

	<p align="center">UNIVERSIDAD DON BOSCO ESCUELA DE COMPUTACIÓN</p>
<p align="center">CICLO I</p>	<p align="center">PROGRAMACIÓN ORIENTADA A OBJETOS JSTL</p>

I. OBJETIVOS

Que el estudiante:

- Comprenda las ventajas del uso de las librerías JSTL.
- Pueda crear mantenimientos básicos con JSTL y utilizando además un pool de conexiones.
- Implementar la internalización en un proyecto web utilizando las librerías JSTL

II. INTRODUCCIÓN

¿Qué es JSTL? (JSP Standard Tag Library)

La librería JSTL es un componente dentro de la especificación del Java 2 Enterprise Edition (J2EE) y es controlada por Sun Microsystems. JSTL no es más que un conjunto de librerías de etiquetas simples y estándares que encapsulan su funcionalidad principal: escribir páginas JSP de una manera más sencilla y estándar. Las etiquetas JSTL están organizadas en 4 librerías:

- **core:** Comprende las funciones de script básicas como bucles, bloques condicionales, y entrada/salida de datos.
- **xml:** Comprende el procesamiento de xml.
- **fmt:** Comprende la internacionalización y formato de valores como de moneda y fechas.
- **sql:** Comprende el acceso a base de datos.

¿Cuál es el problema con los scriptlets JSP?

La especificación JSP ahora se ha convertido en una tecnología estándar para la creación de sitios web dinámicos en Java, y el problema es que han aparecido algunas debilidades:

Los scriptlets tienen la forma: 

```
<% int contador = 100; %>
```

- El código Java embebido en scriptlets es desordenado.
- Un programador que no conoce Java no puede modificar el código Java embebido, anulando uno de los mayores beneficios de los JSP: permitir a los diseñadores y personas que escriben la lógica de presentación que actualicen el contenido de la página.
- El código de Java dentro de scriptlets JSP no pueden ser reutilizados por otros JSP, por lo tanto la lógica común termina siendo re-implementada en múltiples páginas.
- La recuperación de objetos fuera del HTTP Request y Session es complicada. Es necesario hacer el Casting de objetos y esto ocasiona que tengamos que importar más clases en los JSP.

¿Cómo mejora esta situación el uso de JSTL?

- Debido a que las etiquetas JSTL son XML, estas etiquetas se integran limpia y uniformemente a las etiquetas HTML.
- Las 4 librerías de etiquetas JSTL incluyen la mayoría de funcionalidad que será necesaria en una página JSP. Las etiquetas JSTL son muy sencillas de usarlas para personas que no conocen de programación, a lo mucho necesitarán conocimientos de etiquetas al estilo HTML.
- Las etiquetas JSTL encapsulan la lógica como el formato de fechas y números. Usando los scriptlets JSP, esta misma lógica necesitaría ser repetida en todos los sitios donde es usada, o necesitaría ser movida a clases de ayuda.
- Las etiquetas JSTL pueden referenciar objetos que se encuentren en los ambientes Request y Session sin conocer el tipo del objeto y sin necesidad de hacer el casting.
- Los **JSP EL** (Expression Language) facilitan las llamadas a los métodos Get y Set en los objetos Java. Esto no es posible en la versión JSP 1.2, pero ahora está disponible en JSP 2.0. **EL** es usado extensamente en la librería JSTL.

¿Cuáles son las desventajas de JSTL?

- JSTL puede agregar mayor sobrecarga en el servidor. Los scriptlets y las librerías de etiquetas son compiladas como servlets, los cuales luego son ejecutados por el contenedor. El código Java embebido en los scriptlets es básicamente copiado en el servlet resultante. En cambio, las etiquetas JSTL, causan un poco más de código en el servlet. En la mayoría de casos esta cantidad no es mensurable pero debe ser considerado.
- Los scriptlets son más potentes que las etiquetas JSTL. Si desea hacer todo en un script JSP pues es muy probable que insertará todo el código Java en él. A pesar que las etiquetas JSTL proporciona un potente conjunto de librerías reutilizables, no puede hacer todo lo que el código Java puro nos permite realizar. La librería JSTL está diseñada para facilitar la codificación en el lado de presentación que es típicamente encontrado en la capa de Vista si hablamos de la arquitectura Modelo-Vista-Controlador.

Historia de JSTL

Con JSTL se pretendía recopilar las etiquetas JSP más usadas en una biblioteca estándar que pudiera usarse en todos los contenedores JSP. La especificación JSTL se desarrolló bajo el auspicio del JCP [<http://www.jcp.org/en/home/index>](Java Community Process, Proceso Comunitario Java). El JCP es un proceso supervisado por SUN pero abierto a empresas, e individuos particulares, que guía el desarrollo y aprobación de los estándares para el lenguaje Java. Las iniciativas para crear un estándar dentro del proceso JCP se conocen como JSR (Java Specification Request, Petición de Especificación Java). La JSR No. 52 [<http://www.jcp.org/jsr/detail/52.jsp>] se llamó "A Standard Tag Library for JavaServer Pages" o, abreviadamente, JSTL. Fue solicitada originalmente por Eduardo Pelegri-Llopert y Anil Vijendran, empleados de SUN. En su desarrollo participaron individuos como Jason <http://today.java.net/cs/user/print/au/8?x-t=full.view> Hunter http://www.javahispano.org/text.viewer.action?file=jason_hun_es , y representantes de varias organizaciones (ASF, Adobe, BEA, y otras).

La especificación JSTL 1.0 fue terminada el 11 de julio de 2002. Unos días después apareció la primera implementación <http://jakarta.apache.org/taglibs/doc/standard-1.0-doc/intro.htm> creada

por miembros del proyecto Taglibs <http://jakarta.apache.org/taglibs/doc/standard-doc/intro.html> de la fundación Apache. La última versión de JSTL a día de hoy es la 1.2 aunque la versión estable es 1.1, es implementada por el proyecto Taglibs y es parte de Java EE 5 Platform.

Etiquetas JSTL

Una etiqueta JSTL corresponde a una acción; llamándolas acción nos indica que añaden comportamiento dinámico página estática.

Librería	URI	Prefijo Librería
Core	http://java.sun.com/jsp/jstl/core	c
Internacionalización I18N	http://java.sun.com/jsp/jstl/fmt	fmt
SQL/DB	http://java.sun.com/jsp/jstl/sql	sql
Procesamiento XML	http://java.sun.com/jsp/jstl/xml	x
Functions	http://java.sun.com/jsp/jstl/functions	fn

III. PROCEDIMIENTO

Para esta guía necesita crear un proyecto web llamado “GuiaJSTL”.

