

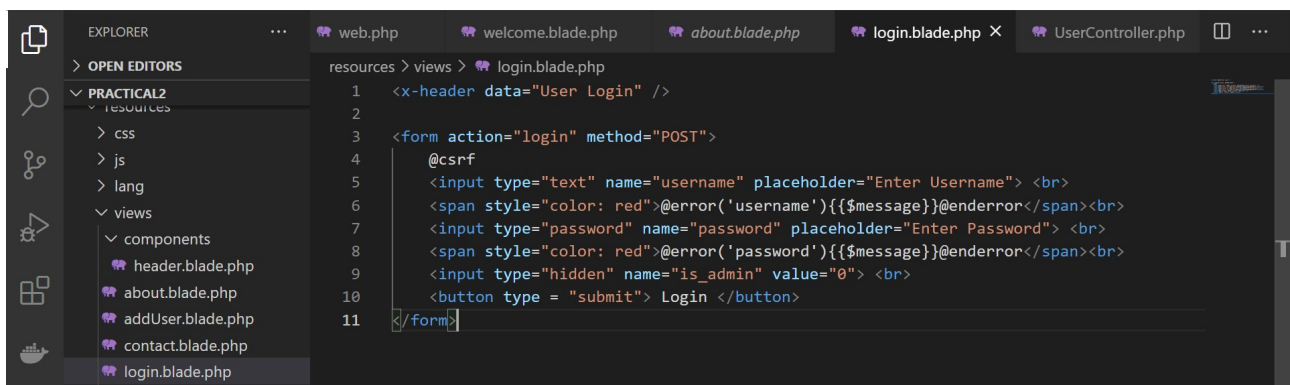
Practical 5 : Maintaining State

In this lab, we will explore the concept of maintaining state in Laravel. Previously, we learned that data can be passed from a form using “POST” method into a HTTP request. With the existing knowledge of form creation and passing of data in HTTP request, let’s explore the concept of keeping the data in a state until the state being deliberately changed in the following practical session.

1. Session.

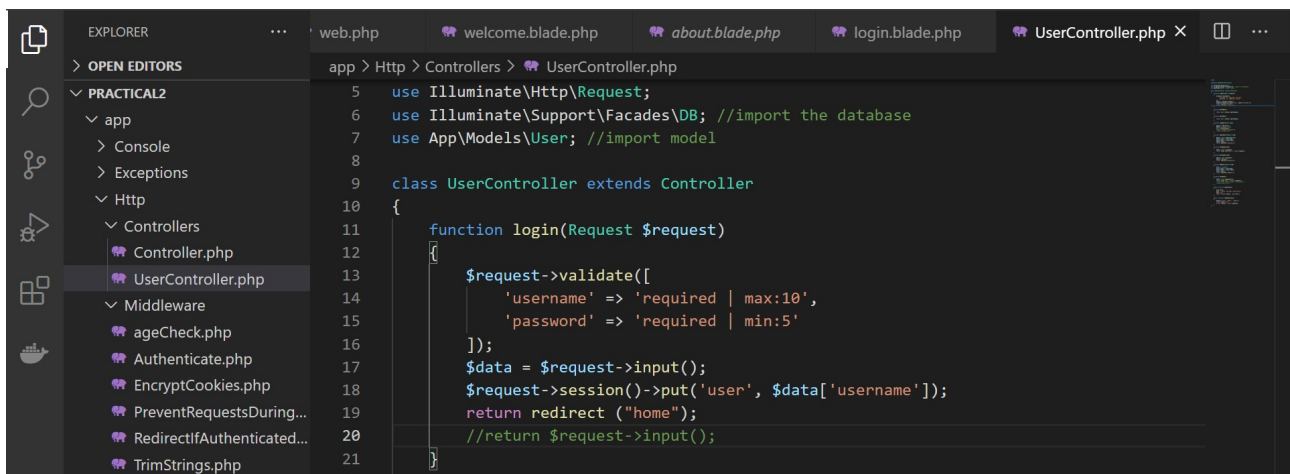
It is inevitable that a stateless web application be developed into a platform with states. For instance, once a user logged in the web application, the state of the web application should change to become “User Logged In”. While this is the web application’s state, the user’s profile should be “viewable” within web pages that the user interact within the web application until the user deliberately log out from the web application and change the state to “User Logged Out”. Such storage of user profile or information during the state of user’s login are stored in backend server and it could be done using session.

