

13. Acceso a bases de datos

En el capítulo 11 vimos cómo almacenar información en ficheros de texto de tal forma que esta información no se pierde cuando el programa se cierra. Normalmente, los ficheros de texto se usan en casos concretos como archivos de configuración o archivos de registro (*log*).

Las bases de datos relacionales entran en juego cuando se necesita almacenar mucha información y, además, esta información es homogénea y se puede representar mediante tablas. Por ejemplo, una aplicación que permita gestionar la matriculación de los alumnos de un centro educativo deberá ser capaz de almacenar los datos de gran cantidad de alumnos (puede que miles). En este caso concreto, cada alumno se representa mediante un **registro** que contiene determinados datos como el número de expediente, el nombre, los apellidos, la nota media, etc. A estos datos se les llama **campos**.

En este capítulo vamos a ver cómo se puede acceder mediante Java a una base de datos relacional **MySQL** para hacer listados, insertar nuevos elementos y borrar elementos existentes.

Los programas de ejemplo que se muestran en este capítulo son aplicaciones JSP, aprovechando lo visto en el capítulo anterior, aunque también se puede acceder a una base de datos desde un programa en Java escrito para la consola.

13.1 Socios de un club de baloncesto

Para ilustrar el acceso a una base de datos mediante Java vamos a utilizar como ejemplo un club de baloncesto. Nuestro club necesita tener almacenada la información de todos los socios. Sobre cada socio se necesita saber su nombre completo, su estatura, su edad y su localidad de residencia. Cada socio tendrá, además, un número de identificación único que será el número de socio.

Nuestra aplicación consistirá en varias páginas JSP que van a permitir ver un listado con los datos de todos los socios, dar de alta un nuevo miembro en el club, y también borrar o modificar los datos de un determinado socio.

Si todavía no tienes instalado en tu equipo el gestor de bases de datos **MySQL**, deberás instalarlo. También es recomendable instalar **PHPMyAdmin** que es una aplicación que permite crear y manejar bases de datos MySQL de una forma muy sencilla, mediante una *interfaz* web.



Instalación de MySQL

```
sudo apt-get install mysql-server
sudo apt-get install mysql-client
```

Para el usuario `root` de MySQL, deberás establecer la contraseña `root` para que te funcionen correctamente los ejemplos.



Instalación de PHPMyAdmin

```
sudo apt-get install phpmyadmin
```

Para acceder a **PHPMyAdmin** debes escribir `http://localhost/phpmyadmin/` en la barra de direcciones del navegador.

Deberás crear la base de datos MySQL a la que llamarás `baloncesto`, que contendrá una tabla de nombre `socio` con los campos `socioID`, `nombre`, `estatura`, `edad` y `localidad` y luego deberás añadir los datos de los socios. Todo esto lo tienes ya preparado en un fichero con nombre `baloncesto.sql` que está en GitHub, en la dirección https://github.com/LuisJoseSanchez/aprende-java-con-ejercicios/blob/master/ejemplos/13_JSP_y_BBDD/baloncesto.sql. Descárgate este fichero, lo utilizaremos en seguida.

