UNIT 6 FRAMEWORKS AVAILABLE FOR J2EE DEVELOPMENT –STRUTS, SPRING BOOT AND HIBERNATE

6.0 INTRODUCTION

A conventional website mainly delivers only static pages, while a web application has the ability to create dynamic responses. A web application receives input from users, interacts with the database and executes the business logic to customize the view as a response to the users.

Web applications use ***Servlet and JSP*** technologies for dynamic response. The Servlet and JSP may contain page design code, control execution code and database code. Mixing design code, control execution, and database code together make a large application difficult or impossible to maintain.

The ***Model-View-Controller (MVC)*** architecture provides the separation of concerns and makes the application easy to maintain and develop. The model represents the business logic, view represents the page design, and controller represents the navigational code. The ***Spring MVC and Struts*** are based on MVC architecture which helps developers to create a maintainable web application efficiently.

Build process of a software project involves a series of processes such as downloading required dependencies, putting jars on classpath, generating source code (for auto-

generated code), source code compilation, tests execution, packaging compiled code along with the dependencies into deployable artifacts such as WAR, EAR or JAR. All processes can be done manually by developers but it's time consuming and error prone.

Apache Maven simplifies the build process and automates all the involved processes. Maven is a popular open-source build tool developed by the Apache Group to build, publish, and deploy several projects at once for better project management.

6.1 OBJECTIVES

After going through this unit, you will be able to:

- explain Model1 and Model2(MVC) architecture,
- describe Struts architecture and execution flow,
- describe Spring Boot and Spring MVC feature,
- describe features of Hibernate/JPA,
- differentiate among Spring, Spring MVC and Spring Boot,
- use Maven tool to automate the build process, and
- describe Maven build life cycles, build phases and build goals

6.2 STRUTS: FEATURES

The Struts framework was developed by Craig McClanahan and he donated it to Apache foundation in May, 2000. In June, 2001 Apache released the first version of Struts as Struts 1.0. Struts 2.5.22 is the most current version of the Struts framework in April, 2021.

Apache Struts2 is an elegant, popular Java Model-View-Controller (MVC) framework to develop MVC based web applications. Struts2 is a complete rewrite of Struts architecture. Struts2 is completely different from Struts1, and it is not the next version of Struts1 framework. The **Webwork** framework was started with **Struts** as base in order to provide a simplified and enhanced framework based on Struts to develop the MVC based web application easily. After a while Webwork framework and Struts community joined hands together to work on this enhanced framework which became famous as Struts2 framework.

The Struts2 is enriched with many important features such as support to POJO based actions, Configurable MVC Components, Integration support to various frameworks such as Spring, Tiles, Hibernate etc., AJAX support, Validation support, support to various theme and Template such as JSP, Velocity, Freemarker etc.

6.2.1 Model 1 Vs Model 2 (MVC) Architecture

The knowledge about design models helps us to decide the architecture of a web application. There are two types of design models.

- Model 1 Architecture
- Model 2 (MVC) Architecture

6.2.1.1 Model 1 Architecture

In earlier days, web application development using Servlet had multiple issues like mixing Business and Presentation logic and recompilation of Servlet if any design code changed.

In model 1 architecture, JSP pages were the focal point for the entire web application. JSP provides the solutions to the problems of Servlet technology. JSP provides a better separation of concern not to mix business logic and presentation logic. Re-Deployment of web applications is not required if the JSP page is modified. JSP supports JavaBean, custom tags and JSTL to keep the Business logic separate from JSP. Model 1 architectural diagram is shown below.

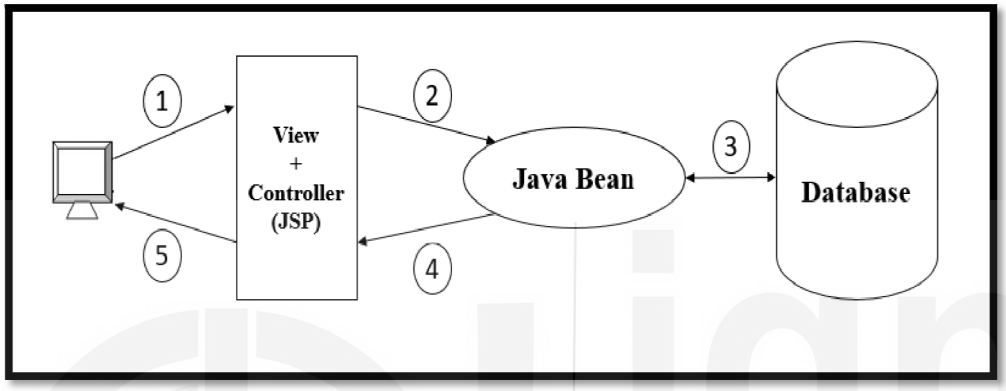

Figure 6.1 : Model 1 Architecture

Notice that there is no **Servlet** involved in the process. The client request is sent directly to a JSP page, which may communicate with JavaBeans or other services, but ultimately the JSP page selects the next page for the client.

Advantages

- Quick and easy way to develop a web application

Disadvantages

- **Decentralized navigation Control:** There is no central control to determine the navigation since every JSP page contains the logic to determine the next page.
- **Hard to maintain:** If a JSP is renamed, then every other JSP which refers to the renamed JSP, needs to be modified.
- **Time Consuming:** Reusable components like custom tags development requires more time.
- **Difficult to extend:** Model1 architecture is not suitable for large and complex applications.

6.2.1.2 Model 2 (MVC) Architecture

In Model 2 architecture contrast to the Model 1 architecture, a Servlet known as ***Controller Servlet*** intercepts all client requests. The Controller Servlet performs all the initial request processing and determines which JSP to display next.

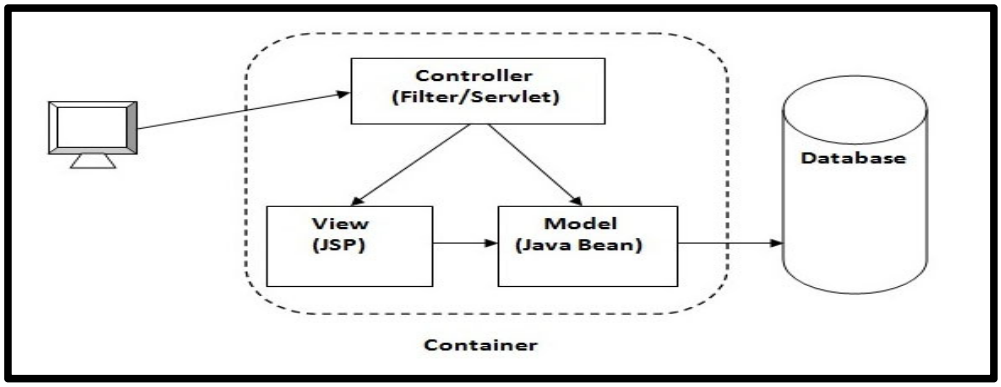

Figure 6.1: Model 2 (MVC) Architecture

In this architecture, the client never sends requests directly to the JSP. This architecture enables us to perform authentication and authorization, centralized logging etc. Once request processing is finished, the servlet directs the request to the appropriate JSP page.

