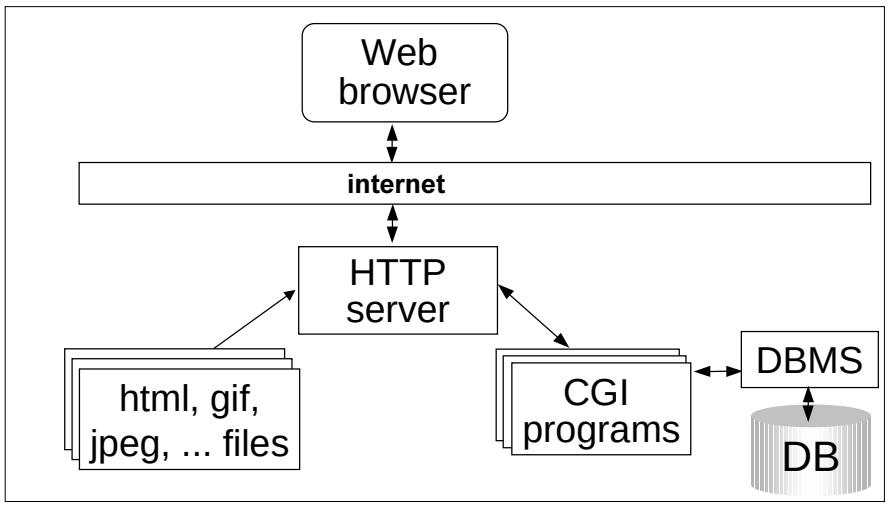
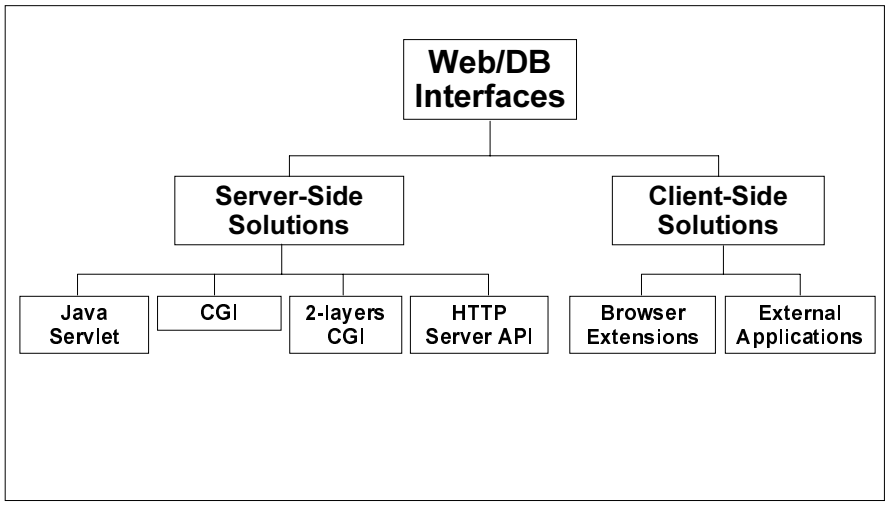
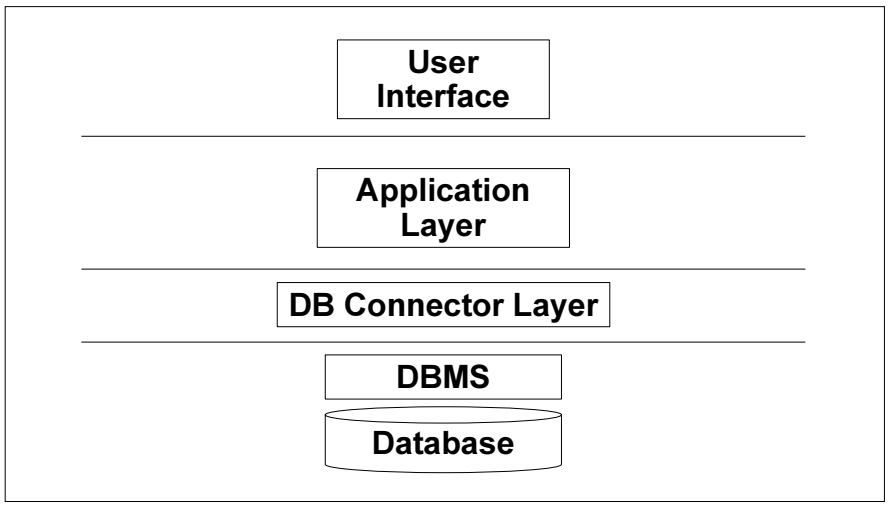
Typical Web server architecture



