Struts 2

Contents

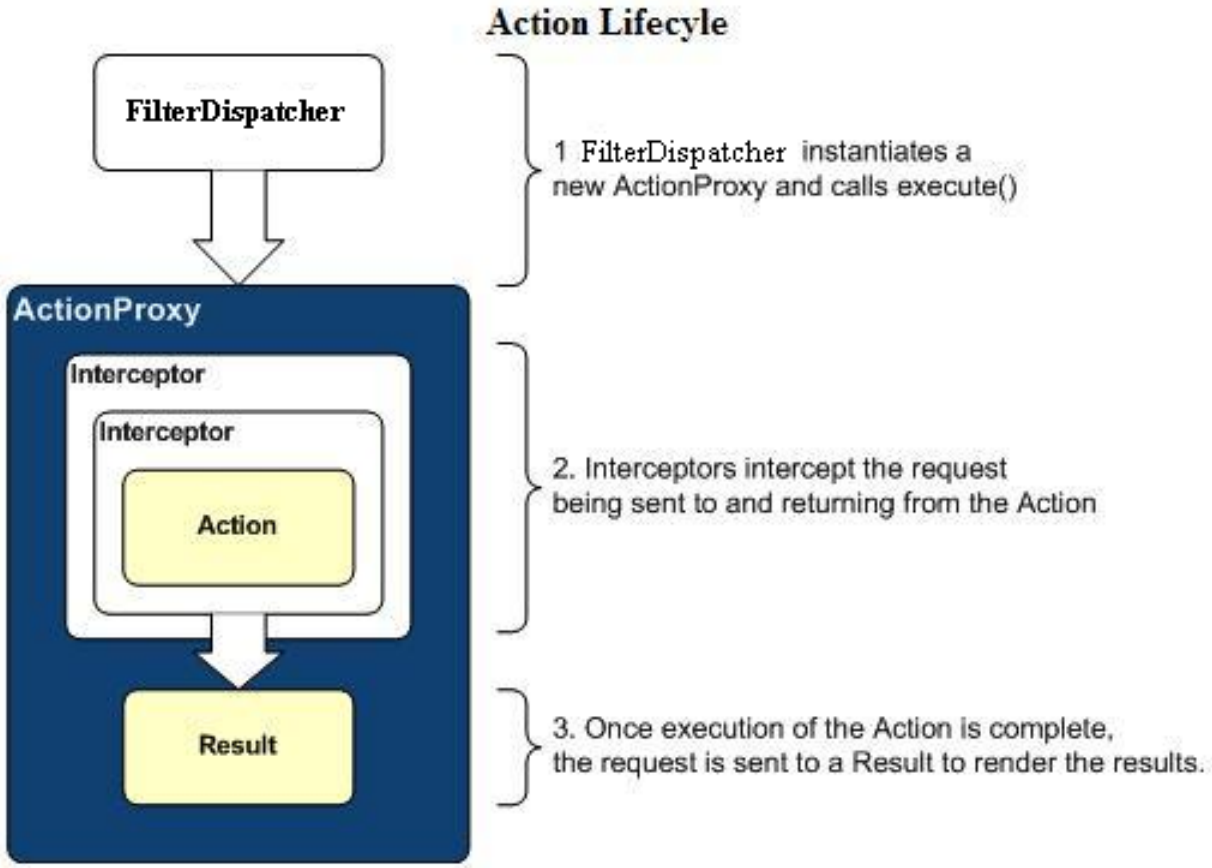
- **Interceptors**
- **File Upload & Download**

Interceptor

Why Interceptors?

- Many Actions share common concerns.
  - Logging, Validation, File Upload, Double-submit guard etc.
- The framework makes it easy to share solutions to these concerns using an "Interceptor" strategy.

