Facultad: **Ingeniería**
Escuela: **Computación**
Asignatura: **Java Avanzado**

Guía N° 10

Tema:JSTL

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-Llopart 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 Plataform.

Etiquetas JSTL

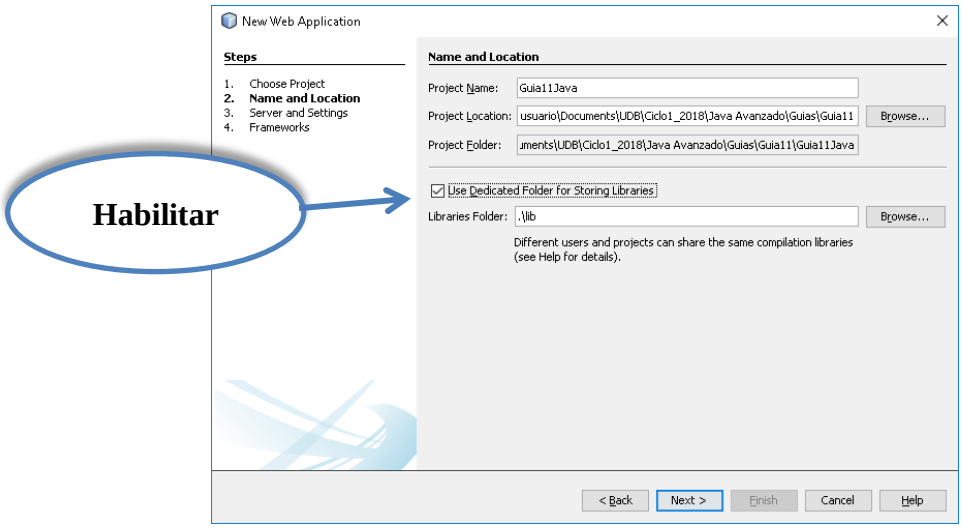
Una etiqueta JSTL corresponde a una acción; llamándolas acción nos indica que añaden comportamiento dinámico página estática.

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 **“Guia10Java”**, tomar en cuenta que para este proyecto debe dejar chequeada la casilla **“Use Dedicated Folder for Storing Libraries”** para poder copiar las librerías al proyecto, ver la siguiente figura**.**

Instalación y configuración del JSTL



Ejercicio 1

La librería JSTL es distribuida como un conjunto de archivos JAR que simplemente tenemos que agregar en el classpath del contenedor de servlets.

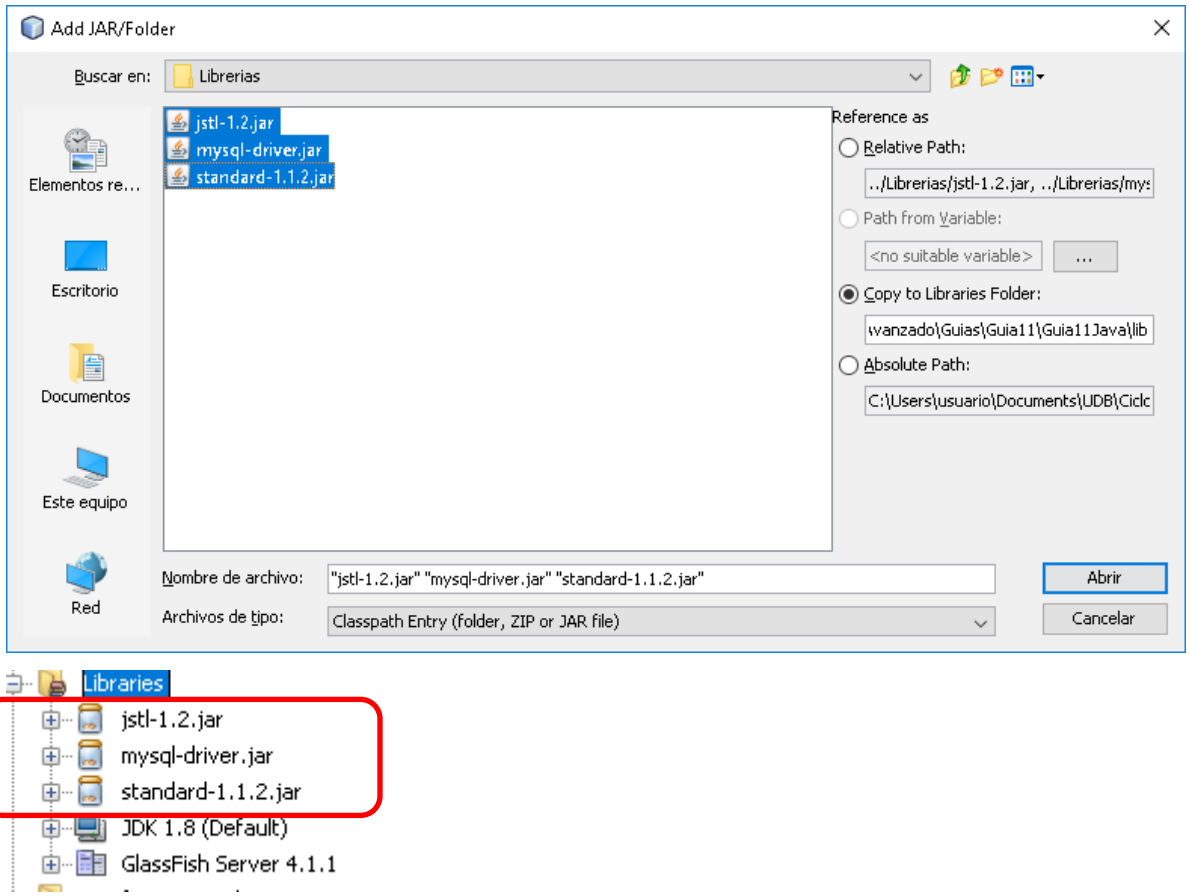
1. Agregar las **librerias "jstl.jar , standard.jar y mysql-connector"** las puede incluir de las que tiene disponibles de netbeans o bien de lasproporcionadas por el docente. Si usted agregará las librerías a partir de los .jar, debe hacerlo como se muestra en la siguiente figura y seleccionar la opción **"Copy to Libraries Folder"** para que estas sean copiadas al proyecto.

