DMIT2503 – ADVANCED WEB CONCEPTS

CODEIGNITER MVC PHP FRAMEWORK – DAY 3

Overview: This lesson is continued from the CodeIgniter Lessons – Day 2.

LET'S TIDY UP…

Ok, to get started, let's tidy up a few more things before we delve into databases, models, validation and god knows what else.

- Some Bootstrap stuff in **views/includes/header.php**:
  - The first link should go to the Home page, so I'm changing mine to this: a class="navbar-brand" href="<?php echo base_url()?>">Home</a>
- Using PHP constants for global data:
  - Open **config/constants.php:**
    - Add this new line at the bottom: **define('APP_NAME', 'My Crazy App');**
  - Open **views/includes/header.php** and let's create a dynamic <title> tag:
    - Change your <title>Hello World!<title> to the following:

```
// views/includes/header.php

<title><?php
   if(APP_NAME){
     $title =  APP_NAME;
   }
   if(isset($heading)){
     $title = $title . " - " . $heading;
   }
   echo $title;
   ?></title>
```

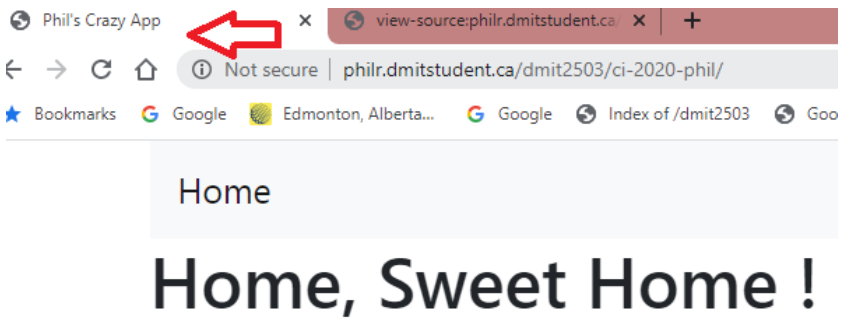
Ok. We should be getting our APP_NAME as the <title> now. However, we are also trying to get various sections of our app to also have more specific <title>'s.

Where was $heading used before? We used that in our Birds section.

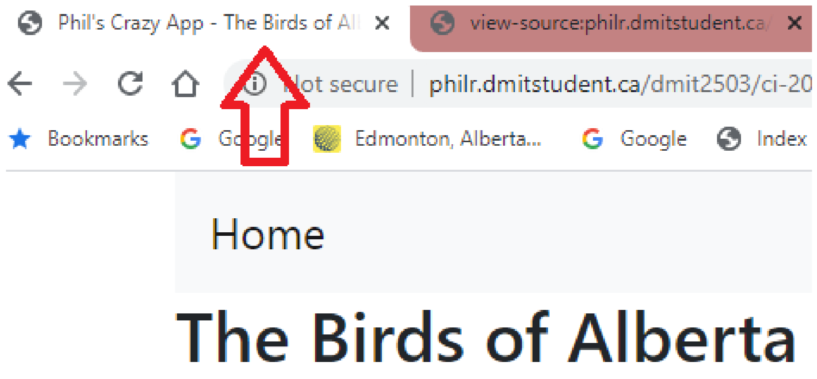
- So, why don't we see that $heading as part of our <title>?
- Because, when we loaded the header, we didn't pass that $data['heading'] array item to that view !
- Ok, let's fix that in Birds. Open controllers/Birds.php and add the following:

```
$this->load->view('includes/header',$data); // here we add the $data array so that the $heading is available to the header view
$this->load->view('bird_view',$data);
$this->load->view('includes/footer');
```

Now, let's test navigating from home to the birds pages:



Home, Sweet Home !


Phil's Crazy App - The Birds of All
view-source:philr.dmitstudent.ca/
Not secure
philr.dmitstudent.ca/dmit2503/ci-20
Bookmarks
Google
Edmonton, Alberta...
Google
Index
Home

The Birds of Alberta

We should now see a dynamic <title> tag. We can also use that APP_NAME constant in headings or anywhere else we like.

DMIT2503 – ADVANCED WEB CONCEPTS

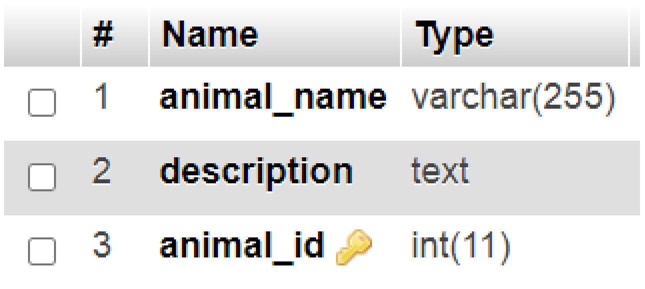
CRUD

Create a new Controller called Crud.php

```
// controllers/Crud.php

<?php
defined('BASEPATH') OR exit('No direct script access allowed');
class Crud extends CI_Controller {

        function __construct()
        {
            parent::__construct();
            $this->load->helper('form'); // loading this for the entire class/controller
            $this->load->library('form_validation'); // loading this for the entire class/controller
            $this->load->database(); // ummm…ditto
        }

        public function index()
        {
                echo "CRUD";
        }

}
```

Note that we now have a constructor. Make sure you have the 2 underscores in front. We are using this to load both a **library** and a **helper**. Both libraries and helpers are extensions of the core functionality. We will load them here as we need them for many of our CRUD functions, but not throughout the app.

