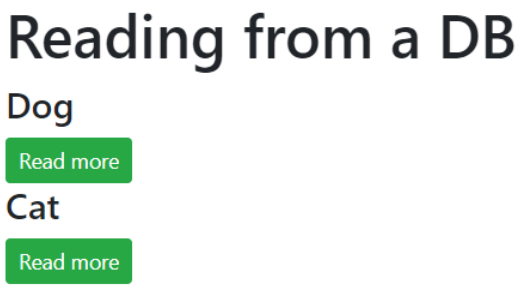
DMIT2503 – ADVANCED WEB CONCEPTS

CODEIGNITER MVC PHP FRAMEWORK – DAY 5

Overview: This lesson is continued from the CodeIgniter Lessons – Day 4.

To get started, did anyone notice my mistake from last classes? Well, not really a huge mistake, but something I noticed as I was testing. Let's take a quick look at my data.



Well, what the problem? Here's my validation for insert:
$this->form_validation->set_rules('animal_name', 'Animal Name', 'required|**min_length[4]**|max_length[40]');

Ok. I set animal_name minimum length as 4, but I have both **Dog** and **Cat** in my DB. Of course, I inserted both of those as testing data in phpMyAdmin, so validation wasn't an issue. Since both animals are legit, I have learned from my testing and changed min_length to 3.

What's the point? *Always test with real data.*

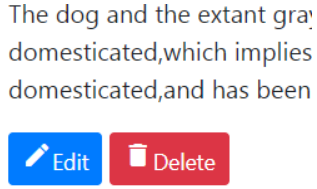
ACTIVITY FROM LAST CLASS – DETAIL VIEW WITH EDIT/DELETE BUTTONS:

Last class, I asked peeps to go back to their detail view and add buttons for both edit and delete. Let's see that before we move on:

```
// views/crud_detail_view.php.

<a href="<?php echo base_url() ."crud/edit/" .$row->animal_id';?>" class="btn btn-primary btn-sm"> <i class="material-icons">edit</i>Edit</a>
<a href="<?php echo base_url() ."crud/delete/" .$row->animal_id';?>"  class="btn btn-danger btn-sm"> <i class="material-icons">delete</i>Delete</a>
```

Note that once again, I am using the Google material icons. Feel free to research this a bit to change size or color.



DMIT2503 – ADVANCED WEB CONCEPTS

CRUD – UPDATE(EDIT)

Ok. Last time we were able to Insert (Create) into a database.

Now, we'll move on with Update, but we'll call this Edit. As we know from other PHP based apps we have done, an edit feature will share much of the functionality as both a detailed read (info from a single item) as well as an insert (form data submitted to the DB).

So, to start with, we can raid our Insert function for some copy/paste code, but make sure we take out the actual insert function we call in the model.

```
// controllers/Crud.php

public function edit($id) // we change the name to edit from write and we add the $id parameter
{
        $this->form_validation->set_error_delimiters('<div class="alert alert-danger">', '</div>');
                $this->form_validation->set_rules('animal_name', 'Animal Name',
'required|min_length[3]|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_edit_view'); // a new view here
                        $this->load->view('includes/footer');
                }
                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($data); // make sure we remove this from our
copy/paste of write
                        // and comment out or remove any redirects, etc. so we can see errors as we test
                        //$this->session->set_flashdata('message', 'Insert Successful');
                        //redirect('crud/index', 'location');
                }
} // \ edit
```

And we create a **new view**. Copy/paste crud_write_view to start. **But…do NOT click the button if you test.** We have some work to do.

```
// views/crud_edit_view.php. This has been copied from crud_write_view
<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>
```

So. Our first order of business is to start populating the form fields. Also not that in the above view code, we haven't yet changed our form action, so if we click the button (I did tell you not to didn't I), you are redirected to write. But if we change the action, we need to set the ID.
How do we populate the fields? The same model function we used to create our detail page.

Let's add a few things to our controller.