   Los .jar lo puede descargar de las siguientes páginas web:

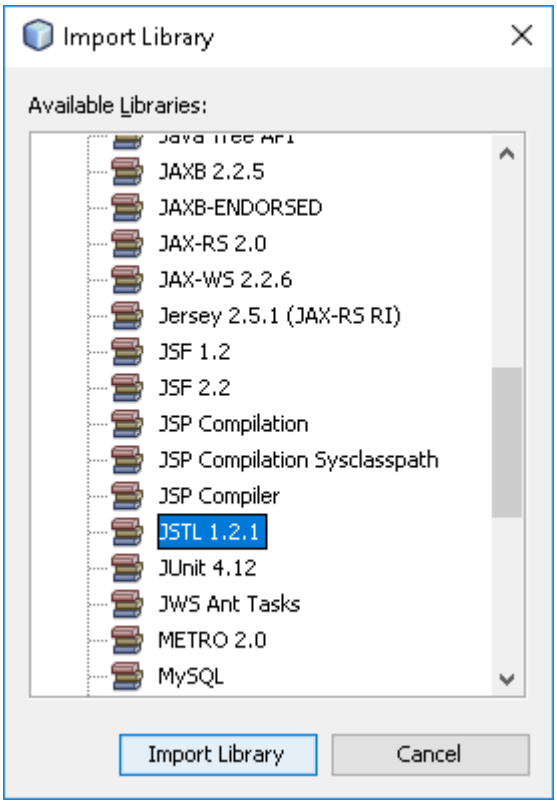
   http://repo2.maven.org/maven2/javax/servlet/jstl/1.2/

   http://repo2.maven.org/maven2/taglibs/standard/1.1.2/

   http://icursos.net/cursos/JavaJDBC/drivers/MySQL-driver.jar



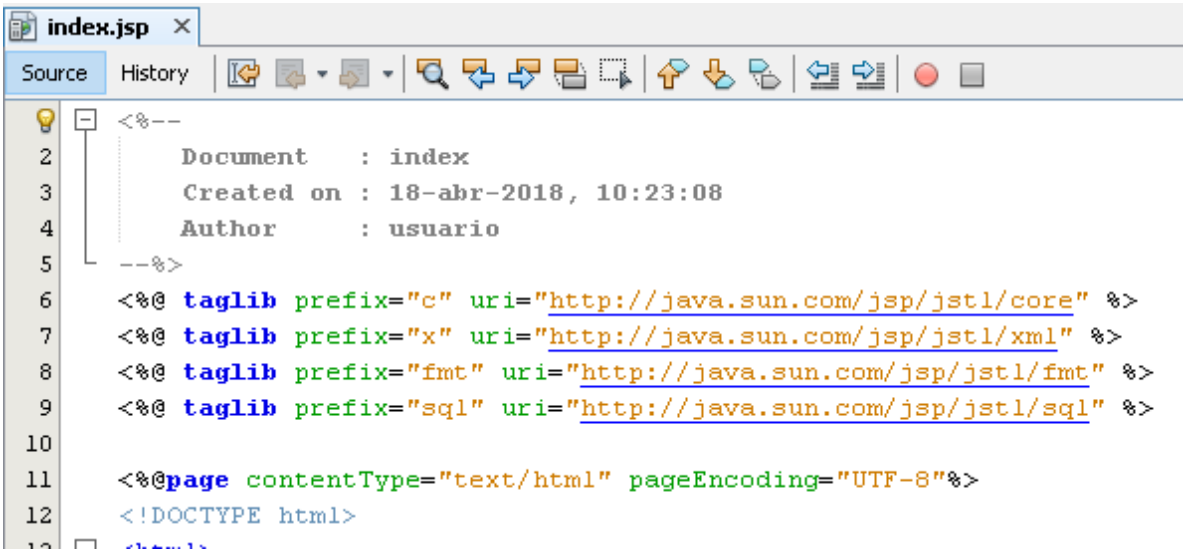
Como NetBeans cuenta con estas librerías ya disponibles.Usted puede incluirlas al proyecto, evitando hacer el paso anterior.



2. Borrar del proyecto Web el archivo index.html
3. Agregar un archivo de tipo JSP, con el nombre index

4. 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" %>
```



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" %>

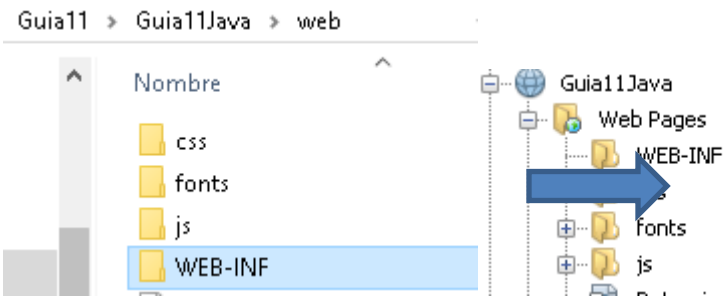
Esta librería implementa acciones de propósito general, como mostrar información, crear y modificar variables de distinto ámbito y tratar excepciones. Veremos algunas de las etiquetas más comunes.

- **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

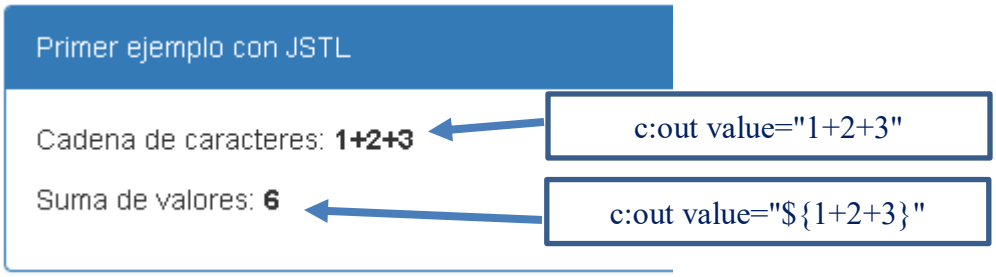
NOTA: Exporte al proyecto todos los archivos correspondientes para hacer uso de **Bootstrap.**



5. Modificar la página de **“index.jsp”** incluir 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>Ejemplo JSTL</title>
</head>
<body>
<div class="container">
<div class="row">
 
</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>
```

6. 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>
</div>
</body>
</html>
```

2. Ahora crearemos la página **“ProcesarC.jsp”** que es la que atrapará 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">
 
</div>
<div class="panel panel-primary">
<div class="panel-heading">Imprimiendo par&aacute;metros 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>
```

