AUT03_02. Actividad BaseDatos ClientesConsumos

1. Objetivos

Los objetivos de esta actividad son:

- Repasar métodos del ciclo de vida de Servlet
- Repasar formularios y peticiones de tipo GET/POST
- Conocer los fundamentos de JDBC

2. Crear aplicación ConsumoEléctrico.

Se creará una aplicación Maven de tipo Web usando Java Enterprise Edition 7 y Glassfish 4.1.x.

2.1 Crear formulario de búsqueda HTML

Será una página **index.html** con un formulario de búsqueda. En este caso con método **GET**.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Buscador</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <h1>Buscador de clientes</h1>
        <form action="/ConsumoElectrico/buscadorClientes" method="GET">

            <fieldset>
                <legend>Datos cliente</legend>
                <label for="txtNombre">Nombre</label>
                <input id="txtNombre" type="text" name="nameToSearch"/>
                <input type="submit" value="Enviar"/>
```

```
            </fieldset>

        </form>

    </body>

</html>
```

2.2 Fichero .properties de configuración de la conexión a base de datos

Se creará un fichero **consumoelectrico.properties** que se ubicará en la carpeta raíz de "source packages". Su contenido será el siguiente:

```
database.driver=com.mysql.jdbc.Driver

database.url=jdbc:mysql://localhost:3306/consumoelectrico?profileSQL=true

database.user=2dawa

database.password=2dawa

database.pageSize=10
```

- database.driver. Nombre cualificado de la clase Java del conector a la base de datos.
- database.url. Cadena de conexión a la base de datos.
- database.user. Usuario de la base de datos.
- database.password. Contraseña del usuario de la base de datos.
- database.pageSize. Número máximo de registros por consulta.

2.3 Web component de tipo servlet que realizará las búsquedas de clientes

Se creará un servlet, mapeado en la URL **/buscadorclientes**, cuyo nombre de clase plenamente cualificado será: **es.cifpcm.consumoelectrico.buscador.web.BuscadorConsumoElectricoServlet**.

Este servlet, definirá un parámetro de inicialización que será usado en la conexión a la base de datos.

i. app.config → consumoelectrico

Esto quiere decir que tendremos un fichero consumoelectrico.properties con la configuración de la aplicación.

El servlet, en el método init, leerá las propiedades del fichero de configuración, usando un ResourceBundle. En el método init cargará la clase del driver, mediante Class.forName utilizando la propiedad driver.classname del fichero properties referido.

Buscando información sobre el ResourceBundle encontrarán que busca por defecto en el paquete `resources` así que colócalo ahí. Además el fichero app.config vendrá definido en web.xml. Ya que el método ConfigServlet busca sus parámetros de inicialización allí. **Importante hazlo como especifico, de otra forma quitará puntos.**

Ejemplo:

```
@Override
public void init(ServletConfig config) throws ServletException {
    super.init(config);
    try {
        String configBundleName = config.getInitParameter("app.config");
        ResourceBundle rb = ResourceBundle.getBundle(configBundleName);
        this.dbUrl = rb.getString("database.url");
        this.dbUser = rb.getString("database.user");
        this.dbPassword = rb.getString("database.password");
        this.dbPageSize = rb.getString("database.pageSize") == null
                                                    ?   DEFAULT_PAGESIZE    :
Integer.parseInt(rb.getString("database.pageSize"));
        String driverClassName = rb.getString("database.driver");

        Class.forName(driverClassName);
    } catch (ClassNotFoundException ex) {
         Logger.getLogger(BuscadorConsumoElectricoServlet.class.getName()).log(Level.S
EVERE, null, ex);
    }
}
```

2.4 Clase Java de tipo (POJO) para almacenar datos de clientes

POJO es un acrónimo que significa **P**lain **O**ld **J**ava **O**bject. Hace referencia a una clase Java muy simple, que únicamente contiene datos, en forma de propiedades de instancia y simplemente métodos get/set para acceder o modificar las propiedades.

```
public class Cliente {
    private Integer clNo;
    private String firstName;
    private String lastName;
```

```
    public Cliente() {

    }

    public Integer getClNo() {

        return clNo;

    }

    public void setCLNo(Integer clNo) {

        this.clNo = clNo;

    }
```

2.5 Código del servlet

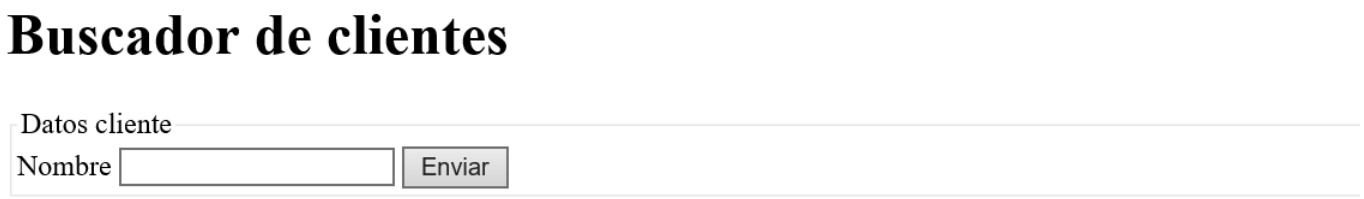
El servlet realizará lo siguiente:

1. El servlet en este primer paso recogerá el valor del campo nameToSearch, aunque no lo utilizará por ahora en la consulta SQL.
2. Obtendrá una conexión del DriverManager, utilizando la cadena de conexión JDBC, así como el nombre de usuario y contraseña de consumoelectrico.properties.
3. A partir de la conexión creará un objeto Statement.
4. A través del método setMaxRows del objeto Statement se establecerá el número máximo de registros devueltos por consulta.
5. Se ejecutará el statement con la siguiente consulta SQL: "SELECT * FROM consumoelectrico", asignándose el resultado a un objeto de tipo ResultSet.
6. Se recorrerá el resultset obteniendo los datos de cada cliente y almacenando los clientes en una colección de tipo ArrayList.
7. Se cerrarán objetos de base de datos.
8. Se devolverá un HTML de respuesta, mostrando en una tabla los datos de los clientes.

   NOTA: Se utilizará una hoja de estilos CSS.

2.6 Ejemplo de petición y respuesta

2.6.1 Página de búsqueda



Nota: Recuerda que en primer lugar sacará por pantalla los registros de 10 clientes. Y luego sacará para el cliente dado, la lista de las ids de sus mediciones, las fechas y su consumo total, además de un link al consumo particular de cada medición.

3. Puntuación.

Mapeado del servlet	3 puntos
Clase POJO cliente	1 punto
Conexión a base de datos	1 puntos
Sacar datos de clientes por pantalla	2 puntos
Sacar datos de cliente buscado	2 puntos
Carga de fichero propiedades Bundle	0,5
Fichero de estilo aparte	0,5