Tecnologie per la generazione di contenuti dinamici

Valeria Cardellini
Università di Roma Tor Vergata

Contenuti dinamici

- Risorse Web che richiedono l'esecuzione di una o più applicazioni prima di poter inviare la risposta al client
  - La risposta può essere generata anche sulla base della richiesta del client
- L'utente non si accorge che la risorsa identificata dall'URL richiesto necessita di un processamento per la generazione del contenuto
  - Il server trasmette il risultato dell'esecuzione
- Attenzione: risorse dinamiche ≠ risorse attive!
  - Le risorse attive contengono codice che viene eseguito sul client
  - Esempi di tecnologie per risorse attive: applet Java, JavaScript, Microsoft VBScript, Active X, Ajax

Contenuti dinamici (2)

- Esempi di utilizzo di contenuti dinamici:
  - Interazione personalizzata sulla base di diversi parametri, sia client-side, sia server-side (ad es., cookie, ora/giorno, stato del sistema)
  - Accesso ad informazioni gestite da server non HTTP (ad es. basi di dati o sistemi legacy)
  - Interrogazioni a motori di ricerca
- Con le risorse dinamiche, il Web è divenuto l'interfaccia per la fornitura di servizi sofisticati
  - Ulteriore evoluzione: i Web service

Livelli logici di un servizio Web-based

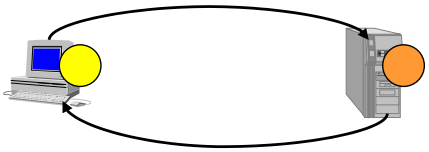
- In un servizio Web-based si possono identificare, a *livello logico*, i seguenti strati:
- **Interfaccia utente**
  - Rappresenta ciò che l'utente percepisce interagendo con il servizio
- **Logica di presentazione**
  - Rappresenta quello che accade quando l'utente interagisce con l'interfaccia del servizio Web-based
- **Logica di applicazione (business logic)**
  - Rappresenta tutte le funzionalità offerte dal servizio; costituisce il filtro bidirezionale tra logica di presentazione e logica dei dati
- **Logica dei dati**
  - Rappresenta la gestione fisica dei dati (memorizzazione, ricerca ed aggiornamento di dati), compresa la verifica della loro integrità e completezza
- Attenzione: non confondere i livelli logici con i *processi* che realizzano i livelli logici e con i *calcolatori* che eseguono i processi

Come mappare i livelli logici sui processi

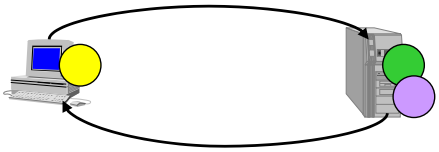
- Esistono 4 alternative
- Prima alternativa: con un solo processo (teorica)
- Seconda alternativa: con due processi
  - Processo client: gestisce il livello logico interfaccia utente
  - Processo server: gestisce tutti e 3 i livelli logici (presentazione, applicazione, dati)
- Terza alternativa: con tre processi
  - Processo client: gestisce il livello logico interfaccia utente
  - 1° processo server: gestisce il livello logico presentazione
  - 2° processo server: gestisce i livelli logici applicazione e dati
- Quarta alternativa: con quattro processi
  - Processo client: gestisce il livello logico interfaccia utente
  - 1° processo server: gestisce il livello logico presentazione
  - 2° processo server: gestisce il livello logico applicazione
  - 3° processo server: gestisce il livello logico dati

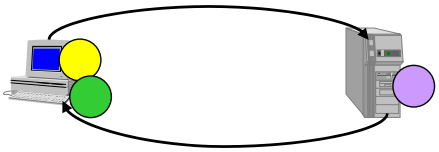
Come mappare i processi sui calcolatori

- 2 o 3 processi su 2 calcolatori
  - 2 processi su 2 calcolatori



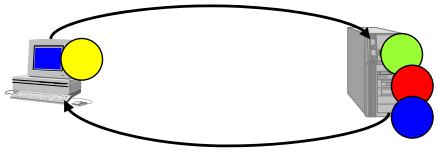
  - 3 processi su 2 calcolatori



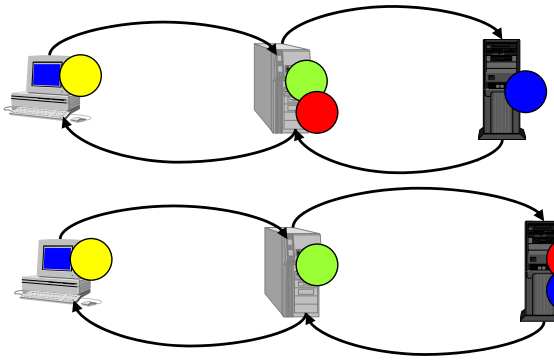


Come mappare i processi sui calcolatori (2)

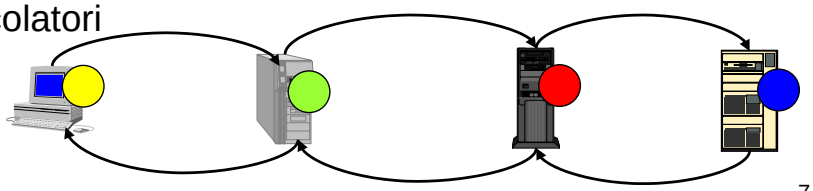
- 4 processi su 2 o più calcolatori
  - 4 processi su 2 calcolatori



  - 4 processi su 3 calcolatori

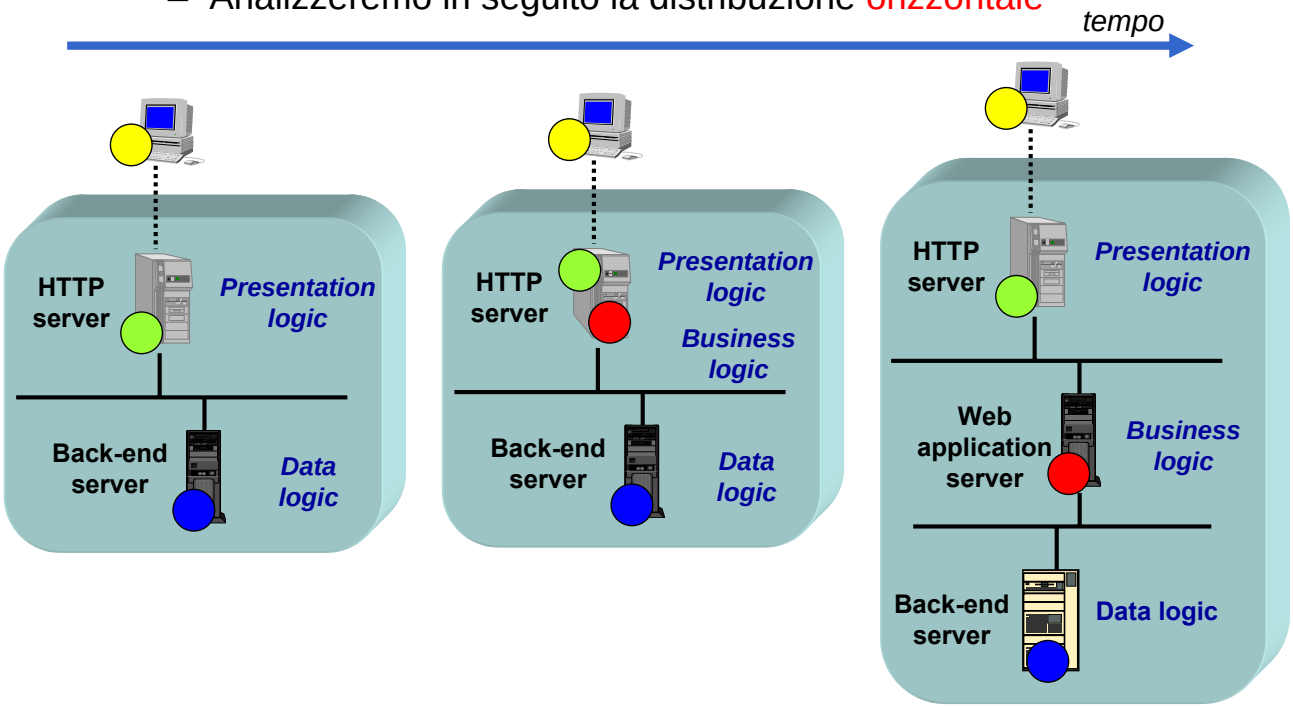


  - 4 processi su 4 calcolatori



Evoluzione delle architetture Web lato server

- Distribuzione *verticale*
  - Analizzeremo in seguito la distribuzione orizzontale

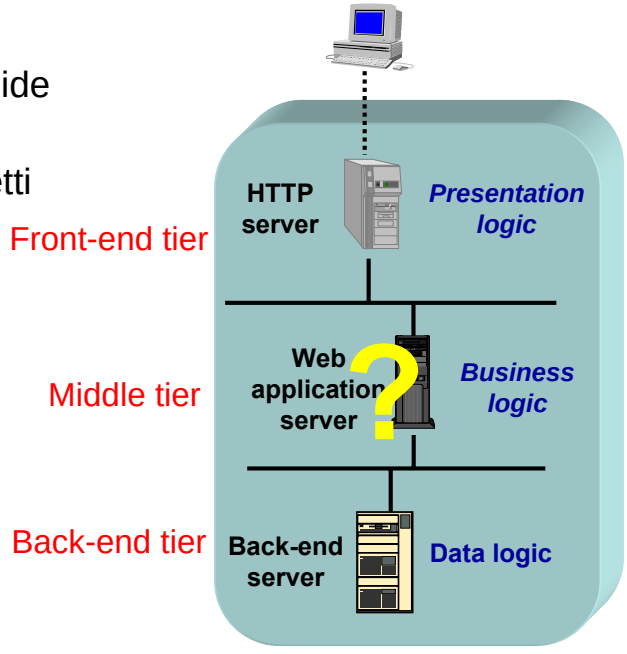


Livelli logica di presentazione e dei dati

- Logica di presentazione: agisce come un'interfaccia tra il livello di interfaccia utente ed il livello di logica dell'applicazione
  - Implementato mediante
    - HTTP server (ad es., Apache)
    - Eventuale plug-in di business logic
- Logica dei dati: gestisce dati strutturati su supporti di memorizzazione permanente
  - Implementato mediante un Database Management System (DBMS), ad esempio:
    - MySql (open source)
    - PostgreSQL (open source)
    - Microsoft SQL server
    - IBM DB2
    - Oracle

Livello logica di applicazione

- Analizziamo le tecnologie software per realizzare il middle tier
  - Processi esterni al server HTTP
  - Linguaggi di scripting server-side
  - Tecnologie distribuite ad oggetti



Tecnologie per middle tier: alberi

- Tecnologia che prevede processi *esterni* al server HTTP
  - La logica di presentazione e la logica di applicazione sono implementate da 2 processi server separati
  - I 3 processi server possono essere mappati su 2 calcolatori
    - Processi server per logica di presentazione e di applicazione sullo stesso calcolatore
- Basata su **Common Gateway Interface (CGI)**
  - Applicazioni realizzare in un qualsiasi linguaggio di programmazione, prevalentemente C o Perl
  - Il server HTTP crea un *nuovo processo* per ogni invocazione dell'applicazione CGI (quindi per ogni richiesta dinamica)
    - Vantaggi: interfaccia standard, isolamento della logica di elaborazione dell'applicazione dal server HTTP, flessibilità nella scelta del linguaggio
    - Svantaggio: stateless, vari overhead (creazione del processo, allocazione della memoria, …) $\rightarrow$ tecnologia non scalabile
      - Inaccettabile per server che devono gestire un elevato numero di richieste

Tecnologie per middle tier: 1° evoluzione

- *Prima evoluzione*: tecnologie che evitano la creazione di nuovi processi
  - La logica di presentazione e la logica di applicazione sono integrate nello stesso processo server
  - I 2 processi server possono essere mappati su 2 calcolatori
- **Fast CGI**
  - Condivisione dell'istanza di un processo CGI (*persistenza* del processo)
  - Processi persistenti organizzati in un pool, con strategia di dimensionamento del pool tipicamente configurabile
  - Anche esecuzione di applicazioni Fast CGI remote (CGI solo locali)
    - 3 processi server mappati su 3 calcolatori

Tecnologie per middle tier: 1° evoluzione (2)

- **Server API proprietarie**
  - Librerie condivise, caricate nello spazio del server HTTP, in grado di servire richieste multiple senza creare nuovi processi
  - Ad es., NSAPI (Netscape), ISAPI (Microsoft), API di Apache
  - Svantaggi: vulnerabilità (mancanza di isolamento) e scarsa portabilità (interfaccia proprietaria)

- **mod_perl**
  - Modulo di Apache (http://perl.apache.org/), in grado di interpretare script Perl all'interno del server HTTP
  - Interprete Perl *persistente* interno ad Apache
    - No overhead per invocare un interprete esterno (come nel caso di script CGI implementato in Perl)
  - Perché Perl? Linguaggio interpretato portabile e con elevate potenzialità

Tecnologie per middle tier: 1° evoluzione (3)

- **Java servlet**
  - Risposta in chiave Java alla programmazione CGI
  - Componenti software Java compilate e successivamente eseguite in ambiente server-side
  - Creazione di un thread per richiesta (anziché di un processo come CGI)
  - Vantaggi: maggiore efficienza del modello di esecuzione rispetto a CGI, estensione standard di Java, portabilità (basate su API standard), caching di computazioni precedenti (persistenza in memoria della servlet), connessioni persistenti a DB, sicurezza, robustezza
  - Svantaggio: presentazione non separata dalla generazione dei contenuti, conoscenza approfondita di Java

Tecnologie per middle tier: 2° evoluzione

- *Seconda evoluzione*: linguaggi di scripting HTML-embedded
  - Inclusione di codice di scripting all'interno di testo HTML statico (*template text*), utilizzando una sintassi orientata ai tag
  - Script HTML-embedded processato da un interprete che elabora l'intero template HTML e incorpora l'output nel testo HTML; al client viene restituita la risorsa finale
  - La logica di presentazione e la logica di applicazione sono generalmente integrate nello stesso processo server
- **Server Side Includes (SSI)**
  - Predecessore dei linguaggi di scripting HTML-embedded
  - Semplici direttive incluse come commenti SGML all'interno del file HTML (estensione .shtml) e processate dal server prima di inviare la risposta HTTP
  - Ad es., inclusione di file esterni, aggiunta di semplici informazioni "on the fly" al documento HTML
    - Es.: `<!--#echo var="DATE_LOCAL" -->` per la data
  - Svantaggio: numero ridotto di funzionalità offerte

Tecnologie per middle tier: 2° evoluzione (2)

- **Hypertest PreProcessor (PHP)**
  - Linguaggio di scripting general-purpose (file con estensione .php)
  - Integrazione come modulo all'interno del server Web
  - Limitazione: logica di presentazione e di applicazione sullo stesso calcolatore
- **Java Server Pages (JSP)**
  - Pagina JSP convertita e compilata in una servlet Java alla prima richiesta di accesso (file con estensione .jsp)
  - Vantaggi: portabilità, maggiore potenza per applicazioni complesse che richiedono componenti riusabili
  - Logica di presentazione e di applicazione anche su calcolatori diversi
- **Active Server Pages (ASP)**
  - Soluzione proprietaria Microsoft (file con estensione .asp)
  - Uso di script in diversi linguaggi (ad es. VBScript)
  - Svantaggio: soluzione specifica per server IIS, problemi di manutenibilità dell'HTML/script integrato

Tecnologie per middle tier: 3° evoluzione

- *Terza evoluzione*: tecnologie distribuite ad oggetti e a componenti
- Con l'evoluzione tecnologica delle piattaforme hardware, aumentano le aspettative sui servizi erogabili via Web
  - Aumenta considerevolmente la complessità del middle tier, che diviene una vera e complessa business logic
  - Non più accesso ad un solo DB ma a DB multipli, insiemi di file XML, directory service, …
- Applicazioni sempre più complesse: necessità di modularità, portabilità (spaziale e temporale), manutenibilità, riusabilità

Tecnologie per middle tier: 3° evoluzione (2)

- Le tecnologie per lo scripting HTML-embedded, che mirano principalmente all'incremento delle prestazioni, permettono un rapido sviluppo di applicazioni ma offrono scarsi strumenti di ingegnerizzazione del software
- La logica di applicazione complessa rende necessaria una sua separazione dalla logica di presentazione
  - Ma non è un ritorno al CGI, gli scopi sono diversi

Tecnologie per middle tier: 3° evoluzione (3)

- Le tecnologie distribuite ad oggetti e a componenti rispondono ai requisiti di modularità, portabilità e manutenibilità di una business logic complessa
  - *Portabilità*: il codice può essere eseguito dove serve
  - Migliora la *generalità*: molte applicazioni possono usare business object comuni
  - Migliora la *manutenibilità*: con una buona interfaccia (insieme di metodi pubblici), i cambiamenti influenzano solo l'oggetto
- Tecnologie distribuite ad oggetti: framework generali, non specifici per il Web
  - Sun Java Platform, Enterprise Edition (Java EE)
  - Microsoft .NET

