NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

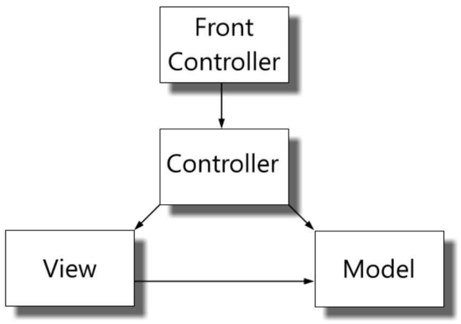
CZ3002 Assignment 2: Login Application Using Struts2 MVC Framework

Group 14

Chester Yeoh Fu Soon (U1620743C)
Goh Jun Le (U1820246G)
Wilson Thurman Teng (U1820540H)

Part 1: Introduction of MVC and MVC frameworks

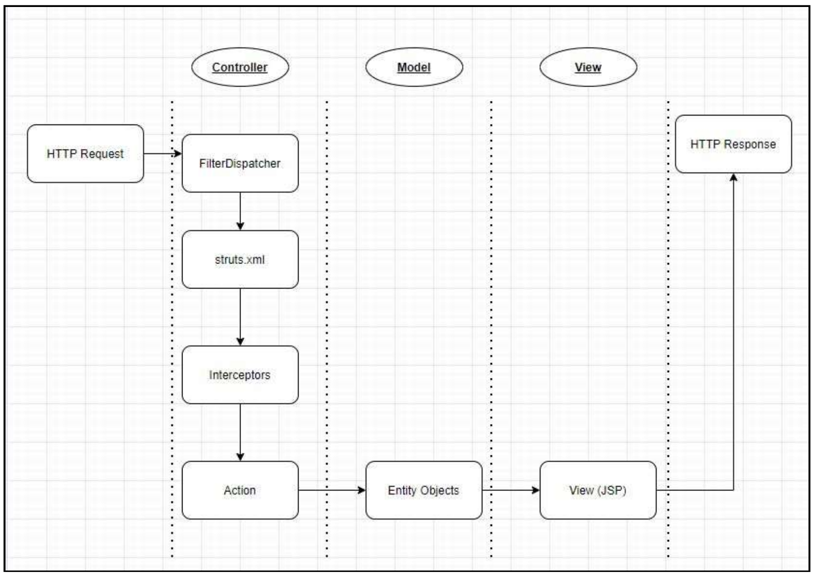
Model-View-Controller (MVC) is a software design architecture that is used to develop applications by decomposing the program logic into three components: Model, View and Controller. More specifically, in web-programming, a Front Controller is introduced.



Front Controller - Consolidates all request handling and common concerns that result from it. (E.g. Security, Session state management)
Model - Manages the data and is responsible for the business logic.
View - Displays data from the Model to the users, often referred to as the user interface.
Controller - Accepts input from the user and alters the View or the Model.

The MVC framework used in this project is Struts 2, a web-application framework for developing Java EE web applications. While its predecessor, Struts 1 has a Push-MVC based architecture, Struts 2's architecture is pull-MVC based. This means that the model data typically constructed in the controller are stored centrally (e.g. in Actions) and retrieved by the View layer for rendering. This is advantageous as the server would not need to store client-side information to update the client state.