In order to explore the concept of session, let’s modify the login form, login controller and login routes created in the previous lab, as shown in Figure 1, 2 and 3.



```
1 <x-header data="User Login" />
2
3 <form action="login" method="POST">
4     @csrf
5     <input type="text" name="username" placeholder="Enter Username"> <br>
6     <span style="color: red">@error('username'){{ $message }}@enderror</span><br>
7     <input type="password" name="password" placeholder="Enter Password"> <br>
8     <span style="color: red">@error('password'){{ $message }}@enderror</span><br>
9     <input type="hidden" name="is_admin" value="0"> <br>
10    <button type="submit"> Login </button>
11 </form>
```

Figure 1: User Login Form.



```
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB; //import the database
7 use App\Models\User; //import model
8
9 class UserController extends Controller
10 {
11     function login(Request $request)
12     {
13         $request->validate([
14             'username' => 'required | max:10',
15             'password' => 'required | min:5'
16         ]);
17         $data = $request->input();
18         $request->session()->put('user', $data['username']);
19         return redirect ("home");
20         //return $request->input();
21     }
22 }
```

Figure 2: Login Controller.

UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

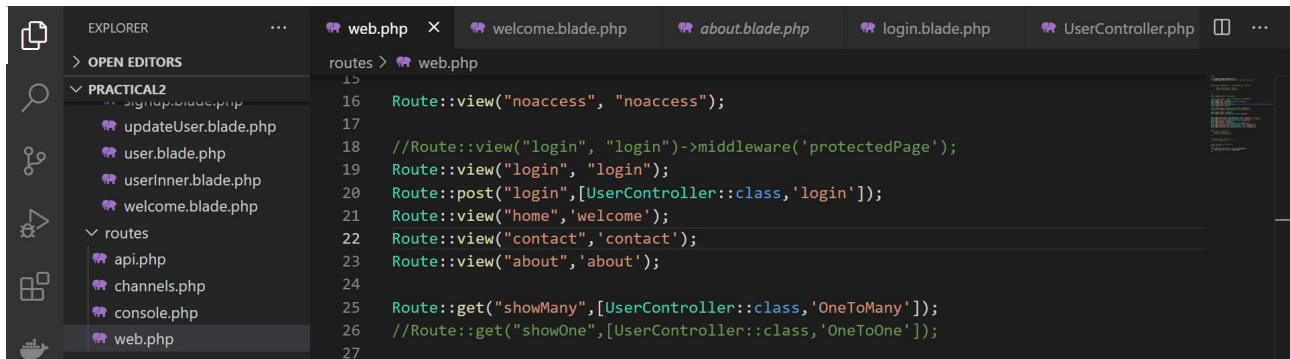


Figure 3: Login Routes.

After the modification scripts, let's modify the scripts in “welcome” blade template in order to display username as a greeting on the page and include the anchor to pages such as “about” and “contact” as shown in Figure 4.