Web Application Server

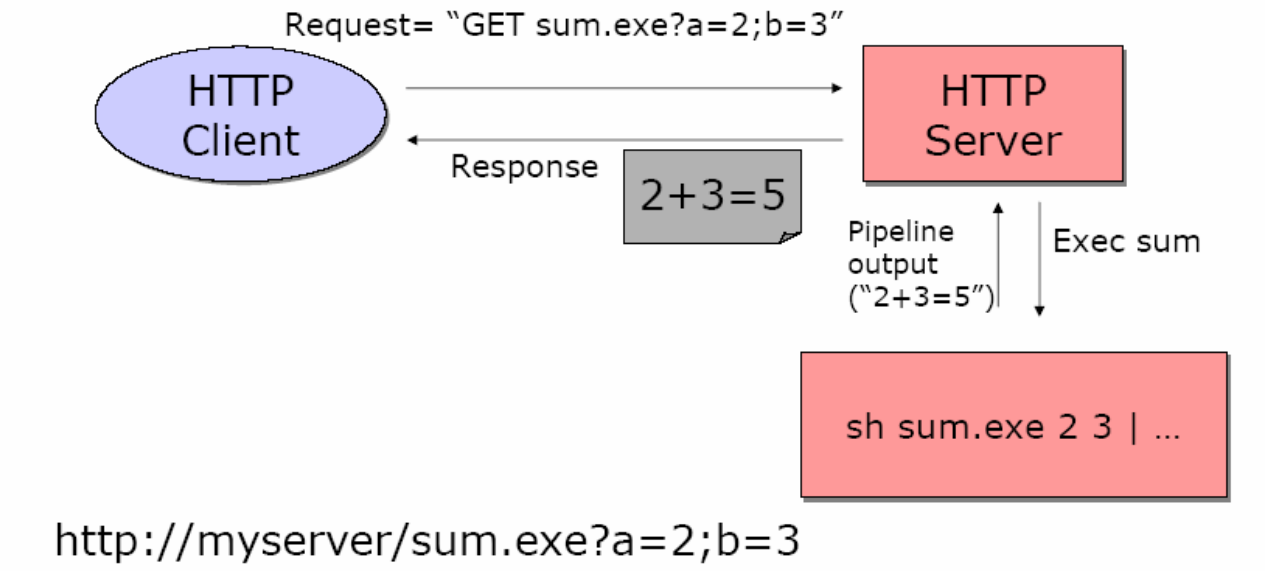
- Implementa il livello di business logic dell'applicazione Web
  - Fornisce l'ambiente di esecuzione per il supporto dei linguaggi che si vogliono utilizzare per realizzare il servizio basato su Web
  - Comprende tutte le possibili funzioni che sottendono le operazioni dinamiche
  - Traduce le richieste utente in operazioni che interagiscono con applicativi e/o con il livello logico dei dati
- Implementato mediante una miriade di tecnologie
- Esempi di Web Application Server (WAS)
  - Apache Tomcat
  - JBoss
  - IBM WebSphere Application Server
  - Oracle Application Server

Framework per applicazioni Web

- Alcuni framework per applicazioni Web e le loro caratteristiche salienti
  - Apache Struct
    - Applicazioni in Java (piattaforma JEE), design pattern architetturale MVC, open source
  - WebWork
    - Applicazioni in Java
  - Apache Tapestry
    - Applicazioni in Java, open source
  - Apache Cocoon
    - Open source, separazione dei componenti dell'applicazione
  - Apache Shale
    - Basato su JavaServer Faces, open source
    - JavaServer Faces è una tecnologia Java che permette di semplificare lo sviluppo dell'interfaccia utente dell'applicazione Web

Tecnologia CGI

- Permette al server Web di interfacciarsi con una applicazione e di passargli la richiesta ed i parametri provenienti dal client
  - Logica simile ad un filtro Unix



Tecnologia CGI (2)

- La prima tecnologia per la generazione di contenuti dinamici, sviluppata da NCSA
  - Specifica 1.1: http://hoohoo.ncsa.uiuc.edu/cgi/
- Ancora abbastanza diffusa per la sua flessibilità
- Stabilisce la modalità di interazione tra server HTTP ed applicazione
  - Permette al client di eseguire un'applicazione sulla piattaforma server
  - Permette al server Web di connettersi ad altri servizi per eseguire applicazioni esterne in grado di creare risorse dinamicamente e di poter recuperare informazioni dall'utente, ad es. mediante form
- *Script*: definizione generica di un qualsiasi programma eseguito dal server
  - Tipicamente, ma non necessariamente, implementato in un linguaggio interpretato
- *Gateway*: uno script che fornisce accesso ad un servizio (es. ad un DB), svolgendo un ruolo di interfaccia tra il server Web ed un altro server

Azioni ed implementazione di uno script

- Tradurre l'input fornito dal client (all'interno della richiesta HTTP) in forma comprensibile al servizio a cui lo script si collega (ad es., query nel linguaggio del DB)
- Invocare l'attivazione di altri programmi eseguibili
- Tradurre l'output del programma in una forma comprensibile al client (ad es., il risultato della query in un formato compatibile con il protocollo HTTP)

- Non importa il linguaggio di programmazione con cui lo script è implementato
- L'importante è che lo script sia in grado di leggere da standard input, scrivere su standard output e leggere le variabili d'ambiente

Tecnologia CGI (3)

- Quando un browser richiede un URL che identifica un'applicazione da eseguire, il server HTTP svolge un semplice ruolo: avvia lo script e passa i dati dal browser allo script e viceversa
  - Il passaggio corretto dei dati tra il server HTTP e lo script è garantito da CGI
- Script tipicamente inseriti in una directory specifica
  - /cgi-bin/
- CGI definisce un insieme di *variabili di ambiente* utili all'applicazione
  - Ad es., parametri inviati dal client
- Il server Web deve permettere la gestione delle variabili di ambiente

Passi nell'esecuzione CGI

- Il client specifica nell'URL il nome del programma
- Il server HTTP deve determinare che la risorsa richiesta è un programma
- Il server HTTP deve localizzare il programma e controllare se può essere eseguito
- Il server HTTP lancia in esecuzione lo script, passandogli l'eventuale input fornito dal client
  - Il server decodifica i parametri inviati dal client e assegna valori alle variabili d'ambiente
- L'applicazione può utilizzare le variabili d'ambiente
- L'applicazione stampa la sua risposta sullo stdout
- Il server HTTP deve leggere da stdout l'output dello script e ridirezionarlo verso il client
- Il server HTTP deve chiudere la connessione quando l'esecuzione dello script è terminata

Interazione tra client e server HTTP

- Invocazione diretta di un URL
  - Metodo GET nella richiesta HTTP
  - L'utente specifica l'URL di un CGI o seleziona un collegamento all'URL di un CGI
- Invio di una *fill-in form*
  - Metodi GET o POST nella richiesta HTTP
  - Il browser visualizza il form all'utente mediante una pagina speciale HTML (detta fill-in form), acquisisce i dati di input inseriti dall'utente e spedisce al server il form compilato

Tag form HTML

- L'uso del tag <FORM> permette di acquisire input inserito dall'utente
  - Tramite l'attributo ACTION del tag <FORM> si identifica l'applicazione CGI, passando come parametro l'input inserito dall'utente
  - Invio dei dati tramite l'attributo SUBMIT del tag <INPUT>

```
<FORM ACTION="http://servername/cgi-bin/test"
  METHOD= "GET">
<INPUT TYPE = "TEXT" NAME="var1"  VALUE="">
<INPUT TYPE = "SUBMIT" NAME="Submit"  VALUE="SEND">
```

- L'informazione della fill-in form è inviata al server come:
  - parte dell'URL (attributo METHOD del tag <FORM> uguale a GET)
  - oppure come corpo della richiesta HTTP (attributo METHOD del tag <FORM> uguale a POST)

Invio di parametri con metodo GET

- GET: i parametri sono codificati nell'URL specificato con ACTION
  - Recuperabili dallo script CGI tramite la variabile di ambiente QUERY_STRING

```
<FORM ACTION="http://servername/cgi-bin/test"
  METHOD= "GET">
<INPUT TYPE = "TEXT" NAME="var1"  VALUE="">
<INPUT TYPE = "TEXT" NAME="var2"  VALUE="">
<INPUT TYPE = "SUBMIT" NAME="Submit"
  VALUE="SEND">
```

