



# Struts

## Sesión 1: MVC y Struts



# Indice

- **Introducción a Struts y repaso de MVC**
- Instalación de Struts
- El “ciclo de control” en Struts
- Las acciones
- Gestión de errores en las acciones

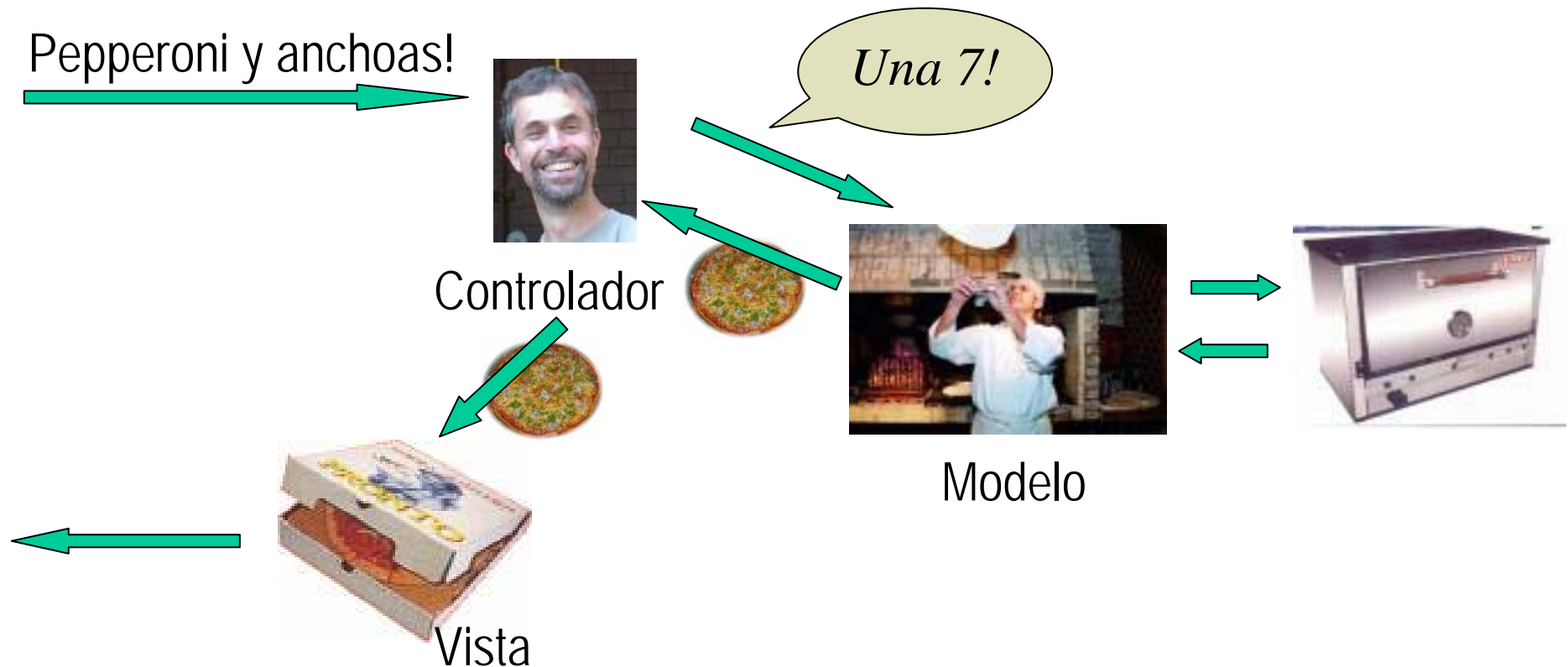


# Struts como framework MVC

- ¿Por qué Struts?
  - Es un estándar “de facto”. Amplia comunidad de desarrolladores. Mucha documentación y ejemplos.
  - Problema: en algunos aspectos es tedioso (configuración). Se ha quedado un poco “antiguo”.
- ¿Hay alternativas?
  - JavaServer Faces (centrado en el GUI)
  - Spring
  - Webwork,...
- Versiones de Struts
  - Veremos la 1.2 (más recursos disponibles) y diferencias con la 1.3 (intentaremos hacer código compatible con ella)
  - Struts 2: próxima generación, fusión con WebWork. Introduce ideas de AOP