As it can be observed that the key difference between the two architectures is the ***Controller Servlet*** introduced into Model 2, which provides a single point of entry and encourages more reuse and extensibility than the Model 1 architecture. The Model 2 architecture clearly provides the separation of business logic, presentation logic, and request processing. This separation is commonly known as ***Model-View-Controller*** **(MVC)** pattern.

Advantages

- Centralized navigation Control
- Easy to maintain
- Separation of concern
- Easy to extend
- Easy to test

Disadvantages

- Developer writes the Controller Servlet code and it needs to be compiled and redeployed if any logic changes for Controller Servlet.

Struts provides the configurable MVC support with declarative approach for defining view components, request mapping in order to resolve the drawbacks of Model2 architecture. In Struts 2, we define all the action classes and view components in ***struts.xml*** *file.*

6.2.2 Struts2 Architecture

Struts2 is a Pull-MVC (MVC2) based architecture, in which the view layer pulls data stored into Value Stack to render. Struts2 implements the Model-View-Controller architecture with the following five components.

- Interceptors
- Actions
- Results/ Result Types
- Value Stack / OGNL
- View Technologies

Struts2 architectural diagram is shown below.

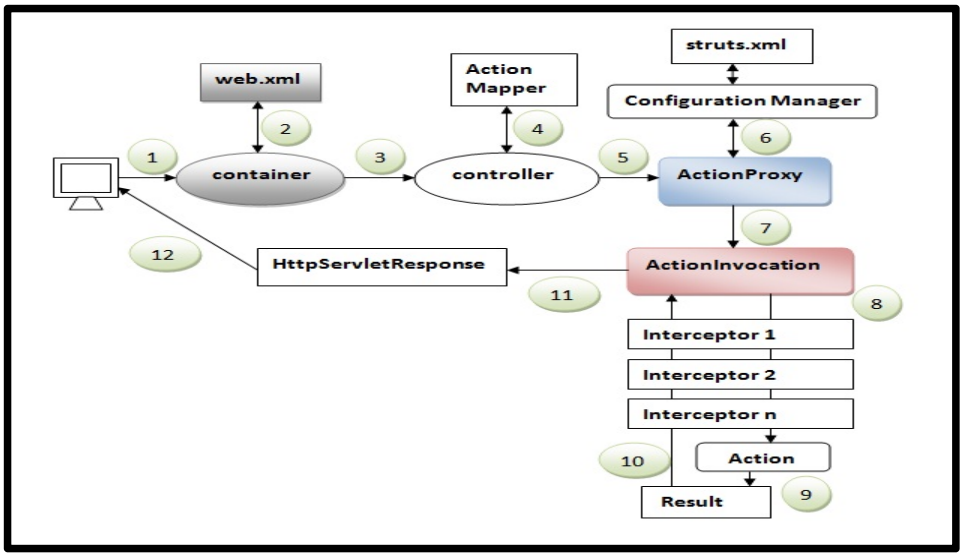

Figure 6.2: Struts2 Architecture

Interceptors: Interceptors are conceptually similar to servlet filters. The Interceptors are used to implement the cross-cutting functionality such as validation, exception handling, internationalization etc. The Struts2 provides many preconfigured and ready to use interceptors out-of-the-box.

Actions: Actions are the core of Struts2 framework. These are used to provide the business logic which executes to serve the client request. Each URL is mapped to a specific action.

In Struts2, the action class is simply POJO (Plane Old Java Object). The only prerequisites for a POJO to be an action is that there must be a no-argument method that returns either a String or Result Object. If the no-argument method is not specified, the execute() method is used to perform business logic processing.

Results / Result Types: The next step after execution of an action is to display a view. The **<results>** tag plays the role of a **view** in the Struts2 MVC framework.

Some navigation rules are associated with the results. For example, if the action method is to register a user, there are three possible results.

1. Successful registration
2. Unsuccessful registration
3. Already registered

For the above, the action method will be configured with three possible outcome strings and three different views to render the outcome. Struts2 does not force us to use JSP as view technology. Struts2 is based on the MVC paradigm, which makes the components configurable.

Struts2 has many predefined **result types,** and the **default** result type is **dispatcher** to dispatch to JSP pages. Struts2 defines different result types for different view technologies such as **Velocities, Tiles, XSLT, Freemarker**.

ValueStack / OGNL: A ValueStack represents a stack that contains application-specific objects such as action objects and other model objects. The objects in the ValueStack are available to the response on UI pages. There are various tags available for JSP, Velocity and Freemarker to access the ValueStack. The ValueStack allows us to put objects, query objects and delete objects using **OGNL**.

The **Object-Graph Navigation Language** (OGNL) is a very powerful expression language similar to the JSP Expression Language. Struts2 OGNL performs two important tasks – **data transfer** and **type conversion**. The OGNL in Struts2 takes the

request parameters from servlet context, performs the type conversion using the in-built type converters to the corresponding type in bean and transfers it to corresponding java variable.

6.2.3 Struts2 Features

The Struts2 is enriched with many important features, and some of them are listed as follows.

1. **Decoupling of view from action classes:** Struts2 allows us to associate a view with multiple action classes.
2. **Ready to use view components:** Struts2 provides many ready to use view components such as stylesheet driven tags which reduces the lines of code into the application for form design and form validation.
3. **Intelligent defaults in all configuration files:** One of the important features of the Struts2 is the default value for the configurable properties. Hence, developers do not need to specify the obvious default values into configuration files.
4. **Use of OGNL:** Struts2 uses the expression language OGNL, which is more powerful and flexible than JSTL.
5. **Type conversion mechanism:** Struts2 supports any type of properties with the help of OGNL, while in Struts1 all the properties are mostly of string type.
6. **Strong validation support:** Struts2 provides a strong validation mechanism with the support of Xwork validation framework, which provides both client-side and server-side validation. It also supports custom validation with the support of Validate() method.

6.2.3 Struts2 Example

The concepts and the components of Struts have been explained in previous sections. This section explains step by step to create a simple Struts2 web application. The required tools and software are as follows.

- Eclipse IDE
- Maven
- Java 8 or above
- Struts 2.x

Step 1: Create a maven project in eclipse with the following group id and artifact id.

