



# Introduzione a Java EE



Corso di Laurea in Informatica, Programmazione Distribuita  
Delfina Malandrino, [dmalandrino@unisa.it](mailto:dmalandrino@unisa.it)  
<http://www.unisa.it/docenti/delfinamalandrino>

1

## Organizzazione della lezione

2

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

2

1

## Organizzazione della lezione

3

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

3

## Motivazioni

4

- 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)

4

## Motivazioni

5

- Necessario anche:
  - ▣ Combinare tutte queste complex 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 in-house data, 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

5

## Enterprise computing

6

- 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 Enterprise Edition (Java EE)



6

## Understanding Java EE

7

- 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 . . .
  - Quindi... 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

7

## Organizzazione della lezione

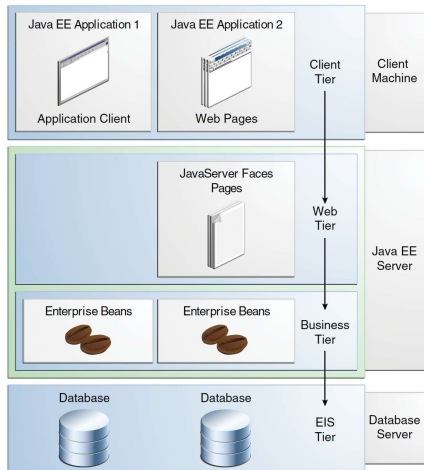
8

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

8

## Architettura multilayer

9



- Modello multilayer per enterprise
- La logica dell'applicazione separate in componenti
- Componenti in diversi layer, mappati su diversi tier in un ambiente Java EE
- Classico il mapping su 3-tier: client, server EE, database/legacy

9

## Architettura

10

- 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

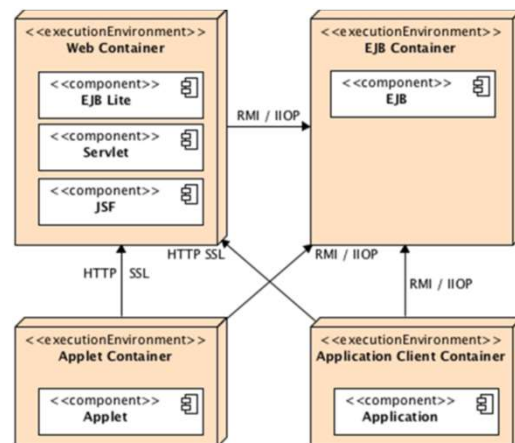


Figure 1-1. Standard Java EE containers

10

## Architettura

11

- Relazione logica fra container
- Le frecce rappresentano i protocolli usati per accedere da un container ad un altro
  - Ad esempio, il web container esegue servlet che possono accedere agli EJB attraverso RMI-IIOP