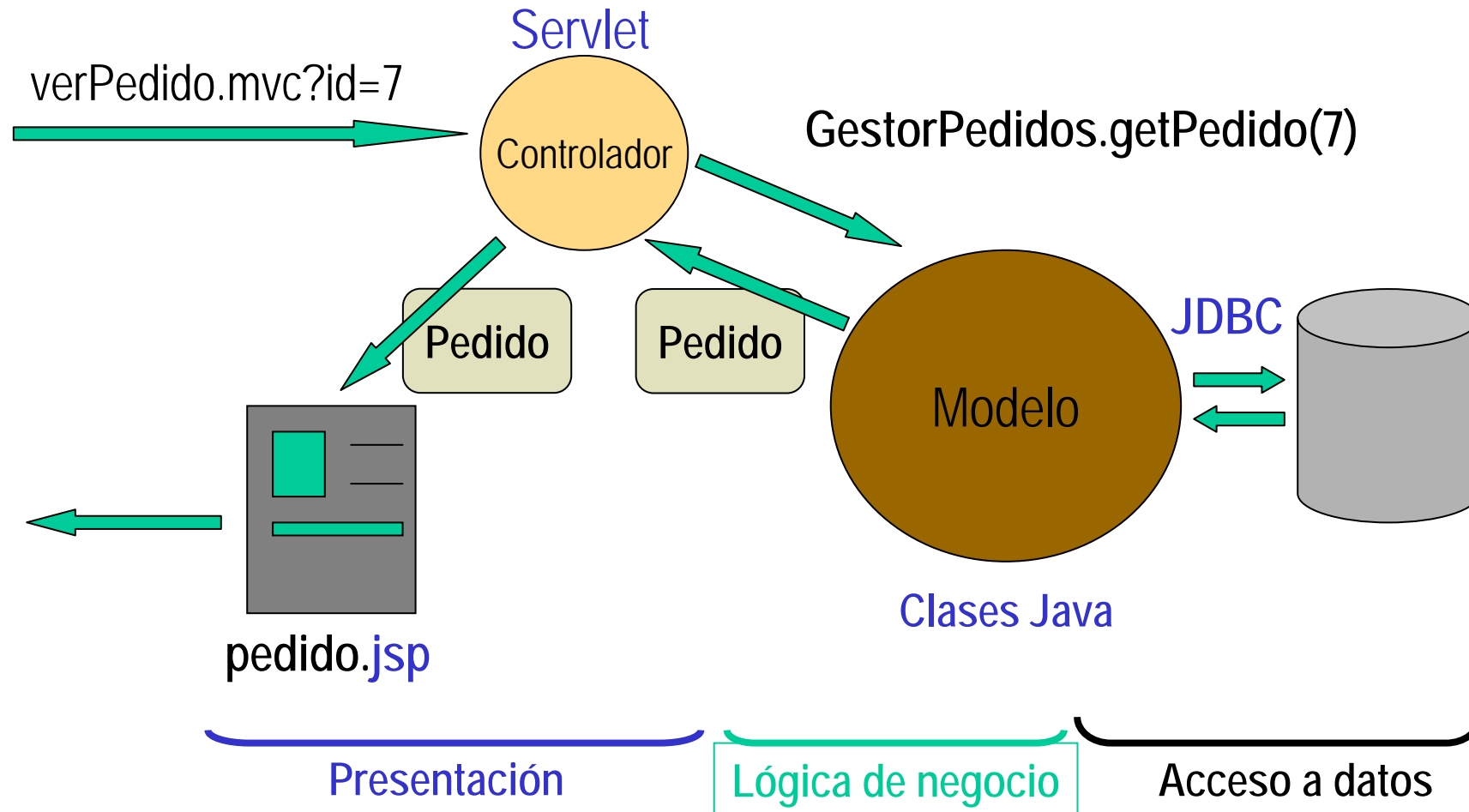


# Ejemplo real de MVC: pizzas para llevar



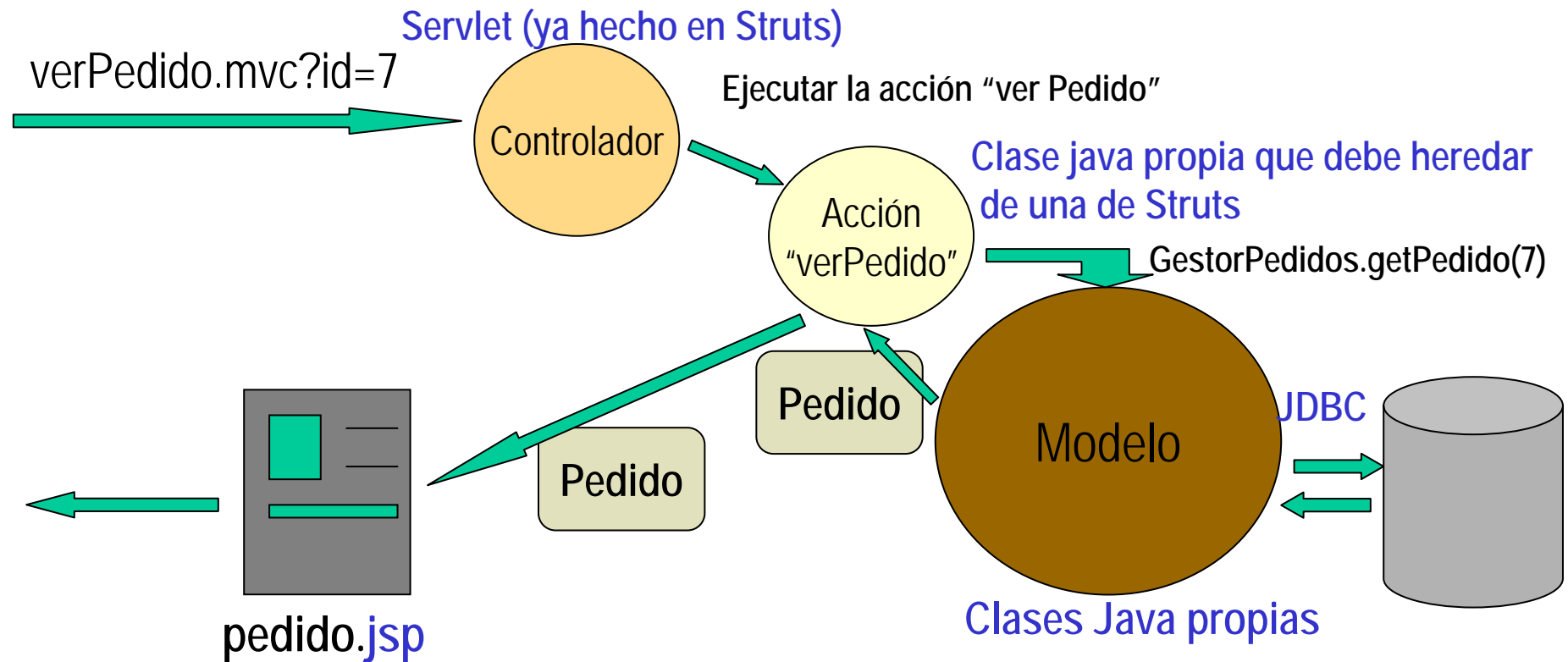


# Ejemplo no tan real (JavaEE)





# Flujo de MVC en Struts





# Indice

- Introducción a Struts y repaso de MVC
- **Instalación de Struts**
- El “ciclo de control” en Struts
- Las acciones
- Gestión de errores en las acciones



# Desarrollar con Struts: instalación

- Incluir una serie de .jar en el WEB-INF
- Crear y mantener el fichero de configuración principal
  - WEB-INF/struts-config.xml
- Crear y mantener ficheros de recursos (.properties)
  - Básicamente mensajes de error, avisos, etc.
- Hay una serie de *plugins* adicionales. Veremos el *validator*.





