CRUD SPRING DE JDBC A PERSISTENCIA

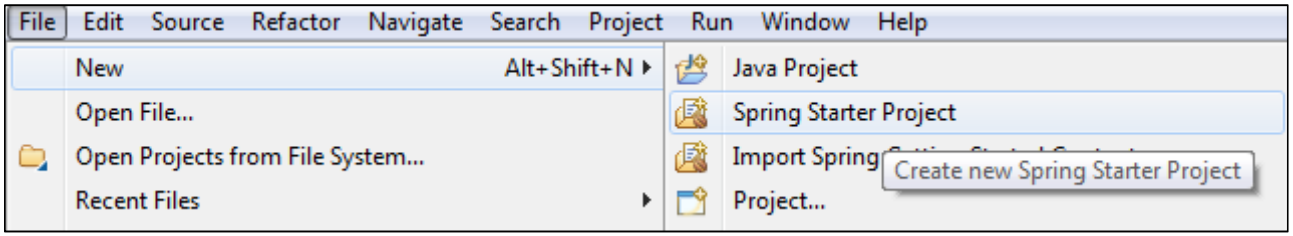
Eduard Lara

INDICE

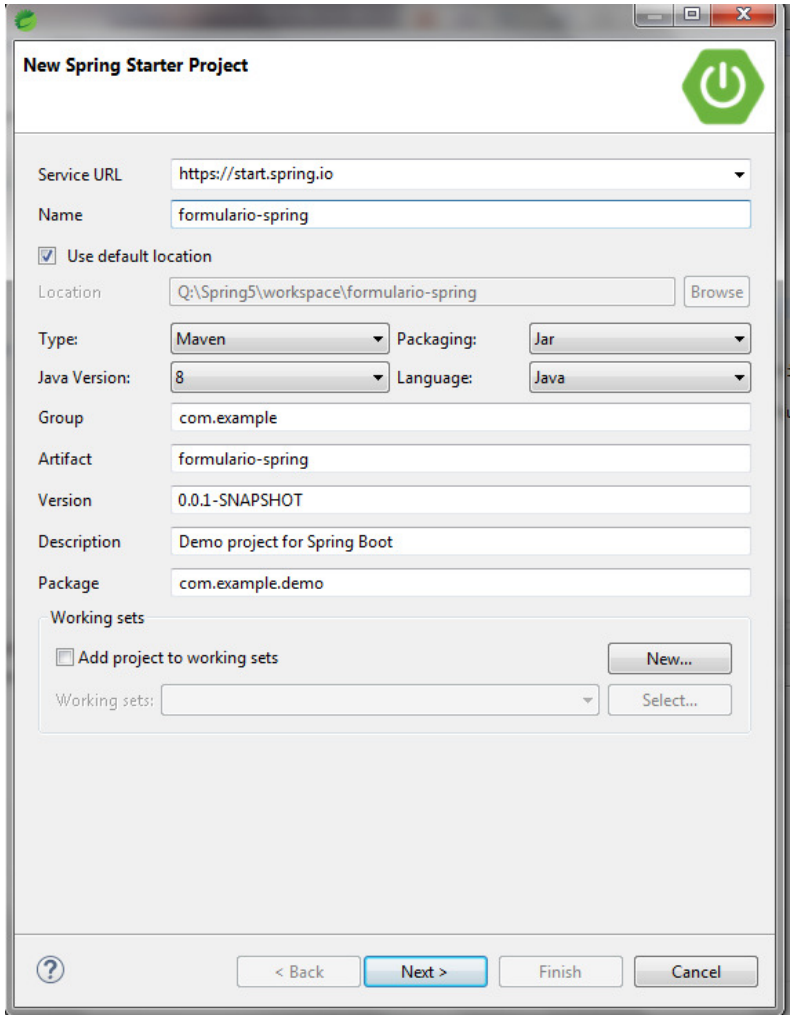
1. Construccion CRUD
2. Backend con ArrayList
3. Backend con JDBC
4. Backend con JPA Persistencia

1. CONSTRUCCION CRUD

Paso 1) Creamos un proyecto Spring Boot, en la opción de menu File/New/Spring Starter Project:



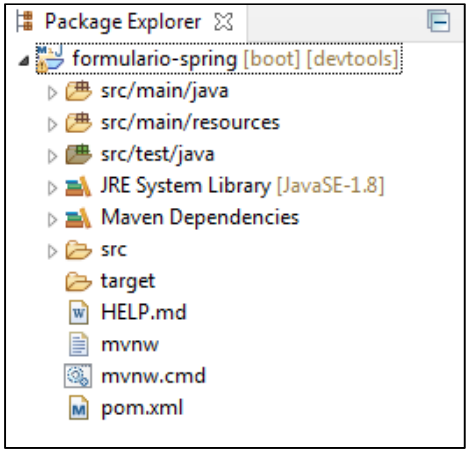
Podemos dejar por defecto los valores que nos presenta el wizard. Si se desea se puede cambiar el nombre de proyecto, el package raíz, el tipo de proyecto (Maven o Gradle) y/o la versión de Java.

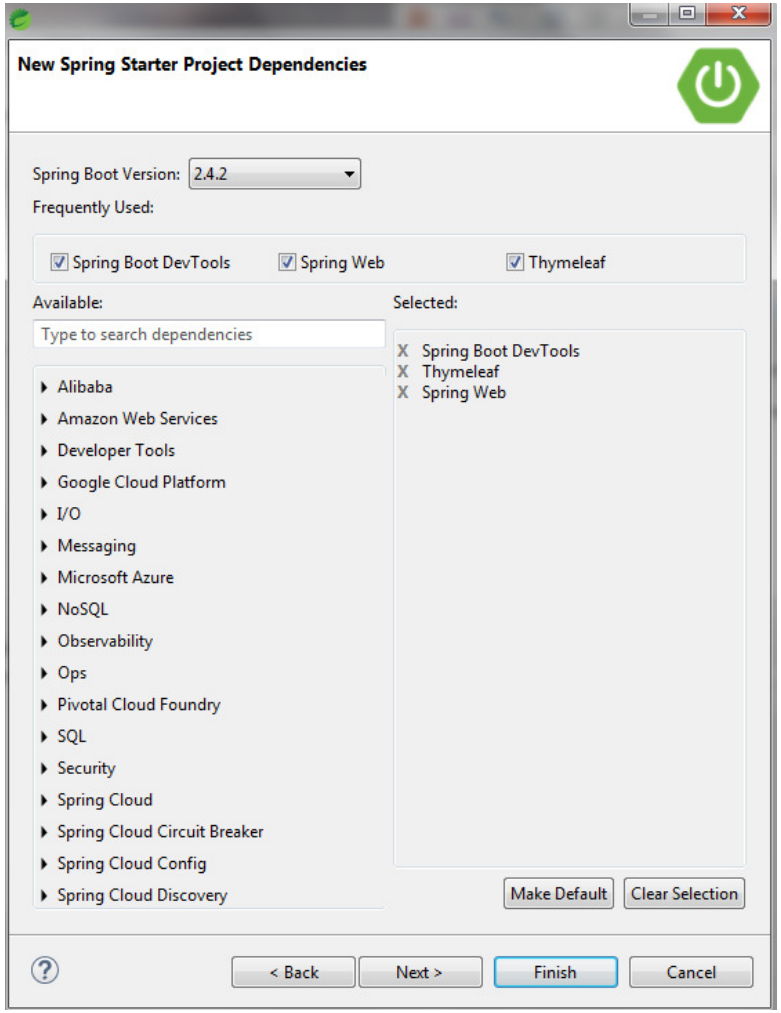


1. CONSTRUCCION CRUD

Paso 2) Agregamos las siguientes dependencias:

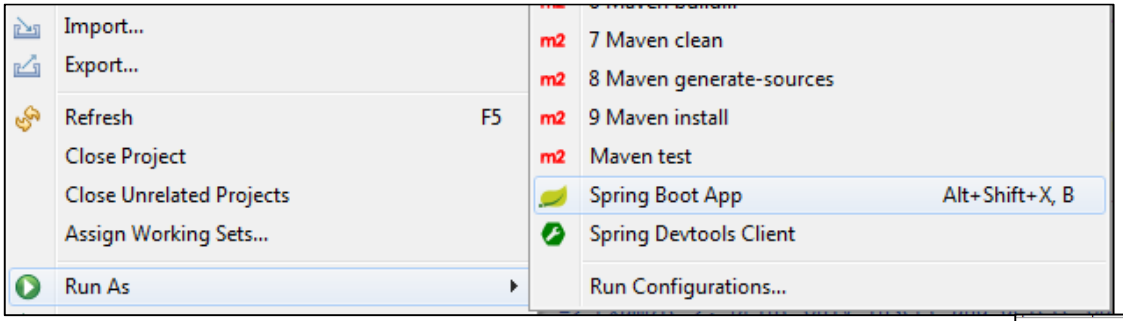
- Spring Web (necesaria)
- Spring Boot Dev Tools (muy importante ya que cualquier cambio que hagamos en nuestro código java, de forma automática se va a actualizar en el despliegue sin tener que reiniciar el servidor)
- Thymeleaf, para hacer uso de estas plantillas, que hacen el papel de las típicas jsp

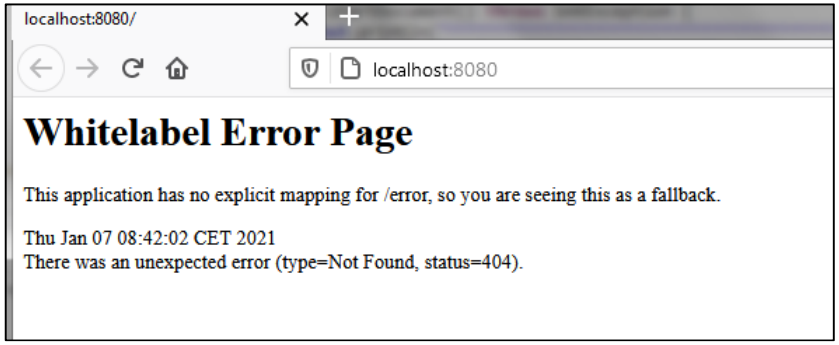


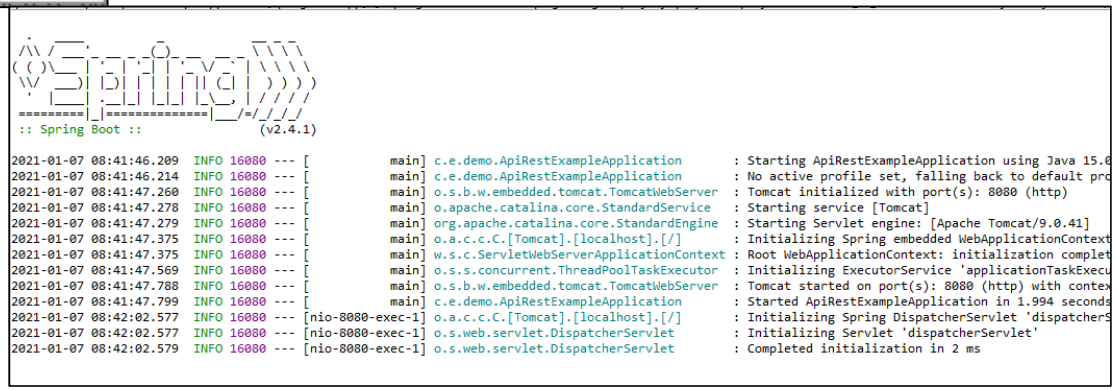


1. CONSTRUCCION CRUD

Paso 3) Probamos de ejecutar el proyecto, para ello levantamos el servidor Tomcat haciendo Run As/Spring Boot App. Una vez vemos que ha arrancado correctamente el servidor, vamos a un navegador y ponemos **localhost:8080**. Nos da error porque no tenemos ninguna página de inicio. Pero también significa que ya hay un servidor respondiendo en el puerto 8080.

