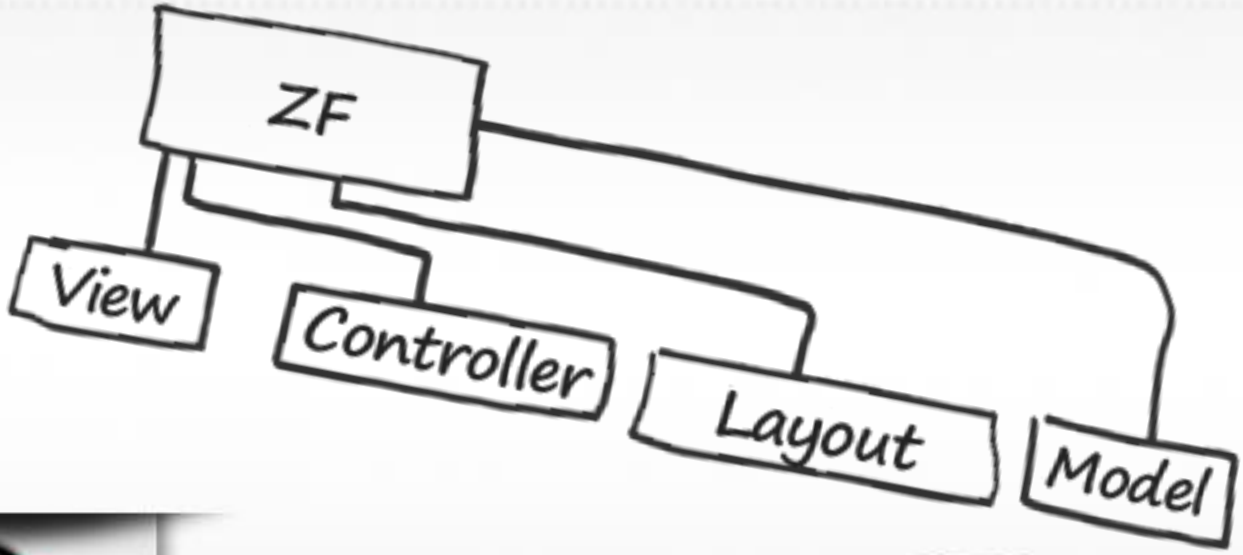
Zend Framework

NOW

ZEND FRAMEWORK



OPEN SOURCE DEPARTMENT

Information Technology Institute

Course Materials

You can access the course materials via this link

http://goo.gl/sef2ZN

Day 1 Contents

- History
- Download and install
- Zend Tool
- MVC
- Zend framework project directory structure
- Controller/Action
- Views
- Getting / Passing Parameters
- View Helpers

