- MODULO 6 -
Introduzione a JEE e componenti Web Oriented

Dynamic Solution TECHnology

Gianluca Contaldi

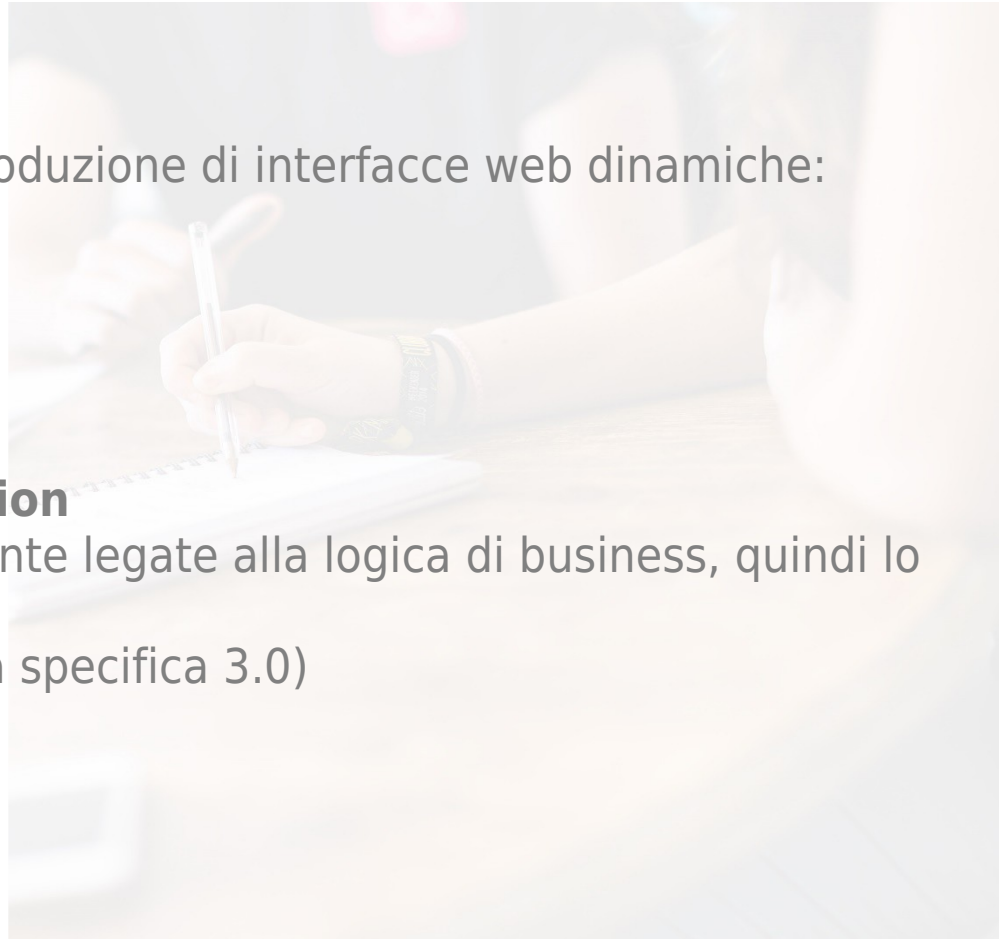
26/08/2016

La tecnologia Java 2 Enterprise Edition (J2EE, nelle versioni più recenti JEE) è diventata negli anni sinonimo di sviluppo di applicazioni aziendali robuste, sicure ed efficienti.

Può facilitare la creazione di modelli B2B (Business to Business) e B2C (Business to Consumer) e quindi permettere all'azienda lo sviluppo di nuovi servizi. Infatti rende facile l'accesso ai dati e la sua rappresentazione in diverse forme (un browser web, un applet, un dispositivo mobile, un sistema esterno, ecc).

Dal punto di vista tecnologico tutto ciò è realizzato con una struttura a livelli, dove ogni livello implementa uno specifico servizio. In pratica J2EE è un raccoglitore di tecnologie che facilitano lo sviluppo di software web based distribuito.[1]

Introduzione a JEE: tecnologie



Tecnologia Web Application

Si tratta delle tecnologie legate alla produzione di interfacce web dinamiche:

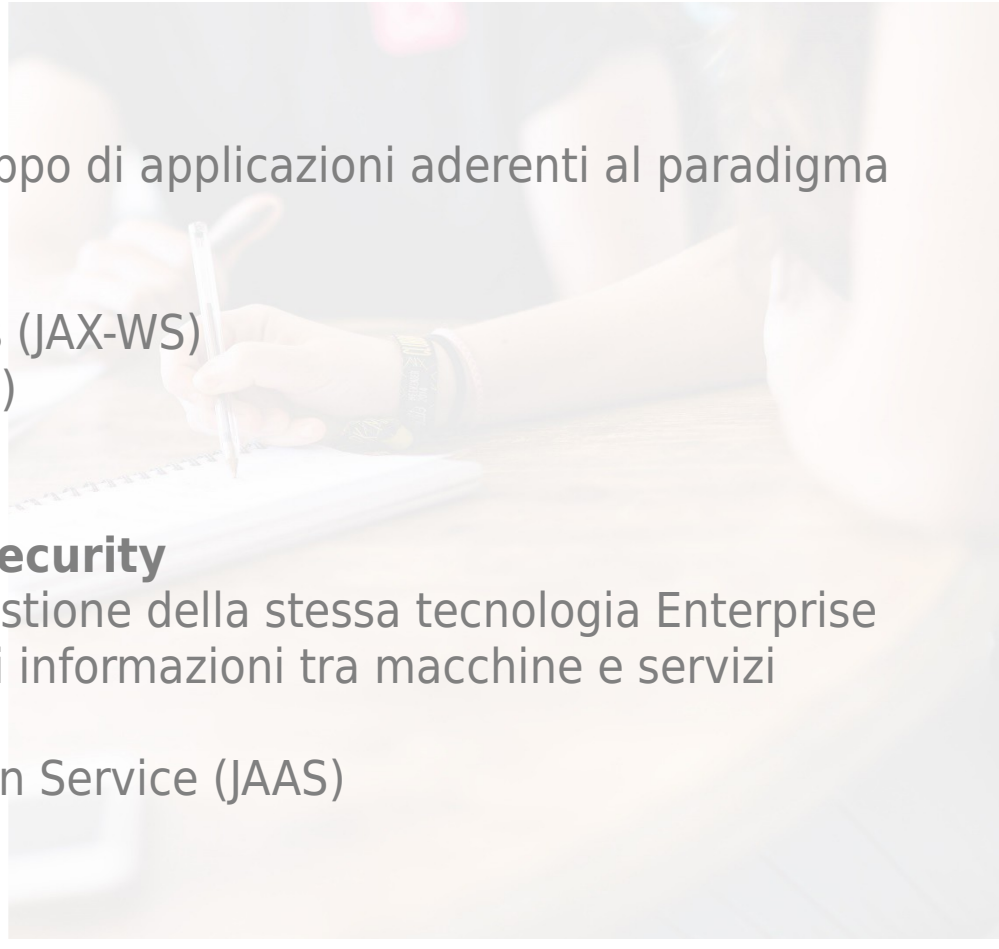
- Java Server Pages (JSP)
- XML
- Java Server Faces (JSF)
- Custom Tag

Tecnologia Enterprise Application

Si tratta delle tecnologie più direttamente legate alla logica di business, quindi lo sviluppo vero e proprio:

- Enterprise JavaBeans (EJB, giunti alla specifica 3.0)
- JNDI
- JavaMail
- Java Message Service (JMS)
- Java Transaction (JTA)[1]

Introduzione a JEE: tecnologie



Tecnologia Web Services

Si tratta delle tecnologie utili allo sviluppo di applicazioni aderenti al paradigma SOA (Service Oriented Architecture):

- Web Services
- Java API for XML-Based Web Services (JAX-WS)
- Java API for XML-Based RPC (JAX-RPC)

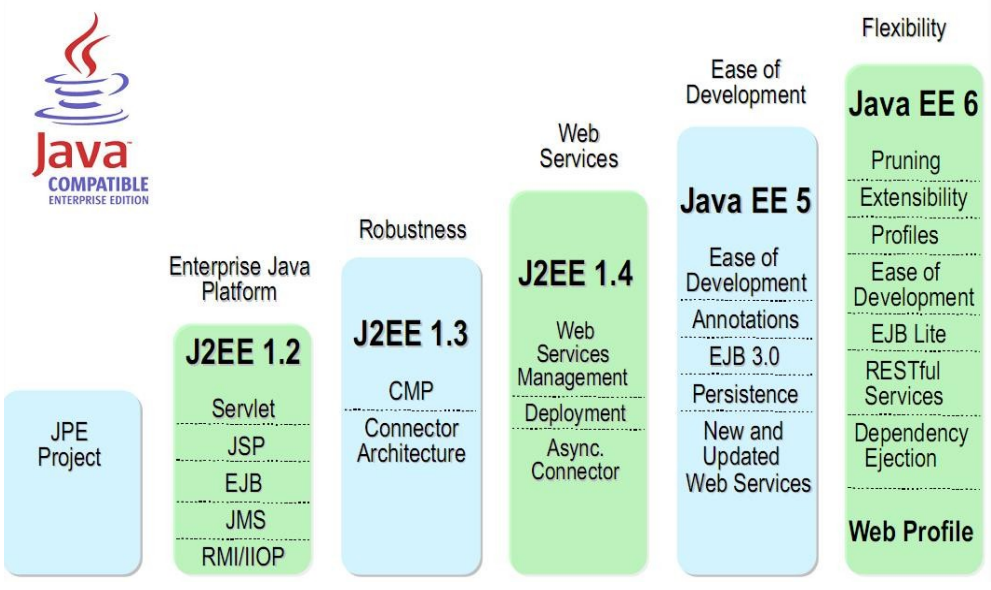
Tecnologia Management and Security

Si tratta delle tecnologie legate alla gestione della stessa tecnologia Enterprise per realizzare l'accesso e lo scambio di informazioni tra macchine e servizi distribuiti.

- Java Authentication and Authorization Service (JAAS)
- Java Connector Architecture (JCA)[1]

Introduzione a JEE: tecnologie

Possiamo immaginare che le “*tecnologie enterprise*” vengono usate per gestire l’accesso ai dati (uno o più database), mentre le “*tecnologie web application*” vengono usate per mostrare i dati al consumatore. In un contesto B2B, inoltre, le “*tecnologie web service*” verranno utilizzate per scambiare informazioni con i partner aziendali, il tutto mentre le “*tecnologie di gestione*” sovrintendono tutti i processi informativi assicurando la sicurezza delle transazioni.[1]



L'*Application Server*

Un Application Server è uno strato software residente su una macchina **server** che fornisce una serie di **servizi**, in particolar modo, fornisce tutti i servizi che la tecnologia Java Enterprise espone.

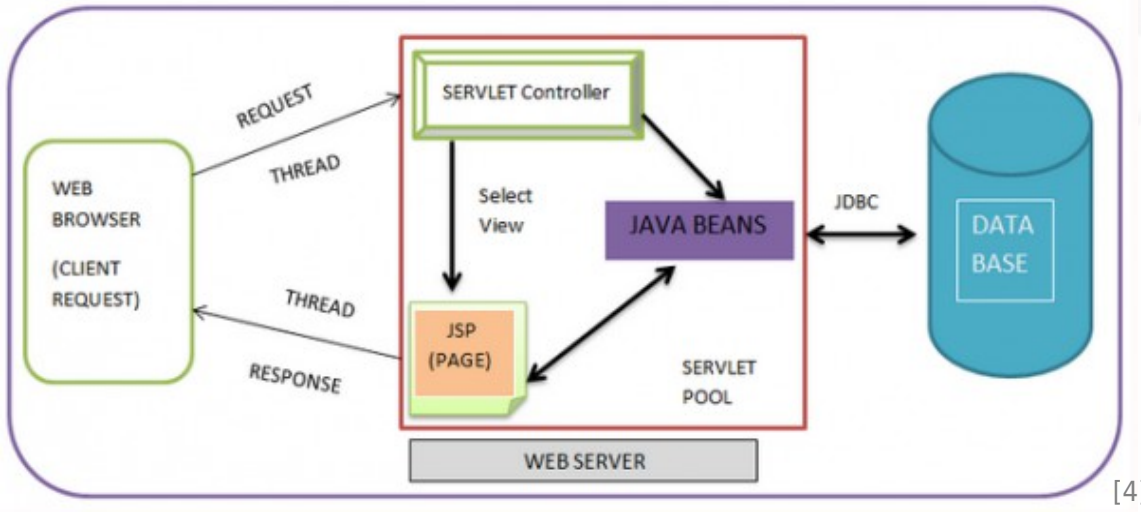
Apache Tomcat è un application server open source, usato per quasi il 70% dei server della rete mondiale ed è la soluzione ottimale per chi sceglie di utilizzare **Java** e **JSP** per realizzare il codice delle pagine Web e della logica.

Lo possiamo scaricare da qui: http://tomcat.apache.org/download-70.cgi

Ad essere proprio pignoli, Tomcat è un **Servlet Container** ed un **JSP Engine**. Un motore quindi in grado di eseguire lato server applicazioni Web basate sulla tecnologia J2EE, costituite da componenti Servlet e da pagine JSP.[2]

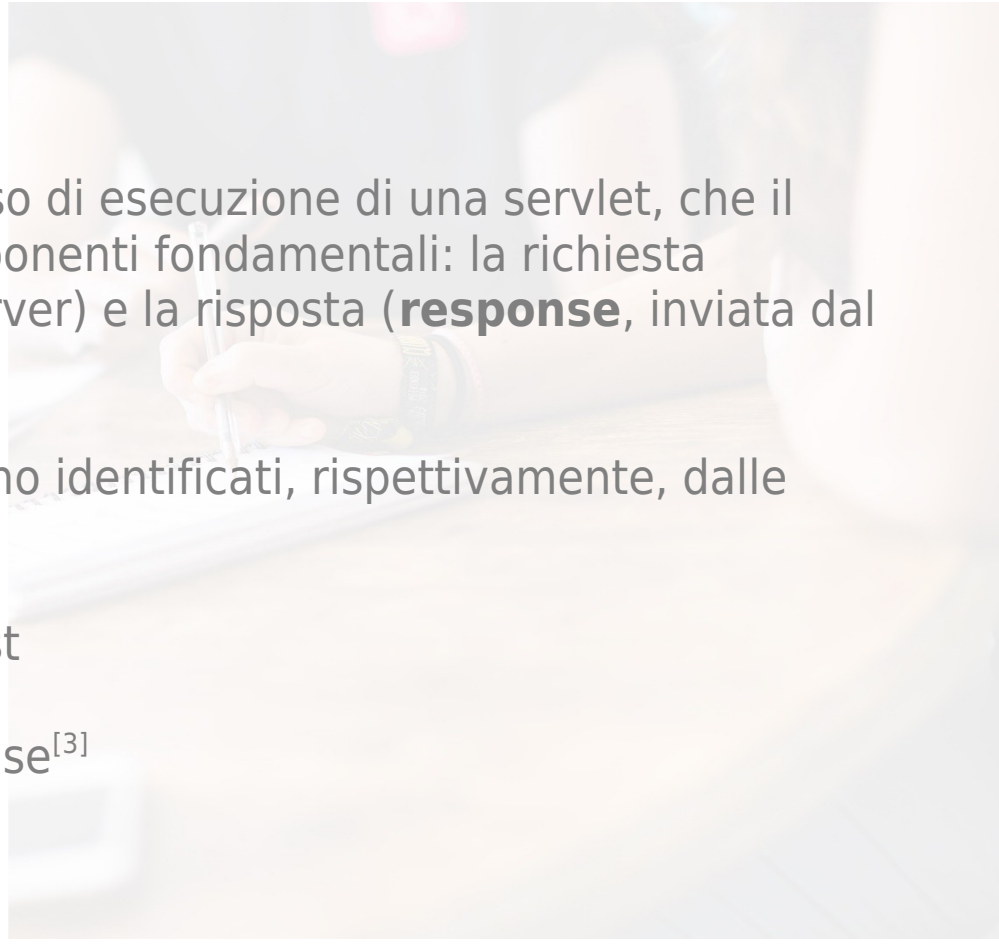
Servlet: che cos'è?

Una **Servlet** è semplicemente, un **programma** scritto in Java (che estende la classe *javax.servlet.http*) e residente su un server, in grado di gestire le richieste generate da uno o più client, attraverso uno scambio di messaggi tra il server ed i client stessi che hanno effettuato la richiesta. Tipicamente sono collocate all'interno di Application Server o Web Application Server come, ad esempio, Tomcat.[3]



[4]

HttpServletRequest e *HttpServletResponse*



Abbiamo visto, esaminando il flusso di esecuzione di una servlet, che il flusso stesso è incentrato su due componenti fondamentali: la richiesta (**request**, inviata dal client verso il server) e la risposta (**response**, inviata dal server verso il client).

In Java, questi due componenti sono identificati, rispettivamente, dalle seguenti interfacce:

- javax.servlet.http.HttpServletRequest
- javax.servlet.http.HttpServletResponse[3]

I metodi principali di una *Servlet*

Creare una Servlet vuol dire, in termini pratici, definire una classe che derivi dalla classe HttpServlet. I metodi più comuni per molti dei quali si è soliti eseguire l'overriding nella classe derivata sono i seguenti:

- *void **doGet** (HttpServletRequest req, HttpServletResponse resp)*
- *void **doPost** (HttpServletRequest req, HttpServletResponse resp)*
- *void **doPut** (HttpServletRequest req, HttpServletResponse resp)*
- *void **doDelete** (HttpServletRequest req, HttpServletResponse resp)*[3]

JSP – Java Server Page

Il livello di presentazione nella tecnologia JEE è composto dalle **JSP**. Si tratta di semplici file testuali, che accanto al codice **HTML**, presentano codice in linguaggio **Java**, permettendo un utilizzo dinamico del servizio web.[1]

Iniziamo con il classico messaggio “Ciao Mondo!” da stampare a video.

```
<!-- prova.jsp -->
<html>
<body>

<% out.println("Ciao Mondo!"); %>

</body>
</html>
```

Come si può notare le sezioni racchiuse tra **<%** e **%>** sono quelle che contengono le istruzioni in linguaggio **Java**. I file per essere riconosciuti dal server come pagine JSP devono essere salvati con estensione *.jsp*

La prima volta che si effettua la richiesta del file, quest'ultimo viene compilato, creando una *servlet*, che sarà archiviata in memoria (per servire le richieste successive); solo dopo questi passaggi viene elaborata la pagina HTML che viene mandata al browser.

Ad ogni richiesta successiva alla prima, il server controlla se sulla pagina .jsp è stata effettuata qualche modifica, in caso negativo richiama la servlet già compilata, altrimenti si occupa di eseguire nuovamente la compilazione e memorizzare la nuova servlet.

JSP: Script, dichiarazioni ed espressioni

Gli script in JSP sono vere e proprie porzioni di codice inserite all'interno di pagine HTML che possono rappresentare: dichiarazioni, espressioni o scriptler. Il codice java del nostro file jsp va racchiuso come già detto all'interno dai tag <% %>, mentre per le dichiarazioni la sintassi cambia leggermente.

Dichiarazioni

La sintassi per le dichiarazioni è la seguente: **`<%! dichiarazione %>`** sia per la dichiarazioni di variabili, sia per la dichiarazione di metodi.

```
<%// dichiarazione di una stringa %>
<% ! String stringa=new string("ciao a tutti") %>

<% // dichiarazione di una funzione che dati due numeri in ingresso
// restituisce la loro somma %>
<% ! public int somma (int primo, int secondo){
return (primo + secondo);
}//somma %>
```

Espressioni

La sintassi per le espressioni è la seguente: **`<%= espressione %>`**

Ad esempio:

```
<%= somma (2,3)%>
```

se inserita all'interno dei tag <BODY> e </BODY> stamperà a video l'output della funzione somma.

JSP: Direttive

Questo tipo di oggetti specificano le **caratteristiche** globali della pagina.

Al momento sono definite tre direttive:

- ***direttiva page***: modifica alcune impostazioni che influiscono sulla compilazione della pagina, attraverso la modifica degli attributi (*language*, *import*, *isThreadSafe*, *info*, *errorPage* e *isErrorPage;*
- ***direttiva include***: modificando l'attributo file è possibile aggiungere dei file sorgenti aggiuntivi da compilare assieme alla pagina (possono per esempio essere file HTML o semplici file di testo);
- ***direttiva taglib***: permette di utilizzare una libreria di tag personalizzati (basta specificare l'URL della libreria e il prefisso da utilizzare per accedere ai tag, rispettivamente settando i parametri url e prefix).

I **JavaBean** sono classi scritte in Java secondo una particolare **convenzione**. Sono utilizzate per **incapsulare** più oggetti in un oggetto singolo, il **bean**, cosicché tali oggetti possano essere passati come un singolo oggetto bean invece che come multipli oggetti individuali.

Al fine di funzionare come una classe *JavaBean*, una classe di un oggetto deve obbedire a certe convenzioni in merito ai nomi, alla costruzione e al comportamento dei metodi:

- La classe deve avere un costruttore senza argomenti;
- Le sue proprietà devono essere accessibili usando *get*, *set*, *is* (usato per i booleani al posto di *get*) e altri metodi (così detti metodi accessori);
- La classe dovrebbe essere ***serializzabile*** *(implements java.io.Serializable)*, ossia capace di salvare e ripristinare il suo stato in modo persistente.[5]

JavaBean: un esempio

```java
// PersonaBean.java

public class PersonaBean implements java.io.Serializable {

    private String nome;
    private boolean sposata;

    // Costruttore senza argomenti
    public PersonaBean() { }

    // Proprietà "nome" (da notare l'uso della maiuscola) lettura / scrittura
    public String getNome() {
        return this.nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }

    // Diversa sintassi per gli attributi boolean ( 'is' al posto di 'get' )
    public boolean isSposata() {
        return this.sposata;
    }
    public void setSposata(boolean sposata) {
        this.sposata = sposata;
    }
}
```

[5]

Questi oggetti sono finalizzati ad un migliore **incapsulamento** del codice, generalmente inteso come inclusione e utilizzo di ***JavaBean***.

Le azioni standard per le pagine JSP sono:

<jsp:useBean> : permette di utilizzare i metodi di un JavaBean;
<jsp:setProperty> : permette di impostare il valore di un parametro di un metodo di un Bean;
<jsp:getProperty> : permette di acquisire il valore di un parametro di un Bean;
<jsp:param> : permette di dichiarare ed inizializzare dei parametri all'interno della pagina.
<jsp:include> : permette di includere risorse aggiuntive di tipo sia statico che dinamico.
<jsp:foward> : consente di eseguire una richiesta di una risorsa statica, un servlet o un'altra pagina JSP interrompendo il flusso in uscita.

Bibliografia

1. Guida JEE 7, EJB 3 e JPA, html.it
2. Guida Application Server, html.it
3. Primi passi con le Servlet, html.it
4. Java Web Hosting, webhostingsearch.com
5. JavaBean, wikipedia.it