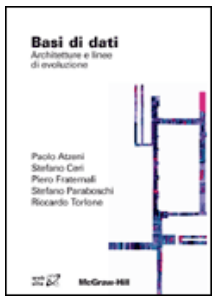
Database & WWW

Capitolo 4



Basi di dati – Architetture e linee di evoluzione
P. Atzeni, S. Ceri, P. Fraternali, S. Paraboschi, R. Torlone

Sommario

- Protocollo HTTP
- CGI
- Java Servlet
- Server-side scripting e librerie di tag JSP, tag eseguibili e ASP.NET
- PHP

Motivazioni

- Produrre pagine "al volo" in base alle esigenze dell'utente e da contenuti strutturati (p.e. database)
- Scripting & componenti client-side non bastano
- Soluzioni:
  - Architetture per produrre contenuti dinamici a lato-server

Protocollo HTTP

- HTTP è un protocollo di comunicazione utilizzato su Internet
- Protocollo client–server (browser–HTTPserver)
- Si basa su scambio di messaggi tra client e server:
  - HTTP **request** messages (da client a server)
  - HTTP **response** messages (da server a client)
- Utilizzato per trasmettere qualunque tipo di risorsa (non solo files), ma in pratica vengono inviati file di testo statici o file dinamici prodotti da server-side scripts

HTTP Request

- Il client stabilisce una connessione con il server ed invia un messaggio di richiesta per accedere ad una risorsa

```
GET index.html HTTP/1.1
User-Agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
```

- La struttura di una richiesta HTTP consiste in varie linee di testo ASCII con una linea vuota finale
  - **Request** line: HTTP Method Field, URI field, HTTP version field
    - Metodi: **GET**, **POST**
  - **User-Agent** line: indica quale tipo di client fa la richiesta
  - **Accept** line: indica quale tipo di file il client è in grado di accettare

HTTP Response

```
HTTP/1.1 200 OK
Date: Mon, 03 May 2004 21:00:00 GMT
Content-Length: 1024
Content-Type: text/html
Last-Modified: Wed, 28 Apr 2004 11:30:00 GMT
<HTML> ... </HTML>
```

- Il server risponde alle richieste del client con un messaggio suddiviso in tre parti
  - linea di **stato**: HTTP version, Status code, Server message
  - varie linee **Header**: varie informazioni (data, tipo di contenuto, lunghezza del messaggio, ultima modifica, ...)
  - **corpo** del messaggio: La risorsa richiesta dal client

Status Code

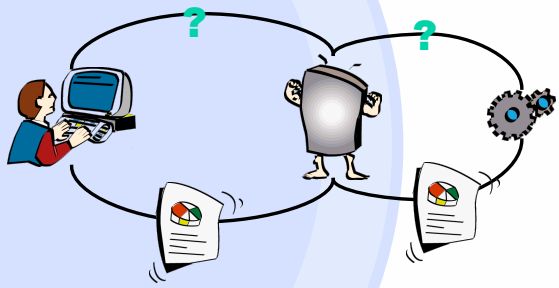
- **200 OK**
  Tutto a posto: il messaggio di risposta contiene la risorsa richiesta
- **400 Bad Request**
  Errore generico che indica che il server non è in grado di far fronte alla richiesta
- **404 Not Found**
  La risorsa richiesta non è stata trovata
- **505 HTTP Version Not Supported**
  La versione del protocollo utilizzata dal client non è supportata dal server

Proprietà di HTTP

- HTTP non prevede metodi per identificare l'utente
- Ogni richiesta è trattata come un evento isolato
- Una nuova richiesta non ricorda nulla della precedente
- Non esiste il concetto di sessione interattiva dell'utente
- Impossibile fare applicazioni personalizzate

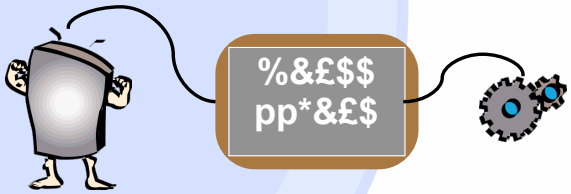
Common Gateway Interface

- Interfaccia che consente al Web Server di eseguire applicazioni esterne in grado di creare pagine dinamicamente



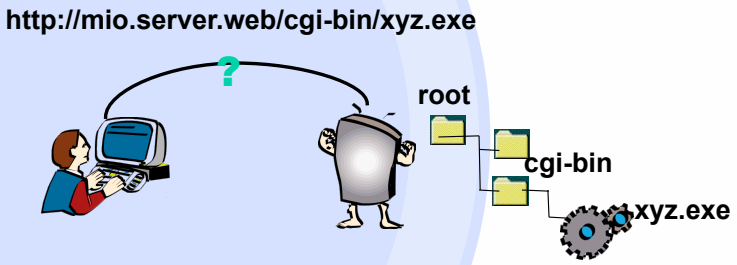
Caratteristiche di CGI

- Non e':
  - un linguaggio di programmazione
  - un protocollo di comunicazione
- Definisce solo un insieme di variabili di ambiente utili alla applicazione (ad es. parametri inviati dal client)



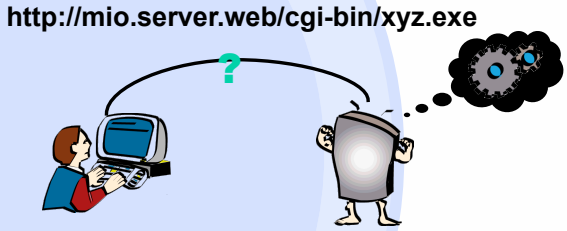
Invocazione

- Il cliente specifica nell'URL il nome del programma da eseguire
- Il programma deve stare in una posizione precisa (di solito il direttorio cgi-bin)



Esecuzione

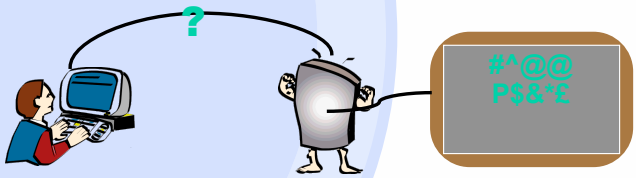
- Il server riconosce dall'URl che la risorsa richiesta dal cliente e' un eseguibile



Esecuzione

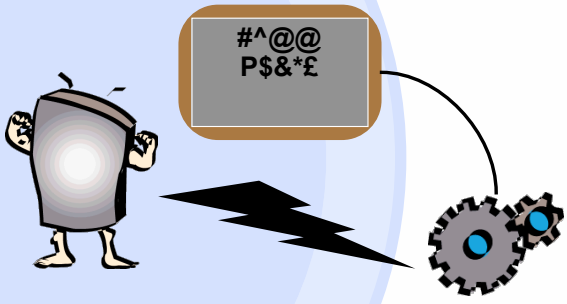
- Il server decodifica i parametri inviati dal cliente e riempie le variabili d'ambiente
  - es: request_method, query_string, content_length, content_type

http://mio.server.web/cgi-bin/xyz.exe?#^@@



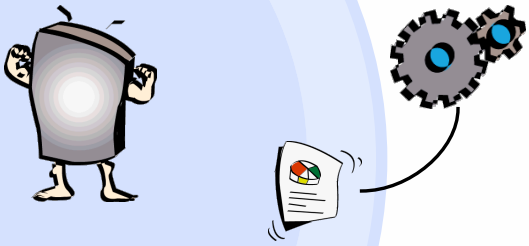
Esecuzione

- Il server lancia in esecuzione l'applicazione richiesta



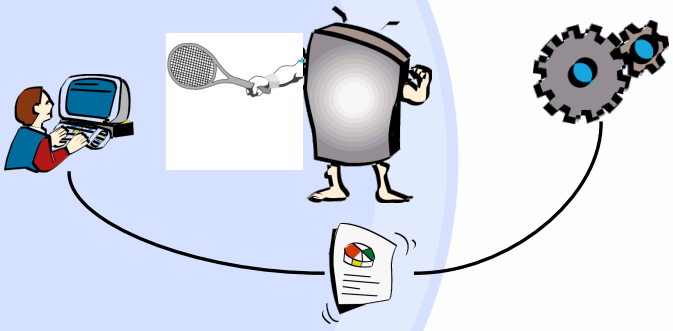
Esecuzione

- L'applicazione stampa la sua risposta sullo standard output



Esecuzione

- Il server ridireziona lo standard output sulla rete e quindi verso il client



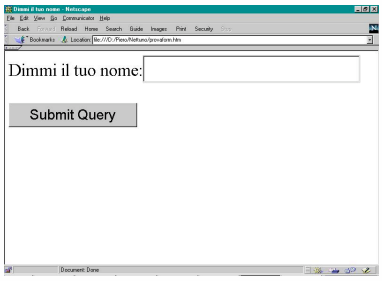
Invio di parametri a un programma CGI

- Il client puo' usare due metodi:
  - GET
  - POST
- **GET**: i parametri sono codificati nell'URL
  http://www.mioserver.it/cgi-bin/xyz?par=val
- **POST**: i parametri sono spediti al server separatamente, usando il body del messaggio di richiesta HTTP
- NB: il metodo POST richiede l'uso di un costrutto HTML chiamato FORM

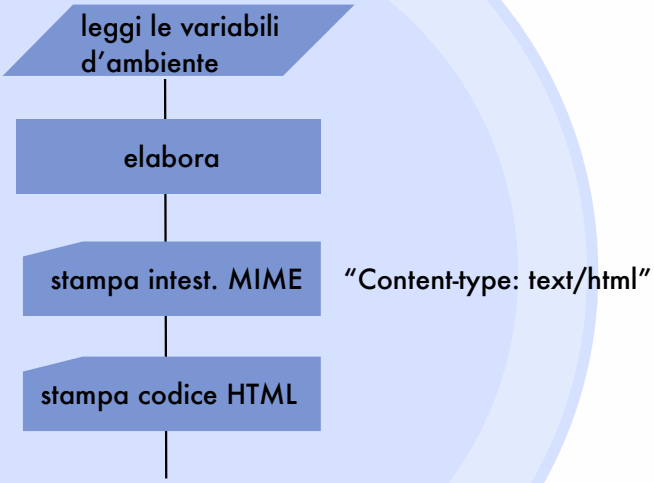
FORM HTML

- Esempio: invio al server il nome dell'utente

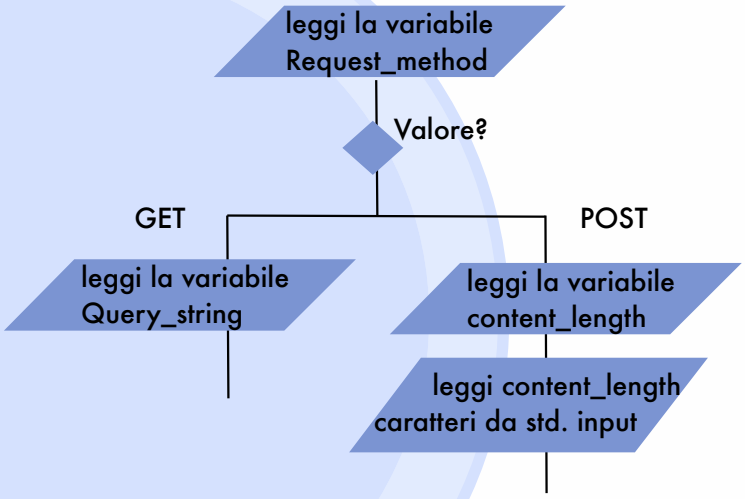
```
<form action=http://www.mysrvr.it/cgi-bin/xyz.exe
  method=post>
<p>Dimmi il tuo nome:
<input type=text name="chisei" ></p>
<input type=submit >
</form>
```



Struttura di un programma CGI



Decodifica dei parametri



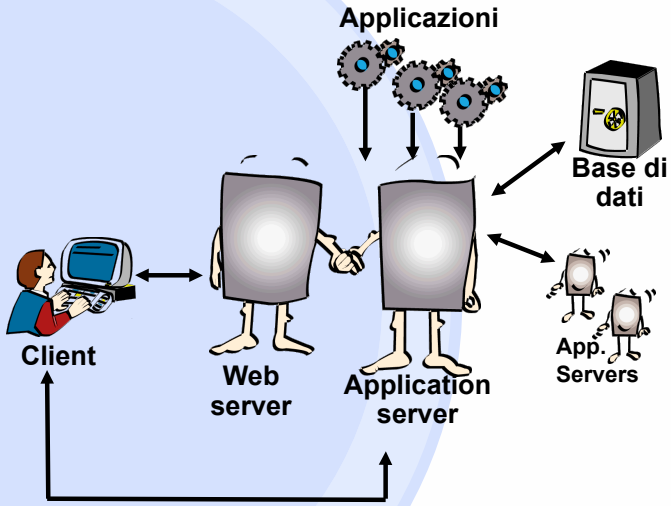
Revisione critica di CGI

- Il web server genera un nuovo processo cgi ad ogni richiesta
- Il processo viene terminato alla fine del computo della risposta
- Altissimo sovraccarico di esecuzione per la creazione e distruzione di processi
- Impossibile:
  - Tenere informazioni sulla sessione dell'utente in memoria centrale (serve un database)
  - Tenere allocate risorse condivise tra più richieste o più utenti (es. pool di connessioni a database)

Obiettivi delle architetture server side

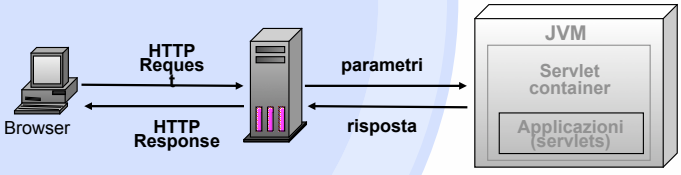
- Migliori prestazioni: creare processi solo in fase di inizializzazione del sistema e riusare un pool di processi pronti
- Mantenimento dello stato: sfruttare la persistenza del processo per mantenere informazioni sulla sessione dell'utente (stateful application)
- Condivisione delle risorse: mantenere allocate le risorse condivise
- Scalabilità: aumentare il numero di processi applicativi all'aumentare del traffico

Architetture "application server"



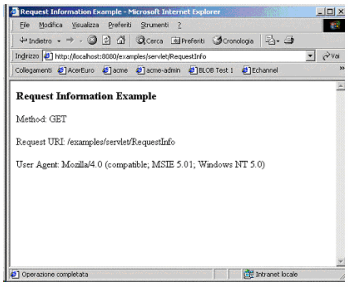
Java Servlet

- **Servlet container**: un programma Java che fornisce un ambiente persistente di esecuzione per applicazioni Web ed espone le caratteristiche del Web server come oggetti Java
- **Servlet**: l'analogo di uno script CGI nel mondo Java



