Struts

Sesión 2: La vista en Struts.
ActionForms y *taglibs* propias

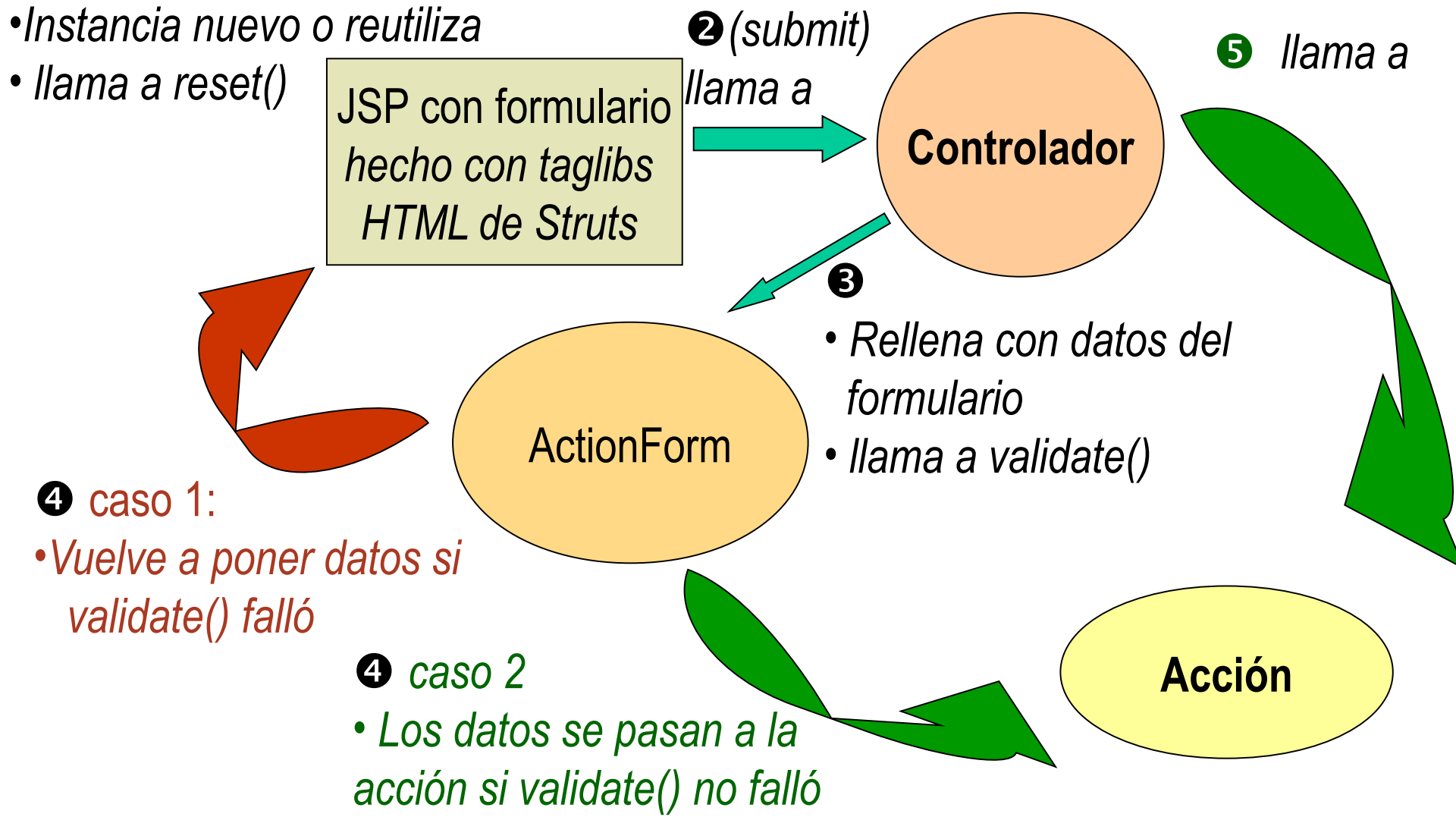
Indice

- **ActionForms. Ciclo de vida**
- Usar ActionForms
- Las taglibs de Struts. HTML y HTML-EL

ActionForms

- Normalmente la acción toma los parámetros directamente de la petición HTTP
- Pero esto implica que debe chequear errores en parámetros
- ¿No sería mejor que los tomara de otro sitio, una vez que ya se han validado?
- Los ActionForm son objetos Java para
  - Recolección de datos a partir de los que hay en la petición HTTP
  - Validación de datos por programa o automática (Validator)
  - Recuperación de datos para volver a rellenar formularios

❶ Ciclo de vida de un ActionForm



Ciclo de vida de un ActionForm (detallado)

- El controlador recibe la petición y chequea si lleva asociada un **`ActionForm`** De ser así, lo crea si no existe ya
- Se llama al método **`reset()`** que hay que reescribir para inicializar los campos
- El **`ActionForm`** se almacena en el ámbito deseado
- Los datos se rellenan con los del mismo nombre de la petición HTTP
- Se llama al método **`validate()`** donde debe estar la lógica de validación
- Si hay errores de validación, se redirecciona a la página especificada, sino se llama a **`execute()`**
- Si en la página JSP se utilizan las taglibs de Struts, se muestran los datos del **`ActionForm`**.

