

JAVA + Spring

Introduzione a JavaEE + Spring

Presented by





Gianluca Avizzano/Donata Uricchio

IBM Client Innovation Center - Bari

Napoli, 12/02/2022

IBM Client Innovation Center
Italy

Agenda

-  1 Introduzione a JavaEE
-  2 Componenti JavaEE
-  3 Container
-  4 Spring

Java SE

```
public class Test {  
    static int x;  
    public static void main (String [] args) {  
        System.out.println(x);  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) *Exception at runtime*
- b) *Compilation Error*
- c) *0*
- d) *1*

Java SE

```
public class Test {  
    int x = 3;  
    public static void main (String [] args) {  
        Test test = new Test();  
        System.out.println(test.x);  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) *Exception at runtime*
- b) *Compilation Error*
- c) *0*
- d) *3*

Java SE

```
public static void main(String[] args) {  
  
    String somethingToSay;  
    int money = 10;  
    if(money < 5) {  
        somethingToSay = "Coffe please";  
    }else if(money >= 5) {  
        somethingToSay = "Pizza please";  
    }  
    System.out.println(somethingToSay);  
}
```

What will be the result of compiling and executing the main method?

- a) *Coffee please*
- b) *Pizza please*
- c) *Exception at runtime*
- d) *Coffee pleasePizza Please*
- e) *Compilation Error*

Java SE

```
public class Test {  
    int x = 3;  
  
    public static void main (String [] args) {  
        System.out.println(x);  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) *Exception at runtime*
- b) *Compilation Error*
- c) *0*
- d) *3*

Java SE

```
public class Test {  
    public static void main(String[] args) {  
        String language = "JAVA";  
        switch (language) {  
            default:  
                System.out.println("Whatever works");  
            case "C":  
                System.out.println("I am the father");  
            case "JAVA":  
                System.out.println("I'm good but verbose");  
            case "python":  
                System.out.println("I'm cute and fast");  
                break;  
        }  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) Exception at runtime
- b) Compilation error
- c) Whatever works
 I'm good but verbose
- d) *I'm good but verbose*
- e) *I'm good but verbose*
 I'm cute and fast

Java SE

```
public static void main (String [] args) {  
    int a = 1;  
    int b = 2;  
    int c = 3;  
  
    int num = a > b ? 1 : a > c ? 2 : 3;  
    System.out.println(num);  
}
```

What will be the result of compiling and executing Test class?

1. 1
2. 2
3. 3
4. Compilation error
5. None of the above

Java SE

```
public class Test {  
    public static void main (String [] args) {  
        int x;  
        System.out.println(x);  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) *Exception at runtime*
- b) *Compilation Error*
- c) *0*
- d) *1*

Java SE

```
public class Test {  
    public static void main (String [] args) {  
        String message = "Hello ";  
        concat(message);  
        System.out.println(message);  
    }  
  
    public static void concat(String s) {  
        s += "world!";  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) Exception at runtime*
- b) Compilation error*
- c) Hello*
- d) Hello world!*

Java SE

```
public class Test {  
    public static void main(String[] args) {  
        String language = "java";  
        switch (language) {  
            default:  
                System.out.println("Whatever works");  
            case "C":  
                System.out.println("I am the father");  
            case "JAVA":  
                System.out.println("I'm good but verbose");  
            case "python":  
                System.out.println("I'm cute and fast");  
                break;  
        }  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) Exception at runtime
- b) Compilation error
- c) Whatever works
 I'm good but verbose
- d) Whatever works
 I am the father
 I'm good but verbose
 I'm cute and fast
- e) *I'm good but verbose*
 I'm cute and fast

Java SE

```
public static void main (String [] args) {  
    List<String> elements = new ArrayList<>();  
    elements.add("j");  
    elements.add("a");  
    elements.add("v");  
    elements.add("a");  
  
    elements.remove("a");  
    System.out.print(elements);  
}
```

What will be the result of compiling and executing Test class?

- a) *Exception at runtime*
- b) *Compilation error*
- c) *[j,a,v,a]*
- d) *[j,v,a]*
- e) *[j,v]*

Java SE

```
public class Test {  
    public static void main (String [] args) {  
        int b = 1;  
        int a = 10;  
        if(a > 10)  
            b = 10;  
        else  
            b++;  
            b--;  
        System.out.println(b);  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) *Exception at runtime*
- b) *Compilation error*
- c) *1*
- d) *2*

Java SE

```
public class Test {  
    public static void main (String [] args) {  
        String message = "Hello ";  
        message.concat("world");  
        System.out.println(message);  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) *Exception at runtime*
- b) *Compilation error*
- c) *Hello*
- d) *Hello world!*

Java SE

```
public class Test {  
    int x = 3;  
    static int y=0;  
    public static void main (String [] args) {  
        Test test1 = new Test();  
        Test test2 = new Test();  
        test1.x = 4;  
        test1.y++;  
        test2.x = 5;  
        test2.y++;  
        System.out.println(test1.x + " " + test1.y);  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) Exception at runtime
- b) Compilation error
- c) 4 1
- d) 5 1
- e) 4 2
- f) 5 2

