Codeigniter 3

Training (part 1)
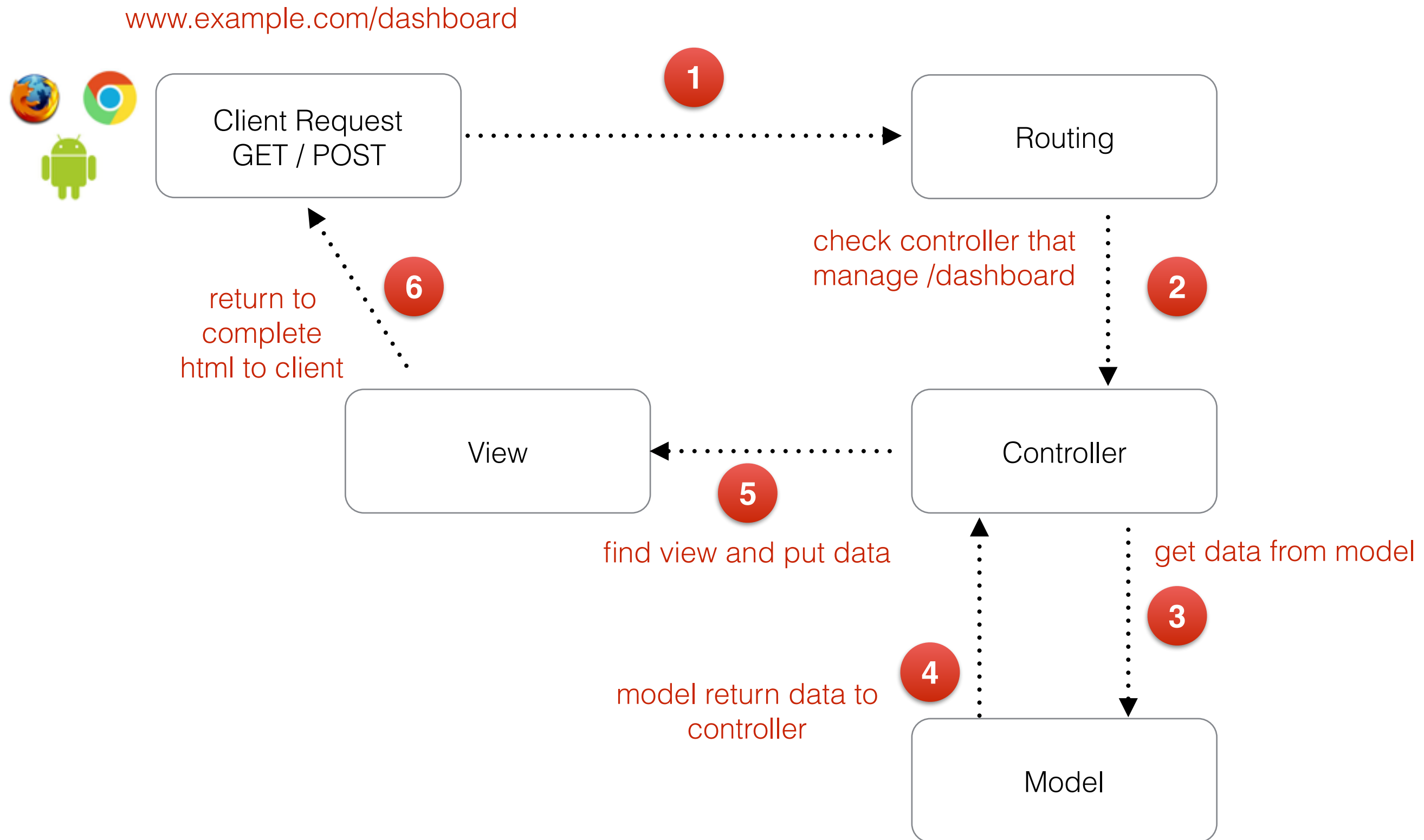
Conducted by Terato Tech Sdn Bhd

Content

- Codeigniter (CI)?

