Jacob Cannamela

CSD 430

7/19/2025

**Module 10 assignment**

Like with any other technology, JavaServer Pages (JSP) has its own share of advantages and shortcomings especially when it comes to curating a balance between best practices and ease of maintenance. Java logic tends to clutter the presentation layer of an application as it becomes increasingly complex and difficult to manage. This problem is alleviated through the use of custom tags which serve as encapsulated reusable components that integrate Java functionality, enhancing the separation between the logic and the layout. Just like HTML or JSTL tags, these custom tags are completely tailored to developers' requirements and as such, provide optimal modularity which is the goal of web development.

The ability to reuse code more effectively is perhaps one of the greatest benefits of custom tags. Instead of repeating the same Java code snippet in several JSP files, developers only need to write the logic once and encapsulate it within a custom tag, which can then be referenced through a simple declarative statement. This not only eliminates duplicitous coding but also guarantees uniform application behavior and styling. Take, for instance, custom tags which fetch user profile data or format dates; they can be reused on numerous pages, which significantly streamlines and speeds up the modification process. With this strategy, programmers can concentrate on maintenance because there is less logic that is repeated and can be handled in isolation.

Custom tags aid in achieving a significant separation of concerns in JSP development. Developers are able to focus on the visual structure of the JSP pages since the logic can now be handled by a tag handler class. This improves the readability and modifiability of the pages, especially for frontend developers that do not know Java. Instead of looking at a tangled mess of Java and HTML, they will encounter well defined, self-contained tags with phrases such as <app:formatDate> or <user:showProfile> which enhances understanding of the code. Following this design pattern helps in adhering to the MVC architecture, often leading to improved development.

Nevertheless, custom tags come with some drawbacks. They add extra overhead to the development processes for a smaller project where such levels of complexity might be unnecessary. In order to implement a custom tag, one needs to write a Java class, create a Tag Library Descriptor (TLD) file, and make sure that the JSP page properly uses and includes the tag library. For projects which only require a few dynamic elements, this approach is unnecessarily complicated. For less experienced developers, unlocking the JSP tag lifecycle puzzle, properly using methods like doStartTag, doEndTag, or doTag, and following through with the business logic can be daunting and, if not done precisely, can result in unexpected behavior. It is more complex to debug the logic of custom tags than standard JSP code because the logic is more deeply buried within backend components.

The performance aspect is yet another possible negative. Whenever a custom tag is employed in a document, the app has to create an instance and run the corresponding tag handler class. If these tags are executed multiple times within loops, performance can be undermined even more. Developers must be cautious not to overuse tags for functions best served by JSTL or Expression Languages.

From a Technical perspective, a Java class must be implemented either by extending SimpleTagSupport or implementing the Tag interface in order to create custom tags. It seems, most developers implement attributes through setter methods and use a core method like doTag() or doStartTag() when the logic tied to the tag is executed. The association between tag names and their handler classes is done through TLD . Tags are defined in the file and as long as the tag library is imported into the JSP page alongside configured TLD, it should be good to go. This part may give the developer some trouble depending on the file tree structure (speaking from experience).

As much as I advocate for the implementation of custom tags in their appropriate scenarios, I also feel that developers need to be mindful of overengineering problems. For small-scale applications, or those created with more recent frameworks such as Spring Boot or modern frontend technologies, the custom tag use may not be warranted. JSTL and EL already provide sufficient abstraction and maintainability in these situations, and adding more frameworks would only add layers of unnecessary complexity. However, in classic JSP settings where backend logic is still predominantly in charge of rendering the views, custom tags present a neat, extensible method for managing complexity and enhancing code clarity.

To summarize, custom tags within Java JSP offer an organized and efficient method for developing complex web components that are reusable and maintainable. Despite the additional initial overhead and performance impact, the gains that they offer with respect to organization and separation of concerns is hard to ignore. In custom tags are indispensable when it comes to maintaining large scale applications or constructing a uniform presentation layer, as they greatly enhance the sustainablity of web applications over an extended period of time. As with all implementation strategies, these will need to be aligned with the requirements of the web project and skill level of the developers working on the project.

**References:**

Pandey, S. (2025, January 6). Advanced Java — Custom tags in JSP - Sachin Pandey - Medium. *Medium*. <https://medium.com/@SachinPandeyOnline/advanced-java-custom-tags-in-jsp-da3f324e8ee0>

*Creating a custom tag | JSP Tutorial | StudyTonight*. (n.d.). https://www.studytonight.com/jsp/creating-a-custom-tag.php

*A JSP custom tag for XML+XSLT*. (n.d.). Object Computing, Inc. https://objectcomputing.com/resources/publications/sett/september-2000-a-jsp-custom-tag-for-xmlxslt

RK Keynotes. (2023, October 2). *#12 How to Create Own Tags in JSP? | Custom Tags | Tag Handler Class | TLD File | doTag() | Taglib* [Video]. YouTube. https://www.youtube.com/watch?v=tox6mq8e8KA