Para poder usar la librería Core en nuestra pagina JSP, debemos incluir la siguiente línea (recuerdas cuando vimos la directiva TagLib?) :

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

Agrega, esa línea antes de la etiqueta de nuestro index.jsp y listo.

1.

<c:out> :
Dicha etiqueta nos permite mostrar valores en nuestra página.
Normalmente, para mostrar valores o mensajes en JSP, usamos por ejemplo: <%=”Texto_del_mensaje”%>, pero también puede hacerse con la etiqueta <c:out> asi:

```
<c:out value="Esto es un mensaje"/>
```

Pero veamos algo un poco más interesante, creamos una clase llamada Mensaje en el paquete beans:

```
package beans;
public class Mensaje {
    private String texto;
    public String getTexto() {
        return texto;
    }
    public void setTexto(String texto) {
        this.texto = texto;
    }
}
```

Ahora nuestro index.jsp:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<jsp:useBean class="beans.Mensaje" id="mensaje"/>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        El valor en la variable <b>texto</b> es: <br/>
        <b><c:out value="${mensaje.texto}" default="valor_nulo"/>&quot</b>
    </body>
</html>
```

Supongo que ya sabes lo que hace la etiqueta jsp:useBean. Asi que la línea importante es:
`<c:out value="${mensaje.texto}" default=" valor_nulo >"/>`
Simplemente muestra el valor que hay en la propiedad “texto” del objeto “mensaje”, pero si es null, se muestra el mensaje por defecto. Ejecutemos a ver qué pasa:

Modifiquemos el index.jsp, de esta manera (solo muestro las etiquetas body, lo demás no cambia):

```
<body>
        <jsp:setProperty name="mensaje" property="texto" value="Esto es un
mensaje"/>
        El valor en la variable <b>texto</b> es: <br/>
        <b>&quot<c:out value="${mensaje.texto}" default="El valor es
null"/>&quot</b>
</body>
```

2.

<c:set>
Nos permite crear una variable y establecer un valor para esta:

```
<body>
        <!-- Creamos la variable texto con un valor String -->
        <c:set var="texto" value="valor_de_la_variable"/>
        <!-- Mostramos el valor de la variable texto -->
        El valor de la variable <b>texto</b> es : <c:out value="${texto}"/>
</body>
```

3.

<c:remove>
Simplemente remueve una variable que hayamos definido:

```
<body>
        <p>
            <!-- Creamos la variable texto con un valor String -->
            <c:set var="texto" value="valor_de_la_variable"/>
            <!-- Mostramos el valor de la variable texto -->
            El valor de la variable <b>texto</b> es : <c:out
value="${texto}"/>
        </p>
        <p>
            <!-- removemos la variable texto -->
            <c:remove var="texto" scope="page"/>
            <!-- Mostramos nuevamente el valor -->
            El valor de la variable <b>texto</b> ahora es :
            <c:out value="${texto}" default="Es Nulo"/>
        </p>
</body>
```

4.

<c:**catch**>
Nos permite capturar excepciones.

```
<body>
        <c:catch var="excepcion">
            <%=3/0%>
        </c:catch>
        <c:if test="${excepcion != null }">
            Ocurrió una excepción : <c:out value="${excepcion}"/>
        </c:if>
    </body>
```

Obtenemos entonces:

5.

<c:if>
Es lo mismo que la instrucción if de Java SE, evalúa una condición:

```
<body>
        <c:set var="a" value="20"/>
```

```
        <c:if test="${a > 100}">
            La variable <b></b> es <b>mayor</b> que 100.
        </c:if>
        <c:if test="${a < 100 }">
            La variable <b></b> es <b>menor</b> que 100.
        </c:if>
    </body>
```

6.

`<c:choose>`, `<c:when>` y `<c:otherwise>`
Funciona parecido a la instrucción switch, es decir, nos permite establecer varias condiciones en un mismo bloque.