  - L'URL generata è

```
http://servername/cgi-bin/test?
  var1=value1&var2=value2
```

  - Il server riceve l'input in QUERY_STRING:

```
var1=value1&var2=value2
```


Invio di parametri con metodo POST

- POST: i dati vengono inviati al server in una linea separata, senza limite al numero di caratteri inviati
  - Recuperabili tramite standard input
  - Dimensione dell'input specificata nell'header Content-Length della richiesta HTTP
  - Possibilità di inviare anche informazioni non testuali, specificando il tipo nell'header Content-Type della richiesta HTTP

```
<FORM ACTION="http://servername/cgi-bin/test"
  METHOD= "POST">
<INPUT TYPE = "TEXT" NAME="Var1"  VALUE="">
<INPUT TYPE = "TEXT" NAME="Var2"  VALUE="">
<INPUT TYPE = "SUBMIT" NAME="Submit"
  VALUE="SEND">
```

  - Il server riceve dallo standard input:

```
http://servername/cgi-bin/test?Var1=Value1&
Var2=Value2
```

Variabili d'ambiente

- Una variabile ambiente è un parametro con un nome per trasferire informazioni dal server allo script CGI
  - Non è necessariamente una variabile dell'ambiente del sistema operativo, anche se questa è l'implementazione più comune
  - Rappresentazione canonica: maiuscole_maiuscole
- Principali variabili:
  - SERVER_SOFTWARE: nome e versione del server
  - SERVER_NAME: hostname o indirizzo del nodo server
  - GATEWAY_INTERFACE: versione CGI (uso: CGI/1.1)
  - QUERY_STRING: informazione contenuta dopo il ? nell'URL
  - REQUEST_METHOD: metodo della richiesta HTTP
  - REMOTE_ADDR: indirizzo IP del nodo client che effettua la richiesta
  - REMOTE_USER: se lo script è protetto da autenticazione utente
  - HTTP_USER_AGENT: nome e versione del client
  - CONTENT_TYPE: tipo di input inviato (con metodo POST)
  - CONTENT_LENGTH: dimensione dell'input inviato (con metodo POST)

Input a CGI: riepilogo

Client	***Metodo HTTP***	***Server → CGI***
Invocazione URL	GET (senza ?)	Linea di comando
Invocazione URL	GET (con ?)	Variabili d'ambiente
Invio form	GET (con ?)	Variabili d'ambiente
Invio form	POST	Standard input

Output da CGI

- Quando il server HTTP restituisce al client una risorsa (statica o generata dinamicamente), include nell'header di risposta alcune informazioni sulla risorsa
- Queste informazioni sono inviate indipendentemente dal risultato dell'applicazione CGI
- Alcune header sono generati dal server Web, altri dallo script CGI
- Lo script CGI può aggiungere tre tipi di informazioni:
  - **Content-type**: formato MIME della risorsa
  - **Location**: URL alternativa per localizzare la risorsa
    `Location: URL`
  - **Status**: risposta HTTP
    `Status: XXX message`

Esempio in C

- Stampa "hello, world."

```
#include <stdio.h>
void main() {
        printf("Content-type: text/html\n\n");
        printf("<html>\n");
        printf("<head><title>CGI Output</title></head>\n");
        printf("<body>\n");
        printf("<h1>Hello, world.</h1>\n");
        printf("</body>\n");
        printf("</html>\n");
        exit(0);
}
```

Esempio in Perl

- Stampa la parola inviata allo script CGI implementato in Perl e l'indirizzo IP del client
  - Usa il modulo CGI del Perl

```
#!/usr/bin/perl -w
use CGI;
$query = new CGI;
$secretword = $query->param('w');
$remotehost = $query->remote_host();
print $query->header;
print "<p>The secret word is
  <b>$secretword</b> and your IP is
  <b>$remotehost</b>.</p>";
```


Configurazione di Apache per CGI

- Si usa la direttiva ScriptAlias

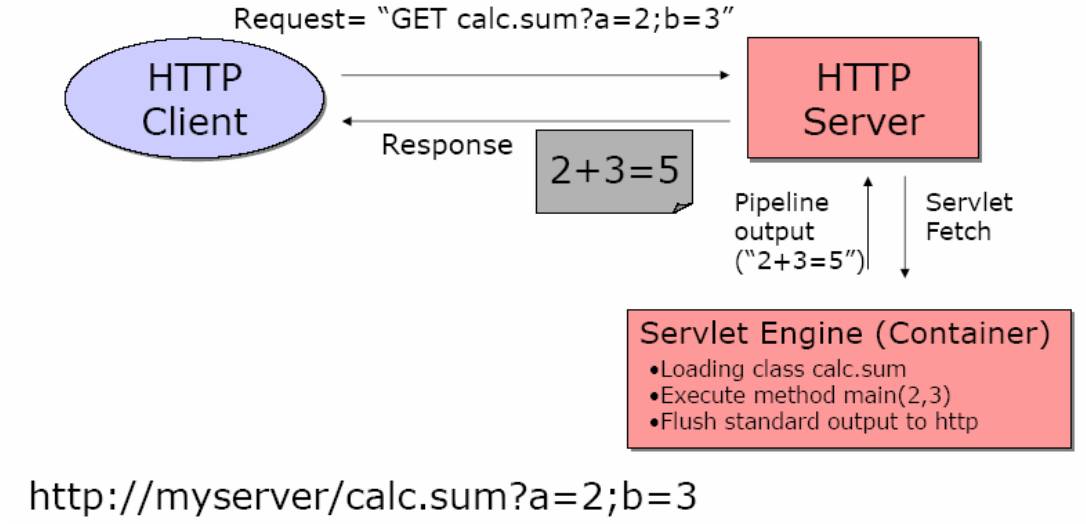
  ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/
  - Programmi CGI spesso confinati nella directory di default per evitare vulnerabilità di sicurezza
- In alternativa, per abilitare una directory arbitraria si usano le direttive AddHandler e Options

  AddHandler cgi-script .cgi

  <Directory /usr/local/apache2/htdocs/somedir>
  Options +ExecCGI
  </Directory>
- Riferimento: http://httpd.apache.org/docs/2.2/howto/cgi.html

Java servlet

- Una servlet è un componente software scritto in Java, gestito da un *container*, in grado di ricevere in modo strutturato i parametri e generare dinamicamente la risorsa richiesta



Java servlet (2)

- Una servlet interagisce con un client in base al paradigma richiesta/risposta
- Le servlet sono una tecnologia cross-platform, in quanto le API non fanno assunzioni sull'ambiente di esecuzione o sul protocollo
  - L'uso più diffuso è con il protocollo HTTP

Vantaggi delle servlet

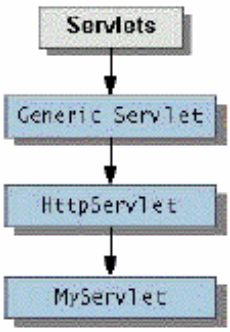
- Portabilità
  - Indipendenti dalla piattaforma, solo JVM
- Integrazione con altri componenti ed API Java (ad es. JDBC)
- Convenienza
  - Molte funzionalità di alto livello
- Efficienza
  - Uso di thread Java anziché processi
- Persistenza della Java servlet
  - Dopo il caricamento, la servlet rimane in memoria e può rispondere a richieste multiple mantenendo alcune informazioni (ad es., connessione ad un DB)
- Gestione delle sessioni
- Robustezza
- Sicurezza (sandbox)

I compiti di una servlet

- Leggere i dati impliciti inviati dal client (nella linea di richiesta e negli header della richiesta HTTP)
- Leggere i dati espliciti inviati dal client (nel form)
- Generare il risultato
- Inviare i dati impliciti al client (linea di risposta e gli header della risposta HTTP)
- Inviare i dati espliciti al client (ad es., il file HTML)

Java servlet (3)

- Le servlet usano le classi e le interfacce definite nei package javax.servlet e javax.servlet.http
- Interfaccia pubblica Servlet
- Una servlet generica estende la classe astra javax.servlet.GenericServlet
  - Generica: indipendente dal protocollo utilizzato
- Una servlet HTTP estende la classe astratta javax.servlet.http.HttpServlet
- Interfacce pubbliche generiche ServletRequest e ServletResponse
- Interfacce pubbliche per HTTP HttpServletRequest e HttpServletResponse
  - Forniscono alla servlet HTTP informazioni rispettivamente sulla richiesta e sulla risposta
- Interfaccia pubblica per HTTP HttpSession
  - Fornisce i meccanismi per identificare una sessione dell'utente e per memorizzare informazioni sull'utente

