CS/INFO 330
Middle Tier Technology

Mirek Riedewald
mirek@cs.cornell.edu

Overview

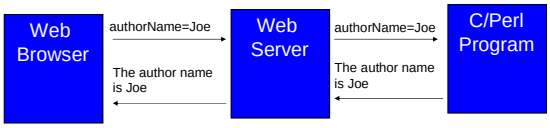
- Recall: Functionality of the middle tier
  - Encodes business logic
  - Connects to database system(s)
  - Accepts form input from the presentation tier
  - Generates output for the presentation tier
- We will cover
  - CGI: Protocol for passing arguments to programs running at the middle tier
  - Application servers: Runtime environment at the middle tier
  - Servlets: Java programs at the middle tier
  - JavaServerPages: Java scripts at the middle tier
  - Maintaining state: How to maintain state at the middle tier

CGI: Common Gateway Interface

- General framework for creating server side web applications
- Instead of returning static web document, web server returns results of a program
  - Transmit arguments from HTML forms to application programs running at the middle tier
  - Details of the actual CGI protocol unimportant; libraries implement high-level interfaces
- First mechanism for creating dynamic web sites
- Can create CGI programs in almost any programming language

CGI Overview

- Browser sends parameter authorName=Joe
- Web server passes request to a Perl program
- Perl Program returns HTML that says “The author name is Joe”



CGI Example

- HTML form:

```
<form action=“findbooks.cgi” method=POST>
Type an author name:
<input type=“text” name=“authorName”>
<input type=“submit” value=“Send it”>
<input type=“reset” value=“Clear form”>
</form>
```

- Perl code:

```
use CGI;
$dataIn=new CGI;
$dataIn->header();
$authorName=$dataIn->param(‘authorName’);
print(“<HTML><TITLE>Argument passing test</TITLE>”);
print(“The author name is “ + $authorName);
print(“</HTML>”);
exit;
```

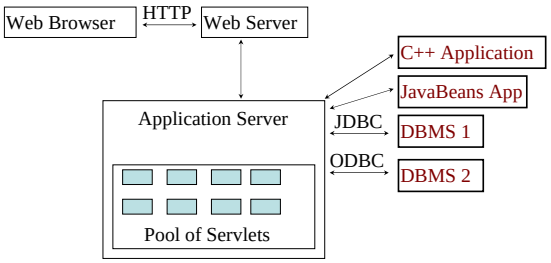

CGI Disadvantages

- The application program is invoked in a new process at every invocation (remedy: FastCGI)
- No resource sharing between application programs (e.g., database connections)
- Remedy: Application servers

Application Servers

- Idea: Avoid overhead of CGI
  - Main pool of threads of processes
  - Manage connections
  - Enable access to heterogeneous data sources
  - Other functionality such as APIs for session management

App Server Process Structure



Servlets

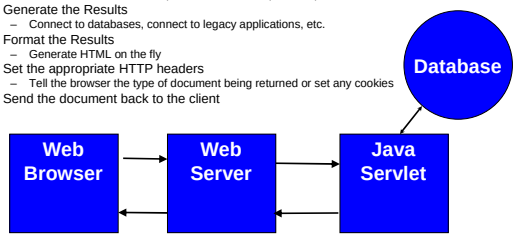
- Java Servlets: Java code that runs on the middle tier, either in web server or application server (Java's answer to CGI)
  - Applet: java program that runs within the web browser
  - Servlet: java program that runs within the web server
  - Platform independent; complete Java API available, including JDBC
- Example:

```
import java.io.*;
import java.servlet.*;
import java.servlet.http.*;

public class ServetTemplate extends HttpServlet {
        public void doGet(HTTPServletRequest request,
        HTTPServletResponse response)
        throws SerletExpection, IOException {
                PrintWriter out=response.getWriter();
                out.println("Hello World");
        }
}
```

Servlet Processing Client Request

- Read any data sent by the user
  - Capture data submitted by an HTML form
- Look up any HTTP information
  - Determine browser version, host name of client, cookies, etc.
- Generate the Results
  - Connect to databases, connect to legacy applications, etc.
- Format the Results
  - Generate HTML on the fly
- Set the appropriate HTTP headers
  - Tell the browser the type of document being returned or set any cookies
- Send the document back to the client



