**Assisted Practice: 4.3 JSP Directives**

This section will guide you to:

* Create a JSP file to test JSP directives and run it in the browser
* Show how an include directive works using a second JSP file

**Development Environment**

* Eclipse IDE for Enterprise Java Developers v2019-03 (4.11.0)
* Apache Tomcat Server v9.0
* JRE: OpenJDK Runtime Environment 11.0.2

This lab has eight subsections, namely:

* + 1. Creating a dynamic web project
    2. Creating a JSP file index.jsp
    3. Creating a JSP file included.jsp
    4. Checking for servlet-api.jar
    5. Building the Project
    6. Publishing and starting the project
    7. Running the project
    8. Pushing the code to your GitHub repositories

**Step 4.3.1:** Creating a dynamic web project

* Open Eclipse
* Go the **File** menu. Choose **New->Dynamic Web Project**
* Enter the project name as **JSPDirectives**. Click on **Next**
* Enter nothing in the next screen and click on **Next**
* Check the checkbox **Generate web.xml deployment descriptor** and click on **Finish**
* This will create the project files in the Project Explorer

**Step 4.3.2:** Creating a JSP file index.jsp

* In the Project Explorer, expand the project **JSPDirectives**
* Expand **WebContent**. Right click on **WebContent**. Choose **New->JSP File**
* Enter the filename as **index.jsp** and click on **Finish**
* Enter the following code:
* <https://stackoverflow.com/questions/8400301/cout-unknown-tag>

You're apparently developing with a servlet container which does not support JSTL out the box, such as Tomcat. In that case, you need to download [jstl-1.2.jar](https://repo.maven.apache.org/maven2/javax/servlet/jstl/1.2/jstl-1.2.jar) and drop in /WEB-INF/lib folder of your webapp. **No** other changes are necessary, also not extracting the JAR file and/or littering the /WEB-INF folder with loose TLD files as some poor online tutorials suggest.

After having dropped the JAR file in the classpath (the /WEB-INF/lib folder is part of the webapp's runtime classpath), you should be able to reference the JSTL core taglib by putting the following line in top of your JSP as per [its documentation](http://docs.oracle.com/javaee/5/jstl/1.1/docs/tlddocs/c/tld-summary.html):

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

JSTL 1.2 requires a minimum of Servlet 2.4 declaration in web.xml. So make sure that your web.xml has the proper root declaration, preferably the highest supported version as supported by your servlet container (Tomcat 7 is Servlet 3.0, Tomcat 6 is Servlet 2.5 and Tomcat 5.5 is Servlet 2.4).

* Add this to the top of jsp page.

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

https://stackoverflow.com/questions/22708241/unknown-tag-cforeach-in-eclipse

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>

https://stackoverflow.com/questions/37395594/unknown-tag-fmtformatnumber

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<!DOCTYPE html>

<!--

Download jstl.jar from http://central.maven.org/maven2/javax/servlet/jstl/1.2/jstl-1.2.jar

-->

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>JSP Directives</**title**>

</**head**>

<**body**>

This is content from the main file.

<%@ include file="included.jsp" %>

<**hr**>

Example of using JSTL taglibs for formatting output<**br**>

<**p**>

Currency = <fmt:formatNumber value = "145" type = "currency"/>

<**p**>

<**p**>

<c:set var = "now" value = "<%= new java.util.Date()%>" />

Current date and time is <fmt:formatDate pattern = "yyyy-MM-dd hh:mm:ss" value = "${now}" />

</**p**>

</**body**>

</**html**>

* Click on the **Save** icon

**Step 4.3.3:** Creating a JSP file included.jsp

* In the Project Explorer, expand the project **JSPDirectives**
* Expand **WebContent**. Right click on **WebContent**. Choose **New->JSP File**
* Enter the filename as **included.jsp** and click on **Finish**
* Enter the following code:

<% out.println("This is content from included file.<**br**><**br**>"); %>

* Click on the **Save** icon

**Step 4.3.4:** Checking for servlet-api.jar

* Before building the project, we need to add **servlet-api.jar** to the project
* Servlet-api.jar file is already present in your practice lab. (Refer FSD: Lab Guide - Phase 2)
* To add it to the project, follow the below-mentioned steps:
  + In the **Project Explorer**, right click on **JSPDirectives** and choose **Properties**
  + Select **Java Build Path** from the options on the left
  + Click on **Libraries** tab on the right
  + Under **ClassPath,** expand the node that says **Apache Tomcat**
  + If there is an existing entry for **servlet-api.jar,** then click on **Cancel** and exit the window
  + If it is not there, then click on **Classpath** entry and click on **Add External JARs** button on the right
  + From the file list, select **servlet-api.jar** file and click on **Ok**
  + Click on **Apply and Close**

**Step 4.3.5:** Building the project

* From the **Project** menu at the top, click on **Build**
* If any compile errors are shown, fix them as required

**Step 4.3.6:** Publishing and starting the project

* If you do not see the **Servers** tab near the bottom of the IDE, go to **Window** menu and click on **Show View->Servers**
* Right click on the **Server** entry and choose **Add and Remove**
* Click the **Add** button to move **JSPDirectives** from the **Available** list to the **Configured** list
* Click on **Finish**
* Right click on the **Server** entry and click on **Publish**
* Right click on the **Server** entry and click on **Start**
* This will start the server

**Step 4.3.7:** Running the project

* To run the project, open a web browser and type: [**http://localhost:8080/**](http://localhost:8080/ServletConcept)**JSPDirectives**

**Step 4.3.8:** Pushing the code to your GitHub repositories

* Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

* Initialize your repository using the following command:

**git init**

* Add all the files to your git repository using the following command:

**git add .**

* Commit the changes using the following command:

**git commit . -m “Changes have been committed.”**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**