A continuación mostramos un fragmento del fichero `baloncesto.sql` para que veas cómo se crea la base de datos, la tabla `socio` y cómo se introducen los datos de muestra.

```
--
-- Base de datos: `baloncesto`
--
CREATE DATABASE IF NOT EXISTS `baloncesto` DEFAULT CHARACTER SET utf8 COLLATE utf8_bin;
USE `baloncesto`;

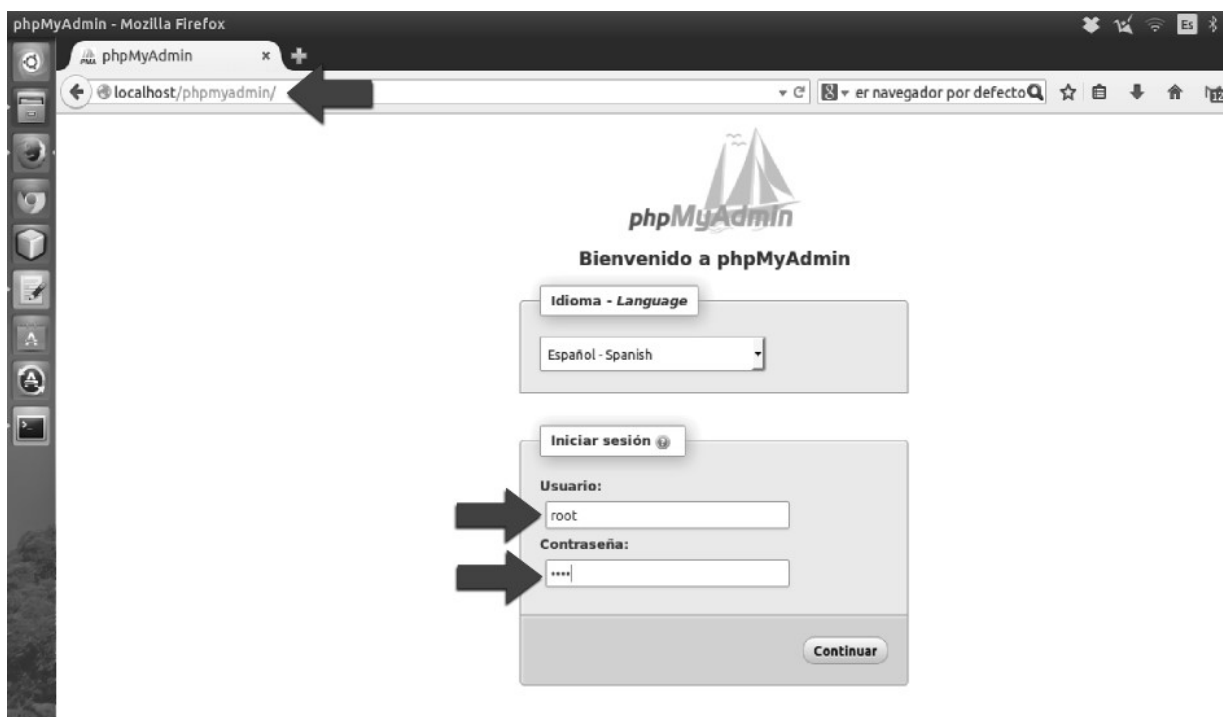
-- -----

--
-- Estructura de tabla para la tabla `socio`
--

CREATE TABLE IF NOT EXISTS `socio` (
  `socioID` int(11) NOT NULL,
  `nombre` varchar(40) COLLATE utf8_spanish2_ci DEFAULT NULL,
  `estatura` int(11) DEFAULT NULL,
  `edad` int(11) DEFAULT NULL,
  `localidad` varchar(30) COLLATE utf8_spanish2_ci DEFAULT NULL,
  PRIMARY KEY (`socioID`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish2_ci;  
  
--  
-- Volcado de datos para la tabla `socio`  
--  
  
INSERT INTO `socio` (`socioID`, `nombre`, `estatura`, `edad`, `localidad`) VALUES  
(1235, 'Bermúdez Espada, Ana María', 186, 46, 'Málaga'),  
(1236, 'Cano Cuenca, Margarita', 161, 48, 'Málaga'),  
...
```

En dos sencillos pasos tendrás preparada y lista para usar la base de datos de ejemplo. Para acceder a PHPMYAdmin debes escribir `http://localhost/phpmyadmin/` en la barra de direcciones de tu navegador. A continuación, teclea el nombre de usuario y la contraseña. Si has instalado MySQL como te hemos indicado, tanto el nombre de usuario como la contraseña deberían ser `root`.



A continuación, haz clic en la pestaña **Importar** y seguidamente en el botón **Examinar**. Selecciona el fichero `baloncesto.sql` que descargaste antes.



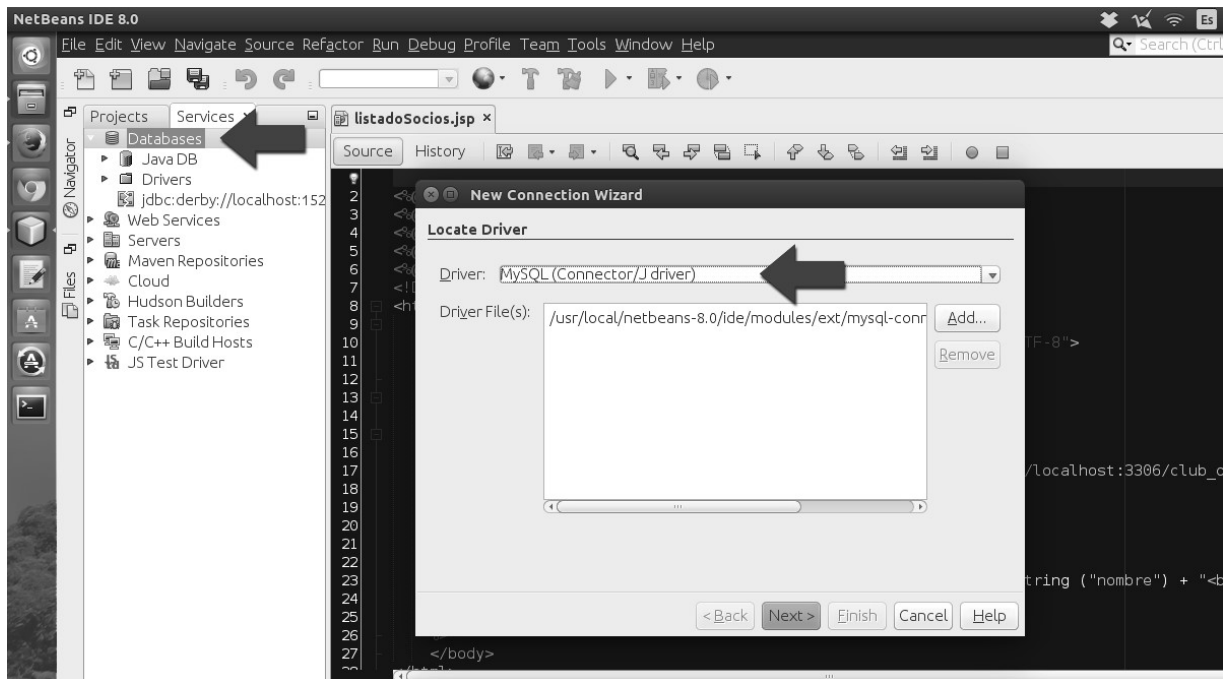
Si has seguido correctamente los pasos, ya tienes la base de datos lista para usar desde un programa escrito en Java.

