Ejercicios de MVC en Spring

Índice

Continuaremos en esta sesión con la aplicación AmigosSpring, refactorizando la capa de presentación para que parte de ella pase de estar basada en servlets a usar Spring MVC

1. Configuración inicial del proyecto

1. Copiar la librería Spring webMVC a la carpeta `WEB-INF/lib` de nuestro proyecto. Dicha librería se incluye en la distribución estándar de Spring, pero no como parte del módulo principal, `spring.jar`.
2. Configurar el *dispatcher servlet* en el `web.xml` para que las URL con .do se asocien con Spring, tal y como aparece en los apuntes de la sesión:

```
<servlet>
   <servlet-name>dispatcher</servlet-name>
   <servlet-class>
      org.springframework.web.servlet.DispatcherServlet
   </servlet-class>
   <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
   <servlet-name>dispatcher</servlet-name>
   <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

3. Crear el fichero de configuración para los beans de la capa web. Ayudarse del plugin de Spring: `File > New > Other...> Spring bean definition`. Recordad que con la configuración por defecto del *dispatcher servlet*, el archivo debe llamarse `dispatcher-servlet.xml` y residir en la carpeta `WEB-INF`. En el asistente de creación, incluye el *namespace* "context" en el fichero, ya que vas a necesitar la etiqueta "context:component-scan".
4. Finalmente, introduce en el fichero `dispatcher-servlet.xml` la etiqueta "context:component-scan" para que Spring explore las clases de la capa web en busca de anotaciones.

2. MVC sin procesamiento de datos de entrada

Cambiar el caso de uso "leer mensaje" para que emplee Spring MVC. Esto supondrá:

1. Cambiar el servlet `LeerMensajeServlet` por un *controller* de Spring:
   1. Crear un paquete es.ua.jtech.spring.mvc para meter el código
   2. Crear en este paquete la clase LeerMensajeController. Añadir las anotaciones necesarias para convertirlo en controlador y asociarlo con la URL "/leerMensaje.do".
   3. El método que va a procesar la petición HTTP debe ser `public String procesar(String id, ModelMap modelo)`. Recuerda que debes usar una anotación para asociar "id" con el parámetro HTTP del mismo nombre. Recuerda

también que el ModelMap es para poder colocar el resultado que devuelve la capa de negocio, y que el String que devuelve el método es el nombre lógico de la vista
2. En el fichero `dispatcher-servlet.xml` debes definir el `ViewResolver` que asocia el nombre lógico de la vista con el nombre físico ("mensaje.jsp").
3. Finalmente, cambia el enlace que aparece en la página "correo.jsp" que ahora es a "leerMensaje.servlet" por "leerMensaje.do", para que la petición la reciba el controller en lugar del servlet.

3. MVC con procesamiento de datos de entrada

Cambiar el caso de uso "buscar usuarios" para que emplee Spring MVC. Esto supondrá:

1. Definir una clase `es.ua.jtech.spring.mvc.CriteriosBusqueda` que pueda capturar los datos que se introducen en el formulario de búsqueda (con propiedades int edadMin, int edadMax, String localidad, Sexo sexo)
2. Cambiar el servlet `BuscarUsuariosServlet` por un *controller* de Spring
    1. Crear la clase `es.ua.jtech.spring.mvc.BuscarUsuariosController`. Usando anotaciones, convertirla en controlador y mapearla con la URL "/busqueda.do"
    2. El controlador tendrá dos métodos: uno de ellos mapeado con GET y otro con POST:
        - `public String preparaForm()`: simplemente devuelve el nombre lógico de la vista con el formulario ("busqueda") (podría crear un `es.ua.jtech.spring.mvc.CriteriosBusqueda` vacío y colocarlo en el modelo, pero no es estrictamente necesario)
        - `public String procesaForm(CriteriosBusqueda criterios, BindingResult result, ModelMap modelo)`: Con @ModelAttribute, asociamos el primer parámetro al modelo. Si `result` contiene errores vuelve otra vez al formulario. Si no, llama a la lógica de negocio, coloca el resultado en el modelo bajo el nombre "encontrados" y devuelve el nombre lógico de la vista que mostrará los datos ("encontrados"). **Por el momento no preocuparse de detectar errores de validación adicionales ni de mostrar mensajes de error en el JSP.**
3. En la página "busqueda.jsp" el action debe ser vacío (action="") y el method="POST"
4. Finalmente, en la barra de navegación (página "navegacion.jsp") cambia el enlace a "busqueda.jsp" por "busqueda.do", para llamar al controller de Spring. En la página "encontrados.jsp" aparece otro enlace a "busqueda.jsp" que debes cambiar.

4. Validación de datos

Implementar validación de datos para el caso de uso "buscar usuarios". En concreto, verificar que tanto la edad mínima como la máxima son como mínimo 18 y que la máxima es efectivamente un número mayor que la mínima.

Ten en cuenta que:

1. Necesitarás añadir código de validación en el método `procesaForm` del controlador
2. Tienes que crear el ".properties" con los mensajes de error y referenciarlo en el `dispatcher-servlet.xml`
3. Debes usar las etiquetas de Spring en el formulario de "busqueda.jsp", para poder mostrar los mensajes de error de validación.
   - Debes usar el mismo nombre para "CriteriosBusqueda" en la anotación @ModelAttribute del método "procesaForm" y en el "modelAttribute" del formulario.
   - Al asociar el formulario al objeto de la clase "CriteriosBusqueda" será necesario que dicho objeto esté disponible cuando se vuelva del `preparaForm`. Por ello en este método debes añadir dicho objeto (vacío) al `ModelMap`.
4. Debes crear el .properties con los mensajes de error y configurarlo en el `dispatcher-servlet.xml`.