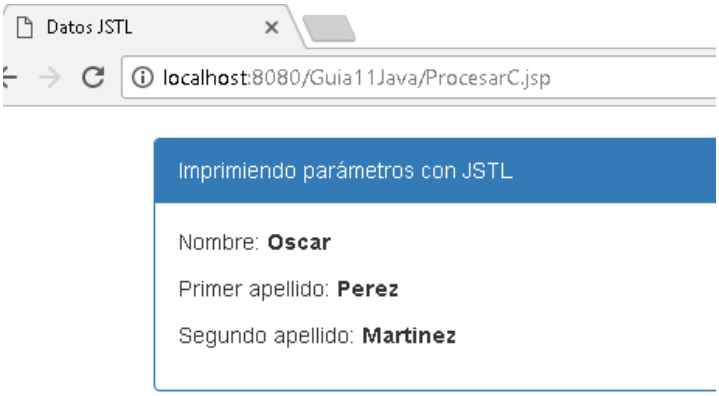
- **c:set**

  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

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

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

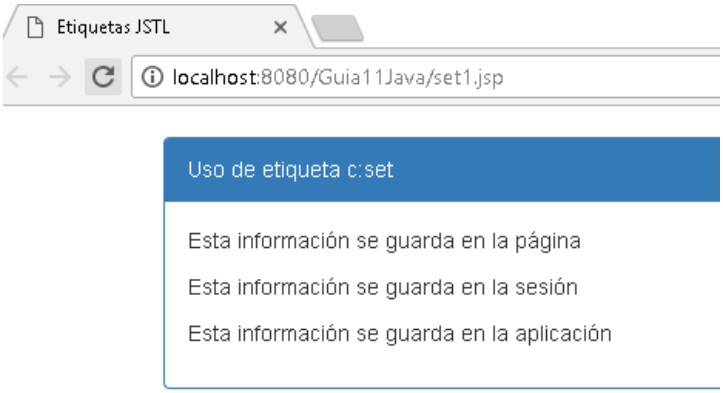


4. Crear una página JSP llamada **"set1.jsp"** y agregar el siguiente código.

```
<%@ 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>
<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>
<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>
</div>
</div>
</div>
</body>
</html>
```

5. Ejecutar la página y observar el siguiente resultado:



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 serácrear laspáginas**"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>
```

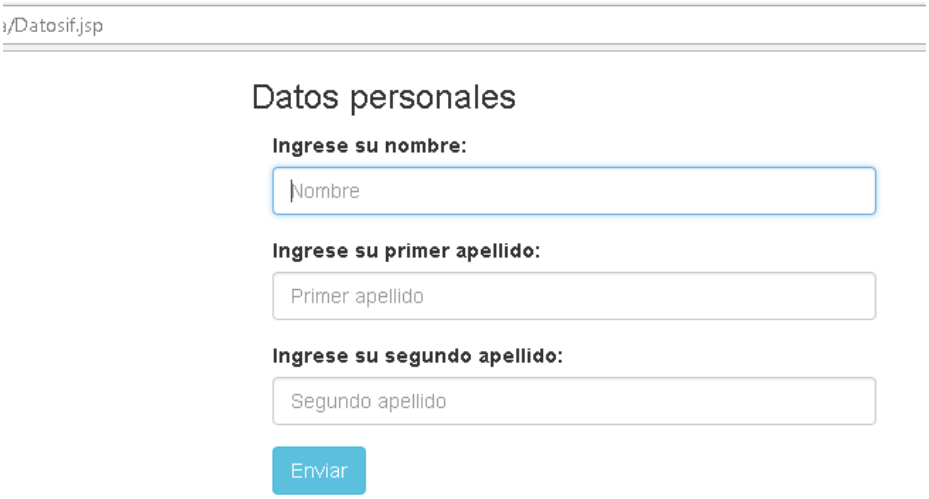
```
<input type="text" class="form-control" name="nombre" id="nombre" placeholder="Nombre">
</div>
<div class="form-group">
<label for="apellido1">Ingrese su primer apellido:</label>
<input type="text" class="form-control" id="apellido1" name="apellido1" placeholder="Primer
apellido">
</div>
<div class="form-group">
<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>
<div class="container">
<div class="row">
 
</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.

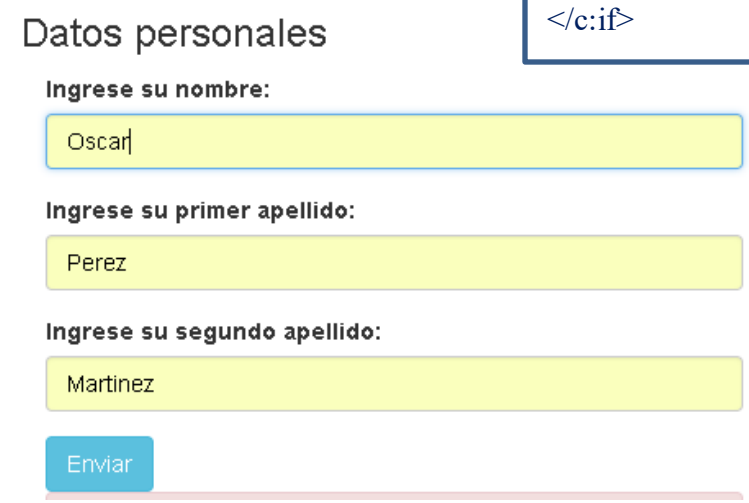


Hacer clic en el botón Enviar, sin haber digitado ningún dato en el formulario

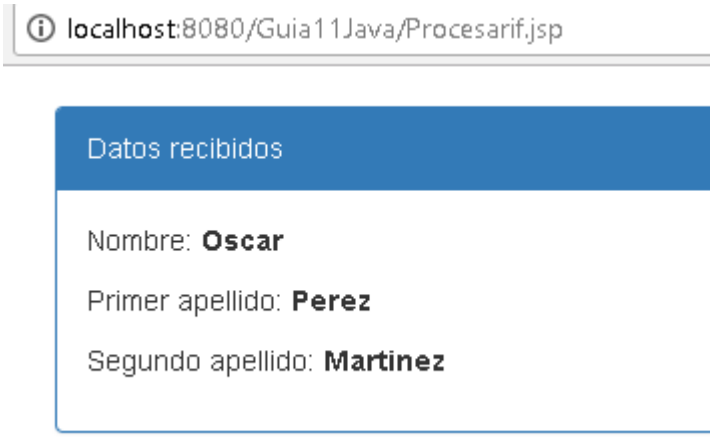
Archivo: Procesarif.jsp

```
<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:



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



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"** yagregar 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>
```

3. 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"> </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>
```

4. 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"> </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
begin	Número	El valor del atributo begin.
current	Número	El elemento actual.
end	Número	El valor del atributo end.
index	Número	Índice del elemento actual dentro de la colección.
count	Número	Número de iteración (empezando en 1).
first	Booleano	Indica si estamos en la primera iteración.
last	Booleano	Indica si estamos en la última iteración.
step	Número	El valor del atributo step.

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"> </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>
</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">
 
</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>
</body>
</html>
```

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

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

indicadaspor 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.
```

Ingreso y muestra de datos con JSTL

1. Ahora crearemos una nueva aplicaciónWeb,por ejemplo: Guia10BaseDatos, debe seleccionar el servidor de **Apache Tomcat.**
   Para ingresar datos y mostrar los datos ingresados, pero para ello realizaremos la conexión creando un **pool de conexiones**, para ello ir al archivo **context.xml** que se encuentra dentro de la carpeta **META-INF**,se deberá editar para que quede de la siguiente manera:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context crossContext="true" debug="5" path="/Guia10BaseDatos" reloadable="true">
<Logger className="org.apache.catalina.logger.FileLogger" prefix="localhost_." suffix=".txt" timestamp="true"/>
<Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" name="jdbc/mysql" password="" type="javax.sql.DataSource" url="jdbc:mysql://localhost:3306/empleados" username="root"/>
</Context>
```

