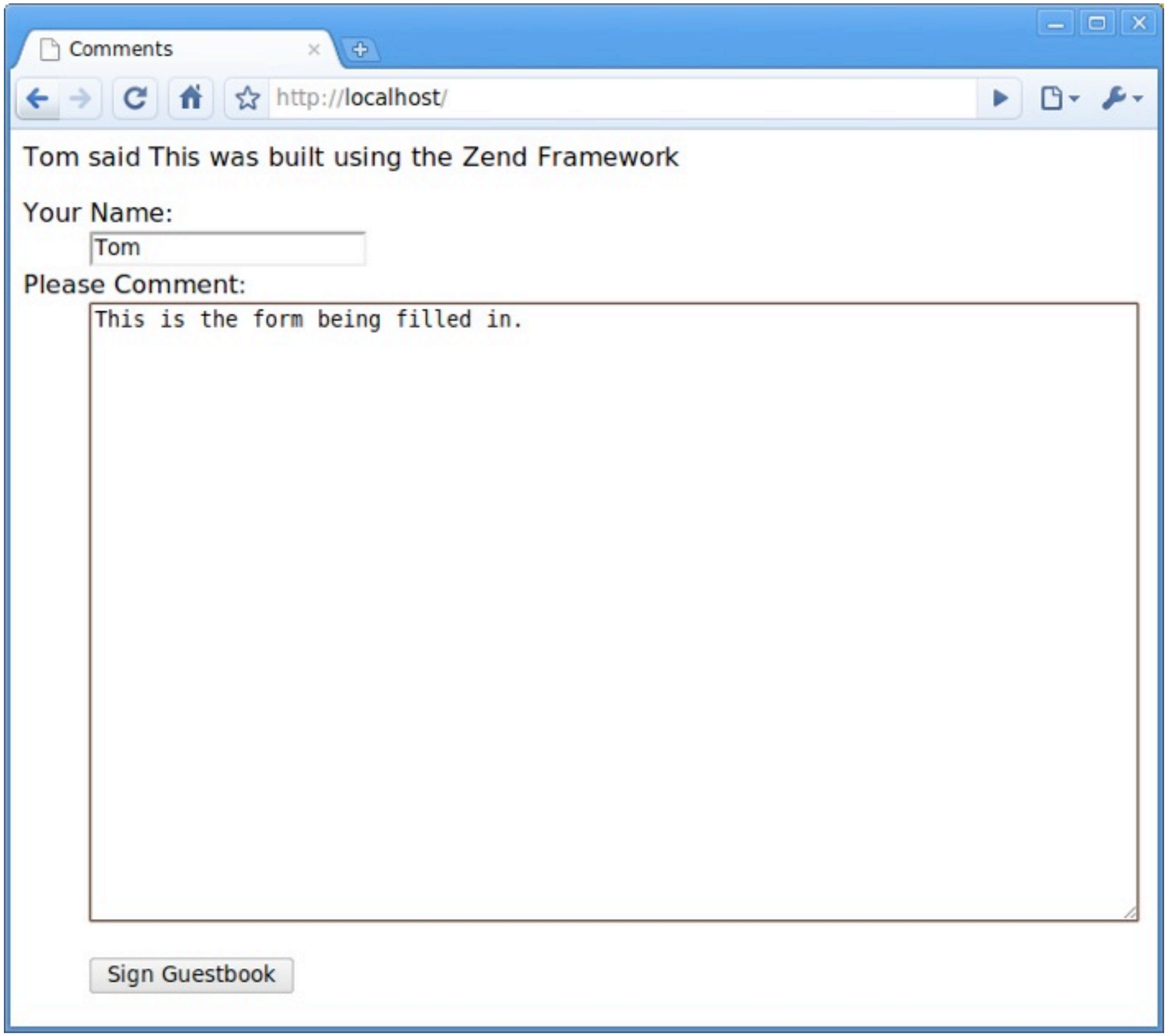
Comment Application



Comment Application (2)



Comment Application (3)



Setting up Autoloading

PENNSTATE 1855

- When using PHP, it is necessary to specify what files to load
  - Typically using the functions require() and require_once()
- This gets to be a pain
  - Zend provides an Autoloading module that will handle loading modules on demand
  - It is necessary to setup the autoloading, typically in the application bootstrapping phase

Autoloading Code

PENNSTATE 1855

- Insert the following code in `application/Bootstrap.php`

```
protected function _initAutoload()
{
    $autoloader = new Zend_Application_Module_Autoloader(array(
        'namespace' => 'Default_',
        'basePath'  => dirname(__FILE__),
    ));
    return $autoloader;
}
```

- The namespace means any classes that need loaded starting with 'Default_' will take advantage of the autoloader
- The autoloader will start the search at the level of the Bootstrap.php file

Mapping Classes to Files

PENNSTATE 1855

- The autoloader will try to map a class name to a file name using the following convention