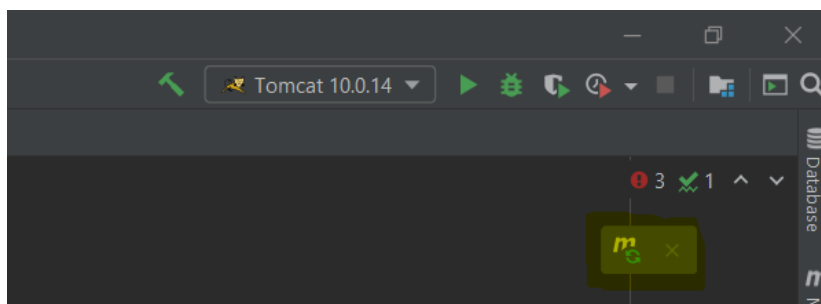
Nota: Utilizar el servidor del Apache Tomcat y la JDK 14.0.2.

Instalación y configuración del JSTL

- Agregaremos las siguientes líneas de código al archivo pom.xml, notara que lo hacemos en la sección de dependencias al igual que con el mysqlconnector.

```
!-- https://mvnrepository.com/artifact/org.glassfish.web/jakarta.servlet.jsp.jstl -->
<dependency>
  <groupId>org.glassfish.web</groupId>
  <artifactId>jakarta.servlet.jsp.jstl</artifactId>
  <version>2.0.0</version>
</dependency>
```

- Deberemos actualizar el proyecto para que dichas dependencias sean integradas correctamente.



- Las dependencias podrán ser vistas en el área de librerías y el error al colocar la configuración en el pom.xml debe desaparecer.

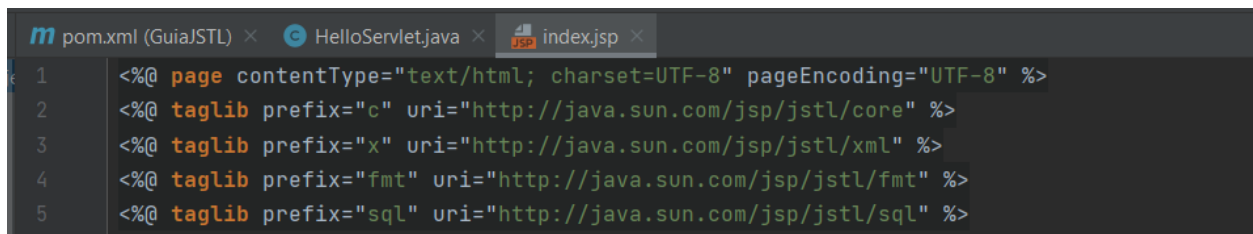


Ejercicio 1

Ahora importamos en las páginas JSP cada librería JSTL que la página necesitará. Eso lo hacemos agregando las directivas taglib apropiadas al inicio de la página JSP. Las directivas son las siguientes:

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

Este paso es imprescindible para el uso de las etiquetas de jstl, por lo cual siempre que se quiera dar uso a esta tecnología se deberá colocar en el encabezado de la jsp, de la siguiente manera:



Taglib: Con JSP es posible hacer una librería de clases Java que hagan una especie de ampliación de las etiquetas posibles de HTML. De esta forma, podríamos llamar con unas etiquetas -tags- especiales a las clases Java que hemos hecho en nuestra librería.

Etiquetas

La librería core

En las páginas que la usen deberemos incluir la siguiente directiva:

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

Esta librería implementa acciones de propósito general, como mostrar información, crear y modificar variables de distinto ámbito y tratar excepciones. Veremos algunas de las etiquetas más comunes.

▪ **c:out**

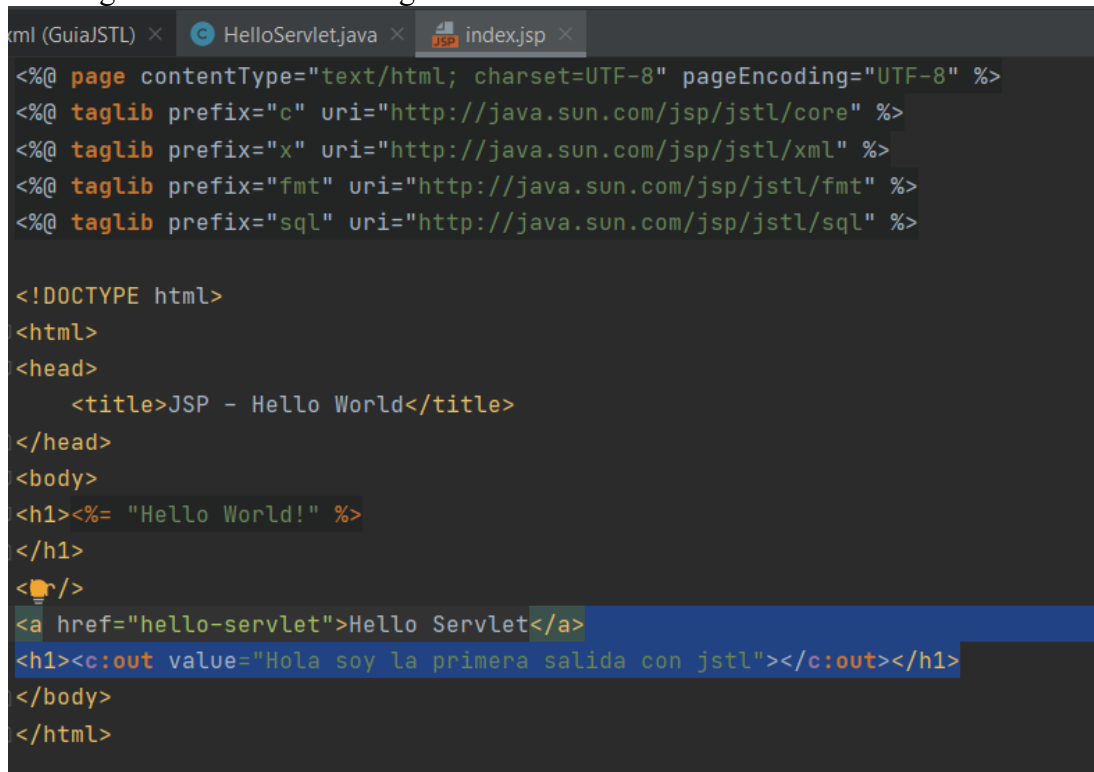
Muestra información en la página, se presenta la expresión contenida en el atributo value. Su funcionalidad es equivalente a la de `<%= %>`.