- CI architecture

- MVC

- View

- Controller

- Model

- Routing

- Useful CI Helpers & Libraries (Classes to make job easy)

- Installation

- Create Static Page

- Divide view to header, footer & content
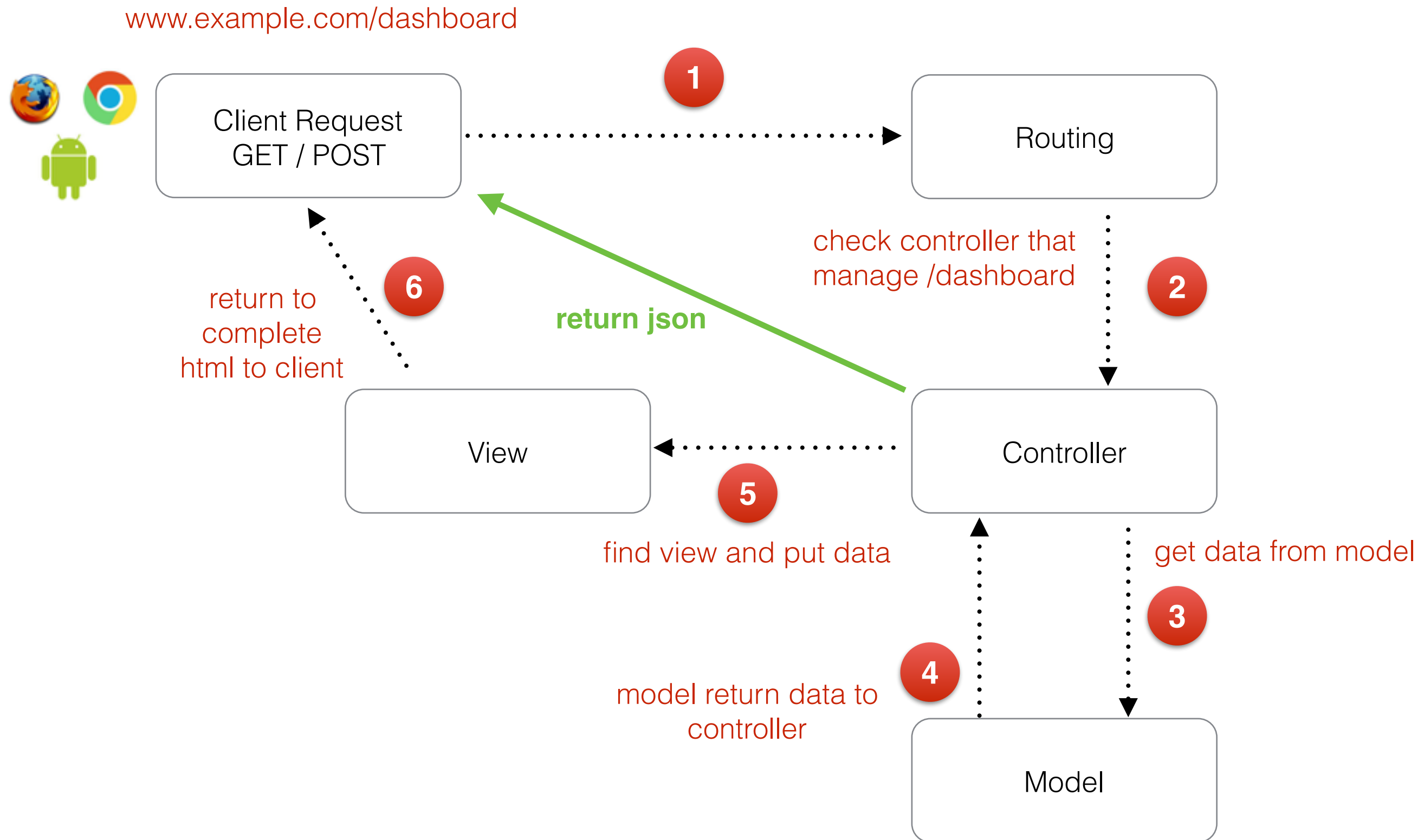
- Pass variable to view

Codeigniter (CI) ?