Java SE

```
public class Test {  
    public static void main (String [] args) {  
        for(int i = 0; i <3; i++) {  
            System.out.print(i);  
        }  
        System.out.print(i);  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) Exception at runtime*
- b) Compilation error*
- c) 012*
- d) 0123*

Java SE

```
public class Test {  
    public static void main (String [] args) {  
        for(int i = 0; i <3; i++) {  
            if(i % 2 == 0) {  
                System.out.print(i);  
            }  
        }  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) *Exception at runtime*
- b) *Compilation error*
- c) *012*
- d) *02*
- e) *0123*
- f) *13*
- g) *024*
- h) *135*

Java SE

```
public class Test {  
    public static void main (String [] args) {  
        for(int i = 0; i <3; i++) {  
            if(i % 2 == 0) {  
                System.out.print(i);  
                break;  
            }  
        }  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) *Exception at runtime*
- b) *Compilation error*
- c) *No output*
- d) *1*
- e) *0*
- f) *2*

Java SE

```
public class Test {  
    public static void main (String [] args) {  
        int condition = 0;  
        String message = "Hello world!";  
        if(condition) {  
            System.out.println(message);  
        } else {  
            System.out.println("Default");  
        }  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) *Exception at runtime*
- b) *Compilation error*
- c) *No output*
- d) *Hello world!*
- e) *Default*

Java SE

```
public class Test {  
    public static void main (String [] args) {  
        String s1 = "HELLO";  
        String s2 = "HELLO";  
        String s3 = new String("HELLO");  
  
        System.out.println((s1 == s2) + " " + (s2 == s3) + " " + (s2.equals(s3)));  
    }  
}
```

What will be the result of compiling and executing Test class?

- a) Exception at runtime*
- b) Compilation error*
- c) true true true*
- d) true false true*
- e) true false false*

Java SE VS JAVA EE

- Java SE (Standard Edition)

È una piattaforma di programmazione Java. Include API di programmazione Java come java.lang, java.io, java.net, java.util, java.sql, java.math ecc. Include argomenti di base come OOPs, String, Regex, Exception, Inner classes, Multithreading, Stream I/O, Networking, AWT, Swing, Reflection, Collection, ecc.

- Java EE (Enterprise Edition)

È una piattaforma aziendale utilizzata principalmente per sviluppare applicazioni Web e aziendali. È costruito sulla piattaforma Java SE. Include argomenti come Servlet, JSP, Web Services, EJB, JPA, ecc.

JAVA EE

Acronimo di Java Enterprise edition

J2EE è un ambiente basato su Java, indipendente dalla piattaforma, applicabile per lo sviluppo, la creazione e la distribuzione online di applicazioni aziendali basate sul Web. J2EE include vari set di servizi, API e protocolli per lo sviluppo di applicazioni basate sul Web multilivello.

Applicazioni a tre livelli :

- Client
- Server
- Database o macchine legacy

Set di API per

- Abbreviare i tempi di sviluppo.
- Riducendo la complessità delle applicazioni.
- Migliorando le prestazioni delle applicazioni.

Architettura multi-tier

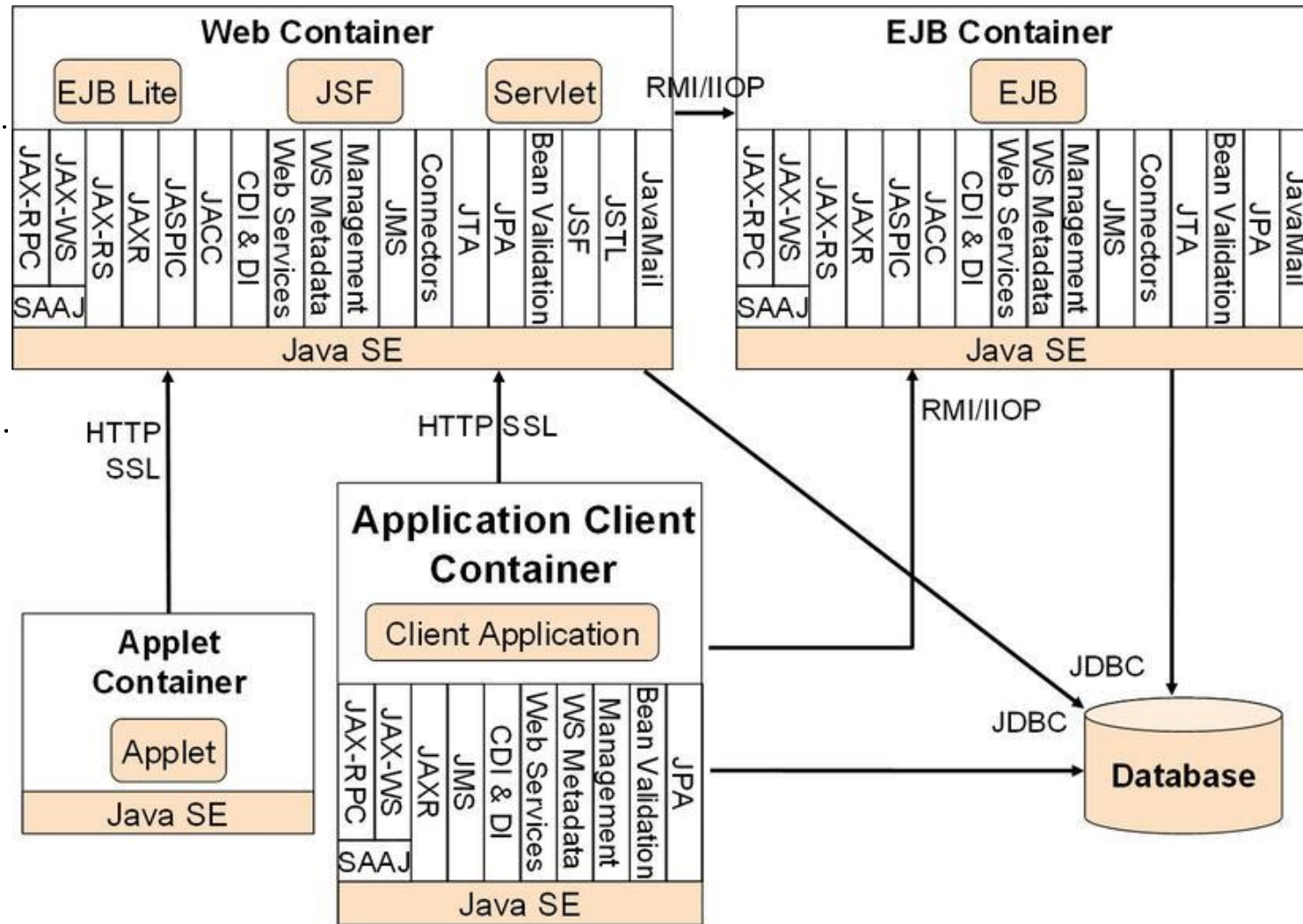
Versioni di Java EE:

- J2EE 1.2 (12 dicembre 1999)
- J2EE 1.3 (24 settembre 2001)
- J2EE 1.4 (11 novembre 2003)
- Java EE 5 (11 maggio 2006)
- Java EE 6 (10 dicembre 2009)
- Java EE 7 (5 aprile 2013)
- Java EE 8 (2017)

Architettura Client Server

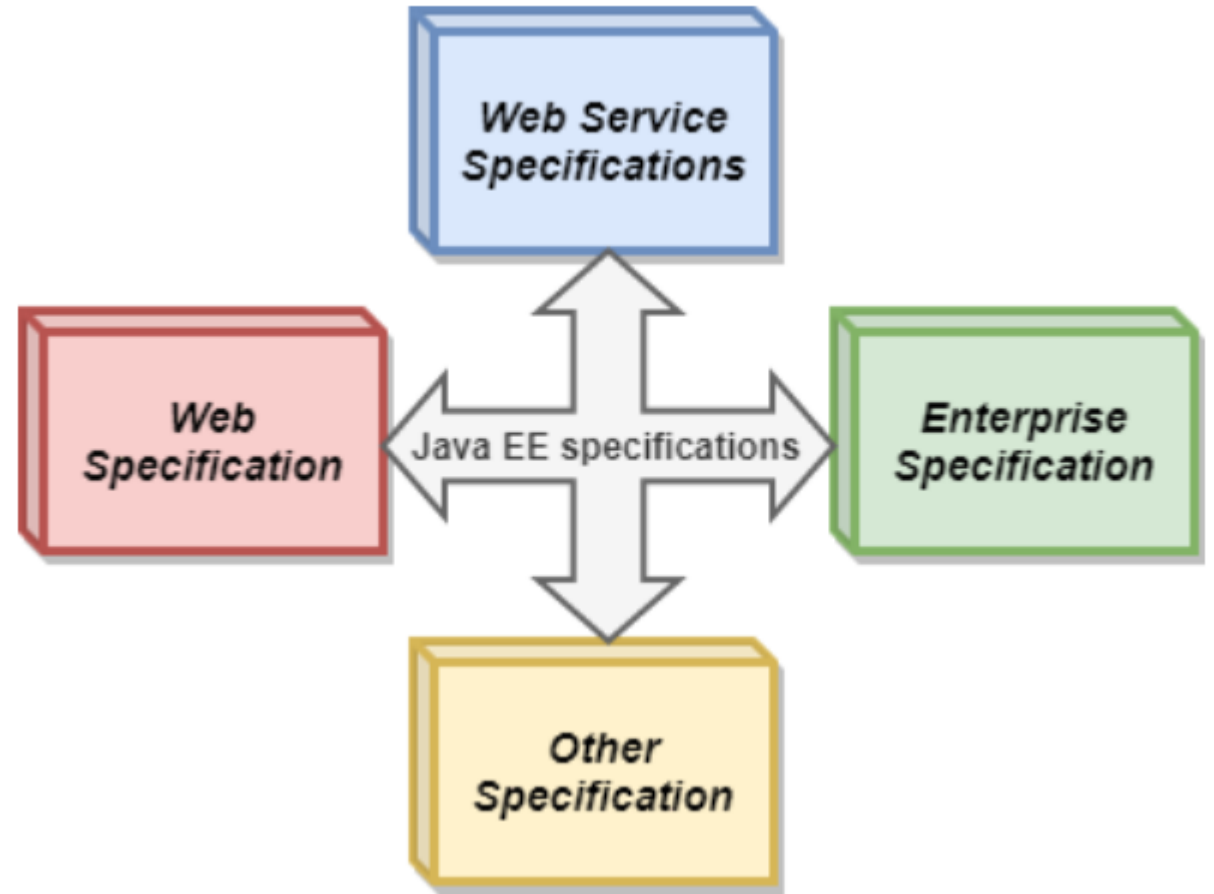


Specifiche J2EE



Specifiche J2EE

Java EE ha diverse specifiche utili per creare pagine web, leggere e scrivere da database in modo transazionale, gestire code distribuite. Java EE contiene diverse API che hanno le funzionalità delle API Java SE di base come Enterprise JavaBeans, connettori, servlet, pagine server Java e diverse tecnologie di servizi Web.



Web Specifications

- Servlet: questa specifica definisce come gestire le richieste HTTP in modo sincrono o asincrono. È di basso livello e altre specifiche dipendono da esso
- WebSocket: WebSocket è un protocollo di comunicazione per computer e questa API fornisce un insieme di API per facilitare le connessioni WebSocket.
- Java Server Faces: è un servizio che aiuta a creare la GUI dai componenti.
- Unified Expression Language: è un linguaggio semplice progettato per facilitare gli sviluppatori di applicazioni web.

Web Service Specifications

- **API Java per servizi Web RESTful:** aiuta a fornire servizi con schema di trasferimento dello stato rappresentativo.
- **API Java per l'elaborazione JSON-** È un insieme di specifiche per gestire le informazioni fornite in formato JSON.