2. Crear la siguiente base de datos (ejecutar las sentencias):

```
create database empleados;

use empleados;

create table empleados (
id varchar(12),
nombres varchar(50),
apellido1 varchar(50),
apellido2 varchar(50),
edad int
);
```

3. Como siguiente paso crearemos un JavaBeans llamado **“CodigoBean”** para que genere el código de la persona, este deberá quedar en el paquete **“sv.edu.udb.guia10”** y contendrá el siguiente código:

```
package sv.edu.udb.guia10;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class CodigoBean {
   private String apellido1;
   private String apellido2;
   private int cantidad_registros;
   private String anio_actual;
   private String soloLetra;
   private String soloLetra2;
   private String cod;

   public String getApellido1() {
return apellido1;
}

   public void setApellido1(String apellido1) {
      this.apellido1 = apellido1;
}
   public String getApellido2() {
      return apellido2;
}

   public void setApellido2(String apellido2) {
      this.apellido2 = apellido2;
   }

   public int getCantidad_registros() {
      return cantidad_registros;
   }

   public void setCantidad_registros(int cantidad_registros) {
      this.cantidad_registros = cantidad_registros;
}
    public String getCod(){
       Date d = new Date();
       SimpleDateFormat anio=new SimpleDateFormat("yy");
DecimalFormat dosDigitos = new DecimalFormat("0000");
        anio_actual=anio.format(d);
        soloLetra = apellido1.substring(0,1);
```

```
        soloLetra2 = apellido2.substring(0,1);
        int aumentregistros=cantidad_registros+1;
        String regitros=String.valueOf(dosDigitos.format(aumentregistros));
        String codigo=soloLetra+soloLetra2+anio_actual+regitros;
        return codigo;
    }
}
```

4. Crear una página JSP llamada **“Informacion.jsp”** yagregar 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="ProcesarInfo.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>
<div class="form-group">
<label for="edad">Ingrese su edad:</label>
```

```
<input type="text" class="form-control" name="edad" id="edad" placeholder="Edad">
</div>
<input type="submit" class="btn btn-info" value="Enviar">
</form>
</div>
</div>
</div>
</body>
</html>
```

5. Y por último creamos la página **“ProcesarInfo.jsp”** con el siguiente contenido:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<jsp:useBean id="cod" scope="page" class="sv.edu.udb.guia10.CodigoBean"/>
<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>
<c:set var="nombre" value="${param.nombre}"/>
<c:set var="apellido1" value="${param.apellido1}"/>
<c:set var="apellido2" value="${param.apellido2}"/>
<c:set var="edad" value="${param.edad}"/>
<sql:query var="q1" dataSource="jdbc/mysql">
        SELECT * from empleados
</sql:query>
<c:set var="nreg" value="${q1.rowCount}"/>
<c:set target="${cod}" property="apellido1" value="${apellido1}"/>
<c:set target="${cod}" property="apellido2" value="${apellido2}"/>
<c:set target="${cod}" property="cantidad_registros" value="${nreg}"/>
<c:set var="codigoUsu" value="${cod.cod}"/>

<div class="container">
<div class="row">&amp;nbsp;</div>
<div class="panel panel-primary">
<div class="panel-heading">Datos recibidos</div>
<div class="panel-body">
```

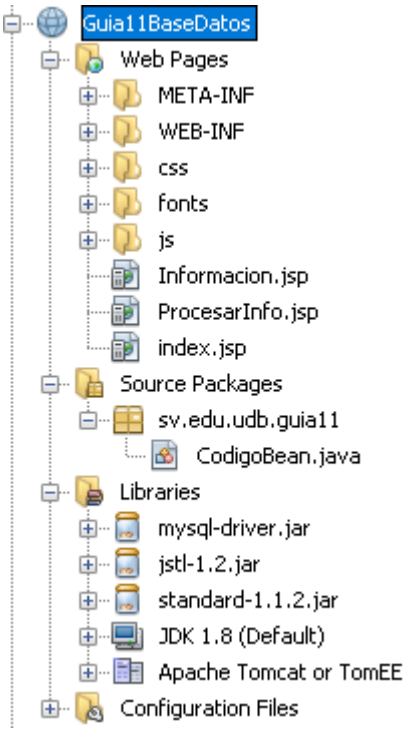
```
<p>Nombre: <strong><c:out value="${nombre} " /></strong></p>
<p>Primer apellido: <strong><c:out value="${apellido1} " /></strong></p>
<p>Segundo apellido: <strong><c:out value="${apellido2} " /></strong></p>
<p>Edad: <strong><c:out value="${edad} " /></strong></p>
<p>Codigo: <strong><c:out value="${codigoUsu} " /></strong></p>
</div>
</div>
</div>
<sql:update var="insertar" dataSource="jdbc/mysql">
insert into empleados  (id,nombres,apellido1,apellido2,edad) values (?,?,?,?,?)
<sql:param value="${codigoUsu}"/>
<sql:param value="${nombre}"/>
<sql:param value="${apellido1}"/>
<sql:param value="${apellido2}"/>
<sql:param value="${edad}"/>
</sql:update>

<sql:query var="q1" dataSource="jdbc/mysql">
        SELECT * from empleados
</sql:query>
<div class="row col-md-3"></div>
<div class="row col-md-6">
<table class="table table-striped table-bordered table-hover">
<thead>
<tr>
<th>Id</th>
<th>Nombres</th>
<th>Apellidos</th>
<th>Edad</th>
</tr>
</thead>
<tbody>
<c:forEach var="name" items="${q1.rows}">
<tr>
<td><c:out value="${name.id}"/></td>
<td><c:out value="${name.nombres}"/></td>
<td><c:out value="${name.apellido1} ${name.apellido2}"/></td>
<td><c:out value="${name.edad}"/></td>
</tr>
</c:forEach>
</tbody>
</table>
</div>
```