Atributo	Descripción	Requerido
value	Información a mostrar.	Sí
default	Información a mostrar por defecto.	No

- 1- Importaremos las directivas taglib dentro **index.jsp** creado por defecto en la creación del proyecto. De no tenerla procesa a crear este archivo jsp dentro de la carpeta **Web App**. Finalmente agregaremos nuestra primera salida al cliente con jstl agregando la siguiente línea de código.

```
<h1><c:out value="Hola soy la primera salida con jstl"></c:out></h1>
```

- 2- El código deberá verse de la siguiente manera.



```
xml (GuiaJSTL) × HelloServlet.java × index.jsp ×
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>

<!DOCTYPE html>
<html>
<head>
  <title>JSP - Hello World</title>
</head>
<body>
<h1><%= "Hello World!" %>
</h1>
<img alt="Java logo" data-bbox="179 764 224 776"/>
<a href="hello-servlet">Hello Servlet</a>
<h1><c:out value="Hola soy la primera salida con jstl"></c:out></h1>
</body>
</html>
```

- 3- Proceda a correr su proyecto para ver el siguiente resultado.



Hello World!

[Hello Servlet](#)

Hola soy la primera salida con jstl

- 4- Notara que el uso de la etiqueta `c:out` es similar a `<%= "Hola soy la primera salida jstl" %>` , para lo cual agregaremos esta línea de código para comparar.

```
<h1><%= "Soy una salida con scriplets" %></h1>
```

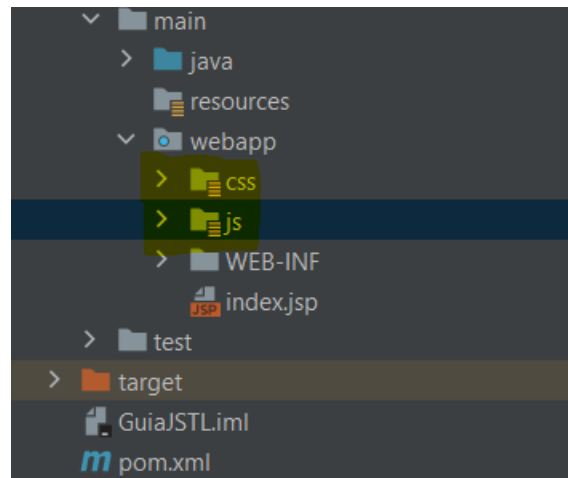
- 5- El código se verá así y el resultado será el mostrado a continuación.

<pre><!DOCTYPE html> <html> <head> <title>JSP - Hello World</title> </head> <body> <h1><%= "Hello World!" %> </h1>
 Hello Servlet <h1><c:out value="Hola soy la primera salida con jstl"></c:out></h1> <h1><%= "Soy una salida con scriplets" %></h1> </body> </html></pre>	
---	--

- 6- Notara que hacemos lo mismo solo que con `jstl` evitamos usar código java dentro de mi `jsp` y procedemos a usar estructuras de etiquetas parecidas al `html`.

Ejercicio 2.

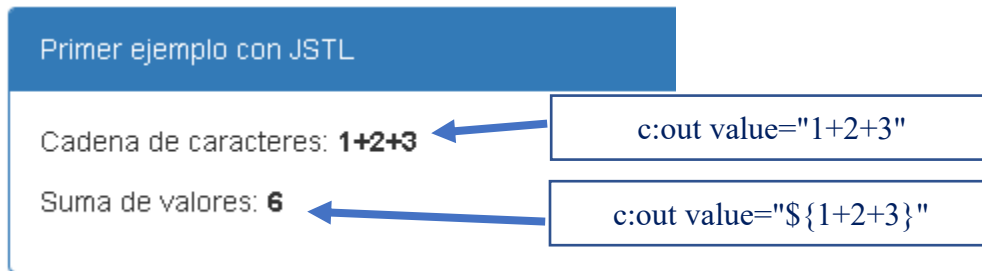
- 1- Agregaremos los recursos proporcionados por su docente, para lo cual deberá colocar estos como **resources root**, de no recordar consultas las guías anteriores.



- 2- Modificar la página de “**index.jsp**” incluir el siguiente código.

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <title>Ejemplo JSTL</title>
</head>
<body>
<div class="container">
  <div class="row">
    &nbsp;
  </div>
  <div class="panel panel-primary">
    <div class="panel-heading">Primer ejemplo con JSTL</div>
    <div class="panel-body">
      <p>Cadena de caracteres: <strong><c:out
value="1+2+3"/></strong></p>
      <p>Suma de valores: <strong><c:out
value="{1+2+3}"/></strong></p>
    </div>
  </div>
</div>
</body>
</html>
```

- 3- Correr la página y observar el resultado.



Ejercicio 2

- 1- Ahora crear una página JSP con el nombre **"Datos.jsp"** y agregar el siguiente código:

```
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/bootstrap.min.css">
<title>Datos JSTL</title>
</head>
<body>
<div class="container">
<div class="row">
<div class="col-sm-4 col-sm-offset-4">
<div class="row">
<h3>Datos personales</h3>
</div>
<form role="form" name="persona" action="ProcesarC.jsp" method="POST">
<div class="form-group">
<label for="nombre">Ingrese su nombre:</label>
<input type="text" class="form-control" name="nombre" id="nombre" placeholder="Nombre">
</div>
<div class="form-group">
<label for="apellido1">Ingrese su primer apellido:</label>
<input type="text" class="form-control" id="apellido1" name="apellido1" placeholder="Primer apellido">
</div>
<div class="form-group">
<label for="apellido2">Ingrese su segundo apellido:</label>
<input type="text" class="form-control" id="apellido2" name="apellido2" placeholder="Segundo apellido">
</div>
<input type="submit" class="btn btn-info" value="Enviar">
</form>
</div>
</div>
</body>
</html>
```



```
</form>
</div>
</div>
</div>
</body>
</html>
```

2- Ahora crearemos la página **“ProcesarC.jsp”** que es la que atraparé los parámetros.

Agregar el siguiente código:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/bootstrap.min.css">
<title>Datos JSTL</title>
</head>
<body>
<div class="container">
<div class="row">
&nbsp;
</div>
<div class="panel panel-primary">
<div class="panel-heading">Imprimiendo parámetros con JSTL</div>
<div class="panel-body">
<p>Nombre: <strong><c:out value="{param.nombre}" /></strong></p>
<p>Primer apellido: <strong><c:out value="{param.apellido1}" /></strong></p>
<p>Segundo apellido: <strong><c:out value="{param.apellido2}" /></strong></p>
</div>
</div>
</div>
</body>
</html>
```

Ejecutar el archivo Datos.jsp y verá el siguiente resultado:

Datos personales

Ingrese su nombre:

Ingrese su primer apellido:

Ingrese su segundo apellido:

Enviar

Agregar datos y hacer clic en botón enviar y observar el siguiente resultado:

Imprimiendo parámetros con JSTL

Nombre: **Rafael**

Primer apellido: **Torres**

Segundo apellido: **Rodriguez**

Ejercicio 3.

