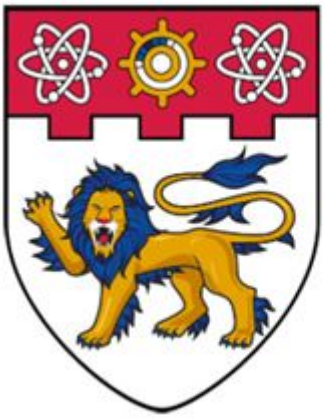


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Assignment 2
CZ3002 Advanced Software Engineering (S12020-2021)

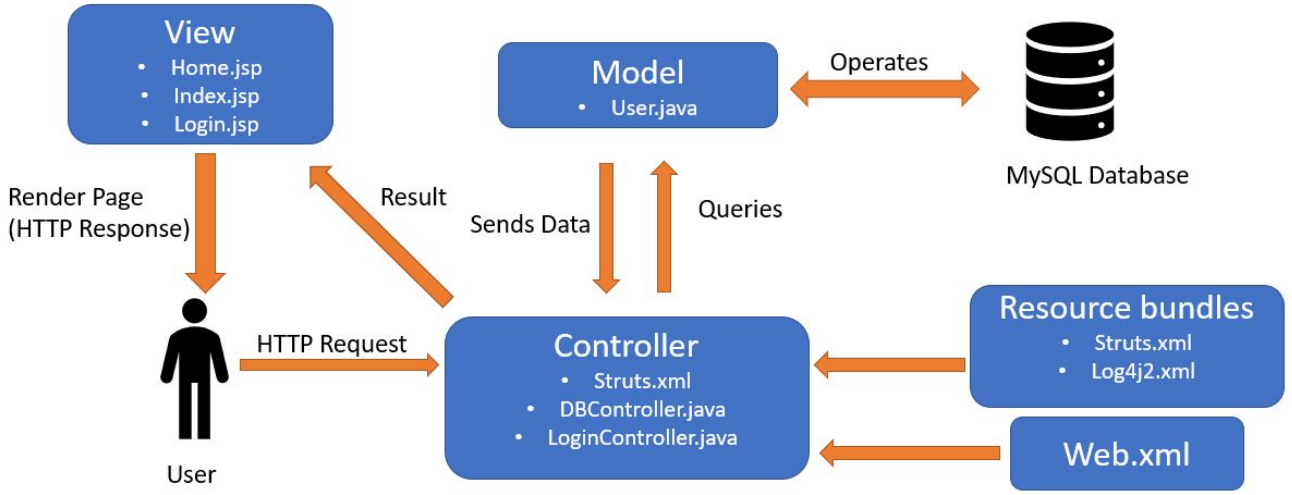
Asok Kumar Gaurav (U1823850F)
Qwek Zhi Hui (U1823304C)
Wong Yuan Neng (U1822431K)

(1) INTRODUCTION

The Model-View-Controller (MVC) is an architectural pattern that divides the application into the following three components - the model, the view, and the controller. It encourages loose coupling between various application components by separating the functionalities of displaying and managing the data. This architecture is useful in applications that require multiple views such as HTML, XML etc.

One of the benefits of the MVC architecture is the fast web application deployment process. Since the architecture is divided into three components, one developer can work on one of the components while the others work on the other components. This kind of modularity of components allows UI developers and core logic developers to work simultaneously without affecting each other.

(II) ARCHITECTURE DIAGRAM



As illustrated in the architecture diagram above, we have utilized the Model View Control (MVC) approach to implement a *Java Web Application* that creates a login function in the *Struts2* framework. The MVC framework is divided into the following components

Controller: The controller forms the brain of the architecture. Residing between the model and view component, it is responsible for processing all business logic and incoming requests. It accepts and converts input into commands for the model or view. An example would be the LoginController. It calls the DBController to authenticate the parsed login details against the database by querying the User model, then the appropriate userbean will be returned to the LoginController for further processing. When no userbean is returned, LoginController adds an ActionError to display in the view, login.jsp, that the username or password input is invalid.

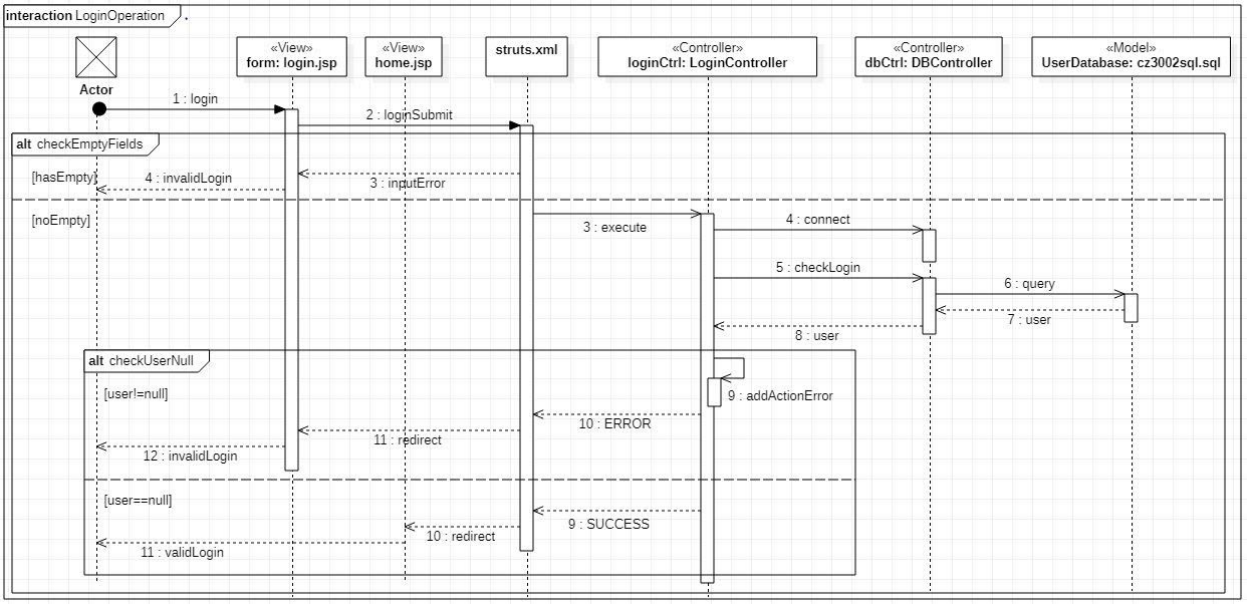
Model: The model is responsible for all the data-related logic used by the user. Independent of any user interface, it holds the basic structure of this web application. Our User model is a data structure that contains the user's full name, username and password, and is queried by the DBController.

