Introduzione a Java EE

Programmazione Distribuita - A.A. 2020/2021

Biagio Cosenza
Dipartimento di Informatica
Università di Salerno
http://cosenza.eu/
bcosenza@unisa.it

Organizzazione della Lezione

- Introduzione
- Architettura
  - Multilayer e multitier
  - Containers
  - Packaging
  - Annotazioni e Deployment Descriptor
- L'ecosistema JEE
  - Standard
  - Storia
  - Tecnologie
- Conclusioni

Motivazioni (1)

- Le imprese vivono in un mondo competitivo, globale
  - hanno necessità di applicazioni software complesse, distribuite anche su continenti diversi
  - eseguono business 24/7, hanno diversi datacenter, sistemi internazionalizzati, diverse valute/time-zone, etc.
  - allo stesso tempo, per tali aziende, è necessario:
    - riduzione dei costi
    - riduzione dei tempi di risposta dei servizi
    - storing dei dati su storage affidabili e sicuri
    - fornire interfacce mobile Web verso i clienti, fornitori (integrazione) ed impiegati (supporto interno)

Motivazioni (2)

- Necessario anche:
  - Combinare tutte queste challenges con gli *Enterprise Information Systems* preesistenti (EIS) di tali aziende
  - Sviluppare applicazioni B2B per la comunicazione con partners o applicazioni B2C usando applicazioni mobile o geolocalizzate
  - Coordinare dati in-house, memorizzati in differenti locazioni (eterogeneità di piattaforme), processati da diversi linguaggi, instradati attraverso protocolli diversi
- Tutto questo garantendo nessuna perdita di denaro, il che significa:
  - evitare system crash, garantire disponibilità, scalabilità, sicurezza
- Per questo motivo nasce Java EE

Enterprise Computing

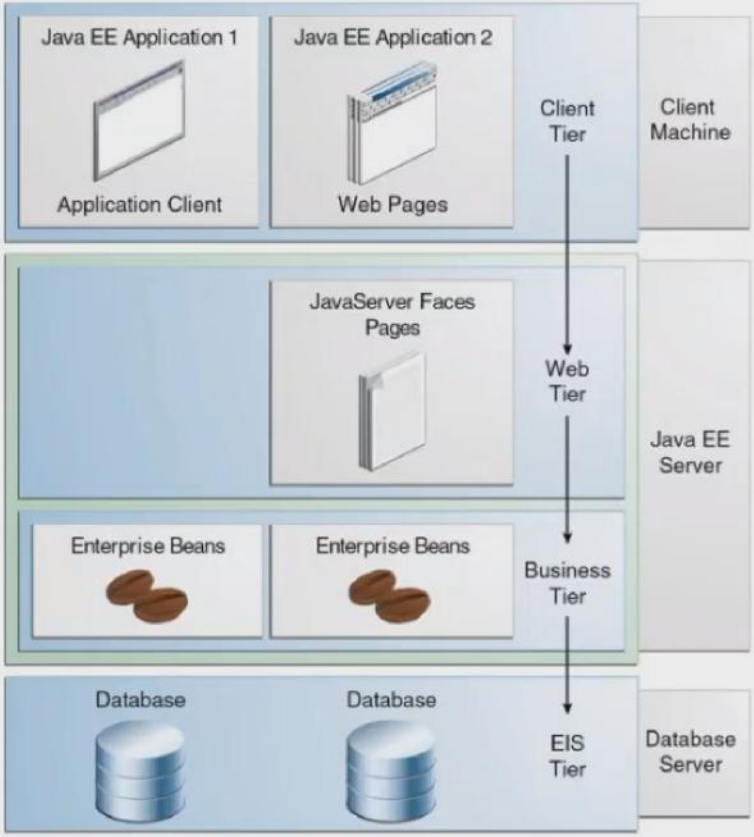
- Applicazioni distribuite, transazionali e portabili, che garantiscono l'efficienza, la sicurezza e l'affidabilità della tecnologia lato server
- Obiettivo: più efficienza con meno risorse e minori investimenti, garantendo alta disponibilità, scalabilità e sicurezza
  - riduzione del tempo di sviluppo e della complessità delle applicazioni
  - aumento delle application performance
- Java Enterprise Edition cerca di rispondere a queste esigenze

Java EE

- L'idea:
  - Quando si ha necessità di lavorare con collections of objects non ci si scrive una propria HashTable, ma si usa una `Collection` di Java SE
  - Se si vuole una applicazione Web-based, transazionale, sicura, interoperabile, scalabile e distribuita, non ci si scrive tutto da zero, ma si usa
    - *Java Transaction API* (JTA),
    - si comunica con *Java Message Service* (JMS) e
    - si realizza la persistenza con *Java Persistence API* (JPA)
- Java EE è un insieme di specifiche progettate per applicazioni enterprise
  - Estensione di Java SE per facilitare lo sviluppo di applicazioni distribuite, robuste, powerful, altamente disponibili

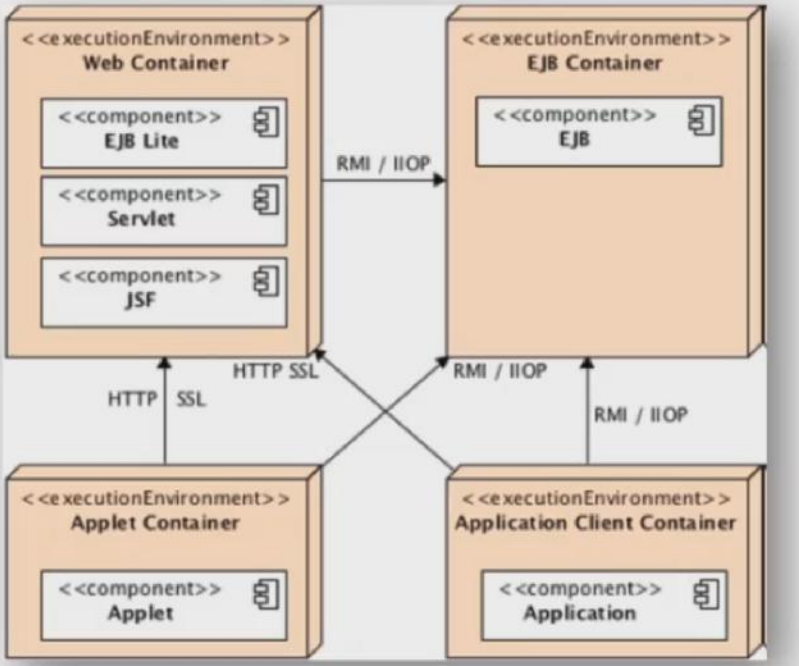
Architettura Multilayer

- Modello multilayer per enterprise
- Tre layer
  1. client
  2. server EE
  3. database/legacy
- La logica dell'applicazione separate in componenti
- Componenti in diversi layer, mappati su diversi tier in un ambiente Java EE



Architettura: Container (1)