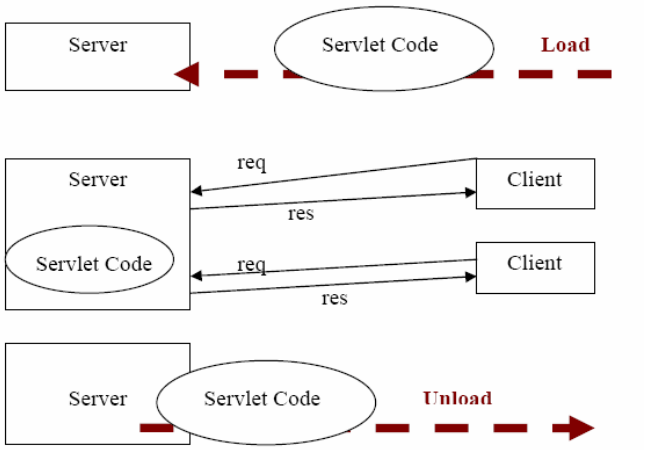


Ciclo di vita di una servlet

- Il servlet container carica ed inizializza la servlet
  - Mediante il metodo init()
  - Invocato solo quando la servlet è caricata per la prima volta, non viene chiamato per ogni richiesta (a meno che la servlet non venga ricaricata)
  - E' possibile costruire un metodo init in overriding
- La servlet gestisce zero o più richieste dai client
  - Mediante il metodo service(ServletRequest req, ServletResponse res)
  - Il metodo service legge la request e produce una response dai suoi due parametri ServletRequest e ServletResponse
  - Chiamato un nuovo thread dal servlet container per gestire ogni richiesta; in service() avviene il dispatching verso doGet(), doPost(), doXxx() in base al metodo HTTP nella richiesta
  - Non in overriding!
  - doGet(), doPost(), doXxx(): metodi di HttpServlet per gestire le richieste HTTP con metodo GET, POST, …
  - E' possibile costruirli in overriding per ottenere il comportamento desiderato

Ciclo di vita di una servlet (2)

- Il servlet container rilascia l'istanza della servlet
  - Mediante il metodo destroy()
  - Non viene chiamato per ogni richiesta
  - Permette alla servlet il clean-up di qualsiasi risorsa (file aperti, connessioni ad DB, …) prima che la servlet sia scaricata
  - E' possibile costruire un metodo destroy in overriding



HttpServletRequest e HttpServletResponse

- HttpServletRequest
  - Consente l'accesso agli header HTTP
    - Accesso ai metodi di richiesta (GET, POST, …)
    - Accesso ai cookie
    - Accesso ai parametri della richiesta
- HttpServletResponse
  - Costruzione dell'header di risposta (ad es., content type)
  - Costruzione della risposta
    - testo, html: getWriter()
    - binario: getOutputStream()

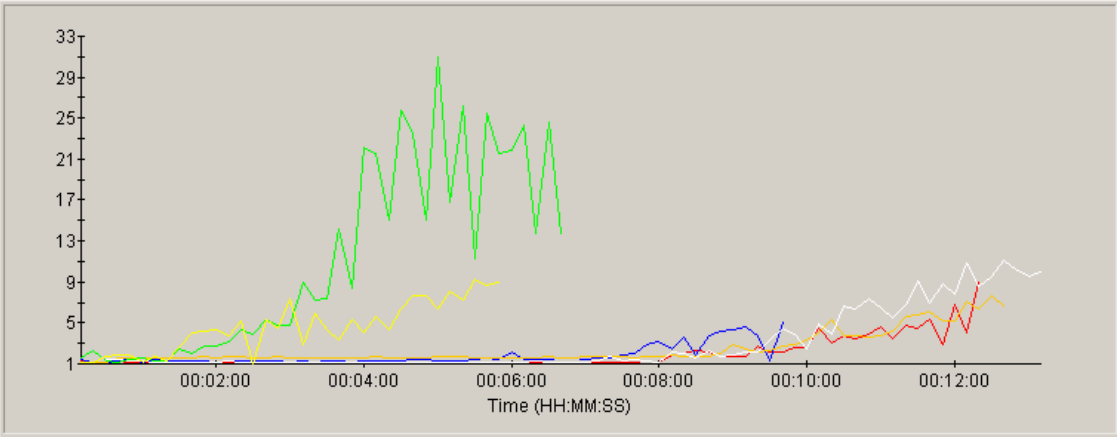
Esempio di Java servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
          HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String docType = "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD
          HTML 4.0 " + "Transitional//EN\">\n";
    out.println(docType + "<HTML>\n" + "<HEAD><TITLE>
          This is the output from HelloServlet</TITLE></HEAD>\n" +
          "<BODY BGCOLOR=\"#FDF5E6\">\n" +
          "<H1>Hello</H1>\n" + "</BODY></HTML>");
  }
}
```


Servlet container

- Servlet container: consente l'esecuzione della servlet e gestisce l'interazione con il server Web
- Tipologie di servlet container
  - Autonomo
    - Componente software separata dal server Web, che include un supporto nativo per le servlet
    - Comunicazione tra le due entità gestita da un connector
    - Ad es.: Apache Tomcat, Jetty
  - Aggiuntivo
    - Integrabile direttamente in un server Web come modulo
    - Ad es: Macromedia JRun
  - Incorporato in un application server
    - Può anche essere usato in modo indipendente dal server Web, se nell'application server è incluso il servizio HTTP
    - Ad es.: Apache Tomcat, Caucho Resin, Oracle Application Server, IBM WebSphere Application Server

Prestazioni dei servlet container

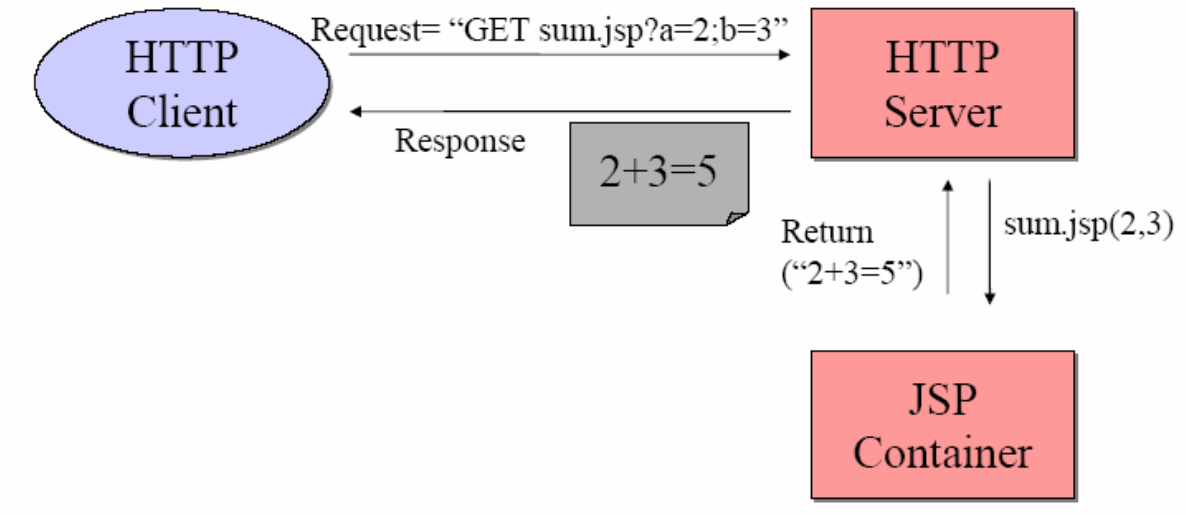


Test	Statistic Name	Color	Scale
Tomcat	Average Duration (ms)	Red	0.0010
Orion	Average Duration (ms)	Blue	0.0010
Resin	Average Duration (ms)	Orange	0.0010
JRun	Average Duration (ms)	White	0.0010
Websphere	Average Duration (ms)	Green	0.0010
Jetty	Average Duration (ms)	Yellow	0.0010

Tempo di risposta per una pagina (sec) all'aumentare del carico

Fonte (2004):
http://www.webperformanceinc.com/library/reports/ServletReport/

JSP



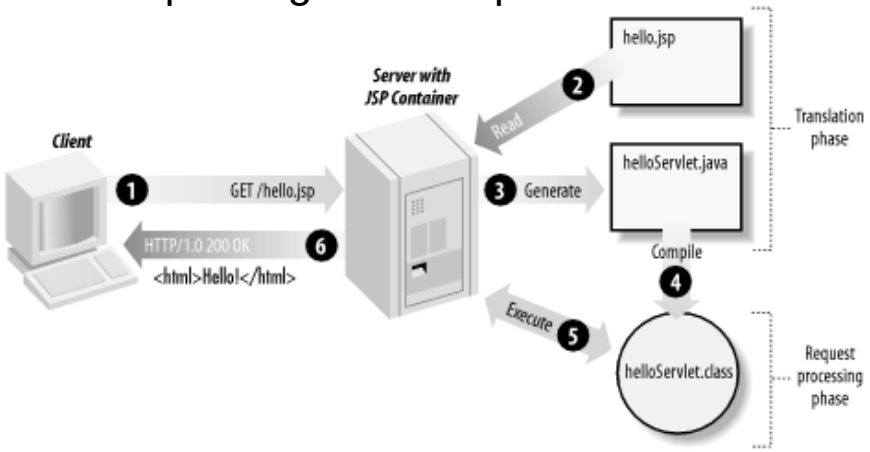
http://myserver/sum.jsp?a=2;b=3

JSP: caratteristiche

- Linguaggio di scripting HTML-embedded server-side
- Una pagina JSP contiene sia HTML sia codice
  - Il client effettua la richiesta per la pagina JSP
  - La parte HTML viene passata al client senza trasformazione
  - Il codice viene eseguito sul server e viene generato il contenuto dinamico
  - La pagina così creata viene inviata al server
- Condivide molte caratteristiche con Java servlet
  - Integrazione con Java
  - Sicurezza
  - Portabilità
- Maggiore semplicità di sviluppo rispetto a Java servlet
- Possibilità di creare librerie di tag JSP che fungono da estensioni dei tag standard
  - Librerie di tag custom e JSP Standard Tag Library (JSTL)

JSP: caratteristiche (2)

- L'esecuzione delle JSP è compito del *JSP container*
- Le pagine JSP vengono trasformate in Java servlet (classi di implementazione della pagina) dal JSP container
- Le classi di implementazione sono conservate in una cache per migliorare le prestazioni



Esempio: JSP

