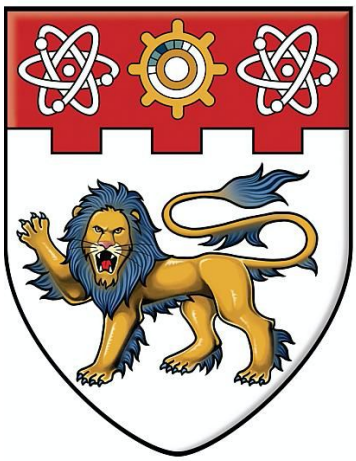


NANYANG TECHNOLOGICAL UNIVERSITY

CZ3002 Advanced Software Engineering

Assignment 2

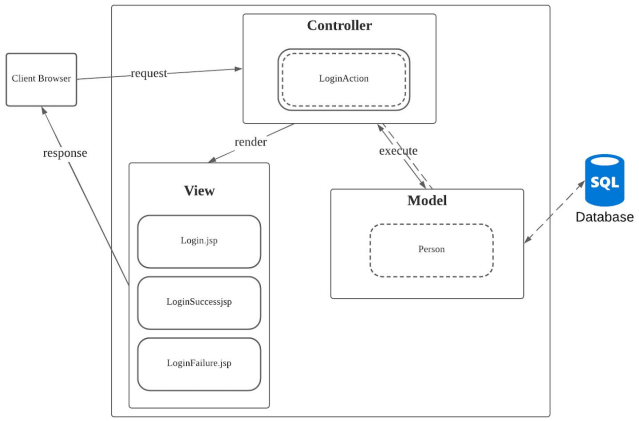
Report on MVC Architectural Design

Name	**Matriculation No.**
Kang Xinhui	U1820019C
Adrian Goh Jun Wei	U1721134D

Understanding MVC (Model-View-Controller)

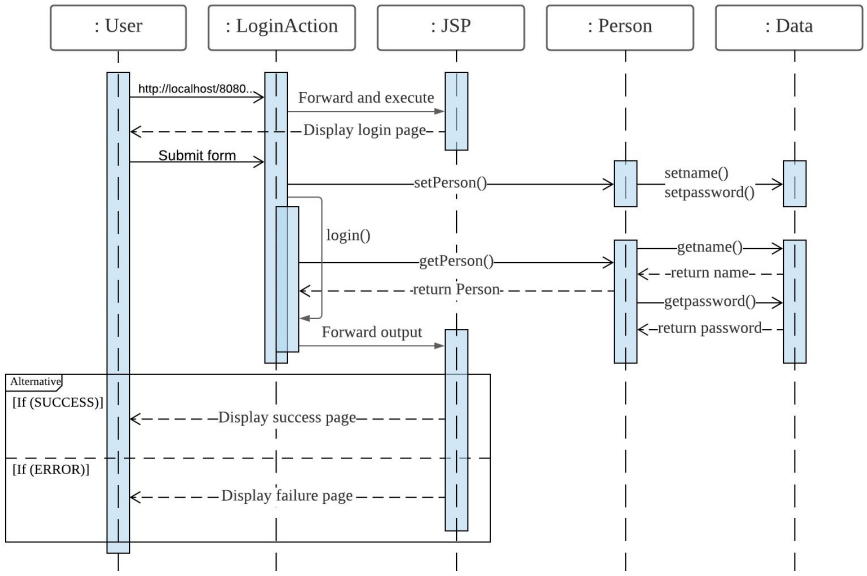
MVC is an architectural design pattern that can be applied to develop software with high maintainability and extensibility. It separates the application into 3 main logical components: Model, View and Controller. Each component is built to handle a specific aspect of the application development. Model contains data-related logic. View handles UI logic and displays information. Controller acts like an interface between Model and View. It processes incoming requests, manipulates data and interacts with View to render the output.

