Agosto | 09

# Un primer JSP

Guía rápida

# Un Primer JSP

## Guía rápida

Este documento te guiará por los primeros pasos para desarrollar **aplicaciones web** usando NetBeans IDE. Te mostrará como crear una aplicación web simple, su despliegue en el servidor y su presentación en un navegador. La aplicación utiliza una página JavaServer Pages (JSP) y su correpondiente clase, uno de los objetivos pero no el principal, es introducirte en el uso de NetBeans como un IDE de desarrollo muy completo, entendible y fácil de usar.
Para completar esta guía, necesitaras el siguiente software y recursos.

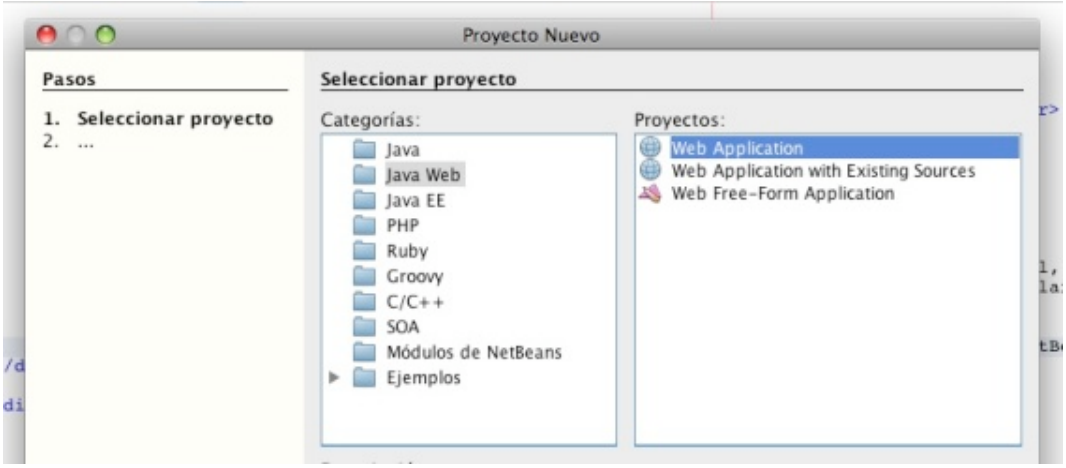
| Software | Versión |
|---|---|
| **NetBeans IDE** | Instalación Web o Java EE versión 6.0 en adelante |
| **Java Developer Kit (JDK)** | Versión 5 en adelante |
| **Servidor de Aplicaciones GlassFish**<br>ó<br>**Contenedor de Servlets Tomcat** | V2<br><br>Versión 6.x |

### Nota

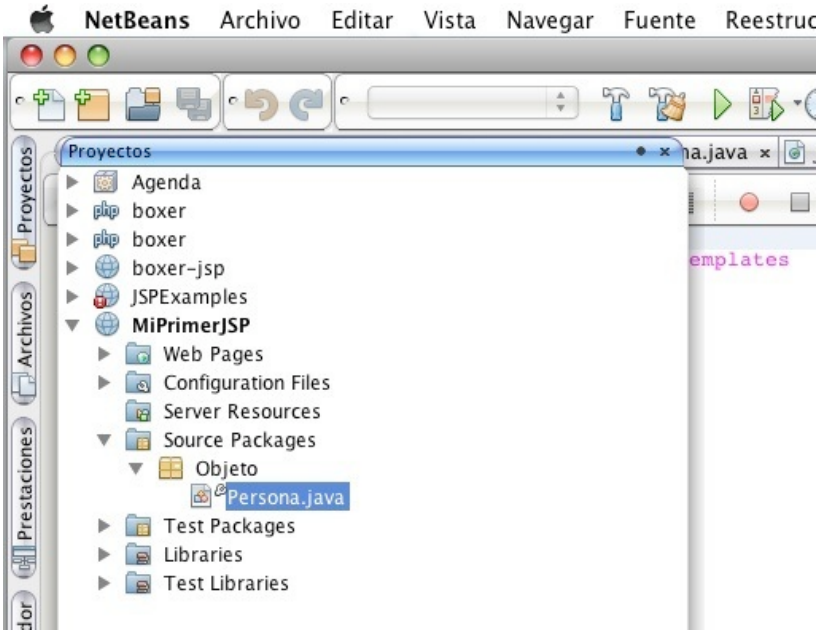
- Las instalaciones Web y Java EE te permitirán que instales, de manera opcional, el servidor de aplicaciones GlassFish V2 y el contenedor de servlets Apache Tomcat 6.0.x. Debes instalar una de las aplicaciones anteriores para realizar el tutorial.

