



Struts

Primera Parte



Struts

¿Qué es?

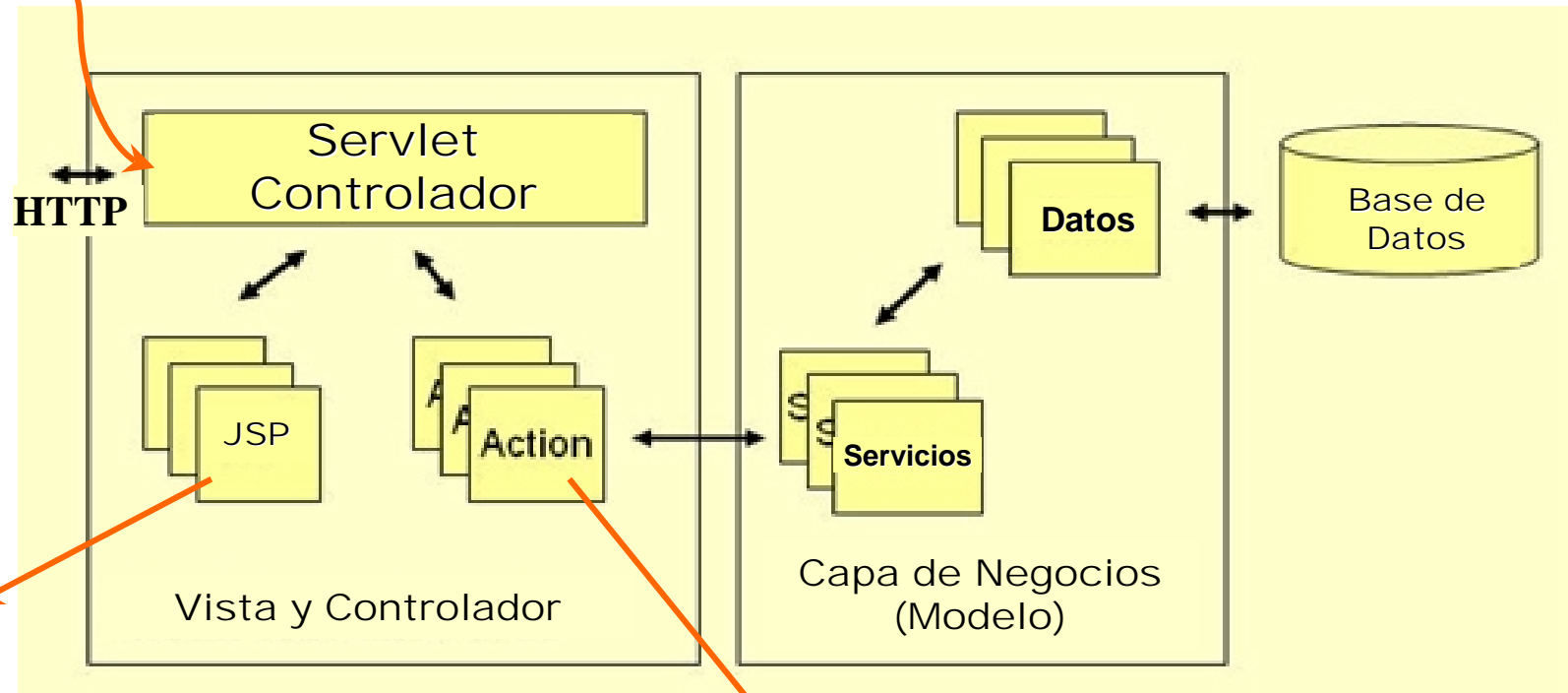
- Es un framework de aplicación, *de código fuente abierto*, escrito en JAVA, que simplifica el desarrollo de aplicaciones web.
- Implementa el patrón de diseño MVC2 o Modelo 2 (Model View Controller para Web).
- Es compatible con la plataforma J2EE y básicamente está construido sobre las tecnologías de Servlets y JSP. Struts combina Servlets, JSP's, custom tags y recursos de la aplicación en un único framework.
- Fue creado por Craig McClanahan y hasta hace pocos meses formó parte del subproyecto Jakarta, del Apache Software Foundation.
<http://jakarta.apache.org/struts>
- Oculta al programador los detalles de HTTP, JSP, Servlets, etc. Un programador Struts puede no conocer estos nombres, sin embargo tener conocimiento de las tecnologías de base de Struts hace que se puedan hacer soluciones creativas.
- Está soportado en la mayoría de los IDE's: Eclipse, JBuilder (de Inprise), WSAD (de IBM), JDeveloper (Oracle), etc.



El framework Struts

Arquitectura General

Todos los requerimientos HTTP entrantes, provenientes de los clientes, son interceptados por el **Servlet Controlador**. El **Servlet Controlador** usa el archivo de configuración **struts-config.xml** para determinar el flujo de la aplicación.



Vista: típicamente son formularios generados dinámicamente o páginas con links a otros recursos de la aplicación

Los objetos **Action** son componentes de la capa **Controlador**, interactúan con el **Modelo**. Los escribe el programador

STRUTS

- **Struts está basado en el patrón de diseño MVC:** la aplicación web íntegra se divide en tres subsistemas
 - **Modelo:** representa el estado (datos) de la aplicación
 - **Vista:** es la interfaz de usuario, son las pantallas con las que interactúa el usuario (JSP, HTML)
 - **Controlador:** implementa el flujo de la aplicación
- Struts provee componentes para las capas Vista y Controlador.
- Struts es de **modelo neutral**; típicamente el Modelo se implementa mediante JavaBeans y EJB.
- Struts almacena la información de ruteo y de mapeos en un archivo llamado **struts-config.xml**