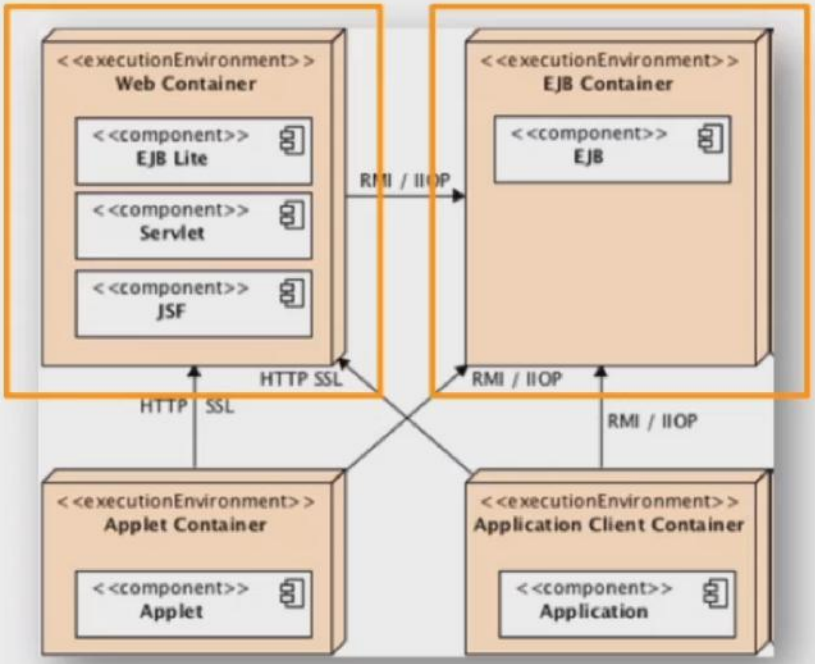
- Java EE infrastructure è partizionata in domini logici chiamati container
- Ogni container:
  - ha uno specifico ruolo
  - supporta un set di API
  - offre servizi alle componenti (security, database access, transaction handling, naming directory, resource injection)
- I container nascondono complessi dettagli tecnici e migliorano la portabilità



Standard Java EE containers

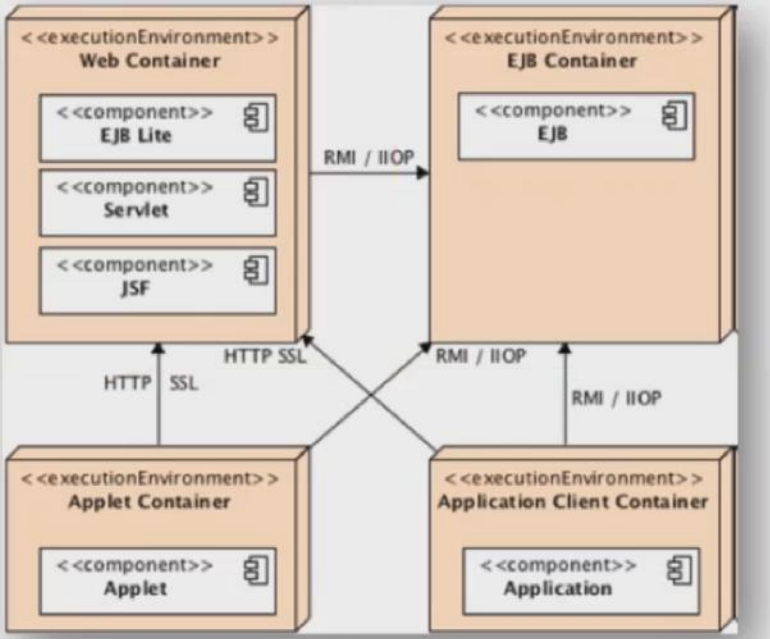
Architettura: Container (2)

- A seconda dell'applicazione che si vuole costruire, bisogna capire funzionalità e limiti di ciascun container
- Ad esempio, se si vuole costruire una web application, bisognerà sviluppare un JSF tier con un EJB Lite tier e farne il deploy in un Web container
- Se però si vuole una web application che invochi un business tier da remoto, e usare messaging e asynchronous calls, allora c'è necessità sia di un web che di un EJB container



Architettura: Container (3)

- Java EE ha 4 differenti container:
  - Applet containers
  - Application client container (ACC)
  - Web container
  - EJB container
- Prima di descrivere i container, vediamo i diversi tipi di componenti che è possibile sviluppare con Java EE



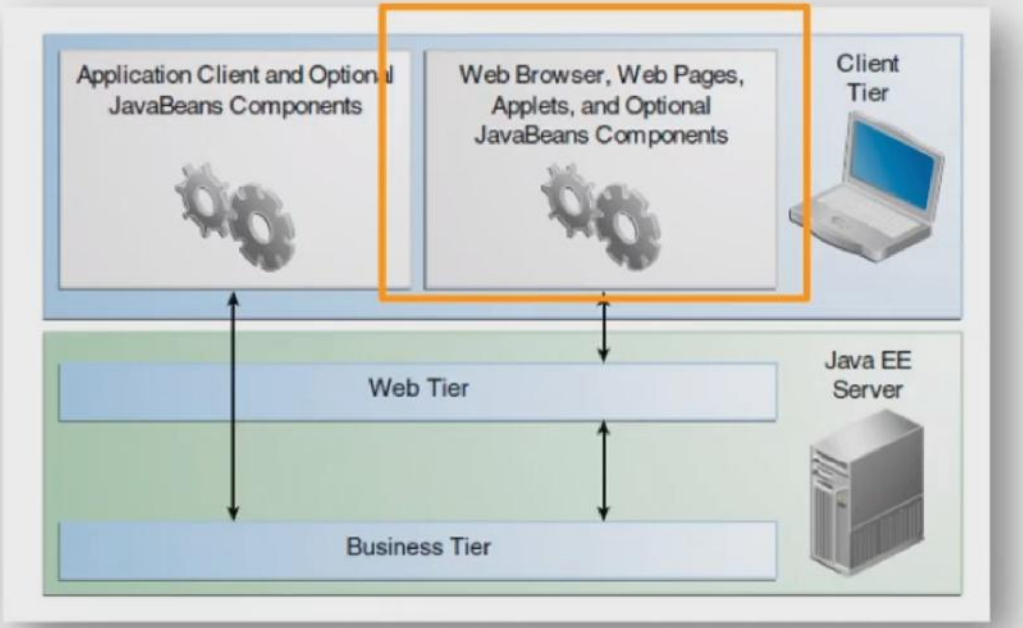
Architettura: Componenti

- Java EE runtime environment definisce diversi tipi di componenti:

1. Client components (Java EE Clients)
   - Web Clients
   - Application Clients
   - Applets
2. Web components
3. Business components

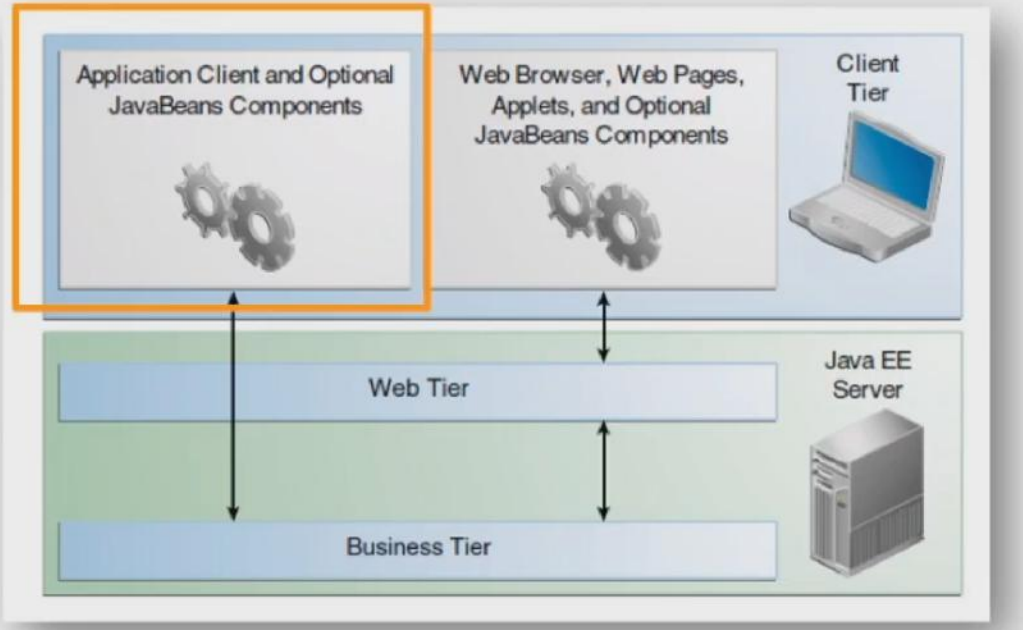
Client components: Web Client (Applets)

- Web client (thin client)
  - Applicazioni eseguite su web browser
  - Pagina dinamiche (HTML, XML, etc.)
  - Possibile che contengano applet: piccola applicazione che viene eseguita sul browser del client
  - Thin client
    - no query a database
    - nessuna logica di business complessa
    - no connessione a legacy applications



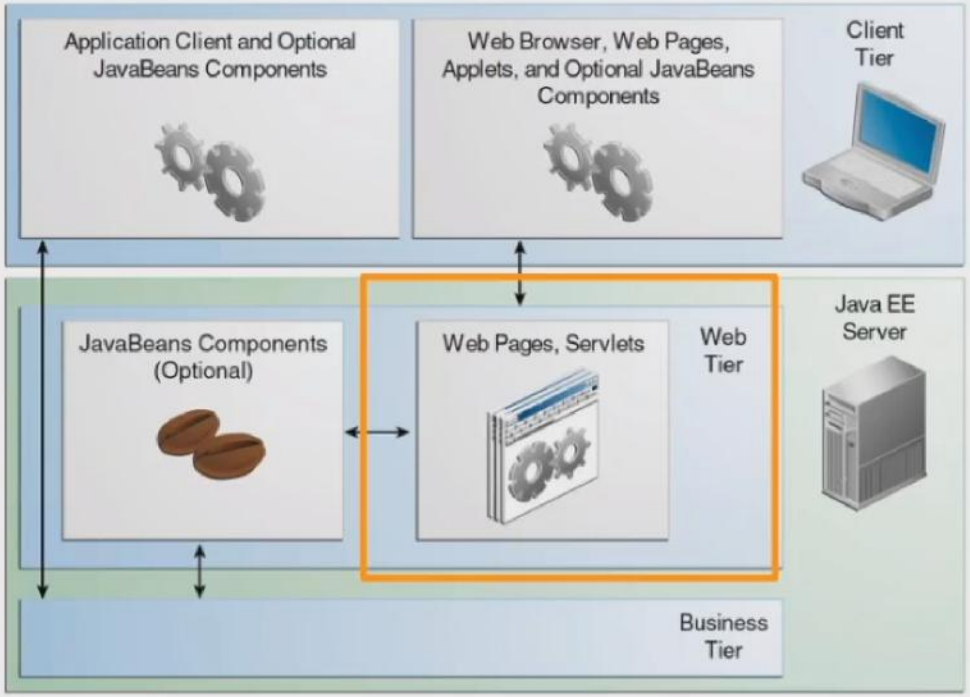
Client components: Application Clients

- Application clients
  - Programmi eseguiti sul client
  - Interfaccia utente più ricca
  - GUI creata con toolkit Java (Swing, AWT, etc.)
  - Accesso diretto allo strato di business (Middle tier)
  - Possibile anche, se necessario, il passaggio via strato Web (HTTP)



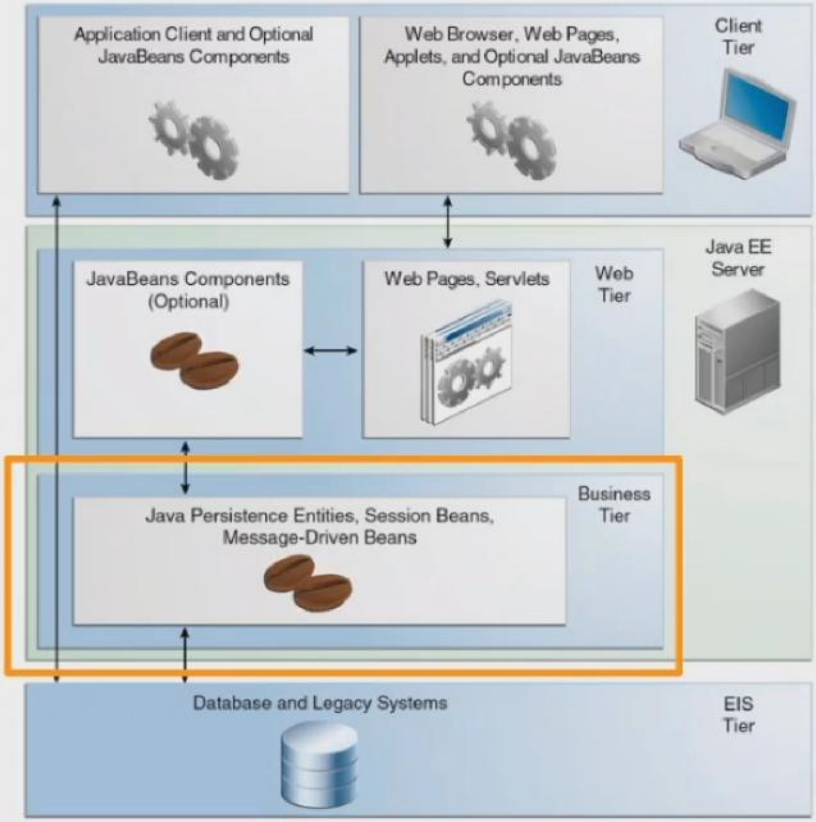
Web Components

- Web Components
  - Applicazioni eseguite in un web container che rispondono a richieste HTTP da web client
  - Servlet, o pagine create usando Java Server Faces /Java Server Pages
    - Servlet: classi che dinamicamente processano richieste e costruiscono risposte
    - JSP pages: Text-based documents che eseguono servlet
    - JavaServer Faces technology: costuite sulla tecnologia delle servlet e JSP per fornire pagine dinamiche
  - Può includere componenti JavaBeans



Business Components

- Business Components
  - Eseguite in un EJB container
  - Enterprise Java Beans, Java Message Service, Java Transaction API, asynchronous calls, RMI/IIOP
  - Gli EJB sono container-managed components for transactional business logic
  - Possono essere acceduti localmente o da remoto attraverso RMI
    - oppure HTTP per SOAP e RESTful Web Service



Java EE Container

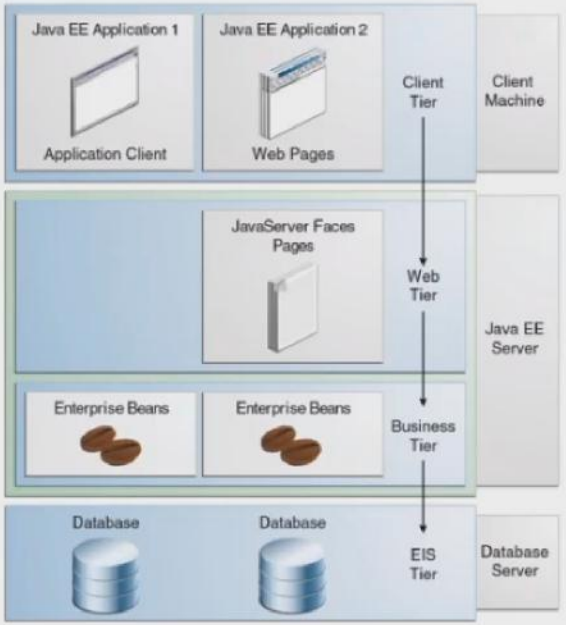
- Il server Java EE fornisce servizi sotto forma di un container per ogni tipo di componente
  - poiché lo sviluppatore non deve sviluppare questi servizi, può concentrarsi sulla logica di business

