	UNIVERSIDAD DON BOSCO **FACULTAD DE ESTUDIOS TECNOLÓGICOS**
CICLO I	**GUIA DE LABORATORIO #9** **Programación Orientada a Objetos** **JSP Y Java-Beans**

I. OBJETIVOS

Que el estudiante:

- Pueda comprenda el funcionamiento de los JavaBeans.
- Manipule los JavaBeans de tal manera que pueda separar la lógica de negocio de la presentación.

II. INTRODUCCIÓN

Un JavaBean se puede definir como un componente de software reutilizable. Lo que esto significa es que podemos escribir un JavaBean que luego puede ser utilizado en una variedad de otras aplicaciones basadas en software de Java tales como aplicaciones, Servlets o páginas JSP. De esta manera podemos definir nuestra lógica de negocio dentro de un JavaBean y luego utilizar sistemáticamente la lógica en las aplicaciones por separado.

Mediante el uso de JavaBeans se puede separar completamente la lógica de negocio de la generación de resultados en la pantalla. Esta es una filosofía importante que conduce a los sistemas mejor estructurados y más fáciles de mantener. En nuestro caso, debe utilizar la página de JavaServer (JSP) para generar dinámicamente la exhibición y también para manejar la interacción del usuario. El JavaBean tomaría el control del flujo de la aplicación cuando se necesite realizar algún procesamiento de datos complejos o cuando se necesita el acceso a bases de datos o el sistema de archivos.

La otra ventaja del uso de JavaBeans es que la lógica de negocio puede ser utilizada por más de una solicitud. Por ejemplo, tanto un cliente de aplicaciones basadas en Java como una JSP pueden acceder al mismo JavaBean garantizando la misma funcionalidad.

Un último punto a destacar es que mediante el uso de JavaBeans puede dividir su equipo de desarrollo de expertos en Java y los expertos en HTML. Los expertos de Java se dedicarán a escribir y desarrollar las JavaBeans y los expertos en HTML se concentrarán en el diseño de la aplicación web.

Uso en JSP

- El protocolo HTTP tiene un funcionamiento muy simple: un cliente hace una petición de documento, el servidor responde y termina la transacción. No almacena el estado de cada petición, es un protocolo “sin estado”.

- Por ello, se van a utilizar los JavaBeans integrados en nuestro formulario para poder

almacenar el estado de éste a lo largo de toda la sesión.

- Existen otras alternativas, como son los cookies, la reescritura de la URL o campos ocultos.

Escribir el Bean

- Lo primero es escribir el código en Java del JavaBean bajo el siguiente esquema:
  - El constructor de la clase no tiene argumentos.
  - Por cada componente del formulario tendremos una variable privada con el mismo nombre: private nombrePropiedad;
  - Cada variable tendrá dos métodos:
    - Un método accesor:
      - **public String getNombrePropiedad();**
    - Y un método mutador:
      - **public void setNombrePropiedad (String name);**

Por ejemplo, si tenemos un formulario con un componente text llamado peticion, el JavaBean sería el siguiente:

```
public class EjemploBean{
   private String peticion;
    public EjemploBean() {
       peticion = null;
    }
    public void setPeticion( String nombre ) {
        peticion = nombre;
    }
    public String getPeticion() {
         return peticion;
     }
}
```

Utilizar el JavaBean en JSP

La acción <jsp:useBean>

La acción <jsp:useBean> indica a la página JSP que deseamos tener a nuestra disposición un JavaBean determinado, el contenedor de páginas JSP creará el JavaBean correspondiente o bien lo recuperará del ámbito adecuado si éste ya existe. La sintaxis básica de esta acción es la siguiente:

<jsp:useBean id="nombre" class="nombreClase"/>

También tenemos esta otra sintaxis:

<jsp:useBean id="nombre" class="nombreClase"/>
//Código de inicialización
</jsp:useBean>

Es decir, podemos utilizar la sintaxis de una única línea, o bien la de varias líneas indicando un código de inicialización que deseamos que se ejecute, este código de inicialización sólo se ejecutará si se crea el JavaBean.

