

Alumno: Alejandro Herrera Mendoza

Descripción: Sistema web para registrar la información de un doctor para ser almacenada en un archivo txt. con el fin de poder tener acceso al sistema con su usuario y contraseña antes registradas y poder visualizar su información en una nueva página.

Tecnologías: Jsp, Spring framework 3.2.13.RELEASE, Maven.

Lenguaje de programación: Java

Código

Archivo de configuración para asignar las características principales para el funcionamiento del proyecto en Spring.

Servlet DispatcherServlet encargado de resolver la vistas apropiadas a mostrar.

```
web.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app id="WebApp_ID" version="3.0" xmlns="http://java.sun.com/xml
3
4 <welcome-file-list>
5     <welcome-file>login.jsp</welcome-file>
6 </welcome-file-list>
7
8 <context-param>
9     <param-name>contextConfigLocation</param-name>
10    <param-value>/WEB-INF/court-service.xml</param-value>
11 </context-param>
12 <listener>
13    <listener-class>
14        org.springframework.web.context.ContextLoaderListener
15    </listener-class>
16 </listener>
17
18 <servlet>
19    <servlet-name>court</servlet-name>
20    <servlet-class>
21        org.springframework.web.servlet.DispatcherServlet
22    </servlet-class>
23    <load-on-startup>1</load-on-startup>
24 </servlet>
25
26 <servlet-mapping>
27    <servlet-name>court</servlet-name>
28    <url-pattern>/</url-pattern>
29 </servlet-mapping>
30 </web-app>
```

Declaración de las clases de servicios que utilizará Spring.

```
court-service.xml
1 <beans xmlns="http://www.springframework.org/schema/beans"
2     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://www.springframework.org/schema/beans
4     http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
5
6
7     <bean id="loginService"
8         class="com.apress.springrecipes.court.service.LoginServiceImpl" />
9     <bean id="registerService"
10        class="com.apress.springrecipes.court.service.RegisterServiceImpl" />
11
12
13 </beans>
```

Sufijos y prefijos para el reconocimiento de las vistas.

```
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/jsp/" />
    <property name="suffix" value=".jsp" />
</bean>
```

Configuración para mostrar una excepción por default en Spring.

```
<bean
    class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
    <property name="exceptionMappings">
        <props>
            <prop
                key="com.apress.springrecipes.court.service.DoctorNotAvailableException">
                doctorNotAvailable
            </prop>
        </props>
    </property>
</bean>

<bean
    class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
    <property name="exceptionMappings">
        <props>
            <prop
                key="com.apress.springrecipes.court.service.CredencialesNotAvailableException">
                loginNotAvailable
            </prop>
        </props>
    </property>
</bean>
```

- Declaración e instancia de los servicios a utilizar.
- En el método GET es necesario hacer la instancia del objeto del formulario para que pueda ser reconocido en la vista.
- Método POST se obtiene el objeto lleno o vacío pero para verificarlo se utiliza el servicio validator para verificar que todos los campos vengan con su información respectiva.

```
public class LoginController {

    private LoginService service;
    private CredencialesValidator validator;

    @Autowired
    public LoginController(LoginService service, CredencialesValidator validator) {
        this.service = service;
        this.validator = validator;
    }

    @RequestMapping(method = RequestMethod.GET)
    public String login(Model model) {
        Credenciales credenciales = new Credenciales();
        model.addAttribute("credenciales", credenciales);
        return "login";
    }

    @RequestMapping(method = RequestMethod.POST)
    public String validation(@ModelAttribute("credenciales") Credenciales credenciales, BindingResult result,
        SessionStatus status, Model model) {
        Date date = new Date();
        validator.validate(credenciales, result);
        if (result.hasErrors()) {
            model.addAttribute("credenciales", credenciales);
            return "login";
        } else {
            if (service.getValidation(credenciales)) {
                model.addAttribute("fecha", date);
                model.addAttribute("infoDoctor", service.getInfoDoctor());
                return "inicio";
            } else {
                model.addAttribute("credenciales", credenciales);
                model.addAttribute("mensaje", Constantes.mensaje);
                return "login";
            }
        }
    }
}
```

- Clase que implementa el contrato del servicio.
- En caso de no poder guardarse la información del doctor en el archivo lanzará una excepción por default controlada por Spring.

```
public class RegisterServiceImpl implements RegisterService {

    @Override
    public boolean createDoctor(Doctor doctor) throws DoctorNotAvailableException {
        GuardarInformacion doc = new GuardarInformacion();
        if (!doc.save(doctor)) {
            throw new DoctorNotAvailableException(doctor.getNombre(), doctor.getApellidos(), doctor.getDireccion(),
            doctor.getTelefono(), doctor.getRfc(), doctor.getEspecialidad(), doctor.getSexo(),
            doctor.getContra(), doctor.getContra());
        } else {
            return true;
        }
    }
}
```