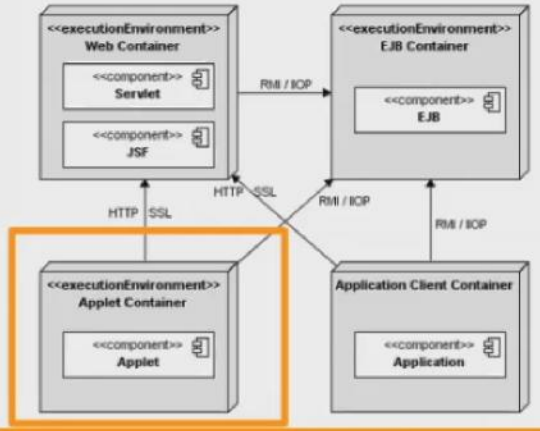
Java EE Container

- I container rappresentano l'interfaccia tra una componente e le funzionalità a basso livello che supporta la componente
- Prima di essere eseguita, una componente (Web, Enterprise bean, o application client) deve essere assemblata in un Java EE module e deployata nel suo container
- Questo packaging specifica per ogni componente i settaggi del container
  - questi settaggi personalizzano il supporto fornito dal Java EE server, ad esempio servizi quali security, transaction management, Java Naming and Directory Interface (JNDI) API lookups, remote connectivity
    - il servizio di sicurezza permette di configurare l'autenticazione e autorizzazione di componenti web o enterprise bean
    - il servizio di transazioni permette di definire una transazione composta dall'invocazione di diversi metodi
    - JNDI fornisce un'unica interfaccia che le componenti usano per accedere ai servizi e risorse del server
    - invocazione remota di metodi

Applet Containers

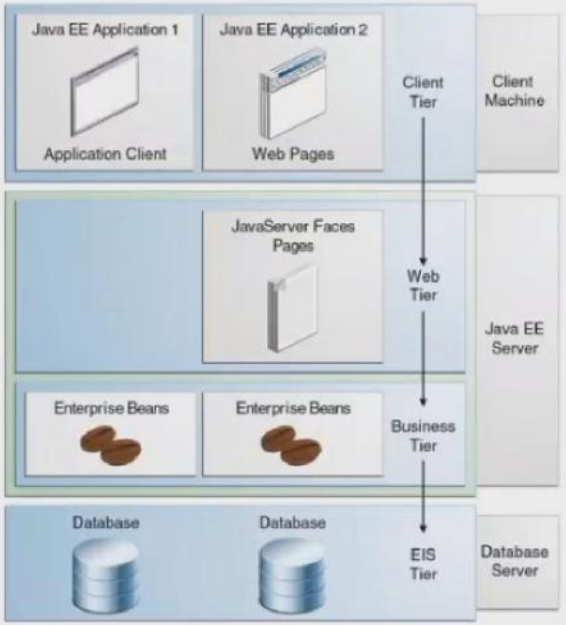
- Fornito dalla maggior parte dei browser per eseguire applet components
- Sicurezza fornita da sandbox
  - accesso limitato alla macchina client
- Impedisce accesso al computer locale per accesso a processi o files

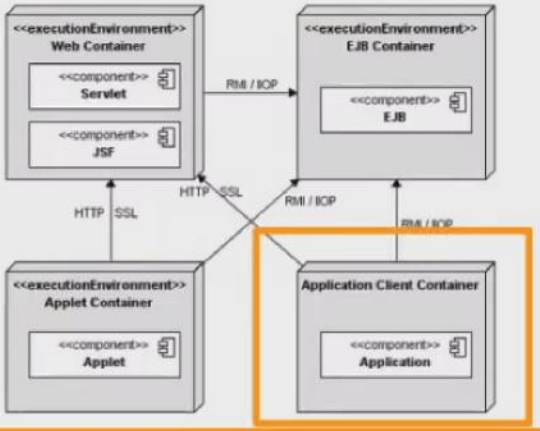




Application Client Container (ACC)

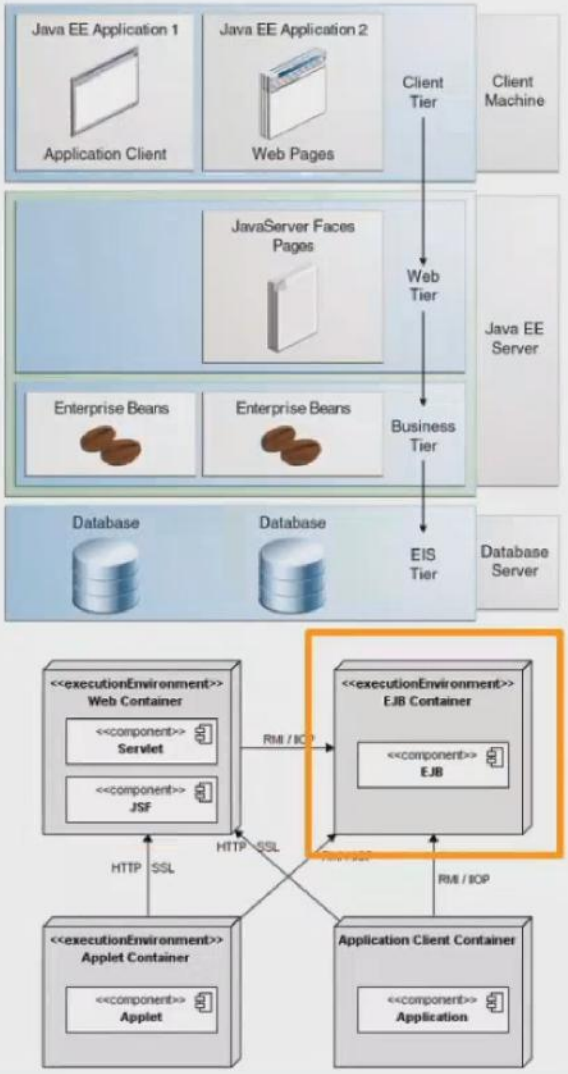
- Insieme di classi Java, librerie etc. che permettono di usare servizi Java EE in applicazioni SE
- Servizi quali sicurezza, naming etc acceduti da una applicazione standard (con un `main()`)
- L'ACC comunica
  - con l'EJB container usando RMI-IIOP e
  - con il Web container usando HTTP (Web services)