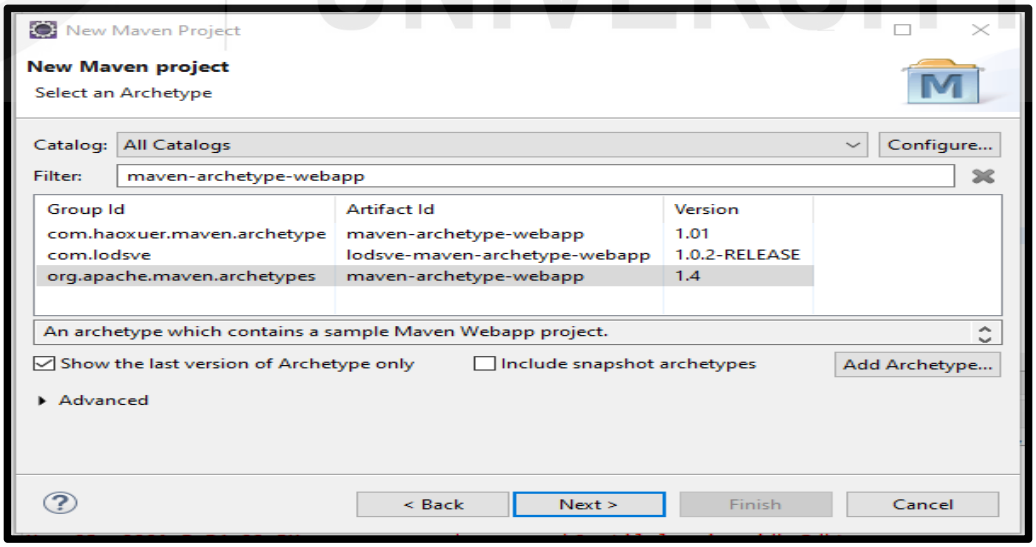

Figure 6.3: Create Maven Project Into Eclipse

Step 2: Right click on the project and go to properties of the project. Check the Java Build Path. If some error is there, select the JRE System Library as shown in the screenshot.

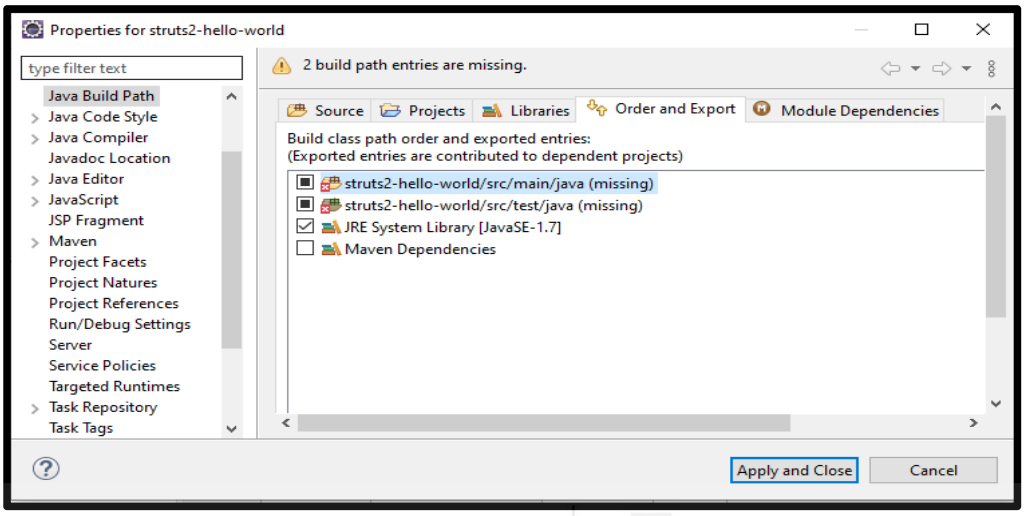

Figure 6.4: Fix Build Path Error

Step 3: Right click on Project folder and click on New option to add a Source Folder as src/main/resources into the project. Final folder structure of the project is as follows.

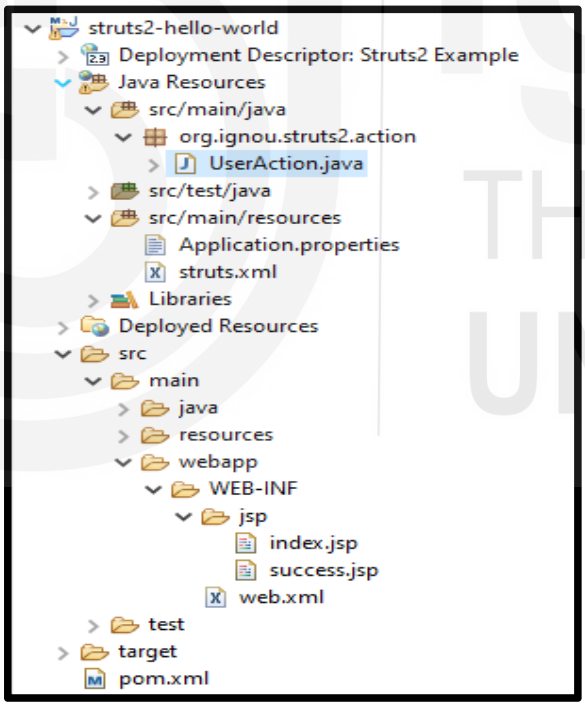

Figure 6.5: Project Structure into Eclipse

Step 4: Add the following maven dependency into pom.xml

```
<!-- https://mvnrepository.com/artifact/javax.servlet/servlet-
api -->
            <dependency>
                  <groupId>javax.servlet</groupId>
                  <artifactId>servlet-api</artifactId>
```

```
                <version>2.5</version>
                <scope>provided</scope>
            </dependency>
<!--
https://mvnrepository.com/artifact/org.apache.struts/struts2-
core -->
            <dependency>
                <groupId>org.apache.struts</groupId>
                <artifactId>struts2-core</artifactId>
                <version>2.5.26</version>
            </dependency>
```

Step 5: Configure the Controller Servlet into web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://java.sun.com/xml/ns/javaee"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
      id="apache-struts-config-example" version="3.0">
      <display-name>Struts2 Example</display-name>

      <filter>
            <filter-name>struts2</filter-name>
            <filter-class>

      org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteF
ilter
            </filter-class>
      </filter>
      <filter-mapping>s
            <filter-name>struts2</filter-name>
            <url-pattern>/*</url-pattern>
      </filter-mapping>

</web-app>
```

Step 6: Create an Action class. UserAction class extends ActionSupport class in order to implement the validation with Validate() method.