- **c:set**

Guarda información en una variable, tiene los siguientes atributos:

Atributo	Descripción	Requerido	Por defecto
value	Información a grabar.	No	Cuerpo
target	Nombre de la variable cuya propiedad será modificado.	No	Ninguno
property	Propiedad a modificar.	No	Ninguna
var	Nombre de la variable en la que guardar el valor.	No	Ninguno
scope	Ámbito de la variable en la que grabar la información (page, request, session o application).	No	page

1- Crear una página JSP llamada **“set1.jsp”** y agregar el siguiente código.

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:set var="variableDePagina" scope="page">
    Esta información se guarda en la página
</c:set>
<c:set var="variableDeSesion" scope="session">
    Esta información se guarda en la sesión
</c:set>
<c:set var="variableDeAplicacion" scope="application">
    Esta información se guarda en la aplicación
</c:set>
<%-- graba la variable cuatro=4 en el ámbito page --%>
<c:set var="cuatro" value="El valor de esta variable es: ${2 + 2}" scope="application"/>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="css/bootstrap.min.css">
        <title>Etiquetas JSTL</title>
    </head>
    <body>
        <div class="container">
            <div class="row">
                &nbsp;
            </div>
            <div class="panel panel-primary">
                <div class="panel-heading">Uso de etiqueta c:set</div>
                <div class="panel-body">
                    <p>${variableDePagina}</p>
                    <p>${variableDeSesion}</p>
                    <p>${variableDeAplicacion}</p>
                    <p>${cuatro}</p>
                </div>
            </div>
            <a href="verambito.jsp">Ir a ver pagina ambito de aplicación</a>
        </div>
    </body>
</html>
```

2- Crear la página JSP llamada **verambito.jsp** y agregar el siguiente código.

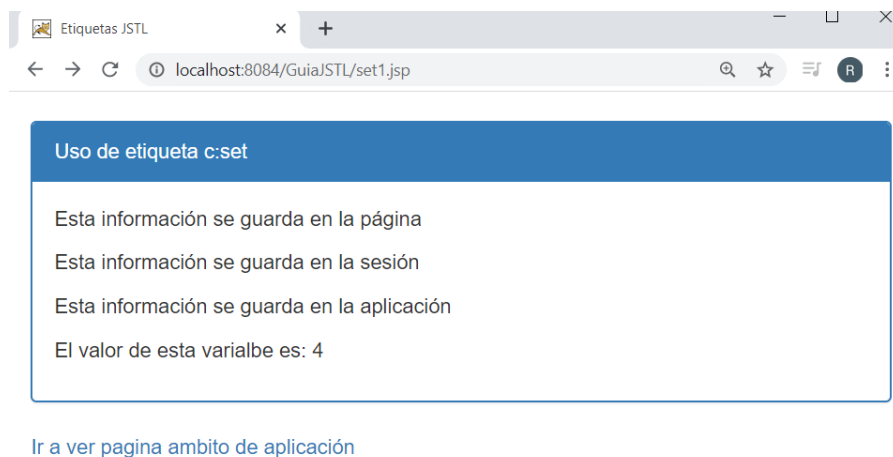
```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

```

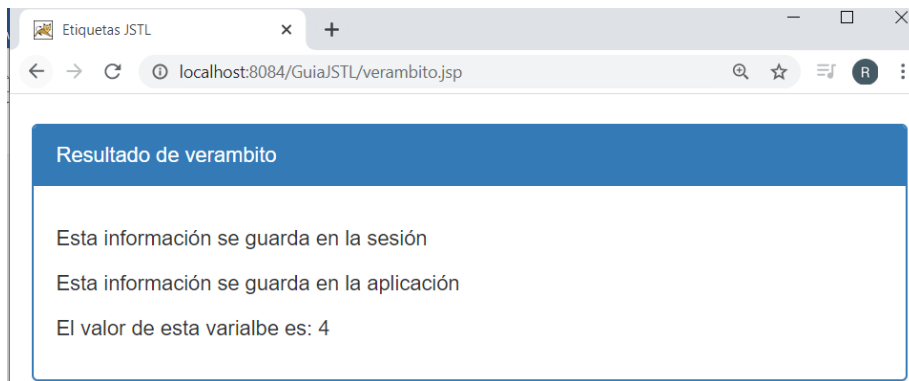
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <title>Etiquetas JSTL</title>
  </head>
  <body>
    <div class="container">
      <div class="row">
        &nbsp;
      </div>
      <div class="panel panel-primary">
        <div class="panel-heading">Resultado de verambito</div>
        <div class="panel-body">
          <p>${variableDePagina}</p>
          <p><c:out value="${variableDeSesion}" /></p>
          <p><c:out value="${variableDeAplicacion}" /><br></p>
          <p><c:out value="${cuatro}" /></p>
        </div>
      </div>
    </div>
  </body>
</html>

```

3- Ejecutar la página **set1.jsp** y observar el siguiente resultado:



El resultado al dar clic en el link, obtener la url resultante y colocarla en un navegador diferente y observar que variables quedan activas.



Para eliminar una variable podemos usar:

```
<c:remove var="nombreVariable" scope="ambito"/>
```

Cuando no se especifica ámbito, la etiqueta busca en todos los ámbitos por turno, desde el más específico al más general (page, request, session, application), hasta encontrar una variable con ese nombre. Si la variable no se encuentra, la etiqueta termina sin error.

Ejercicio 3

▪ c:if

Procesa el cuerpo de la etiqueta si la condición se evalúa como verdadera. La condición se indica en el atributo test.

Ejemplo

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:if test="${empty param.nombre}">
Parámetro 'nombre' no definido.
</c:if>
```

Atributo	Descripción	Requerido	Por defecto
test	Condición a evaluar, solo procesa el cuerpo si es verdadera.	Sí	--
var	Nombre del atributo con el que grabar el resultado booleano de la condición.	No	Ninguno
scope	Ámbito en el que exponer el atributo anterior.	No	page

1. Para ese ejemplo lo que haremos será crear las páginas **"Datosif.jsp"** y **"Procesarif.jsp"**.
2. Agregar el código siguiente en la página **"Datosif.jsp"**.

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/bootstrap.min.css">
<title>Etiquetas JSTL</title>
</head>
<body>
<div class="container-fluid">
<div class="row">
<div class="col-sm-4 col-sm-offset-4">
<div class="row">
<h3>Datos personales</h3>
</div>
<form role="form" name="persona" action="Procesarif.jsp" method="POST">
<div class="form-group">
<label for="nombre">Ingrese su nombre:</label>
<input type="text" class="form-control" name="nombre" id="nombre" placeholder="Nombre">
</div>
<div class="form-group">
<label for="apellido1">Ingrese su primer apellido:</label>
<input type="text" class="form-control" id="apellido1" name="apellido1" placeholder="Primer apellido">
</div>
<div class="form-group">
<label for="apellido2">Ingrese su segundo apellido:</label>
<input type="text" class="form-control" id="apellido2" name="apellido2" placeholder="Segundo apellido">
</div>
<input type="submit" class="btn btn-info" value="Enviar">
</form>
<c:if test="${not empty param.error}">
<div class="alert alert-danger">
<strong>Error!</strong><c:out value="${param.error}" />
<br>
</div>
</c:if>
</div>
</div>
</div>
</body>
</html>
```

