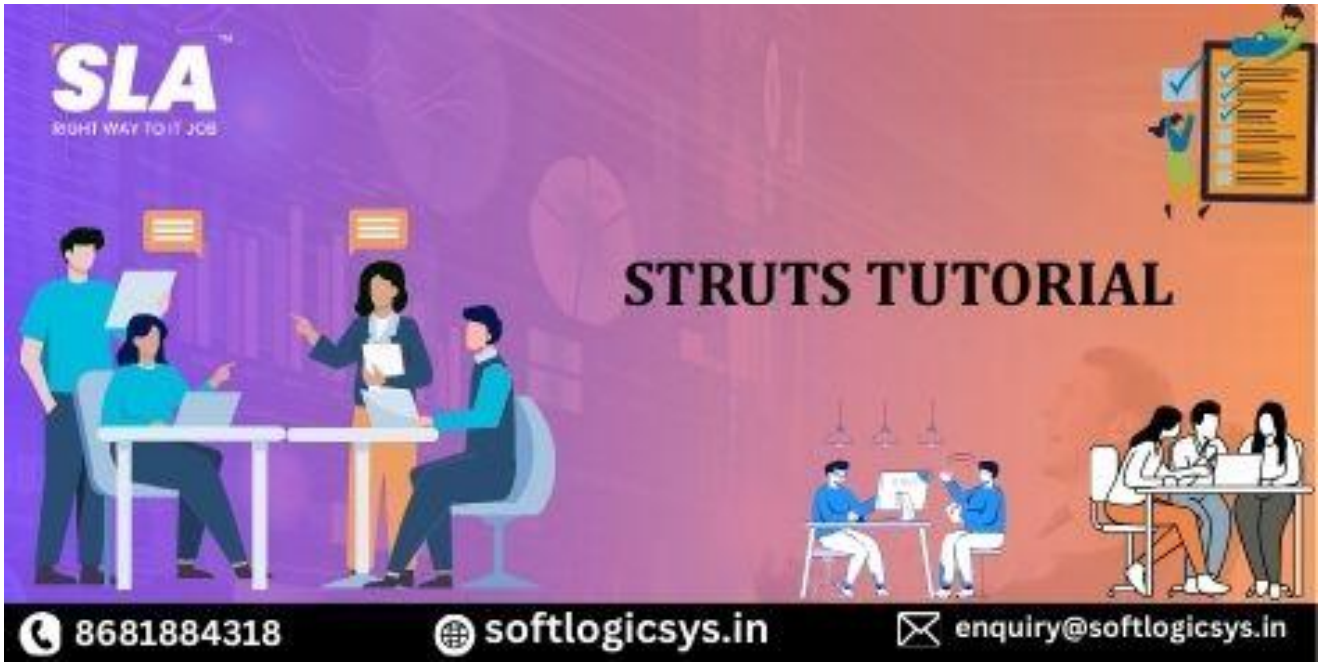

Struts Tutorial

Introduction

Apache Struts is an open-source MVC web application framework for creating beautiful Java web apps. Beginners find it difficult due to its XML configuration and MVC flow (Action, Result, Interceptors). This Struts tutorial for beginners will demystify these complicated concepts with an emphasis on real-world implementation. Ready to create enterprise-class apps? Download our entire **Struts course syllabus** to view your entire learning journey!

Why Students or Freshers Learn Struts

Learning Apache Struts provides substantial career benefits to students and Java development freshers:

- **Legacy Systems:** Large companies continue to employ Struts 1 or Struts 2 within their stable, mission-critical applications. Familiarity with Struts makes you a desirable candidate for supporting or updating these systems.

- **Core Concepts:** Struts adheres to core MVC (Model-View-Controller) design concepts, which are vital in learning contemporary frameworks such as Spring MVC and Jakarta EE (Java EE).
- **Job Market:** It places a universally accepted, sought-after skill on your resume, increasing your prospects in a saturated job market in which support and legacy maintenance are key roles.

Ready to ace your first tech interview? Download our list of top **Struts Interview Questions and Answers**!

Step-by-Step Struts Tutorial for Beginners

Here is the step-by-step Struts tutorial in one response, covering installation, basic concepts, and code, will be difficult to accommodate within the platform's boundaries without compromising on quality.

Step 1. Project Setup and Installation

The simplest start is with a contemporary IDE such as Eclipse and a build tool such as Maven.