```
public class UserAction extends ActionSupport
{
  private static final long serialVersionUID = 1L;
  private String firstName;
  private String lastName;
  private String message;
  @Override
  public String execute() throws Exception
  {
    int timeOfDay =
Calendar.getInstance().get(Calendar.HOUR_OF_DAY);
    if(timeOfDay >= 0 && timeOfDay < 12)
    {
      message = "Good Morning";
    }
    else if(timeOfDay >= 12 && timeOfDay < 16)
    {
```

```
      message = "Good Afternoon";
    }
    else if(timeOfDay >= 16 && timeOfDay < 21)
    {
      message = "Good Evening";
    }
    else if(timeOfDay >= 21 && timeOfDay < 24)
    {
      message = "Good Morning";
    }
    return ActionSupport.SUCCESS;
  }
  @Override
  public void validate()
  {
    if (null == firstName || firstName.length() == 0)

addActionError(getText("error.firstName.required"));
    if (null == lastName || lastName.length() == 0)

addActionError(getText("error.lastName.required"));
  }

  // getter setter...
}
```

Step 7: Configure the request mapping and view in **struts.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
    <include file="struts-default.xml"/>

    <constant name="struts.enable.DynamicMethodInvocation"
value="false" />
    <constant name="struts.devMode" value="true" />
    <constant name="struts.custom.i18n.resources"
value="Application" />

    <package name="default" extends="struts-default">
        <action name="">
            <result>/WEB-INF/jsp/index.jsp</result>
        </action>
       <action name="user"
class="org.ignou.struts2.action.UserAction">
          <result name="error">/WEB-INF/jsp/index.jsp</result>
          <result name="input">/WEB-INF/jsp/index.jsp</result>
          <result name="success">/WEB-INF/jsp/success.jsp</result>
       </action>
    </package>

</struts>
```

Step 8: Add the required properties into Application.properties.

```
label.welcome = Struts 2 Hello World!!!
label.firstName =  First Name
label.lastName =  Last Name
error.firstName.required = First Name is required!
error.lastName.required = Last Name is required!

submit = Go!!
```

Step 9: Add the jsp file inside folder /WEB-INF/jsp.

index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-
1" pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
 <head>
     <title>Struts 2 Maven Hello world!!!</title>
     <s:head/>
 </head>

 <body>
        <h1 style="color: green;text-align: center;"><s:text
name="label.welcome" /></h1>
        <s:if test="hasActionErrors()">
            <div id="fieldErrors">
                <s:actionerror/>
            </div>
        </s:if>

        <s:form action="user" namespace="/" method="post"
name="strutsForm">
            <s:textfield name="firstName" size="30" maxlength="50"
key="label.firstName"/>
            <s:textfield name="lastName" size="30" maxlength="50"
key="label.lastName"/>
            <s:submit key="submit" align="right"/>
        </s:form>
    </body>

</html>
```

success.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-
8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <title>Struts 2 Maven Welcome page!</title>
```

```
    </head>
    <body>
        <h2 style="color: green"><s:property
value="message"/>,</h2>
        <h3 style="color: green">    Mr./Mrs.
<s:property value="lastName" /> <s:property value="firstName"
/></h3>
    </body>

</html>
```

Step 10: Run the application in eclipse by right clicking on project > **Run As** > **Run on Server.** Configure tomcat server into eclipse in order to execute the web application.

Step 11: Access the URL http://localhost:8080/struts2-hello-world/. Outputs of the execution are as followings.

Figure6. 6: Execution Result 1

Validation has been implemented for both the fields. Hence, if any field validation fails, the user will be notified with error messages.

Figure 6.7: Execution Result 2

Success response after valid form submission is as follows.

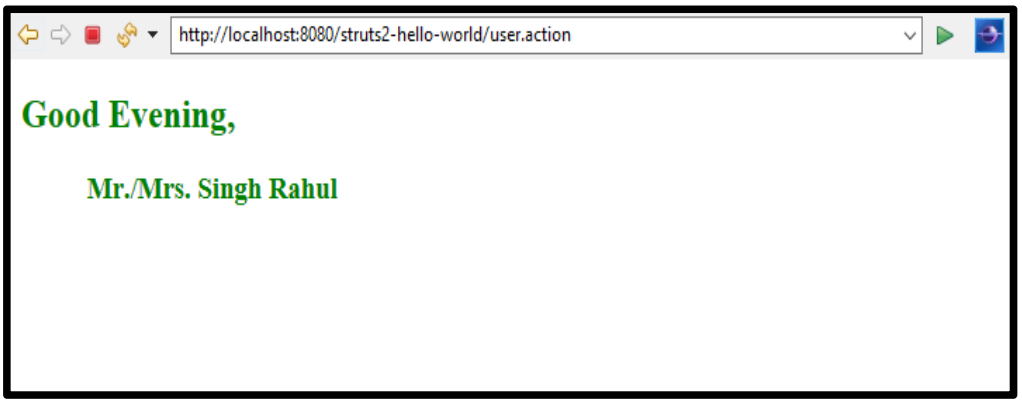

Figure 6.8: Execution Result 3

☛ Check Your Progress 1:

1) Explain Model 1 architecture with its advantages and disadvantages.

2) Explain Model 2 architecture with its merits and demerits.

3) What are Actions in Struts2?

4) List down the features of Struts.

6.3 SPRING BOOT AND MVC: FEATURES

Spring, Spring MVC and Spring Boot do not compete for the same space; instead, they solve the different problems very well. This section briefs about Spring, Spring MVC and Spring Boot. Later it explains the features of Spring MVC and Spring Boot.

6.3.1 Spring MVC features

The **dependency injection (DI)** is the core concept of the **Spring framework,** which enables the development of loosely coupled and easily unit testable applications. Based on the DI, Spring framework created various modules such as ***Spring MVC, Spring JDBC, Spring Test, Spring ORM, Spring AOP*** etc. These modules do not bring any new functionality since we can do all these with J2EE. **These modules simply bring in abstraction** and the abstraction aims-

- Reduce Duplication/ Reduce Boilerplate Code
- Promote Decoupling/ Increase Unit testability

Spring Web MVC framework enables us to develop **decoupled web applications**. It introduces the concepts of Dispatcher Servlet, ModelAndView and View Resolvers to develop loosely coupled web applications. Spring MVC holds many unique features listed below.

- **Clear separation of responsibility:** The DispatcherServlet, controller, handler mapping, view resolver, form object, model object, command object, validation and so on are specialized to perform a single responsibility.
- **Adaptability and flexibility:** Spring MVC is very flexible and adaptable. One of the features to define the controller method signature as per need, possibly using one of the parameter annotations such as @PathVaribale, @RequestParam, @RequestHeader etc.
- **Reusable business code:** It allows us to reuse the existing business object as a form object or command object instead of creating a duplicate business object.
- **Flexible model transfer:** It supports model transfer as name/ value map, which can be easily integrated with any view technology.
- **Customizable handler mapping and view resolution:** Spring Web MVC supports various Handler mapping and View resolution strategies ranging from simple URL-based configuration to sophisticated, purpose-built resolution strategies.
- **Robust JSP form tag library:** Spring 2.0 introduced a powerful JSP form tag library that simplifies the form design in JSP.