- **API Java per JSON Binding-** Si tratta di un insieme di specifiche per l'associazione o l'analisi di un file JSON in classi Java.
- **Java Architecture for XML Binding-** Consente il binding di xml in oggetti Java.
- **API Java per servizi Web XML-SOAP** è un protocollo basato su xml per accedere ai servizi Web su http. Questa API consente di creare servizi Web SOAP

Enterprise Specifications of Java EE

- Contexts and Dependency Injection: fornisce un contenitore per inserire le dipendenze come in Swing.
- Enterprise JavaBean: è un insieme di API leggere che un contenitore di oggetti possiede per fornire transazioni, chiamate di procedure remote e controllo della concorrenza.
- Java Persistence API- Queste sono le specifiche della mappatura relazionale degli oggetti tra le tabelle di database relazionali e le classi Java.
- Java Transaction API- Contiene le interfacce e le annotazioni per stabilire l'interazione tra il supporto delle transazioni offerto da Java EE. Anche le API in questo estratto dai dettagli di basso livello e le interfacce sono considerate di basso livello.
- Java Message Service: fornisce un modo comune al programma Java per creare, inviare e leggere i messaggi del sistema di messaggistica aziendale.

Un componente Java EE è un'unità software funzionale autonoma

Componenti

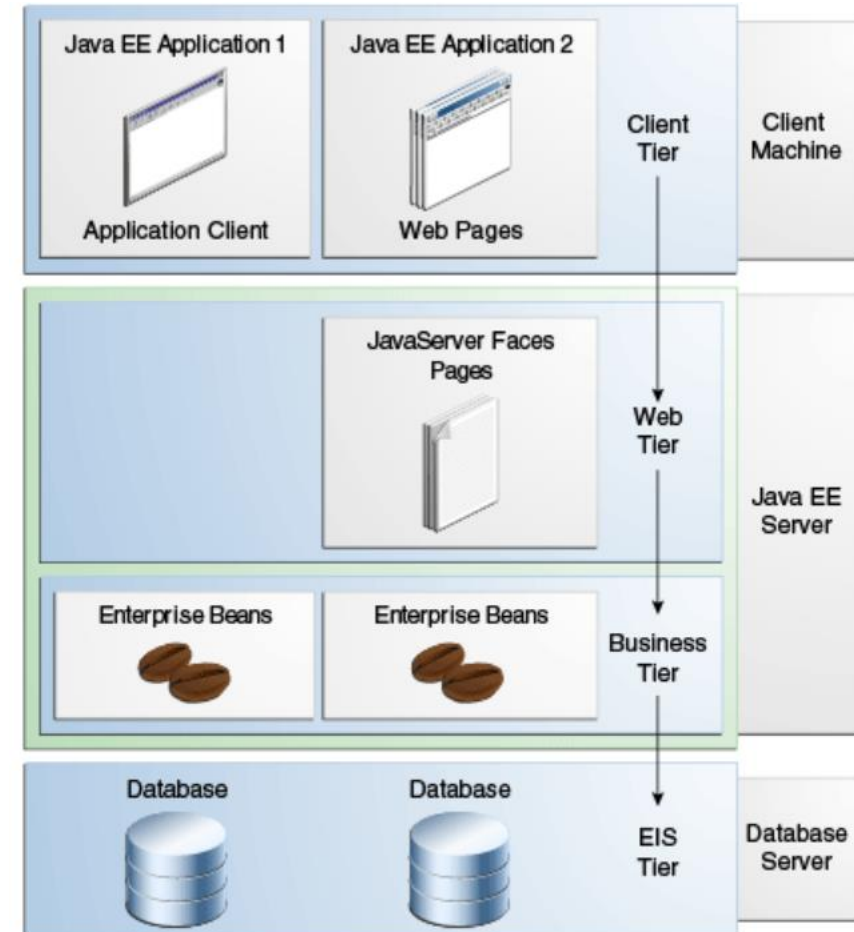
Client tier:

- Web Client
- Application Client

Business tier:

- Entity Beans
- Session Beans
- Message Driven Beans.