Clase con el fin de tener los atributos que van a ser verificados en el formulario por el validator.

```
DoctorNotAvailableException.java
1 package com.apress.springrecipes.court.service;
2
3 public class DoctorNotAvailableException extends RuntimeException {
4
5     private String nombre;
6     private String apellidos;
7     private String direccion;
8     private String telefono;
9     private String rfc;
10    private String especialidad;
11    private String sexo;
12    private String correo;
13    private String contra;
14
15    public DoctorNotAvailableException() {
16        super();
17    }
18
19    public DoctorNotAvailableException(String nombre, String apellidos, String direccion, String rfc,
20        String especialidad, String sexo, String telefono, String correo, String contra) {
21        super();
22        this.nombre = nombre;
23        this.apellidos = apellidos;
24        this.direccion = direccion;
25        this.telefono = telefono;
26        this.rfc = rfc;
27        this.especialidad = especialidad;
28        this.sexo = sexo;
29        this.correo = correo;
30        this.contra = contra;
31    }
32
33    public String getNombre() {
34        return nombre;
35    }
36
37    public void setNombre(String nombre) {
38        this.nombre = nombre;
39    }
40}
```

Clase Validator para verificar los campos requeridos y devolver un mensaje de error para cada uno de ellos.

```
DoctorValidator.java
1 package com.apress.springrecipes.court.domain;
2
3 import org.springframework.stereotype.Component;
4
5
6
7
8 @Component
9 public class DoctorValidator implements Validator {
10
11     @Override
12     public boolean supports(Class<?> arg0) {
13         return Doctor.class.isAssignableFrom(arg0);
14     }
15
16     @Override
17     public void validate(Object target, Errors errors) {
18         ValidationUtils.rejectIfEmpty(errors, "nombre", "required.nombre", "El nombre es requerido.");
19         ValidationUtils.rejectIfEmpty(errors, "apellidos", "required.apellidos", "Los apellidos son requeridos.");
20         ValidationUtils.rejectIfEmpty(errors, "direccion", "required.direccion", "La dirección es requerida.");
21         ValidationUtils.rejectIfEmpty(errors, "telefono", "required.telefono", "El teléfono es requerido.");
22         ValidationUtils.rejectIfEmpty(errors, "rfc", "required.rfc", "El rfc es requerido.");
23         ValidationUtils.rejectIfEmpty(errors, "especialidad", "required.especialidad", "La especialidad es requerida.");
24         ValidationUtils.rejectIfEmpty(errors, "sexo", "required.sexo", "El tipo de sexo es requerido.");
25         ValidationUtils.rejectIfEmpty(errors, "correo", "required.correo", "El correo es requerido.");
26         ValidationUtils.rejectIfEmpty(errors, "contra", "required.contra", "La contraseña es requerida.");
27         Doctor doctor = (Doctor) target;
28     }
29 }
30
31
```

Resultados

Validación de los campos requeridos para el formulario de registro.

Insert title here x

http://localhost:8081/SpringDoctorTxt/formulario

Agregar Doctor

Nombre:*

Ingresa tu nombre

El nombre es requerido.

Apellidos:*

Ingresa tus apellidos

Los apellidos son requeridos.

Direccion:*

Ingresa tu dirección

La dirección es requerida.

Teléfono:*

Ingresa tu teléfono

El teléfono es requerido.

RFC:*

Ingresa tus rfc

El rfc es requerido.

Especialidad:*

Ingresa tu especialidad

La especialidad es requerida.

Captura de la información

http://localhost:8081/SpringDoctorTxt/formulario

Agregar Doctor

Nombre:*

Pedro

El nombre es requerido.

Apellidos:*

Martinez

Los apellidos son requeridos.

Direccion:*

privada 3 norte

La dirección es requerida.

Teléfono:*

23455858

El teléfono es requerido.

RFC:*

UHIFE

El rfc es requerido.

Especialidad:*

Cirujano

Especialidad:*

Cirujano

La especialidad es requerida.

Sexo:*

☒ Hombre

☐ Mujer

El tipo de sexo es reuqrido.

Correo:*

pedro@gmail.com

El correo es requerido.

Contraseña:*

pedro

La contraseña es requerida.

Registrar

Campos requeridos para el formulario de inicio de sesión.

