Noel Miranda

May 3, 2025

Server Side Development

Module 10

Understanding JSF Tag Libraries: HTML, Core, and Facelets

JavaServer Faces (JSF) is a framework that started to make more sense the deeper I ventured into it. At first glance, it might seem like just another way to build Java based web applications, but compared to the other approaches learned by far, JSF actually brings structure and reusability to the table. It uses components to help build user interfaces and manage the flow of a web application in a way that feels organized and scalable (GeeksforGeeks, 2019). This paper specifically explores how JSF works, with a special focus on three key tag libraries: HTML, Core, and Facelets. To further clarify the framework, this paper also makes references to a personal supplemental dynamic web project that utilizes JSF to simulate an online student survey. In essence, JSF relies on these libraries to provide the functionality that makes it such a powerful web framework.

To begin with, JSF is used to create Java based web applications that run on a server and use reusable user interface components (Manelli & Zambon, 2020). To be specific, it provides a structured way to link the front end with server-side logic, allowing developers to build dynamic, data-driven pages efficiently (GeeksforGeeks, 2019). To build these interfaces, developers rely on tags from the mentioned libraries, which define what appears on the page. One can think of these tag libraries as toolkits filled with ready to use building blocks for web pages, making the development process both powerful and intuitive. What makes these building blocks special is that they hide much of the complexity involved in handling tasks such as validation, state

management, and back-end integration, allowing developers to focus more on what the application does rather than how it works under the hood (GeeksforGeeks, 2019).

When it comes to the HTML tag library in JSF, it gives developers smarter tools that look and feel like standard HTML elements but come with extra capabilities. For example, using <h:form> to create a form or <h:inputText> to let users enter data is as easy as using regular HTML. But these tags do more than just render the user interface, they connect directly to server-side Java code through managed beans (Tutorialspoint, n.d.). It is like placing an order at a restaurant kiosk that sends our selection straight to the kitchen without requiring additional instructions. When it comes to my supplemental web dynamic project, I utilized the HTML tag library to build a feedback form for a mock Bellevue University course survey. The form fields include student ID, course name, rating selection, and a text area for open response feedback. Tags such as <h:inputText>, <h:selectOneMenu>, <h:inputTextarea> and <h:commandButton> were central to the development process. I also included <h:message> to show validation errors. As mentioned before, these tags did not just render the user interface elements but also linked them to the corresponding properties in its managed bean.

In view of the Core tag library, it plays more of a behind the scenes role. It does not handle how things look, but rather how they work. I like to think of it as a set of helpers that make sure the input values are correct and processed properly on the server side. For instance, I used <f:validateRegex> to ensure the student ID followed a specific pattern, and <f:validateLongRange> to restrict the numeric inputs to valid ranges. That level of control gave me confidence that the data would be validated before being passed to the application's business logic or database layer, reducing potential errors or misuse (Pareek, 2015). On top of that, one cannot forget the amazing benefit of not having to write complex logic manually in the form of scriptlets or JavaScript. However, one is able to make their own validators if needed for more

complex validation or special use cases (Tutorialspoint, n.d.). This overall makes the application more secure, reliable, and easier to maintain.

The Facelets tag library is where layout and structure come into play. Facelets enable developers to create templates, in other words blueprints, primarily using the <ui:insert> tag for web pages that define shared elements like headers, footers, and sidebars (Jakarta EE, 2025). For the supplemental project, I personally made my template file hold a common page structure and then applied it to my other individual pages using <ui:composition> and <ui:define>. This allowed me to change only the content while keeping the overall layout consistent across multiple pages. As a result, this templating system allowed me to create a polished interface without repeating code. For example, I defined the logo and stylesheet once in the template and reused them throughout the application. Then, I inserted dynamic content, like the survey form and thank you message, into designated spots using the appropriate Facelets tags. This made my code cleaner and gave me the flexibility to expand it more easily later on, in the case I would like to add a results page or additional survey sections.

At first, the multitude of JSF tags was overwhelming. But once one sees how each tag serves a purpose, whether to control layout, validate inputs, or connect to the backend, the learning curve flattens. One analogy that stuck with me was thinking of JSF like building a Lego structure. Each tag is like a Lego piece designed for a specific job. Once one understands what the pieces do, it becomes easier to build something strong and reusable. JSF also reduces the need for JavaScript and avoids mixing Java and HTML in the same file, as seen in JSP with scriptlets. All in all, JSF offers a more organized, modular approach to development. This separation of behavior from layout is a major advantage for both security and long-term maintenance (Manelli & Zambon, 2020).

In conclusion, learning to use JSF and its three major tag libraries gave me a deeper appreciation for how web applications can be both robust and well organized. Additionally, all three tag libraries support the Model-View-Controller (MVC) pattern. HTML tags define the view layer, Core tags handle control logic like validation, and Facelets manage the layout and structure of the view (Manelli & Zambon, 2020). All in all, these libraries do more than help build interfaces, they promote better coding practices, encourage reusability, and make it easier for teams to collaborate.

References

GeeksforGeeks. (2019, January 2). JSF | Java Server Faces. GeeksforGeeks. https://www.geeksforgeeks.org/jsf-java-server-faces/

Jakarta EE. (2025). Introduction to Facelets. Jakarta.ee. https://jakarta.ee/learn/docs/jakartaee-tutorial/9.1/web/faces-facelets/faces-facelets.html

Manelli, L., & Giulio Zambon. (2020). *Beginning Jakarta EE web development : using JSP, JSF, MySQL, and Apache Tomcat for building Java web applications*. Apress.

Pareek, M. (2015). Jsf Tutorial. Javasafari.com. https://javasafari.com/jsf/jsf-core-tag-library.php

SF core tag library. (Apache MyFaces JSF-1.1 Implementation Tag library documentation). (n.d.). Svn.apache.org. https://svn.apache.org/repos/asf/myfaces/site/publish/core11/myfaces-impl/tlddoc/f/tld-summary.html

Tutorialspoint. (n.d.). JSF - Basic Tags - Tutorialspoint. Www.tutorialspoint.com. https://www.tutorialspoint.com/jsf/jsf_basic_tags.htm

Tutorialspoint. (n.d.). JSF Validation Tags. Tutorialspoint.com. https://www.tutorialspoint.com/jsf/jsf_validation_tags.htm