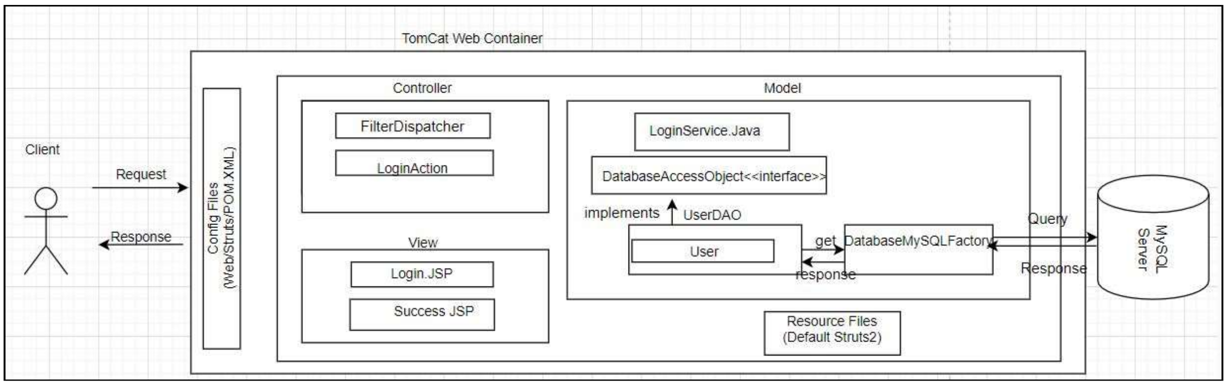
Part 2: Architectural Diagram of Struts 2 from MVC perspective



The controller is implemented via the dispatch servlet filter together with the interceptors, as well as the actions, while the model refers to the entity objects in the application which handles the business logic. The view is implemented by the template JSP files, which displays data from the Model. First, the user makes an HTTP request and the request is received by the Filter Dispatcher, which forwards the request to struts.xml, where the mapping information between a URL and a view page can be found. The request is then forwarded to the interceptors which help invoke the relevant Action, which will reach into the entity objects to retrieve or update business logic related data. Finally, the data from the Model will be displayed in the View which is returned to the client via an HTTP Response.

Part 3: Code Explanation with Diagrams

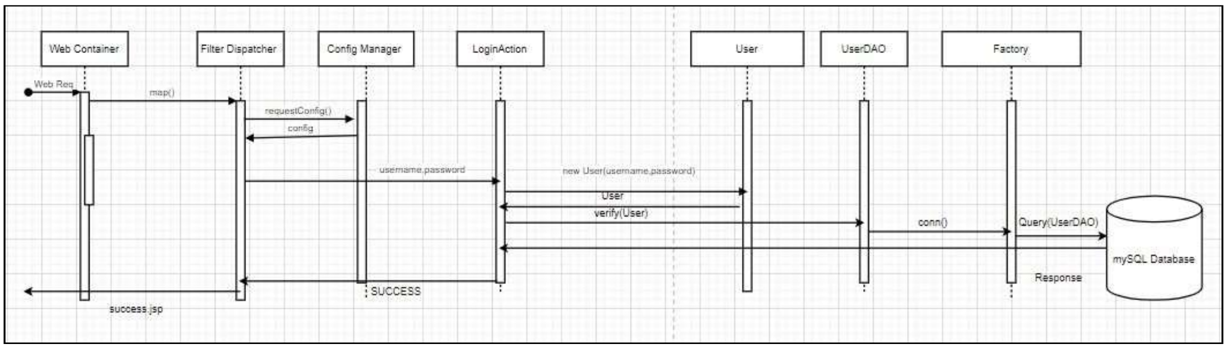
Architectural Diagram



In our architectural diagram, Tomcat was used as the web container to host JSP and servlet codes. Default and commonly used resource JAR files were also downloaded from the apache struts website.

- Config files: **Web** XML are used to define mappings between URL paths and servlets.**Struts** XML files are used in determining the model view controller classes to be dynamically wired respectively.
- Model Classes: ***User*** is the model component and it consists of class attributes username and password which is used for verification during the login procedure. The ***UserDAO*** implements the ***DatabaseAccessObject*** interface and calls the DatabaseMySQL Factory which will then query the MySQL Server.
- View Classes: ***Login.JSP*** is the initial view component in the framework. Success.JSP will be the next screen shown when the login is successful. It consists of html and css code which depict the graphical interface to the user.
- Controller Classes: ***LoginAction*** & ***FilterDispatcher*** form the logic components in the framework. The first class determines whether a username and password is authorised while the second class determines whether the which MVC classes shall be invoked for a particular web request.