## El Servlet controlador

- Ya está hecho en Struts
- Debemos configurar la aplicación para que todas las peticiones vayan a parar al mismo servlet (en el web.xml)

```
<servlet>
  <servlet-name>controlador</servlet-name>
  <servlet-class>
    org.apache.struts.action.ActionServlet
  </servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>controlador</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
```



# Indice

- Introducción a Struts y repaso de MVC
- Instalación de Struts
- **El “ciclo de control” en Struts**
- Las acciones
- Gestión de errores en las acciones



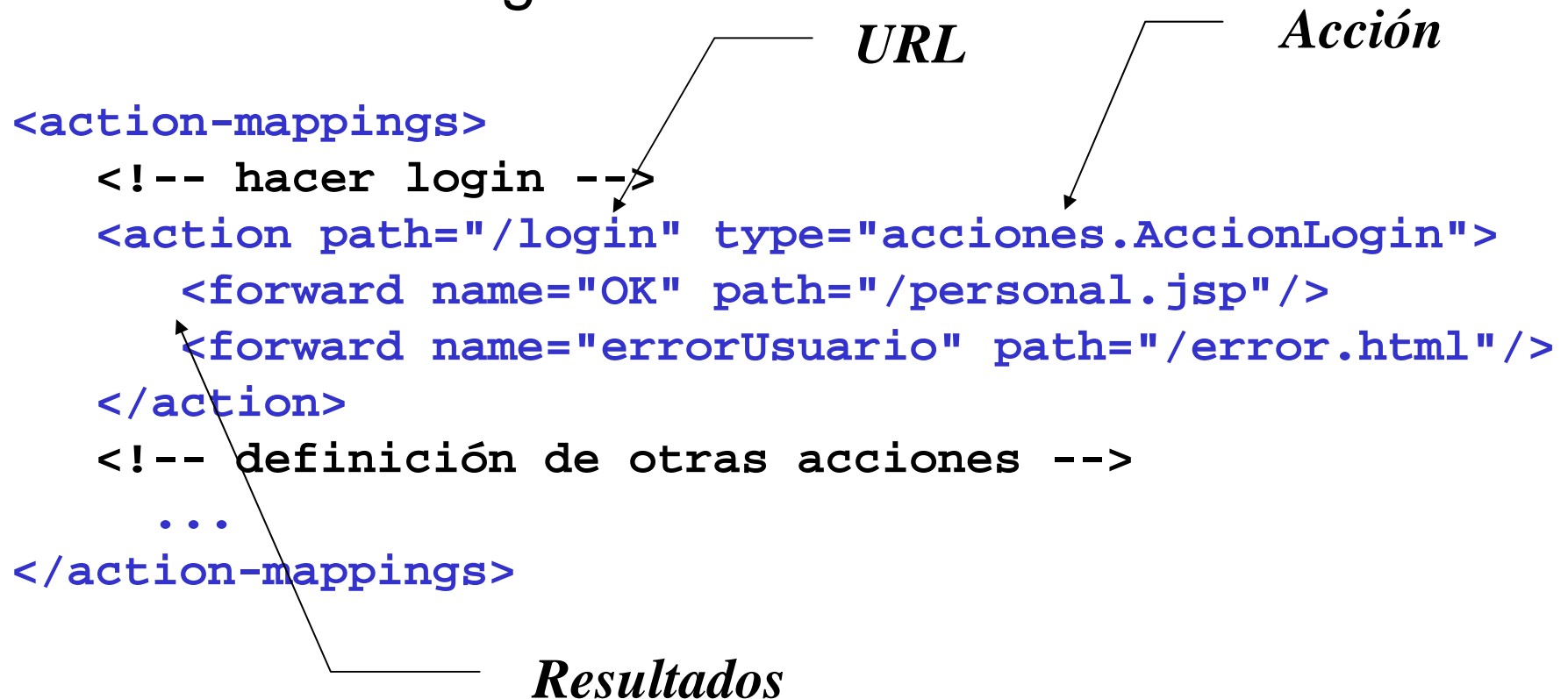
# El trabajo del servlet controlador

1. Recibir la petición: xxx.do
2. Buscar en struts-config.xml “lo que viene antes del `.do`” para encontrar una clase que se corresponde con una acción  
`login.do`  $\Rightarrow$  `es.ua.jtech.struts.presentacion.acciones.AccionLogin`
3. Crear un nuevo objeto de esa clase y llamar al método `execute`
4. El método `execute` devolverá un objeto `ActionForward` que indica a qué página JSP hay que llamar
5. Volver al paso 1