13.2 Preparación del proyecto de ejemplo

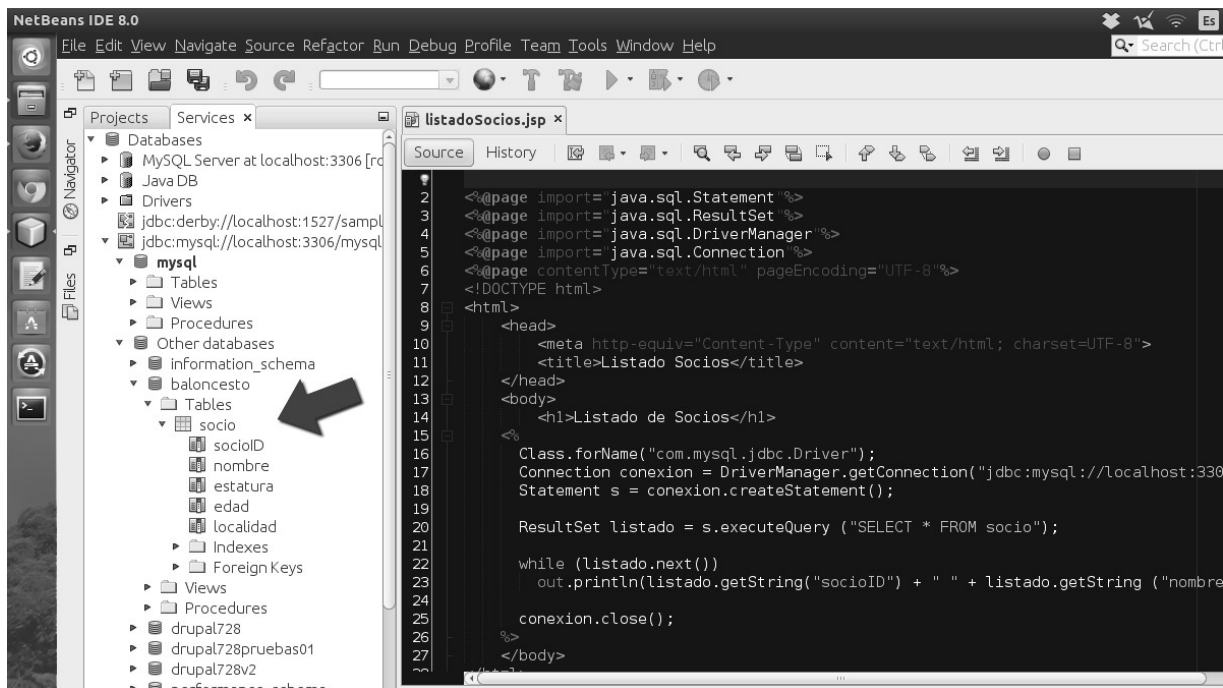
Activar la conexión a MySQL

Abre el entorno Netbeans y crea un nuevo proyecto del tipo **Java Web** - como vimos en el capítulo 12 - y nómbralo **Baloncesto**.

Ahora es necesario activar el servicio de bases de datos. Haz clic en la pestaña **Servicios** (está junto a la pestaña **Proyectos**). Haz clic con el botón derecho en **Bases de datos** y luego selecciona **Nueva conexión**. A continuación aparece una ventana para seleccionar el driver, debes seleccionar **MySQL (Connector/Jdriver)**.



