COSA E JAVA EE

E una piattaforma per lo sviluppo e l'esecuzione di applicazioni aziendali, web e distribuite aggiunge API e runtime per gestire le esigenze di applicazioni di livello enterprise, come transazioni, sicurezza, scalabilità, messaggistica e persistenza.

COME E STRUTTURATO

Java EE è organizzato in più livelli, ognuno dei quali fornisce funzionalità specifiche per supportare lo sviluppo di applicazioni enterprise, i componenti chiave sono
//
Servlets: Per gestire le richieste HTTP e costruire applicazioni web.
JSP: Per creare pagine HTML(web) dinamiche.
EJB: Per separare la logica di business e gestire servizi distribuiti.
JPA: Per gestire l'accesso e la persistenza dei dati nei database.
JAX-RS e JAX-WS: Per sviluppare API RESTful e servizi web SOAP.
CDI: Per la gestione avanzata dei bean e l'iniezione di dipendenze.
//

LA SUA STRUTTURA E:
client---> web layer ----> buisness layer -----> persistence layer

Client: Può essere un'applicazione browser, desktop o mobile, che invia richieste HTTP/HTTPS al server.
Web Layer: È il punto di ingresso del server, gestisce richieste HTTP (Servlet, JSF, REST API) e spesso include funzionalità di autenticazione, routing e trasformazione dati.
Business Layer: Contiene la logica applicativa vera e propria. In Java EE, spesso è implementato tramite Enterprise JavaBeans (EJB) o CDI (Contexts and Dependency Injection).
Persistence Layer: Si occupa della gestione dei dati, generalmente interfacciandosi con un database tramite JPA o altre tecnologie ORM.

Connessione al Database

Ecco i modi principali per collegarsi a un database in Java EE, dal più vecchio al più moderno:

a. JDBC (Java Database Connectivity)

• **Cos'è:** API di basso livello per connettersi a un database usando query SQL dirette.

• **Come si usa:**

1. Caricare il driver JDBC.
2. Aprire una connessione (DriverManager.getConnection()).
3. Creare uno Statement o PreparedStatement per eseguire query.
4. Gestire i risultati (ResultSet).
5. Chiudere risorse manualmente.

• **Pro:** Molto semplice.

• **Contro:** Richiede molto codice boilerplate e non è scalabile per progetti grandi.

b. JPA (Java Persistence API)

• **Cos'è:** API che permette la gestione degli oggetti (modello orientato agli oggetti) e li mappa direttamente alle tabelle del database (ORM - Object Relational Mapping).

• **Implementazioni comuni:** Hibernate, EclipseLink.

• **Come si usa:**

1. Definire entità con annotazioni come @Entity e @Table.
2. Utilizzare un EntityManager per gestire operazioni CRUD.

• **Pro:** Scrittura di query più semplice e meno codice boilerplate.
• **Contro:** Curva di apprendimento iniziale.

d. Spring Data JPA (Estensione di Spring Framework)

• **Cos'è:** Framework che semplifica ulteriormente JPA, eliminando quasi tutto il codice boilerplate.
• **Come si usa:**

1. Creare repository usando interfacce come JpaRepository.
2. Usare metodi predefiniti o definire query con nomi di metodo (findByNome).

• **Pro:** Super intuitivo e veloce da configurare.
• **Contro:** Legato al framework Spring.

3. Connessione al Browser

Dal più vecchio al più moderno:

a. Servlets

• **Cos'è:** API per creare applicazioni web ricevendo e rispondendo a richieste HTTP.
• **Come si usa:**

1. Creare una classe che estende HttpServlet.
2. Implementare i metodi doGet e doPost.
3. Configurare il servlet in web.xml o con annotazioni (@WebServlet).

• **Pro:** Semplice per iniziare.
• **Contro:** Richiede molto codice e non è pratico per progetti grandi.

b. JSP (JavaServer Pages)

• **Cos'è:** Tecnologia che combina HTML e Java (scriptlet).
• **Come si usa:** Scrivere HTML con tag speciali <% %> per integrare codice Java.
• **Pro:** Semplice per creare pagine dinamiche.
• **Contro:** Mescolare logica di business e presentazione non è consigliato.

c. JSF (JavaServer Faces)

• **Cos'è:** Framework component-based per costruire interfacce utente web.
• **Come si usa:** Utilizzare file XML o XHTML con tag JSF per definire interfaccia e logica.
• **Pro:** Componente standard di Java EE.
• **Contro:** Non più molto popolare.

d. Framework moderni

1. **Spring MVC**
   - Permette di creare controller (@Controller) per gestire richieste HTTP.
   - Supporta template engine moderni come Thymeleaf o integrazione con frontend JS.
3. **RESTful API con JAX-RS**
   - **Cos'è:** Specifica per creare servizi RESTful.
   - **Come si usa:** Annotare metodi con @GET, @POST, @Path per rispondere alle richieste.
   - **Pro:** Facilmente combinabile con frontend moderni (Angular, React).