## Preparando el proyecto

Hay que comenzar un nuevo proyecto Tomcat en Eclipse, o **Java Web/Web Application** en NetBeans. Lo nombramos **MiPrimerJSP**.




En el siguiente paso vamos a crear una clase Java común y corriente muy sencilla para probar. Primero creamos un **package** llamado **Objetos**, hacemos clic con el botón derecho del ratón sobre Web-inf en Eclipse o en panel de la izquierda, **Proyectos -> MiprimerJSP -> Source Packages** en NetBeans y elegir **New -> Package** ( Java Package en Netbeans ). Ahora hacemos clic con el boton derecho del ratón en dicho package, elegir **New -> Class** ( Java Class en NetBeans ) y la nombramos **Persona**.




Escribimos la clase:

```
package Objetos;

public class Persona {

    private String Nombre;
    private String Ciudad;
    private int id;

    public String getCiudad(){

        return Ciudad;
    }

    public void setCiudad(String Ciudad){
        this.Ciudad=Ciudad;
    }

    public String getNombre(){

        return Nombre;
    }

    public void setNombre(String Nombre){

        this.Nombre=Nombre;
    }

    public int getId(){

        return id;
    }

    public void setId(int id){

        this.id=id;
    }
}
```

NetBeans te crea por defecto el **index.jsp**. En Eclipse hay que crearlo a mano, en la carpeta Web-inf, pueden hacerlo a desde la carpeta del S.O. o bien **New -> File** sobre dicha carpeta y nombrarlo **index.jsp**.

## Escribiendo la página JSP

Lo primero que hay que indicarle a la página ( index.jsp ) son una serie de parámetros o directivas, como por ejemplo qué clases importar:

```
<%@ page contentType="text/html; charset=utf-8" import="Objetos.Persona"
    errorPage=""%>
```

`contentType:` especifica el tipo de salida, en este caso text/html.

`import:` importa las clases o paquetes que vamos a utilizar en la página. Se puede separar por coma o usando varias veces import. En este caso `Objetos.Persona` ( también pueden ser clases del entorno claro, como `java.sql` ).

`erroPage:` muestra simplemente la página de error en caso de que ocurra uno y no sea tomado por las excepsiones del código.
Hay otros parámetros más pero por ahora no viene al caso.

Hay varias formas de manipular la información de las clases en el código HTML, y yendo más lejos podemos utilizar frameworks. Ahora vamos a ver dos formas básicas, código java enbebido y los javaBeans. Se utilizan tags especiales para separlo del HTML común.

Para código java enbebido se utiliza:

<%= expresión %>: para mostrar datos, es decir expresiones evaluadas para la salida (por ejemplo: `persona.getNombre()` en una tabla HTML).

<% código %>: algoritmos y cálculos comunes, tomar una variable, manipular clases (código java común y corriente). No se puede mostrar algo en el navegador, para eso de utiliza el tag anterior.

<%! código %>: se insertan en el cuerpo de la clase del servlet, fuera de cualquier mótodo existente.

Por ahora con los los primeros tags nos alcanza.
Con esto ya se puede practicar algo, vamos a ver un ejemplo, tan fácil que casi no hacen falta comentarios:

```
<%@page contentType="text/html" pageEncoding="UTF-8"
import="Objetos.Persona" errorPage=""%>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Mi primer JSP</title>
    </head>
    <body>
        <%
        Persona persona= new Persona();
        persona.setNombre("Boxer");
        persona.setCiudad("Lomas Altas");
        persona.setId(1);
        %>

        <h1>Datos de la Persona</h1>
        <p>Datos de la persona: </p>
```

```
        <p>Nombre: <%= persona.getNombre() %></p>
        <p>Ciudad <%= persona.getCiudad() %></p>

        <p>Num. de Identificación: <%= persona.getId() %></p>
    </body>
</html>
```

Abajo del cuerpo se utilizó el tag para setearle los datos al objeto, y en el párrafo se muestran esos datos con la etiqueta . Para tomar un parámetro que viene por `POST` o `GET`, pueden tomarlo con `request.getParameter()`.