Una vez se haya establecido el servicio, verás las bases de datos disponibles en una estructura de árbol. Puedes ir desplegando hasta encontrar la base de datos `baloncesto` y la tabla `socio`, que contiene los campos `socioID`, `nombre`, `estatura`, `edad` y `localidad`

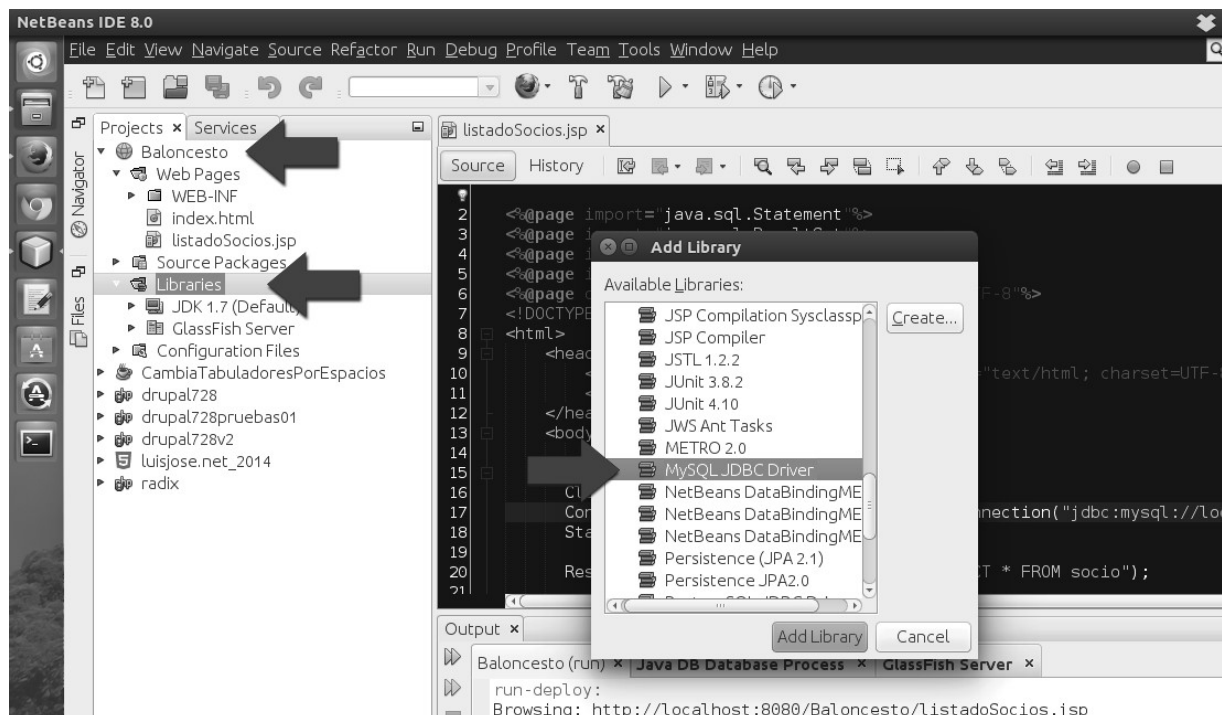


Una vez activada la conexión a MySQL, no es necesario realizarla de nuevo para cada

proyecto.

Incluir la librería MySQL JDBC

Cada proyecto de aplicación en Java que haga uso de una base de datos MySQL deberá incluir la librería **MySQL JDBC**. Para incluir esta librería, despliega el proyecto **Baloncesto**, haz clic con el botón derecho en **Librerías**, selecciona **Añadir librería** y, por último, selecciona **MySQL JDBC Driver**.



13.3 Listado de socios

El primer ejemplo de acceso a una base de datos mediante Java que veremos será un listado. Si echas un vistazo desde PHPMyAdmin a la tabla `socio` de la base de datos `baloncesto` verás que ya contiene información sobre más de 20 socios. Nuestro primer ejemplo será una aplicación en Java que muestre todos esos datos en una página web. Copia el archivo `listadoSocios.jsp`¹ a la carpeta principal del proyecto (donde está `index.html`). Para ejecutarlo, haz clic con el botón derecho y selecciona **Ejecutar**.

A continuación tienes el código JSP del listado de socios.

¹ https://github.com/LuisJoseSanchez/aprende-java-con-ejercicios/blob/master/ejemplos/13_SP_y_BBDD/Baloncesto/listadoSocios.jsp

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Listado Socios</title>
  </head>
  <body>
    <h1>Listado de Socios</h1>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/baloncesto"
, "root", "root");
      Statement s = conexion.createStatement();

      ResultSet listado = s.executeQuery ("SELECT * FROM socio");

      while (listado.next()) {
        out.println(listado.getString("socioID") + " " + listado.getString("nombre") + "<br>");
      }

      conexion.close();
    %>
  </body>
</html>
```

Si todo ha salido bien, al ejecutar el archivo - haciendo clic con el botón derecho sobre el archivo y seleccionando **Ejecutar** - verás por pantalla un listado como el siguiente, con el número y el nombre de cada socio.



Vamos a “diseccionar” el código. Las siguientes tres líneas son obligatorias y deberás copiarlas en todas las páginas JSP que accedan a una base de datos. Lo único que tendrás que cambiar es el nombre de la base de datos - la de nuestro ejemplo se llama `baloncesto` - según el caso.