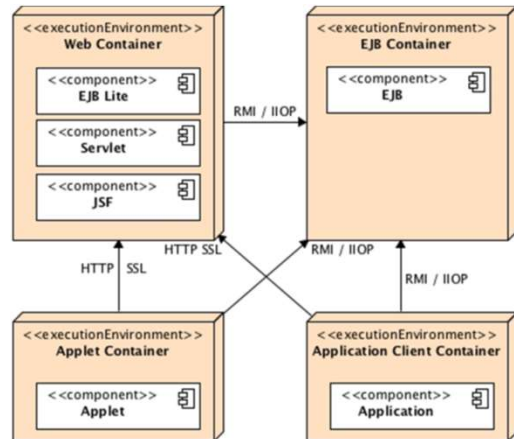


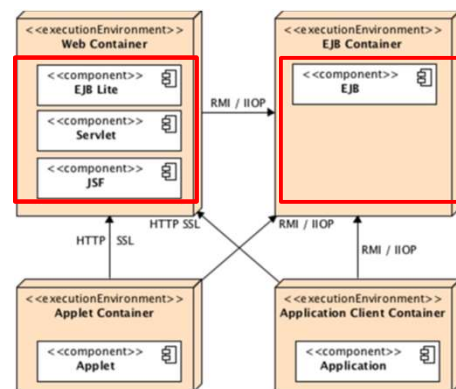
Figure 1-1. Standard Java EE containers

11

## Container

12

- 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

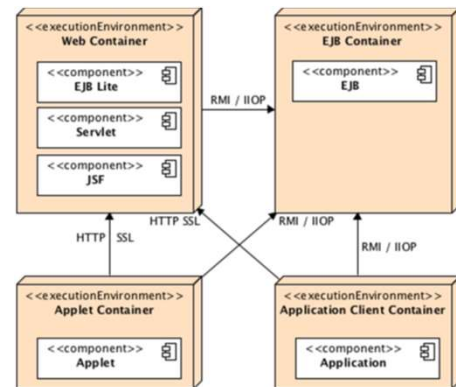


12

## Container

13

- 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



13

## Componenti

14

- Java EE runtime environment definisce differenti tipi di componenti:
  - ▣ Client components (Java EE Clients)
    - Web Clients
    - Application Clients
    - Applets
  - ▣ Web components
  - ▣ Business components

14

## Componenti

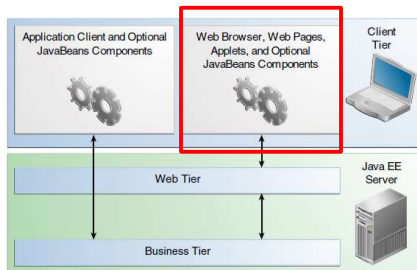
15

- Java EE runtime environment definisce differenti tipi di componenti:
  - ▣ Client components (Java EE Clients)
    - Web Clients
    - Application Clients
    - Applets
  - ▣ Web components
  - ▣ Business components

15

## Client components: Web Client (Applets)

15



### Web client (thin client)

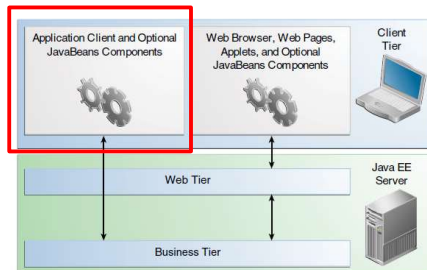
- Applicazioni eseguite su web browser
- Pagina dinamiche (HTML, XML, etc.)
- Possibile che contengano applet: piccola applicazione che viene eseguita sul browser del client
- Thin client
  - ▣ no query a database
  - ▣ nessuna logica di business complessa
  - ▣ no connessione a legacy applications

16



## Client components: Application Clients

16



### 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)

17

## Componenti

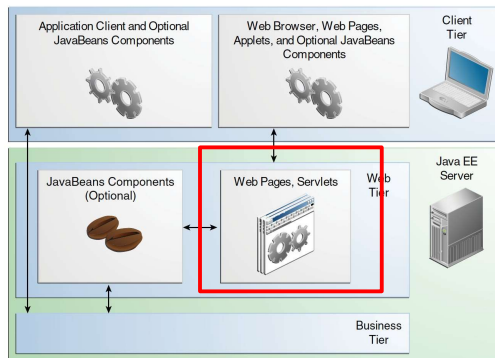
18

- ☐ Java EE runtime environment definisce differenti tipi di componenti:
  - ☐ Client components (Java EE Clients)
    - ☐ Web Clients
    - ☐ Application Clients
    - ☐ Applets
  - ☐ Web components
  - ☐ Business components

18

## Web components

17



### 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:** costituita sulla tecnologia delle servlet e JSP per fornire pagine dinamiche
- Può includere componenti JavaBeans

19

## Componenti

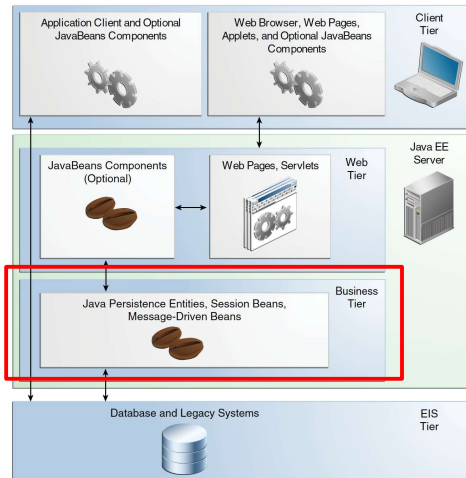
20

- Java EE runtime environment definisce differenti tipi di componenti:
  - ▣ Client components (Java EE Clients)
    - Web Clients
    - Application Clients
    - Applets
  - ▣ Web components
  - ▣ Business components

20

## Business components

18



### 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)

21

## Organizzazione della lezione

19

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

22

## Java EE Container

23

- 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

23

## Java EE Container

24

- 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

24

## Java EE Container

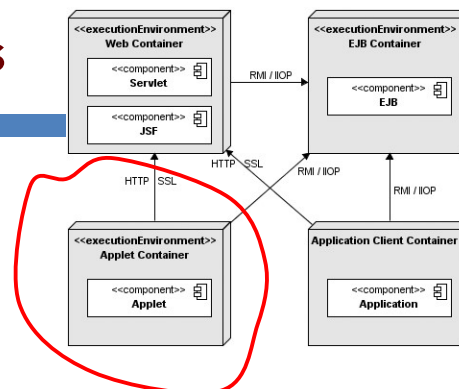
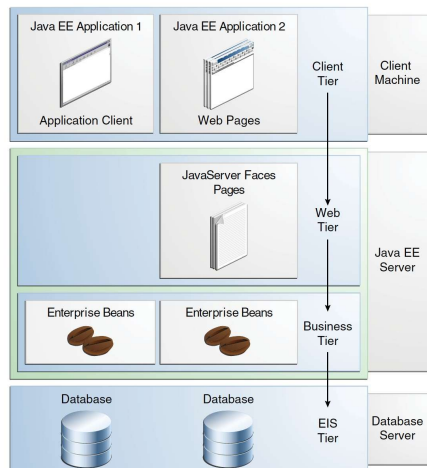
25

- Java EE ha 4 differenti container:
  - ▣ Applet containers
  - ▣ Application client container (ACC)
  - ▣ Web container
  - ▣ EJB container

25

## Applet containers

23

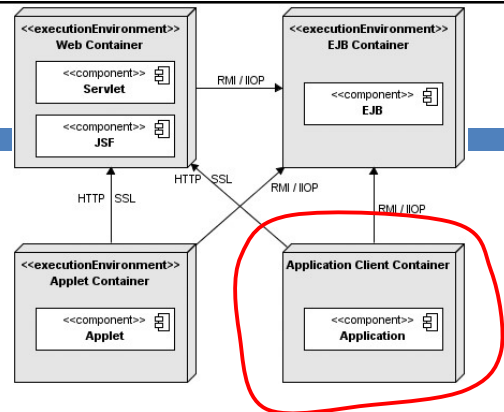
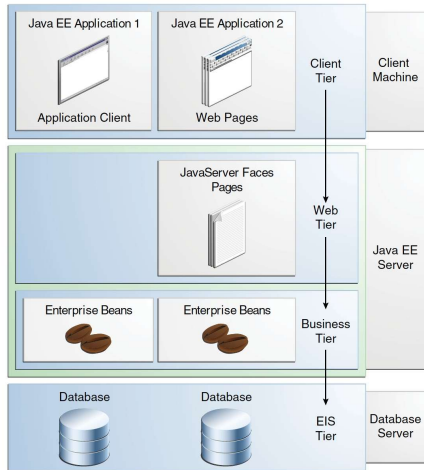


- Fornito dalla maggior parte dei browser per eseguire applet components
- Sicurezza fornita da sandbox (accesso limitato alla macchina client)
- Impedisce accesso al computer locale per accesso a processi o files

26

## Application Client Container (ACC)

25

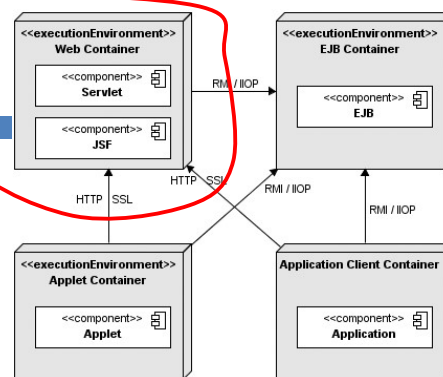
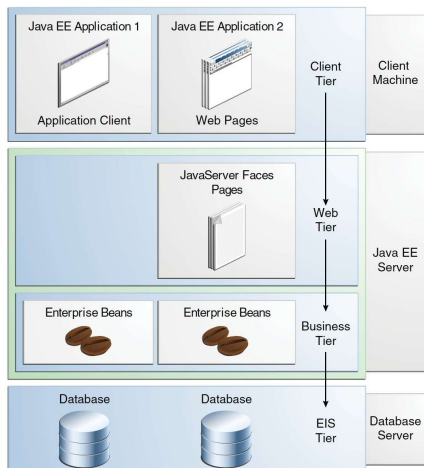


- Insieme di classi Java, librerie etc. che permettono di usare servizi Java EE in applicazioni SE
- Servizi quali sicurezza, naming etc. acceduti da una applicazione standard (con un main())
- L'ACC comunica con l'EJB container usando RMI-IIOP e con il Web container usando HTTP (Web services)

27

## Web Container

25

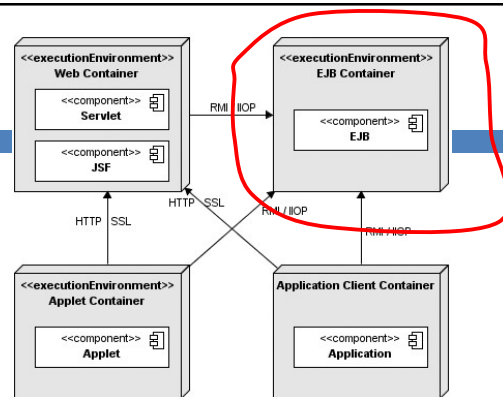
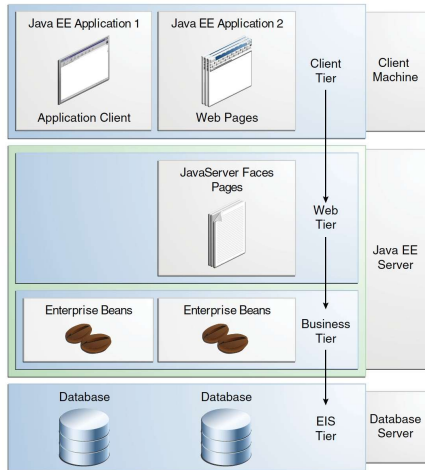


- Servizi per gestione e esecuzione di componenti web
- Ad esempio, servlet, JSPs, filtri, listeners, pagine JSF, web services
- Responsabile per inizializzazione, invocazione e gestione del ciclo di vita delle servlet

28

# EJB Container

26

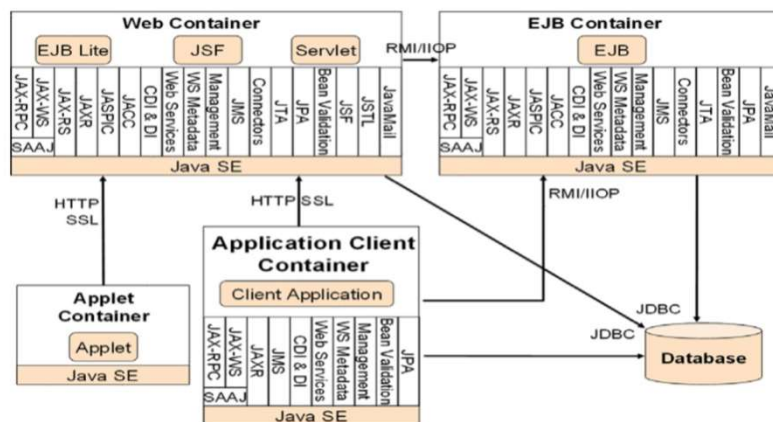


- ❑ Responsabile della gestione dei bean
- ❑ Business logic layer
- ❑ Gestisce il ciclo di vita degli EJB
- ❑ Fornisce transazioni, sicurezza, concorrenza, distribuzione, servizio di naming, invocazioni asincrone

29

## I servizi forniti dal container

27



30

## Alcuni servizi importanti

28

- Java Transaction API
- Java Persistence API: standard API per object-relational mapping (ORM). Con il linguaggio Java Persistence Query Language (JPQL) si possono fare query su oggetti
- Java Message Service: comunicazione asincrona tra le componenti. 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"*

31

## Organizzazione della lezione

29

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

32



## Packaging

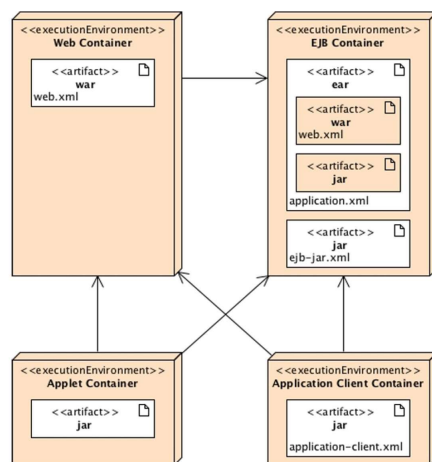
33

- 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

33

## Packaging: Application client module

34

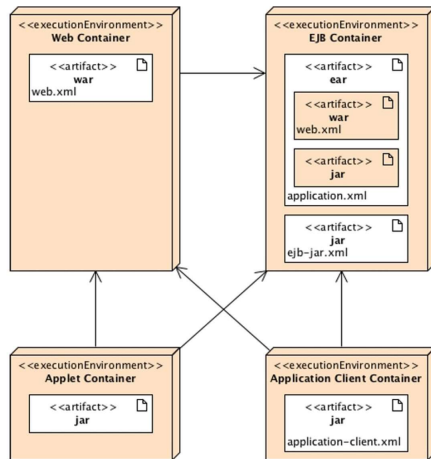


- Application client module: contiene classi Java e altre risorse packaged in a jar file
- Il jar può essere eseguito in un Java SE environment o in un application client container
- Il jar contiene la directory META-INF con meta information che descrivono l'archivio
  - ▣ MANIFEST.MF usato per definire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

34

## Packaging: EJB module

35

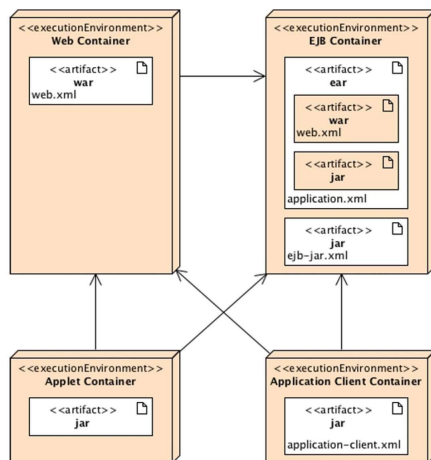


- EJB module: contiene uno o più session o Message-driven beans (MDBs) impacchettati in un jar file
- Deployment descriptor in META-INF/ejb-jar.xml

35

## Packaging: web application module

36

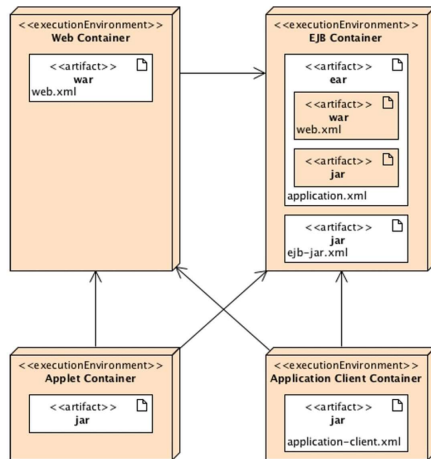


- Web application module: contiene servlet, JSP, JSF, HTML, CSS, JS, media, etc.
- Jar file con estensione .war
- Deployment descriptor in WEB-INF/web.xml
- Se il war contiene un EJB Lite beans il deployment descriptor settato in WEB-INF/ejb-jar
- Le classi in WEB-INF/classes e altri jar in WEB-INF/lib

36

## Packaging: enterprise module

37



- 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

37

## Organizzazione della lezione

35

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

38

## Annotazioni e Deployment Descriptor

39

- 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

39

## Annotazioni e Deployment Descriptor

40

- 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 aggiuntive 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

40

## Un EJB con annotazioni

Listing 1-1

```
@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);
    }
}
```

Senza stato da mantenere

La definizione remota

41

## Un EJB con annotazioni

Listing 1-1

```
@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);
    }
}
```

Senza stato da mantenere

La definizione remota

42

## Un EJB con annotazioni

Listing 1-1

```

@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);
    }
}

```

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

43

## Un EJB con annotazioni

Listing 1-1

```

@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);
    }
}

```

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

44

## Un deployment descriptor equivalente

```
<?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>
```

Il nome del bean

45

## Un deployment descriptor equivalente

```
<?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>
```

Il nome del bean

La classe remota

46

## Un deployment descriptor equivalente

```
<?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>
```

› Il nome del bean

› La classe remota

› ... e quella locale

47

## Un deployment descriptor equivalente

```
<?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>
```

› Il nome del bean

› La classe remota

› ... e quella locale

› Il tipo

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

48



## Differenza fra i due stili

49

- 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

49

## Deployment descriptor in Java EE

50

*Table I-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

50

## Organizzazione della lezione

48

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

51

## L'importanza degli standard

52

- 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.
- Oggi, le applicazioni possono essere deployate in un qualunque application server con pochi cambiamenti

52

## Java Community Process

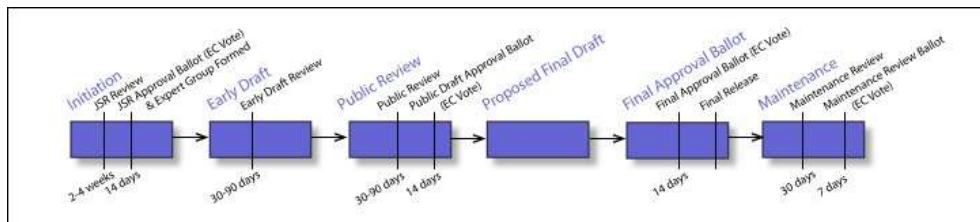
53

- 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 e un Compatibility Test Kit
- L'approvazione avviene da parte dell'Executive Committee di JCP

53

## Il processo di JCP

54



### Initiation

L'idea iniziale deve essere sottoposta al JCP da un Membro, come richiesta di modifica, aggiornamento o creazione di una nuova specifica (la JSR).

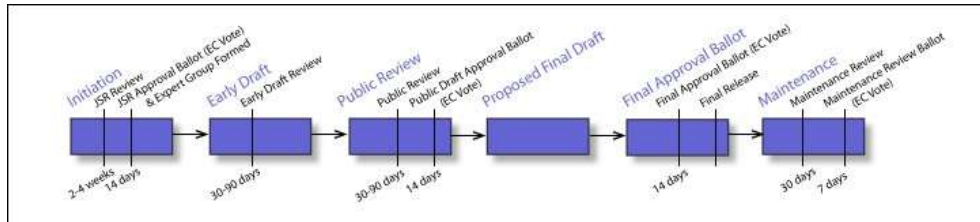
Nelle due settimane successive dalla richiesta, pubblico, Membri ed Esperti hanno la possibilità di rivederla e commentarla, fino a darle l'eventuale forma definitiva che sarà sottoposta, al termine di tale periodo, alla votazione del Comitato Esecutivo (EC).

Se la proposta passa positivamente la votazione, la richiesta va avanti nel suo iter e passa alla fase successiva.

54

## Il processo di JCP

55



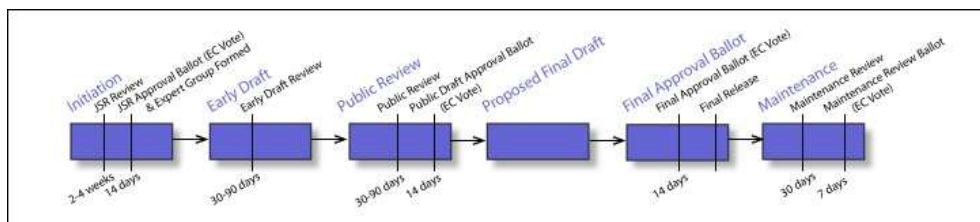
### Early Draft

Il primo passo da compiere è quello di fornire una prima bozza della futura JSR. Per fare ciò, nei 2-3 mesi successivi, il capo progetto, normalmente chi ha effettuato la richiesta, lavora per formare un gruppo di Esperti, che si preoccupano di scrivere una prima bozza della specifica. Al termine, la bozza viene presentata, e sottoposta alla revisione da parte di altri Esperti, Membri o del pubblico, per ricevere commenti ed eventualmente aggiornarne i contenuti.

55

## Il processo di JCP

56



### Complete the Specification, Public Draft/Final Release

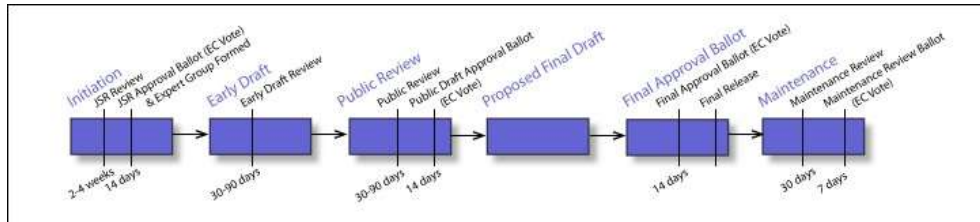
Quando viene stabilito che la prima bozza può andar bene, si procede con lo sviluppo dettagliato della JSR. Questo processo, che normalmente dura da uno a tre mesi, termina con la preparazione di una bozza pubblica, che viene sottoposta nuovamente al voto del Comitato Esecutivo, incaricato di vagliarne la correttezza complessiva.

Se la bozza passa positivamente la votazione, il gruppo di Esperti si deve occupare di proporre una bozza finale, mentre il capo progetto deve preparare un prototipo che esplicitamente faccia riferimento all'implementazione.

56

## Il processo di JCP

57



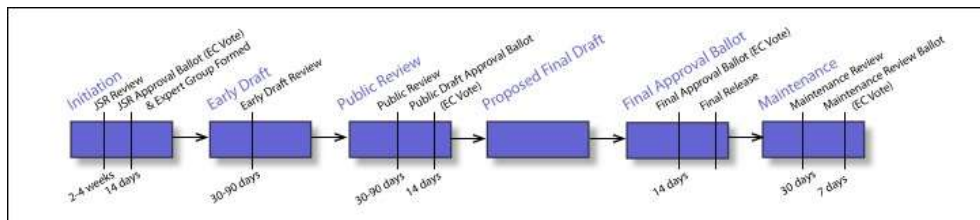
### Complete the Specification, Public Draft/Final Release

Sempre in questa fase, il capo progetto deve preoccuparsi di preparare il Kit di Compatibilità Tecnologica (TCK), ovvero quell'insieme di test, strumenti e documenti che mettano in condizione, chi volesse implementare la specifica, di comprendere se la propria implementazione è aderente alla JSR proposta. Al termine, tutta questa documentazione passa ancora al vaglio del Comitato Esecutivo, che può approvarne o meno la sua pubblicazione sotto forma di **"Versione Finale"**. In caso di votazione positiva, il gruppo di Esperti associato alla JSR si scioglie, e la proposta diventa standard a tutti gli effetti.

57

## Il processo di JCP

58



### Maintenance

Qualora il capo progetto reputi che siano necessarie alcune modifiche, possono essere distribuiti degli aggiornamenti di manutenzione. Nel mese seguente, Esperti, Membri e pubblico possono partecipare e contribuire alla stesura dell'aggiornamento, che infine viene passata alla votazione del Comitato Esecutivo: lo scopo è quello di approvare l'inclusione nella JSR come aggiornamento minore.

58

## Organizzazione della lezione

56

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

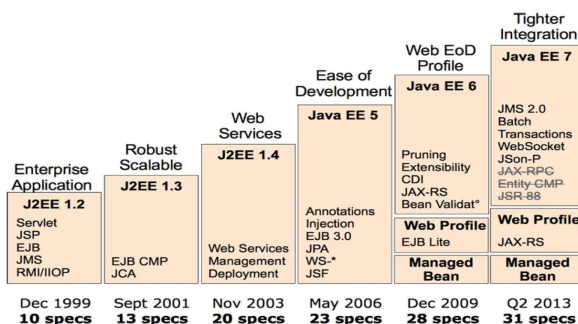
59

## L'evoluzione di Java EE

60



Java EE 8  
Modernization - Simplification



### Java EE 8

- 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)

2017

60

## L'evoluzione di Java EE

61



Java EE 8  
Modernization - Simplification

### Java EE 8

- 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)

2017

Java EE 8 builds on Java EE 7. The following JSRs are new or updated in Java EE 8:

- JSR 366 – Java EE 8 Platform
- JSR 365 – Contexts and Dependency Injection (CDI) 2.0
- JSR 367 – The Java API for JSON Binding (JSON-B) 1.0
- JSR 369 – Java Servlet 4.0
- JSR 370 – Java API for RESTful Web Services (JAX-RS) 2.1
- JSR 372 – JavaServer Faces (JSF) 2.3
- JSR 374 – Java API for JSON Processing (JSON-P) 1.1
- JSR 375 – Java EE Security API 1.0
- JSR 380 – Bean Validation 2.0
- JSR 250 – Common Annotations 1.3
- JSR 338 – Java Persistence 2.2
- JSR 356 – Java API for WebSocket 1.1
- JSR 919 – JavaMail 1.6

<https://www.oracle.com/it/java/technologies/java-ee-glance.html>

61

## Organizzazione della lezione

59

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

62

## Specifiche di Java EE 7

63

- 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

63

## Specifiche di Java EE 7

61

**Table 1-2.** Java Enterprise Edition Specification

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

**Table 1-3.** Web Services Specifications

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

64



## Specifiche di Java EE 7

62

*Table 1-4. Web Specifications*

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

65

## Specifiche di Java EE 7

63

*Table 1-5. Enterprise Specifications*

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

66

## Specifiche di Java EE 7

64

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

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

67

68



## Java EE 8

JSR-000366 Java™ Platform, Enterprise Edition 8 (Java EE 8) Specification  
(Final Release)

<https://jcp.org/aboutJava/communityprocess/final/jsr366/index.html>

<https://www.oracle.com/java/technologies/javaee-8-sdk-downloads.html>

68

## CD-BookStore

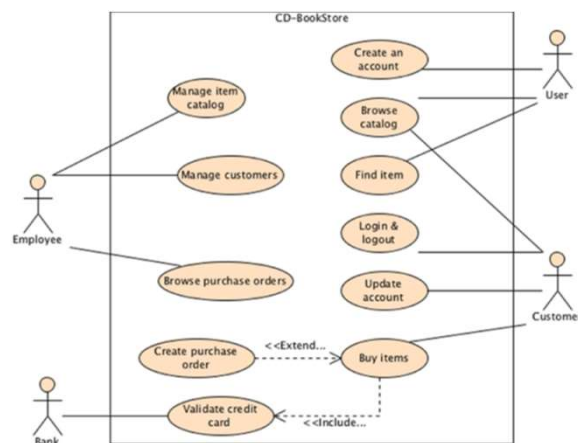
66

- 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

69

## CD-BookStore

70

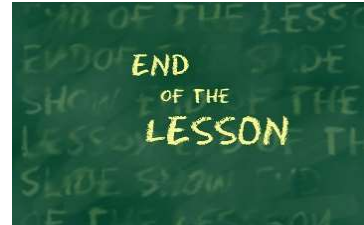


70

## Organizzazione della lezione

68

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



Nelle prossime lezioni:

Context and Dependency Injection