



## Struts 2



- Framework?
  - Es una pieza de software estructurada
  - Intenta proveer una plataforma sobre la cual se pueden construir aplicaciones rápidamente
    - Automatiza algunas tareas
    - Agrega una solución de arquitectura elegante para el “workflow” común de la aplicación
- Por que usar uno?
  - Ahorro de tiempo en aplicaciones grandes
    - Se dedica el tiempo a tareas más importantes que las comunes de flujo de datos
  - Masificación de frameworks

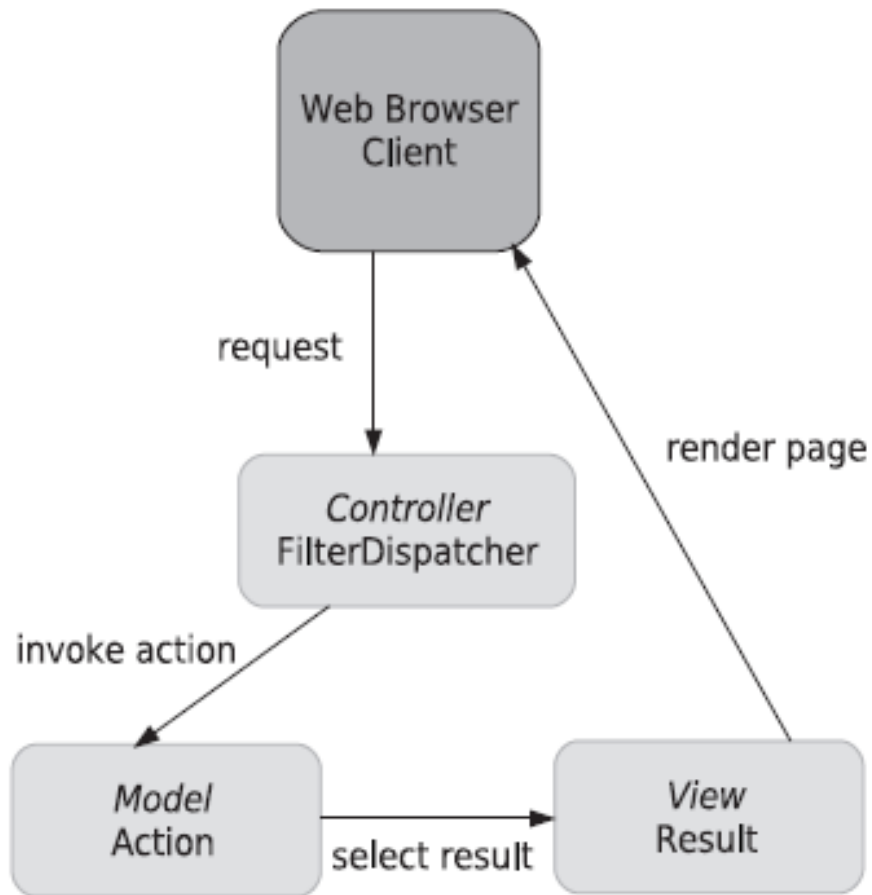


- Struts 2
  - No es la segunda versión de Struts
  - Framework para aplicaciones web de “segunda-generación”
    - Implementa el patrón de diseño MVC
  - Se aprendió de las debilidades de Struts 1 para hacer una mejor implementación de MVC
- MVC
  - Provee una separación de conceptos que se aplican a sistemas web
  - Se maneja la complejidad de un sistema dividiéndolo en componentes de alto nivel



- MVC
  - El patrón de diseño identifica 3 componentes distintas: modelo, vista y controlador
  - En Struts 2 son implementadas por “action”, “result” y “FilterDispatcher”
- Controlador
  - Su trabajo es asociar requerimientos con acciones
  - No es necesario preocuparse por manipular los requerimientos
    - Solo se necesita informar la asociación entre solicitudes y acciones
    - Se puede hacer con un archivo XML o con anotaciones

# Struts





- Struts 2 apunta a tener cero configuraciones de aplicaciones web
  - Deja el trabajo a los metadatos de la aplicación
  - Se usa las anotaciones Java como eje principal
- Modelo
  - Es el estado interno de la aplicación
  - El controlador entrega el control del requerimiento que se está atendiendo
    - Se preparan los datos y se ejecuta la lógica de negocio
  - Cuando termina la ejecución, reenvía los resultados a la componente de la Vista

