

MP0486
Acceso a datos
UF5. Programación de
componentes de acceso a datos

5.6. Los JavaBeans

Índice

☰	Objetivos	3
☰	Concepto	4
☰	Proyecto web con un JavaBean	7
☰	JSP y los JavaBeans	9
☰	Resumen	13

Objetivos

En esta lección perseguimos los siguientes objetivos:

1

Recordar el concepto de *JavaBeans*.

2

Utilizar *JavaBeans* desde páginas JSP.

¡Ánimo y adelante!

Concepto

En el siguiente apartado vamos a recordar el concepto de *JavaBeans*, que ya vimos en la Unidad 3, con el objeto de aplicarlo después a las páginas JSP.

Para comenzar, recordemos que un *JavaBean* es una **clase** que debe cumplir una serie de condiciones.

1

Debe implementar la **interfaz Serializable**, que otorga a sus objetos la capacidad de persistencia.

2

Debe tener un **constructor vacío** (que no reciba argumentos), aunque luego puede tener otros constructores. De esta forma, permite crear objetos estándar.

3

Todas las propiedades del objeto serán **privadas y accesibles mediante métodos get/set** que serán públicos.

4

Para los métodos *get/set*, hay que **seguir cuidadosamente la nomenclatura estándar**. Para una propiedad privada llamada *precio*, los métodos *get/set* serán *getPrecio* y *setPrecio*, es decir, con las palabras *get* y *set* en minúscula y el nombre de la propiedad con la primera letra mayúscula.

El concepto *JavaBean* fue creado por Sun Microsystems, aunque esta compañía fue adquirida por Oracle en 2010. Los *JavaBean* nacieron con el objetivo de ser utilizados como componentes software reutilizables.

Veamos un ejemplo de clase que cumple con las especificaciones de los *JavaBeans*:

```
import java.io.Serializable;
import java.time.LocalDateTime;

public class Llamada implements Serializable {
    private static final long serialVersionUID = 6164080316086841480L;

    private LocalDateTime fechaHora;
    private String emisor; // Nombre de la persona que llamo.
    private String motivo; // Motivo de la llamada.

    public Llamada() {
        this.fechaHora = LocalDateTime.now();
    }

    public LocalDateTime getFechaHora() {
        return fechaHora;
    }
    public void setFechaHora(LocalDateTime fechaHora) {
        this.fechaHora = fechaHora;
    }
    public String getEmisor() {
        return emisor;
    }
    public void setEmisor(String emisor) {
        this.emisor = emisor;
    }
    public String getMotivo() {
        return motivo;
    }
    public void setMotivo(String motivo) {
        this.motivo = motivo;
    }

    @Override
    public String toString() {
        return this.emisor + " llamo el " + this.fechaHora + " para " + this.-motivo;
    }
}
```

La clase *Llamada* cumple con la especificación *JavaBeans* porque es serializable, tiene un constructor sin argumentos, y sus métodos son privados y accesibles mediante métodos *get/set*, generados cumpliendo la nomenclatura estándar.

Podríamos construir un objeto en una clase con método *main()* de la siguiente forma:

```
public class Principal {  
  
    public static void main(String[] args) {  
        Llamada unaLlamada = new Llamada();  
        unaLlamada.setEmisor("Carlos Pérez");  
        unaLlamada.setMotivo("Pedir información");  
        System.out.println(unaLlamada); // Invoca al método toString()  
    }  
}
```