6.3.2 Spring Boot features

Spring MVC requires lots of configurations such as dispatcher servlet, component scan, handler mapping, view resolver and many more. When an application uses Hibernate/JPA, it requires configurations such as data source, entity manager, transaction manager etc.

Spring Boot project aims at simplifying and make it easier to develop a Spring based web application. Spring Boot brings intelligence and provides auto configuration features. Spring Boot looks at the framework / jars available in the classpath and

existing configuration of the application in order to provide the basic configuration needed to configure the application. Features of Spring Boot are as follows.

- The guiding principle of Spring Boot is **convention over configuration.**
- Spring Boot starter simplifies dependency management.
- It also supports the development of stand-alone Spring applications.
- Supports auto-configuration wherever possible.
- It supports production-ready features such as metrics, health checks, and externalized configuration.
- XML configuration is not required any more.
- It supports embedded servers such as Jetty, Tomcat or Undertow.

6.4 HIBERNATE WITH JPA: FEATURES

Object-relational mapping (ORM, O/RM, and O/R mapping tool) in computer science is a programming technique for converting data between incompatible type systems using object-oriented programming languages. ORM makes it possible to perform ***CRUD*** operations without considering how those objects relate to their data source. It manages the mapping details between a set of objects and the underlying database. It hides and encapsulates the changes in the data source itself. Thus, when data sources or their APIs change, only ORM change is required rather than the application that uses ORM.

Hibernate is a pure Java object-relational mapping (ORM) and persistence framework which maps the POJOs to relational database tables. The usage of Hibernate as a persistence framework enables the developers to concentrate more on the business logic instead of making efforts on SQLs and writing boilerplate code. Hibernate is having many features as listed below –

- Open Source
- High Performance
- Light Weight
- Database independent
- Caching
- Scalability
- Lazy loading
- Hibernate Query Language (HQL)

Java Persistence API (JPA) is a specification that defines APIs for object relational mappings and management of persistent objects. JPA is a set of interfaces that can be used to implement the persistence layer. JPA itself doesn't provide any implementation classes. In order to use the JPA, a provider is required which implements the specifications. Hibernate and EclipseLink are popular JPA providers.

Spring Data JPA is one of the many sub-projects of Spring Data that simplifies the data access for relational data stores. Spring Data JPA is not a JPA provider; instead it wraps the JPA provider and adds its own features like a no-code implementation of the repository pattern. Spring Data JPA uses Hibernate as the default JPA provider. JPA provider is configurable, and other providers can also be used with Spring Data JPA. Spring Data JPA provides a complete abstraction over the DAO layer into the project.

6.5 COMPARISON AMONG THESE FRAMEWORKS

Spring, Struts and Hibernate are parts of MVC architecture. **These frameworks serve different purposes and can exist independently or together**. It depends on the project requirement to choose the combination of frameworks. This section gives an

overview of comparison among the various frameworks such as Spring Vs Struts, Spring Boot Vs Spring MVC and Spring Vs Hibernate. In the subsequent units Spring, Spring Boot and Hibernate have been explained in detail.

6.5.1 Struts Vs Spring

Struts	Spring
An open source framework which enables to extend Java Servlet API and MVC framework.	An open source framework to implement inversion of Control (IOC) and dependency Injection (DI).
It is a Heavyweight framework.	It is a Lightweight framework.
It is less flexible than Spring.	It is more flexible than Struts.
It has non layered architecture.	It has layered architecture.
It is tightly coupled.	It is loosely coupled.
It also provides integration with ORM and JDBC but manual coding is required.	It provides an easy integration with ORM and JDBC technologies.

6.5.2 Spring Boot Vs Spring MVC

Spring Boot	Spring MVC
Spring Boot removes all boilerplate code and supports the auto configuration based on jars available into classpath.	Spring MVC requires a lot of configuration and it contains a lot of boilerplate code.
Spring Boot provides many spring boot starters. Spring boot starter is a unit of related dependencies wrapped together. It solves the problem of dependency management.	Dependency management is a tough process since every dependency with the correct version needs to be declared.
Spring Boot makes the application development easier and faster.	Spring MVC requires a lot of configurations and dependency management is time consuming. Hence, development is slow.
Spring Boot supports embedded servers to be packaged along with applications to run the jar stand alone.	A lot of manual configuration is required to attain the same level of packaging.
Spring Boot supports many production ready features such as Actuators, Process monitoring out-of-box.	Spring MVC does not support any production ready feature.
Spring Boot is very flexible and provides many modules which can be used to build many different other applications.	Spring MVC is designed only for the development of dynamic web pages and RESTful web services.

6.5.3 Spring Vs Hibernate

Spring	Hibernate
Spring is an open source, complete and modular application framework to develop enterprise applications in java.	Hibernate is an ORM framework specialized in data persisting and data retrieval from DB.
Spring provides many useful features such as transaction management, aspect-oriented programming and dependency injection (DI).	Hibernate provides Object-Relational Persistence and Query service for applications.
Spring framework has many modules such as Spring-Core, Spring-MVC, Spring-Rest, Spring-Security and many more.	Hibernate does not have modules like Spring framework.
Spring framework has support for connection pooling by changing the configuration in the spring configuration file.	Hibernate supports robust connection pooling features. Hibernate supports two levels of cache which improves the application performance.

☛ Check Your Progress 2:

1) What are the advantages of Spring Web MVC?

.....

.....

.....

.....

2) Discuss the features of Spring Boot.

.....

.....

.....

.....

3) Explain Hibernate with its advantages.

.....

.....

.....

.....

4) Compare the Struts and Spring frameworks.

5) Compare Spring Boot and Spring MVC.

6.6 MAVEN: INTRODUCTION, OVERVIEW AND CONFIGURATION

Building the process of a software project may involve a series of tasks among downloading required dependencies, putting jars on classpath, generating source code (for auto generated code), generating documentation from source code, source code compilation, tests execution, packaging compiled code along with the dependencies into deployable artifacts such as WAR, EAR or JAR and deploying the generated artifacts to the application server or repository.

Maven is a simple and powerful build automation and management tool used primarily for Java Projects. It can also be used to build and manage other language projects such as C#, Scala, Ruby, etc. A manual build process is error prone and time consuming. Maven minimizes the risk of humans making errors and speeds up the build process of a software project.