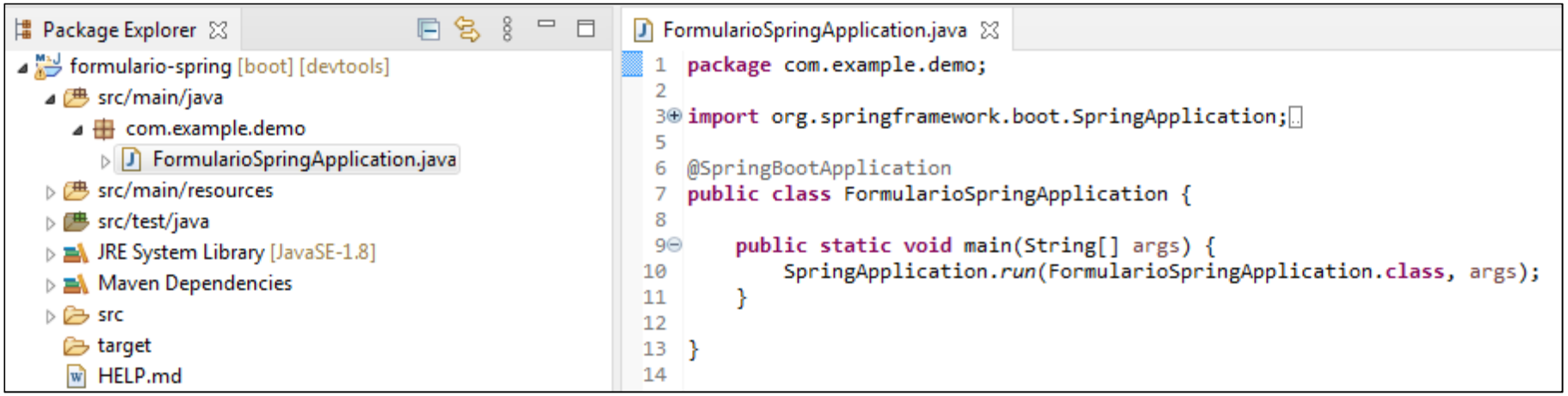






1. CONSTRUCCION CRUD

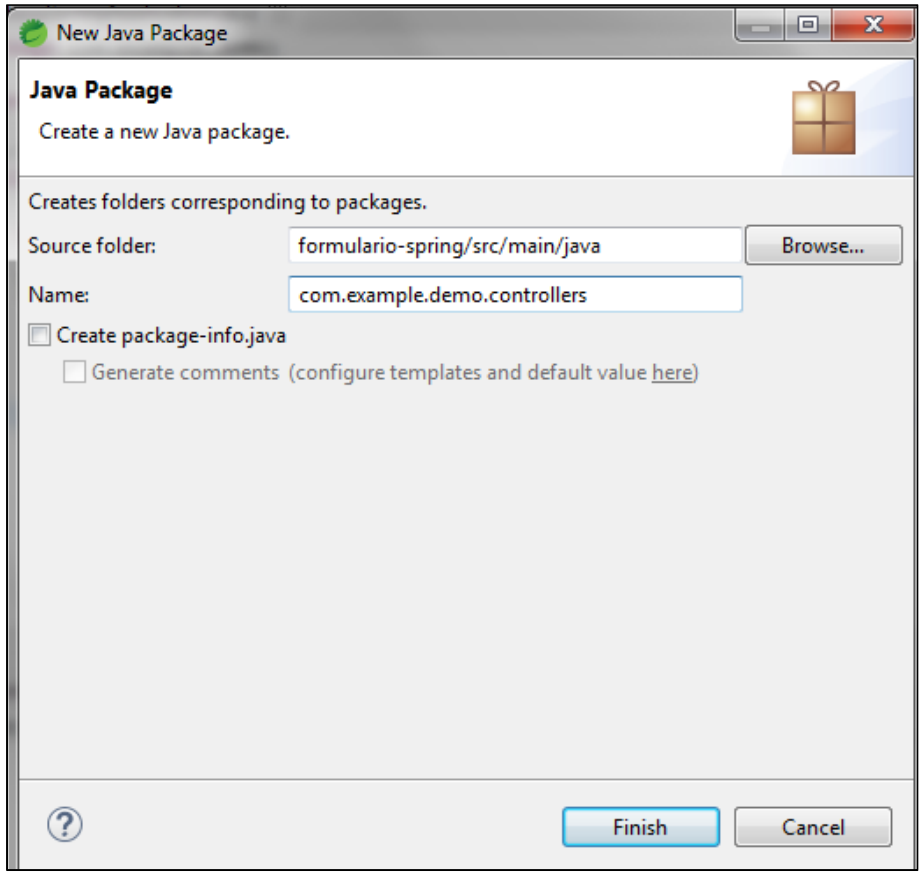
Paso 4) Podemos observar en el package raíz indicado al principio en la creación del proyecto, la clase generada automáticamente que inicia nuestro servidor y la aplicación:

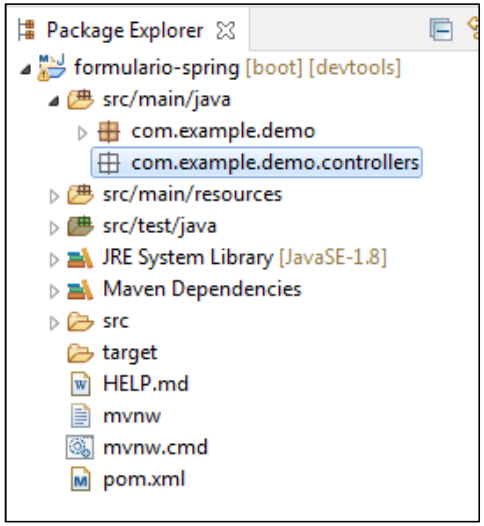


1. CONSTRUCCION CRUD

Controlador

Paso 5) Generamos un package dentro del existente con la extensión controllers:

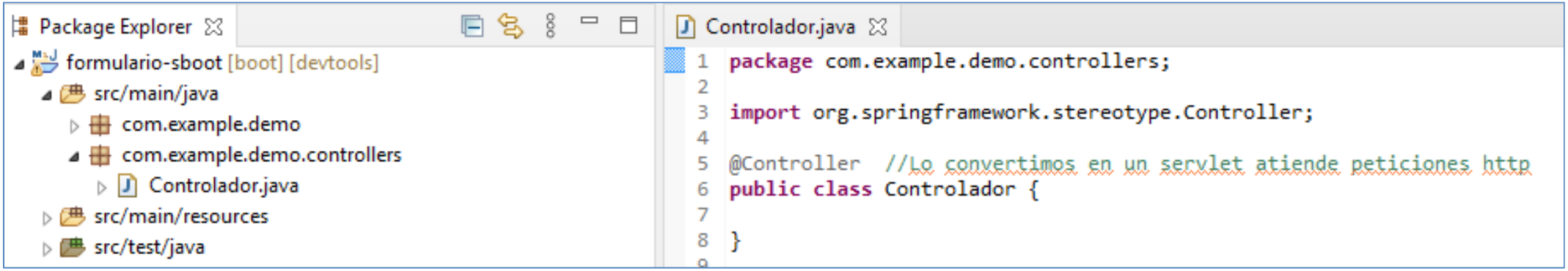




1. CONSTRUCCION CRUD

Controlador

Paso 6) Dentro de este package creamos una clase a la que le pondremos la etiqueta de controlador @Controller.



1. CONSTRUCCION CRUD

Controlador

Paso 7) Creamos dos métodos handler:

- iniciar → el cual nos dará entrada al formulario de login. Será accesible desde localhost:8080 con método Get.
- login → una vez logados correctamente nos dara entrada al listado de libros

Controlador.java

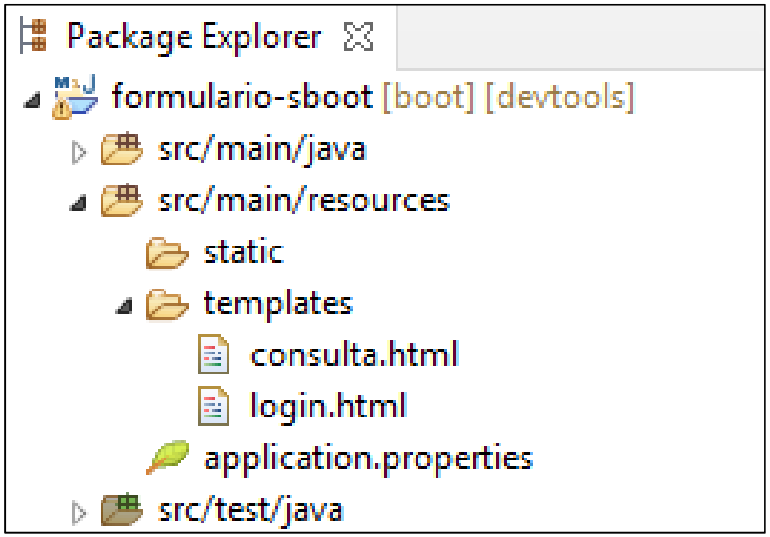
```
1  package com.example.demo.controllers;
2
3  import org.springframework.stereotype.Controller;
7
8  @Controller  //Lo convertimos en un servlet atiende peticiones http
9  @RequestMapping("")
10 public class Controlador {
11
12     @GetMapping("/")
13     public String iniciar() {
14         return "login";
15     }
16
17     @PostMapping("/")
18     public String login() {
19         return "consulta";
20     }
21 }
22
```

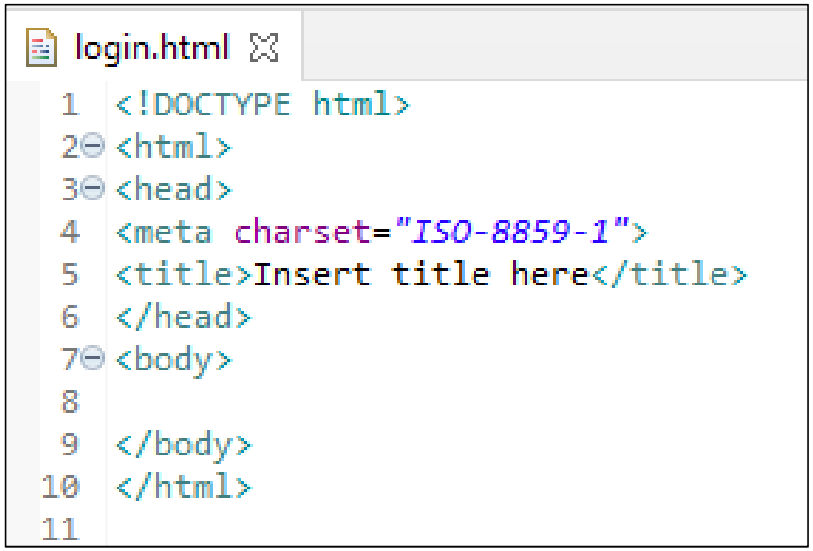
1. CONSTRUCCION CRUD

Vistas

Paso 8) En resources/templates vamos a crear dos plantillas:

- login.html → que mostrará el formulario de acceso.
- consulta.html → que mostrará el listado de libros y permitirá realizar un mantenimiento sobre esos datos





1. CONSTRUCCION CRUD

Vistas

Paso 9) En login.html crea un formulario de acceso a la aplicación. Su action será "/", es decir localhost:8080/, y su método http de envío post:

login.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>FORMULARIO DE ACCESO</title>
6  </head>
7  <body>
8  <H1>FORMULARIO DE ACCESO</H1>
9      <form action="/" method="post">
10         Login: <input type=text name=nombre><br><br>
11         Password: <input type="password" name=password><br><br>
12         <input type=submit name=submit value="Login">
13     </form>
14 </body>
15 </html>
16
17
```

1. CONSTRUCCION CRUD

Vistas

Paso 10) En consulta.html, mostraremos un inicial listado de libros (ya lo complementaremos posteriormente):

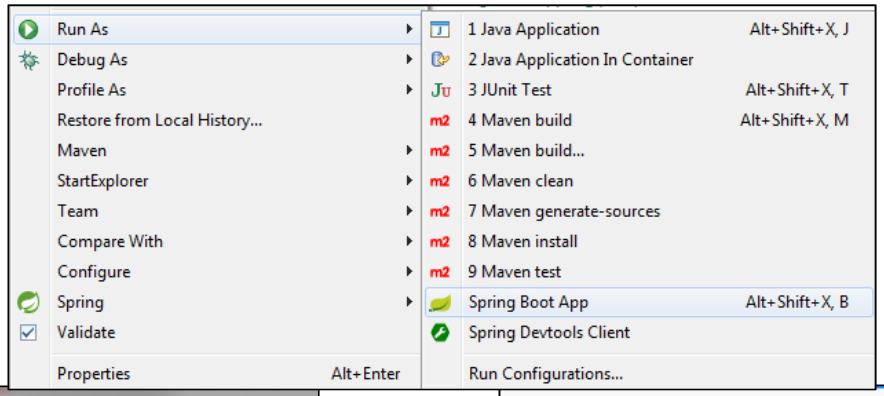
consulta.html

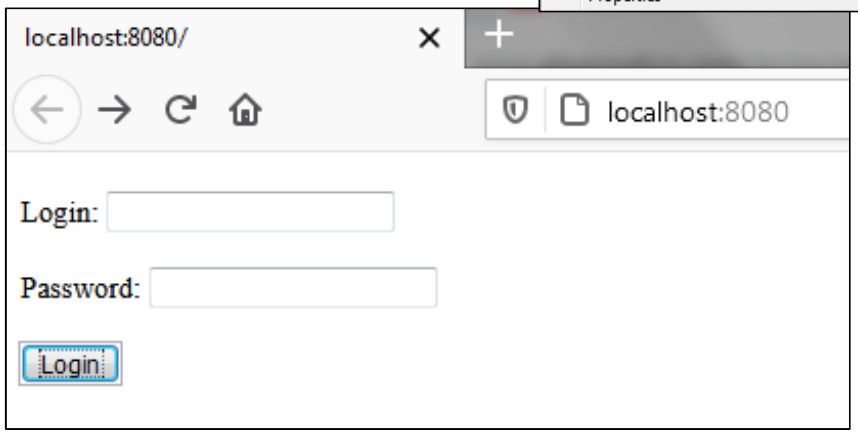
```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3  <head>
4      <meta charset="ISO-8859-1">
5      <title>LISTADO DE LIBROS</title>
6  </head>
7  <body>
8
9  <H1>LISTADO DE LIBROS</H1>
10
11 </body>
12 </html>
13
```

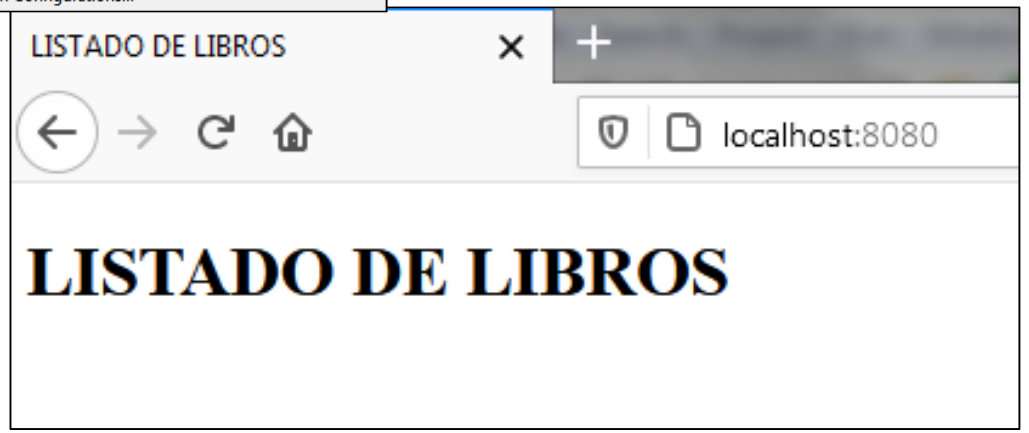
1. CONSTRUCCION CRUD

Vistas

Paso 11) En este punto podemos probar el funcionamiento de nuestra aplicación. Hacemos sobre el proyecto click botón derecho/Run As/Spring Boot App.







1. CONSTRUCCION CRUD

Vistas

Paso 12) Podemos convertir login.html en una plantilla thymeleaf para pasarle el parámetro titulo desde el handler iniciar del controlador:

login.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title th:text="${titulo}"> /title>
</head>
<body>
<H1 th:text="${titulo}"></H1>
    <form action="/" method="post">
        Login: <input type=text name=nombre><br><br>
        Password: <input type="password" name=password><br><br>
        <input type=submit name=submit value="Login">
    </form>
</body>
</html>
```

Controlador.java

```
package com.example.demo.controllers;

import org.springframework.stereotype.Controller;

@Controller  //Lo convertimos en un servlet atiende peticiones http
@RequestMapping("")
public class Controlador {

    @GetMapping("/")
    public String iniciar(Model model) {
        model.addAttribute("titulo","FORMULARIO DE ACCESO");
        return "login";
    }

    @PostMapping("/")
    public String login() {
        return "consulta";
    }
}
```

1. CONSTRUCCION CRUD

Vistas

Paso 13) El método handler “login” del controlador, puede recibir los datos enviados por el formulario de acceso uno a uno o empaquetados en una clase.

Método uno a uno

Debemos de insertar tantos parámetros con la etiqueta @RequestParam, como elementos html (básicamente input type) hayamos definido en el formulario.

Es muy importante que coincida el valor del atributo name del input type con el nombre de la variable usada para recuperar su valor en el controlador.

```
<form action="/" method="post">
    Login: <input type=text name=nombre><br><br>
    Password: <input type="password" name=password><br><br>
    <input type=submit name=submit value="Login">
</form>
```

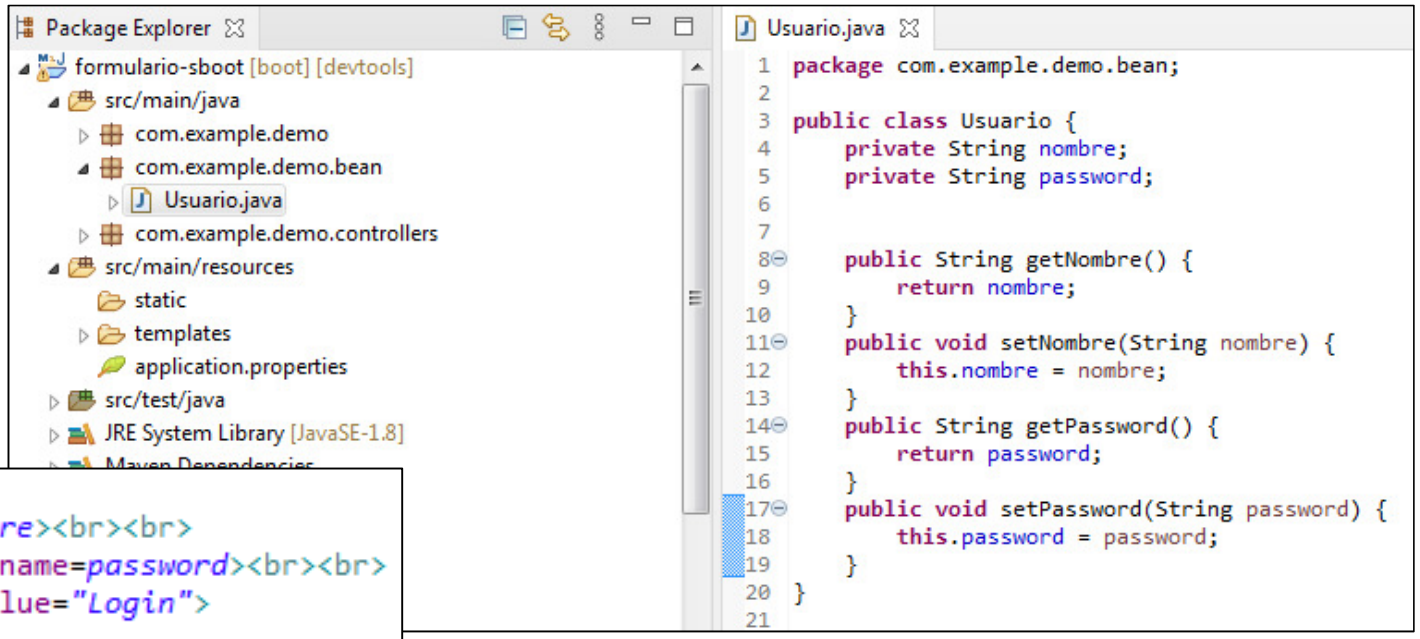
Controlador.java

```
3  import org.springframework.stereotype.Controller;
9
10 @Controller  //Lo convertimos en un servlet atiende peticiones http
11 @RequestMapping("")
12 public class Controlador {
13
14     @GetMapping("/")
15     public String iniciar(Model model) {
16         model.addAttribute("titulo","FORMULARIO DE ACCESO");
17         return "login";
18     }
19
20     @PostMapping("/")
21     public String login(Model model,
22                         @RequestParam String nombre,
23                         @RequestParam String password) {
24         if (nombre.equals("edu") && password.equals("edu"))
25             return "consulta";
26         else
27             return "login";
28     }
29 }
30
```

1. CONSTRUCCION CRUD

Vistas

Paso 14) Para el método de todos los parámetros **empaquetados en una clase**, debemos definir una clase llamada Usuario, cuyos atributos deben de coincidir con el atributo name de los input text del formulario. La crearemos dentro de un package de nombre bean.




```
<form action="/" method="post">
    Login: <input type=text name=nombre><br><br>
    Password: <input type="password" name=password><br><br>
    <input type=submit name=submit value="Login">
</form>
```

1. CONSTRUCCION CRUD

Vistas

Paso 15) El método handler tendrá un parámetro que será un objeto de la clase Usuario. El acceso a las variables del formulario se hará mediante los getters de la clase:

Controlador.java

```
1  package com.example.demo.controllers;
2
3  import org.springframework.stereotype.Controller;
11
12 @Controller  //Lo convertimos en un servlet atiende peticiones http
13 @RequestMapping("")
14 public class Controlador {
15
16     @GetMapping("/")
17     public String iniciar(Model model) {
18         model.addAttribute("titulo","FORMULARIO DE ACCESO");
19         return "login";
20     }
21
22     @PostMapping("/")
23     public String login(Usuario usuario, Model model) {
24         if (usuario.getNombre().equals("edu") && usuario.getPassword().equals("edu"))
25             return "consulta";
26         else
27             return "login";
28     }
29 }
30
```