Software EIS (Enterprise Information System) viene eseguito sul server EIS.



Web Tier

Funzioni principali:

- Generazione dinamica dei contenuti per diversi client
- Raccolta dei dati di input che gli utenti inviano tramite interfaccia web client
- Generazione output sulla base delle componenti del business tier
- Controllo del flusso di navigazione sul client
- Mantenimento dello stato per una sessione utente
- Logica applicative di base e memorizzazione temporanea di informazione all'interno di componenti Java (Java Bean)

Tecnologia	Scopo
Servlets	Classi Java che processano le richieste HTTP e generano dinamicamente le risposte (HTML)
JavaServer Faces	Framework per il design dell'interfaccia utente di applicazioni Web
JavaServer Faces Facelets	Particolari applicazioni JavaServer Faces che usano pagine XHTML anziché JSP
Expression Language	Insieme di tags standard usati in JSP e Facelets per riferirsi a componenti Java EE
JavaServer Pages (JSP)	Documenti testuali compilati e trasformati in Servlets per aggiungere contenuto dinamico a pagine HTML
JavaServer Pages Standard Tag Library	Tag library che raccoglie funzionalità comuni a pagine JSP
JavaBeans Components	Oggetti Java per la memorizzazione temporanea dei contenuti di un'applicazione

Business tier

- Consiste di component che forniscono la logica di business dell'applicazione
- “Business logic” insieme di software che si occupa delle funzionalità di un determinato contesto di business
- Costituisce il “core” dell'intera applicazione

Tecnologia	Scopo
Enterprise JavaBeans (EJB)	Componenti gestite dall'Application Server che incapsulano le funzionalità principali dell'applicazione
JAX-RS RESTful Web Services	API per la creazione di Web Services REST (via HTTP GET e POST)
JAX-WS Web Service Endpoints	API per la creazione ed il consumo di Web Services XML/SOAP
Java Persistence API Entities	API per il mapping tra i dati contenuti nei sistemi di memorizzazione persistente e corrispondenti oggetti Java
Java EE Managed Beans	Essenzialmente EJB che non richiedono requisiti di sicurezza/transazionalità

Data tier

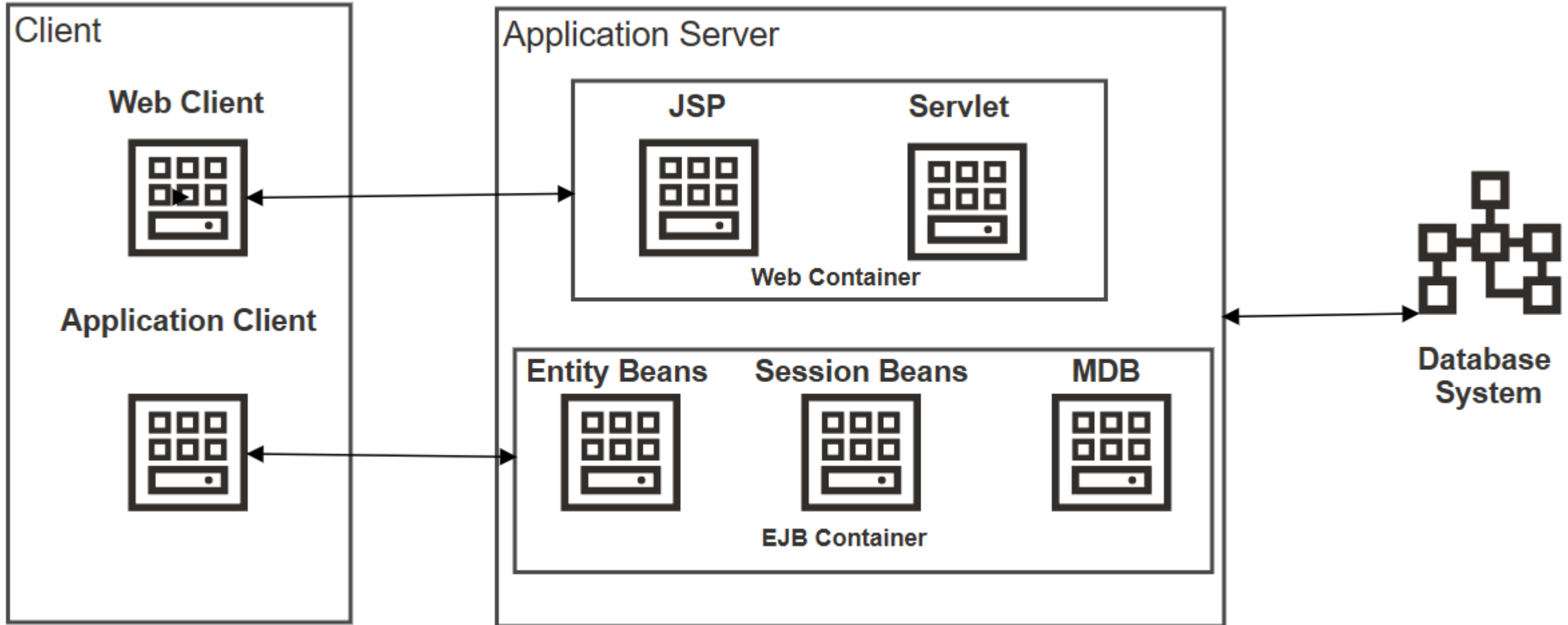
Si riferisce alle varie sorgenti di dati cui può attingere l'applicazione:

- Relational Database Management System
- Enterprise Resource Planning System (SAP)
- Mainframes(IBM AS/400)

Tecnologia	Scopo
Java Database Connectivity API (JDBC)	API a basso livello per l'accesso ed il recupero dei dati memorizzati su supporti permanenti. Tipicamente usata per eseguire query SQL ad un particolare RDBMS
Java Persistence API	API per la creazione di Web Services REST (via HTTP GET e POST)
Java EE Connector Architecture	API per la creazione ed il consumo di Web Services XML/SOAP
Java Transaction API (JTA)	API per la definizione e la gestione delle transazioni tra sorgenti dati multiple e distribuite

J2EE fornisce dei container ad hoc sia per i componenti web che per quelli di business

Container



WebContainer

E' un contenitore J2EE che ospita applicazioni Web. Il contenitore Web estende la funzionalità del server Web fornendo agli sviluppatori l'ambiente per eseguire servlet e JavaServer Pages (JSP), una servlet.

Interfaccia tra le componenti web e il server web, gestisce il ciclo di vita del componente

Application Client Container

Interfaccia tra le applicazioni client JAVA EE ed il server JAVA EE
In esecuzione su machine client differentidal server

Ejb Container

I bean Enterprise (componenti EJB) sono componenti del server del linguaggio di programmazione Java che contengono la logica aziendale. Il contenitore EJB fornisce l'accesso locale e remoto ai bean enterprise. Esistono tre tipi di bean enterprise: bean di sessione, bean di entità e bean a messaggi. I bean di sessione rappresentano oggetti e processi transitori e in genere vengono utilizzati da un singolo client. I bean di entità rappresentano dati persistenti, generalmente mantenuti in un database. I bean basati sui messaggi vengono utilizzati per passare i messaggi in modo asincrono ai moduli e ai servizi dell'applicazione.

Application Server

Le applicazioni enterprise si basano fortemente sull'ausilio di alcuni sistemi, denominati Application Server.

Un Application Server, è uno strato software residente su una macchina software, fornisce dei servizi di fondamentale importanza per le enterprise application, come ad esempio la facilitazione della comunicazione, la gestione delle connessioni ad un database e quella delle transazioni. Ne esistono differenti : Apache, Jboss, Websphere.

Un application server si installa e si avvia come un web server. Infatti un web server è uno dei tanti servizi che l'application server ha in sé.

Per rendere meglio l'idea del container possiamo paragonarlo ad una console per video giochi che consenta all'utente di inserirvi una cartuccia a propria scelta ed eseguire i giochi in essa contenuti. La console fornisce, in questo caso, un insieme di servizi che permettono all'utente di giocare ed interagire con il video game senza che questi debba minimamente preoccuparsi di come ciò avvenga (esonelandolo, quindi, da problemi del tipo: quale tipologia di controller sia utilizzato o come il gioco stesso venga visualizzato a video).



Tomcat

Un servlet container è un componente J2EE responsabile di accettare request, processarle, e fornire una response.

Tomcat è un servlet container e web server. Non è propriamente considerato un application server in quanto non fornisce l'intero set di funzionalità J2EE.

È di forte utilizzo all'interno delle aziende perchè implementa le seguenti funzionalità J2EE:

- Java Servlet
- JavaServer Page
- Java Expression Language
- Java Websocket

Tomcat verrà utilizzato come servlet container per i successivi esercizi di Spring.

- Standalone application

Le applicazioni standalone sono anche note come applicazioni desktop o applicazioni basate su finestre.

Questi sono software tradizionali che dobbiamo installare su ogni macchina.

Esempi di applicazioni standalone sono :

- Media player
- Antivirus.

AWT e Swing sono usati in Java per creare applicazioni standalone.

- Web application

Un'applicazione che viene eseguita sul lato server e crea una pagina dinamica è chiamata applicazione web.

Per la creazione di applicazioni Web in Java sono utilizzate le seguenti tecnologie:

- Servlet
- JSP
- Struts
- Spring
- Hibernate
- JSF
- ecc

- Enterprise application

Un'applicazione enterprise o distribuita ad esempio applicazioni bancarie e così via, è denominata applicazione aziendale.

Presenta vantaggi come

- sicurezza di alto livello
- bilanciamento del carico
- clustering

In Java, EJB viene utilizzato per creare applicazioni aziendali.

- Mobile application

Un'applicazione creata per dispositivi mobili è chiamata applicazione mobile.