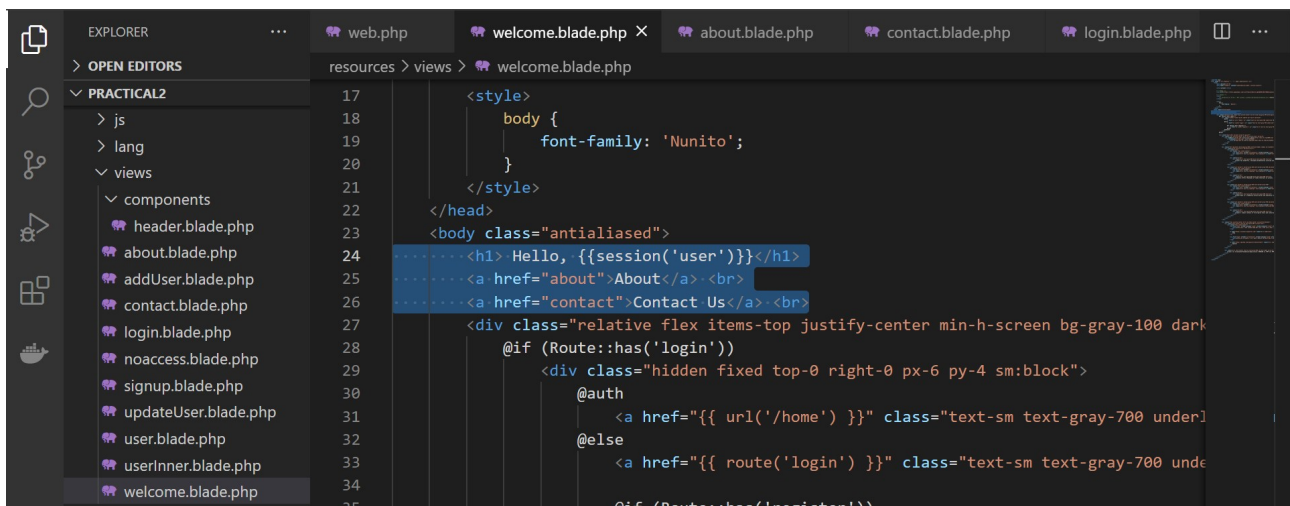


Figure 4: Modification to “welcome” blade template.

Host the web application and perform login. See that when a user login to the web application, the user will be brought to the “welcome” page with the username shown by the call from session, as illustrated in Figure 5.