```
<body>
       <c:set var="a" value="50"/>
        <c:choose>
            <c:when test="${a == 1}">
                <b>a</b> es 1.
            </c:when>
            <c:when test="${a == 2 }">
                <b>a</b> es 2.
            </c:when>
            <c:otherwise>
                <b>a</b> tiene un valor diferente de 1 y de 2.
            </c:otherwise>
        </c:choose>
</body>
```

En el anterior código, la etiqueta `<c:when>` es para definir alguna condición, mientras que la etiqueta `<c:otherwise>` es para definir el caso por defecto, en el cual no se cumple ninguna de las condiciones.

7.

`<c:forEach>`
Nos permite hacer ciclos en una jsp.
Crea la clase Lista en el paquete beans.

```
import java.util.*;
public class Lista {
    private List<String> listaDeNombres;
    public Lista() {
        listaDeNombres = new ArrayList<String>();
        listaDeNombres.add("Juan");
        listaDeNombres.add("Maria");
        listaDeNombres.add("Alberto");
        listaDeNombres.add("Lucia");
    }
    public List<String> getListaDeNombres() {
        return listaDeNombres;
    }
}
```

Ahora vamos a iterar sobre el arreglo que contiene la clase Lista, para saber qué valores contiene:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<jsp:useBean id="lista" class="beans.Lista"/>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <p>Los elementos en la lista son: </p>
        <p>
            <c:forEach var="nombreActual" items="${lista.listaDeNombres}">
            <b><c:out value="${nombreActual}"/></b> <br/>
        </c:forEach>
        </p>
    </body>
</html>
```

En la etiqueta <c:forEach> el parámetro **var="nombreActual"** es simplemente un apuntador hacia el objeto actual de la iteración.

8.

<c:forTokens>
Nos permite iterar sobre un String y separarlo por tokens.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <c:set var="oracion" value="Estos,son,los,tokens,de,la,frase"/>
        Los tokens son los siguientes:
        <p>
            <c:forTokens
items="${oracion}" delims="," var="token" varStatus="i" >
                <b><c:out value="${i.count}"/>.</b>  <c:out
value="${token}"/> <br/>
            </c:forTokens>
        </p>
    </body>
</html>
```

El parámetro **delims=","** es simplemente para identificar cual es el carácter separador de los diferentes tokens de la cadena que vamos a separar y **varStatus="i"**, es simplemente una variable que guarda el estado actual de la iteración, por lo que podemos saber en qué iteración estamos haciendo uso de la instrucción **${i.count}** . Al ejecutar obtenemos:

9.

<c:**import**>
Recuerdas lo que hace la **directiva include** o la acción <jsp:include>?
Pues <c:**import**> hace básicamente lo mismo, solo con el hecho de que esta etiqueta

permite también incluir contenido **que no esté en la misma aplicación web** que estemos desarrollando. Creamos una nueva pagina JSP:

Cuyo contenido es:

```
<h2>Contenido importado</h2>
```

Y ahora nuestra index.jsp:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <c:import url="nuevaJSP.jsp"/>
        Contenido de <b>index.jsp</b> .....
    </body>
</html>
```

Obtenemos:

10.

<c:url>, <c:param> :
Nos permiten establecer URL's y parámetros en nuestra JSP respectivamente.
Nuestro index.jsp:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        Click en el enlace para enviar el parametro :
        <a href="
           <c:url value="nuevaJSP.jsp">
               <c:param name="nombre" value="Juan"/>
           </c:url>
           ">link</a>
    </body>
</html>
```

Como ves en el anterior codigo, definimos una url que apunta hacia **"nuevaJSP.jsp"**, pero además de eso, dicha URL, envía un parámetro llamado **"nombre"**. Ahora nuestra "nuevaJSP.jsp", es asi:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        El parametro enviado es : <br/>
        <b><c:out value="${param.nombre}"/></b>
```

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

Ejecutamos y obtenemos:

11.

`<c:redirect>`

Nos permite redirigir una petición.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <c:redirect url="nuevaJSP.jsp">
            <c:param name="nombre" value="Juan"/>
        </c:redirect>
    </body>
</html>
```