1. CONSTRUCCION CRUD

Vistas

Paso 16) Como complemento, una vez logados satisfactoriamente, se pueden pasar los datos del usuario a la consulta

consulta.html

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3  <head>
4      <meta charset="ISO-8859-1">
5      <title>LISTADO DE LIBROS</title>
6  </head>
7  <body>
8   <h3>Usuario: <span th:text="${usuario.nombre}"> </span></h3>
9   <h3>Password: <span th:text="${usuario.password}"> </span></h3>
10
11 <H1>LISTADO DE LIBROS</H1>
12
13 </body>
14 </html>
15
```

Controlador.java

```
1  package com.example.demo.controllers;
2
3  import org.springframework.stereotype.Controller;
11
12 @Controller  //Lo convertimos en un servlet atiende peticiones http
13 @RequestMapping("")
14 public class Controlador {
15
16     @GetMapping("/")
17     public String iniciar(Model model) {
18         model.addAttribute("titulo","FORMULARIO DE ACCESO");
19         return "login";
20     }
21
22     @PostMapping("/")
23     public String login(Usuario usuario, Model model) {
24         if (usuario.getNombre().equals("edu") && usuario.getPassword().equals("edu")) {
25             model.addAttribute("usuario",usuario);
26             return "consulta";
27         }else
28             return "login";
29     }
30 }
31
```

2. BACKEND CON ARRAYLIST

Paso 1) Primero crearemos la clase Libro dentro del package bean, que utilizaremos como clase Entity para la transferencia de información entre vistas y controlador:

Libro.java

```java
package com.example.demo.bean;

public class Libro {
    private int id;
    private String titulo;
    private String autor;
    private String editorial;
    private String fecha;
    private String tematica;

    public Libro(int id, String titulo, String autor, String editorial, String fecha, String tematica) {
        this.id = id;
        this.titulo = titulo;
        this.autor = autor;
        this.editorial = editorial;
        this.fecha = fecha;
        this.tematica = tematica;
    }

}
```

2. BACKEND CON ARRAYLIST

Paso 2) Creamos la clase BaseDatos(), la cual en su constructor carga el ArrayList que utilizaremos como backend:

BaseDatos.java

```
5
6  public class BaseDatos {
7
8      ArrayList<Libro> libros = new ArrayList<Libro>();
9      public BaseDatos() {
10         libros.add(new Libro(1,"HARRY POTTER Y EL PRISIONERO DE AZKABAN","J.K ROWINS","SALAMANDRA","26/9/2006 0:00:00","INFANTIL"));
11         libros.add(new Libro(2,"EL GRAN LABERINTO","FERNANDO SABATER PEREZ","ARIEL","26/9/2006 0:00:00","FICCION"));
12         libros.add(new Libro(3,"ROMEO Y JULIETA","WILLIAM SHAKESPEARE","SALAMANDRA","26/9/2006 0:00:00","ROMANTICA"));
13         libros.add(new Libro(4,"LA CARTA ESFERICA","ARTURO PEREZ LOPEZ","SALAMANDRA","29/9/2006 0:00:00","FICCION"));
14         libros.add(new Libro(5,"CODIGO DA VINCI","DAN BROWN","ARIEL","29/9/2006 0:00:00","FICCION"));
15         libros.add(new Libro(6,"MUCHO RUIDO Y POCAS NUECES","WILLIAM SHAKESPEARE","SALAMANDRA","29/9/2006 0:00:00","ROMANTICA"));
16         libros.add(new Libro(7,"PROTOCOLO","JOSE LOPEZ MURILLO","SALAMANDRA","6/9/2006 0:00:00","SOCIAL"));
17         libros.add(new Libro(8,"LINUX","FERNANDO SABATER PEREZ","ARIEL","6/9/2006 0:00:00","INFORMATICA"));
18         libros.add(new Libro(9,"EL TUMULTO","H.P LOVERCRAFT","DEBATE","6/9/2006 0:00:00","CIENCIA"));
19         libros.add(new Libro(10,"PERSONAJES MITICOS","RICHARD HOLLIGHAM","DEBATE","7/9/2006 0:00:00","ENTRETENIMIENTO"));
20         libros.add(new Libro(11,"EL TIEMPO","J.K ROWINS","SALAMANDRA","7/9/2006 0:00:00","CIENCIA"));
21         libros.add(new Libro(12,"DIETAS MEDITERRANEAS","ARTURO PEREZ LOPEZ","ARIEL","16/9/2006 0:00:00","ASTRONOMIA"));
22         libros.add(new Libro(13,"ANGELES Y DEMONIOS","DAN BROWN","ARIEL","17/9/2006 0:00:00","FICCION"));
23         libros.add(new Libro(14,"FORTALEZA DIGITAL","DAN BROWN","ARIEL","6/10/2006 0:00:00","FICCION"));
24         libros.add(new Libro(15,"CAPITAN ALATRISTE","ARTURO PEREZ LOPEZ","ALFAGUARA","9/10/2006 0:00:00","FICCION"));
25         libros.add(new Libro(16,"PIEL DE TAMBOR","ARTURO PEREZ LOPEZ","ALFAGUARA","16/10/2006 0:00:00","FICCION"));
26         libros.add(new Libro(17,"TIEMPOS DE COLERA","GABRIEL GARCIA GARCIA","OVEJA NEGRA","1/9/2006 0:00:00","OCIO"));
27         libros.add(new Libro(18,"NOTICIA DE UN SECUESTRO","GABRIEL GARCIA GARCIA","ALFAGUARA","7/12/2006 0:00:00","FICCION"));
28     }
29     public ArrayList<Libro> getLibros() {
30         return libros;
31     }
32     public void setLibros(ArrayList<Libro> libros) {
33         this.libros = libros;
34     }
```

2. BACKEND CON ARRAYLIST

Paso 3) Dentro del controlador creamos una instancia de la clase BaseDatos. La utilizaremos dentro del método login para obtener el ArrayList de Libros, el cual pasaremos a la vista consulta:

Controlador.java

```
16 @Controller  //Lo convertimos en un servlet atiende peticiones http
17 @RequestMapping("")
18 public class Controlador {
19
20     BaseDatos bd = new BaseDatos();
21
22     @GetMapping("/")
23     public String iniciar(Model model) {
24         model.addAttribute("titulo","FORMULARIO DE ACCESO");
25         return "login";
26     }
27
28     @PostMapping("/")
29     public String login(Usuario usuario, Model model) {
30         if (usuario.getNombre().equals("edu") && usuario.getPassword().equals("edu")) {
31             ArrayList<Libro> libros = bd.getLibros();
32             model.addAttribute("usuario",usuario);
33             model.addAttribute("libros",libros);
34             return "consulta";
35         }else
36             return "login";
37     }
38
39 }
```

2. BACKEND CON ARRAYLIST

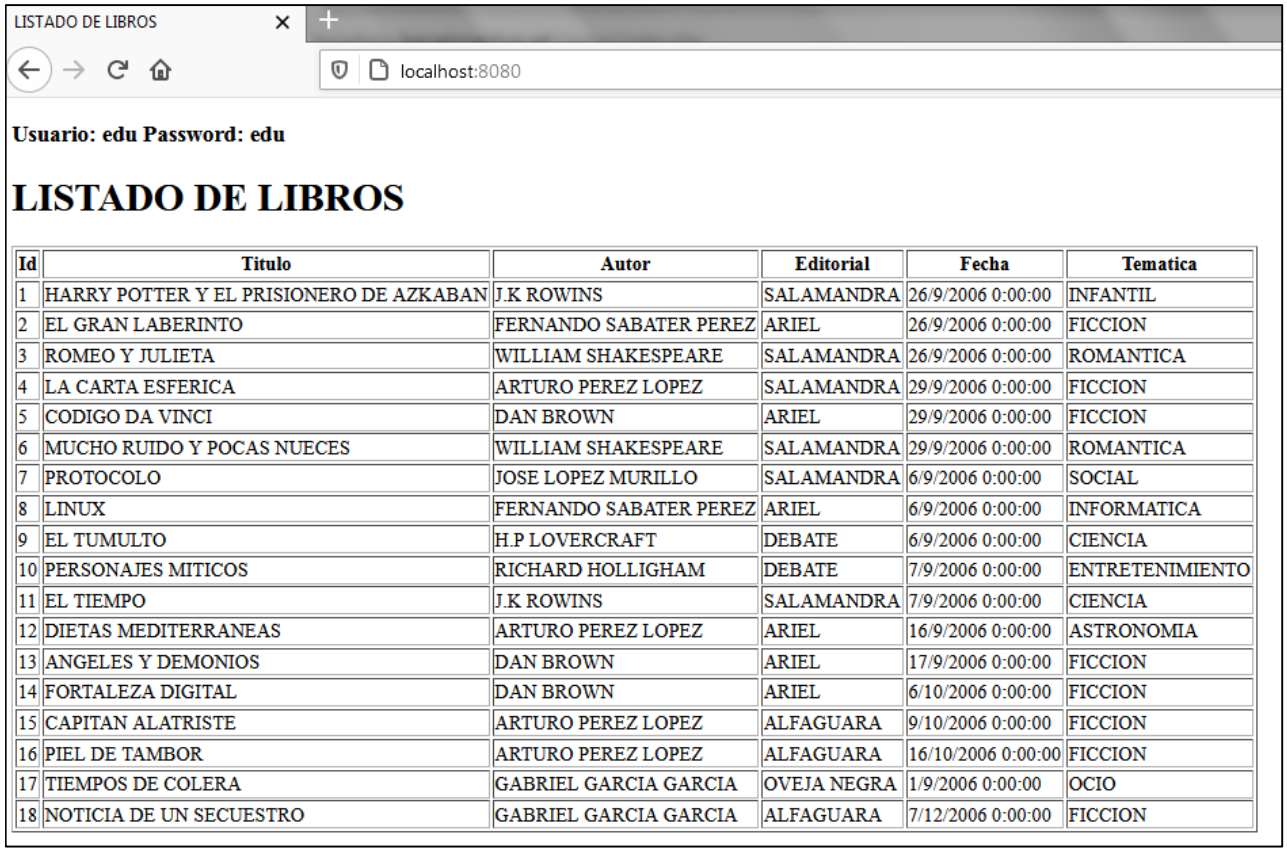
Paso 4) Modificamos consulta para que reciba la lista de libros y la muestre formateada en una tabla html:

consulta.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="ISO-8859-1">
    <title>LISTADO DE LIBROS</title>
</head>
<body>
 <h3>Usuario: <span th:text="${usuario.nombre}"> </span> - Password: <span th:text="${usuario.password}"> </span></h3>

<H1>LISTADO DE LIBROS</H1>
    <table border=1>
        <thead>
            <tr><th> Id <th> Titulo <th> Autor <th> Editorial <th> Fecha <th> Tematica
            </tr>
        </thead>
        <tbody>
        <tr th:if="${libros.empty}">
                <td colspan="2"> No Books Available </td>
            </tr>
            <tr th:each="libro : ${libros}">
                <td><span th:text="${libro.id}"></span></td>
                <td><span th:text="${libro.titulo}"></span></td>
                <td><span th:text="${libro.autor}"></span></td>
                <td><span th:text="${libro.editorial}"></span></td>
                <td><span th:text="${libro.fecha}"></span></td>
                <td><span th:text="${libro.tematica}"></span></td>
            </tr>
        </tbody>
    </table>
```