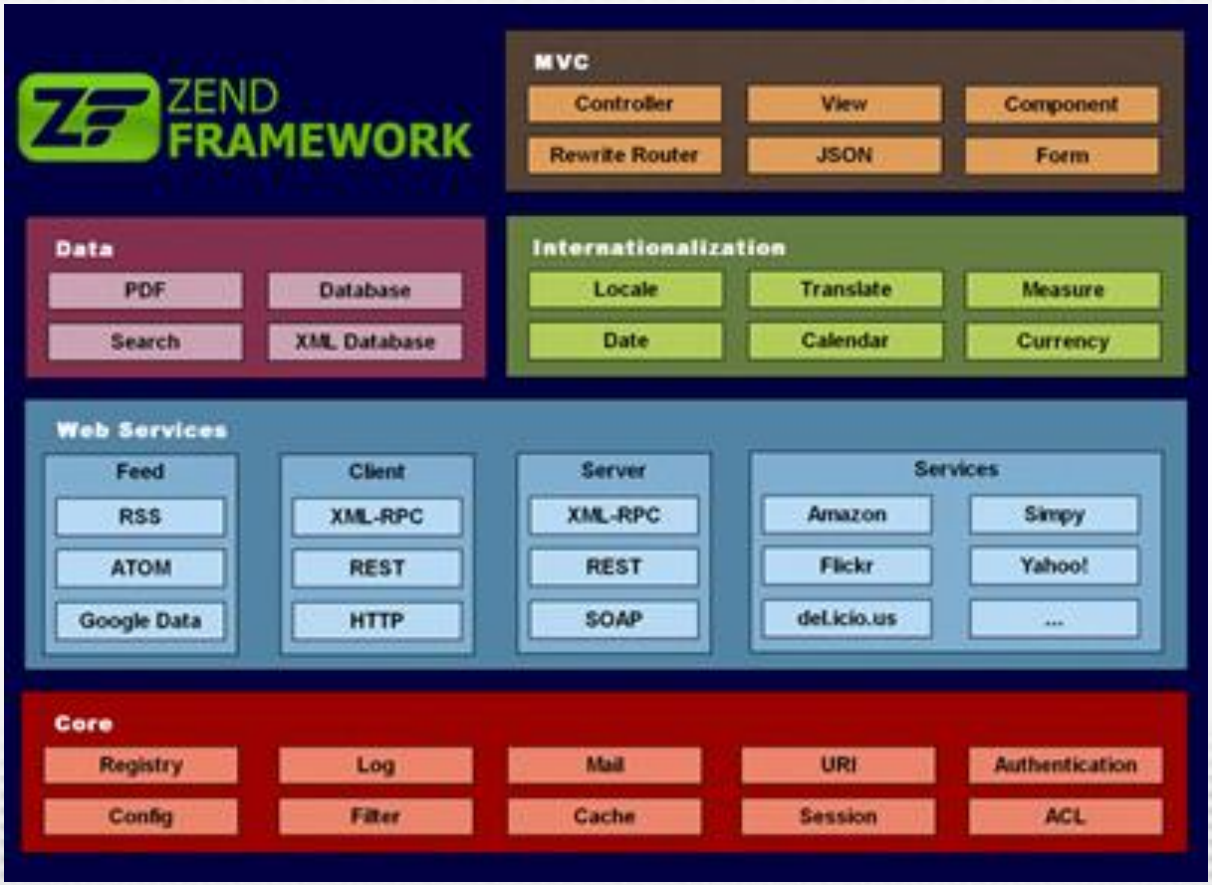
History

- Coding on Zend Framework officially started in July 2005. It was announced to the general public in the same year at the first ZendCon.
- The first public release was on March 4, 2006, version 0.1.2. More than a year later, the first version (1.0) was released on July 2, 2007.
- Intial development of Zend Framework 2.0 was released on August 6, 2010. The first stable release of Zend Framework 2.0 was released 5 September, 2012.

Why Zend Framework?

- Free and open source framework
- Extreme Simplicity
- Flexible Architecture
- Support for multiple database systems and vendors, including MariaDB, MySQL, Oracle, IBM DB2, Microsoft SQL Server, PostgreSQL, SQLite, and Informix Dynamic Server

Zend Framework Libraries



Using Zend Framework

- Download the latest version of Zend Framework 1.0 from this link

http://framework.zend.com/downloads/latest#ZF1

- Extract the downloaded file to your localhost directory.
- Configure Zend Tool

Using Zend Framework

- Zend Tool is command line tool to manage your zend project (Create project, model, controller, view, enable layout and configure db adabter).
- To configure Zend Tool , Open your ~/.bashrc add these lines:

```
PATH=$PATH:/var/www/ZendFramework-1.12.1/bin/
alias zf="/var/www/ZendFramework-1.12.1/bin/zf.sh“
```

Using Zend Framework

- Create Your first project using Zend Tool:

```
$zf create project/var/www/site
```

- Create symbolic link to the library:

```
ln -s /var/www/ZendFramework-
1.12.1/library/Zend
/var/www/site/library
```

Using Zend Framework

- You need to configure your apache by adding this directive to your conf. file:

```
<Directory /var/www/site >
DirectoryIndex index.php
AllowOverride All
</Directory>
$ sudo a2enmod rewrite
$sudo service apache2 restart
```

