

Uso de formularios para captura de información

Uso de formularios para captura de información	1
¿Qué aprenderás?	2
Introducción	2
Formulario JSP a formulario JSP	3
Composición	3
Input Text	3
Input Text de password	3
CheckBox	4
Radio Button	4
Boton de envio de formulario	4
Estructura de un formulario	5
Comunicación de JSP a JSP	6
Objetos implícitos	10
El objeto request	11



¡Comencemos!

¿Qué aprenderás?

- Conocer el concepto de formulario.
- Utilizar controles html de formularios.
- Conocer los objetos implícitos.

Introducción

Como se ha estudiado en capítulos anteriores, una página JSP es capaz de manejar lógica de negocio ya sea mediante código nativo java entre scriptlet y también utilizando la librería JSTL. Estas características, están enfocadas a proveer al sistema de inteligencia para procesar datos y en base a ellos generar conocimiento, almacenar datos y en definitiva dar un valor al usuario.

El elemento principal que permite que un usuario alimente de datos a un sistema web son los formularios, los cuales están diseñados para capturar la información del usuario y enviarlos al servidor para su posterior procesamiento. En este capítulo trabajaremos con los formularios y se estudiarán los mecanismos que ofrecen para la comunicación entre capas.

Formulario JSP a formulario JSP

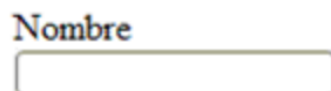
En una arquitectura empresarial, es común la comunicación entre entidades del mismo tipo, ya sea para generar un flujo de trabajo en donde la información solo deba pasar de página a página o definitivamente procesar la información hasta almacenarla en una base de datos. En el caso de la tecnología JSP es posible transferir información desde dos páginas java server page. Veremos a base de ejemplos prácticos los mecanismos para empezar la comunicación.

Composición

Un formulario está compuesto por componentes HTML que tienen como misión capturar información ingresada por el usuario. Algunos de estos controles son:

Input Text

Se trata del elemento más utilizado en los formularios. En el caso más sencillo, se muestra un cuadro de texto vacío en el que el usuario puede escribir cualquier texto:

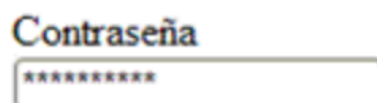


Nombre

Imagen 1. Ejemplo de etiqueta input (type=text).
Fuente: Desafío Latam

Input Text de password

La única diferencia entre este control y el cuadro de texto normal es que el texto que el usuario escribe en un cuadro de contraseña no se ve en la pantalla. En su lugar, los navegadores ocultan el texto utilizando asteriscos o círculos, por lo que es ideal para escribir contraseñas y otros datos sensibles.



Contraseña

Imagen 2. Ejemplo de etiqueta input (type=password).
Fuente: Desafío Latam

CheckBox

Los checkbox o "casillas de verificación" son controles de formulario que permiten al usuario seleccionar y deseleccionar opciones individualmente. Aunque en ocasiones se muestran varios checkbox juntos, cada uno de ellos es completamente independiente del resto. Por este motivo, se utilizan cuando el usuario puede activar y desactivar varias opciones relacionadas pero no excluyentes.

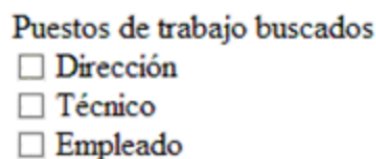


Imagen 3. Ejemplo de etiqueta input (type=checkbox).
Fuente: Desafío Latam

Radio Button

Los controles de tipo radiobutton son similares a los controles de tipo checkbox, pero presentan una diferencia muy importante: son mutuamente excluyentes. Los radiobutton se utilizan cuando el usuario solamente puede escoger una opción entre las distintas opciones relacionadas que se le presentan. Cada vez que se selecciona una opción, automáticamente se deselecciona la otra opción que estaba seleccionaba.

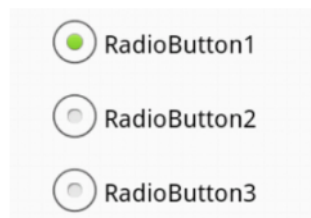


Imagen 4. Ejemplo de etiqueta input (type=radio).
Fuente: Desafío Latam

Boton de envio de formulario

La mayoría de formularios dispone de un botón para enviar al servidor los datos introducidos por el usuario:

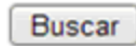


Imagen 5. Ejemplo de etiqueta input (type=submit).
Fuente: Desafío Latam

El valor del atributo type para este control de formulario es *submit*. El navegador se encarga de enviar automáticamente los datos cuando el usuario pincha sobre este tipo de botón. El valor del atributo value es el texto que muestra el botón. Si no se establece el atributo value, el navegador muestra el texto predefinido “Enviar consulta”. Estos controles permiten la interacción de un humano con el sistema. Para más información es posible investigar controles html en google.