Web and Databases:
classification of architectural solutions



Web and Databases: layers



Web Browser

- Netscape Navigator
- Microsoft Internet Explorer
- …

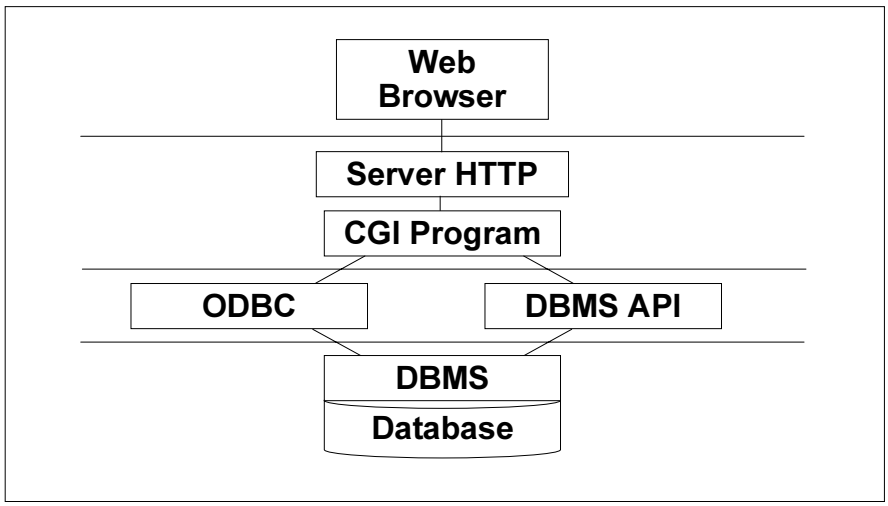
Server HTTP

- Netscape Enterprise Server
- Microsoft Internet Information Server
- NCSA
- Apache
- AOL
- ...

Gateway

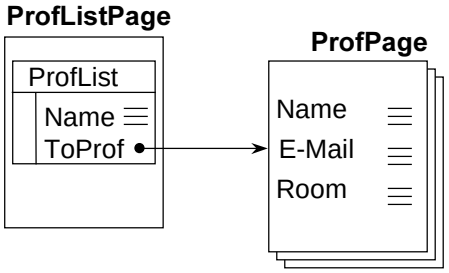
- ODBC
- JDBC
- TCP Socket
- Proprietary protocols (e.g. Oracle SQL*Net)

Web Server Architecture:
Common Gateway Interface (CGI)



Weaving the Web

NAME	E-MAIL	ROOM	...
Paolo Atzeni	atzeni@dia.uniroma3.it	213	...
Gianni Mecca	mecca@dia.uniroma3.it	212	...
...	...	...	...



Weaving the Web
by CGI-programs (ProfListPage)

```
main() {
  char ProfName[20];
  $OPEN CONNECTION(DeptDB);
  $DECLARE ProfCursor CURSOR FOR
    SELECT Name
    FROM ProfTable;
  $OPEN ProfCursor;
  $FETCH ProfCursor INTO :ProfName;
  printf("<HTML> ...<BODY ...');
  printf("<UL>");
  while (sqlcode==0) {
            printf(" <LI><A HREF=/cgi-bin/ProfPage?Name=%s> %s </A>",
                    ProfName,ProfName);
            $FETCH ProfCursor INTO :ProfName;
            }
  $CLOSE CURSOR ProfCursor;
  printf("</UL> ... </BODY></HTML>");
  $CLOSE CONNECTION(DeptDB);
 }
```

Weaving the Web
by CGI-programs (ProfPage)

```
main(int argc, *char argv[ ]) {
 char Name[20], Email[20], Room[20];
 $OPEN CONNECTION(DeptDB);
 $DECLARE ProfCursor CURSOR FOR
  SELECT *
  FROM ProfTable
  WHERE Name = argv[1]
 $OPEN ProfCursor
 $FETCH ProfCursor INTO :Name, :Email, :Room;
 printf(" <HTML> ...<BODY … ');
 printf(" <B> %s </B> ",Name);
 printf(" <BR>E-mail: <I> %s </I> ",Email);
 printf(" <BR>Room: %s ",Room);
 $CLOSE CURSOR ProfCursor;
 printf(" ... </BODY></HTML> ");
 $CLOSE CONNECTION(DeptDB);
}
```

CGI Problems

- **Programming**
- **Portability**
- **Performance**
- **Maintenance**

Portability

- Changing platform imposes to re-compile source code
- Changing DBMS imposes to update DBMS accesses and to re-compile source code
- Interpreted languages (e.g. Perl, TCL, etc.) and standard DBMS connectors (e.g. ODBC) partially solve the problem