Using Zend Framework

- You can also create Virtualhost (recommended)

```
$ sudo a2enmod rewrite
$ sudo vi /etc/apache2/sites-
available/site.conf
<VirtualHost *:80>
        ServerName os.iti.gov.eg
        DocumentRoot /var/www/site/public
         SetEnv APPLICATION_ENV "development"
      <Directory /var/www/site>
            DirectoryIndex index.php
            AllowOverride All
      </Directory>
</VirtualHost>
```

Using Zend Framework

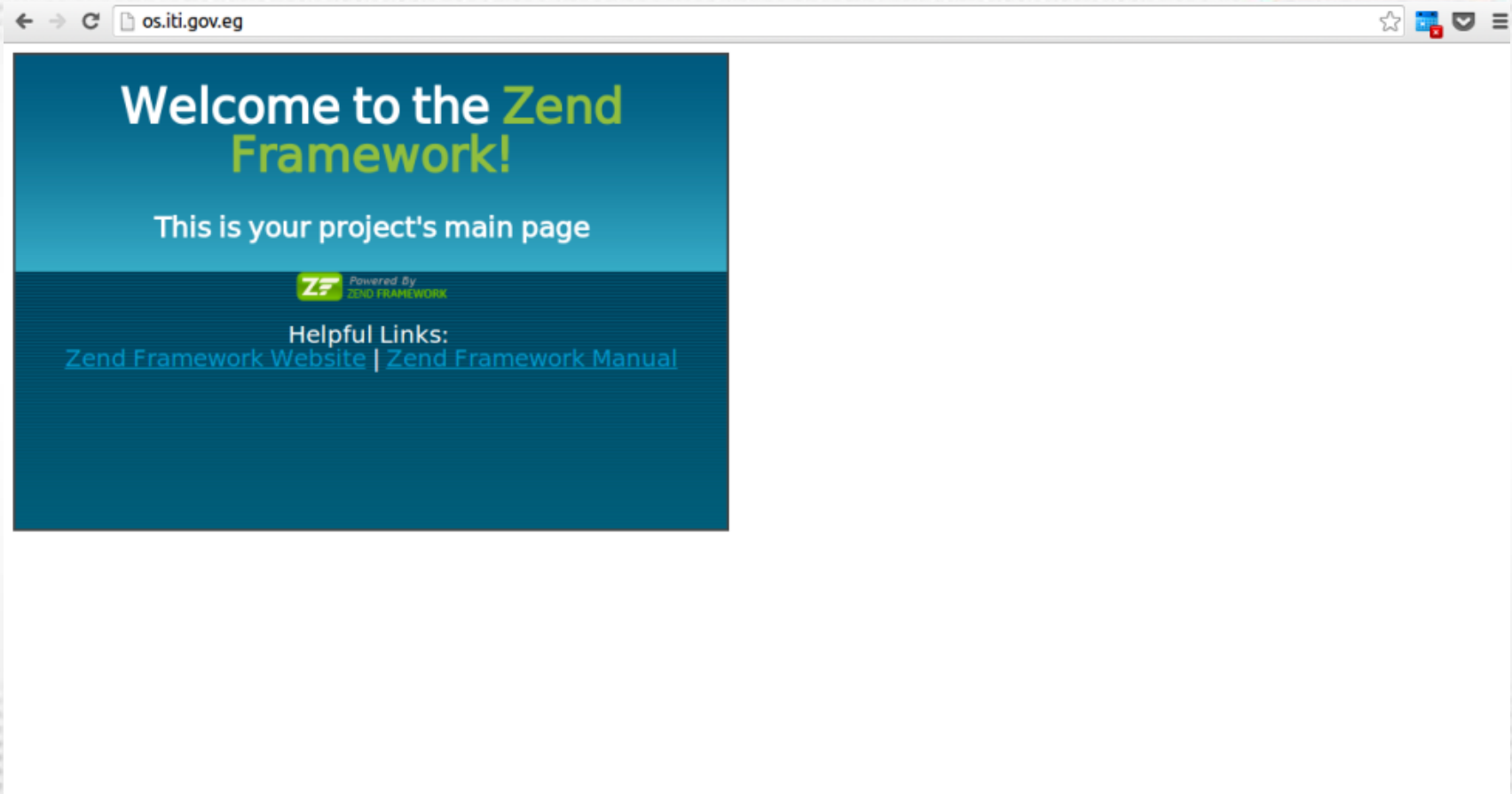
- Edit your /etc/hosts, add:

```
$ gksudo gedit /etc/hosts
127.0.0.1   os.iti.gov.eg
$sudo a2ensite site.conf
$sudo service apache2 reload
```

