Internet Programming

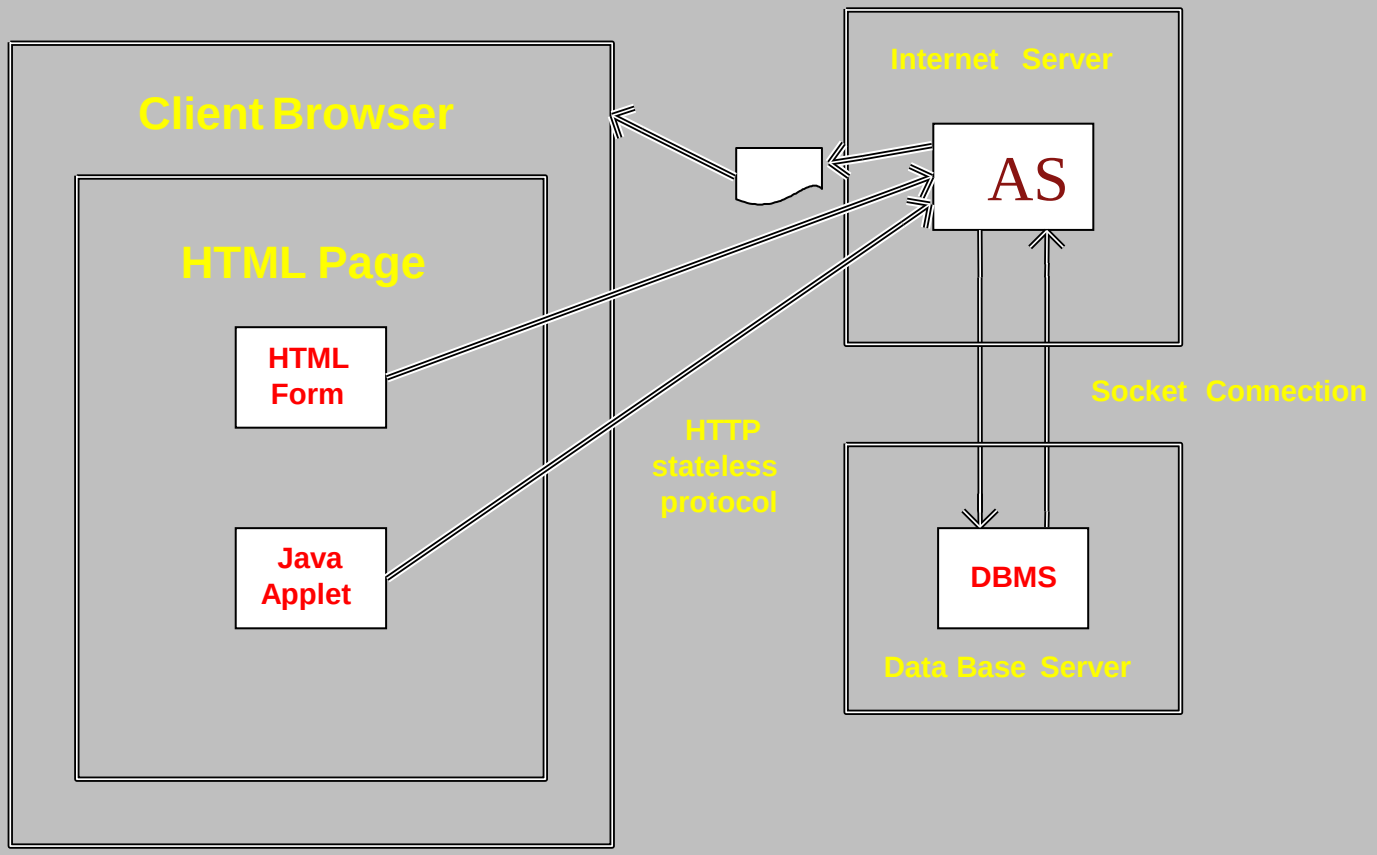
Server Side Processing

Dan Brandon, Ph.D., PMP

Server Program Execution

- Application Server or API (Application Program Interface) – Software becomes “part of information server” process
  - ISAPI (Active Server Pages, etc.)
    - Microsoft specific
    - Uses *.dll’s
    - Master process started once
  - NSAPI (Java Products, PHP, Cold Fusion, etc.)
    - Platform independent
    - Master process started once
- CGI (C, C++, PERL, etc.)
  - General solution
  - Separate host processes (with separate memory areas) for each program executing

DB Query/Update via HTTP-API or CGI



Internet Server Processing Options

- PHP or Python (also RubyRails, NodeJS, etc.)
  - All platforms
  - Language (within tag) based
- Active Server Pages
  - Microsoft only
  - Language (within tag) based
  - Can get some Unix versions but not fully compatible
- Java Servlets
  - All Platforms
  - Language based
- Java Server Pages
  - All Platforms
  - Language (within tag) based
- Cold Fusion
  - All Platforms
  - Tag based (some language capabilities)

Web Servers

- Web servers set up a client/server communication between browser clients and processes on an information server
- Clients make requests using the HTTP protocol, a connection is opened, the request is serviced by sending information back to the browser, and then the connection is closed
- As well as communicating with clients (on devices running browsers), Web (Information) Servers - organize and administer internet resources (directories and files consisting of html pages, executable programs, images, and other web related information)

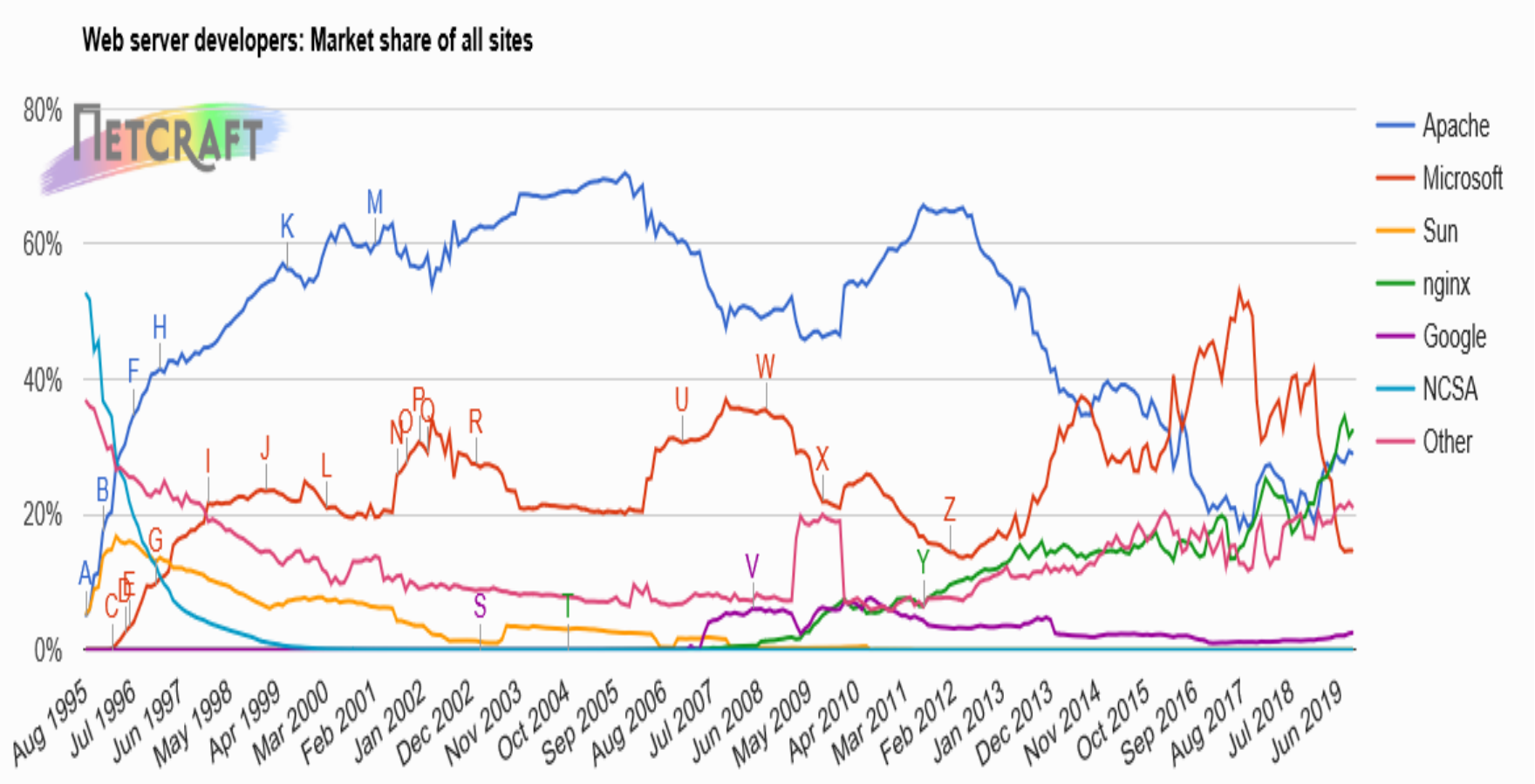
Typical Microsoft Web Server

- Windows Server Operating System
- Web Server - IIS
- Server Processing
  - ISAPI (direct calls with C++)
  - Active Server Pages
  - PHP
  - Python
  - Java Server Pages
  - Java Servlets
  - Cold Fusion
- Database Access (ODBC, ADO, OLE/DB, JDBC, native)
  - SQLServer
  - Access
  - MySQL (open source)

Typical Unix Web Server

- Operating System - Unix (Sun, HP, Linux, AIX, …)
- Web Server - Apache, NGINX, NCSA
- Server Processing
  - CGI (Perl, C/C++)
  - NSAPI (direct calls with C++)
  - PHP
  - Python
  - Java Server Pages
  - Java Servlets
  - Cold Fusion
- Database (ODBC, JDBC, native)
  - Oracle
  - Sybase
  - DB2
  - MySQL (open source)

Web Server Market Share



nginx

- NGINX is a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server
- NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption.
- Unlike traditional servers, NGINX doesn't rely on threads to handle requests. Instead it uses a much more scalable event-driven (asynchronous) architecture. This architecture uses small, but more importantly, predictable amounts of memory under load.
- It is used on many heavily loaded Russian sites including Yandex, Mail.Ru, VK, and Rambler
- Also used by: Dropbox, Netflix, Wordpress.com, FastMail.FM

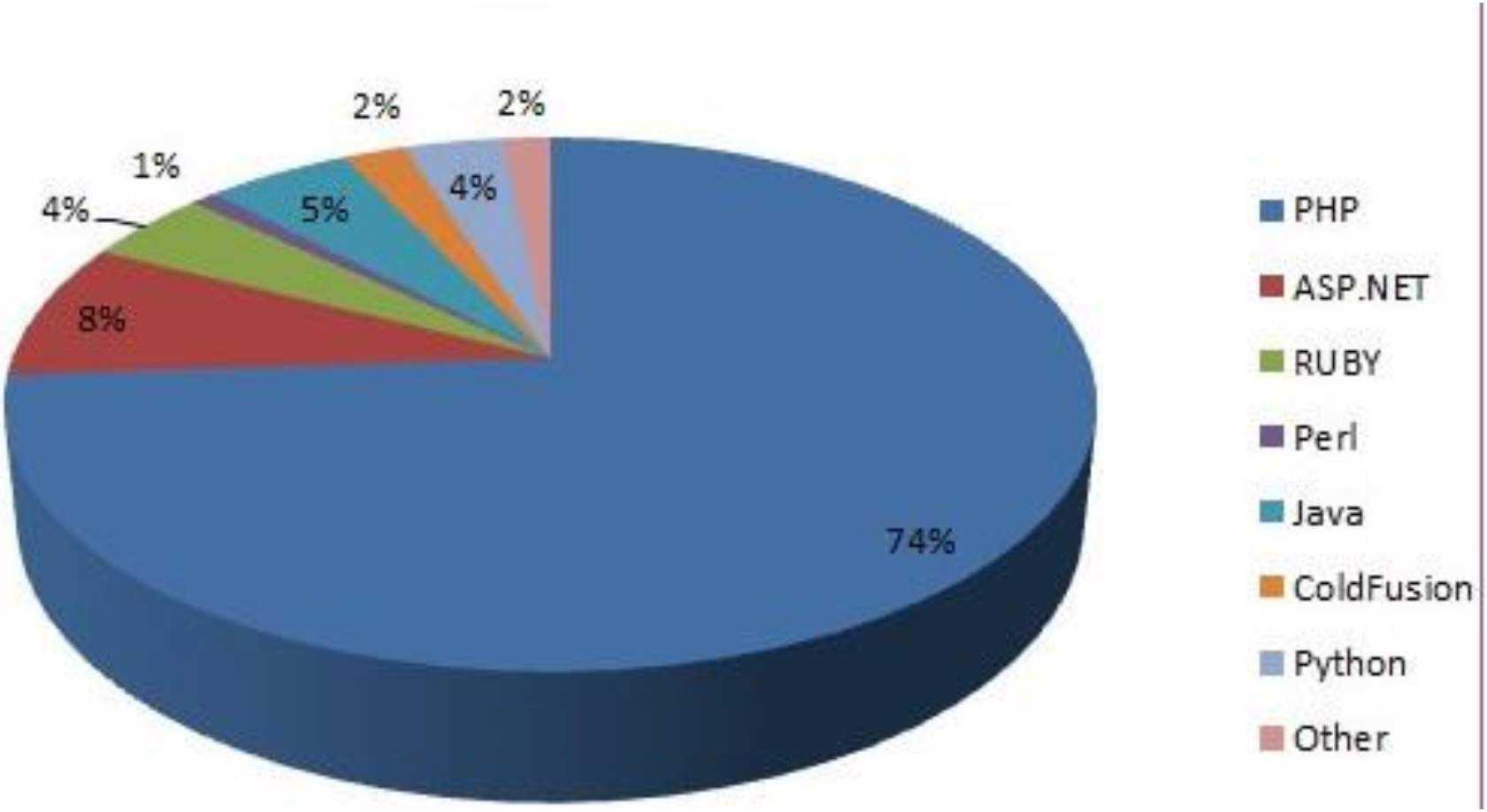
Rasmus Lerdorf, who wrote the original Common Gateway Interface (CGI) component, together with Andi Gutmans and Zeev Suraski, who rewrote the parser that formed PHP 3.

- PHP Hypertext Processor
- Open Source
- Platform independent
- Developed by Rasmus Lerdorf in 1994
- Server Side Scripting Language using “Zend” [www.zend.com] – much more powerful than Perl
  - Perl usually requires add-on libraries, PHP contains basic functions already
- Excellent native interface to MySql database and LDAP (lightweight directory access protocol)

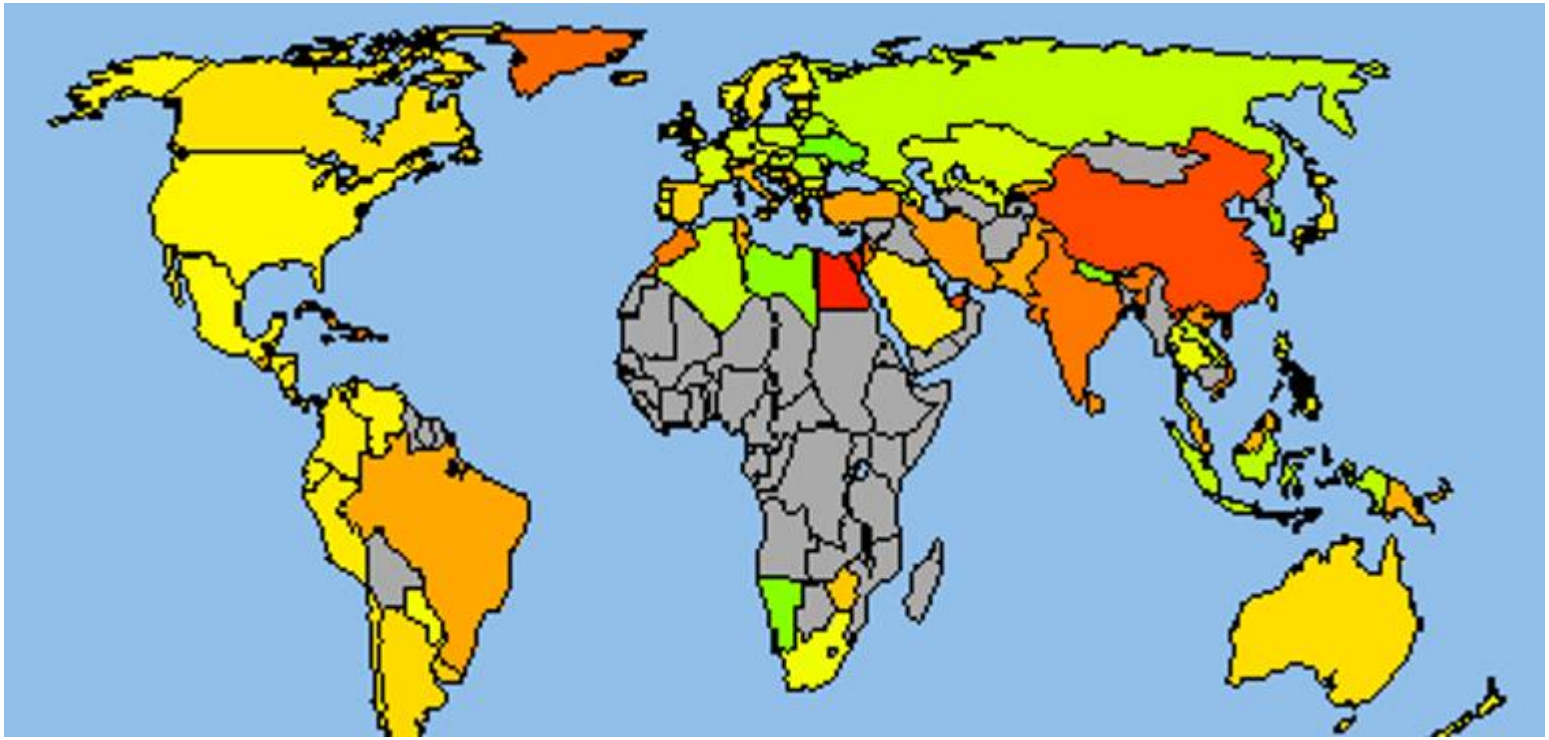
PHP (con't)

- Originally, PHP stood for Personal Home Page, but eventually evolved to "PHP: Hypertext Preprocessor"
- PHP is the most used server-side language on the Web, driving 77% of all database-driven websites (including Facebook and Wikipedia)
- Now installed on more than 244 million websites and 2.1 million web servers

PHP Market Share

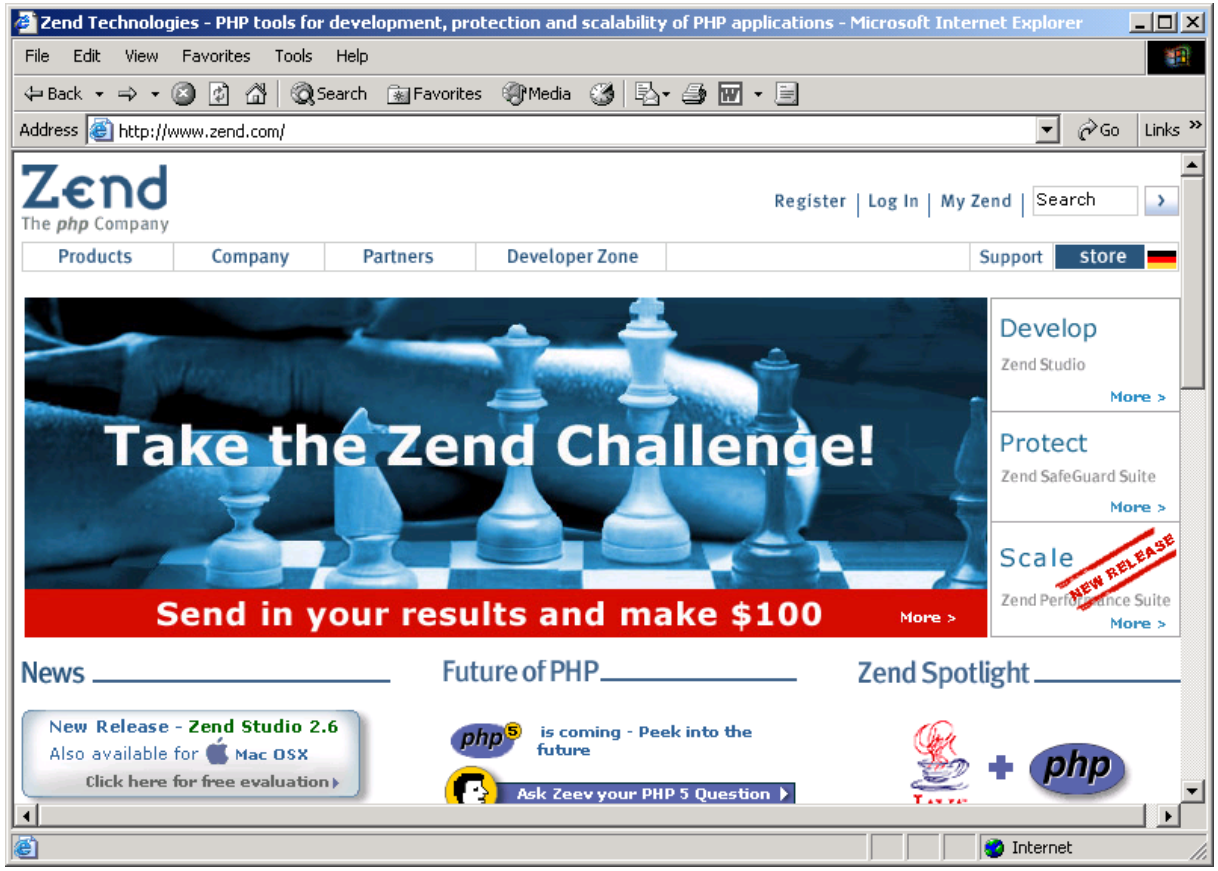


PHP Worldwide Usage



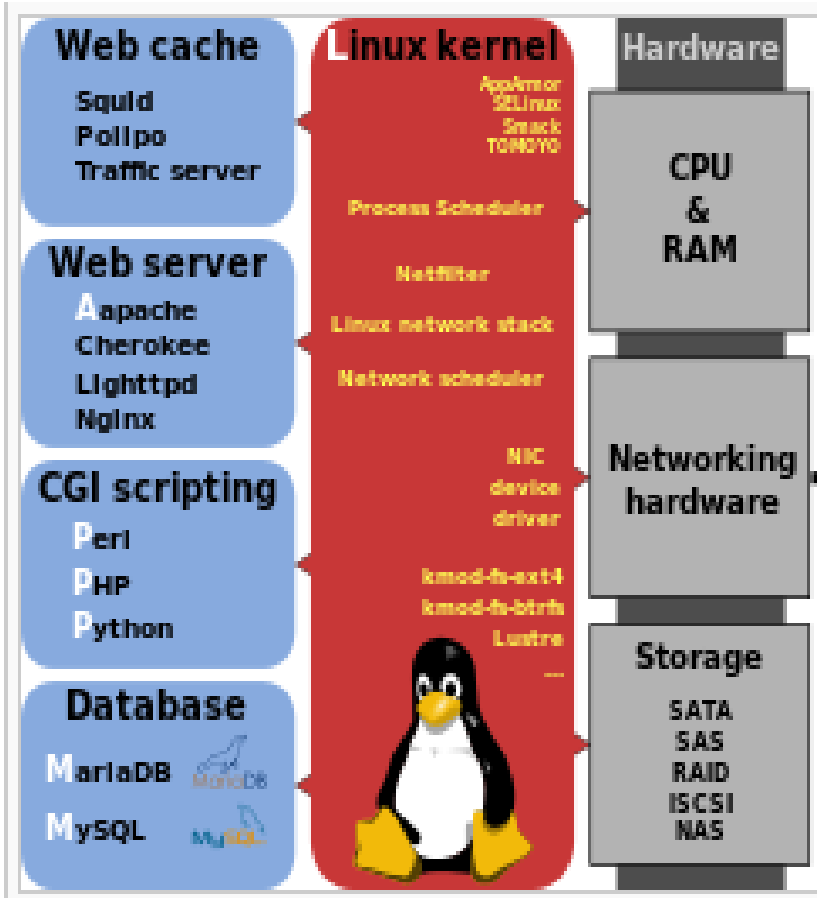
Green = high (80%+) Red = low (5%-)

