Struts

Sesión 4: Internacionalización. Pruebas

Indice

- **Internacionalización de aplicaciones**
- Pruebas con StrutsTestCase

Internacionalización (i18n)

- El proceso de diseño y desarrollo que lleva a que una aplicación pueda ser adaptada fácilmente a diversos idiomas y regiones sin necesidad de cambios en el código
- El factor más costoso es la traducción de textos, aunque no es lo único que varía
  - Formatos de fechas, números, moneda, etc.
- **Localización**: adaptación de una aplicación a un idioma/región concreto (l10n)

El soporte de i18n de Java

- La plataforma Java ofrece soporte nativo a la *i18n*,en el que se basa Struts, así que conviene conocerlo antes de ver cosas propias de Struts.
- Tres clases básicas:
  - **Locale**: combinación de idioma y país
  - **ResourceBundle**: permite almacenar textos fuera del código fuente,para no tener que recompilar si cambia algo (p.ej. idioma)
  - **MessageFormat**: permite hacer mensajes con parámetros a los que se da valor en tiempo de ejecución

Locale

- Combinación de **idioma**, y opcionalmente **país** y dialecto/región (este último no se suele usar). Tanto el idioma como el país se especifican con códigos ISO

```
Locale pibeLocale = new Locale("es","AR");
```

- Algunos métodos de Java son “sensibles” al *locale*, aceptándolo como parámetro

```
NumberFormat nf = NumberFormat.getCurrencyInstance(new Locale("es","ES"));
//Esto imprimirá "100,00 €"
System.out.println(nf.format(100));
```

ResourceBundle

- Almacena textos **aparte** del código fuente. En **Struts** se usan ficheros .properties
  - Recordemos en struts-config.xml cómo se decía dónde estaban los mensajes

```
//El fichero será mensajes.properties, estando en cualquier sitio del CLASSPATH
<message-resources parameter="mensajes"/>
```

  - Para **i18n**, basta con tener varios .properties que tengan al final el idioma y el país. Se usará automáticamente el del *locale* actual (luego veremos cómo cambiar el *locale* actual)

```
mensajes_es_ES.properties
mensajes_es_AR.properties
mensajes_en.properties      //No es necesario especificar país
mensajes.properties         //Por defecto, si el sistema no encuentra el apropiado
```

MessageFormat

- Para generar mensajes con parámetros. Ya lo hemos usado en *validator*

```
Se ha producido un error con el campo {0} del formulario
```

- Aunque se le puede dar valor a los parámetros con un API estándar de Java, lo habitual es que los "rellene" Struts por nosotros
- También se pueden formatear números, fechas y horas (*ver el API de Java SE*)

```
Se ha realizado un cargo de {0,number,currency} a su cuenta con fecha {1,date,long}
```

¿Qué aporta Struts a todo esto?

- Nos permite obtener y cambiar el locale actual en las acciones
- Nos automatiza la recuperación de mensajes del fichero .properties adecuado
- Las *taglibs* son “sensibles” al locale
  - Si las usamos adecuadamente, la aplicación estará automáticamente internacionalizada

Obtener y cambiar el locale actual en una acción

- Struts guarda el locale actual en la sesión HTTP

```
Locale locale = request.getSession().getAttribute(Globals.LOCALE_KEY);
```

  - Por defecto, el locale actual es el del servidor
  - Si queremos saber el del navegador del usuario, podemos hacer

```
request.getLocale();
```

- Para cambiar el locale se crea uno nuevo (ya que son objetos inmutables) y se guarda en la sesión

```
Locale nuevo = new Locale("es","ES");
request.getSession().setAttribute(Globals.LOCALE_KEY,nuevo);
```

Recuperar mensajes localizados

- A través de la clase MessageResources
- Es automático, aunque también se puede hacer a través de un API

```
Locale locale = request.getSession().getAttribute(Globals.LOCALE_KEY);
MessageResources mens = getResources(request);
String m = mens.getMessage(locale, "error");
```

Componentes de Struts “sensibles” al locale

- Los mensajes gestionados con ActionError y ActionMessage
- Muchas etiquetas de las *taglibs*
  - Sustituir todos los textos de los JSP por *tags* de Struts. La típica usada para esto es <bean:message>

```
<%@ taglib uri="http://struts.apache.org/tags-bean-el" prefix="bean" %>
…
<bean:message key="saludo“/>
<bean:message key="saludo“ arg0="${usuarioActual.login}" />
```

Localización de *validator*

- Los mensajes de error de los validadores estarán localizados automáticamente
- Podemos hacer validadores dependientes del *locale*

```
<form name="registro" locale="es" country="ES">
 <field property="codigoPostal" depends="required,mask">
   <var> <var-name>mask</var-name> <var-value>^[0-9]{5}$</var-value> </var>
 </field>
</form>
<!-- en Argentina, los C.P. tienen 1 letra seguida de 4 dígitos y luego 3 letras más -->
<form name="registro" locale="es" country="AR">
 <field property="codigoPostal" depends="required,mask">
     <var>
       <var-name>mask</var-name> <var-value>^[A-Z][0-9]{4}[A-Z]{3}$</var-value>
    </var>
</field>
</form>
```

Indice

- Internacionalización de aplicaciones
- **Pruebas con StrutsTestCase**

StrutsTestCase

- Una especie de “JUnit para Struts”
- Permite ejecutar una acción y comprobar
  - Si el *ActionForward* es el esperado
  - Si se han generado *ActionMessage* o no
  - …
- Hay dos enfoques
  - *Mock* (pruebas fuera del contenedor)
  - Basado en Cactus, para hacer las pruebas dentro del servidor

Ejemplo de prueba dentro del contenedor

```
import servletunit.struts.CactusStrutsTestCase;

public class LoginAccionTest extends CactusStrutsTestCase {
        public LoginAccionTest(String testName) {
                super(testName);
        }

        public void testLoginOK() {
                setRequestPathInfo("/login");
                addRequestParameter("login", "struts");
                addRequestParameter("password", "mola");
                actionPerform();
                verifyForward("OK");
                verifyNoActionErrors();
        }
}
```

Preparar petición

Ejecutar acción

Verificar resultados

Ejemplo de prueba dentro del contenedor (II)

```
//Ahora con ActionForm y verificando errores de validación
import servletunit.struts.CactusStrutsTestCase;

public class NuevoUsuarioAccionTest extends CactusStrutsTestCase {
        public NuevoUsuarioAccionTest(String testName) {
                super(testName);
        }

        public void testNuevoUsuarioFallido() {
                UsuarioForm miForm = new UsuarioForm();
                miForm.setLogin("");
                miForm.setPassword("notengologin!!!");
                setActionForm(miForm);
                setRequestPathInfo("/nuevoUsuario");
                actionPerform();
                verifyInputForward();
                verifyActionErrors(new String[] {"ERROR.ALTA"});
        }
}
```