

FAST-CGI

<http://www.fastcgi.com>

- Il web server genera un unico **processo fast-cgi** in fase di inizializzazione
- Il processo esegue una routine di inizializzazione e si pone in attesa
- Ad ogni richiesta, il web server apre una connessione verso il processo fast-cgi
- Il processo genera output sulla connessione http col client passatagli dal server http
- Il processo fast-cgi chiude la connessione e rimane in attesa di nuove connessioni

FAST-CGI: vantaggi

- **Migliori prestazioni:** creazione di processi fast-cgi solo in fase di inizializzazione
- **Mantenimento dello stato:** la persistenza del processo fast-cgi consente di superare la natura stateless di http
- **Disponibilità:** distribuito gratuitamente da Open Market come fast-cgi library

Java Servlet

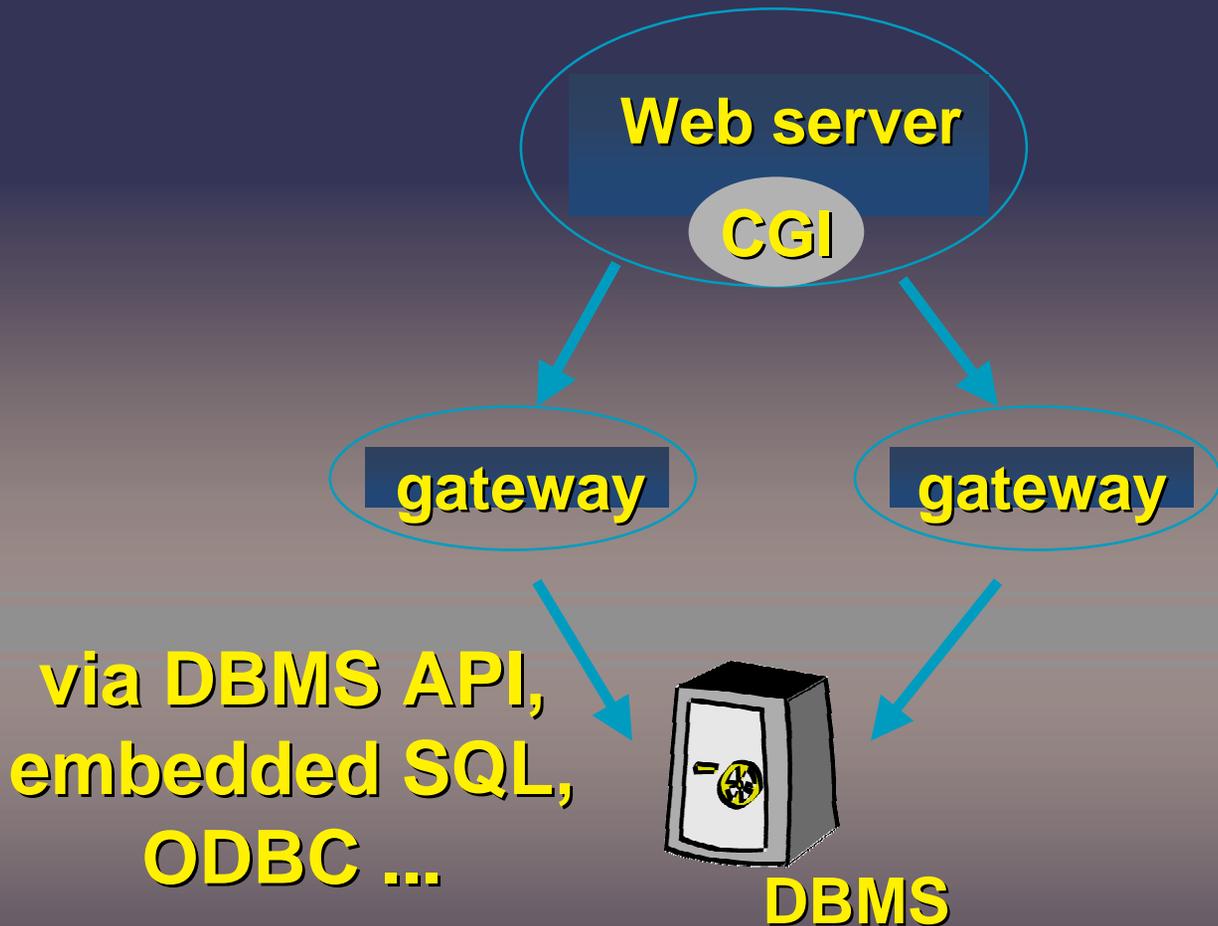
<http://www.javasoft.com/products/java-server/servlets/index.html>