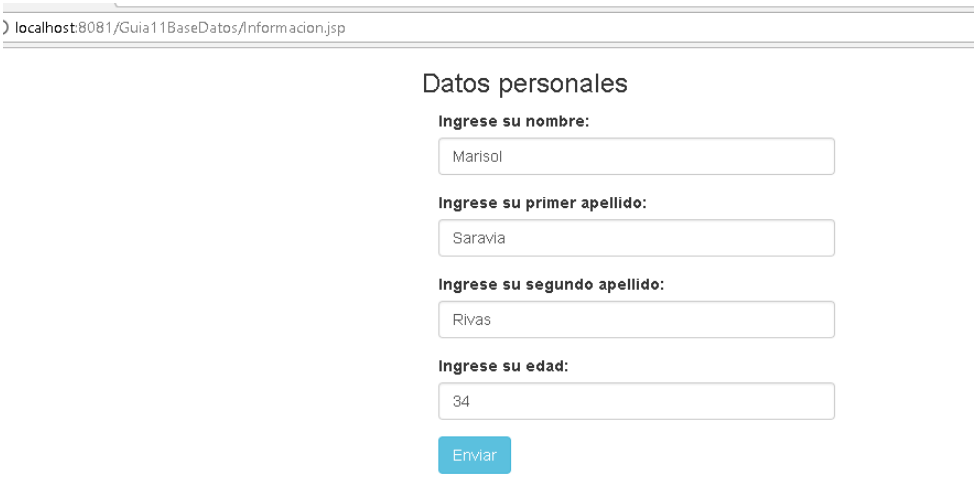
```
<div class="row col-md-3"></div>
</body>
</html>
```

NOTA: IMPORTAR LAS LIBRERÍAS PARA JDBC y JSTL A SU PROYECTO.

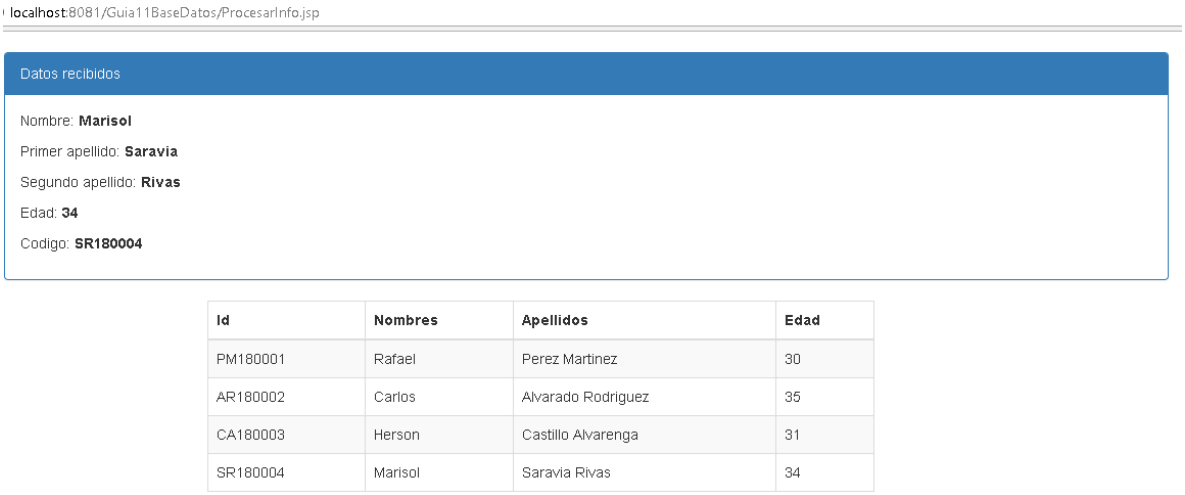
Al final su proyecto debe tener la siguiente estructura:



6. Correr la página llamada "**Informacion.jsp"** y ver el resultado.



Hacer clic en botón Enviar



Ejercicio 9.

Internacionalización con JSTL

La internacionalización es el proceso de diseño de un producto para que el mismo pueda adaptarse a varios idiomas y regiones sin cambios de ingeniería. Ello asegura que el producto funcione en más de un idioma.

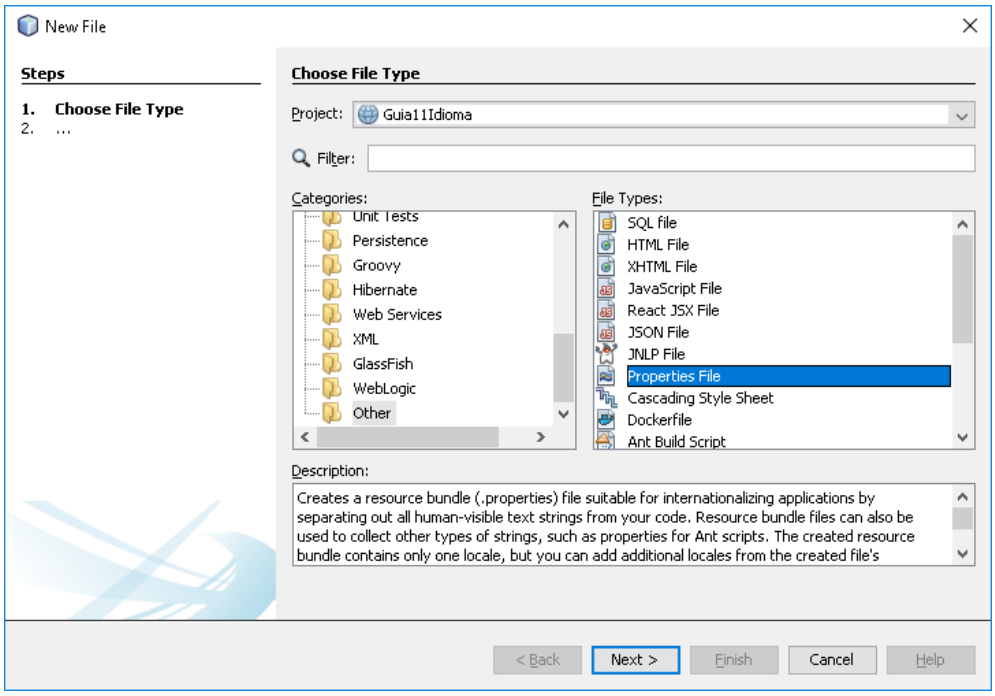
1. Crear un nuevo proyecto Web con el nombre Guia10Idioma
2. Deberá crear una página que se llame **"internacionalizacion.jsp"** y agregar el siguiente código:

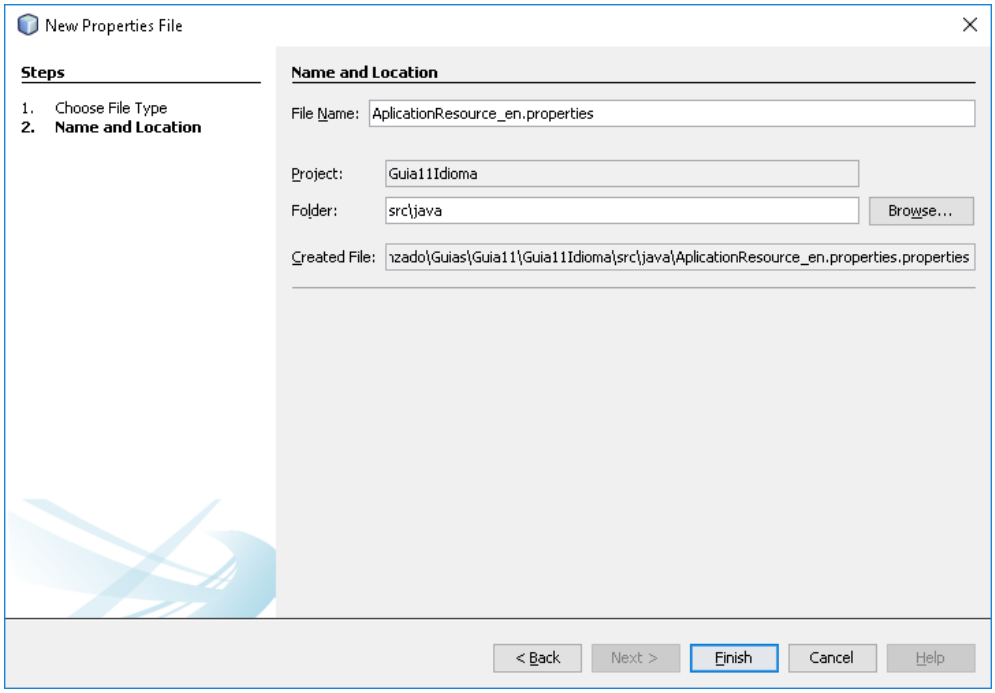
```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
<link href="css/tabla1.css" rel="stylesheet" type="text/css" />
</head>
<body>
<h2><fmt:message key="label.datos"/></h2>
<form name="persona" action="ProcesarInfo.jsp" method="post">
<table border="0">
<tbody>
<tr>
<td><fmt:message key="label.nombre"/></td>
<td><input type="text" name="nombre" value="" /></td>
</tr>
<tr class="odd">
<td><fmt:message key="label.apellido1"/></td>
<td><input type="text" name="apellido1" value="" /></td>
</tr>
```

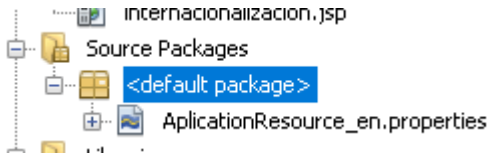
```
<tr>
<td><fmt:message key="label.apellido2"/></td>
<td><input type="text" name="apellido2" value="" /></td>
</tr>
<tr class="odd">
<td><fmt:message key="label.edad"/></td>
<td><input type="text" name="edad" value="" /></td>
</tr>
<tr>
<td colspan="2"><input type="submit" value="<fmt:message key="label.ingresar"/>"></td>
</tr>
</tbody>
</table>
</form>
<br/>
<center>
<jsp:useBean id="now" class="java.util.Date" />
<fmt:formatDate value="${now}" pattern="dd 'de' MMMM 'de' yyyy, hh:mm a." />
</center>
</body>
</html>
```

3. Ahora deberá crear 2 archivos con extensión **.properties** que serán llamados así:
    - AplicationResource_en.properties
    - AplicationResource_es.properties

    Para ello deberá dar clic derecho sobre el paquete del proyecto, seleccionar la opción **Other**, en donde aparecerá una pantalla como la siguiente en la cual seleccionará**Other→Properties File**


New File
Steps
1. Choose File Type
2. ...
Choose File Type
Project: Guia11Idioma
Filter:
Categories:
Unit Tests
Persistence
Groovy
Hibernate
Web Services
XML
GlassFish
WebLogic
Other
File Types:
SQL file
HTML File
XHTML File
JavaScript File
React JSX File
JSON File
JNLP File
Properties File
Cascading Style Sheet
Dockerfile
Ant Build Script
Description:
Creates a resource bundle (.properties) file suitable for internationalizing applications by separating out all human-visible text strings from your code. Resource bundle files can also be used to collect other types of strings, such as properties for Ant scripts. The created resource bundle contains only one locale, but you can add additional locales from the created file's
< Back
Next >
Finish
Cancel
Help


New Properties File
Steps
1. Choose File Type
2. Name and Location
Name and Location
File Name: AplicationResource_en.properties
Project: Guia11Idioma
Folder: src\java
Browse...
Created File: ızado\Guias\Guia11\Guia11Idioma\src\java\AplicationResource_en.properties.properties
< Back
Next >
Finish
Cancel
Help

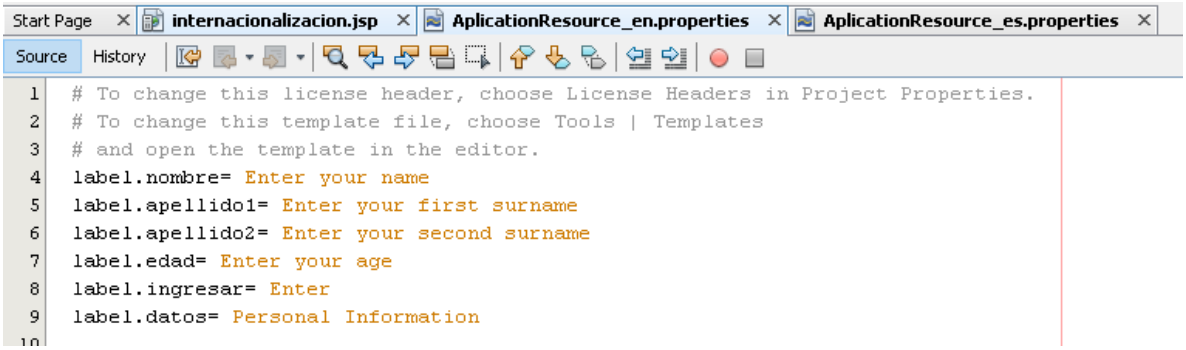

internacionalizacion.jsp
Source Packages
<default package>
AplicationResource_en.properties

Seguir el asistente para crear el archivo respetando los nombres que se le han proporcionado.