- Open your browser, type:

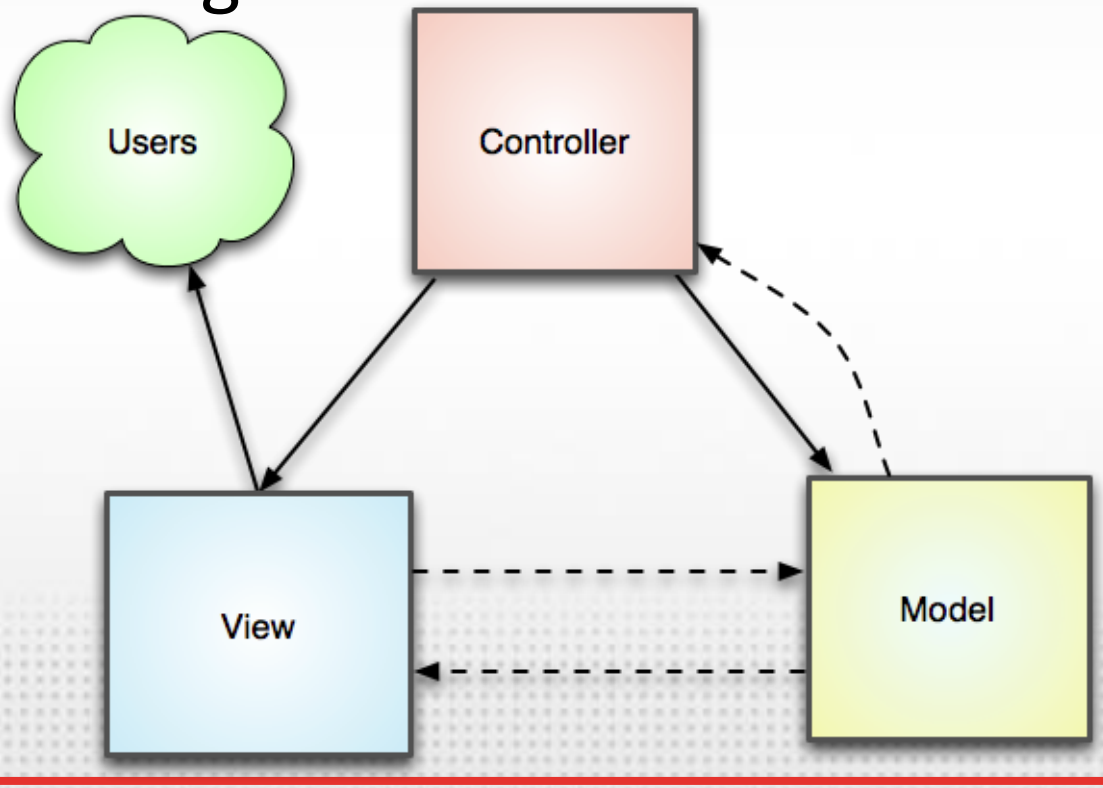
http://os.iti.gov.eg

Using Zend Framework



MVC

- Most web application code falls under one of the following three categories:
  - presentation,
  - business logic,
  - and App flow