```
<HTML>
<HEAD>
<TITLE>hello jsp</TITLE>
<!-- the variable, message, is declared and initialized -->
<%! String message = "Hello, World, from JSP"; %>
</HEAD>
<BODY>
<!-- the value of the variable, message, is inserted between h2 tags -->
<h2><font color="#AA0000"><%= message%></font></h2>
<h3><font color="#AA0000">
<!-- the java.util.Date method is executed and the result inserted
    between h3 tags -->
Current time: <%= new java.util.Date() %>
</font></h3>
</BODY>
</HTML>
```

Dichiarazione variabile

Espressione

Espressione

Esempio: Java Servlet corrispondente

```
public class HelloWorldServlet implements Servlet {
  public void service(ServletRequest request, ServletResponse
     response)
     throws ServletException, IOException {
     response.setContentType("text/html");
     PrintWriter out = response.getWriter();
     out.println("<html><head><title>Hello</title></head>");
     out.println("<body><h2>Hello, World, from Java Servlet</h2>");
     out.println("It's" + (new java.util.Date()).toString());
     out.println("</body></html>);
  }
}
```

Nota: non è il codice della Java Servlet generata dal JSP container

Elementi JSP

- Una pagina JSP è un'alternanza di:
  - Modelli di testo (HTML o altro)
  - Elementi JSP
- JSP permettono di utilizzare elementi di programmazione differenti (HTML, EJB, servlet)
- **Elementi di direttiva**
  - Informazioni sulla pagina, sono gli stessi per ogni richiesta
  - Formato: *<%@ direttiva {attributo="valore"}%>*
- **Elementi di azione**
  - Gestione di informazioni disponibili al momento della richiesta
  - Forniscono un livello di astrazione per incapsulare agevolmente compiti comuni; generalmente impiegati per creare o manipolare oggetti, soprattutto JavaBeans
- **Elementi di scripting**
  - Per includere porzioni di codice vero e proprio (tipicamente codice Java): scriplet generica (*<% code %>*), espressioni *<%= expression %>*) e dichiarazioni di variabili (*<%! code %>*)

Apache Tomcat

- Apache Tomcat è l'implementazione di riferimento, free e open-source delle tecnologie Java Servlet e JavaServer Pages
- Sviluppato dalla Apache Software Foundation
  - http://tomcat.apache.org/
  - Ultima release: Tomcat 6 (implementa le specifiche Java Servlet 2.5 and JavaServer Pages 2.1)

Apache Tomcat (2)

- In modalità standalone: può essere usato in modo indipendente dal server Web Apache
  - Include il supporto del protocollo HTTP
  - Tomcat serve anche pagine statiche
- In alternativa, può essere usato insieme al server Web Apache
  - Tramite il connettore mod_jk
  - Apache serve pagine statiche, Tomcat solo JSP
  - Protocollo AJP per gestire la comunicazione tra Apache e Tomcat
    - Comunicazioni su connessioni TCP persistenti in formato binario
  - Con Apache di fronte a Tomcat aumentano le possibilità di configurazione sofisticate
    - Ad es., URL rewriting, bilanciamento del carico, …

Riferimenti per Java servlet e JSP

- http://java.sun.com/products/servlet/
- http://java.sun.com/products/jsp/
- http://www.coreservlets.com/

PHP

- Linguaggio di scripting HTML-embedded server-side
- In realtà, è un linguaggio general purpose
  - Ad es., sviluppo di applicazioni shell (o desktop) in PHP (SAPI CLI o Command Line Interface), applicazione GUI (estensione PHP-GTK)
- Molto usato (20 milioni di domini ad ottobre 2006)
  - Sistema LAMP (Linux Apache MySQL PHP)
- L'esecuzione del PHP è compito del *PHP preprocessor*
  - Elabora la pagina eseguendo il codice PHP prima di inviarla al client
- Riferimento
  - http://www.php.net/

Caratteristiche di PHP

- Linguaggio non strettamente tipato
- Sintassi di base simile a Perl e C
- Costrutti base per
  - Salto condizionale (if, elseif, else)
  - Cicli di iterazione (for, while)
  - Definizione di funzioni
- Non necessaria la dichiarazione esplicita di variabili
  - Variabile inizia con $
  - L'engine tiene traccia del primo tipo di dato assegnato (intero, reale, stringa, array)
- Array associativi
  - Contengono coppie nome-valore; es.:

```
$role = array {
"Valeria" => "teacher",
"You" => "student",
};
$role["Your neighbor"] = "student";
$me = $role["Valeria"];
```

Caratteristiche di PHP (2)

- Per default, uno script PHP genera un file HTML: si può fornire anche un altro tipo di contenuto

```
<? $len = filesize("foo.pdf");
header("Content-type: application/pdf");
header("Content-Length: $len");
readfile("foo.pdf"); ?>
```

- Supporto per gestire il controllo degli accessi
  - Variabili standard $PHP_AUTH_USER e $PHP_AUTH_PW
- Interazione con i cookie
  - Cookie standard PHPSESSID e funzione setcookie()
- Interazione con un DB
  - Supportati molti DB; per ognuno funzioni PHP proprie

Esempio PHP

