**Spring Boot**

DAY-1 (09-JUL-25)

-Download Eclipse developer edition and run .exe file after extracting the zip file

WebSite Link - <https://www.eclipse.org/downloads/packages/release/2025-06/r/eclipse-ide-enterprise-java-and-web-developers>

Direct Windows installer Link - <https://mirror.kakao.com/eclipse/technology/epp/downloads/release/2025-06/R/eclipse-jee-2025-06-R-win32-x86_64.zip>

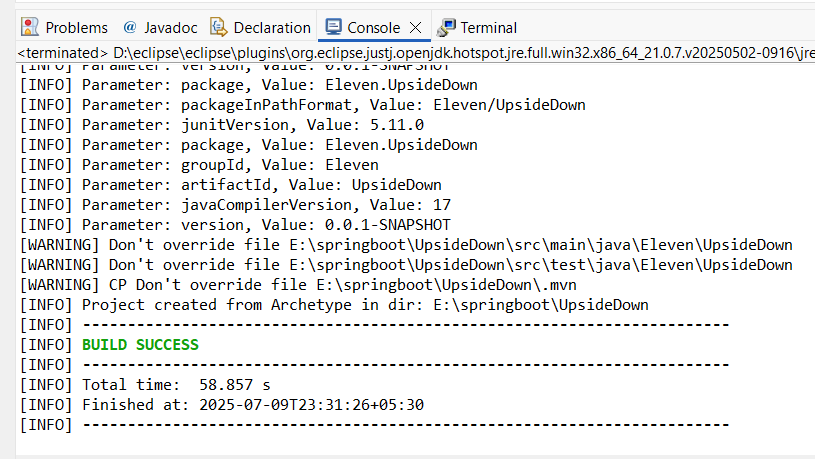
AFTER OPENING ECLIPSE

- click on file->new Maven Project-> give work space location->next

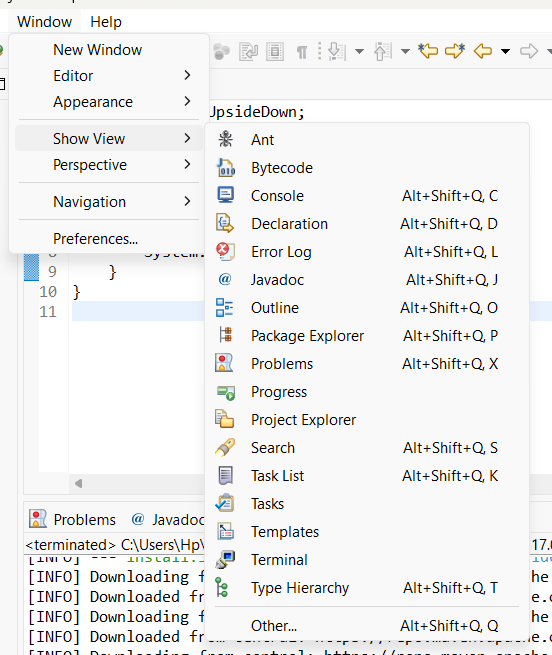
- In archetype selection type org.apache.maven.quickStart 🡪 and select latest version 1.5-> next

- GroupID – anyName, ArtifactID -ProjectName🡪 click on finish

-- you need to type y in console for confirming the project creation and you will able to see build success in console itself.

  
Now the project is created we need to navigate to it and open it.

-- window->showview->PackageExplorer

-

-click on run, to run the application and see output in console

HOW TO CREATE .jar file

-right click on current working maven project and hover on run as option, you could able to see the maven install option so click on it then the .jar file will be created. Here also you can see build success in console itself

