JSTL

JSTL (JAVASERVER PAGES STANDARD TAG LIBRARY)

- JSTL extiende los tags de los JSP's, agregando principalmente las siguientes librerías:

- **core**: Permite leer y manipular datos, así como iterar, agregar condiciones y otras funciones básicas

- **xml**: Permite la manipulación y transformación de documentos XML

- **sql**: Permite ejecutar consultas a una Base de Datos, así como crear conexiones a las mismas

- **fmt**: Permite dar formato a las cadenas, apoyado de conceptos como Internacionalización (Locale)

JSTL significa JavaServer Pages Standard Tag Library, esta es una librería de los JSPs que extiende la funcionalidad básica de los mismos, agregando principalmente las siguientes librerías: core, xml, sql y fmt.

La librería de **core** nos va a permitir leer y manipular datos así como iterar y agregar condiciones y otras funcionalidades básicas.

La librería de **xml** permite la manipulación y transformación de documentos xml.

La librería de **sql** tiene tags que nos van a permitir tanto ejecutar consultas a una base de datos así como crear conexiones a las mismas.

Finalmente la librería **fmt** nos va a permitir dar formato a las cadenas que estemos manipulando para convertirlas por ejemplo de cadena a número o fecha. Incluso esta librería nos permite apoyarnos de los conceptos como es *internacionalización* o en inglés *locale*.

En esta lección estudiaremos algunos de estos tags para tener mayor conocimiento de cómo podemos extender la funcionalidad básica de los JSP utilizando JSTL.

CONFIGURACIÓN DE JSTL

- Agregar las siguientes librerías al Classpath (No es necesario si el servidor Web como Glassfish ya cuenta con estas librerías):
  - standar.jar
  - jstl.jar
- Configuración de la directiva JSP:
  - JSP clásico:

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

  - Documento JSP (jspx)

```
<html
        xmlns:c   = "http://java.sun.com/jsp/jstl/core"
        xmlns:jsp = "http://java.sun.com/JSP/Page"   >
```

Para poder utilizar las librerías de JSTL en nuestra aplicación web en caso que usemos un servidor como Glassfish ya cuenta con las librerías JSTL, si por alguna razón no estuvieran en el servidor debemos agregar el .jar de estándar.jar y jstl.jar al classpath, sin embargo con Glassfish este paso no es necesario.

Posteriormente para poder utilizar las librerías dentro de un JSP debemos agregar la siguiente directiva: **<%@ taglib uri="http://java.sun.com/jstl/core"prefix="c"%>**

Como podemos observar estamos utilizando la sintaxis de las directivas en los JSPs según vimos anteriormente. En este caso el atributo **uri="http://java.sun.com/jstl/core"** es un uri que se resuelve de manera local en nuestro servidor web al agregar los jars (estándar.jar, jstl.jar), por lo que al solicitar desde un JSP un taglib, lo que hace el web server es buscar dentro de estos jars cualquier archivo con terminación .tld (tag library definition), por lo que esta uri no es algo que se tenga que buscar o resolver en internet, sino que simplemente se va a buscar un uri idéntica dentro de los jars que tenemos en nuestra aplicación web.

Si utilizamos archivos jspx, es decir un documento JSP, lo que tenemos que hacer es lo siguiente dentro de nuestro elemento root, por ejemplo en este caso dentro del tag de html lo que hacemos es especificar el namespace de nuestro tag library con elprefijo core y posteriormente el uri que es exactamente el mismo que especificamos anteriormente **<%@ taglib uri="http://java.sun.com/jstl/core"prefix="c"%>** y de igual manera el prefijo es el mismo que estamos utilizando para nuestro namespace. Hay que recordar que nuestro namespace únicamente es un prefijo de igual manera como lo estamos especificando en la notación clasica **"prefix="c".**

Existen más detalles respecto a la configuración de JSTL pero de manera básica y para empezar a utilizar esta tecnología con esto es suficiente.

LIBRERÍA CORE DE JSTL

- Tags de Despliegue de información:

```
<c:out value="${persona.nombre}">
```

- Tags de Creación y Manipulación de variables:

```
<c:set var="nombre" value="Carlos" scope="page" />
```

- Tags de Elementos condicionales:

```
<c:if test="${ i > 0 } />   y   <c:choose> <c:when test="a"> …
```

- Tags de Iteración de elementos:

```
<c:forEach   var="persona"   items="${personas}>  …
```

Dentro de la librería de core de jstl tenemos varios tags que nos van a permitir desplegar información, veamos como podemos hacer esto:

Tenemos un prefijo llamado **c** para el tag de core, según especificamos en el JSP. Posteriormente especificamos el nombre del tag que vamos a utilizar, en este caso es como si utilizáramos una función pero debido a que estamos utilizando JSPs en lugar de una función se le conoce como un tag o etiqueta y posteriormente utilizamos ciertos atributos y valores para terminar de configurar estos tags, por ejemplo: <c:out value="${persona.nombre}">.