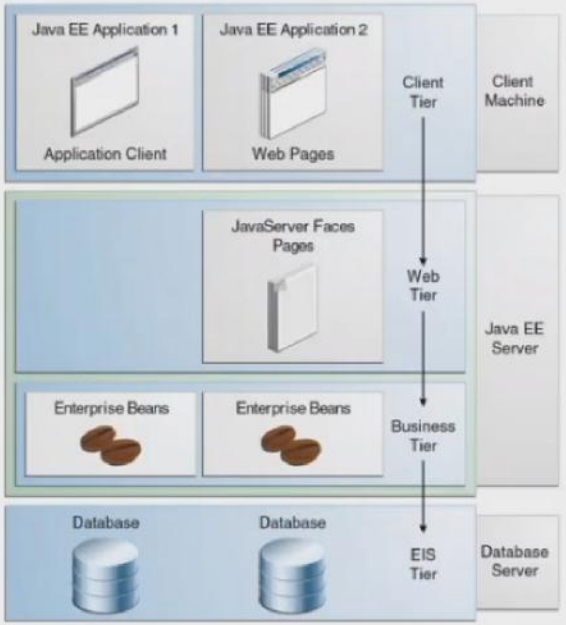
EJB Containers

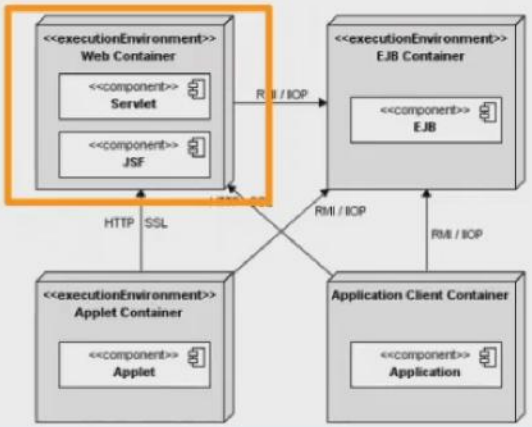
- Responsabile della gestione dei bean
- Business logic layer
- Gestisce il ciclo di vita degli EJB
- Fornisce transazioni, sicurezza, concorrenza, distribuzione, servizio di naming, invocazioni asincrone



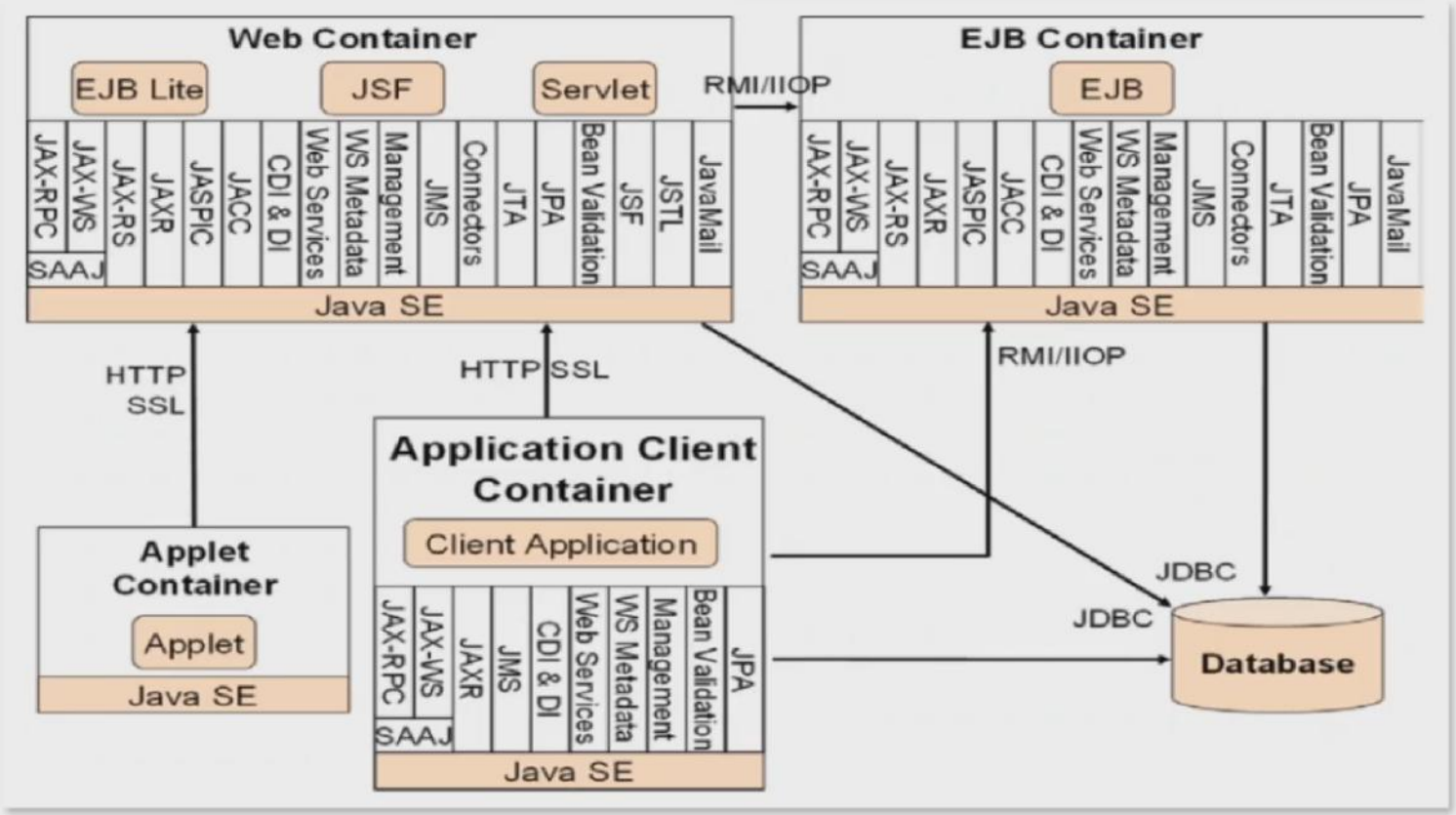
Web Containers

- Servizi per gestione e esecuzione di componenti web
- Ad esempio, servlet, JSPs, filtri, listeners, pagine JSF, web services
- Responsabile per inizializzazione, invocazione e gestione del ciclo di vita delle servlet





I Servizi Forniti dal Container



Container: i servizi piu importanti

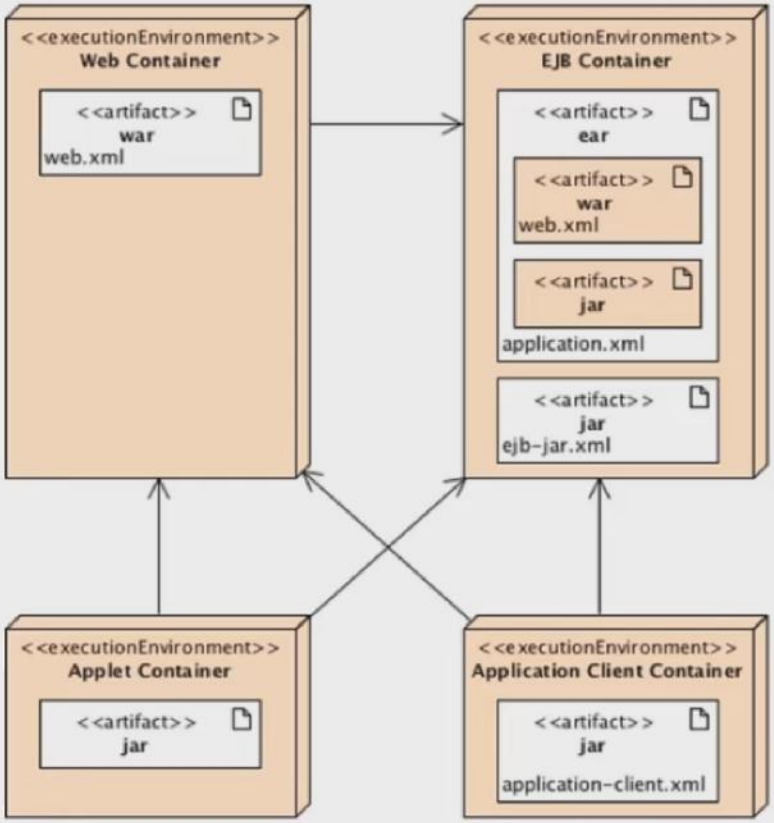
- Java Transaction API: abilita le transazioni distribuite
- Java Persistence API: standard API per object-relational mapping (ORM)
  - col linguaggio Java Persistence Query Language (JPQL) si possono fare query su oggetti
- Java Message Service: comunicazione asincrona tra le component
  - comunicazione affidabile point-to-point e publish-subscribe
- Servizi di sicurezza: Java Authentication and Authorization Service (JAAS)
- Web Services: Java API for XML Web Services (JAX-WS) e Java API for RESTful Web Services (JAX-RS)
- Dependency injection: risorse possono essere iniettate nei componenti "managed"