Zend Web Site



LAMP Framework

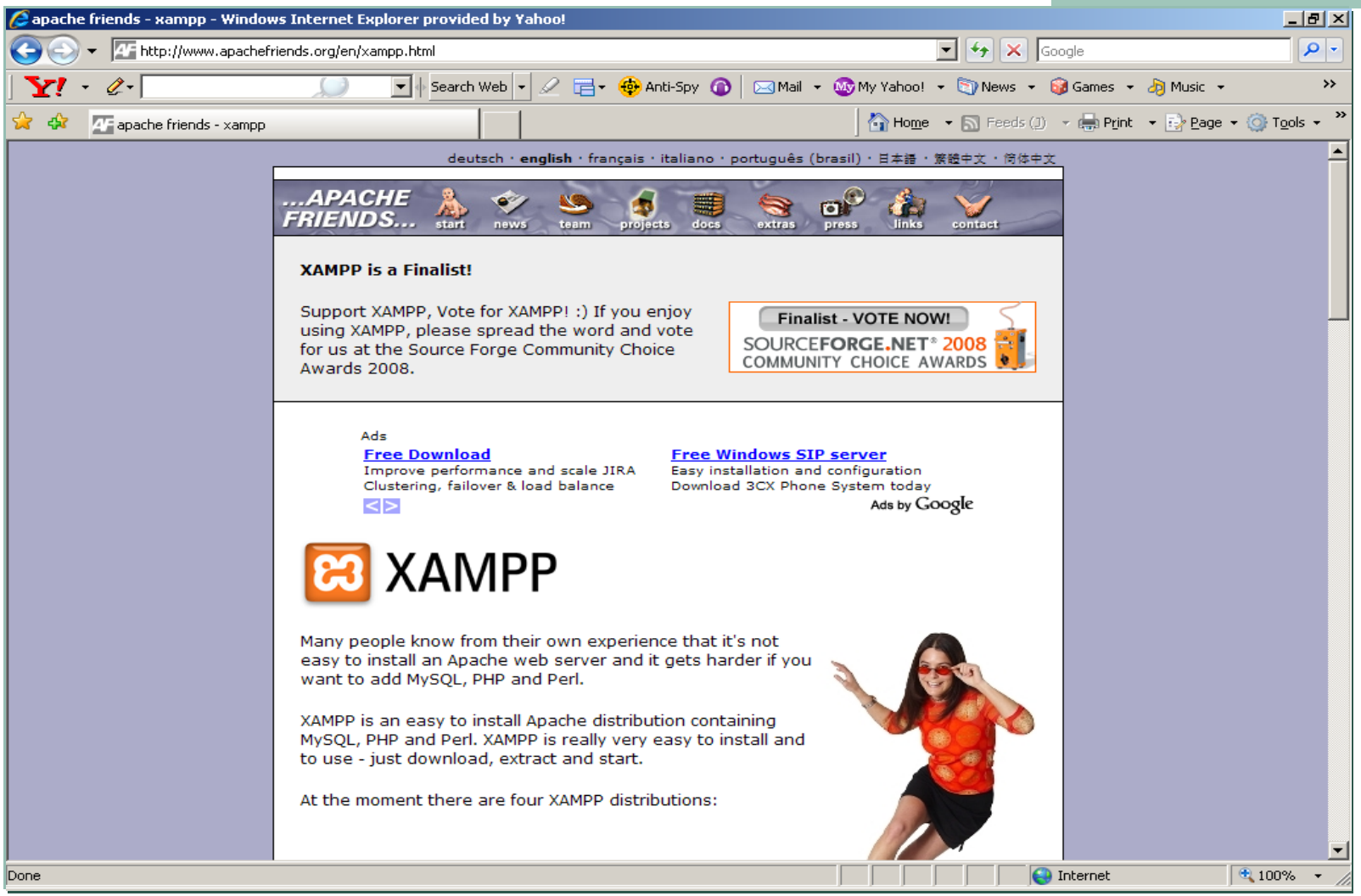
[Linux-Apache-MySQL-PHP]



PHP Usage

- PHP runs on the web server
- You can install PHP on your own client, and emulate server side processing by running a local server instance of IIS (or Apache) and PHP – that way you can test your server side PHP programs by using your PC client as a server
- There are also consolidated client/server install programs such as XAMPP

Consolidated Install



Server-side Scripting Languages

- PHP is a server-side scripting language (can also be run from command line)
- C type syntax
- “php” filename extension
- PHP can be included with HTML/XML tags or not
- A server-side scripting language is similar to JavaScript in many ways, as they both allow you to embed little programs (scripts) into the HTML of a Web page
- When executed, such scripts allow you to control what will actually appear in the browser window with more flexibility than is possible using straight HTML

Server-side Scripting Languages (con't)

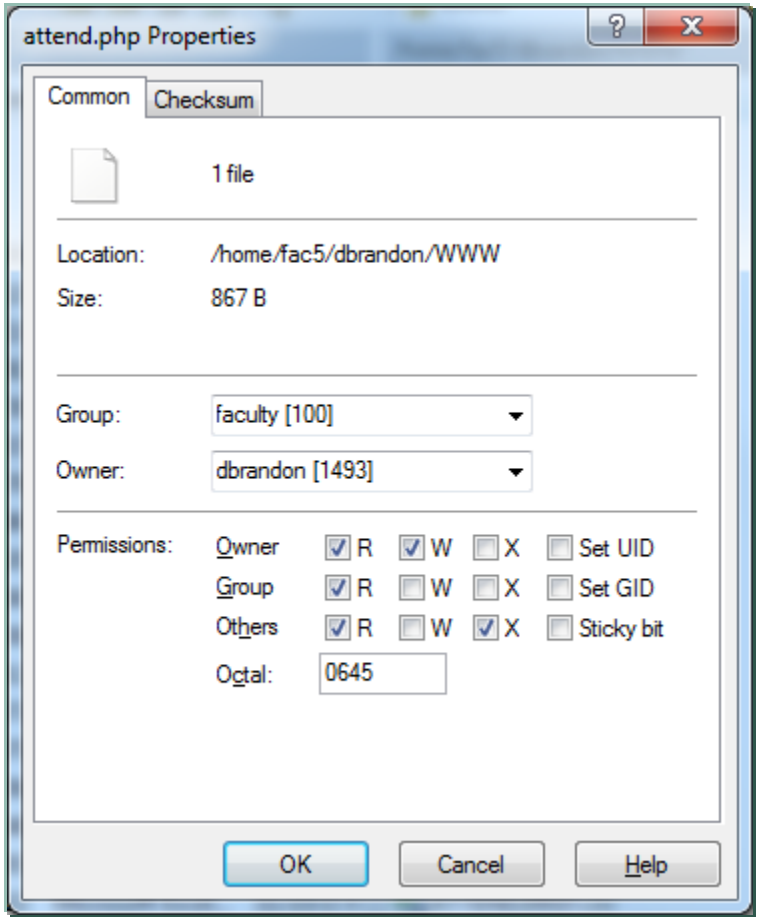
- The key difference between JavaScript and PHP is that JavaScript (a client side scripting language) is interpreted by the Web browser once the Web page that contains the script has been downloaded
- Meanwhile, server-side scripting languages like PHP are interpreted by the Web server before the page is even sent to the browser
- Once it's interpreted, the results of the script replace the PHP code in the Web page itself, so all the browser sees is a standard HTML/JavaScript file
- The PHP script is processed entirely by the server, hence the designation: server-side scripting language

PHP TAGS

- Starting Tag:
  - <?php
- Ending Tag:
  - ?>
- Within an HTML document:
  - <HTML><HEAD>…</HEAD>
  - <BODY>
  - <?php
    - php statements …
  - ?>
  - </BODY>
  - </HTML>

Unix File Permissions

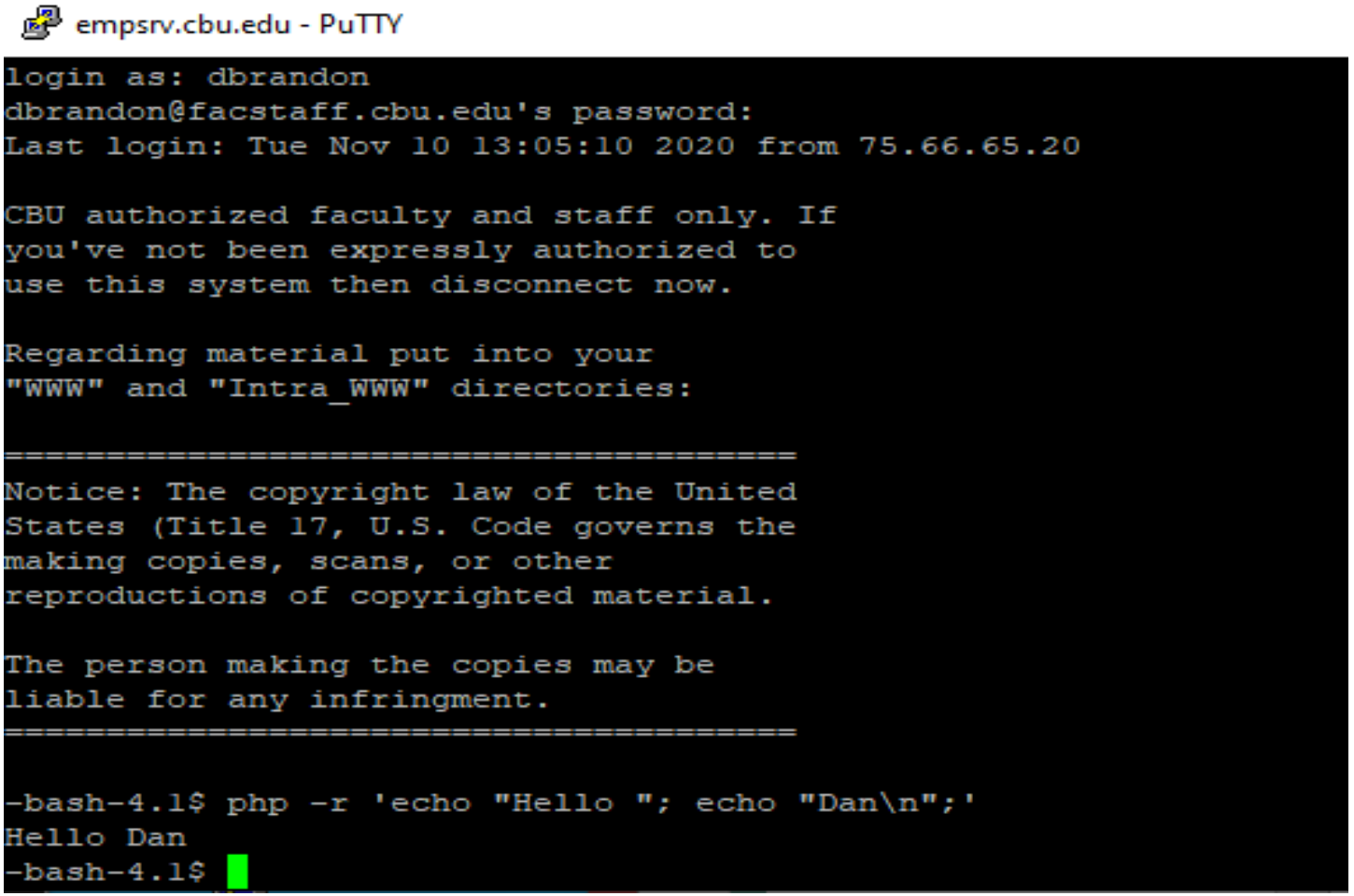
- PHP programs are placed under the “web root” – for the CBU sheba server this is your WWW folder
- PHP programs need to have read and execute permissions for all



PHP Command Line Execution

- There are several different ways to execute PHP from the Unix/Linux command line:
- Executing a literal command:
  - php -r 'echo "Hello "; echo "Dan\n";'
- Executing a php script file:
  - $ php my_script.php
  - Note that there is no restriction on which files can be executed; in particular, the filename is not required have a .php extension

Command Line PHP via Putty



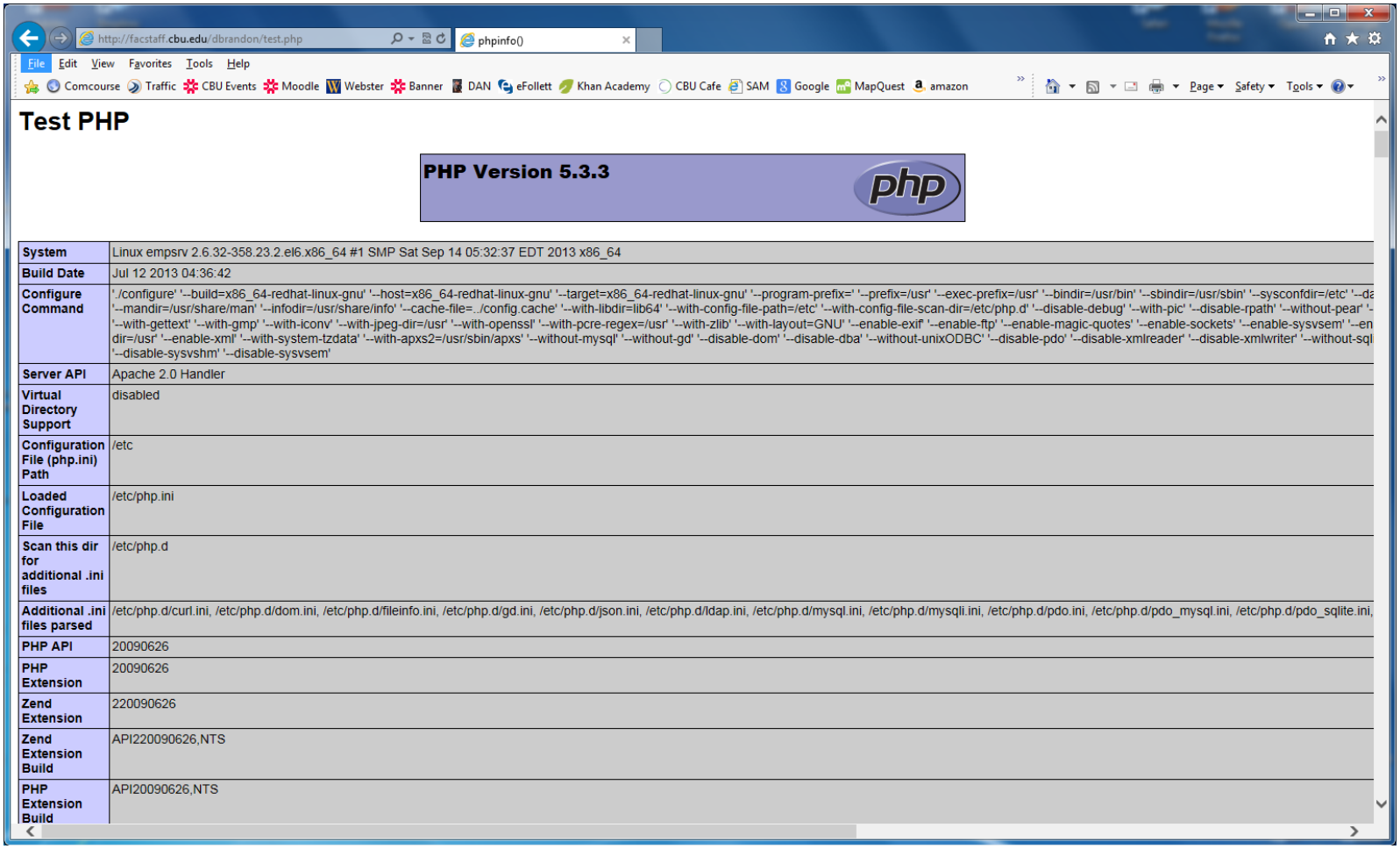
Web Server PHP Testing Program (test.php)

- <HTML>
- <BODY>
- <H1>Test PHP</H1>
- <P>
- <?php phpinfo(); ?>
- </BODY>
- </HTML>

Use Notepad to create this text file
Name it test.php
FTP it to your WWW (public-html) folder on server
Set permissions

Open in browser:
URL: your-url/test.php

http://facstaff.cbu.edu/dbrandon/test.php



System	Linux empsrv 2.6.32-358.23.2.el6.x86_64 #1 SMP Sat Sep 14 05:32:37 EDT 2013 x86_64
Build Date	Jul 12 2013 04:36:42
Configure Command	'./configure' '--build=x86_64-redhat-linux-gnu' '--host=x86_64-redhat-linux-gnu' '--target=x86_64-redhat-linux-gnu' '--program-prefix=' '--prefix=/usr' '--exec-prefix=/usr' '--bindir=/usr/bin' '--sbindir=/usr/sbin' '--sysconfdir=/etc' '--da '--mandir=/usr/share/man' '--infodir=/usr/share/info' '--cache-file=../config.cache' '--with-libdir=lib64' '--with-config-file-path=/etc' '--with-config-file-scan-dir=/etc/php.d' '--disable-debug' '--with-pic' '--disable-rpath' '--without-pear' '- '--with-gettext' '--with-gmp' '--with-iconv' '--with-jpeg-dir=/usr' '--with-openssl' '--with-pcre-regex=/usr' '--with-zlib' '--with-layout=GNU' '--enable-exif' '--enable-ftp' '--enable-magic-quotes' '--enable-sockets' '--enable-sysvsem' '--en dir=/usr' '--enable-xml' '--with-system-tzdata' '--with-apxs2=/usr/sbin/apxs' '--without-mysql' '--without-gd' '--disable-dom' '--disable-dba' '--without-unixODBC' '--disable-pdo' '--disable-xmlreader' '--disable-xmlwriter' '--without-sql '--disable-sysvshm' '--disable-sysvsem'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/curl.ini, /etc/php.d/dom.ini, /etc/php.d/fileinfo.ini, /etc/php.d/gd.ini, /etc/php.d/json.ini, /etc/php.d/ldap.ini, /etc/php.d/mysql.ini, /etc/php.d/mysqli.ini, /etc/php.d/pdo.ini, /etc/php.d/pdo_mysql.ini, /etc/php.d/pdo_sqlite.ini,
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626
Zend Extension Build	API220090626,NTS
PHP Extension Build	API20090626,NTS

PHP Program (all in PHP)

[you can mix php and HTML/JS or not]

- <?php
  - echo “<HTML>”;
  - echo “<BODY>”;
  - echo “<H1>Test PHP</H1>”;
  - echo “<P>”;
  - phpinfo();
  - echo “</BODY>”;
  - echo “</HTML>”;
- ?>

PHP Syntax

- C type syntax
- Case sensitive
- Statements end with ;
- C or C++ style comments
- Operators same as C/Java
- Conditional and Looping similar to C/Java
- When a variable is inside a double-quoted string PHP replaces the variable name with its value
- Output (to standard output, which is mapped to the web server HTTP socket [80]) is via either the “echo” or “printf” functions

PHP Variables

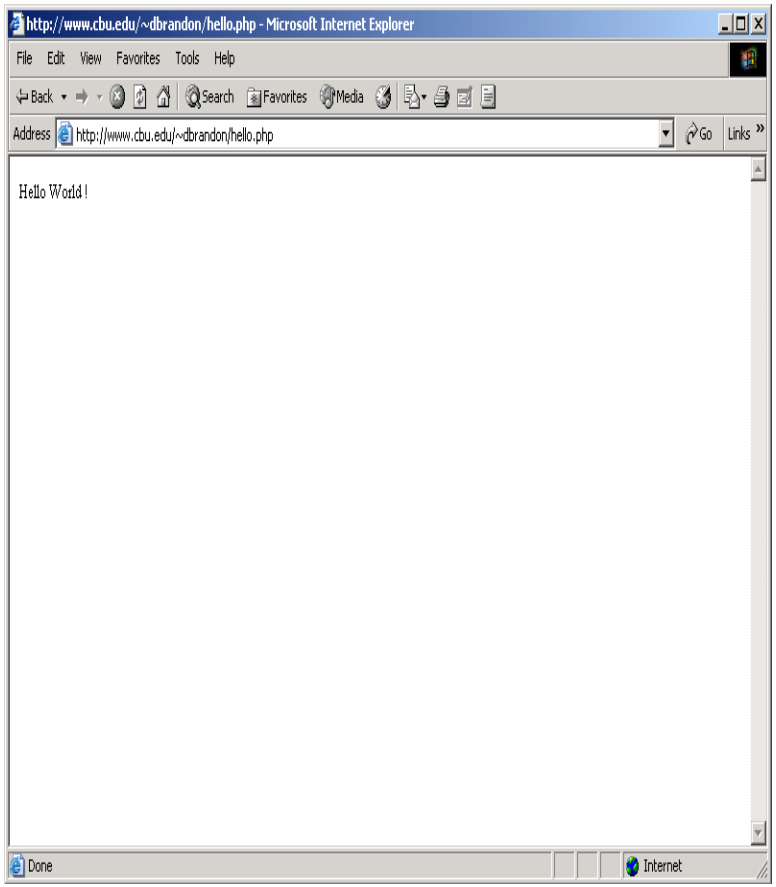
- Start with $
- Case sensitive
- Camel notation, all caps for constants (“define” function)
- Type is determined by the value equated to the variable unless a conversion is made via the “settype” function which results in a permanent change (can do a temporarily via casting)
  - Integer, Double, String, Boolean
- Arrays (incl. associative arrays), Objects, Resources and Nulls also supported
- Strings can be concatenated via the period (“.”)
- Environment variables are stored in an array called $GLOBALS

String Concatenation

Operator	Languages
+	ALGOL 68, BASIC, C++, C#, Cobra, Pascal, Object Pascal, Eiffel, Go, JavaScript, Java, Python, Turing, Ruby, Rust, Windows PowerShell, Objective-C, Swift, F#, Scala, Ya
++	Haskell, Erlang
$+	mIRC Scripting Language
&	Ada, AppleScript, COBOL (for literals only), Curl, Seed7, VHDL, Visual Basic, Visual Basic .NET, Excel, FreeBASIC
nconc	Common Lisp
.	Perl, PHP, and Maple (up to version 5), Autohotkey
~	Raku and D
||	Icon, Standard SQL, PL/I, Rexx, and Maple (from version 6)
<>	Mathematica, Wolfram Language
..	Lua
:	Pick Basic
,	J programming language, Smalltalk, APL
^	OCaml, Standard ML, F#, rc
//	Fortran
*	Julia

Class Lab

- Write “Hello World” in PHP
- Use a string variable to contain the “Hello World!” text
- Create source code with text editor on client
- Upload source file to web server WWW folder (file needs read & execute permissions)
- Run via browser
  - URL: your-url/hello.php



Do not look ahead !