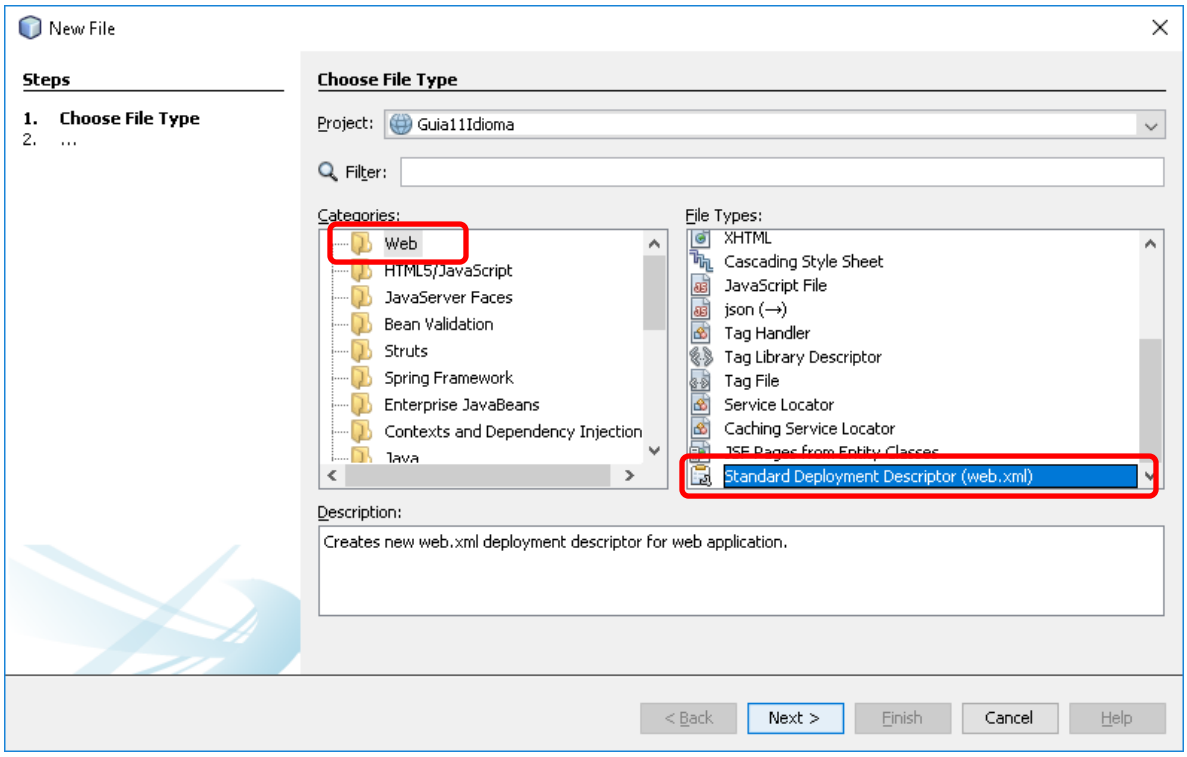
4. Como siguiente punto deberemos agregar cada identificador en los archivos .properties como se muestra en la siguiente tabla.

AplicationResource_es.properties	AplicationResource_en.properties
label.nombre=Ingrese su nombre	label.nombre= Enter your name
label.apellido1=Ingrese su primer apellido	label.apellido1= Enter your first surname
label.apellido2=Ingrese su segundo apellido	label.apellido2= Enter your second surname
label.edad=Ingrese su edad	label.edad= Enter your age
label.ingresar=Ingresar	label.ingresar= Enter
label.datos=Datos personales	label.datos= Personal Information

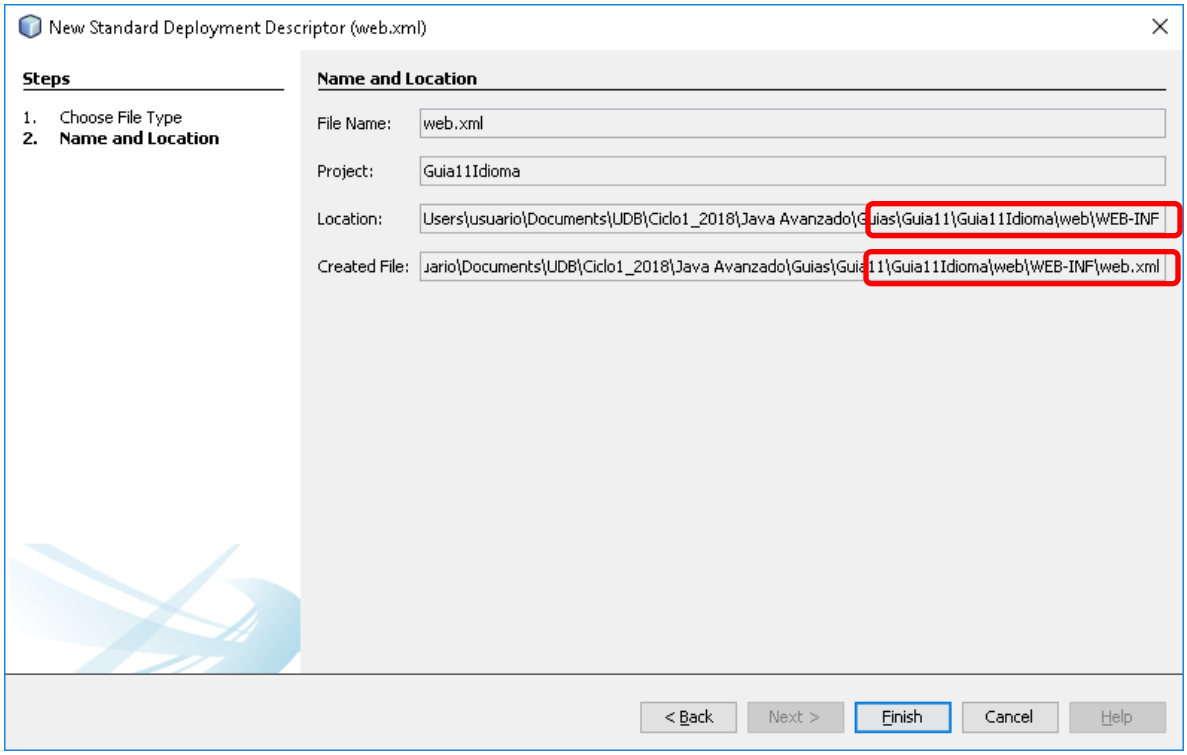
Así como se ve a continuación:



5. Crear el archivo web.xml en la carpeta \Guia10idioma\web



Hacer clic en Next



Hacer clic en Finish

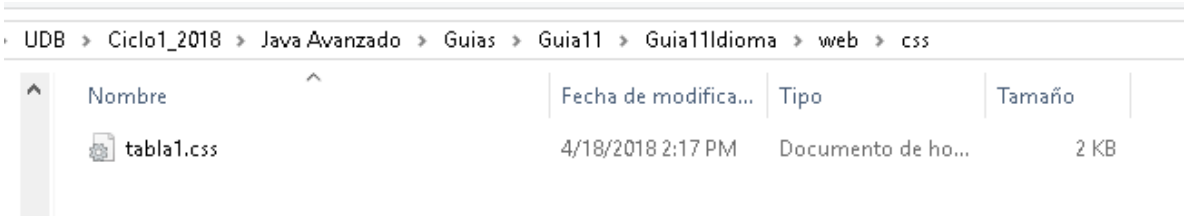
6. Ahora deberá agregar lo siguiente al archivo **“web.xml”**:

```
<context-param>
<param-name>javax.servlet.jsp.jstl.fmt.localizationContext</param-name>
<param-value>AplicationResource</param-value>
</context-param>
```