MVC

- **Model** - the part of your application that defines its basic functionality. Data access routines and some business logic can be defined in the model.
- **View** - Views define exactly what is presented to the user. Usually controllers pass data to each view to render in some format. Views will often collect data from the user, as well. This is where you're likely to find HTML.
- **Controller** - Controllers bind the whole pattern together. They manipulate models, decide which view to display based on the user's request and other factors, pass along the data that each view will need, or hand off control to another controller entirely

Zend Framework structure

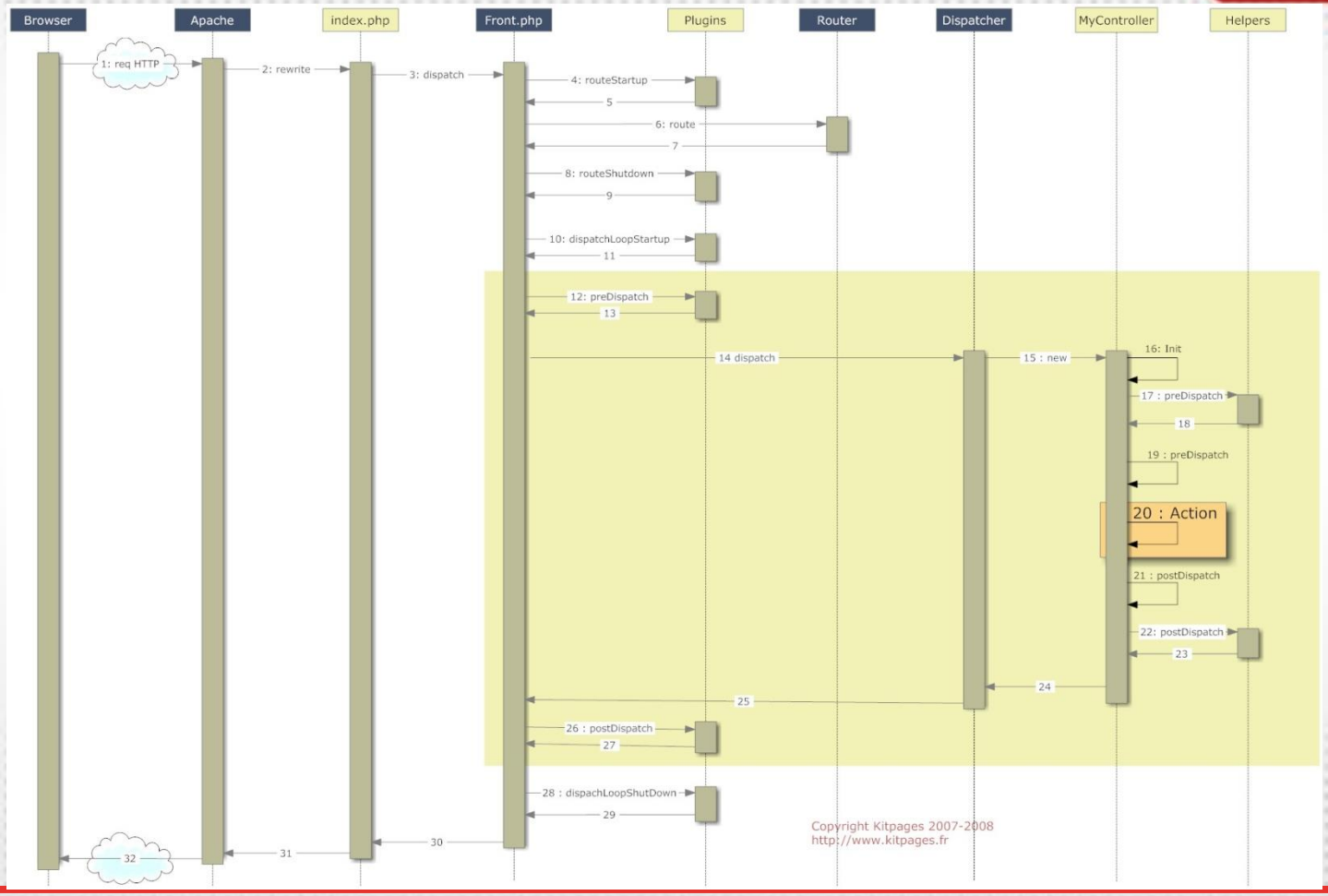

```
site
|-- application
|   |-- Bootstrap.php
|   |-- configs
|   |   `-- application.ini
|   |-- controllers
|   |   |-- ErrorController.php
|   |   `-- IndexController.php
|   |-- models
|   `-- views
|       |-- helpers
|       `-- scripts
|           |-- error
|           |   `-- error.phtml
|           `-- index
|               `-- index.phtml
|-- library
|-- public
|   |-- .htaccess
|   `-- index.php
`-- tests
    |-- application
    |   `-- bootstrap.php
    |-- library
    |   `-- bootstrap.php
    `-- phpunit.xml
```