"Complete" Servlet Example

- Webserver forwards request to Servlet container
- Container creates Servlet instance (calls init() method; at deallocation time calls destroy())
- Container calls service() method
  - service() calls doGet() for HTTP GET or doPost() for HTTP POST
  - Usually, don't override service(), but override doGet() and doPost()

```
public class ReadUserName extends HttpServlet {
    public void doGet( HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        reponse.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.println("<HTML><BODY>\n <UL> \n" + "<LI>" +
                request.getParameter("userid") + "\n" + "<LI>" +
                request.getParameter("password") + "\n" +
                "<UL>\n<BODY></HTML>");
    }

    public void doPost( HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        doGet(request,response);
    }
}
```


What Can You Build with Servlets?

- Search Engines
- Personalization Systems
- E-Commerce Applications
- Shopping Carts
- Product Catalogs
- Intranet Applications
- Groupware Applications: bulletin boards, file sharing, etc.

Java Server Pages

- Servlets
  - Generate HTML by writing it to the "PrintWriter" object
  - Code first, webpage second
- JavaServerPages
  - Written in HTML, Servlet-like code embedded in the HTML
  - Webpage first, code second
  - Usually compiled into a Servlet

Java Servlet :
Looks like a regular
Java program

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

public class HelloWorld extends HttpServlet {
  public void doGet(HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Hello World</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("<BIG>Hello World</BIG>");
    out.println("</BODY></HTML>");
  }
}
```


JSP Page :
Looks like a regular
HTML page.