```
Class.forName("com.mysql.jdbc.Driver");
Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/baloncesto","root", "root");
Statement s = conexion.createStatement();
```

La siguiente línea ejecuta la consulta

```
SELECT * FROM socio
```

sobre la base de datos y guarda el resultado en el objeto `listado`. Esta consulta extrae todos los datos (*) de la tabla `socio`. Si quisiéramos obtener esos datos ordenados por nombre, la consulta sería

```
SELECT * FROM socio ORDER BY nombre
```

Aquí está la línea en cuestión:


```
ResultSet listado = s.executeQuery ("SELECT * FROM socio");
```

Una vez extraídos los datos en bruto, hace falta ir sacando cada uno de los registros, es decir, cada una de las líneas. En cada línea se van a mostrar los datos de un socio. Para ello usamos un `while` que va extrayendo líneas mientras quede alguna en el objeto `listado`.

```
while (listado.next()) {  
    out.println(listado.getString("socioID") + " " + listado.getString("nombre") + "<br>");  
}
```

Observa que el método `getString` toma como parámetro el nombre de un campo. En este ejemplo estamos mostrando únicamente los números de socio con sus correspondientes nombres. Si quisiéramos mostrar también la altura de cada socio, la podemos extraer mediante `listado.getString("altura")`.

13.4 Alta

Para dar de alta un nuevo socio, necesitamos recoger los datos mediante un formulario que puede ser un programa JSP o simplemente una página HTML. Estos datos recogidos por el formulario serán enviados a otra página encargada de grabarlos en la tabla `socio` de la base de datos `baloncesto` mediante el comando `INSERT` de SQL.

A continuación se muestra el código de `formularioSocio.jsp` que recoge los datos del nuevo socio. Como puedes comprobar es un simple formulario HTML y no contiene código en Java.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
    </head>  
    <body>  
        <h2>Introduzca los datos del nuevo socio:</h2>  
        <form method="get" action="grabaSocio.jsp">  
            Nº socio <input type="text" name="numero"/><br>  
            Nombre <input type="text" name="nombre"/><br>  
            Estatura <input type="text" name="estatura"/><br>  
            Edad <input type="text" name="edad"/><br>  
            Localidad <input type="text" name="localidad"/><br>  
            <input type="submit" value="Aceptar">  
        </form>  
    </body>  
</html>
```

El siguiente programa - `grabaSocio.jsp` - se encarga de tomar los datos que envía `formularioSocio.jsp` y grabarlos en la base de datos.

Igual que en los programas del capítulo 12, incluimos esta línea para poder recoger correctamente cadenas de caracteres que contienen tildes, la letra ñ, etc.

```
request.setCharacterEncoding("UTF-8");
```

La variable `insercion` es una cadena de caracteres en la que se va componiendo, a base de ir juntando trozos, la sentencia SQL correspondiente a la inserción.

```
String insercion = "INSERT INTO socio VALUES ("
    + Integer.valueOf(request.getParameter("numero"))
    + ", '" + request.getParameter("nombre")
    + "', " + Integer.valueOf(request.getParameter("estatura"))
    + ", " + Integer.valueOf(request.getParameter("edad"))
    + ", '" + request.getParameter("localidad") + "')";
```

Una vez que se han recogido los datos del formulario y se ha creado la cadena, en la variable `insercion` debería quedar algo como esto:

```
INSERT INTO socio VALUES (6789, "Brito Fino, Alan", 180, 40, "Benalmádena")
```

Por último, se ejecuta la sentencia de inserción mediante `s.execute(insercion)`.

A continuación tienes el código completo de `grabaSocio.jsp`:

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <%
            Class.forName("com.mysql.jdbc.Driver");
            Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/baloncest\
o","root", "root");
            Statement s = conexion.createStatement();

            request.setCharacterEncoding("UTF-8");
```

```
String insercion = "INSERT INTO socio VALUES ("
    + Integer.valueOf(request.getParameter("numero"))
    + ", '" + request.getParameter("nombre")
    + "', " + Integer.valueOf(request.getParameter("estatura"))
    + ", " + Integer.valueOf(request.getParameter("edad"))
    + ", '" + request.getParameter("localidad") + "'";

s.execute(insercion);
conexion.close();
%>
Socio dado de alta.
</body>
</html>
```



Al componer una sentencia SQL concatenando trozos, es frecuente que se produzcan errores porque faltan o sobran comillas, paréntesis, comas, etc. Si se produce un error al ejecutar la sentencia, la manera más fácil de detectarlo es mostrar por pantalla la sentencia. Por ejemplo, si la sentencia es una inserción de datos y la cadena de caracteres que la contiene se llama `insercion`, como en el ejemplo que acabamos de ver, simplemente tendríamos que escribir:

```
out.println(insercion).
```

13.5 Borrado

El borrado, al igual que el alta, se realiza en dos pasos. En el primer paso, se carga la página `pideNumeroSocio.jsp` que muestra en una tabla todos los socios, cada uno en una fila. Junto a cada socio se coloca un botón **borrar** que al ser pulsado envía el número de socio a la página `borraSocio.jsp` que se encarga de ejecutar la sentencia SQL de borrado.

La tabla con los socios y los botones de borrado quedaría como se muestra a continuación:

Código	Nombre	Estatura	Edad	Localidad	
1235	Bermúdez Espada, Ana María	186	46	Málaga	<input type="button" value="borrar"/>
1236	Cano Cuenca, Margarita	161	48	Málaga	<input type="button" value="borrar"/>
1237	Doña Enríquez, Adrián Manuel	158	31	Málaga	<input type="button" value="borrar"/>
1238	Fernández Padilla, Esther	183	26	Málaga	<input type="button" value="borrar"/>
1239	Galán Bazán, Ester María	184	52	Málaga	<input type="button" value="borrar"/>
1240	Guzmán Puyol, Estefanía	182	30	Málaga	<input type="button" value="borrar"/>
1241	Martín Jurado, Eva	180	44	Málaga	<input type="button" value="borrar"/>
1242	Moreno Blanco, Carlos	191	17	Campanillas	<input type="button" value="borrar"/>
1243	Narváez Gálvez, Juan Antonio	155	22	Campanillas	<input type="button" value="borrar"/>
1244	Pinto Echeverri, Jhon Diver	167	17	Campanillas	<input type="button" value="borrar"/>

A continuación se muestra el código de `pideNumeroSocio.jsp`. El fichero `estilos.css`² se encuentra disponible en GitHub, igual que el resto de archivos.

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" type="text/css" href="estilos.css" />
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/baloncesto\
o","root", "root");
      Statement s = conexion.createStatement();
```

²https://github.com/LuisJoseSanchez/aprende-java-con-ejercicios/blob/master/ejemplos/13_JSP_y_BBDD/Baloncesto/estilos.css

```

        ResultSet listado = s.executeQuery ("SELECT * FROM socio");
    %>
    <table>
        <tr><th>Código</th><th>Nombre</th><th>Estatura</th><th>Edad</th><th>Localidad</th></tr>
    <%
        while (listado.next()) {
            out.println("<tr><td>");
            out.println(listado.getString("socioID") + "</td>");
            out.println("<td>" + listado.getString("nombre") + "</td>");
            out.println("<td>" + listado.getString("estatura") + "</td>");
            out.println("<td>" + listado.getString("edad") + "</td>");
            out.println("<td>" + listado.getString("localidad") + "</td>");
        }
    %>
    <td>
        <form method="get" action="borraSocio.jsp">
            <input type="hidden" name="codigo" value="<%=listado.getString("socioID") %>" />
            <input type="submit" value="borrar">
        </form>
    </td></tr>
    <%
        } // while
        conexion.close();
    %>
    </table>
</body>
</html>

```

A continuación se muestra el código de `borraSocio.jsp`. Como puedes comprobar, la sentencia SQL que borra el registro deseado es un `DELETE` que toma como referencia el número de socio para realizar el borrado.

```

<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <%
            Class.forName("com.mysql.jdbc.Driver");
            Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/balconcest\



















```

```
o","root", "root");
    Statement s = conexion.createStatement();

    s.execute ("DELETE FROM socio WHERE socioID=" + request.getParameter("codigo"));
    %>
    <script>document.location = "pideNumeroSocio.jsp"</script>
</body>
</html>
```

13.6 CRUD completo con Bootstrap

Los programas que ofrecen la posibilidad de hacer listados y de realizar modificaciones y borrado sobre los registros de una tabla, se denominan CRUD que son las siglas en inglés de *Create*, *Read*, *Update* y *Delete*; en español se diría que es un programa de alta, listado, modificación y borrado. El código fuente completo está disponible en la carpeta *BaloncestoMejorado* de nuestro repositorio en GitHub³.

