Java Server Faces o Jakarta Server Faces (JSF) es un framework web muy interesante para crear aplicaciones web con Java, muy popular hasta hace algunos años, surgía como una alternativa ideal para desarrolladores backend Java con poco o nulo conocimiento de librerías clientes como JavaScript.

En el transcurrir de los años a pesar de que sigue siendo una excelente alternativa, tiene un ciclo de vida complejo y que lastimosamente muchos no entienden lo que hace que se tengan malas prácticas y por ende aplicaciones lentas y mal optimizadas.

Algunos de sus más defensores sostienen que es una alternativa incluso mejor que los populares frameworks de JavaScript como NextJS, Nuxt o Angular Universal y el argumento principal es que permite hacer todo lo que estos frameworks ya hacen, pero con la diferencia de que JSF lo lleva haciendo desde hace muchos años.

Específicamente la controversia es con el Server Side Rendering y es que aunque JSF se ejecuta en el servidor y genera HTML dinámico que se envía al cliente, no es exactamente lo mismo que el Server-Side Rendering (SSR) moderno tal como se entiende en el desarrollo web actual y no, no son solamente tecnisismos, son conceptos diferentes.

¿Qué es JSF?

JavaServer Faces (JSF) es un framework para construir interfaces de usuario basadas en componentes para aplicaciones web Java. JSF proporciona un conjunto de componentes reutilizables de interfaz de usuario, gestión del estado del servidor y validación de entrada de usuario. Además cuenta con excelentes frameworks como es el caso de PrimeFaces.

Características de JSF

1. Basado en componentes

JSF utiliza un enfoque basado en componentes para construir la interfaz de usuario. Los componentes son objetos Java que representan elementos de la interfaz de usuario como botones, campos de texto y tablas, generalmente en JSF no utilizamos los tags tradicionales para su maquetación si no tags personalizados que pueden añadir atributos adicional al HTML que se renderiza posteriormente.

2. Gestión del Estado del Servidor

JSF mantiene el estado de la vista del lado del servidor, lo que significa que cuando se

requiera modificar cualquier componente de la aplicación de forma dinámica, debemos realizar una petición al servidor, ya que el estado de la aplicación se almacena ahí.

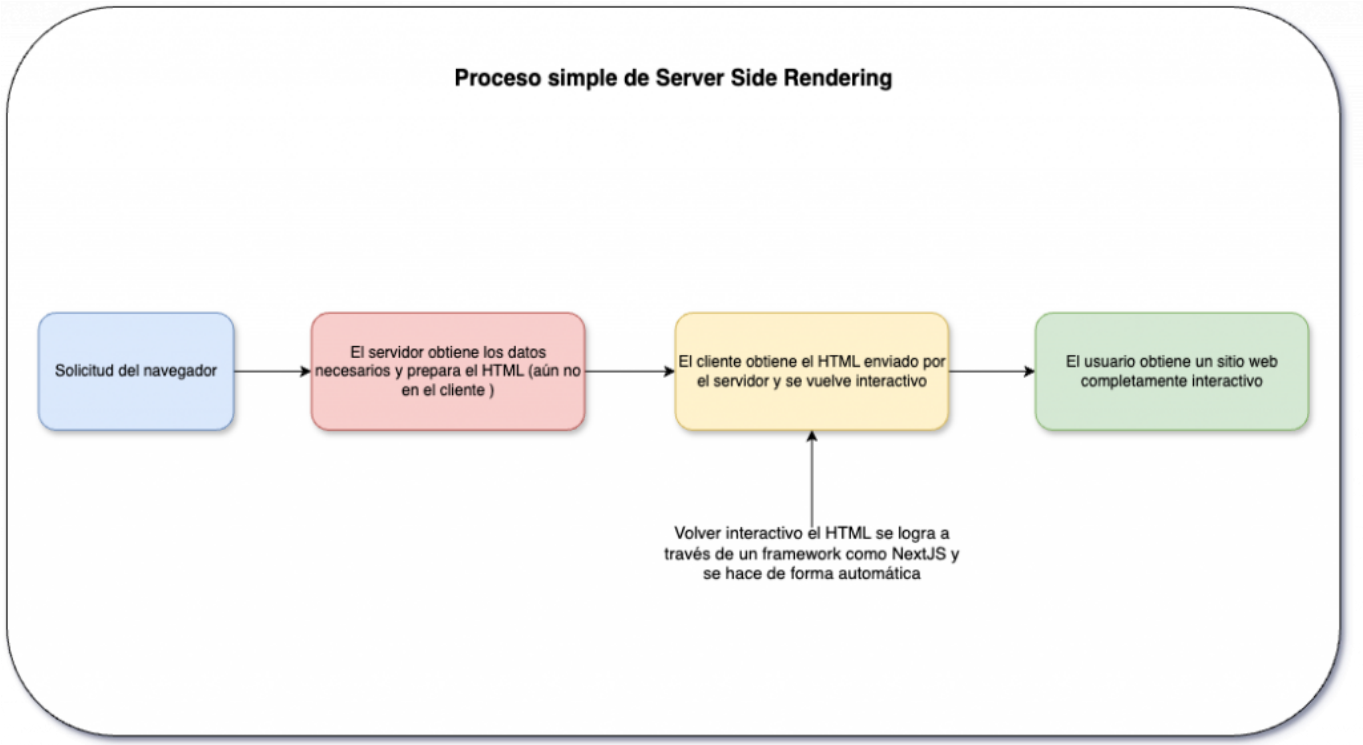
3. Integración con AJAX

JSF soporta AJAX para permitir la actualización parcial de la página sin recargar toda la página. Esto mejora la interactividad y la experiencia del usuario.