Ok. Open phpMyAdmin and create the following simple table called **ci_animals**:



	#	Name	Type
☐	1	animal_name	varchar(255)
☐	2	description	text
☐	3	animal_id 🔑	int(11)

Then, using the **Insert tab in phpMyAdmin**, insert a couple animals of your choice: Input both a name and description.

Now, we will create our first Model. A Model is where we should be keeping our Database queries. In CodeIgniter, we can actually do DB queries anywhere, but for Best Practice, we should do them in our models.

Create a file in your /models/ folder called Crud_model.php

CRUD READ

```
// models/Crud_model.php
<?php
class Crud_model extends CI_Model {

  function __construct()
  {
    // Call the Model constructor
    parent::__construct();
  }

        function get_animals(){
                $query = $this->db->get('ci_animals');

                if ( $query->num_rows() > 0 ){
                        return $query->result();
                }else{
                        return FALSE;
                }

        }

}
```

Now, let’s get back to our Crud controller and grab some content from our DB.

```
// controllers/Crud.php
        public function index(){
                $data['heading'] = "Reading from a DB";
                $this->load->model('crud_model');
                $data['results'] = $this->crud_model->get_animals();

                echo "<pre>";
                print_r($data);
                echo "</pre>";
        }
```

We should see some content printed to the screen as we test this. Once again, we are using print_r() to visualize the array, and the <pre> tag just to keep some line breaks so it’s not one big mess.

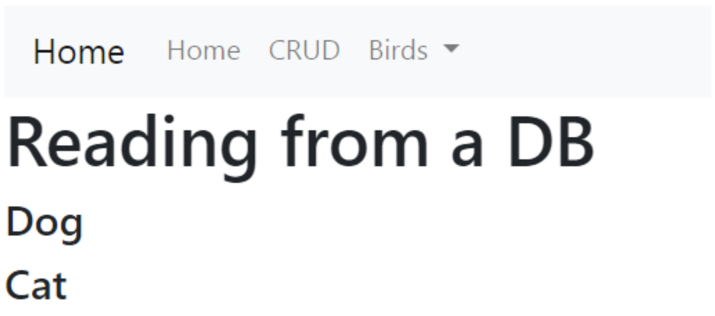
Now, let's move this data to a view and deal with it there.

Create a new view file called **crud_read_view.php**

```
// views/crud_read_view.php

<h1><?php echo $heading?></h1>

<?php if(($results)) : ?>
<?php foreach($results as $row): ?>
<div class="whatever">
                        <h4><?php echo $row->animal_name ?></h4>

</div>
<?php endforeach;?>
<?php endif; ?>
```

And change our controller to load this view (and the header/footer).

```
// controllers/Crud.php

public function index(){
            $data['heading'] = "Reading from a DB";
            $this->load->model('crud_model');
            $data['results'] = $this->crud_model->get_animals();

            $this->load->view('includes/header', $data);
            $this->load->view('crud_read_view',$data);
            $this->load->view('includes/footer');
}
```



Reading from a DB

Dog

Cat

So what are we doing here?

- We sent an array from the controller called $data. One item in that was called $results ($data['results'] = $this->crud_model->get_animals();) which we got back from the model.
    - Well, $result is itself an array. It's an array IN another array !!!

        - We call this a ***multidimensional array*** !
        - Yup. Mind = Blown
    - Once we get to the view, we can loop though that (we should test for it first).
    - We then use some OOP language and our Object Access Operator (aka, "arrow thingy") to grab each column name.

Ok. So why didn't we echo out the animal description here? Because we're going to have a **detail page** as well.

CRUD READ – DETAIL

Let's add a detailed read function to all our MVC components.

```
// models/Crud_model.php

function get_animal_detail($id){

		$this->db->where(animal_id', $id);
		$query = $this->db->get('ci_animals');
		if ( $query->num_rows() > 0 ){
				return $query->result();
		}else{
				return FALSE;
		}
}
```