Attualmente

- Android
- Java ME (Micro Edition) è una micro platform utilizzata per mobile application

- Packaging di applicazioni

Un'applicazione JAVAEE viene consegnata in :

- JAR
- WAR
- EAR

Un modulo JAVAEE è costituito da una o più componenti

I Moduli sono :

- EJB
- WEB
- Client
- Adapter Resources

- Packaging Enterprise Beans

I bean Enterprise vengono consegnati in :

- JAR
- WAR

Un file JAR EJB è portatile e può essere utilizzato per varie applicazioni.

I bean enterprise forniscono spesso la logica aziendale di un'applicazione Web. In questi casi, il packaging del bean enterprise all'interno del modulo WAR dell'applicazione Web semplifica la distribuzione e l'organizzazione dell'applicazione

- Packaging webarchive

Un modulo Web è la più piccola unità distribuibile e utilizzabile delle risorse Web.

Classi di utilità lato server, ad esempio carrelli della spesa
Classi lato client, ad esempio classi di utilità

- Packaging Resources Adapter Archives

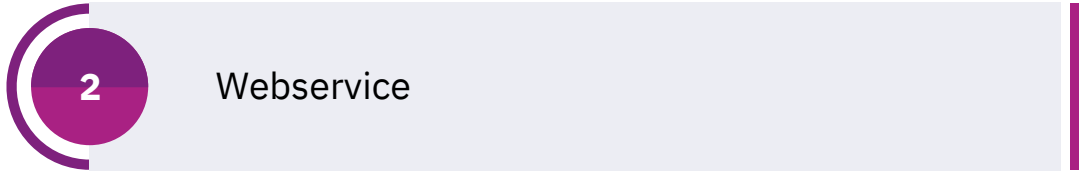
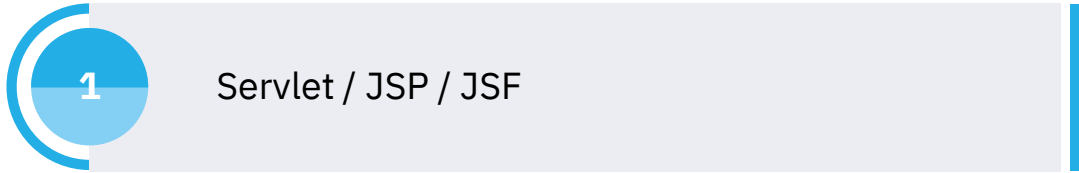
Estensione RAR

Contiene

- Xml
- Classi java

Può essere contenuto in un file .ear oppure da solo

Tecnologie utilizzate



Servlet

Le servlet sono oggetti java residenti sul server.

