INTERNET & WEB APPLICATION DEVELOPMENT SWE 444

Fall Semester 2008-2009 (081)

Module 5.1: Server-Side Programming

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives/Outline

- Objectives
  - Understand the power of server-side coding
  - Learn about the application architecture and sever-side technologies
  - Learn how to use ASP

- Outline
  - What is a Server?
  - Server Architectures
  - Why Server-Side Coding?
  - Server-Side Technologies
  - Overview of ASP

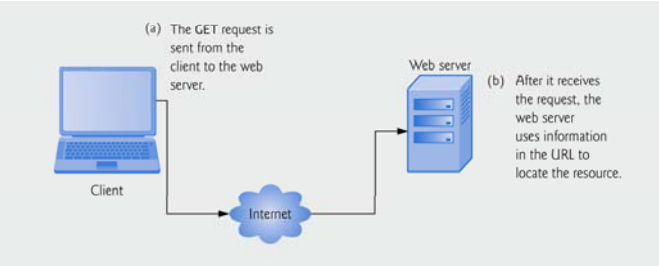
What is a Server?

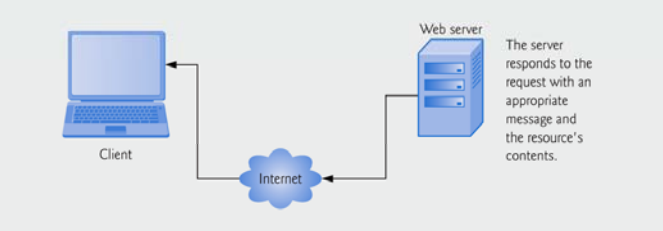
- Server
  - A computer (or the software that runs on it) that provides services to others
- Many types of servers
  - File server — file: networked file space
  - FTP server — ftp: remote file space, often read-only
  - Web server — http: web pages and more
  - Mail server — mail: email system
  - News server — news: newsgroups messages

Web Server

- Web server provides web resources such as XHTML documents to web browsers (clients)
- Web browser interacts with a web server using HTTP protocol for transferring requests and files over the Internet or Intranet.
  - HTTP protocol allows clients and servers to exchange information in a uniform and reliable manner
- HTTP uses URIs to identify data on the network

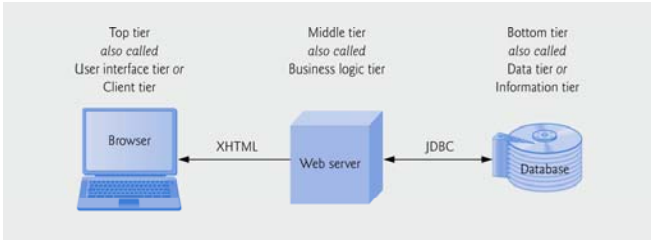
Client interacting with web server



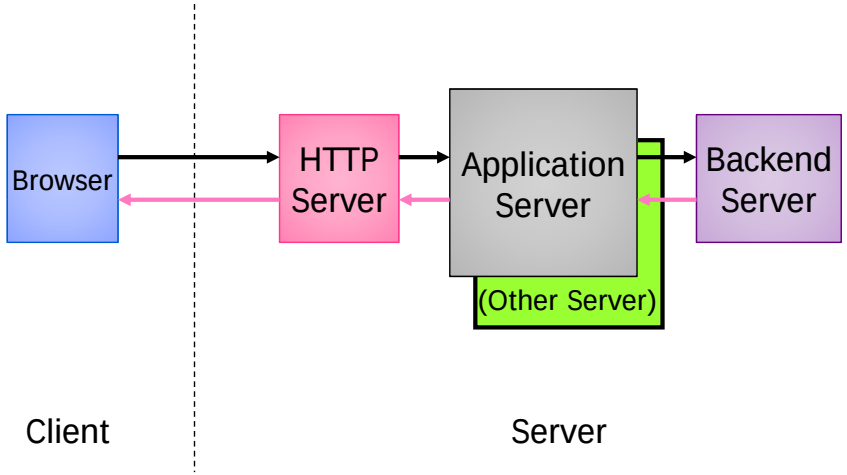


Application Architectures

- Multi-tier applications divide functionality into separate tiers (i.e., logical groupings of functionality)
- Tiers can be located on the same computer or on separate computers
- A three-tier architecture consists of
  - The client/top tier -- is the application's user interface
  - Information/data/bottom-tier -- maintains data for the application, typically stores data in relational DBMS
  - Middle-tier -- implements *business logic* to control interactions between application clients and application data