Para poder comunicar un JSP con otro hace falta un formulario que capture y envíe la información.

Estructura de un formulario

La etiqueta de inicio de un formulario se presenta a continuación:

```
<form action="proceso.jsp" method="POST"></form>
```

La etiqueta comienza con la palabra form la cual declara el inicio de un formulario en lenguaje HTML (No olvidar que una página JSP es también una página con html). Todo lo que esté dentro de estas etiquetas se considera un formulario y será procesado como tal.

La segunda palabra clave que se ve en el código es el atributo action, que en palabras simples, dirige los datos que el usuario ingrese en el formulario al archivo que está llamando entre las comillas dobles, en este caso todo lo que el usuario coloque en los formularios se irá a un archivo (que aún no existe) que se llama proceso.jsp.

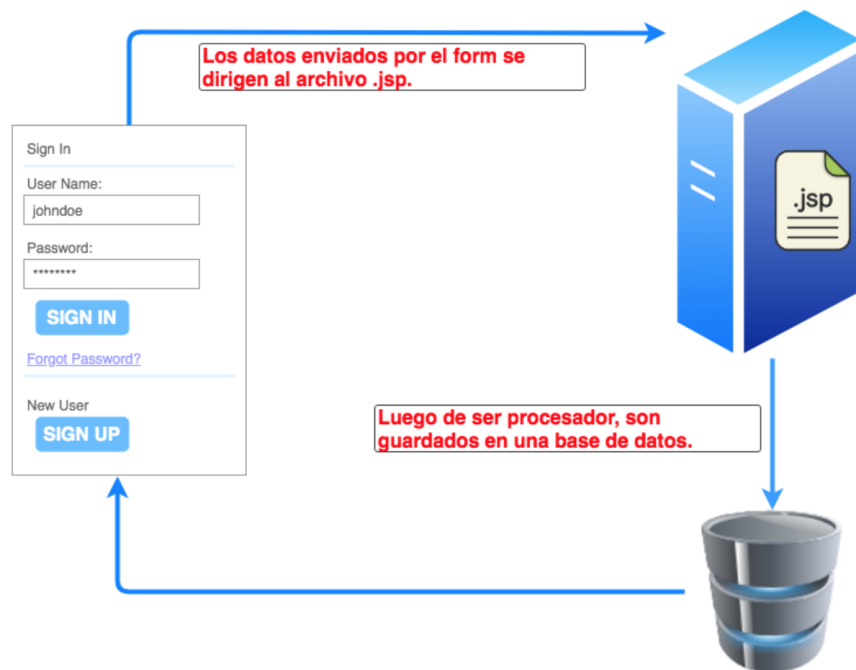


Imagen 6. Flujo entre formulario y servidor.
Fuente: Desafío Latam

Este simple, pero importante flujo convierte una página web estática en un sistema de información.

Comunicación de JSP a JSP

Continuando con el ejemplo de comunicación entre JSP y JSP se crearán dos archivos Java Server Page, uno encargado de recopilar los datos que ingrese el usuario y otro que los capture y despliegue en la página. Crear un proyecto Dynamic Web Project de nombre *Patrones_Diseño_Sesion1_ejemplo012* y dos archivos jsp, *contacto.jsp* y *validacion.jsp*.

El primer archivo jsp es un humilde formulario:

Usuario

Comentario

Imagen 7. Formulario Contacto.jsp.
Fuente: Desafío Latam

El código se presenta a continuación:

Código formulario Contacto.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
</head>
<body>
    <div class="container" style="border:1px solid black;height:
600px;">
        <form action="validacion.jsp" method="get">
            <div>
                <label>Usuario</label>
                <input type="text" id="user" name="user">
            </div>

