"Spring MVC: Vistas PDF y Excel, Layout, Ajax Localización y Manejo de Errores"

Módulo 5 / 1

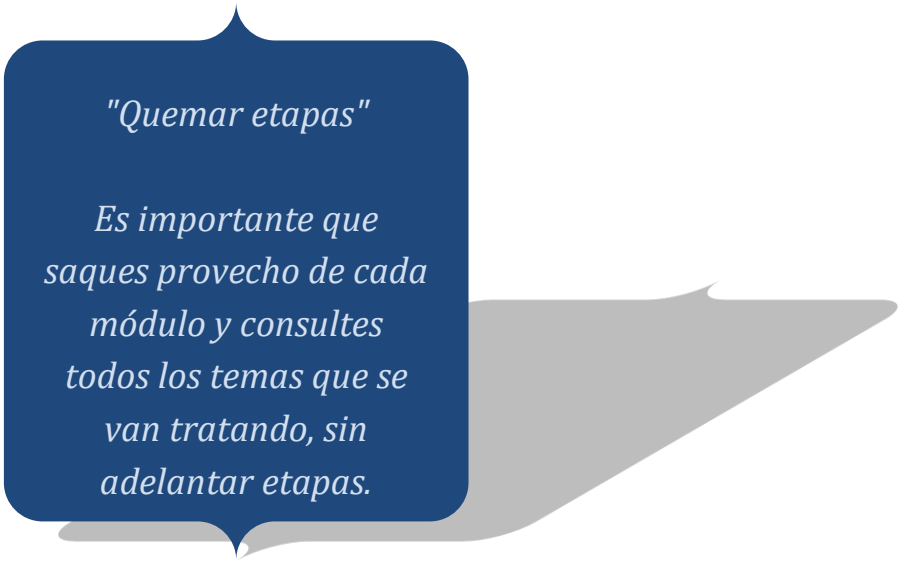
AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

Objetivo

El objetivo de esta práctica es aprender a trabajar con vistas de Spring MVC, una de las áreas en las que sobresale Spring es en la separación las vistas (view technologies) del resto de la aplicación, por ejemplo, la decisión de utilizar Velocity o Pdf o bien Excel, en lugar de una ya existente JSP, es sólo un tema de configuración. Veremos algunos ejemplos para exportar a pdf y excel, además veremos un ejemplo de uso de layout con tiles, también conocidos como **plantilla global**, almacenan código HTML5/CSS3 que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página (vista *.jsp). El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout inserta el contenido de las vistas jsp que corresponde a la acción que es solicitada en la petición.

La plantilla global o layout puede ser adaptada completamente para cada aplicación. Se puede añadir todo el código HTML que sea necesario. **Normalmente se utiliza el layout para mostrar la navegación, el logotipo del sitio, un encabezado de página (header), pie de página (footer) etc**.

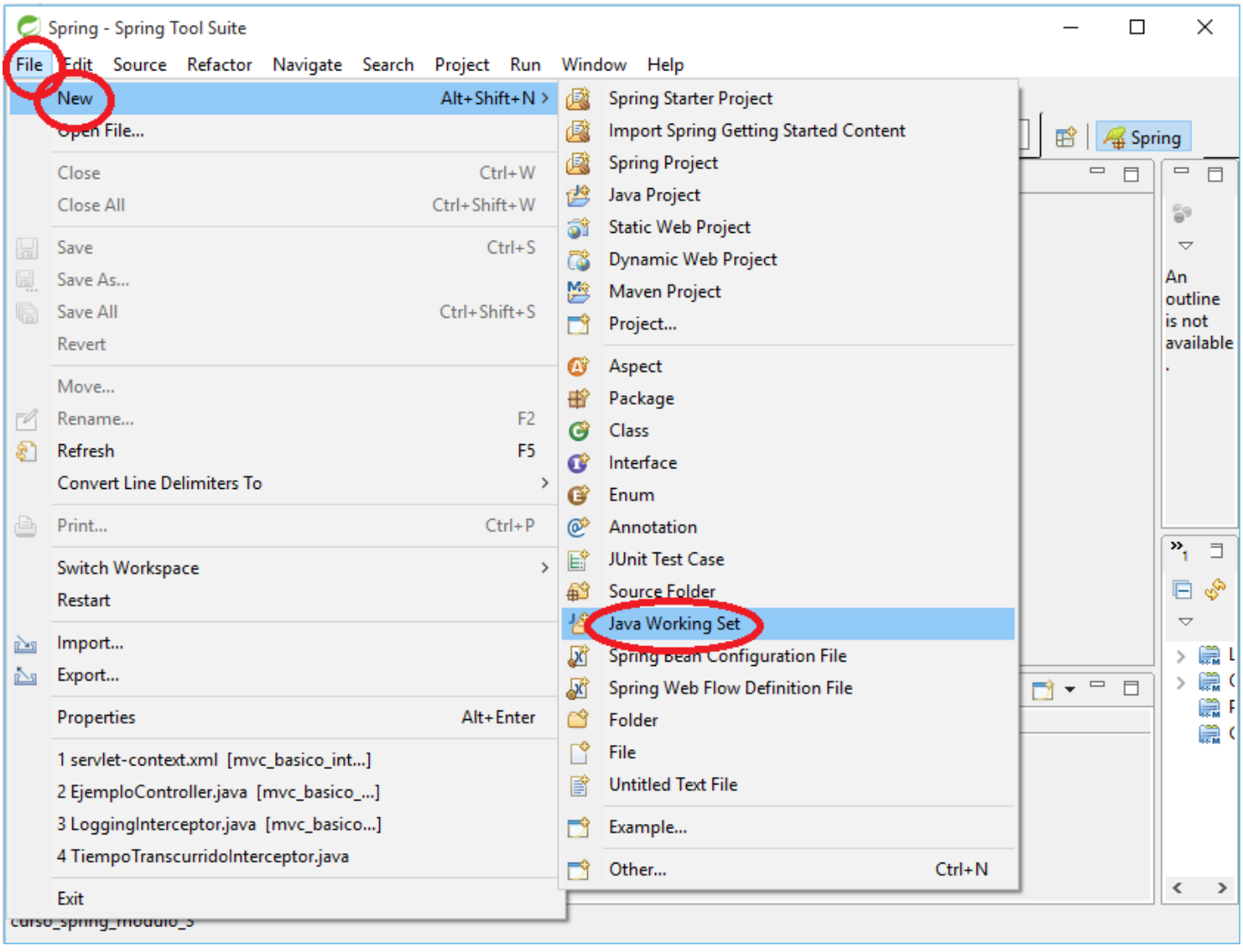
Finalmente veremos un par de ejemplos más, para un proyecto multi-idioma implementado con Spring y el último para aprender sobre el manejo y control de errores/excepciones.

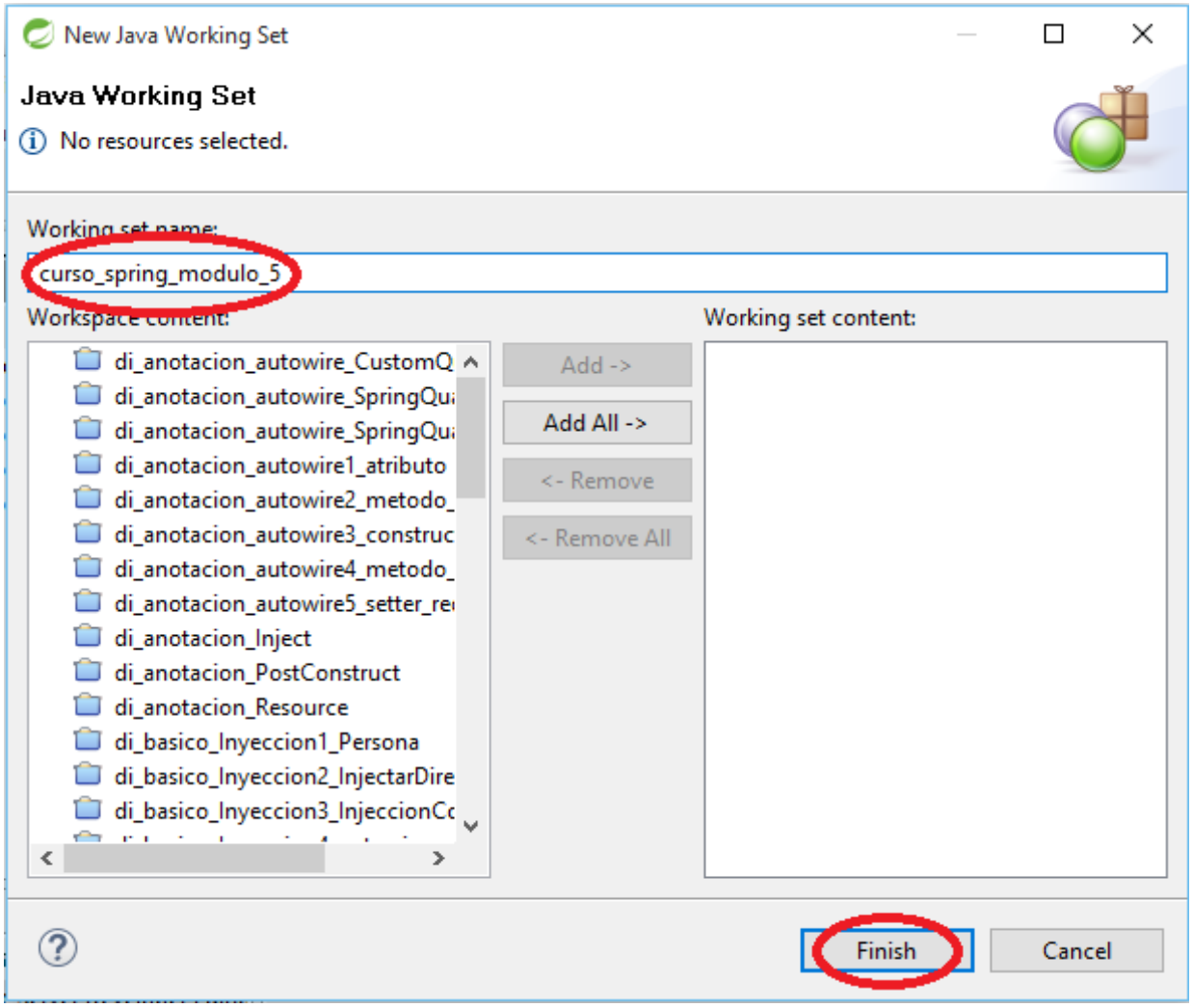


AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

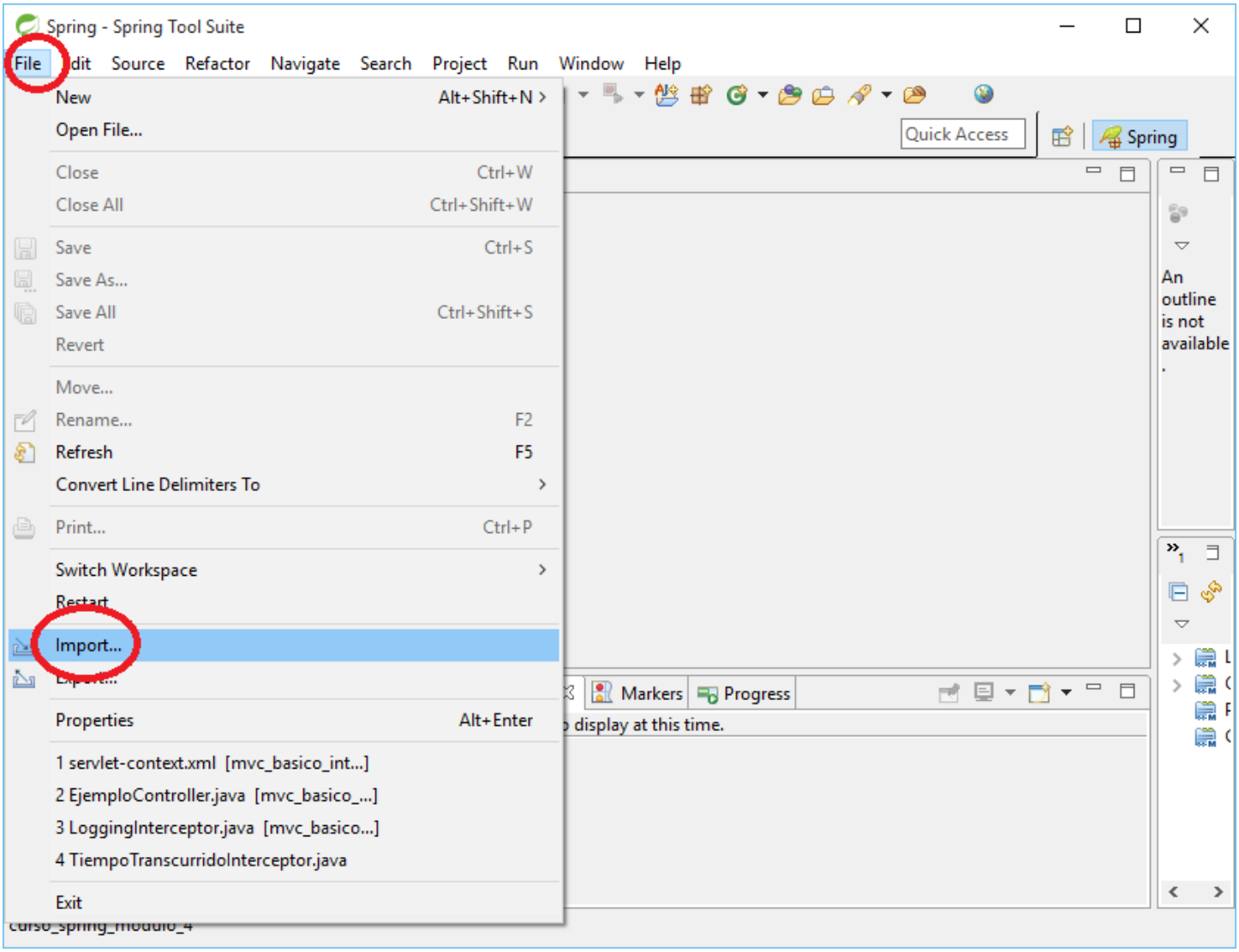
Ejercicio 0: Importar todos los proyectos de ejemplo

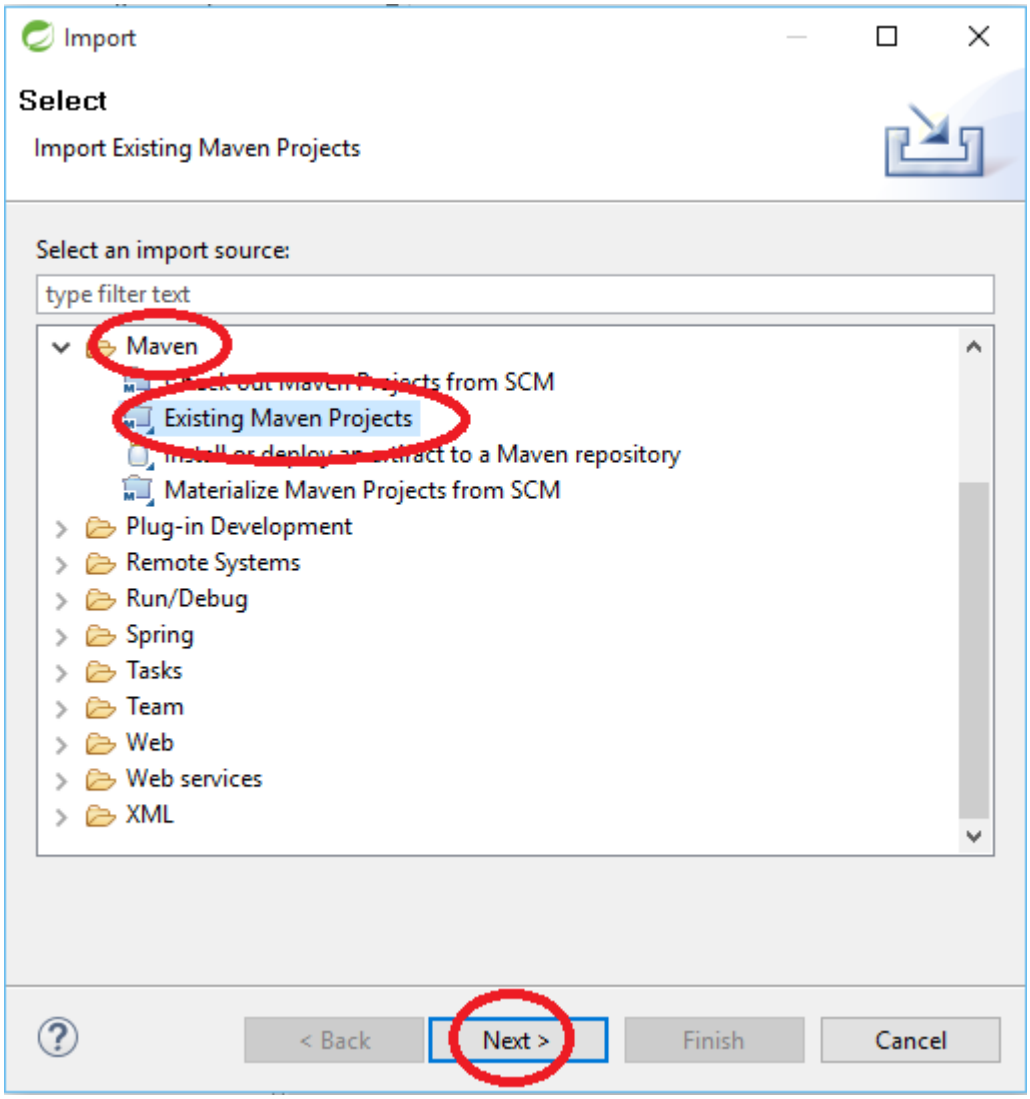
1. Crear un nuevo "**Java Working Set**" llamado "**curso_spring_modulo_5**". Esto es para organizar los proyectos bajo un esquema llamado **Working Set**, similar a como organizamos archivos en directorios.
    - Seleccionar **File->New->Java Working Set**.