Hello World in PHP [hello.php]

- <html> <body>
- <?php
  $myvar = ‘Hello World’;
- echo $myvar;
- ?>
- </body></html>

today.php

```
<html>
<head><title>Today's Date</title></head>
<body>
        <p>Today's Date is
        <?php
            echo( date("l, F dS Y.") );
        ?>
        </p>
</body>
</html>
```

date(*format,timestamp);*

- •d - The day of the month (from 01 to 31)
- •D - A textual representation of a day (three letters)
- •j - The day of the month without leading zeros (1 to 31)
- •l (lowercase 'L') - A full textual representation of a day
- •N - The ISO-8601 numeric representation of a day (1 for Monday, 7 for Sunday)
- •S - The English ordinal suffix for the day of the month (2 characters st, nd, rd or th. Works well with j)
- •w - A numeric representation of the day (0 for Sunday, 6 for Saturday)
- •z - The day of the year (from 0 through 365)
- •W - The ISO-8601 week number of year (weeks starting on Monday)
- •F - A full textual representation of a month (January through December)
- •m - A numeric representation of a month (from 01 to 12)
- •M - A short textual representation of a month (three letters)
- •n - A numeric representation of a month, without leading zeros (1 to 12)
- •t - The number of days in the given month
- •L - Whether it's a leap year (1 if it is a leap year, 0 otherwise)
- •o - The ISO-8601 year number
- •Y - A four digit representation of a year
- •y - A two digit representation of a year
- •a - Lowercase am or pm
- •A - Uppercase AM or PM
- •B - Swatch Internet time (000 to 999)
- •g - 12-hour format of an hour (1 to 12)
- •G - 24-hour format of an hour (0 to 23)
- •h - 12-hour format of an hour (01 to 12)
- •H - 24-hour format of an hour (00 to 23)
- •i - Minutes with leading zeros (00 to 59)
- •s - Seconds, with leading zeros (00 to 59)
- •u - Microseconds (added in PHP 5.2.2)
- •e - The timezone identifier (Examples: UTC, GMT, Atlantic/Azores)
- •I (capital i) - Whether the date is in daylights savings time (1 if Daylight Savings Time, 0 otherwise)
- •O - Difference to Greenwich time (GMT) in hours (Example: +0100)
- •P - Difference to Greenwich time (GMT) in hours:minutes (added in PHP 5.1.3)
- •T - Timezone abbreviations (Examples: EST, MDT)
- •Z - Timezone offset in seconds. The offset for timezones west of UTC is negative (-43200 to 50400)
- •c - The ISO-8601 date (e.g. 2013-05-05T16:34:42+00:00)

Client (form) Data

- $HTTP_USER_AGENT (browser type)
- $REMOTE_ADDR (IP address)
- $_GET (info passed via URL or form get method) [old format -> $HTTP_GET_VARS]
- $_POST (info passed via form post method) [old format -> $HTTP_POST_VARS]
  - In latest version of PHP, one can simply use the form field name (ie. “city”) instead of:
    - “$_POST[‘city’]” or “$HTTP_POST_VARS['city']”
- $HTTP_COOKIE_VARS (cookie data)

PHP Echo Program

- <html><head><title>Form Echo</title></head><body>
- <h1 align='center'>Form Echo</h1>
- <?php
- while ($i = each($_POST)) {
- $k = ($i['key']);
- $v = ($i['value']);
- echo "
Key: ".$k." Value: ".$v."</br>";
- }
- ?>
- </body></html>

PHP Database Access

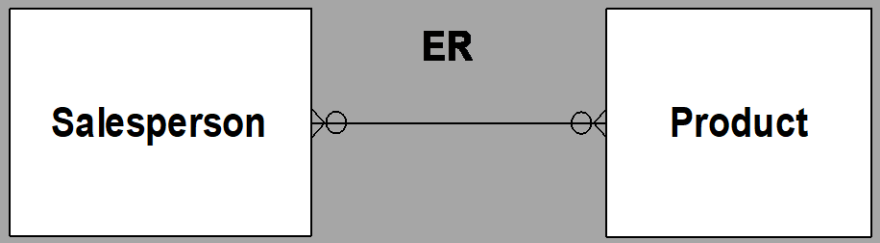
[printf characters preceded by the % sign are placeholders (or tokens); they will be replaced by a variable passed as an argument]

```
<html><body>
<?php
        $db = mysql_connect("localhost", "root");
        mysql_select_db("mydb",$db); //database on selected host
        $sql = 'SELECT * FROM employees WHERE id = 3';
       $result = mysql_query($sql, $db); //perform query $sql on $db
        printf("First Name: %s<br>\n", mysql_result($result,0,"first"));
        printf("Last Name: %s<br>\n", mysql_result($result,0,"last"));
        printf("Address: %s<br>\n",
mysql_result($result,0,"address"));
        printf("Position: %s<br>\n", mysql_result($result,0,"position"));
?>
</body></html>
```

Example Table Creation File

[SQL Create Table Syntax]

```
create table S
( sid char(2) not null primary key,
  sname varchar(30) not null,
  city varchar(15) not null
);
create table P
( pid char(2) not null primary key,
  pname varchar(20) not null,
  size tinyint unsigned not null,
  price decimal(5,2)not null
);
create table SP
( sid char(2) not null,
  pid char(2) not null,
  qty smallint unsigned not null,
  primary key (sid, pid)
);
```



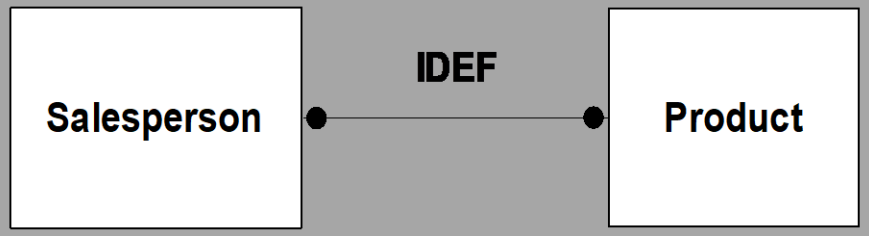


Table Insertion Text File
[SQL Script File]

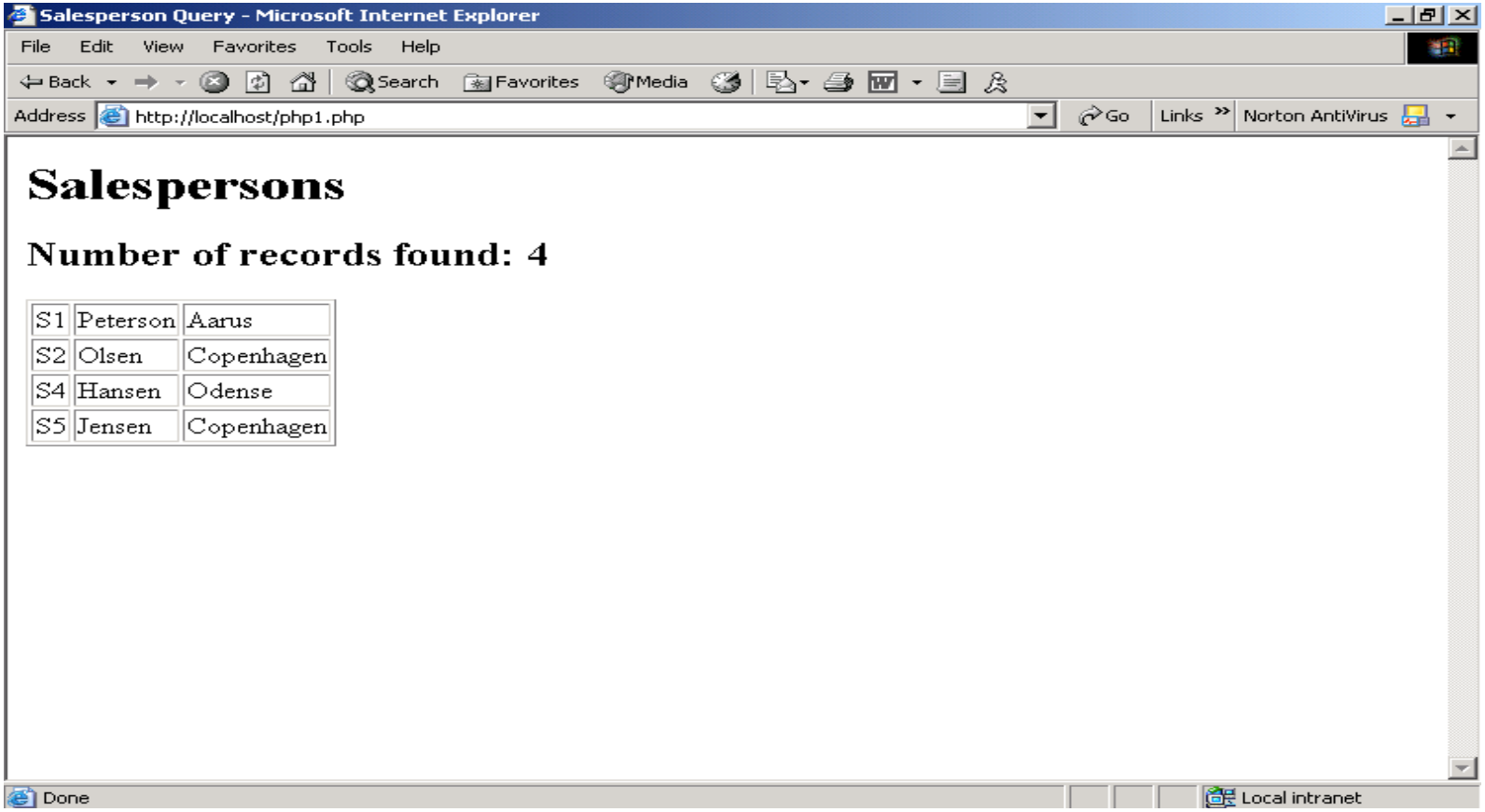
```
use sp;

insert into s values
  ("S1", "Peterson", "Aarus"),
  ("S2", "Olsen", "Copenhagen"),
  ("S4", "Hansen", "Odense"),
  ("S5", "Jensen", "Copenhagen");

insert into p values
  ("P1", "Shirt", 6, 50),
  ("P3", "Trousers", 5, 90),
  ("P4", "Socks", 7, 20),
  ("P5", "Blouse", 6, 50),
  ("P8", "Blouse", 8, 60);

insert into sp values
  ("S2", "P1", 200 ),
  ("S2", "P3", 100 ),
  ("S4", "P5", 200 ),
  ("S4", "P8", 100 ),
  ("S5", "P1", 50 ),
  ("S5", "P3", 500 ),
  ("S5", "P4", 800 ),
  ("S5", "P5", 500 ),
  ("S5", "P8", 100 );
```

PHP Program to Show SalesPersons

[the stripslashes() function removes backslashes; the htmlspecialchars() function converts some predefined characters such as < and > to HTML code]

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<TITLE>Salesperson Query</TITLE>
<META NAME="Author" CONTENT="Dan Brandon">
</HEAD>
<BODY>
<H1>Salespersons</H1>
<?php
        @ $db = mysql_pconnect('localhost', 'student', '');
        if (!$db)
        {
                echo 'Error: Could not conect to database !';
                exit;
        }
        mysql_select_db('sp');
        $query = "select * from s";
        $result = mysql_query($query);
        $num_results = mysql_num_rows($result);
        echo '<P><H2>Number of records found: '.$num_results.'</H2></P>';
        echo '<TABLE BORDER>';
        for ($i=0; $i<$num_results; $i++)
        {
                $row = mysql_fetch_array($result);
                echo '<TR>';
                echo '<TD>'.htmlspecialchars(stripslashes($row['sid'])).'</TD>';
                echo '<TD>'.htmlspecialchars(stripslashes($row['sname'])).'</TD>';
                echo '<TD>'.htmlspecialchars(stripslashes($row['city'])).'</TD>';
                echo '</TR>';
        }
        echo '</TABLE>';
?>
</BODY>
</HTML>
```

PHP Program Output



PHP Form and Database Processing

```
<html><body>
<?php
      if ($submit) {
         // process form
         $db = mysql_connect("localhost", "root");
         mysql_select_db("mydb",$db);
         $sql = "INSERT INTO employees (first,last,address,position) VALUES
                  ('$first','$last','$address','$position')";
          $result = mysql_query($sql);
          echo "Thank you! Information entered.\n";
        } else{
           // display form
   ?>
      <form method="post" action="<?php echo $PHP_SELF?>">
      First name:<input type="Text" name="first"><br>
      Last name:<input type="Text" name="last"><br>
      Address:<input type="Text" name="address"><br>
      Position:<input type="Text" name="position"><br>
      <input type="Submit" name="submit" value="Enter information">
     </form>
<?php
      } // end if
?>
</body></html>
```

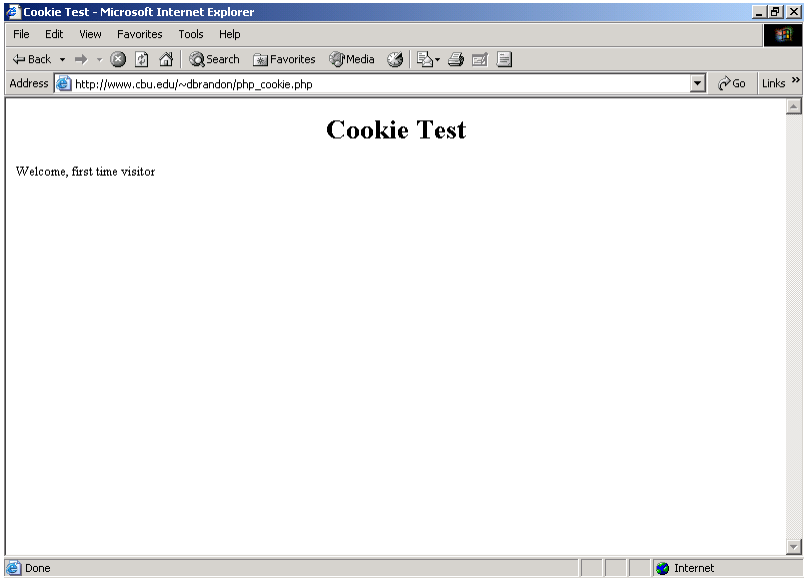
PHP Features

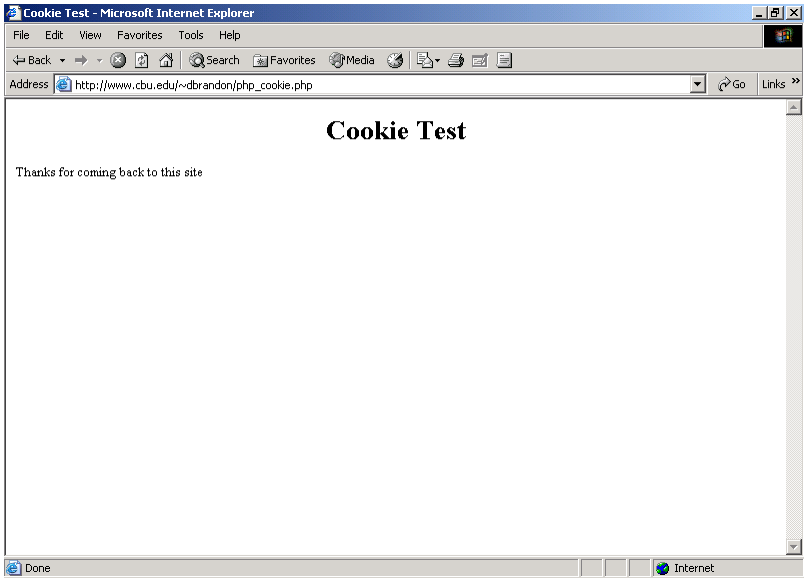
- Also supported in PHP:
  - Regular expressions
  - Cookies
  - Session Management
  - Object Orientation
  - Java interface
  - TCP/IP application interfaces (mail, ftp, etc.)
  - LDAP interface
  - OS Command execution
  - Compilation/execution accelerators
  - Source code encryptors

Cookies via PHP

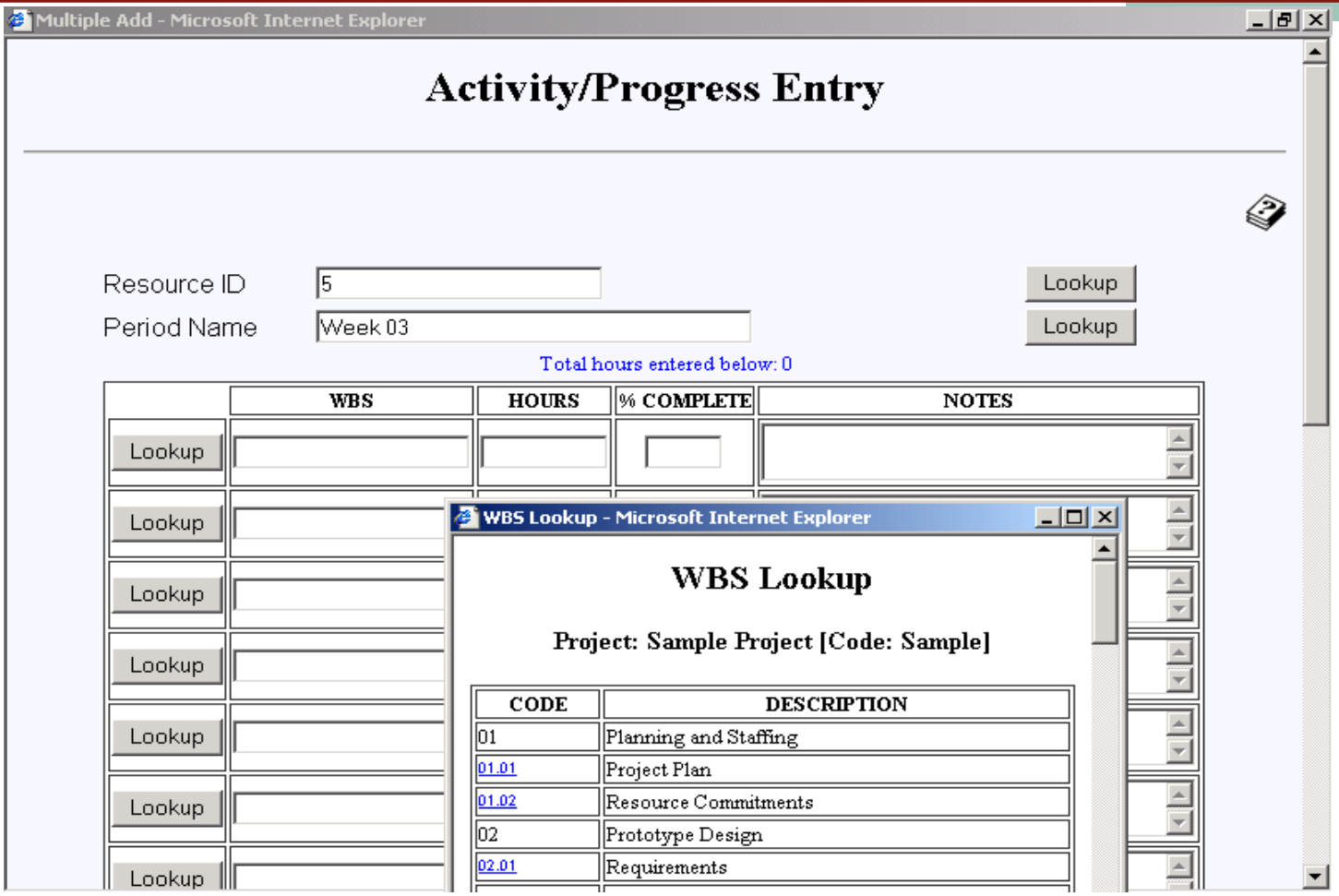
```
<?php
        setcookie ("been_here", "yes", time()+604800);  // one week in duration
?>
<HTML>
<HEAD><TITLE>Cookie Test</TITLE></HEAD>
<BODY>
<H1 align="center">Cookie Test</H1>
<P>
<?php
        $c = $_COOKIE['been_here'];
        if ($c == "yes") {
                echo "Thanks for coming back to this site";
        }
        else {
                echo "Welcome, first time visitor";
        }
?>
</BODY>
</HTML>
```

Example of PHP Cookies





PHP Form with Foreign Key Lookup



PHP Frameworks

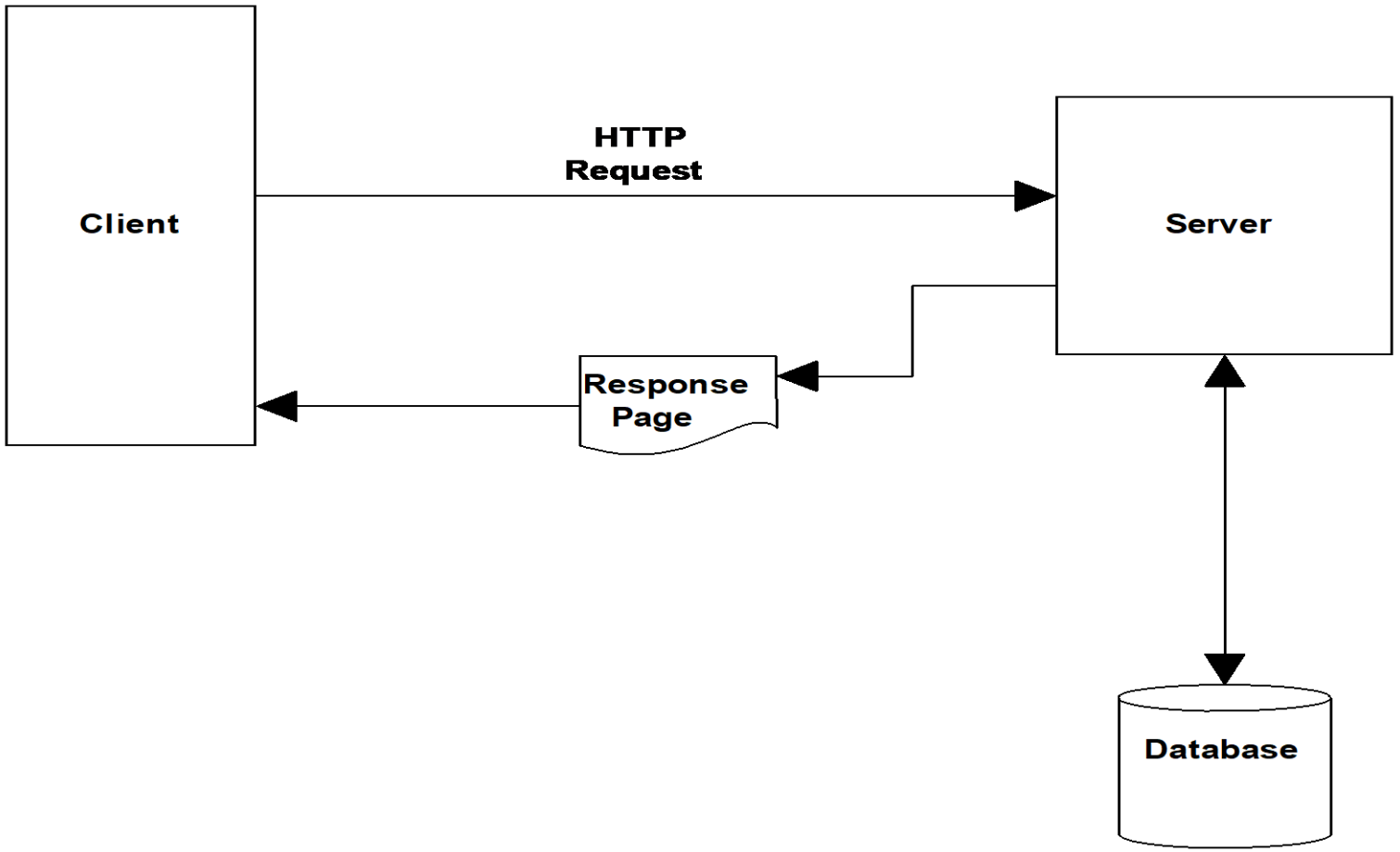
Python Server Programs

- In HTML web page:
  - <form method="post" action="programname.cgi">
- In Python server program:
  - formData = cgi.fieldStorage()
  - firstName = formData.getvalue('firstname')

Asynchronous JavaScript and XML

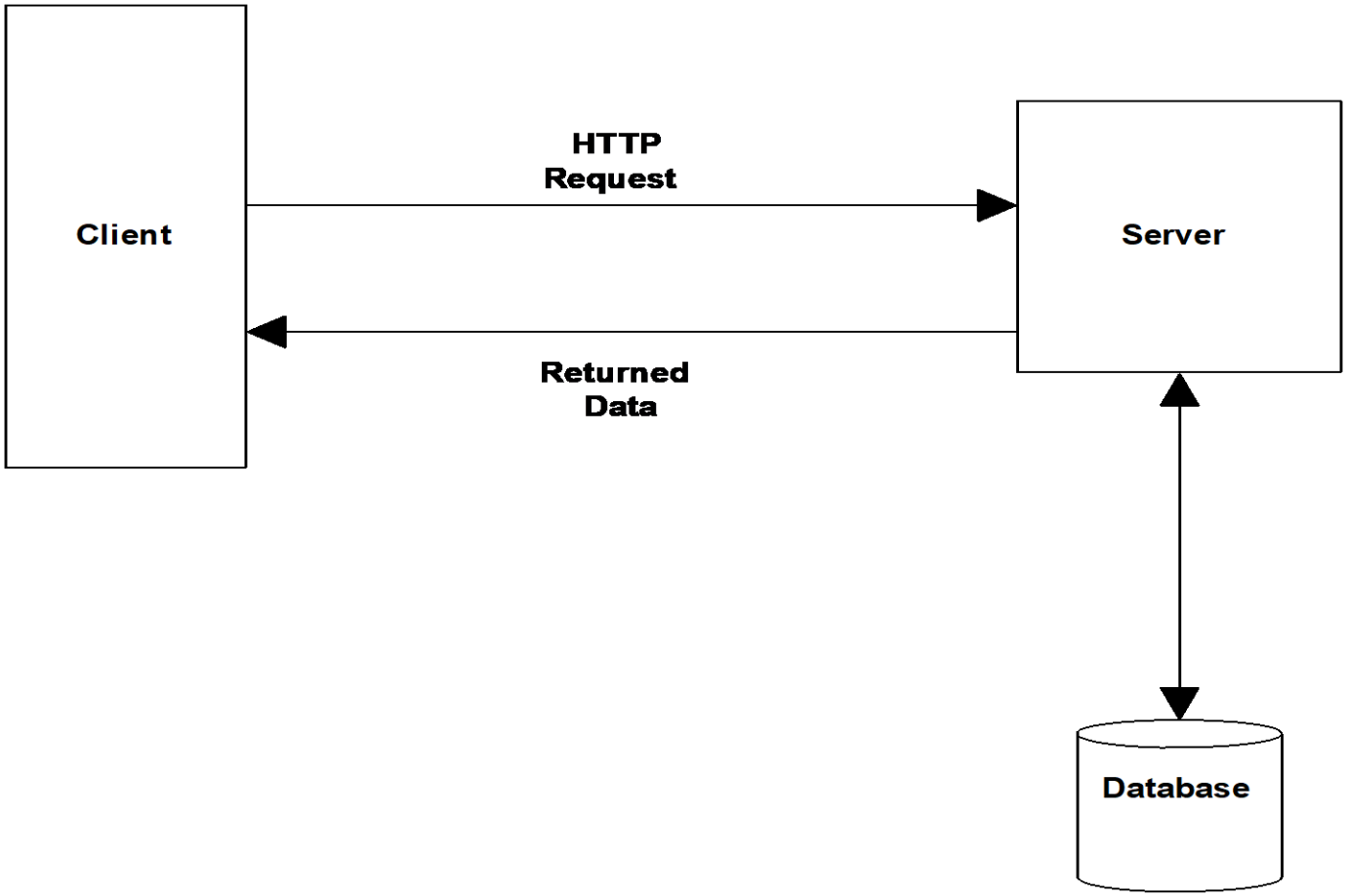
AJAX

Traditional HTTP



AJAX

(client stays focused on same page)



XMLHttpRequest Object

- The XMLHttpRequest object is a JavaScript object that offers a convenient way for webpages to get information from servers without refreshing the HTML page
- The benefit to end users is that they don't have to type as much and they don't have to wait as long
  - For example, having the user's city and state show up in a webpage automatically after the ZIP code has been typed in is a big time saver
- A good way to think of the XMLHttpRequest object is as you would think of the JavaScript Image object
  - As we know, with the Image object you can dynamically specify a new URL for the image source without reloading the page; similarly with the XMLHttpRequest object, you can dynamically specify a URL to get some server data without reloading the page
- AJAX not required for Project 3

JavaScript's XMLHttpRequest Object

```
// create object
xh = new XMLHttpRequest();
// open connection, do http "get" on object at specified URL
xh.open("GET", URL, true);
// if data was successfully transferred, change content of current page
xh.onreadystatechange=
      function () {
        var textElement;
        if (xh.readyState == 4) { // 4 = operation complete
          textElement = document.getElementById(someElementId);
          textElement.innerHTML = xh.responseText;
        }
      }
// close connection
xh.send(null);
```

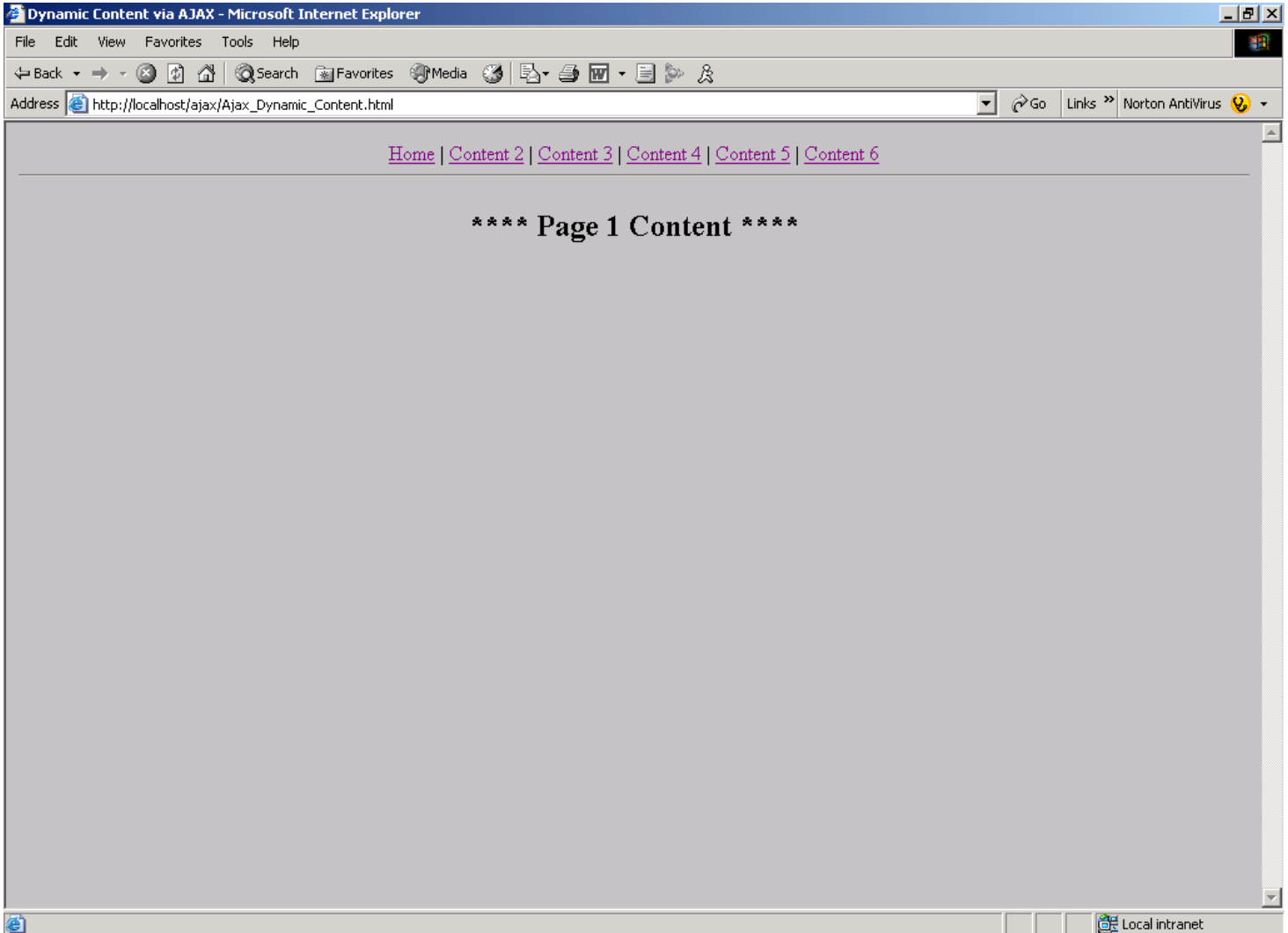
XMLHttpRequest Object Methods

Method	Description
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(method,url,async,uname,pswd)	Specifies the type of request, the URL, if the request should be handled asynchronously or not, and other optional attributes of a request method: the type of request: GET or POST url: the location of the file on the server async: true (asynchronous) or false (synchronous)
send(string)	send(string) Sends the request off to the server. string: Only used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number (e.g. "404" for "Not Found" or "200" for "OK")
statusText	Returns the status-text (e.g. "Not Found" or "OK")

AJAX Dynamic Content



Ajax_Dynamic_Content.html

```
<html>
<head><title>Dynamic Content via AJAX</title>
<script type="text/JavaScript" src="Ajax_Dynamic_Content.js"></script>
</head>
<body onLoad="getPage(1);" bgcolor="silver">
<center>
<div align="center">
<a href="JavaScript:getPage(1);">Home</a> |
<a href="JavaScript:getPage(2);">Content 2</a> |
<a href="JavaScript:getPage(3);">Content 3</a> |
<a href="JavaScript:getPage(4);">Content 4</a> |
<a href="JavaScript:getPage(5);">Content 5</a> |
<a href="JavaScript:getPage(6);">Content 6</a>
</div>
<hr>
<div id="responseText">Loading...</div>
</body>
</html>
```

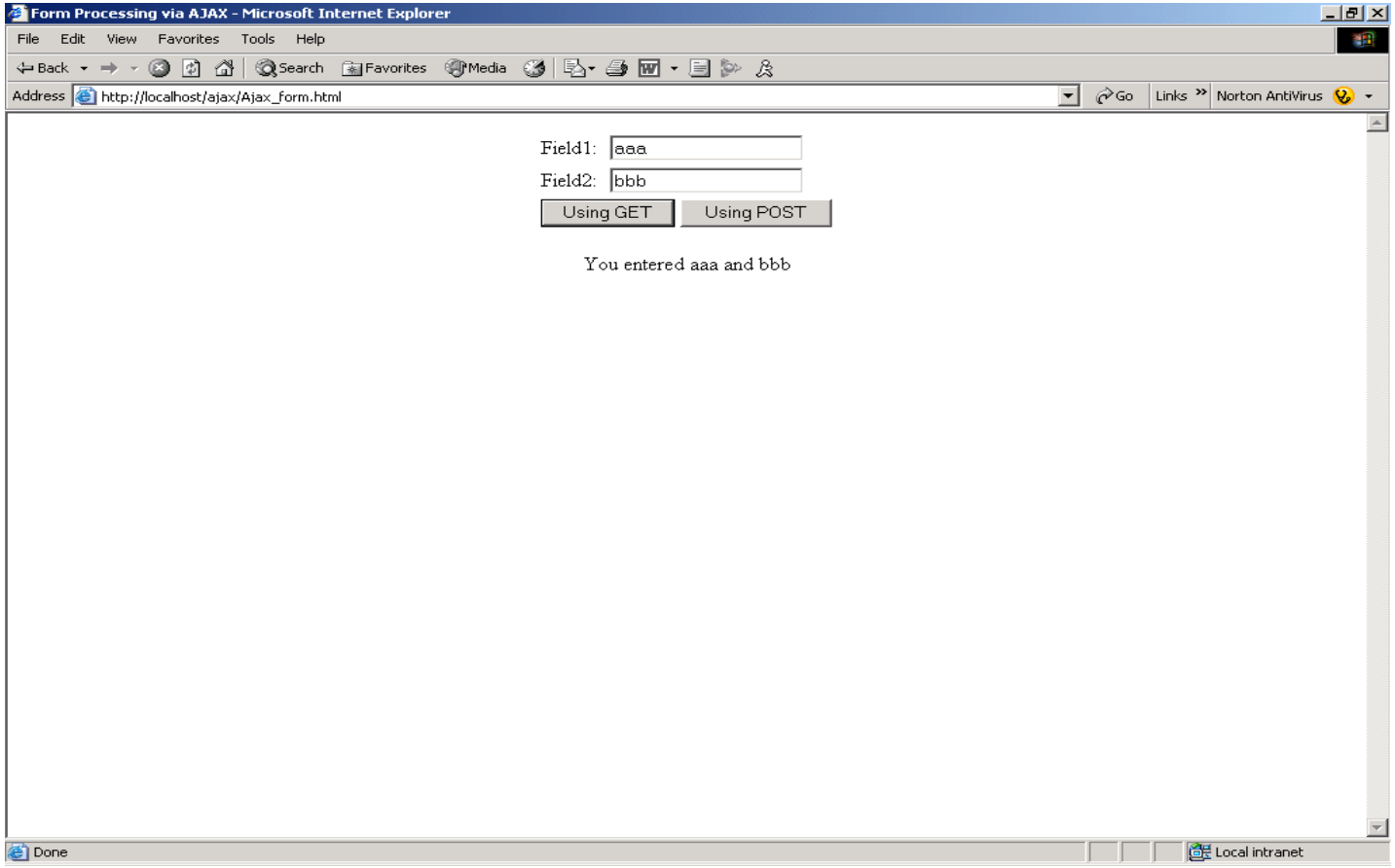
Ajax_Dynamic_Content.js

```
function getPage(pageNo) {
  switch (pageNo) {
    case 1: loadTextFromServer("Page1.html", "responseText");break;
    case 2: loadTextFromServer("Page2.html", "responseText");break;
    case 3: loadTextFromServer("Page3.html", "responseText");break;
    case 4: loadTextFromServer("Page4.html", "responseText");break;
    case 5: loadTextFromServer("Page5.html", "responseText");break;
    case 6: loadTextFromServer("Page6.html", "responseText");break;
    default: loadTextFromServer("Page1.html", "responseText");break;
  }
}
function loadTextFromServer(URL, destinationElementId) {
  var xh;
  try { xh = new ActiveXObject("Msxml2.XMLHTTP");}
  catch (e) {
    try { xh = new ActiveXObject("Microsoft.XMLHTTP"); }
    catch (e) { xh = new XMLHttpRequest(); }
  }
  xh.open("GET", URL, true);
  xh.onreadystatechange=
    function () {
      var responseTextElement;
      if (xh.readyState == 4) { // 4 = operation complete
        responseTextElement = document.getElementById(destinationElementId);
        responseTextElement.innerHTML = xh.responseText;
      }
    }
  xh.send(null);
}
```

Other HTML Files on Server

- Page1.html
  - <H2 align="center">**** Page 1 Content ****</h2>
- Page2.html
  - <H2 align="center">**** Page 2 Content ****</h2>
- Page3.html
  - <H2 align="center">**** Page 3 Content ****</h2>
- Page4.html
  - <H2 align="center">**** Page 4 Content ****</h2>
- Page5.html
  - <H2 align="center">**** Page 5 Content ****</h2>
- Page6.html
  - <H2 align="center">**** Page 6 Content ****</h2>

Ajax Form Echo



Ajax_form.html

```
<html>
<head>
<title>Form Processing via AJAX</title>
<script type="text/JavaScript" src="Ajax_form.js"></script>
</head>
<body>
<center><form action="#">
<table>
<tr><td>Field1:</td><td><input type="text" id="f1"/></tr>
<tr><td>Field2:</td><td><input type="text" id="f2"/></tr>
<tr><td colspan="2" align="center">
<input type="button" value="Using GET" onclick="GETrequest();" />
<input type="button" value="Using POST" onclick="POSTrequest();"/>
</td></tr>
</table>
<p><div id="serverResponse"></div>
</body>
</html>
```

Ajax_form.js

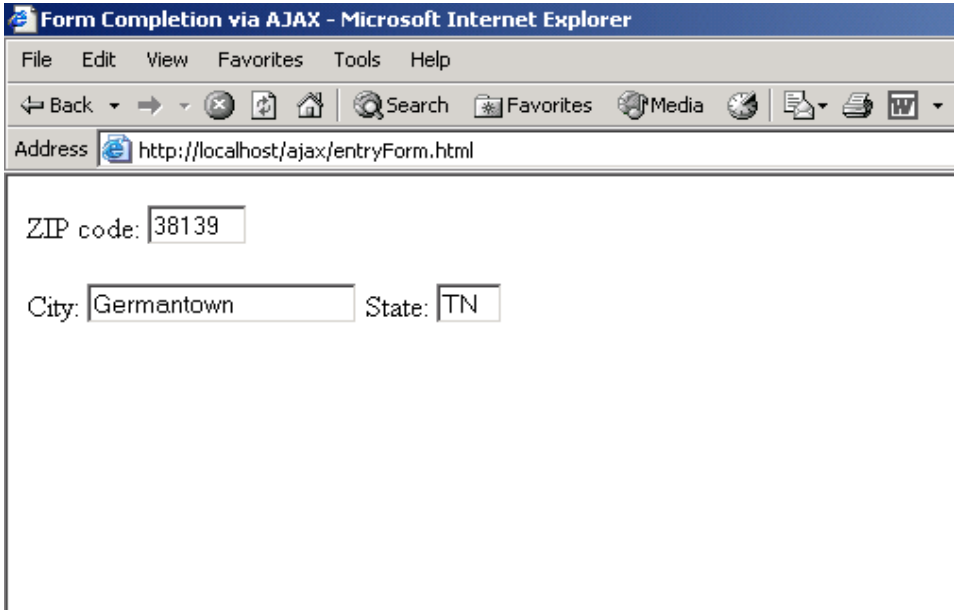
```
function createXMLHttpRequest() {
  try { xmlHttp = new ActiveXObject("Msxml2.XMLHTTP"); }
  catch (e) {
  try {xmlHttp = new ActiveXObject("Microsoft.XMLHTTP"); }
  catch (e) { xmlHttp = new XMLHttpRequest(); }
  }
}
function createQueryString() {
  var f1 = document.getElementById("f1").value;
  var f2 = document.getElementById("f2").value;
  var queryString = "f1=" + escape(f1) + "&f2=" + escape(f2); //url encode form fields
  return queryString;
}
function GETrequest() {
  createXMLHttpRequest();
  var queryString = "returnAjaxMessage.php?";
  queryString = queryString + createQueryString()
  + "&timeStamp=" + new Date().getTime();
  xmlHttp.onreadystatechange = handleStateChange;
  xmlHttp.open("GET", queryString, true);
  xmlHttp.send(null);
}
function POSTrequest() {
  createXMLHttpRequest();
  var url = "returnAjaxMessage.php?timeStamp=" + new Date().getTime();
  var queryString = createQueryString();
  xmlHttp.open("POST", url, true);
  xmlHttp.onreadystatechange = handleStateChange;
  xmlHttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
  xmlHttp.send(queryString);
}
function handleStateChange() {
  if(xmlHttp.readyState == 4) {
    if(xmlHttp.status == 200) {
      var responseDiv = document.getElementById("serverResponse");
      if(responseDiv.hasChildNodes()) {
        responseDiv.removeChild(responseDiv.childNodes[0]);
      }
      var responseText = document.createTextNode(xmlHttp.responseText);
      responseDiv.appendChild(responseText);
    }
  }
}
```

returnAjaxMessage.php

- <?php
  - $first = $_REQUEST['f1'];
  - $second = $_REQUEST['f2'];
  - echo "You entered " . $first . " and " . $second;
- ?>

Form Completion via AJAX

[enter zip code, then look up city and state on server]



Form Completion via AJAX

```
<html>
<head>
<title>Form Completion via AJAX</title>
<script language="javascript"  type="text/javascript">
var url = "getCityState.php?param="; // The server api program
function handleHttpResponse() {
  if (http.readyState == 4) {
    // Split the comma delimited response into an array
    results = http.responseText.split(",");
    document.getElementById('city').value = results[0];
                document.getElementById('state').value = results[1];
  }
}
function updateCityState() {
  var zipValue = document.getElementById("zip").value;
  http.open("GET", url + escape(zipValue), true);
  http.onreadystatechange = handleHttpResponse;
  http.send(null);
}
function getHTTPObject() {
   var xh;
   try { xh = new ActiveXObject("Msxml2.XMLHTTP");}
   catch (e) {
      try { xh = new ActiveXObject("Microsoft.XMLHTTP"); }
      catch (e) { xh = new XMLHttpRequest(); }
   }
   return xh;
}
var http = getHTTPObject(); // Create the HTTP Object
</script>
</head>
<body>
<form action="post">
  <p>
  ZIP code:
  <input type="text" size="5" name="zip" id="zip" onblur="updateCityState();" />
  </p>
  City:
  <input type="text" name="city" id="city" />
  State:
  <input type="text" size="2" name="state" id="state" />
</form>
</body>
</html>
```

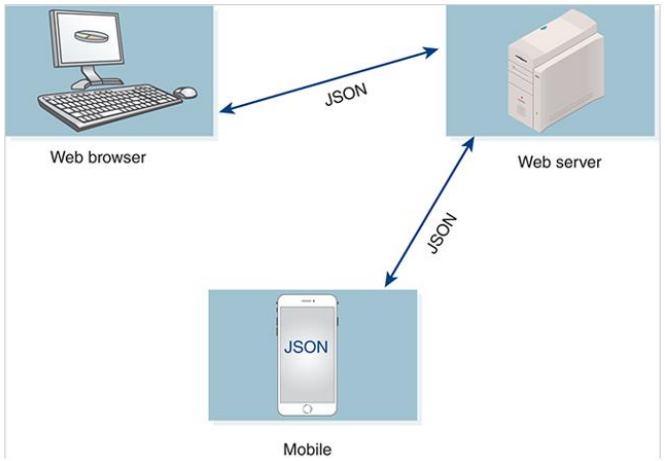
getCityState.php

```
<?php
$zip = $_REQUEST['param'];
// this would normally be a database lookup
if ($zip=='38139') echo "Germantown,TN";
else echo "Memphis,TN";
?>
```

JSON

- JSON (JavaScript Object Notation) was developed as a format for objects between JavaScript programs on a client and server
- JSON for a student object:

```
{"name": "Laurie Jones", "GPA": 3.5, "Grade": 97}
```

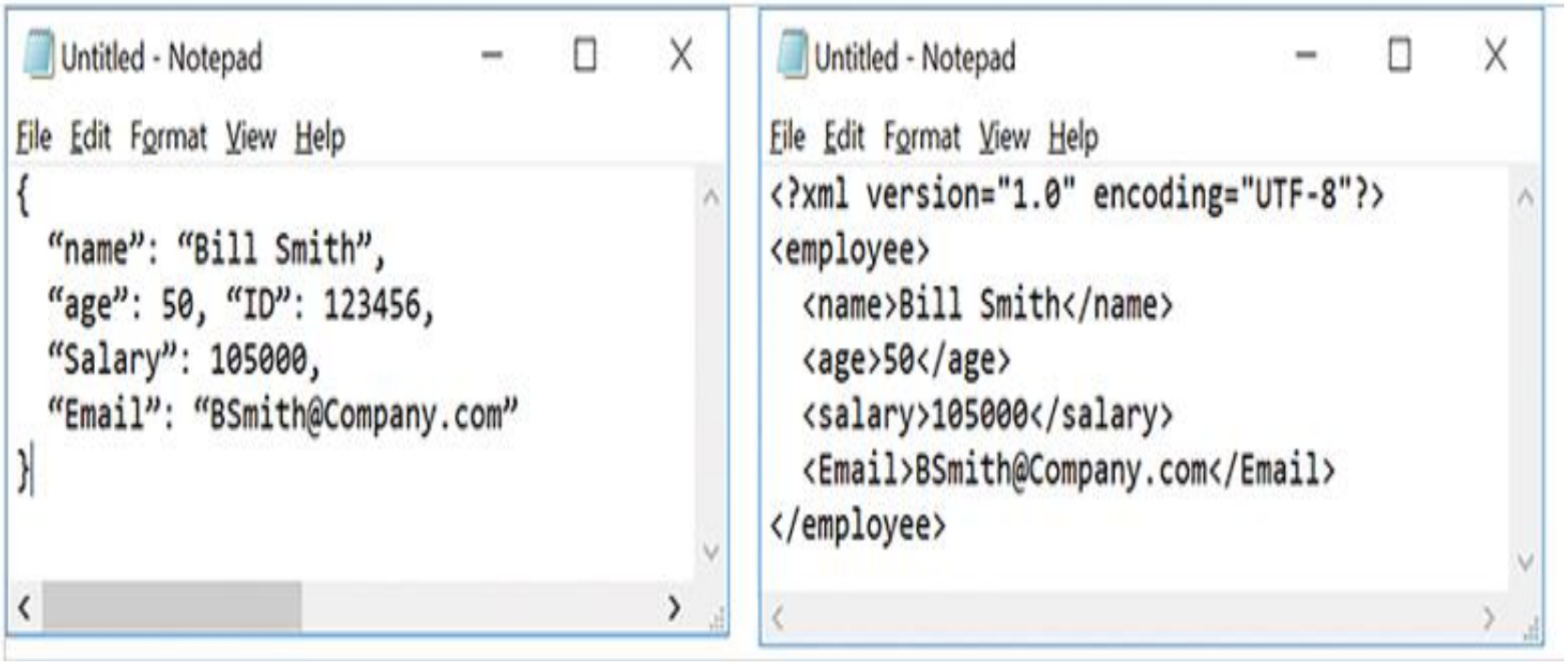


JSON Data Types

String: Contains a value within quotes
Number: An integer or floating-point value
Object: A nested object enclosed in left and right braces { }
Array: A collection of values grouped in left and right brackets [ ]
Boolean: A true or false value
Null: Represented by the keyword null

```
{
   "Title": "Data Mining and Analytics",
   "Chapters": 16,
   "Author": {"Last": "Jamsa", "First": "Kris", "ID": 1983},
  "OtherBooks": true,
   "Universities": [ "ASU", "UNLV", "San Diego State
University"],
    "NextBook": null
  }
```

JSON vs XML


Untitled - Notepad
File Edit Format View Help
{
"name": "Bill Smith",
"age": 50, "ID": 123456,
"Salary": 105000,
"Email": "BSmith@Company.com"
}
Untitled - Notepad
File Edit Format View Help
<?xml version="1.0" encoding="UTF-8"?>
<employee>
<name>Bill Smith</name>
<age>50</age>
<salary>105000</salary>
<Email>BSmith@Company.com</Email>
</employee>

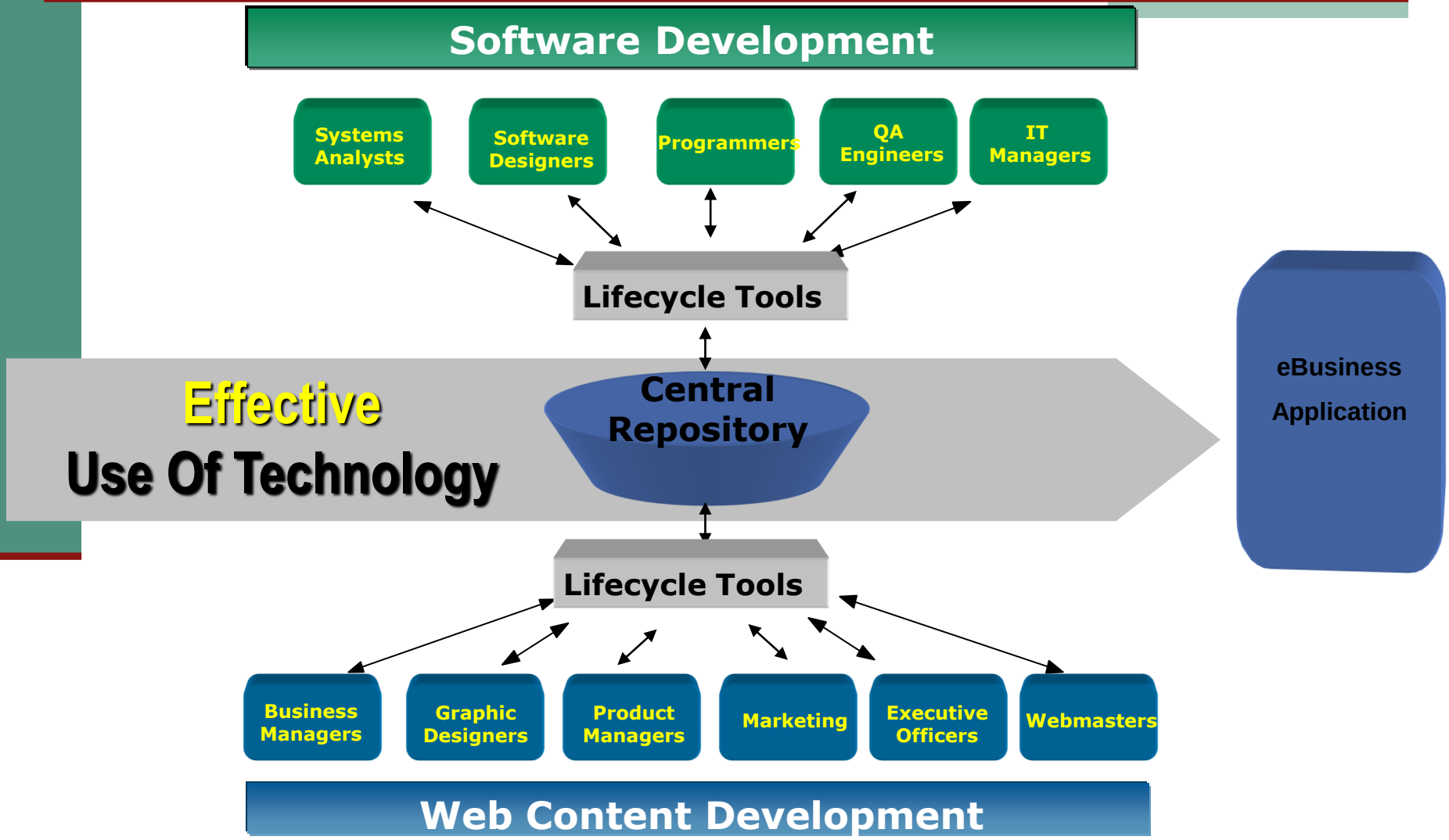
AJAJ

- Asynchronous JavaScript and JSON (or AJAJ) refers to the same dynamic web page methodology as Ajax, but instead of XML, JSON is the data format
- Like AJAX, AJAJ is a web development technique that provides to the ability of a webpage to request new data after it has loaded into the web browser
- Typically it renders new data from the server in response to user actions on that webpage
- For example, what the user types into a search box, client-side code then sends to the server, which immediately responds with a drop-down list of matching database items
- The following JavaScript code is an example of a client using XMLHttpRequest to request data in JSON format from a server. (The server-side programming is omitted; it must be set up to service requests to the url containing a JSON-formatted string.)

```
var my_JSON_object;
var http_request = new XMLHttpRequest();
http_request.open("GET", url, true);
http_request.onreadystatechange = function () {
    var done = 4, ok = 200;
    if (http_request.readyState === done && http_request.status === ok) {
    my_JSON_object = JSON.parse(http_request.responseText);
    }
};
http_request.send(null);
```

Modern Web Development Environment

[Program design/coding as well as Content Management]

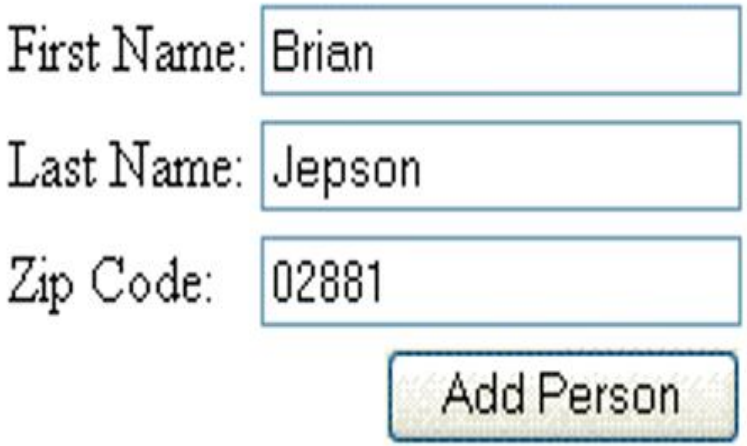


Case Study

Internet Database Interaction Using Modern Server Technology

Case Study Form

(xxx in action will be php, asp, etc.)




```
<HTML>
<HEAD>
  <TITLE>Add a New Person</TITLE>
</HEAD>
<BODY>
  <FORM ACTION="changeme.xxx">
    <TABLE>
      <TR>
        <TD>First Name:</TD>
        <TD><INPUT TYPE="text" NAME="firstname"></TD>
      </TR>
      <TR>
        <TD>Last Name:</TD>
        <TD><INPUT TYPE="text" NAME="lastname"></TD>
      </TR>
      <TR>
        <TD>Zip Code:</TD>
        <TD><INPUT TYPE="text" NAME="zipcode"></TD>
      </TR>
      <TR>
        <TD COLSPAN="2" ALIGN="RIGHT">
        <INPUT TYPE="submit" VALUE="Add Person">
        </TD>
      </TR>
    </TABLE>
  </FORM>
</BODY>
</HTML>
```

SQL To Be Executed on Server

- Add row to PEOPLE table:
  - INSERT INTO PEOPLE VALUES (‘first', ‘last', ‘zip')
- Retrieve all data for all rows in PEOPLE table:
  - SELECT * FROM PEOPLE

HTML Response page

I added Brian Jepson to the database. Here is the complete list of people:

Name	Zip Code
Brian Jepson	02881

SavePerson.cgi (Perl)

```
#!/usr/bin/perl -w

use strict; # restrict unsafe constructs
use DBI;    # import the DBI module

use CGI qw(:standard *table); # import the CGI functions

# Move form data into local variables.
#
my $firstname = param("firstname");
my $lastname  = param("lastname");
my $zipcode   = param("zipcode");

# Print the standard text/html header before you do
# anything that might raise an error.
#
print header();

# Connect to the local MySQL server.
#
my $user   = "bjepson";
my $passwd = "password";
my $dsn    = "DBI:mysql:database=bjepson";
my $dbh = DBI->connect($dsn, $user, $passwd);
if (!$dbh) { die "Error connecting: $DBI::errstr.\n" };

# Insert the new person.
#
my $sql = "INSERT INTO PEOPLE VALUES
           ('$firstname', '$lastname', '$zipcode')";

if (!$dbh->do($sql)) {
  print "Warning: SQL statement:
        <PRE>$sql</PRE>
        failed!!!<BR>";
  exit;
}
```

```
# Display the HTML document.
#
print start_html("New Person: $firstname $lastname");
print "I added $firstname $lastname to the database.
       Here is the complete list of people:";

# Retrieve the people with a SELECT statement.
#
my $stmt = $dbh->prepare("SELECT * FROM PEOPLE");

if (!$stmt->execute()) {
  print "<BR>Warning: SELECT statement failed!!!<BR>";
  exit;
}
# Fetch all the rows and display them in a table.
#
print start_table({-border => 1});
while (my @row = $stmt->fetchrow_array) {

  my ($first, $last, $zip) = @row;
  print "<TR>
         <TD>$first $last</TD><TD>$zip</TD>
        </TR>";

}
$stmt->finish;
print end_table;

print end_html;
$dbh->disconnect;
```

SavePerson.asp (Active Server Pages)

```
<%@ page language="C#" contentType="text/html" %>
<%@ import namespace= "System.Data" %>
<%@ import namespace= "System.Data.OleDb" %>

<%

  /*
   * Move the form data into local variables.
   */
  string firstname = Request["firstname"];
  string lastname  = Request["lastname"];
  string zipcode   = Request["zipcode"];

  /*
   * Connect to the database server.
   */
  string conn_str =
    "provider=SQLOLEDB;server=(local)\\NetSDK;uid=sa;";
  OleDbConnection conn = new OleDbConnection(conn_str);
  conn.Open();

  /*
   * Create an SQL statement to insert the new person.
   */
  string sql_template =
    "INSERT INTO PEOPLE VALUES ('{0}', '{1}', '{2}')";
  string sql = String.Format(sql_template,
                             firstname,
                             lastname,
                             zipcode);
  /*
   * Send the SQL to the database.
   */
  OleDbCommand cmd = new OleDbCommand(sql, conn);
  cmd.ExecuteNonQuery();

%>
```

```
<HTML>
<HEAD>
  <TITLE>
    New Person: <%= firstname + " " + lastname %>
  </TITLE>
</HEAD>

<BODY>
  I added <%= firstname + " " + lastname %> to the
  database.  Here is the complete list of people:

  <%

    /* Fetch all the customers and bind to the
     * HTML table.
     */
    sql = "SELECT firstname + ' ' + lastname " +
          "AS fullname, zipcode FROM PEOPLE";
    OleDbDataAdapter da =
      new OleDbDataAdapter(sql, conn);

    DataSet ds = new DataSet();
    da.Fill(ds, "PEOPLE");
    Table1.DataSource =
      ds.Tables["PEOPLE"].DefaultView;
    Table1.DataBind();

  %>

  <!-- Insert an ASP.NET DataGrid component. -->
  <asp:DataGrid id="Table1"
                AutoGenerateColumns="false"
                runat="server">
    <Columns>
      <asp:BoundColumn
          HeaderText="Name"
          DataField="fullname"/>
      <asp:BoundColumn
          HeaderText="Zip Code"
          DataField="zipcode"/>
    </Columns>

  </asp:DataGrid>

</BODY>
</HTML>
```

SavePerson.JSP (Java Server Pages)

```
<%@ page language="java"
contentType="text/html" %>
<%@ page import = "java.sql.*" %>
<%
/*
* Move the form data into local
variables.
*/
String firstname =
request.getParameter("firstname");
String lastname =
request.getParameter("lastname");
String zipcode =
request.getParameter("zipcode");
/*
* Load the JDBC driver for
PostgreSQL.
*/
Class.forName("org.postgresql.Driver");
/*
* Connect to the database server.
*/
String connection_string =
"jdbc:postgresql:testdb";
String username = "bjepson";
String password = "password";
Connection conn = DriverManager.
getConnection(connection_string,
username, password);
Statement stmt = conn.
createStatement();
/*
* Create an SQL statement to insert
the new person.
*/
String sql = "INSERT INTO PEOPLE " +
"VALUES('" + firstname + "', " +
"'" + lastname + "', " +
"'" + zipcode + "') ";
/*
* Send the SQL to the database.
*/
stmt.executeUpdate(sql);
%>
```

```
<HTML>
<HEAD>
<TITLE>
New Person: <%= firstname + " " +
lastname %>
</TITLE>
</HEAD>
<BODY>
I added <%= firstname + " " +
lastname %> to the database. Here is
the complete list of people:
<TABLE BORDER>
<TH>Name</TH><TH>Zip Code</TH>
<%
/*
* Fetch each row in the table,
and display it as an HTML table row.
*/
ResultSet rs =
stmt.executeQuery("SELECT *
FROM PEOPLE");
while(rs.next()) {
firstname = rs.getString(1);
lastname = rs.getString(2);
zipcode = rs.getString(3);
%>
<TR>
<TD><%= firstname + " " +
lastname %></TD>
<TD><%= zipcode %></TD>
</TR>
<% } /* end of the while loop */ %>
</TABLE>
</BODY>
</HTML>
```

SavePerson.cfm (Cold Fusion)

```
<!-- Import the form data. -->
<cfset firstname=form.firstname>
<cfset lastname=form.lastname>
<cfset zipcode=form.zipcode>

<!-- Insert the new person. -->
<cfquery name="MyUpdate" dataSource="Local">
  INSERT INTO PEOPLE VALUES (
    <cfqueryparam value="#firstname#"  CFSQLType="CF_SQL_CHAR">,
    <cfqueryparam value="#lastname#"   CFSQLType="CF_SQL_CHAR">,
    <cfqueryparam value="#zipcode#"    CFSQLType="CF_SQL_CHAR">  )
</cfquery>

<HTML>
<HEAD>
  <TITLE>
  New Person:  <cfoutput>#firstname# #lastname#</cfoutput>
  </TITLE>
</HEAD>
```

Cfm (con't)

- <BODY>
- I added <cfoutput>#firstname# #lastname#</cfoutput>
- to the database.
- Here is the complete list of people:

- <!-- Retrieve all the people. -->
- <cfquery name="MyQuery" dataSource="Local">
- SELECT * FROM PEOPLE
- </cfquery>

- <!-- Create a ColdFusion HTML table and associate it with MyQuery -->
- <cftable border HTMLTable query="MyQuery">
- <cfcol header="Name" text="#firstname# #lastname#">
- <cfcol header="Zip" text="#zipcode#">
- </cftable>

- </BODY>
- </HTML>

SavePerson.PHP

```
<?php
/*
* Connect to a MySQL database server.
*/
$db = mysql_connect("localhost", "bjepson', "password");
mysql_select_db("bjepson", $db);
/*
* Create an SQL statement to insert the new person.
*/
$sql = "INSERT INTO PEOPLE VALUES
("$firstname", "$lastname", "$zipcode")";
/*
* Send the SQL to the database.
*/
mysql_query($sql, $db);
?>
```

```
<HTML>
<HEAD>
<TITLE>
New Person: <?php echo "$firstname
$lastname"; ?>
</TITLE>
</HEAD>
<BODY>
I added <?php echo "$firstname
$lastname"; ?> to the
database. Here is the complete list of
people:
<TABLE BORDER>
<TH>Name</TH><TH>Zip Code</TH>
<?php
/*
* Fetch each row in the table, and
display it as an HTML table row.
*/
$rs = mysql_query("SELECT * FROM
PEOPLE", $db);
while($row = mysql_fetch_array($rs))
{
$first = $row["firstname"];
$last = $row["lastname"];
$zip = $row["zipcode"];
echo "<TR>
<TD>$first
$last</TD><TD>$zip</TD>
</TR>";
}
?>
</TABLE>
</BODY>
</HTML>
```

References

- Modern PHP: New Features and Good Practices by Josh Lockhart
- PHP & MySQL: Server-side Web Development by Jon Duckett
- Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 (Learning PHP, MYSQL, Javascript, CSS & HTML5) by Robin Nixon

Homework

- Develop an HTML page with a form that has a text box for temperature and a radio button for C or F conversion
- Validate form data, and use echo site to validate data transmission
- Hybrid Appendices on:
  - Web Page Cyber Security
  - Java (JSP and Servlets)
  - Cold Fusion
  - Active X Server Pages (ASP)

Web Page Cyber Security Issues

SQL Injection

- SQL injection is a form of cyber security attack where one attempts to pass SQL commands to a server program, typically via HTML forms
- For example we may have a form to update a customer's telephone number which has a PHP program as its "action" and the form variable of pnum for the phone number and cnum for the ID of the customer being updated:
  - $SQL = "UPDATE CUSTOMER SET PHONE = '$pnum' WHERE CID = '$cnum'";
  - $result = mysql_query($SQL);
- If the value of pnum is not carefully checked (on client and/or server) then, someone could enter
  - 9013214567; DELETE FROM CUSTOMER;
- Into the pnum field, and delete all the customers

Cross Site Scripting

- **Cross-site scripting** (**XSS**) is a type of computer security vulnerability typically found in web applications
- XSS enables attackers to inject client-side scripts into web pages viewed by other users
- A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same-origin policy (session control)
  - A web browser permits scripts contained in a first web page to access data in a second web page, but only if both web pages have the same *origin (*a combination of URI scheme, host name, and port number)
- Cross-site scripting carried out on websites accounts for a majority of all web page security vulnerabilities
  - XSS effects vary in range from petty nuisance to significant security risk, depending on the sensitivity of the data handled by the vulnerable site and the nature of any security mitigation implemented by the site's owner

Allowing Info To Be Added By Users

In short, if you are allowing any information to be added by users to your site, then you need to make sure you are protecting yourself. Some of the user interactions that pose a risk include:

- Allowing for comments
- Allowing image, article, or file submissions
- Using Form submissions
- Allowing data to be imported from other sites — such as allowing YouTube videos or Facebook posts to be pulled into your site.
- User registration and logins
- Accepting and using added parameters on the web address

If you are doing any of these things, then you should make sure your code is secure. The following are a few tips to make your site more secure.

Client and Server Validation

It is easy to know that you should validate what is entered when the user is filling out a form on your site and performing validation to check what is being entered is, generally considered, common sense. If a person is entering a phone number, you can check to see that it is a phone number using JavaScript on the page. Validating on entry on the client side is great, and you should be doing that!

If that data is also being stored on your web site's server in a database or otherwise, then you should also validate the data on the server side. You should always verify that what you are receiving on the server is what you expect and that it hasn't be changed or that someone isn't sending something directly to the server from outside of your web page. Validate again.

Get Rid Of Dangerous Characters

There are certain characters that are more dangerous than others. For example, the letters in the alphabet (A to Z) are generally safe. Characters such as &, <, and quotes; however, are a bit more shady. These characters mean something in HTML, so you need to make sure they don't get treated like HTML. The easiest way to do this is to encode the characters when they are used in text submitted by users. Some of the characters you should watch out for are:

- Don't use: <

Use; <

- Don't use: >

Use: >

- Don't use: '

Use: '

- Don't use: "

Use: "

- Don't use: /

Use: &#x2F

- Don't use: &

Use: &

- Don't use: -

Use: `

Check URL's Before Trusting Them

If you are allowing links to be added by users, then you should make sure they do not contain anything that could cause problems. For example, if you store bookmarks, or if you have links to profile pages that might be based on a user name entered, then you might be opening a vulnerability. One way to provide protection is to make sure that any URLs that have user input are treated with `encodeURIComponent()`. This will replace the dangerous characters with escape values.

```
var uri = "http://www.htmlgoodies.com/mytest?alert('stop it!');"
var res  encodeURIComponent(uri);
```

You can then later chose to use `decodeURIComponent()()` if you want to get back to what the user provided.

```
uri =  decodeURIComponent(res);
```

Treat User Submitted Data As Text, Not Markup

It might sound goofy to say treat text as text, but it isn't! When you put information on your webpage, you end up using HTML. There are commands in JavaScript for displaying HTML, and then there are commands for displaying just text.

When displaying text that was created or saved by a user, make sure it is always displayed as text. You can do this by using text commands and by avoiding HTML display commands. In your JavaScript, you should avoid displaying any user data using `innerHTML()`. Instead you should be using `textContent()` or `innerText()`.

If you find that you need to use `innerHTML()` to display user submitted text, then it is critical that you verify that all the text uses escaped with no markup within it.

Be Careful Where You Put A User's Data

Stating the obvious, don't put data from users you don't know or trust into dangerous areas of your site. For example, don't accept user data and place it within a script block, within your CSS, within an attribute on an HTML tag, or even within an HTML comment.

As an example, it might seem harmless to put text in a comment, after all, it is just a comment. Consider the following text, and think about what it would do in a comment if it isn't encoded correctly:

```
usertext = "this is text that could be add within a comment to say "--> Hi Mom! <!--" when you might not have intended it to! "
```

Clearly this text could have bad consequences on your site if it were placed within a comment tags.

```
<!-- this is text that could be add within a comment to say "--!> Hi Mom! <!-" when you might not have intended it to!--!>
```

With this text, you might find that a message saying, "Hi Mom!" displayed on your page!

Don’t Accept Anything From Unknown Users

If you don't have a way to validate or know who is entering content onto your site, then you should think very seriously about whether you should allow commenting at all. This is why many commenting systems require registration before you are permitted to post. If you can't trust a user, then don't let them add content to *your* site!

Java (JSP & Servlets) and Cold Fusion

Cold Fusion

- Cold Fusion is an API product from Macromedia
- Cold Fusion supports both “tags” (like ASP or JSP) and/or scripting
  - Unlike PHP, database access is only via tags not script
- Operating System independent
- Built on Java (Java Servlets and EJB’s)
- Can write custom tag processes in Java

CFML (Cold Fusion Markup Language)

- <CFQUERY NAME="ProductInfo"
- DATASOURCE="SP">
- SELECT PName, Size, Price FROM P;
- </CFQUERY>
- …
- <CFOUTPUT QUERY="ProductInfo">
- #PName#, #Size#, #Price#
- </CFOUTPUT>

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
        <TITLE>...Some Title...</TITLE>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <!--- Get user's details from the database --->
        <cfquery name="GetUser" datasource="#master.ds#">
                SELECT      *
                FROM        USERS
                WHERE       USER_NAME   = '#Form.username#'
                AND         PASSWORD    = '#Form.Password#'
                AND   EXP_DATE    > #CreateODBCDateTime(Now())#
        </cfquery>
</HEAD>
<BODY bgcolor="EEEEEE" text="466188">
        <H1 align="center">...Page Title...</H1><BR>
        <!--- Check if user is authorized --->
        <cfif GetUser.RecordCount gt 0>
                <!--- update profile information --->
                <cfset TotLogins = getUser.TOTAL_LOGINS>
                <!--- Store the new info in variables --->
                <cfset TotLogins = TotLogins + 1>
                <cfquery name="UpdateUser" Datasource="#master.ds#" >
                        UPDATE      USERS
                        SET          TOTAL_LOGINS = #TotLogins#,
                                     LAST_LOGIN    = #CreateODBCDateTime(Now())#
                        WHERE       USER_NAME = '#Form.username#'
                </cfquery>
                <cflocation url="...url of next page...">
        <cfelse>
                <!--- User does not exist --->
                <!--- Pass the variables back in the URL - message and UserName --->
                <cfset loginpage = "bk_login.cfm?Message=" & URLEncodedFormat("Invalid User Name/Password or has
expired")>
                <cfset loginpage = loginpage & "&UserName=" & URLEncodedFormat(#Form.UserName#)>
                <cflocation url="#loginpage#">
        </cfif>
</BODY>
</HTML>
```

Example Cold Fusion File for User Login Validation

Java Servlets

- Servlets represent an efficient way of doing CGI type processing on the server
- Like ASP, a separate process is not started for each server HTTP request as it is with CGI
- However, Java Servlets are platform independent
- Servlets can maintain “state” between a browser client and a server (via Session objects)
- Servlets can directly access DBMS via native interface or JDBC
- Java language and technology covered in the Java courses

Form Posting to Servlet

```
<HTML>
   <HEAD>
      <TITLE>Servlet HTTP Post Example</TITLE>
   </HEAD>
   <BODY>
      <FORM METHOD="POST" ACTION=
         "http://sheba.cbu.edu:8080/servlet/HTTPPostServlet">
         What is your favorite pet?<BR><BR>
         <INPUT TYPE=radio NAME=animal VALUE=dog>Dog<BR>
         <INPUT TYPE=radio NAME=animal VALUE=cat>Cat<BR>
         <INPUT TYPE=radio NAME=animal VALUE=bird>Bird<BR>
         <INPUT TYPE=radio NAME=animal VALUE=snake>Snake<BR>
         <INPUT TYPE=radio NAME=animal VALUE=none CHECKED>None
         <BR><BR><INPUT TYPE=submit VALUE="Submit">
         <INPUT TYPE=reset>
      </FORM>
   </BODY>
</HTML>
```

Servlet Java Code

```
// A simple survey servlet
import javax.servlet.*;
import javax.servlet.http.*;
import java.text.*;
import java.io.*;
import java.util.*;

public class HTTPPostServlet extends HttpServlet {
   private String animalNames[] = { "dog", "cat", "bird", "snake", "none" };
   public void doPost( HttpServletRequest request,
                       HttpServletResponse response )
      throws ServletException, IOException
   {
      int animals[] = null, total = 0;
      File f = new File( "survey.txt" );
      if ( f.exists() ) {
         // Determine # of survey responses so far
         try {
            ObjectInputStream input = new ObjectInputStream(new FileInputStream( f ) );
            animals = (int []) input.readObject();
            input.close();   // close stream
            for ( int i = 0; i < animals.length; ++i )
               total += animals[ i ];
         }
         catch( ClassNotFoundException cnfe ) {
            cnfe.printStackTrace();
         }
      }
      else
         animals = new int[ 5 ];
      // read current survey response
      String value =  request.getParameter( "animal" );
      ++total;   // update total of all responses
      // determine which was selected and update its total
      for ( int i = 0; i < animalNames.length; ++i )
         if ( value.equals( animalNames[ i ] ) )
            ++animals[ i ];
```

```
   // write updated totals out to disk
         ObjectOutputStream output = new ObjectOutputStream(
            new FileOutputStream( f ) );
         output.writeObject( animals );
         output.flush();
         output.close();
         // Calculate percentages
         double percentages[] = new double[ animals.length ];
         for ( int i = 0; i < percentages.length; ++i )
            percentages[ i ] = 100.0 * animals[ i ] / total;
         // send a thank you message to client
         response.setContentType( "text/html" ); // content type
         PrintWriter responseOutput = response.getWriter();
         StringBuffer buf = new StringBuffer();
         buf.append( "<html>\n" );
         buf.append( "<title>Thank you!</title>\n" );
         buf.append( "Thank you for participating.\n" );
         buf.append( "<BR>Results:\n<PRE>" );
         DecimalFormat twoDigits = new DecimalFormat( "#0.00" );
         for ( int i = 0; i < percentages.length; ++i ) {
            buf.append( "<BR>" );
            buf.append( animalNames[ i ] );
            buf.append( ": " );
            buf.append( twoDigits.format( percentages[ i ] ) );
            buf.append( "%  responses: " );
            buf.append( animals[ i ] );
            buf.append( "\n" );
         }
         buf.append( "\n<BR><BR>Total responses: " );
         buf.append( total );
         buf.append( "</PRE>\n</html>" );
         responseOutput.println( buf.toString() );
         responseOutput.close();
      }
   }
```

JSP (Java Server Pages)

- JSP offer a capability similar to ASP, but using the Java virtual machine on web servers to provide a platform independent approach
- In direct use of Servlets, the Java Servlet program formats the resulting HTML response page; JSP's eliminate that programming difficulty by being embedded in an HTML page
- JSP is easier to use (like ASP) but less flexible than direct Servlet programming

JSP (con't)

- JSP represent a simplified way to use Java server programs (Servlets)
- Reference:
  - java.sun.com/products/jsp
- The JSP Application Server (free download) is a superset of the Java Servlet Engine and supports both direct Servlets and JSP's
- Most JSP's use JavaBeans (just as ASP uses ActiveX objects); these can come from libraries or you can write your own

JSP (con't)

- A JSP consists of HTML/JavaScript, JSP directives, and Java code
- The JSP server engines compiles and caches this when the page is first referenced (via browser HTML)
- For HTML forms processing, when the browser user hits the "submit" button, the "set" method is called for each form field (corresponding the form field's name=… attribute)

