**Creating Custom Tags in JSP**

**Julio Pochet Edmead**  
Bellevue University – CSD 430 Module 10  
10/05/2025

**Introduction**

When a web application grows in size, it can become hard to manage if business logic and HTML markup live together in the same JSP file. A simple solution to this problem is creating **custom tags**. Custom tags allow developers to wrap reusable Java logic inside easy-to-read elements that look like regular HTML tags. This makes code cleaner, easier to maintain, and more team-friendly. In this paper, I’ll explain what custom tags are, why they matter, the pros and cons of using them, the steps required to build one, and my personal opinion about when they’re worth the effort.

**What Custom Tags Are**

A custom tag in JSP is a user-defined component that runs Java logic when the tag is processed on a page. Instead of writing raw Java code between <% … %> scriptlets, a developer can use a tag such as <demo:hello/> that triggers Java logic behind the scenes. According to Oracle (n.d.), a custom tag acts as a bridge between presentation and business logic by letting developers build reusable functions that can be dropped into any JSP page.

Each custom tag is made up of two main parts:

1. A **Tag Handler Class** – a Java class that usually extends SimpleTagSupport and defines what the tag does.
2. A **Tag Library Descriptor (TLD)** – an XML file that tells JSP the tag’s name, class location, and attributes.

When the JSP engine encounters the tag, it creates the class, runs its doTag() method, and outputs the result to the browser.

**Advantages of Custom Tags**

The first major advantage is **reusability**. Once a tag is built, it can be reused across pages or projects without rewriting code.

Another strength is **readability**. A short tag like <user:welcome/> looks much cleaner than a large block of Java code. This helps other developers—and even designers—understand what’s happening without digging through logic.

Custom tags also help **organize teamwork**. Developers who focus on the back end can handle tag logic, while front-end designers simply use those tags. This separation of concerns makes development faster and reduces errors.

Finally, custom tags encourage **consistency**. Teams can create a shared library of tags for common functions such as formatting, validation, or authentication. According to the Apache Software Foundation (n.d.), using the SimpleTagSupport class simplifies building such reusable features.

**Disadvantages and Limitations**

While useful, custom tags are not perfect. They require extra setup and an understanding of both Java and JSP’s tag API, which can intimidate beginners. For small applications, using JSTL or Expression Language (EL) might be simpler.

There’s also a minor **performance cost**. Each tag adds a bit of overhead because the container must create and run its class at runtime. Although this cost is small, it can add up on very busy sites.

Another issue is **maintenance**. If a tag library grows large without clear naming or documentation, new developers might find it confusing. Consistent structure and comments help avoid this problem.

**How to Create a Custom Tag**

Creating a custom tag involves four key steps:

1. **Write the Tag Handler Class.**  
   Extend SimpleTagSupport and place your logic in the doTag() method.
2. **Add Attributes (if needed).**  
   Define private variables with setter methods for each attribute you want to support.
3. **Create the TLD file.**  
   Specify the tag’s name, class, and attributes in XML.
4. **Register the Tag Library.**  
   Add a directive such as <%@ taglib uri="/WEB-INF/mytags.tld" prefix="demo" %> at the top of your JSP page.

These steps allow the container to locate and execute the tag correctly.

**My Perspective on Custom Tags**

Personally, I think custom tags are one of the best ways to keep JSP pages organized. They make complex pages easier to read and maintain. In my own experience, I like using them for repetitive tasks such as formatting user names or displaying dynamic messages.

However, I wouldn’t use them for every project. For smaller or temporary applications, creating a full tag library can be more work than it’s worth. But for long-term or team-based projects, custom tags are a huge help. They enforce structure, reduce duplicated code, and keep the project professional-looking.

**Conclusion**

Custom tags make JSP development more modular, readable, and maintainable. They help separate layout from logic and make team collaboration easier. Although they take some time to learn and configure, the benefits in organization and clarity easily outweigh the initial effort. When used thoughtfully, custom tags are one of the most effective tools for building clean and scalable Java web applications.

**References**

Apache Software Foundation. (n.d.). *SimpleTagSupport class (Java EE Servlet JSP API documentation).* <https://tomcat.apache.org/tomcat-7.0-doc/api/javax/servlet/jsp/tagext/SimpleTagSupport.html>

Oracle. (n.d.). *Using custom tags.* In *The Java EE 5 Tutorial.* <https://docs.oracle.com/javaee/5/tutorial/doc/bnaiy.html>

**Sample Code**

**Tag Handler Class**

public class HelloTag extends SimpleTagSupport {

private String name;

public void setName(String name) {

this.name = name;

}

@Override

public void doTag() throws JspException, IOException {

JspWriter out = getJspContext().getOut();

out.println("Hello, " + (name != null ? name : "Guest") + "!");

}

}

**Tag Library Descriptor (TLD)**

<tag>

<name>hello</name>

<tag-class>com.example.HelloTag</tag-class>

<body-content>empty</body-content>

<attribute>

<name>name</name>

<required>false</required>

</attribute>

</tag>

**JSP Usage**

<%@ taglib uri="/WEB-INF/mytags.tld" prefix="demo" %>

<demo:hello name="Julio"/>