Architettura: Packaging

- Prima di effettuare il deploy in un container, le componenti devono essere formattate in un archivio standard
  - Java SE definisce Java Archive (jar) files, usato per aggregare diversi tipi di files in un file compresso (zip format)
    - Java classes, deployment descriptors, resources, external libraries
- Java EE definisce differenti tipi di moduli con il proprio packaging format, basato sul proprio jar format

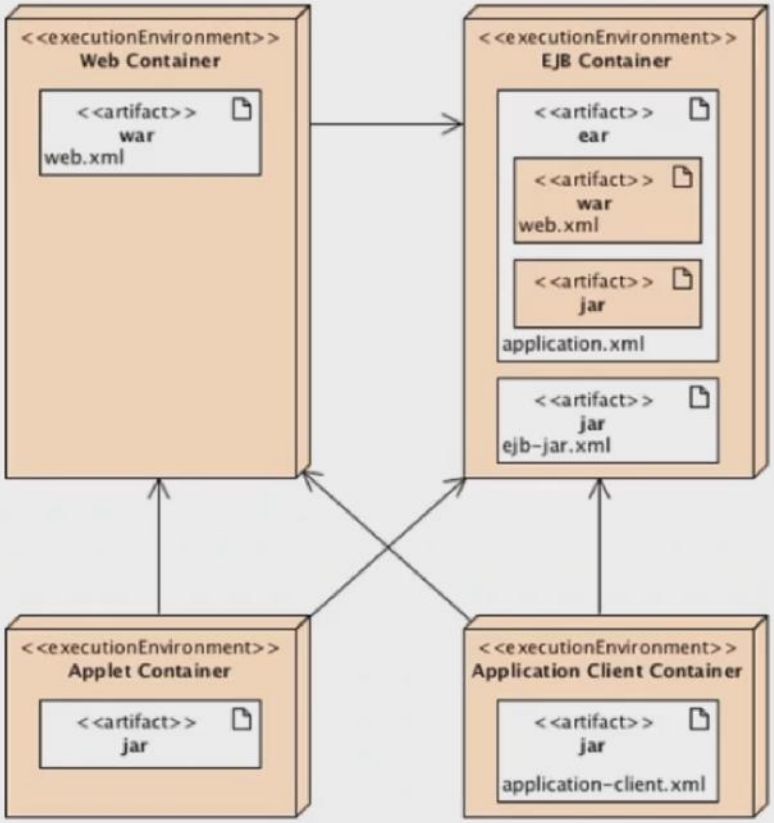
Packaging: Application Client Module

- Application client module: contiene classi Java e altre risorse packaged in a jar file
- Il jar può essere eseguito in un Java SE environment o in un application client container
- Il jar contiene la directory META-INF con meta information che descrivono l'archivio
  - `MANIFEST.MF` usato per definire extension and package related data
- Se *deployato* in un ACC, il deployment descriptor si troverà nel file `META-INF/application-client.xml`



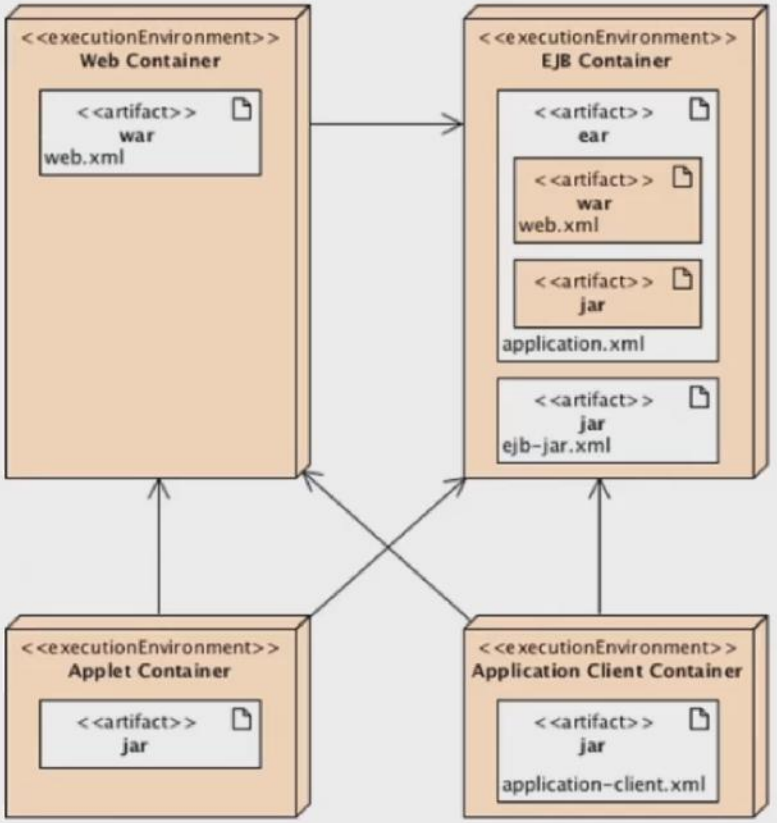
Packaging: EJB Module

- EJB module: contiene uno o più session o *Message-driven beans* (MDBs) impacchettati in un jar file
- Deployment descriptor in `META-INF/ejb-jar.xml`



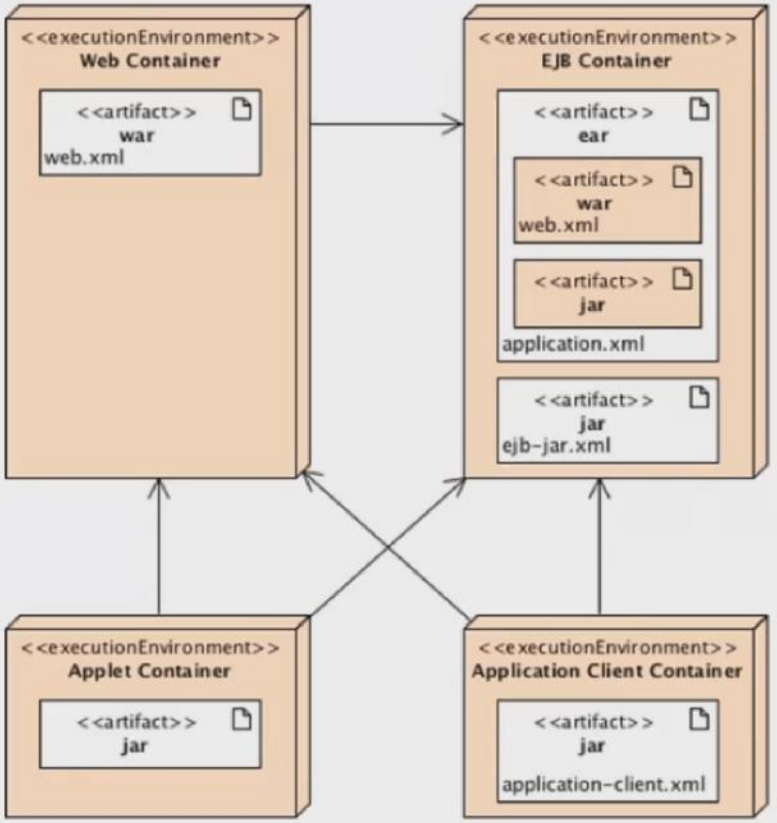
Packaging: Web Application Module

- Web application module: contiene servlet, JSP, JSF, HTML, CSS, JS, media, etc.
- File `jar` con estensione `.war`
- Deployment descriptor in `WEB-INF/web.xml`
- Se il war contiene un EJB Lite beans il deploymnt descriptor settato in `WEB-INF/ejb-jar`
- Le classi in `WEB-INF/classes` e altri jar in `WEB-INF/lib`