App directory

Zend Library

Public directory

Request Life Cycle



Zend Framework files

- Index.php

```
<?php

// Define path to application directory
defined('APPLICATION_PATH')
    || define('APPLICATION_PATH', realpath(dirname(__FILE__) . '/../application'));

// Define application environment
defined('APPLICATION_ENV')
    || define('APPLICATION_ENV', (getenv('APPLICATION_ENV') ? getenv('APPLICATION_ENV') :
'production'));

// Ensure library/ is on include_path
set_include_path(implode(PATH_SEPARATOR, array(
    realpath(APPLICATION_PATH . '/../library'),
    get_include_path(),
)));

/** Zend_Application */
require_once 'Zend/Application.php';

// Create application, bootstrap, and run
$application = new Zend_Application(
    APPLICATION_ENV,
    APPLICATION_PATH . '/configs/application.ini'
);
$application->bootstrap()
            ->run();
```

Controller

- Your application's action controllers contain your application workflow, and do the work of mapping your requests to the appropriate models and views
- A controller should have one or more methods ending in "Action"; these methods may then be requested via the web. By default, Zend Framework URLs follow the schema /controller/action, where "controller" maps to the action controller name (minus the "Controller" suffix) and "action" maps to an action method (minus the "Action" suffix).

Controller

- Typically, you always need an IndexController, which is a fallback controller and which also serves the home page of the site, and an ErrorController, which is used to indicate things such as HTTP 404 errors (controller or action not found) and HTTP 500 errors (application errors).

```
// application/controllers/IndexController.php
class IndexController extends Zend_Controller_Action
{
    public function init()
    {
        /* Initialize action controller here */
    }
    public function indexAction()
    {
        // action body
    }
}
```

Controller

- To create a new controller

```
zf create controller Auth
```

- This will create a controller named Auth, specifically it will create a file at application/controllers/AuthController.php as well as their corresponding view script directories and files.

Controller

- To create an action within an existing controller:

```
zf create action login Auth
zf create action login -c Auth
zf create action login --
controller-name=Auth
```

This will create the corresponding view.

Views

- Views in Zend Framework are written in plain old PHP. View scripts are placed in application/views/scripts/, where they are further categorized using the controller names. In our case, we have an IndexController and an ErrorController, and thus we have corresponding index/ and error/ subdirectories within our view scripts directory. we thus have the view scripts index/index.phtml and error/error.phtml.

Passing Parameter

- Parameters are sent in URL (GET) after action as:

controller/action/param/value

- To get these parameters (POST/GET) inside the action using:

```
$this->_request->getUserParams();
$this->_request->getParam($key);
```

Passing Parameter

- To send a parameter to the corresponding view:

```
$this->view->param = $value;
$this->view-
>assign('param','value');
$this->render();
$this->redirect($url);
```

- TO get the full path inside the view

```
$this->baseUrl();
```

View Helpers

- In your view scripts, often it is necessary to perform certain complex functions over and over: e.g., formatting a date, generating form elements, or displaying action links. You can use helper classes to perform these behaviors for you.
- By default, the class is prefixed with 'Zend_View_Helper_Name as example Zend_View_Helper_FooBar. This class should contain at the minimum a single method, named after the helper, and camelCased: fooBar().