VRIJE
UNIVERSITEIT
BRUSSEL

SOFTWARE ARCHITECTURES

Report assignment 2

Christophe Van den Eede

December 28, 2021

Sciences and Bio-Engineering Sciences

1 Introduction

In this assignment we were instructed to implement an MVC web application using Akka Play. In this report I will cover the design decisions made.

2 User stories

In this section I will try to cover all of the actions a user is able to make and how they are performed in the application.

- General concept: For all web pages(Views) A simple method was implemented in the most relevant controller. For example the page for registering a new user is given by a method in the UserController.
- A user can register a new account: From the register view, a user can send a form via POST request. This form is checked to contain all fields belonging to the RegisterAttempt, a case class that holds the new username, password and a duplicate of the password. After the form is confirmed correct, the controller checks if the UserDao already contains a user with the specified username and reacts accordingly. The way this form management works is taken from the lab we had on Akka Play. The only difference I implemented is that the case class can be cast to the same object usually used for logging in, by removing the additional password. The UserDao.addUser method is implemented to take this object and create a fully fledged User object from it. If the completed User object needed to be specified from the POST request itself, the view would need to contain hidden input fields and this seemed impractical.
- A user can log in to the website: From the login view, a user can send a form via POST request. this form was taken exactly from the lab on Akka Play.
- A user can post a new tweet: From the timeline view, a user can use the provided tweet form to POST a tweet. This POST again follows the same structure as seen in the lab. However an extra piece of code needed to be added to allow the user to add a picture to their tweet. I found code to perform this task in the Play documentation and i have provided a reference to it in the code. The NewTweetAttempt will be combined with the username of the logged in user and an Option for the filename of the image. the TweetDao will also add an incremented Id to new tweets added from here. In the lab we received an implementation of an AuthenticatedUserAction to use when a user should be logged in to perform an action, however this code does not work when posting forms so I either had to rewrite it to work in this case or find another solution. For this is settled on checking at the start of an action whether the session key "username" has been set. If this has not been set, the action will redirect the user to the login page instead.
- A user can delete their own tweet: From either the timeline, profile or hashtag view the user will see a delete button on all of their own tweets which allows them to submit an invisible form. This form does not apply any form control on the controller side, however the controller does check whether the active user is the owner of the tweet. The response to this request will either be Ok if the tweet was succesfully deleted NoContent if the tweet did not exist or the owner of the tweet was not the authenticated user or BadRequest if the user was not logged in at all. This request is handled in JQuery using ajax. I found JQuery code that allows me to hijack the usual submitting of a form and replace it with an ajax request that removes the tweet in question from the page if Ok is returned. I provided a reference to the source of this code in each of the .js files that implements it.
- A user can like a tweet: From the same views that support deleting tweets, a user can send an ajax POST request to like a tweet. The controller acts much in the same way as delete, however here the TweetDao.likeTweet function returns either true/false depending on the new state of liking the tweet. This is reflected in the response sent back to ajax and allows JQuery to color the like button to reflect the current state.

- A user can share a tweet: This method is identical to liking a tweet and requires no further explanation.
- A user can comment on a tweet: From the same views that support deleting, liking and sharing tweets a user can submit a POST form to place a comment on a tweet. This method works in the same way as the post request we saw in the lab. Ajax will reload the page on a successful comment.
- A user can go to another users profile by clicking on their name: All views that print the name of a user, do so in an ¡a¿ tag, making them link to a URL that will call the ProfilePageController.
- A user can follow another user: From the profile page view, a user can click the follow button to trigger a request identical to like and share. This requires no further explanation.
- A user can use the search bar to look for users or hashtags: When any page containing the "main" view is loaded. An ajax request is sent to retrieve the names of all users and all hashtags, the return value for this is a view containing ¡option¿ tags for each of these items. These are placed into the ¡datalist¿ tag of the search bar. This makes it so the search bar automatically shows and filters between these items as a user types. I created this implementation myself but was inspired by the bootstrap 5.0 documentation. As such I have provided a reference for the html chunk I did take from the documentation.

3 Other functionalities

Besides the actions a user can take, there are a lot of things that happen "unbeknownst" of the user.

- Showing the explore/profile buttons and hiding the login/register buttons is done via a simple if-statement in the view and does not require much explanation. It is however worth mentioning.
- The datasources for showing tweets on the timeline/profile page/hashtag page are different from each other and are managed in the methods for these in their respective controllers. On the timeline only tweets made/shared by users the current user is following are shown. On a profile page only that users' tweets and the ones they shared are shown. On the hashtag page only tweets with a specific hashtag are shown.

4 Data storage

The way I implemented the holding of data in memory was via DAO's. We had already seen the implementation of the UserDao in the lab, I implemented a tweetDao in the same manner. Both of these were outfitted with many methods to retrieve data from them with different degrees of precision.

Beyond these manners of data storage I want to explain how the information for likes on tweets, users being followed and tweets being shared are saves.

- A user has an amount of shared tweets, This information is saved in the User object since this is where the information is most relevant. It is interesting to ask what posts the user has shared, but it is not interesting to ask by what users a post has been shared.

- Contrary to this, a tweet holds the information on which users have liked it. This information can be used to show an amount of likes.
- A user also holds the information on what other users they follow. It is not as important to check what users follow a certain user.