CodeIgniter®

Modern Programming paradigms in Web Engineering

Lecture 04

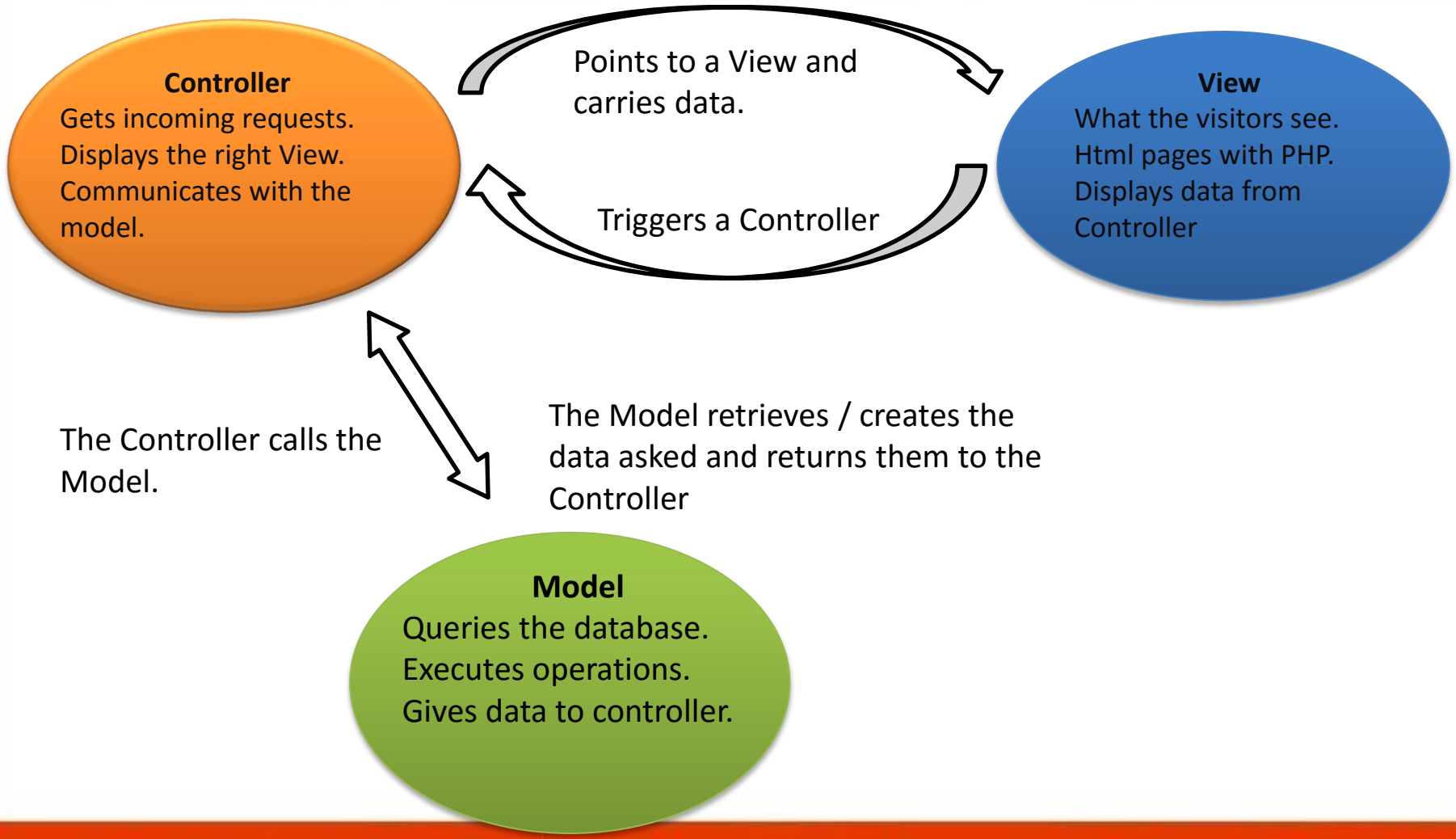
MVC

By: Ashiqullah Alizai
Herat University
Computer Science Faculty
0795642400
Alizai.csf@hotmail.com

MVC

- The MVC architecture pattern separates the representation of data from the logic of the application.
- The **View** is what the visitors of the web application see.
- The **Controller** is responsible for handling the incoming requests, validating input and showing the right view.
- The **Model** is responsible for accessing the database or executing other operations.