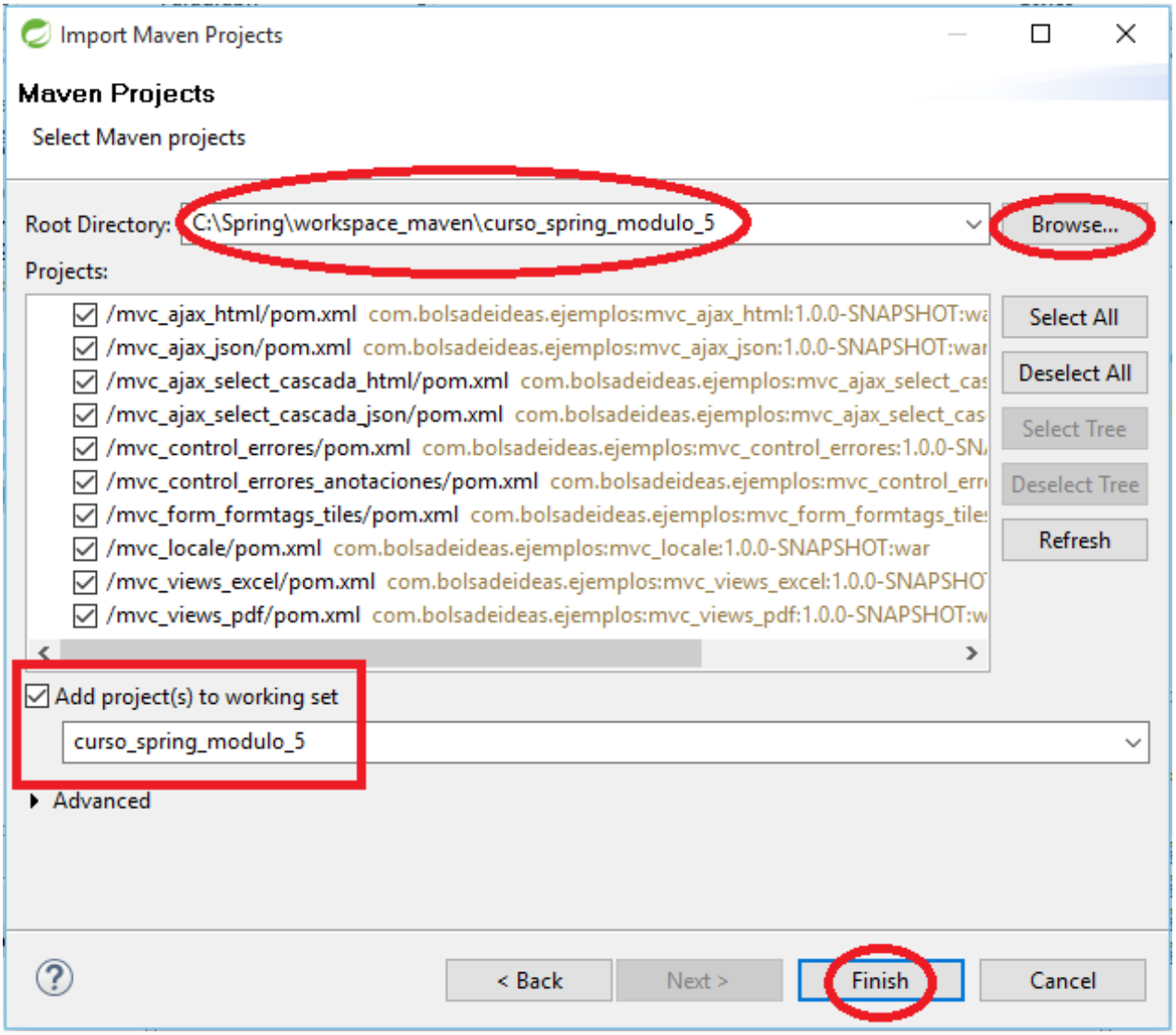

New Java Working Set
Java Working Set
No resources selected.
Working set name:
curso_spring_modulo_5
Workspace content:
di_anotacion_autowire_CustomQ
di_anotacion_autowire_SpringQu
di_anotacion_autowire_SpringQu
di_anotacion_autowire1_atributo
di_anotacion_autowire2_metodo_
di_anotacion_autowire3_construc
di_anotacion_autowire4_metodo_
di_anotacion_autowire5_setter_re
di_anotacion_Inject
di_anotacion_PostConstruct
di_anotacion_Resource
di_basico_Inyeccion1_Persona
di_basico_Inyeccion2_InjectarDire
di_basico_Inyeccion3_InjeccionCo
Add ->
Add All ->
<- Remove
<- Remove All
Working set content:
Finish
Cancel

2. Importar los proyectos de ejemplos en maven.
    - Seleccionar **File->Import**.




Import
Select
Import Existing Maven Projects
Select an import source:
type filter text
Maven
Existing Maven Projects
Materialize Maven Projects from SCM
Plug-in Development
Remote Systems
Run/Debug
Spring
Tasks
Team
Web
Web services
XML
< Back
Next >
Finish
Cancel

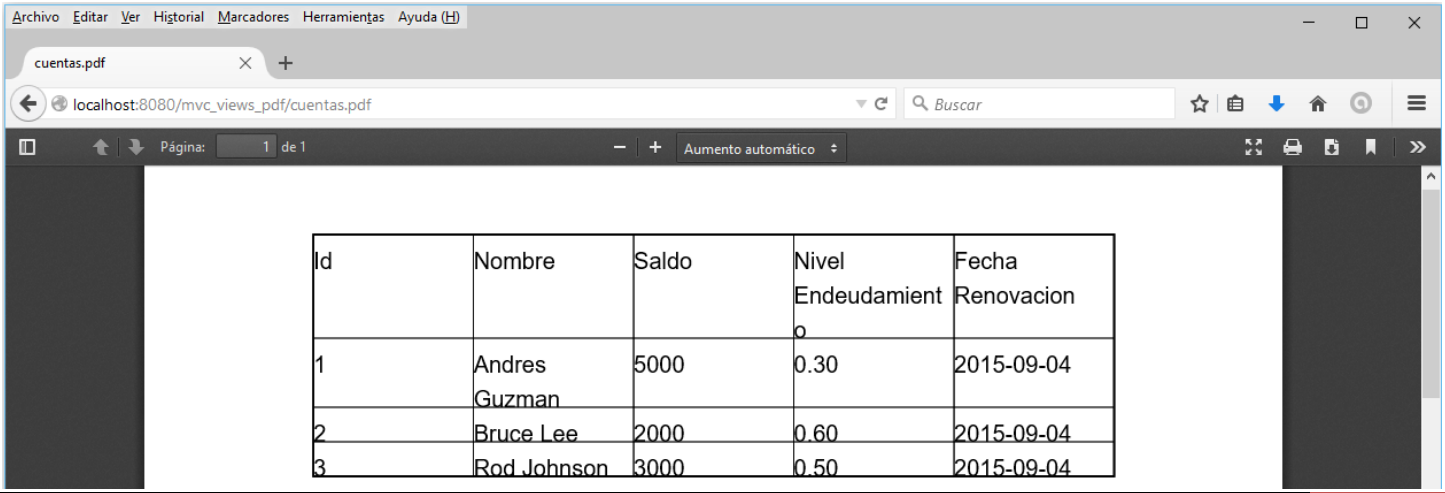
- Clic **Browse**
- Seleccionamos el directorio donde vienen los proyecto de ejemplo del laboratorio
- Agregamos los proyectos al **Working Set curso_spring_modulo_5**
- **Finish**



Ejercicio 1: Generar y ejecutar el ejemplo " Generar PDF como una vista"

En este ejercicio, aprenderemos a generar una vista en PDF, heredando de la súper clase de Spring **AbstractPdfView**.

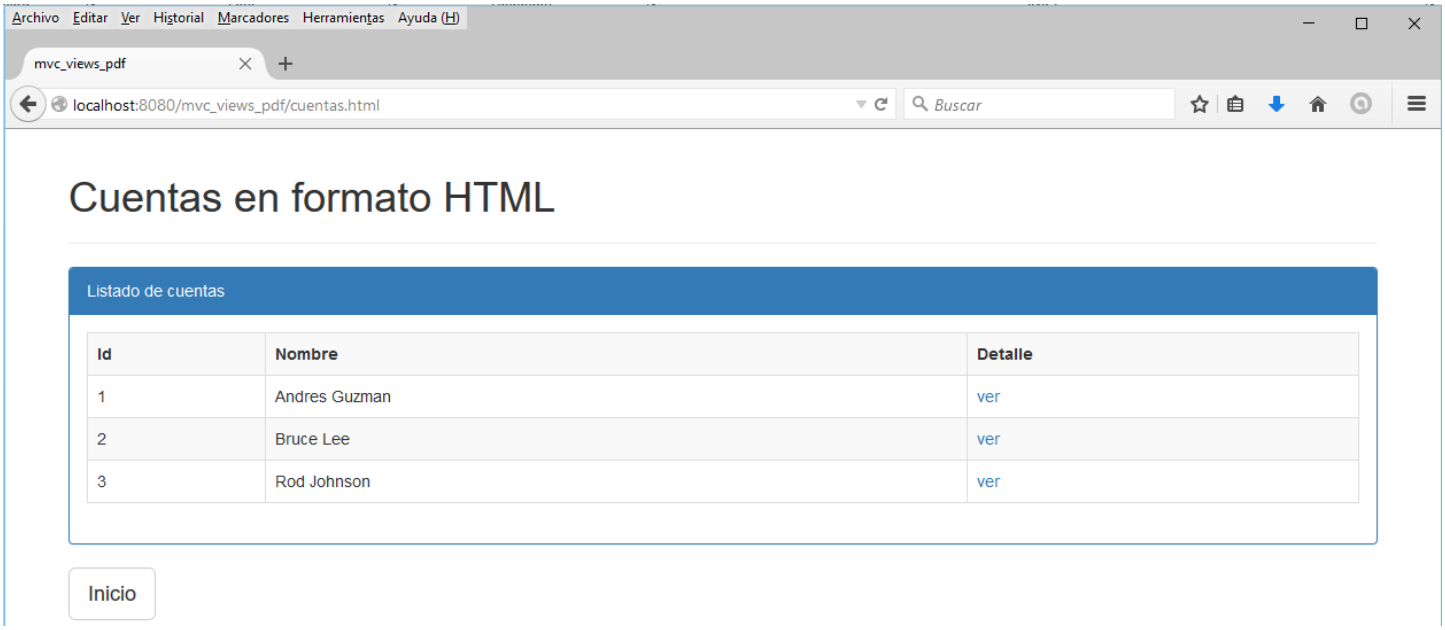
1. Clic derecho sobre el proyecto y **Run As->Maven Clean**
2. Clic derecho sobre el proyecto y **Run As->Maven Install**
3. Clic derecho **Maven->Update Project…**
4. Clic derecho sobre el proyecto **mvc_views_pdf -> Run As on Server**
5. Observe el resultado.
6. Clic en **Exportar a PDF**



Id	Nombre	Saldo	Nivel Endeudamiento	Fecha Renovacion
1	Andres Guzman	5000	0.30	2015-09-04
2	Bruce Lee	2000	0.60	2015-09-04
3	Rod Johnson	3000	0.50	2015-09-04

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

7. Volvemos atrás
8. Clic en **Ver en HTML**



9. Abrir y estudiar la clase controladora **/src/main/java/ com.bolsadeideas.ejemplos.cuenta.controllers/CuentaController.java**

```
package com.bolsadeideas.ejemplos.cuenta.controllers;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Date;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import com.bolsadeideas.ejemplos.cuenta.domain.Cuenta;

@Controller
@RequestMapping(value="/cuentas")
public class CuentaController {

  private Map<Long, Cuenta> cuentas = new ConcurrentHashMap<Long, Cuenta>();

  @RequestMapping(method=RequestMethod.GET)
  public String getAccounts(Model model) {
    // Agregamos algunos datos de ejemplo
    Cuenta account = new Cuenta( new Long(0001), "Andres Guzmán", new BigDecimal("5000"),
        new BigDecimal("0.30"), new Date());
    cuentas.put(new Long(0001), account);

    account =  new Cuenta( new Long(0002), "Bruce Lee", new BigDecimal("2000"),
        new BigDecimal("0.60"), new Date());
    cuentas.put(new Long(0002), account);

    account =  new Cuenta( new Long(0003), "James Gosling", new BigDecimal("3000"),
        new BigDecimal("0.50"), new Date());
        cuentas.put(new Long(0003), account);

    // Agregamos el map de cuentas al atributo del model "cuentas",
    // para desplegarlos en el reporte de la vista excel
    model.addAttribute("cuentas", new ArrayList<Cuenta>(cuentas.values()));
    return "cuentas";
  }

  @RequestMapping(value="{id}", method=RequestMethod.GET)
  public String getView(@PathVariable Long id, Model model) {
    Cuenta cuenta = this.cuentas.get(id);
    model.addAttribute(cuenta);
    return "detalle";
  }
}
```

10. Abrir y estudiar la clase PdfResumenCuenta **/src/main/java/com.bolsadeideas.ejemplos.cuenta.views/PdfResumenCuenta.java**

```
package com.bolsadeideas.ejemplos.cuenta.views;

import java.math.BigDecimal;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.view.document.AbstractPdfView;

import com.bolsadeideas.ejemplos.cuenta.domain.Cuenta;

import com.lowagie.text.Document;
import com.lowagie.text.Table;
import com.lowagie.text.pdf.PdfWriter;

public class PdfResumenCuenta extends AbstractPdfView {
    @Override
    protected void buildPdfDocument(Map<String, Object> model, Document document,
            PdfWriter writer, HttpServletRequest request, HttpServletResponse response)
                    throws Exception {

        List<Cuenta> cuentas = (List<Cuenta>) model.get("cuentas");
        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        Table table = new Table(5);
        table.addCell("Id");
        table.addCell("Nombre");
        table.addCell("Saldo");
        table.addCell("Nivel Endeudamiento");
        table.addCell("Fecha Renovacion");

        if (!cuentas.isEmpty()) {
            for (Cuenta cuenta : cuentas) {
                table.addCell(Long.toString(cuenta.getId()));
                table.addCell(cuenta.getNombre());
                BigDecimal bigDecimal = cuenta.getSaldo();
                table.addCell(bigDecimal.toString());
                bigDecimal = cuenta.getNivelEndeudamiento();
                table.addCell(bigDecimal.toString());
                table.addCell(dateFormat.format(cuenta.getFechaRenovacion()));
            }
        }
        document.add(table);
    }
}
```