Atributos de la acción <jsp:useBean>

- **id:** Es el identificador que vamos a utilizar dentro de la página JSP, y durante el resto del ciclo de vida del JavaBean para hacer referencia al mismo. Se puede elegir cualquier nombre para hacer referencia a un JavaBean, aunque se deben seguir una serie de normas: este identificador debe ser único en la página, se distingue entre mayúsculas y minúsculas, el primer caracter debe ser una letra, sólo se permiten letras, números y guión bajo (_), no se permiten espacios en blanco.

- **class:** En este atributo se indica el nombre de la clase del JavaBean; a cada uno de ellos le va a corresponder una clase, al igual que sucede con los applets, que poseen una clase principal que representa la clase del componente, para organizar los componentes.

- **scope:** Indica el ámbito que le va a corresponder al JavaBean, existen cuatro ámbitos distintos, y por lo tanto este atributo podrá tener los valores page, request, session o application. Por defecto se utiliza el ámbito de página (page).

Ámbito	Accesibilidad	Existencia
page	Únicamente la página actual.	Hasta que la página se ha terminado de mostrar o el control es redirigido hacia otro recurso.
request	Página actual y cualquiera que se incluya o redirija.	Hasta que la petición se ha procesado completamente y la respuesta ha sido enviada al usuario.
session	La petición actual y cualquiera de las siguientes peticiones que tengan lugar en la misma ventana (sesión) del navegador.	La vida de la sesión del usuario.
application	La petición actual y cualquiera de las siguientes peticiones que tengan lugar en la misma aplicación web.	La vida de la aplicación web.

En el código siguiente se muestra un sencillo ejemplo de una página JSP para crear un Bean con ámbito de página. Como se puede observar, se ha utilizado la clase java.util.Date como clase del Bean, lo normal es utilizar un componente JavaBeans pero para este ejemplo la clase Date nos sirve perfectamente para mostrar como más tarde se puede acceder en la página al Bean que se ha creado utilizando su identificador.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
        <head>
            <title>Acción jsp:useBean</title>
        </head>
        <body>
                <jsp:useBean id="fecha" scope="page" class="java.util.Date"/>
                <%=fecha%>
        </body>
</html>
```

La acción <jsp:getProperty>
Esta acción es una de tantas que nos permite utilizar componentes JavaBeans dentro de nuestras páginas JSP, y tiene como función permitirnos obtener el valor de la propiedad de un JavaBean creado en la página con el ámbito correspondiente, convirtiéndolo en un objeto de la clase String que posteriormente imprime en el flujo de salida de la página JSP.

Su sintaxis es muy sencilla, no posee cuerpo y únicamente presenta dos atributos o propiedades, como se puede observar a continuación:

<jsp:getProperty name="nombreBean" property="nombrePropiedad"/>

- **name:** Indica el identificador del JavaBean que hemos creado con la acción <jsp:useBean>, y cuyo valor de cierta propiedad que posee queremos obtener. Se corresponderá con el valor del atributo id de la acción <jsp:useBean> adecuada.

- **property:** Indica el nombre de la propiedad del JavaBean cuyo valor se desea obtener. El valor de la propiedad se mostrará como código HTML, reemplazando en tiempo de ejecución a la acción <jsp:getProperty> pertinente.

La acción <jsp:setProperty>
Esta acción permite modificar las propiedades de los JavaBeans a los que hacemos referencia en nuestras páginas JSP a través de la acción <jsp:useBean>, es la acción complementaria a <jsp:getProperty>. Su sintaxis general es la que se muestra a continuación:

<jsp:setProperty name="nombreBean" detallesPropiedad/>

Donde:

- **name:** Tiene el mismo significado que en la acción vista anteriormente, es decir, es el identificador del componente JavaBean al que se hace referencia en la página.

- Los detalles de la propiedad son una serie de atributos que combinados entre sí permiten asignar el valor a la propiedad del JavaBean de distintas formas. Así por ejemplo, la forma de establecer el valor de la propiedad de un Bean puede ser cualquiera de las que aparecen a continuación:
  - property="*"
  - property="nombrePropiedad"
  - property="nombrePropiedad" param="nombreParámetro"
  - property="nombrePropiedad" value="valorPropiedad"

- **property:** El nombre de la propiedad del JavaBean cuyo valor se desea establecer. Este atributo puede tener asignado un valor especial que el asterisco ("*"). Si indicamos el asterisco, de forma automática la etiqueta iterará sobre todos los parámetros del objeto request correspondiente estableciendo los nombres de las propiedades del JavaBean que coincidan con el nombre de los parámetros del objeto request, asignándole el valor del parámetro cuando se dé dicha coincidencia. Si un parámetro del objeto request posee el valor de vacío ("") no se modificará el valor de la propiedad del JavaBean. Con el asterisco podemos establecer el valor de varias propiedades del JavaBean de una sola vez, más adelante lo veremos mediante un ejemplo.

- **param:** Este atributo permite indicar el nombre del parámetro del objeto request que se va a utilizar para establecer el valor de la propiedad del JavaBean indicadas en el atributo

property**.** Gracias a este atributo no es necesario que el JavaBean posea el mismo nombre de propiedad que el parámetro del objeto request cuyo valor deseamos establecer para la propiedad. Si no se especifica el atributo param se asume que el nombre de la propiedad y el nombre del parámetro del objeto request es el mismo.

- **value:** Contiene el valor que se va a asignar a la propiedad, puede ser una cadena una expresión válida. Una acción <jsp:setProperty> no puede presentar los atributos value y param al mismo tiempo.

El valor de una propiedad de un JavaBean se puede establecer a partir de varios elementos:

- En el tiempo de la petición de la página a partir de los parámetros existentes en el objeto integrado request.
- En el tiempo de ejecución de la página a partir de la evaluación de una expresión válida de JSP.
- A partir de una cadena de caracteres indicada o como una constante en la propia página.

III. PROCEDIMIENTO

Uso de JavaBeans y JSP

NOTA: Para el desarrollo de los ejemplos se hará uso nuevamente del framework Bootstrap, por lo que será necesario que vuelva a colocar en el proyecto los archivos .js, .css y las fuentes correspondientes.

1. Para esta guía crear un nuevo proyecto con el nombre de **Guia9POO1.**

2. Crear una página JSP que se llame **"Bean1"** y que contendrá el siguiente código.

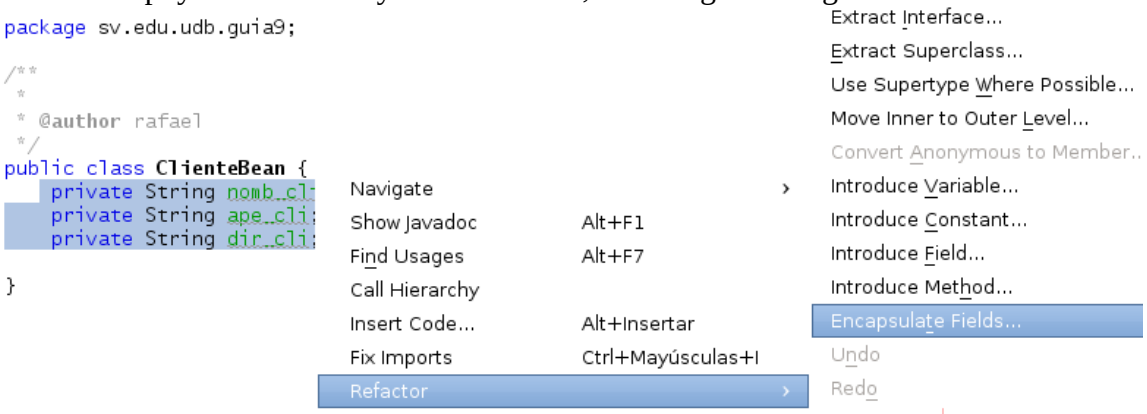
```
<%
        String nom_cli="Rafael";
    String ape_cli="Torres";
    String dir_cli="My_House";
%>
<jsp:useBean id="cli_bean" class="sv.edu.udb.guia9.ClienteBean"/>
<jsp:setProperty name="cli_bean" property="nomb_cli" value="<%=nom_cli%>"/>
<jsp:setProperty name="cli_bean" property="ape_cli" value="<%=ape_cli%>"/>
<jsp:setProperty name="cli_bean" property="dir_cli" value="<%=dir_cli%>"/>