## 2. Mapeo petición-acción

- En struts-config.xml





## 3. Instanciar la acción y llamar a execute

- Clase que hereda de `org.apache.struts.action.Action`

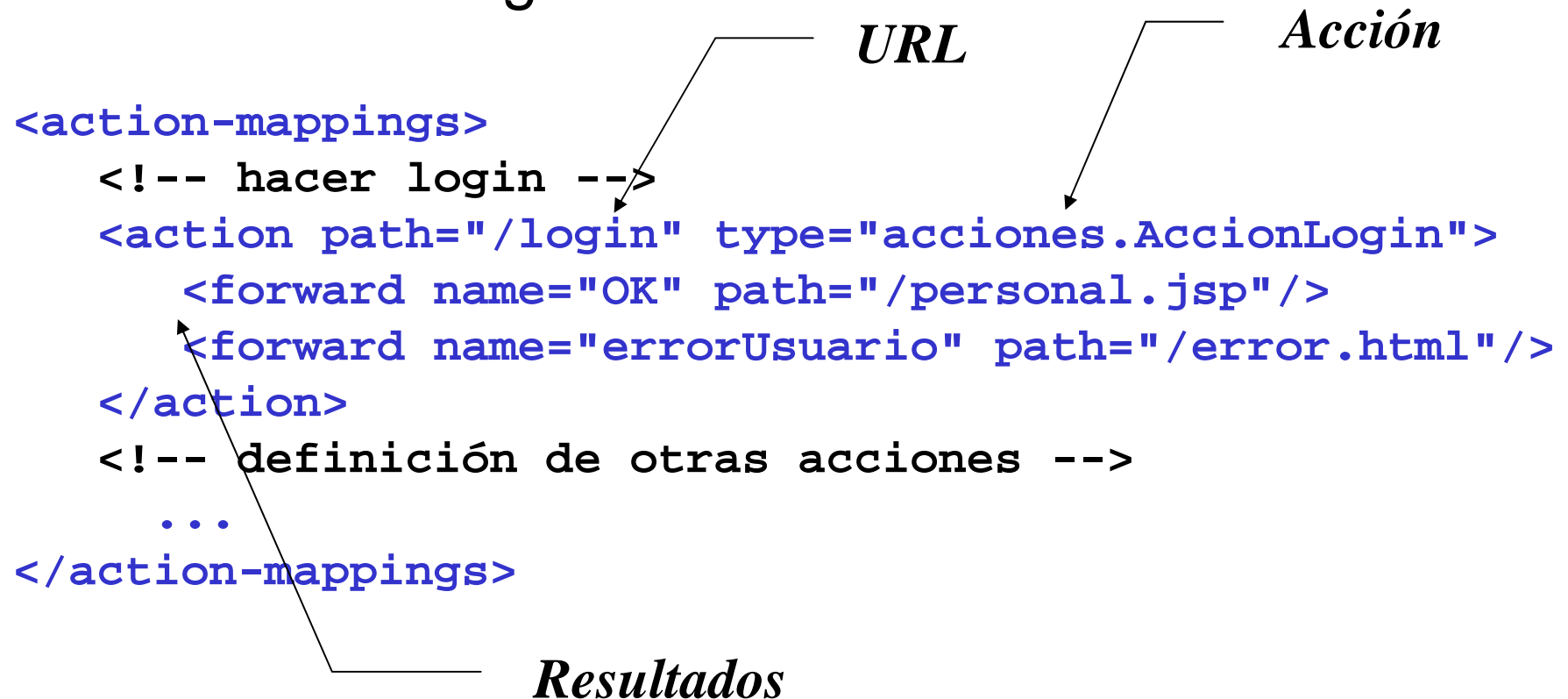
```
import javax.servlet.http.*;
import org.apache.struts.action.Action;

public class AccionLogin extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest req, HttpServletResponse resp){
        //obtener login y password y autenticar al usuario
        ...
        if (...)
            return mapping.findForward("OK");
        else
            return mapping.findForward("errorUsuario");
    }
}
```