¿Qué es el Server-Side Rendering (SSR) Moderno?

El SSR moderno se refiere a una técnica en la que las aplicaciones web se renderizan inicialmente en el servidor y luego se "hidratan" en el cliente para convertirse en aplicaciones de una sola página (SPA) completamente interactivas. Este enfoque es popular en frameworks modernos como Next.js (para React), Nuxt.js (para Vue.js) y Angular Universal (para Angular).

Proceso de SSR



1. Renderizado en el Servidor

La aplicación genera HTML completo en el servidor y lo envía al cliente, en este momento el HTML no es interactivo. Es decir si hemos añadido algún evento o dinamismo a través de JavaScript no va a tener ningún efecto aún.

2. Envío al Navegador

El navegador recibe el HTML y hace su trabajo, es decir lo renderiza de forma inmediata pero continua sin ser interactivo aunque en esta etapa se descargan los JS necesarios para lograr volver interactivo ese JS.

3. Hidratación

En esta fase es cuando un framework como NextJS, Nuxt o Angular Universal, se encarga de tomar ese HTML y hacerlo completamente interactivo, esto permite que la velocidad sea mucho mayor y además podemos controlar el estado, tanto a nivel del cliente como a nivel del servidor.

Diferencias Clave entre JSF y SSR

1. Renderizado Inicial

- **JSF**: El servidor renderiza HTML dinámico usando componentes JSF y lo envía al cliente. Este HTML es estático y se actualiza con AJAX según sea necesario. También podemos agregar código JavaScript personalizado pero suele ser una tarea un tanto compleja porque debemos interactuar con componentes que ya tienen un comportamiento definido y podemos tener algunos problemas de compatibilidad.
- **SSR** : El servidor genera HTML completo inicialmente y luego el JavaScript del cliente “hidrata” el HTML para convertirlo en una web completamente interactiva.

2. Gestión del Estado:

- **JSF**: Mantiene el estado de la vista en el servidor. Cada interacción del usuario puede resultar en una solicitud al servidor para actualizar el estado, simplemente necesitas abrir la pestaña network del navegador para darte cuenta la cantidad de peticiones que se hacen al servidor en procesos sencillos.
- **SSR**: El estado inicial se puede establecer en el servidor, pero una vez que la página

está “hidratada”, el estado se gestiona principalmente en el cliente.

3. Interactividad:

- **JSF**: Utiliza AJAX para actualizar partes de la página sin recargarla, pero la lógica de interactividad es menos fluida que en una web moderna. No significa que AJAX no sea una buena alternativa, lo es. Pero se queda corta comparada con las soluciones modernas.
- **SSR**: Después de la hidratación, la aplicación se comporta como una web completa, proporcionando una interactividad fluida y rápida. Y esto es gracias a que el estado se mantiene principalmente del lado del cliente.

4. SEO y Rendimiento:

- **JSF**: Proporciona buenas capacidades de SEO ya que el HTML estático se genera en el servidor, pero puede tener problemas de rendimiento debido a la gestión del estado del servidor y al desconocimiento del ciclo de vida de un ManagedBean.
- **SSR**: Ofrece excelentes capacidades de SEO y rendimiento debido al renderizado inicial en el servidor y la rápida interactividad proporcionada por la SPA.

Limitaciones de JSF

1. Complejidad en la Gestión del Estado

Mantener el estado de la vista en el servidor puede ser complejo y aumentar la carga del servidor. Además esto en un mundo donde casi todo está en la nube de terceros (AWS, GCP, Azure), puede significar en un aumento significativo de costos.

2. Interactividad Limitada

Aunque JSF soporta AJAX, no puede igualar la fluidez y la rapidez de las SPA modernas, es que es incomparable, no tiene caso seguir discutiendo en si AJAX es una competencia a las SPA modernas, como mucho es una alternativa, pero no es comparable.

3. Menos Flexibilidad:

JSF puede ser menos flexible y más difícil de escalar que las aplicaciones construidas con frameworks modernos de JavaScript.

Conclusión

Aunque JSF genera HTML en el servidor y puede actualizar parcialmente la página con AJAX, no es lo mismo que el Server-Side Rendering (SSR) moderno. El SSR moderno, utilizado en frameworks como Next.js, Nuxt.js y Angular Universal, combina lo mejor del renderizado del servidor y la interactividad de las SPA, ofreciendo una experiencia de usuario superior en términos de rendimiento y fluidez.

JSF sigue siendo una herramienta poderosa para aplicaciones empresariales basadas en Java, pero no se clasifica como SSR en el sentido moderno. No deberías dudar en elegir el SSR Moderno para tu nuevo proyecto teniendo en cuenta las ventajas de uno y otro.