11. Ahora estudiemos el archivo de configuración de spring **servlet-context.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/mvc
                    http://www.springframework.org/schema/mvc/spring-mvc.xsd
                    http://www.springframework.org/schema/beans
                    http://www.springframework.org/schema/beans/spring-beans.xsd
                    http://www.springframework.org/schema/context
                    http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- Escanea o busca en el package base de la aplicación clases beans anotados
        con @Components, @Controller, @Service, @Repository -->
    <context:component-scan base-package="com.bolsadeideas.ejemplos.cuenta" />

    <!-- Habilita la anotacion de Spring MVC @Controller -->
    <mvc:annotation-driven />

    <!-- Views configuradas y mapeadas en un bean, ej: id="cuentas" (clases PDF, XLS, etc) -->
     <bean id="contentNegotiatingResolver"
         class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
         <property name="order"
             value="#{T(org.springframework.core.Ordered).HIGHEST_PRECEDENCE}" />
     </bean>

    <!-- BeanNameViewResolver -->
     <bean id="beanNameViewResolver"
         class="org.springframework.web.servlet.view.BeanNameViewResolver">
         <property name="order" value="#{contentNegotiatingResolver.order+1}" />
     </bean>

    <!-- La vista "cuentas" que es manejada por la clase "PdfResumenCuenta" -->
    <bean id="cuentas"
         class="com.bolsadeideas.ejemplos.cuenta.views.PdfResumenCuenta" />

    <!-- Resuelve la ubicacion de las vistas .jsp de @Controllers en la ruta /WEB-INF/views -->
    <bean id="internalResourceResolver"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
        <property name="order" value="#{beanNameViewResolver.order+1}" />
    </bean>
</beans>
```

12. Abrir y estudiar la vista jsp "index.jsp"

```
<div class="container">
     <div class="jumbotron">
       <h1>ContentNegotiatingViewResolver!</h1>
       <p>El ContentNegotiatingViewResolver selecciona el View Resolver más apropiado para
manejar el request comparando el "media type" (conocido también como Content-Type) para
seleccionar la vista adecuada según el "media type" enviado en la URL (como extensión)</p>
       <p>Si el request URL es con extensión .pdf, entonces se seleccionará la vista PdfView
de la lista DefaultViews independientemente del nombre de la vista retornada.</p>
      <p>
          <a class="btn btn-primary btn-lg" href="cuentas.html" role="button">Ver en HTML</a>
          <a class="btn btn-danger btn-lg" href="cuentas.pdf" role="button">Exportar a PDF</a>
      </p>
  </div>
</div>
```

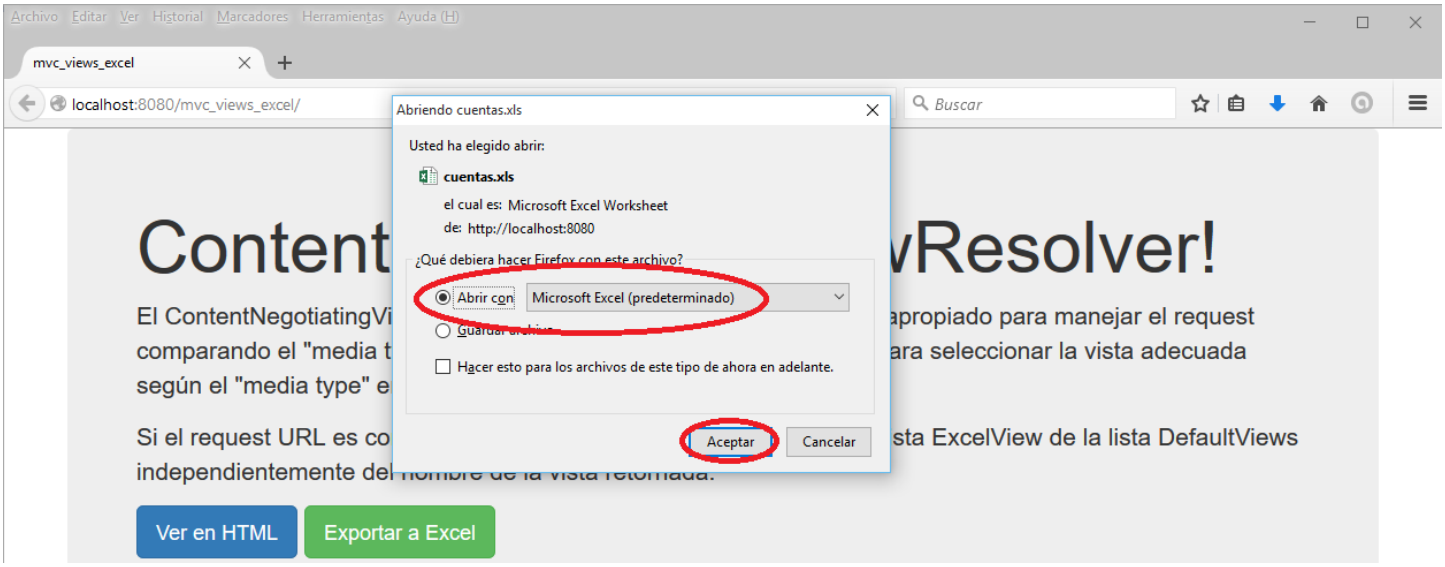
Otro punto importante a tener en cuenta, es tener el pom.xml con las dependencias maven de itext PDF:

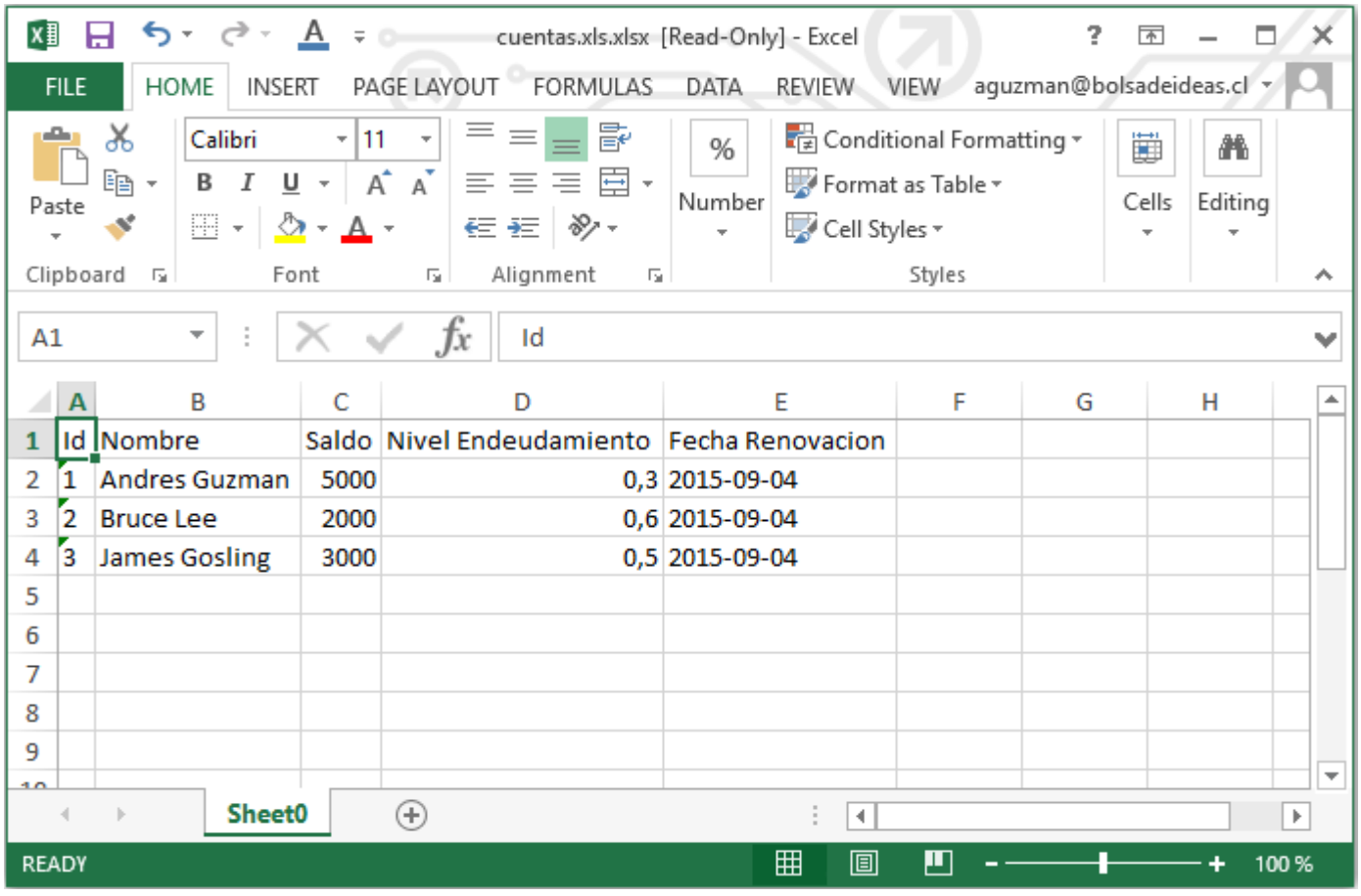
```
ETC…

<!-- PDF support -->
<dependency>
    <groupId>com.lowagie</groupId>
    <artifactId>itext</artifactId>
    <version>2.1.7</version>
</dependency>

ETC…
```

Ejercicio 2: Generar y ejecutar el ejemplo " Generar XLS como una vista"

En este ejercicio, aprenderemos a generar una vista en Excel, heredando de la súper clase de Spring AbstractXlsxView.

1. Clic derecho sobre el proyecto y **Run As->Maven Clean**
2. Clic derecho sobre el proyecto y **Run As->Maven Install**
3. Clic derecho **Maven->Update Project…**
4. Clic derecho sobre el proyecto **mvc_views_xls-> Run As on Server**
5. Observe el resultado.





	A	B	C	D	E
1	Id	Nombre	Saldo	Nivel Endeudamiento	Fecha Renovacion
2	1	Andres Guzman	5000	0,3	2015-09-04
3	2	Bruce Lee	2000	0,6	2015-09-04
4	3	James Gosling	3000	0,5	2015-09-04

6. Abrir y estudiar la clase controladora **/src/main/java/com.bolsadeideas.ejemplos.cuenta.controllers/CuentaController.java**

```
package com.bolsadeideas.ejemplos.cuenta.controllers;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Date;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import com.bolsadeideas.ejemplos.cuenta.domain.Cuenta;

@Controller
@RequestMapping(value="/cuentas")
public class CuentaController {

  private Map<Long, Cuenta> cuentas = new ConcurrentHashMap<Long, Cuenta>();

  @RequestMapping(method=RequestMethod.GET)
  public String getAccounts(Model model) {

    // Agregamos algunos datos de ejemplo
    Cuenta account = new Cuenta( new Long(0001), "Andres Guzmán", new BigDecimal("5000"),
        new BigDecimal("0.30"), new Date());
    cuentas.put(new Long(0001), account);

    account =  new Cuenta( new Long(0002), "Bruce Lee", new BigDecimal("2000"),
        new BigDecimal("0.60"), new Date());
    cuentas.put(new Long(0002), account);

    account =  new Cuenta( new Long(0003), "James Gosling", new BigDecimal("3000"),
        new BigDecimal("0.50"), new Date());
    cuentas.put(new Long(0003), account);

    // Agregamos el map de cuentas al atributo del model "cuentas",
    // para desplegarlos en el reporte de la vista excel
    model.addAttribute("cuentas", new ArrayList<Cuenta>(cuentas.values()));
    return "cuentas";
  }

  @RequestMapping(value="{id}", method=RequestMethod.GET)
  public String getView(@PathVariable Long id, Model model) {
    Cuenta cuenta = this.cuentas.get(id);
    model.addAttribute(cuenta);
    return "detalle";
  }
}
```

7. Abrir y estudiar la clase ExcelResumenCuenta **/src/main/java/com.bolsadeideas.ejemplos.cuenta.views/PdfResumenCuenta.java**

```
package com.bolsadeideas.ejemplos.cuenta.views;

import java.math.BigDecimal;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.poi.ss.usermodel.RichTextString;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.springframework.web.servlet.view.document.AbstractXlsxView;

import com.bolsadeideas.ejemplos.cuenta.domain.Cuenta;

public class ExcelResumenCuenta extends AbstractXlsxView {
    @Override
    protected void buildExcelDocument(Map<String, Object> model, Workbook workbook,
            HttpServletRequest request, HttpServletResponse response) throws Exception {

        List<Cuenta> cuentas = (List<Cuenta>) model.get("cuentas");
        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        Sheet sheet = workbook.createSheet();

        Row header = sheet.createRow(0);

        header.createCell((short) 0).setCellValue(workbook.getCreationHelper()
                                    .createRichTextString("Id"));
        header.createCell((short) 1).setCellValue(workbook.getCreationHelper()
                                    .createRichTextString("Nombre"));
        header.createCell((short) 2).setCellValue(workbook.getCreationHelper()
                                    .createRichTextString("Saldo"));
        header.createCell((short) 3).setCellValue(workbook.getCreationHelper()
                                    .createRichTextString("Nivel Endeudamiento"));
        header.createCell((short) 4).setCellValue(workbook.getCreationHelper()
                                    .createRichTextString("Fecha Renovacion"));
        int rowNum = 1;
        for (Cuenta cuenta : cuentas) {
            Row row = sheet.createRow(rowNum++);
            row.createCell((short) 0).setCellValue(workbook.getCreationHelper()
                                     .createRichTextString(Long.toString(cuenta.getId())));
            row.createCell((short) 1).setCellValue(workbook.getCreationHelper()
                                     .createRichTextString(cuenta.getNombre()));
```

```
            BigDecimal bigDecimal = cuenta.getSaldo();
            row.createCell((short) 2).setCellValue(bigDecimal.doubleValue());
            bigDecimal = cuenta.getNivelEndeudamiento();
            row.createCell((short) 3).setCellValue(bigDecimal.doubleValue());

            RichTextString fechaRenovacion = workbook.getCreationHelper()
                        .createRichTextString(dateFormat.format(cuenta.getFechaRenovacion()));
            row.createCell((short) 4).setCellValue(fechaRenovacion);
        }
    }
}
```

8. Ahora estudiemos el archivo de configuración de spring **servlet-context.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns=http://www.springframework.org/schema/beans
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
                        http://www.springframework.org/schema/mvc/spring-mvc.xsd
                        http://www.springframework.org/schema/beans
                        http://www.springframework.org/schema/beans/spring-beans.xsd
                        http://www.springframework.org/schema/context
                        http://www.springframework.org/schema/context/spring-context.xsd">
    <!-- Escanea o busca en el package base de la aplicación clases beans anotados
        con @Components, @Controller, @Service, @Repository -->
    <context:component-scan base-package="com.bolsadeideas.ejemplos.cuenta" />

    <!-- Habilita la anotacion de Spring MVC @Controller -->
    <mvc:annotation-driven />

    <!-- Views configuradas y mapeadas en un bean, ej: id="cuentas" (clases PDF, XLS, etc) -->
    <bean id="contentNegotiatingResolver"
        class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
        <property name="order"
            value="#{T(org.springframework.core.Ordered).HIGHEST_PRECEDENCE}" />
    </bean>

    <!-- BeanNameViewResolver -->
    <bean id="beanNameViewResolver"
        class="org.springframework.web.servlet.view.BeanNameViewResolver">
        <property name="order" value="#{contentNegotiatingResolver.order+1}" />
    </bean>

    <!-- La vista "cuentas" que es manejada por la clase "ExcelResumenCuenta" -->
    <bean id="cuentas"
        class="com.bolsadeideas.ejemplos.cuenta.views.ExcelResumenCuenta" />
```

```
    <!-- Resuelve la ubicacion de las vistas .jsp de @Controllers en la ruta /WEB-INF/views -->
     <bean id="internalResourceResolver"
         class="org.springframework.web.servlet.view.InternalResourceViewResolver">
         <property name="prefix" value="/WEB-INF/views/" />
         <property name="suffix" value=".jsp" />
         <property name="order" value="#{beanNameViewResolver.order+1}" />
     </bean>
</beans>
```