## 4. Mapeo ActionForward-página JSP

- En struts-config.xml





## Las acciones

- Normalmente habrá una clase por caso de uso
  - AccionLogin
  - AccionSeleccionarLibro
  - AccionListarLibros
- También se puede hacer una acción para varias tareas: AccionLibro, con un parámetro HTTP diferencia entre seleccionar, listar,...
- Las acciones:
  - Toman los parámetros de la petición HTTP
  - Chequean errores en dichos parámetros
  - Le piden al modelo que haga “el trabajo sucio” y devuelva resultados
  - Ponen los resultados en un ámbito accesible a los JSP (request, response, session)
  - Ceden el control al controlador



## Ejemplo de código de acción

```
...
UsuarioDAO dao = UsuarioDAO.getInstance();
Usuario u = dao.login(request.getParameter("login"), request.getParameter("password"));
if (u==null) {
    ActionMessages errores = new ActionMessages();
    errores.add(ActionMessages.GLOBAL_MESSAGE, new
                ActionMessage("error.login"));
    saveErrors(request, errores);
    //si hay error se debe volver al formulario de login
    return mapping.findForward("error");
}
else {
    request.getSession().setAttribute("usuario", u);
    //si todo va bien se debe mostrar la lista de tareas (tareas.jsp)
    return mapping.findForward("OK");
}
...
```





# Indice

- Introducción a Struts y repaso de MVC
- Instalación de Struts
- El “ciclo de control” en Struts
- Las acciones
- **Gestión de errores en las acciones**



## Gestión de errores: pasos

1. Crear una lista de errores vacía (`ActionMessages`)
2. Añadir errores (`ActionMessage`) a la lista
3. Si la lista contiene errores
  - I. Guardar la lista en la petición HTTP (`saveErrors`)
  - II. Devolver un resultado (`findForward`) indicando error
4. En la página web a la que se salta, habrá que mostrarlos ( `<html:messages>` )



## Gestión de errores (II): código de la acción

```
ActionMessages errores = new ActionMessages();
try {
    //código que ejecuta la lógica de negocio.
    ...
}
catch(Exception e) {
    //añadir errores a la lista
    errores.add(ActionMessages.GLOBAL_MESSAGE, new ActionMessage("error.bd"));
}
//comprobar si la lista de errores está vacía
if (!errores.empty()) {
    //guardar los errores en la petición HTTP
    saveErrors(request, errores);
    //devolver un resultado que indique error
    return mapping.findForward("error");
}
```



## Gestión de errores (III): mostrar los errores

- Se usa una taglib de Struts (HTML)

```
<!-- referenciar la taglib de Struts que incluye la etiqueta -->  
<%@taglib uri="http://struts.apache.org/struts/tags-html" prefix="html" %>  
  
...  
<!-- mostrar los mensajes almacenados -->  
<html:messages id="e">  
  <ul>  
    <li>${e}</li>  
  </ul>  
</html:messages>
```



## ¿Dónde están los mensajes de error?

- NO en el código Java, sino en un .properties

```
errores.add(ActionMessages.GLOBAL_MESSAGE,  
            new ActionError("error.bd");
```

- Ficheros de recursos: definidos en `struts-config.xml`

```
<message-resources parameter="util.recursos"/>
```

- En el fichero `/WEB-INF/classes/util/recursos.properties`

```
error.bd = se ha producido un error con la base de datos
```



## Mensajes de error con “nombre”

- Asociar el error a un nombre arbitrario

```
...  
errors.Add("password",newActionError("error.pw"));  
...
```

- Mostrar el error en el JSP

```
...  
<html:messages id="e" property="password">  
  ${e}  
</html:messages>  
...
```



# ¿Preguntas...?