Packaging: Enterprise Module

- Enterprise module: include moduli EJB e moduli Web applications e altre librerie esterne
- Archivio `jar` con estensione `.ear`
- Permette il deployment coerente in un unico passo
- Deployment descriptor in `META-INF/application.xml`



Architettura: Annotazioni e Deployment Descriptor

- In un programming paradigm esistono due differenti approcci
  - imperative programming
  - declarative programming
- Imperative programming: specifica l'algoritmo per raggiungere un obiettivo
  - what has to be done
- Declarative programming: specifica come raggiungere questo obiettivo:
  - how it has to be done
- In Java EE, il declarative programming è realizzato attraverso l'uso di metadata
- Annotations e/o deployment descriptors

Architettura: Annotazioni e Deployment Descriptor

- Le componenti sono eseguite in un container ed ogni container offre un insieme di servizi
- I metadati sono usati per dichiarare e personalizzare questi servizi
  - associando informazioni addizionali a: classi Java, interfacce, costruttori, metodi, campi, parametri
- Il tutto viene realizzato con l'uso di annotazioni
- Annotazioni nel codice: keyword `@xxx` (possibili con parametri)
- Permettono ad un "*Plain Old Java Object*" di diventare una componente
- Altri meccanismi sono quelli di dichiarare un deployment descriptor scritto in XML

Esempio di EJB con Annotazioni

- Senza stato da mantenere
- La definizione remota
  - ...e quella locale
- La persistenza

```
@Stateless
@Remote(ItemRemote.class)
@Local(ItemLocal.class)
@LocalBean

public class ItemEJB implements ItemLocal,
   ItemRemote {

   @PersistenceContext(unitName="chapter01PU")
   private EntityManager em;

   public Book findBookById(Long id) {
       return em.find(Book.class, id);
   }
}
```

Un Deployment Descriptor Equivalente

- Il nome del bean
- La classe remota
  - ...quella locale
- Il tipo

- I deployment descriptor devono essere *packaged* con le componenti
  - nelle directory `META-INF` o `WEB-INF`

```
<?xmlversion="1.0"?>

<ejb-jar xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
  http://xmlns.jcp.org/xml/ns/javaee/ejb-jar_3_2.xsd"
  version="3.2">
  <enterprise-beans>
    <session>
      <ejb-name>ItemEJB</ejb-name>
      <remote>org.agoncal.book.javaee7.ItemRemote</remote>
      <local>org.agoncal.book.javaee7.ItemLocal</local>
      <local-bean/>
      <ejb-class>org.agoncal.book.javaee7.ItemEJB</ejb-
      class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
</ejb-jar>
```

Differenza fra i due stili

- La maggior parte dei deployment descriptor sono opzionali e si possono usare le annotazioni
- Le annotazioni riducono la quantità di codice da scrivere
  - meno file e meno testo
- Il vantaggio dei deployment descriptor è che possono essere modificati senza richiedere modifiche al codice sorgente e ricompilazioni
- Se esistono entrambi, XML ha precedenza su annotazioni
  - Il meccanismo preferito è attualmente quello delle annotazioni

Deployment Descriptor in Java EE

Table 1-1. Deployment Descriptors in Java EE

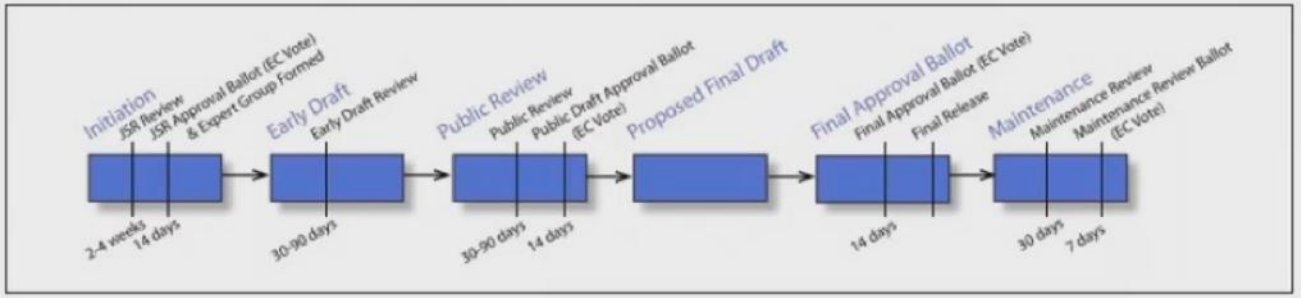
File	Specification	Paths
application.xml	Java EE	META-INF
application-client.xml	Java EE	META-INF
beans.xml	CDI	META-INF or WEB-INF
ra.xml	JCA	META-INF
ejb-jar.xml	EJB	META-INF or WEB-INF
faces-config.xml	JSF	WEB -INF
persistence.xml	JPA	META-INF
validation.xml	Bean Validation	META-INF or WEB-INF
web.xml	Servlet	WEB-INF
web-fragment.xml	Servlet	WEB-INF
webservices.xml	SOAP Web Services	META-INF or WEB-INF

Ecosistema Java EE: l'importanza degli standard

- Java EE è basato su standard
  - sviluppati attraverso il Java Community Process (JCP)
  - Java EE come una "specifica ombrello" che copre tante altre specifiche
- Importante la standardizzazione: facilità di comunicazione e di scambio (valute, tempo, misure, ferrovie, elettricità, telefoni, protocolli, linguaggi)
- Java EE fornisce un ambiente open (no vendor lock-in) con diversi server commerciali (WebLogic, Websphere, MQSeries, etc.) e open source (GlassFish, JBoss, Hibernate, Open JPA, Jersey, etc.) per gestire transazioni, security, persistenza, ecc.
  - le applicazioni possono essere deployate in un qualunque application server con pochi cambiamenti

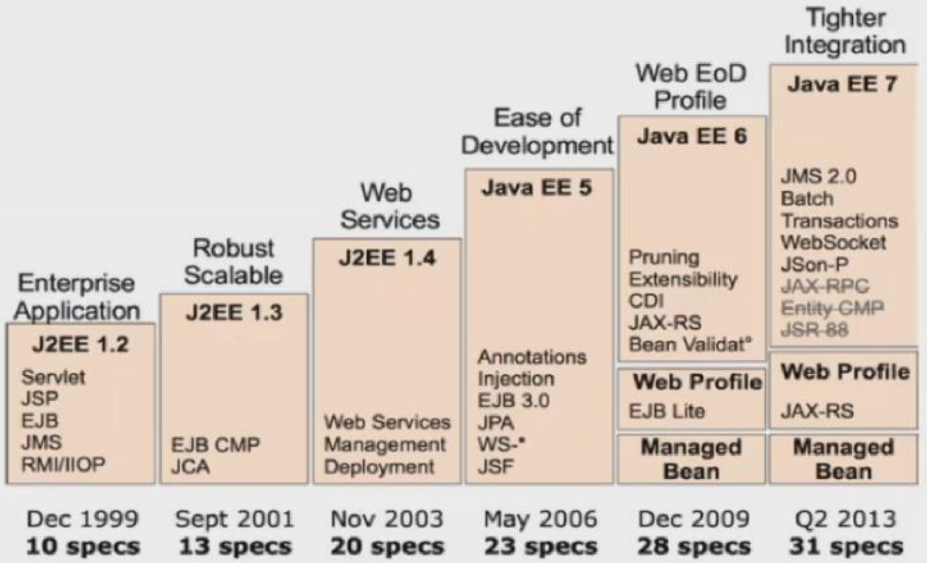
Java Community Process

- Creato nel 1998 da Sun, organizzazione coinvolta per la definizione di versioni e caratteristiche future di Java
- Quando c'è necessità, viene creato un Java Specification Request (JSR)
  - con un gruppo di esperti rappresentanti di aziende, organizzazioni, università, individui singoli
- Un JSR sviluppa
  - le specifiche
  - una Reference Implementation
  - un Compatibility Test Kit
- L'approvazione avviene da parte dell'Executive Committee di JCP