9. Abrir y estudiar la vista jsp "index.jsp"

```
<div class="container">
  <div class="jumbotron">
        <h1>ContentNegotiatingViewResolver!</h1>
        <p>El ContentNegotiatingViewResolver selecciona el View Resolver más apropiado para
manejar el request comparando el "media type" (conocido también como Content-Type) para
seleccionar la vista adecuada según el "media type" enviado en la URL (como extensión)</p>
        <p>Si el request URL es con extensión .xls, entonces se seleccionará la vista
ExcelView de la lista DefaultViews independientemente del nombre de la vista retornada.</p>
   <p>
       <a class="btn btn-primary btn-lg" href="cuentas.html" role="button">Ver en HTML</a>
       <a class="btn btn-success btn-lg" href="cuentas.xls" role="button">Exportar a Excel</a>
   </p>
  </div>
</div>
```

Otro punto importante a tener en cuenta, es tener el pom.xml con las dependencias maven de poi:

```
ETC…
<!-- Excel View support -->
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>3.12</version>
</dependency>

<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>3.12</version>
</dependency>
ETC…
```

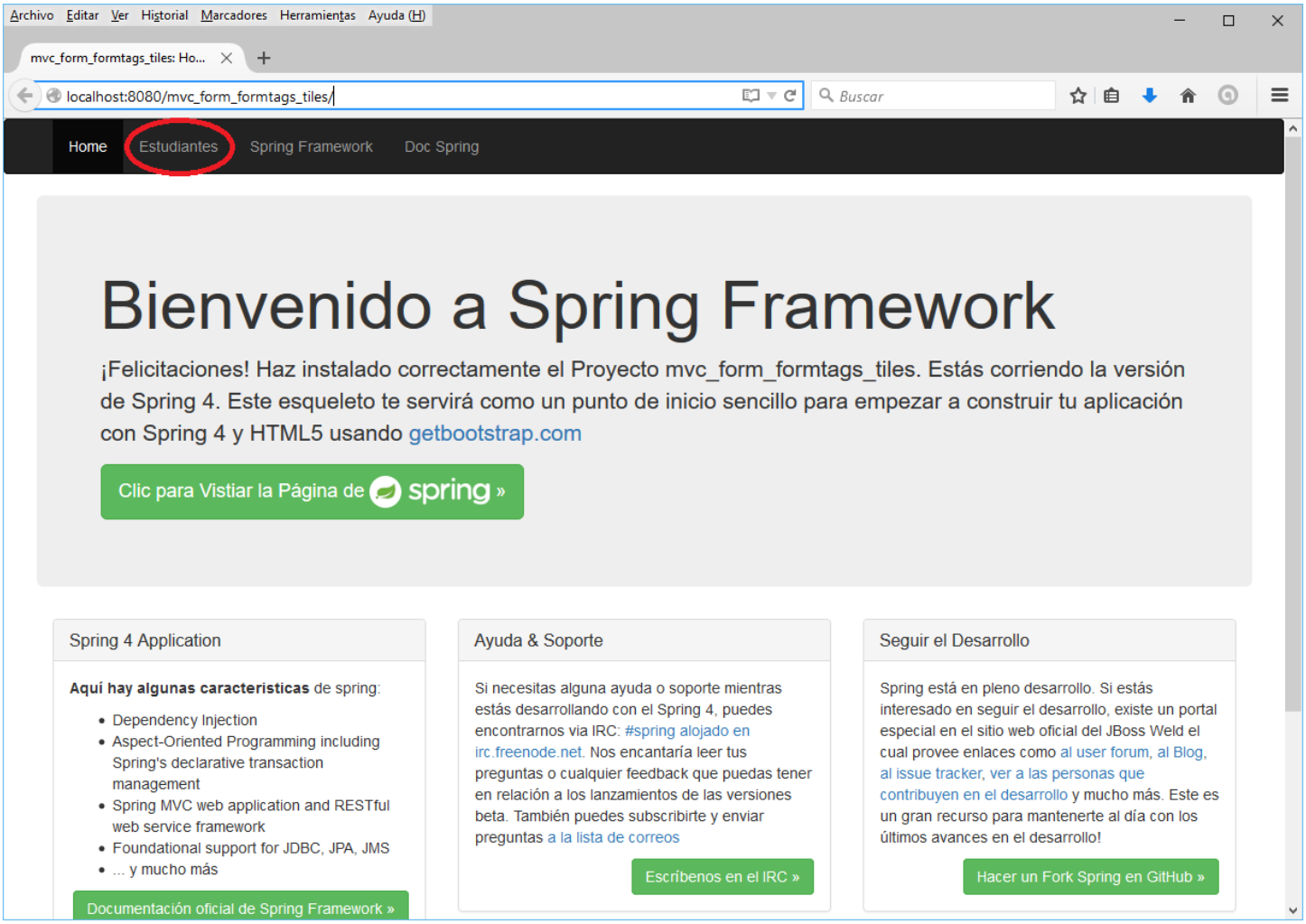
AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

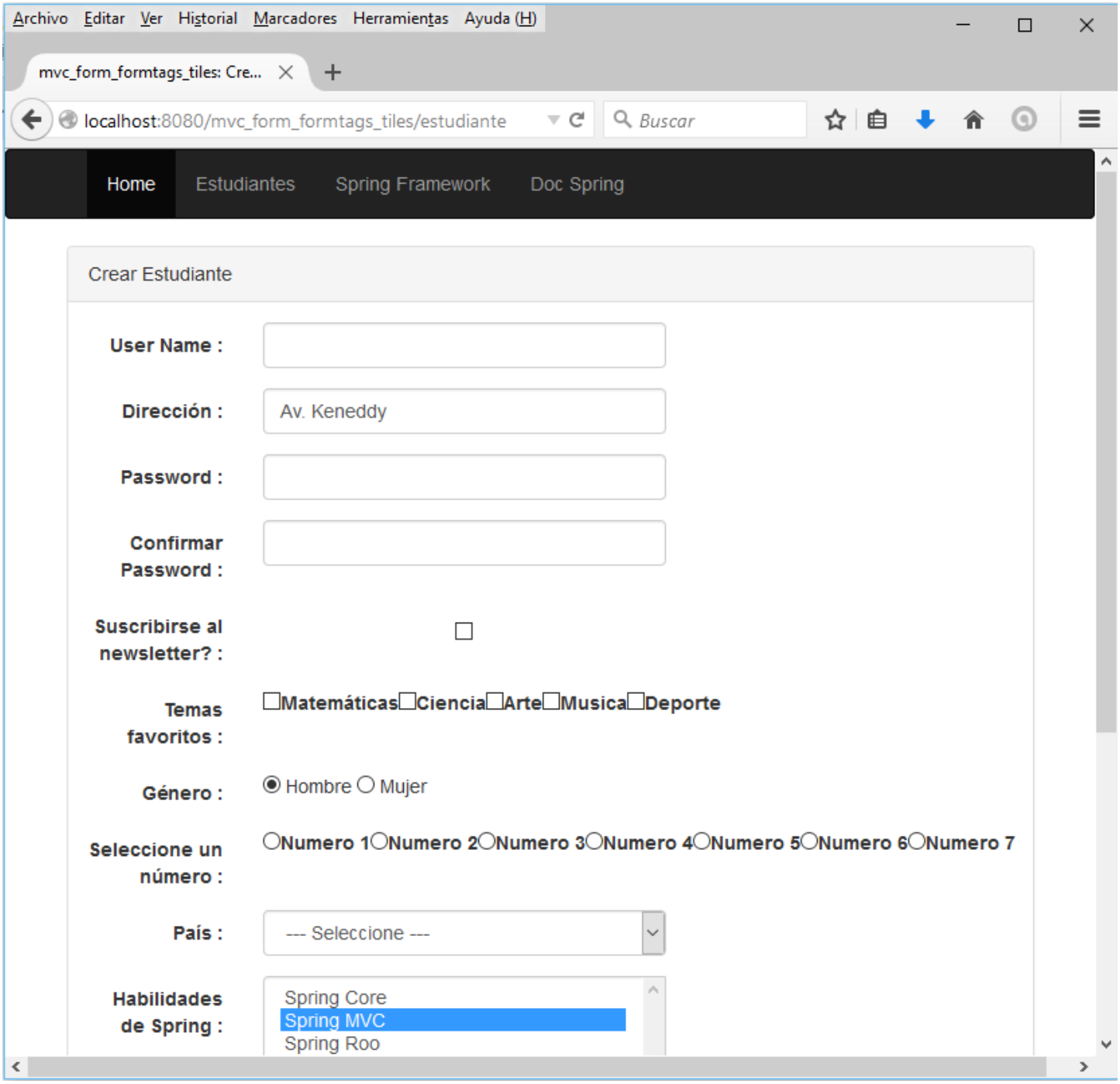
Ejercicio 3: Generar y ejecutar el ejemplo " Usando Layout Tiles como vista"

En este ejercicio, aprenderemos a configurar e implementar un sistema de Layout con Tiles.

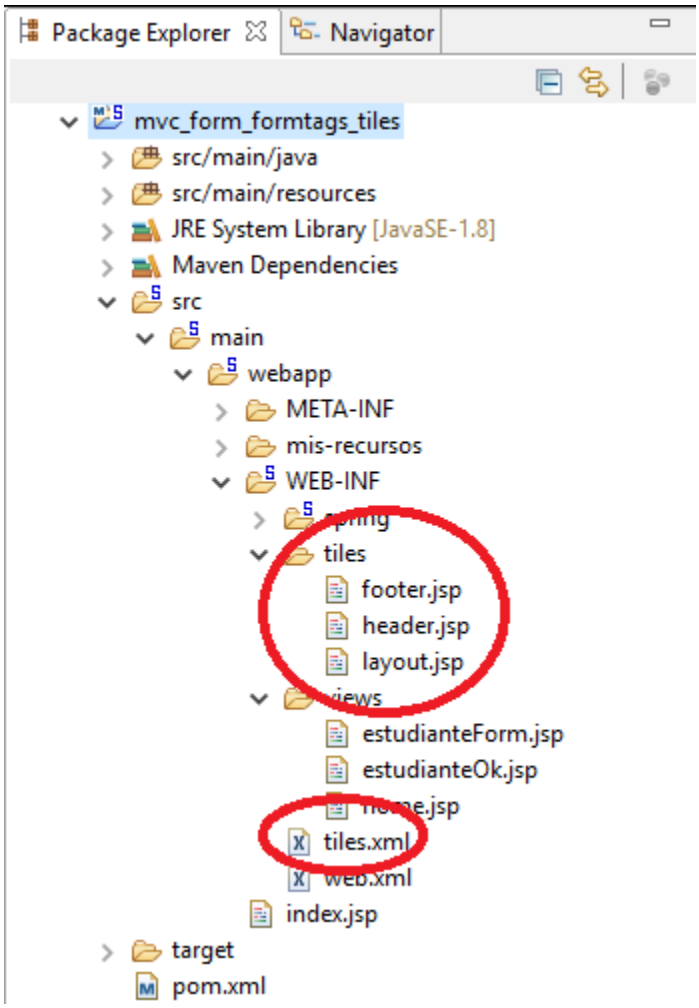
Los layout, también conocidos como plantilla global, almacenan código de etiquetas JSP, HTML5/CSS3 y JavaScript que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página jsp, centralizando los cambios del diseño en un único lugar, luego crearemos las plantillas clientes que usen el diseño definido en esa plantilla global.

1. Clic derecho sobre el proyecto **Run As->Maven Clean** y **Run As->Maven Install**
2. Clic derecho **Maven->Update Project…**
3. Ejecutamos la aplicación: clic derecho sobre **mvc_form_formtags_tiles-> Run As on Server**
4. Clic en **Estudiante** (menú superior)


Archivo Editar Ver Historial Marcadores Herramientas Ayuda (H)
mvc_form_formtags_tiles: Ho...
localhost:8080/mvc_form_formtags_tiles/
Buscar
Home
Estudiantes
Spring Framework
Doc Spring
Bienvenido a Spring Framework
¡Felicitaciones! Haz instalado correctamente el Proyecto mvc_form_formtags_tiles. Estás corriendo la versión de Spring 4. Este esqueleto te servirá como un punto de inicio sencillo para empezar a construir tu aplicación con Spring 4 y HTML5 usando getbootstrap.com
Clic para Vistiar la Página de spring »
Spring 4 Application
Aquí hay algunas caracteristicas de spring:
Dependency Injection
Aspect-Oriented Programming including Spring's declarative transaction management
Spring MVC web application and RESTful web service framework
Foundational support for JDBC, JPA, JMS
... y mucho más
Documentación oficial de Spring Framework »
Ayuda & Soporte
Si necesitas alguna ayuda o soporte mientras estás desarrollando con el Spring 4, puedes encontrarnos via IRC: #spring alojado en irc.freenode.net. Nos encantaría leer tus preguntas o cualquier feedback que puedas tener en relación a los lanzamientos de las versiones beta. También puedes subscribirte y enviar preguntas a la lista de correos
Escríbenos en el IRC »
Seguir el Desarrollo
Spring está en pleno desarrollo. Si estás interesado en seguir el desarrollo, existe un portal especial en el sitio web oficial del JBoss Weld el cual provee enlaces como al user forum, al Blog, al issue tracker, ver a las personas que contribuyen en el desarrollo y mucho más. Este es un gran recurso para mantenerte al día con los últimos avances en el desarrollo!
Hacer un Fork Spring en GitHub »


Archivo Editar Ver Historial Marcadores Herramientas Ayuda (H)
mvc_form_formtags_tiles: Cre...
localhost:8080/mvc_form_formtags_tiles/estudiante
Buscar
Home
Estudiantes
Spring Framework
Doc Spring
Crear Estudiante
User Name :
Dirección :
Av. Keneddy
Password :
Confirmar Password :
Suscribirse al newsletter? :
Temas favoritos :
Matemáticas Ciencia Arte Musica Deporte
Género :
Hombre Mujer
Seleccione un número :
Numero 1 Numero 2 Numero 3 Numero 4 Numero 5 Numero 6 Numero 7
País :
--- Seleccione ---
Habilidades de Spring :
Spring Core
Spring MVC
Spring Roo

5. Estudiar el proyecto.



6. Abrir y estudiar archivo pom.xml. Hay que tener las dependencias relacionadas con Tiles:

ETC…

```
<!-- Apache Tiles -->
<dependency>
    <groupId>org.apache.tiles</groupId>
    <artifactId>tiles-api</artifactId>
    <version>3.0.5</version>
    <scope>compile</scope>
</dependency>
<dependency>
    <groupId>org.apache.tiles</groupId>
    <artifactId>tiles-core</artifactId>
    <version>3.0.5</version>
    <scope>compile</scope>
</dependency>
<dependency>
    <groupId>org.apache.tiles</groupId>
    <artifactId>tiles-jsp</artifactId>
    <version>3.0.5</version>
    <scope>compile</scope>
</dependency>
<dependency>
    <groupId>org.apache.tiles</groupId>
    <artifactId>tiles-servlet</artifactId>
    <version>3.0.5</version>
    <scope>compile</scope>
</dependency>
<dependency>
    <groupId>org.apache.tiles</groupId>
    <artifactId>tiles-template</artifactId>
    <version>3.0.5</version>
    <scope>compile</scope>
</dependency>
<dependency>
    <groupId>org.apache.tiles</groupId>
    <artifactId>tiles-el</artifactId>
    <version>3.0.5</version>
    <scope>compile</scope>
</dependency>
```