  Class Name: Default_Form_Comment
  File Name: application/forms/Comment.php

- Another example

  Class Name: Default_Model_User_Prefs
  File Name: application/models/User/Prefs.php

- '_' maps to '/' which is the directory separator

Create Form

PENNSTATE 1855

- First part of application: the form
- Zend provides some convenience classes for handling forms
  - `Zend_Form`
- This class can also double as a *filter/validator* for input

Zend_Form Example

PENNSTATE 1855

```
<?php
class Default_Form_Comment extends Zend_Form
{
    public function init()
    {
        // Set the method for the display form to POST
        $this->setMethod('post');
        $this->setAction('/index/processform');

        // Add an email element
        $this->addElement('text', 'name', array(
            'label'      => 'Your Name:',
            'required'   => true,
            'filters'    => array('StringTrim'),
            'validators' => array(array('validator' => 'StringLength', 'options' => array(0, 20)))));

        // Add the comment element
        $this->addElement('textarea', 'comment', array(
            'label'      => 'Please Comment:',
            'required'   => true,
            'validators' => array(array('validator' => 'StringLength', 'options' => array(0, 100)))));

        // Add the submit button
        $this->addElement('submit', 'submit', array('ignore'   => true, 'label'    => 'Sign Guestbook', ));
    }
}
```

Include Form

- Once we have the form class, we can create instances of the form in the PHP code

```
$form = new Default_Form_Comment();
```

- This object can be used for multiple purposes
  - *Creating* HTML markup

```
echo $form
```

  - *Validating* user input

```
$form->isValid($formData)
```

  - $formData is an array of input values
    - e.g. $formData['name'] = 'Thomas Moyer'
    - 'name' is the name of an element in the form

Adding a New Action

PENNSTATE 1855

- When forms are submitted, there is some backend code that processes the input
  - We will handle this in a new action within the Index controller
  - We use the 'zf' tool to create the relevant code stubs

```
zf create action processform index
```

Action Name Controller Name

- This creates the function `processformAction()` in `application/controllers/IndexController.php`

Processing Form Inputs

PENNSTATE 1855

```
public function processformAction()
{
    $request = $this->getRequest();
    $form    = new Default_Form_Comment();

    if ($this->getRequest()->isPost()) {
        if ($form->isValid($request->getPost())) {
            // Write the name and comment to the text file.
            $formData = $this->getRequest()->getPost();
            $fp = fopen('comments/comments.txt', 'a');
            fwrite($fp, "<p>" . htmlspecialchars($formData['name']) .
                " said " . htmlspecialchars($formData['comment']) .
                "</p>\n");
            fclose($fp);
            return $this->_helper->redirector('index');
        }
    }
    return $this->_helper->redirector('index');
}
```

Displaying Comments

PENNSTATE 1855

```
public function indexAction()
{
    $this->view->comments = "<p>No comments yet!</p>";
    // Read all the comments to date and put them here.
    if(file_exists("comments/comments.txt")) {
        $handle = fopen("comments/comments.txt", "rb");
        $contents = '';
        while (!feof($handle)) {
                $contents .= fread($handle, 8192);
        }
        fclose($handle);
        $this->view->comments = $contents;
    }
    $this->view->commentForm = new Default_Form_Comment();
}
```

Directory for comments

PENNSTATE 1855

- Apache needs someplace to store files it writes
- Create a directory in `<path>/public/` called `comments`
- Change the group to `www-data`*

```
sudo chgrp www-data comments
```

- Make it writeable by the group*

```
sudo chmod g+w comments
```

*Only need sudo if you are not a member of the www-data group

Displaying Comments (2)

```
<html>
    <head>
        <title>Comments</title>
    </head>
    <body>
        <?php echo $this->comments;?>
        <?php echo $this->commentForm; ?>
    </body>
</html>
```

Debugging

PENNSTATE 1855

- Debugging a web application can be somewhat difficult
  - Part of the code runs on the server and part on the client
- There are modules for PHP that aid in debugging
  - XDebug and Zend Debugger
  - Easy to install XDebug on your VM

```
sudo apt-get install php5-xdebug
```

  - PHP debuggers allow external debuggers to interact with the running PHP code (similar to GDB)
  - Need a client to use the debugger (see http://xdebug.org/docs/remote)

Using an IDE

PENNSTATE 1855

- Several popular IDE's exist for PHP development
- My personal choice is *Eclipse*
  - With the PHP Development Toolkit (PDT)
- Others that I have worked with
  - NetBeans
  - Vim (requires a fair bit of work to use as an IDE)

More Information

PENNSTATE 1855

- Zend Homepage
  - http://framework.zend.com
- Zend Quickstart Guide
  - http://framework.zend.com/docs/quickstart
- Zend Reference Guide
  - http://framework.zend.com/manual/en/
- Zend API Documentation
  - http://framework.zend.com/apidoc/core/

More Information (2)

PENNSTATE 1855

- Eclipse Homepage
  - http://www.eclipse.org
- NetBeans
  - http://www.netbeans.org
- XDebug
  - http://xdebug.org