6.6.1 Maven Installation

JDK installation is the prerequisite to execute maven into the system. Visit http://maven.apache.org/download.cgi to download the maven. Extract the downloaded file and perform the following configurations in order to execute maven.

- Set JAVA_HOME environment variable pointing to valid JDK installation path.
- Set M2_HOME environment variable pointing to the directory of extracted maven.
- Add the executable maven path into PATH variable as **%M2_HOME%\bin** on Windows and **%M2_HOME%/bin** on Unix.

Open a command prompt and execute the command ***mvn –version*** to check the maven installation and configuration. You will get the output similar to below.

```
$ mvn -version
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: D:\Rahul\software\apache-maven-3.6.3-bin\apache-maven-3.6.3
Java version: 11.0.10, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-11.0.10
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

Figure 6.9: Maven Version Verification

6.6.2 Maven Core Concepts

POM file (Project Object Model) is the focal point of a maven project. A maven project is configured using a POM file known as pom.xml. A POM file is an XML file which describes the project and contains all the references of project resources like source code, test code, dependencies etc. Maven execution and the main content of the POM file has been illustrated below.

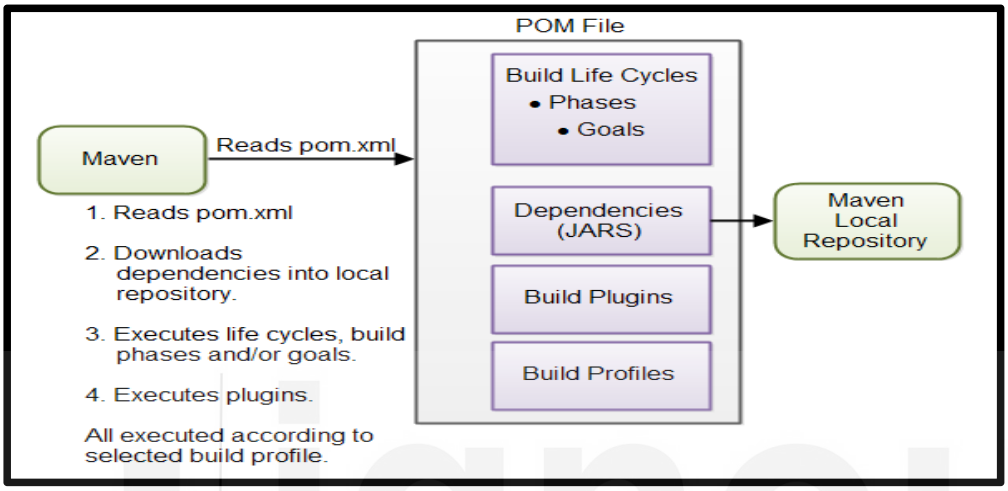

Figure 6.10: Maven Execution Flow

6.6.2.1 Basic Structure of the POM File

Basic structure of a typical POM file contains project identifiers, dependencies, properties, build etc. POM file also supports the concept of profile. Profiles are used to customize the build configuration for different environments such as dev, test and prod. A sample POM file example is shown below.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
      <modelVersion>4.0.0</modelVersion>
      <groupId>org.test</groupId>
      <artifactId>DemoWebApp</artifactId>
      <version>1.0-SNAPSHOT</version>
       <packaging>war</packaging>
      <name>DemoWebApp Maven Webapp</name>
      <url>http://maven.apache.org</url>
      <properties>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
       </properties>
      <dependencies>
            <dependency>
                  <groupId>javax.servlet</groupId>
                  <artifactId>servlet-api</artifactId>
                  <version>2.5</version>
                  <scope>provided</scope>
            </dependency>
            <dependency>
```

```
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>3.8.1</version>
                <scope>test</scope>
            </dependency>
        </dependencies>
        <build>
            <finalName>Demo WebApp</finalName>
        </build>
</project>
```

- **Project Identifiers:** Maven combines groupId:artifactId:version (GAV) to form the unique identifier to identify a project uniquely.
    - groupId – a unique base name of the group or company of the project creator
    - artifactId – a unique name for the project
    - version – a version of the project
    - packaging – a packaging format of the project such as JAR, WAR, EAR, ZIP. If the packaging type is ***pom***, Maven does not create anything for this project since it is just meta-data.
- **Dependencies:** A mid-scale or large scale project uses external Java APIs or frameworks which are packaged in their own JAR files. These Jar files for Java APIs or frameworks are known as dependencies. A dependency JAR may again depend on other dependencies. Keeping projects up-to-date with the correct version of external dependencies a comprehensive task. Downloading all these external dependencies (JAR files) recursively and making sure that the right versions are downloaded is cumbersome.

  Maven has a built-in powerful dependency management feature. To make the project dependencies available into classpath, we just need to specify all the required dependencies with GAV (groupId, artifactId, version) as shown in sample pom.xml. Maven downloads all the dependencies recursively and puts them into the local **maven repository**.
- **Maven Repositories:** Maven repositories are directories of packaged JAR files with extra Meta data. There are three types of repository in Maven.
    - **Local Repository –** As the name indicates, it's a repository located on the developer's machine itself. Maven downloads the dependencies from Central repository, Remote repositories and stores it into the Local Repository. By default, the Local Repository location is **user-home/.m2.**
    - **Central Repository –** The Central Repository is maintained and provided by the maven community. The Central Repository access does not require any specific configuration. It can be accessed at ***https://mvnrepository.com/*** and maven dependencies can be searched.

- **Remote Repository – A** Remote Repository is like the Central Repository from where maven can download the dependencies and store it into Local Repository. It may be located on any web server on the internet or the local network**.** It can be configured into pom.xml by putting the following contents just after **<dependencies>** element**.**

```
<repositories>
        <repository>
          <id>test.code</id>
          <url>http://myremote.maven.com/maven2/lib</url>
        </repository>
</repositories>
```

- **Properties:** Custom properties make the pom file readable and maintainable. Example to use custom properties is shown below.

```
<properties>
   <spring.version>4.3.5.RELEASE</spring.version>
</properties>
<dependencies>
   <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>${spring.version}</version>
   </dependency>
   <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${spring.version}</version>
   </dependency>
</dependencies>
```

If we want to upgrade the spring to a newer version, we just need to change the version at one place only for custom property <spring.version> to update all required dependencies versions for spring.

- **Build:** The build section provides information about the default maven **goal,** the final name of the artifact and the directory for the compiled project. The default build section is like shown below.

```
<build>
     <defaultGoal>install</defaultGoal>
     <finalName>${artifactId}-${version}</finalName>
     <directory>${basedir}/target</directory>
    //...
</build>
```

  - Default goal is install
  - Final name of the artifact contains artifactId and version. It can be modified at any time into the build section.
  - The default output folder for the generated artifact is target folder.