            <div>
                <label>Comentario</label>
                <textarea id="comentario"
name="comentario"></textarea>
            </div>
            <div id="boton">
                <button type="submit" class="btn
btn-primary">Enviar</button>
            </div>
        </form>
    </div>
</body>
</html>
```

Como se puede apreciar en el código, se puede ver el inicio de la etiqueta , el cual tiene como referencia un archivo *validacion.jsp* y el método post. En apartados posteriores estudiaremos los métodos de envío, por el momento solo tenemos en cuenta que se declara para enviar un formulario al servidor mediante este protocolo.

El resto de componentes son solamente controles HTML, pero tienen una característica especial la cual es su atributo name. Para enviar valores desde un formulario a algún lenguaje de programación por el lado de servidor (en nuestro caso es java con jsp, pero podría ser cualquier otro) es muy importante recordar que los elementos html que reciben los datos del usuario tienen que obligatoriamente tener un atributo name de identificación, ya que mediante con este elemento es posible que el archivo que procesa los valores sepa

en donde estan ubicados. Por ejemplo en el código anterior el elemento que recibe el usuario tiene un *name* de nombre user y cuando el archivo jsp reciba la petición, sabrá con claridad que el nombre de usuario que tiene que evaluar esta en el input text que tiene por id el nombre 'usuario'.

Para graficar este importante concepto se continuará con el ejercicio. Creamos un segundo archivo JSP que en términos prácticos, es el mismo archivo anterior, pero con la gran diferencia que solo mostrará los valores que ingresamos en el primer formulario. El código es el siguiente:

Código formulario contacto.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
</head>
<body>
    <div class="container" style="border:1px solid black;height:
600px;">
        <h2>Recepcion de datos desde Contacto.jsp</h2>
        <%
            String nombre = request.getParameter("user");
            String comentario =
request.getParameter("comentario");
        %>
        <form action="Contacto.jsp" method="post">
            <div>
                <label>Usuario</label>
                <input type="text" id="user"
value="<%out.println(nombre);%>">
            </div>
            <div>
                <label>Comentario</label>
                <textarea
id="comentario"><%out.println(comentario);%></textarea>
            </div>
            <div id="boton">
                <button type="submit" class="btn
btn-secondary">Volver</button>
            </div>
        </form>
    </div>
</body>
```


Tal como se vio en el capítulo anterior, se utilizan directivas scriptlet en donde se incrusta código java, declarando dos variables de tipo String. Cada una de estas variables se encargará de capturar los valores que se ingresarán en el primer formulario para luego mostrarla en los *input text* y en el *text area*.

Las variables declaradas contienen el elemento capturado en el request mediante el nombre del atributo name (el name que identifica al componente de la página jsp). Este ejemplo ilustra de manera básica como es el proceso para enviar variables desde archivos jsp y otro jsp.

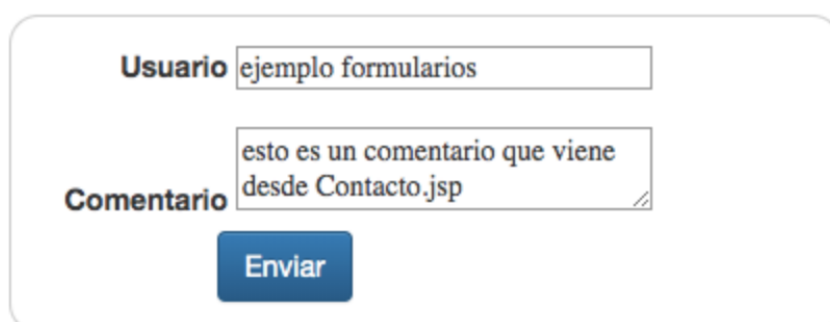
A screenshot of a web form titled 'Formulario Contacto.jsp'. It contains two input fields: 'Usuario' with the value 'ejemplo formularios' and 'Comentario' with the value 'esto es un comentario que viene desde Contacto.jsp'. Below the fields is a blue button labeled 'Enviar'.

Imagen 8. Formulario Contacto.jsp con datos ingresados.
Fuente: Desafío Latam

Al presionar en el botón enviar, los valores son enviados al archivo.jsp declarado en el action del formulario. Al momento de recibir el request el segundo jsp los captura y los muestra en el formulario.

Recepcion de datos desde Contacto.jsp

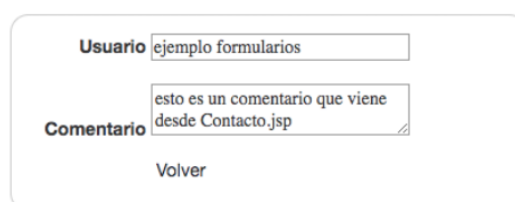
A screenshot of a web form titled 'Formulario validación.jsp'. It contains two input fields: 'Usuario' with the value 'ejemplo formularios' and 'Comentario' with the value 'esto es un comentario que viene desde Contacto.jsp'. Below the fields is a button labeled 'Volver'.