MVC Architecture



Person class is the JavaBean Model class. It stores attributes like username and password for users. View component is implemented through JSP files. We have 3 JSP files to display login form, login success and login failure page respectively. LoginAction class is the Controller class. It takes in the form submitted by users, check with the Model class, which interacts with the database to authenticate the password provided. The Controller class when interacts with View to display login success or failure page to the user. Please refer to the sequence diagram and the execution flow for detailed implementation.

Sequence Diagram



Execution flow of our code for MVC

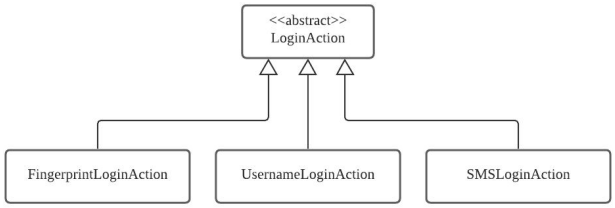
Once a request is sent to the server for the URL http://localhost:8080/myproject/index.action:

1. The application receives the request for the resource index.action. According to the settings loaded, it finds that all requests are being routed to org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteFilter, including the *.action requests. The StrutsPrepareAndExecuteFilter is the entry point into the framework.
2. The framework looks for an action mapping named “index”, and it finds that this mapping corresponds to the class LoginAction. The framework instantiates the Action and calls the action’s “login” method.
3. The action creates the Person object (model) with the values submitted by the user. It then retrieves the password for this object from MySQL database. After comparing the password stored with the value submitted, it returns SUCCESS (="success") if it matches and ERROR (="error") if otherwise.
4. The framework checks the action mapping to see what page to load, and was told by the framework to render the resource LoginSuccess.jsp as the response to the request if SUCCESS was returned, and LoginFailure.jsp if ERROR was returned.
5. As the page LoginSuccess.jsp is being processed, the <s:property value="person"/> tag calls the getter getPerson of LoginAction, which calls the toString() of the Person object returned by getPerson, and merges the value of the message attribute into the response.
6. A pure HTML response is sent back to the browser.

How does dynamic bindings work

Dynamic binding is the mechanism that the method called only binds with the method body during runtime. It helps to achieve high maintainability and hence ease in development when maintenance change occurs. Beside login using username and password, the application might incorporate new login methods like fingerprint login or mobile phone SMS login. We could create an abstract LoginAction Class, and create a FingerprintLoginAction class and a SMSLoginAction class which extend the abstract class and override the non-static method *login()*. During compilation time, the compiler would not know which login method is to be called. The binding is only resolved during runtime according to the referencing variable. If any other need for enhancement arises in the future, the development could also be done without affecting the main application.

We then simply need to update the wiring accordingly in the structs.xml file. Components in the application are then wired during runtime according to the configuration file.



Setup

Ensure that the following requirements are met before proceeding with the installation:

- MySQL version 8.0.22 installed
- Java SE Development Kit 8 (JDK 8) downloaded and installed
- Apache Maven downloaded and installed

Organization

The Model files (Person.java) can be found in */src/main/java/org/apache/struts/myproject/model*. The View files (e.g. Login.jsp) can be found in */src/main/webapp*. The Controller files (LoginAction.java) can be found in */src/main/java/org/apache/struts/myproject/action*.

struts.xml is a XML configuration file to specify the relationship between a URL, a Java class, and a view page (e.g. Login.jsp), and serves to route the flow of the application from one component to another. It can be found in */src/main/java/org/apache/struts/myproject/resources*.

Installation and user manual

To populate the database, open up MySQL workbench and run the following set of SQL queries that can be found in */src/main/webapp/sql/Person.sql*

Once you are in the repository's root directory, run the following command in your console. The server will be running, so you can go to http://localhost:8080/myproject/index.action

```
cd assignment2 & cd myproject & mvn jetty:run
```

To test the application, enter the name and password fields with "adrian" and "secret" respectively, and submit the form. You should be redirected to a login success page. Any other values entered for the fields will result in a login failure page.