<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" href="css/bootstrap.min.css">
   </head>
   <body>
      <div class="container">
         <div class="row">
             
         </div>
```

```
        <div class="panel panel-primary">
          <div class="panel-heading">Datos personales</div>
          <div class="panel-body">
            <%
               out.println("<h3>Nombre: " + cli_bean.getNomb_cli() + "</h3>");
               out.println("<h3>Apellido: " + cli_bean.getApe_cli() + "</h3>");
               out.println("<h3>Direcci&oacute;n: " + cli_bean.getDir_cli() + "</h3>");
            %>
          </div>
        </div>
      </div>
    </body>
</html>
```

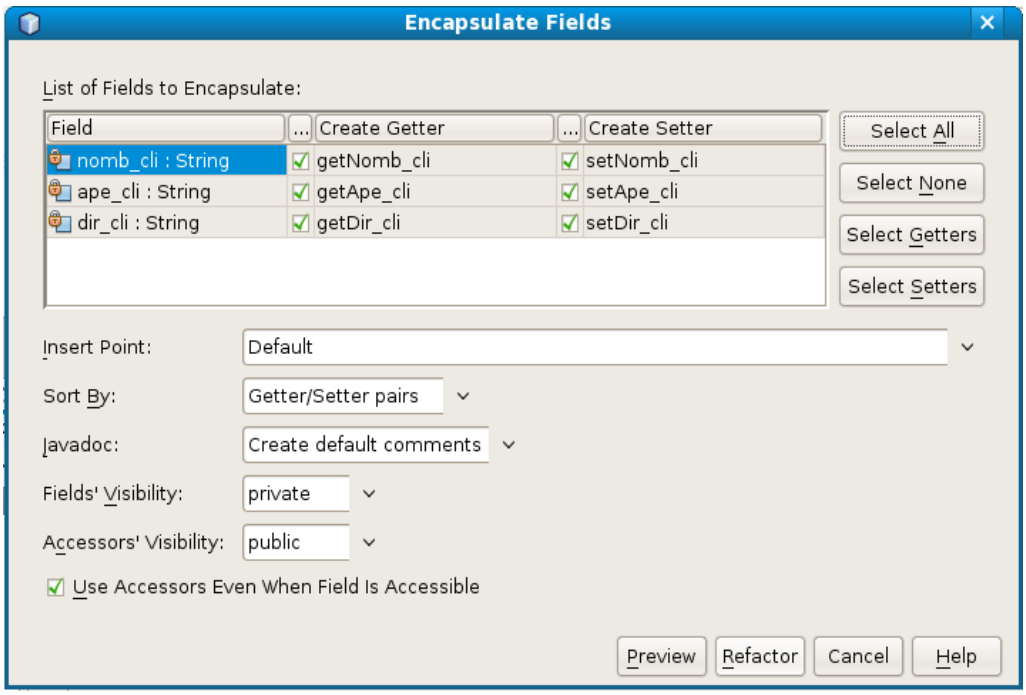
3. Crear un clase llamada **“ClienteBean”** que será nuestro JavaBean en el paquete **“sv.edu.udb.guia9”.**

4. Agregar las siguientes propiedades:
   private String nomb_cli;
   private String ape_cli;
   private String dir_cli;

5. Ahora crearemos los metodos setXXX y getXXX correspondientes a estas propiedades. Para ello nos apoyaremos de la ayuda de netbens, ver la siguiente figura.



Para generar los métodos se debe de sombrear las propiedades indicadas en el punto cuatro, luego seleccionamos **“Refactor”** y por ultimo **“Encapsulate Fields...”** deber da aparecer una pantalla como la siguiente.

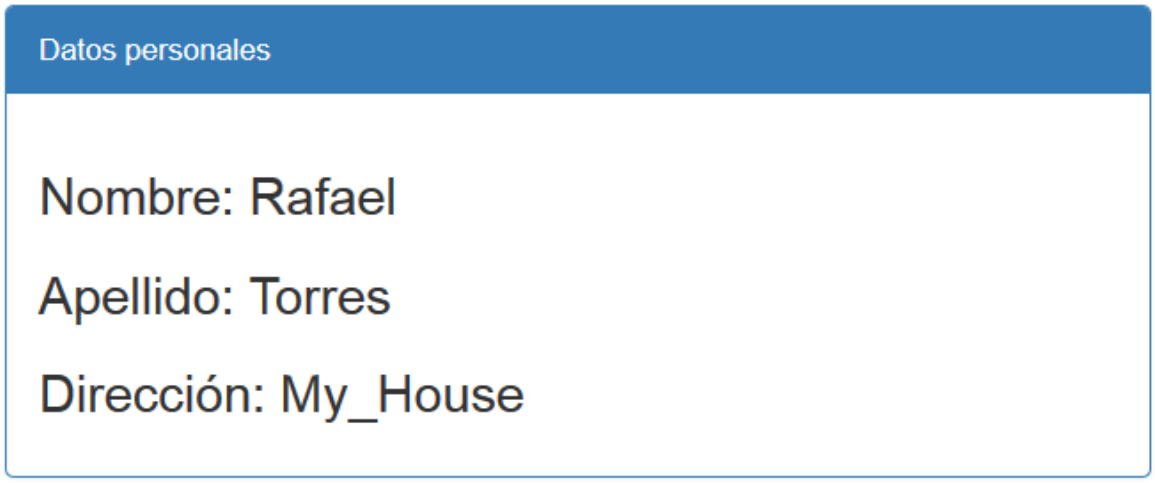


Seleccionamos la opcion **"Select All"** y damos click en **"Refactor"**, todo ello nos debería generar la siguiente estructura en la clase:

```
public class ClienteBean {
    private String nomb_cli;
    private String ape_cli;
    private String dir_cli;

    /**
     * @return the nomb_cli
     */
    public String getNomb_cli() {
        return nomb_cli;
    }

    /**
     * @param nomb_cli the nomb_cli to set
     */
    public void setNomb_cli(String nomb_cli) {
        this.nomb_cli = nomb_cli;
    }

    /**
     * @return the ape_cli
     */
    public String getApe_cli() {
        return ape_cli;
    }

    /**
     * @param ape_cli the ape_cli to set
     */
    public void setApe_cli(String ape_cli) {
```

```
        this.ape_cli = ape_cli;
    }

    /**
     * @return the dir_cli
     */
    public String getDir_cli() {
        return dir_cli;
    }

    /**
     * @param dir_cli the dir_cli to set
     */
    public void setDir_cli(String dir_cli) {
        this.dir_cli = dir_cli;
    }
}
```

El resultado en el navegador será el siguiente:



JavaBeans y acciones <jsp:getProperty> y <jsp:setProperty>

Para esta parte manipularemos desde la JSP las propiedades creadas con las acciones correspondientes.

1. Crear una nueva página jsp llamada **"LenguajeFavorito".**
2. Digitar el siguiente código.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" href="css/bootstrap.min.css">
   </head>
   <body>
      <div class="container">
         <div class="row">
             
         </div>
         <div class="panel panel-primary">
            <div class="panel-heading">Uso de JavaBean</div>
            <div class="panel-body">
               <h1>P&aacute;gina de prueba del uso de beans</h1>
               <p>Se env&iacute;a el formulario al servicio cuyo fichero es
<mark>beans.jsp</mark></p>
            </div>
         </div>
         <div class="row col-md-12" >
            <form role="form" action="beans.jsp" method="POST">
               <div class="col-md-10">
                  <div class="form-group">
                     <label for="nombre">Por favor, introduce tu nombre:</label>
                     <div class="input-group">
                        <input type="text" class="form-control" name="nombre" id="nombre"
placeholder="Ingresa tu nombre">
                        <span class="input-group-addon"></span>
                     </div>
                  </div>
                  <div class="form-group">
                     <label for="lenguaje">¿Cuál es tu lenguaje de programación favorito? </label>
                     <div class="input-group">
                        <select name="lenguaje" class="form-control">
                           <option value="Java">Java
                           <option value="C++">C++
                           <option value="Perl">Perl
                        </select>
                        <span class="input-group-addon"></span>
                     </div>
                  </div>
                  <input type="submit" class="btn btn-info" value="Enviar">
               </div>
            </form>
         </div>
      </div>
   </body>
</html>
```