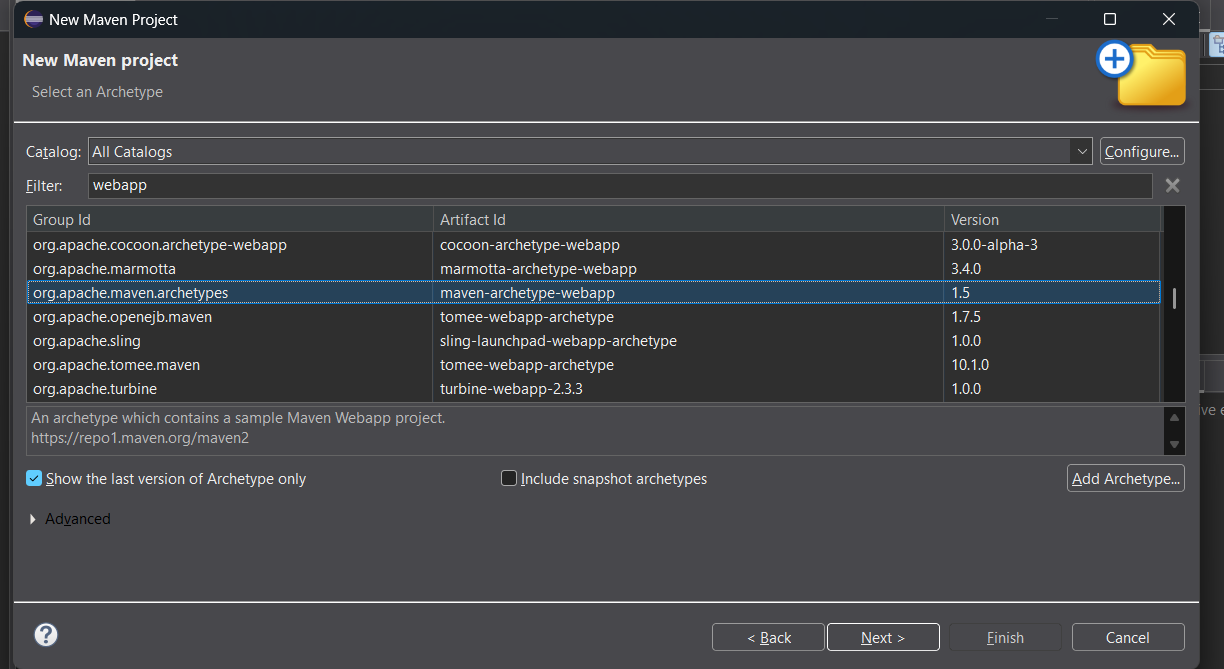
-- To navigate to that jar file copy the path and paste in file explorer



Day – 2 (10-JUL-25)

**Create new maven webApp project:**

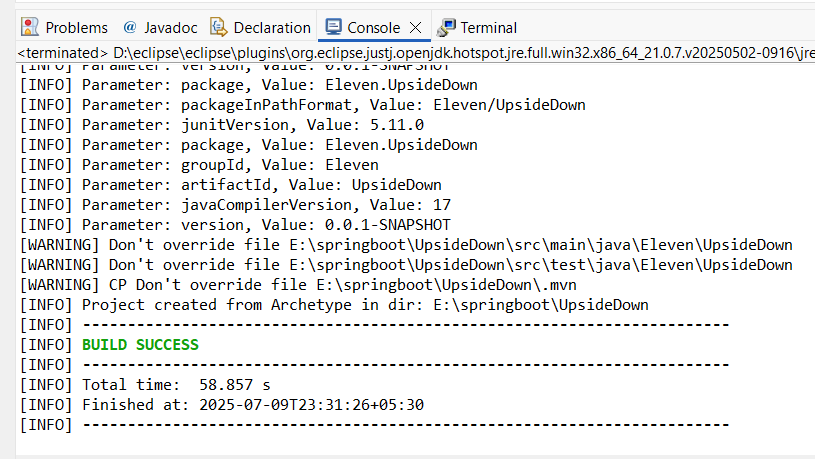
- click on file->new Maven Project-> give work space location->next



- In archetype selection type maven.archtype.webApp 🡪 and select latest version 1.5-> next

- GroupID – anyName, ArtifactID -ProjectName🡪 click on finish

-- you need to type y in console for confirming the project creation and you will able to see build success in console itself.



* Right click on project name and
  + run as maven install
* It’ll creates a war file then it need to deploy in Tomcat.

