Seguimiento

Consultas API

Sebastian Ordoñez Giraldo

Universidad Alexander Von Humboldt

Ingeniería de software

Armenia

2024

Beans:

JavaBeans es una clase Java que sigue ciertas convenciones para que pueda ser utilizada como un componente reutilizable en diversas aplicaciones. Los JavaBeans se utilizan comúnmente en entornos de desarrollo visual, como aplicaciones web o de escrito

Características de los JavaBeans:

- **Encapsulamiento:** Los atributos (propiedades) están encapsulados y se accede a ellos mediante métodos *getter* y *setter*.
- **Constructor sin parámetros:** Un JavaBean debe tener un constructor público sin parámetros para que las herramientas de desarrollo visual puedan instanciarlo fácilmente.
- **Serializable:** Los JavaBeans implementan la interfaz `Serializable`, lo que significa que pueden ser serializados (convertidos en una secuencia de bytes) y almacenados o enviados a través de la red.
- **Propiedades accesibles:** Las propiedades de un JavaBean pueden ser accedidas a través de métodos públicos denominados *getters* y *setters*, que siguen una convención de nomenclatura (`getPropiedad` o `setPropiedad`).

Estructura EAR:

Un archivo **EAR (Enterprise Archive)** es un archivo que encapsula y empaqueta aplicaciones empresariales que se ejecutan en servidores de aplicaciones compatibles con **Jakarta EE** (antes **Java EE**). Un archivo EAR permite agrupar y desplegar varios módulos relacionados, como **EJB**, **servlets**, **JSP**, **aplicaciones web**, y otros componentes necesarios para la ejecución de una aplicación empresarial.

El archivo EAR utiliza el formato de empaquetado **ZIP** y generalmente tiene la extensión `.ear`. Dentro del archivo EAR se encuentran otros archivos de tipo **JAR (Java Archive)** y **WAR (Web Archive)**, que corresponden a los diferentes módulos de la aplicación empresarial.

Características del archivo EAR:

- **Despliegue conjunto:** Agrupa todos los componentes de la aplicación empresarial para que puedan desplegarse juntos en un servidor de aplicaciones.
- **Modularidad:** Permite combinar diferentes módulos (Web, EJB, bibliotecas) en un solo paquete, facilitando la gestión de una aplicación compleja.
- **Soporte para múltiples aplicaciones:** Un archivo EAR puede contener más de un módulo Web o EJB, permitiendo gestionar múltiples partes de una aplicación desde un único archivo.
- **Configuración centralizada:** Al usar archivos de descriptor como `application.xml`, la configuración de la aplicación se maneja en un único lugar.

JSF:

JavaServer Faces (JSF) es un **framework** de desarrollo web basado en componentes que forma parte de la especificación **Jakarta EE (anteriormente Java EE)**. JSF está diseñado para simplificar la creación de interfaces de usuario (UI) para aplicaciones web basadas en Java, proporcionando una arquitectura bien estructurada que separa la lógica de negocio de la presentación.

Características

- **Basado en componentes**: Proporciona un conjunto reutilizable de componentes UI.
- **Gestión del ciclo de vida**: Define un ciclo de vida claro para las solicitudes web.
- **Soporte MVC**: Separa la lógica de negocio (Modelo) de la presentación (Vista).
- **Backing Beans**: Usa beans manejados para gestionar la lógica de negocio.
- **Validación y conversión de datos**: Valida datos de entrada automáticamente.
- **Navegación declarativa**: Gestiona la navegación entre páginas mediante configuración.
- **Extensible y personalizable**: Permite crear y extender componentes.
- **Integración con Ajax**: Facilita actualizaciones parciales sin recargar toda la página.
- **Compatibilidad con otras tecnologías Java EE**: Se integra fácilmente con tecnologías como EJB y JPA.

Soap Web Services

(Simple Object Access Protocol) es un protocolo estándar basado en XML para intercambiar mensajes entre aplicaciones a través de redes, como Internet. SOAP permite la comunicación entre aplicaciones que están en diferentes plataformas y lenguajes de programación, utilizando estándares bien definidos.

Características de SOAP Web Services:

- **Basado en XML**: Utiliza XML para formatear mensajes, siendo independiente de plataforma y lenguaje.
- **Protocolo estandarizado**: Sigue un estándar de la W3C, garantizando interoperabilidad.
- **Uso de HTTP y otros protocolos**: Generalmente se transmite por HTTP, pero puede usar otros como SMTP o FTP.
- **Mensajes estructurados**: Los mensajes incluyen un sobre con un encabezado opcional y un cuerpo obligatorio.
- **WSDL**: Utiliza WSDL para describir las operaciones y parámetros del servicio.
- **Seguridad**: Soporta seguridad avanzada con WS-Security.
- **Transacciones**: Permite transacciones distribuidas entre varios servicios.
- **Estado y estilo**: Soporta servicios con o sin estado.
- **Extensible**: Permite agregar funcionalidades a través de encabezados personalizados.

WebSocket

WebSocket es un protocolo de comunicación que permite una conexión bidireccional y persistente entre un cliente y un servidor a través de un único canal de comunicación, generalmente utilizando el protocolo **TCP**. Fue diseñado para superar las limitaciones de HTTP, que es de naturaleza unidireccional (solo el cliente puede iniciar solicitudes), proporcionando una comunicación en tiempo real y eficiente entre las aplicaciones.

Características de WebSocket:

Comunicación bidireccional: Permite que tanto cliente como servidor envíen y reciban mensajes en tiempo real.

Conexión persistente: La conexión permanece abierta durante toda la sesión, reduciendo la sobrecarga.

Bajo overhead: Después del handshake inicial, los mensajes se envían eficientemente sin encabezados innecesarios.

Tiempo real: Ideal para aplicaciones que requieren actualizaciones instantáneas como chats o juegos.

Basado en frames: Los datos se envían en forma de cuadros (frames), que pueden ser texto o binarios.

RESTful Web Services

RESTful Web Services (Representational State Transfer) son servicios web que siguen los principios de la arquitectura REST, que es un estilo arquitectónico para construir aplicaciones distribuidas. REST se basa en el uso de métodos HTTP estándar para realizar operaciones sobre recursos, y es muy popular debido a su simplicidad y escalabilidad.

Características de RESTful Web Services:

Uso de HTTP: Utiliza métodos HTTP estándar como GET, POST, PUT y DELETE.

Recursos identificados por URIs: Los recursos son representados y accedidos mediante URIs únicas.

Operaciones CRUD: Los métodos HTTP permiten crear, leer, actualizar y eliminar recursos.

Stateless: No se mantiene estado en el servidor; cada solicitud es independiente.

Intercambio de datos en JSON o XML: Los datos se intercambian en formatos estándar, siendo JSON el más común.

Cacheable: Las respuestas pueden ser almacenadas en caché para mejorar la eficiencia.

Interfaz uniforme: Se siguen convenciones consistentes para la interacción con recursos.

Separación cliente-servidor: Separa completamente el frontend del backend.

Escalable y flexible: Facilita la escalabilidad y modificación sin afectar al cliente.

Bajo overhead: Usa formatos ligeros y operaciones eficientes.