2. BACKEND CON ARRAYLIST

Paso 5) Podemos observar como queda el resultado:



Usuario: edu Password: edu

LISTADO DE LIBROS

Id	Titulo	Autor	Editorial	Fecha	Tematica
1	HARRY POTTER Y EL PRISIONERO DE AZKABAN	J.K ROWINS	SALAMANDRA	26/9/2006 0:00:00	INFANTIL
2	EL GRAN LABERINTO	FERNANDO SABATER PEREZ	ARIEL	26/9/2006 0:00:00	FICCION
3	ROMEO Y JULIETA	WILLIAM SHAKESPEARE	SALAMANDRA	26/9/2006 0:00:00	ROMANTICA
4	LA CARTA ESFERICA	ARTURO PEREZ LOPEZ	SALAMANDRA	29/9/2006 0:00:00	FICCION
5	CODIGO DA VINCI	DAN BROWN	ARIEL	29/9/2006 0:00:00	FICCION
6	MUCHO RUIDO Y POCAS NUECES	WILLIAM SHAKESPEARE	SALAMANDRA	29/9/2006 0:00:00	ROMANTICA
7	PROTOCOLO	JOSE LOPEZ MURILLO	SALAMANDRA	6/9/2006 0:00:00	SOCIAL
8	LINUX	FERNANDO SABATER PEREZ	ARIEL	6/9/2006 0:00:00	INFORMATICA
9	EL TUMULTO	H.P LOVERCRAFT	DEBATE	6/9/2006 0:00:00	CIENCIA
10	PERSONAJES MITICOS	RICHARD HOLLIGHAM	DEBATE	7/9/2006 0:00:00	ENTRETENIMIENTO
11	EL TIEMPO	J.K ROWINS	SALAMANDRA	7/9/2006 0:00:00	CIENCIA
12	DIETAS MEDITERRANEAS	ARTURO PEREZ LOPEZ	ARIEL	16/9/2006 0:00:00	ASTRONOMIA
13	ANGELES Y DEMONIOS	DAN BROWN	ARIEL	17/9/2006 0:00:00	FICCION
14	FORTALEZA DIGITAL	DAN BROWN	ARIEL	6/10/2006 0:00:00	FICCION
15	CAPITAN ALATRISTE	ARTURO PEREZ LOPEZ	ALFAGUARA	9/10/2006 0:00:00	FICCION
16	PIEL DE TAMBOR	ARTURO PEREZ LOPEZ	ALFAGUARA	16/10/2006 0:00:00	FICCION
17	TIEMPOS DE COLERA	GABRIEL GARCIA GARCIA	OVEJA NEGRA	1/9/2006 0:00:00	OCIO
18	NOTICIA DE UN SECUESTRO	GABRIEL GARCIA GARCIA	ALFAGUARA	7/12/2006 0:00:00	FICCION

2. BACKEND CON ARRAYLIST

Inserción

Paso 6) Para el proceso de inserción debemos crear un formulario debajo del listado de la vista consulta.html, tal y como se ve en la figura:

consulta.html

```
                <td><span th:text="${libro.titulo}"> Title </span></td>
                <td><span th:text="${libro.autor}"> Author </span></td>
                <td><span th:text="${libro.editorial}"> Editorial </span></td>
                <td><span th:text="${libro.fecha}"> Author </span></td>
                <td><span th:text="${libro.tematica}"> Author </span></td>
            </tr>
        </tbody>
    </table>
    <form action="/insertar" method="post">
        <br>ID: <input type=text name=id>
        TITULO: <input type=text name=titulo>
        AUTOR: <input type=text name=autor><br>
        EDITORIAL: <input type=text name=editorial>
        FECHA: <input type=text name=fecha>
        TEMATICA: <input type=text name=tematica><br>
        <input type=submit name=submit value="Insertar Libro">
    </form>
</body>
</html>
```

Id	Titulo	Autor	Editorial	Fecha	Tematica
1	HARRY POTTER Y EL PRISIONERO DE AZKABAN	J.K ROWINS	SALAMANDRA	26/9/2006 0:00:00	INFANTIL
2	EL GRAN LABERINTO	FERNANDO SABATER PEREZ	ARIEL	26/9/2006 0:00:00	FICCION
3	ROMEO Y JULIETA	WILLIAM SHAKESPEARE	SALAMANDRA	26/9/2006 0:00:00	ROMANTICA
4	LA CARTA ESFERICA	ARTURO PEREZ LOPEZ	SALAMANDRA	29/9/2006 0:00:00	FICCION
5	CODIGO DA VINCI	DAN BROWN	ARIEL	29/9/2006 0:00:00	FICCION
6	MUCHO RUIDO Y POCAS NUECES	WILLIAM SHAKESPEARE	SALAMANDRA	29/9/2006 0:00:00	ROMANTICA
7	PROTOCOLO	JOSE LOPEZ MURILLO	SALAMANDRA	6/9/2006 0:00:00	SOCIAL
8	LINUX	FERNANDO SABATER PEREZ	ARIEL	6/9/2006 0:00:00	INFORMATICA
9	EL TUMULTO	H.P LOVERCRAFT	DEBATE	6/9/2006 0:00:00	CIENCIA
10	PERSONAJES MITICOS	RICHARD HOLLIGHAM	DEBATE	7/9/2006 0:00:00	ENTRETENIMIENTO
11	EL TIEMPO	J.K ROWINS	SALAMANDRA	7/9/2006 0:00:00	CIENCIA
12	DIETAS MEDITERRANEAS	ARTURO PEREZ LOPEZ	ARIEL	16/9/2006 0:00:00	ASTRONOMIA
13	ANGELES Y DEMONIOS	DAN BROWN	ARIEL	17/9/2006 0:00:00	FICCION
14	FORTALEZA DIGITAL	DAN BROWN	ARIEL	6/10/2006 0:00:00	FICCION
15	CAPITAN ALATRISTE	ARTURO PEREZ LOPEZ	ALFAGUARA	9/10/2006 0:00:00	FICCION
16	PIEL DE TAMBOR	ARTURO PEREZ LOPEZ	ALFAGUARA	16/10/2006 0:00:00	FICCION
17	TIEMPOS DE COLERA	GABRIEL GARCIA GARCIA	OVEJA NEGRA	1/9/2006 0:00:00	OCIO
18	NOTICIA DE UN SECUESTRO	GABRIEL GARCIA GARCIA	ALFAGUARA	7/12/2006 0:00:00	FICCION

2. BACKEND CON ARRAYLIST

Inserción

Paso 7) En el controlador debemos crear un nuevo handler con PostMapping insertar, que atenderá las peticiones de inserción de un nuevo libro

Como inicialmente ya pasábamos el usuario y el password a consulta.html, ahora debemos realizar una estrategia para continuar haciendo lo mismo.

Controlador.java

```
16 @Controller  //Lo convertimos en un servlet atiende peticiones http
17 @RequestMapping("")
18 public class Controlador {
19
20     BaseDatos bd = new BaseDatos();
21     Usuario usuario;
22
23     @GetMapping("/")
24     public String iniciar(Model model) {
25         model.addAttribute("titulo","FORMULARIO DE ACCESO");
26         return "login";
27     }
28
29     @PostMapping("/")
30     public String login(Usuario usuario, Model model) {
31         if (usuario.getNombre().equals("edu") && usuario.getPassword().equals("edu")) {
32             ArrayList<Libro> libros = bd.getLibros();
33             model.addAttribute("usuario",usuario);
34             this.usuario = usuario;
35             model.addAttribute("libros",libros);
36             return "consulta";
37         }else
38             return "login";
39     }
40
41     @PostMapping("/insertar")
42     public String insertar(Libro libro, Model model) {
43         bd.inserta(libro);
44         ArrayList<Libro> libros = bd.getLibros();
45         model.addAttribute("usuario",this.usuario);
46         model.addAttribute("libros",libros);
47         return "consulta";
48     }
49 }
```

2. BACKEND CON ARRAYLIST

Inserción

Paso 8) Finalmente en el servicio BaseDatos.java, se debe de crear la función inserta libro.

BaseDatos.java

```
7
8  public class BaseDatos {
9
10     ArrayList<Libro> libros = new ArrayList<Libro>();
11     public BaseDatos() {
12         libros.add(new Libro(1,"HARRY POTTER Y EL PRISIONERO DE AZKABAN","J.K ROWINS","SALAMANDRA","26/9/2006 0:00:00","INFANTIL"));
13         libros.add(new Libro(2,"EL GRAN LABERINTO","FERNANDO SABATER PEREZ","ARIEL","26/9/2006 0:00:00","FICCION"));
14         libros.add(new Libro(3,"ROMEO Y JULIETA","WILLIAM SHAKESPEARE","SALAMANDRA","26/9/2006 0:00:00","ROMANTICA"));
15         libros.add(new Libro(4,"LA CARTA ESFERICA","ARTURO PEREZ LOPEZ","SALAMANDRA","29/9/2006 0:00:00","FICCION"));
16         libros.add(new Libro(5,"CODIGO DA VINCI","DAN BROWN","ARIEL","29/9/2006 0:00:00","FICCION"));
17         libros.add(new Libro(6,"MUCHO RUIDO Y POCAS NUECES","WILLIAM SHAKESPEARE","SALAMANDRA","29/9/2006 0:00:00","ROMANTICA"));
18         libros.add(new Libro(7,"PROTOCOLO","JOSE LOPEZ MURILLO","SALAMANDRA","6/9/2006 0:00:00","SOCIAL"));
19         libros.add(new Libro(8,"LINUX","FERNANDO SABATER PEREZ","ARIEL","6/9/2006 0:00:00","INFORMATICA"));
20         libros.add(new Libro(9,"EL TUMULTO","H.P LOVERCRAFT","DEBATE","6/9/2006 0:00:00","CIENCIA"));
21         libros.add(new Libro(10,"PERSONAJES MITICOS","RICHARD HOLLIGHAM","DEBATE","7/9/2006 0:00:00","ENTRETENIMIENTO"));
22         libros.add(new Libro(11,"EL TIEMPO","J.K ROWINS","SALAMANDRA","7/9/2006 0:00:00","CIENCIA"));
23         libros.add(new Libro(12,"DIETAS MEDITERRANEAS","ARTURO PEREZ LOPEZ","ARIEL","16/9/2006 0:00:00","ASTRONOMIA"));
24         libros.add(new Libro(13,"ANGELES Y DEMONIOS","DAN BROWN","ARIEL","17/9/2006 0:00:00","FICCION"));
25         libros.add(new Libro(14,"FORTALEZA DIGITAL","DAN BROWN","ARIEL","6/10/2006 0:00:00","FICCION"));
26         libros.add(new Libro(15,"CAPITAN ALATRISTE","ARTURO PEREZ LOPEZ","ALFAGUARA","9/10/2006 0:00:00","FICCION"));
27         libros.add(new Libro(16,"PIEL DE TAMBOR","ARTURO PEREZ LOPEZ","ALFAGUARA","16/10/2006 0:00:00","FICCION"));
28         libros.add(new Libro(17,"TIEMPOS DE COLERA","GABRIEL GARCIA GARCIA","OVEJA NEGRA","1/9/2006 0:00:00","OCIO"));
29         libros.add(new Libro(18,"NOTICIA DE UN SECUESTRO","GABRIEL GARCIA GARCIA","ALFAGUARA","7/12/2006 0:00:00","FICCION"));
30     }
31
32     public void inserta(Libro libro) {
33         libros.add(libro);
34     }
35
```

