FIT2104 Web Database Interface
Week 11 Lab Exercises

CakePHP Configuration, Model and Controller

Objectives

On completion of this tutorial you should have:

1. Configure CakePHP project
2. Completed tasks on manipulating Model-related features
3. Completed tasks on manipulating Controller-related features

PART 0: Complete the CakePHP project in Lab Week 10

To complete the tasks in this lab exercise sheet, you'll need a complete CakePHP project from last week's lab exercises. You can either:

- **Continue your work on the same CakePHP project used in week 10**
  However, you'll need to ensure that your schema is fully compatible with CakePHP conventions and codes are generated by Bake and working as expected (including all relationships between tables, especially the many to many relationship and joinTable [furniture_types_rooms])
- **OR, start from the demo project of week 10 labs**

Hint: The `.gitignore` file from CakePHP app skeleton will automatically leave out `/config/app_local.php` and `/vendor` when committing files. Therefore if you're cloning a CakePHP project new onto your computer, you'll need to execute

```
composer install
```

command to install missing vendor libraries, temporary folders and local configuration files.

Also, don't use Bake to generate/regenerate codes throughout this lab

Once you're settled, continue...

PART 1: Configure CakePHP project

Familiarise yourself with various configuration options in CakePHP by:

1. **Change the locale and timezone of the project**

> *Before starting, make sure to check if your DBMS (either MySQL or MariaDB) has timezone data. This can be checked by converting the current time into a timezone. Execute the following query:*
>
> ```
> SELECT CONVERT_TZ(now(), 'UTC', 'Australia/Melbourne');
> ```
>
> *If you're getting `NULL` as result, follow the instructions here for macOS to resolve the issue. This should not be an issue on Windows or Linux. Consult tutors if you need help.*

CakePHP app skeleton is by default using United States formats and UTC time zone. Check pages with date and time, the room booking index for example, and you may notice that the date is in month/day/year form, which can be annoying for Australian users.

Once the database is confirmed to support timezone, open `/config/app.php` file of your CakePHP project, change the following attributes so that your CakePHP project is using Australian (and Melbourne) formats and time zone when processing data:

- `App[defaultLocale] - change from en_US to` **`en_AU`**
- `App[defaultTimezone] - change from UTC to` **`Australia/Melbourne`**
- `Datasource[default][timezone] - change from UTC to` **`Australia/Melbourne`**

Now check any pages with date and time data - you may noticed that the format is not "correct"

2. **Configure email sending profiles (overriding an attribute)**

In CakePHP there are two app configuration files: `app.php` and `app_local.php`. Any values that exist in `app_local.php` will override its counterpart in `app.php`. This allows no sensitive data to be committed into the git repository, and different deployment of the same codebase can have different settings.

By default, how CakePHP sends email (i.e. which email transport method) is only configured in `app.php`, where the key EmailTransport[default][className] controls its behaviour. You'll need to override this value in the `app_local.php` file, and change its value from the default value (which is `MailTransport::class`) to Debug Mail Transport instead. Make sure the overridden attribute also exists in `app_local.example.php` file so it will propagate to future-generated local configurations.

3. **Make and modify custom routes**

Another important CakePHP configuration file is `/config/routes.php`. This file controls how URLs are translated and routed in a CakePHP application.

Open the file, analyse the existing routes provided along with their comments, and:

- Change the home page (i.e. base path of the website) from `PagesController::display` to the `index` action of the `RoomBookingsController`, without any additional argument
- Route `/foobar` to bar action of FoosController, with one argument of "placeholder"

Test those changes and see if anticipated pages or error messages are displayed.

PART 2: Manipulating CakePHP Models

For this part you'll need to complete the following tasks related to CakePHP Models:

1. **Create a new field in the database and modify the related Model to reflect the change**

First, add a new column in the "Rooms" table named "general availability" (ensure to follow CakePHP conventions) with a boolean (tinyint(1)) type that cannot be null, and default to true (or 1).