1.1 Prerequisites

- **Java Development Kit (JDK):** Version 8 or higher installed.
- **IDE:** Eclipse IDE for Java EE Developers.
- **Web Server:** Apache Tomcat (version 8.5 or higher) set up in Eclipse.
- **Build Tool:** Maven (typically integrated into Eclipse).

1.2 Create a Maven Web Project

- In Eclipse, navigate to **File → New → Maven Project.**
- Choose **“Create a simple project (skip archetype selection)”**.
- Fill in Group Id *(e.g., com.mycompany),* Artifact Id *(e.g., Struts2HelloWorld),* and make **Packaging** equal to *war* (Web Application Archive).
- Right-click the project, navigate to **Properties → Project Facets**, and mark **Dynamic Web Module** as selected (version 3.0 or higher) and **Java** as a recent version.

1.3 Add Struts 2 Dependencies (pom.xml)

Open the *pom.xml* file and include the following necessary dependencies. Struts 2 has a minimal requirement for these two:

```
<dependencies>

    <dependency>

        <groupId>org.apache.struts</groupId>

        <artifactId>struts2-core</artifactId>

        <version>2.5.30</version> </dependency>

    <dependency>

        <groupId>javax.servlet</groupId>

        <artifactId>javax.servlet-api</artifactId>

        <version>4.0.1</version>

        <scope>provided</scope>
```

```
    </dependency>

</dependencies>
```

Action: Right-click on the project → Maven → Update Project in order to download the JAR files.

Step 2. Setting up the Deployment Descriptor *(web.xml)*

The *web.xml* file (within *src/main/webapp/WEB-INF/*) needs to be set up in order to utilize the **Struts 2 Filter Dispatcher**. This filter traps all the incoming requests and sends them through the Struts framework.

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

                        http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"

    version="4.0">

    <display-name>Struts2HelloWorld</display-name>

    <filter>

        <filter-name>struts2</filter-name>


<filter-class>org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteFilter</filter-class>
```

```
    </filter>
    <filter-mapping>
        <filter-name>struts2</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

Concept: The **Filter Dispatcher** is the Struts 2 **Front Controller** in the MVC implementation. It's a single point of entry for web requests.

Step 3. The Model Component: The Action Class

The **Action** class is the Struts 2 **Model** component where business logic is performed. A standard Struts 2 *Action* class implements the Action interface (although not strictly necessary) and has the *execute()* method as the default entry point.

3.1 Make the Action Class

Make a package (for example, *com.mycompany.action*) and the *HelloWorldAction.java* file.

```
package com.mycompany.action;

import com.opensymphony.xwork2.ActionSupport;
```

```
// ActionSupport provides a default implementation of the Action interface and useful methods like getText() for i18n.
public class HelloWorldAction extends ActionSupport {
    // 1. Model Property (Input/Output)
    private String username;
    // 2. Default Action Method – Business Logic
    @Override
    public String execute() throws Exception
{
        // Simple business logic: check if the username is set
        if (username != null && !username.isEmpty()) {
            return SUCCESS; // SUCCESS is a static String defined in ActionSupport
        }
        return INPUT; // INPUT is returned if validation/input is needed
    }
    // 3. Getter/Setter for the 'username' property
    public String getUsername() {
        return username;
```

```
    }

    public void setUsername(String username) {

        this.username = username;

    }

}
```

Concept: Struts 2 employs an **Interceptor Stack** to populate automatically the username property in the Action class through the *setUsername()* method using the form parameter of the same name. This is one of the fundamental aspects of the framework's **data transfer mechanism**.

Step 4. The Controller Component: struts.xml

The *struts.xml* file (located directly in *src/main/resources/ or src/main/java/*) is the core configuration file. It takes an incoming URL request and maps it to an Action class and its resulting View page.

4.1 Create struts.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE struts PUBLIC

    "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"

    "http://struts.apache.org/dtds/struts-2.5.dtd">

<struts>

    <package name="default" namespace="/" extends="struts-default">
```

```
        <action name="hello" class="com.mycompany.action.HelloWorldAction">
            <result name="success">/WEB-INF/jsp/helloSuccess.jsp</result>
            <result name="input">/index.jsp</result>
        </action>
    </package>
</struts>
```

Concept: *struts-default* package plays an important role; it holds the default **Interceptor Stack** (responsible for parameter setting, validation, etc.) and a default set of **Result Types** (such as dispatcher for JSP).

Step 5. The View Component: JSP Pages

The View component is usually a JSP (JavaServer Page) that displays data. We will require two JSPs: one for input *(index.jsp)* and one for success *(helloSuccess.jsp).*

5.1 Input View (index.jsp)

Put this file in *src/main/webapp/.*

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<head>
    <title>Hello World Input</title>
</head>
```

```
<body>
    <h1>Welcome to Struts 2!</h1>
    <s:form action="hello">
        <s:textfield name="username" label="Enter Your Name" />
        <s:submit value="Say Hello" />
    </s:form>
</body>
</html>
```

5.2 Success View *(helloSuccess.jsp)*

Put sensitive JSPs in the *WEB-INF* directory *(src/main/webapp/WEB-INF/jsp/)* to avoid direct URL access.