El framework Struts

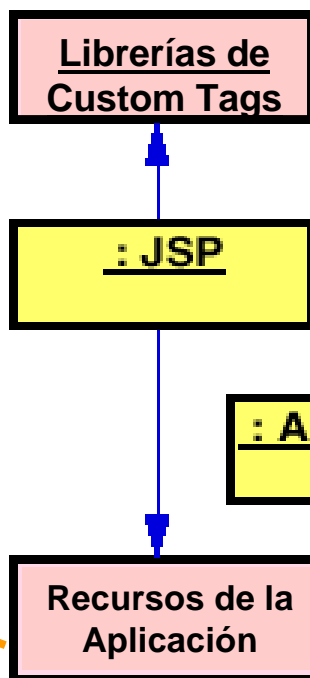
Arquitectura Detallada

- Recibe todos los requerimientos HTTP y los delega a un manejador apropiado (objeto Action).
- Recibe las respuestas de los Action y las redirecciona a la vista apropiada. Para ello, consulta un conjunto de mapeos definidos en el archivo de configuración struts-config.xml

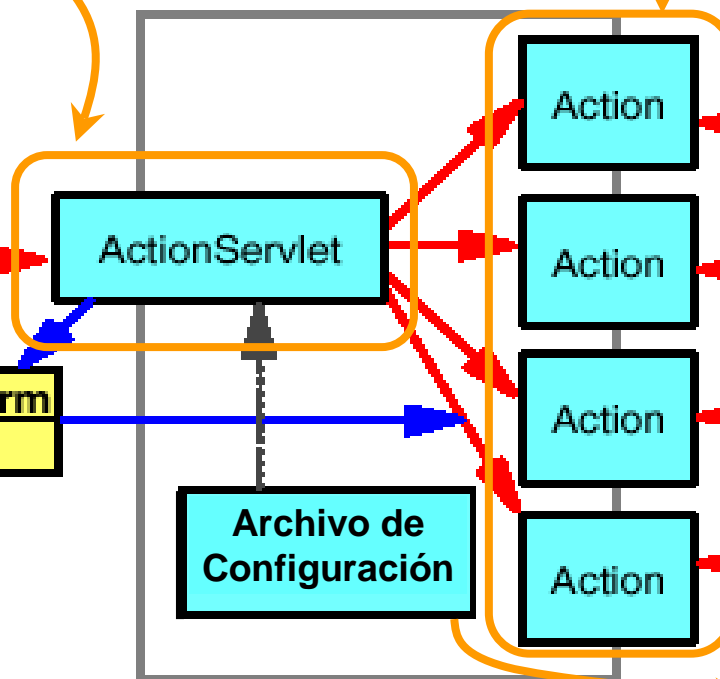
Actúan como *adaptadores* entre la capa web (requerimiento HTTP) y la de negocio (el modelo)

Representa la lógica y el estado del negocio

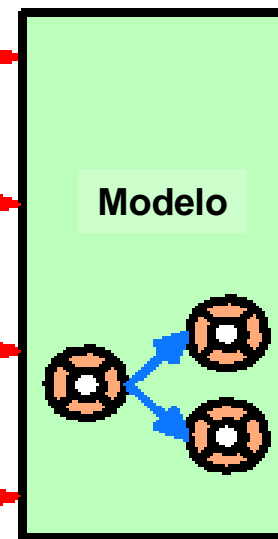
Vista



Controlador



Modelo



JavaBeans

EJB

struts-config.xml

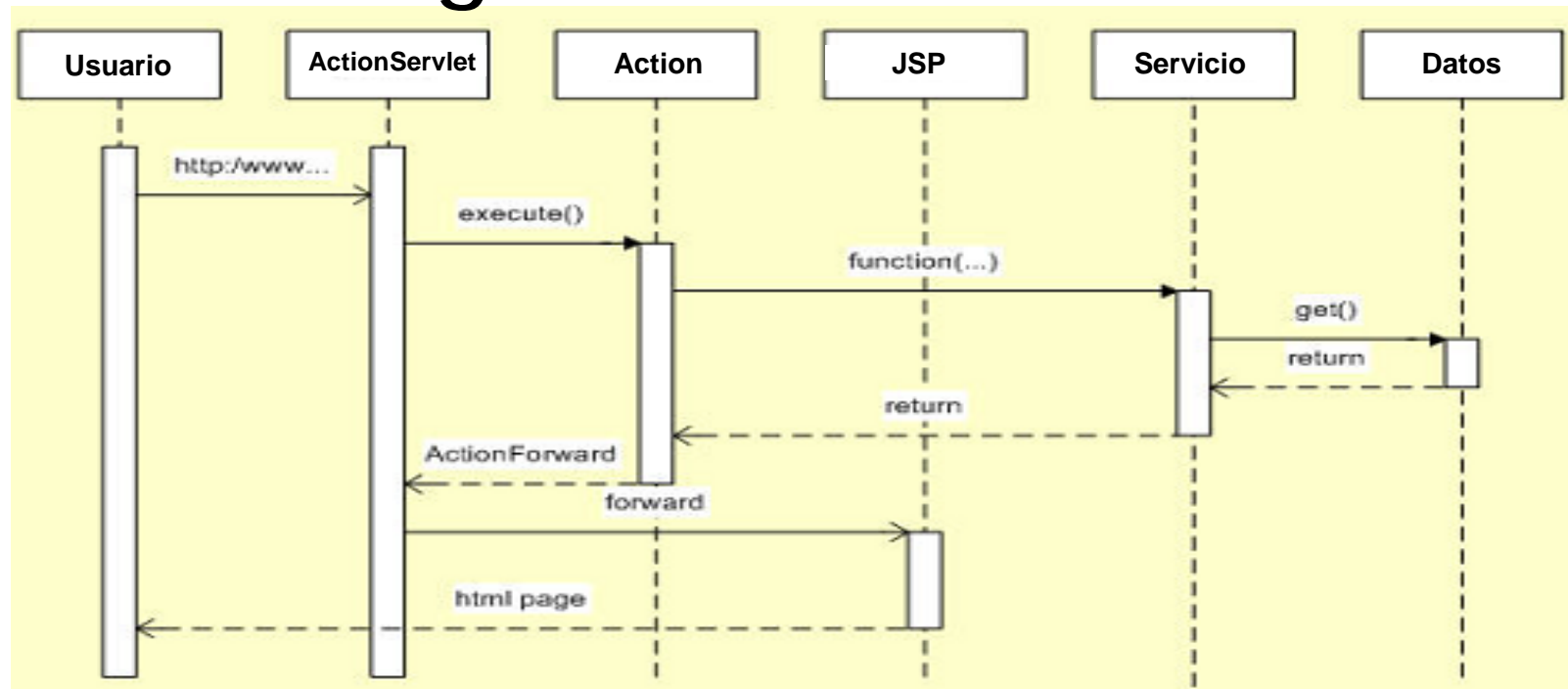
Soporte de Struts

Prompts y mensajes usados por los custom tags



El framework Struts

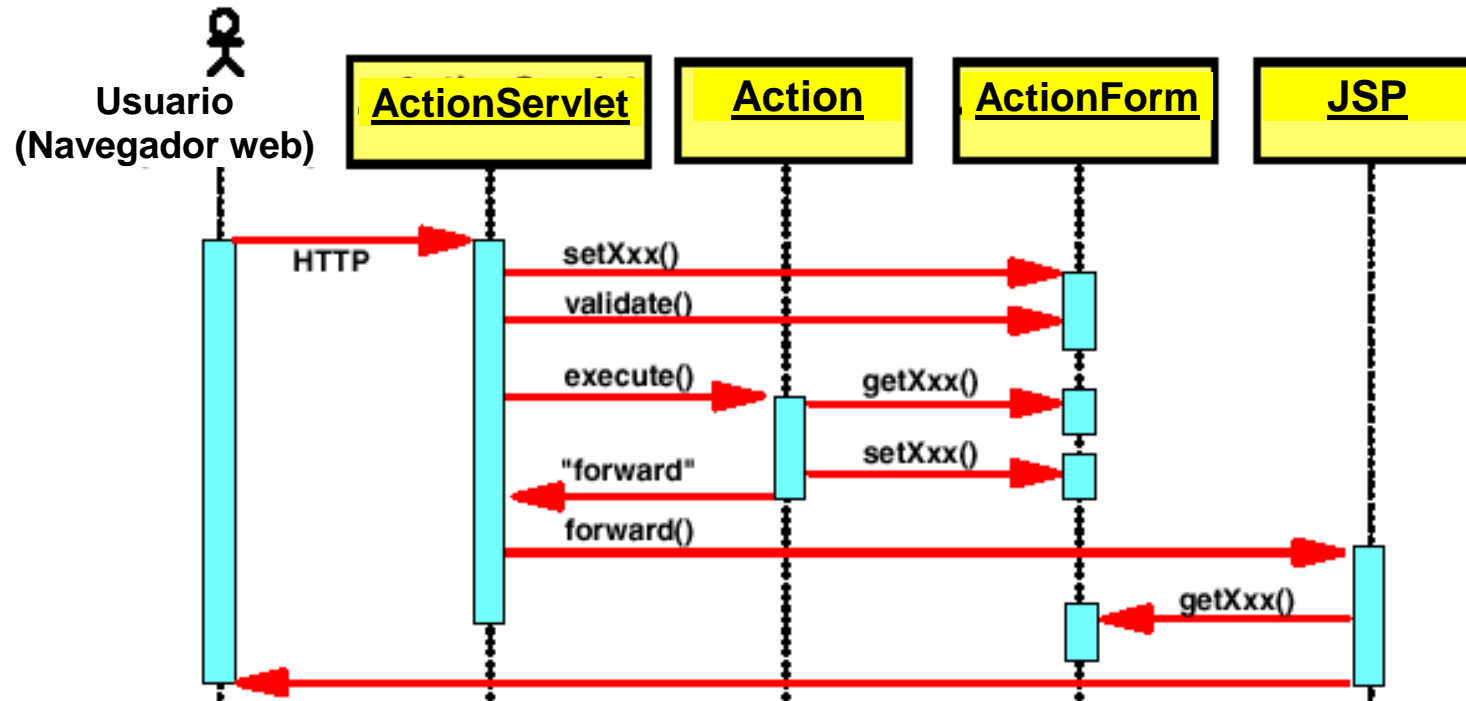
Diagrama de Secuencia



- 1) El **Usuario** presiona sobre un link de una página HTML o envía un formulario mediante un botón submit
- 2) El **Action Servlet** (Servlet Controlador) recibe el requerimiento, busca información de mapeo en el archivo **struts-config.xml** que usará para enrutar el requerimiento al objeto **Action** apropiado.
- 3) El framework invoca al método **execute()** del objeto Action apropiado, cuya función principal es **procesar el requerimiento entrante** y devolver un objeto **ActionForward** que identifica a donde direccionar el flujo (por ejemplo a una JSP, a otro objeto Action).
- 4) El requerimiento es procesado invocando a funciones apropiadas del Modelo.
- 5) La componente del Modelo invocada retorna al Action.
- 6) El objeto **ActionForward** retornado por el Action es tomado por ActionServlet quién lo usa para seleccionar el próximo recurso de la aplicación a ejecutar, usualmente una **JSP** que muestra la respuesta.
- 7) La **JSP** es invocada por el ActionServlet y devuelve al navegador una página HTML.
- 8) Se le presenta al **Usuario** una nueva página en el navegador.

El framework Struts

Diagrama de Secuencia



Considerando objetos **ActionForm** (*form beans*), se agregarían los siguientes pasos:

Antes de invocar al método `execute()` del objeto action:

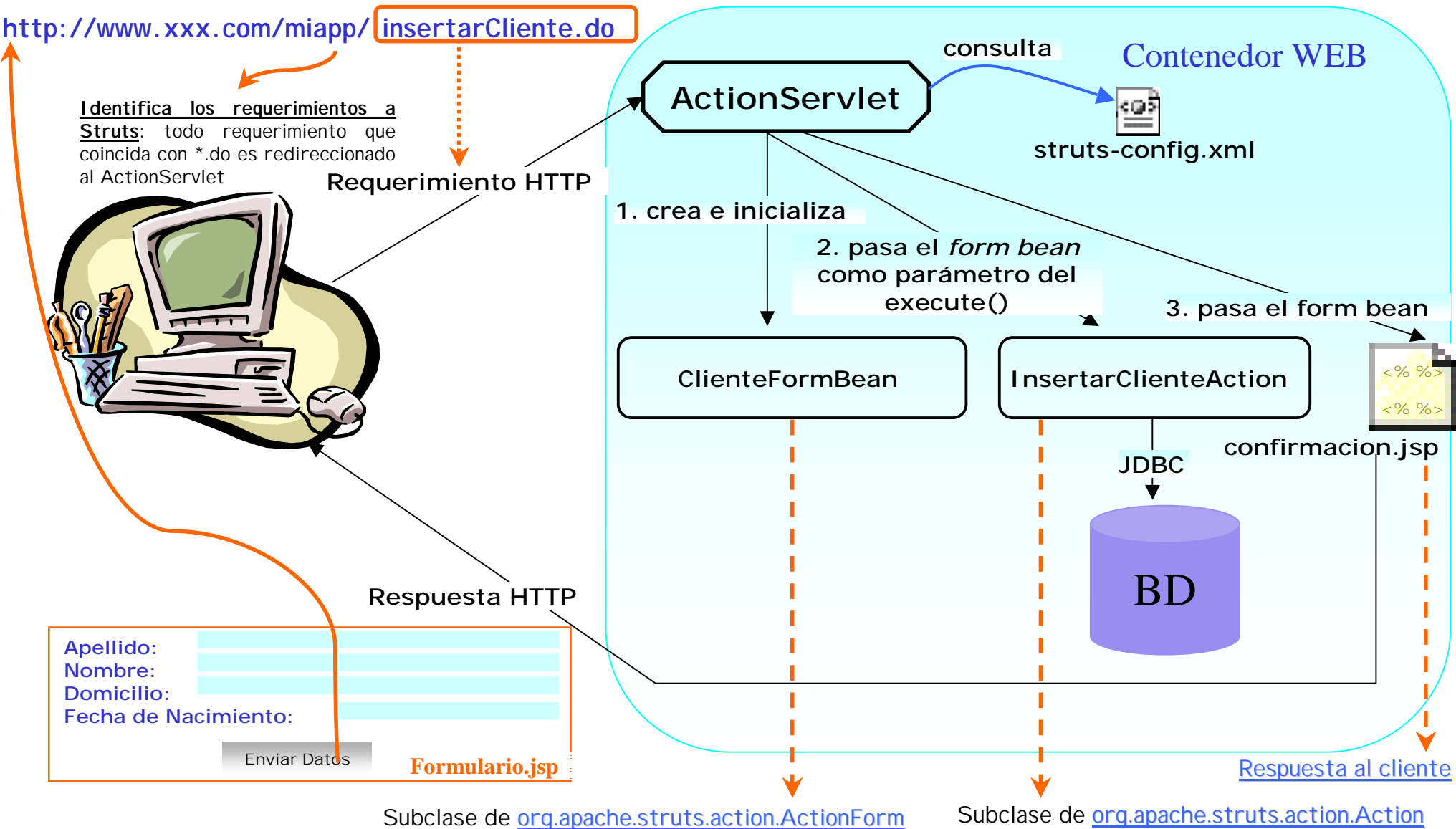
- Si el objeto **Action** que atenderá el requerimiento, tiene un *form bean* (**ActionForm**) asociado, el framework crea una instancia (o usa una existente) y la *setea* con los datos del formulario de entrada invocando al método `setXxx()`.
- Luego, se invoca al método `validate()` del *form bean* para validar los datos de entrada. Si la validación falla, el usuario es retornado a la página de entrada, para corregir los datos.
- Si la validación es exitosa, el framework invoca al método `execute()` del objeto Action pasándole la instancia del *form bean*

Luego de invocar al método `execute()` del objeto action:

- El objeto Action recupera los datos del form bean, invocando a los métodos `getXxx()` y realiza la lógica apropiada invocando a funciones de las componentes del modelo.
- La JSP consultará el *form bean* para generar la página de respuesta.



Una Aplicación Struts





Componentes del Framework Struts

Componentes de la capa *Controladora*

- **ActionServlet:** es la “columna vertebral” del framework Struts. Como todos los Servlets, vive en el Contenedor Web. Su función principal es dirigir el flujo de control de la aplicación. Es el ***controller***.
- **Action:** es el objeto que atiende el requerimiento HTTP; puede hacer todo lo que hace un Servlet, pero **No es un Servlet**. Interactúa con el modelo.
- **ActionForwards:** representan las URI's de la aplicación; usualmente se definen en el archivo de configuración **struts-config.xml**. Tienen un nombre lógico y un *path*.
- **ActionMappings:** le proveen a los objetos Action de una URI o *path*. Funciona como una “hoja de ruta” para el ActionServlet.
- **ActionForms:** son JavaBeans con un par de métodos adicionales que permiten validar datos de entrada y *resetear* datos. Manejan automáticamente los *inputs* provenientes de formularios HTML's.
 - **Basados en Maps:** soportan propiedades que se almacenan en un Map; son propiedades para las que no es necesario declarar un campo, un *getter* y un *setter*. Son útiles cuando se manejan muchas propiedades.
 - **DynaActionForms:** permiten especificar las propiedades del JavaBean en el archivo de configuración **struts-config.xml**. Evita codificar por cada propiedad del JavaBean un campo, un método *getter* y un *setter*.
- El archivo de configuración **struts-config.xml**: es usado para cargar las componentes críticas del framework. El archivo **struts-config.xml** es a Struts lo que es el **web.xml** es para el Contenedor Web. El ActionServlet lee el archivo de configuración **struts-config.xml** para crear y configurar los objetos que el framework necesita.



Componentes del Framework Struts

ActionMappings

| ActionMapping: | ActionMapping: | ActionMapping: | |
|--------------------------------|--------------------------------|--------------------------------|-------|
| -Nombre de la clase del Action | -Nombre de la clase del Action | -Nombre de la clase del Action | |
| -Nombre del ActionFormBean | -Nombre del ActionFormBean | -Nombre del ActionFormBean | |

Elemento Action
del archivo de
configuración de
Struts

ActionMappings.findMapping(String path)

El **ActionServlet** está en el centro de la aplicación, su función es invocar a otros objetos para que realicen servicios

ActionServlet

Crea una instancia

objeto Action

Crea una instancia

objeto ActionForm

ActionFormBeans.findFormBean(String name)

ActionFormBeans

| ActionFormBean: | ActionFormBean: | ActionFormBean: | |
|--|--|--|-------|
| -Nombre de la clase del ActionFormBean | -Nombre de la clase del ActionFormBean | -Nombre de la clase del ActionFormBean | |

Elemento ActionForm del
archivo de configuración
de Struts



Componentes del Framework Struts

El ActionServlet

- El ActionServlet es un **singleton**, hay exactamente un ActionServlet por aplicación Struts.
- El ActionServlet es una **caja negra**, a diferencia de otras componentes del framework, como los objetos Action y ActionForm que son diseñadas para extenderse y personalizarse (cajas blancas).
- El ActionServlet pasa la mayor parte de su tiempo invocando a otros objetos. El programador codifica los objetos que invoca el ActionServlet (patrón Inversión de Control).
- El ActionServlet coordina las actividades de la aplicación: los métodos definidos por el programador son invocados por el ActionServlet, NO son declarados adentro del ActionServlet.
- Una aplicación Struts puede dividirse en múltiples módulos y todos ellos comparten el mismo ActionServlet.

El ActionServlet delega la mayor parte del trabajo en el RequestProcessor y en las clases Action.

- Cuando arriba un requerimiento, el ActionServlet determina el módulo que atenderá el requerimiento, invocando al método **process(HttpServletRequest, HttpServletResponse)** y luego invoca al método **process(HttpServletRequest, HttpServletResponse)** del **RequestProcessor** del módulo seleccionado.

El RequestProcessor

- El **RequestProcessor** es el corazón del procesamiento de cada requerimiento HTTP. El método **process** tiene la responsabilidad de procesar un requerimiento HTTP y de crear su respuesta. El RequestProcessor de defecto, **org.apache.struts.action.RequestProcessor**, viene con el framework.
- El **RequestProcessor** puede usarse como una “caja negra” o extenderse para proveer un comportamiento especial.



Componentes del Framework Struts

Configurar el ActionServlet

- El **ActionServlet** se configura en el [web.xml](#) usando el *elemento* servlet, bajo un nombre, por ej. [action](#), y como una instancia de la clase [org.apache.struts.action.ActionServlet](#).
- Es posible configurar múltiples parámetros de inicialización para el **ActionServlet**, uno de ellos es el archivo de configuración, [struts-config.xml](#), en el que se especifican todos los objetos de la aplicación.
- Para identificar los requerimientos HTTP que atenderá el **ActionServlet**, se usa algunos de los mapeos de url estándares de la API de Servlets. En este caso, se usará el mapeo por extensión: todos los requerimientos a recursos de nuestra aplicación que finalicen con **.do**, por ejemplo: productos.do, el Contenedor Web los redireccionará al **ActionServlet**. Todos los requerimientos que no coincidan con ese patrón, por ejemplo *.jsp, serán atendidos por el servicio JSP del Contenedor Web.

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>WEB-INF/struts-config.xml</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

[web.xml](#)

```
<FORM action="login.do">
  Nombres: <INPUT type="text" name="nombre"><BR>
  Apellido: <INPUT type="text" name="apellido"><BR>
  <INPUT type="submit" value="Procesar">
</FORM>
```



Componentes del Framework Struts

El Action

- Es una clase JAVA, en la que el **ActionServlet** delega el manejo del requerimiento y de la respuesta.
- Los objetos **Action** son instancias de subclases de org.apache.struts.action.Action. Funcionan como mini-servlets, aunque **NO SON SERVLETS**; sus responsabilidades fundamentales son: **acceder a la capa de negocios, preparar objetos para la capa de presentación y manejar errores**.
- El **Action** es parte del **Controller**; por lo tanto se recomienda que el comportamiento relacionado a la lógica del negocio, como por ejemplo acceder a la bd, sea realizado por objetos de negocio (clases separadas) y no adentro del mismo Action; el Action es el lugar ideal para codificar tareas específicas de la web, por ejemplo manejo de la sesión de usr., del requerimiento, del contexto de la aplicación.
- El **ActionServlet** crea una **única instancia de cada subclase de Action** por aplicación y las usa para servir a todos los requerimientos que recibe. Es un objeto **multithread**, no es un objeto **thread-safe**.
- El **ActionServlet** usa la colección de objetos **ActionMappings** para determinar el **Action** que manejará cada requerimiento entrante. Finalmente, invocará al método **execute()** del Action al que le envía un conjunto de objetos útiles. Cuando el método **execute()** termina, devuelve un objeto **ActionForward**, que es usado por el **ActionServlet** para determinar a donde pasará el control para completar el requerimiento. Generalmente el **ActionForward** define que la próxima componente a mostrar es de presentación como una JSP aunque puede referirse a otro Action (encadenamiento de Action) o a otro recurso de la aplicación, como HTML. Si el **ActionForward** es **null**, el **ActionServlet** asume que el Action generó la respuesta y completó el requerimiento (en cuyo caso no hace nada).
- Como mejor práctica se establece **un Action por requerimiento HTTP** (no encadenar Actions).



Componentes del Framework Struts

Ejemplo de un Action

Devuelve un objeto `ActionForward` que identifica la próxima componente a invocar por el `ActionServlet`

```
public class AgregaUsrAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)
                                                                    throws Exception
    {
        // Crea el objeto ActionErrors en donde se almacenarán los errores que puedan ocurrir (clave, valor)
        ActionErrors errors = new ActionErrors();
        // Crea el objeto ActionForward en donde se almacenará el siguiente paso que retornará el método execute al ActionServlet
        ActionForward forward = new ActionForward();
        // Recupera el "form bean" que instanció el framework, haciendo casting al tipo particular
        UsuarioFormBean usuario= (UsuarioFormBean)form;
        String email = usuario.getEmail();
        String password = usuario.getPassword();
        try {
            // Invoca a objetos que implementan la lógica de negocio
            ActualizarBD.insertarFila("Usuario", usuario);
        }
        // Captura las excepciones de generadas por los objetos de negocio
        catch (SQLException e) {
            // Mensaje de error general
            errors.add(ActionErrors.GLOBAL_ERROR, new ActionError("error.usuarioDuplicado"));
        }
        if (!errors.isEmpty()) {
            // Guarda en el requerimiento los errores, para ser recuperados usando tags específicos de Struts <html:errors>
            saveErrors(request, errors);
            forward= mapping.findForward("error");
        }
        else
            forward=mapping.findForward("exito");
        // Retorna el siguiente paso (componente) a la que el ActionServlet invocará
        return forward;
    }
}
```



Componentes del Framework Struts

Manejo de Errores en los Action

Struts tiene un sistema de manejo errores muy bien desarrollado, que permite:

- Capturar errores
- Pasar el error como un dato del requerimiento
- Mostrar mensajes realcionados a los errores producidos

Involucra los objetos: ActionErrors y ActionError y un método utilitario saveErrors

Usando custom tags como `<html:errors>`

La registración de errores en el Action, consiste en:

- Crear una instancia vacía de **ActionErrors**
`ActionErrors errors = new ActionErrors();`
- Agregar “claves” para los mensajes de error
`errors.add(ActionErrors.GLOBAL_ERROR, new ActionError("error.usuarioDuplicado"));`
- Chequear si se agregaron errores
- Guardar la colección de ActionErrors en el requerimiento
- Pasar el control a una página de error para que muestre los mensajes y en caso de no haber errores, continuar el procesamineto normamente

```
if (!errors.isEmpty()) {  
    // Guarda en el requerimiento los errores, para ser recuperados usando tags específicos de Struts <html:errors>  
    saveErrors(request, errors);  
    forward= mapping.findForward("error");  
} else  
forward=mapping.findForward("exito");
```



Componentes del Framework Struts

ActionForwards

Los **ActionForwards** definen “los lugares a donde ir o pasar el control”. Son los **links** de la aplicación

Los objetos **Action** devuelven un objeto **ActionForward** al **ActionServlet** en el que se especifica un **nombre** (por ej. “exito”, “error”) y que el **ActionServlet** lo usará para recuperar un **path** que indica el lugar al que se le pasará el control.

Los objetos **ActionForward** representan las URI's de la aplicación. Tienen 4 propiedades:

| Propiedad | Descripción |
|------------------|---|
| name | Especifica el nombre lógico del ActionForward. Las componentes hacen referencia al ActionForward por nombre |
| path | Especifica la URI para el ActionForward. |
| redirect | Si es true , el redireccionamiento al path especificado se hace usando sendRedirect. Por defecto es false , indicando que se usará un forward. |
| className | Es opcional y especifica una subclase de org.apache.struts.action.ActionForward usada para instanciar el ActionForward <small>forward=mapping.findForward("exito");</small> |

Struts provee dos tipos **ActionForward**:

- **ActionForward Globales**: están disponibles para cualquier objeto **Action** de la aplicación.
- **ActionForward Locales**: están disponibles solamente para el objeto **Action** donde se definen.



Componentes del Framework Struts

ActionForwards

El objeto **Action**, elige el **ActionForward**, por nombre, de la siguiente manera:

```
forward=mapping.findForward("exito");
```

Asociar un nombre lógico a los *forwards*, permite cambiar el destino de un link (path) sin modificar el código de las componenetes que hacen referencia a ese link

- El objeto **mapping** es pasado como parámetro al **Action** e incluye una lista de **ActionForward** locales y un link a los **ActionForward** globales.
- El método **findForward** chequeará primero los **ActionForwards** locales y si la búsqueda falla disparará otra en los **ActionForward** globales. Si no encontrará un **ActionForward** con el nombre dado en alguno de los dos alcances, entonces retorna null.

Los **ActionForwards** se definen en el archivo de configuración **struts-config.xml**, de la siguiente manera:

```
<global-forwards>
  <forward name="login" path="/logon.jsp"/>
  <forward name="error" path="/paginaerror.jsp"/>
</global-forwards>
<action-mappings>
<action path="/listaCuentas" name="clienteInfoForm" scope="session"
        type="curso.strutsweb.actions.ListaCuentasAction" input="/index.jsp">
  <forward name="exito" path="/listaAccounts.jsp"/>
  <forward name="fracaso" path="/index.jsp" redirect="true"/>
</action>
</action-mappings>
```

} **ActionForward globales**

} **ActionForward locales**

Indicación para el ActionServlet: invoca al sendRedirect o al forward del RequestDispatcher (default)



Componentes del Framework Struts

ActionMappings

- Un **ActionMapping** es una instancia de `org.apache.struts.action.ActionMapping` y describe cómo Struts maneja cada operación o **Action**.
- El **ActionMapping** funciona como una **hoja de ruta** para el ActionServlet.
- Cada **ActionMapping** está asociado con una URI específica a través de la propiedad **path**. Cuando el **ActionServlet** recibe un requerimiento, usa la propiedad **path** para elegir el correspondiente objeto ActionMapping. Este ActionMapping le podría indicar que se **redireccione el control** a un recurso de la aplicación, o que se **llene y valide un ActionForm** para luego pasarle el **control a un objeto Action** y cuando éste retorna, **busque el ActionForward asociado con el ActionMapping**.
- Los **ActionMapping** permiten usar el **mismo objeto Action** con diferentes *mappings*, por ejemplo un *mapping* podría requerir validación y otro no sobre el mismo Action.
- El conjunto de objetos **ActionMapping** se guarda en una colección de **ActionMappings**, que es una instancia de `org.apache.struts.action.ActionMappings`.
- De la misma manera que los **ActionForwards**, Struts crea el objeto **ActionMappings** *parseando* el archivo **struts-config.xml**.

```
<action-mappings>
```

```
<action path="/login1" name="loginFormBean" scope="request" type="app.LoginAction" validate="true" input="/index.jsp"/>
```

```
<action path="/login2" name="loginFormBean" scope="request" type="app.LoginAction" input="/index.jsp"/>
```

```
</action>
```