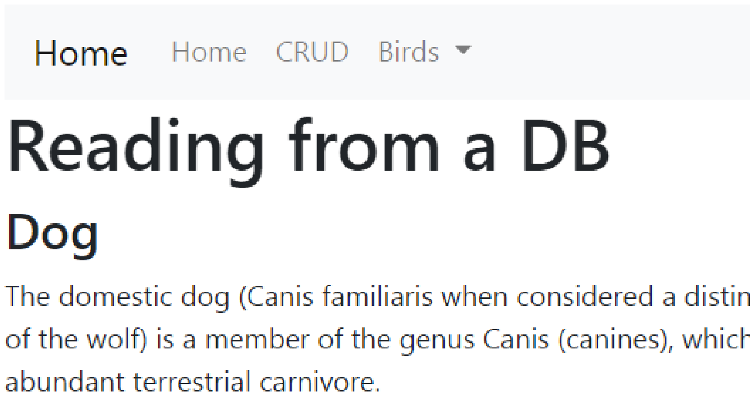
```
// controllers/Crud.php
public function detail($id)
		{
				/* We need to add some security and a "graceful exit: in case of a URL manipulation or other
error that prevents us from getting the required $id */
				if(!is_numeric($id)){ /* if this parameter is missing, or wrong format...*/
						/* best to just redirect*/
						redirect('/', 'location');

				}
				$this->load->library('typography');
				$data['heading'] = "Reading from a DB";
				$this->load->model('crud_model');
				$data['results'] = $this->crud_model->get_animal_detail($id);
				$this->load->view('includes/header',$data);
				$this->load->view('crud_detail_view',$data);
```

```
                $this->load->view('includes/footer');
        }// \ detail
```

```
// views/crud_detail_view.php

<h1><?php echo $heading?></h1>

<?php if(($results)) : ?>
        <?php foreach($results as $row): ?>
        <div class="well">

                <h3><?php echo $row->animal_name ?></h3>
                <p><?php echo $this->typography->nl2br_except_pre($row->description); ?></p>

        </div>
        <?php endforeach;?>
<?php else:?>
        <p>No results</p>
<?php endif; ?>
```

Whew! Ok, that's a lot of MVC files. So, how do we see all this?

In your address bar, type in the following after your base URL: **/crud/detail/1** (where 1 is the primary key value of one of your animals in the DB).

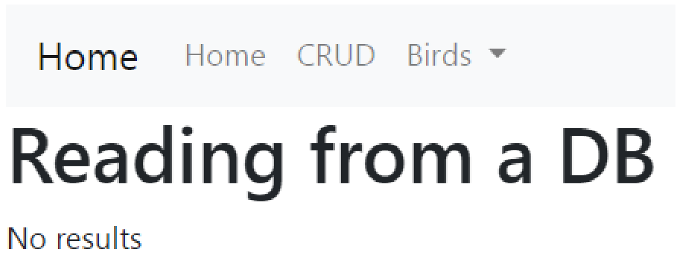


Reading from a DB

Dog

The domestic dog (Canis familiaris when considered a distin
of the wolf) is a member of the genus Canis (canines), which
abundant terrestrial carnivore.

And, it works ! Now try a primary key number that isn't in your DB:



Reading from a DB

No results

We should see a "No Results" message as we got that if the DB returns nothing for that value.

Now let's try hacking the URL. Try **/crud/detail/drop_from_db**

It should redirect us to the home page. We actually did that in the controller by testing to see if the 3rd segment is numeric. If not, then redirect this user anywhere.

Note: We are now seeing CodeIgniter's default segment handling.

- 1st segment: The controller (or class) name and it's index() function.
- 2nd segment: A function within that class.
- 3rd (and 4th) segments: Parameters that are passed to that function.

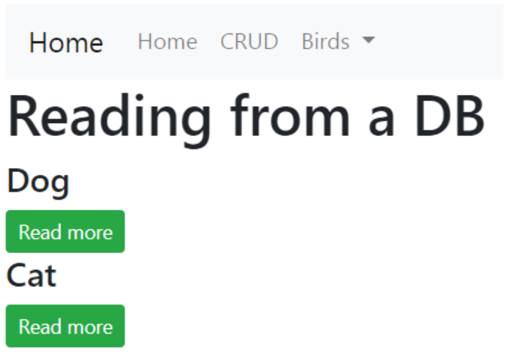
Take a look at the functions in our controller and model:

- **public function detail($id)** // controller: This reads the ID from the URL
- **$data['results'] = $this->crud_model->get_animal_detail($id);** // and sends that to the function in the Model
- **function get_animal_detail($id){}** // model : This then uses that ID to query the DB.

So, how do we get our UI to incorporate this?

ACTIVITY

On your own, go back to crud_read_view.php and have a Detail link that send the user to the correct detail page for that animal. You WILL need to use the ID.



Reading from a DB

Dog

Cat

SUMMARY

- One thing we didn't go over was the Typography library. Note that we loaded this in the controller so that we can use a **nl2br** type function in the view. Lots of little libraries and helpers in CI that you may learn as you go along.
- Models:
    - Note that in our controller, we load the model first: **$this->load->model('crud_model');**
    - Then, we access that by referring that new model : **$this->crud_model->**
    - And then, access one of that models functions: **$this->crud_model->get_animal_detail($id);**
    - We also set one array item to the results we get back from that DB query in the models functions: **$data['results'] = $this->crud_model->get_animal_detail($id);**
    - Remember: If you're having trouble visualizing an array, try **print_r($data)** to see it.

Enough for now. Next time, we'll start off with forms, validation, and then inserting, editing, and deleting DB info.