1. First, let's move the loading of the model up because we will need it before we actually update the DB.
2. Use the same model function we had in *detail* to grab the existing data.
3. Move that to the view to populate the form fields.
4. While we're at it, lets add a little security to the top of the function to check that a correct numeric ID has been provided.

```
// controllers/Crud.php
public function edit($id)
{
        if(!is_numeric($id)){
                redirect('/', 'location');
          }
                $this->form_validation->set_error_delimiters('<div class="alert alert-danger">', '</div>');
                $this->form_validation->set_rules('animal_name', 'Animal Name',
'required|min_length[3]|max_length[40]');
                $this->form_validation->set_rules('description', 'Description',
'required|min_length[20]|max_length[2000]');

                $this->load->model('crud_model'); // just moved this up so we can call the model function

      if ($this->form_validation->run() == FALSE)
                {
                        $data['results'] = $this->crud_model->get_animal_detail($id);
                        $this->load->view('includes/header');
```

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

Now, back to the view and let's see the existing data. Add the following to the top and see the animal name to test.

```
// views/crud_edit_view.php.
<?php
if($results){
        foreach($results as $row){
                $animal_name = $row->animal_name;
                $description = $row->description;
                $id = $row->animal_id;
               echo $animal_name; //just to test
        }
}
```

Ok. If that works, we can:

1. Set the action of the form with the correct ID.
2. Populate the form fields.

In your view, change the action of the form and add the ID. Note that **we must switch to double quotes** in order to use a variable ($id) within a text string. If using single quotes, we would have to append the value to the string.

```
// views/crud_edit_view.php.
<?php echo form_open_multipart("crud/edit/$id"); ?>
```

Try submitting the form, and make sure the URL stays the same (crud/edit/{number}).

Now, let's populate the form fields. Add the following 2nd parameter to your *set_value()* in your form fields:

```
// views/crud_edit_view.php.
<input type="text" name="animal_name" class="form-control form-width" value="<?php echo
set_value('animal_name',$animal_name); ?>" >
```

Do the same to your description field and test. We should now have the correct data for whichever animal you are editing.

THE ACTUAL UPDATE

Once we have the form fields populated with the existing data, we now have to grab the form values when the user submits the form, check they are valid (validation), and then move all the data to the model to do the update.

The model will need 2 things in order to update:

1. The $data[] array with the form data.
2. The $id variable.

```
// models/Crud_model.php
function edit_animal($data,$id){
                $this->db->where('animal_id', $id);
                $this->db->update('ci_animals', $data);
        }
```

And we can call this from our controller:

```
// controllers/Crud.php
        $data['animal_name']  = $this->input->post('animal_name');
       $data['description']= $this->input->post('description');
         $this->crud_model->edit_animal($data,$id);
```

If we run this, we should see the DB data change, Well, the first thing we might see is a blank screen. So make an edit you can remember, then run over the *crud/detail* and check to see the new data.

If that works, then we can add our status message, and a quick redirect at the end of that if statement.

```
// controllers/Crud.php

        $this->session->set_flashdata('message', 'Edit Successful');
       redirect('crud/edit/' . $id, 'location');
   }
}
```

Note that here I have elected to redirect to the same page (edit) with the same ID. Also not that this time, I have appended the ID to the 2 segments ('crud/edit' . $id) and stayed with the single quotes.

Try that and see if your edit page works as expected.

Note that I have not supplied the complete code to be copy/pasted. I think students should be able to follow the steps and figure things out as they go.

CRUD - DELETE

ACTIVITY - ON YOUR OWN…

1. Please compete a delete function.
    - Don't need a view. The user can access this from the detail page.
    - 1 controller function

- 1 model function.
- You will need the ID.

2. Do something (anything you like) with your home page. Feel free to just design straight to the home_view, or else you can pass some data from the controller to the view. No need for any DB stuff here though.
3. Take a look at Ion Auth: http://benedmunds.com/ion_auth/
    - This is an authentication library for CI. I have used an older one in the past (Tank Auth) and had upgraded it, but it doesn't work for PHP 7+, so time to move on.
    - We will integrate that next class.

See you next time folks. Expect that Lab 7 will be given out next week sometime.