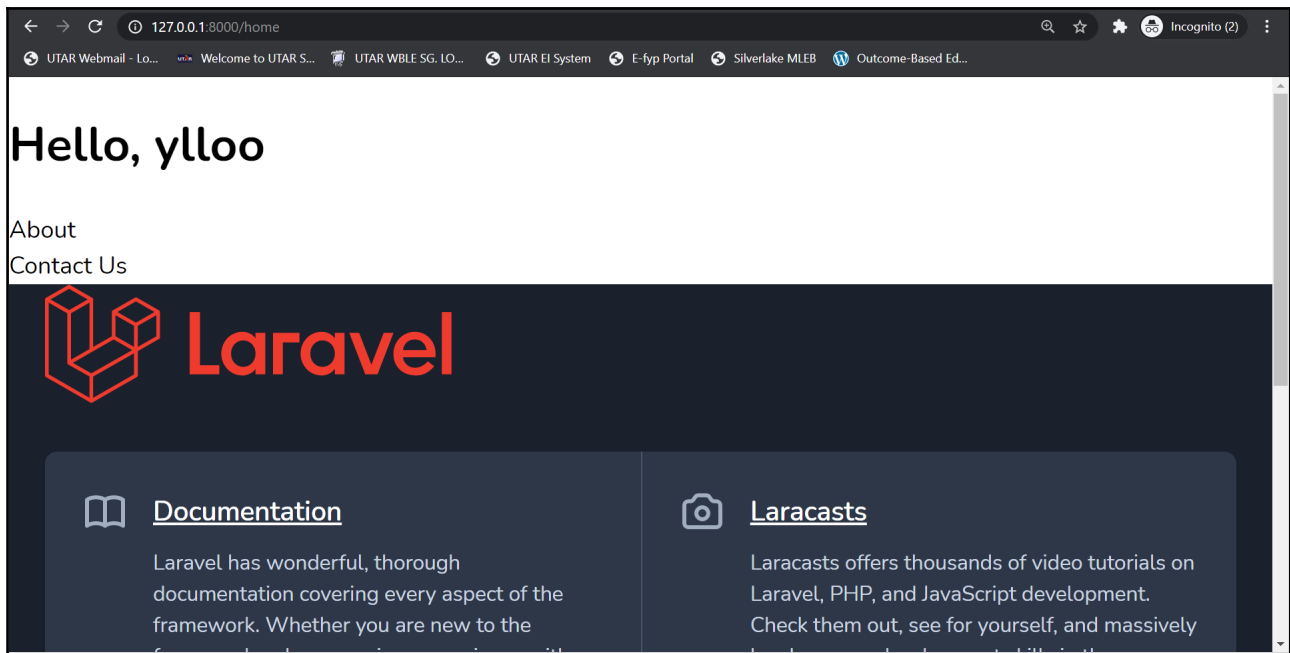


Figure 5: Redirection to login page with user's username greetings.

Exercise:

Modify scripts in “about” and “contact” blade template as well so that it will show the logged in username even if the user refresh the page or going back and forth “welcome”, “about” and “contact” page.

UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

Now, let's go to "login" page once again. It shouldn't be like this, isn't it? User should not be able to go to the web application's login page before log out. In other words, once the web application is in "User Logged In" state, "login" page should be inaccessible to the user.

Thus, let's create a "logout" anchor and "logout" route for the "welcome" page in order to pass a "User Logged Out" state to the web application. Modify the login route to only allow user to access login page when the state is "User Logged Out" as shown in Figure 6.

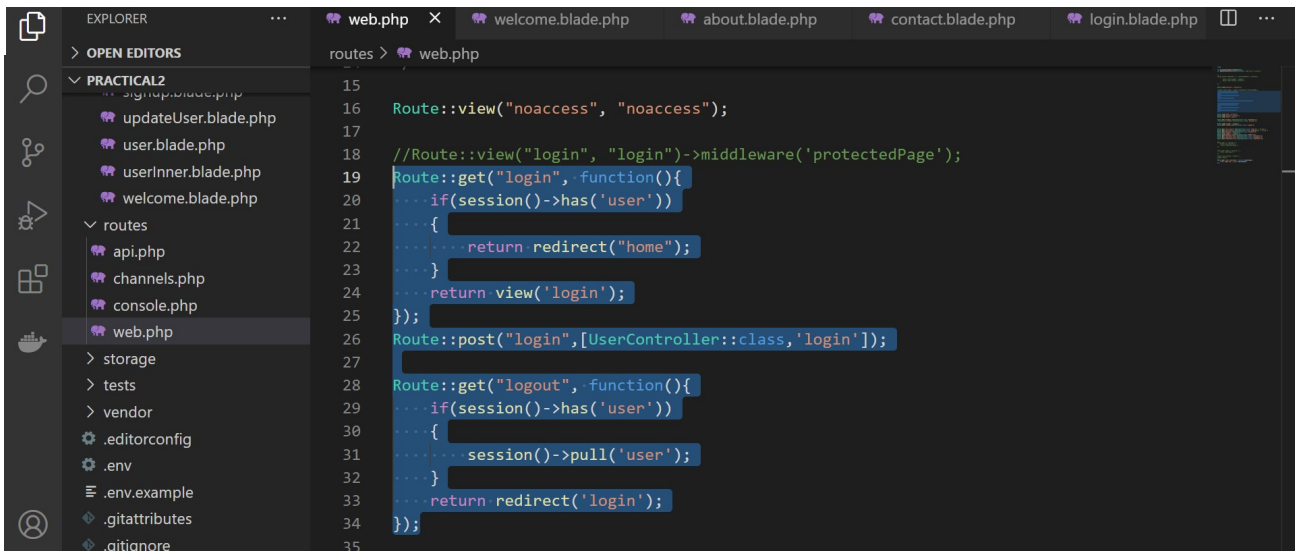


Figure 6: "User Logged In" and "User Logged Out" states.