Insert title here

http://localhost:8081/SpringDoctorTxt/login

Inicio de sesión

Correo electrónico

Usuario requerido

Contraseña

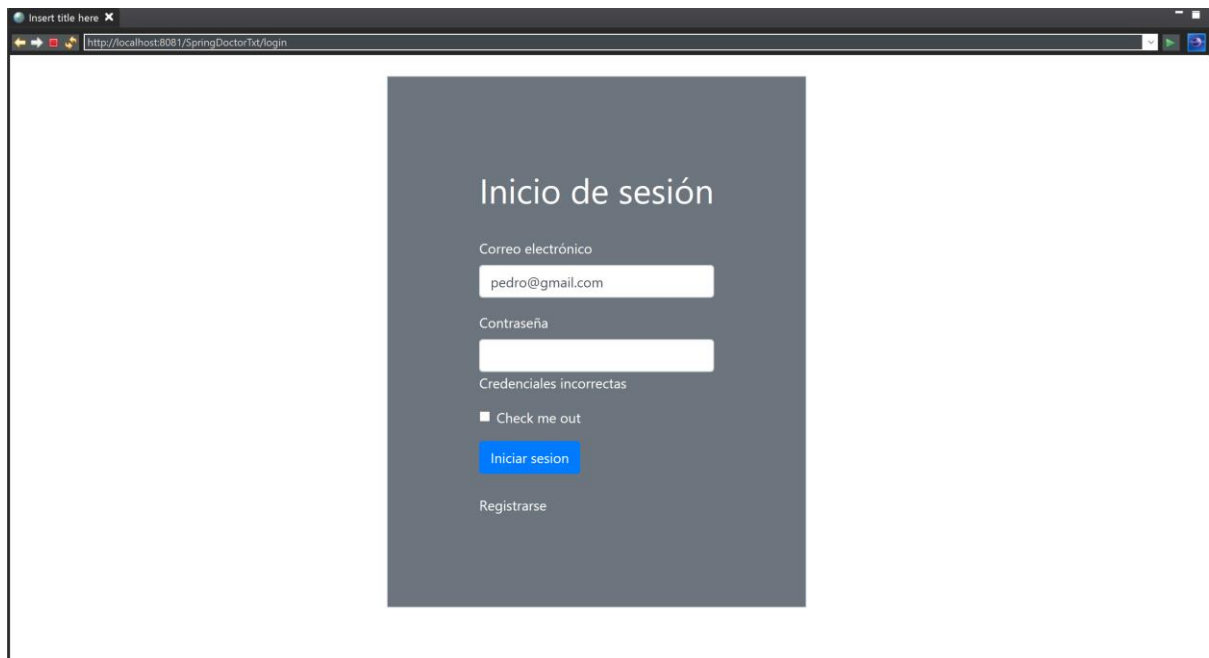
Contraseña requerida

☐ Check me out

Iniciar sesion

Registrarse

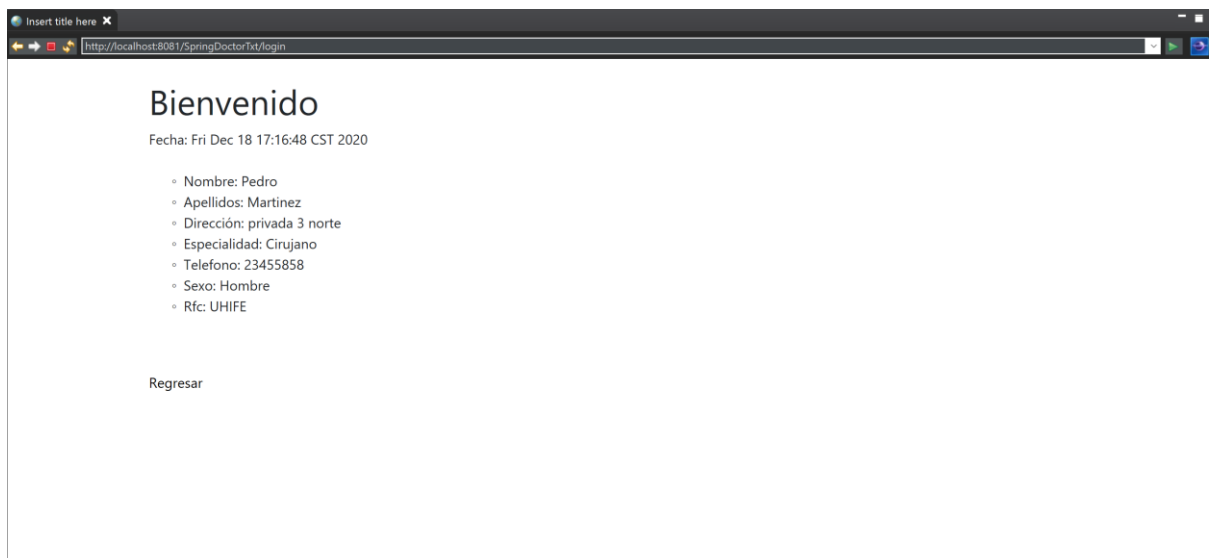
Validación de credenciales.



The screenshot shows a web browser window with the address bar displaying `http://localhost:8081/SpringDoctorTxt/login`. The main content area features a dark gray login form with the following elements:

- Inicio de sesión**: The title of the login form.
- Correo electrónico**: A text input field containing `pedro@gmail.com`.
- Contraseña**: A password input field.
- Credenciales incorrectas**: A message indicating that the credentials are incorrect.
- ☐ **Check me out**: A checkbox for remembering the user.
- Iniciar sesion**: A blue button to submit the login form.
- Registrarse**: A link to the registration page.

Inicio de sesión correcta.



The screenshot shows a web browser window with the address bar displaying `http://localhost:8081/SpringDoctorTxt/login`. The main content area features a white welcome page with the following elements:

- Bienvenido**: The title of the welcome page.
- Fecha: Fri Dec 18 17:16:48 CST 2020**: The current date and time.
- **Nombre: Pedro**
 - **Apellidos: Martinez**
 - **Dirección: privada 3 norte**
 - **Especialidad: Cirujano**
 - **Telefono: 23455858**
 - **Sexo: Hombre**
 - **Rfc: UHIFE**
- Regresar**: A link to return to the previous page.