ETC…

7. Ahora estudiemos el archivo de configuración de spring **servlet-context.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">
    <!-- Escanea o busca en el package base de la aplicación clases beans anotados con
@Components, @Controller, @Service, @Repository -->
    <context:component-scan base-package="com.bolsadeideas.ejemplos" />

    <!-- Habilita la anotacion de Spring MVC @Controller -->
    <mvc:annotation-driven />

    <!-- Controla las peticiones HTTP GET /recursos/** para los recursos estáticos
        (ej: imagenes), estos se almacenan en el directorio ${webappRoot}/mis-recursos -->
    <mvc:resources mapping="/recursos/**" location="/mis-recursos/" cache-period="10000" />

    <!-- La url /home.html inmediatamente esta mapeada hacia la vista home.jsp -->
    <mvc:view-controller path="/home" view-name="home" />

    <!-- Para resolver vistas Tiles -->
    <bean id="viewResolver"
        class="org.springframework.web.servlet.view.UrlBasedViewResolver">
        <property name="order" value="1" />
        <property name="viewClass"
                  value="org.springframework.web.servlet.view.tiles3.TilesView"/>
    </bean>

    <!-- Configuracion del layout Tiles -->
    <bean id="tilesConfigurer"
        class="org.springframework.web.servlet.view.tiles3.TilesConfigurer">
        <property name="definitions">
            <list>
                <value>/WEB-INF/tiles.xml</value>
            </list>
        </property>
    </bean>

   <!-- Resuelve la ubicacion de las vistas .jsp de @Controllers en la ruta /WEB-INF/views -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="order" value="2" />
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
    </bean>
</beans>
```

8. Abrir y estudiar "/WEB-INF/tiles.xml"

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE tiles-definitions PUBLIC
       "-//Apache Software Foundation//DTD Tiles Configuration 3.0//EN"
       "http://tiles.apache.org/dtds/tiles-config_3_0.dtd">
<tiles-definitions>

     <!-- La definición principal del layout o root -->
    <definition name="definicion.principal" template="/WEB-INF/tiles/layout.jsp">
      <put-attribute name="title" value="" />
      <put-attribute name="header" value="/WEB-INF/tiles/header.jsp" />
      <put-attribute name="body" value="" />
      <put-attribute name="footer" value="/WEB-INF/tiles/footer.jsp" />
    </definition>

    <!-- Esta es una definición Tiles hija que extiende de la principal/padre
"definicion.principal" sobreescribe los componentes "title" y "body" -->
    <definition name="estudianteForm" extends="definicion.principal">
      <put-attribute name="title" value="mvc_form_formtags_tiles: Crear Estudiante" />
      <put-attribute name="body" value="/WEB-INF/views/estudianteForm.jsp" />
    </definition>

    <!-- Esta es otra definición hija que extiende de la principal "definicion.principal"
         sobreescribe los componentes "title" y "body" -->
    <definition name="estudianteOk" extends="definicion.principal">
      <put-attribute name="title" value="mvc_form_formtags_tiles: Datos enviados con éxito" />
      <put-attribute name="body" value="/WEB-INF/views/estudianteOk.jsp" />
    </definition>

    <definition name="home" extends="definicion.principal">
        <put-attribute name="title" value="mvc_form_formtags_tiles: Home" />
        <put-attribute name="body" value="/WEB-INF/views/home.jsp" />
    </definition>
</tiles-definitions>
```

Las vista hay que manejarlas y definirlas en el tiles.xml, como se ilustra arriba, observamos que hay dos vistas con su determinado nombre, **estudianteForm** y **estudianteOk**, mapeadas a los JSP **estudianteForm.jsp** y **estudianteOk.jsp** respectivamente, que extienden de la definición principal para ser incluida dentro del **body** del Layout.

Luego en el controlador **EstudianteController** se cargan las vistas haciendo referencia a lo definido en el tiles.xml, aparecen marcada en color rojo, el nombre de la vista debe coincidir con el nombre de la vista definida en el XML de tiles, ejemplo:

Etc…

```
	// Metodo handler formulario, para crear al estudiante
	@RequestMapping(method = RequestMethod.GET)
	public String getCreateForm(ModelMap model) {

		Estudiante estudiante = new Estudiante();

		estudiante.setDireccion("Av. Keneddy");
		estudiante.setTemas(new String[] { "Matematicas" });
		estudiante.setGenero("H");
		estudiante.setExperienciaSpring("Spring MVC");
		estudiante.setValorSecreto("Algún valor oculto");

		// Guardamos al objeto comando como "estudianteCommand" el cual
		// será accesible en la vista estudianteForm.
		model.addAttribute("estudianteCommand", estudiante);

		// retornamos la vista tiles estudianteForm
		return "estudianteForm";
	}

	// Metodo handler que procesa el envio de datos del form
	@RequestMapping(method = RequestMethod.POST)
	public String processSubmit(
			@ModelAttribute("estudianteCommand") Estudiante estudiante,
			BindingResult result, SessionStatus status) {

		// Validamos
		estudianteValidator.validate(estudiante, result);

		if (result.hasErrors()) {
			// Si ocurre un error en la validación retornamos al formulario
			// con los mensajes de error.
			return "estudianteForm";
		} else {
			status.setComplete();
			// Si pasa bien la validacion, retornamos vista tiles "estudianteOk"
			return "estudianteOk";
		}
	}
```

Etc…

9. Abrir y estudiar archivo de Layout Tiles "tiles/layout.jsp"

```
<%@ taglib uri="http://tiles.apache.org/tags-tiles" prefix="tiles"%>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title><tiles:insertAttribute name="title" ignore="true" /></title>
<!-- Bootstrap -->
<link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
<style>
.error-campo {
    color: #ff0000;
}
</style>
</head>
<body>
<tiles:insertAttribute name="header" />
<div class="container">
     <tiles:insertAttribute name="body" />
     <hr>
     <tiles:insertAttribute name="footer" />
</div>
</body>
</html>
```

10. Abrir y estudiar archivo "tiles/header.jsp"

```
<nav class="navbar navbar-inverse">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse"
                data-target=".navbar-collapse">
                <span class="icon-bar"></span> <span class="icon-bar"></span> <span
                    class="icon-bar"></span>
            </button>
        </div>
        <div class="collapse navbar-collapse">
            <ul id="barramenu" class="nav navbar-nav">
                <li class="active"><a href="<%=request.getContextPath()%>/home">Home</a></li>
                <li><a href="<%=request.getContextPath()%>/estudiante">Estudiantes</a></li>
                <li><a href="http://projects.spring.io/spring-framework/">
                        Spring Framework</a>
                </li>
                <li><a href="http://docs.spring.io/spring/docs/current/javadoc-api/">
                        Doc Spring</a>
                </li>
            </ul>
        </div>
        <!--/.nav-collapse -->
    </div>
</nav>
```

11. Abrir y estudiar archivo "tiles/footer.jsp"

```
<footer style="clear: both;">
    <img src="/mvc_form_formtags_tiles/recursos/images/spring.png" alt="Spring logo" />
    <p>
        Powered by <a href="http://projects.spring.io/spring-framework/">Spring Framework</a>
        y <a href="https://tiles.apache.org/">Tiles 3</a>.
    </p>
    <p>
        This project was generated from a Spring Tools maintained by the Spring team.<br />
         © 2015 Pivotal Software, Inc. All Rights Reserved. Terms of Use and Privacy.<br />
    </p>
</footer>
```

Ejercicio 4: Generar y ejecutar el ejemplo "Multi-lenguaje"

En este ejercicio, aprenderemos a configurar e implementar un sistema de multi-idioma en nuestros proyectos con Spring. La mayoría de las plataformas más conocidas y utilizadas hoy en días, como lo son Google, Facebook, Twitter, Youtube y Flickr, por nombrar unas pocas, están disponibles en varios idiomas. Este tipo de característica es la clave del éxito de cualquier proyecto que se propone como meta una estrategia global, asegurando que los usuarios de todo el mundo serán capaces de navegar y utilizar la aplicación en su propio idioma, de esta forma el proyecto y sus gestores serán capaces de captura un mercado de usuarios mucho más amplio y garantizar una experiencia en común para todos ellos.

Si queremos implementar un proyecto web que apunte hacia un mercado globalizado, necesariamente necesitamos contar con una herramienta que nos resuelva el problema.

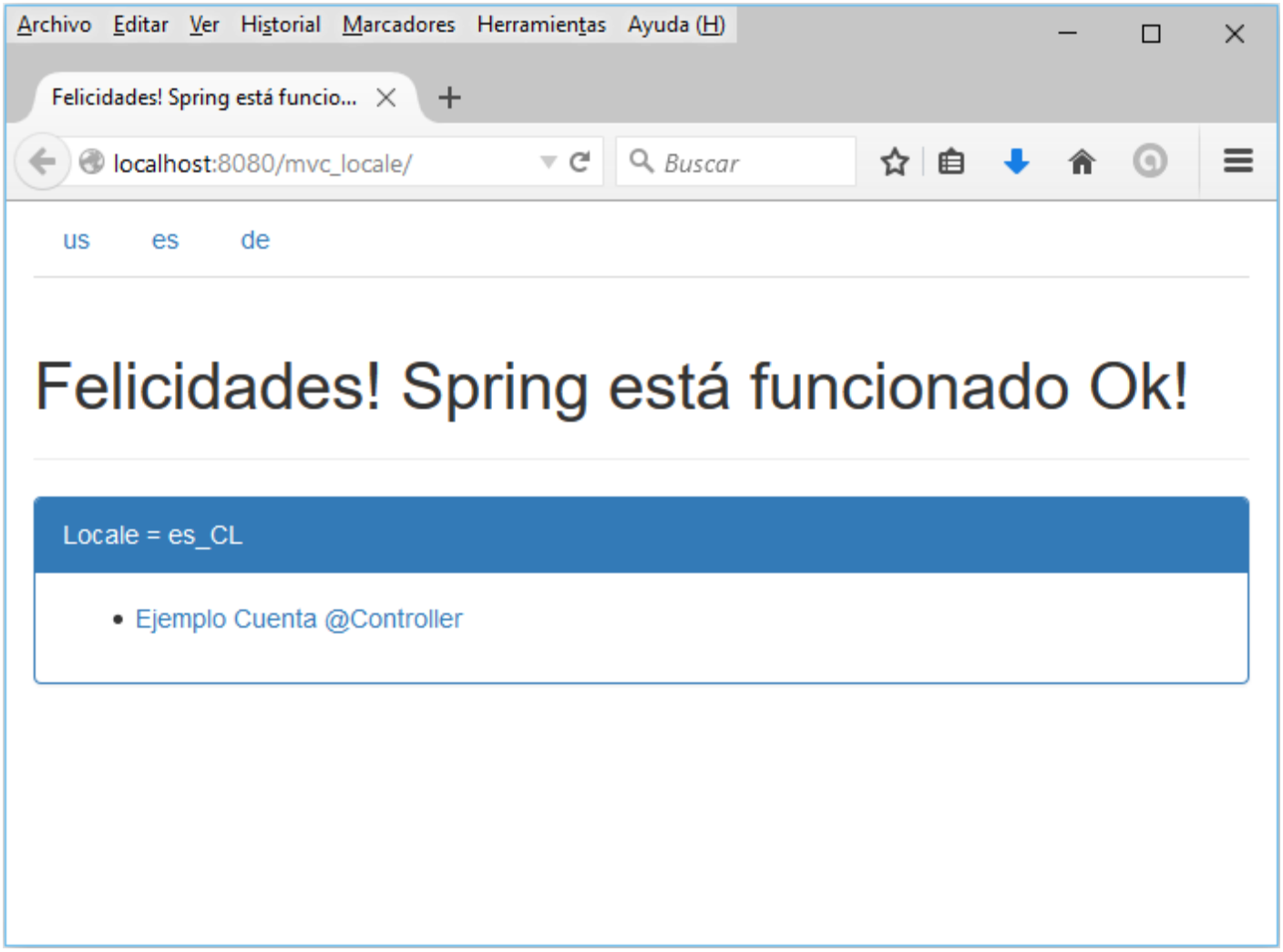
> **Afortunadamente**, hay un componente en Spring Framework, llamado **Locale**, que nos permite resolver el tema e implementar de forma muy sencilla un soporte multi-idioma a nuestra aplicación.

Aprenderemos y veremos de forma sencilla cómo configurar el interceptor *org.springframework.web.servlet.i18n.LocaleChangeInterceptor* para interceptar el parámetro del request pasado por URL 'locale' ej. /?locale=de

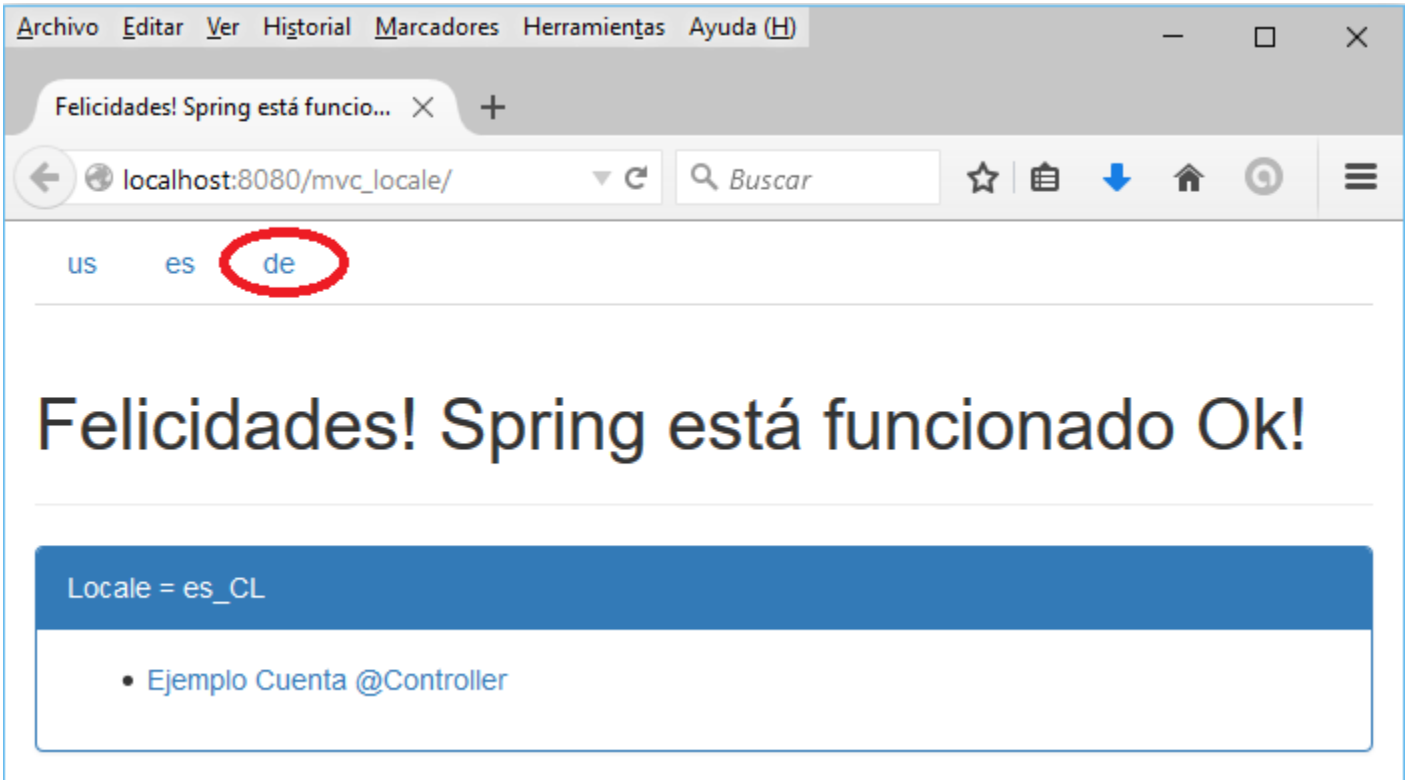
Además ver la configuración de *org.springframework.web.servlet.i18n.CookieLocaleResolver* para guardar el cambio del locale en una Cookie.