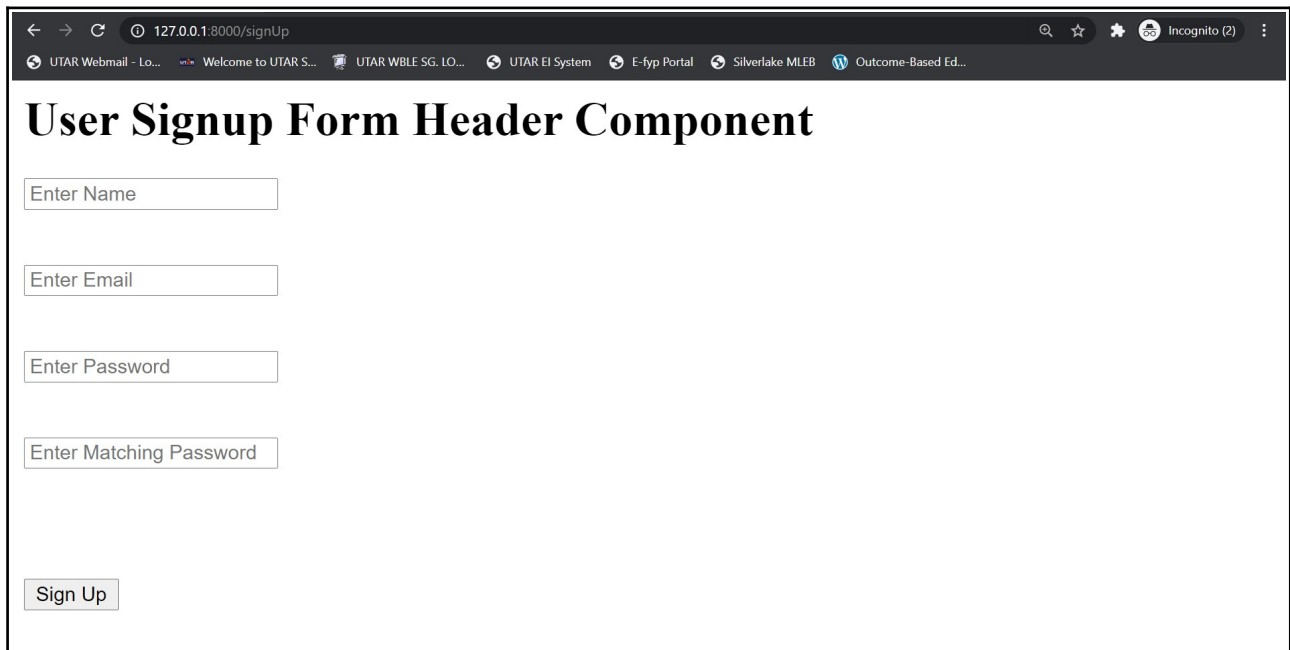
2. Flash Session.

In order to store items in the session just for the next request, one may do so using the flash method. Data stored in the session using this method will be available immediately and during the subsequent HTTP request. After the subsequent HTTP request, the flashed data will be deleted. Flash session data is useful for registering a new user to the web application; responding to the user with some of the registered data as a proof of successful registration; in other words, a confirmation message.

Session data is distinguished from flash data by simply assigning "flash" method to the assignment of session instead of "put" method as learned in previous practical session. Let's explore the concept by the exercises below:

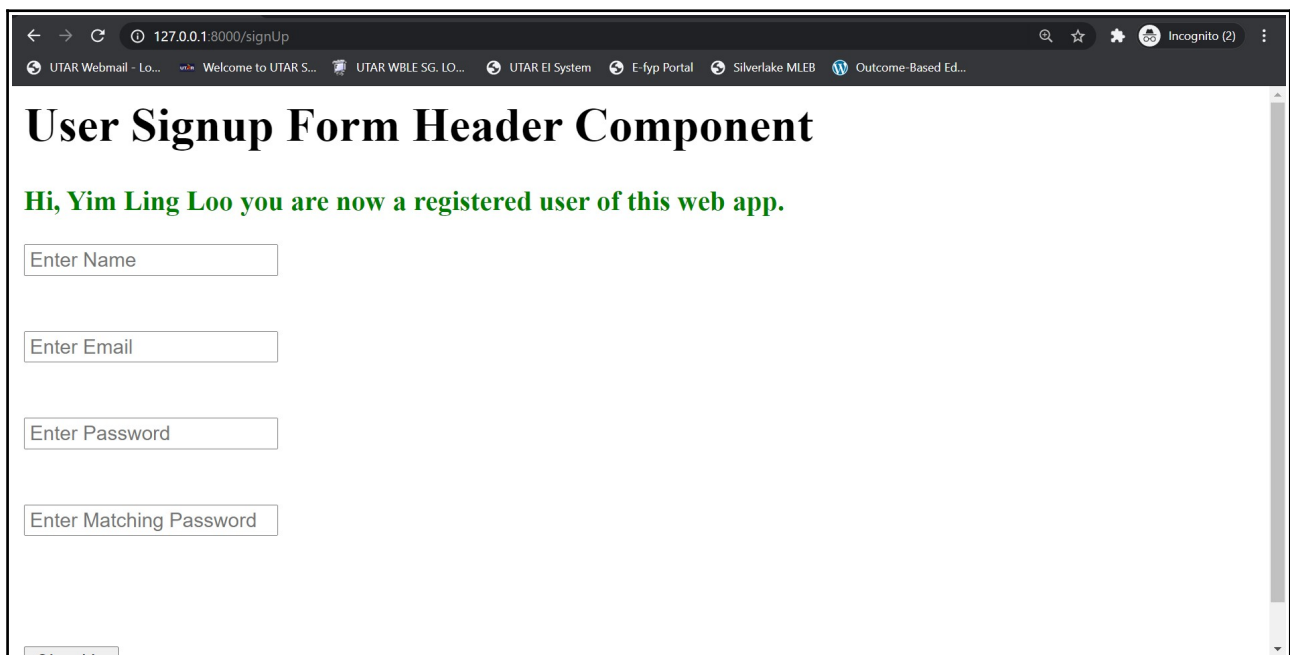
Exercises:

1. Modify previous practical's "signup" blade template to have another data input field called "confirm_password" so that user will enter a matching password as shown in Figure 7.
2. Modify previous practical's "signup" controller to validate the all data input fields so that users will not input empty string. Insert more validation logics so that the email will be validated and password will contain lower case and upper case characters, numbers and special character within 8 to 12 length. Validate the "confirm_password" field to be having same password as
3. Insert the "name" into flash session data and redirect the data to be shown on "signup" page, once the user successfully added him/herself as a new registered user in the web application as shown in Figure 8.



A screenshot of a web browser showing the 'User Signup Form Header Component'. The browser's address bar displays '127.0.0.1:8000/signup'. The page title is 'User Signup Form Header Component'. The form consists of four text input fields stacked vertically: 'Enter Name', 'Enter Email', 'Enter Password', and 'Enter Matching Password'. Below these fields is a 'Sign Up' button. The browser's tab bar shows several open tabs, including 'UTAR Webmail - Lo...', 'Welcome to UTAR S...', 'UTAR WBLE SG. LO...', 'UTAR EI System', 'E-fyp Portal', 'Silverlake MLEB', and 'Outcome-Based Ed...'. The browser is in Incognito mode.

Figure 7: User signup form.



A screenshot of a web browser showing the 'User Signup Confirmation' page. The browser's address bar displays '127.0.0.1:8000/signup'. The page title is 'User Signup Form Header Component'. A green message is displayed: 'Hi, Yim Ling Loo you are now a registered user of this web app.' Below the message are the same four text input fields as in Figure 7: 'Enter Name', 'Enter Email', 'Enter Password', and 'Enter Matching Password'. The 'Sign Up' button is no longer visible. The browser's tab bar shows the same open tabs as in Figure 7. The browser is in Incognito mode.

Figure 8: User signup confirmation.