Por ejemplo para llevar un parámetro "`nombre`" a una variable, dentro del tag que corresponda ponemos: `String variable=request.getParameter("nombre")`. Recuerden que llega como `string`, si es un número deben convertirlo a número con código Java.

Veamos el mismo caso utilizando **javaBeans**. Básicamente un Bean hace referencia a una clase fácilmente reutilizable y "sencilla" por así decirlo. La clase **Persona** que creamos sería una especie de Bean. Para que funcione debe tener los setters y getters que ya los hemos puesto.

```
<%@page contentType="text/html; charset=utf-8" import="Objetos.Persona"%>

<html>
  <head>
    <title>Mi primer JSP</title>
  </head>

  <body>

    <jsp:useBean id="persona" scope="page" class="Objetos.Persona" />
    <jsp:setProperty name="persona" property="nombre" value="Boxer" />
    <jsp:setProperty name="persona" property="ciudad" value="Lomas Altas" />
    <jsp:setProperty name="persona" property="id" value="1" />

    <h1>Datos de la persona:</h1>
    <p>Nombre: <jsp:getProperty name="persona" property="nombre" /></p>
    <p>Ciudad: <jsp:getProperty name="persona" property="ciudad" /></p>
    <p>Numero de identificacion: <jsp:getProperty name="persona"
       property="id" /></p>

  </body>
</html>
```

Veamos una pequeña explicación

`<jsp:useBean id="algo" scope="page" class="package.class" />`: nos sirve para declarar un Bean. Hay que indicarle cuál paquete y clase a utilizar. Observa que le damos el id "persona" en el ejemplo, por lo que a partir de ahora usamos ese id. El scope no viene al caso por ahora.

`<jsp:setProperty name="algo" property="nombre" value="Boxer" />`: sirve para setearle un valor al Bean creado, usando el id que le hemos dado al crearlo.
`<jsp:getProperty name="algo" property="nombre" />`: para mostrar los valores del Bean.

## Probando nuestro JSP

Para probar la página, tiene que estar iniciado **Tomcat** en los respectivos IDE. En NetBeans hacemos clic con el botón derecho del ratón sobre el proyecto (el panel de la izquierda) y elegir **"Run"**; nos llevará automáticamente al navegador por defecto. En Eclipse pueden entrar desde el navegador siguiendo este formato (por lo general): `http://localhost:8080/MiPrimerJSP/index.jsp` o la dirección donde hayan puesto la página.

## Recomendaciones

Con lo que hemos visto ya podras hacer algunos ejemplos mas completos, y más si sabes usar Bases de Datos. Un ejercicio que puedes probar por ejemplo es el siguiente: crear otra página, pongamosle "**datos.jsp**" y con "**index.jsp**" pasarle los parámetros por `GET` desde un link común (o `POST` desde un formulario), tomarlos y hacer cálculos o mostrarlos.

La idea de este artículo es solamente comenzar a jugar con Java y páginas webs en la parte básica. Para hacer un sitio profesionalmente se utilizan otros mecanismos y hay miles de alternativas.

**Ultimo tip**: si quieres usar imágenes, Flash, CSS o JavaScript en la página no hay ningún problema. Guarda el material en alguna carpeta e indica la ruta en el HTML como siempre, ya sea dinámicamente o estáticamente. Si yo tengo una carpeta "**paginas**" con "**index.jsp**" y dentro de la misma tengo otra carpeta llamda "**css**" con un archivo de estilos, la ruta del css sería `css/estilos.css`

Con paciencia, buscando, preguntando, probando se puede ir entendiendo la onda.

**Suerte**.