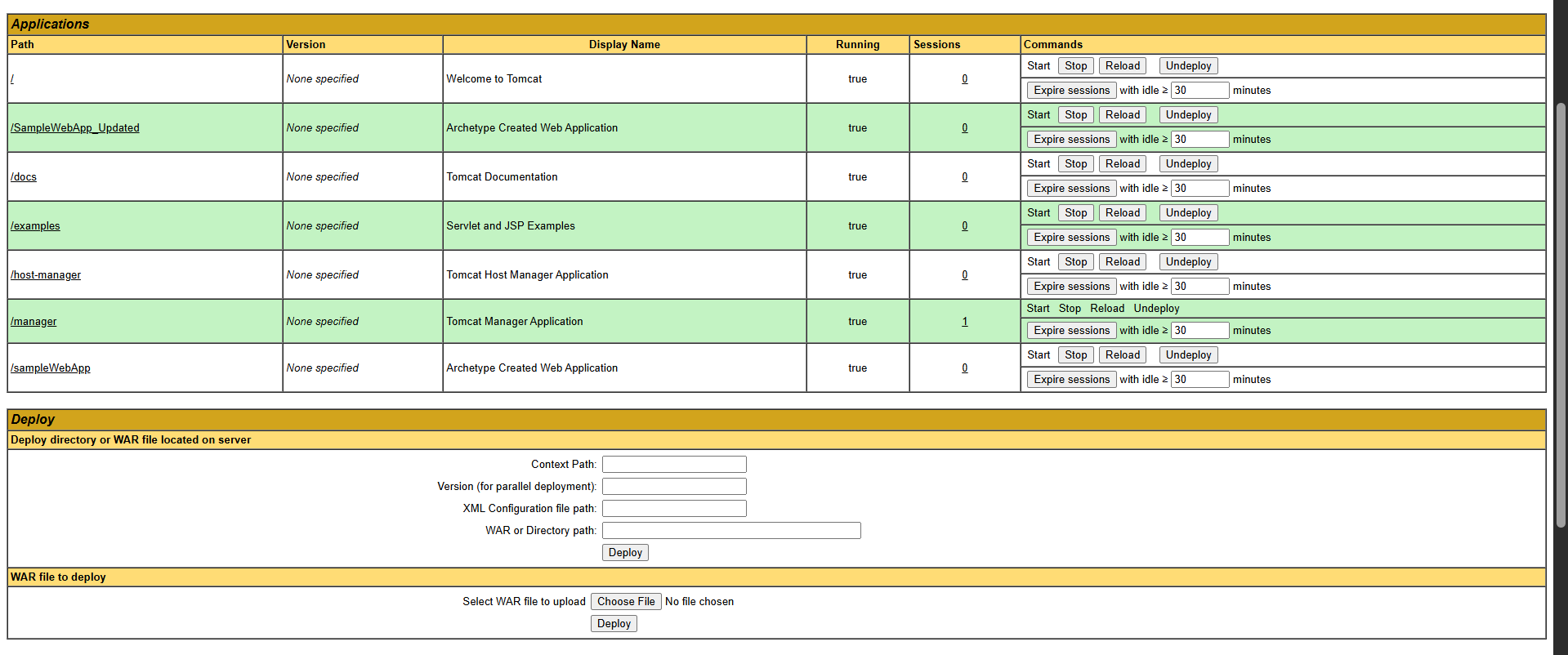
***Install TOMCAT SERVER***

WebSite link - <https://tomcat.apache.org/download-11.cgi>

Windows Direct link - <https://dlcdn.apache.org/tomcat/tomcat-11/v11.0.9/bin/apache-tomcat-11.0.9.exe>

Tomcat installation

* Click nextsss and change below features if you find
* **Select 2 unselected checkboxes** and next
* change http/1.1 connector port num to **8087**
* Mention username & paswd = cse & cse
* install & finish.
* Open chrome type – **localhost:8087/manager/html** (make sure that tomcat is running at BG if not then **windows button -> services -> Apache tomcat -> start the service**)
* Enter username and pswd
* Deploy the above created war file by selecting choose file



* To run that deployed file tap on the **file name** of **Applications Section**

**For Example :** Tap on **/sampleWebApp** file name in Applications section**.**

DAY - 3 (11-JUL-25)

JSP(java servlet pages) – jsp files are converted into java and jvm compiled that into biteCode.

*(Projectname – one)*

**Spring Core – context Dependency injection**

* Create a new maven quickstart project
* Open pom.xml file in that project
* Add below spring Core and context dependencies inside <dependencies> </denependencies>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>7.0.0-M6</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>7.0.0-M6</version>

</dependency>

* Then save it, if it’s in red clr then try to change the version num
  + Eg - 7.0.0-M6 to 7.1.0-M6
  + Eg - 7.0.0-M6 to 7.0.0-M5

IMP for beans:

* Creats a class inside src/main/java/projName then objs of that class are created inside ”config.xml” which is created at src/main/java and stores in a variable.
* Retrieve that objs(beans) from the variable of config.xml, Use those beans at src/main/ App.java file.And **calling the methods of classFile from App.java through config.xml**

**Creating objects(beans) in config.xml file and call those object wherever we want.**

* So create an xml file named config inside ***src/main/java***
* Paste the below code and edit ***filename.className*** as yours

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="

http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean class=" ***filename.className*** " name="student1"> </bean>

<bean class=" ***filename.className*** " name="student2"> </bean>

<bean class=" ***filename.className*** " name="mine"> </bean>

</beans>

* This XML is the **Spring configuration file**.It Creates the objects(beans) of the specified class and stores in name=” ”. Which stores in the spring container **ApplicationContext**,
* <bean> tags define **objects (beans)** that Spring should manage.