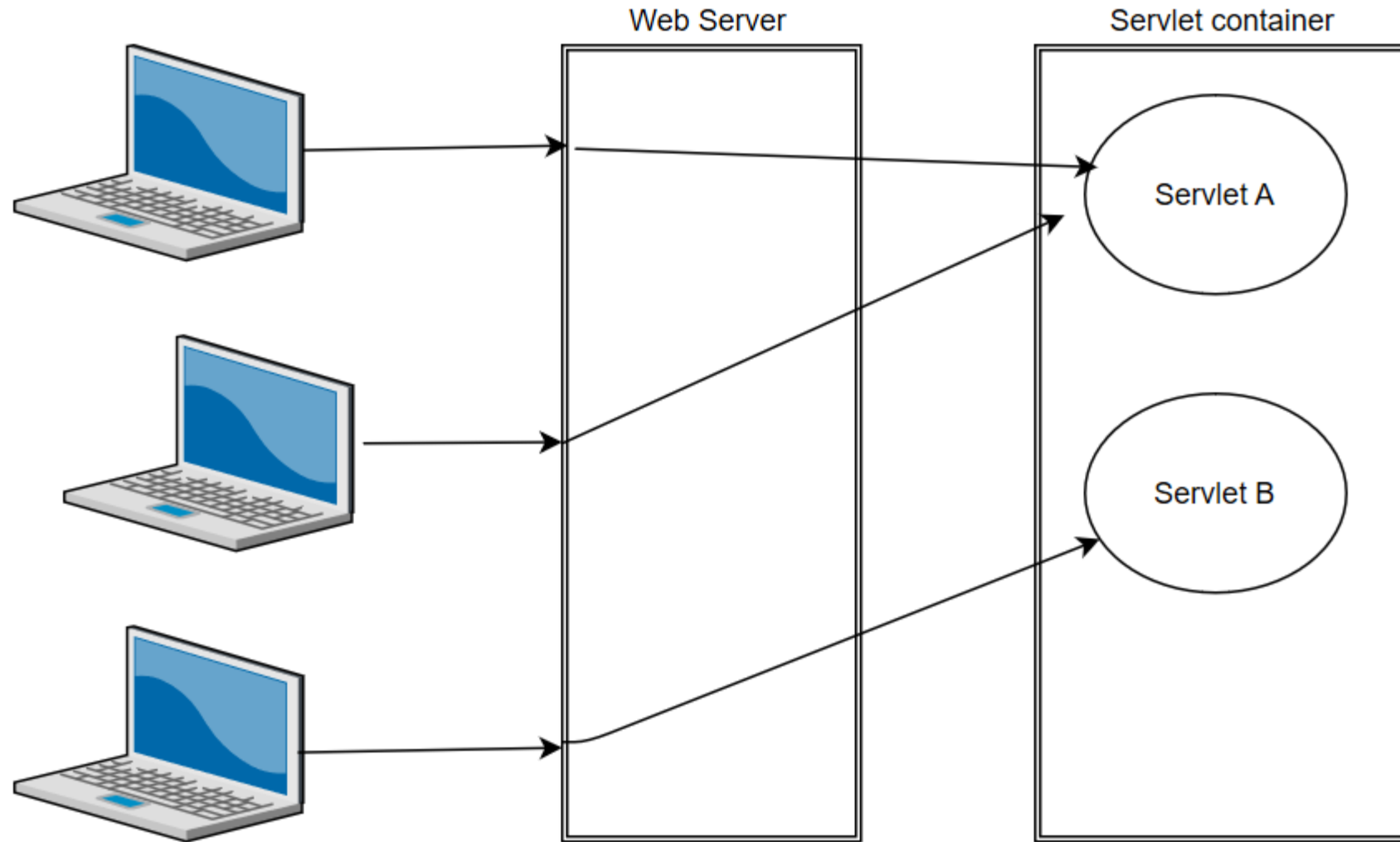
Le servlet sono puri oggetti java

Ereditano da un supertipo appartenente al package `javax.servlet`

Il modo di funzionare ricalca il paradigma client/server

La simulazione di queste due funzioni è garantita dagli oggetti :

- `HttpServletRequest`
- `HttpServletResponse`



Ciclo di vita di una servlet

INIT

Questo metodo viene chiamato una sola volta, subito dopo la sua istanziamento. È in questo metodo che la servlet istanzia tutte le risorse che utilizzerà poi per gestire le richieste

SERVICE

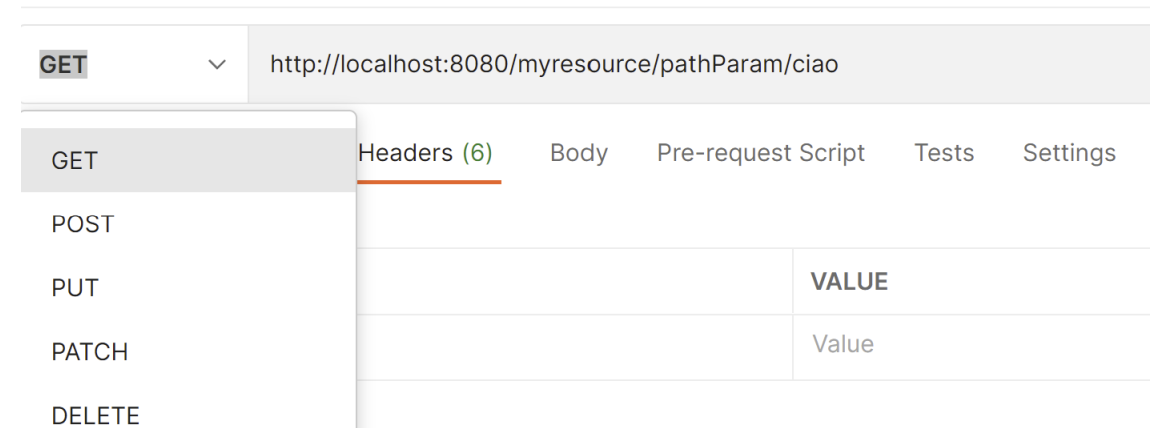
Questo incaricato di gestire le richieste effettuate dal client.

DESTROY

Questo metodo segna la chiusura della servlet. In questo modo si effettuato eventuali salvataggi di informazioni utili.

Metodi Http

- **GET** - Richiede una risorsa al server. (Idempotente)
- **POST** - Invia informazioni all'URI specificato. (Aggiorna parzialmente una risorsa)
- **PUT** - Istanza una risorsa sul server, creandola o riscrivendola
- **DELETE** - Cancella una risorsa (file) sul server.
- **TRACE** - Traccia una richiesta, visualizzando come viene trattata dal server.
- **OPTIONS** - Richiede l'elenco dei metodi permessi dal server.
- **HEAD** - Richiede solo l'header, senza la risorsa (il file HTML, l'immagine, ecc.). Di fatto viene usato soprattutto per diagnostica.



Metodi principali di una servlet

- `void doGet(HttpServletRequest req, HttpServletResponse resp)`
- `void doPost (HttpServletRequest req, HttpServletResponse resp)`
- `void service (HttpServletRequest req, HttpServletResponse resp)`
viene invocato dopo la `init()`. Smista le richieste pervenute dal client
- `void doPut (HttpServletRequest req, HttpServletResponse resp)`
consente a un client di inserire un file sul server
- `void delete (HttpServletRequest req, HttpServletResponse resp)`
consente a un client di cancellare un file sul server
- `void init()`
viene invocato solo una volta al caricamento della server e prima che inizi a esaudire le richieste
- `void destroy()`
scarica la servlet dalla memoria



Experience.
Create.
Inspire.

Gianluca Avizzano