WEB TECHNOLOGIES

Overview

- Simple Web Pages
  - HTTP
  - HTML
- Web Application Architectures
- Dynamic Web Pages
  - Scripting
  - CGI
  - Server Extensions (APIs)
  - Java: Servlets, JSP, JDBC
  - Microsoft: ASP, ADO, ODBC

History of the Internet

- Developed in 60s and 70s by US Department of Defense
  - called ARPANET (Advanced Research Projects Agency NETwork)
  - project to build a network that could withstand physical attacks
- 1982: TCP/IP adopted as ARPANET standard protocol
- 1986: project shifted from military to government/universities by grant money from National Science Foundation
  - renamed NSFNET (National Science Foundation NETwork)
- 1995: NSFNET ceased control of network backbone; network becomes known as Internet.

Intranet vs. Extranet

- **Intranet** : a web site or group of sites belonging to an organization, accessible only by the members of the organization (behind firewall)

- **Extranet** : an intranet that is partially accessible to authorized outsiders

HTTP (HyperText Transfer Protocol)

The protocol used to transfer Web pages through the Internet.

Version history:

HTTP/0.9: early development of the web

HTTP/1.0: released in 1995

HTTP/1.1: current version

HTTP (continued)

Based on a “request-response” paradigm:

- connection – the client establishes a connection with the web server
- request – the client sends a request message to the web server
- response – the web server sends a response to the client
- close – the connection is closed by the web server

NOTE: HTTP is inherently a “stateless” protocol;

HTTP Request and Response

HTTP (Hypertext Transfer Protocol)

- Exchanged information can be static or dynamic
- Every resource, accessible over the Web has a URL(Uniform resource locator)
- HTTP mechanism is based on client/server model typically using TCP/IP sockets

HTTP Request and Response

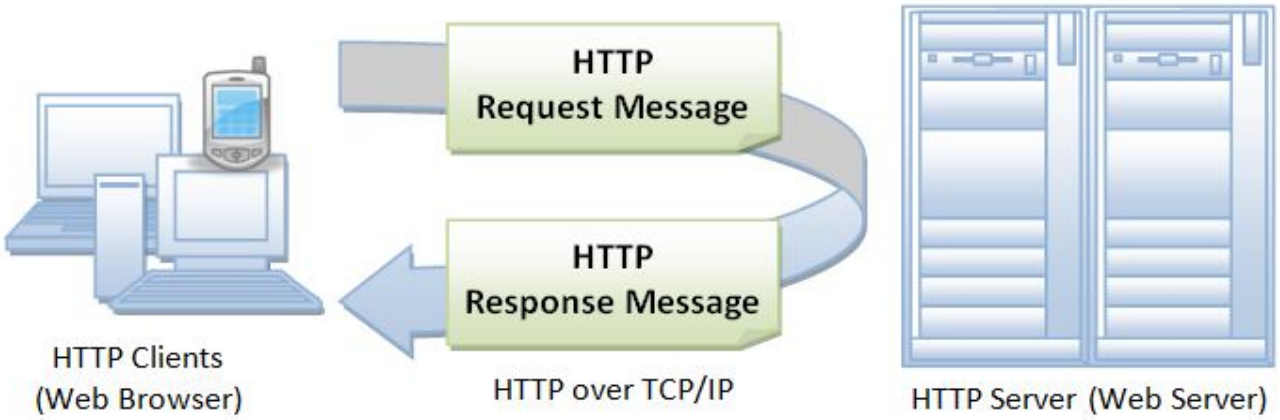


Figure illustrates request-response client-server protocol

HTTP Request

Request = Request-Line
***((general-header**
| request-header
| entity-header) CRLF)
CRLF
[message-body]

Example :

URL: http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

Request Headers | Request Body | Response Headers | Response Body | Cookies | Initiator | Timings

Key	Value
Request	GET /Protocols/rfc2616/rfc2616-sec14.html HTTP/1.1
Accept	text/html, application/xhtml+xml, */*
Referer	http://www.google.com/url?sa=t&source=web&cd=3&ved=0CC4QFjAC8
Accept-Language	en-US
User-Agent	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5
Accept-Encoding	gzip, deflate
Host	www.w3.org
If-Modified-Since	Wed, 01 Sep 2004 13:24:52 GMT
If-None-Match	"1edec-3e3073913b100"
Connection	Keep-Alive

Source : http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html#sec5

HTTP Request

- Request-Line
  - begins with a method token,
  - followed by the Request-URI & protocol version & ending with CRLF
  - elements are separated by SP characters
  - No CR or LF is allowed except in the final CRLF sequence

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Request-Line

Method

```
Method =          "OPTIONS"
                  | "GET"
                  | "HEAD"
                  | "POST"
                  | "PUT"
                  | "DELETE"
                  | "TRACE"
                  | "CONNECT"
                  | extension-method
extension-method = token
```

