## Java EE 7 Platform Highlights

L’obiettivo più importante della piattaforma Java EE7 è di semplificare lo sviluppo offrendo basi comuni per i vari tipi di componenti della piattaforma Java EE.

Gli sviluppatori beneficiano di questi miglioramenti alla produttività usando più annotazioni e meno XML, più POJO e packaging semplificati. Java EE7 include le seguenti novità:

Nuove tecnologie quali:

* Batch Applications per la piattaforma Java
* Concurrency Utilities
* Java API per processare JSON (JSON-P)
* Java API per WebSocket

Nuove caratteristiche per i componenti EJB

Nuove caratteristiche per i servlet.

Nuove caratteristiche per i componenti JavaServer Faces

Nuove caratteristiche per il Java Message Service (JMS)

## Java EE application model

Il Java EE application model è formato dal linguaggio di programmazione Java e dalla Java Virtual Machine.

La portabilità, sicurezza e l’aumento della produttività dello sviluppatore che fornisce creano le basi per l’application model. Java EE è costruito per supportare applicazioni che implementano servizi enterprise per clienti, dipendenti, partners di aziende.

Questo tipo di applicazioni sono complesse per natura , fornendo accesso ai dati da un varietà di fonti e distribuire le applicazioni a una varietà di clienti.

Per controllare e gestire al meglio queste applicazioni le funzioni business che supportano i vari utenti sono condotte nel **middle tier**.

Il middle tier rappresenta un ambiente che è strettamente controllato da un reparto di enterprise information tecnology e generalmente funziona su un server dedicato e ha accesso a tutti i servizi dell’enterprise.

Il Java EE application model definisce un’architettura per implementare servizi come le applicazioni **multitier (a strato)** che garantiscono scalabilità, accessibilità e governabilità indispensabili alle applicazioni enterprise level.

Questo modello divide il lavoro necessario a implementare un servizio multitier nelle seguenti parti:

* La **logica business e di presentazione** che deve implementare lo sviluppatore
* I **sistemi di servizio standard** offerti dalla piattaforma Java EE

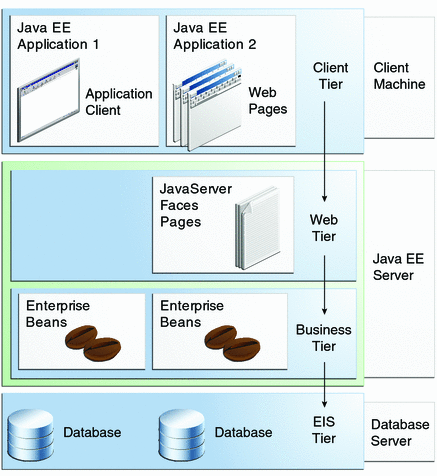
Lo sviluppatore può fare affidamento sulla piattaforma per gestire i problemi di sistema che si presentano sviluppando un servizio multitier.

## Distributed Multitiered Applications

La piattaforma Java EE usa il **distributed multitiered application model** per le applicazioni enterprise.

La logica dell’applicazione è divisa in componenti in base alla loro funzine e i componenti che creano una applicazione Java EE sono installati su varie macchine a seconda dello strato, all’interno dell’ambiente multistrato di Java EE, a cui appartengono.

L’immagine sotto mostra gli strati di un’applicazione:



* i componenti **client tier** vanno sulla macchina del client.
* I componenti Web-tier vanno sul server Java EE
* I componenti Business-tier vanno sul server Java EE
* I software Enterprise information system(EIS) vanno sull’EIS Server.

Anche se un’applicazione Java EE può consistere di tutti e 4 i tier mostrati in figura, le applicazioni Java EE multitiered generalmente sono considerate applicazioni a tre tier perché sono distribuite in tre posti: il client, il server Java EE e il database nel back end.

Le applicazioni a tre tier estendono lo standard di quelle a due tier (client/server) andando a inserire un server multithread per l’applicazione tra il client e il back end.

## Sicurezza

Mente altri modelli di applicazioni enterprise necessitano misure di sicurezza specifiche per ogni applicazione, l’ambiente di sicurezza di Java EE permette di definire i vincoli di sicurezza in fase di sviluppo e le rende capaci di una grande varietà di implementazioni di sicurezza evitando allo sviluppatore di doverle implementare lui stesso.

La piattaforma Java EE offre regole dichiarative di controllo di accesso che sono definite dal programmatore e interpretate quando l’applicazione viene messa sul server.

Inoltre Java EE offre meccanismi logici standard tali che lo sviluppatore non debbano implementare questi meccanismi all’interno delle loro applicazioni.

La stessa applicazione funzionerà in molti ambienti di sicurezza diversi senza cambiare il codice.

## Java EE Components

Le applicazioni Java EE sono formate da componenti.

Un componente è un’inutà software funzionale indipendente assemplata dentro l’applicazione Java EE con le sue classi e files e che comunica con altri componenti.

La specifica di Java EE definisce i seguenti **componenti**:

* Application clients e appletssono componenti **che funzionano sul client.**
* Java Servlet, JavaServer Faces e JavaServer Pages (JSP) sono componenti web e **funzionano sul server**.
* I componenti EJB (Enterprise beans) cono componenti business che **funzionano sul server**

I componenti Java EE sono scritti nel linguaggio di programmazione Java e scritti nello stesso modo di ogni altro programma scritto in Java.

La differenza tra i componenti Java EE e le classi standard di Java è che i componenti EE sono assemblati in un’applicazione Java EE, subiscono una verifica per sapere se sono ben formati e in linea con le specifiche Java EE e vengono posti nella produzione, dove funzionano e vengono gestiti dal server Java EE.

## Java EE Clients

Un client Java EE è solitamente un web client o un’application client.

## Web clients

Un web client consiste di due parti:

* + Le **pagine web dinamiche** checontengono vari tipi di linguaggi di mark-up (HTML, XML, etc) che sono generati dai componenti web che vanno sul web tier.
  + Un **web browser** che interpreta le pagine ricevute dal server

Il web client è chiamato anche **thin client.** I thin client solitamente non fanno query su database, né eseguono complesse regole di business né connettono con applicazioni.

Quando si usa un thin client queste operazioni vengono lasciate agli enterprise beans eseguiti sul server Java EE che possono alzare il livello di sicurezza, velocità, riutilizzabilità e servizio delle tecnologie lato server di Java EE.

## Application clients

Un application client funziona sulla macchina del client e offre un modo all’utente per gestire compiti che richiedono una ricca interfaccia utente che può essere offerta tramite il linguaggio di mark-up.

Un’application client ha solitamente un’interfaccia grafica per l’utente (GUI) creata dalla Swing aPI o dall’Abstract Window Toolkit (AWT) API, ma si può utilizzare anche la linea di comando.

Le application client hanno accesso diretto agli enterprise beans in esecuzione nel business tier.

Comunque, se un’applicazione lo richiede, può aprire una connessione http per stabilire una comunicazione con una servlet in esecuzione nel web tier.

Le application client scritte in altri linguaggi possono interagire con i server Java EE, abilitare la piattaforma Java EE a operare con sistemi dipendenti, clients e linguaggi non Java.

## 1.3.3.3 Applets

Una pagina web ricevuta dal web tier può includere un’applet incorporata.

Scritta in linguaggio Java, un’applet è una piccola applicazione client che viene eseguita nella Java Virtual Machine installata nel browser web.

Spesso i sistemi client necessitano del plug-in Java e di un file per le security policy per eseguirlo con successo nel web browser.

I componenti web sono le API migliori per creare programmi web client perché non necessitano di alcun plug-in né file per la sicurezza per funzionare, inoltre i componenti web rendono più pulito e modulare il design dell’applicazione perché forniscono un modo per separare la programmazione dell’applicazione dal design della pagina web.

## Architettura dei componenti Java Bean

I tier client e server possono inoltre includere componenti basati sull’architettura dei componenti JavaBeans per gestire il flusso di dati tra:

* Un client o un’applet e i componenti in esecuzione sul server Java EE
* I componenti del server e il database

I componenti JavaBeans non sono considerati componenti Java EE dalla specifica.

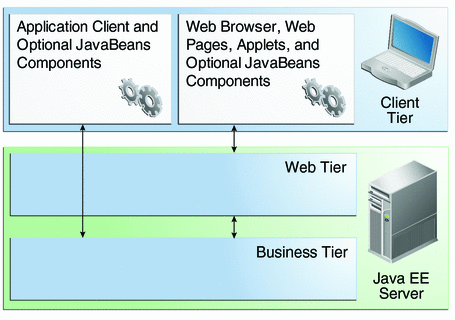
I componenti JavaBeans hanno proprietà i metodi getter e setter per accedere a tali proprietà.

I JavaBeans usati in questi modo sono semplici nel design e nell’implementazione ma devono essere conformi alle convenzioni di naming e design dell’architettura JavaBeans.

## Java EE server communications

La figura di cui sotto mostra i vari elementi che possono comporre il client tier.

Il client comunica con il business tier in esecuzione sul server Java EE direttamente o, come nel caso di un client che funziona su un browser, attraverso pagine web o servlets che vanno nel web tier.



## Componenti Web

I componenti web di Java EE sono servlets o pagine web create usando la tecnologia JavaServer Faces e/o JSP.

Le **servlets** sono classi scritte in Java che processano **dinamicamente** le richieste e forniscono le risposte.

Le pagine **JSP** sono documenti testuali che funzionano come le servlets ma forniscono un approccio migliore per creare **contenuti** **statici**.

La tecnologia **JavaServer Faces** è basata sulla tecnologia delle servlet e dei JSP e offre un framework con **interfaccia utente** per le applicazioni web.

Le pagine HTML statiche e le applets sono unite ai componenti web durante l’assemblaggio dell’applicazione ma non sono considerati parte dei componenti web dalle specifiche di Java EE.

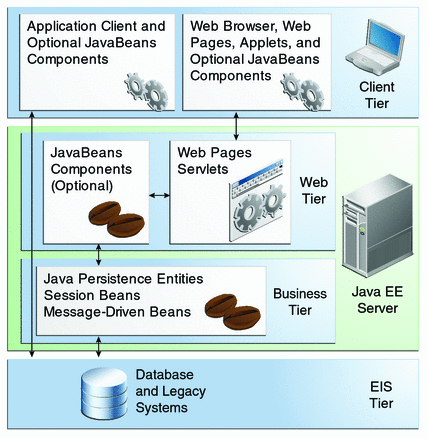
Anche le classi con utilità lato server sono unite con i componenti web ma, come HTML, non fanno parte dei componenti web.

## Componenti business

Il codice business, che è la logica che risolve e incontra le necessità di un particolare dominio business come ad esempio banking, retail o finance è gestito dagli Enterprise Beans eseguiti nel business o nel web tier.

La figura sotto mostra come un Enterprice Bean riceve i dati dai programmi client, li processa (se necessario) e li invia al sistema di Enterprise Information per archiviarli.

Un Enterprise Bean recupera inoltre i dati dall’archivio, li processa (se neacessario) e li restituisce al programma client.



## Enterprise Information System Tier

L’Enterprise Information System Tier gestisce il software EIS e include sistemi a infrastruttura enterprise, come un enterprise resource planning (ERP), mainframe di processamento delle transazioni, sistemi database e altri sistemi che ereditano informazioni.

Ad esempio i componenti dell’applicazione Java EE possono necessitare di accedere ai sistemi di enterprise information per la connettività col database.

## Java EE Containers

Normalmente le applicazioni thin-client multitiered sono difficili da scrivere perché hanno molte linee di codice intricato per gestire le transazioni, il multithread, la mole di risorse e altri complessi dettagli di basso livello.

L’architettura basata su componenti e l’indipendenza dalla piattaforma di Java EE rende le applicazioni semplici da scrivere perché la logica business è organizzata in componenti riutilizzabili.

In più il server Java EE offre servizi per ogni tipo di componente sotto forma di **conteiners**.

Visto che non necessita che lo sviluppatore implementi questi servizi da solo, è più semplice concentrarsi sul risolvimento dei problemi di business.

## 1.4.1 Container Services

I containers sono interfacce tra un componente e le funzionalità specifiche a basso livello della piattaforma che supporta il componente.

Prima di essere eseguito un componente web, enterprise bean o application client deve essere assemblato dentro un modulo Java EE e posto nel suo container.

Il processo di assemblaggio specifica i settaggi del container per ogni componente nell’applicazione Java EE e per l’applicazione stessa.

Il setting del container personalizza il supporto offerto in background dal server Java EE, incluso inclusi i servizi di sicurezza, gestione delle transazioni, Java Naming and Directory Interface (JNDI) e connettività remota.

* Il **Java EE security model** da modo di configurare un componente web o un enterpreis bean in modo che le risorse di sistema siano accessibili solo dagli utenti autorizzati.
* Il **Java EE transation model** da modo di specificare le relazioni tra metodi che costituiscono una singola transazione in modo che tutti i metodi in una transazione siano trattati come una singola unità
* **JNDI** offre un’interfaccia unificata per servizi di naming e directory multipli nell’enterprise in modo che i componenti dell’applicazione possano avere accesso a tali servizi
* **Il Java EE remote connectivity model** gestisce le comunicazioni a basso livello tra clients e enterprise beans. Dopo la creazione di un enterprise bean il client invoca i metodi sul bean come se fossero sulla stessa virtual machine.

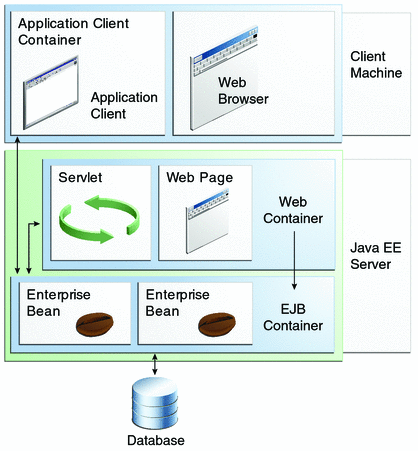
Dato che l’architettura Java EE offre servizi configurabili, i componenti all’interno della stessa applicazione possono comportarsi diversamente in base a dove sono posti.

Ad esempio un enterprise bean può avere settaggi di security che gli permettano un certo livello di accesso ai dati del database in un ambiente di produzione e un altro livello di protezione in un altro ambiente.

Il container gestisce anche dei servizi non configurabili, come i cicli di vita degli enterpreis beans delle servlet, il modo di accumulare le risorse della connessione al database, la persistenza dei dati e l’accesso alle API della piattaforma Java EE.

## 1.4.2 Tipi di container

Il processo di sviluppo installa i componenti delle applicazioni Java EE nei containers come illustrato nella figura seguente:



I server e containers sono i seguenti:

* **Server Java EE:** La porzione runtime di un prodotto Java EE. Offre EJB e web containers.
* **EJB Container:** Gestisce l’esecuzione degli enterprise beans per le applicazioni Java EE. Gli enterpreis beans e i loro container sono eseguiti sul server Java EE.
* **Web Container:** Gestisce l’esecuzione delle pagine web, delle servlets e di alcuni componenti EJB per l’applicazione Java EE. I componenti web e i loro container sono eseguiti sul server Java EE.
* **Application client container:** Gestisce l’esecuzione dei componenti application client. Le application client e i loro cointainer sono eseguiti sul client.
* **Applet container:** Gestisce l’esecuzione delle applets. Consiste di un browser e di un Java plug-in che vengono eseguiti insieme sul client.

## Web Services Support

I web services sono applicazioni basate sul web che usano standard e protocolli di trasporto XML per scambiare dati con i client. La piattaforma Java EE offre le XML API’s e strumenti atti a velocizzare il design, lo sviluppo, il test el’allocazione dei servizi web e clients che cooperano con altri servizi web e clients eseguiti su altre piattaforme indipendentemente se siano basate o no su Java.

Tutto ciò che serve per scrivere web services e clients con le Java EE XML API’s è passare i dati dei parametri ai metodi e processare i dati di ritorno.

Per servizi web orientati ai documenti si mandano documenti contenenti i dati avanti e indietro.

Grazie alle XML API’s non c’è bisogno di nessuna programmazione a basso livello in quanto implementando le API’s queste svolgono il lavoro di tradurre i dati dell’applicazione da e per un stream con dati XML che è inviato tramite i protocolli standard di trasporto di XML che andremo a vedere a breve.

La traduzione dei dati dallo stream XML è ciò che rende i servizi web e i clients scritti con leAPI Java EE XML pienamente interoperanti con altre piattaforme.

Ciò non signifca necessariamente che i dati trasportati debbano includere tag XML, perché i dati trasportati possono essere anche testuali, dati XML o ogni tipo di dato binario come audio, video, mappe, file di programmi, documenti CAD, etc.

Andiamo ora a vedere XML e come usarlo per scambiare i dati in maniera più significativa.

## 1.5.1 XML

L’Extensible Markup Language (XML) è uno standard multi piattaforma, estensibile e testuale di rappresentazione dati.

Le parti che si scambiano i dati XML possono creare i propri tag per descriverli, creare uno schema che specifichi quali tag sian consentiti in un particolare documento XML e usare l’ XML stylesheet per gestire la rappresentazione e la gestione dei dati.

Ad esempio un servizio web può usare XML e uno schema che produce una lista di prezzi e la compagnia che riceve la lista di prezzi e lo schema può usare i propri stylesheets per gestire i dati in una visualizzazione che risulti più pratica. Esempio:

* Una società può inserire le informazioni sui prezzi XML in un programma che traduce l’XML in HTML in modo da poter postare la lista sul proprio intranet.
* Una società partner potrebbe mettere le informazioni XML in uno strumento per creare una rappresentazione di marketing.
* Un’altra società potrebbe leggere le informazioni dell’XML in un’applicazione per processarli.

## 1.5.2 Protocollo di trasporto SOAP

Le richieste del cliente e le risposte del web service sono trasmesse come messaggi SOAP (Simple Object Access Protocol) via HTTP per abilitare una completa interoperabilità di scambi tra i clients e i web services eseguiti su diverse piattaforme e in vari siti su internet.

HTTP è uno standard di richiesta-risposta per inviare messaggi sul web e SOAP è un protocollo basato su XML che segue il modello richiesta-risposta di HTTP.

La porzione SOAP di un messaggio trasportato fa ciò che segue:

* Crea una “busta” che descrive il contenuto del messaggio e come processarlo
* Include delle regole di codifica XML per esprimere le istanze dei tipi di dati definiti nell’applicazione, dentro il messaggio
* Definisce un convenzione XML per rappresentare le richieste e i servizi remoti e le rispettive risposte.

## Formato Standard WSDL

Il WSDL (Web Services Description Language) è un formato standard XML per descrivere servizi di rete.

La descrizione include il nome del servizio, l’allocazione del servizio e i modi per comunicare col servizio stesso.

La WSDL service description può essere pubblicato nel web.

GlassFish Server offre uno strumento per generare le specifiche WSDL per un servizio web che utilizza procedure di chiamata remota per comunicare coi clients.

## Assemblaggio e distribuzione di applicazioni Java EE

Un’applicazione Java EE è impacchettata in una o più unità standard per distribuirla a un qualunque sistema di compilazione della piattaforma Java EE. Ogni unita consiste di:

* Uno o più componenti funzionali (enterpreis beans, pagine web, servlet, applet).
* Un descrittore di distribuzione opzionale che descrive il contenuto.

Una volta prodotta, l’unità Java EE è pronta per essere distribuita. La distribuzione usualmente prevede l’utilizzo di uno strumento di distribuzione della piattaforma per specificare le informazioni specifiche di allocamento, come il nome del database locale e una lista di utenti che possono accedervi.

Una volta distribuita su una piattaforma locale l’applicazione è pronta all’uso.

## Java EE 7 API’s



La figura mostra le relazioni tra i containers Java EE



La figura mostra la disponibilità delle Java API nel web container



La figura mostra la disponibilità delle Java EE7 API’s nell’EJB container.



La figura mostra la disponibilità delle API Java EE7 nel conteiner dell’application client.

I capitoli seguenti danno una spiegazione sommaria delle tecnologie richieste dalla piattaforma Java EE e delle API’s usate nelle applicazioni Java EE.

## 1.7.1 Tecnologia Enterprise JavaBean

Un componente Enterprise JavaBean è un blocco di codice che ha campi e metodi che implementano moduli di logica business.

Si può immaginare un enterprise bean come un blocco di costruzione che può essere usato da solo o con altri enterprise beans per eseguire la logica business sul server EE.

Gli enterprise beans possono essere session beans o message-driven beans.

* Una session bean rappresenta un conversazione transitoria col client. Quando il client finisce l’esecuzione il session bean e i dati che contiene vengono persi.
* Un message-driven bean combina le caratteristiche del sission bean e di un message listener, permette ai componenti business di ricevere messaggi asincronamente. Solitamente questi messaggi sono JSM (Java Message Service).

Nella piattaforma Java EE7 le nuove funzioni degli enterprise beans includono:

* Session beans asincroni in EJB Lite
* Timers nonpersistent in EJB Lite

La piattaforma Java EE7 richiede JavaBeans 3.2 e Interceptors 1.2.

La specifica di Interceptors è parte della specifica di EJB.

## 1.7.2 Tecnologia Java Servlet

La tecnologia Java Servlet permette di definire specifiche classi servlet HTTP. Una classe servlet estende le funzionalità dei server che ospitano l’applicazione e a cui si accede tramite un modello di programmazione richiesta-risposta comunemente usato.

Dato che le servlet possono rispondere a ogni tipo di richiesta, sono comunemente usate per estendere le applicazioni ospitate dai server web.

Nella piattaforma Java EE7 le nuove funzionalità delle servlet includono:

* Nonblocking I/O
* Upgrade del protocollo HTTP

## Tecnologia JavaServer Faces

La tecnologia JavaServer Faces è un framework ad interfaccia usato per costruire applicazioni web.

I componenti principali di questa tecnologia sono:

* Un framework GUI
* Modelli flessibili per renderizzare componenti di diversi tipi di HTML o differenti linguaggi di mark-up.
* Un oggetto renderizzato genera il mark-up per renderizzare i componenti e convertire i dati conservati in un oggetto model in tipi che possono essere rappresentati in una view.
* Renderkit standard per generare mark-up HTML 4.01

Le seguenti funzionalità supportanto i componenti GUI:

* Validazione input
* Gestione degli eventi
* Conversione dati tra componenti e oggetti modello
* Gestione della creazione di oggetti modello
* Configurazione della navigazione della pagina
* Expression Language (EL)

Tutte queste funzionalità sono disponibili usando API Java standard e files di configurazione XML.

Nella piattaforma Java EE7 le nuove funzionalità disponibili per la tecnologia JavaServer Faces includono:

* Mark-up compatibile con HTML5
* Faces Flows
* Contratti con librerie di risorse (resource library contracts)

La piattaforma Java EE7 richiede JavaServer Faces 2.2 e Expression Language 3.0.

## Tecnologia JavaServer Pages

La tecnologia Java Server Pages (JSP) permette di inserire pezzetti del codice della servlet direttamente all’interno di un documento testuale. Una pagina JSP è un documento testuale che contiene due tipi di dati:

* Dati statici che possono essere espressi in ogni formato testuale, come HTML o XML.
* Elementi JSP che determinano come la pagina costruisce i contenuti dinamici

La piattaforma Java EE7 richiede JavaServer Pages 2.3 per la retrocompatibilità con le più vecchie versioni ma raccomanda l’uso di Facelets per le nuove applicazioni.

## JavaServer Pages Standard Tag Library

Il JavaServer Pages Standard Tag Library (JSTL) incorpora le principali funzionalità comuni a molte applicazioni JSP. Invece di mixare tag di vari tipi nelle applicazioni JSP si utilizza una libreria standard di tag.

Questa standardizzazione permette di sviluppare l’applicazione su ogni contenitore JSP che supporti JSTL e rende più semplice l’implementazione delle tag.

JSTL ha tag per iterazioni e condizionali per gestire il flusso, altre per manipolare i documenti XML, tag internazionali, tag per accedere ai database tramite SQL e tag per funzioni comunemente utilizzate.

La piattaforma Java EE7 richiede JSTL 1.2.

## Java Persistence API

La Java Persistence Api (JPA) p una soluzione basata su Java per gestire la persistenza.

La persistenza utilizza un approccio di mapping oggetto/relazionale per collegare un modello orientato agli oggetti a un database relazionale.

La Java Persistence API può inoltre essere usata in applicazioni Java SE all’esterno dell’ambiente EE.

La Java persistence è formata dalle seguenti aree:

* Java Persistence API (JPA)
* Il linguaggio per le query
* Matadata del mapping oggetto/relazionale

La piattaforma Java EE7 richiede la Java Persistence API 2.1.

## Java Transaction API

La Java Transaction API (JTA) offre un’interfaccia standard per demarcare le transazioni.

L’architettura Java EE offre un auto commit di default per gestire i commit (modifica) e i rollback (ripristino) della transazione.

Commit automatico significa che ogni altra applicazione che stia vedendo i dati vedrà i dati aggiornati dopo ogni operazione di scrittura/lettura su database.

Se l’applicazione performa due diverse operazioni di accesso al database che dipendono l’una dall’altra si potrebbe voler usare la API JTA per demarcare dove l’intera transazione, incluse le due operazioni, iniziano, fanno il rollback e il commit.

La piattaforma Java EE7 richiede Java Transaction API 1.2.

## Java API for RESTful Web Service

La Java API for RESTful Web Service (JAX-RS) definisce le API per lo sviluppo di un servizio web costruito secondo lo sile architetturale REST (Representation Stat Tranfer).

Un’applicazione JAX-RS è un’applicazione web che consiste di classi impacchettate come servlet in un file WAR con le librerie richieste.

La piattaforma Java EE7 necessita di JAX-RS 2.0.

## Managed Beans

I managed beans, oggetti gestiti da container leggeri (POJO’s), hanno requisiti minimali, supportano un piccolo set di servizi base come l’iniezione di risorse, il callback del ciclo di vita e gli interceptors, e rappresentano una generalizzazione dei managed beans specificati dalla tecnologia JavaServer Faces che possono essere usati ovunque in un’applicazione Java EE, non solo nei moduli web.

I managed beans fanno parte della specifica e la piattaforma Java EE7 necessita di Managed Beans 1.0.

## Contesti e iniezione di dipendenze in Java EE

Context and Dependency Injection (CDI) definisce un set di servizi contestuali, offerti dai containers Java EE, che rendono semplice utilizzare gli enterprise beans unitamente alle JavaServer Faces nelle applicazioni web.

Costruito per essere utilizzato con oggetti dotati di stato, CDI ha anche molte utilità di contorno, permette agli sviluppatori grande flessibilità nell’integrazione di differenti tipi di componenti in un modo approssimativo ma indipendente dal tipo e quindi sicuro.

La piattaforma Java EE7 necessita di CDI 1.1.

## 1.7.11 Iniezione di dipendenze in Java

L’iniezione di dipendenze in Java usa un set di annotazioni standard (e un’interfaccia) per utilizzare classi iniettabili.

Nella piattaforma Java, il supporto per l’iniezione delle dipendenze è implementato dal CDI, per cui è possibile utilizzare le iniezioni solo in un’applicazione che abbia il CDI abilitato.

La piattaforma Jave EE7 richiede Dependency Injection for Java 1.0.

## Bean Validation

Specificare il Bean Validation permette di definire il modello di metadata e le API per validare i dati dei componenti JavaBean.

Invece di distribuire la validazione dei dati sui diversi livelli, come il browser e il lato server, si possono definire le regole di validazione in un posto e condividerle tra i diversi livelli dell’applicazione.

La piattaforma Java EE7 necessita di Bean Validation 1.1.

## Java Message Service API

Il Java Message Service API (JSM) è uno standard di messaggi che permette ai componenti dell’applicazione Java EE7 di creare, spedire, ricevere e leggere messaggi.

Le nuove funzionalità del JSM includono:

* Una nuova API semplificata che offre una più semplice alternativa all’Api precedente. Questa API include un oggetto JMSContext che combina le fonzuini di una Connection e quelle di una Session.
* Tutti gli oggetti con un metodo close implementano l’interfaccia Autocloseable.

La piattaforma Java EE7 necessita di JMS 2.0.

## Architettura del Java EE Connector

L’architettura Java EE Connector è usata per creare adattatori per risorse che supportano l’accesso al sistema di enterprise information che possono essere aggiunte a ogni prodotto Java EE.

Un resource adapter è un softwareche permette ai componenti dell’applicazione Java EE di accedere e interagire con il resourge manager dell’EIS. Dato che il resource adaptor è specifico per il suo resource manager, esiste un adattatore diverso per ogni tipo di database o information system.

L’architettura del Java EE Connector offre anche un’integrazione transazionale orientata alle performance, sicura, scalabile e basata sui messaggi per i servizi web basati sulla piattaforma Java EE con EIS esistenti che possono essere sia sincroni che asincroni.

Le applicazioni esistenti e gli EIS integrati tramite l’architettura dei Java Connector nella piattaforma Java EE possono essere esposti come servizi web basati su XML usando JAX-WS e altre tecnologie complementari per l’integrazione di applicazioni enterprise (EAI) e integrazioni business end-to-end.

La piattaforma Java EE7 necessita della Java EE Connector Architecture 1.7.

## JavaMail API

Le applicazioni Java EE usano l’API JavaMail per spedire notifiche tramite e-mail.

La JavaMail API ha due parti:

* Un’interfaccia a livello dell’applicazione usata dai componenti dell’applicazione per spedire le mail.
* Un’interfaccia che fornisce il servizio

La piattaforma Java EE include l’API JavaMail con il service provider (l’interfaccia).

La piattaforma Java EE7 richiede JavaMail 1.5.

## Java Authorization Contract for Containers

La Java Authorization Contract for Containers (JACC) definisce un contratto tra il server dell’applicazione Java EE e unpolicy provider autorizzato. Tutti i containers Java EE supportano questo contratto.

La specifica di JACC definisce le classi java.security.Permission che soddisfano il modello di autorizzazione di Java EE.

La specifica definisce i vincoli del container nell’operare su istanze delle classi che lo permettono.

Definisce la semantica dei policy providers che usano le nuove classi che lo permettono per accedere ai requisiti per l’autorizzazione sulla piattaforma Java EE, inclusa la definizione e l’uso dei ruoli.

La piattaforma Java EE7 necessita di JACC 1.5.

## Java Authentication Service Provider Interface for Containers

La specifica del Java Authentication Service Provider Interface for Containers (JASPIC) definisce un’interfaccia service provider (SPI) presso cui i providers di autenticazione che implementano meccanismi di autenticazione per messaggi possono essere integrati in container di processamento messaggi client o server o runtimes.

Gli authentication providers integrati con questa interfaccia operano su messaggi network forniti dai containers chiamati.

Il provider di autenticazione trasforma i messaggi in uscita in modo che la fonte (source) di ogni messaggio possa essere autenticato dal mittente.

Inoltre il provider di autenticazione autentica ogni messaggio in entrata e rimanda ai loro containers il risultato dell’autenticazione del messaggio.

La piattaforma Java EE7 necessita di JASPIC 1.1.

## Java API per WebSocket

WebSocket è un protocollo di applicazione che fornisce comunicazioni interamente duplici tra due peers sul TCP. La Java API WebSocket permette alla piattaforma Java EE di creare degli endpoints usando le annotazioni che specificano i parametri di configurazione dell’endpoint e designano i metodi di callback del suo ciclo di vita.

L’API WebSocket è nuova e pertanto la piattaforma Java EE necessita della Java API for WebSocket 1.0.

## Java API for JSON Processing

JSON è un formato testuale per lo scambio di dati derivato da JavaScript usato nei servizi web e in altre applicazioni connesse.

La Java API for JSON Processing (JSOP-P) permette alle applicazioni Java di parsare, trasformare e creare query di dati JSON usando l’object model o lo streaming model.

JSON-P è nuovo per la piattaforma Java EE7 che necessita di JSON-P 1.0.

## Concurrency Utilities for Java EE

Concurrency Utilities for Java EE è un’API standard che mette a disposizione proprietà asincrone per i componenti di un’applicazione Java EE attraverso i seguenti tipi di oggetti: servizio di gestione dell’esecuzione, servizio di gestione dell’esecuzione programmato, gestione del thread factory e context service.

La piattaforma Java EE necessita di Concurrency Utilities for Java EE 1.0.

## Batch Applications per la piattaforma Java

I batch sono compiti che possono essere eseguiti senza interazione con l’utente.

Il Batch Applications for the Java Platform è un batch framework che offre supporto per creare e far funzionare questo tipo di operazoni in un’applicazione Java EE.

Il batch framework consiste di un batch runtime e un’API che implemente gli oggetti batch.

La piattaforma Java EE necessita del Batch Applications for the Java Platform 1.0.

## Le API Java EE7 nella piattaforma Java Standard Edition 7

Molte API necessarie per la piattaforma Java EE7 sono incluse nel piattaforma Java SE7 e sono disponibili anche per le applicazioni EE.

## 1.8.1 Java Database Coonectivity API

L’API Java Database Connectivity (JDBC) permette di invocare comandi SQL dai metodi definiti nel linguaggio di programmazione Java.

L’API JDBC si usa in un enterprise bean quando si vuole avere una session bean che acceda al database.

Si può inoltre usare l’API JDBC da una servlet o una pagina JSP per accedere direttamente al database senza passare da un enterprise bean.

L’API JDBC è formata da due parti:

* Un’interfaccia a livello dell’applicazione usata dai componenti dell’applicazione per accedere al database.
* Un’interfaccia service provider per unire il driver JDBC alla piattaforma EE.

La piattaforma Java SE7 necessita del JDBC 4.1.

## 1.8.2 Java Naming and Directory Interface API

La Java Naming and Directory Interface API (JNDI) offre funzionalità di naming e directory e permette alle applicazioni di accedere a molteplici servizi di naming e directory, come LDAP, DNS e NIS.

JNDI offre inoltre metodi alle applicazioni per performare operazioni standard di directory, come associare attributi agli oggetti e cercare oggetti usando i loro attributi.

Usando JNDI un’applicazione Java EE può conservare e recuperare qualunque tipo di oggetto nominato in Java, permettendo alle applicazioni Java EE di coesistere con molti sistemi e applicazioni derivate.

I servizi **di naming** di Java EE permettono all’applicazione del client, agli enterprise beans e ai componenti web di **accedere all’ambiente** del JNDI naming.

Un **ambiente naming** permette ai componenti di essere modificati senza bisogno di accedere al o cambiare il source code del componente.

Un container implementa l’ambiente dei componente e lo mette a disposizione dello stesso come un **JNDI naming context.**

L’ambiente di naming offre quattro namespace logici per gli oggetti disponibili per componenti, moduli o applicazioni o condivisi da tutte le applicazioni deprecate: java:comp, java:module, java:app, e java:global.

Un componente Java EE può accedere a oggetti supportanti il named system e definiti dall’utente.

I nomi di alcuni oggetti standard supportati dal sistema quali l’oggetto JDBC DataDource, la JMS connection factory e l’oggetto JTA UserTransaction sono archiviati nel java:comp.

La piattaforma Java EE permette ai componenti di nominare oggetti definiti dall’utente, quali enterprise beans, entrate in un ambiente, oggetti JDBC DtaSource e destinazioni di messaggi.

Un componente Java EE può anche collocare il suo environment naming context usando le interfacce JNDI.

Un componente può creare ad esempio un javax.naming.InitialContext col nome java:comp/env.

Il naming environment di un componente è conservato direttamente nell’environment naming context o in uno dei suoi sottocontesti (subcontexts).

## 1.8.3 JavaBeans Activation Framework

Il JavaBeans Activation Framework (JAF) è usato dall’API JavaMail e offre servizi standard per determinare il tipo di un pezzetto arbitrario di dati, incapsulare l’accesso agli stessi, scoprire le operazioni possibili sugli stessi e creare i componenti JavaBeans appropriati per fare queste operazioni.

## 1.8.4 Java API for XML processing

Il Java API for XML Processing (JAXP), parte della specifica della piattaforma Java SE, supporta la processazione dei documenti XML usando il Document Object Model (DOM), semplici API per XML (SAX) e Extensible Stylesheet Language Transformations (XSLT).

JAXP offre inoltre supporto per i nomi, rendendo possibile lavorare con schemi che altrimenti produrrebbero conflitti tra nomi.

Creato per essere flessibile, JAXP permette di usare qualunque parser di compilazione XML o processore XSL da qualunque applicazione e supporta lo schema W3C.

## 1.8.5 Java Architecture for XML Binding

Il Java Architecture for XML Binding (JAXB) offre un modo conveniente di unire uno schema XML a una rappresentazione con il linguaggio di programmazione Java.

JAXB può essere usato indipendentemente da o in combinazione con JAX-WS, in quest’ultimo caso offre un’unione standard per i messaggi del web service. Tutti i container client, i web containers e gli EJB containers delle applicazioni Java EE supportano JAXB API.

La piattaforma Java EE7 necessita di JAXB 2.2.

## 1.8.6 Java API for XML Web Services

La Java API for XML Web Services (JAX-WS) offre supporto per i servizi web che usano l’API JAXB per legare i dati XML agli oggetti Java.

La specifica di JAX-WS propone client API’s per accedere ai servizi web come tecniche per implementare gli endpoints dei servizi web.

La specifica Implementing Enterprise Web Services descrive lo sviluppo di clients e servizi basati su JAX-WS.

La specifica di EJB e Java Servlet descrivono ulteriori aspetti di questo sviluppo.

Le applicazioni basate su JAX-WS possono essere sviluppate usando uno di questi modelli di sviluppo.

La specifica descrive il support per il gestionale dei messaggi che può processare le richieste e le risposte dei messaggi.

Solitamente questi gestionali di messaggi vengono eseguiti nello stesso container e con gli stessi privilegi e contesto di esecuzione dei componenti client o endpoint del JAX-WS a cui sono associati e hanno accesso allo stesso namespace JNDI dei componenti associati

Serializzazioni e deserializzazioni personalizzati, se supportati, sono trattati allo stesso modo del gestore dei messaggi.

La piattaforma Java EE7 necessita di JAX-WS 2.2.

## 1.8.7 SOAP with Attachments API for Java

Il SOAP for Attachment API for Java (SAAJ) è un’API di basso livello da cui JAX-WS dipende.

SAAJ permette la produzione e la fruizione di messaggi conformi alle specifiche SOAP 1.1 e 1.2 e al SOAP with Attachment note.

La maggioranza degli sviluppatori non lo utilizza preferendo l’API di alto livello JAX-WS.

## 1.8.8 Java Authentication and Authorization Service

Il Java Authentiation and Authorization Service (JAAS) offre modo a un’applicazione Java di autenticare e autorizzare un gruppo specifico di utenti a utilizzarla.

JAAS è la versione in Java del framework standard Pluggable Authentication Module (PAM) che estende l’architettura di sicurezza della piattaforma Java per supportare le autorizzazioni.

## 1.8.9 Common Annotations for the Java Platform

Le Annotations abilitano uno stile di programmazione dichiarativo nella piattaforma Java.

La piattaforma Java EE7 richiede Common Annotations for the Java Platform 1.2.

## GlassFish Server Tools

GlassFish Server è un’implementazione conforme alla piattaforma Java. Oltre a supportare tutte le API’s descritte negli articoli precedenti, GlassFish Server include alcuni strumenti Java EE che non fanno parte della piattaforma ma sono offerti come strumenti di aiuto agli sviluppatori.

Questo capitolo da una sommaria introduzione agli strumenti che costituiscono un GlassFish Server.

Le istruzioni per far partire e stoppare il GlassFish Server, come far funzionare la Administration Console e come far partire e stoppare il Java DB server saranno nel capitolo 2.

Il GlassFish Server contiene gli strumenti illustrati nella seguente tabella.

