

John Sebastian Castaneda Hoyos

Programación II

Conceptos de APIs y ejercicio de ejemplo

Docente

Hernan Henao

Universidad Alexander Von Humboldt

Armenia, Quindío
2024

¿Qué es un BEANS?

En Java, un **Bean** es una clase que sigue ciertas convenciones o reglas específicas para que pueda ser utilizada en distintos contextos, como en el framework de JavaBeans, aplicaciones Java EE , o en entornos de desarrollo como frameworks de inyección de dependencias.

Características de un Bean en Java:

1. Constructor público sin argumentos:

Los Beans deben tener un constructor público sin argumentos (constructor por defecto). Esto permite que se puedan crear instancias de los Beans de forma fácil, tanto por frameworks como por herramientas de desarrollo.

2. Propiedades accesibles mediante métodos getters y setters:

Las propiedades de un Bean son privadas y se acceden mediante métodos públicos.

3. Serializable:

Para que un Bean pueda ser persistido o enviado a través de la red, debe implementar la interfaz `java.io.Serializable`. Esto es importante para que el Bean pueda ser almacenado y transferido entre diferentes capas de una aplicación.

4. Encapsulamiento:

Las propiedades de los Beans son privadas, lo que significa que no se accede directamente a ellas desde fuera de la clase. Los **getters** y **setters** permiten controlar el acceso y modificación de las propiedades.

5. Sin comportamiento complejo:

Un Java Bean es típicamente una clase simple que contiene solo propiedades, métodos de acceso (getters y setters) y, en algunos casos, lógica simple.

¿Qué es la estructura EAR?

En Java, un archivo **EAR (Enterprise Archive)** es un archivo comprimido que contiene todos los módulos necesarios para una aplicación empresarial de Java EE. Estas aplicaciones suelen estar compuestas por varios tipos de módulos, como archivos **JAR** (Java ARchive) y **WAR** (Web ARchive), que se empaquetan juntos dentro del archivo EAR. El archivo EAR facilita la distribución y despliegue de aplicaciones empresariales complejas en servidores de aplicaciones Java EE.

Características de la estructura EAR

META-INF/:

Este directorio contiene los archivos de configuración de la aplicación EAR, principalmente el archivo `application.xml`, que define los módulos que forman parte de la aplicación y otras configuraciones necesarias para el despliegue.

Módulos WAR:

Los módulos **WAR** son archivos que contienen la parte web de la aplicación (Java Servlets, JSPs, archivos HTML, etc.). Estos módulos se colocan directamente en el archivo EAR. Cada módulo WAR tiene su propia estructura interna con su directorio `WEB-INF/`, donde se colocan los archivos de configuración y las clases específicas de la web.

Módulos JAR:

Los módulos **JAR** contienen la lógica de negocio, como los Enterprise JavaBeans (EJBs) y otros componentes de backend. Estos JARs también se colocan directamente en el archivo EAR.

¿Qué es JSF?

JavaServer Faces (JSF) es un **framework** de desarrollo web basado en componentes para la plataforma Java EE. JSF facilita la creación de interfaces de usuario (UI) en aplicaciones web al proporcionar una arquitectura basada en componentes reutilizables, y está integrado con otras tecnologías Java EE como **EJB** y **JPA**.

Características principales de JSF:

1. **Framework basado en componentes:**

JSF permite el desarrollo de páginas web mediante componentes reutilizables (botones, tablas, formularios, etc.). Estos componentes pueden ser configurados y renderizados de diferentes maneras, permitiendo que sean reutilizados en varias partes de la aplicación.

2. **Separación de lógica de presentación y lógica de negocio:**

Con JSF, la capa de presentación está bien separada de la lógica de negocio. Las páginas web son diseñadas utilizando archivos **XHTML** que definen la estructura de la UI, mientras que la lógica de negocio se maneja en **managed beans**.

3. **Sistema de navegación:**

JSF tiene un sistema de navegación que define cómo la aplicación debe navegar entre diferentes páginas. Esto se puede definir explícitamente en el archivo `faces-config.xml` o mediante anotaciones en las clases.

4. **Validación y conversión de datos integrados:**
JSF permite la validación automática de formularios y la conversión de tipos de datos sin necesidad de escribir mucho código. Los errores de validación se gestionan fácilmente y se muestran de forma clara al usuario.
5. **Soporte para controladores de eventos:**
JSF permite manejar eventos del lado del servidor como los clics de botones o cambios en los campos del formulario. Estos eventos están vinculados a métodos en los managed beans.
6. **Integración con otros frameworks:**
JSF puede integrarse fácilmente con otros frameworks y tecnologías, como **JPA** para persistencia de datos o **EJB** para la lógica de negocio, permitiendo la construcción de aplicaciones empresariales completas.