```
<html>
<head>
<title>PHP Hello</title>
</head>
<body>
<? echo "Hello world!"; ?>
<? if(strstr($HTTP_USER_AGENT,"MSIE")) { ?>
    <center><b>You are using Internet Explorer</b></center>
<? } else { ?>
    <center><b>You are not using Internet Explorer</b></center>
<? } ?>
</body>
</html>
```

Apache e PHP

- Interprete PHP come modulo di Apache (mod_php)
  - Nel file di configurazione httpd.conf di Apache

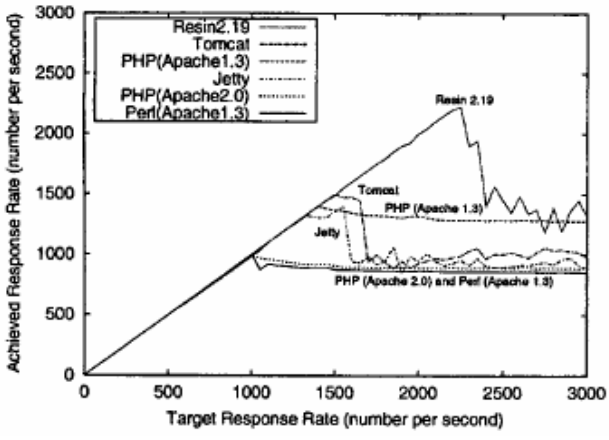
    LoadModule php5_module modules/libphp5.so

    AddType application/x-httpd-php .php

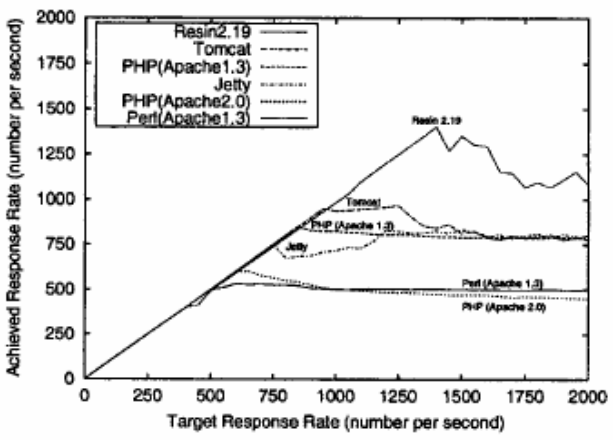
    DirectoryIndex index.php index.htm index.html
- Il modulo ridefinisce la fase di fixup nel servizio delle richieste per risorse di tipo x-httpd-php
  - La fase di fixup avviene subito prima della fase di risposta
- Durante la fase di fixup viene invocato l'interprete PHP
- In questa modalità di funzionamento, PHP deve essere integrato nel Web server
- Necessità di avere logica di presentazione e di applicazione insieme

Prestazioni PHP e JSP a confronto

- Per sole richieste statiche, le prestazioni di Apache sono nettamente migliori di quelle dei servlet container Tomcat, Jetty e Resin
- Per richieste dinamiche (dimensione 2 KB):



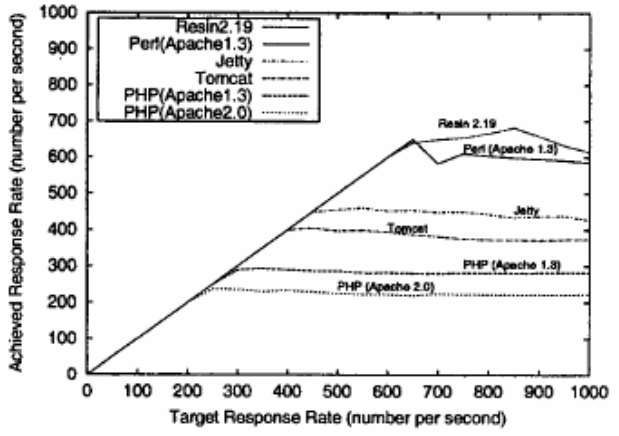
Senza accesso al DB



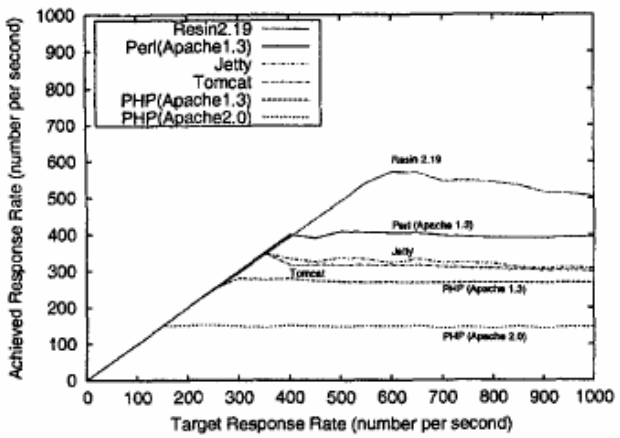
Con accesso al DB

Prestazioni PHP e JSP a confronto

- Per richieste dinamiche (dimensione 64 KB):
  - Con accesso al DB, riduzione fino ad un fattore pari a 8 del throughput di picco rispetto a richieste statiche



Senza accesso al DB



Con accesso al DB

Fonte: L. Titchkosky, M. Arlitt, C. Williamson, "A Performance Comparison of Dynamic Web Technologies", ACM Performance Evaluation Review, Dec. 2003.

Nota: PHP5 ha prestazioni migliori di PHP4 (usato nel confronto)

Middleware

- Middleware = "software che sta in mezzo"
  - E quindi, … grande confusione sul termine …
- ObjectWeb Consortium (http://middleware.objectweb.org/): "In un sistema distribuito, il middleware è il livello software che si trova tra il sistema operativo e le applicazioni su ogni lato del sistema"
- Middleware per sviluppo di sistemi distribuiti
  - Software di supporto alle applicazioni distribuite tra il supporto di comunicazione e le applicazioni:
    - RPC (Remote Procedure Call), RMI (Remote Method Invocation), …
- Middleware come collante per realizzare sistemi informatici complessi

Middleware

- Il middleware permette di mascherare molti dettagli dipendenti dalle diverse piattaforme hw/sw e la distribuzione, facilita l'integrazione e consente allo sviluppatore di applicativi di lavorare sul problema ad un più alto livello di astrazione
- Essendo i sistemi veramente complessi, il primo (e spesso) l'unico obiettivo è riuscire a *farli funzionare*
- Quindi, la maggior parte del processo di sviluppo è speso nell'integrazione delle componenti e nel farle funzionare
- Le prestazioni sono spesso considerate un fattore secondario

Due tecnologie middleware

- SUN Java Platform, Enterprise Edition (Java EE)
  - Open source
  - Ultima versione: Java Platform, Enterprise Edition 5 (Java EE 5)
  - In precedenza nota come Java 2 Enterprise Edition (J2EE)
  - Riferimento: http://java.sun.com/javaee/
- Microsoft .NET
  - Proprietaria

Obiettivi di Java EE

- Creare uno standard che consenta alle applicazioni basate su Web di essere portabili tra server
  - Paradigma *Write Once, Run Everywhere*
- Dare al server il controllo del ciclo di vita delle componenti e delle altre risorse in termini di:
  - Scalabilità
  - Concorrenza
  - Gestione flessibile delle transazioni
