



Struts 2 - Vista



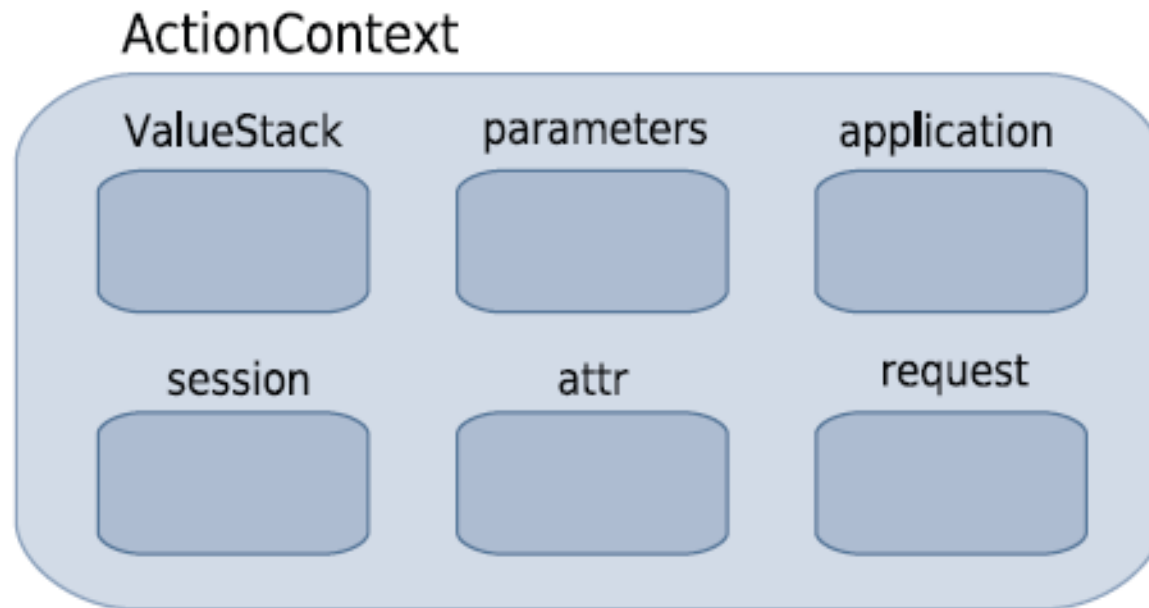
- En Struts 2, la capa de vista está encapsulada en la componente “*Result*”
- Biblioteca de tags de Struts 2
 - Provee un conjunto de tags para acceder a datos y generación de HTML
- ¿De donde vienen los datos que se muestran en la vista?
 - Cuando llega una solicitud al framework, se crean los objetos necesarios para gestionar la solicitud
 - Corresponde a los datos a nivel de aplicación, que se mantienen en el ValueStack
 - Pero el procesamiento necesita manejar más datos que los mantenidos en el ValueStack



- **ActionContext**
 - Contiene todos los datos disponibles para el procesamiento de una solicitud en el framework
 - Incluyendo el ValueStack
 - Con expresiones OGNL se puede acceder a los objetos que se mantienen
 - Por omisión, se utiliza el ValueStack para la **resolución** de objetos
- **Resolución de una expresión OGNL**
`user.account.balance`
 - Se está accediendo a la propiedad “balance” del objeto “account” del objeto “user”
 - ¿Donde está almacenado el objeto “user”?



- **ActionContext**



- `ActionContext` mantiene todos los objetos necesarios para el procesamiento de una solicitud
- `OGNL` puede acceder a cualquiera de ellos



- **ActionContext**

Name	Description
parameters	Map of request parameters for this request
request	Map of request-scoped attributes
session	Map of session-scoped attributes
application	Map of application-scoped attributes
attr	Returns first occurrence of attribute occurring in page, request, session, or application scope, in that order
ValueStack	Contains all the application-domain-specific data for the request

- Nombre y contenido de los objetos del **ActionContext**



- **ActionContext**
 - Parameters: conjunto de datos asociados a la solicitud que se está procesando
 - Request y session: conjunto de atributos a nivel de solicitud y sesión
 - Application: conjunto de atributos a nivel de aplicación
 - Attribute: concepto de atributo a nivel de servlet-api
 - Objetos arbitrarios, asociados a un nombre en el *scope* respectivo
 - Attr: conjunto especial que busca atributos a nivel de page, request, session



- OGNL selecciona uno de los objetos del `ActionContext` como raíz para las resoluciones
 - Por omisión, `ValueStack` es la raíz si es que no se especifica otro

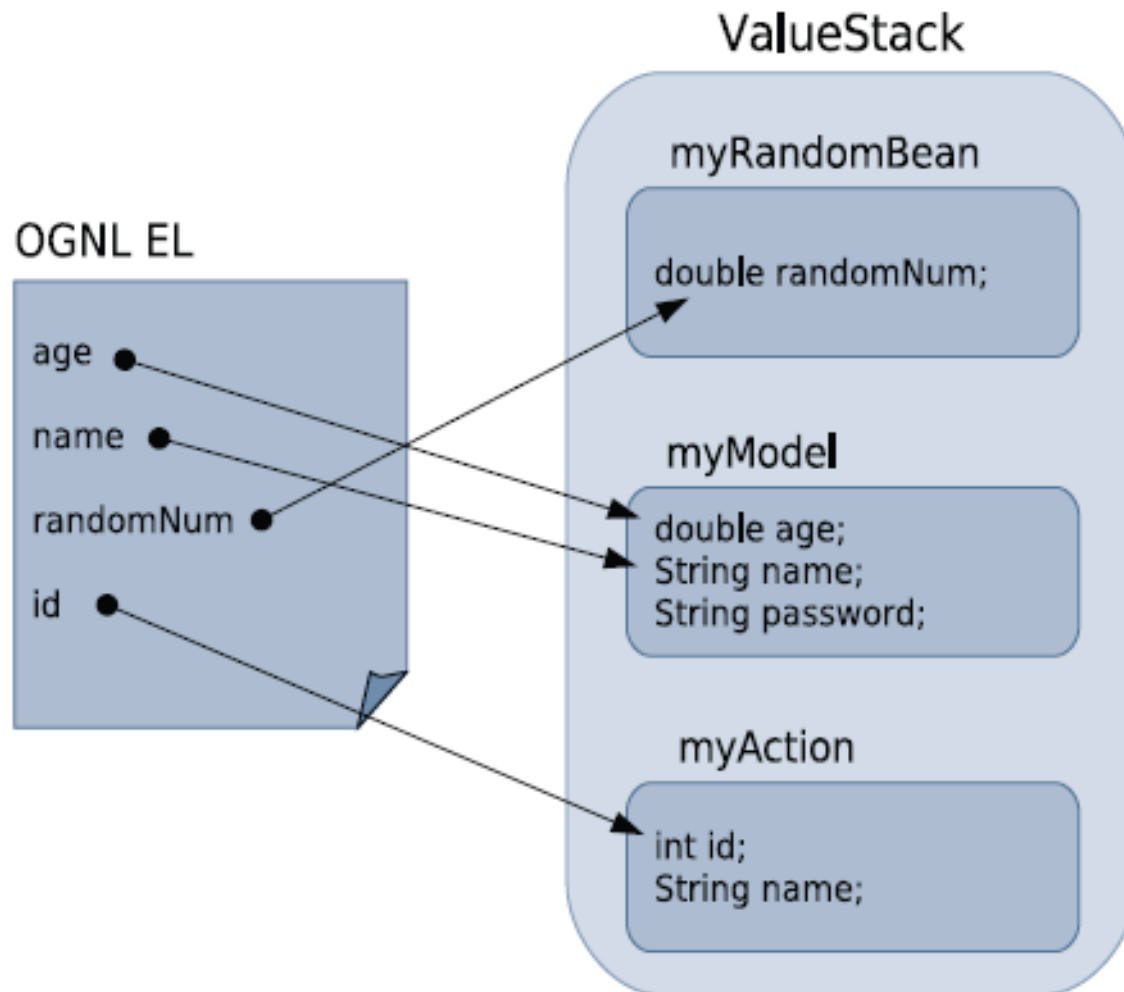
```
#session[ 'user' ]
```

- Con `#session` se está indicando que se usarán para la resolución los objetos que están a nivel de “*session*”
- Esta expresión apunta al cualquier tipo de objeto que ha sido almacenado a nivel de sesión con el nombre “*user*”