Proyecto web con un JavaBean

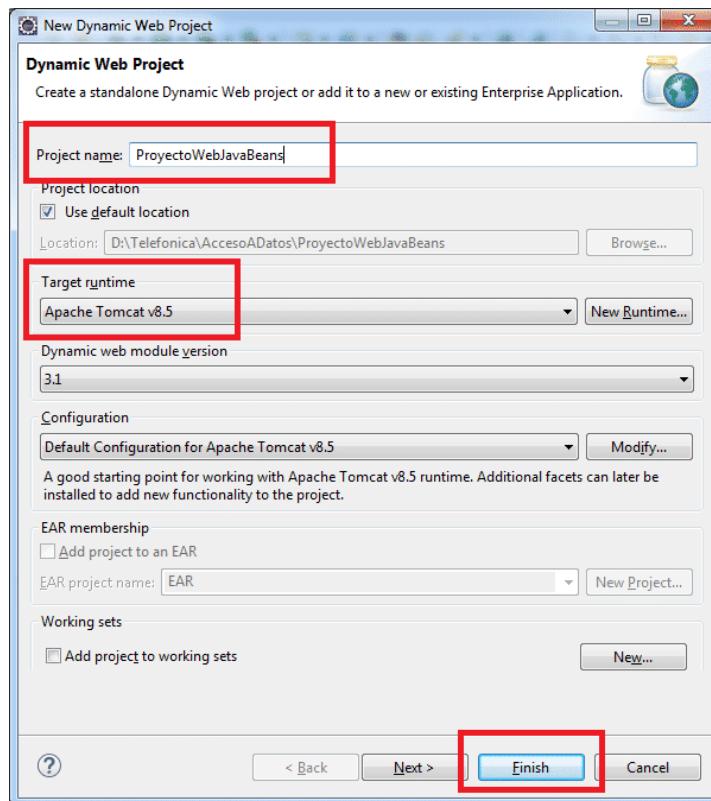
En este apartado, crearemos un proyecto web dinámico con un *JavaBean* para después utilizarlo dentro de una página JSP.

Sigue estos pasos:

1

Crea un proyecto web dinámico llamado *ProyectoWebJavaBeans*.

Sítuate en la perspectiva Java EE, si es que no estás ya en ella. Después, selecciona en el menú *File / New / Dynamic Web Project*. Recuerda que debes especificar el nombre del proyecto y el *Target runtime*, apuntando a la ubicación de la instalación de Apache Tomcat que tengas en tu equipo.



2

Crea la clase que servirá de JavaBean

Vamos a crear una clase llamada *Estudiante* en un paquete llamado *escuela*. Como para cualquier otro tipo de clase, haz clic derecho sobre el nombre del proyecto y selecciona *New / Class*.

Luego, edita el código hasta dejarlo así:

```
package escuela;

import java.io.Serializable;
import java.time.LocalDateTime;

public class Estudiante implements Serializable {
    private static final long serialVersionUID = 7556602068002620452L;

    String nombre;
    String curso;
    LocalDateTime fechaHoraMatricula;

    public Estudiante() {
        this.nombre = "Desconocido";
        this.curso = "Java";
        this.fechaHoraMatricula = LocalDateTime.now();
    }

    public String getNombre() {
        return nombre;
    }

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

    public String getCurso() {
        return curso;
    }

    public void setCurso(String curso) {
        this.curso = curso;
    }

    public LocalDateTime getFechaHoraMatricula() {
        return fechaHoraMatricula;
    }

    public void setFechaHoraMatricula(LocalDateTime fechaHoraMatricula) {
        this.fechaHoraMatricula = fechaHoraMatricula;
    }
}
```

Como puedes comprobar, la clase *Estudiante* cumple todos los requisitos para ser un *JavaBean*.