3. Como siguiente punto modificaremos el archivo **“Procesarif.jsp”** para que quede de la siguiente manera.

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/bootstrap.min.css">
<title>Etiquetas JSTL</title>
</head>
<body>
<c:if test="${empty param.nombre}">
<c:redirect url="Datosif.jsp">
<c:param name="error" value="Nombre obligatorio"/>
</c:redirect>
</c:if>
<c:if test="${empty param.apellido1}">
<c:redirect url="Datosif.jsp">
<c:param name="error" value="Primer apellido obligatorio"/>
</c:redirect>
</c:if>
<c:if test="${empty param.apellido2}">
  <c:redirect url="Datosif.jsp">
    <c:param name="error" value="Segundo apellido obligatorio"/>
  </c:redirect>
</c:if>
<div class="container">
<div class="row">
&nbsp;
</div>
<div class="panel panel-primary">
<div class="panel-heading">Datos recibidos</div>
<div class="panel-body">
<p>Nombre: <strong><c:out value="${param.nombre}" /></strong></p>
<p>Primer apellido: <strong><c:out value="${param.apellido1}" /></strong></p>
<p>Segundo apellido: <strong><c:out value="${param.apellido2}" /></strong></p>
</div>
</div>
</div>
</body>
</html>
```

4. Ejecutar la página de **“Datosif.jsp”** para ver el resultado.

Datos personales

Ingrese su nombre:

Ingrese su primer apellido:

Ingrese su segundo apellido:

Hacer clic en el botón Enviar, sin haber digitado ningún dato en el formulario

error=Nombre+obligatorio

Datos personales

Ingrese su nombre:

Ingrese su primer apellido:

Ingrese su segundo apellido:

Error! Nombre obligatorio

Archivo: Procesarif.jsp

```
<c:if test="${empty param.nombre}">
<c:redirect url="Datosif.jsp">
<c:param name="error" value="Nombre obligatorio"/>
</c:redirect>
</c:if>
```

Agregue datos en el formulario:

localhost:8084/GuiaJSTL/Datosif.jsp?error=Nombre+obl

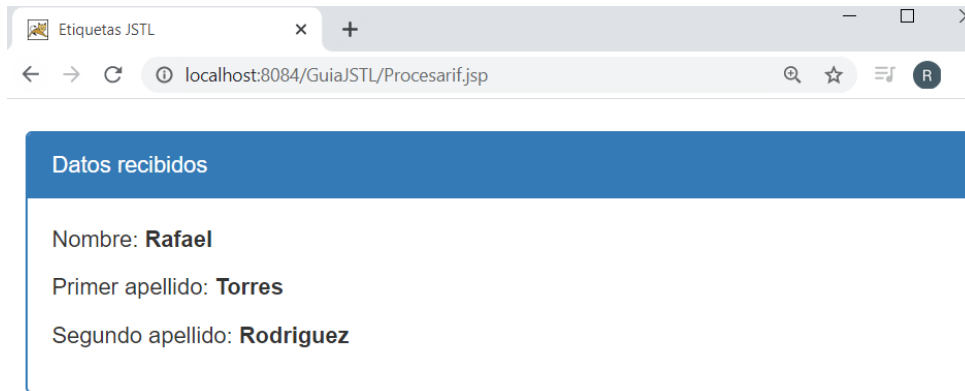
Datos personales

Ingrese su nombre:

Ingrese su primer apellido:

Ingrese su segundo apellido:

Hacer clic en el botón enviar y obtendrá el siguiente resultado:



Etiquetas JSTL

localhost:8084/GuiaJSTL/Procesarif.jsp

Datos recibidos

Nombre: **Rafael**

Primer apellido: **Torres**

Segundo apellido: **Rodriguez**

Ejercicio 4

▪ **c:choose, when y otherwise**

Procesa condiciones múltiples, se procesa el cuerpo del primer when cuya condición especificada en el atributo test se evalúe a cierto. Si ninguna de las condiciones se cumple, se procesa el cuerpo de otherwise en caso de que aparezca.

1- Crear una página JSP llamada **“lenguaje”** y agregar el siguiente código:

```
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/bootstrap.min.css">
<title>Etiquetas JSTL</title>
</head>
<body>
<div class="container">
<div class="row">
<div class="col-sm-4 col-sm-offset-4">
<div class="row">
<h3>Pagina de prueba del uso de choose, when y otherwise</h3>
</div>
<form role="form" name="lenguaje" action="ProcesarC2.jsp" method="POST">
<div class="form-group">
<label for="lenguaje">¿Cuál es tu lenguaje de programación favorito?</label>
<select name="lenguaje" id="lenguaje" class="form-control">
<option value="">--Seleccionar un Lenguaje
```

```

<option value="Java">Java
<option value="C++">C++
<option value="Perl">Perl
</select>
</div>
<input type="submit" class="btn btn-info" value="Enviar">
</form>
</div>
</div>
</div>
</body>
</html>

```

2- Ahora crear la página **“ProcesarC2.jsp”** y agregar el código siguiente:

```

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/bootstrap.min.css">
<title>Etiquetas JSTL</title>
</head>
<body>
<div class="container">
<div class="row">&nbsp;</div>
<div class="panel panel-primary">
<div class="panel-heading">Resultado</div>
<div class="panel-body">
<c:choose>
<c:when test="{param.lenguaje == 'Java'}">
<p>El rey de los lenguaje orientados a objetos</p>
</c:when>
<c:when test="{param.lenguaje == 'C++'}">
<p>Ideal para aprender</p>
</c:when>
<c:when test="{param.lenguaje == 'Perl'}">
<p>Lenguaje de scripting muy potente</p>
</c:when>
<c:otherwise>
<p>No se seleccionó ninguno</p>
</c:otherwise>
</c:choose>
</div>

```

```

</div>
<div class="row">
<a class="btn btn-info" href="lenguaje.jsp">Regresar</a>
</div>
</div>
</body>
</html>

```

3- Ejecutar el archivo lenguaje.jsp y observe los resultados.

Ejercicio 5

▪ c:catch

Con <c:catch> podemos capturar excepciones, sin que se aborte la ejecución de la página al producirse un error. En el atributo var indicamos el nombre de la variable donde debe guardarse la información de la excepción, podremos saber que se ha producido un error comprobando que el valor de esa variable no es nulo.

1- Crear una página llamada **“catch.jsp”** y agregar el siguiente código:

```

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%!int valor=0;%><%--Declarando variable tipo int--%>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/bootstrap.min.css">
<title>Etiquetas JSTL</title>
</head>
<body>
<div class="container">
<div class="row">&nbsp;</div>
<div class="row">
<div class="col-sm-4 col-sm-offset-4">
<c:catch var="error01">
<%
                                valor=Integer.parseInt(request.getParameter("parametro"));
%>
</c:catch>
<c:if test="${not empty error01}">
<div class="alert alert-danger">
<strong>Se produjo un error:</strong> ${error01}
<br>
</div>
</c:if>

```

```

<c:if test="${valor!=0 && empty error01}">
<div class="alert alert-info">
<strong>Valor recibido: <%out.print(valor);%></strong>
<br>
</div>
</c:if>
<form role="form">
<input type="hidden" name="parametro" value="prueba"/>
<input type="submit" class="btn btn-info" value="Enviar 'prueba'"/>
</form>
<form role="form">
<input type="hidden" name="parametro" value="1234"/>
<input type="submit" class="btn btn-info" value="Enviar '1234'"/>
</form>
<form role="form">
<input type="submit" class="btn btn-info" value="No enviar el parámetro"/>
</form>
</div>
</div>
</div>
</body>
</html>

```

2- Ejecutar el archivo y observe los resultados

Ejercicio 6

▪ c:forEach

Permite iterar sobre los elementos siguientes:

- Arrays de objetos o tipos primitivos.
- Instancias de java.util.Collection, java.util.Map, java.util.Iterator,
- java.util.Enumeration.
- Cadenas delimitadas por comas.
- Instancias de javax.servlet.jsp.jstl.sql.Result (resultantes de una consultaSQL con JSTL).

Es posible anidar varias etiquetas c:forEach.

Atributo	Descripción	Requerido	Por defecto
items	Colección sobre la cual hay que iterar.	No	Ninguno
begin	Elemento con el que empezar (0=primero).	No	0
end	Elemento con el que terminar.	No	Último
step	Procesa solo cada step elementos.	No	1 (todos)
var	Nombre del atributo con el que exponer el elemento actual.	No	Ninguno
varStatus	Nombre de la variable con la que exponer el estado de la iteración.	No	Ninguno

La variable `varStatus` tiene propiedades que describen el estado de la iteración:

Atributo	Tipo	Descripción
<code>begin</code>	Número	El valor del atributo <code>begin</code> .
<code>current</code>	Número	El elemento actual.
<code>end</code>	Número	El valor del atributo <code>end</code> .
<code>index</code>	Número	Índice del elemento actual dentro de la colección.
<code>count</code>	Número	Número de iteración (empezando en 1).
<code>first</code>	Booleano	Indica si estamos en la primera iteración.
<code>last</code>	Booleano	Indica si estamos en la última iteración.
<code>step</code>	Número	El valor del atributo <code>step</code> .

1- Crear una página llamada **“foreach.jsp”** y agregar el siguiente código:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/bootstrap.min.css">
<title>Etiquetas JSTL</title>
</head>
<body>
<div class="container">
<div class="row">
<h3>Uso de c:forEach</h3>
</div>
<div class="panel panel-primary">
<div class="panel-heading">&nbsp;</div>
<div class="panel-body">
<c:forEach begin="1" end="24" step="2" var="hour" varStatus="status">
<p><c:out value="{hour}" />
<c:if test="{status.first}">
<strong>Estoy en uno</strong>
</c:if>
<c:if test="{status.count == 5}">
<strong>Estoy en la iteración numero 5</strong>
</c:if>
</p>
</c:forEach>
</div>
</div>
</div>
</body>
</html>
```

- 2- Ejecutar el archivo y vea los resultados

Ejercicio 7

- 1- Crear una página llamada **“ForTokens.jsp”** y agregar el siguiente código:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/bootstrap.min.css">
<title>c:forTokens Demo</title>
</head>
<body>
<div class="container">
<div class="row">
<div class="col-sm-4 col-sm-offset-4">
<div class="row">
<h3>c:forTokens Demo</h3>
</div>
<form role="form" name="forTokensForm" action="ResultTokens.jsp" method="POST">
<div class="form-group">
<label for="delimText">Enter some text with delimiter:</label>
<input type="text" class="form-control" name="delimText" id="delimText">
</div>
<div class="form-group">
<label for="delim">Enter the delimiter:</label>
<input type="text" class="form-control" id="delim" name="delim">
</div>
<input type="submit" class="btn btn-info" value="Tokenize">
</form>
</div>
</div>
</div>
</div>
</body>
</html>
```

- 2- Crear una página llamada **“ResultTokens.jsp”** y agregar el siguiente código

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
```

```

<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/bootstrap.min.css">
<title>c:forTokens Demo</title>
</head>
<body>
<div class="container">
<div class="row">
&nbsp;
</div>
<div class="panel panel-primary">
<div class="panel-heading">Your tokens</div>
<div class="panel-body">
<c:forTokens items="{param.delimText}" delims="{param.delim}" var="myToken">
<p><c:out value="{myToken}" /></p>
</c:forTokens>
</div>
</div>
</div>
</div>
</body>
</html>

```

3- Luego ejecutar la página **ForTokens.jsp** para ver el resultado.

c:forTokens Demo

Enter some text with delimiter:

Enter the delimiter:

Tokenize

Your tokens

Emerson
Ernesto
Torres

Ejercicio 8. Bases de datos

▪ **sql:DataSource**

Atributo	Descripción	Requerido	Por defecto
dataSource	Base de datos a usar.	No	Ninguno
driver	Nombre de la clase JDBC a usar como driver.	No	Ninguno
url	URL de la base de datos.	No	Ninguno
user	Nombre del usuario de la base de datos.	No	Ninguno
password	Password del usuario de la base de datos.	No	Ninguno
var	Nombre de la variable que representa a la base de datos.	No	Ninguno
scope	Ámbito de la variable anterior.	No	page

Ejemplo. Establecer un dataSource por defecto:

```
<sql:setDataSource driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/test"
user="root"
password="" />
```

▪ **sql:query**

Se usa para consultar la base de datos.

Atributo	Descripción	Requerido	Por defecto
sql	Consulta SQL a ejecutar.	No, si ponemos la consulta en el cuerpo de la etiqueta.	Cuerpo
dataSource	Proveedor de conexiones.	No	--
startRow	Primeras filas a ignorar (ej: 10=ignora las primeras 10 filas).	No	0 (Primero)
maxRows	Máximo número de filas.	No	--
var	Nombre de la variable con la que exponer el resultado.	Sí	Ninguno
scope	Ámbito de la variable anterior.	No	page

Esta etiqueta no muestra datos, solo los graba en la variable indicada por var.

El atributo maxRows indica el número por defecto de filas a recuperar. Podemos asignar un valor a este atributo para cambiar el límite de filas, o asignar -1 si no queremos límite.

