Ejercicios sesión 2 - MVC

Índice

1. Login de usuario (1 punto)

Vamos a añadir una pequeña regla de navegación a nuestra aplicación, que consistirá en el login del usuario.

Crearemos un controlador al que llamaremos `es.ua.jtech.jsf.AccessController`. Éste deberá tener un método llamado `doLogin`, que comprobará que tanto el login como el password sean la palabra 'admin'. En caso de que no sea así, vuelve a la página de registro y muestra en ella un mensaje de error.

Pista: En la solución que se os dará, el mensaje de error aparecerá en la cabecera de esta manera:

```
<h:panelGroup
     layout="block"
     rendered="#{accessController.loginError}"
     style="color:#B94A48; background-color: #F2DEDE; border: 1px solid
#B94A48; padding: 5px">
    <h:outputText value="#{accessController.errorMsg}" />
</h:panelGroup>
```

2. Guardando el usuario recién logueado (1 punto)

Cuando hayamos realizado el login, deberemos almacenar la información del usuario en algún lado. Para ello, nos crearemos un *managed bean*, que tendrá el siguiente esqueleto:

```
package es.ua.jtech.jsf.model;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;
import javax.faces.bean.ManagedBean;

@ManagedBean(name="user")
//DETERMINAR ÁMBITO
public class UserBean implements Serializable {
   private String name;
   private List<TaskBean> tasks = null;

   public UserBean(){
     tasks = new ArrayList<TaskBean>();
   }

   public void setTasks(List<TaskBean> tasks) {
     this.tasks = tasks;
   }

   public void addTask(TaskBean t){
      ...
   }

   public void removeTask(int index){
     ...
   }
```

```
    //GETTERS Y SETTERS

}
```

Por su parte, el objeto TaskBean tendrá la forma:

```
package es.ua.jtech.jsf.model;

import java.io.Serializable;

public class TaskBean implements Serializable {
   int id=-1;
   String title;
   String description;

   //GETTERS & SETTERS
}
```

Como aún no tenemos la lógica de la parte de las tareas realizada, una vez hagamos login se nos redirigirá a una vista donde tengamos un mensaje que diga "Bienvenido *admin*", y un `commandLink` para poder hacer logout

Deberemos crear una regla de navegación en el fichero `faces-config.xml` que nos lleve de la página de login a esta que acabamos de crear.

3. Logout del usuario

En el ejercicio anterior, hemos hecho el login del usuario y lo hemos llevado a una página donde la única posibilidad es hacer logout. Así, ahora lo que tendremos que hacer será un método `doLogout` en el `AccessController`. Éste se encargará de invalidar la sesión y redirigirnos a la página de login nuevamente.

Como los logouts suelen poder hacerse en distintos puntos de una aplicación, la regla de navegación que insertemos aquí deberá hacer uso de *wildcards* en la etiqueta `from-view-id`