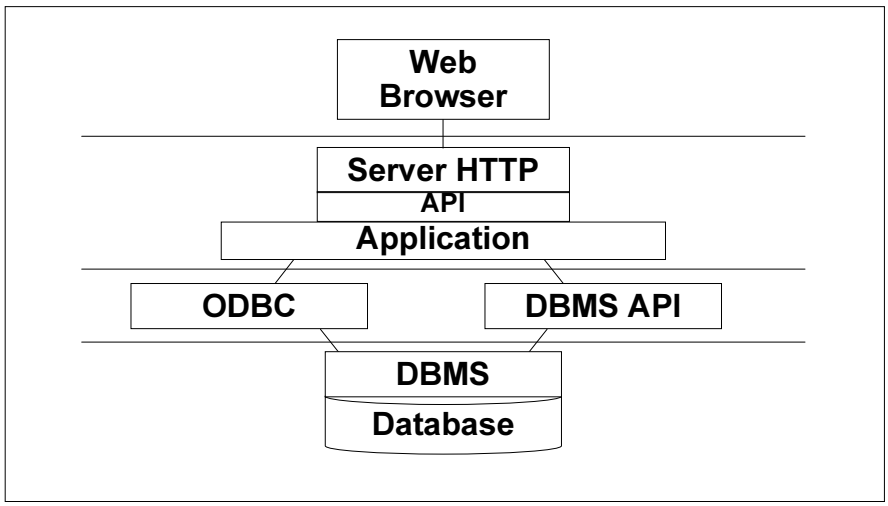
Performance

- Each access needs:
  - to load a program in main memory
  - to open a new connection with DBMS
  - to execute a (bunch of) query
  - to close the connection
- then
  - the DBMS is overloaded
  - the O.S. is overloaded

Maintenance

- Presentation is encoded into source code
- Hypertext structure is encoded into source code
- Data access is encoded into source code

Web Server Architecture:
HTTP Server API



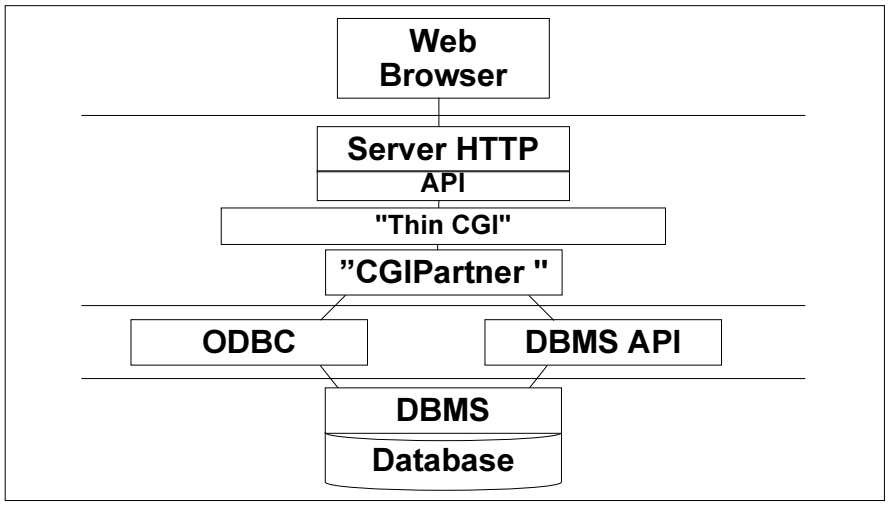
HTTP Server API:
main features

- It is possible to establish direct and continuous connections between the DBMS and the HTTP Server
- It is possible to maintain connections throughout Web applications as well as across invocations of Web applications
- Since the database connections are continuous, applications don't experience the overhead of a connect and subsequent disconnect from the database
- Use specific languages for CGI programs

Web Server Architecture:
HTTP Server API

- Advantages
  - Applications are built "within" the HTTP server by means of specific APIs
  - Persistent applications
- Disadvantages
  - Portability

Web Server Architecture:
2-layers CGI



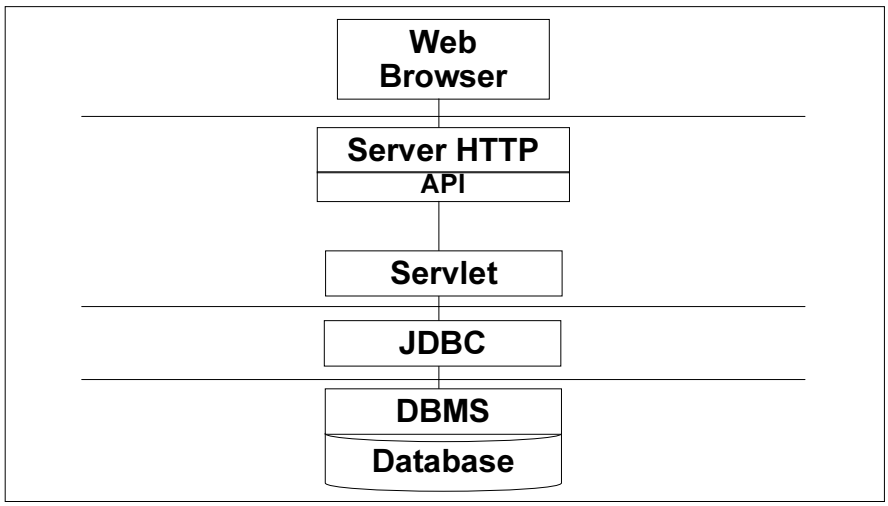
Web Server Architecture: 2-layers CGI

- Each client request activates the execution of a ThinCGI
- ThinCGIs generate a session Id and communicate the request to the CGIPartner
- CGIPartner is a demon, with a pool of open connections to the DBMS
- CGIPartner queries the database, format query's results, and give them back to a ThinCGI, which ends the process sending the results to the clients

Web Server Architecture: 2-layers CGI

- Main Advantages
  - (Each) CGIPartner offers a pool of open DBMS connections
  - (Each) CGIPartner can manage multiple requests
  - ThinCGIs are "thin" applications: their execution doesn't overload the O.S.
  - Session Id can be used to manage connections with the client
- Disadvantages
  - ODBC is a bottleneck !
  - This approach is effective when the DBMS connection is through DBMS vendors API

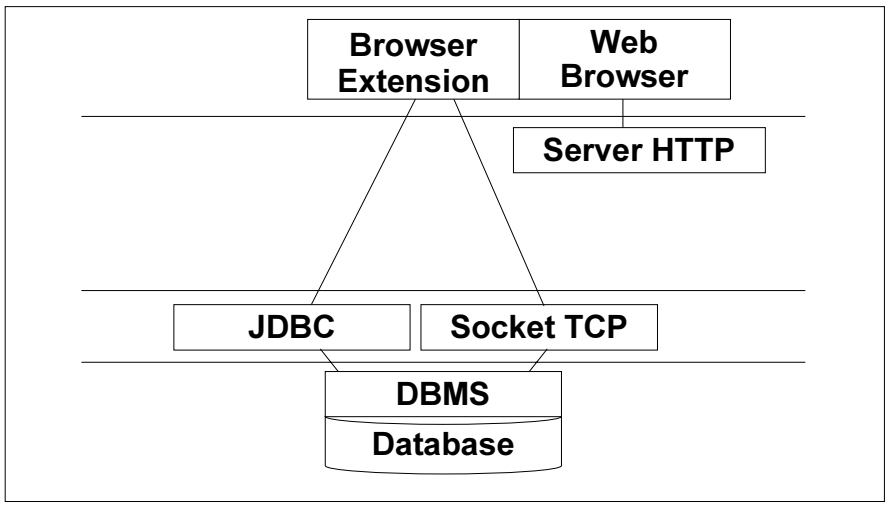
Web Server Architecture:
Java Servlet



Java Servlet

- protocol and platform-independent server side components
- do not require creation of a new process for each request
- allows for three tier applications

Client side solutions: browser extensions



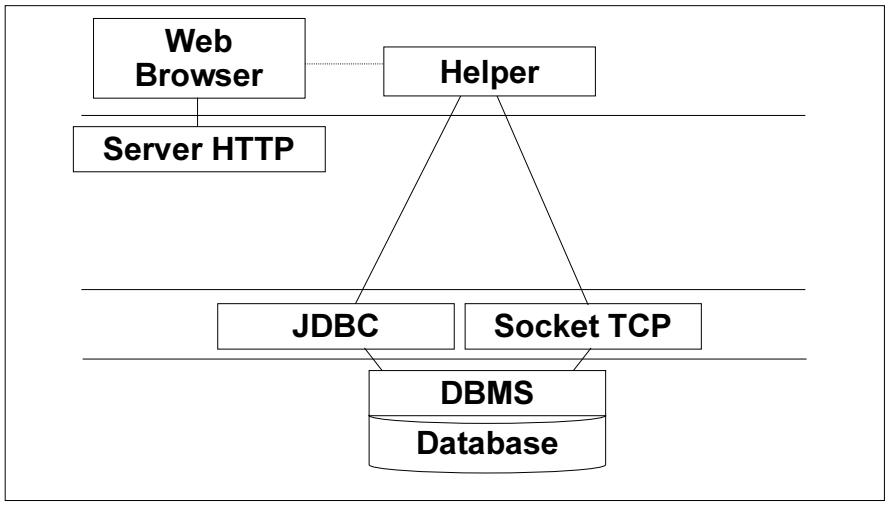
Client side solutions: browser extensions

- Products:
  - Java Applets
  - Netscape Plug-Ins
  - Microsoft ActiveX
  - Netscape JavaScript

Client side solutions: browser extensions

- Advantages
  - allow to manage data entry
  - easy to use
  - flexible
- Disadvantages
  - software updates
  - overload clients

Web Server with client side solutions: external apps



Web Server with client side solutions: external apps

- Examples
  - MS Word-Excel-…
  - Terminal emulators
  - ...

Web Server with client side solutions: external apps

- Advantages
  - may be useful for proprietary solutions
  - audit and security
- Disadvantages
  - software updates
  - process control

Databases and information systems over the Web: a great opportunity

- **Outline**
  - Introduction
  - Web-based information systems: a database perspective
  - Architectures
  - **Tools and techniques**
  - The Web evolution: XML

Tools and techniques for developing WBIS

- **Push and pull**
- **Managing sessions**
- **Market solutions**
  - (architectures)
  - languages
  - tools
- **Research Proposals**

Push and pull

- **The "pull" approach**
  - generate pages dynamically
- **The "push" approach**
  - materialize pages (in text files)

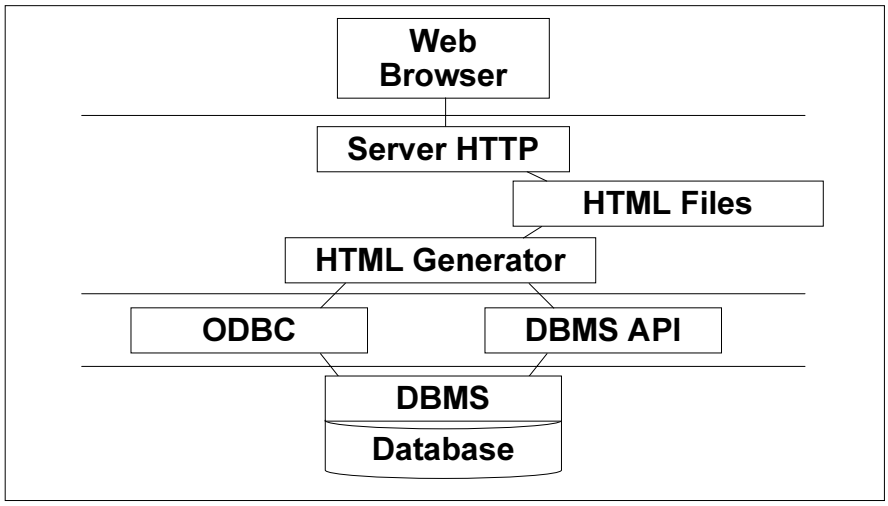
The pull approach

- **Advantages**
  - pages are always up to date
  - easy maintenance
- **Disadvantages**
  - increases the DBMS overload (but recent architectures seem to be able to overcome the problem)
  - portability (but Java servlets ...)
- **Main features**
  - supported by most of the tools available on the market

The push approach

- **Advantages**
  - portability
  - reduces the DBMS overload (if any)
- **Disadvantages**
  - needs techniques to maintain consistency between database and hypertext
- **Main features**
  - needs a mechanism for URL invention

Web Server:
push approach



Materializing Page-schemes (ProfListPage)

```
main( ) {
  char ProfName[20], Fname[20];
  FILE *fp;
  strcpy(GenerateURL("ProfListPage"), Fname);
  fp= fopen(Fname, "w");
  $OPEN CONNECTION(DeptDB);
  $DECLARE ProfCursor CURSOR FOR
    SELECT Name
    FROM ProfTable;
  $OPEN ProfCursor;
  $FETCH ProfCursor INTO :ProfName;
  fprintf(fname, "<HTML> ...<BODY … <UL>");
  while (sqlcode==0) {
            fprintf(fname, " <LI><A HREF=/ProfPage/%s> %s </A>",
                      GenerateURL(ProfName), ProfName);
            $FETCH ProfCursor INTO :ProfName;
            }
  $CLOSE CURSOR ProfCursor;
  fprintf(fname, "</UL> ... </BODY></HTML>");
  $CLOSE CONNECTION(DeptDB);
 }
```

Materializing Page-schemes (ProfPage)

```
main( ) {
 char ProfName[20], Email[20], Room[20], Fname[20];
 FILE *fp;
 $OPEN CONNECTION(DeptDB);
 $DECLARE ProfCursor CURSOR FOR
   SELECT *
   FROM ProfTable
 $OPEN ProfCursor
 $FETCH ProfCursor INTO :ProfName, :Email, :Room;
 while (sqlcode==0) {
            sprintf(Fname, ”/ProfPage/%s”, GenerateURL(”ProfName"));
            fp= fopen(Fname, "w");
            fprintf(Fname, "<HTML> ...<BODY> … ");
            fprintf(Fname, "<B> %s </B> ",ProfName);
            fprintf(Fname, "<BR>E-mail: <I> %s </I> ",Email);
            fprintf(Fname, "<BR>Room: %s ",Room);
            fprintf(" ... </BODY></HTML> ");
   $FETCH ProfCursor INTO :ProfName, :Email, :Room;
   }
 $CLOSE CURSOR ProfCursor;
 $CLOSE CONNECTION(DeptDB);
 }
```

Managing Sessions

- Browser session: a sequence of related requests from a particular user to a particular application
- The HTTP protocol is stateless
- It cannot manage a sequence of requests

Managing Sessions: Examples

- E-Commerce applications
- Educational applications
- ...
  - shopping baskets
  - user navigation track
  - user-customized presentation
  - ...

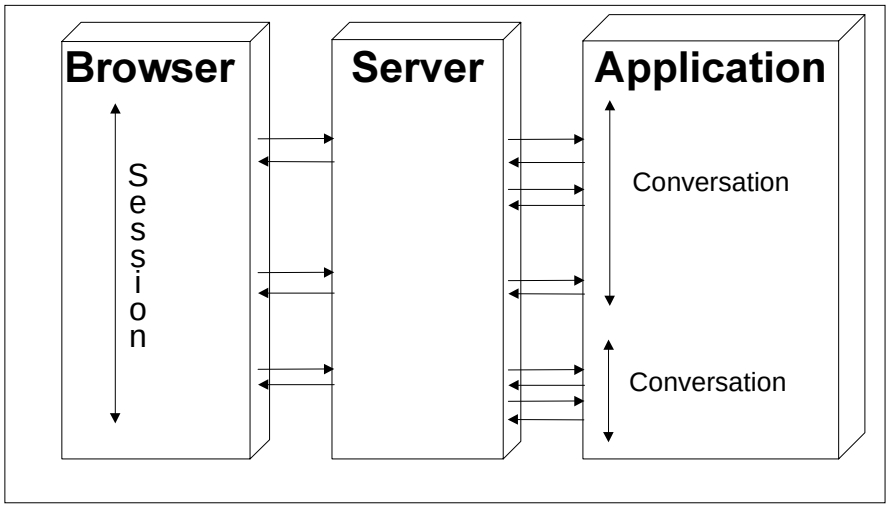
Managing sessions: techniques

- A “token” is used to represent the session
- the token is passed between browser and server as:
  - part of the URL
  - hidden field in a form
  - cookie

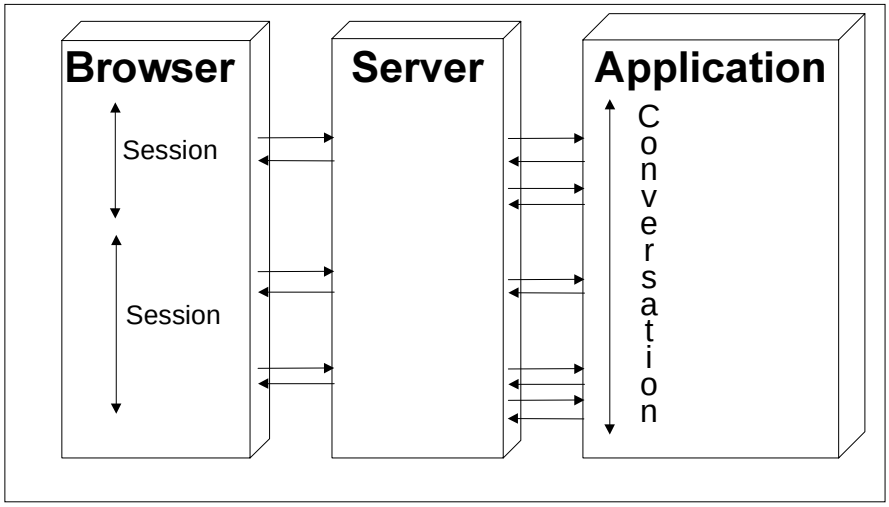
Session Management

- Session initialization
  - the browser accesses an initialization page (e.g. a login form) able to launch an init program on the server
  - the init program generates a session id
- Session Id transmission
  - a page containing the session id is sent back to the browser
  - every further request will contain the session id