Las propiedades disponibles son:

- columnName: Lista de nombres de columnas. Podemos acceder a ella con paréntesis cuadrados o iterando sobre ella.
- limitedByMaxRows: Booleano que indica si el resultado contenía más de las filas indicadas por maxRows.
- rows: Acceso a filas usando por nombre.
- rowsByIndex: Acceso a filas por índice.
- rowCount: Número de filas.

Ejemplo

```
<sql:query var="users">
    SELECT * FROM USERS
</sql:query>
```

Forma equivalente a la anterior:

```
<sql:query var="users" sql="SELECT * FROM USERS"/>
```

Suponiendo que el SQL este en una variable:

```
<sql:query var="users" sql="${sql}"/>
```

▪ **sql:update**

Se usa para modificar la base de datos.

Atributo	Descripción	Requerido	Por defecto
sql	Consulta SQL a ejecutar.	No	Cuerpo
dataSource	Proveedor de conexiones.	No	--
var	Nombre de la variable para guardar el número de filas actualizadas.	No	Ninguno
scope	Ámbito de la variable anterior.	No	page

Varios ejemplos

```
<sql:update>
    INSERT INTO  citas
    SET cita = "es reinventar el tornillo, digo.. la rueda, bueno, y el tornillo
    autor = "Luis"
    date = "2004-03-03";
</sql:update>

<sql:update sql="DELETE FROM citas WHERE autor = 'Luis'"/>

<sql:update var="n">
    DELETE FROM citas
    WHERE autor = 'Luis'
</sql:update>

Hemos borrado <c:out value="${n}"/> filas.
```

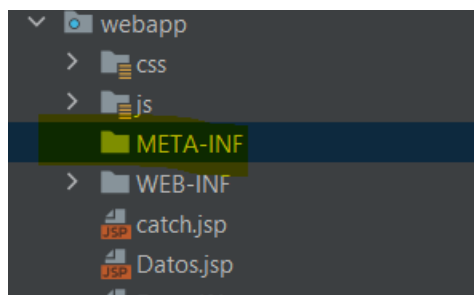
CRUD JSTL

- 1- Lo primero que haremos es crear y configurar un pool de conexiones, esto para no implementar un sqlDataSource con etiquetas de JSTL cada vez que queramos hacer una conexión a una base de datos, Para lo cual debemos implementar un nuevo directorio al cual llamaremos **META-INF**, el cual estará dentro de **Web App**. Finalmente crearemos el archivo de configuración **context.xml**.

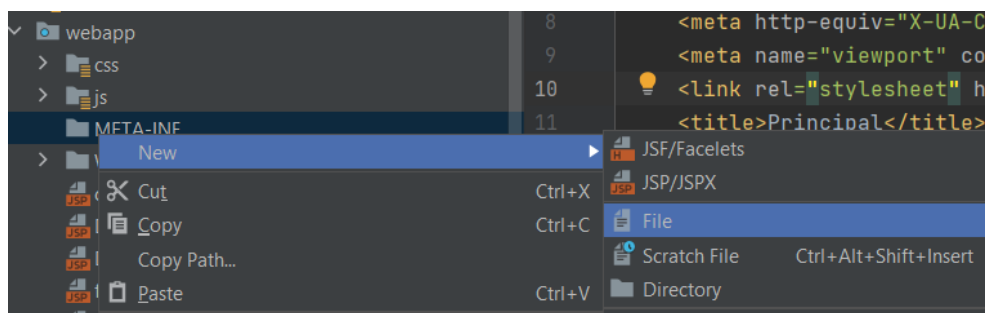
```
<artifactId>junit-jupiter-engine</artifactId>
<version>${junit.version}</version>
<scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>8.0.27</version>
</dependency>

<!-- Fin de importacion para todas las dependencias -->
</dependencies>
```

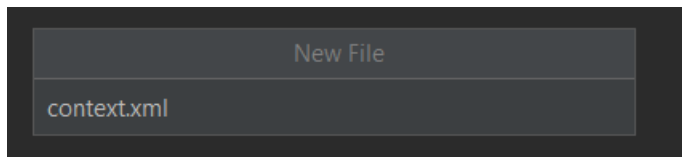
Paso 1. Importaremos el mysqlconnector mediante el archivo pom.xml, agregando la siguiente dependencia. Esto ya se ha hecho en guías pasadas, revisar la documentación si tiene dudas respecto a este paso.



Paso 2. Creación del directorio **META-INF**.



Paso 3. Damos click en **New-> File**



Paso 4. Creamos con todo y su extensión el archivo **context.xml**. Notara que es simplemente un archivo de configuración el cual contendrá toda la información para generar la conexión a la base de datos.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <Context>
3   <Resource name="jdbc/mysql" auth="Container" type="jakarta.sql.DataSource"
4     username="root" password="" driverClassName="com.mysql.cj.jdbc.Driver"
5     url="jdbc:mysql://localhost/personabdd"/>
6 </Context>
```

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

<Context>

  <Resource name="jdbc/mysql" auth="Container" type="jakarta.sql.DataSource"

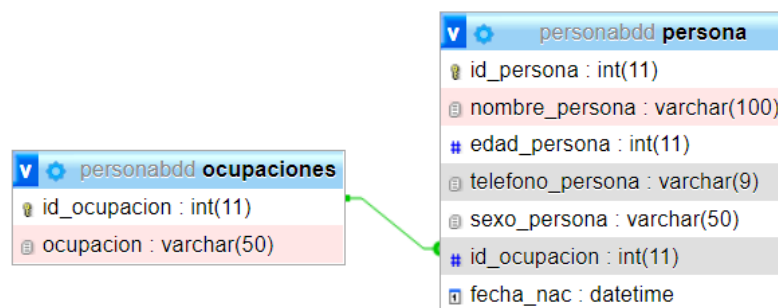
    username="root" password="" driverClassName="com.mysql.cj.jdbc.Driver"

    url="jdbc:mysql://localhost/personabdd"/>

</Context>
```

Paso 5. Copiaremos el siguiente código dentro del archivo **context.xml**.

- 2- Notara que estaremos trabajando con la base de datos proporcionada como recurso en la guía 4 “**personabdd**”.



3- Crear la pagina **principal.jsp**.