- Differenze rispetto a fast-cgi:
 - **protocollo** basato su NCGI, versione semplificata di fast-cgi
 - applicazioni (**servlet**) scritte in Java
 - **servlet eseguite** nello stesso processo del web server, con minor carico di comunicazione interprocesso risultante

Servlet: vantaggi

- **Indipendenza** dalla piattaforma grazie a Java
- **Sicurezza** gestita mediante Security Manager della JVM
- **Gestione degli errori** con il meccanismo delle eccezioni Java
- **Disponibilità:** distribuzione gratuita di Java Servlet Development Kit contenente la libreria Java servlet

Programmi CGI per accesso a DB



Programma CGI

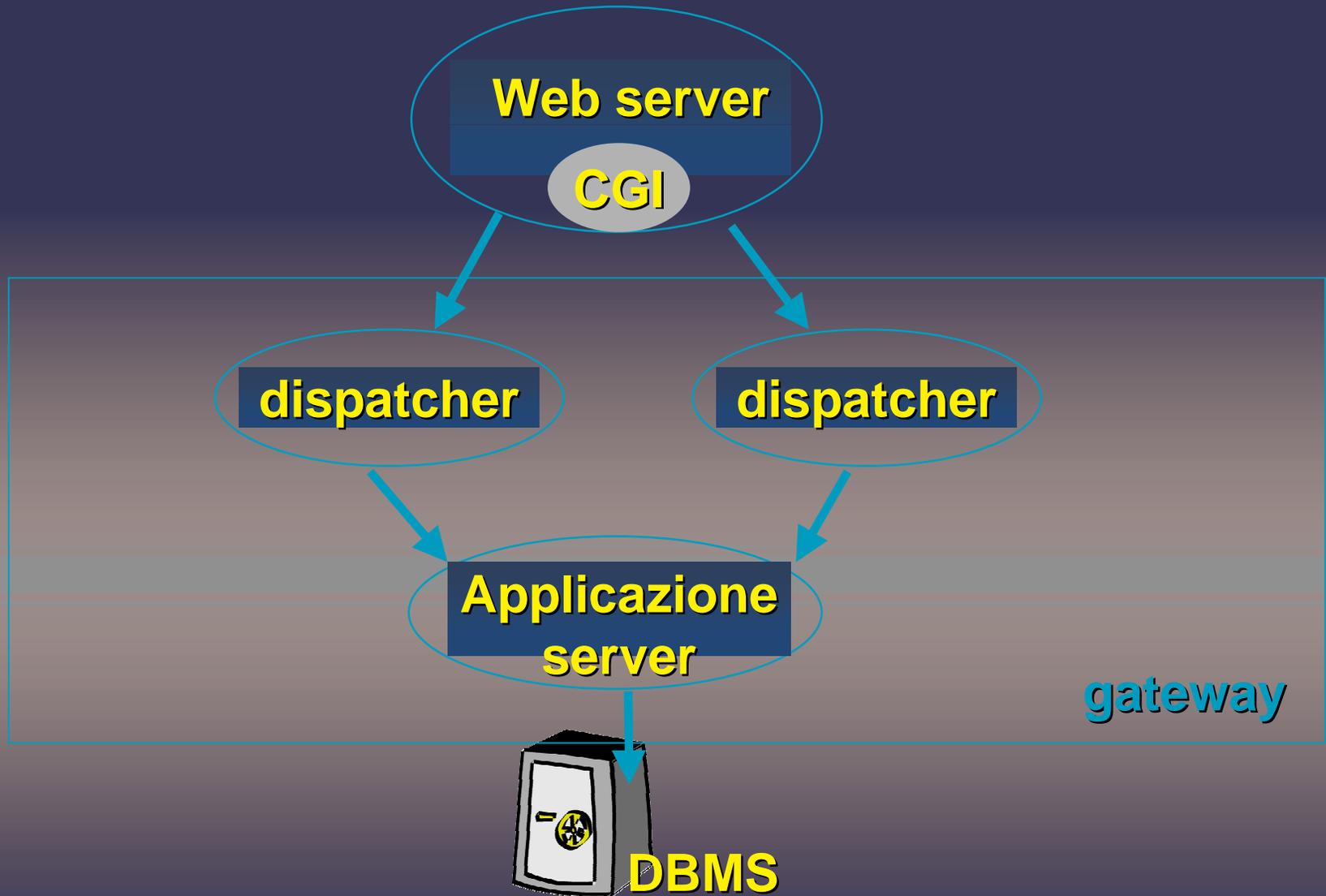
- **PRO**

- **portabilità: usa solo standard aperti (URL, HTTP, CGI, HTML)**

- **CONTRO**

- **prestazioni: creazione di un sotto-processo per ogni richiesta**
- **aperture e chiusure ripetute della connessione con la base di dati**

(thin CGI +) Server CGI



Server CGI

- **PRO**

- **evita** apertura e chiusura della connessione ad ogni richiesta
- **sfrutta** le ottimizzazioni del DBMS
- **basato** su standard CGI

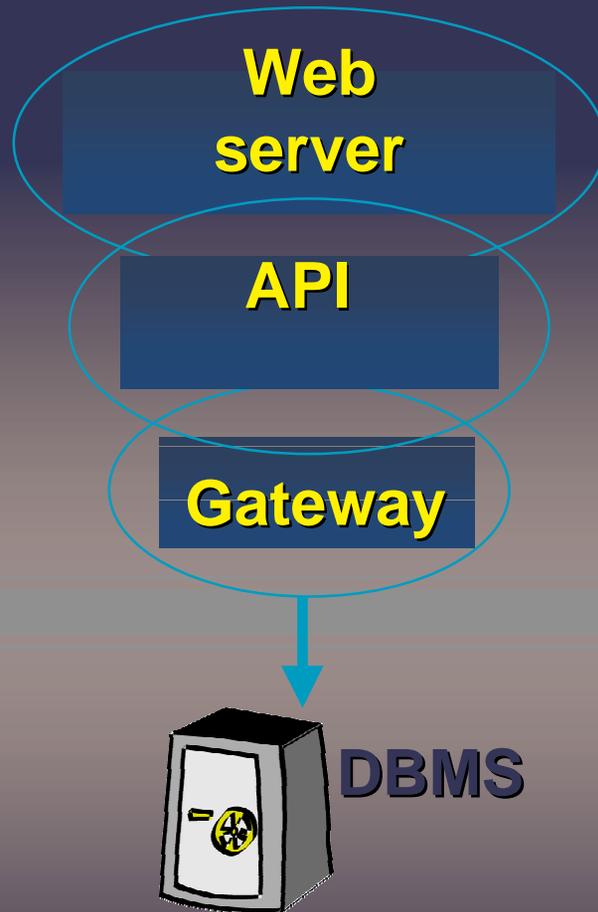
- **CONTRO**

- **prestazioni:** tempo di commutazione tra processi diversi (dispatcher - server)
- **complessità** realizzativa dell'interfaccia tra dispatcher e applicazione server

Scripting Server-side Embedded

- **PHP - prodotto opensource**
 1. interprete plug-in in Web server Apache
 2. interprete esterno con wrapper CGI
 - linguaggio di facile apprendimento
 - accesso a DBMS con funzioni specifiche!!
- **ASP - estensione di Microsoft IIS**
 - supporta diversi linguaggi di scripting (VBScript, JScript, Perl...)
 - accesso a DBMS tramite ADO/OLEDB
 - esistono prodotti per usare pagine ASP anche su diverse piattaforme hw/sw

Server API



API = interfaccia per estendere il server con servizi non standard

Server API

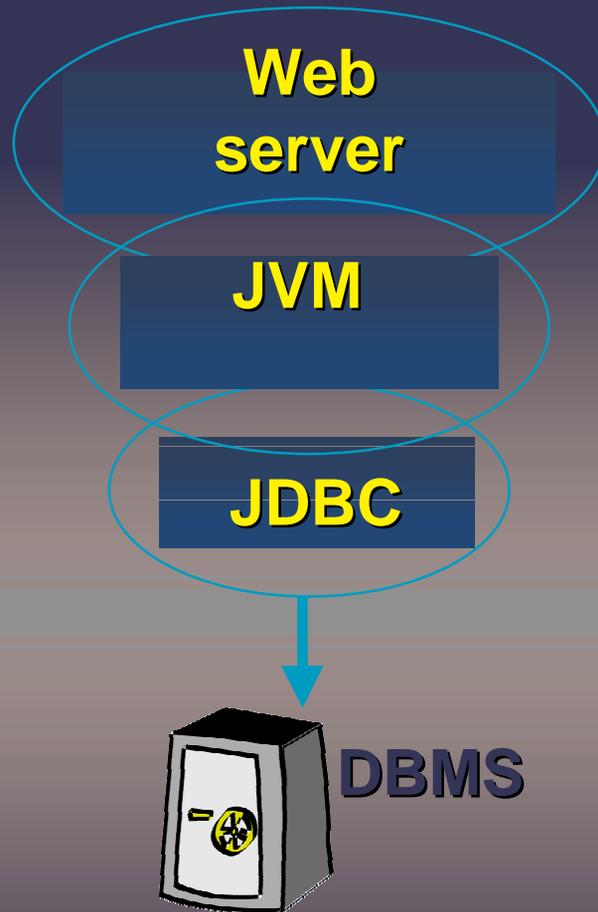
- **PRO**

- **efficienza**: il gateway è nello stesso processo del Web server (es. è una libreria collegata dinamicamente)
- ancora parzialmente **basato** su standard aperti (URL, HTTP, HTML)

- **CONTRO**

- **dipendenza** da API proprietarie (NSAPI, ISAPI) non standardizzate

Servlet API e JSP



API = Java Virtual Machine consente la costruzione dinamica di pagine JSP e l'esecuzione di Java Servlet

Servlet API e JSP

- **Java Virtual Machine** attiva sul server Web (in forma nativa o come plug-in)
- il **Servlet Engine** genera dinamicamente le pagine Web:
 - interpretando **pagine JSP** (soluzione simile ad ASP ma che usa Java come linguaggio di scripting)
 - eseguendo **Servlet Java** (programmini tipo “applet” ma eseguiti sul lato server)

Servlet API e JSP

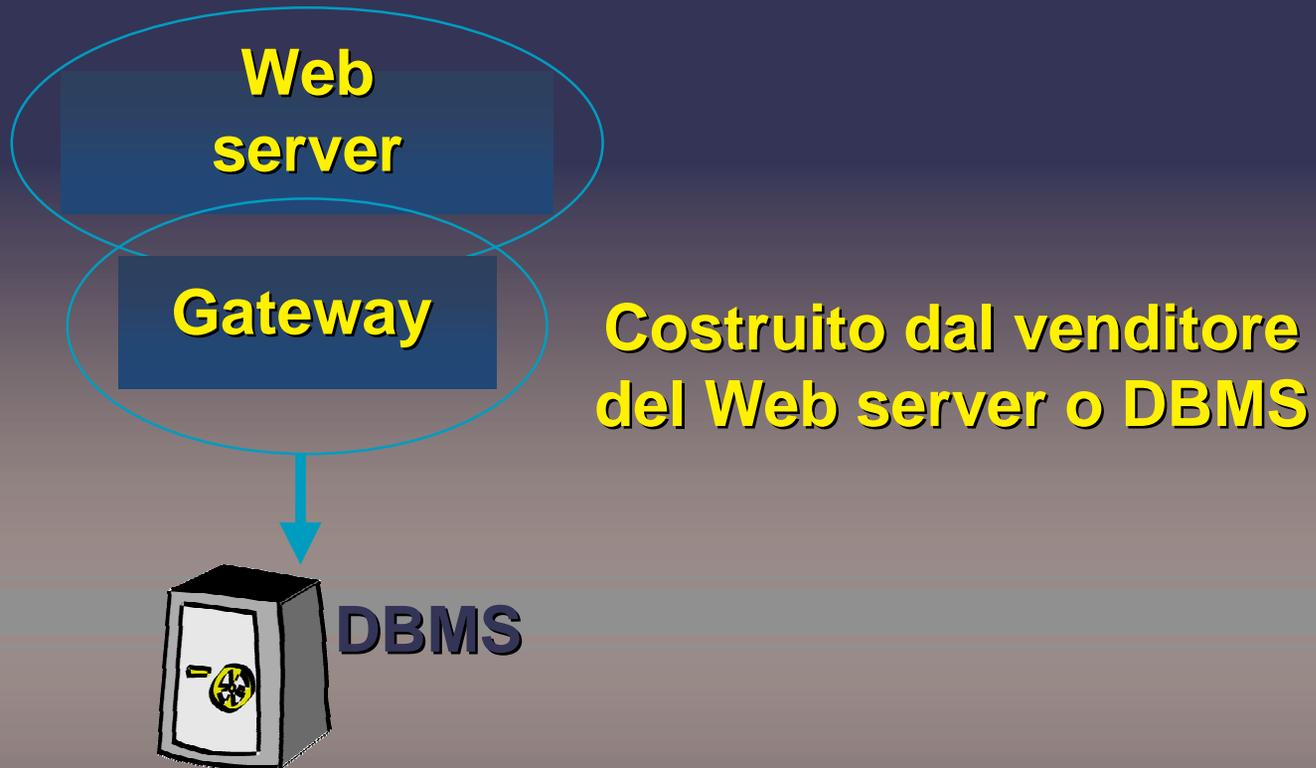
- **PRO**

- **efficienza:** richieste eseguite da un thread leggero Java
- **portabilità:** soluzione disponibile sull'ammaggior parte delle piattaforme e basata su standard (Java, JDBC)
- **flessibilità e affidabilità** dovute a Java

- **CONTRO**

- **complessità dell'architettura** risultante
- **difficoltà nell'uso di Java** rispetto ai linguaggi di scripting

Server Proprietario



Server Proprietario

- **PRO**

- **efficienza:** il server Web diventa una applicazione client della base di dati

- **CONTRO**

- **dipendenza** dallo specifico Web Server e DBMS (a meno che il gateway non utilizzi prodotti di connettività come ODBC)