- ValueStack
 - Cuando se recibe una solicitud en Struts 2, se crea inmediatamente el ActionContext, ValueStack y el objeto action
 - Para el transporte de datos, el action se almacena en el ValueStack, par que sus propiedades se accedan vía OGNL
 - Pretende ser el único objeto contra el cual se resuelven las expresiones OGNL
 - Contiene todas las propiedades, de todos los objetos que han sido dejados en el stack
 - Si hay múltiples ocurrencias de la misma propiedad, la más reciente será accedida

- ValueStack





- Struts 2 Tags, divididos en 4 categorías
 - Datos, control de flujo, UI, varios
 - Datos: se preocupa de las formas de traer o colocar datos en el ValueStack
 - Control de flujo: provee herramientas para controlar condicionalmente el proceso de representación
- Sintaxis API
 - Una vez que se conoce el TAG que se desea usar, simplemente se lleva a la tecnología de vista seleccionada
 - JSP, Velocity o FreeMarker



- JSP
 - Tags de Struts 2 se ven de la misma forma que otros Tags JSP

```
<s:property value="name" />
```

- Es necesario tener la declaración del tag-lib al principio del archivo

```
<%@ page contentType="text/html; charset=UTF-8" %>  
<%@ taglib prefix="s" uri="/struts-tags" %>
```



- Sintaxis JSP - Velocity

```
<s:form action="Register">
  <s:textfield name="username" label="Username"/>
  <s:password name="password" label="Password"/>
  <s:textfield name="portfolioName" label="Enter a name"/>
  <s:submit value="Submit"/>
</s:form>
```

```
#sform ("action=Register")
  #stextfield ("label=Username" "name=username")
  #spassword ( "label=Password" "name=password")
  #stextfield ( "label=Enter a name" "name=portfolioName")
  #ssubmit ("value=Submit")
#end
```



- Estableciendo atributos en un TAG

```
nonExistingProperty on the ValueStack =  
    <s:property value="nonExistingProperty" />
```

```
nonExistingProperty on the ValueStack =  
    <s:property value="nonExistingProperty"  
        default="doesNotExist" />
```

- Forzando la resolución de un atributo:

```
nonExistingProperty on the ValueStack =  
    <s:property value="nonExistingProperty"  
        default="%{myDefaultString}" />
```

- Buscará el valor de “myDefaultString” en el ValueStack



- Ejemplos de tag → “chapter 6” de la aplicación de ejemplo
- Property Tag

Attribute	Required	Default	Type	Description
value	No	<top of stack>	Object	Value to be displayed
default	No	null	String	Default value to be used if value is null
escape	No	True	Boolean	Whether to escape HTML

– Ejemplo:

```
<h4>Property Tag</h4>
```

```
The current user is <s:property value="user.username"/>.
```



- Set Tag
 - Asigna una propiedad a otro nombre
 - Se puede especificar la ubicación de la nueva referencia

Attribute	Required	Type	Description
name	Yes	String	Reference name of the variable to be set in the specified scope.
scope	No	String	application, session, request, page, or action. Defaults to action.
value	No	Object	Expression of the value you wish to set.

– Ejemplo

```
<s:set name="username" value="user.username"/>  
Hello, <s:property value="#username"/>. How are you?
```



- Set Tag
 - Crea una nueva referencia al objeto
 - Se no se indica el scope quedará en el “default scope”: ActionContext
 - Especificando el scope:

```
<s:set name="username" scope="application"  
      value="user.username"/>
```

```
Hello, <s:property value="#application['username']"/>. How are  
you?
```




- Push Tag

- Permite agregar propiedades al ValueStack

```
<s:push value="user">
```

```
This is the "<s:property value="portfolioName"/>" portfolio,  
created by none other than <s:property value="username"/>  
</s:push>
```

- Bean Tag

- Híbrido entre set y push
- No se necesita trabajar con un objeto existente



- Bean Tag

Attribute	Required	Type	Description
name	Yes	String	Package and class name of the bean that is to be created
var	No	String	Variable name used if you want to reference the bean outside the scope of the closing bean tag

- Ejemplo

```
<s:bean name="org.apache.struts2.util.Counter" var="counter">
  <s:param name="last" value="7"/>
</s:bean>
<s:iterator value="#counter">
  <li><s:property/></li>
</s:iterator>
```



- Revisar Struts Tag Reference

<http://struts.apache.org/2.x/docs/tag-reference.html>