2. BACKEND CON ARRAYLIST

Borrado

Paso 9) Para el borrado crearemos una nueva columna paralela al listado de libros. Insertaremos un enlace a la url /borrado/{id}, de manera que nos redirigirá a un método del controlador que borrará el libro a través de su id.

consulta.html

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3  <head>
4      <meta charset="ISO-8859-1">
5      <title>LISTADO DE LIBROS</title>
6  </head>
7  <body>
8   <h3>Usuario: <span th:text="${usuario.nombre}"> </span> - Password: <span th:text="${usuario.password}"> </span></h3>
9
10 <H1>LISTADO DE LIBROS</H1>
11     <table border=1>
12         <thead>
13             <tr><th> Id <th> Titulo <th> Autor <th> Editorial <th> Fecha <th> Tematica <th> Borrado
14             </tr>
15         </thead>
16         <tbody>
17         <tr th:if="${libros.empty}">
18                 <td colspan="2"> No Books Available </td>
19             </tr>
20             <tr th:each="libro : ${libros}">
21                 <td><span th:text="${libro.id}"></span></td>
22                 <td><span th:text="${libro.titulo}"></span></td>
23                 <td><span th:text="${libro.autor}"></span></td>
24                 <td><span th:text="${libro.editorial}"></span></td>
25                 <td><span th:text="${libro.fecha}"></span></td>
26                 <td><span th:text="${libro.tematica}"></span></td>
27                 <td><a th:href="@{/borrado/}+${libro.id}">Borrado</a></td>
28             </tr>
29         </tbody>
30     </table>
```

2. BACKEND CON ARRAYLIST

Borrado

Paso 10) Creamos un método Get en la url /borrado/{id} del controlador. Después de borrar el libro, se vuelve a obtener la lista actualizada de libros para pasársela a la vista.

Hay más parámetros que se pasan a la vista.

- El parámetros usuario es para mantenerlo ya que se pasó al inicio.
- Los parámetros botón y action es para modificar el formulario durante el proceso de modificación que veremos a continuación

Controlador.java

```
43
44      @PostMapping("/insertar")
45      public String insertar(Libro libro, Model model) {
46          bd.inserta(libro);
47          ArrayList<Libro> libros = bd.getLibros();
48          model.addAttribute("usuario",this.usuario);
49          model.addAttribute("libros",libros);
50          model.addAttribute("boton","Inserta Libro");
51          model.addAttribute("action","/insertar");
52          model.addAttribute("libro",null);
53          return "consulta";
54      }
55
56      @GetMapping("/borrado/{id}")
57      public String borrar(@PathVariable int id, Model model) {
58          bd.borrar(id);
59          ArrayList<Libro> libros = bd.getLibros();
60          model.addAttribute("libros",libros);
61          model.addAttribute("usuario",this.usuario);
62          model.addAttribute("boton","Inserta Libro");
63          model.addAttribute("action","/insertar");
64          return "consulta";
65      }
```

2. BACKEND CON ARRAYLIST

Borrado

Paso 11) Finalmente para el proceso de borrado debemos crear la función borrar en BaseDatos.java:

BaseDatos.java

```
35      public void inserta(Libro libro) {
36          libros.add(libro);
37      }
38
39      public void borrar(int id) {
40          Iterator<Libro> it = libros.iterator();
41          while(it.hasNext()) {
42              Libro li = it.next();
43              if (li.id==id) {
44                  it.remove();
45                  break;
46              }
47          }
48      }
```

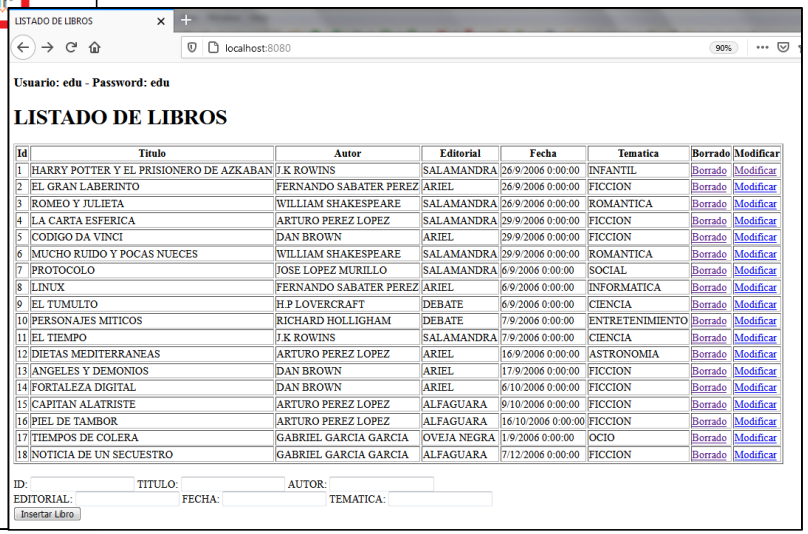
2. BACKEND CON ARRAYLIST

Modificación

Paso 12) Creamos una nueva columna “Modificar” al lado de Borrado, que contenga un enlace con el identificador del libro. Esto nos permita realizar la primera acción de la modificación que es mostrar el libro en el formulario de inserción:

consulta.html

```
<body>
  <h3>Usuario: <span th:text="${usuario.nombre}"> </span> - Password: <span th:text="${usuario.password}"> </span></h3>

<H1>LISTADO DE LIBROS</H1>
    <table border=1>
        <thead>
            <tr><th> Id <th> Titulo <th> Autor <th> Editorial <th> Fecha <th> Tematica <th> Borrado <th> Modificar
            </tr>
        </thead>
        <tbody>
        <tr th:if="${libros.empty}">
                <td colspan="2"> No Books Available </td>
            </tr>
            <tr th:each="libro : ${libros}">
                <td><span th:text="${libro.id}"></span></td>
                <td><span th:text="${libro.titulo}"></span></td>
                <td><span th:text="${libro.autor}"></span></td>
                <td><span th:text="${libro.editorial}"></span></td>
                <td><span th:text="${libro.fecha}"></span></td>
                <td><span th:text="${libro.tematica}"></span></td>
                <td><a th:href="@{/borrado/}+${libro.id}">Borrado</a></td>
                <td><a th:href="@{/modificar/}+${libro.id}">Modificar</a></td>
            </tr>
        </tbody>
    </table>
    <form action="/insertar" method="post">
        <br>ID: <input type=text name=id>
        TITULO: <input type=text name=titulo>
```



Id	Titulo	Autor	Editorial	Fecha	Tematica	Borrado	Modificar
1	HARRY POTTER Y EL PRISIONERO DE AZKABAN	J.K ROWINS	SALAMANDRA	26/9/2006 0:00:00	INFANTIL	Borrado	Modificar
2	EL GRAN LABERINTO	FERNANDO SABATER PEREZ	ARIEL	26/9/2006 0:00:00	FICCION	Borrado	Modificar
3	ROMEO Y JULIETA	WILLIAM SHAKESPEARE	SALAMANDRA	26/9/2006 0:00:00	ROMANTICA	Borrado	Modificar
4	LA CARTA ESFERICA	ARTURO PEREZ LOPEZ	SALAMANDRA	29/9/2006 0:00:00	FICCION	Borrado	Modificar
5	CODIGO DA VINCI	DAN BROWN	ARIEL	29/9/2006 0:00:00	FICCION	Borrado	Modificar
6	MUCHO RUIDO Y POCAS NUECES	WILLIAM SHAKESPEARE	SALAMANDRA	29/9/2006 0:00:00	ROMANTICA	Borrado	Modificar
7	PROTOCOLO	JOSE LOPEZ MURILLO	SALAMANDRA	6/9/2006 0:00:00	SOCIAL	Borrado	Modificar
8	LINUX	FERNANDO SABATER PEREZ	ARIEL	6/9/2006 0:00:00	INFORMATICA	Borrado	Modificar
9	EL TUMULTO	H.P LOVERCRAFT	DEBATE	6/9/2006 0:00:00	CIENCIA	Borrado	Modificar
10	PERSONAJES MITICOS	RICHARD HOLLIGHAM	DEBATE	7/9/2006 0:00:00	ENTRETENIMIENTO	Borrado	Modificar
11	EL TIEMPO	J.K ROWINS	SALAMANDRA	7/9/2006 0:00:00	CIENCIA	Borrado	Modificar
12	DIETAS MEDITERRANEAS	ARTURO PEREZ LOPEZ	ARIEL	16/9/2006 0:00:00	ASTRONOMIA	Borrado	Modificar
13	ANGELES Y DEMONIOS	DAN BROWN	ARIEL	17/9/2006 0:00:00	FICCION	Borrado	Modificar
14	FORTALEZA DIGITAL	DAN BROWN	ARIEL	6/10/2006 0:00:00	FICCION	Borrado	Modificar
15	CAPITAN ALATRISTE	ARTURO PEREZ LOPEZ	ALFAGUARA	9/10/2006 0:00:00	FICCION	Borrado	Modificar
16	PIEL DE TAMBOR	ARTURO PEREZ LOPEZ	ALFAGUARA	16/10/2006 0:00:00	FICCION	Borrado	Modificar
17	TIEMPOS DE COLERA	GABRIEL GARCIA GARCIA	OVEJA NEGRA	1/9/2006 0:00:00	OCIO	Borrado	Modificar
18	NOTICIA DE UN SECUESTRO	GABRIEL GARCIA GARCIA	ALFAGUARA	7/12/2006 0:00:00	FICCION	Borrado	Modificar

2. BACKEND CON ARRAYLIST

Modificación

Paso 13) Creamos dos handlers dentro del controlador, uno que nos permita rellenar el formulario de actualización y otro que trate el envio post de este formulario.

Controlador.java

```
66
67      @GetMapping("/modificar/{id}")
68      public String modificar(@PathVariable int id, Model model) {
69          Libro libro = bd.getLibro(id);
70          ArrayList<Libro> libros = bd.getLibros();
71          model.addAttribute("libros", libros);
72          model.addAttribute("libro", libro);
73          model.addAttribute("usuario", this.usuario);
74          model.addAttribute("boton","Actualiza Libro");
75          model.addAttribute("action","/modificar");
76          return "consulta";
77      }
78
79      @PostMapping("/modificar")
80      public String modificar2(Libro libro, Model model) {
81          bd.modifica(libro);
82          ArrayList<Libro> libros = bd.getLibros();
83          model.addAttribute("usuario", this.usuario);
84          model.addAttribute("libros", libros);
85          model.addAttribute("libro",null);
86          model.addAttribute("boton","Inserta Libro");
87          model.addAttribute("action","/insertar");
88          return "consulta";
89      }
90  }
91
```

2. BACKEND CON ARRAYLIST

Modificación

Paso 14) Modificamos el formulario de consulta para que acepte los parámetros de un hipotético libro que se quiera modificar, mediante la sintaxis condicional ${libro?.id}

consulta.html

```
                <td><span th:text="${libro.titulo}"></span></td>
                <td><span th:text="${libro.autor}"></span></td>
                <td><span th:text="${libro.editorial}"></span></td>
                <td><span th:text="${libro.fecha}"></span></td>
                <td><span th:text="${libro.tematica}"></span></td>
                <td><a th:href="@{/borrado/}+${libro.id}">Borrado</a></td>
                <td><a th:href="@{/modificar/}+${libro.id}">Modificar</a></td>
            </tr>
        </tbody>
    </table>

