JavaBeans en JSP

- JavaBeans surge de una necesidad de la Ingeniería del Software: contar con componentes reutilizables e independientes de la plataforma. El programador que hace el JB no hace nada misterioso, simplemente define una clase, tratando de encapsular (ocultar) su implementación y mostrando al exterior (el programador que usa el bean) solamente los métodos y propiedades que son públicos. Sólo se muestra aquello que forma parte del servicio que el bean ofrece al exterior.
- El programador que usa el bean sólo debe preocuparse de lo QUE hace el bean (el servicio que ofrece), no tiene que enfrentarse al trabajo realizado por el diseñador del bean, es decir, le queda oculta la implementación (el COMO el bean ofrece el servicio).
- Los JavaBeans son componentes reusables se puedan utilizar en diversos contextos: desde JavaBeans gráficos de capa cliente (controles de interfaz) hasta JavaBeans de capa web (normalmente representan entidades o reglas de negocio).

JavaBean de ejemplo

Aunque no ha sido necesario en este ejemplo, los JavaBean **deben ser serializables** (o heredar de una clase serializables), si es que queremos usar la serialización de objetos.

```
package objeto;

public class asignatura
{
   public String  Nombre;
   public double nota;

   public void asignatura()
   {

   }

   public String getNombre()
   {
      return Nombre;
   }

   public void setNombre(String Nombre)
  {
      this.Nombre = Nombre;
   }

  public double getNota() {
      return nota;
   }

   public void setNota(double nota)
  {
      this.nota = nota;
```

```
    }
    public String getresultado ()
    {
        if (nota <3)
           return "Reprueba" ;

        else
          return  "Aprueba";
    }

}
```

Para instanciar las clases del API de Java no hay más que utilizar el viejo operador new.

Pero las clases que representan entidades y reglas de negocio se llaman JavaBeans y exigen unas etiquetas específicas:

<**jsp:usebean id**="id_del_objeto" **scope**="page | request | session | Application" **class**="paquete...subpaquete.clase.class" **beanName**="nombre_del_bean"/>

El id es un nombre identificativo, seleccionado por el programador. Además debemos declarar su alcance o ámbito (scope). Un alcance "request" implica que el bean es accesible hasta otra JSP que haya sido invocada por medio de jsp:forwar o jsp:include. El beanName es opcional, sigue la lógica Java de paquete.subpaquete y se utiliza si se usa el método instantiate() de java.beans.Beans.

En nuestro ejemplo:

<jsp:useBean id="nota" scope="page" class="objeto.asignatura">

De esta sencilla línea se puede deducir que al menos debemos definir en el JavaBean el constructor vacío, que es el que utiliza el motor JSP en la creación del objeto.

Existe una segunda sintaxis:

<**jsp:usebean id**="id_del_objeto" **scope**="page | request | session | application" **class**="paquete...subpaquete.clase.class" **beanName**="nombre_del_bean">

Instanciacion del Bean

</jsp:useBean>

En este caso se ejecutan las instrucciones **si el bean es instanciado**. Dichas instrucciones pueden ser cualquier contenido JSP, aunque normalmente consiste en Scriptlets y acciones setProperty.

Los métodos setXXX() y getXXX() son puramente convencionales, podríamos haberlos llamado defXXX() y obtXXX(), sin embargo es conveniente **usar los métodos setXXX() y getXXX()** para propiciar la interoperabilidad e integración con software de otros fabricante. Además esto facilita el uso de acciones setProperty y getProperty.

Uso de setProperty y getProperty

jsp:setProperty se usa en conjunción con jsp:useBean para definir valores de propiedades. Las etiquetas **jsp:setProperty y jsp:getProperty** nos evitan los scriptlets y se encargan de invocar a los métodos setXXX() y getXXX() del JavaBean.

```
<jsp:setProperty
     name="id_del_objeto"
     property="nombre_propiedad"|"*"
     param="nombre_parametro_de_request"|
     value="valor" />
```

Esta acción puede aplicarse a **una** propiedad:

```
<jsp:setProperty name="nota" property= "nota"></jsp:setProperty>
```

O a **todas aquellas propiedades cuyo nombre coincide con parámetros de la petición** (request), es decir, selecciona los parámetros que coinciden en nombre con las propiedades y copia los valores en sus correspondientes propiedades:

```
<jsp:setProperty name="nota" property="*" />
```

Si queremos solamente extraer el valor de **un parámetro** (en el siguiente ejemplo es 'identificacion') y copiarlo a **un** atributo ('nombre'):

```
<jsp:setProperty
     name="nota"
     property="nombre"
     param="nombre" />
```

Con **value** podemos especificar un valor para la propiedad, pero esta etiqueta es **incompatible** con **param**.

En el siguiente ejemplo se obtiene el nombre de un formulario mediante request.getParameter() y se almacena en el Bean:

```
<%! String tuNombre; %>
<% tuNombre = request.getParameter( "nombre" ); %>
<jsp:setProperty name="nota" property="nombre"
value="<%=tuNombre%>" />
```

Como ya hemos visto, esto se puede hacer de forma más breve:

```
<jsp:setProperty name="nota" property="nombre"
param="nombre" />
```

Incluso se puede hacer durante la instanciación:

```
<jsp:useBean id="nota"
             scope="page"
             class="objeto.asignatura">

   <jsp:setProperty
        name="nota"
       property="*" />

</jsp:useBean>
```

El ejemplo completo

```
<%@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">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>JSP Page</title>
   </head>
   <body>
      <h1>Trabajando con los beans  !</h1>
<jsp:useBean id="nota"  scope="page" class="objeto.asignatura">
   <jsp:setProperty name="nota" property= "nota"></jsp:setProperty>
     <%
      nota.setNota(4);

      %>
    <jsp:getProperty name="nota" property= "nota"></jsp:getProperty>
     <p>Hemos creado una instancia del JavaBean. La clase se llama
       <%=nota.getClass().getName()%>.<p></p>

          El nombre de la instancia es <%=nota.getClass().getSuperclass()%>
      <p ></p>
      y la Nota es <%= nota.getNota()%> <br>
     <jsp:getProperty name="nota" property= "resultado"></jsp:getProperty>
     El resultado es <%nota.getresultado(); %><p></p>
</jsp:useBean>
   </body>
</html>
```