6.6.2.2 Maven Build Life Cycles, Phases and Goals

Maven follows a **build life cycle** while building an application. The build life cycle consists of phases and phase consists of build goals.

- **Build Life Cycle:** Maven has 3 built-in build life cycles which are responsible for different aspects of building a software project. These build life cycles execute independent of one another. For two different build life cycles, you have to use two different maven commands and these build life cycles will execute sequentially. The build life cycles are as followings-
  - **default -** The *default* build lifecycle is of most interest since this is what builds the code. It is responsible to handle everything related to compiling and packaging the software project. ***This build life cycle can't be executed directly***. You need to execute a build **phase** or **goal** from the default build life cycle.
  - **clean -** The *clean* build life cycle performs the tasks related to cleaning such as deleting compiled classes, removing previous jar files, deleting the generated source codes, removing temporary files from the output directory etc. The command **mvn clean** cleans the project.
  - **site -** The *site* build life cycle performs the task related to generating documents for the project.
- **Build Phases:** A Maven phase represents a stage in the Maven build life cycle. Each phase performs a specific task. The most commonly used build phases defined for default build life cycle are as follows.

Build Phase	Description
validate	Verifies the correctness of the project and make sure that all required dependencies are downloaded.
Compile	Compiles the source code to produce the binary artifacts
Test	Executes the unit test cases using a suitable unit testing framework without packaging the binary artifacts
Package	Packs the binary artifacts into distributable format such as JAR,WAR,EAR
intergration-test	Executes integration test cases which require packaging.
Verify	Verifies the correctness of package
Install	Installs the package into local maven repository
Deploy	Copies the final package to the remote repository for sharing with other developers and projects.

Maven allows us to execute either a whole build life cycle such as ***clean or site***, a build phase like ***deploy*** which is a phase of default build life cycle, or a build goal like ***sunfire:test***.

*Note: The execution of a phase using command **mvn <phase>** does not execute only the specified phase. Instead it executes all the preceding phases as well. The execution of **mvn install** will execute all other phases in the above table.*

- **Build Goals:** The finest step in the maven build process is a Goal. A goal can be bound to zero or more build phases. A phase helps to determine the order in which the *goals* are executed.
  - Command **mvn <goal>** executes a goal which is not bound to any phase.
  - Command **mvn <phase>:<goal>** executes a goal which is bound to a phase.

6.6.2.3 Maven Build Profiles

Different build configurations are required for different environments such as production, test, dev. The concepts of Maven Build Profiles enable us to keep the multiple environments build configuration into a single pom.xml file. A sample example to create the configuration based on profiles is shown below.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

<groupId>com.test</groupId>
<artifactId>maven-demo</artifactId>
<version>1.0.0</version>
<profiles>
   <profile>
      <id>production</id>
      <build>
         <plugins>
            <plugin>
            //...
            </plugin>
         </plugins>
      </build>
   </profile>
   <profile>
      <id>development</id>
      <activation>
         <activeByDefault>true</activeByDefault>
      </activation>
      <build>
         <plugins>
            <plugin>
            //...
            </plugin>
         </plugins>
      </build>
   </profile>
 </profiles>
</project>
```

In the above configuration development profile is set as default. If we want to execute production profile build, it can be executed as **mvn clean install –Pproduction**. The flag –P is used to provide the profile to be used for the build process.

6.7 CREATE FIRST PROJECT USING MAVEN

This section describes how to create a simple Hello World java application using Maven. The Maven project will be created using the command line, and pom.xml will be modified to execute the jar file using maven plug-in. Perform the following steps and execute the application using maven command.

Step 1: Create a simple Hello World Java project using the below command. The command will generate the directory structure and pom.xml file. The generated directory structure is shown in the screenshot.

mvn archetype:generate -DgroupId=org.test -DartifactId=hello-world -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

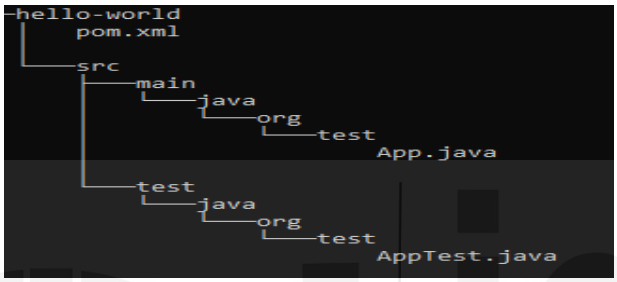

Figure 6.11: Maven Project Folder Structure

Step 2: Simple Hello World application is ready to be compiled and packaged. Update the pom.xml as below.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.test</groupId>
  <artifactId>hello-world</artifactId>
  <name>hello-world</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
    </dependency>
  </dependencies>
  <build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.6.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
</build>
</project>
```

Execute the following commands and check the output on the console.

- mvn compile : Maven will execute all the required phases for compile phase and build the binary artifact.
- mvn test: Only test phase can be executed with this command.
- mvn package : It packages the binary artifact with the specified package type such as jar, war, ear etc. Default packaging type is jar and the generated jar file is located into the target folder.

Step 3: Add the following plug-in into the build section after compiling the plug-in. The plugin requires the main class into configuration. Update the main class. This added plug-in will enable the jar file execution from maven command.

```
<plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>exec-maven-plugin</artifactId>
        <version>3.0.0</version>
        <configuration>
           <mainClass> org.test.App</mainClass>
        </configuration>
  </plugin>
```

The command **mvn exec:java** will execute the jar file. The output for the command execution is shown below:

Figure 6.12: Maven Execution Result

☛ Check Your Progress 3:

1) What is Maven? Discuss its usage.

.....

.....

.....

.....

2) Explain about Maven repositories.

.....

.....

3) Explain the different build life cycles available in the Maven tool.

4) Write down the command to execute a build goal into Maven.

6.8 SUMMARY

The unit has explained the Model 1 and Model 2 architecture of a web application with its advantages and disadvantages. Model 2 fixes the concerns into Model 1 architecture and Struts fixes the issues into Model 2 architecture and provides Pull-MVC (MVC2) based framework to develop enterprise web applications. The features of different frameworks such as Struts, Spring Boot, Spring MVC and Hibernate have been explained and these frameworks have been compared. The last section of this unit has explained Maven as a simple and powerful build and deployment management tool. The highlights of this unit are as follows.

