DMIT2503 – ADVANCED WEB CONCEPTS

CODEIGNITER MVC PHP FRAMEWORK – DAY 4

Overview: This lesson is continued from the CodeIgniter Lessons – Day 3.

CRUD - CREATE

VALIDATION LIBRARY

Ok. Last time we were able to Read from a database. We had a *read all* query that got all our animals, and a *detailed read* that got all info on only one animal.

Now, we'll move on with Create (putting info into our DB). We'll be working with:

- The same Crud controller, but a new function.
- The same Crud model, but a new function.
- A new view.
- A new library: Validation.

Let's start off by creating a new function in our Crud controller.

```
// controllers/Crud.php

public function write()
	{
	// Validation Library was loaded in the constructor.
	if ($this->form_validation->run() == FALSE)
	{
		$this->load->view('includes/header');
		$this->load->view('crud_write_view');
		$this->load->view('includes/footer');
	}
	else
	{

	}
} // \ write
```

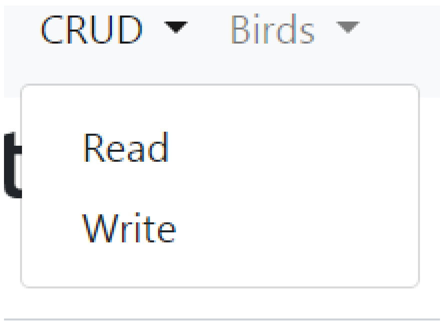
And a new view file:

```
// views/crud_write_view.php

<h1>New Animal</h1>
<?php echo form_open_multipart('crud/write'); ?>
<div class="form-group">
	<label for="animal_name">Animal Name</label>
```

```
		<input type="text" name="animal_name" class="form-control form-width" value="<?php echo
set_value('animal_name'); ?>"  />
		<?php echo form_error('animal_name'); ?>
</div>
<div class="form-group">
		<label for="description">Description</label>
		<textarea name="description" class="form-control form-width textarea-height"><?php echo
set_value('description'); ?></textarea>
		<?php echo form_error('description'); ?>
</div>
<div class="form-group"><input type="submit" value="Submit" class="btn" /></div>
</form>
```

And, make a link in your nav to your crud/write function: Here, I'm using a BS4 navbar dropdown item:
<a class="dropdown-item" href="<?php echo base_url()?>crud/write">Write</a>



Don't expect anything to happen here. We're just framing some stuff and have a way to go yet.

Let's rundown what we're setting up here:

- **Form Helper:** Last class we loaded the form helper in the class constructor*($this->load->helper('form');).* This way it is available to all functions in that class.
    - **View**: Note the line that starts the form: *<?php echo form_open_multipart('crud/write'); ?>*
        - This creates a <form> tag with the necessary attributes. Take a look at the HTML code it writes by Viewing Page Source in your browser.
        - *<form action="http://philr.dmitstudent.ca/dmit2503/ci-2020-phil/crud/write" enctype="multipart/form-data" method="post" accept-charset="utf-8">*
        - As we can see, we have the action of the form (in CI's *class/function* segment based nav).
        - We have added the *enctype="multipart/form-data"* in case we want to add file uploads (but not necessary in this case).
        - Method is POST.
- **Validation Library:** This was also loaded in the class constructor *($this->load->library('form_validation').*
    - Note the *if statement* in the controller function. Here we will test to see if validation has passed or not.

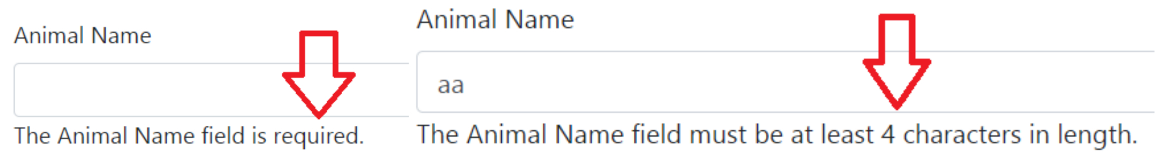
- If the run() method == FALSE, then validation has not passed (either we're just loading the page or else user has not filled out everything correctly).
- The else part of the if statement means that validation has passed and we can insert some data.
- **View**: We have a couple things in our view that are part of the Validation Library:
  - *<?php echo set_value('description');* // this will prepopulate the form
  - *<?php echo form_error('description'); ?>* // this will write any validation errors.

Ok. Let's add some validation rules and see the validation in action:

```
// controllers/Crud.php

public function write()
	{
	$this->form_validation->set_rules('animal_name', 'Animal Name',
'required|min_length[4]|max_length[40]');
	$this->form_validation->set_rules('description', 'Description',
'required|min_length[20]|max_length[2000]');
		if ($this->form_validation->run() == FALSE)
		{
			$this->load->view('includes/header');
			$this->load->view('crud_write_view');
			$this->load->view('includes/footer');
		}
		else
		{
			echo "SUCCESS";
		}
	} // \ write
```

Run that and see what you get:


Animal Name

The Animal Name field is required.