MVC


Controller
Gets incoming requests.
Displays the right View.
Communicates with the model.
Points to a View and carries data.
Triggers a Controller
View
What the visitors see.
Html pages with PHP.
Displays data from Controller
The Controller calls the Model.
The Model retrieves / creates the data asked and returns them to the Controller
Model
Queries the database.
Executes operations.
Gives data to controller.

Intro to CodeIgniter

- How does a Controller trigger?

  Each Controller triggers by a certain URI.

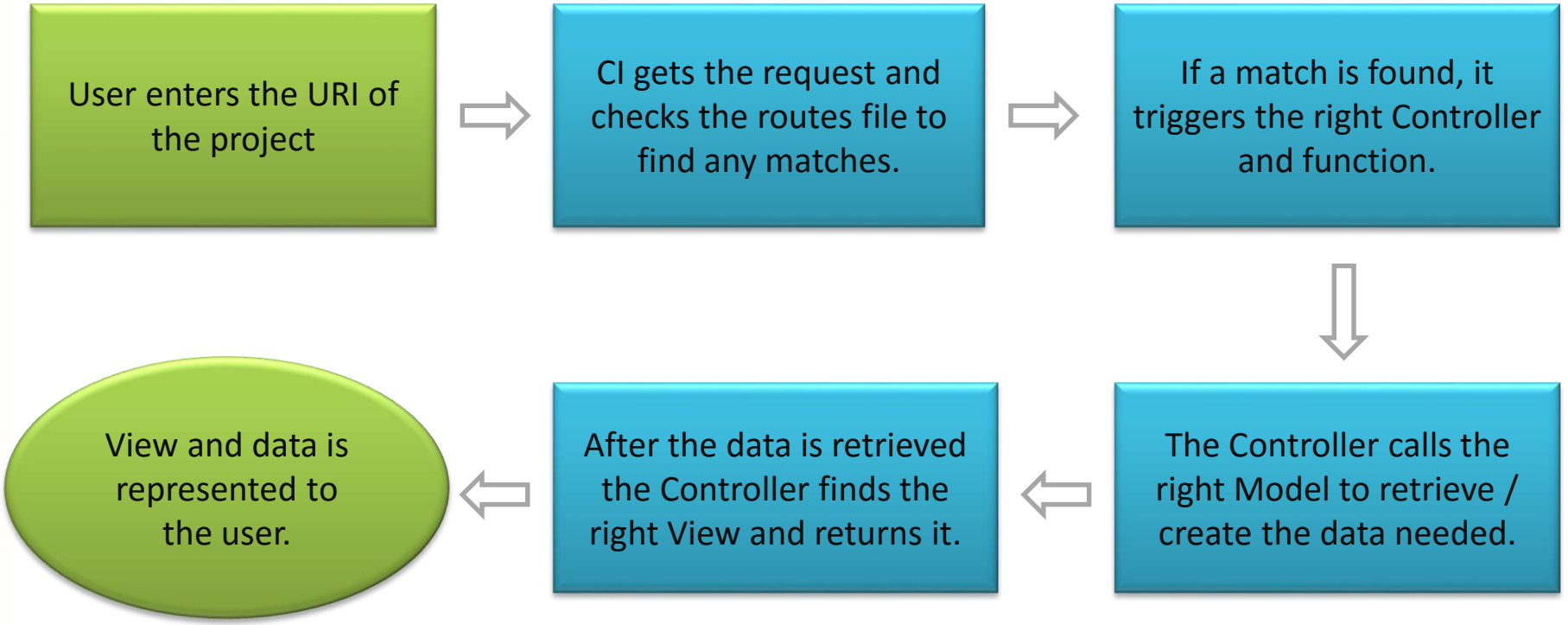
- What happens when a Controller is triggered?

