Jacob Achenbach

10/4/2025

Csd 430

When using JSP and creating custom tags they can provide a powerful way to push repetitive or a lot of logic out of JSP pages and into more reusable Java components. Using custom tags let JSP pages stay cleaner, with mostly markups, rather than clustered.The ability to put behaviors into tags which in return can enhance maintainability and separate any problems, though it also introduces overhead and some greater complexity. Overall though there are many advantages and disadvantages of custom tags, throughout my paper I will explain what is required to properly build them, and give my perspective on when they should or shouldn't be used.

One advantage I will point out is that custom tags help maintain a clear separation between HTML and Java logic. With custom tags, the JSP page stays clear and concise and calling tags instead of embedding scriptlets. This improves readability, especially for the front end developers, which would probably prefer not to read Java code embedded in JSP. As GeeksforGeeks explains, custom tags like “separate the business logic from JSP” and reduce the need for scriptlets which is a mix between Java and HTML (GeeksforGeeks). Also the Java EE tutorial describes custom tags as user defined JSP elements that are translated into handler operations, thus separated logic in Java classes rather than letting it crowd and over complicate pages (Oracle “Using Custom Tags”).

Another benefit is the reuse of code. Once you build a tag library with a Tag Library Descriptor and handler classes, you can easily use those tags in many JSP pages or even across multiple applications. This avoids repeated code and maintains fluid code logic changes. The Oracle documentation states that to use custom tags the tag library must be made available to the web application and then declared in the JSP via a <%@ taglib %> directive (Oracle “Using Custom Tags”). With that said, tags can accept attributes which can map to setter methods in the tag handler or process body content, this gives them flexibility in behavior (Oracle “Understanding and Creating Custom JSP Tags”).

I will say custom tags are not perfect and definitely do come with some in other words flaws/problems. First is that they require up front overhead. So you must write the tag handler class, produce a proper TLD file defining tag names, attributes, and behavior, include a tag lib directive in JSPs, and deploy everything correctly. So for logic or one-time use, this overhead may outweigh any benefits. Second is debugging. When it becomes the debugging part and when logic is hidden in handler classes, the developer may need to jump among JSPs, Java classes, and TLD files. So mistakes or mismatches in names or class names often cause runtime errors. Third is performance cost is a concern. Wich each tag implies a method call, possible object instantiation or reuse, and handler logic execution. So in high traffic systems or when many tags are used per page, this overhead can pile up. Fourth is that custom tags may

hinder designers. If HTML is generated in Java code rather than markup, many people that do not code may find it difficult to adjust the output. Lastly, there is the danger of over-abstraction. So when creating tags that are generic or all-purpose that they become hard to understand or maintain. If a tag tries to do everything it loses clarity and becomes brittle.

Now to get started making correct custom tags you need a certain custom component. So the first step is that you need a tag handler class. In modern JSP, this is often done by extending SimpleTagSupport and overriding doTag(), or by implementing the classic Tag or BodyTag interfaces. The handler must provide a setter method for each attribute like public void so that the container can inject the attribute values before executing the tag logic in other words (GeeksforGeeks). One crucial part is that the handler must also reset internal state often using release() or even by reinitializing fields to avoid leaking state across multiple uses. Next thing to do is make a Tag Library Descriptor (TLD) XML file which maps tag names to handler classes which allows attributes with names, rtexprvalue, and then sets body-content policies to empty (Oracle “Understanding and Creating Custom JSP Tags”). The TLD must stay where the container can locate it commonly under WEB-INF/. The third step is that the JSP pages must declare the tag library that should look like “<%@ taglib prefix="ex" uri="/WEB-INF/custom.tld" %>”, which then tells the container prefix to use and where to find the tag definitions (Oracle “Using Custom Tags”). Finally, the correct packaging is very crucial as well. The tag handler classes must be in WEB-INF/classes or in JARs under WEB-INF/lib, the TLD should be packaged correctly, and classes must all agree on names. So the best practices include keeping tag logic lightweight and avoiding heavy database or network calls inside tags, validating attribute values, and handling exceptions properly by throwing JspException or IOException.

Here’s a small snippet:

```
public class HelloTag extends SimpleTagSupport {
   private String name;
   public void setName(String name) {
      this.name = name;
   }
   @Override
   public void doTag() throws JspException, IOException {
      if (name != null) {
         getJspContext().getOut().print("Hello, " + name + "!");
      } else {
         getJspContext().getOut().print("Hello, world!");
      }
   }
}
```

Then here is a piece of a TLD:

```
<taglib version="2.1" xmlns="http://java.sun.com/xml/ns/javaee"
```

```
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="…">
  <tag>
    <name>hello</name>
    <tag-class>com.example.HelloTag</tag-class>
    <body-content>empty</body-content>
    <attribute>
      <name>name</name>
      <required>false</required>
      <rtexprvalue>true</rtexprvalue>
    </attribute>
  </tag>
</taglib>
```

Works Cited

GeeksforGeeks. “Custom Tags in JSP.” *GeeksforGeeks*, 5 months ago, https://www.geeksforgeeks.org/java/custom-tags-in-jsp/. Accessed 5 Oct. 2025.

Oracle. “Using Custom Tags.” *The Java EE 5 Tutorial*, Oracle, https://docs.oracle.com/javaee/5/tutorial/doc/bnaiy.html.

Oracle. “Understanding and Creating Custom JSP Tags.” *Oracle Help Center*, https://docs.oracle.com/cd/E11035_01/wls100/taglib/quickstart.html.

TutorialsPoint. “JSP Custom Tags.” *TutorialsPoint*, https://www.tutorialspoint.com/jsp/jsp_custom_tags.htm.