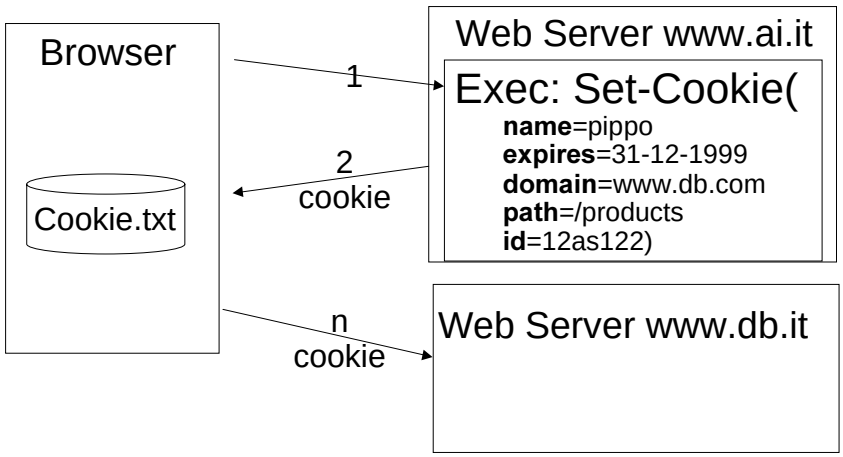
Managing sessions and conversations



Managing sessions and conversations



Cookies



Cookies

- **Name**: cookie name
- **Expires**: expiration date
- **Domain**: domain where to cookie has to be sent
- **Path**: path where the cookie has to be sent
- **Secure**: indicates that the cookie can be transmitted only through SSL
- Other Information (<4k): application data

Market Solutions

- **Languages**
  - HTML Extensions
  - 4GL Extensions
- **Tools**
  - Tools to support the implementation
  - Tools to support design activities

Languages

- **HTML Extensions**
- 4GL Extensions
  - add specialized functions to a 4GL
  - the result of a program execution is HTML code

HTML Extensions

- **Microsoft Active Server Page (ASP)**
- **Java Server Page**
- **Microsoft Internet Database Connector (IDC)**
- **Allaire Inc. Cold Fusion**
- **Sybase Web.SQL**
- **...**

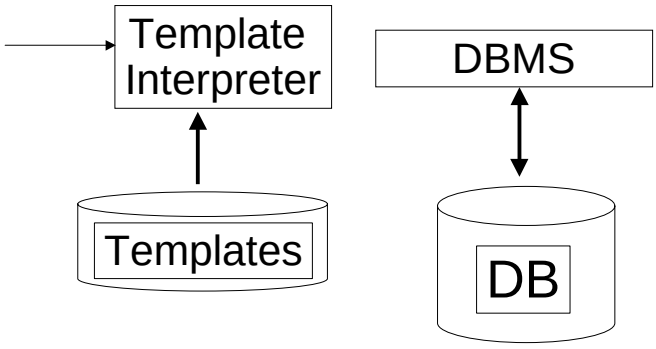
HTML Templates: Example

(source Sybase)

```
TEMPLATE: WORK-LIST

<HTML>
<BODY BGCOLOR="WHITE">

...

<SYB_TYPE=SQL>
SELECT lname, fname, title, price
FROM   authors a,
       titleauthor ta,
       titles t
WHERE (a.id = ta.id
       and t.title_id = ta.title_id)
</SYS>

...

</BODY>
</HTML>
```

HTML Templates: Architecture

Browser Request:
/tpl-int?TPLName=Work-List



Languages

- HTML Extensions
  - embed SQL statement in specific HTML tags
  - writing page templates becomes an implementation paradigm
- **4GL Extensions**
  - add specialized functions to a 4GL
  - the result of a program execution is HTML code

4GL Extensions

- **IBM Net.Data**
- **Oracle PL/SQL Web/Toolkit**
- **Informix Web Datablade**
- …

4GL Extensions: Example
(source Informix)

```
CREATE FUNCTION ArtistPage(text) RETURN TEXT
AS
SELECT UNIQUE
 "<TABLE WIDTH=100%><TR>
        <TD><IMG SRC=Webdriver?LO=" ||
                logo::text || "&Type=“image/gif” " ||
                "<BR>" ||
                <STRONG> " || name || "</STRONG>"
                <P><B>Birth:</B><EM>” || birth || "</EM><BR>" ||
                <P><B>Death:</B><EM> "||death||" </EM><TD>
                <TD ALIGN=LEFT><P>" ||biography ||
        " </TD></TR></TABLE><P> "
FROM  ArtistTable
WHERE Name LIKE $$1
```

4GL Extensions: Example (results)

```
<TABLE WIDTH=100%><TR>
 <TD><IMG SRC=Webdriver?LO=
 l0109766443206 &Type="image/gif"
 <BR>
 <STRONG>Botticelli</STRONG>
 <P><B>Birth:</B><EM>1445</EM><BR>
 <P><B>Death:</B><EM>1510</EM><TD>
 <TD ALIGN=LEFT><P>Alessandro Filipepi,
                  called "il Botticelli'',
                  was borns in Florence...
 </TD></TR></TABLE><P>
```

Tools

- **HTML Editors**
  - WYSWYG Editors
  - Provide Database import facilities
- **Database Publishing Wizards**
  - Export database tables, views, reports, forms for Web publishing
  - Target formats: HTML code (for static pages), HTML or 4GL extension source code (for dynamic page creation)