COSA E UN API

è un insieme di regole, protocolli e strumenti che consente a un'applicazione software di comunicare con un'altra. In altre parole, un'API è un'interfaccia che permette a diversi programmi di interagire tra loro, scambiando dati o utilizzando funzionalità senza dover conoscere i dettagli dell'implementazione.

COSA E UN FRAMEWORK

E una struttura software riutilizzabile progettata per fornire supporto per lo sviluppo di applicazioni
E un insieme di strumenti librerie e regole che stabiliscono come costruire e organizzare un applicazione in modo efficiente.
Quindi il framework:
controlla il flusso di un applicazione e fornisce una struttura completa per sviluppare un applicazione.
e sono:

Hibernate:
MyBatis

COSA E UNA SERVLET JSP, EJB, JPA, CDI

SERVLET: Un servlet e una classe java utilizzata per estendere le funzionalita di un server web, consentendo di gestire richieste e risposte in modo dinamico.
I servlet sono una parte fondamentale della piattaforma jakarta ee e javax e vengono utilizzati per creare applicazioni web che rispondono a richieste e risposte in modo dinamico.

COME FUNZIONA UNA SERVLET

Ricezione della richiesta (Request):

Una servlet riceve una richiesta HTTP (ad esempio, GET o POST) da parte di un client (spesso un browser web).
Elaborazione della richiesta:

La servlet analizza la richiesta usando l'oggetto HttpServletRequest per accedere ai dati della richiesta (parametri, intestazioni, ecc.).
Può eseguire operazioni di elaborazione come interrogare un database, calcolare risultati, o elaborare la logica di business.
Generazione della risposta (Response):

La servlet crea una risposta usando l'oggetto HttpServletResponse.
inoltra la richista alla jsp usando RequestDispathcer generando una risposta html
Questa può essere:
Una pagina HTML o contenuto dinamico generato sul momento.
Dati in formato JSON o XML per applicazioni web o API REST.
Un file scaricabile come un PDF o un'immagine.
inoltra la richista alla jsp usando RequestDispathcer generando una risposta html
Invio della risposta al client:

La servlet restituisce la risposta al server web, che a sua volta la invia al client.

COSA E UNA JSP

JSP è un’estensione delle servlet che consente di separare la logica di presentazione (HTML, CSS) dalla logica di business (Java). Ogni JSP, una volta richiesta, viene convertita in una servlet dal server, compilata e poi eseguita. Questo rende le JSP molto simili alle servlet ma più comode per progettare l’interfaccia utente.
quindi:
crea interfacce utente per applicazioni web dinamiche.
integra com servlet e framework mvc

COSA E UNA EJB

progettata per sviluppare componenti server-side modulari, scalabili, e riutilizzabili in applicazioni aziendali distribuite, e utilizzato per creare componenti buisness logici che devono essere eseguiti in un contenitore ejb,(ad esempio come su un server come wildfly, glassfish, o weblogic).
Serve per semplificare lo sviluppo di applicazioni aziendali grazie a funzionalita preconfigurate fornite dal contenitore.

L'EJB viene spesso usato insieme a quello che e la session bean
Session Bean:
Stateful: Mantiene lo stato della sessione tra client e server (es. un carrello della spesa).
Stateless: Non mantiene lo stato tra le chiamate; ogni richiesta è indipendente (es. un servizio di calcolo).
Singleton: Unico per l'intera applicazione, utile per operazioni condivise (es. cache o gestione delle configurazioni).
Ti permette di creare un Local e un Remote, che ti permettono di gestire le operazioni sul lato server sia in locale quindi sulla stessa macchina e in remote ovvero in macchine diverse.

Local:e un interfaccia utilizzata quando il client e l'ejb si trovano nella stessa JVM usa il @Local non usa la serializzazione ed e piu veloce rispetto al remote

Remote: e un interfaccia usata quando il client e lejb si trovano su macchine diverse, usa il @Remote. il remote usa la serializzazione per inviare oggetti attraverso la rete.

COSA E JPA

fornisce un insieme di API per la gestione della persistenza dei dati nelle applicazioni Java.
Consente agli sviluppatori di mappare le classi Java alle tabelle di un database relazionale, facilitando le operazioni di archiviazione, recupero e aggiornamento dei dati
JPA si basa sul concetto di Object-Relational Mapping (ORM), che permette di rappresentare le entità del database come oggetti Java. Questo approccio semplifica l'interazione con il database, riducendo la necessità di scrivere manualmente query SQL e migliorando la portabilità del codice.
ha bisogno di un framework come hibernate.
In sintesi, JPA standardizza l'accesso ai dati nelle applicazioni Java, fornendo un modello coerente e indipendente dal database sottostante, semplificando lo sviluppo e la manutenzione del codice