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<head>
    <title>Success!</title>
</head>
<body>
    <h1>Hello, <s:property value="username"/>!</h1>
```

* This data was processed by your Action class. *

* Go Back *

</body>

</html>

Concept: The *<s:property value="username"/>* tag is an example of the Struts 2 Tag Library. It directly retrieves the value of the *username* property from the *HelloWorldAction* object that had been saved into the Action Context by the framework.

Step 6. Running the ApplicationDeploy

- **Deploy:** Right-click on the project → **Run As** → **Run on Server** and choose your configured **Tomcat** server.
- **Access:** Open a web browser and go to:

http://localhost:8080/Struts2HelloWorld/index.jsp (or the root of the project).

- **Test:**
    - The form *index.jsp* is shown.
    - Enter a name **(e.g., "Alice")** and click on **Say Hello**.
    - The request is routed to */hello.action*.
    - The Struts Filter calls the *HelloWorldAction.execute()* method.
    - The *execute()* method returns *SUCCESS*.
    - Struts looks up the *SUCCESS* result in *struts.xml* and redirects the user to */WEB-INF/jsp/helloSuccess.jsp*.
    - The success page shows the customized greeting based on the data from the Action class.

Step 7. Fundamental Struts 2 Concepts for Expertise

This simple app illustrates the fundamental **MVC process**. To move on, concentrate on these concepts:

7.1 The Interceptor Stack

- **What it does:** A series of interceptors (pre-processors) which are called before and after the execute() method of the Action.
- **Typical Use:** Parameter setting (params interceptor), workflow control, validation, file upload, and error handling.
- **Configuration:** Specified in struts-default.xml and overridden in struts.xml.

7.2 Action Context and Value Stack (OGNL)

- **Action Context:** A holder that stores all request-related information, such as session, application attributes, and parameters.
- **Value Stack:** The heart of Struts 2. It's a stack-based framework (utilizing **OGNL – Object-Graph Navigation Language)** that stores the Action object and other entities. Struts 2 tags <s:property> resolve values from this stack. This simplifies data retrieval in JSPs considerably.

7.3 Validation

Struts 2 provides two primary means to validate user input:

- **Programmatic Validation:** Override your Action class's validate() method. If validation is unsuccessful, invoke addActionError() or addFieldError(), and INPUT result is returned automatically.
- **XML Validation:** Put a file named ActionClassName-validation.xml in the same package where your Action class is. It's declarative and neater for complicated validation rules.

7.4 Separate Configuration (Convention Plugin)

Though XML *(struts.xml)* is the old method, contemporary Struts 2 applications commonly employ the Convention Plugin.

- This plugin takes advantage of naming conventions (e.g., all Actions in a package suffixed with *action*) to set up the framework automatically, **removing most of the *struts.xml*** file and accelerating development.

Want to Address Real-World Challenges? The basic flow is the first step, but actual applications require dealing with security, rich validation, and AJAX integration.

Download our handpicked collection of **Advanced Struts Challenges and Solutions** to ace subjects such as Interceptor creation, file upload, and Spring integration!

Real Time Examples for Struts Tutorial for Learners

Here are some real time examples showing how Apache Struts is utilized within web applications, ideal for students:

1. User Authentication System (Login/Logout) → The Classic MVC Flow

Objective: Authenticate user access securely.

- **Struts Parts:** An **Action Class** (eg, *LoginAction*) executes the business logic (checking credentials against a database). An **Interceptor** (such as the *authentication interceptor*) may verify if the user is logged in already before entering restricted pages. The *struts.xml* directs the **SUCCESS or ERROR** string returned to the dashboard or login page, respectively.
- **Key Concept:** This illustrates the basic Model-View-Controller (MVC) request cycle and Interceptors for cross-cutting issues (security).

2. Basic Employee Data Management (CRUD Application) → Form and Data Handling

Objective: Create, Read, Update, and Delete **(CRUD)** employee records.

- **Struts Components:** Various **Action Methods** (e.g., *save(), edit(), delete()*) in the same Action class are utilized to manage various form submissions. Struts Tags (*<s:textfield>, <s:select>*) are used in the **JSP View** to bind form data directly to the Action class properties *(Model)*.
- **Key Concept:** Demonstrates Form Handling, Data Binding via the Value Stack (OGNL), and using multiple methods in one Action for clear organization.

3. Product Search and Listing → Displaying Collections using Value Stack