- The Model-View-Controller (MVC) architecture provides the separation of concerns and makes the application easy to maintain and easy to develop.
- The Struts2 is enriched with many important features such as support to POJO based actions, Configurable MVC Components, Integration support to various frameworks such as Spring, Tiles, Hibernate etc, AJAX support, Validation support, support to various themes and Template such as JSP, Velocity, Freemarker etc.
- In model 1 architecture, JSP pages were the focal point for the entire web application.
- In Model 2 architecture contrast to the Model 1 architecture, a Servlet known as ***Controller Servlet*** intercepts all client requests. The Controller Servlet performs all the initial request processing and determines which JSP to display next.
- Spring, Spring MVC and Spring Boot do not compete for the same space instead, they solve the different problems very well.
- Spring Web MVC framework enables us to develop **decoupled web applications**. It introduces the concepts of Dispatcher Servlet, ModelAndView and View Resolvers to develop loosely coupled web applications.
- **Spring Boot** project aims at simplifying and making it easier to develop a Spring based web application. Spring Boot brings intelligence and provides

auto-configuration features. Spring Boot looks at the framework / jars available in the classpath and existing configuration of the application in order to provide the basic configuration needed to configure the application.

- Maven is a simple and powerful build automation and management tool used primarily for Java Projects. Maven minimizes the risk of humans making errors and speeds up the build process of a software project.
- The concepts of Maven Build Profiles enable us to keep the multiple environments build configuration into a single pom.xml file.

6.9 SOLUTIONS/ ANSWER TO CHECK YOUR PROGRESS

Check Your Progress 1

1) In model 1 architecture, JSP pages are the focal point for entire web applications. JSP provides the solutions to the problems of Servlet technology. JSP provides better separation of concern not to mix business logic and presentation logic. Re-Deployment of web applications is not required if the JSP page is modified. JSP supports JavaBean, custom tags and JSTL to keep the Business logic separate from JSP. **Refer section 6.2.1.1 for architecture diagram and advantages and disadvantages.**
2) In Model 2 architecture contrast to the Model 1 architecture, a Servlet known as ***Controller Servlet*** intercepts all client requests. The Controller Servlet performs all the initial request processing and determines which JSP to display next. **Refer section 6.2.1.2 for architecture diagram and advantages and disadvantages.**
3) Actions are the core of Struts2 framework. These are used to provide the business logic which executes to serve the client request. Each URL is mapped to specific action. In Struts2, the action class is simply POJO (Plane Old Java Object). The only prerequisites for a POJO to be an action is that there must be a no-argument method that returns either a String or Result Object. If the no-argument method is not specified, the execute() method is used to perform business logic processing. **Refer section 6.2.3**

Check Your Progress 2

1) Spring Web MVC framework enables us to develop **decoupled web applications**. It introduces the concepts of Dispatcher Servlet, ModelAndView and View Resolvers to develop loosely coupled web applications. Spring MVC holds many unique features listed below.
    - Clear separation of responsibility
    - Adaptability and flexibility
    - Reusable business code
    - Flexible model transfer
    - Customizable handler mapping and view resolution
    - Robust JSP form tag library
2) **Spring Boot** project aims at simplifying and making it easier to develop a Spring based web application. Spring Boot brings intelligence and provides auto configuration features. Spring Boot looks at the framework / jars available into the classpath and existing configuration of the application in order to provide the basic configuration needed to configure the application. Features of Spring Boot are as follows.
    - The guiding principle of Spring Boot is **convention over configuration.**
    - Spring Boot starter simplifies dependency management.
    - It also supports the development of stand-alone Spring applications.

- Supports auto-configuration wherever possible.
- It supports production-ready features such as metrics, health checks, and externalized configuration.
- XML configuration is not required anymore.
- It supports embedded servers such as Jetty, Tomcat or Undertow.

3) Hibernate is a pure Java object-relational mapping (ORM) and persistence framework which maps the POJOs to relational database tables. The usage of Hibernate as a persistence framework enables the developers to concentrate more on the business logic instead of making efforts on SQLs and writing boilerplate code. Hibernate is having many features as listed below –

   - Open Source
   - High Performance
   - Light Weight
   - Database independent
   - Caching
   - Scalability
   - Lazy loading
   - Hibernate Query Language (HQL)

4) Refer the section 6.5.1
5) Refer the section 6.5.2

Check Your Progress 3

1) Maven is a simple and powerful build automation and management tool used primarily for Java Projects. It can also be used to build and manage other language projects such as C#, Scala, Ruby, etc. Manual build process is error prone and time consuming. Maven minimizes the risk of human making errors and speeds up the build process of a software project. Maven is used to perform many tasks like:
   - We can easily build a project using maven.
   - We can add jars and other dependencies of the project easily using the help of maven.
   - Maven provides project information (log document, dependency list, unit test reports etc.)
   - Maven is very helpful for a project while updating the central repository of JARs and other dependencies.
   - With the help of Maven, we can build any number of projects into output types like the JAR, WAR etc, without doing any scripting.
   - Using maven, we can easily integrate our project with source control systems (such as Subversion or Git).

2) Maven repositories are directories of packaged JAR files with extra Meta data. There are three types of repositories in Maven.
   - Local Repository
   - Central Repository
   - Remote Repository

   Refer the section 6.6.2.1

3) Refer Build life cycle in **section 6.6.2.2**

4) The finest step in the maven build process is a Goal. A goal can be bound to zero or more build phases. A phase helps to determine the order in which the goals are executed.
   - Command mvn <goal> executes a goal which is not bound to any phase.
   - Command mvn <phase>:<goal> executes a goal which is bound to a phase.

6.10 REFERENCES/ FURTHER READING

- Craig Walls, "Spring Boot in action" Manning Publications, 2016. (https://doc.lagout.org/programmation/Spring%20Boot%20in%20Action.pdf)
- Christian Bauer, Gavin King, and Gary Gregory, "Java Persistence with Hibernate", Manning Publications, 2015.
- Ethan Marcotte, "Responsive Web Design", Jeffrey Zeldman Publication, 2011(http://nadin.miem.edu.ru/images_2015/responsive-web-design-2nd-edition.pdf)
- Tomcy John, "Hands-On Spring Security 5 for Reactive Applications", Packt Publishing,2018
- https://mkyong.com/tutorials/struts-2-tutorials/
- https://www.tutorialspoint.com/struts_2/struts_overview.htm
- https://www.javatpoint.com/struts-2-tutorial
- https://www.codejava.net/frameworks/struts/introduction-to-struts-2-framework#vsStruts1
- https://stackoverflow.com/questions/17441926/push-vs-pull-model-in-mvc
- https://www.baeldung.com/maven
- http://tutorials.jenkov.com/maven/maven-tutorial.html#maven-pom-files
- https://howtodoinjava.com/maven/maven-dependency-management/
- https://www.quora.com/Whats-the-difference-between-J2EE-Struts-with-Hibernate-and-J2EE-Hibernate-with-Spring-framework