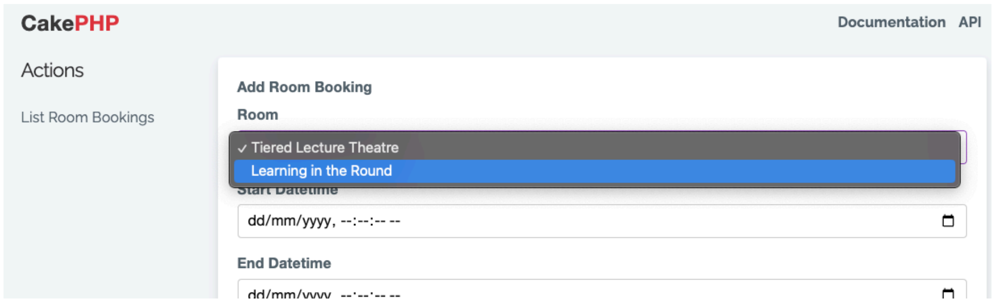
Once completed, clear the CakePHP schema cache so that the system knows the change:

```
bin/cake cache clear_all
```

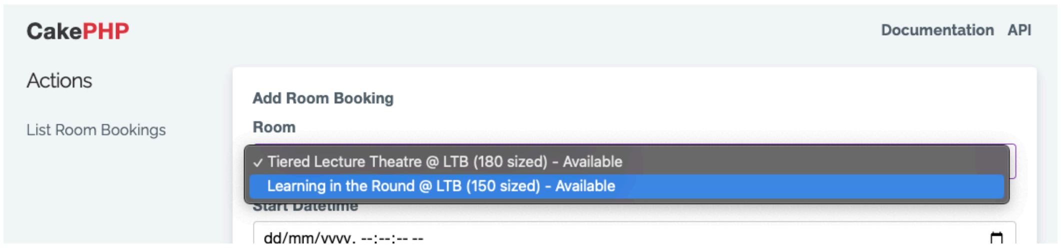
Modify Room Model files (both entity and table files) and add the new column accordingly. Ensure that the new field is protected from `newEntity()` or `patchEntity()` while mass assigning values, and the related validators are also being added.

2. **Create a virtual field and make it the display field**

By default, when creating a new room booking record, only the room name is displayed when selecting a room for the booking record:



Now, using the accessor and mutator feature in the CakePHP entity class, create a virtual field that aggregates the name, building, capacity and general availability of a room, and use it as the display field of Rooms table. Make sure you're giving the field a meaningful name. Once completed, the dropdown of rooms may be look like this:



PART 3: Manipulating CakePHP Controllers

For this part you'll need to complete the following tasks related to CakePHP Controller:

1. **Change how Room Bookings listing is sorted**

By default, a model's listing page is not sorted. Modify the existing index() action function in RoomBookingsController so it is sorted by descending order in the end date/time column.

This can be achieved by adding more/building up the existing query used in the function. Once completed, check "Sql Log" in the DebugKit toolbar and see if the SQL query generated by CakePHP ORM has been modified successfully.

Hints:

1) `Query::order()` is deprecating, and `Query::orderBy()` is preferred in CakePHP 5 instead;
2) When referring a column, it's recommended to add the model name along with the column name, like "`RoomBookings.end_datetime`" instead of only using "`end_datetime`"

2. **Create a action function to change the availability of a room**

Create a new action function in `RoomsController` that takes the primary key of a room as the only argument, and then flips the "general availability" column of that record. Save the changes, and send the user back to the detail (view) page of the record.

Hints:

1) You can use the `delete()` as a starting point - both functions take the same arguments, and do similar things, however they are running a very different action, and upon completion of the task, the user goes to a different page
2) In part 2, the accessibility of “general availability” column in Rooms model was set to false, meaning the field can only be updated in the controller programmatically (assigning values directly), instead of using a client-side form (using `patchEntity()`)
3) You can read the original value of the field by accessing the entity object once it’s been retrieved using Model
4) Don’t forget to save the record after making changes. Definitely don’t delete the record
5) Since view templates are not yet updated to accommodate the new column yet, you won’t see the changes immediately. You can check if your new function does its job by either retrieving the record from the database in phpMyAdmin directly, or inspecting the variable (object) that contains the entity using the DebugKit toolbar

Student Evaluation on Teaching and Units (SETU)

Please complete the SETU survey for FIT2104 when you have time - Thank you!

https://lms.monash.edu/mod/page/view.php?id=5667603