```
<html>
<head>
<title>Hello, World JSP Example</title>
</head>
<body>
 <h2> Hello, World!
The current time in milliseconds is
<%= System.currentTimeMillis() %>
</h2>
</body>
</html>
```

Embedded Java
command to
print current time.

Some Server Side Options

- Common Gateway Interface (CGI)
- Fast CGI
- Mod Perl
- Server Extensions
  - NSAPI
  - ISAPI
- ASP
- PHP
- Cold Fusion
- Ruby on Rails

Common Features

- All server side frameworks share a common set of features:
  - Read data submitted by the user
  - Generate HTML dynamically based on user input
  - Determine information about the client browser
  - Access Database systems
  - Exploit the HTTP protocol

Decision Points

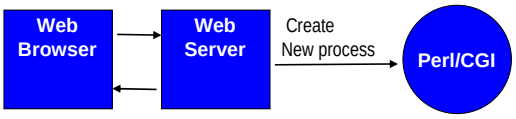
- When evaluating which server side framework to use, you need to consider a number of critical factors:
  - Ease of development:
    - How easily can you build new applications?
  - Performance:
    - How fast can the framework respond to queries?
  - Scalability:
    - Can the framework scale to thousands, millions of users?
  - Security:
    - Are there any inherent security vulnerabilities?

Option 1: CGI

- One of the earliest, practical methods for generating web content
- Primarily written in the Perl programming language
- Unfortunately, traditional CGI programs suffer from scalability and performance problems
- Let's examine these two problems…

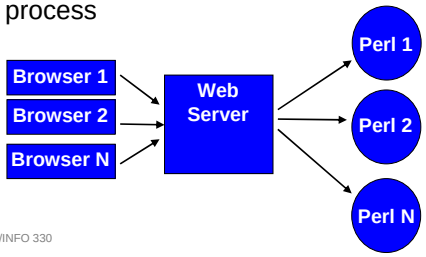
CGI Architecture

1) Browser initiates request
2) Web server receives request
3) For each request, web server spawns a new operating system process to execute the CGI/Perl program



CGI Architecture

- For each browser request, web server must spawn a new operating system process



CGI Architecture

- Spawning new operating system process for each request takes time and memory
  - Inherent performance and scalability problems (for traditional CGI)
- Every other server architecture tries to address these problems

Option 2: Fast CGI

- Option for developing faster, more scalable CGI programs
- Works by creating a pool of processes for handling CGI requests
- When a CGI request comes in, Fast CGI picks one of the processes from the pool and assigns it to the task
- Without the overhead of creating new operating system processes, FastCGI is much faster than traditional CGI

Option 3: Mod_Perl

- Module for the Apache Web Server (most popular web server on the planet)
- Embeds the Perl interpreter directly within the web server
- Perl programs are precompiled
  - No need to re-launch Perl interpreter for each request
- Because Perl is embedded within the Server, Mod_Perl does not need to create a new process for each request
- Much faster than traditional CGI

Option 4: Server Extensions

- Several web servers provide extension APIs
  - Netscape: NSAPI
  - Microsoft: ISAPI
- Much like Mod_Perl, these programs run directly within the web server
  - Hence much faster than traditional CGI
- Usually written in C/C++, not portable across web servers
  - If you develop to Netscape NSAPI, you cannot run it on ISAPI

Option 5: ASP.NET

- Active Server Pages
- Runs on Microsoft's Web Server: Internet Information Server (IIS)
- Programmers add ASP code directly into their HTML pages (like JSP)
- When client requests a page, web server takes the HTML page, runs the ASP code within the page, and returns a complete HTML page
- Faster than traditional CGI, but only works on Microsoft IIS

Option 6: Cold Fusion

- Developed by Allaire Corporation, now owned by Adobe
- Provides excellent database access and database tools
- Great platform for rapid prototyping and rapid development
- Expensive…

Option 7: PHP

- "Personal Home Page" tool or "PHP: Hypertext Preprocessor"
- Open source project written entirely by volunteers
- Provides simple, but powerful database access
- Also great for rapid development
- Very popular

Option 8: Ruby on Rails

- Open source project written in Ruby programming language
- Fundamental principles
  - Convention over Configuration
    - Only need to specify unconventional aspects, e.g., default naming convention for classes and database tables
  - Don't repeat yourself
    - Information located in single unambiguous place, e.g., no need to specify database column names in class definitions—retrieved from database
- Simplifies web development
  - Out of the box "scaffolding" for interaction with databases
  - Extensive use of JavaScript libraries for AJAX

Advantages of Servlets

- Servlets have six main advantages:
  - Efficient
  - Convenient
  - Powerful
  - Portable
  - Secure
  - Inexpensive

Advantage 1: Efficient

- For each browser request, Servlet spawns a lightweight thread
  - Faster and more efficient than spawning a new operating system process
  - Results in better performance and better scalability than traditional CGI

Advantage 2: Convenient

- Servlets include built-in functionality for:
  - Reading HTML form data
  - Handling cookies
  - Tracking user sessions
  - Setting HTTP headers
- Java is object oriented

Advantage 3: Powerful

- Servlets can talk directly to the web servers
- Multiple servlets can share data
  - Particularly important for maintaining database connections
- Includes powerful techniques for tracking user sessions

Advantage 4: Portable

- Java advantage: portability across different operating systems
  - Servlets have the same advantages
- Write Servlets on Windows, then deploy them on UNIX
- Can run servlets on any Java-enabled web server, with no code changes

Advantage 5: Secure

- Traditional CGI programs have a number of known security vulnerabilities
  - Hence, need to include a separate Perl/CGI module to supply the necessary security protection
- Java has a number of built-in security layers
- Java servlets are considered more secure than traditional CGI programs

Advantage 6: Inexpensive

- You can download free Servlet kits for development use
  - Can get started for free!
- Nonetheless, production quality Servlet web servers can get quite expensive
  - But there are good free ones
- All Java EE tools used for this course can be downloaded for free…

Application State

- Recall: HTTP is stateless
- Server-side state
  - Information is stored in a database, or in the application layer's local memory
- Client-side state
  - Information is stored on the client's computer in the form of a cookie
- Hidden state
  - Information is hidden within dynamically created web pages

Server-Side State

- Many types of Server side state:
  1. Store information in a database
     - Data will be safe in the database
     - BUT: requires a database access to query or update the information
  2. Use application layer's local memory
     - Can map user's IP address to some state
     - BUT: this information is volatile and takes up lots of server main memory

Server-Side State

- Should use server-side state maintenance for information that needs to persist
  - Old customer orders
  - "Click trails" of a user's movement through a site
  - Permanent choices a user makes