Request-Line

Request-URI

Request-URI = "*" | absoluteURI | abs_path | authority

HTTP Request

General header

```
general-header = Cache-Control
                 | Connection
                 | Date
                 | Pragma
                 | Trailer
                 | Transfer-Encoding
                 | Upgrade
                 | Via
                 | Warning
```

HTTP Request

Request header

```
request-header =    Accept
                  | Accept-Charset
                  | Accept-Encoding
                  | Accept-Language
                  | Authorization
                  | Expect
                  | From
                  | Host
                  | If-Match
                  | If-Modified-Since
                  | If-None-Match
                  | If-Range
                  | If-Unmodified-Since
                  | Max-Forwards
                  | Proxy-Authorization
                  | Range
                  | Referer
                  | TE
                  | User-Agent
```

HTTP Request

Entity header

```
entity-header  = Allow
               | Content-Encoding
               | Content-Language
               | Content-Length
               | Content-Location
               | Content-MD5
               | Content-Range
               | Content-Type
               | Expires
               | Last-Modified
               | extension-header
      extension-header = message-header
```

HTTP Request

Message body

```
message-body = entity-body
             | <entity-body encoded as per Transfer-Encoding>
```

HTTP Response

```
Response      = Status-Line
                *(( general-header
                 | response-header
                 | entity-header ) CRLF)
                CRLF
                [ message-body ]
```

Example :

URL: http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

Request Headers | Request Body | Response Headers | Response Body | Cookies | Initiator | Timings

Key	Value
Response	HTTP/1.1 200 OK
Date	Sun, 05 Dec 2010 01:27:11 GMT
Server	Apache/2
Last-Modified	Wed, 01 Sep 2004 13:24:52 GMT
ETag	"1edec-3e3073913b100"
Accept-Ranges	bytes
Content-Length	126444
Cache-Control	max-age=21600
Expires	Sun, 05 Dec 2010 07:27:11 GMT
P3P	policyref="http://www.w3.org/2001/05/P3P/p3p.xml"
Connection	close
Content-Type	text/html; charset=iso-8859-1

HTTP Response

Status Line

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

Status Code and Reason Phrase

- 1xx: Informational - Request received, continuing process –
- 2xx: Success - The action was successfully received, understood, and accepted
- 3xx: Redirection - Further action must be taken in order to complete the request
- 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
- 5xx: Server Error - The server failed to fulfill an apparently valid request

For a detailed list of status codes refer : http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6

Source : http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6

HTTP Response

Response Header

```
response-header = Accept-Ranges
                | Age
                | ETag
                | Location
                | Proxy-Authenticate
                | Retry-After
                | Server
                | Vary
                | WWW-Authenticate
```

Source : http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6

HTTP Request

HTTP header indicating:

1. type of request
   - GET: gets the requested resource
   - POST: transfer posted data to the specified resource
   - HEAD: similar to get but returns the HTTP header ONLY
   - PUT (HTTP/1.1): uploads the resource to the server
   - DELETE (HTTP/1.1): deletes the resource from the server
   - OPTIONS (HTTP/1.1):request's the server's configuration options
2. name of a resource
3. HTTP version
4. body (*optional)

HTTP Response

HTTP header indicating:

1. HTTP version
2. status of the response
3. information to control the response behavior
4. body (*optional)

MIME types

Multipurpose Internet Mail Extensions

used by HTTP header to determine how to handle multiple media types

Example:

- text/html (html document; *.html)
- application/java (java class file; *.class)