Imagen 9. Formulario validación.jsp con datos rescatados desde el request.
Fuente: Desafío Latam

El objeto request es el utilizado para obtener los valores enviados desde el formulario por lo tanto hay que entender que es y cómo se utiliza. A continuación se estudiará más a fondo.

Objetos implícitos

Como se vio en unidades anteriores, los servlets pueden acceder a los valores enviados mediante los request al igual que las páginas JSP, las cuales también utilizan este mismo mecanismo.

Estas variables son llamadas implícitas porque pueden ser utilizadas en cualquier lugar de las páginas JSP con tan solo llamarlas, y además de la variable request existen un conjunto de ellas las cuales son:

- **Request:** Método de petición para indicar la acción que se desea realizar para un recurso determinado.
- **Response:** Respuesta generada por el servidor.
- **out:** Representa el flujo de salida del cuerpo de la respuesta HTTP.
- **session:** Permite mantener una sesión para cada uno de los usuarios conectados a la aplicación Web.
- **config:** Contiene información relativa a la configuración del servlet generado.
- **exception:** Excepción que se ha producido en una página JSP.

En este caso nos enfocaremos en los request, ya que es esencial para el manejo de formularios y la transferencia de valores entre páginas jsp.

El objeto request

Las páginas JSP son componentes web que responden a los request mediante el protocolo HTTP. El objeto implícito request es quien representa este mensaje que se envía desde el navegador y puede contener la información de los valores enviados desde el cliente además de información relevante de las cabeceras del protocolo.

Cuando un navegador envía información al servidor lo hace a través del objeto request el cual viaja por la red mediante la url del sitio. Existen dos formas de transferencia de request mediante la url:

- **Url encoded parameters:** En esta modalidad los valores que deben viajar desde el navegador al servidor van incrustados en la misma url en formato de parámetros explícitamente visibles. Los parámetros compuestos por su clave-valor (nombre de la variable y su contenido) van después de un símbolo de interrogación y separados mediante el símbolo &.

http://www.miservidor.com/carpetaproyecto/proyecto?var1=nombre1&var2=nombre2

- **Form encoded parameters:** Estos parámetros son enviados como resultado de la ejecución del envío de un formulario mediante un submit. Se cuenta con el mismo formato de url de *url encoded parameters*, pero con la diferencia que los valores van envueltos en un body y no incrustados en la misma url.

```
<%  
String nombre = request.getParameter("user");  
String comentario = request.getParameter("comentario");  
%>
```

Imagen 10. Utilización del objeto implícito request.

En el ejemplo del formulario anterior, si miramos la imagen 10 en la línea 92, vemos que en el formulario está declarado el método POST, lo que significa que el envío de parámetros es mediante el método Form encoded parameters.

Si ejecutamos el formulario con algunos datos, vemos que la url de envío no muestra los valores que se están enviando explícitamente.

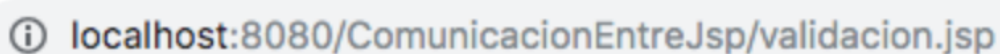
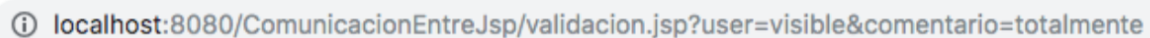


Imagen 11. No se muestran los valores enviados.
Fuente: Desafío Latam

Si cambiamos el método de envío a GET, veremos como los parámetros son expuestos en la url de esta forma:



localhost:8080/ComunicacionEntreJsp/validacion.jsp?user=visible&comentario=totalmente

Imagen 12. Se muestran los valores enviados.

Fuente: Desafío Latam

Como se aprecia, los parámetros que coloques en el formulario son visibles en la URL. Esa es la diferencia entre el envío de parámetros con el método GET y el método POST.

El método `request.getParameter("identificador")` devuelve el valor contenido en la variable designada por el identificador que está entre paréntesis. En caso de querer capturar una lista de valores, como por ejemplo de una lista de *checkbox*, se utiliza el método `request.getParameterValues("identificador")` que se encarga de recuperar todos los valores seleccionados.

Este mecanismo es primordial para la comunicación entre capa vista y controlador, por lo cual en los siguientes capítulos se estudiará esta arquitectura para generar un sistema web con todo lo aprendido hasta ahora.