Usaremos los archivos mensajes.properties y mensajes.properties_<locale> para guarder nuestras traducciones en cada idioma.

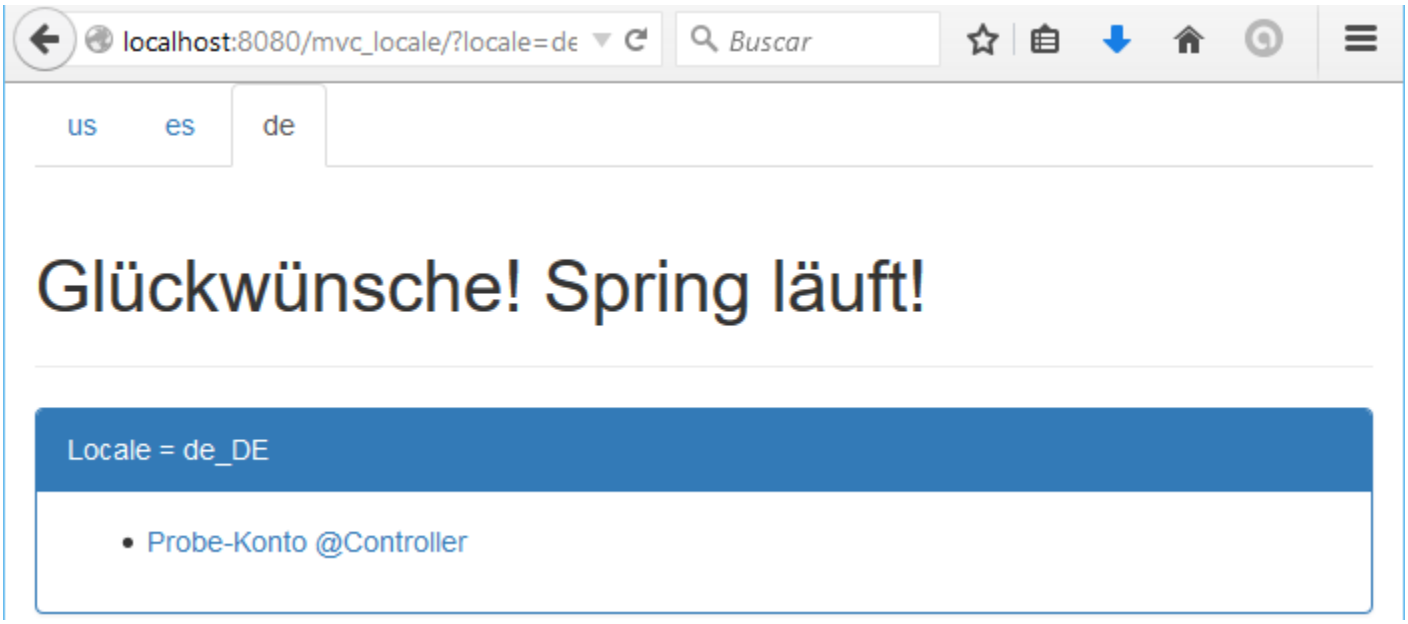
1. Clic derecho sobre el proyecto **Run As->Maven Clean** y **Run As->Maven Install**
2. Clic derecho **Maven->Update Project…**
3. Ejecutamos la aplicación: clic derecho sobre **mvc_locale-> Run As on Server**