Sequence Diagram

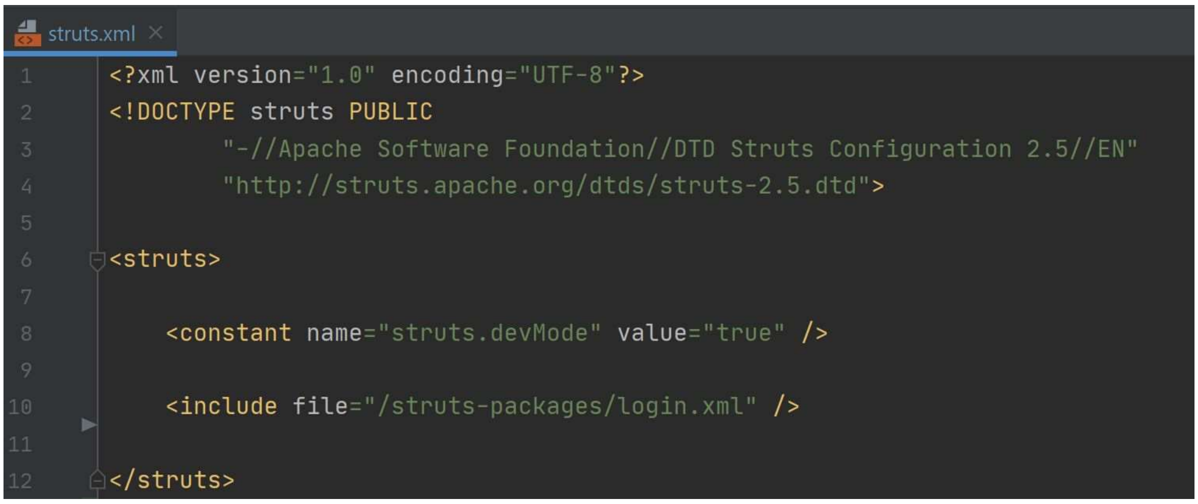


During initial execution when the client raises a web request, the web configuration file will be used to call the relevant dispatcher ("org.apache.struts2.dispatcher.FilterDispatcher"). The Filter Dispatcher will then check the struts.xml file to determine which MVC classes are to be called, followed by forwarding the necessary input parameters to the controller classes. The controller class (LoginAction) will check with the Model class attributes to ensure that the information is verified before calling the necessary view (Login.JSP) to the filter dispatcher , which will then forward it back to the client.

Part 4: Dynamic Binding during Maintenance Change

In the Struts2 framework, dynamic binding occurs at run-time through fly-by-wiring using the Struts.XML configuration file. There is no need to re-compile the project.

A more detailed view of the XML file used is shown below. During a maintenance change (i.e. a view class has to be taken offline), a user can remove the line of code (line 10) without the need to recompile the whole project. When maintenance is completed, the XML file can then be re-wired to the upgraded components.



```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!DOCTYPE struts PUBLIC
3           "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
4           "http://struts.apache.org/dtds/struts-2.5.dtd">
5
6   <struts>
7
8       <constant name="struts.devMode" value="true" />
9
10      <include file="/struts-packages/login.xml" />
11
12  </struts>
```

Part 5: Code Organisation, Installation and User Manual

Code Organisation

	Examples	Location	Description
Model	UserDAO.Java	src>main>cz3002>grp14>dao	User created classes which extends abstract classes specified in default Apache Struts JAR files
View	login.jsp	src>webapp	JSP file that provides graphical views using HTML and CSS
Controller	LoginAction.java	src>main>cz3002>grp14>action	Default JAR files Provided by Apache Struts website(https://struts.apache.org/download.cgi#struts2525)
Configuration	struts.xml	resources	Configuration files that link all MVC objects for a user

Installation and User Manual

For full installation and user manual guide, please refer to the README file in the source folder.