Animal Name

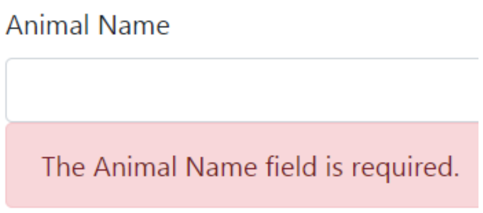
aa

The Animal Name field must be at least 4 characters in length.

Ok. We can see the validators doing their job. Now, let's add some styling with Bootstrap. Add this to the top of your write() function.

```
// controllers/Crud.php
public function write()
	{
	$this->form_validation->set_error_delimiters('<div class="alert alert-danger">', '</div>');
	$this->form_validation->set_rules('animal_name', 'Animal Name',
'required|min_length[4]|max_length[40]');
```

Ok. Run that again.


Animal Name

The Animal Name field is required.

Now we have added some HTML <div>'s with a little Bootstrap love around our error messages. Much better!

Now try testing your form and getting through your validation rules. Should see the SUCCESS message we are using to test.

Ok, let's have the form actually put something in our DB if validation has padded. Add the following to your controller function:

```
// controllers/Crud.php

    else
                {
                        //echo "SUCCESS";
                        // retrieve POSTED form data
                        $data['animal_name']  = $this->input->post('animal_name');
                        $data['description']= $this->input->post('description');

                        $this->load->model('crud_model');
                        $this->crud_model->insert_animal($data);


                }
```

And add a function to your Model.

```
// models/Crud_model.php

function insert_animal($data){
        $this->db->insert('ci_animals', $data);

}
```

Run that and then take a peek at your DB (in either phpMyAdmin, or CRUD Read page).

We should have now inserted new data. You may have noticed that we left the user starting at a blank white screen though. Let's go fix that with a simple redirect for now.

Add this as the last line in your crud/write controller function:

```
// controllers/Crud.php
    else
        {
                $data['animal_name']  = $this->input->post('animal_name');
                $data['description']= $this->input->post('description');
                $this->load->model('crud_model');
                $this->crud_model->insert_animal($data);
                redirect("crud/index", 'location');
        }
```

Voila: After the successful insert, we redirect the user to the read function where they can see their new animal.

SUMMARY

- The validation library offers many easy and useful validators. We have only seen the *string length* and *required* validators so far, but there are many more to learn. https://codeigniter.com/user_guide/libraries/form_validation.html
- Our DB insert (Crud_model). With just one line of code *($this->db->insert('ci_animals', $data)* we are able to insert data into our DB. But how is this possible?
    - Using CI's Query Builder class, we can do an insert with one simple line of code.
    - https://codeigniter.com/user_guide/database/query_builder.html#inserting-data
    - We start by populating an array (*$data* but you can call it anything), moving that to the Model function as an argument (parameter), and then the Insert statement will put each array item into a DB field **of the same name.**
    - *In order for this to be this simple, please try to call your form elements, array items and DB fields the same thing.*
    - CodeIgniter can also utilize actual MySQL queries if you choose. The Query Builder is a shortcut to actual queries.

THE SESSION LIBRARY

Ok. Let's pretty things up a bit and start learning a bit about CI's session library.

First of all, what we would like to do with the session library is going to be global, so we need to load it in our *config/autoload.php* file. Around line 61, add this: **$autoload['libraries'] = array('session');**

Now, let's add a nice little message to the user anytime they do something, like inserting a new animal.

Add this as the last line in your crud/write controller function:

```
// controllers/Crud.php
    else
        {
                $data['animal_name']  = $this->input->post('animal_name');
                $data['description']= $this->input->post('description');
                $this->load->model('crud_model');
                $this->crud_model->insert_animal($data);

                $this->session->set_flashdata('message', 'Insert Successful');

                redirect("crud/index", 'location');
        }
```

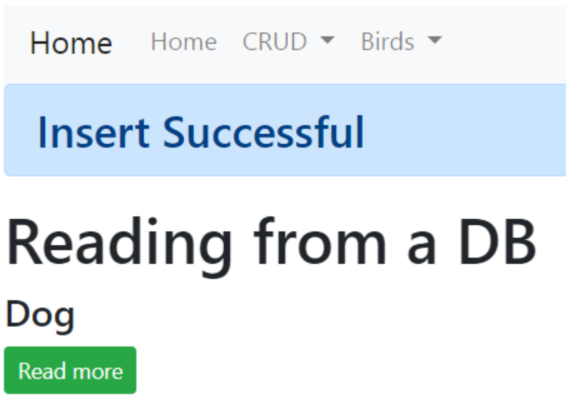
Then, we will add something to our global view (includes/header.php) that will reflect our message to the user. Add the following to the very end of your header file:

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

<?php $message = $this->session->flashdata('message'); ?>
<?php if ($message): ?>
<h3 class="alert alert-primary"><?php echo $message ?></h3>
<?php endif; ?>