  - Gestione delle sessioni utente
  - Sicurezza

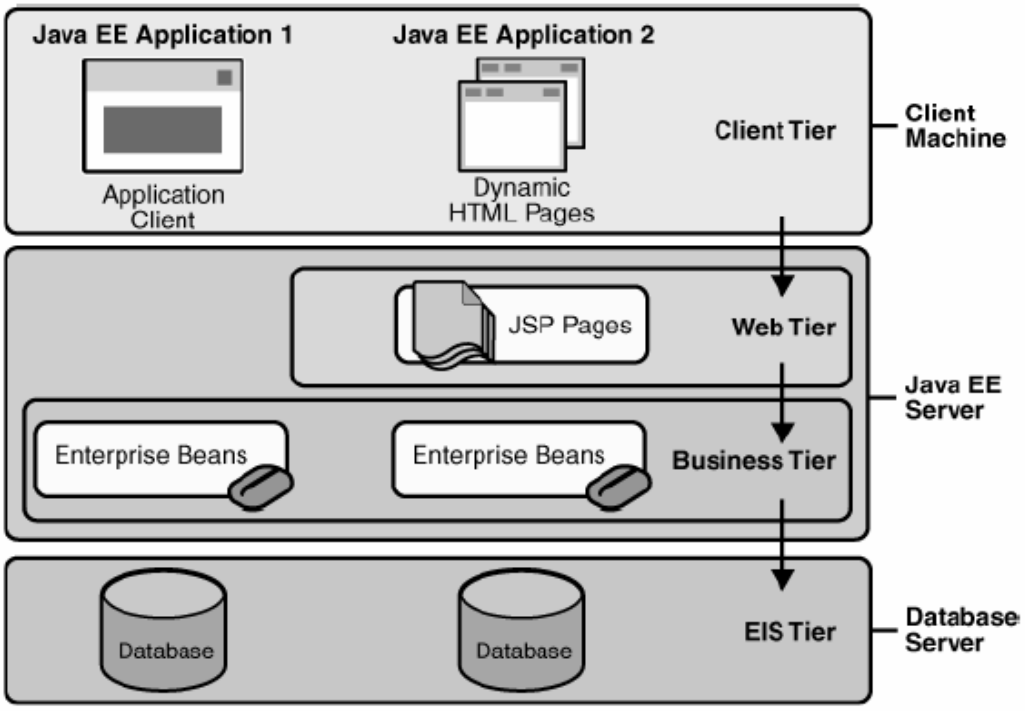
Caratteristiche di Java EE

- Java EE fornisce
  - un'architettura basata su **componenti** per il progetto, lo sviluppo e l'assemblaggio di applicazioni *enterprise e mission critical* basate su Web
  - un modello di sviluppo di applicazioni distribuite multi-livello (architetture *n-tier*) indipendenti dalla piattaforma
- Enterprise (impresa): organizzazione di business
- Enterprise software application: applicazione SW che facilita la gestione delle attività di impresa
  - interagire con clienti e partner via Internet
  - facilitare l'interazione tra le varie parti di un'impresa, eventualmente distribuite geograficamente
  - gestione del business: resource planning, gestione inventari, ...
- Java EE non è un linguaggio, ma un insieme di diverse tecnologie software (nuove e preesistenti)

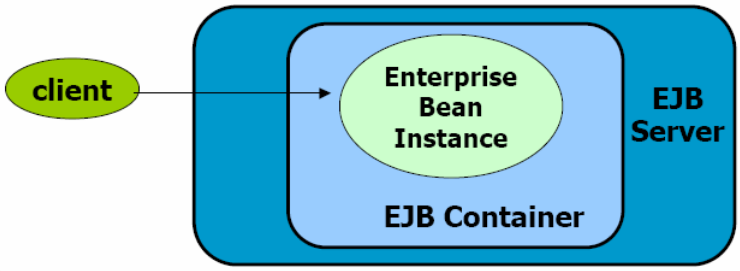
Componenti Java EE

- E' un componente software auto-contenuto (*building block*), riusabile e che viene assemblato con altri componenti per formare un'applicazione enterprise distribuita
- Un componente è eseguito all'interno di un contesto detto **container**
  - In pratica, il supporto a run-time
- Componenti fondamentali nelle specifiche Java EE:
  - Applet Java e applicazioni client
    - Eseguite sulla macchina del client
  - Tecnologie Java Servlet e JSP
    - Componenti Web eseguite sul server
  - Enterprise Java Beans (EJB)
    - Componenti business eseguite sul server

Il modello applicativo Java EE

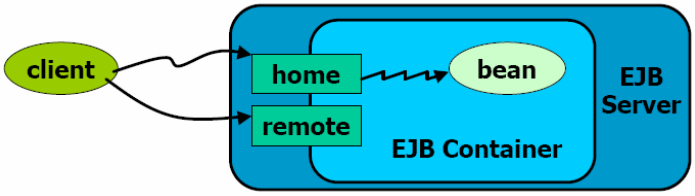


EJB: architettura



- **EJB server**
  - Application server
  - E' il motore che permette di usare i componenti da parte di client remoti
  - Fornisce ai container servizi di:
    - Gestione delle risorse del sistema
    - Mantenimento dello stato
    - Gestione/attivazione di processi/thread
    - Sicurezza
  - Gestisce le politiche di ottimizzazione delle risorse
  - Esempi: JBoss (open source), IBM WebSphere Application server, BEA Web Logic Server

EJB: architettura (2)



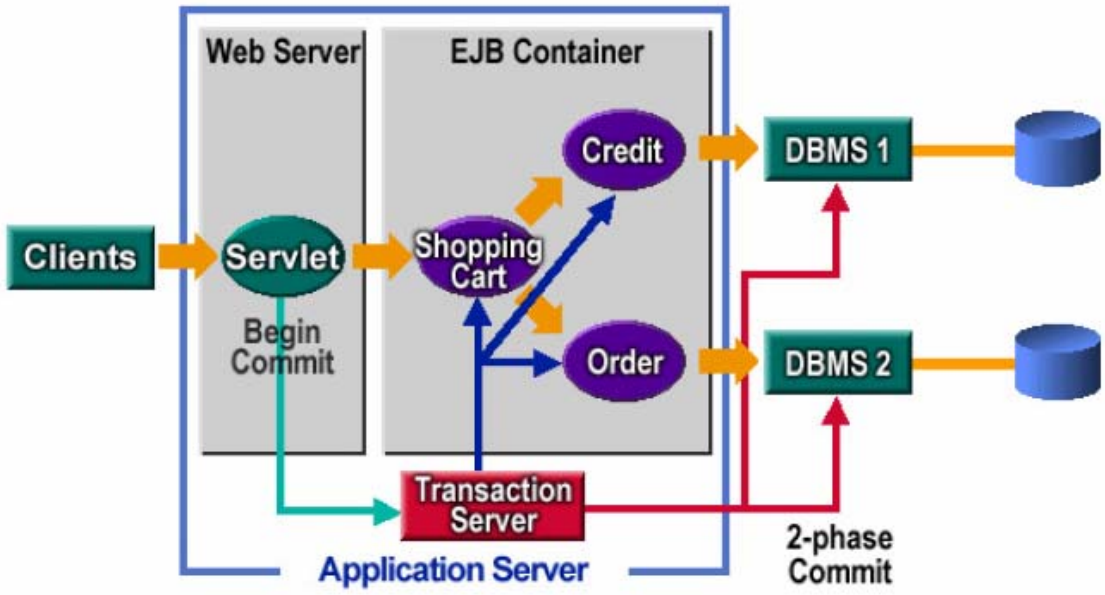
- **EJB container**
  - Ospita ed esegue gli EJB
- **Enterprise Bean**
  - Componente vero e proprio
  - Classe Java specializzata in cui risiede la logica di business dell'applicazione
  - Deve essere installato (deployment) sull'application server
  - Usa i servizi offerti dall'ambiente di esecuzione EJB:
    - Transazioni
    - Sicurezza
    - Persistenza
  - Due tipologie principali di EJB
    - Session Bean (stateful o stateless): oggetto distribuito con o senza stato (mantiene o meno lo stato della conversazione), implementa la logica di business
    - Message Driven Bean: oggetto distribuito per la gestione di messaggi asincroni

EJB: vantaggi

- Semplificazione del processo di sviluppo
- Riusabilità del codice
- Modularità
- Robustezza
- Gestione automatica di:
  - transazioni (Commit, Rollback e Recovery)
  - scalabilità
  - sicurezza
- Alte prestazioni
  - bilanciamento dinamico dei carichi di lavoro
  - caching delle connessioni al database

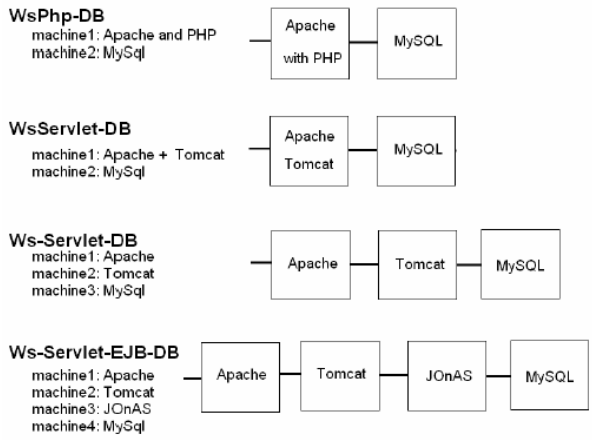
Esempio di applicazione Web-based

- Applicazione multi-tier con accesso Web, business logic e DB

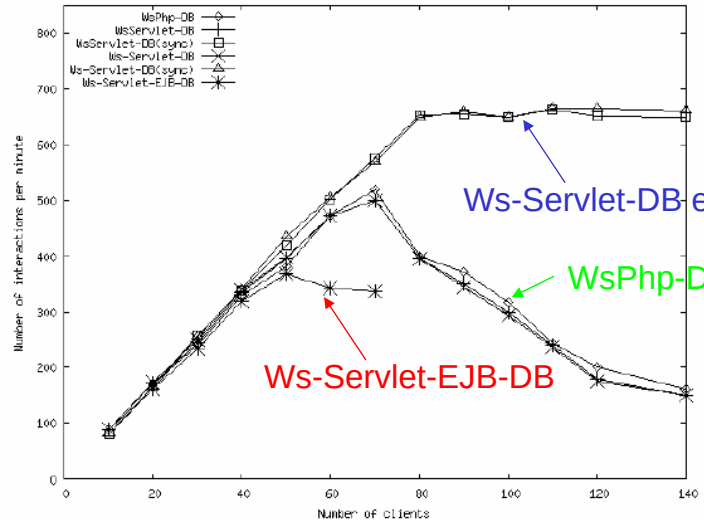


Prestazioni di architetture middleware

Configurazioni esaminate



Benchmark TPC-W, shopping mix



Ws-Servlet-DB e WsServlet-DB

WsPhp-DB

Ws-Servlet-EJB-DB

Fonte: E. Cecchet *et al.*, “Performance Comparison of Middleware Architectures for Generating Dynamic Web Content”, Proc. of Middleware 2003.