Interceptor in Action Life-cycle



Interceptors

- Interceptors can execute code before and after an Action is invoked.
- Most of the framework's core functionality is implemented as Interceptors.
  - Features like double-submit guards, type conversion, object population, validation, file upload, page preparation, and more, are all implemented with the help of Interceptors.
- Each and every Interceptor is pluggable

Interceptors

- The default Interceptor stack is designed to serve the needs of most applications.
- Most applications will **not** need to add Interceptors or change the Interceptor stack.

Interceptors & Actions

- In some cases, an Interceptor might keep an Action from firing, because of a double-submit or because validation failed.
- Interceptors can also change the state of an Action before it executes.

Configuration of Interceptors

- Interceptors can be configured on a per-action basis.
- Your own custom Interceptors can be mixed-and-matched with the Interceptors bundled with the framework.
- The Interceptors are defined in a stack that specifies the execution order.
  - In some cases, the order of the Interceptors on the stack can be very important.

Configurating Interceptors

- **struts.xml**

```
<package name="default" extends="struts-default">
    <interceptors>
        <interceptor name="timer" class=".."/>
        <interceptor name="logger" class=".."/>
    </interceptors>


    <action name="login" class="tutorial.Login">
        <interceptor-ref name="timer"/>
        <interceptor-ref name="logger"/>
        <result name="input">login.jsp</result>
        <result name="success"
                type="redirect-action">/secure/home</result>
    </action>
</package>
```

Stacking Interceptors

- Most web applications want to apply the same set of Interceptors over and over again.
- Instead of reiterating the same list of Interceptors, we can use an Interceptor Stack to bundle these Interceptors together.

Stacking Interceptors

```
<package name="default" extends="struts-default">
    <interceptors>
        <interceptor name="timer" class=".."/>
        <interceptor name="logger" class=".."/>
        <interceptor-stack name="myStack">
            <interceptor-ref name="timer"/>
            <interceptor-ref name="logger"/>
        </interceptor-stack>
    </interceptors>
<action name="login" class="tutuorial.Login">
    <interceptor-ref name="myStack"/>
    <result name="input">login.jsp</result>
    <result name="success"
            type="redirect-action">/secure/home</result>
</action>
</package>
```

Interceptor Interface

- Interceptors must implement the

com.opensymphony.xwork2.interceptor.Interceptor interface

- The *AbstractInterceptor* class provides an empty implementation of init and destroy, and can be used if these methods are not going to be implemented.

```
public interface Interceptor extends Serializable {
        void destroy();
        void init();
    String intercept(ActionInvocation invocation)
        throws Exception;
}
```

Example: Interceptor

```
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;

public class SimpleInterceptor extends AbstractInterceptor {

    public String intercept(ActionInvocation invocation) throws
    Exception {
       MyAction action = (MyAction)invocation.getAction();
       action.setDate(new Date());
       return invocation.invoke();
    }
}
```

Framework Interceptors

- Struts 2 framework provides an extensive set of ready-to-use interceptors
  - Parameter interceptor: Sets the request parameters onto the Action.
  - Scope interceptor: Simple mechanism for storing Action state in the session or application scope.
  - Validation interceptor: Performs validation using the validators defined in action-validation.xml
  - Many more
- Configured in *struts-default.xml*

File Upload & Download

File Upload & Download

- Struts 2 utilizes the service of **File Upload Interceptor** to add the support for uploading files in the Struts applications.
- The Struts 2 File Upload Interceptor is based on MultiPartRequestWrapper, which is automatically applied to the request if it contains the file element.
- The Struts 2 FileUpload component can be used to upload the multipart file in your Struts 2 application.

File Upload & Download

- The file upload interceptor also does the validation and adds errors, these error messages are stored in the struts-messsages.properties file. The values of the messages can be overridden by providing the text for the following keys:
- **struts.messages.error.uploading** - error when uploading of file fails
- **struts.messages.error.file.too.large** - error occurs when file size is large
- **struts.messages.error.content.type.not.allowed** - when the content type is not allowed

File Upload & Download

- **Parameters** to control the file upload functionality.
  - **maximumSize:** optional. The default value is 2MB.
  - **allowedTypes:** optional. It specify the allowed content type.