<!-- This ends the header include -->
```

Now try a successful insert.



Insert Successful

Reading from a DB

Dog

And we should see our message.

Now try navigating to another page and notice that the message is gone. Using the session library, we have utilized the **flashdata** method (https://codeigniter.com/user_guide/libraries/sessions.html) .

This will set a session variable for the following one pageload only. Very handy for setting a message like this. And, we can re-use this again and again since it echo's out in our header which is global. So, anytime we want to set this, we use the *$this->session->set_flashdata('message', 'Insert Successful')* and then redirect the user *(redirect('crud/index', 'location')*. The message will show at the redirected page view and then disappear.

SOME MORE PRETTINESS

Ok. Let's add a couple little client-side things to pretty up our app.

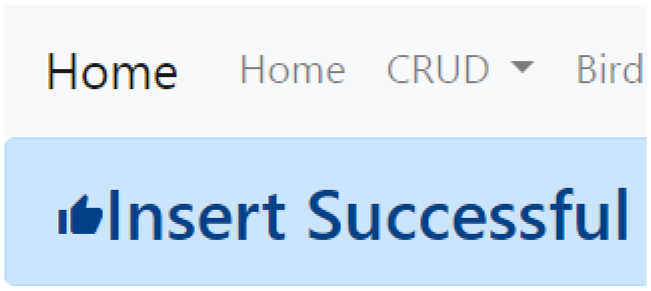
ICONS

First, let's incorporate some icons. Whereas I used to use Font-Awesome, they are going to more of a paid model and don't advertise their CDN service anymore. So, let's try some Google icons:

Add this to your header: *<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">*

Now, let's put an icon in our Message prompt:

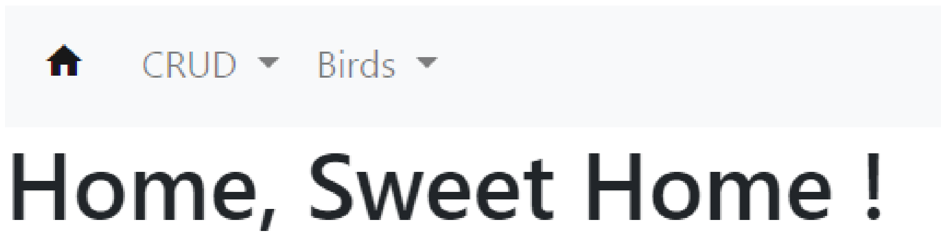
```
// views/includes/header.php
<h3 class="alert alert-primary"> <i class="material-icons">thumb_up</i> <?php echo $message ?></h3>
```



Good.

Now, let's make our Home link an icon as well. Add this if you're using Bootstrap 4:

<a class="navbar-brand" href="<?php echo base_url()?>"> <i class="material-icons">home</i></a>



SOME JQUERY PRETTINESS

And finally, let’s incorporate a bit of jQuery for a simple fade of our Message.

If you are using the basic Bootstrap template like I am, first remove the .slim version of jQuery and replace it with the full version.

Find the jQuery in your footer (or header) and make sure it is the **full version**.
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

Now, add a <script> tag, a document ready, and test your jQuery readiness.

```
// views/includes/header.php
<script type="text/javascript">
     $(document).ready(function(){
       console.log('ready');

     });
   </script>
```

Now, go back to your Message and give it an ID:
<h3 class="alert alert-primary" **id="message"><**i class="material-icons">thumb_up</i><?php echo $message ?></h3>

```
// views/includes/header.php
<script type="text/javascript">
     $(document).ready(function(){
       // fade #message if exists
       if($('#message').length){
          $( "#message" ).delay(3000).fadeOut({}, 3000);
       }

     });
   </script>
```

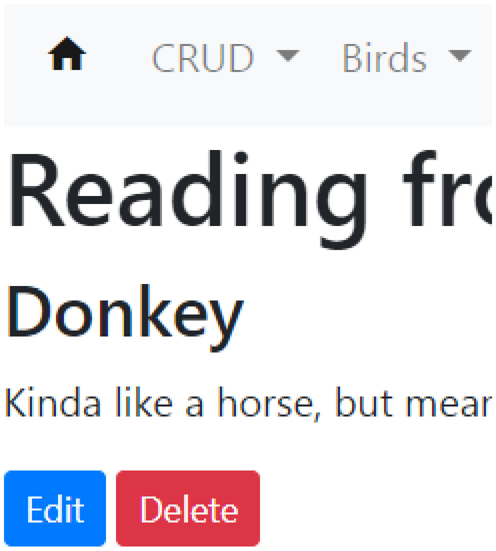
Test that. Ooooohhhhh !

ACTIVITY FOR NEXT CLASS

Next class, we will finish off our CRUD learning with some Update and Delete. Here’s some steps to get started on that:

1. In your Crud controller, create a new function called Edit(). You WILL need to grab the ID of whatever item you wish to edit, so it should look something like this: **public function edit($id){}**

2. In your detail view of an item, add both an Edit and a Delete link (or button). Each of these will link to their respective functions in the Crud controller and will need an ID (like detail).



Reading fr

Donkey

Kinda like a horse, but mear

3. See how far you can get with your edit function.

See you next time folks. Wash your hands and stay safe. ☺