Hay que recordar que el objetivo de un JSP es únicamente manipular tags o etiquetas y no manipular código de Java (aunque pueda realizar esta tarea) por ello la combinación de Expression Language con JSTL es muy práctica y son de las mejores prácticas de las que vamos a estar hablando en los siguientes ejercicios.

Para desplegar el valor de un JavaBean, por ejemplo la clase Persona,y queremos accedemos a la propiedad de *nombre* simplemente basta con utilizar el tag de **out** y especificamos el valor que queremos desplegar a nuestro cliente, ej. **"${persona.nombre}"**. Podríamos poner aquí una variable estática o una variable en código duro, pero en este caso podemos observar que al combinar JSTL con Expression Language podemos acceder al JavaBean que se encuentran en algún alcance que puede ser page, request, session o application según hemos comentado anteriormente.

También tenemos etiquetas para crear y manipula variables, debido a que el Expression Language tiene esa limitante respecto a modificar valores de los JavaBeans o crear JavaBeans. JSTL nos va a permitir crear ese tipo de variables con el tag de **set.** Podemos declarar una nueva variable y especificar el valor de dicha variable con**<c:set var="nombre" value...** y una vez que hemos creado la variable opcionalmente podemos declarar el alcance de esta variable, en este caso estamos especificando el alcance de manera explícita **scope="page"** pero si omitiéramos este atributo ~~scope="page" />~~ sucedería que esta variable **nombre** se agrega de manera automática al alcance de page. Entonces con este tag vamos a cubrir la limitante de expression Language para crear variables y también por ello es que vamos a poder omitir el uso de scriptles, utilizando código mucho más fácil de mantener y de entender en un JSP.

Posteriormente también tenemos tags para manejar elementos condicionales, por ejemplo el tag de **if** si es que necesitamos manejar algún código condicional, ej. **<c:if test="${ i > 0 }>** o también el tag de **choose,** este es un tag muy similar al switch de java, y los casos del switch se pueden poner con **<c:when test="a">**.

También tenemos etiquetas de iteración **<c:forEach var="persona" items="{personas}>**... para poder procesar los elementos de una lista. Cada uno de estos elementos se van a ir iterando y almacenando en el objeto **persona.** Esto va a ser un nuevo complemento respecto al Expression Language debido a que en un Expression Language no podemos iterar los elementos pero si podemos ir accediendo a cada uno de ellos de manera individual, entonces combinaremos JSTL con Expression Language para poder acceder a los elementos de una colección.

LIBRERÍA CORE DE JSTL

- Tags de Importación de recursos web:

```
<c:import  url = “recursoInternoExterno” >
```

- Tags de redireccionamiento:

```
<c:redirect  url =“${ nuevaUrl } />
```

- Tags de manejo de parámetros:

```
<c:import url = "c-import-param.jsp“ >
        <c:param name = “nombre" value = "${ param.select }“ />
</c:import>
```

Uno de los tags bastante útiles respecto a JSTL es el tag de import <c:import url="recursoInternoExterno"> en este caso la diferencia con un include que hemos comentado anteriormente con los JSPs es que los includes únicamente están restringidos a elementos que están dentro de nuestra aplicación web, en cambio este tag de import nos va a permitir incluir recursos internos o externos a nuestra aplicación web. Por lo que podemos incluir formularios de otras aplicaciones web o incluso páginas completas de otras aplicaciones. Esto nos va a permitir manipular nuestra aplicación de tal manera que el usuario no tenga que cambiarse de aplicación para estar navegando en diferentes aplicaciones web.

También tenemos el tag de redirect <c:redirect url="${nuevaUrl}"/> este caso es algo similar al send redirect que hemos utilizado en los Servlets, pero en este caso es un tag que nos va a permitir hacer una manipulación más sencilla del redirect. Únicamente lo que vamos a especificar es la nueva URL, que puede estar también de igual manera dentro de nuestra aplicación o en una aplicación externa a la nuestra.

Y finalmente tenemos el tag de manejo de parámetros, debido a que podemos incluir recursos de manera dinámica podríamos necesitar de enviar cierta información a estos recursos externos y debido a que no podemos proporcionar parámetros de manera dinámica en este url, vamos a poder combinar el manejo del tag de **import** y enviar la información de los parámetros por medio del tag de **param,** ej. **<c:param name="nombre" value = "${param.select}"/>** especificando el nombre del parámetro y el valor del mismo, incluso podemos observar cómo podemos combinar de nueva cuenta Expression Language para poder acceder a otros parámetros que estemos recibiendo y posteriormente propagarlos a otro componente JSP.

Estos son simplemente alguno de los ejemplos de tags que podemos utilizar con JSTL. A continuación vamos a crear algunos ejemplos para poner en práctica el uso de JSTL combinándolos con nuestros JSPs.