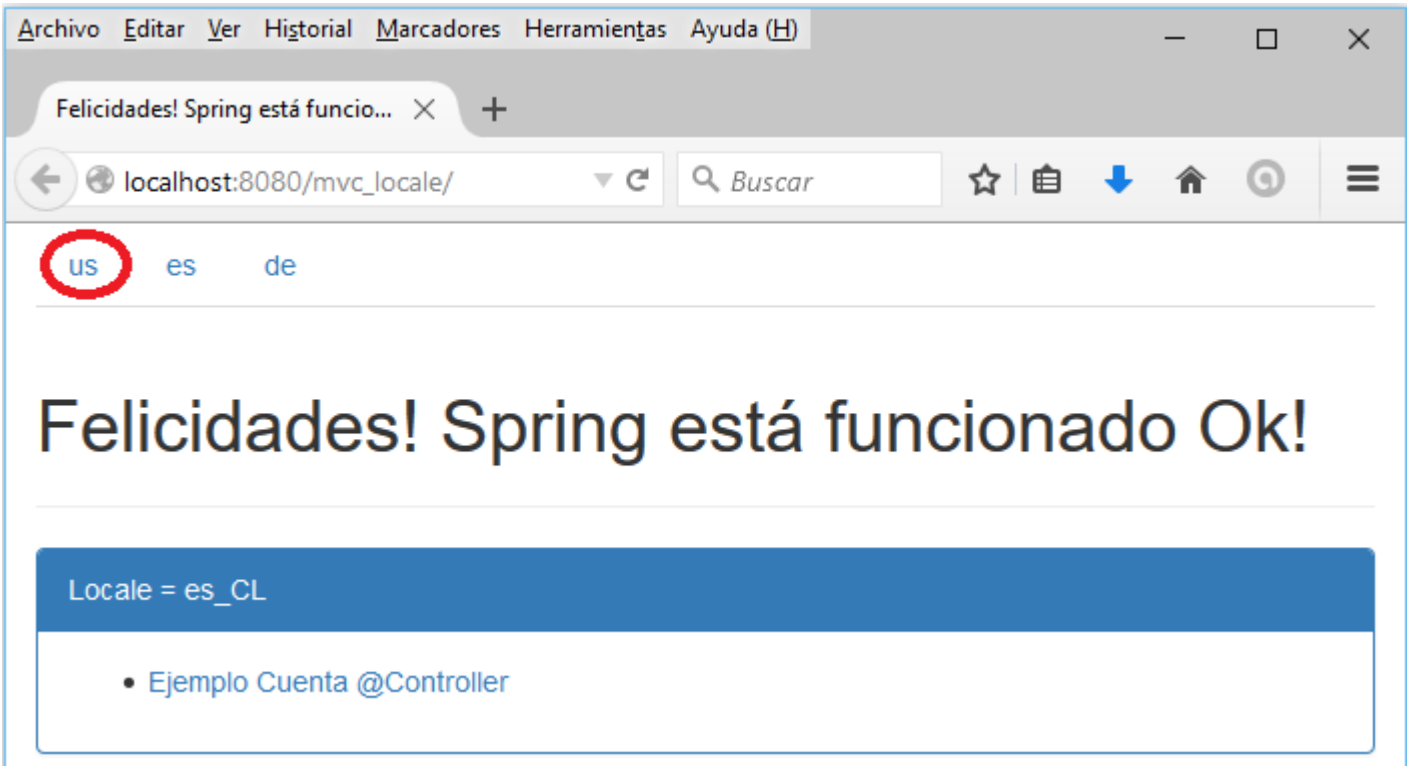
4. Seleccionemos un locale o idioma



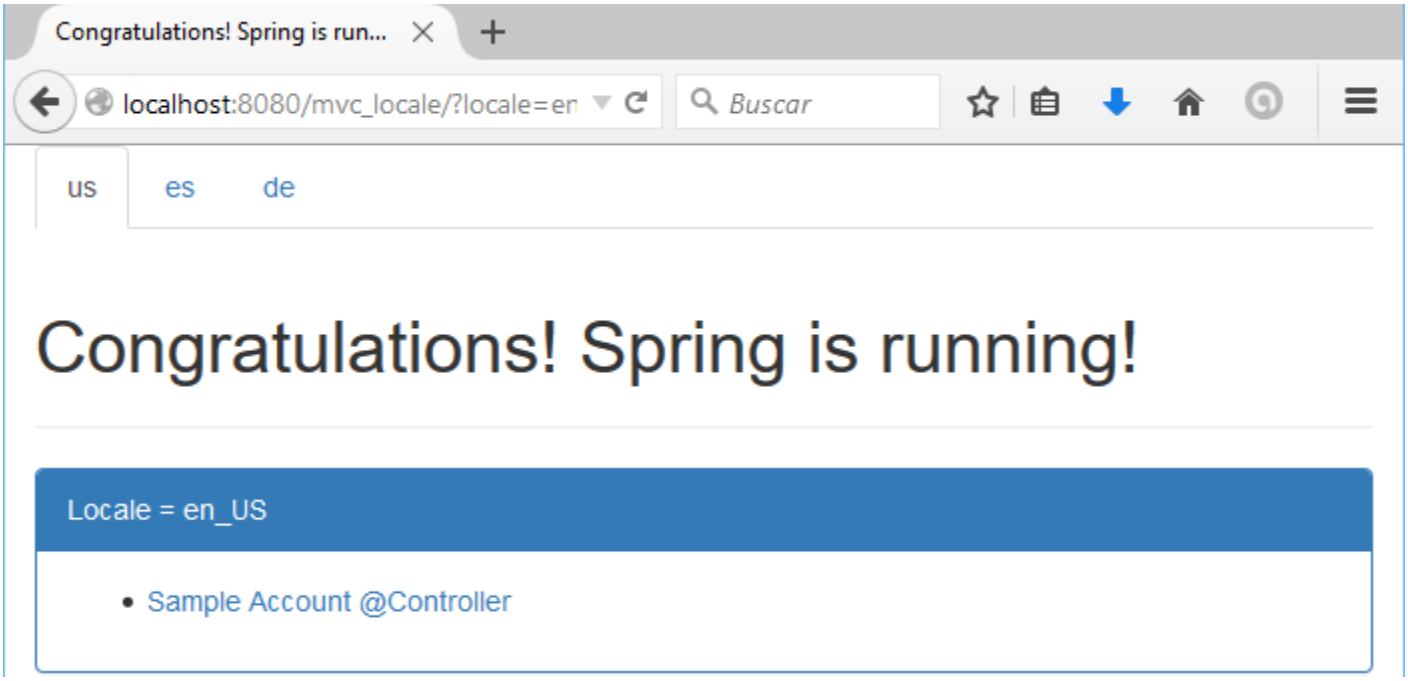
5. Observamos que cambia todo al alemán



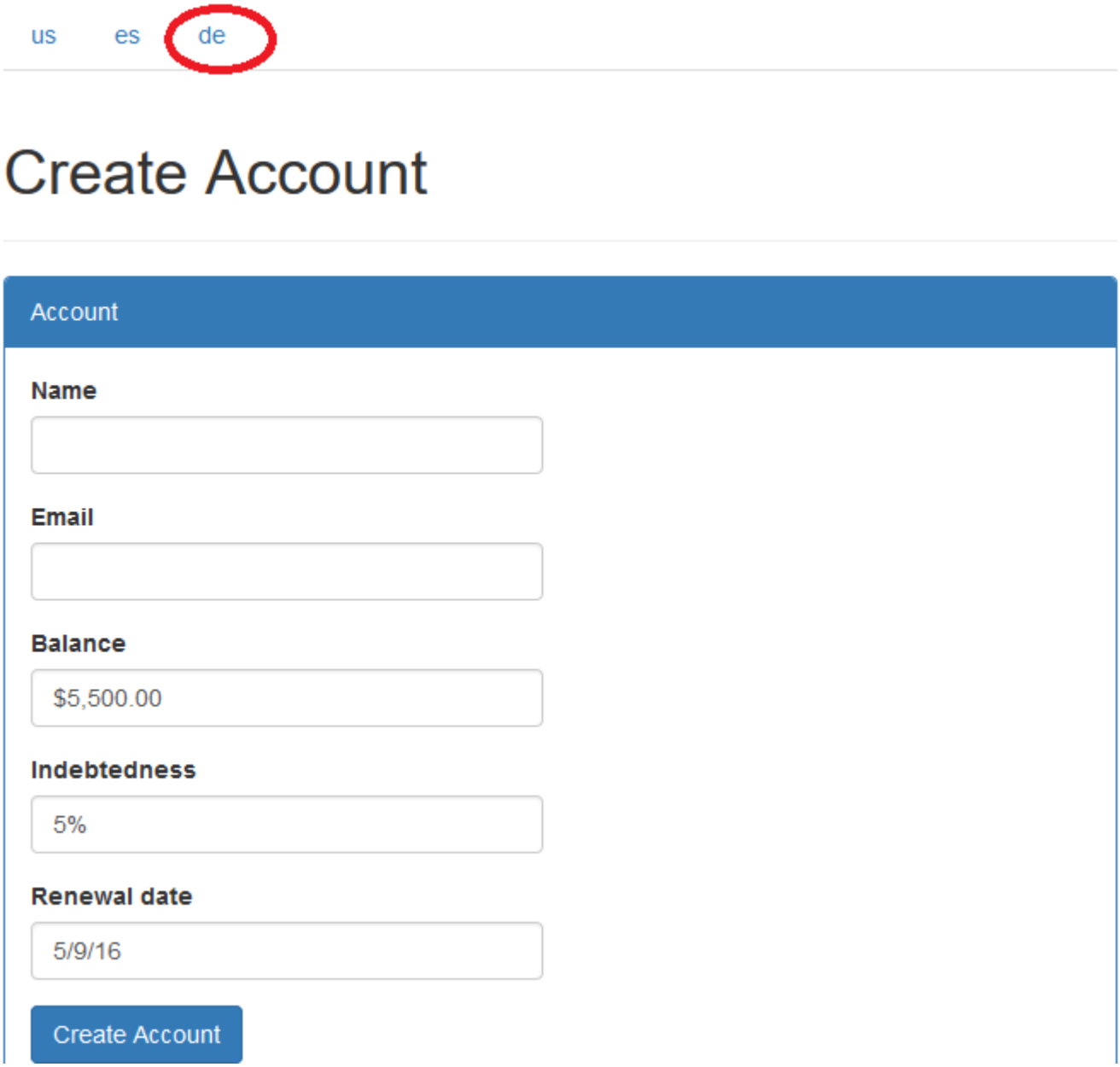
6. Ahora un clic en us



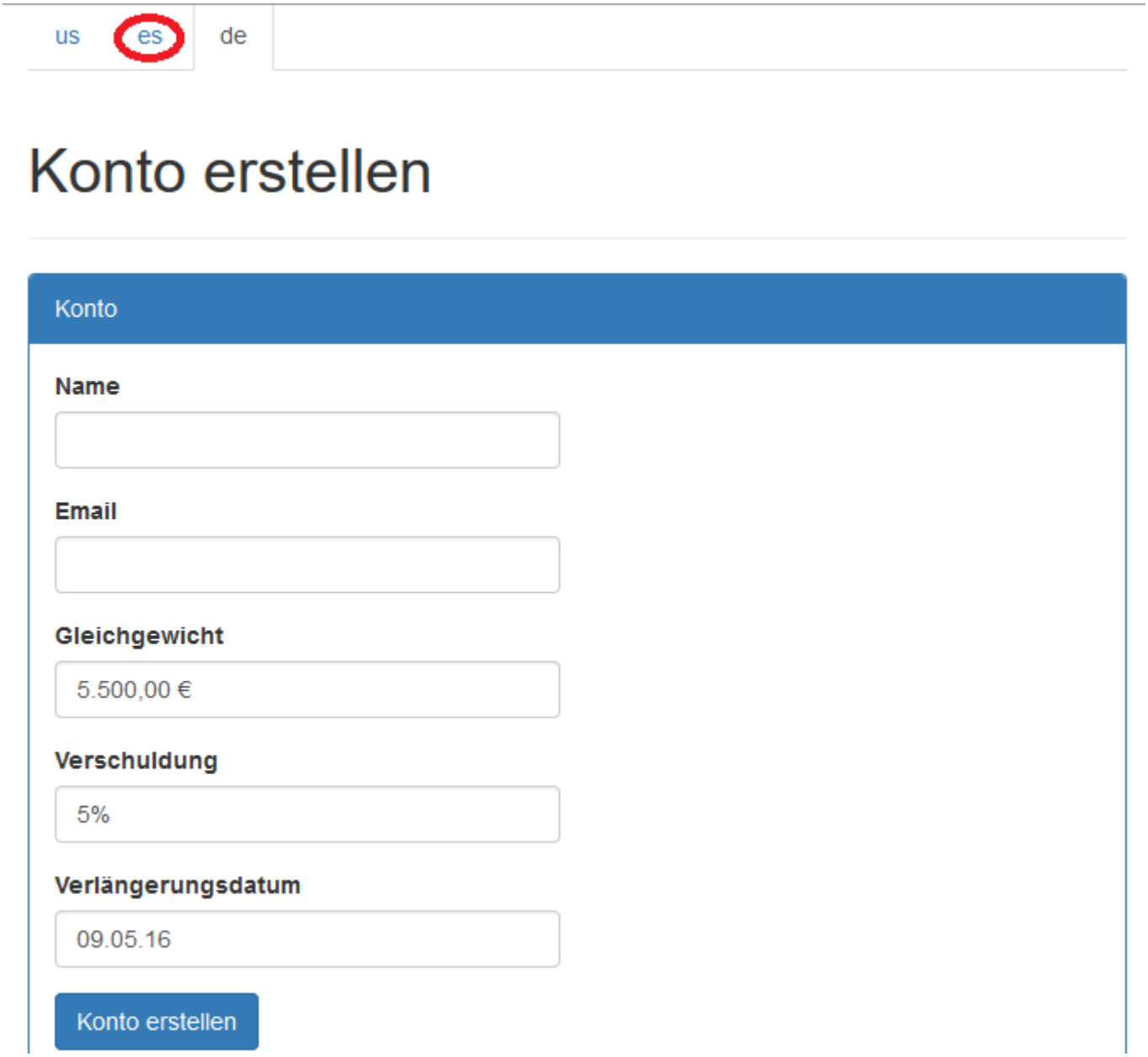
7. Cambia todos los textos al inglés

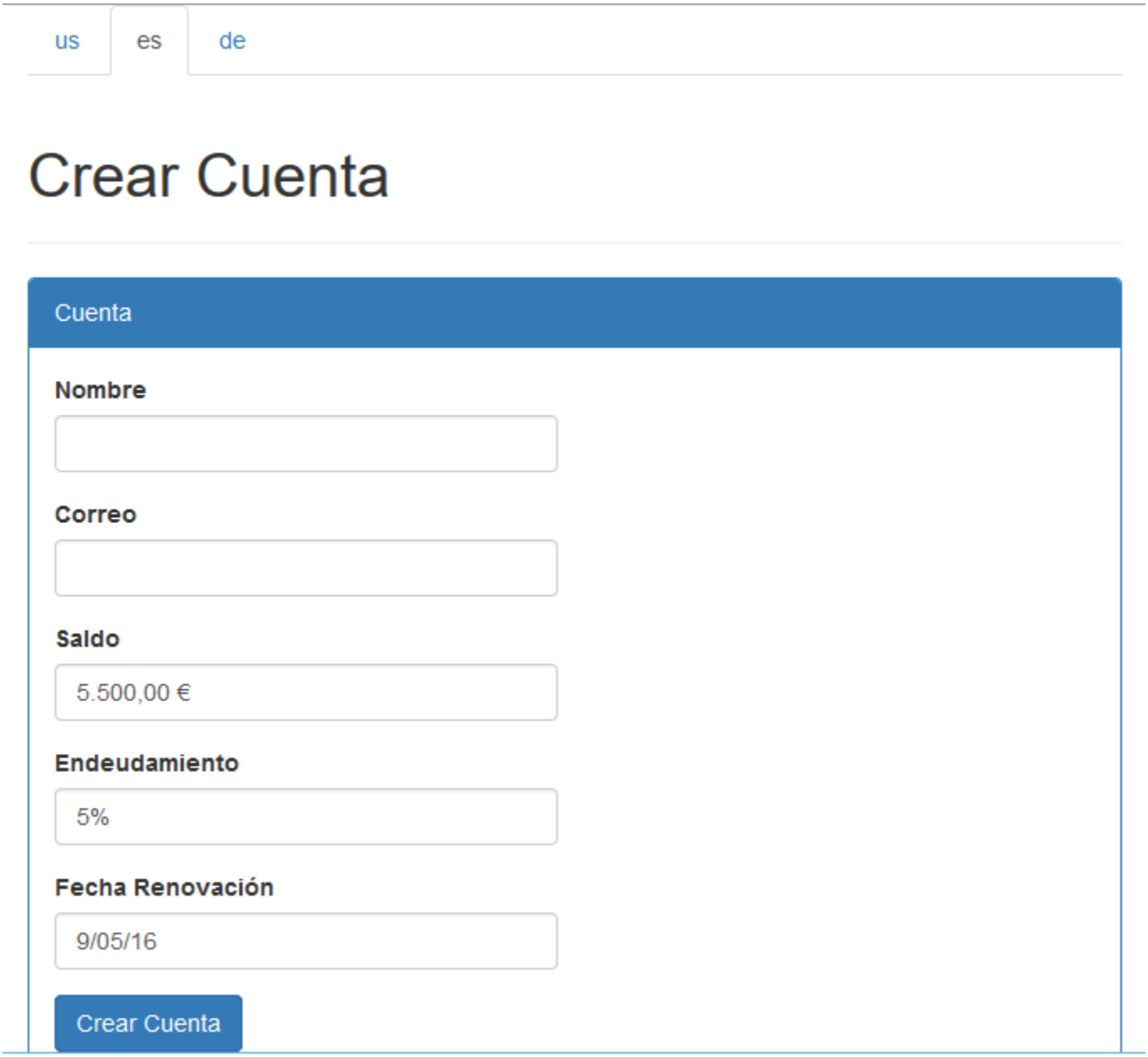


8. Ahora seleccionamos con un clic **Sample Account @Controller**
9. Apaprece el formulario con los textos y Local en inglés, ahora un clic en de, para traducir al alemán.

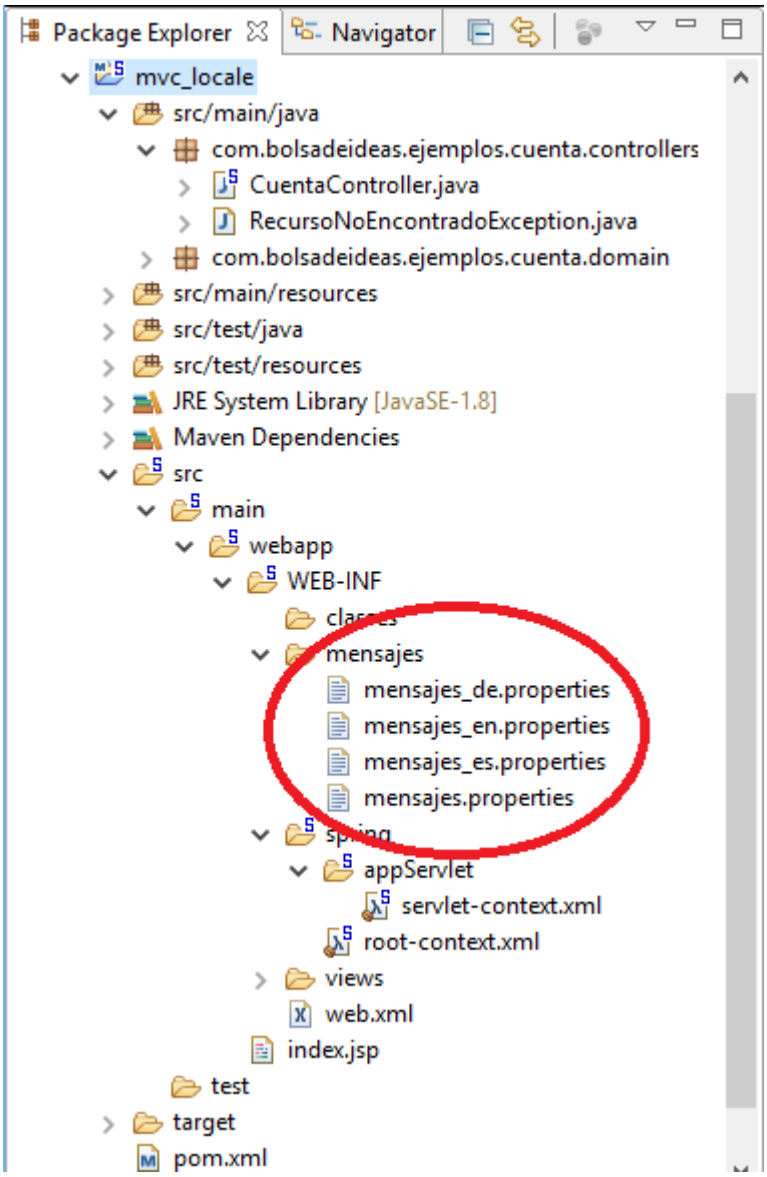


10. Aparece todo en alemán
11. Clic en español en **es**




us
es
de
Crear Cuenta
Cuenta
Nombre
Correo
Saldo
5.500,00 €
Endeudamiento
5%
Fecha Renovación
9/05/16
Crear Cuenta

12. Estudiar el proyecto.



13. Abrir y estudiar la clase CuentaController, ningún cambio para el soporte multilenguaje.

```
package com.bolsadeideas.ejemplos.cuenta.controllers;

import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.bolsadeideas.ejemplos.cuenta.domain.Cuenta;

@Controller
@RequestMapping(value="/cuenta")
public class CuentaController {

  private Map<Long, Cuenta> cuentas = new ConcurrentHashMap<Long, Cuenta>();

  // Metodo handler formulario, para crear la cuenta
  @RequestMapping(method=RequestMethod.GET)
  public String crearCuentaForm(Model model) {
    // La instancia de "Cuenta" es creada y guardada como atributo "cuenta"
    // en el objeto "model", el cual es accesible desde
    // la vista, "cuenta/crearForm.jsp".
    model.addAttribute("cuenta", new Cuenta());
    return "cuenta/crearForm";
  }

  // Metodo handler que procesa el envio de datos del form
  @RequestMapping(method=RequestMethod.POST)
  public String crearCuenta(Cuenta cuenta, BindingResult result) {

    // Si ocurre un error en la validación aparecera el formulario
    // "cuenta/crearForm.jsp" con los mensajes.
    if (result.hasErrors()) {
      return "cuenta/crearForm";
    }

    // Si todo está bien, crea la cuenta y la agrega a la lista
    // "cuentas", luego redirige hacia el detalle /cuenta/{id},
    // controlado por el método handler "verDetalle(..) de más abajo.
    this.cuentas.put(cuenta.asignarId(), cuenta);
    return "redirect:/cuenta/" + cuenta.getId();

  }
```

```
  @RequestMapping(value="{id}", method=RequestMethod.GET)
  public String verDetalle(@PathVariable Long id, Model model) {
    Cuenta cuenta = this.cuentas.get(id);
    if (cuenta == null) {
      throw new RecursoNoEncontradoException(id);
    }
    model.addAttribute("cuenta", cuenta);
    // return "cuenta/detalle"; // Muestra el detalle de la cuenta en formato formulario
    return "cuenta/detalle2";  // Muestra el detalle de la cuenta en formato tabla html
  }

}
```

14. Ahora estudiemos el archivo de configuración de spring **root-context**.

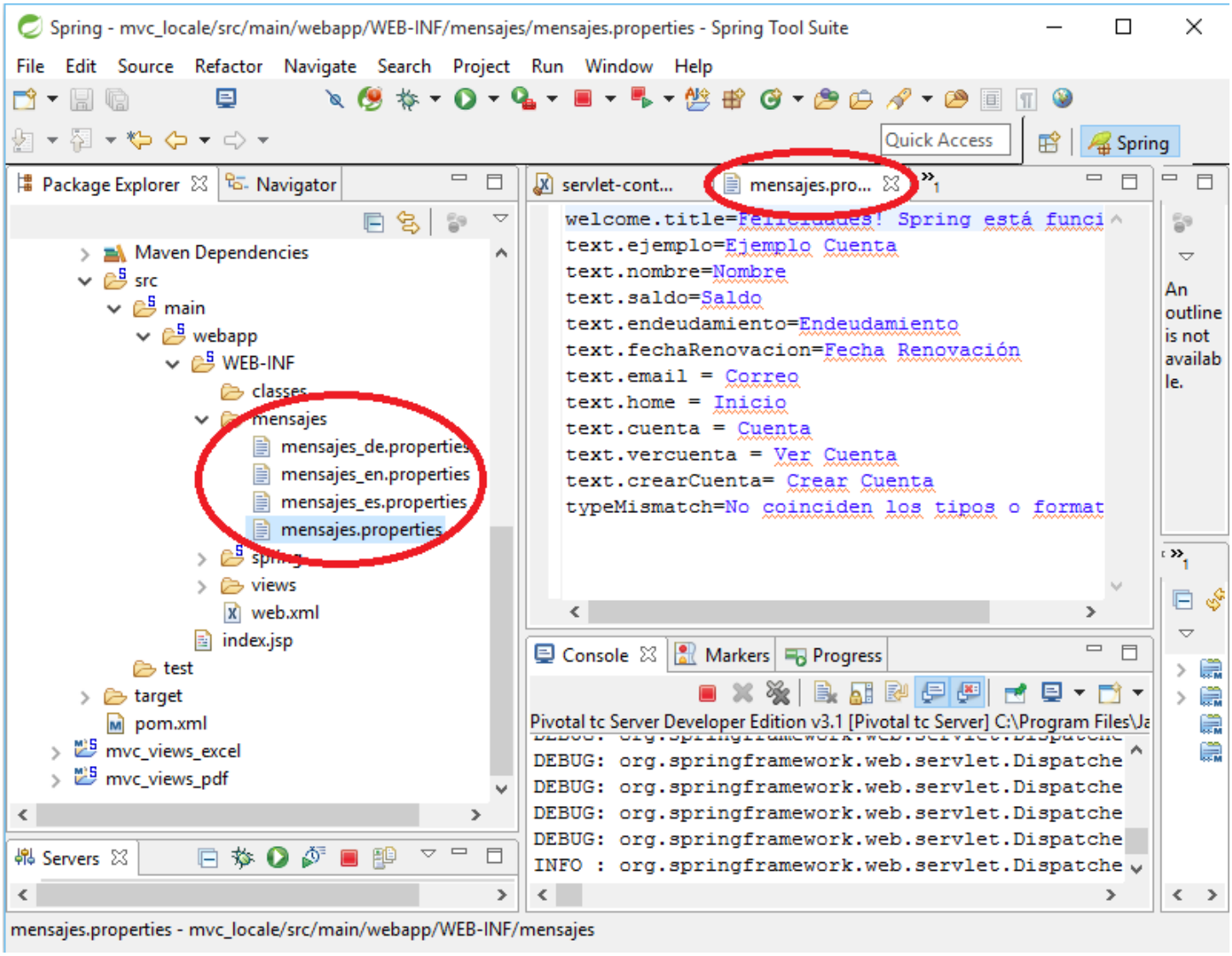
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Root Context: defines shared resources visible to all other web components -->
    <!-- Mensajes de idiomas -->
    <bean id="messageSource"
        class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
        <property name="basename" value="/WEB-INF/mensajes/mensajes" />
        <property name="cacheSeconds" value="0" />
    </bean>
</beans>
```

15. Estudiar el archivo de configuración de spring **servlet-context**.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- Escanea o busca en el package base de la aplicación clases beans anotados
        con @Components, @Controller, @Service, @Repository -->
    <context:component-scan base-package="com.bolsadeideas.ejemplos.cuenta" />

    <!-- Habilita la anotacion de Spring MVC @Controller -->
    <mvc:annotation-driven />

    <!-- La url /bienvenidos inmediatamente esta mapeada hacia la vista bienvenidos -->
    <mvc:view-controller path="/bienvenidos" view-name="bienvenidos" />

    <!-- Interceptores que son aplicados a todos los controladores anotados
        @Controller -->
    <mvc:interceptors>
        <!-- Cambiamos el local cuando envia un parametro del request 'locale'
            e.j. /?locale=es -->
        <bean class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor" />
    </mvc:interceptors>

    <!-- Guardamos los cambios del local usando cookie -->
    <bean id="localeResolver"
        class="org.springframework.web.servlet.i18n.CookieLocaleResolver" />

    <!-- View Resolvers -->
    <!-- Resuelve la ubicion de las vistas .jsp de @Controllers en la ruta /WEB-INF/views -->
    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
    </bean>
</beans>
```