Estensione esterna del browser

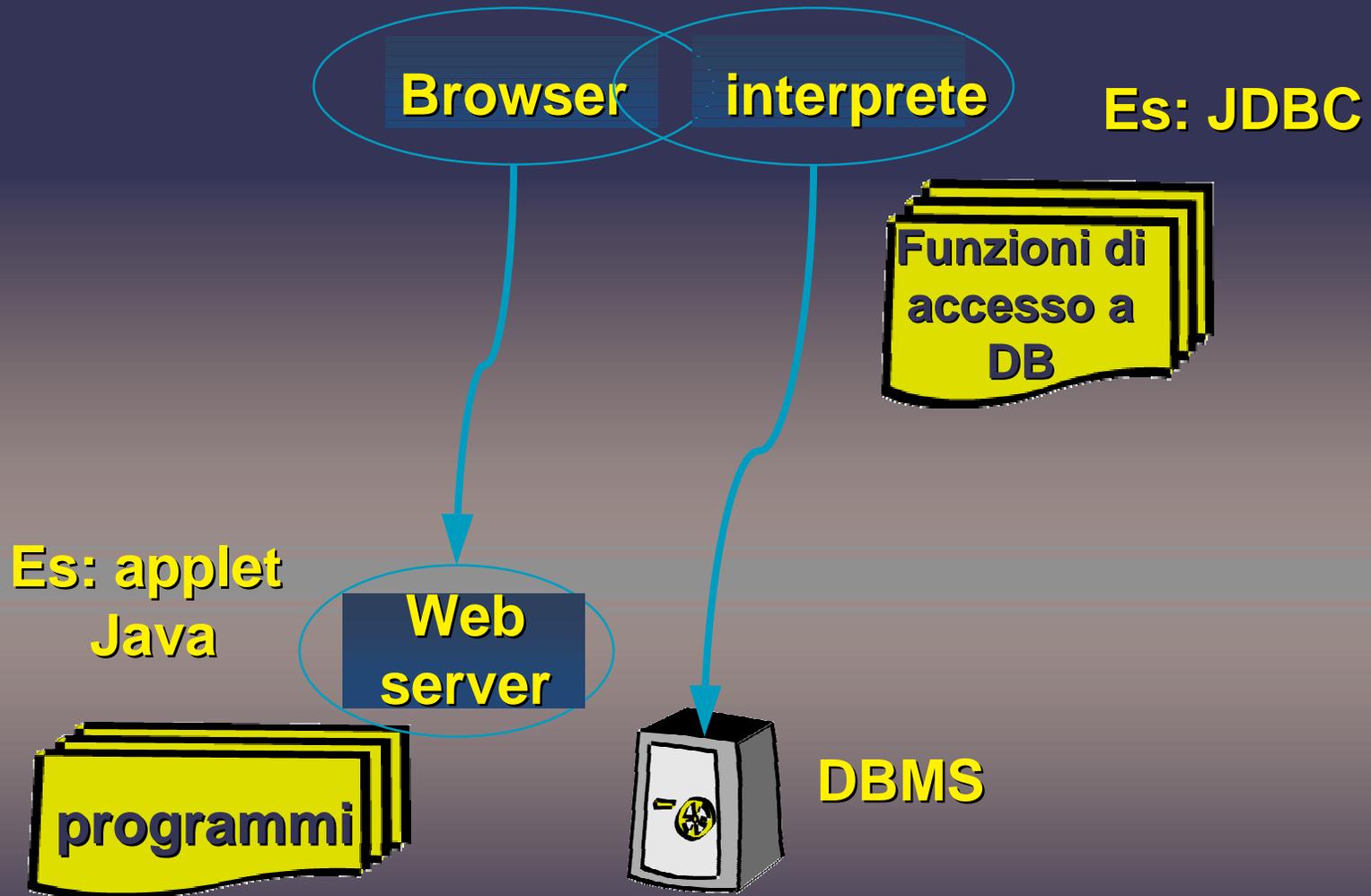
- **PRO**

- **semplice**: stesse modalità di sviluppo di applicazioni client-server tradizionali
- **riuso** di applicazioni pre-esistenti

- **CONTRO**

- **non è una vera integrazione**
- **non sfrutta appieno le potenzialità del Web**
- **i file da trasferire in rete possono essere di grandi dimensioni**

Estensione interna del browser



Estensione interna del browser

- **PRO**

- portabilità (es. Java è uno standard aperto)
- non servono applicazioni sul server
- connessione al DBMS semplificata

- **CONTRO**

- il browser è più complesso
- prestazioni ridotte a causa dell'interpretazione e della necessità di scaricare codice dalla rete
- tecnologia immatura

Il futuro

- **Architetture basate su application server polifunzionali:**
 - **gateway per basi di dati**
 - **controllo degli accessi e sicurezza**
 - **registrazione degli utenti e sessioni durevoli**
 - **gestione della distribuzione, del lavoro cooperativo, di workflow**
- **Supporto di servizi forniti da terze parti (Application Server Providers)**