- PHP Framework

- It has set of libraries that make your job easy

- Easy to use and configure database

- Clean routing / url - example.com/index.php/news/article/1

- Secure

- MVC

- Form and Data Validation

- Query Builder Database Support

- Session Management

- Easy to debug error

How CI works?

www.example.com/dashboard

1

Client Request
GET / POST

Routing

check controller that
manage /dashboard

2

6

return to
complete
html to client

View

5

find view and put data

Controller

get data from model

3

4

model return data to
controller

Model

How CI works with **json**?

www.example.com/dashboard

Client Request GET / POST

1

Routing

check controller that manage /dashboard

2

return json

6

return to complete html to client

View

5

find view and put data

Controller

get data from model

3

4

model return data to controller

Model

File Structure

/ **Project Directory**

  / **system** - contains framework libraries and helpers

   / **application**

      / **config**

      / **controllers**

      / **models**

      / **views**

MVC

- Stands for **M**odel-**V**iew-**C**ontroller

- Separates **application logic** from **presentation** and **data**

- **Model** is your data either store in array, database of files

- **View** is your html page that will be presented to user request

- **Controller** is an intermediary (middle-man) between Model and View. It gets data from Model and pass to View and vice versa

View

- It is **html** code **mixing** with **php variables** passed to the view

- We need to create all files under **application/views/** so controller can find the file and render it

- `<?php echo $title ?>` is now simplefied to `<?= $title ?>`

- Example:

```
<h1><?= $title ?></h1>

<?php forech($news as $article): ?>
   <p> <?= $article->text ?></p>
<?php endforeach; ?>
```

Controller

- Create all controller files under **application/controllers/**

- Has many methods with different pages / tasks

- Load all libraries and helpers

- Get data from database using Model and pass to View

- Store data to database using Model

- Contains:

  - form Validation

  - data and Variables will be used View

  - any logic codes

Model

- It is a place where we **retrieve, insert, and update** information in **database**

- Dont write queries in controller

- Do write in model so it can be re-use by other controller as well

- Model class located at **application/models/**

- Model will always use 'Query Builder' and avoid raw SQL statement

  - Query builder: $query = $this->db->get('employee');

  - Raw SQL: $query = "SELECT * FROM employee";

Routing

- In CI Framework, url are determines by developer, not by file name

- It can be different url from its file name

- Convention / normal php:

localhost/student/form.php

- CI Framework:

localhost/index.php**/student/add**
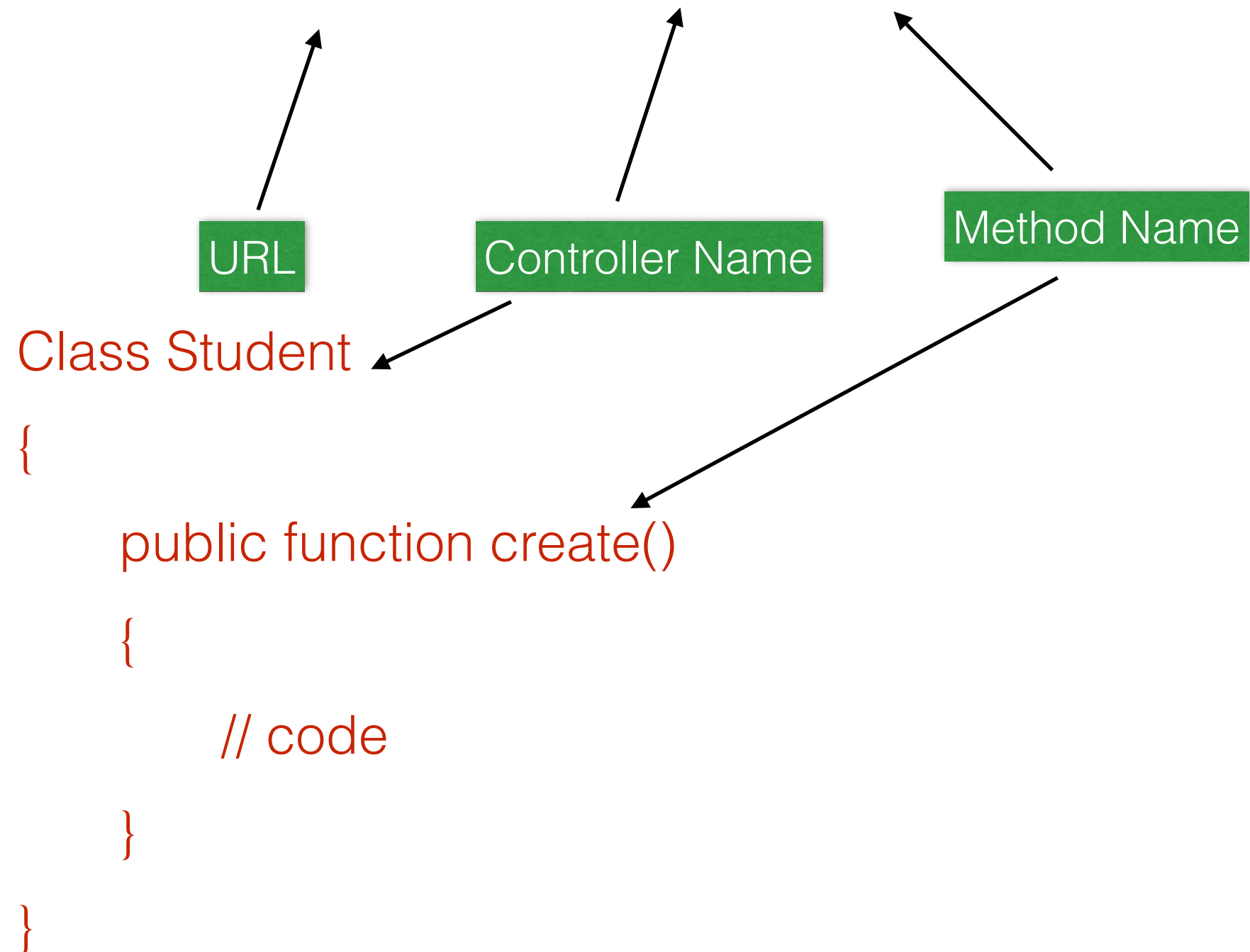
- How it works? it is easy as

$route['**student/add**'] = 'student/create';

Routing

- It is about how we set the URL for each page

- Basic pattern

- **http://example.com/[controller-class]/[controller-method]/[method arguments]**

- For example:

- Files located at config/routes.php
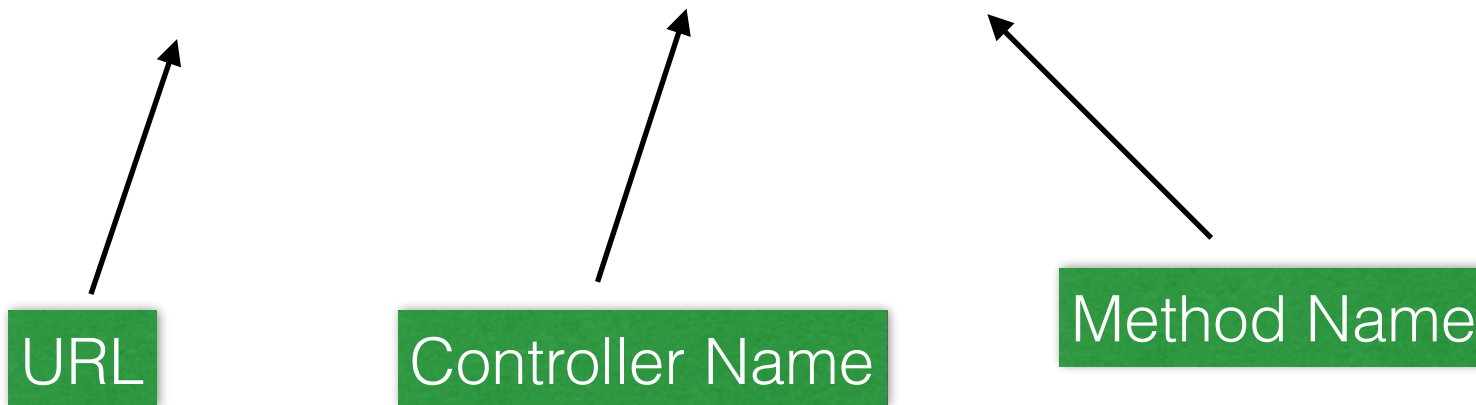
- It also can be custom