Objective: Show a list of products corresponding to user search parameters.

Struts Components: The **Action Class** retrieves a list of *Product* objects from a service layer (the **Model**). The **View** (JSP) utilizes the strong Struts tag *<s:iterator>* to iterate on the list property that is exposed by the Action and output a table or grid of results.

Key Concept: Concentrates on passing and displaying intricate **Collection** data types efficiently from the Action to the View through the **Value Stack.**

Let's put these concepts into practice? Check out our list of **Struts Project Ideas** to get started on building your portfolio today!

FAQs About Struts Tutorial for Beginners

1. What is a Strut?

A Strut (or Apache Struts) is an open-source, free web application development framework for developing sophisticated, enterprise-level Java web applications. It offers a rich, standardized framework founded on proven design patterns, significantly cutting development time in comparison to using Java Servlets and JSPs alone.

2. What is the use of Struts?

Struts is mainly applied to enforce the Model-View-Controller (MVC) architectural pattern in Java web applications. It controls the flow of control from a user request to the execution of business logic and ultimately to displaying the result (View), keeping code organized and maintainable.

3. What is a Strut in Java?

Struts, in the context of Java, is a full-featured MVC framework. It provides needed pieces such as a Front Controller (FilterDispatcher), Action classes, and XML configuration files (struts.xml) to integrate Java code (Model) with HTML/JSP (View) seamlessly during web development.

4. Is Java Struts still used?

Yes, but primarily Struts 2, not Struts 1. Numerous big, established companies (particularly in government and finance) still use and support applications developed with Struts 2. Newer frameworks dominate new development, yet knowledge of Struts remains useful to maintain legacy systems and to modernize them. Explore **Struts salary for freshers** here.

5. What is MVC in Struts?

MVC (Model-View-Controller) in Struts divides the application into three layers: the Model (the Action class and business logic), the View (JSP/HTML pages), and the Controller (the Struts Filter and the configuration that maps requests). This division makes the application simpler to test and maintain.

6. Why are Struts important?

Struts are valuable since they give a solid structure, impose best practices such as MVC, and supply features such as input validation, internationalization, and security through its Interceptor Stack. Standardization is essential for big teams developing long-term enterprise applications that are complex.

7. What is the difference between Spring and Struts?

Spring is a modular, full-featured framework that provides solutions for all but the last layer (DI, ORM, Web, Security). Struts is largely a web-tier MVC framework. Spring

MVC is Spring's web module, and it directly competes with Struts 2, with more configurable and integrated configuration with the broader Spring environment.

8. Can we integrate Spring with Struts?

Yes, indeed. There is a typical pattern of using Struts 2 for the web tier (Controller/View) and Spring for the Model/Service tier (business logic, database transactions, and dependency injection). Struts 2 provides a dedicated plugin that makes it easy to integrate with the Spring container for handling Action object lifecycles.

9. What is the future of Java?

Java's future is still solid and relevant. It remains the most pervasive language for enterprise software, large backend systems, Android applications, and big data technology. With ongoing advancements such as modern language features, improved performance (Project Loom), and improved memory management, its reliability provides a stable future.

10. Is Java outdated in 2025?

No, Java is not old in 2025. It is extremely well-maintained by Oracle and the open-source developers, with significant releases every six months. Its huge ecosystem, great tooling, better performance for high-throughput applications, and strong integration with cloud-native development frameworks like Spring Boot make it continue to lead enterprise development.

Conclusion

You have now successfully made it through the fundamental steps of creating your initial Struts 2 MVC application through this Struts tutorial for beginners. You've learned the critical sequence: from the Filter Dispatcher intercepting the request, the Action Class executing the logic, and the Value Stack passing data to the eventual JSP View.

This sets you up to comprehend any large, enterprise Java application. Struts 2 proficiency is a most excellent resume enhancer.

Ready to take your skills a step higher and be a master of advanced subjects such as Interceptors, Validation, and Spring integration? Join now our in-depth **Struts course in Chennai** and become certified!