General Multi-tier Architecture

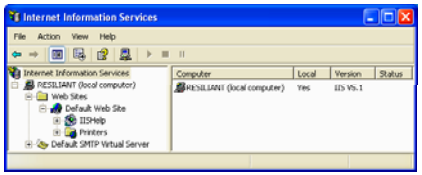


Accessing Web Servers

- To request documents from web servers, users must know the hostnames on which the web server software resides.
- Users can request documents from local web servers or remote web servers.
- Local web servers can be accessed through your computer's name or through the name `localhost`—a hostname that references the local machine and normally translates to the IP address `127.0.0.1` (also known as the loopback address).

Common Web Servers

- Microsoft Internet Information Services (IIS)



- Apache HTTP Server
  - maintained by the Apache Software Foundation, is currently the most popular web server. It is open source software that runs on UNIX, Linux, Mac OS X, Windows and numerous other platforms.

Server-side vs. client-side script

- Client-side scripting
  - can be used to validate user input, to interact with the browser, to enhance web pages by manipulating the DOM of a page, and to add Ajax functionality
  - does have limitations, such as browser dependency; the browser or scripting host must support the scripting language and capabilities.
  - can be viewed by the client by using the browser's source-viewing capability
    - Sensitive information, such as passwords or other personally identifiable data, should not be stored or validated on the client

Server-side vs. client-side script (cont.)

- Placing large amounts of JavaScript on the client can open web applications to attack and other security issues
- Code executed on the server often generate custom responses for clients
- Server-side scripting languages have a wider range of capabilities than their client-side equivalents.
  - E.g., server-side scripts often can access the server's file directory structure, whereas client-side scripts cannot access the client's directories
- Properly configured server-side scripts are not visible to the client

Why Server-Side Coding?

- Security – Source code is not exposed
  - Once user is authenticated, can only allow certain actions
- Manageability – Does not require distribution of application code
  - Easy to change code
- Accessibility – You can reach the Internet from any browser, any device, any time, anywhere
- Scalability -- Web-based 3-tier architecture can scale out
  - If bottlenecks in terms of performance occur, the server process can be moved to other servers at runtime

Server-Side Technologies

Scripting Languages:
- Server Side Includes (SSI)
- Perl
- PHP
- ASP (VBScript)
- Python

Compiled Languages:
- C
- C++
- C#
- ASP .Net
- Java Servlets
- Java Server Pages (JSP)
  - Looks like a scripting language, but is actually compiled into a Java Servlet

- Can download and render the page while interpreted
- Portability

- Either portable byte code (such as a Java .class file) or a true executable (native to the microprocessor) is produced

- Common to all scripting languages is some sort of real-time interpreter that parses text and turns it into executable instructions for the server

Choosing Technologies ?

- Some criteria affecting decisions
  - Web server availability
  - Knowledge of language
  - Scalability and efficiency
  - Personal preference

What is ASP?

- ASP stands for Active Server Page
  - ASP is a Microsoft technology for server-side programming
  - ASP program runs inside IIS (Internet Information Services)
    - ChiliASP and InstantASP are other technologies that run ASP without windows
- What is an ASP file?
  - An ASP file (with ".asp" extension) can contain text, HTML and XML elements, and client-side scripts
    - Also it can contain server scripts, surrounded by the delimiters <% and %>.
    - Server scripts are executed on the server, and can contain any expressions, statements, procedures, or operators valid for the scripting language you prefer to use
- How does ASP differ from HTML?
  - When a browser requests an HTML file, the server returns the file
  - When a browser requests an ASP file, IIS passes the request to the ASP engine
    - Which reads the ASP file, line by line, and executes the server-side scripts in the file. Finally, the ASP file is returned to the browser as plain HTML
    - Thus allowing making dynamic and interactive Web pages