Club de Baloncesto						
Nº de socio	Nombre	Estatura	Edad	Localidad		
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	+ Añadir	
1235	Bermúdez Espada, Ana María	186	46	Málaga	 Modificar	 Eliminar
1236	Cano Cuenca, Margarita	161	48	Málaga	 Modificar	 Eliminar
1237	Doña Enríquez, Adrián Manuel	158	31	Málaga	 Modificar	 Eliminar
1238	Fernández Padilla, Esther	183	26	Málaga	 Modificar	 Eliminar
1239	Galán Bazán, Ester María	184	52	Málaga	 Modificar	 Eliminar
1241	Martín Jurado, Eva	180	44	Málaga	 Modificar	 Eliminar
1242	Moreno Blanco, Carlos	191	17	Campanillas	 Modificar	 Eliminar
1243	Narváez Gálvez, Juan Antonio	155	22	Campanillas	 Modificar	 Eliminar
1244	Pinto Echeverri, Jhon Diver	167	17	Campanillas	 Modificar	 Eliminar

Para dar de alta un nuevo socio, simplemente habría que rellenar los campos del formulario que aparecen en la primera línea y hacer clic en el botón *Añadir*. La aplicación comprueba si el número de socio introducido ya existe en la base de datos y, en tal caso, muestra un mensaje de error; recuerda que el número de socio es único.

³https://github.com/LuisJoseSanchez/aprende-java-con-ejercicios/tree/master/ejemplos/13_JSP_y_BBDD/BaloncestoMejorado

El borrado se lleva a cabo de la misma manera que en el ejemplo del apartado anterior.

El botón *Modificar* nos lleva a otra página que contiene un formulario con todos los datos del socio, donde podemos modificar la información necesaria. A continuación se muestra una captura.



Modificación de socio

Nº de socio: 1235

Nombre: Bermúdez Espada, Ana María

Estatura (en cm): 186 Edad: 40

Localidad: Málaga

✕ Cancelar ✓ Aceptar

Para los estilos - colores, botones, iconos, etc. - hemos utilizado Bootstrap⁴. Bootstrap es un **framework** que incluye unos estilos predefinidos para que con solo incluirlo en nuestro proyecto, la aplicación tenga una apariencia bonita y homogénea. Puedes encontrar más información de cómo utilizar este **framework** en la página oficial de Bootstrap⁵.

⁴<http://getbootstrap.com/>

⁵<http://getbootstrap.com/>

13.7 Ejercicios





Ejercicio 1


Establece un control de acceso mediante nombre de usuario y contraseña para alguno de los programas de la relación anterior (por ejemplo el que pinta una pirámide). Lo primero que aparecerá por pantalla será un formulario pidiendo el nombre de usuario y la contraseña. Si el usuario y la contraseña son correctos, se podrá acceder al ejercicio; en caso contrario, volverá a aparecer el formulario pidiendo los datos de acceso y no se nos dejará ejecutar la aplicación hasta que iniciemos sesión con un nombre de usuario y contraseña correctos. Los nombres de usuario y contraseñas deben estar almacenados en la tabla de una base de datos.

1

Control de acceso


 Usuario

 Contraseña


ACEPTAR 

2

Control de acceso




Acceso permitido a la aplicación.

ACEPTAR 

3

Pinta una pirámide

Altura

ACEPTAR 

4



Ejercicio 2

Mejora el programa anterior de tal forma que se puedan dar de alta nuevos usuarios para acceder a la aplicación. Si se introduce un nombre de usuario que no sea el administrador (admin) y una contraseña correcta, la aplicación funcionará exactamente igual que el ejercicio anterior. Si se introduce el usuario `admin` y la contraseña correcta, la aplicación entra en la gestión de usuarios donde se podrán dar de alta nuevos usuarios indicando nombre de usuario y contraseña. No puede haber dos nombres de usuario iguales aunque sí puede haber claves repetidas.

1

Control de acceso

Usuario

 admin


Contraseña



ACEPTAR ✓

2

Control de acceso



Tiene acceso al área de gestión de usuarios.

ACEPTAR ✓

3

Gestión de usuarios

Usuario	Contraseña
admin	123456
tux	linux
usuario	usuario
root	toor
usuario2	123

Usuario

Contraseña

AÑADIR USUARIO ✓



Ejercicio 3

Amplía el programa anterior para que se pueda asignar o quitar permiso para ejecutar las aplicaciones de la relación anterior a los distintos usuarios. Por ejemplo, que se pueda especificar que el usuario “jaimito” pueda ejecutar los ejercicios 2, 3 y 5. Para ello, en la base de datos deberá existir una tabla con las parejas (usuario, nº ejercicio). Por ejemplo, si el usuario “jaimito” tiene acceso a los ejercicios 2, 3 y 5; en la tabla correspondiente estarán las parejas (jaimito, 2), (jaimito, 3) y (jaimito, 5). Lo ideal es que la asignación de permisos se haga mediante el marcado de múltiples “checkbox”.

1

Control de acceso

Usuario
admin

Contraseña
.....

ACEPTAR ✓

3


Gestión de usuarios

Usuario	Contraseña	Permisos
admin	123456	EDITAR
tux	linux	EDITAR
usuario	usuario	EDITAR
root	toor	EDITAR
usuario2	123	EDITAR

Usuario _____ Contraseña _____ AÑADIR USUARIO

2

Control de acceso



Tiene acceso al área de gestión de usuarios.