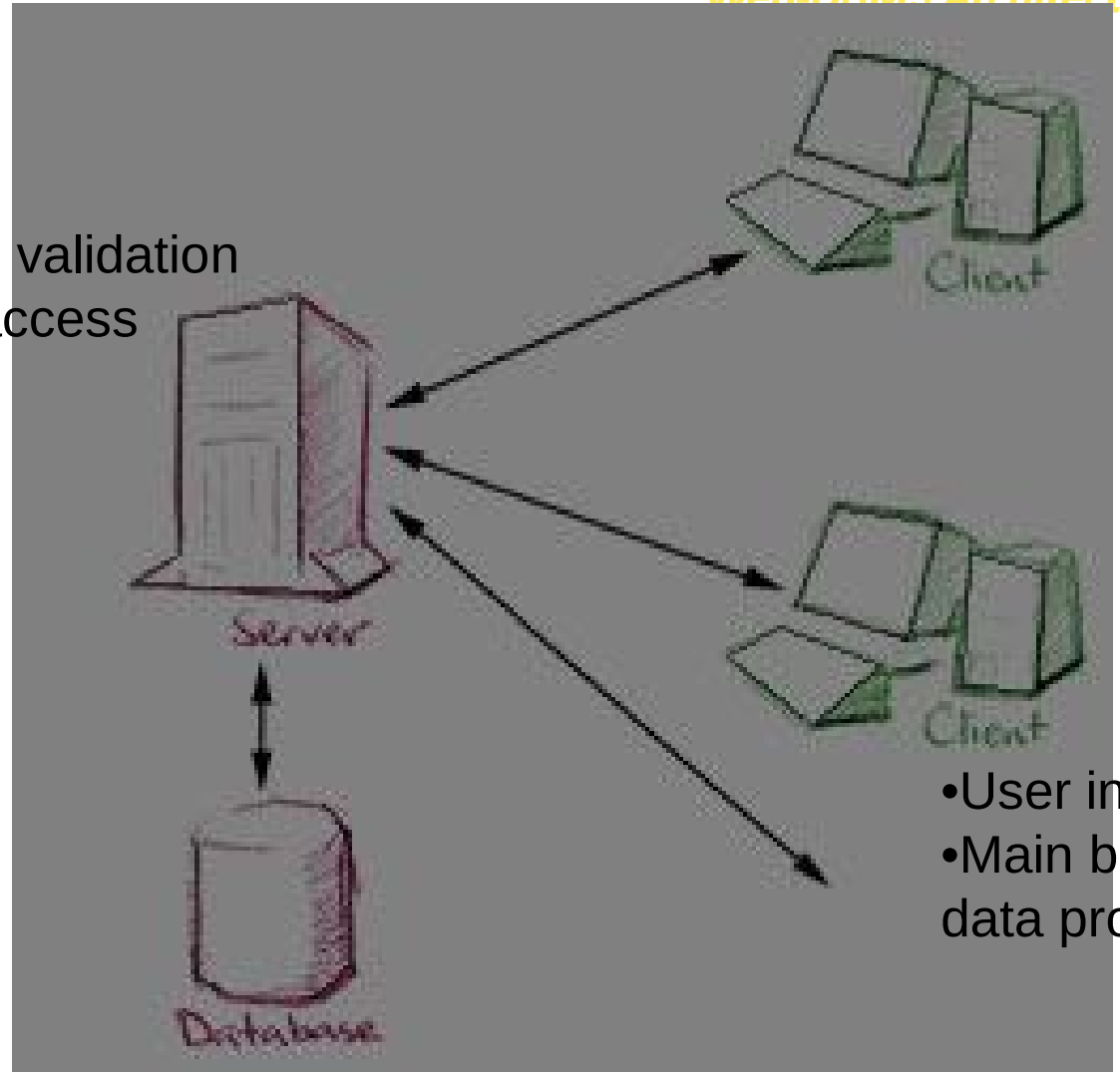
Static vs. Dynamic Web Pages

- **static**: content stored in an html page
  - content does not update unless the file is updated
- **dynamic**: content is generated “on the fly”
  - content is gathered and delivered based on the user’s request; usually content here is stored in a database

two-tier client-server architecture

- **Client (tier 1)** : primarily responsible for *presentation* of data to the user
  - user interface actions
  - main business application logic
- **Server (tier 2)** : primarily responsible for supplying *data services* to the client
  - limited business application logic (i.e. verification not able to be processed by the client)
  - access to the requested data

two-tier architecture (continued)