Sample Java Server Page

[will echo back "Hello there: xxx" when submit button is picked (xxx is entry in form field)]

- <jsp:useBean id="mine" scope="page" class="mine.TestBean"/> "mine" is object of class "TestBean" in package mine
- <jsp:setProperty name="mine" property="*"/> init all properties of object
- <HTML><HEAD><TITLE>…</TITLE></HEAD>
- <BODY>
- <FORM METHOD="post">
- <INPUT TYPE="text" name="text" size = 25>
 // name must match bean set
- <INPUT TYPE="submit" VALUE="submit"></FORM>
- <% if mine.getText() != null) { %> java code enclosed in <%…%>
- Hello there:
- <% =mine.getText() %> = is short for out.println(…)
- <% } %> </BODY></HTML>

Sample Java Bean

- Package mine;
- public class TestBean {
  - private String text;
  - public TestBean { text = null; }
  - // need a get/set for each form field with matching names
  - public void setText (String s) {text = s;}
  - public String getText() {return text;}
- }

Java Database Connectivity

- An API that allows Java programs to be written and interact with databases via SQL
- Similar capabilities to ODBC & OLE-DB
- Java programs can be written in a database independent manner
- JDBC can be used from server programs that are invoked from clients (Servlets or JSP)
- Java applets can also use JDBC directly (with appropriate choice of drivers), and thus internet applications can be made fully database interactive

JDBC (con't)

- Provides for a much more responsive and capable client application
  - No need to go back to the information (web) server to validate fields and show error messages (can reposition immediately to field in error)
  - No need to go back to server for "lookup" fields (foreign key fields)
  - Network services can be directly initiated on client side (ie sending e-mail, ftp, paging, etc.)
  - Complex business rules and constraints can be implemented
  - Full power of Java language

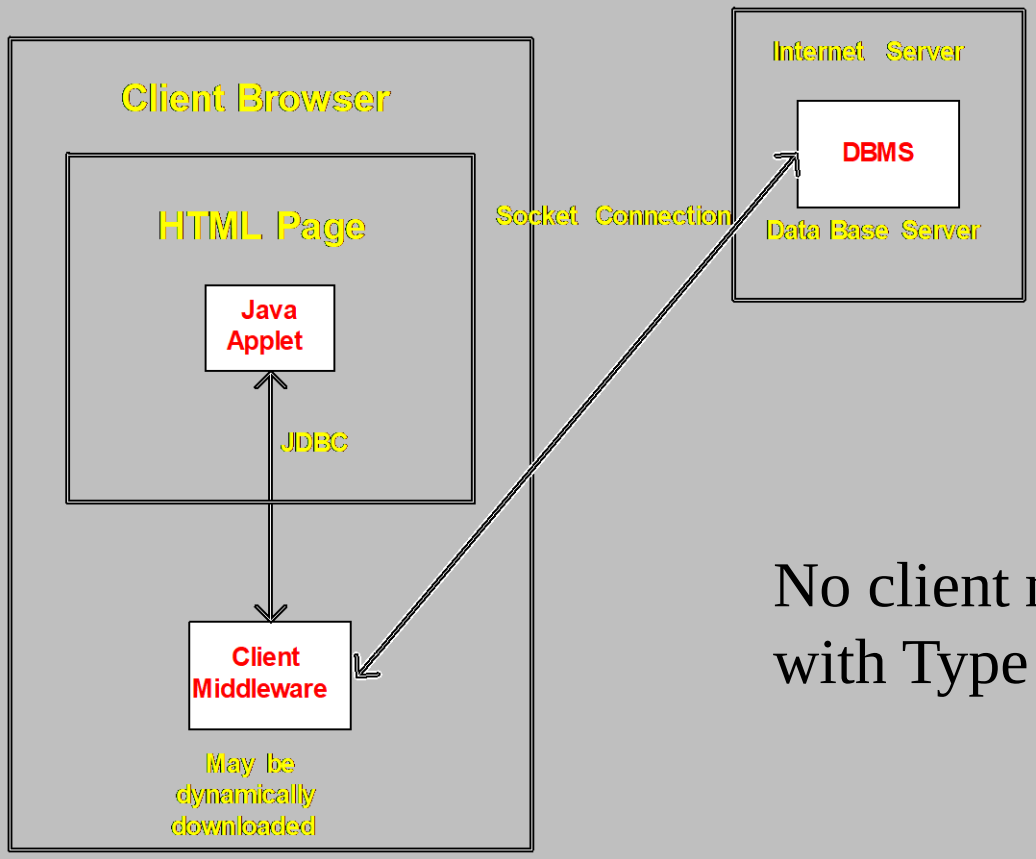
JDBC Driver Types

- Type 1 - JDBC/ODBC Bridge
  - Uses ODBC binary software stored on client (comes with Windows 95/98) and the bridge software which comes with the JDK (built into compiled programs/applets [.class files]). Need to set up ODBC data source on the client.
- Type 2 - Driver is part Java, part native
  - Requires client side binary (.dll) for native portion which is database specific (not for use with applets)

JDBC Driver Types (con't)

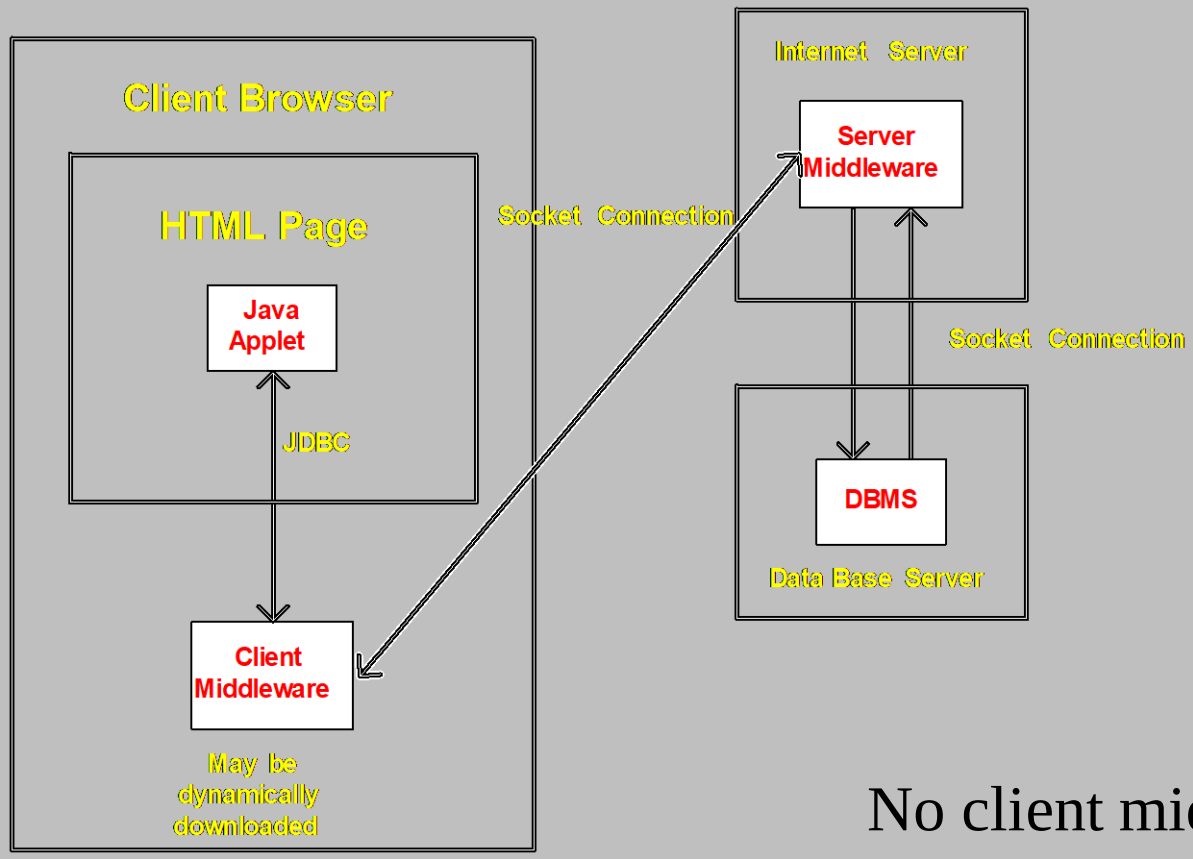
- Type 3 - Pure Java driver with database specific middleware
  - Needs middleware installed onclient
- Type 4 - Pure Java driver
  - Compiled into program/applet [.class files] and can be use with applets (no client side installation required) - Best choice, but also most expensive.

Java JDBC (Two Tier)



No client middleware with Type 4 driver

Java JDBC (three tier)

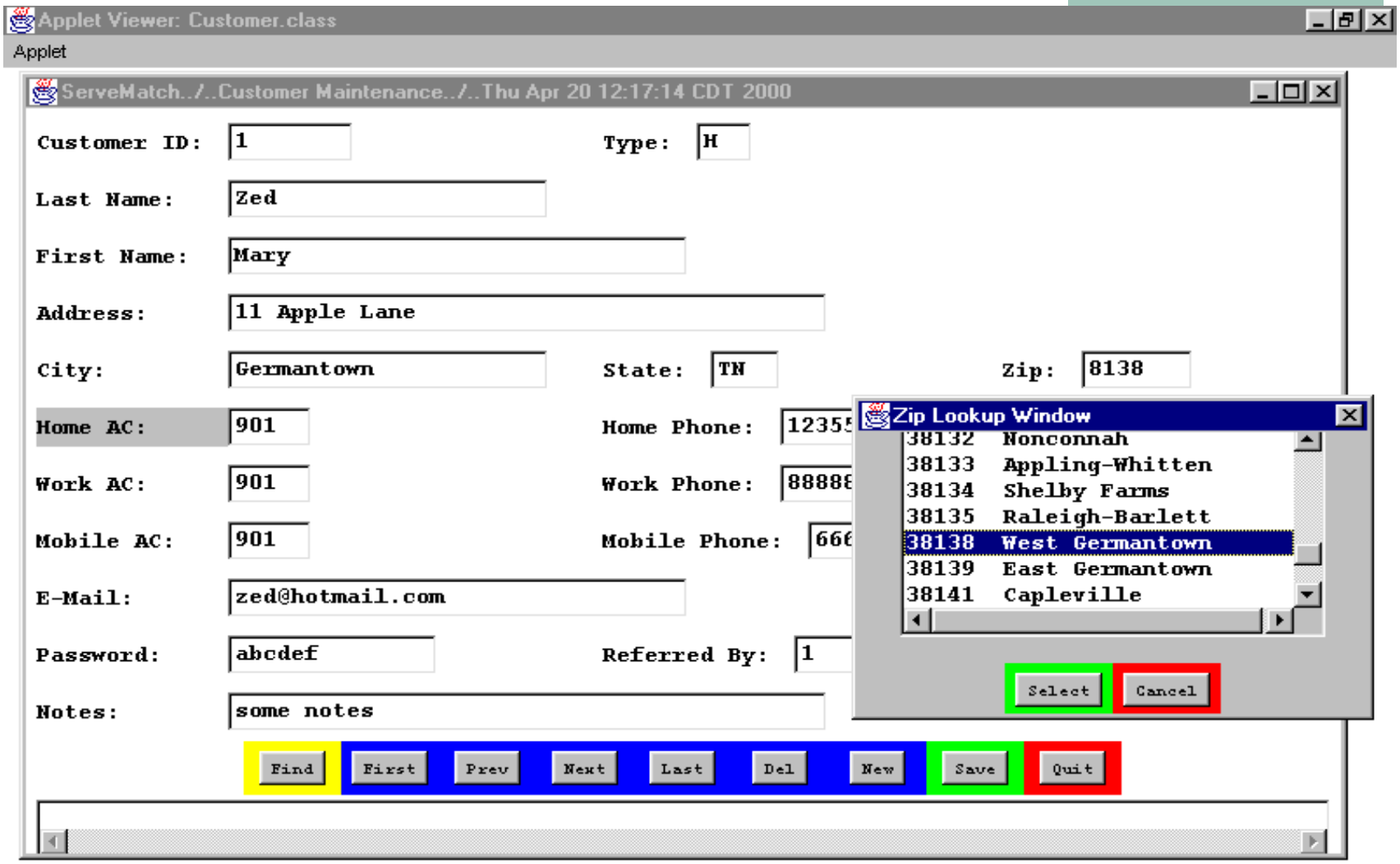


No client middleware with Type 4 driver

Java Database Coding (simplified)

- …
- try{
  - Class.forName(“sun.jdbc.odbc.JdbcOdbcDriver”); // load driver
  - String url = “jdbc:odbc:SP_Access”; //ODBC Data source
  - String un = “”;
  - String pw = “”;
  - Connection con = DriverManager.getConnection(url, un, pw);
  - String query = “SELECT * FROM S;” // SQL query
  - Statement s = con.createStatement();
  - ResultSet rs = s.executeQuery(query); // get result set
- } catch (SQLException ex) {ex.printStackTrace();}
- … // process result set

Java Database Applet – FK Lookup



APPENDIX

Microsoft Active Server Pages (ASP)

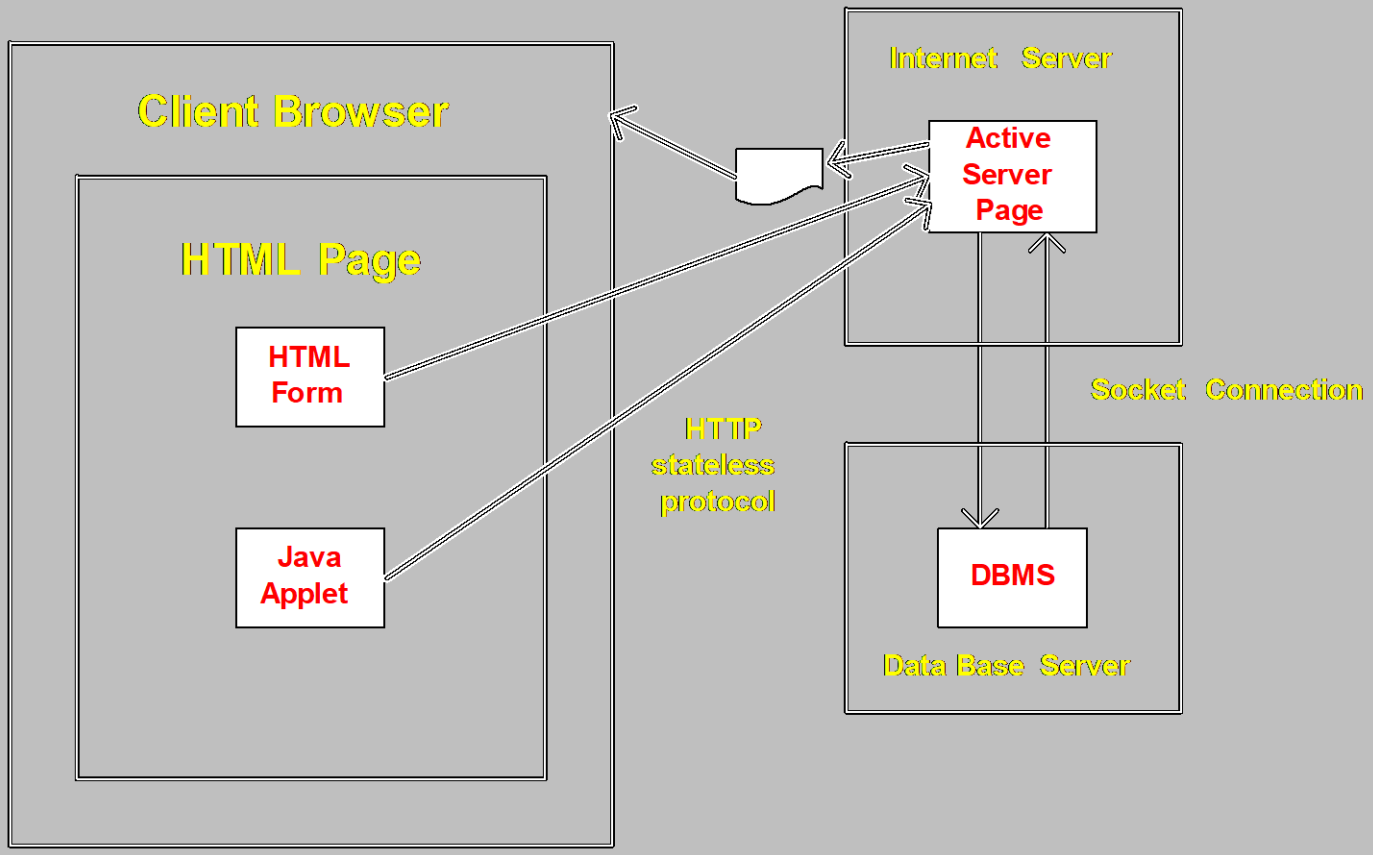
Active Server Pages
(or ActiveX Server Pages)

- ASP is a Microsoft developed technology for sending dynamic Web content - which includes HTML, Dynamic HTML, ActiveX controls, client side scripts, and Java Applets to an internet client
- ASP allows one to mix server side scripting within an HTML like file
- For database applications, ASP is like running an Access form (or datasheet) locally except, it occurs over the internet; it is relatively slow and provides minimal validation; also if Access is used as the database (instead of SQL Server, there is no built-in transaction or concurrency control)

ActiveX (con't)

- A client sends an HTTP request to an information server (an NT or Windows 2000 server, or a Unix server with special third party software) running an ASP process (a server side ActiveX control "asp.dll")
- The asp dll receives the request and directs it the the designated ASP file
- The ASP program executes (which often involves interacting with a database) and returns its result to the client (typically in the form of an HTML document)

Active Server Pages



ASP (con't)

- ASP files use the .asp file extension
- A number of scripting languages can be used in the ASP
- Typically Visual Basic Script (VBScript) is used and is the default (another scripting language can be specified with the @LANGUAGE tag)
- Script (enclosed between <% and %> can be intermixed within HTML code (and vice versa)
- The ASP process reads the script and replaces it with HTML before sending the page to the client

Client Side vs Server Side Scripting

- Server side scripting allows greater flexibility
- Source code of server scripts is not visible on client side (directly via the browser)
- Server scripting is implemented on a particular server (ie Windows NT/200*), and cross platform compatibility is not an issue (VBScript is typically used for Windows NT/2000)
- A typical example is for a client to request to see flights from one place to another; the server script would formulate and send an SQL request to the database (on the same or another server); when it got the results it would format them in an HTML table and send it back to the client

VB Syntax

- VB Script is a derivative of Basic (instead of a derivative of C, like JavaScript); it has weaker syntax and less consistency than C derived scripting
- It is case insensitive
- Operators are somewhat different: two division operators [/ and \], exponentiation operator [^], not equal [<>], logical operators [AND, OR, NOT, XOR]
- Data types are somewhat different, as is function calling syntax
- Control structures are also somewhat different: if/then/end if, if/then/else/end if, while/wend, do/loop while, select case/end select

Personal Web Server/IIS

- PWS is a single user server that runs on a PC under the Windows 9+
- A limited IIS comes with 200*/XP operating system
- PWS is a free download from Microsoft
- Both can be used to test CGI scripts and Active Server Pages
- They can be setup to run both CGI, ASP, PHP, Cold Fusion and other server processes
- ASP examples in these slides use IIS on a Microsoft server

PWS/IIS Setup

[C:\Inetpub\wwwroot]

Standard ASP Objects

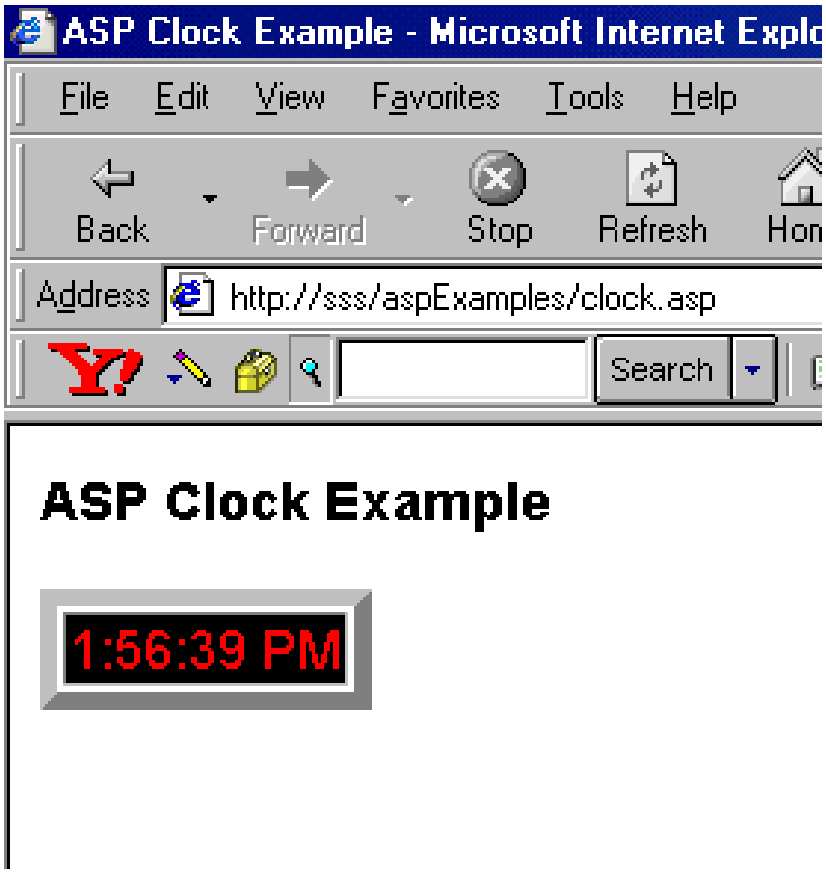
- ASP has several built in objects that the user can use to invoke the methods (functions) provided in these objects:
  - Request – methods provide access to data sent from the client browser
  - Response - methods provide access to data going back to the client browser
  - Server – methods (and properties) provide access to information about the server environment

clock.asp

```
<% @LANGUAGE = VBScript %>
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>ASP Clock Example</TITLE>
<META HTTP-EQUIV = "REFRESH" CONTENT = "1; URL=clock.asp">
</HEAD>
<BODY>
<FONT FACE = "Arial" SIZE = "4"><STRONG>ASP Clock Example</STRONG>
</FONT><P>
   <TABLE BORDER = "6">
   <TR>
   <TD BGCOLOR = "#000000">
      <FONT FACE = "Arial" COLOR = "red" SIZE = "4">
         <% =Time() %>
      </FONT>
   </TD>
   </TR>
   </TABLE>
</BODY>
</HTML>
```

"=Time()" is short for "Call Response.Write(Time())"
Calling the Write method of the Response Object

http://SSS/aspExamples/clock.asp



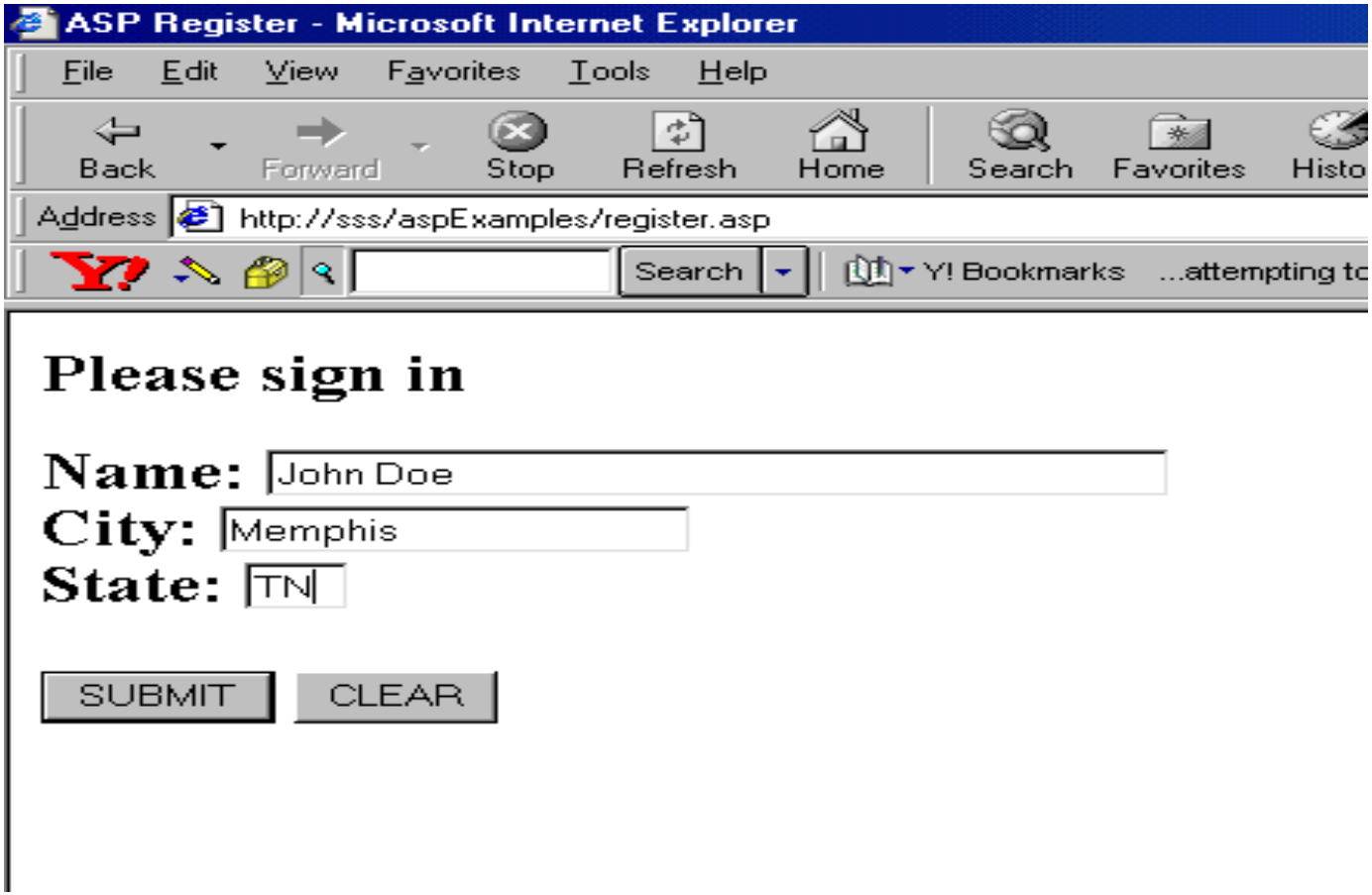
SSS is a directory under wwwroot !

ASP Version of Our Register Program [register.asp]

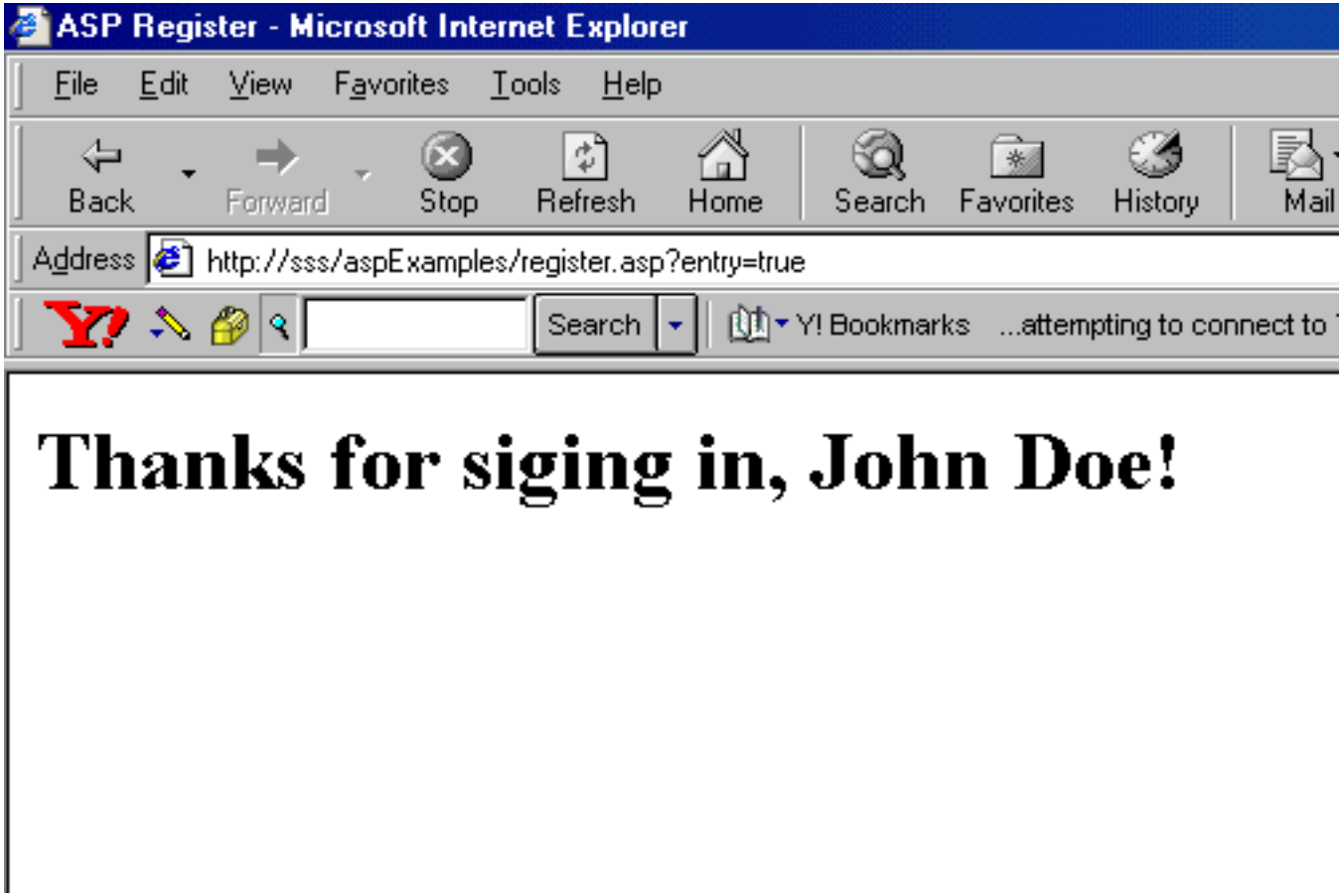
```
<% @LANGUAGE = VBScript %>
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>ASP Register</TITLE>
<BODY>
<H1>
<%
   If Request( "entry" ) = "true" Then
      Call Response.Write( "Thanks for siging in, " )
      Call Response.Write( Request( "name" ) & "!" )
   Else
%>
</H1>
<H2>Please sign in<BR>
<FORM ACTION = "register.asp?entry=true" METHOD = "POST">
Name:  <INPUT TYPE = "text" FACE = "Arial"    SIZE = "40" NAME = "name"><BR>
City:  <INPUT TYPE = "text" FACE = "Arial" SIZE = "20" NAME = "city"><BR>
State:  <INPUT TYPE = "text" FACE = "Arial" SIZE = "3" NAME = "state"><BR><BR>
<INPUT TYPE = "submit" VALUE = "SUBMIT">
<INPUT TYPE = "reset" VALUE = "CLEAR">
<%
      End If
%>
</BODY></HTML>
```

We have previously looked at this process in CGI (C or Perl)

ASP (con't)


ASP Register - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Histo
Address http://sss/aspExamples/register.asp
Search Y! Bookmarks ...attempting to
Please sign in
Name: John Doe
City: Memphis
State: TN
SUBMIT CLEAR

ASP (con't)


ASP Register - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail
Address http://sss/aspExamples/register.asp?entry=true
Search Y! Bookmarks ...attempting to connect to
Thanks for siging in, John Doe!

Server Side ActiveX Components

- Server side ActiveX controls (which can be invoked from ASP script) provide functionality for common server side functions such as:
  - AdRotator (automatically rotates images, usually advertisements, send to clients)
  - ContentRotator (provides content rotation)
  - Counters
  - Client Info
  - Server Info
  - File I/O
  - ADO (database access)

File System Objects

- File System Objects (FSO) allow one to read and write to text files and also to manipulate files and directories
- The objects are:
  - FileSystemObject
  - File
  - Folder
  - Drive
  - TextStream

FileSystem Object Methods

- The FileSystemObject provides for file system related manipulation including:
  - Creating files and directories
  - Copying
  - Deleting
  - Checking existence
  - Moving
  - Getting (finding)

Folder Methods & Properties

- The Folder object provides for directory related manipulation including:
  - Copying
  - Deleting
  - Moving
- Also provided are access to properties as: dates, path, size, etc.

File Object Methods & Properties

- The File object provides for file related manipulation including:
  - Copying
  - Deleting
  - Moving
  - Opening
- It also provides access to properties as file dates, path, size , etc.

Guestbook ASP Example

```
<% @LANGUAGE = VBScript %>
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>ASP GuestBook</TITLE>
<BODY>
<FONT SIZE = "4" FACE = "Arial" COLOR = "blue">
<%
   Dim fileObject, textFile, guestBook, mailtoUrl
   If Request( "entry" ) = "true" Then
        ' Print a thank you
      Call Response.Write( "Thanks for your entry, " )
      Call Response.Write( Request( "name" ) & "!" )
%>
```

Write Guest Record to File (guestbook.txt)

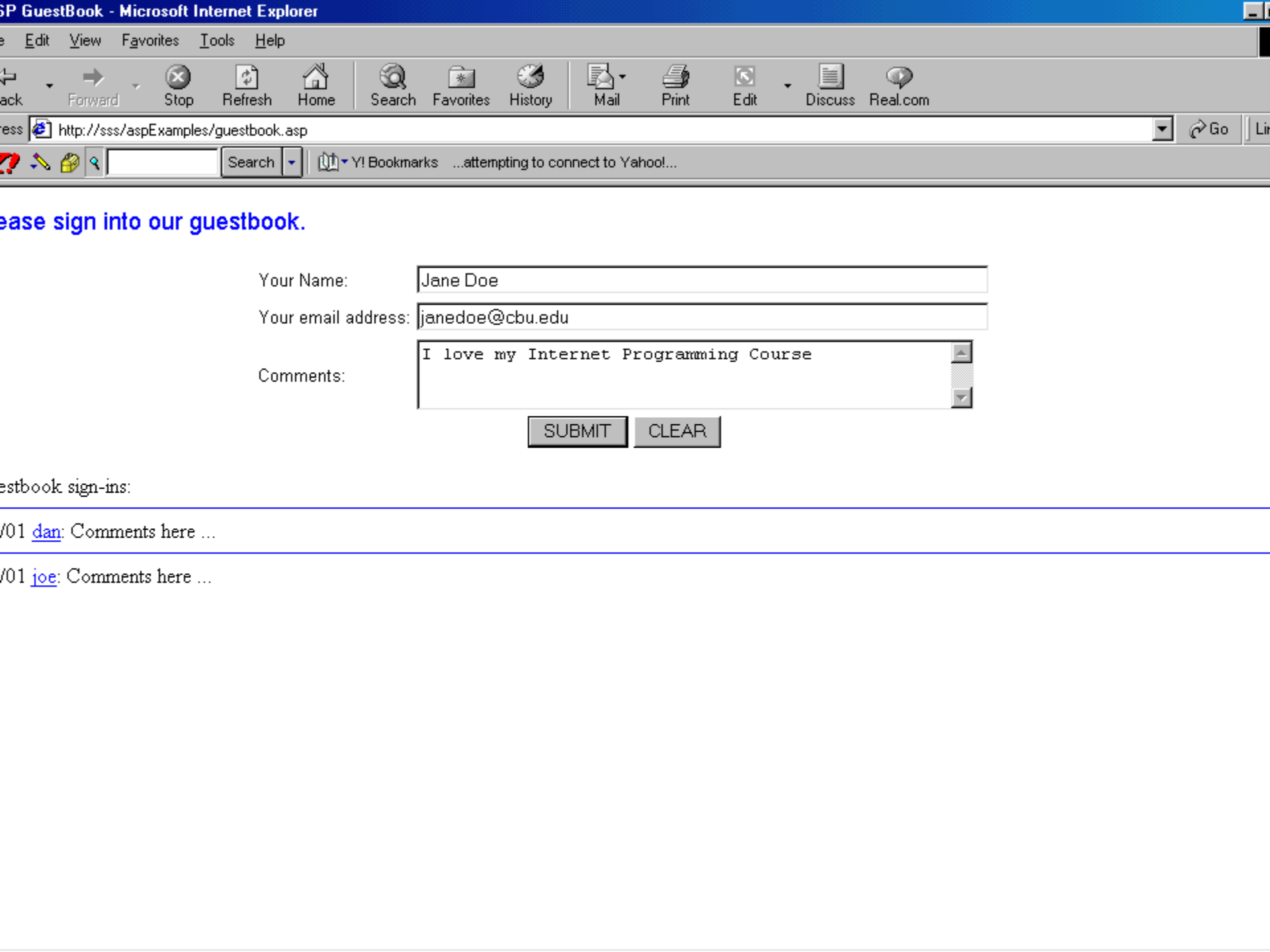
```
<HR COLOR = "blue" SIZE = "1">
<%    ' Instantiate a FileSystemObject
      Set fileObject = Server.CreateObject( _
         "Scripting.FileSystemObject" )
      guestBook = "c:\Inetpub\wwwroot\aspExamples\" & "guestbook.txt"
      ' Check if the file exists. If not, create it.
      If fileObject.FileExists( guestbook ) <> True Then
         Call fileObject.CreateTextFile( guestBook )
      End If
      ' Guestbook must be open for writing, 8 is for appending.
      Set textFile = fileObject.OpenTextFile( guestbook, 8, True )
      ' Build the mailtoUrl
      mailtoUrl = Date() & " <A HREF = " & Chr( 34 ) _
         & "mailto:" & Request( "email" ) & Chr( 34 ) _
         & ">" & Request( "name" ) & "</A>:  "
      ' Write data to guestbook.txt
      Call textFile.WriteLine( "<HR COLOR = " & Chr( 34 ) _
         & "blue" & Chr( 34 ) & " SIZE = " & Chr( 34 ) _
         & "1" & Chr( 34 ) & ">" )
      Call textFile.WriteLine( mailtoUrl )
      Call textFile.WriteLine( Request( "comment" ) )
      Call textFile.Close()
   End If
%>
```

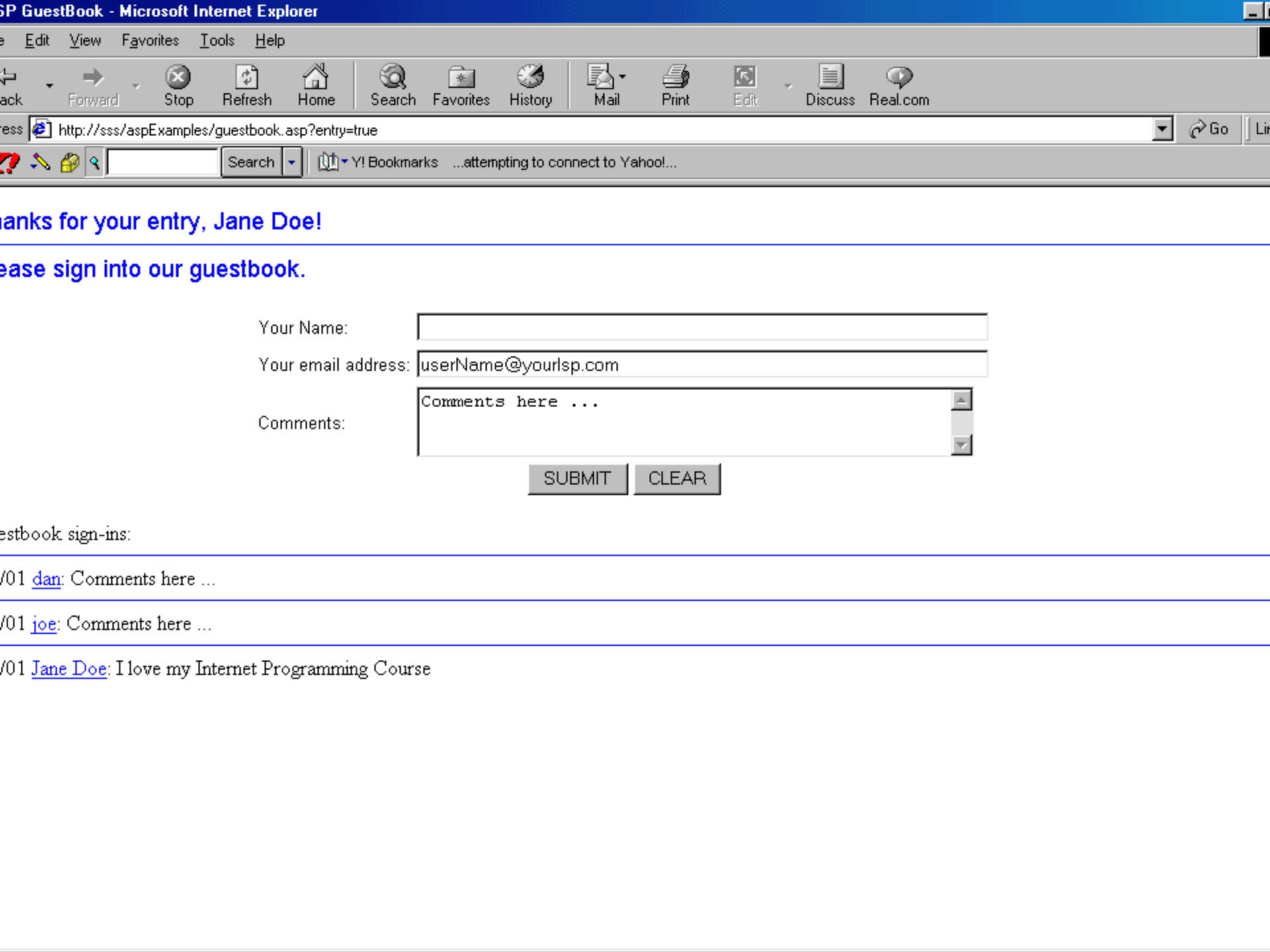
Form Portion of Guestbook

```
Please sign into our guestbook.
</FONT>
<FORM ACTION = "guestbook.asp?entry=true" METHOD = "POST">
<CENTER>
<TABLE>
<TR>
<TD><FONT FACE = "Arial" SIZE = "2">Your Name:  </FONT></TD>
     <TD><INPUT TYPE = "text" FACE = "Arial"
     SIZE = "60" NAME = "name"></TD></TR>
<TR>
<TD><FONT FACE = "Arial" SIZE = "2">Your email address:</FONT></TD>
     <TD><INPUT TYPE = "text" FACE = "Arial" SIZE = "60"
     NAME = "email" VALUE = "userName@yourIsp.com"></TD></TR>
<TR>
<TD><FONT FACE = "Arial" SIZE = "2">Comments:  </FONT></TD>
     <TD><TEXTAREA NAME = "comment" ROWS = "3" COLS = "50">
     Comments here ...
    </TEXTAREA></TD></TR>
</TABLE>
<INPUT TYPE = "submit" VALUE = "SUBMIT">
<INPUT TYPE = "reset" VALUE = "CLEAR">
</CENTER></FORM>
```

Print Guestbook Records

```
<%
   Dim fileObject2, textFile2
    ' Instantiate a FileSystemObject
    Set fileObject2 = Server.CreateObject( _
       "Scripting.FileSystemObject" )
    guestBook = "c:\Inetpub\wwwroot\aspExamples\" & "guestbook.txt"
    ' Check if the file exists. If not, create it.
    If fileObject2.FileExists( guestBook ) = True Then
       ' Guestbook must be open for writing.
       ' Open the guestbook, "1" is for reading.
       Set textFile2 = fileObject2.OpenTextFile( guestbook, 1 )
       ' Read the entries from the file and write them to
       ' the client.
       Call Response.Write( "Guestbook sign-ins:<BR>" )
       Call Response.Write( textFile2.ReadAll() )
       Call textFile2.Close()
    End If
%>
</BODY>
</HTML>
```


SP GuestBook - Microsoft Internet Explorer
Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss Real.com
http://sss/aspExamples/guestbook.asp
Go
Search
Y! Bookmarks ...attempting to connect to Yahoo!...
ease sign into our guestbook.
Your Name:
Jane Doe
Your email address:
janedoe@cbu.edu
Comments:
I love my Internet Programming Course
SUBMIT
CLEAR
estbook sign-ins:
/01 dan: Comments here ...
/01 joe: Comments here ...


SP GuestBook - Microsoft Internet Explorer
Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss Real.com
http://sss/aspExamples/guestbook.asp?entry=true
Go
Search
Y! Bookmarks ...attempting to connect to Yahoo!...
anks for your entry, Jane Doe!
ease sign into our guestbook.
Your Name:
Your email address: userName@yourIsp.com
Comments: Comments here ...
SUBMIT
CLEAR
estbook sign-ins:
/01 dan: Comments here ...
/01 joe: Comments here ...
/01 Jane Doe: I love my Internet Programming Course

Session Tracking

- The ASP process can maintain “state” between client/server internet connection (unlike CGI which is “stateless”)
- A “Session” object is available, and variables can be saved by setting and reading theses Session variables
- “Cookies” can also be setup with ASP

Open Database Connectivity [ODBC]

- Developed in 1990's by X/Open and SQL Access Group committees to provide an open DBMS independent API for relational databases (can also be used for non-relational databases)
- First fully implemented by Microsoft; now implemented by most all DBMS vendors and third party suppliers

ODBC Data Sources

- An ODBC Data Source is the combination of the database, it's associated DBMS, operating system, and network middleware (if any).
- A data source can be an Oracle database, Sybase database, SQLServer database, Access database (.mdb file), Excel spreadsheet, Btrieve file server, etc.

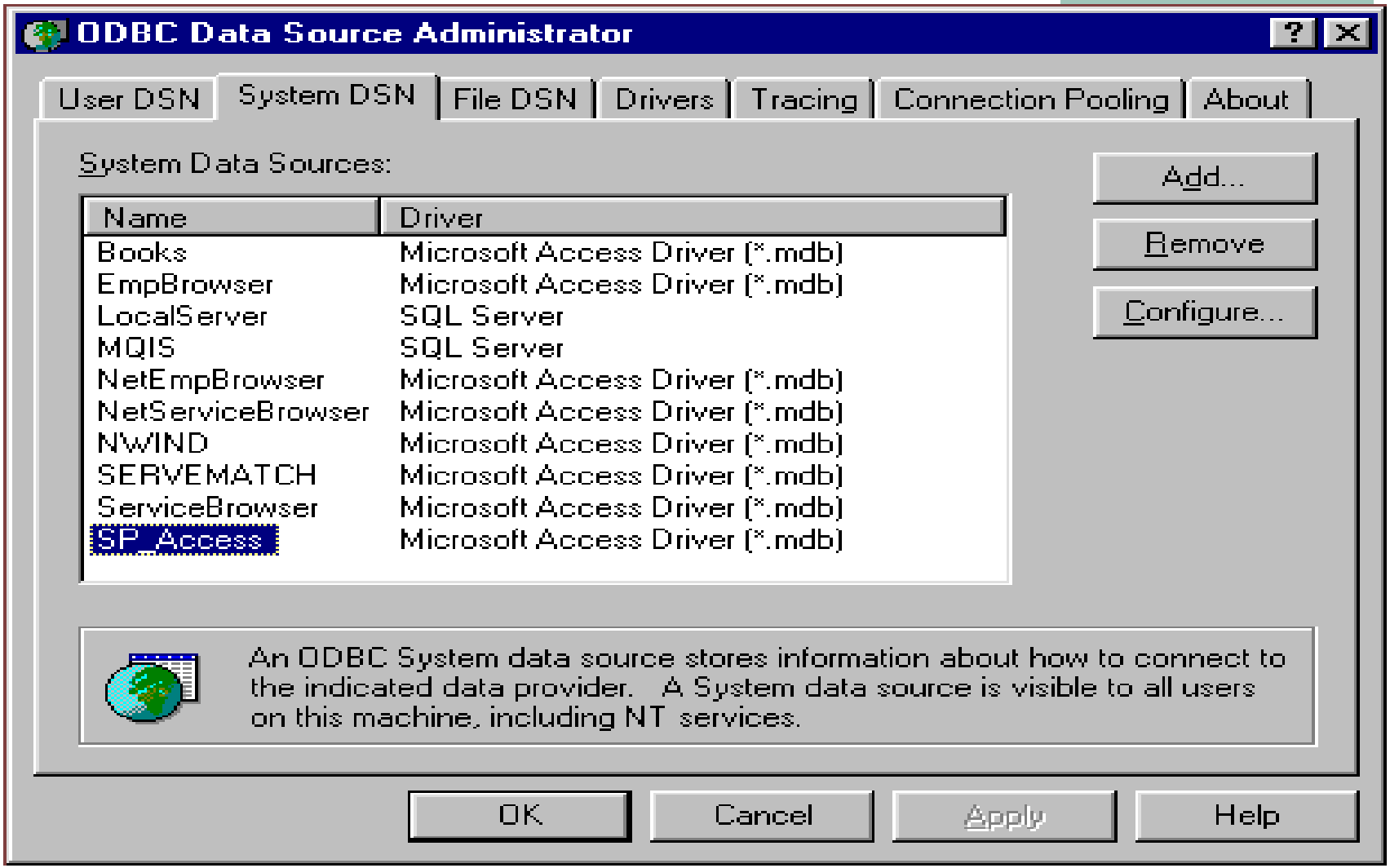
ODBC Components

- Application - makes calls using ODBC API functions to create a connection to a data source, issues dynamic SQL statements, get back result sets, start-commit-rollback transactions, and receive status/error conditions
- Driver Manager - loads appropriate driver and handles communication between the application and the specific data source driver
- DBMS Driver - turns driver manager commands into specific SQL for target database

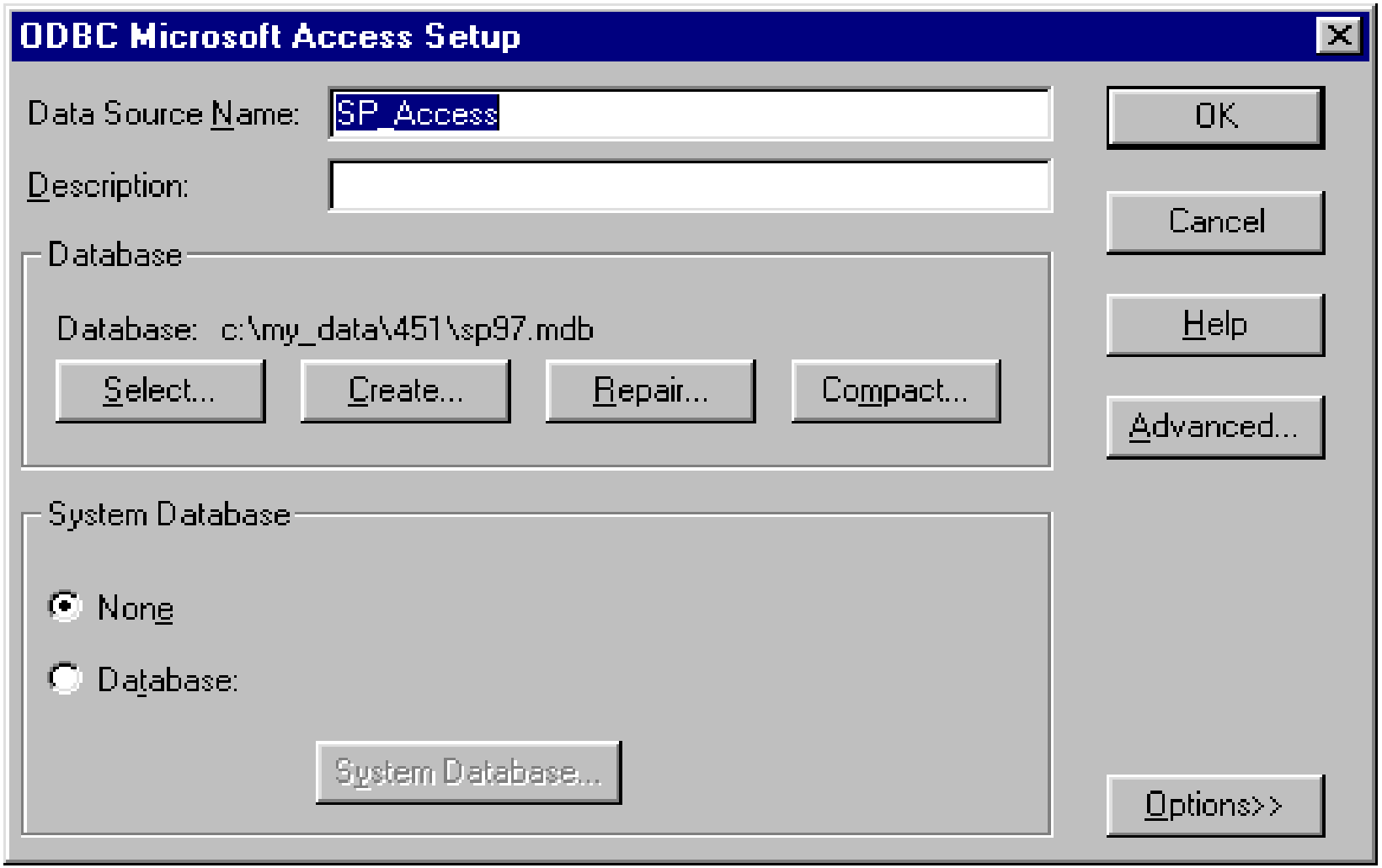
Data Source Types

- File Data Source - A file that is shared among users; users all have same driver and privileges
- System Data Source - A source that is local to a single computer
- User Data Source - only available to user who created it
- Generally you set up a System Data Source on a local PC or a File Data Source on a server on which your CGI/ISAPI/ASP/JSP programs execute

ODBC Data Source



ODBC (con't)


ODBC Microsoft Access Setup
Data Source Name:
SP_Access
Description:
Database
Database: c:\my_data\451\sp97.mdb
Select...
Create...
Repair...
Compact...
System Database
None
Database:
System Database...
OK
Cancel
Help
Advanced...
Options>>

ODBC Conformance Levels

- There are three conformance levels that databases may adhere to:
  - Core
    - Connect, Prepare & Execute SQL, Retrieve data from result set, commit/rollback transactions, get error information
  - Level 1
    - Driver specific info upon connect,send/receive partial results, retrieve meta data, get driver info
  - Level 2
    - Browse connections/sources, use native SQL, scrollable cursors, translation libraries

SQL Conformance Levels (con't)

- Specifies SQL capability of driver:
  - Minimal
    - Create/Drop Table, simple Select, Insert/Update/Delete, simple expressions, character data types
  - Core
    - Indexes, Views, Grant/Revoke, Full Select (incl. Subqueries), aggregate functions, numeric types
  - Extended
    - Outer Joins, Positioned Update/Delete, scalar functions (abs,…), date/time/timestamp, batch sql, stored procedures

Microsoft Component Models

- Either C++ or Visual Basic can be used to create ActiveX objects (C++ is more powerful); objects can be GUI or batch (batch versions are commonly used in ASP's)
- ActiveX is a specialized implementation of Microsoft's Component Object Model (COM) – now replaced by ".NET"
- COM is based on sets of interface specifications (specifications for object function calls [classes without data, only functions]- does not support inheritance
- DCOM is Distributed COM
- OLE objects are COM objects that support interfaces for embedding
- ActiveX Controls are ActiveX objects with "introspection"

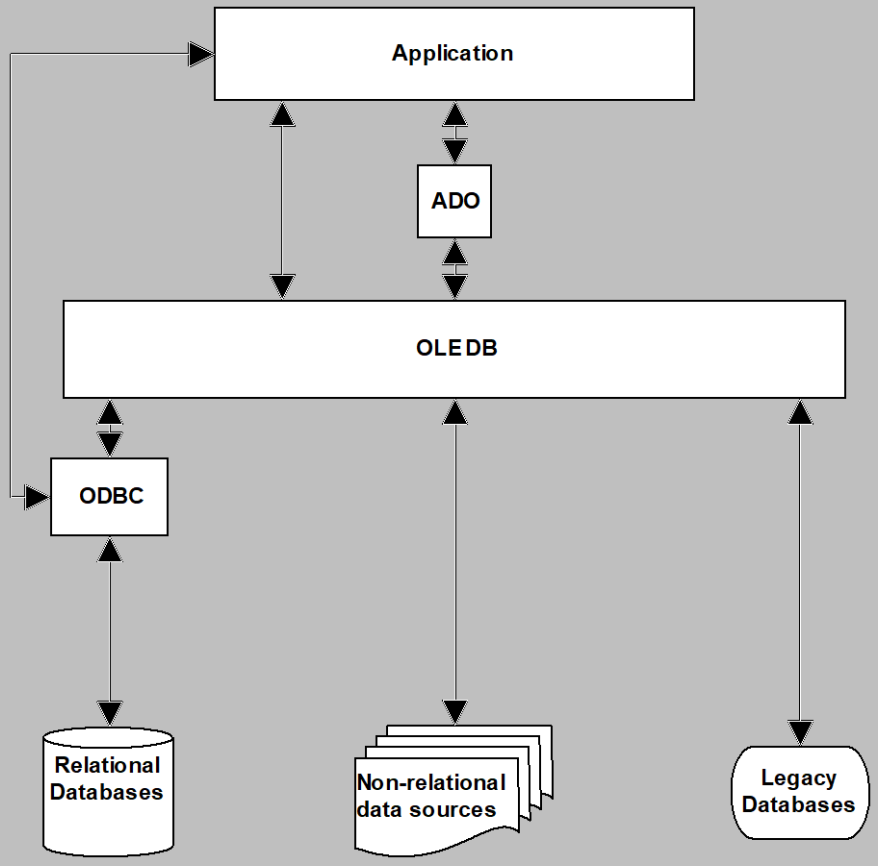
OLE DB

- OLE DB is an object oriented API created by Microsoft that is best utilized via C++ programs
- It allows programs running on web servers (such as CGI programs) to access both ODBC compliant and native database drivers thru a common set of API calls

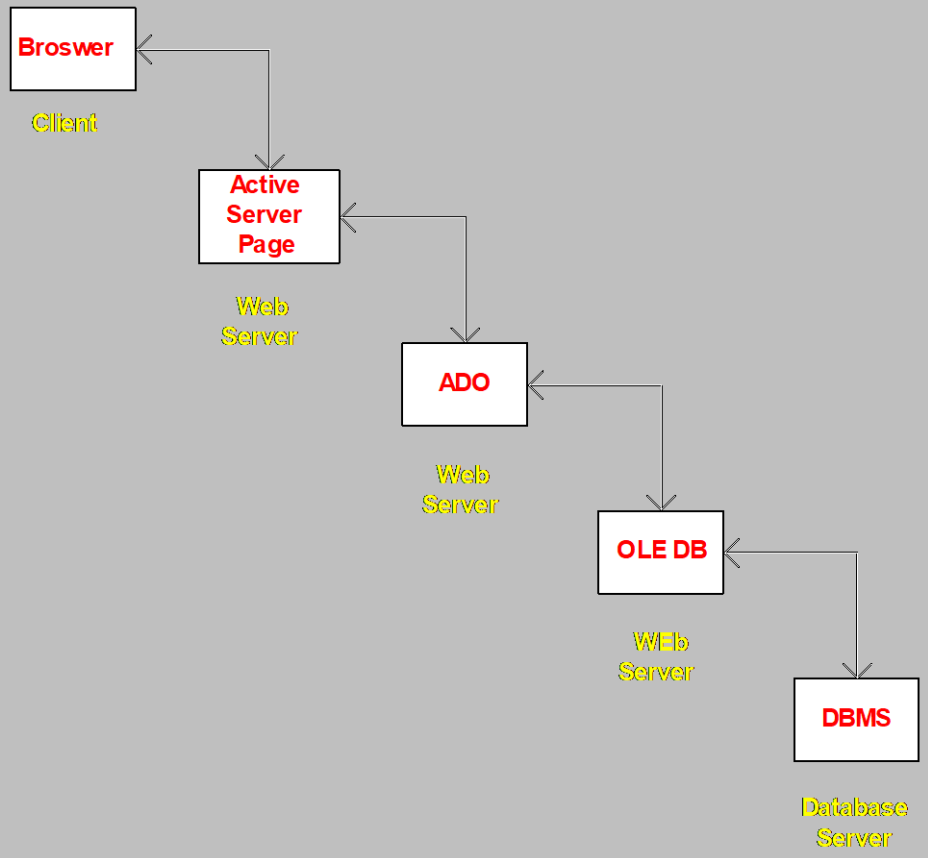
Active Data Objects (ADO)

- ADO provides a preprogrammed cover over OLE DB objects
- So that a non object oriented languages (like Visual Basic or VBScript) can make use of many OLE DB functions

Microsoft's UDA (Universal Data Access)



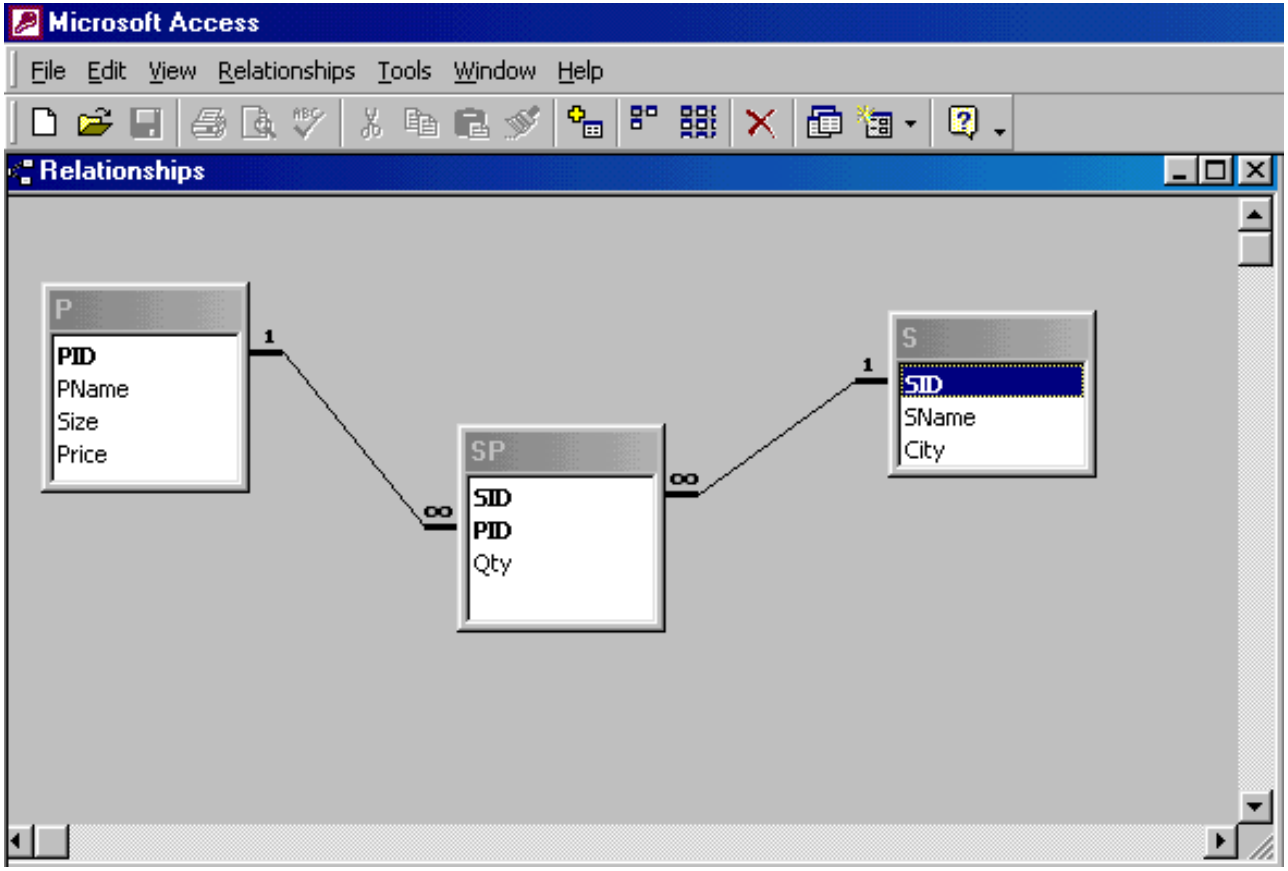
Using Active Server Pages for Database



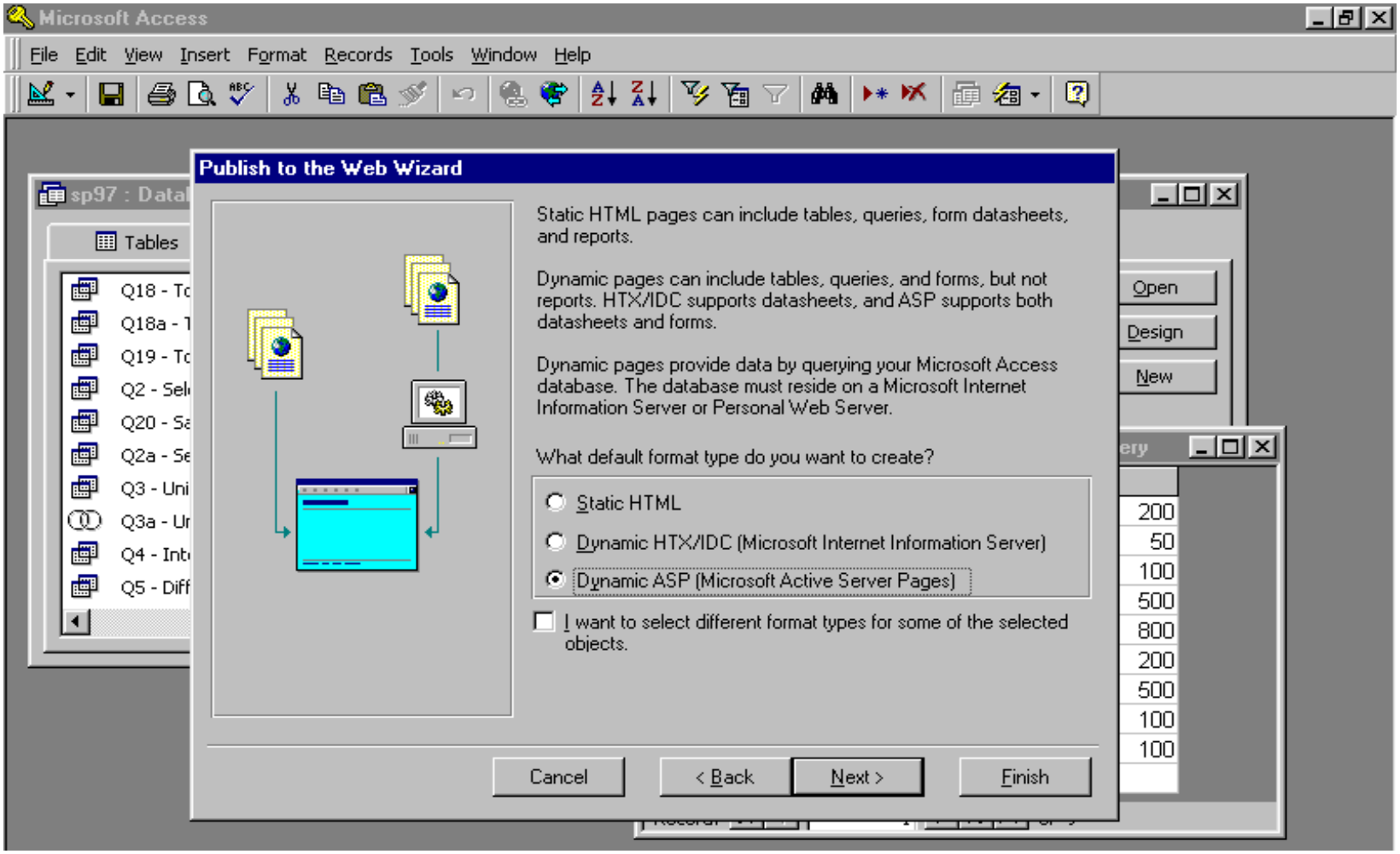
Creating Database ASP's

- Microsoft Access
- Microsoft Visual Studio (now .NET products)
- Case Packages
- Manually

Access Database Example



Using Access to Generate ASP



Generated ASP

```
<HTML><HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>Q7 - Simple Join (Class Exercise)</TITLE></HEAD><BODY>
<%
Param = Request.QueryString("Param")
Data = Request.QueryString("Data")
%>
<%
If IsObject(Session("SP_Access_conn")) Then
    Set conn = Session("SP_Access_conn")
Else
    Set conn = Server.CreateObject("ADODB.Connection")
    conn.open "SP_Access","",""
    Set Session("SP_Access_conn") = conn
End If
%>
<%
    sql = "SELECT DISTINCTROW SP.SID, P.PName, SP.Qty  FROM P INNER JOIN SP ON P.PID = SP.PID
"
    If cstr(Param) <> "" And cstr(Data) <> "" Then
        sql = sql & " WHERE [" & cstr(Param) & "] = " & cstr(Data)
    End If
    Set rs = Server.CreateObject("ADODB.Recordset")
    rs.Open sql, conn, 3, 3
%>
```

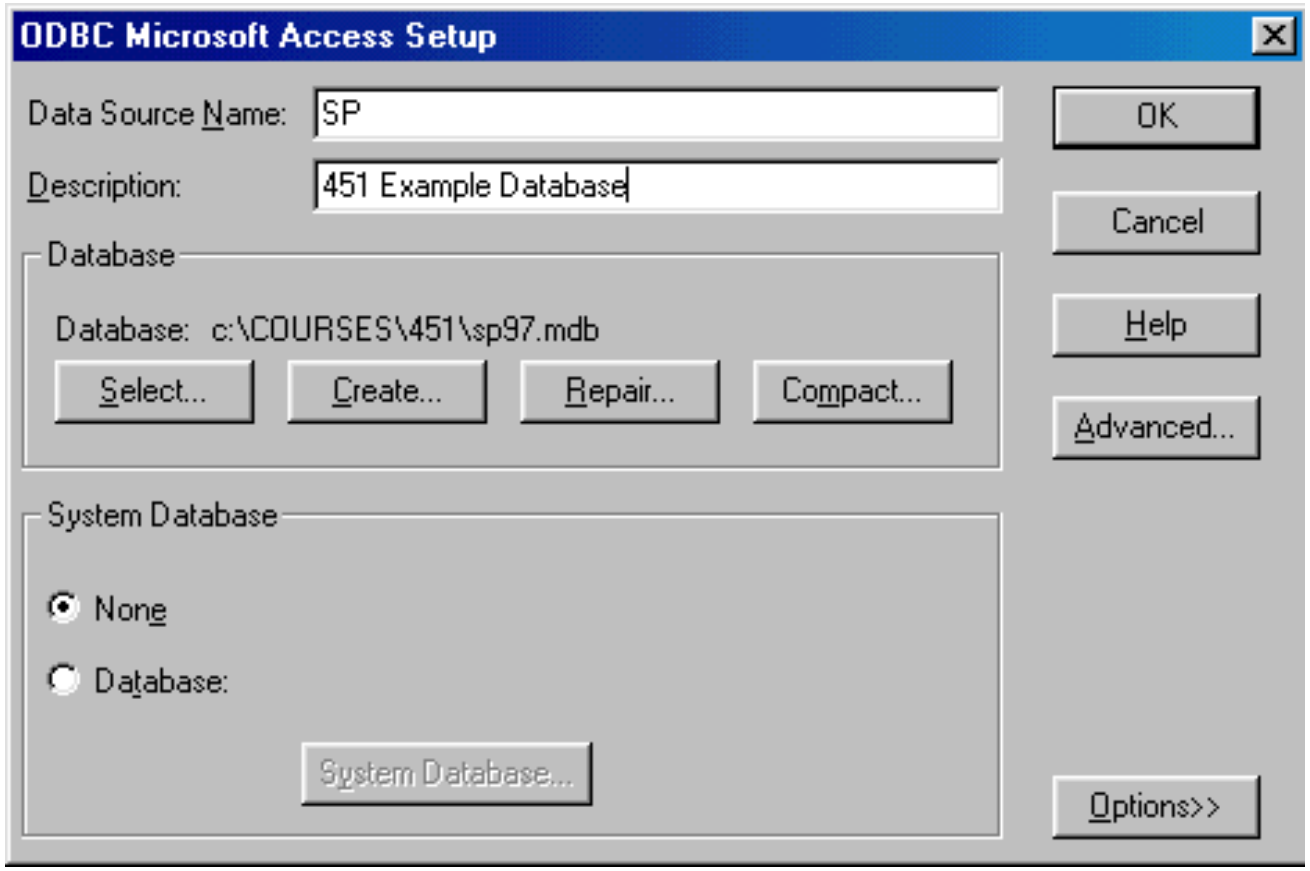
Write results into html table:

```
<TABLE BORDER=1 BGCOLOR=#ffffff CELLSPACING=0><FONT FACE="Arial"
COLOR=#000000><CAPTION><B>Q7 - Simple Join (Class Exercise)</B></CAPTION>
<THEAD><TR>
<TH BGCOLOR=#c0c0c0 BORDERCOLOR=#000000 ><FONT SIZE=2 FACE="Arial"
COLOR=#000000>SID</FONT></TH>
<TH BGCOLOR=#c0c0c0 BORDERCOLOR=#000000 ><FONT SIZE=2 FACE="Arial"
COLOR=#000000>PName</FONT></TH>
<TH BGCOLOR=#c0c0c0 BORDERCOLOR=#000000 ><FONT SIZE=2 FACE="Arial"
COLOR=#000000>Qty</FONT></TH></TR>
</THEAD><TBODY>
<%On Error Resume Next
rs.MoveFirst
do while Not rs.eof %>
<TR VALIGN=TOP>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial"
COLOR=#000000><%=Server.HTMLEncode(rs.Fields("SID").Value)%><BR></FONT></TD>
<TD BORDERCOLOR=#c0c0c0 ><FONT SIZE=2 FACE="Arial"
COLOR=#000000><%=Server.HTMLEncode(rs.Fields("PName").Value)%><BR></FONT></TD>
<TD BORDERCOLOR=#c0c0c0  ALIGN=RIGHT><FONT SIZE=2 FACE="Arial"
COLOR=#000000><%=Server.HTMLEncode(rs.Fields("Qty").Value)%><BR></FONT></TD>
</TR>
<%rs.MoveNextloop%>
</TBODY><TFOOT></TFOOT></TABLE></BODY></HTML>
```

Manually Written ASP

```
<% @LANGUAGE = VBScript %>
<% Option Explicit %>
<HTML>
<HEAD><TITLE>ODBC Example</TITLE></HEAD>
<% Dim connection, query, data
   Set connection = Server.CreateObject( "ADODB.Connection" )
   Call connection.Open( "SP" )
   ' Create the SQL query
   query = "SELECT * FROM S"
   ' Create the recordset
   Set data = Server.CreateObject( "ADODB.Recordset" )
   Call data.Open( query, connection )
   ' If an error occurs, ignore it
   On Error Resume Next
%>
<BODY>
<H1>List of Salespersons</H1>
<%While Not data.EOF%>
      <H2><% =data( "SName" ) %></H2>
      <%Call data.MoveNext()
Wend%>
</BODY></HTML>
```

ODBC System DSN



Running ASP


ODBC Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favor
Address http://sss/aspExamples/sp.asp
Search Y! Bookmarks
List of Salespersons
Peterson
Olsen
Hansen
Jensen