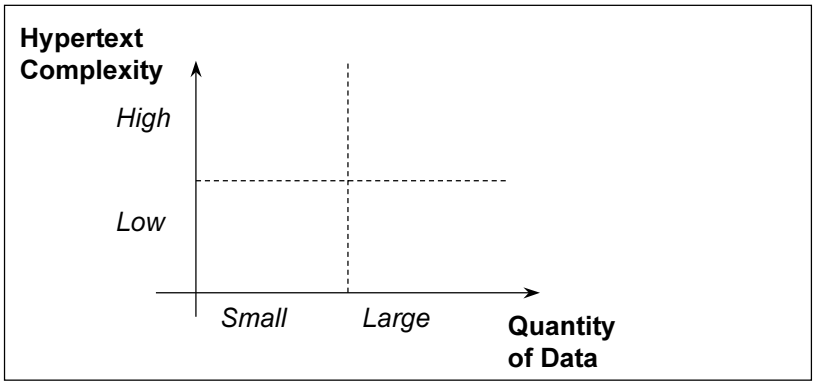
Tools (2)

- **Web site managers**
  - provide graphic interface on the content of a Web site for a tree-like presentation and manipulation of HTML files
  - utilities to check link consistency
- **RADs and Web form editors**
  - provide support for exporting tables, views, reports in HTML (or Java)
  - Assist developers in the construction of interactive form based applications for accessing and updating data

Tools **(3)**

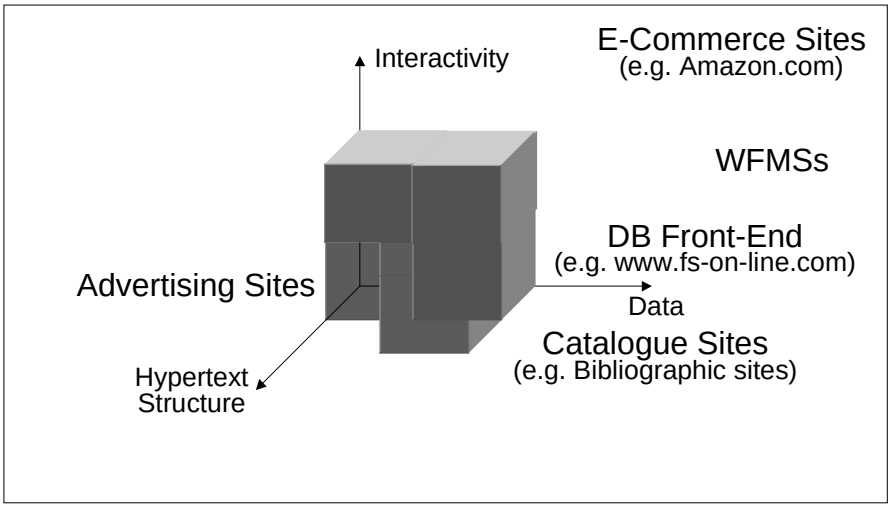
- **CASE Tools**
  - provide full support in both the design and development activities
  - use a model driven approach

Web site Characterization

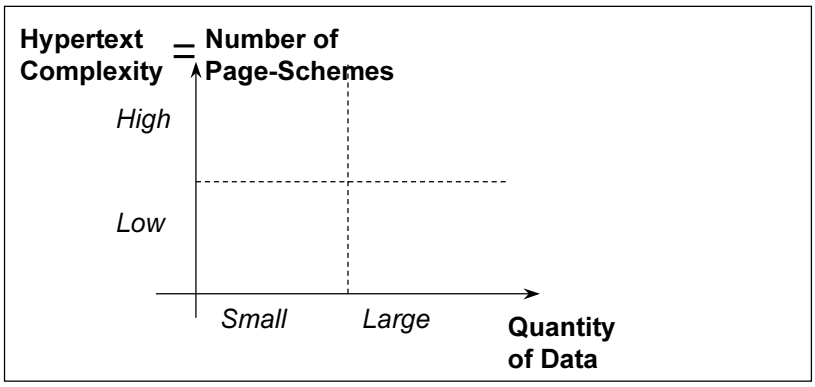


[Fernandez et al., SIGMOD98]

WBIS Features



Web site Characterization: measuring the hypertext complexity



Hypertext Complexity -> Number of Page-schemes

Sites with small quantities of data

- **Features**
  - Several "Unique" pages
  - Focus on graphic layout of pages
  - Maintenance: adding, removing unique pages; link consistency
- **Tools**
  - HTML editors
  - Site managers

Sites with
large quantities of data
and low complexity

- **Features**
  - Few page-schemes with "table data"
  - Focus on data
  - Maintenance: updating the pages' contents
- **Tools**
  - Site managers
  - Database publishing wizards
  - Web form editors

Sites with
large quantities of data
and high complexity

- **Features**
  - Several interconnected page-schemes
  - Focus on data and navigation
  - Maintenance: updating contents and structure
- **Tools**
  - RADs
  - CASE Tools