Evoluzione di HTML

level 1

la versione originale: comandi di formattazione, liste, riferimenti ipertestuali, immagini

level 2

tabelle, form di inserimento dati, frames, mappe client-side

level 3

tabelle avanzate, font, allineamento del testo, sub/superscript, testo attorno a immagini

level 4

cascading style sheets (controllo posizione e oggetti grafici sovrapponibili), simboli matematici

Oltre HTML:

**HTML dinamico, DOM, CSS, HTML 4.0,
XML, XSSL, XLL, RDF**

Obiettivi

Migliorare la resa grafica su video e carta

Ridurre l'interazione client/server

Rendere HTML estensibile in modo consistente

Produrre documenti autodescrittivi

Due linee evolutive:

HTML: DOM, CSSL, HTML 4.0

XML: XSSL, XLL, RDF

Document Object Model (DOM)

<http://w3c.org/DOM>

- Un modello ad oggetti degli elementi tipici di una pagina HTML
- Fornisce una API standard per accedere dinamicamente e modificare gli elementi di una pagina (ad esempio, aggiungere voci a un menu')
- Interfacce degli oggetti specificate in IDL
- Programmi di manipolazione in diversi linguaggi di scripting (JavaScript, VBScript)
- Consente di ridurre le chiamate al server

Cascading Style Sheet (CSS)

<http://w3c.org/Style>

Specifica della presentazione indipendente dal contenuto e dal mezzo (stampa, video, audio)

Style Sheet: specifica testuale di regole di formattazione da applicare al testo

Regola: when <pattern> do <action>

Pattern: configurazione di elementi del testo

Azione: produzione di sezioni di testo contenenti oggetti grafici (control flow obj.s)

Migliora la resa grafica dei documenti

eXtended Markup Language

<http://w3c.org/XML/> -- <http://www.microsoft.com/xml/>

HTML: insieme fisso di tag

**XML: standard per creare linguaggi di markup con tag personalizzati
(erede di SGML)**

HTML vs XML

```
<h1> The Idea Methodology </h1>
```

```
<ul>
```

```
<li> di S. Ceri, P. Fraternali
```

```
<li> Addison-Wesley
```

```
<li> US$ 49
```

```
</ul>
```

```
<book>
```

```
<title>The Idea  
Methodology </title>
```

```
<author> S. Ceri </author>
```

```
<author> P. Fraternali </author>
```

```
<ed> Addison-Wesley </ed>
```

```
<price> US$ 49 </price>
```

```
</book>
```

Esempio di documento XML

```
<?xml version="1.0"?>
<oldjoke>
  <burns>Say <quote>goodnight</quote>,
  Gracie.</burns>
  <allen><quote>Goodnight,
    Gracie.</quote></allen>
  <applause/>
</oldjoke>
```

Tipi di marcature

- Elementi: <quote>
- Entità: < (sta per <), ℞ (Unicode)
- Commenti: <!-- qualsiasi testo -->
- Istruzioni: <? Nome-istruzione dati ?>
- Sezioni CDATA (character data)

```
<![CDATA[
    *p = &q;
    b = (i <= 3);
]]>
```

Document Type Definition (DTD)

- Detta il tipo di un documento, cioè:
 - i tag ammessi
 - le regole di annidamento dei tag
- Esempio di dichiarazione di un elemento:
<!ELEMENT oldjoke (burns+, allen, applause?)>
- Il TAG **oldjoke** può contenere uno o più tag **burns**, seguiti da un tag **allen** e un **applause** opzionale

Modello di contenuto

- Di tipo elemento:
`<!ELEMENT oldjoke (burns+, allen, applause?)>`
- Misto:
`<!ELEMENT burns (#PCDATA | quote)*>`
 - PCDATA (parsed character data) identifica un brano di testo qualsiasi
 - Il simbolo | denota disgiunzione (OR)
- Altri: EMPTY, ANY

Document Type Definition (DTD)

- Detta il tipo di un documento, cioè:
 - i tag ammessi
 - le regole di annidamento dei tag
- Esempio di dichiarazione di un elemento:
<!ELEMENT oldjoke (burns+, allen, applause?)>
- Il TAG **oldjoke** può contenere uno o più tag **burns**, seguiti da un tag **allen** e un **applause** opzionale