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/n
         version="3.1">
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <context-param>
    <param-name>javax.servlet.jsp.jstl.fmt.localizationContext</param-name>
    <param-value>AplicationResource</param-value>
  </context-param>

</web-app>
```

7. Para dar una buena presentación a la tabla que contiene el formulario crear una carpeta que se llame “**css**” y un archivo que se llame **“tabla1.css”**



8. Agregar lo siguiente al archivo tabla1.css:

```
table {
        width: 650px;
        border-collapse:collapse;
        border:1px solid #FFCA5E;
}
caption {
        font: 1.8em/1.8em Arial, Helvetica, sans-serif;
        text-align: left;
        text-indent: 10px;
```

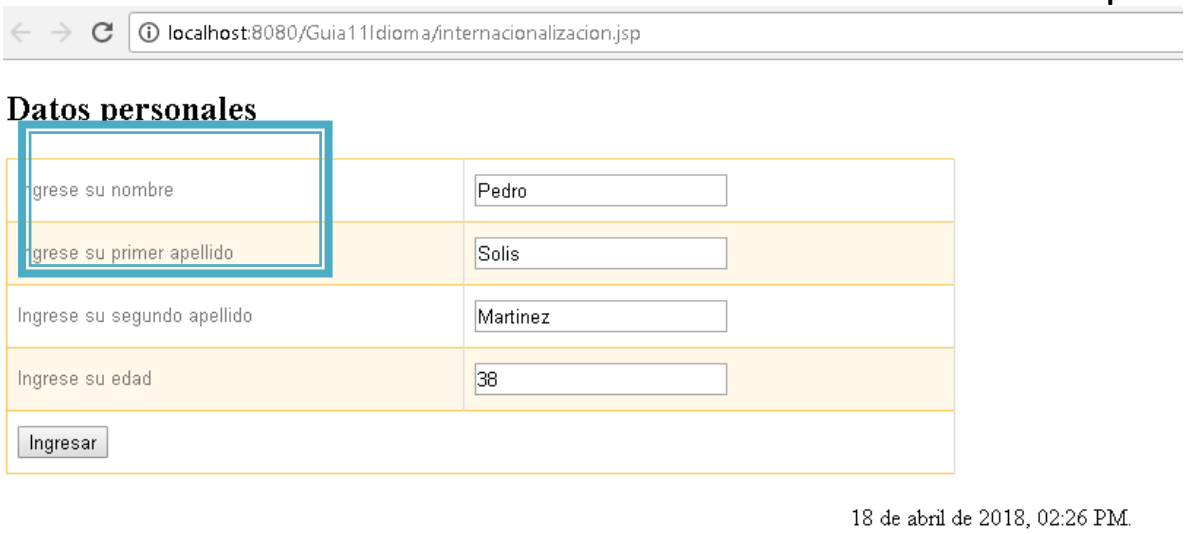
```
	background: url(bg_caption.jpg) right top;
	height: 45px;
	color: #FFAA00;
}
thead th {
	background: url(bg_th.jpg) no-repeat right;
	height: 47px;
	color: #FFFFFF;
	font-size: 0.8em;
	font-weight: bold;
	padding: 0px 7px;
	margin: 20px 0px 0px;
	text-align: left;
	border-right: 1px solid #FCF1D4;
}
tbody tr {
background: url(bg_td1.jpg) repeat-x top;
}
tbody tr.odd {
	background: #FFF8E8 url(bg_td2.jpg) repeat-x;
}

tbody th,td {
	font-size: 0.8em;
	line-height: 1.4em;
	font-family: Arial, Helvetica, sans-serif;
	color: #777777;
	padding: 10px 7px;
	border-top: 1px solid #FFCA5E;
	border-right: 1px solid #DDDDDD;
	text-align: left;
}
a {
	color: #777777;
	font-weight: bold;
	text-decoration: underline;
}
a:hover {
	color: #F8A704;
	text-decoration: underline;
}
tfoot th {
	background: url(bg_total.jpg) repeat-x bottom;
```

```
        color: #FFFFFF;
        height: 30px;
}
tfoot td {
        background: url(bg_total.jpg) repeat-x bottom;
        color: #FFFFFF;
        height: 30px;
}
```

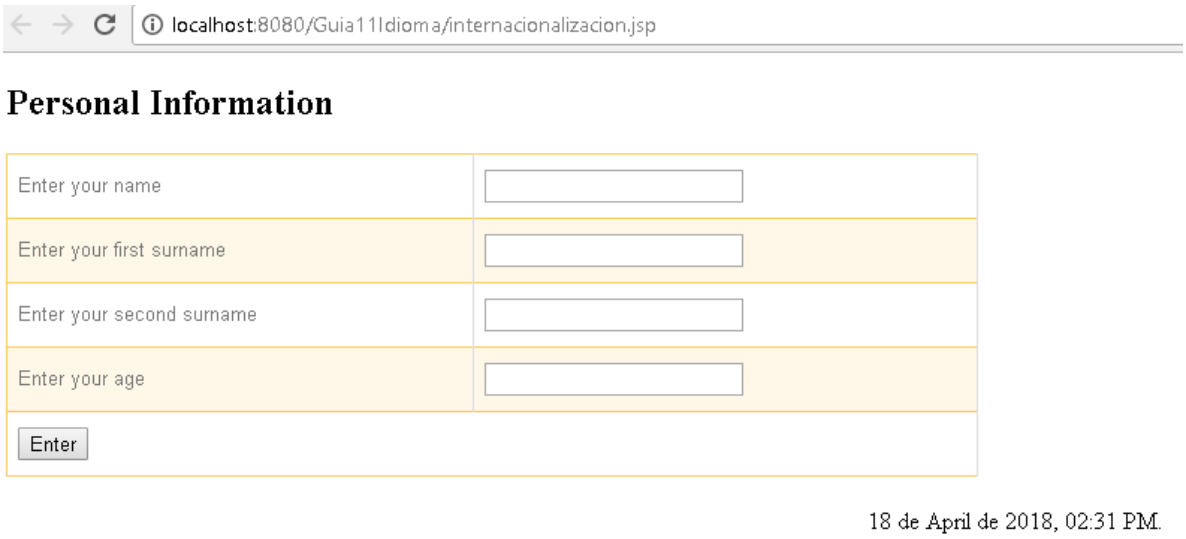
9. Al correr el archivo **internacionalizacion.jsp** dependiendo de la configuración del navegador el resultado será similar al siguiente.

Idioma: **español**



Idioma: inglés

Cambiar el idioma en el navegador a ingles actualice su página y verá los cambios:



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 (recuerde que ya debe haber creado las páginas de inserción, que corresponde a su actividad anterior. De no tenerlas, puede crearlas utilizando JSTL).

V. REFERENCIA BIBLIOGRÁFICA

- http://jimenez303.blogspot.com/
- http://wiki.netbeans.org/PoolConexionesGlassfishNetBeans