Routing (cont.)

$route['student/add'] = 'student/create';

URL Controller Name Method Name

Class Student

{

    public function create()

    {

        // code

    }

}

13

Routing (cont.)

$route['student/add'] = 'student/create';

URL

Controller Name

Method Name

$route['default_controller'] = 'home/index';

$route['students'] = 'student/index';

$route['students/(:any)'] = 'student/detail/$1';

For dynamic ID

Student ID that will be passed to method.
i.e **public function detail($id)**

Libraries and Helpers

- Both are classes that can be used to speed up code writing and improve performance

- Common libraries and helpers used in project

  - database

  - session

  - url

  - form

"Let's create a project"

Installation

- Just use the project folder and ready to start development!

Welcome to CodeIgniter!

The page you are looking at is being generated dynamically by CodeIgniter.

If you would like to edit this page you'll find it located at:

```
application/views/welcome_message.php
```

The corresponding controller for this page is found at:

```
application/controllers/Welcome.php
```

If you are exploring CodeIgniter for the very first time, you should start by reading the User Guide.

Page rendered in **0.0237** seconds. CodeIgniter Version **3.0.0**

Installation (cont.)

- Now change database configuration. Open file **application/config/database.php** and change the *hostname*, *username*, *password* and *database*

Create Static Page

- We will learn how to use the framework

- Static page means, no data to database

- All **controllers, views and routing** will be using here except we are not using model

- Step 1: Add URL that we want in **application/config/routes.php**

- Step 2: Create new controller file in **application/controllers**

- Step 3: Create new view file in **application/views**

- 1 file modified, 2 files created

Step 1

- Add URL that we want in application/config/ routes.php

- Let say we want to have url:

  'localhost/index.php/**home**'

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

our new URL

Step 2

- Create new controller file in **application/ controllers: Home**.php

```php
<?php
class Home extends CI_Controller
{

    public function index( ) {
        $this->load->view('home/index');
    }

}
?>
```

This is a view file located at **application/views/home/index.php**

Step 3

- Create new view file in **application/views:**

  - create folder 'home' (optional - easy to seperate from others)

  - create file index.php

  - so it will be called by controller as 'home/index'

```html
<html>
    <head><title>Home Page</title></head>
    <body>
        <h1>Home</h1>
        <p>This is home page content</p>
    </body>
</html>
```

Pass variable to view

- Variable is passed from controller to view

- Later, data from database also passes to the view in the same way

- In Controller **Home.php**

```
$data['name'] = 'Mohd Hafiz';
$this->load->view('home/index', $data);
```

- In View **application/views/home/index.php**

```
<h1>Home</h1>
<p>This is home page content. Assalamualaikum <?= $name ?></p>
```

More variable to view

- In Controller **Home.php**

  $data['name'] = 'Mohd Hafiz';
  $data['country'] = 'Malaysia';
  $this->load->view('home/index', $data);

- In View **application/views/home/index.php**

  <h1>Home</h1>
  <p>This is home page content. Assalamualaikum **<?= $name ?>**.
  Welcome to **<?= $country ?>** </p>

"static page done!"

Contact

Address:
No 1, Jalan Saga,
Petaling Jaya,
Selangor

Enquiry Form

Email:

Description

Submit