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <title>Principal</title>
</head>
<body>
<div class="container">
<h1 class="text-center">Lista personas</h1>
<sql:query var="q1" dataSource="jdbc/mysql">
    SELECT * FROM persona p INNER JOIN ocupaciones o on p.id_ocupacion =
o.id_ocupacion
</sql:query>
    <a class="btn btn-info" href="controller.jsp?operacion=insertar">Insertar persona</a>
    <br>
    <br>
<table class="table table-striped table-bordered table-hover">
    <thead>
    <tr>
        <th>Id</th>
```

```
<th>Nombres</th>

<th>Edad</th>

<th>Sexo</th>

<th>Ocupacion</th>

<th>Telefono</th>

<th>Fecha de nacimiento</th>

<th>Eliminar</th>

<th>Modificar</th>

</tr>

</thead>

<tbody>

<c:forEach var="persona" items="${q1.rows}">

<tr>

<td>${persona.id_persona}</td>

<td>${persona.nombre_persona}

</td>

<td>${persona.edad_persona}

</td>

<td>${persona.sexo_persona}

</td>

<td>${persona.ocupacion}

</td>

<td>${persona.telefono_persona}

</td>

<td>${persona.fecha_nac}
```

```

        </td>

        <td>

            <button class="btn btn-danger"
onclick="alerta('${persona.id_persona}')">Eliminar</button>

        </td>

        <td>

            <a class="btn bg-primary"
href="controllercontroller.jsp?operacion=modificar&id=${persona.id_persona}">Modificar</
a>

            </form>

        </td>

    </tr>

</c:forEach>

</tbody>

</table>

<c:if test="${not empty param.mensaje}">

    <div class="alert alert-success">

        <strong>Correcto! </strong><c:out value="${param.mensaje}"/>

        <br>

    </div>

</c:if>

<script>

    function alerta(id)

    {

        var mensaje;

        var opcion = confirm("Esta seguro de eliminar este registro");

```

```

        if (opcion == true) {
            location.href="controller.jsp?operacion=eliminar&id="+id;
        }
    }
</script>
</div>
</body>
</html>

```

4- Crear el recurso **controller.jsp** y pegar el siguiente código.

```

<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
<html>
<head>
    <title>Controlador</title>
</head>
<body>
<c:if test="${param.operacion == 'eliminar'}">
    <sql:update var="insertar" dataSource="jdbc/mysql">
        DELETE FROM persona WHERE id_persona = ?
        <sql:param value="${param.id}" />
    </sql:update>

    <c:redirect url="principal.jsp">
        <c:param name="mensaje" value="Persona eliminada con exito"/>

```

```
</c:redirect>

</c:if>

<c:if test="${param.operacion == 'insertar'}">

    <c:redirect url="formulario.jsp">

        <c:param name="cabecera" value="Insertar persona"/>

        <c:param name="operacion" value="insertarinf"/>

    </c:redirect>

</c:if>

<c:if test="${param.operacion == 'insertarinf'}">

    <sql:update var="insertar" dataSource="jdbc/mysql">

        INSERT INTO persona(nombre_persona, edad_persona, telefono_persona, sexo_persona,
id_ocupacion, fecha_nac) VALUES (?, ? , ?, ? ,?, ?)

        <sql:param value="${param.nombre}"/>

        <sql:param value="${param.edad}"/>

        <sql:param value="${param.telefono}"/>

        <sql:param value="${param.sexo}"/>

        <sql:param value="${param.idocupacion}"/>

        <sql:param value="${param.fecha}"/>

    </sql:update>

    <c:redirect url="principal.jsp">

        <c:param name="mensaje" value="Persona ingresada con exito"/>

    </c:redirect>

</c:if>

</body>

</html>
```


5- Finalmente crearemos **formulario.jsp**.

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Formulario</title>
    <link rel="stylesheet" href="css/bootstrap1.min.css">
</head>
<body>
<div class="container">
    <h1 class="text-center">${param.cabecera}</h1>
    <form action="controller.jsp" method="POST">
        <input type="hidden" name="id" value="${not empty param.id}">
        <input type="hidden" value="${param.operacion}" name="operacion" id="operacion">
        <div class="form-group">
            <label for="nombre">Ingrese el nombre de la persona:</label>
            <div class="input-group">
                <input type="text" class="form-control" value="${param.nombre}"
name="nombre" id="nombre"
                placeholder="Ingresa el nombre" required/>
            </div>
        </div>
        <div class="form-group">
            <label for="edad">Ingrese la edad de la persona:</label>
```

```
<div class="input-group">

    <input type="number" class="form-control" id="edad" value="{param.edad}"
name="edad"

        placeholder="Ingresa la edad" required/>

</div>

</div>

<div class="form-group">

    <label for="telefono">Ingresa el telefono de la persona:</label>

    <div class="input-group">

        <input type="tel" class="form-control" id="telefono" value="{param.telefono}"
name="telefono"

            placeholder="Ingresa el telefono" required/>

        </div>

    </div>

</div>

<div class="form-group">

    <label for="sexo">Ingresa el sexo de la persona:</label>

    <div class="input-group">

        <select name="sexo" id="sexo" class="form-control" required>

            <option value="Masculino">Maculino</option>

            <option value="Femenino">Femenino</option>

        </select>

    </div>

</div>

<div class="form-group">

    <label for="ocupacion">Ingresa la ocupacion de la persona:</label>

    <div class="input-group">
```

```

        <select name="idocupacion" id="ocupacion" class="form-control" required>
            <option value="1">Doctor</option>
            <option value="2">Emprendedor</option>
            <option value="3">Profesor</option>
        </select>
    </div>
</div>
<div class="form-group">
    <label for="fecha">Ingrese la fecha de nacimiento de la persona:</label>
    <div class="input-group">
        <input type="date" class="form-control" id="fecha" value="{param.fecha}"
name="fecha"
        placeholder="Ingresa la fecha" required/>
    </div>
</div>
<div style="margin-left: 30%;">
    <input type="submit" class="btn btn-success col-md-6 " value="Enviar"/>
</div>
</form>
</div>
</body>
</html>

```

6- Ahora correremos página **principal.jsp** y ver los resultados.

Lista personas

Insertar persona

Id	Nombres	Edad	Sexo	Ocupacion	Telefono	Fecha de nacimiento	Eliminar	Modificar
1	Alejandro Pineda Saravia	45	Masculino	Profesor	7722-4455	1999-02-05	Eliminar	Modificar
2	Fernando Calderón	21	Masculino	Emprendedor	7667-7890	2001-05-07	Eliminar	Modificar
3	Emerson Torres	22	Masculino	Profesor	7123-9800	1999-08-03	Eliminar	Modificar

Insertar persona

Ingrese el nombre de la persona:

Ingrese la edad de la persona:

Ingrese el telefono de la persona:

Ingrese el sexo de la persona:

Ingrese la ocupacion de la persona:

Ingrese la fecha de nacimiento de la persona:



Enviar

IV. EJERCICIOS COMPLEMENTARIOS

- A partir de su proyecto, **manipular las opciones de modificar y eliminar elementos utilizando 2 de las tablas de su base de datos** haciendo uso de JSTL y Pool de conexiones
- En el ejercicio 8 falta implementar la función modificar, todo esta preparado en la vista, realice de forma creativa dicha función.

V. REFERENCIA BIBLIOGRÁFICA

- <http://jimenez303.blogspot.com/>