3. Ahora crearemos el beans llamado **“LenguajeBean”** en el paquete **“sv.edu.udb.guia9”**, este contendrá las siguientes propiedades.
   private String nombre;
   private String lenguaje;

   Crear los métodos setXXX y getXXX correspondientes, después de que sean creados

deberemos crear un nuevo método llamando **getComentarios** y que debe de quedar de la siguiente manera.

```
public String getComentarios(){
      if (lenguaje.equals("Java")){
         return "El rey de los Lenguajes Orientados a Objetos";
      }
      else if (lenguaje.equals("C++")){
         return "Demasiado complejo";
      }
      else if (lenguaje.equals("Perl")){
         return "OK si te gusta el código incomprensible";
      }
      else {
         return "Lo siento, no conozco el lenguaje " + lenguaje ;
      }
   }
```

4. Crear la página que recibirá los parámetros esta sera llamada **"Beans.jsp"**, digitar el código siguiente.

```
<jsp:useBean id="lenguajeBean" scope="page" class="sv.edu.udb.guia9.LenguajeBean"/>
<jsp:setProperty name="lenguajeBean" property="*"/>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
   <head>
      <title>Resultado de prueba del uso de beans</title>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" href="css/bootstrap.min.css">
   </head>
   <body>
      <div class="container">
         <div class="row">
             
         </div>
         <div class="panel panel-primary">
            <div class="panel-heading">Resultado</div>
            <div class="panel-body">
               <p>Hola: <mark><jsp:getProperty name="lenguajeBean"
property="nombre"/></mark>.</p>

               <p>Tu lenguaje favorito es: <mark><jsp:getProperty name="lenguajeBean"
property="lenguaje"/></mark></p>

               <p>Mis comentarios acerca del lenguaje son: </p>
               <p class="bg-info"><jsp:getProperty name="lenguajeBean"
property="comentarios"/> </p>
            </div>
```

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

5. Corriendo la aplicación, para ello ejecute la página llamada **LenguajeFavorito.jsp**, ingresar un nombre y elegir un lenguaje y por ultimo dar click en enviar.

<u>Un segundo ejemplo con acciones <jsp:getProperty> y <jsp:setProperty></u>

1. Crear una página JSP llamada **"EnviarPersona"** y digitar el siguiente código.

```
<!DOCTYPE html>
<html>
   <head>
      <title>Personas</title>
      <meta charset="utf-8">
```

```
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" href="css/bootstrap.min.css">
   </head>
   <body>
      <div class="container">
         <div class="row">
            <h3>Personas</h3>
         </div>
         <div class="row">
         <form role="form" name="persona" action="Persona.jsp" method="POST">
            <div class="col-md-10">
               <div class="form-group">
                  <label for="nombre">Ingrese su nombre:</label>
                  <div class="input-group">
                     <input type="text" class="form-control" name="nombre" id="nombre"
placeholder="Nombre">
                     <span class="input-group-addon"></span>
                  </div>
               </div>
               <div class="form-group">
                  <label for="edad">Ingrese su edad:</label>
                  <div class="input-group">
                     <input type="text" class="form-control" id="edad" name="edad"
placeholder="Edad">
                     <span class="input-group-addon"></span>
                  </div>
               </div>
               <input type="submit" class="btn btn-info" value="Enviar">
            </div>
         </form>
         </div>
      </div>
   </body>
</html>
```

2. Crear el JavaBean con el nombre **“PersonaBean”** y el paquete **“sv.edu.udb.guia9”.**
3. Agregar las propiedades
   private String nombre;
   private int edad;

   Generar los métodos setXXX y getXXX correspondientes.
4. Crear los métodos getTipo() y getJoven() y digitar el siguiente código.

```
   public String getTipo() {
         if (edad < 40)
         return "joven";

         return "no joven";
   }

   public boolean getJoven() {
```

```
    if (edad < 40){
        return true;
    }
    return false;
  }
```