Esempio di DTD

```
<!ELEMENT oldjoke (burns+, allen, applause?)>  
<!ELEMENT burns (#PCDATA | quote)*>  
<!ELEMENT allen (#PCDATA | quote)*>  
<!ELEMENT quote (#PCDATA)*>  
<!ELEMENT applause EMPTY>
```

Dichiarazioni di attributi

- Per ogni elemento dice:
 - quali attributi può avere il tag
 - che valori può assumere ciascun attributo
 - qual è il valore di default

- Esempio di dichiarazione di attributo:

```
<!ATTLIST oldjoke
```

name	ID	#REQUIRED
label	CDATA	#IMPLIED
status	(funny notfunny)	'funny'>

- Il tag `oldjoke` può contenere 3 attributi

Tipi di attributi

- CDATA: stringa
- ID: identificatore
- IDREF, IDREFS: valore di un attributo di tipo ID nel documento (o insieme di valori)
- ENTITY, ENTITIES: nome (nomi) di entità
- NMTOKEN, NMTOKENS: caso ristretto di CDATA (una sola parola o insieme di parole)

Vincoli sugli attributi

- #REQUIRED: il valore deve essere specificato
- #IMPLIED: il valore può mancare
- #FIXED "valore": se presente deve coincidere con "valore"
- "valore": il valore può non essere specificato, nel qual caso si assume "valore" come default

Dichiarazioni di entità

- Analoghe alle dichiarazioni di macro con #define in C:
- Esempio di dichiarazioni di entità:
 - `<!ENTITY ATI "ArborText, Inc.">`
 - `<!ENTITY boilerplate SYSTEM "/standard/legalnotice.xml">`
 - `<!ENTITY ATIllogo SYSTEM "/standard/logo.gif" NDATA GIF87A>`
- Le entità possono essere **interne** (&ATI;), **esterne** (&boilerplate; &ATIllogo;) oppure **parametriche** (utilizzabili solo nei DTD)

Documenti con DTD

```
<?XML version="1.0" standalone="no"?>
```

```
<!DOCTYPE chapter SYSTEM "dbook.dtd" [  
  <!ENTITY %ulink.module "IGNORE">  
  <!ELEMENT ulink (#PCDATA)*>  
  <!ATTLIST ulink  
    xml:link      CDATA #FIXED "SIMPLE"  
    xml-attributes CDATA #FIXED "HREF URL"  
    URL          CDATA #REQUIRED>  
>
```

DTD esterno

```
<chapter>...</chapter>
```

DTD
interno

XSSL, XLL, RDF

eXtended Style Sheet Language: la versione per XML del concetto di style sheet

eXtended Pointer & Link Language: notazione per collegamenti link tra documenti piu' espressivi dei link HTML (href)

ispirato allo standard ISO/ANSI HyTime

Resource Description Format: una **applicazione XML** per la specifica di meta-informazioni su documenti WWW

si definisce un modello dei dati per i documenti

ogni documento puo' riferirsi ad un altro che ne describe lo schema

Risorse su Web

Extensible Markup Language (XML) 1.0:

<http://www.w3.org/TR/REC-xml>

XML Pointer Language (XPointer):

<http://www.w3.org/TR/1998/WD-xptr-19980303>

eXtended Link Language (XLink):

<http://www.w3.org/TR/1998/WD-xlink-19980303>

Extensible Style Language (XSL):

<http://www.w3.org/TR/1998/WD-xsl>

Dynamic HTML, HTML 4.0

DHTML: una dizione generica che indica uso congiunto di DOM e SS

HTML 4.0: recepisce caratteristiche da DOM, CSS e altre iniziative

style sheets, scripting e stampa

migliori frames, forms e tabelle

annidamento di oggetti

internazionalizzazione (ISO/IEC:10646) e orientamento non

standard del testo

supporto all'accesso per disabili (es. braille)