Tipos de datos en un ActionForm

- Puede ser cualquiera
- Pero hay que tener en cuenta que Struts copia desde la petición HTTP (Strings)
  - Struts sabe convertir tipos primitivos
- ¿Pero qué pasa si algo que debería ser un entero resulta ser una “a”?
  - Fallo en la conversión. La propiedad del ActionForm pasa a valer 0
  - Si el formulario vuelve a mostrarse aparecerá un 0 donde había una “a”. Usuario confuso.
- Enfoque pragmático y tedioso: todas las propiedades Strings. Haremos nosotros la conversión

Indice

- ActionForms. Ciclo de vida
- **Usar ActionForms**
- Las taglibs de Struts. HTML y HTML-EL

Usar ActionForms

- Cuatro pasos:
  1. Definirlo en struts-config.xml
     Si es ActionForm, solo el nombre
     Si es DynaActionForm, también los campos
  1. Asociarlo a la acción, también en struts-config.xml
  2. Escribir la clase Java (si es ActionForm)
  3. Obtener/cambiar props. desde las acciones

Usar ActionForms

1. Definirlo

"Clásico"

```
<form-beans>
  <form-bean name="FormLogin"
       type="es.ua.jtech.struts.presentacion.actionforms.FormLogin" />
  ...
</form-beans>
```

DynaActionForm

```
<form-bean name="FormLogin"
   type="org.apache.struts.action.DynaActionForm">
  <form-property name="login" type="java.lang.String"/>
  <form-property name="password" type="java.lang.String"/>
</form-bean>
```

Usar ActionForms (II)

2. Asociarlo a la acción en el **struts-config.xml**
   - **name:** nombre del ActionForm
   - **scope**: ámbito donde se almacena (request,session, …)
   - **validate**: si *false*, no se llama a validate()
   - **input:** página a la que volver si validate() falla

```
<action path="/login" type="acciones.AccionLogin"
        name="FormLogin" scope="session"
        validate="true" input="/index.jsp">
   <forward name="OK" path="/personal.jsp"/>
   <forward name="errorUsuario" path="/error.html"/>
</action>
```

Usar ActionForms (III)

3. Escribir el código Java (si no es dinámico)

```
public class FormLogin extends ActionForm {
   private String login;
   private String password;

   public void setLogin(String login) {
      this.login = login;
   }

   public String getLogin() {
      return login;
   }

   public ActionErrors validate(ActionMapping mapping,  HttpServletRequest request){
      ...
   }
}
```

Usar ActionForms (y IV)

- Para referenciar los campos del ActionForm desde una acción:

```
import javax.servlet.http.*;
import org.apache.struts.action.*;
public class AccionLogin extends Action {
   public ActionForward execute(ActionMapping mapping,
                                ActionForm form,
                                                ...
            …
            FormLogin f = (FormLogin) form;
            f.getLogin();
            f.get("login") //si fuera DynaActionForm
```

Indice

- ActionForms. Ciclo de vida
- Usar ActionForms
- **Las taglibs de Struts. HTML y HTML-EL**

Las *taglibs* de Struts

- Casi todas han quedado obsoletas tras la aparición de JSTL
- HTML: fundamental para trabajar con ActionForms

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %> ...
<html:form action="/login">
  <html:text property="login"/>
  <html:password property="password" redisplay="false"/>
  <html:submit>Entrar</html:submit>
</html:form>
```

Cuadros de lista

- Poner las opciones “a mano”

```
<html:select property="sexo">
        <html:option value="H">Hombre</html:option>
        <html:option value="M">Mujer</html:option>
        <html:option value="N">No especificado</html:option>
</html:select>
```

- Generarlas automáticamente a partir de un getXXX()

```
public class FormRegistro { …                                    (ActionForm)
        private static String[] listaSexos = { "Hombre", "Mujer", "No especificado"};
        public String[] getListaSexos() {
                return listaSexos;
        } ...
```

```
(el ActionForm asociado al formulario debe ser FormRegistro)
<html:select property="sexo">                                    (JSP)
        <html:options property="listaSexos"/>
</html:select>
```

Botones “de radio”

- Poner las opciones “a mano”

```
<html:radio property="sexo" value="hombre"/> hombre
<html:radio property="sexo" value="mujer"/> mujer
```

- Generarlas automáticamente a partir de un getXXX()
  - Suponemos el ActionForm de la traspa anterior
  - Struts no itera como en html:options, hay que hacerlo con JSTL
  - Para usar EL en un tag de struts, se usa la taglib HTML-EL

```
<%@ taglib uri="http://struts.apache.org/tags-html-el" prefix="html-el" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
...
<c:forEach items="${FormRegistro.listaSexos}" var="sexo">
            <html-el:radio property="sexo" value="${sexo}"/> ${sexo}
 </c:forEach>
```

Casillas de verificación

- Poner las opciones “a mano”

Una sola casilla:
```
<html:checkbox property="publi"/> Sí, deseo recibir publicidad sobre sus productos
```

Varias agrupadas:
```
<html:multibox property="aficiones">cine</html:multibox> Cine
<html:multibox property="aficiones">música</html:multibox> Música
```

- Generarlas automáticamente a partir de un getXXX()
  - Muy parecido a los botones de radio

```
<%@ taglib uri="http://struts.apache.org/tags-html-el" prefix="html-el" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
...
<c:forEach items="${FormRegistro.listaAficiones}" var=“a”>
        <html-el:multibox property=“aficiones“> ${a} </html-el:multibox> ${a}
</c:forEach>
```

Otras tags HTML

- Enlaces

```
<html:link action="/login">login</html:link>

(HTML generado)
<a href="login.do">login</a>
```

- Poner parámetros procedentes de un bean
  - Supongamos que tenemos un bean llamado usuario, con un método getLogin()

```
<html:link action="/editUsuario" paramId="login" paramName="usuario"
paramProperty="login"> Editar usuario </html:link>

(HTML generado)
<a href="editUsuario.do?login=pepe">Editar usuario</a>
```

¿Preguntas...?