5. Crear la página JSP que recibirá los datos y que debe ser llamada **“Persona.jsp”** y digitar el código siguiente.

```
<!DOCTYPE html>
<jsp:useBean id="cientifico" scope="request" class="sv.edu.udb.guia9.PersonaBean">
   <jsp:setProperty name="cientifico" property="*"/>
</jsp:useBean>
<html>
   <head>
      <title>jsp:useBean</title>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" href="css/bootstrap.min.css">
   </head>
   <body>
      <div class="container">
         <div class="row">
             
         </div>
         <div class="panel panel-primary">
            <div class="panel-heading">Uso de jsp:useBean coordinado con parámetros de la
petición</div>
            <div class="panel-body">
               <p>El científico es: <mark><jsp:getProperty name="cientifico"
property="nombre"/></mark>. <br>
                  Su edad es: <mark><jsp:getProperty name="cientifico" property="edad"/></mark>
años.</p>
               <p>A continuación usamos <code>getProperty</code>, sin que haya una propiedad
de clase Bean para soportar los métodos <code>getTipo()</code> y
<code>getEsJoven()</code>:</p>
               <ul>
                  <li>Tipo: <mark><jsp:getProperty name="cientifico"
property="tipo"/></mark></li>
                  <li>¿Joven?: <mark><jsp:getProperty name="cientifico"
property="joven"/></mark></li>
               </ul>
            </div>
         </div>
      </div>
   </body>
</html>
```

6. Correr el ejemplo anterior y visualizar el resultado.

Crear un código para una persona con JavaBeans

1. Para esta parte deber crear una base de datos llamada **"guia9poo1"** con una tabla llamada **"empleados"** y con los siguientes campos.
   -id
   -nombres
   -apellidos
   -edad
2. Ingresar unos cuantos registros como se ve a continuación.

id	nombres	apellidos	edad
VR1	Kelman	Guzman	23
VR2	Manuel	Valdez	24
VR3	Carmen	Quintanilla	23
VR4	Claudia	Juarez	23

3. Crear la página JSP que contiene el formulario, llamarla **"Datos"** y debe de quedar de la siguiente manera.

```
<!DOCTYPE html>
<html>
   <head>
      <title>Datos personales</title>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" href="css/bootstrap.min.css">
   </head>
   <body>
      <div class="container">
         <div class="row">
            <h3>Datos personales</h3>
         </div>
         <div class="row">
         <form role="form" name="persona" action="Procesar.jsp" method="POST">
            <div class="col-md-10">
               <div class="form-group">
                  <label for="nombre">Ingrese su nombre:</label>
                  <div class="input-group">
                     <input type="text" class="form-control" name="nombre" id="nombre"
placeholder="Nombre">
                     <span class="input-group-addon"></span>
                  </div>
               </div>
               <div class="form-group">
                  <label for="apellido1">Ingrese su primer apellido:</label>
                  <div class="input-group">
                     <input type="text" class="form-control" id="apellido1" name="apellido1"
placeholder="Primer Apellido">
                     <span class="input-group-addon"></span>
```

```
                </div>
            </div>
            <div class="form-group">
                <label for="apellido2">Ingrese su segundo apellido:</label>
                <div class="input-group">
                    <input type="text" class="form-control" id="apellido2" name="apellido2"
placeholder="Segundo Apellido">
                    <span class="input-group-addon"></span>
                </div>
            </div>
            <input type="submit" class="btn btn-info" value="Enviar">
        </div>
    </form>
    </div>
  </div>
 </body>
</html>
```