View: The entire user interface (UI) of the application is handled by the view component. As the component that directly interacts with the user, it renders information given by the controller and displays it on the UI. The login.jsp view takes inputs (through forms) from the user and upon submitting, sends a request to the LoginController. If valid, the LoginController returns the home.jsp view.

(III) EXECUTION FLOW

The execution of the framework begins with a user HTTP request. The request triggers a series of events across the MVC framework that completes the login authentication process. The web application is first accessed by the user through the localhost port 8080 with the action name index.action. The struts.xml file, with the ActionSupport class, then redirects the user to the login page login.jsp. The primary function of the struts.xml file is to collaborate with the ActionSupport class to control and route the execution flow of HTTP requests by mapping action classes to resulting views that will be returned to the user as HTTP response.

The sequence diagram illustrating the execution flow of the login operation is explained below.



1. User/Actor inputs username and password and clicks "Login" using the form in login.jsp which is part of the View component. Both data inputs are passed to struts.xml, which then checks for empty inputs.
2. If exists an empty input, an empty input error message will show up on login.jsp, indicating an invalid login. This is seen by the User/Actor, indicating an invalid login.
3. Struts.xml then uses the LoginController execute() function. This function in turn uses the DBController. Note both LoginController and DBController are Controllers.
4. DBController connects the LoginController to the *MySQL* database, and allows for the database to be queried to authenticate the login details. If the user exists and is successfully authenticated, its information is stored in a userbean Model component before it is returned to the DBController and subsequently the LoginController.
5. The LoginController returns either an *"SUCCESS"* or *"ERROR"* message to struts.xml, depending on whether there exists a user that is successfully authenticated or not.
6. On receiving a *"SUCCESS"* message, struts.xml redirects to home.jsp showing "Successful login", which is a View component. This is seen by the User/Actor, indicating a valid login.
7. Else, on receiving an *"ERROR"* message, struts.xml redirects to login.jsp showing "Incorrect username/password", which is a View component. This is seen by the User/Actor, indicating an invalid login.
8. Every modified view will result in a HTTP Response to the user web browser accordingly.

(IV) DYNAMIC BINDING

Dynamic binding in the MVC architecture is the binding of the 3 components (M,V and C) during runtime. When a maintenance change occurs in the Controller or the View, the changes will have minimal effect on the other components, preventing the "ripple effect". This is because the

implementation of MVC architecture with *Struts2* greatly reduces the dependency and coupling between components. For example, let there be a change to a different login function. This change can be in the Controller where the implementation of some action methods are changed or in the View where the definition of some JSP files are changed. The Model stays unchanged because the basic data structure of the User class remains as *fullname*, *username* and *password*. The new implementation of actions method(s) will not change the definition of any JSP file in the View and similarly, the new definition of some JSP files will not change the implementation of any action method in the Controller. At runtime, dynamic binding will bind the 3 components together and allow them to interact with each other according to the new changes.

(V) INSTALLATION MANUAL

Environment used:

1. Maven 3.6.3
2. J2SE-1.5
3. junit 3.8.1 - Maven dependency
4. javax.servlet-api 3.1.0 - Maven dependency
5. org.apache.struts 2.5.25 - Maven dependency
6. org.apache.logging.log4j 2.13.0 - Maven dependency

Setting up of MySQL database

The *MySQL* database serves as a means of storage for the user model. This database resides on the localhost alongside with the Java webapp. This assignment assumes that the MySQL database is ready on the localhost. Otherwise, the user MUST run the SQL script "cz3002sql.sql" with user accounts creation permissions first in order to set up the database. This is because a user is created specifically for the database queried by the *Java* webapp.

Running the web server

1. Ensure that the SQL script cz3002sql.sql is executed with root privileges.
2. Navigate to the LoginStruts2 folder.
3. Run the start.bat script, then the client.bat script to open the webpage.

Alternative to run the web server via Eclipse IDE for Enterprise Java Developers

1. Ensure that the SQL script cz3002sql.sql is executed with root privileges.
2. Import the WAR file OR project folder into the workspace
3. Right click the imported project then Run As maven config with the goal set as jetty:run
    - If an error occurs, do Maven --> Update Project and repeat.
    - Required Maven dependencies should be automatically updated by Maven.
4. The webpage can then be retrieved at http://localhost:8080/LoginStruts2/index.action.

(V) REFERENCES

1. https://struts.apache.org/getting-started/index.html
2. https://struts.apache.org/maven/struts2-core/apidocs/com/opensymphony/xwork2/
3. https://www.codejava.net/coding/how-to-code-login-and-logout-with-java-servlet-jsp-and-mysql
4. http://websystique.com/maven/create-a-maven-web-project-with-eclipse/
5. https://www.javatpoint.com/struts-2-registration-form-example
6. https://www.programcreek.com/2010/03/struts-2-tutorials-sample-application-login-module/
7. https://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException