**INTRODUCCION**

Struts 2 es un framework de presentación, dentro de las capas en las que se divide una aplicación en la arquitectura JEE, el cual implementa el controlador del patrón de diseño MVC (Modelo Vista Controlador), y que podemos configurar de varias maneras; además proporciona algunos componentes para la capa de vista. Por si fuera poco, proporciona una integración perfecta con otros frameworks para implementar la capa del modelo (como Hibernate y Spring).

**Historia de Struts2**

Struts2 es un framework de código abierto usado para desarrollar aplicaciones web. Fue originalmente desarrollado por Craig R. MacClanahan, y en el año 202 pasa a pertenecer a Apache Software Foundation.

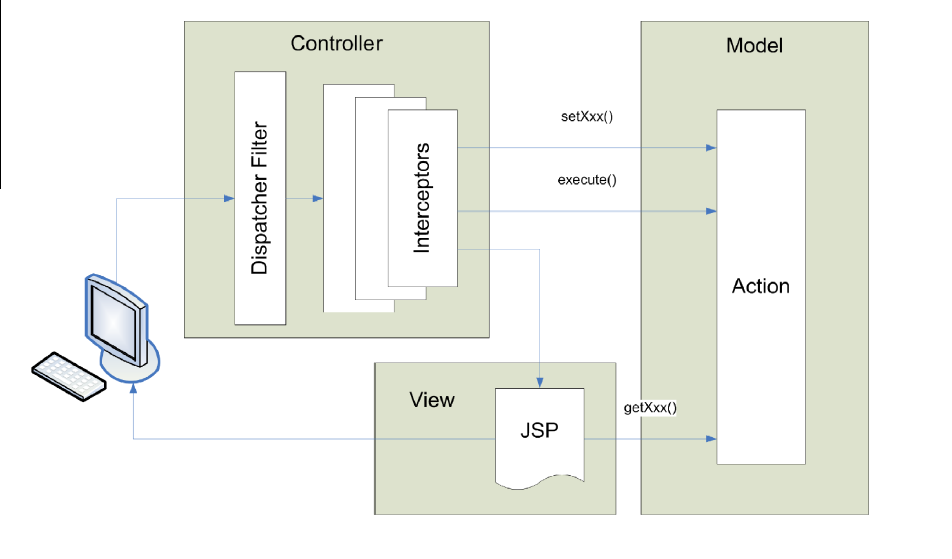
Strust 2 no está basado en Struts 1, se encuentra basado en el framework de desarrollo web de Java WebWork.

**Arquitectura MVC**

El Modelo Vista Controlador es un patrón de diseño de software para desarrollar aplicaciones WEB. El modelo vista controlador consta de las siguientes partes:

* Modelo: Es el responsable de administrar los datos de la aplicación. Responde a las peticiones de la vista y a las instrucciones del controlador
* Vista: Presenta los datos en un formato particular
* Controlador: Comunica el modelo con la vista respondiendo a eventos generados por el usuario.

Fig. Modelo Vista controlador



<https://joeljil.files.wordpress.com/2010/05/struts2mvc.png>

**Características Sruts2**

* Soporte AJAX.
* Uso de acciones y formas POJO (Plain Old Java Object).
* Fácil integración
* Soporte de Plantillas
* Soporte de Pluggins
* Facil modificación de las etiquetas.
* Soporte en etiquetas.

**Desventajas**

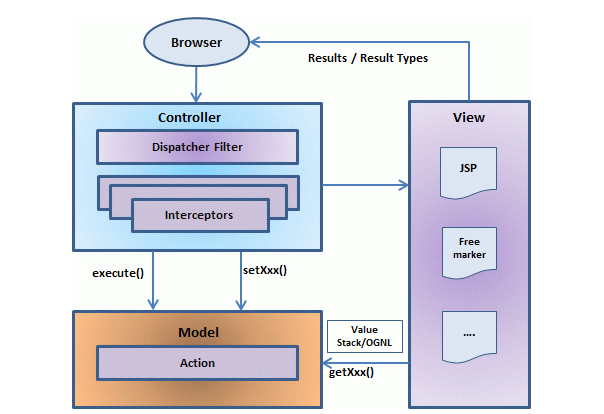
* Gran curva de aprendizaje
* Poca documentación
* Menos transparencia

**Componentes principales**

* Actions
* Interceptors
* Value Stack / OGNL
* Results / Result types
* View technologies

**Ciclo de vida de una petición.**

1. Se envía una petición por parte del usuario.
2. El elemento FilterDispatcher determina que acción que tendrá que responder a la petición
3. Se aplican los interceptores definidos.
4. Se ejeccuta el action.
5. Se renderiza la salida
6. Se devuelve la petición
7. Se muestra el resultado al usuario.



Actions

Las acciones o Actions son clases encargadas de realizar la lógica para servir una petición. Cada URL es mapeada a una acción específica, la cual proporciona la lógica necesaria para servir a cada petición hecha por el usuario.

Results

Después que un Action ha sido procesado se debe enviar la respuesta de regreso al usuario, esto se realiza usando results.

El tipo de result por default es "dispatcher".

Un Action puede tener más de un result asociado. Esto nos permitirá enviar al usuario a una vista distinta dependiendo del resultado de la ejecución del Action. Por ejemplo en caso de que todo salga bien, enviaremos al usuario al result "sucess", si algo sale mal lo enviaremos al result "error", o si no tiene permisos lo enviaremos al result "denied".

El tipo del result indica cómo debe ser tratado el resultado que se le regresará al cliente. Por ejemplo un tipo de Result puede enviar al usuario de vuelta una JSP (lo que haremos más a menudo), otro puede redirigirlo hacia otro sitio, mientras otro puede enviarle un flujo de bytes (para descargar un archivo por ejemplo).

Interceptores

Los interceptores son clases que siguen el patrón interceptor. Estos permiten que se implementen funcionalidades cruzadas o comunes para todos los Actions, pero que se ejecuten fuera del Action (por ejemplo validaciones de datos, conversiones de tipos, población de datos, etc.).

Ejemplos de interceptores

| **Interceptor** | **Nombre** | **Descripción** |
| --- | --- | --- |
| **Alias** | **alias** | Permite que los parámetros tengan distintos nombres entre peticiones. |
| **Chaining** | **chaining** | Permite que las propiedades del **Action** ejecutado previamente estén disponibles en el **Action** actual |
| **Checkbox** | **checkbox** | Ayuda en el manejo de checkboxes agregando un parámetro con el valor "**false**" para checkboxes que no están marcadas (o checadas) |
| **Conversion Error** | **conversionError** | Coloca información de los errores convirtiendo cadenas a los tipos de parámetros adecuados para los campos del **Action**. |
| **Create Session** | **createSession** | Crea de forma automática una sesión **HTTP** si es que aún no existe una. |