  It then calls a **Model** (if any data should be retrieved or created), it finds the **View** that should be shown to the visitor and then it returns that **View** with the corresponding **data**.

- How does CI knows what Controller to trigger?

  This is defined by **routes**. Routes is a PHP configuration file that maps each URL of our web project to a Controller and a certain function.

Intro to CodeIgniter



Controllers – The Class

A controller in CI is basically a custom made class, which inherits from the CI_Controller class.

```
class WELCOME extends CI_Controller {
  }
```

Create a controller with name "welcome.php". The new controller is located into applications/controllers folder.

In every controller we create, the constructor method must be declared. In the constructor we can load libraries, models, the database, create session variables and generally define content that will be used by all methods. The constructor must have the same name as the class.

Controllers – The Constructor

```
public function __construct() {
parent::_construct();

$this->load->helper('url');
$this->load->helper('file');

$this->load->database();

$this->load->model('welcome_model');
}
```

The first function of the controller is the constructor, where we can load…

… the helper libraries we may need….

… the database of our project…

… and the models we need.

Controllers – The Functions

Then we can write the functions of the controller, that will be triggered by the system. The functions can either perform some operations (through the model), load a view, or even both.

```
public function home () {

if (!file_exists(application/views/home.php)){
        show_404();
}

$this->load->view('home');
}
```

The most usual operation is to check if a view file exists and load it.

View files are located in application/views folder. Views are what the visitors see.

In case it does not exist, a 404 error message is displayed to the visitor.

Controllers – The Functions

```
public function home () {

        if (!file_exists(application/views/home.php)){
                show_404();
        }

        $data['files']=$this->welcome_model->get_data();
        $this->load->view('home',$data);
}
```

In case we need to send some data to the view, we retrieve them by calling the right function from the model and then load them with the view. In the above example, the data is then accessible by the name "files" in the "home" view.

Routes File

The **routes** file contains the matches for the URIs and the Controllers / Functions.

There is a **default controller** which is triggered from the Base URL of our project (e.g. www.my_project.com).

```
$route['default_controller']="controller_name/function_name";
```

So, if we try to access the www.my_project.com, the routes file understands it as the default controller and triggers the controller and function we define, e.g. the welcome controller and the home function.

```
$route['default_controller']="welcome/home";
```

Routes File

- There is a pattern which we have to follow in order to create mappings of URIs and controllers / functions.
- my_project.com/class/function/id/
- The first segment is reserved for the **controller class**, the second for the **function** and the third of any values we want to pass as **arguments** (optional). In case we don't want to follow this pattern, the URI handler has to be reconfigured.
- If we want to map another URI, we have to follow that pattern. In the following example if the URI is **www.my_project.com/welcome/blog**, CI triggers the **welcome controller** and the **blog method**.
- $route['welcome/blog']="welcome/blog";

Models – Class/ Constructor

In a model we perform some tasks such as execute database queries, read / write files or perform other operations. The models are located in applications/models folder.

```
class Welcome_model extends CI_Model {

        public function _construct () {
                $this->load->database();
        }
}
```

Each model we create, extends from the CI_Model.

At first we have to write the constructor of the model. In the constructor we load the database or other helper libraries.

Models - Functions

```
class Welcome_model extends CI_Model {

public function get_data() {

        $query=$this->db->get(....);
        /* perform other operations */
        return $files;
}
}
```

In a model function we can perform any operation we need, and then return the data to the corresponding function in the controller.

Conclusion

- The previous topics complete a basic intro into CodeIgniter and how it essentially works.

- CodeIgniter supports **helpers**, which is essentially a collection of functions in a category, for example the helper for working with files (read / write) is “file” and **libraries** as form validation. All of these can come in handy and help a lot in developing your projects.

- The **database** class of CodeIgniter supports both traditional structures as Active Records patterns. Also, someone could set up CodeIgniter to run with **Doctrine** (ORM).

- For the complete CodeIgniter documentation visit here.

This Is The End For This Lecture