tier 2

- Server-side validation
- Database access

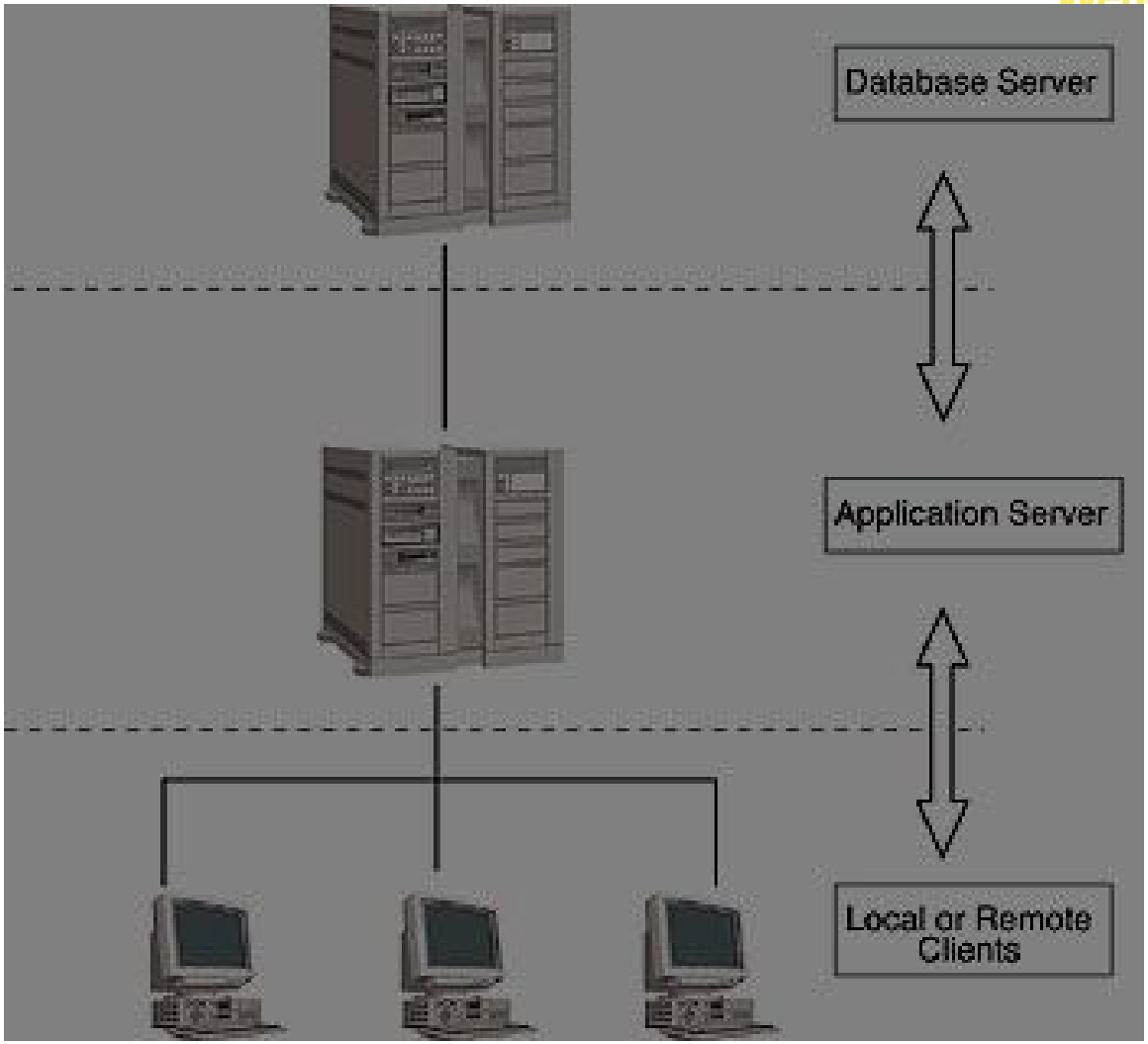
tier 1

- User interface
- Main business and data processing logic

three-tier architecture

- **Client (tier 1)** : primarily responsible for *presentation* of data to the user
- **Application Server (tier 2)** : primarily responsible for supplying *data processing* and *business logic*
- **Database Server (tier 3)** : responsible for *data validation* and *database access*

three-tier architecture (continued)



Web-DBMS Architecture

Tier 3

- data validation
- database access

Tier 2

- business logic
- data processing logic

Tier 1

- user interface

two-tier vs. three-tier

2 tier

- A ‘fat’ client, requiring considerable resources on the client’s computer to run effectively. This includes RAM, disk space, and CPU power.
- A significant client-side administration overhead.

3 tier

- Less expensive because client is ‘thin’
- Maintenance is centralized
- Centralized business logic makes deployment easier
- Added modularity allows modifications to any tier without affecting others
- Load balancing is easier with separation of servers

n-tier architecture

- Done by extending 3-tier's middle tier into any # of tiers
- Is more modular, therefore changes can be more independent
- Load balancing is better because of distribution of work

Web-DBMS Advantages

- Simplicity (minor)
- Platform Independence
- GUI
- Standardization (minor)
- Cross-Platform Support (minor)
- Transparent Network Access
- Scalable Deployment
- Innovation

Integrating Web and DBMSs

- Scripting languages – JavaScript, VBScript, Perl
- CGI
- HTTP Cookies
- Web Server Extensions (APIs) – NetscapeAPI, Microsoft's IIS API
- Java and JDBC, SQLJ, Servlets, JSP
- ASP and ActiveX Data Objects (ADO)

Scripting Languages

- Scripts are embedded in HTML
- Some can generate HTML ‘on-the-fly’
- Interpreted, NOT compiled

Examples:

JavaScript, VBScript, Perl & PHP

JavaScript

- Object-based scripting language
- Relatively simple
- Data Types: numeric, String, and boolean values
- Syntax similar to Java
- No API functions that interact with the filesystem

JavaScript IS NOT Java

JavaScript vs. Java

JavaScript

- Code sent to client; Interpreted by client (browser)
- Object-based. Built in, extensible objects, but no class inheritance
- Code embedded in HTML
- Loose typing (variable data types not declared)
- Dynamic binding. Object references checked at runtime.
- Cannot automatically write to hard disk

