Adrian Marquez

Module 10.2

When I first started working with JavaServer Pages (JSP), I didn’t think much about how the code was structured. I just wanted the page to work. But as the number of features on a site grows, things get complicated. It’s easy to end up with chunks of Java code stuck right in the middle of your HTML. After a while, it becomes hard to read, and even harder to update. After learning about custom tags, it’s changing the way I look at JSP development.

Custom tags in JSP basically let you create your own HTML-like tags to handle repetitive Java logic. It’s like giving yourself a shortcut for code you keep using over and over. Let’s say you’re formatting dates in five different JSPs. Instead of writing the same logic five times, you can build a tag like <mytags:formatDate /> and reuse it. That way, if you need to change the formatting later, you just change the tag's code once and you’re done (Oracle, 2025). It makes your life easier and your code cleaner.

What really stands out to me was is much cleaner JSP files look using custom tags. Scriptlets break up the flow of HTML and make it hard to see what the page is trying to do. But when you use custom tags, your code reads more like a layout or a template, which is easier for everyone to understand, including people who don’t write Java every day (GeeksforGeeks, 2023). It’s also easier to spot bugs when your logic isn’t buried in a sea of angle brackets and percent signs.

Beyond just organization, custom tags help with something called separation of concerns. That basically means the Java code (logic) and the HTML (design) should be in different places. Custom tags help make that happen. You write the logic once, in Java, and then call it in your JSP. This makes testing and updating much less painful. Plus, once you’ve built a good tag library, you can use it across multiple projects or share it with your team (Baeldung, 2016).

There are also some drawbacks to custom tags. Setting them up is kind of a hassle the first time. You need a Java class, an XML file (called a TLD), and a special directive in the JSP to make the tag work. If you're just getting started with JSP, this can feel like a lot just to replace one line of code. And sometimes, it *is* too much. If you only need something once, using JSTL or EL (Expression Language) is faster and simpler (GeeksforGeeks, 2023).

There’s also the fact that custom tags are less popular now in modern Java development. A lot of new projects are built with Spring Boot and Thymeleaf, which have better support for components. And front-end frameworks like React or Vue handle most of the page structure now, leaving JSP behind in many cases (Baeldung, 2016). So, if you’re working on something new, you may not even need to bother with JSP custom tags.

So, how exactly do you make one of these custom tags? The setup isn’t complicated once you’ve done it a few times, but it can feel like a lot at first. Basically, you need four pieces:

1. The Java class that does the work. This class usually extends SimpleTagSupport and contains the logic inside the doTag() method.
2. A Tag Library Descriptor (TLD) file, which is just an XML file where you define what the tag is called, which Java class it uses, and what attributes it accepts.
3. The taglib directive in the JSP page that tells the page where to find your custom tag library. This usually looks like:  
   <%@ taglib prefix="mytags" uri="/WEB-INF/mytags.tld" %>
4. And finally, packaging everything so your JSP page can use it—putting the class and the TLD file into the right folders (usually inside WEB-INF) and making sure they’re compiled and ready to go (Oracle, 2025).

Once those pieces are in place, you can call your custom tag like any built-in one, and the JSP engine will know what to do.

While custom tags can be very useful, they’re not always the most practical choice. They tend to be most valuable when the same logic needs to be applied in multiple places within an application. For example, tasks like formatting currency consistently across several pages or enforcing user permissions throughout a site are ideal candidates for encapsulation in a custom tag. This approach allows developers to make updates in a single location, which then automatically apply everywhere the tag is used, reducing maintenance effort and potential errors (Oracle, 2025). Also, custom tags help improve page readability, making it easier for multiple developers to understand and maintain the codebase.

Custom tags are especially helpful in larger teams or projects with shared codebases. When individual developers create their own versions of formatting or security checks, the application can become inconsistent and harder to maintain. Using a common set of custom tags promotes uniformity in functionality and design, helping keep the application’s behavior consistent across different modules (Baeldung, 2016).

On the other hand, custom tags may be unnecessary or useless for simpler tasks or one-off logic that only appears once. In these situations, using standard tag libraries like JSTL or simple Expression Language (EL) expressions is usually easier. Building a custom tag for minimal or infrequently used functionality can introduce unnecessary complexity and overhead.

Additionally, in modern development environments, the use of custom tags is less common. Many contemporary Java applications rely on frameworks such as Spring Boot combined with Thymeleaf, which provide more streamlined methods for creating reusable UI components. Front-end development often takes place in JavaScript frameworks like React, Vue, or Angular, where JSP and custom tags are typically not involved. In these contexts, investing time in creating custom JSP tags may not be worthwhile (GeeksforGeeks, 2022).

To sum it all up, custom tags in JSP can be really helpful, especially in legacy Java projects or bigger apps that need to stay organized and easy to maintain. They make your pages cleaner, promote code reuse, and let you keep logic separate from layout, which is always a win in most developer’s books. But they also have their drawbacks. They come with some setup work, and in modern Java web development, there are often better tools for the job.

So, I wouldn’t say custom tags are outdated exactly, but I’d definitely say they’re situational. If you’re working in an environment that still uses JSP, it’s a good idea to learn how to build and use them. Just be smart about when you do. If it simplifies your code and saves time in the long run, use them. But if it’s just extra overhead for a one-off task, keep it simple.

References:

*Java Platform, Enterprise Edition: The Java EE Tutorial Release 7 - Contents*. (2025). Oracle.com. <https://docs.oracle.com/javaee/7/tutorial/>

GeeksforGeeks. (2022, December 28). *Custom Tags in JSP*. GeeksforGeeks. <https://www.geeksforgeeks.org/java/custom-tags-in-jsp/>

baeldung. (2016, December 1). *Guide to JavaServer Pages (JSP) | Baeldung*. www.baeldung.com. <https://www.baeldung.com/jsp>

**Code Examples Below:**

**Java Handler Class**A computer screen shot of a computer code

AI-generated content may be incorrect.**Tag Library Descriptor**A computer screen with white text

AI-generated content may be incorrect.

**JSP Page**

A computer screen with white text

AI-generated content may be incorrect.