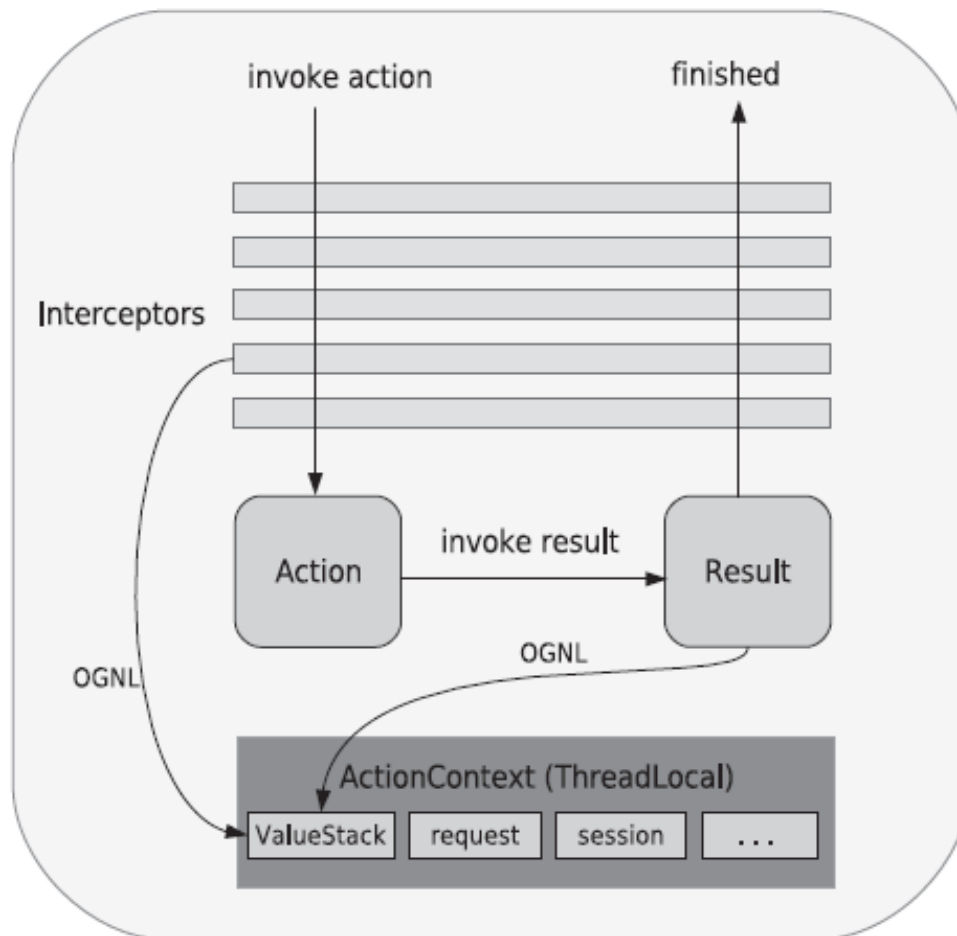


- Vista
  - Es la componente de visualización del patrón de diseño MVC
  - Es la que representa el estado de la aplicación al usuario
  - Comúnmente son páginas JSP
  - Con clientes enriquecidos y aplicaciones Ajax se complican los detalles de la vista
    - Ayuda tener una clara separación de conceptos con MVC

# Struts



- Workflow de procesamiento de un requerimiento







- Interceptores
  - La invocación de una acción pasa a través del stack de interceptores
    - Es una parte clave de Struts 2
  - Son invocados antes y después de la acción
  - No es necesario que tengan que ejecutar algo, pero siempre tienen la oportunidad
  - Permite que las tareas comunes sean definidas de manera limpia
    - Componente reusable que se puede mantener separada del código de la acción
  - Ejemplo: tareas de log, subir archivos, conversiones...