Esempio di servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class RequestInfo extends HttpServlet {
   public void doGet(HttpServletRequest request,
                     HttpServletResponse response)
                     throws IOException, ServletException {
     response.setContentType("text/html");
     PrintWriter out = response.getWriter();
     out.println("<HTML>");
     out.println("<HEAD>");
     out.println("<TITLE>Request Information Example</TITLE>");
     out.println("</HEAD>");
     out.println("<BODY>");
     out.println("<H3>Request Information Example</H3>");
     out.println("Method:    " + request.getMethod());
     out.println("<BR>");
     out.println("Request URI:    " + request.getRequestURI());
     out.println("<BR>");
     out.println("User Agent:"+request.getHeader("User-Agent"));
     out.println("</BODY>");
     out.println("</HTML>");
    }
}
```



Classi/interfacce di utilità

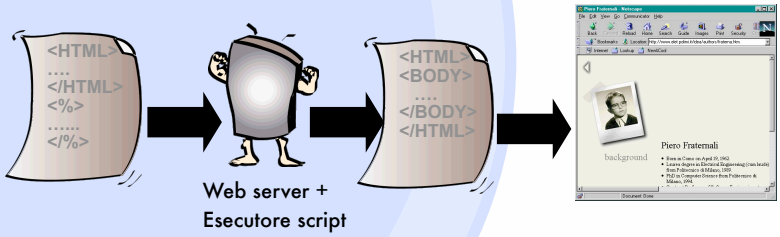
- HTTPServletRequest
  - METODI: getHeader(), getMethod(), getQueryString(),..
- HTTPServletResponse
  - METODI: getWriter(), setContentLenght, setContenType(), setHeader(), setStatus(), ..
- HTTPSession
  - METODI: getAttribute(), invalidate(), isNew, setMaxInactiveInterval()
- Cookie
  - METODI: setValue(), GetValue(), getName(), setMaxAge(),..

Valutazione

- Astrazione object-oriented delle proprietà di un server HTTP esteso (request, response, session ecc)
- Programmazione complessa perchè bisogna produrre sia il testo statico che i contenuti dinamici
- La presentazione HTML è mescolata al codice applicativo
- **Esigenza**: serve una soluzione più semplice per produrre pagine in modo dinamico a lato server

Server-side scripting

- Idea: inserire istruzioni per il calcolo dei contenuti dinamici all'interno della pagina HTML
- Il codice è interpretato dal server
- Il browser riceve HTML puro



Template: testo+script da interpretare sul server

Java Server Pages (JSP)

- JSP è una architettura per server-side scripting proposta da Sun (come alternativa a ASP di Microsoft)
- Si fonda su tecnologia Java: linguaggio Java, Java Servlet, Java Beans

File JSP

- Un file .asp contiene una combinazione di:
  - Testo
  - Tag HTML
  - Istruzioni di server-side scripting
- E' necessario disporre di un ambiente di compilazione apposito (ad es. Tomcat di Apache Jakarta)

Procedure ed espressioni

- Scriptlets
  - procedure racchiuse tra delimitatori (<%.....%>) scritte in Java
- Espressioni
  - Un'espressione ha la forma:
    <%= variable%>
    La variabile viene valutata e il valore risultante, convertito in stringa, viene inserito nella pagina al posto dell'espressione

Direttive

- Direttive: <%@ variable="value" %>
  - il tipo di linguaggio da utilizzare all'interno del file JSP (<%@ language="java"%>)
  - il file .jsp o .html che viene ritornato al client se l'esecuzione della pagina JSP restituisce un errore (<%@ errorepage="contactwebmaster.html"%>)
  - i moduli (packages) importati dagli oggetti usati nella pagina (<%@ import="java.io.*, java.util.Hashtable%>)

Esempio

```
<HTML>
  <HEAD>
  <TITLE>Request Information Example</TITLE>
  </HEAD>
  <BODY>
  <H3>Request Information Example</H3>
  Method: <%= request.getMethod() %> <BR>
  Request URI: <%= request.getRequestURI() %> <BR>
  User Agent: <%= request.getHeader("User-Agent")
  %>
  </BODY>
</HTML>
```

Risultato: la stessa pagina computata dal servlet mostrato in precedenza

Compilazione

- Il file JSP viene prima tradotto in un servlet
- Il servlet viene compilato in bytecode
- La versione compilata viene tenuta in memoria per rendere più veloce una successiva richiesta della pagina
- NB: La versione originaria di Microsoft ASP (non ASP.NET) non compilava i template

Valutazione

- Indipendenza dal tipo di browser utilizzato
  - Il browser vede solamente pagine HTML
  - All'utente non sono necessari programmi proprietari o estensioni del browser
- Maggior facilità di apprendimento e utilizzo rispetto a servlet
- Oggetti di utilità
- Nasconde la presenza di programmi script agli utenti e ad eventuali hacker
- Resta però codice (anche se poco) frammisto a markup HTML