**Create a new class inside src/main/java ->projectname(groupID.artifactId)**

* Paste the below code

package projectName;

public class csedata {

public void disp()

{

System.out.println("this is from cse data");

}

}

* Edit at pojectname & class

**GoTo App.java inside src/main/java**

* Paste the below code.
* Edit the below code at classNames

**import org.springframework.context.ApplicationContext;**

**import org.springframework.context.support.ClassPathXmlApplicationContext;**

**public class App {**

**public static void main(String[] args) {**

**System.out.println("welcome to final cse spring world");**

**ApplicationContext cse=new ClassPathXmlApplicationContext("config.xml");**

**csedata cd=(csedata)cse.getBean("student1");**

**cd.disp();**

**}**

**}**

* ApplicationContext is the **Spring container** responsible for managing beans.
* ClassPathXmlApplicationContext("config.xml"):
  + Loads the config.xml file from your classpath.
  + Reads bean definitions (objects to be created) from XML.
  + Creates and manages those objects (beans).
* getBean("student1"):
  + Retrieves the bean with name student1 (i.e., the Csedata object.
* cd.disp():
  + Calls the disp() method on the retrieved Csedata object.

**Finally run the app.java file to get matter inside disp() of classFile**

DAY – 4 (12-JUL-25)

* Automatic Getter and setter,constructor,to string,constructor superclass methods creation with variables.
* Ryt click on variables -> source ->GetterSetter/ to string/ constructor ..etc
* **To String** method is mandatory for output to be printed.
* Day 5 will give more clarity.

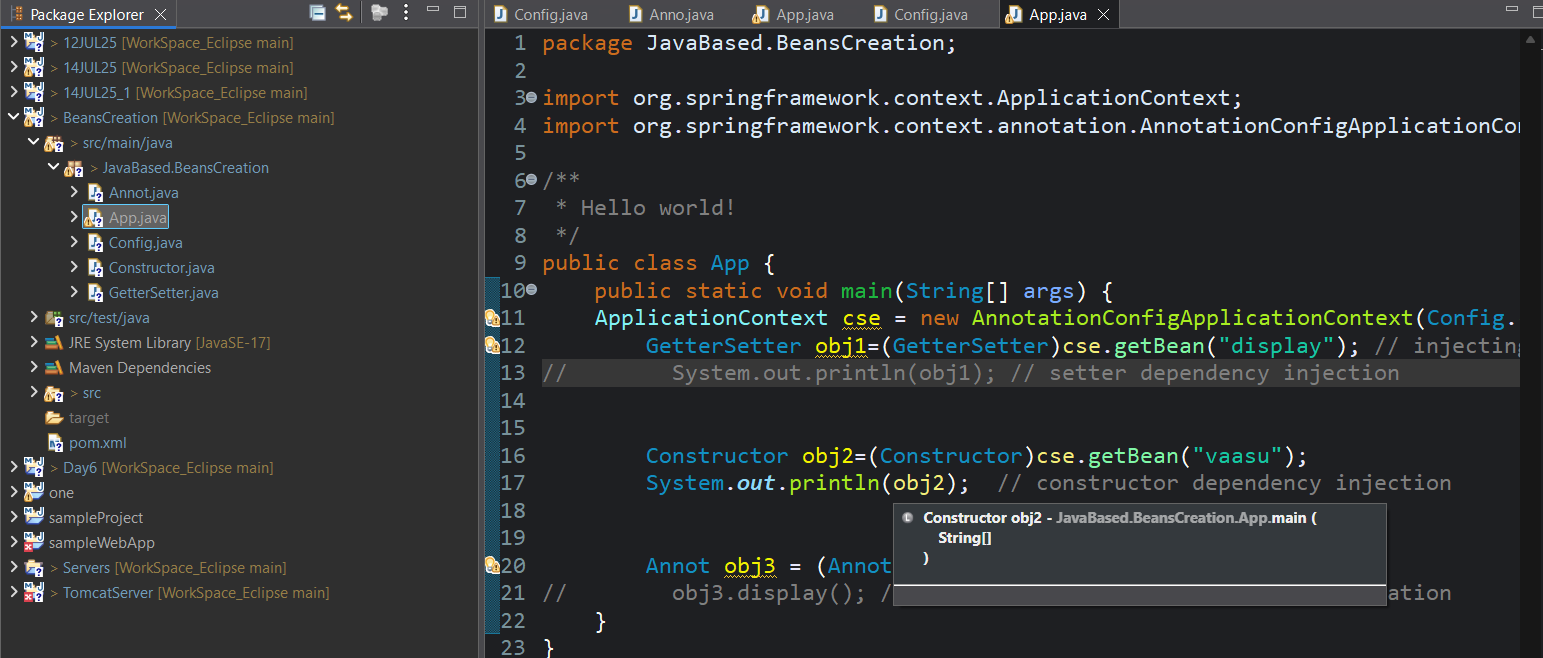
DAY- 5 (14-JUL-25)