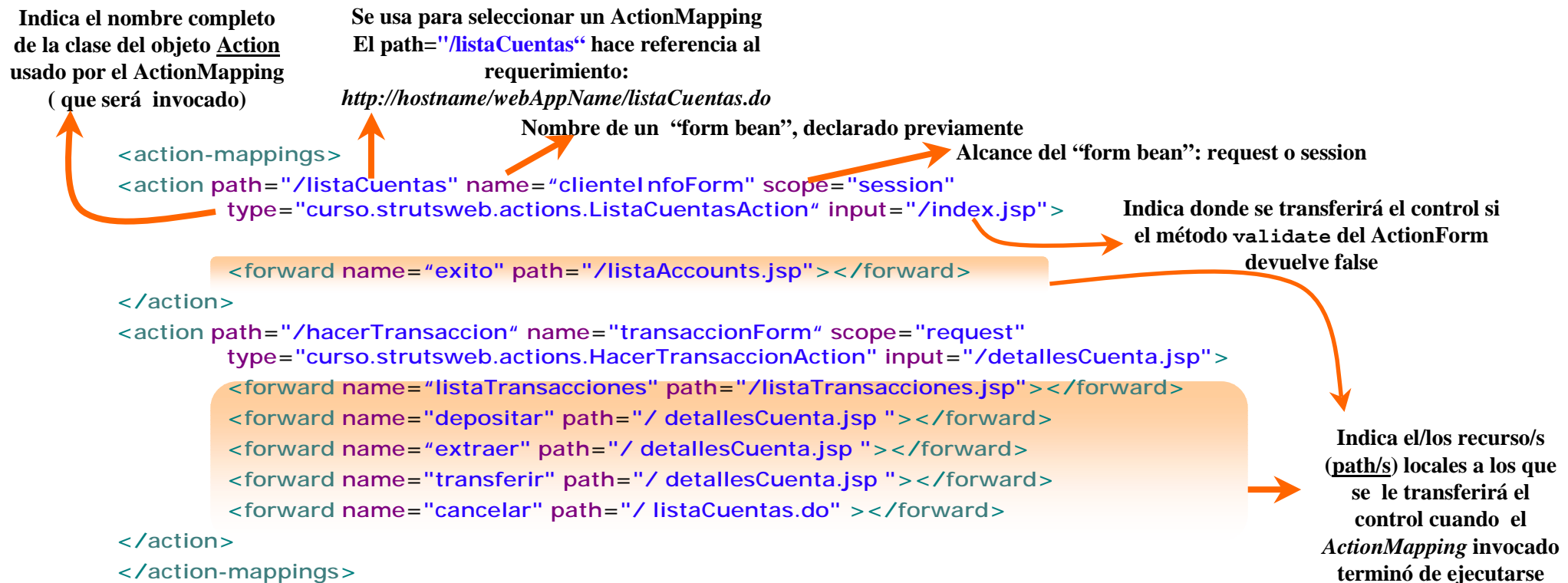
```
</action-mappings>
```



Componentes del Framework Struts

ActionMappings

- El programador define los **ActionMapping** de la aplicación en el **struts-config.xml** mediante el elemento *action*.
- Cada objeto **ActionMapping** tiene un conjunto de propiedades cuyos valores se toman del elemento *action*.





Componentes del Framework Struts

ActionForm

- Es subclase de [org.apache.struts.action.ActionForm](#). La clase base ActionForm NO puede instanciarse.
- Comúnmente se los denomina “**form bean**”. Son JavaBeans que cumplen los **siguientes roles**:
 - **Recolector de datos:** los [ActionForm](#) permiten transformar parámetros de entrada del requerimiento HTTP en propiedades de JavaBeans. Struts automáticamente *setea* las propiedades del [ActionForm](#) que coinciden con los parámetros del requerimiento HTTP, con el valor del parámetro. El programador trabaja con JavaBeans. NO hay más código del tipo: “**request.getParameter(...)**”.
 - **Buffer de datos:** los [ActionForm](#) sirven como *buffer* de los controles HTML. Mantienen la entrada hasta que sea validada y transferida a un campo del tipo apropiado. Por ej.. Si el usr. ingresa letras en un campo numérico, la entrada original se devuelve al cliente, incluyendo los caracteres inválidos. El usr. puede ver lo que tipeó mal, corregir y volver a enviar los datos. Las propiedades de los [ActionForm](#) deben ser Strings o booleans, de manera tal que cualquier entrada puede capturarse y validarse (opcionalmente) antes de transferirse a la aplicación.
 - **Validador de datos:** el método **validate** de la clase [ActionForm](#) permite invocar a métodos de la capa de negocios que saben cómo validar datos. Usualmente la validación consiste en determinar si el tipo de la entrada es correcto y luego si cimple con otros requerimientos de la capa de negocios (Ej.: 12/10/1900, es una fecha válida, pero no para el sistema).
 - **Transformador de tipos:** las propiedades de los [ActionForms](#) son Strings o booleans, sin embargo muchas aplicaciones usan propiedades como números de teléfonos o cantidades. En estos casos, se definen en el clase del [ActionForm](#) métodos *helper* que ayudan a hacer la conversión de tipos.
- Un [ActionForm](#) debe definir una propiedad por cada control HTML que debe recolectar del requerimiento.




Componentes del Framework Struts

Ejemplo de un ActionForm

```
public class ClienteInfoForm extends ActionForm {  
    private String numeroCliente = null;  
    private String nombreCliente = null;  
    private String numeroCuenta = null;  
    public String getNumerocuenta () {  
        return numeroCuenta;  
    }  
    public void setNumerocuenta(String c) {  
        this.numeroCuenta = c;  
    }  
    public void setNumerocliente(String c) {  
        this.numeroCliente= c;  
    }  
    public String getNumerocliente() {  
        return numeroCliente;  
    }  
    public String getNombrecliente () {  
        return nombreCliente;  
    }  
    public void setNombrecliente(String c) {  
        this.nombreCliente = c;  
    }  
}
```

```
public void reset(ActionMapping mapping, HttpServletRequest request)  
{  
    numeroCliente = null;  
    nombreCliente = null;  
    numeroCuenta = null;  
}  
public ActionErrors validate(ActionMapping mapping,  
                             HttpServletRequest request)  
{  
    ActionErrors errors = new ActionErrors();  
    // Bloque de validación de datos  
    return errors;  
}  
} // Fin de la clase ClienteFormBean
```



El método validate() es invocado por el ActionServlet después que las propiedades del *formbean* se setearon (se ejecutaron los `setXxx()`), pero antes que el método `execute()` del Action correspondiente se ejecute.

-Si devuelve `null` o la instancia `ActionError` tiene longitud cero, el ActionServlet invoca al `execute()` del objeto Action apropiado

-Si devuelve errores, el ActionServlet guarda estos errores como un atributo del requerimiento y quedan disponible para que se puedan mostrar en las páginas JSP por ejemplo usando el tag `<html:errors>` en la JSP. Finalmente transfiere el control al formulario de entrada identificado por el atributo *input* del ActionMapping



Componentes del Framework Struts

ActionForm

- Los **ActionForm** se declaran en el archivo [struts-config.xml](#).
- Es un barrera entre el requerimiento HTTP y el objeto Action.
- Si se quiere que el **ActionForm** valide las entradas antes de pasárselos al **Action**, entonces debe implementar el método **validate**.
- Si se quiere inicializar las propiedades antes que se completen con datos, debe implementarse el método **reset**, que será invocado antes de que el **ActionForm** sea inicializado.
- Las propiedades del **ActionForm** pueden recuperarse para mostrarse en la páginas JSP de resultado (por ej. Usando el **jsp:useBean**).
- Si un *ActionMapping* especifica un “form-bean”, el **ActionServlet**, automáticamente realizará los siguientes servicios, antes de invocar al **Action** apropiado:
 - Chequear su existencia, bajo el nombre y alcance (request, session) declarados.
 - Si no existe ninguna instancia disponible, una nueva será automáticamente creada y ligada al alcance apropiado.
 - Para cada parámetro del requerimiento HTTP cuyo nombre coincida con el nombre de una propiedad en el “form-bean”, el método *setter* correspondiente se invocará. Opera de manera similar al:

`<jsp:setProperty name="usr" property="*">`

```
<form-beans>
<form-bean name="clienteInfoForm" type="curso.strutsweb.forms.ClienteInfoForm"></form-bean>
</form-beans>
<action-mappings>
<action path="/listaCuentas" name="clienteInfoForm" scope="session" type="curso.ListaCuentasAction" input="/index.jsp">
.....
</action-mappings>
```



Componentes del Framework Struts

ActionForm- DynaActionForms

- Son instancias de [org.apache.struts.action.DynaActionForm](#)
- Permiten especificar las propiedades de los JavaBeans usando el archivo de configuración **struts-config.xml** evitando al programador la tarea de codificar un JavaBean.
- Pueden usarse en cualquier situación que se usa un ActionForm, sin necesidad de modificar ningún código Java (Actions) ni JSP.

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



Componentes del Framework Struts

```
<struts-config>
```

```
<!-- Form Beans -->
```

```
<form-beans>
```

```
<form-bean name=" clienteInfoForm " type="curso.forms.ClienteInfoForm" />
```

```
<form-bean name=" transaccionForm " type="curso.forms.TransaccionForm" />
```

```
</form-beans>
```

```
<!-- Global Forwards -->
```

```
<global-forwards>
```

```
<forward name="login" path="/logon.jsp" />
```

```
<forward name="error" path="/paginaerror.jsp" />
```

```
</global-forwards>
```

```
<!-- Action Mappings -->
```

```
<action-mappings>
```

```
<action path="/listaCuentas" name="clienteInfoForm" scope="session" type="curso.action.ListaCuentasAction" input="/index.jsp">
```

```
<forward name="exito" path="/listaAccounts.jsp" />
```

```
</action>
```

```
<action path="/transaccion" name="transaccionForm" type="curso.action.TransaccionAction" input="/detallesCuenta.jsp">
```

```
<forward name="listaTransacciones" path="/listaTransacciones.jsp" />
```

```
<forward name="depositar" path="/ detallesCuenta.jsp " />
```

```
<forward name="extraer" path="/ detallesCuenta.jsp " />
```

```
<forward name="transferir" path="/ detallesCuenta.jsp " />
```

```
<forward name="cancelar" path="/ listaCuentas.do" />
```

```
</action>
```

```
</action-mappings>
```

```
<!-- Message Resources -->
```

```
<message-resources parameter="curso.strutsweb.resource.RecursosApp" />
```

```
</struts-config>
```

Identificador única de un “form bean”, usado para ser referenciarlo en un <action-mapping>

Información sobre los “form beans” de la aplicación. Es usada para crear los objetos ActionForm en ejecución. Se define un elemento <form-bean> por cada “form bean”

Nombre completo de la subclase de ActionForm) usada para implementar el “form bean”

Los *forward globales* están disponibles para todos los Action de la aplicación web.

Los *forwards* locales sobrescriben a los globales. Si un Action retorna el mapping “error”, la página “/paginaerror.jsp” se mostrará

struts-config.xml

Especifica el nombre del archivo de recursos de la aplicación, donde se guardarán los mensajes de texto y los de error



Componentes del Framework Struts

Resumiendo, en una aplicación Struts:

- Los objetos **Action** implementan la funcionalidad de la aplicación
- El resto de las componentes de aplicación Struts proveen la infraestructura necesaria para que los objetos **Action** realicen su tarea.

Por ejemplo, si la aplicación **necesita guardar un registro en una BD:**

- El **ActionForward** proveerá un link a la página del formulario de entrada de datos.
- El **ActionForm** captura los datos de entrada.
- El **ActionMapping** configura el Action con el ActionForm y los ActionForwards apropiados.
- Al **Action** envía los datos de entrada a la BD, invocando a objetos de negocio, aunque podría hacerlo directamente sobre la BD vía JDBC.