    <form th:action="${action}" method="post">
        <br>ID: <input type=text name=id th:value="${libro?.id}">
        TITULO: <input type=text name=titulo th:value="${libro?.titulo}">
        AUTOR: <input type=text name=autor th:value="${libro?.autor}"><br>
        EDITORIAL: <input type=text name=editorial th:value="${libro?.editorial}">
        FECHA: <input type=text name=fecha th:value="${libro?.fecha}">
        TEMATICA: <input type=text name=tematica th:value="${libro?.tematica}"><br>
        <input type=submit name=submit th:value="${boton}">
    </form>


</body>
</html>
```

2. BACKEND CON ARRAYLIST

Modificación

Paso 15) Finalmente para el proceso de modificación debemos crear la función modifica en BaseDatos.java

BaseDatos.java

```
49
50      public void modifica(Libro libro) {
51          Iterator<Libro> it = libros.iterator();
52          while(it.hasNext()) {
53              Libro li = it.next();
54              if (li.id==libro.id) {
55                  li.titulo=libro.titulo;
56                  li.autor=libro.autor;
57                  li.editorial=libro.editorial;
58                  li.fecha=libro.fecha;
59                  li.tematica=libro.tematica;
60                  break;
61              }
62          }
63      }
```

3. BACKEND CON JDBC

Paso 1) Introducimos las dependencias del driver mysql connector en el fichero pom.xml de maven.

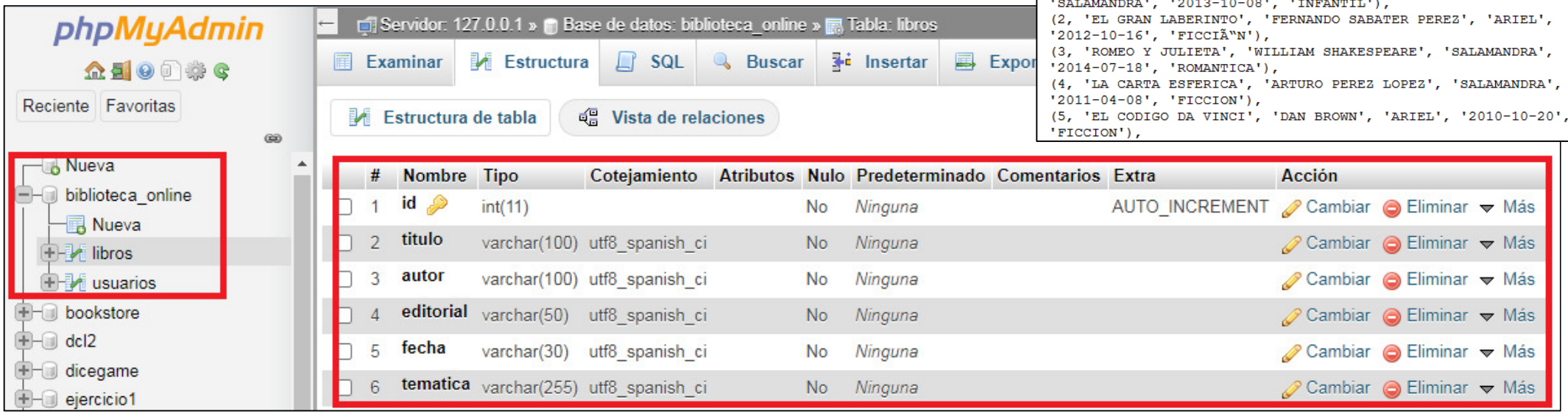
formulario-sboot/pom.xml

```
23          </dependency>
24          <dependency>
25              <groupId>org.springframework.boot</groupId>
26              <artifactId>spring-boot-starter-web</artifactId>
27          </dependency>
28
29          <dependency>
30              <groupId>org.springframework.boot</groupId>
31              <artifactId>spring-boot-devtools</artifactId>
32              <scope>runtime</scope>
33              <optional>true</optional>
34          </dependency>
35          <dependency>
36              <groupId>org.springframework.boot</groupId>
37              <artifactId>spring-boot-starter-test</artifactId>
38              <scope>test</scope>
39          </dependency>
40
41          <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
42          <dependency>
43              <groupId>mysql</groupId>
44              <artifactId>mysql-connector-java</artifactId>
45              <version>8.0.23</version>
46          </dependency>
47
48      </dependencies>
```

3. BACKEND CON JDBC

Paso 2) Se debe crear una base de datos con la tabla libros. Mediante un script cargar en la tabla el contenido anterior del arrayList.

```
CREATE TABLE `libros` (
  `id` int(11) NOT NULL primary key auto_increment,
  `titulo` varchar(100) NOT NULL,
  `autor` varchar(100) NOT NULL,
  `editorial` varchar(50) NOT NULL,
  `fecha` varchar(30) NOT NULL,
  `tematica` varchar(50) NOT NULL
);

--
-- Volcado de datos para la tabla `libros`
--

INSERT INTO `libros` (`id`, `titulo`, `autor`, `editorial`,
`fecha`, `tematica`) VALUES
(1, 'HARRY POTTER Y EL PRISIONERO DE AZKABÃ N', 'J.K ROWLING',
'SALAMANDRA', '2013-10-08', 'INFANTIL'),
(2, 'EL GRAN LABERINTO', 'FERNANDO SABATER PEREZ', 'ARIEL',
'2012-10-16', 'FICCIÃ"N'),
(3, 'ROMEO Y JULIETA', 'WILLIAM SHAKESPEARE', 'SALAMANDRA',
'2014-07-18', 'ROMANTICA'),
(4, 'LA CARTA ESFERICA', 'ARTURO PEREZ LOPEZ', 'SALAMANDRA',
'2011-04-08', 'FICCION'),
(5, 'EL CODIGO DA VINCI', 'DAN BROWN', 'ARIEL', '2010-10-20',
'FICCION'),
```



3. BACKEND CON JDBC

Paso 3) Creamos el fichero BaseDatos2.java con las instrucciones típicas JDBC

BaseDatos2.java

```
13 public class BaseDatos2 {
14
15     private Connection conexion;
16
17     public BaseDatos2() {
18         try {
19             Class.forName("com.mysql.cj.jdbc.Driver");
20             String conex="jdbc:mysql://localhost:3306/biblioteca_online";
21             this.conexion = DriverManager.getConnection (conex,"root","");
22
23         } catch (Exception e) {
24             e.printStackTrace();
25         }
26     }
27
28     public void inserta(Libro libro) {
29         String query = " insert into libros (id, titulo, autor, editorial, fecha, tematica)"
30                 + " values (?, ?, ?, ?, ?, ?)";
31         try {
32             PreparedStatement preparedStmt;
33             preparedStmt = conexion.prepareStatement(query);
34             preparedStmt.setInt (1, libro.getId());
35             preparedStmt.setString (2, libro.getTitulo());
36             preparedStmt.setString (3, libro.getAutor());
37             preparedStmt.setString (4, libro.getEditorial());
38             preparedStmt.setString (5, libro.getFecha());
39             preparedStmt.setString (6, libro.getTematica());
40             preparedStmt.executeUpdate();
41         } catch (SQLException ex) {
42             System.out.print(ex.getMessage());
43         }
44     }
```

BaseDatos2.java

```
78     public Libro getLibro(int id) {
79         Libro libro = null;
80         try {
81             Statement s = conexion.createStatement();
82             String sql = "SELECT * FROM LIBROS WHERE ID="+id;
83             s.execute(sql);
84             ResultSet rs = s.getResultSet();
85             rs.next();
86             libro = new Libro(rs.getInt(1),rs.getString(2),rs.getString(3), rs.getString(4),
87                     rs.getString(5), rs.getString(6));
88         } catch (SQLException ex) {
89             System.out.print(ex.getMessage());
90         }
91
92         return libro;
93     }
```

3. BACKEND CON JDBC

Paso 4) Aquí tenemos las funciones modifica, borrar, getLibros y compruebaUsuario:

BaseDatos2.java

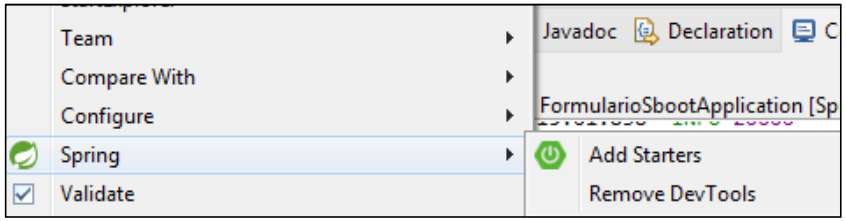
```
46    public void borrar(int id) {
47        String query = " delete from libros where id="+id;
48
49        try {
50            PreparedStatement preparedStmt = conexion.prepareStatement(query);
51            preparedStmt.executeUpdate();
52        } catch (SQLException ex) {
53            System.out.print(ex.getMessage());
54        }
55    }
56
57    public void modifica(Libro libro) {
58
59        String query = " update libros set titulo=?, autor=?, editorial=?, fecha=?, tematica=? "
60                + " where id=?";
61        try {
62            PreparedStatement preparedStmt = conexion.prepareStatement(query);
63            preparedStmt.setString (1, libro.getTitulo());
64            preparedStmt.setString (2, libro.getAutor());
65            preparedStmt.setString (3, libro.getEditorial());
66            preparedStmt.setString (4, libro.getFecha());
67            preparedStmt.setString (5, libro.getTematica());
68            preparedStmt.setInt (6, libro.getId());
69            System.out.print(preparedStmt.toString());
70
71            preparedStmt.executeUpdate();
72        } catch (SQLException ex) {
73            System.out.print(ex.getMessage());
74        }
75    }
```

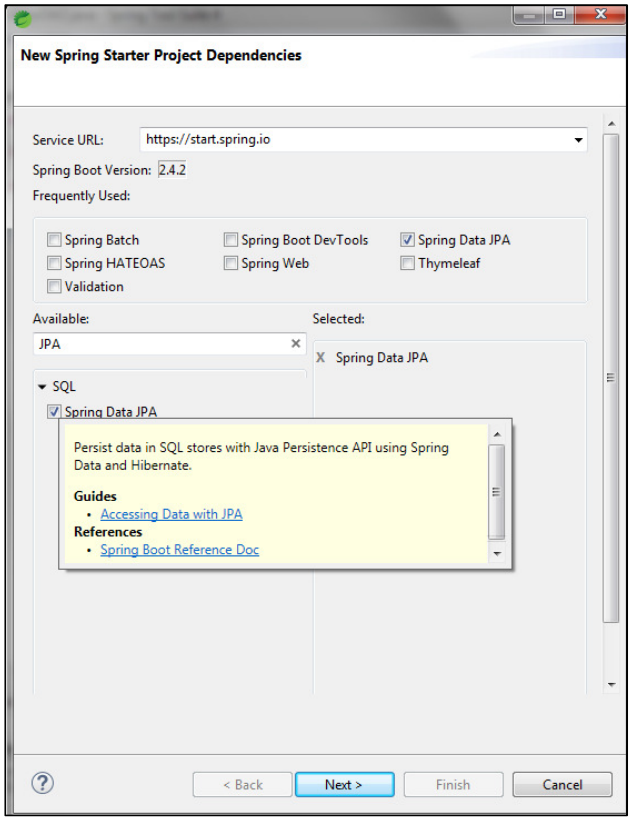
BaseDatos2.java