Java

- Compiled on server before execution on client
- Object-oriented. Object classes with inheritance.
- Code distinct from HTML
- Strong Typing (variable data types must be declared)
- Static binding. Object references must exist at compile time
- Cannot automatically write to hard disk

VBScript

- Virtually identical to JavaScript
- Syntax similar to Visual Basic instead of Java
- No API functions that interact with the file system
- Client side scripts – Mac, *nix, Netscape do not handle VBScript (Internet Explorer alone)
- Server side – mostly used with ASP

Perl & PHP

- Perl combines features of C and Unix utilities
- The most widely used languages for server-side programming
- Founded on Unix, but now cross-platform
- PHP is HTML embedded Perl scripting language
- Very popular – Apache HTTP Server, PHP, and mySQL or PostgreSQL (very simple and quick)

Common Gateway *Interface* (CGI)

- CGI : A specification for transferring information between a Web server and a CGI program

- Program accepts information from STDIN and outputs to STDOUT (web server)
- Output must also first send MIME header
- Since it is a specification, any language can be used; Perl, however, is by far the most common
- Using a CGI script is transparent to the user (web browser)

CGI (continued)

Steps in CGI script execution:

- 1: user initiates the CGI script
- 2. browser contacts server asking for permission to use script
- 3. server checks user permission and that script exists
- 4. server prepares ENV variables and launches script
- 5. script executes and reads ENV and STDIN
- 6. script sends MIME header and contents to STDOUT
- 7. server sends data in STDOUT to browser and closes connection
- 8. browser displays information

CGI (continued)

Passing information to a CGI script:

- Command line : HTML provides ISINDEX tag (must be placed inside <HEAD>)
- Environment variables : QUERY_STRING contains name > values from URL
  - http://localhost/test.pl?var1=val1&var2=val2 … **QUERY_STRING** now contains var1=val1&var2=val2

- ENV is the most popular and easiest; data must be parsed to get relevant information

Advantages of CGI

- *de facto* standard for interfacing web servers with external programs
- Simplicity
- Language independence (minor)
- Web server independence (minor)
- Wide acceptance

Disadvantages of CGI

- Communication between client and DB server must always go through Web Server; this creates a bottleneck
- Lack of efficiency and transaction support (inherited statelessness from HTTP)
  - Validating user input
  - Form filling
- Server must create new process for each CGI script
  - Large overhead
  - Concurrency issues
- Security
  - If script forks a shell, passed parameters can cause serious damage
  - Because GET is used, sometimes hackers can hack the script

Cookies

- Cookies store information on the client by the server
- Application programmers can store information and retrieve it if needed
- Clients can disable use of cookies
- Used heavily in all methods of web development (sessions, customization, login information, browsing patterns)
- Think of it as a persistent ENV table associated with each server
- Mostly insecure, and should be used with care

Extending Web Server with APIs

- Also called non-CGI gateways
- Can be better than CGI (if API is good), but is much more complicated
- Must use proprietary software
  - Netscape's LiveWire Pro
  - Microsoft's IIS
- A better alternative is JSP/Servlets or ASP/ADO

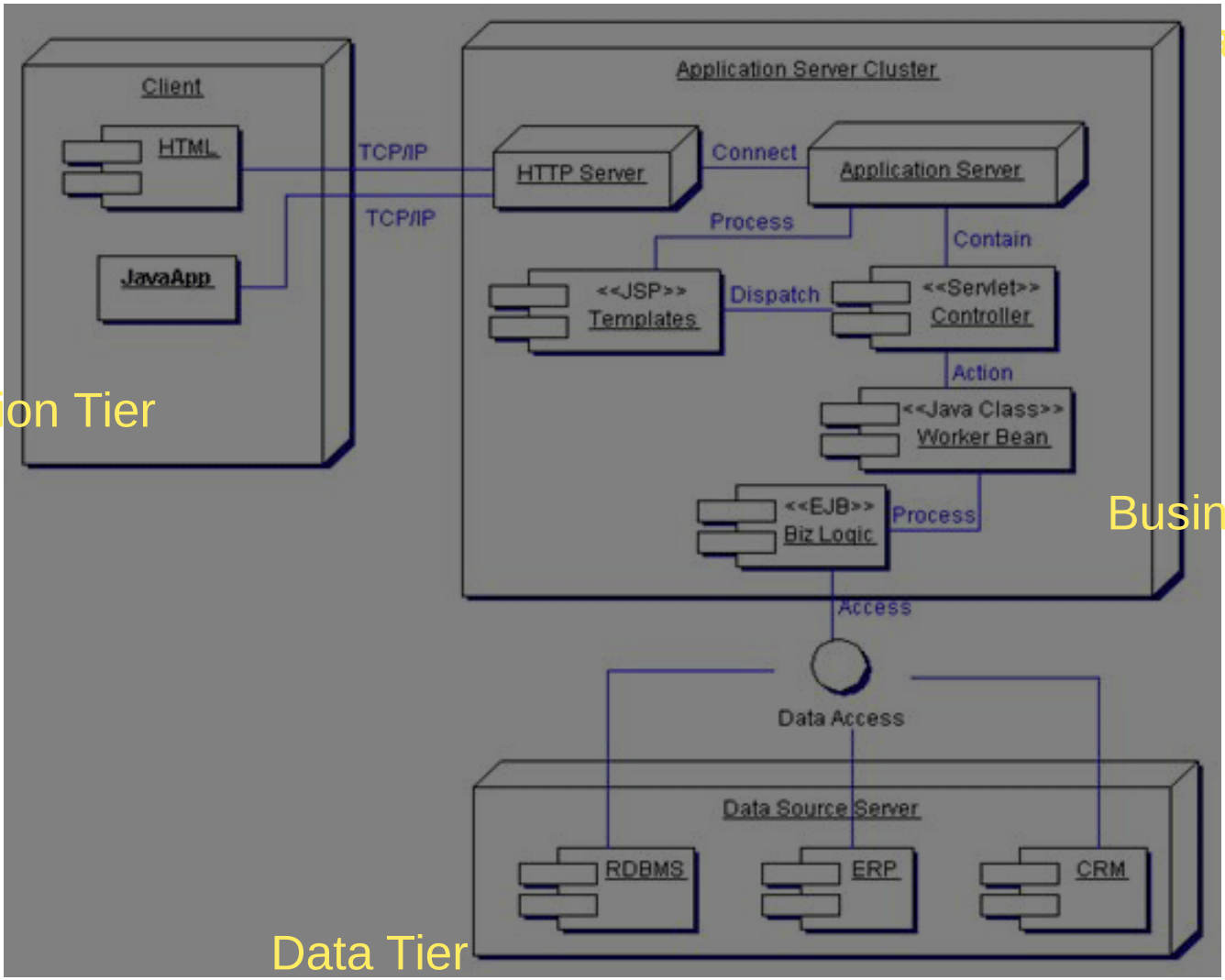
- Examples:
  - NetscapeAPI (NSAPI)
  - Microsoft Internet Information Server API (ISAPI)

Java EE, Servlets, JSP

Java

- Allows for development of Web Applications using tested design patterns (MVC)
- Separates Presentation from Model/Controller
- Allows for multiple views (HTML, Swing, GTK+, etc) to be applied to single application
- Platform independent
- Relies on Bean and Enterprise Java Beans (EJB)

Java EE Architecture



Enterprise Java Beans

Java

- Server-side component architecture for the business tier, encapsulating business and data logic

- EJB Session Beans: components implementing business logic, business rules, and workflow
  - Lives for the lifetime of the session; can be used by only 1 client at a time
  - Performs order entry, banking transactions, DB operations, etc.
- EJB Entity Beans: components encapsulating some data contained by the Enterprise
  - Persistent: may live longer than the session; may be shared by multiple clients
  - Bean-Managed Persistence (BMP) Entity Beans: component developer writes code to make bean persistent, using JDBC, or Java serialization
  - Container-Managed Persistence (CMP) Entity Beans: persistence is provided automatically by the container

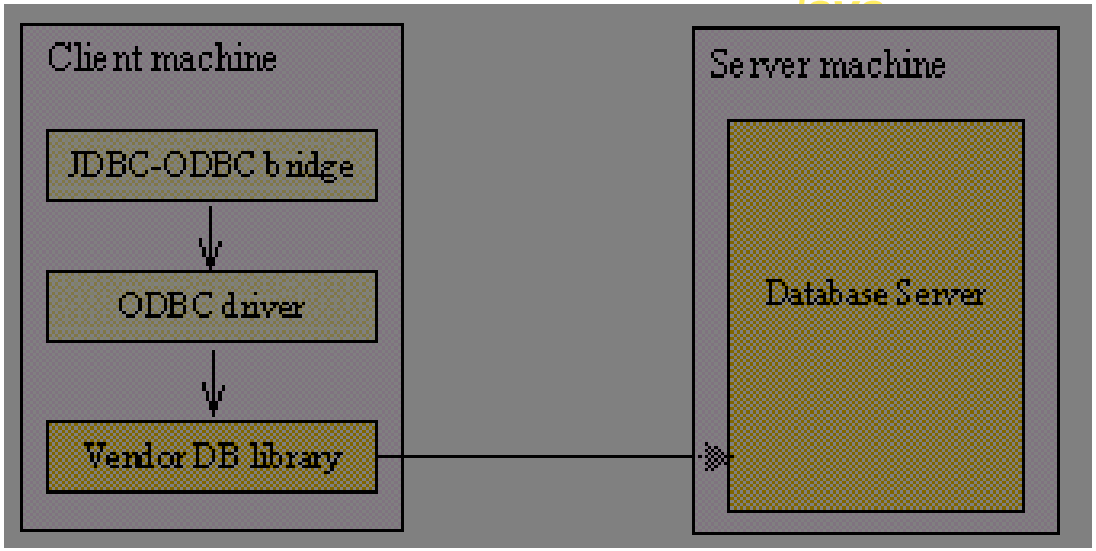
JDBC

Java

- Most prominent and mature approach for accessing R-DBMS
- Modeled after ODBC

- Embedded SQL for Java – JDBC requires SQL statements be passed as strings to Java methods
- Direct mapping of R-DBMS tables to Java classes – each row of the table becomes an instance of that class, and each column value corresponds to an attribute of that instance

JDBC-ODBC Bridge



•Provides JDBC access using ODBC drivers

•Performance overhead associated with translation between JDBC and ODBC

•User is limited by the functionality of underlying ODBC driver

•ODBC drivers are VERY common (more common than JDBC drivers)

•Can use existing components (ODBC) instead of writing new drivers (native JDBC drivers)

Partial JDBC driver

Java

- Converts JDBC calls to calls on the client API for the DBMS (skips ODBC, and directly uses vendor driver)
- Better performance than JDBC-ODBC bridge
- Requires DB client software to be installed on each client

Java JDBC driver for DB Middleware

Java

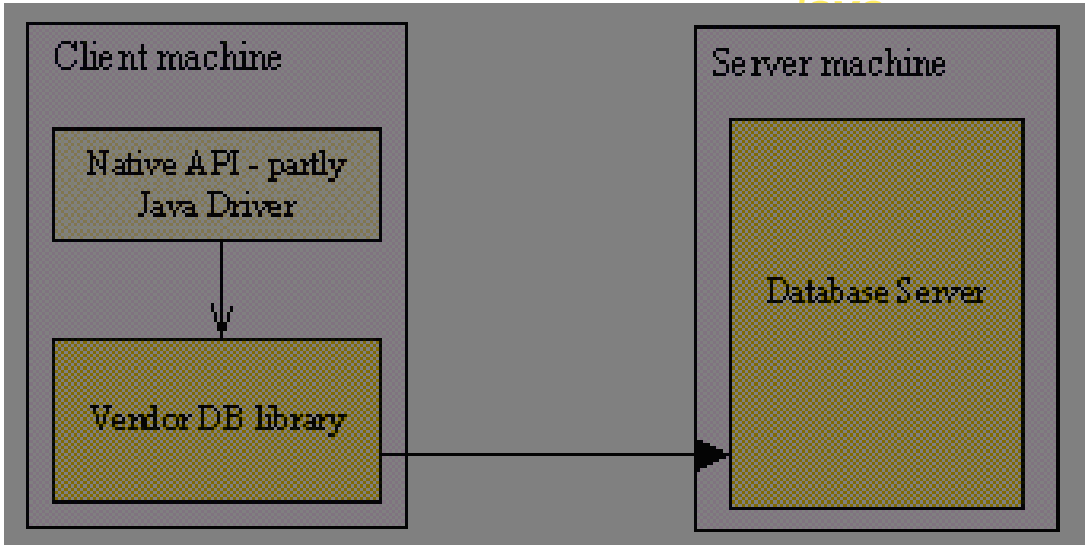
- Translates JDBC calls into middleware vendor's protocol, which is translated to DBMS protocol
- In general, most flexible JDBC alternative
- Middleware provides connectivity to many different DBs

Native Java JDBC

•Converts JDBC calls into network protocol used directly by DBMS

•Allows direct call from client to DBMS server

•Drivers can be downloaded dynamically



•Drivers are completely implemented in Java; therefore completely platform-independent

•Limits deployment issues (since completely native Java)

•Requires a different driver for each DBMS

•Drivers come from DB vendors; most have implemented these

Advantage: pure JDBC

Java

- If a pure JDBC driver exists from the Database Vendor, it should be used instead of JDBC-ODBC bridge
- Better integration
- Little overhead

Java Servlets

Java

- Similar to CGI, with many added benefits (especially with increased number of users):

- Improved performance
  - Compiled code instead of interpreted code
  - Handled by thread of JVM instead of new process
  - Remains in memory, instead of being loaded for each request
- Extensibility
  - Employs fully object-oriented language
  - Larger set of APIs to work from (JDBC, email, directory servers, etc)
- Simpler session management
  - Whereas CGI uses cookies, Servlets maintain persistence until the web server shuts down
  - Session management is handled through the web server instead of by the developer
- Improved security
  - Java's implementing security model
- Improved reliability
  - Java's inherent type checking/safety

Java Servlets (continued)

Java

Mainly Java code with embedded HTML: HelloWorldServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorldServlet
    extends HttpServlet
{
    public void service(HttpServletRequest req,
        HttpServletResponse resp)
    {
out.println("<HTML>");
out.println("<HEAD>");
out.println("<TITLE>HelloWorld Servlet</TITLE>");
out.println("</HEAD>");
out.println("<BODY>");
out.println("hello world!");
out.println("</BODY>");
out.println("</HTML>");
    }
}
```

JavaServer Pages (JSP)

Java

- Whereas Servlets are mainly Java code with embedded HTML, JSP are mainly HTML with embedded Java
- Allows for separation of presentation from development (different teams with different skill sets)

- Scriptlets
  - <% ... %>
- Directives
  - <%@ ... %>
- Actions
  - <jsp:useBean ...> ... </jsp:useBean>

- JSP engine compiles JSP into a Servlet upon first request
- After which the JSP behaves exactly as a Servlet

JSP (continued)

Java

Mainly HTML with embedded Java: HelloWorld.jsp

```
<HTML>
<HEAD>
 <TITLE>HelloWorld JSP</TITLE>
</HEAD>
<BODY>

<%= "Hello World!" %>

</BODY>
</HTML>
```

Microsoft Web Solution Platform

Microsoft

- Object Linking and Embedding (OLE)
  - Object-oriented technology enabling development of reusable software components
- Component Object Model (COM)
  - OLE extension that allow services to be OLEs
  - Object-based model consisting of both a specification defining interface between objects and system, and a concrete implementation
  - Packaged as a Dynamic Link Library (DLL)
- Distributed COM (CDOM)
  - Allows COM architecture across the Enterprise
  - Replaces inter-process communication (IPC) between component and client with appropriate network protocol
- Web Solution Platform (COM+)
  - Provides more application infrastructure, allowing developers to focus on core application logic
  - ASP and ADO are core components of this architecture

Active Server Pages (ASP)

Microsoft

- Analogous to JSP, but developed by MS
- Initially only supported by IIS, but now Apache supported
- Supports ActiveX scripting and ActiveX components (which are readily available)
- Not compiled upon first use (ASP engine must process every time *.asp is requested)
- Runs in thread forked by the Web Server instead of in separate process

ActiveX Data Objects (ADO)

Microsoft

- ASP extension supported by IIS for database connectivity
  - combines RDS & ADO

Key Features:

- Independently created objects
- Support for stored procedures
- Different cursor types, including potential for support of different back-end specific cursors
- Batch updating
- Support for limits on numbers of returned rows and other query goals
- Support for multiple record sets returned from procedures or batch statements

Benefits:

- Ease of use
- High speed
- Low memory overhead
- Small disk footprint

Remote Data Services (RDS)

Microsoft

- Technology for client-side database manipulation (primarily across the Internet)
- Mechanism to directly interact with the database at the client level
- Implemented as a client-side ActiveX control, included with Internet Explorer

Example:

```
<OBJECT CLASSID="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33" ID="ADC">
 <PARAM NAME="SQL" VALUE="SELECT * FROM employees">
 <PARAM NAME="Connect" VALUE="DSN=EmployeeDB;">
 <PARAM NAME="Server" VALUE="localhost">
</OBJECT


<TABLE DATASRC="#ADC">
 <TR><TD><Span DATAFLD="empID"></SPAN></TD></TR>
</TABLE>
```

Microsoft Access

Microsoft

- Export wizards for automatically generating HTML based on data

- Static pages
  - Export data to *.html
  - Pages can become out of date quickly
  - Can use templates to customize pages
- Dynamic pages, using ASP
  - Export data to *.asp
  - Data will be generated dynamically
- Dynamic pages, using DAP (data access pages)
  - Used like access forms
  - Written in Dynamic HTML (DHTML)
  - Requires > Internet Explorer 5.0

Future of ASP and ADO (.NET style)

Microsoft

ASP.NET (more like current JSP and Java Servlets)

- Language-neutral common runtime framework
- Web forms
- Web services
- Rich controls (server-side *complicated* HTML generation controls; i.e. Calendars)
- Server controls

ADO.NET

- Connected Layer (similar to ADO)
  - Uses XML to exchange data
- Disconnected Layer (similar to RDS)
  - Maintains relationship information in memory

JSP vs. ASP

JSP

- Platform-independent
- Easier portability
- Extensible tags (custom tag libraries)
- EJBs, JavaBeans, and custom tags reusable across platforms
- Potentially more reliable

ASP

- Primarily Microsoft platform
- Easier Interoperability
- Non-extensible tags
- ActiveX Controls not reusable (windows platform only)