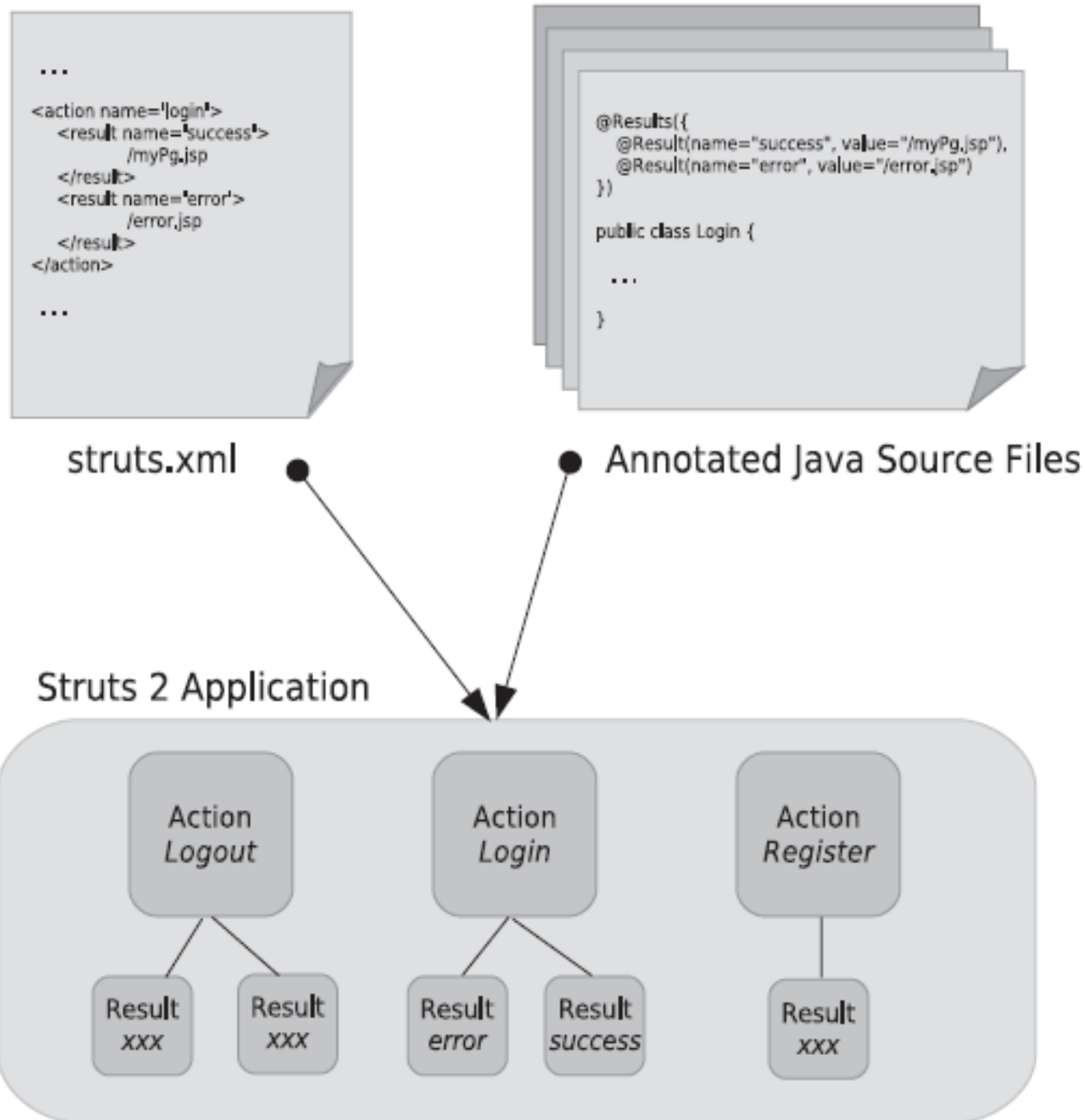


- ValueStack y OGNL
  - ValueStack: es un área de almacenamiento
    - mantiene todos los datos relacionados con el procesamiento de un requerimiento
    - En lugar de pasar los datos por varios lados, los mantiene en una ubicación central
  - OGNL: herramienta que permite acceder al ValueStack
    - Es un “Expression language” que permite manipular y referenciar a los datos en el ValueStack
  - El “ActionContext” contiene todos los datos del contexto en que ocurre una acción
    - Esto incluye el ValueStack



- Arquitectura Declarativa
  - Tipo de configuración que permite describir la arquitectura a un alto nivel
  - Se describe la arquitectura en artefactos del alto nivel:
    - Anotaciones Java o archivos XML
    - Desde estos artefactos se crearán las instancias para tiempo de ejecución de la aplicación
  - El desarrollador necesita declarar que objetos servirán las acciones, resultados e interceptores
    - Principalmente se especifica que clase Java implementará la interfaz necesaria

# Struts





- Arquitectura declarativa XML
  - Ejemplo de archivo *struts.xml*

```
<action name="Login" class="manning.Login">
  <result>/AccountPage.jsp</result>
  <result name="input">/Login.jsp</result>
</action>

<action name="Registration" >
  <result>/Registration.jsp</result>
</action>

<action name="Register" class="manning.Register">
  <result>/RegistrationSuccess.jsp</result>
  <result name="input">/Registration.jsp</result>
</action>
```



- Arquitectura declarativa Java-annotation
  - Se agrega metadata directamente a clase que implementa “*Action*”

```
@Results({
    @Result(name="input", value="/RegistrationSuccess.jsp" )
    @Result(value="/RegistrationSuccess.jsp" )
})

public class Login implements Action {

    public String execute() {

        //Business logic for login

    }
}
```



- Qué método usar?
  - Lo más importante es entender los conceptos detrás de la arquitectura declarativa
  - Si se entienden bien será trivial moverse de XML a Java-annotations
- “Intelligent defaults”
  - Componentes predefinidas
  - Provee componentes que permiten realizar las tareas más comunes de una aplicación con el mínimo desarrollo
    - Se declaran en struts-default.xml

# Struts



- Ejemplo: HelloWorld
  - Usando XML
  - Usando anotations



# Struts



Form Rendered by NameCollector.jsp

```
<form action="HelloWorld.action" >  
  <input type="text"  
    name="name"/>  
  <input type="submit"/>  
</form>
```

ValueStack

*HelloWorld Action*

String name  
String customGreeting

HelloWorld.jsp

Custom Greeting Page  
<s:property value="customGreeting"/>