```
95    public ArrayList<Libro> getLibros() {
96        ArrayList<Libro> lista =new ArrayList<Libro>();
97        try {
98            Statement s = conexion.createStatement();
99            String sql = "SELECT * FROM LIBROS";
100           s.execute(sql);
101           ResultSet rs = s.getResultSet();
102           while (rs.next()) {
103               Libro libro = new Libro(rs.getInt(1), rs.getString(2), rs.getString(3),
104                                       rs.getString(4), rs.getString(5), rs.getString(6));
105               lista.add(libro);
106           }
107       } catch (SQLException ex) {
108           System.out.print(ex.getMessage());
109       }
110       return lista;
111   }
112
113   public boolean compruebaUsuario(String usuario, String password){
114       boolean check=false;
115       try {
116           Statement s = conexion.createStatement();
117           String sql = "SELECT count(*) FROM USUARIOS WHERE usuario='"+usuario+"' "
118                   + "and password='"+password+"'";
119           s.execute(sql);
120           ResultSet rs = s.getResultSet();
121           rs.next();
122           if (rs.getInt(1)>0)
123           check=true;
124       } catch (SQLException ex) {
125           System.out.print(ex.getMessage());
126       }
127       return check;
128   }
```

4. BACKEND CON JPA SPRING

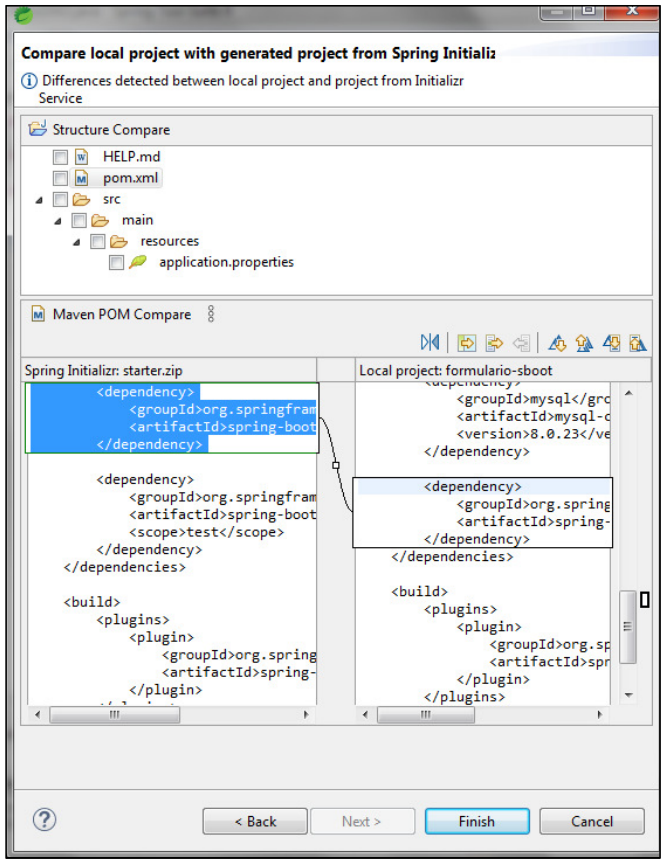
Paso 1) Debemos agregar la libreria JPA Spring Data necesaria para la persistencia. Hacemos click botón derecho encima del proyecto y seleccionamos Spring/Add Starters, Buscamos la librería JPA y la seleccionamos:





4. BACKEND CON JPA SPRING

Paso 2) Al final de este proceso vemos que ha quedado agregada la librería en el fichero pom.xml de Maven:




```
    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.23</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
</dependencies>
```

4. BACKEND CON JPA SPRING

Paso 3) A continuación transformamos la clase Libro en una clase Entity mediante las anotaciones JPA:

@Entity y @Table → Indicamos la tabla a la que referencia esta clase

@Id → Indicamos cual es el clave primaria de la tabla.

@GeneratedValue → Indicamos que se trata de un atributo auto incremento

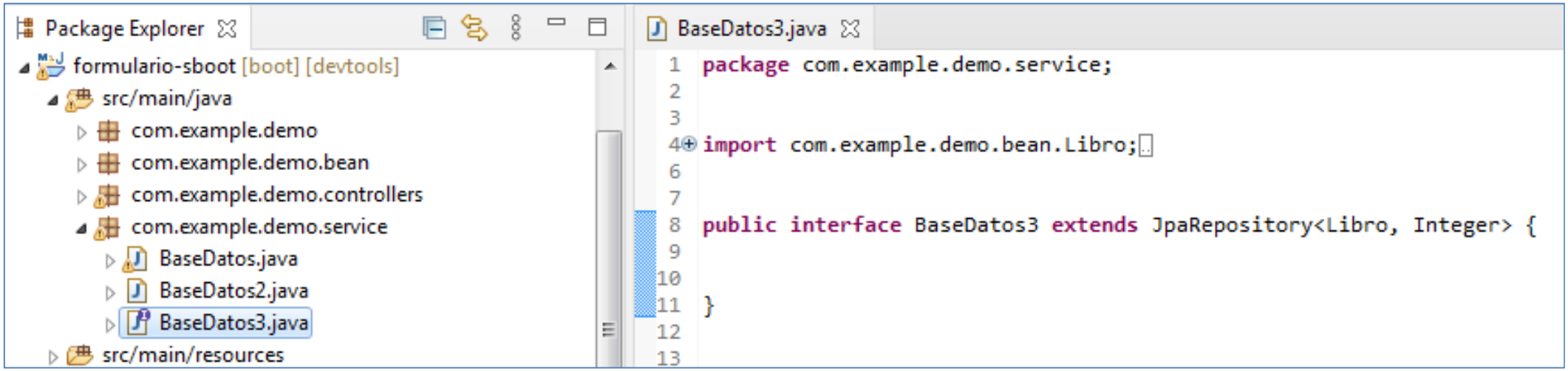
@Colum → Indicamos la columna de la tabla a la cual corresponde el atributo.
Si coincide el nombre del atributo y el de la columna de la tabla no hace falta realizar esta indicación.

Libro.java

```java
package com.example.demo.bean;

import javax.persistence.Column;

@Entity
@Table(name="libros")
public class Libro {

    @Id
    @Column(name="id")
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    public int id;

    @Column(name="titulo", nullable=false, length=30)
    public String titulo;
    public String autor;
    public String editorial;
    public String fecha;
    public String tematica;

    public Libro(int id, String titulo, String autor, String editorial, String fecha, String tematica) {
        this.id = id;
        this.titulo = titulo;
        this.autor = autor;
        this.editorial = editorial;
        this.fecha = fecha;
        this.tematica = tematica;
    }
    public Libro() {
    }
```

4. BACKEND CON JPA SPRING

Paso 4) Crearemos la interfaz DAO BaseDatos3.java que contendrá todas las funciones necesarias del CRUD al heredar de JpaRepository.
Mediante la nomenclatura <Libro, Integer> indicamos la clase entity que va a participar en la persistencia y el tipo de dato de su clave primaria, en este caso un entero.



4. BACKEND CON JPA SPRING

Paso 5) Debemos definir en el archivo application.properties las propiedades de conexión a nuestra base de datos (igual que el connect string de una aplicación JDBC) para que JPA Spring sepa los parámetros de conexión a Mysql

application.properties

```
1  spring.jpa.hibernate.ddl-auto=update
2  spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
3  spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver
4  spring.datasource.url=jdbc:mysql://localhost:3306/biblioteca_online
5  spring.datasource.username=root
6  spring.datasource.password=
```

4. BACKEND CON JPA SPRING

Paso 6) En el controlador creamos un atributo de tipo BaseDatos3, sin el constructor, sólo con la anotación @Autowired. Indicamos al sistema que en caso de que lo necesite, busque una interfaz que implemente BaseDatos3

Esto recibe el nombre de inyección de dependencias: dejo que el sistema llame a una clase que implemente dicha interfaz y de esta manera ya podemos utilizar las funciones de dicha interfaz que se corresponde con las funciones de JpaRepository

Controlador.java

```
1  package com.example.demo.controllers;
2
3  import java.util.ArrayList;
18
19 @Controller  //Lo convertimos en un servlet atiende peticiones http
20 @RequestMapping("")   //localhost:8080
21 public class Controlador {
22
23     Usuario usuario;
24     //BaseDatos bd = new BaseDatos();
25     BaseDatos2 bd2 = new BaseDatos2(); //Para la funcion compruebaUsuario
26     @Autowired
27     BaseDatos3 bd;
28
29     @GetMapping("/") //Da salida al formulario de login
30     public String iniciar(Model model) {
31         model.addAttribute("titulo","FORMULARIO DE ACCESO");
32         return "login";
33     }
```

4. BACKEND CON JPA SPRING

Paso 7) Debemos adaptar las funciones de acceso a base de datos con las funciones que nos ofrece la interfaz:

Controlador.java

```
35    @PostMapping("/")
36    public String login(Usuario usuario, Model model) {
37        if (bd2.compruebaUsuario(usuario.getNombre(),usuario.getPassword())) {
38            //ArrayList<Libro> libros = bd.getLibros();
39            ArrayList<Libro> libros = (ArrayList<Libro>) bd.findAll();
40
41            model.addAttribute("usuario",usuario);
42            this.usuario = usuario;
43            model.addAttribute("libros",libros);
44            model.addAttribute("boton","Inserta Libro");
45            model.addAttribute("action","/insertar");
46            return "consulta";
47        }else {
48            model.addAttribute("titulo","FORMULARIO DE ACCESO");
49            return "login";
50        }
51    }
52
53    @PostMapping("/insertar")
54    public String insertar(Libro libro, Model model) {
55        //bd.inserta(libro);
56        bd.save(libro);
57
58        //ArrayList<Libro> libros = bd.getLibros();
59        ArrayList<Libro> libros = (ArrayList<Libro>) bd.findAll();
60
61        model.addAttribute("usuario",this.usuario);
62        model.addAttribute("libros",libros);
63        model.addAttribute("boton","Inserta Libro");
64        model.addAttribute("action","/insertar");
65        model.addAttribute("libro",null);
66        return "consulta";
67    }
```

Controlador.java

```
84    @GetMapping("/modificar/{id}")
85    public String modificar(@PathVariable int id, Model model) {
86        //Libro libro = bd.getLibro(id);
87        Libro libro = bd.findById(id).get();
88
89        //ArrayList<Libro> libros = bd.getLibros();
90        ArrayList<Libro> libros = (ArrayList<Libro>) bd.findAll();
91
92        model.addAttribute("libros", libros);
93        model.addAttribute("libro", libro);
94        model.addAttribute("usuario", this.usuario);
95        model.addAttribute("boton","Actualiza Libro");
96        model.addAttribute("action","/modificar");
97        return "consulta";
98    }
99
00    @PostMapping("/modificar")
01    public String modificar2(Libro libro, Model model) {
02        //bd.modifica(libro);
03        bd.save(libro);
04
05        //ArrayList<Libro> libros = bd.getLibros();
06        ArrayList<Libro> libros = (ArrayList<Libro>) bd.findAll();
07
08        model.addAttribute("usuario", this.usuario);
09        model.addAttribute("libros", libros);
10        model.addAttribute("libro",null);
11        model.addAttribute("boton","Inserta Libro");
12        model.addAttribute("action","/insertar");
13        return "consulta";
14    }
15 }
```

4. BACKEND CON JPA SPRING

Paso 8) Faltaría integrar el tema de la rutina compruebaUsuario dentro de un modelo Entity. Por ahora lo hemos solucionado mediante la instanciación de BaseDatos2 que va sobre JDBC: