DIREZIONI CORRENTI di EVOLUZIONE

Sicuramente il campo applicativo più ampio di dimensioni è il **sistema Web** stesso

Le sfide sono di:

- utilizzo al meglio i **sistemi Web** partendo dalla computazione locale per una visione coordinata
- uso di tutte le **risorse disponibili** in rete per migliorare i servizi offerti
- creazione di un unico sistema di **calcolo globale** ed **accessibile**

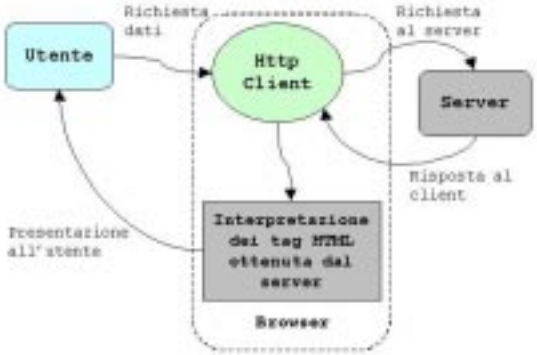
Molte aree applicative di interesse

- **accesso sicuro a dati via Web**
- **e-commerce & e-market**
- **Web computing**

Si arriva a vedere il sistema Web come una **infrastruttura per ottenere servizi (con Qualità e Costo negoziabile)**

EVOLUZIONI DEL CALCOLO WEB

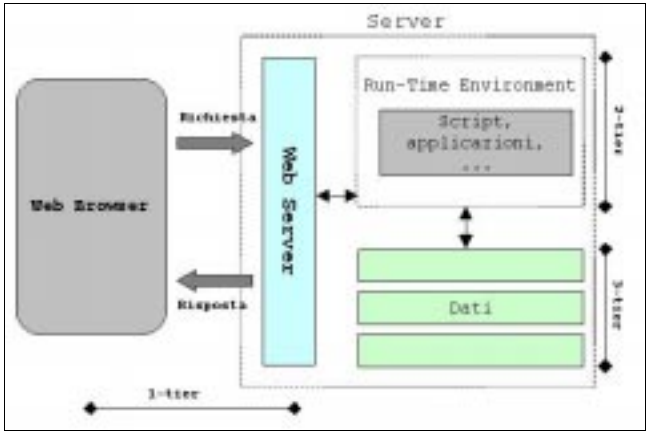
Browser tradizionali e server web



Sistemi con stato

Accesso a risorse del server

attraverso la interazione con il sistema locale al server



PROBLEMI WEB

I limiti più sentiti sono:

- limiti di operazione e alle opzioni
- **mancanza di stato nel protocollo**
- **mancanza di sicurezza**

Web server e le evoluzioni del protocollo tendono a fare permanere la connessione per consentire di usare il **canale** per trasferimenti multipli

Server Web (Apache)

canali che vengono mantenuti per una serie di trasferimenti di informazioni coordinate
Si bilancia il costo del canale con una sequenza di operazioni sullo stesso

i browser tendono a memorizzare localmente una serie di attributi che forniscono automaticamente ai server da cui li hanno ottenuti
per simulare lo stato della interazione

Un **dominio** mantiene una storia delle visite precedenti

dalla parte del **cliente**
dalla parte del **server**

STATO

cookies

un cliente può memorizzare (con scadenza) attributi (**stato**) da usare per successive interazioni
cookies anche per specificare preferenze utente

spesso uno stesso server ha molti cookies per pagina che vengono ripresentati solo al servitore corretto

formato **nome = valore**
mantenuti sul disco permanente del cliente
cookies con scadenza e anche cifrati

log attività

un **server** può tenere (con scadenza) la storia delle interazioni da usare successivamente

si possono memorizzare molti eventi

Pagina richiesta, Host remoto, Tipo del Browser, Pagina Riferita, Data eTempo

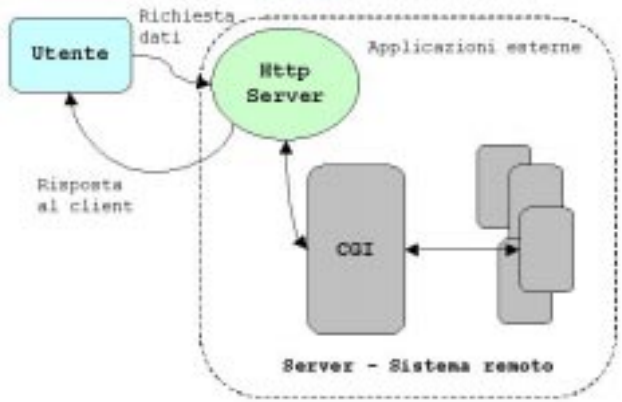
Necessità di applicazioni di esplorazione dei dati

WEB COMPUTING

Il primo passo è la possibilità di superare i vincoli del protocollo HTTP e delle interazioni consentendo di integrare i diversi componenti e di ottenere nuove forme di accesso

Se si vogliono variare le suddivisioni tra le due parti interagenti

Elaborazione sul client via applet
Elaborazione sul server via CGI

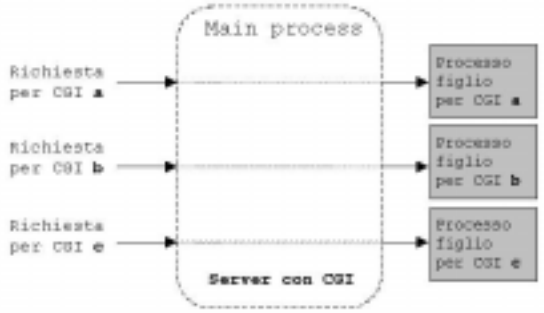


Applet scaricate da una richiesta dal server
CGI per accesso alle risorse del server

Automatismi nella invocazione
Trasparenza per l'utilizzatore

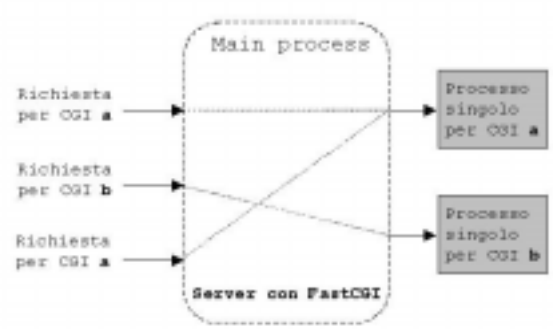
CGI: problemi, limiti e costi

Ad ogni richiesta, viene attivato un **processo** che specifica la CGI (overhead della generazione)
Tempo di attesa per eventuali altre richieste contemporanee (o problemi di mutua esclusione)



Primi passi evolutivi

FAST CGI prevedono un processo già attivo per ogni servizio CGI specificato



Sono state proposte API per funzioni standard
ISAPI Microsoft, NISAPI Netscape
Specifiche dei servizi tipicamente compilate

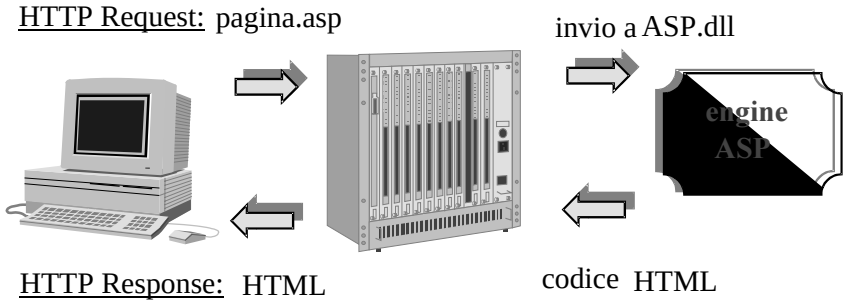
Uso di linguaggi script

Il linguaggio HTML è interpretato …
Allo stesso modo, possiamo pensare a
linguaggi script
intrinsecamente portabili (interpretati)
sia sul **client**
Visual basic
Jscript (piccole elaborazioni grafiche locali)
Sia sul server
Javascript, Perl, PHP, …

Spesso le parti di script sono integrate nelle pagine HTML e vengono trattate da un processore comandi (**engine**) che produce in uscita HTML

Active Server Pages

Definite dalle Microsoft per
mescolare HTML e componenti script

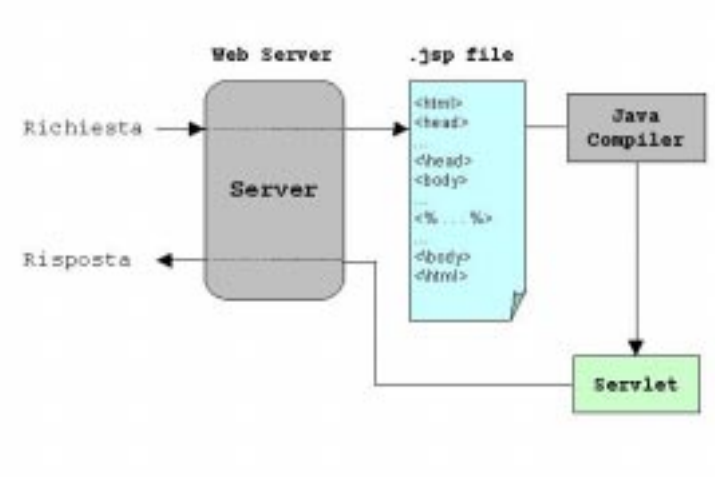


non portabili e supportate da
Microsoft **IIS** Internet Information Server

Operazioni Server-side

Java Server Pages JSP

Parte della pagina HTML contiene specifiche in Java
Queste sono passate alla macchina virtuale integrata nel server e producono pagine HTML dinamiche (usando **servlet**)



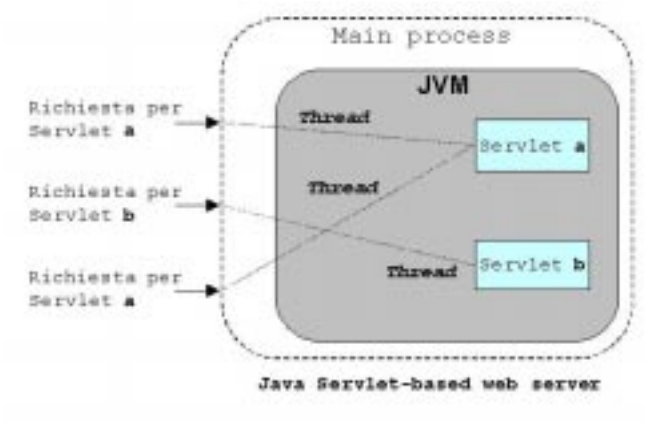
Le **JSP** sono portabili in quanto non assumono una specifica architettura di Web server (come le ASP), ma solo la presenza di una JVM

In genere, supportate da qualunque Web server

Java servlet

Estensioni di attività in esecuzione sul server e integrabili facilmente con il server (via JVM)

Le **servlet** sono componenti di codice Java residenti sul Web server
se **invocate** producono attività nella JVM eseguendo come processi leggeri



☺ i costi di attivazione sono molto limitati
☺ non usciamo dall'ambiente del server

☺ possiamo gestire facilmente mutua esclusione o parallelismo

Java Servlet API specification v2.2 (1999)

Servlet

Le servlet si basano sul concetto di automatizzazione del supporto alla attivazione e alla esecuzione

Le servlet vengono gestite e sono inserite in un
container o **engine**
È responsabilità del container il dispatching delle servlet provvedendo il corretto passaggio dei **parametri** e la raccolta dei **risultati**

> *Si consideri una analogia con il middleware CORBA che fornisce una serie di strutture di supporto per le invocazioni in formato evoluto (POA, repository, ecc.)*

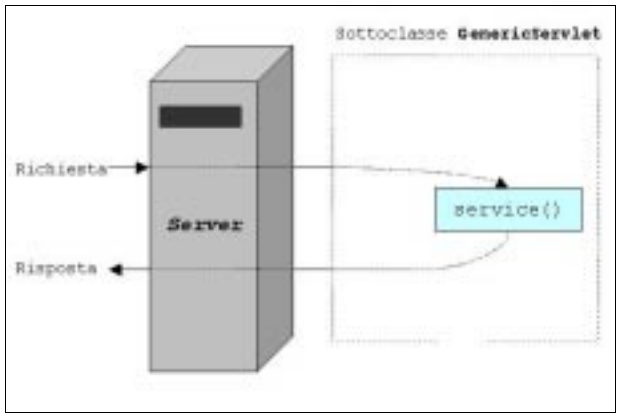
Il **container** è responsabile della:
- **istanziazione** e **caricamento** di una servlet
- esecuzione delle **operazioni** delle servlet
- fase di **scaricamento finale**

Le **servlet**
Non hanno limitazioni alle funzioni che si possono richiedere
tutta la visibilità di Java
azioni sul file system, accessi a database, ecc.

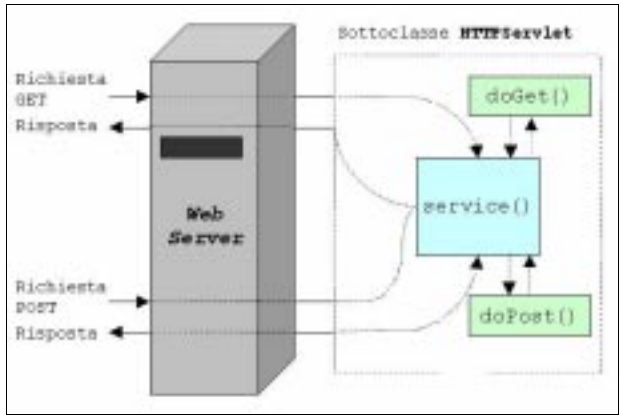
Estrema portabilità
Supporto su i più comuni Web server

Servlet

La operazione fondamentale per una servlet è il **servizio** (**service**) che rappresenta la operazione attuata dalla servlet
Due tipi di servlet, **generiche** e **HHTP**

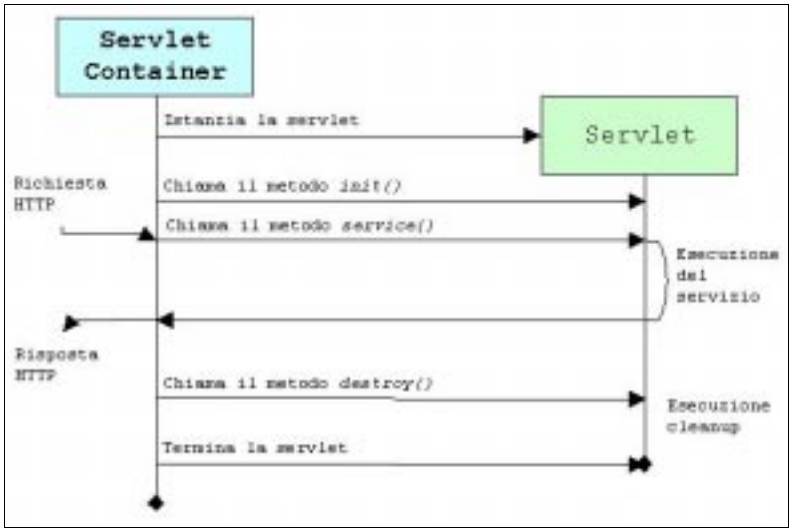


Per le servlet che trattano http



Ciclo di vita delle Servlet

Le servlet consentono di prevedere tre operazioni fondamentali,
correlate alla gestione (prologo ed epilogo)
`init(), destroy()`
e servizio
`service()`



in tre fasi:

1. creazione ed inizializzazione della servlet
2. gestione di uno o più servizi richiesti dai client
3. distruzione della servlet e deallocazione della memoria

Oggetti per servlet

Una servlet è una istanza di una classe che estende (eredita da)
`javax.servlet.GenericServlet`
`javax.servlet.http.HttpServlet`

le classi devono implementare la interfaccia
`javax.servlet.ServletInterface`

devono avere i metodi detti sopra
`init()`
`service()`
`destroy()`

Si usano oggetti per gestire:
HTTP Richieste (e ottenere parametri)
HTTP Risposte (e fornire risultati) e
basate su equivalenti del protocollo HTTP
La service si manifesta con metodi di
`doGet(); doPost();`
`doPut(); doDelete();`
con due parametri (in e out)
`HttpServletRequest`
`HttpServletResponse`

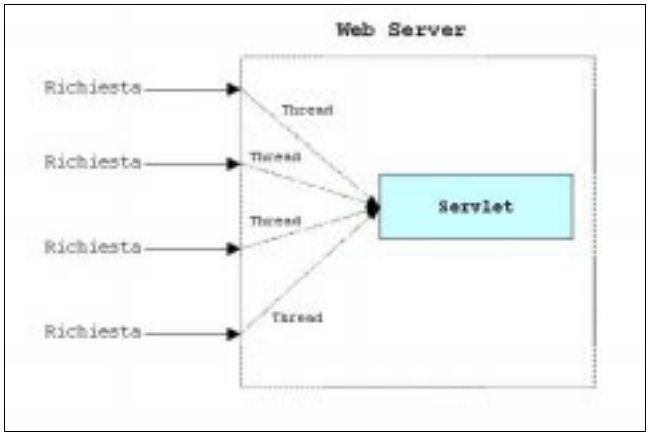
HTTP Sessioni
Sessioni come **sequenze di richieste HTTP dallo stesso cliente**
le sessioni sono mantenuti

SERVLET MULTITHREADED

Possibilità di avere più attività concorrenti per una stessa servlet (permesse dal container)

Se **mutua esclusione**
blocchi synchronized con eccessivo overhead
per la riattivazione di tutti i thread per ogni rilascio

In genere, lavorano in concorrenza
Possibilità di accessi contemporanei alle risorse



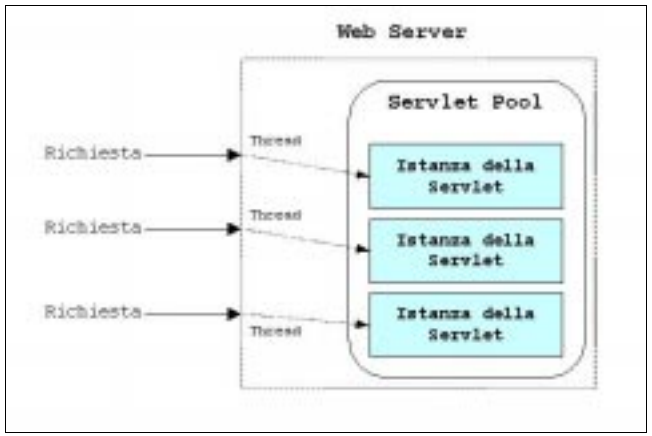
Se **esecuzioni parallele**
La servlet, dopo la inizializzazione, può servire **richieste** contemporaneamente

SERVLET MULTITHREADED

Si può prevedere di avere un pool di istanze delle servlet che sono state precreate e sono pronte per essere richieste

Una di queste viene attribuita ad ogni possibile richiesta appena è libera

e in caso non ce ne siano di disponibili?




```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class esempio extends HttpServlet {
  int contatore; /* numero di accessi */

public void init(ServletConfig c)
  throws  ServletException
{ super.init();
try { FileReader fr = new
           FileReader(“valoreiniziale”);
      BufferedReader br =
           new BufferedReader(fr);
      String appoggio = br.readLine();
      contatore = Integer.parseInt(appoggio);
      return;
    }
catch (FileNotFoundException a)
catch (IOException b)
catch (NumberFormatException c)
{ /* per inconsistenza dello stato */ }
  contatore = 0;// Valore iniziale di default
}
public void doGet(HttpServletRequest req,
      HttpServletResponse res)
      throws ServletException, IOException
{ res.setContentType("text/html");
  PrintWriter out = res.getWriter();
  contatore++;
  out.println("La servlet invocatata " +
      contatore + " volte");
}
public void destroy() {salvaStato();}
protected void salvaStato() {
try {
 FileWriter fw = new
          FileWriter(“valoreiniziale”);
 String appoggio = Integer.toString(contatore);
 fw.write(appoggio, 0, appoggio.length());
 fw.close();
 return;}
catch (IOException a) {}
}}
```

FUNZIONI DI UTILITÀ

ServletRequest/ HttpServletRequest

getInputStream (), getProtocol (), getRemoteAddress(),
getHeader (), getMethod (), getQueryString()
getRemoteUser (), getSession (), …

ServletResponse/ HttpServletResponse

getOutputStream (), setContentType (),
getWriter(), sendRedirect (), …

ENGINE

- stand-alone unico JVM come strumento
- plug-in inserito in tempi successivi

come attivazione (**DCOM**)

- ***In-process engine***
  plugin apre JVM con invocazione nativa

- ***Out-of-process engine***
  plugin comunica con socket con JVM

Sessioni e servizi con stato

Overhead

Le sessioni sono mantenute per un intervallo di tempo definito

Dopo un certo intervallo di inattività, una sessione viene invalidata automaticamente dal container

Session-tracking: tecniche tradizionali

Cookie

Un *cookie* contiene un insieme di coppie *chiave=valore*, generato dal web server e inviato al client con la risposta e il cliente lo fornisce per ogni richiesta

Hidden Form Field

session-tracking anonimo (non riferito ad alcun utente in particolare) utilizzano i campi HIDDEN previsti dal linguaggio HTML

User Authorization

si limitano gli accessi a risorse a soli quegli utenti in possesso di username e password

URL Rewriting

ogni URL utilizzato dall'utente può venire dinamicamente modificato o riscritto per permettergli di contenere informazioni aggiunte

Sessioni con servlet

la tecnologia servlet si integra con queste tecniche per sfruttare i metodi delle API Servlet

ad esempio
`getParameterValues()`
`getPathInfo()`
`getRemoteUser()`
`getCookies())`

oltre ad un supporto *built-in* per il servizio con stato

Servlet Session API: l'interfaccia `HttpSession`

Esempio:
creazione di una sessione per una richiesta `req`

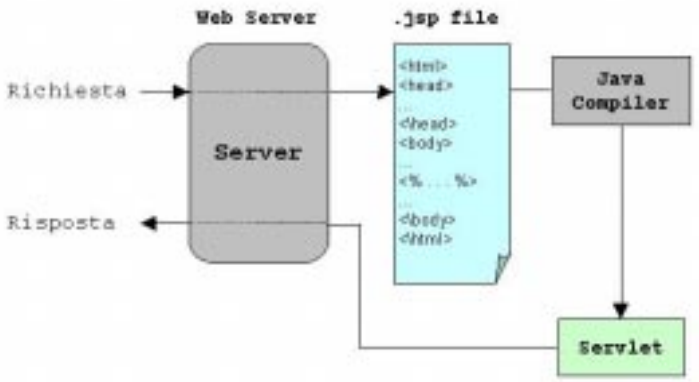
```
…
/* Ottengo un riferimento all'oggetto HttpSession
corrente se esiste, o ne creo uno nuovo

nel metodo getSession() un parametro true richiede
una sessione con il comportamento descritto sopra */

HttpSession sessione = req.getSession(true);
…
```


Java Server Pages

come collante per unire
- codice HTML
- componenti riusabili (*Enterprise JavaBeans*)
- applicazioni remote (servlet)
- codice Java
- script basati su linguaggio Java

JSP

1. Parte della pagina HTML contiene specifiche in **Java** tra tag <% %>
2. Il codice Java passato alla macchina virtuale integrata nel server per produrre una **servlet**
3. Si **compila** on the fly il codice Java e si **attiva** la servlet (verifica che non sia cambiata prima dopo la compilazione precedente)
4. Si produce la ***pagina HTML risultato***



Java Beans

tecnologia per sviluppare facilmente

componenti *Java riusabili*

applet *per pagine Web*

applicazioni *indipendenti*

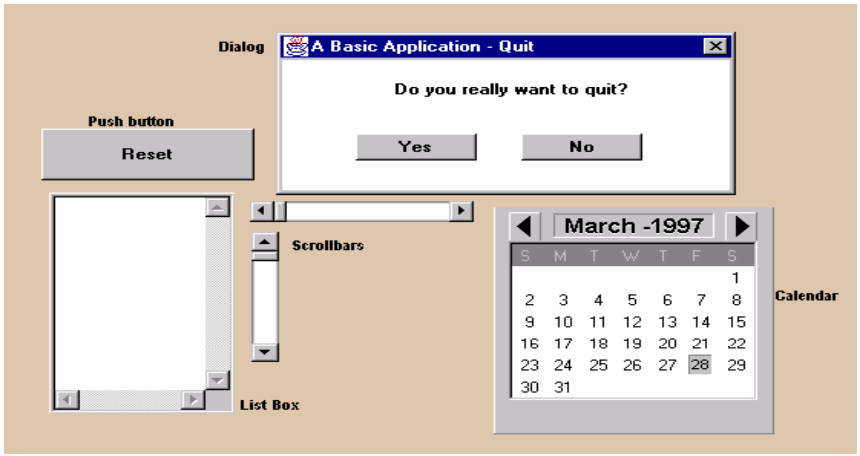
JavaBeans come componenti

- sviluppati senza scrivere codice Java
- che possono essere **innestati** e **combinati**

filosofia

Write once, run anywhere, reuse everywhere

Si passa da elementi GUI (bottoni, ecc.) fino a applicazioni di accesso a database



Uso di **riflessione** e **introspezione**

EJB e Architetture

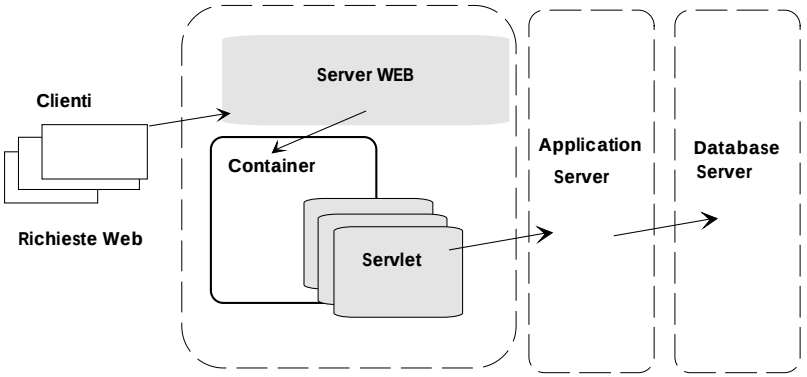
Esistono varietà specializzate di Beans

In particolare, gli **E**nterprise **J**ava **B**eans, pensati per la automatizzazione dei lavori di impresa

Architetture multilivello

Con divisioni funzionali tra diversi livelli

- Cliente (browser)
- Web server
- Application server (comandato dalle servlet)
- Database Server



Sistemi

con una migliore **suddivisione dei compiti** e

con una migliore separazione tra i **diversi livelli**