ACEPTAR ✓

4

Permisos

Usuario	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
tux	✓	✓	<input type="checkbox"/>	<input type="checkbox"/>	✓	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

✓



Ejercicio 4

Crea una aplicación web que permita hacer listado, alta, baja y modificación sobre la tabla cliente de la base de datos banco. Un cliente tiene su identificador, nombre completo, dirección, teléfono y fecha de nacimiento.

Gestibank						
Código	Nombre	Dirección	Teléfono	Fecha de nacimiento		
3534534	Cacerolo Tontoñez	Almogía	123456	1963-04-08	EDITAR	BORRAR
456478	Hernán Sánchez	Calle Finlandia, 11	678098867	1972-05-24	EDITAR	BORRAR
45678	Mota	Calle Falsa, 123	555 444333	1980-06-28	EDITAR	BORRAR
555	Luis José	Larios, 10	5555 234233	2013-02-17	EDITAR	BORRAR
65767	Pepito Lupiañez	Alhaurín	867867867	1992-01-10	EDITAR	BORRAR
76859	ignacio	Periquito, 333	555 325476	1995-08-08	EDITAR	BORRAR
789654	Yren	Calle Verdadera, 98	555 98765	1950-06-06	EDITAR	BORRAR
873475933	Maria Sol	Calle Flor	555 123456	1945-01-01	EDITAR	BORRAR
código	nombre	dirección	teléfono	fecha de nacim.	AÑADIR	



Ejercicio 5

Amplía el programa anterior para que se pueda hacer una búsqueda por nombre. El programa buscará la cadena introducida dentro del campo “nombre” y, si hay varias coincidencias, se mostrará una lista de clientes para poder seleccionar uno de ellos y ver todos los datos. Si solo hay una coincidencia, se mostrarán directamente los datos del cliente en cuestión.

1

Gestibank

Código	Nombre	Dirección	Teléfono	Fecha de nacimiento		
3534534	Cacerolo Tontoñez	Almogía	123456	1963-04-08	EDITAR	BORRAR
456478	Hernán Sánchez	Calle Finlandia, 11	678098867	1972-05-24	EDITAR	BORRAR
45678	Mota	Calle Falsa, 123	555 444333	1980-06-28	EDITAR	BORRAR
555	Luis José	Larios, 10	5555 234233	2013-02-17	EDITAR	BORRAR
65767	Pepito Lupiañez	Alhaurín	867867867	1992-01-10	EDITAR	BORRAR
789654	Yren	Calle Verdadera, 98	555 98765	1950-06-06	EDITAR	BORRAR
873475933	Maria Sol	Calle Flor	555 123456	1945-01-01	EDITAR	BORRAR

código

nombre

dirección

teléfono

fecha de nacim.

ANADIR

nombre

ez

BUSCAR

2

Gestibank

Nombre

Cacerolo Tontoñez

DETALLE

Hernán Sánchez

DETALLE

Pepito Lupiañez

DETALLE

3

Hernán Sánchez

Código: 456478

Nombre: Hernán Sánchez

Dirección: Calle Finlandia, 11

Teléfono: 678098867

Fecha de nacimiento: 1972-05-24

ACEPTAR



Ejercicio 6

Crea una versión web del programa GESTISIMAL (GESTIÓN SIMPLificada de Almacén) para llevar el control de los artículos de un almacén. De cada artículo se debe saber el código, la descripción, el precio de compra, el precio de venta y el stock (número de unidades). La entrada y salida de mercancía supone respectivamente el incremento y decremento de stock de un determinado artículo. Hay que controlar que no se pueda sacar más mercancía de la que hay en el almacén. Aprovecha las opciones que puede ofrecer una interfaz web, por ejemplo para eliminar un artículo ya no será necesario pedir el código sino que se puede mostrar un listado de todos los artículos de tal forma que se puedan borrar un artículo directamente pulsando un botón.

G e s t i s i m a l					
Código	Descripción	Precio de compra	Precio de venta	Stock	
2324	MERMELADA DE ALBARICOQUE	1.9	2.3	40	  
34240	TURRÓN BLANDO ALMENDRA	1.75	2.45	250	  
36548	PAN DE MOLDE	0.95	1.8	70	  
4444	CHOCOLATE 150GR	1.9	3.3	5	  
45654	PAN DE MOLDE	1.78	2.14	23	  
6767676	TOFU CON SÉSAMO	1.6	2.85	40	  
67904	CROQUETAS DE SETAS	2.95	3.5	91	  
89567	PATATAS FRITAS	0.37	0.67	90	  
<input type="text" value="código"/>	<input type="text" value="descripción"/>	<input type="text" value="precio de compra"/>	<input type="text" value="precio de venta"/>	<input type="text" value="stock"/>	