16. Abrir y estudiar archivos mensajes.properties

```
welcome.title=Felicidades! Spring está funcionado Ok!
text.ejemplo=Ejemplo Cuenta
text.nombre=Nombre
text.saldo=Saldo
text.endeudamiento=Endeudamiento
text.fechaRenovacion=Fecha Renovación
text.home = Inicio
text.cuenta = Cuenta
text.vercuenta = Ver Cuenta
text.crearCuenta= Crear Cuenta
typeMismatch=No coinciden los tipos o formato
```



mensajes_en.properties

```
welcome.title=Congratulations! Spring is running!
text.ejemplo=Sample Account
text.nombre=Name
text.saldo=Balance
text.endeudamiento=Indebtedness
text.fechaRenovacion=Renewal date
text.home = Home
text.cuenta = Account
text.vercuenta = View Account
text.crearCuenta= Create Account
typeMismatch=could not be parsed
```

mensajes_de.properties

```
welcome.title=Glückwünsche! Spring läuft!
text.ejemplo=Probe-Konto
text.nombre = Name
text.saldo = Gleichgewicht
text.endeudamiento = Verschuldung
text.fechaRenovacion = Verlängerungsdatum
text.home = nach Hause
text.cuenta = Konto
text.vercuenta = Konto anzeigen
text.crearCuenta= Konto erstellen
typeMismatch=konnte nicht eingelesen werden
```

mensajes_es.properties

```
welcome.title=Felicidades! Spring está funcionado Ok!
text.ejemplo=Ejemplo Cuenta
text.nombre=Nombre
text.saldo=Saldo
text.endeudamiento=Endeudamiento
text.fechaRenovacion=Fecha Renovación
text.home = Inicio
text.cuenta = Cuenta
text.vercuenta = Ver Cuenta
text.crearCuenta= Crear Cuenta
typeMismatch=No coinciden los tipos o formato
```

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

17. Abrir y estudiar archivo web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1">
    <filter>
        <filter-name>characterEncodingFilter</filter-name>
        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
        <init-param>
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
        <init-param>
            <param-name>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>characterEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/root-context.xml</param-value>
    </context-param>
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
    <servlet>
        <servlet-name>mvc_locale</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>mvc_locale</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

18. Abrir y estudiar vista jsp "bienvenidos.jsp"

```
<%@page contentType="text/html;charset=UTF-8"%>
<%@page pageEncoding="UTF-8"%>
<%@ page session="false" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<html>
<head>
   <meta content="text/html; charset=UTF-8" http-equiv="content-type" />
   <title><fmt:message key="welcome.title"/></title>

</head>
<body>
<div class="container">
   <h1>
         <fmt:message key="welcome.title"/>
   </h1>
   <p>
         Locale = ${pageContext.response.locale}
   </p>
   <hr>
   <ul>
         <li> <a href="?locale=en_us">us</a> |  <a href="?locale=es_cl">cl</a> | <a
href="?locale=es_es">es</a> | <a href="?locale=de_de">de</a> </li>
   </ul>
   <ul>
         <li><a href="cuenta"><fmt:message key="text.ejemplo"/> @Controller</a></li>
   </ul>
</div>
</body>
</html>
```

19. Abrir y estudiar archivo "crearForm.jsp"

```
<%@page contentType="text/html;charset=UTF-8"%>
<%@page pageEncoding="UTF-8"%>
<%@ page session="false" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

<html>
<head>
  <meta content="text/html; charset=UTF-8" http-equiv="content-type" />
  <title><fmt:message key="text.crearCuenta"/></title>
</head>
<body>
<div class="container">
  <h3>
    <fmt:message key="text.crearCuenta"/>
  </h3>
  <div class="span-12 last">
    <!-- El objeto del model "cuenta" se crea dentro del metodo "crearCuentaForm(Model model)"
      de la clase controladora "CuentaController". -->
    <form:form modelAttribute="cuenta" action="cuenta" method="post">
        <fieldset>
        <legend><fmt:message key="text.cuenta"/></legend>
        <p>
          <form:label  for="nombre" path="nombre" cssErrorClass="error">
<fmt:message key="text.nombre"/></form:label><br/>
          <form:input path="nombre" /> <form:errors path="nombre" />
        </p>
        <p>
          <form:label for="saldo" path="saldo" cssErrorClass="error">
<fmt:message key="text.saldo"/></form:label><br/>
          <form:input path="saldo" /> <form:errors path="saldo" />
        </p>
        <p>
          <form:label for="nivelEndeudamiento" path="nivelEndeudamiento"
cssErrorClass="error"><fmt:message key="text.endeudamiento"/></form:label><br/>
          <form:input path="nivelEndeudamiento" /> <form:errors path="nivelEndeudamiento" />
        </p>
        <p>
          <form:label for="fechaRenovacion" path="fechaRenovacion"
cssErrorClass="error"><fmt:message key="text.fechaRenovacion"/></form:label><br/>
          <form:input path="fechaRenovacion" /> <form:errors path="fechaRenovacion" />
        </p>
        <p>
          <input type="submit" value="<fmt:message key="text.crearCuenta"/>"/>
        </p>
      </fieldset>
    </form:form>
  </div>
  <hr />
```

```
  <ul>
    <li> <a href="?locale=en_us">us</a> |  <a href="?locale=es_cl">cl</a> | <a
href="?locale=es_es">es</a> | <a href="?locale=de_de">de</a> </li>
  </ul>
</div>
</body>
</html>
```

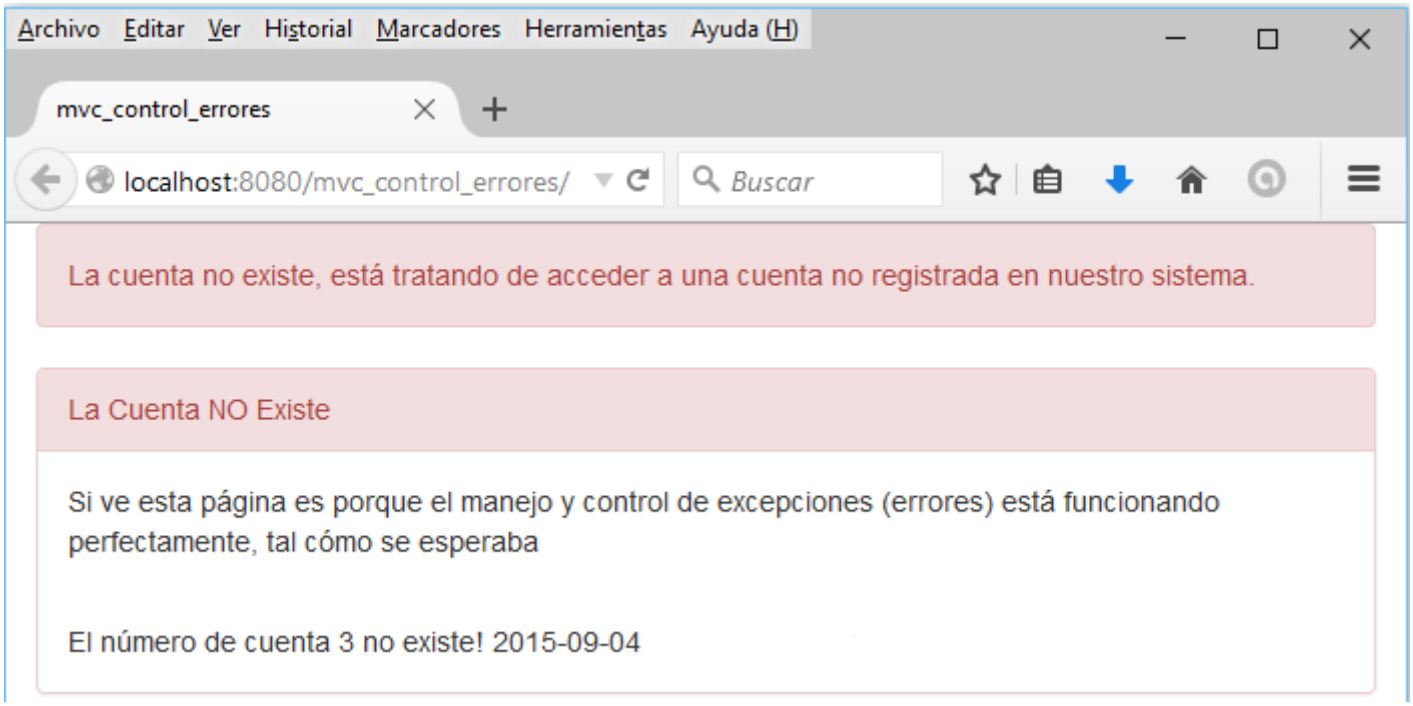
20. Abrir y estudiar archivo "index.jsp" para forzar una redirección por defecto cuando arranca la aplicación.

```
<html>
<body>
<jsp:forward page="bienvenidos"/>
</body>
</html>
```

Ejercicio 5: Generar y ejecutar el ejemplo "Control de errores"

Aprenderemos a tratar/manejar los errores del sistema, veremos de forma sencilla cómo configurar el mapeo de excepciones con *org.springframework.web.servlet.handler.SimpleMappingExceptionResolver* para interceptar una excepción y cargar una vista correspondiente para su presentación.

1. Clic derecho sobre el proyecto **Run As->Maven Clean** y **Run As->Maven Install**
2. Clic derecho **Maven->Update Project…**
3. Clic derecho sobre **mvc_control_errores-> Run As on Server**



4. Abrir y estudiar la clase CuentaController.

```
package com.bolsadeideas.ejemplos.cuenta.controllers;

import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.bolsadeideas.ejemplos.cuenta.domain.Cuenta;

@Controller
@RequestMapping(value="/cuenta")
public class CuentaController {

  private Map<Long, Cuenta> cuentas = new ConcurrentHashMap<Long, Cuenta>();

  // Metodo handler formulario, para crear la cuenta
  @RequestMapping(method=RequestMethod.GET)
  public String crearCuentaForm(Model model) {
    // La instancia de "Cuenta" es creada y guardada como atributo "cuenta"
    // en el objeto "model", el cual es accesible desde
    // la vista, "cuenta/crearForm.jsp".
    model.addAttribute("cuenta", new Cuenta());
    return "cuenta/crearForm";
  }

  // Metodo handler que procesa el envio de datos del form
  @RequestMapping(method=RequestMethod.POST)
  public String crearCuenta(Cuenta cuenta, BindingResult result) {
    // Si ocurre un error en la validación aparecera el formulario
    // "cuenta/crearForm.jsp" con los mensajes.
    if (result.hasErrors()) {
      return "cuenta/crearForm";
    }

    // Si todo está bien, crea la cuenta y la agrega a la lista
    // "cuentas", luego redirige hacia el detalle /cuenta/{id},
    // controlado por el método handler "verDetalle(..) de mas abajo.
    this.cuentas.put(cuenta.asignarId(), cuenta);
    return "redirect:/cuenta/" + cuenta.getId();
  }
```

```
    @RequestMapping(value="{id}", method=RequestMethod.GET)
    public String verDetalle(@PathVariable Long id, Model model) {
      Cuenta cuenta = this.cuentas.get(id);

      if (cuenta == null) {
        throw new RecursoNoEncontradoException(id);
      }

      model.addAttribute("cuenta", cuenta);
      return "cuenta/detalle";
    }
  }
```

5. Ahora estudiemos la clase Exception: **RecursoNoEncontradoException.java**.

```
package com.bolsadeideas.ejemplos.cuenta.controllers;

import java.util.Date;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

public class RecursoNoEncontradoException extends RuntimeException {

  private static final long serialVersionUID = 1L;

  private Long recursoId;
  private Date fecha;

  public RecursoNoEncontradoException(Long recursoId) {
    this.recursoId = recursoId;
    fecha = new Date();
  }

  public Date getFecha() {
    return fecha;
  }

  public void setFecha(Date fecha) {
    this.fecha = fecha;
  }

  public Long getRecursoId() {
    return recursoId;
  }

}
```

6. Estudiar el archivo de configuración de spring **servlet-context.xml**.
    - Observamos que cuando ocurre la excepción **RecursoNoEncontradoException**, debe cargar y mostrar la vista jsp **cuentaNotExiste.jsp**.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">

    <context:component-scan base-package="com.bolsadeideas.ejemplos.cuenta" />

    <!-- Habilita la anotacion de Spring MVC @Controller -->
    <mvc:annotation-driven />

    <!-- Interceptores que son aplicados a todos los controladores anotados
        @Controller -->
    <mvc:interceptors>
        <!-- Cambiamos el local cuando envia un parametro del request 'locale'
            e.j. /?locale=es -->
        <bean class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor" />
    </mvc:interceptors>

    <!-- Guardamos los cambios del local usando cookie -->
    <bean id="localeResolver"
        class="org.springframework.web.servlet.i18n.CookieLocaleResolver" />

    <!-- View resolver de las Excepciones -->
    <!-- Mapeando las Excepciones -->
    <bean
        class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
        <property name="exceptionMappings">
            <props>
                <prop
              key="com.bolsadeideas.ejemplos.cuenta.controllers.RecursoNoEncontradoException">
                    cuentaNotExiste
                </prop>
                <!-- Definimos todas las clases de excepciones en particular y el property
                    defaultErrorView (para los errores en general) -->
                <!-- <prop key="java.lang.Exception">error</prop> -->
            </props>
        </property>
        <property name="defaultErrorView" value="error" />
    </bean>

... ETC ...
```

7. Abrir y estudiar vista jsp "cuentaNotExiste.jsp"

```
<%@page contentType="text/html;charset=UTF-8"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<html>
<head>
<title>La Cuenta NO Existe</title>
</head>

<body>
<h3>La cuenta no existe, está tratando de acceder a una cuenta no registrada en nuestro
sistema.</h3>
<p>Si ve esta página es porque el manejo y control de excepciones (errores) está funcionando
perfectamente, tal cómo se esperaba</p>
<br/>
El número de cuenta ${exception.recursoId} no existe!
<fmt:formatDate value="${exception.fecha}" pattern="yyyy-MM-dd" />
</body>
</html>
```

8. Abrir y estudiar archivo "index.jsp" para forzar una redirección inicial cuando arranca la aplicación a una cuenta inexistente.

```
<html>
<body>
<jsp:forward page="cuenta/3"/>
</body>
</html>
```

Resumen

En este capítulo aprendimos a trabajar con diferentes tecnologías de vistas de Spring MVC, principalmente estudiamos ejemplos para exportar a vistas pdf y excel, además analizamos un ejemplo de uso de layout con tiles, también conocidos como **plantilla global**.

Finalmente nos adentramos en el funcionamiento y soporte multilenguaje de Spring, las funcionalidades que puede involucrar y las diferentes posibilidades creando archivos de idiomas properties e implementando un completo proyecto multi-idioma a través de Spring Locale.

FIN.

Envía tus consultas a los foros!

Aquí es cuando debes sacarte todas las dudas haciendo consultas en los foros correspondientes

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com

Lectura Recomendada y Bibliografía

- Spring View technologies: Recomendable lectura de esta sección del manual oficial para complementar con este workshop.
- Spring MVC Tiles 3 Integration Tutorial: recomendable lectura de un artículo en inglés.
- Spring 4 Tiles 3 programatic configurations: recomendable lectura de un artículo en inglés.
- Creating and using Tiles pages: manual oficial de tiles.

AUTOR: Andrés Guzmán Fontecilla.
Email: andresguzf@gmail.com
WEB: http://www.bolsadeideas.com