ASP Intrinsic Objects

- Six built-in objects in the ASP environment used to provide services:
  - Request
    - Used to access information passed by an HTTP request (get information from users)
  - Response
    - Used to control information sent to the client (send information to users)
  - Server
    - Provides access to methods and properties on the server
  - Application
    - Used to hold information that can be used by many pages (e.g., DB connections)
  - Session
    - Used to maintain information about a user session
    - `Application` variables store info for all users while `Session` variables store info about a single user
  - ObjectContext
- Commonly uses ADO to interact with databases

Example 1: Salam Shabab

```
<%@ language="javascript"%>
<html>
<head><title>Salam.asp</title></head>
<body>
<%
Response.Write("Salam Shabab!")
%>
</body>
</html>
```

server script is surrounded by the delimiters **<%** and **%>**

- Shorthand method for the response.write command

```
<%= "Salam Shabab!" %>
```


Example 2: Salam Shabab

```
<%@ language="javascript"%>
<html>
<head><title>Salam2.asp</title></head>
<body>
<form method="post">
<input type="submit" id="bt" name="bt" value="Push Me" />
<%
if (Request.Form("bt") != "")
  Response.Write("<p>Salam, the time is " + Now()+"</p>");
%>
</form>
</body>
</html>
```


Server-Side Form Processing

- We have seen examples of client-side processing in earlier sessions
  - Where all processing is local to the Web page, encapsulated within browser scripts or event handlers.
- Form controls can be used for submitting information from a Web page to a processing program located on the Web server.
  - In this case, the controls to capture information are surrounded by a **<form>** tag containing **action** and **method** attributes.
  - **action** gives the URL of the external page that handles the processing
  - **method** indicates how the form information is transmitted (normally through the **post** method).
  - The form includes
    - a "submit" button to activate the form submission
    - an optional "reset" button can be coded to automatically clear all form fields.

Example 3: Handling User Input

```
<form method="get" action="simpleform.asp">
First Name:
  <input type="text" name="fname">
 <br />
Last Name:
  <input type="text" name="lname">
<br /><br />
<input type="submit" value="Submit">
</form>
```

… Handling User Input

- User input can be retrieved in two ways: With `Request.QueryString` or `Request.Form`.
- The `Request.QueryString` command is used to collect values in a form with `method="get"`
  - Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send.
- If a user typed "`Feras`" and "`Nabulsi`" in the form example above, the URL sent to the server would look like this:
  - http://www.w3schools.com/simpleform.asp?fname=Feras&lname=Nabulsi
- The `Request.Form` command is used to collect values in a form with `method="post"`.
  - Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.
- If a user typed "Feras" and "Nabulsi" in the form example above, the URL sent to the server would look like this:
  - http://www.w3schools.com/simpleform.asp

… Handling User Input

- Assume `simpleform.asp` has the following simple script:

```
<body> Welcome
<%
  response.write(request.querystring("fname"));
   response.write(" " +
  request.querystring("lname"));
 %>
</body>
```

- For both cases above, the browser will display:
  `Welcome Feras Nabulsi`

ASP Challenges

- Coding overhead (too much code)
  - Everything requires writing code!
- Code readability (too complex; code and UI intermingled)
- Maintaining page state requires more code
  - After submit button is clicked, if we click the back button, we expect to maintain scroll position, maintain which control had focus, and restore focus, or allow server code to focus a new control
- Reuse is difficult – lack of modularity
- Supporting many types of browsers is difficult – ASP.NET has better browser support
- Deployment issues (e.g. DLL locking) – ASP.NET easier to deploy
- Session state scalability and availability – ASP uses cookies to maintain state while ASP.NET uses cookieless means

Q & A

References

- H. M. Deitel, P. J. Deitel, and A. B. Goldberg, Internet and World Wide Web How to Program, 4/e, Pearson Education Inc., 2008.
- Some useful links with examples and other resources:
  - W3School ASP Tutorial
    - http://www.w3schools.com/asp/default.asp
  - W3School ADO Tutorial
    - http://www.w3schools.com/ado/default.asp
  - W3School SQL Tutorial
    - http://www.w3schools.com/sql/default.asp
  - W3School PHP Tutorial
    - http://www.w3schools.com/php/default.asp