Topic : Java – Class based Beans Creation

* Personal (File name - JavaBased.BeansCreation)

**Steps :**

* Create a quick start maven project with Spring core & context dependencies.
* Create 3 classes and 1 Cofig.java files inside GroupID.ArtifactID which is inside the src/main/java
* 3 classes are for getting output from GetterSetter-Constructor-Annotation classes individually by creating Objects(Beans) based on java.
* Previously we created beans based using ApplicationContext which is on Config.xml now we are using AnnotationContext.



DAY – 6 (15-JUL-25)

**Purpose : Integrating Tomcat & eclipse**

***Steps:***

- Create a maven webApp project.

- Add spring mebMvc dependencies in pom.xml

-> Adding Spring webApp mvc from <https://mvnrepository.com/artifact/org.springframework/spring-webmvc/> in pom.xml dependency

**<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->**

**<dependency>**

**<groupId>org.springframework</groupId>**

**<artifactId>spring-webmvc</artifactId>**

**<version>7.0.0-M6</version>**

**</dependency>**

* Add spring mebMvc dependencies in pom.xml
  + Adding Spring webApp mvc from <https://mvnrepository.com/artifact/org.springframework/spring-webmvc/> in pom.xml dependency

<!-- https://mvnrepository.com/artifact/org.springframework/springwebmvc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>7.0.0-M5</version>

</dependency>

* By Adding spring webmvc we additionally get spring core & context dependencies automatically.
* To create server Window->show ->servers/other->servers->choose tomcat any version.
* Double tap on **Tomcat v11.0 Server** **at localhost** in console & set the port num 8090 on 1st then 8089 on 2nd.
* Now Ryt click on **Tomcat v11.0 Server at localhost** in console and start.
* Ryt click on project run as -> run on server = It'll open a jsp file on browser.
* Install tomcat library dependencies = Ryt click on project -> Build path - > configure path -> Add library -> Server Runtime library (It’s an integrated apache tomcat webServer)

DAY-7 (16-JUL-25)

- Web.xml is the heart of our serverSide web application

- Open web.xml which is inside src/main.webapp/web-inf/web.xml

- Here we import the Dispatcher servlet and creating object for it.WKT Dispatcher servlet is

<servlet> <! --creating afriend contlr that is Dispatcher servlet. -->

<servlet-name>spring</servlet-name> <! -- Name Of the friend controller is Spring

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class> Importing Dispatcher servelt from this webSite

</servlet>

<servlet-mapping>

<servlet-name>spring</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

* Create a spring config file which should be named as NameOfDispatcherServlet-servlet.xml inside WEB-INF then paste the below code
* <beans xmlns="http://www.springframework.org/schema/beans"
* xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
* xmlns:context="http://www.springframework.org/schema/context"
* xmlns:mvc="http://www.springframework.org/schema/mvc"
* xsi:schemaLocation="http://www.springframework.org/schema/beans
* https://www.springframework.org/schema/beans/spring-beans-3.0.xsd
* http://www.springframework.org/schema/mvc
* https://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd
* http://www.springframework.org/schema/context
* https://www.springframework.org/schema/context/spring-context-3.0.xsd">
* <context:component-scan base-package="controllers"/> ///
* <mvc:annotation-driven /> ///
* <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver" id="viewResolver">
* <property name="prefix"> ///view resolver has to solve the issue of dispatcherServlet - prefix provide the views inside web-inf/views
* <value>/WEB-INF/views/</value>
* </property>
* <property name="suffix"> ///Suffix provide the data in the form of .jsp
* <value>.jsp</value>
* </property>
* </bean>
* </beans>
* In this NameOfDispatcherServlet-servlet.xml, ***The contrlr asks the dispatcher servlet to provide path then suffix gives the output in .jsp format and it stored in views folder of prefix and get that using annotation then finally return to the contrlr.***
* Create a controller folder inside Java Resources -> src/main/java
* Inside the controller create a class named CseHome.
  + Import @Controller
  + Write below info inside the file
    - @RequestMapping(“/csehome) //if we execute this hrl then below method will be executed.

Public String demo() {  
return “cseindex”;

}

* + Create a cseindex.jsp file inside webapp & then Run this as run on server.