4. Ahora crearemos el JavaBeans y lo llamaremos **"CodigoBean"** siempre en el paquete **"sv.edu.udb.guia9"** y digite el siguiente código.

```
package sv.edu.udb.guia9;
import java.text.SimpleDateFormat;
import java.util.Date;

public class CodigoBean {
   private String apellido1;
   private String apellido2;
   private int cantidad_registros;
   private String anio_actual;
   private String soloLetra;
   private String soloLetra2;

   /**
   * @return the apellido
   */
   public String getApellido1() {
      return apellido1;
   }

   /**
   * @param apellido the apellido to set
   */
   public void setApellido1(String apellido1) {
      this.apellido1 = apellido1;
   }

   /**
   * @return the apellido2
   */
   public String getApellido2() {
      return apellido2;
```

```
    }

    /**
    * @param apellido2 the apellido2 to set
    */
    public void setApellido2(String apellido2) {
        this.apellido2 = apellido2;
    }

    /**
    * @return the cantidad_registros
    */
    public int getCantidad_registros() {
        return cantidad_registros;
    }

    /**
    * @param cantidad_registros the cantidad_registros to set
    */
    public void setCantidad_registros(int cantidad_registros) {
        this.cantidad_registros = cantidad_registros;
    }

    public String getFecha() {
        Date d = new Date();
        SimpleDateFormat anio=new SimpleDateFormat("yy");
        anio_actual=anio.format(d);
        return anio_actual;
    }

    public String getLetra() {
        soloLetra = apellido1.substring(0,1);
        return soloLetra;
    }

    public String getLetra2() {
        soloLetra2 = apellido2.substring(0,1);
        return soloLetra2;
    }

    public String getCod(){
        int aumentregistros=cantidad_registros+1;
        String regitros=String.valueOf(aumentregistros);
        String codigo= soloLetra + soloLetra2+anio_actual+regitros;
        return codigo;
    }
}
```

5. Ahora creamos la página que utilizara el JavaBeans y la llamaremos **“Procesar”.**

```
<!DOCTYPE html>
<%@ include file="conexion.jsp"%>
```

```
<jsp:useBean id="codigo" class="sv.edu.udb.guia9.CodigoBean"/>
<%
int total_registros=0;
rs = s.executeQuery("select count(*) from empleados");
%>

<% while (rs.next()) {
   total_registros=rs.getInt(1);
}%>

<jsp:setProperty name="codigo" property="apellido1" param="apellido1"/>
<jsp:setProperty name="codigo" property="apellido2" param="apellido2"/>
<jsp:setProperty name="codigo" property="cantidad_registros" value="<%=total_registros%>"/>
<html>
   <head>
      <title>jsp:useBean</title>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" href="css/bootstrap.min.css">
   </head>
   <body>
      <div class="container">
         <div class="row">
             
         </div>
         <div class="panel panel-primary">
            <div class="panel-heading">Resultado</div>
            <div class="panel-body">
               <p>Su apellido1 es: <strong><jsp:getProperty name="codigo"
property="apellido1"/></strong></p>
               <p>Su apellido2 es: <strong><jsp:getProperty name="codigo"
property="apellido2"/></strong></p>
               <p>Año: <strong><jsp:getProperty name="codigo" property="fecha"/></strong></p>
               <p>Primera Letra: <strong><jsp:getProperty name="codigo"
property="letra"/></strong></p>
               <p>Segunda Letra: <strong><jsp:getProperty name="codigo"
property="letra2"/></strong></p>
               <p>Codigo obtenido: <code><jsp:getProperty name="codigo"
property="cod"/></code></p>
            </div>
         </div>
      </div>
   </body>
</html>
```

6. Crear el archivo de conexion y agregar el conector de MySQL.
7. Correr el archivo de Datos.jsp

IV. EJERCICIOS COMPLEMENTARIOS

- Realice el mantenimiento de dos tablas del proyecto, utilizando JavaBeans.

Hoja de cotejo.

HOJA DE EVALUACIÓN

Carnet:	Alumno:
Fecha:	Docente:
No.:	Título de la guía:

Actividad a evaluar	Criterio a evaluar	Cumplió		Puntaje
		SI	NO	
Desarrollo	Realizó los ejemplos de guía de práctica (40%)			
	Presentó todos los problemas resueltos (20%)			
	Funcionan todos correctamente y sin errores (30%)			
	Envió la carpeta comprimida y organizada adecuadamente en subcarpetas de acuerdo al tipo de recurso (10%)			
	PROMEDIO:			
Investigación complementaria	Entregó la investigación complementaria en la fecha indicada (20%)			
	Resolvió todos los ejercicios planteados en la investigación (40%)			
	Funcionaron correctamente y sin ningún mensaje de error a nivel de consola o ejecución (40%)			
	PROMEDIO:			