Evoluzione di Java EE

- Java EE 8 - 2017
  - Java Servlet 4.0 API with HTTP/2 support
  - Enhanced JSON support including a new JSON binding API
  - A new REST Reactive Client API
  - Asynchronous CDI Events
  - A new portable Security API
  - Server-Sent Events support (Client & Server-side)
  - Support for Java SE 8 new capabilities
    - e.g. Date & Time API, Streams API, annotations enhancements



Ecosistema JEE: Tecnologie

- Specifiche di Java EE 7
  - Definita da JSR 342 contiene altre 31 specifiche
  - Possibile raggruppare per profili (da JAVA EE6).
    - In Java EE7 esiste il profilo "Web profile" che specifica applicazioni strettamente orientate al web (poca logica di business)
    - Il Web profile ha una sua specifica JSR

Ecosistema JEE: Specifiche di Java EE 7

Table 1-2. *Java Enterprise Edition Specification*

Specification	Version	JSR	URL
Java EE	7.0	342	http://jcp.org/en/jsr/detail?id=342
Web Profile	7.0	342	http://jcp.org/en/jsr/detail?id=342
Managed Beans	1.0	316	http://jcp.org/en/jsr/detail?id=316

Table 1-3. *Web Services Specifications*

Specification	Version	JSR	URL
JAX-WS	2.2a	224	http://jcp.org/en/jsr/detail?id=224
JAXB	2.2	222	http://jcp.org/en/jsr/detail?id=222
Web Services	1.3	109	http://jcp.org/en/jsr/detail?id=109
Web Services Metadata	2.1	181	http://jcp.org/en/jsr/detail?id=181
JAX-RS	2.0	339	http://jcp.org/en/jsr/detail?id=339
JSON-P	1.0	353	http://jcp.org/en/jsr/detail?id=353

Ecosistema JEE: Specifiche di Java EE 7

Table 1-4. *Web Specifications*

Specification	Version	JSR	URL
JSF	2.2	344	http://jcp.org/en/jsr/detail?id=344
JSP	2.3	245	http://jcp.org/en/jsr/detail?id=245
Debugging Support for Other Languages	1.0	45	http://jcp.org/en/jsr/detail?id=45
JSTL (JavaServer Pages Standard Tag Library)	1.2	52	http://jcp.org/en/jsr/detail?id=52
Servlet	3.1	340	http://jcp.org/en/jsr/detail?id=340
WebSocket	1.0	356	http://jcp.org/en/jsr/detail?id=356
Expression Language	3.0	341	http://jcp.org/en/jsr/detail?id=341

Table 1-5. *Enterprise Specifications*

Specification	Version	JSR	URL
EJB	3.2	345	http://jcp.org/en/jsr/detail?id=345
Interceptors	1.2	318	http://jcp.org/en/jsr/detail?id=318
JavaMail	1.5	919	http://jcp.org/en/jsr/detail?id=919
JCA	1.7	322	http://jcp.org/en/jsr/detail?id=322
JMS	2.0	343	http://jcp.org/en/jsr/detail?id=343
JPA	2.1	338	http://jcp.org/en/jsr/detail?id=338
JTA	1.2	907	http://jcp.org/en/jsr/detail?id=907

Ecosistema JEE: Specifiche di Java EE 7

Table 1-6. *Management, Security, and Other Specifications*

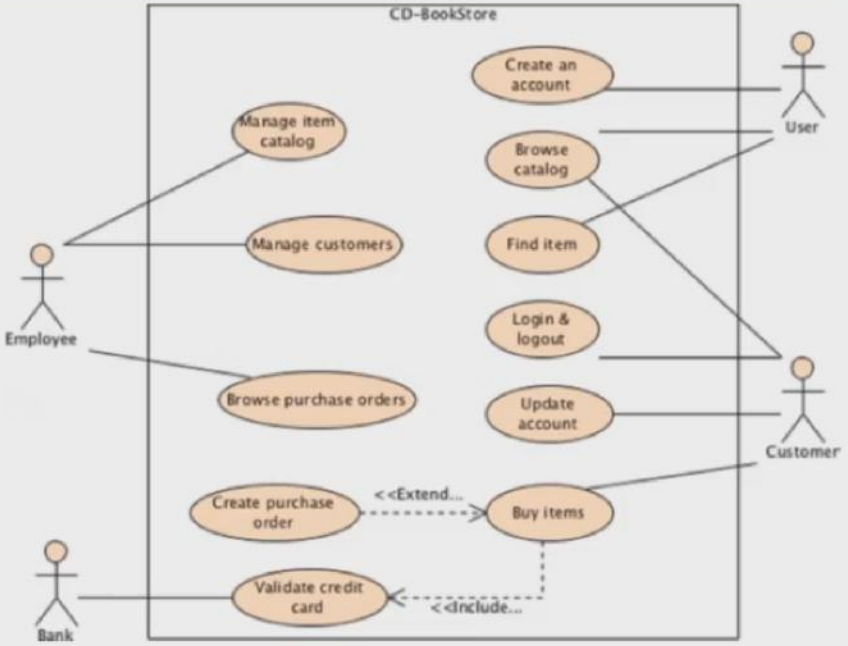
Specification	Version	JSR	URL
JACC	1.4	115	http://jcp.org/en/jsr/detail?id=115
Bean Validation	1.1	349	http://jcp.org/en/jsr/detail?id=349
Contexts and Dependency Injection	1.1	346	http://jcp.org/en/jsr/detail?id=346
Dependency Injection for Java	1.0	330	http://jcp.org/en/jsr/detail?id=330
Batch	1.0	352	http://jcp.org/en/jsr/detail?id=352
Concurrency Utilities for Java EE	1.0	236	http://jcp.org/en/jsr/detail?id=236
Java EE Management	1.1	77	http://jcp.org/en/jsr/detail?id=77
Java Authentication Service Provider Interface for Containers	1.0	196	http://jcp.org/en/jsr/detail?id=196

Java EE 8

- Java EE 8
- JSR-000366 JavaTM Platform, Enterprise Edition 8 (Java EE 8) Specification (Final Release)
  - https://jcp.org/aboutJava/communityprocess/final/jsr366/index.html

CD-BookStore

- E-commerce Web site
- Permette ad un cliente di:
  - sfogliare un catalogo di libri e CD in vendita
  - usare uno shopping cart per aggiungere e rimuovere items
  - controllare se si è in grado di pagare ed ottenere un ordine di acquisto



- Il sistema ha collegamenti con un sistema bancario per la validazione di numeri di carte di credito

Conclusioni

- Introduzione
- Architettura
  - Multilayer e multitier
  - Containers
  - Packaging
  - Annotazioni e Deployment Descriptor
- L'ecosistema JEE
  - Standard
  - Storia
  - Tecnologie
- Conclusioni