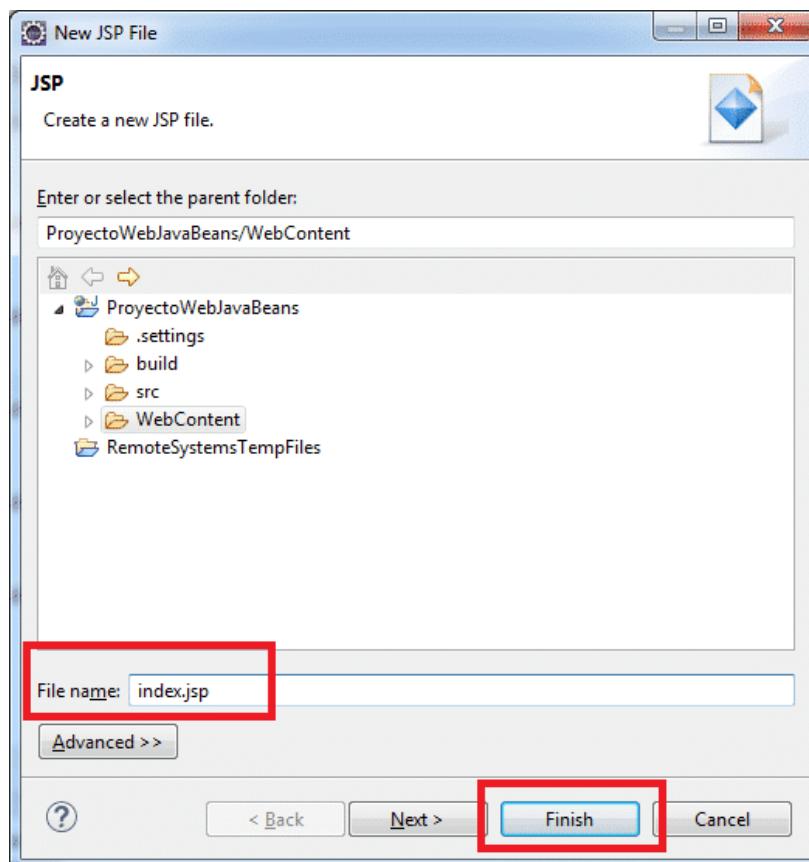
JSP y los JavaBeans

En el siguiente apartado, vamos a hacer uso de la clase *Estudiante* dentro de una página dinámica.

1

Comienza por crear tu archivo JSP.

Haz clic derecho sobre el nombre del proyecto y selecciona *New / JSP File* en el menú contextual. El nombre de la página será *index.jsp*.



2

Sustituye el código que aparece por defecto por el que te presentamos a continuación:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<jsp:useBean id="yo" class="escuela.Estudiante" />
<jsp:setProperty name="yo" property="nombre" value="Amelia González" />
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Página con un JavaBean</title>
</head>
<body>

</body>
</html>
```

Hemos aplicados dos directivas de JSP que no habías utilizado hasta ahora. Vamos a comprobarlo.

Las directivas *useBean* y *setProperty* de JSP

La directiva *jsp:useBean* permite construir un objeto a partir de una clase que cumpla con los requisitos de los *JavaBeans*. A los objetos creados de esta forma los denominamos *beans*.

```
<jsp:useBean id="yo" class="escuela.Estudiante" />
```

La línea anterior es equivalente a una declaración Java de este tipo:

escuela.Estudiante yo = new Estudiante();

El atributo *id* es el que determinará el nombre del objeto.

La directiva *jsp:setProperty* se utiliza para asignar valores a las propiedades de un *bean*.

```
<jsp:setProperty name="yo" property="nombre" value="Amelia González" />
```

La línea anterior es equivalente a cuando en Java escribimos lo siguiente:

yo.setNombre("Amelia González");

Añadir Scripts y expresiones JSP

Recuerda que puedes incluir código Java dentro de un script JSP que va encerrado entre los símbolos *<% %>*. En esta ocasión, utilizaremos un script para asignar el valor *Acceso a datos* a la propiedad *curso*; de este modo, comprobarás que también podemos seguir utilizando el sistema clásico de Java.

Completa el código hasta dejarlo así:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<jsp:useBean id="yo" class="escuela.Estudiante" />
<jsp:setProperty name="yo" property="nombre" value="Perico de los Palotes" />
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Página con un JavaBean</title>
</head>
<body>
<%
    yo.setCurso("Acceso a datos");
%>
<h1>Estudiante: <%=yo.getNombre() %></h1>
<h2>Fecha/hora matricula: <%=yo.getFechaHoraMatricula() %></h2>
<h3>Curso: <%=yo.getCurso() %></h3>
</body>
</html>
```

Dentro del código HTML hemos incluido expresiones JSP para mostrar los valores de las propiedades.

Recuerda que una expresión JSP va encerrada entre los símbolos `<%= %>`.

Ver el resultado en el navegador

Como sabes, puedes utilizar el botón *Run* dentro de Eclipse para ver cómo quedará la página *index.jsp* en el navegador.



Resumen

Has terminado la lección, vamos a ver los puntos más importantes que hemos tratado.

- Un *JavaBean* es una clase que debe cumplir una serie de condiciones: implementar la interfaz *Serializable*, tener un constructor vacío y propiedades privadas accesibles mediante métodos *get/set*.
- Por medio de la directiva *useBean* podemos construir un objeto a partir de una clase que cumpla con la especificación de los *JavaBean* dentro de una página JSP.
Ejemplo:
`<jsp:useBean id="yo" class="escuela.Estudiante"/>`
- Por medio de la directiva *setProperty* podemos asignar valores a las propiedades de un *bean* (objeto creado con *useBean* dentro de una página JSP).
`<jsp:setProperty name="yo" property="nombre" value="Perico de los Palotes" />`



PROEDUCA