¿Qué es Soap web services?

SOAP (Simple Object Access Protocol) es un protocolo estándar basado en XML que se utiliza para intercambiar mensajes estructurados entre aplicaciones a través de la red. En el contexto de Java, **SOAP Web Services** son servicios web que siguen el protocolo SOAP para comunicarse, y son comúnmente implementados en aplicaciones empresariales Java EE

Características de SOAP Web Services en Java:

1. **Protocolos basados en XML:**
SOAP usa **XML** para estructurar los mensajes, lo que permite que los datos sean independientes de la plataforma y el lenguaje de programación. Esto facilita la interoperabilidad entre diferentes sistemas.
2. **Estándar bien definido:**
SOAP es un protocolo estricto con un formato bien definido para los mensajes, lo que garantiza la consistencia en la comunicación. A diferencia de otros enfoques como REST, SOAP sigue un formato específico que incluye un **envelope** con un **header** y un **body**.
3. **Transporte independiente:**
Aunque SOAP se utiliza comúnmente sobre **HTTP** (Hypertext Transfer Protocol), también puede trabajar sobre otros protocolos de transporte como **SMTP** o **JMS**.
4. **WSDL (Web Services Description Language):**
Los SOAP Web Services suelen estar descritos por un archivo **WSDL**, que es un documento XML que describe los servicios web, incluyendo los métodos disponibles, los tipos de datos que manejan y cómo deben ser llamados.

¿Qué es WebSocket en Java?

WebSocket en Java es una tecnología que permite la **comunicación bidireccional** entre un cliente (generalmente un navegador) y un servidor. A diferencia de los protocolos tradicionales como HTTP, en los que el cliente envía una solicitud y el servidor responde, con WebSockets, **el servidor y el cliente pueden intercambiar mensajes de forma continua** y en tiempo real. Esto es ideal para aplicaciones que requieren actualizaciones en tiempo real, como chats, notificaciones en vivo, videojuegos multijugador.

Características principales de WebSocket en Java:

1. Comunicación bidireccional:

A diferencia de HTTP, donde la comunicación es solicitada solo por el cliente, WebSocket permite que tanto el cliente como el servidor envíen datos en cualquier momento.

2. Persistencia de la conexión:

En lugar de abrir y cerrar conexiones continuamente como lo hace HTTP, WebSocket establece una conexión persistente entre el cliente y el servidor, reduciendo el overhead de las conexiones repetidas.

3. Bajo consumo de recursos:

Debido a que WebSocket mantiene una conexión abierta, reduce la carga del servidor y la latencia, ya que no hay necesidad de reabrir la conexión para cada mensaje que se envía.

4. Comunicación en tiempo real:

WebSocket permite enviar y recibir mensajes en tiempo real sin necesidad de recargar la página o esperar respuestas prolongadas, lo que es crucial para aplicaciones que necesitan alta interacción en vivo.

¿Qué es RESTful web services?

RESTful Web Services en Java son una forma de implementar servicios web utilizando el paradigma **REST (Representational State Transfer)**. REST es un estilo arquitectónico que utiliza el protocolo **HTTP** y se basa en el uso de recursos, representados por URLs, para permitir la comunicación entre un cliente y un servidor.

Características de RESTful Web Services:

1. Basado en HTTP:

Los servicios RESTful aprovechan los métodos estándar de HTTP, como **GET**, **POST**, **PUT**, **DELETE**, etc., para realizar operaciones sobre recursos.

2. Uso de recursos:

En REST, todo se trata como un **recurso**. Cada recurso tiene una representación única en el sistema, accesible a través de una **URI** (Uniform Resource Identifier).

3. Stateless (Sin estado):

Cada solicitud HTTP desde un cliente a un servidor es independiente, lo que significa que el servidor no almacena información sobre el estado del cliente entre solicitudes. Esto hace que REST sea más escalable.

4. Mensajes en formato estándar:

Los mensajes intercambiados entre el cliente y el servidor pueden ser en varios formatos, pero los más comunes son **JSON** y **XML**. JSON es más popular debido a su simplicidad y menor sobrecarga en comparación con XML.

5. Operaciones CRUD:

Los servicios RESTful se alinean naturalmente con las operaciones CRUD (Create, Read, Update, Delete) usando los métodos HTTP:

- **GET**: Obtener un recurso.
- **POST**: Crear un nuevo recurso.
- **PUT**: Actualizar un recurso existente.
- **DELETE**: Eliminar un recurso.

Repositorio ejemplo consumo de APIs

<https://github.com/VirtualViking/DemoAPI.git>