

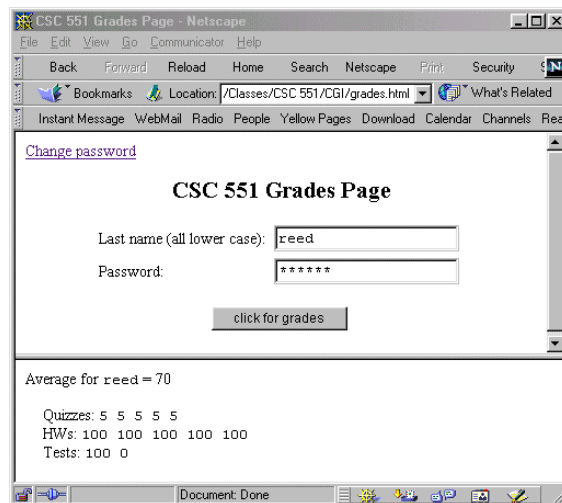
CSC 551: Web Programming

Fall 2001

More CGI & related server-side technologies

- more CGI examples
 - online grades
 - NCAA tournament
 - server-side includes
- new technologies & tools
 - server-specific technologies: ONE vs. WindowsDNA
 - server-independent technologies: Java servlets
 - Web development tools: Cold Fusion vs. WebObjects

Online grades access



want to:

- allow students to access grades
- hide password from view
- allow student to change password

requires server-side:

- must store grades & passwords on server
- allow access based on ID & password

Grades interface

```
<html>
<head>
  <title>CSC 551 Grades Page</title>
</head>

<body>
<a href="password.html" target="_blank">Change password</a><br>
<center>
<h2>CSC 551 Grades Page</h2>

<form action="http://duck.creighton.edu/cgi-bin/student.cgi" method="post" target="output">
<p>
<table border=0>
  <tr><td>Last name (all lower case): <td><input type="text" name="login">
  <tr><td>Password: <td><input type="password" name="passwd">
</table>
<p>
<input type="submit" value="click for grades">
</form>

</center>
</body>
</html>
```

in input frame, allow user to enter login ID and password

submit using POST

direct response to output frame

Changing the password

to change the password, must be able to rewrite the file

- pass login ID, old & new passwords to CGI program
- as the program reads from "grades.txt", echo data to "temp.txt"
- when find matching login ID and password, substitute new password
- if password was changed, then copy "temp.txt" back to "grades.txt"

note: CGI file access presents serious security problems



password program

```
...
int main()
{
    // CODE FOR READING CGI INPUTS, OUTPUTTING HTTP HEADER

    if (newPasswd1 != newPasswd2) {
        cout << "INPUT ERROR: you must enter the new password twice. <br>"
             << "<font color='red'><blink>PASSWORD NOT UPDATED</blink></font>" << endl;
    }
    else {
        ifstream ifstr("grades.txt");
        ofstream ofstr("temp.txt");

        string fileLogin, filePasswd, fileQuizzes, fileHWS, fileTests;
        bool found = false;
        while (ifstr >> fileLogin) {
            ifstr >> filePasswd;
            getline(ifstr, fileQuizzes); getline(ifstr, fileQuizzes);
            getline(ifstr, fileHWS);      getline(ifstr, fileTests);

            if (!found && fileLogin == login && filePasswd == oldPasswd) {
                ofstr << fileLogin << " " << newPasswd1 << endl << fileQuizzes << endl
                    << fileHWS << endl << fileTests << endl;
                cout << "Password for " << login << " was successfully updated. <br>" << endl;
                found = true;
            }
            else {
                ofstr << fileLogin << " " << filePasswd << endl << fileQuizzes << endl
                    << fileHWS << endl << fileTests << endl;
            }
        }
        ifstr.close();
        ofstr.close();
    }
    ...
}
```

password (cont.)

```
...
if (!found) {
    cout << "INPUT ERROR: no match for login ID " << login << ". <br>"
         << "<font color='red'><blink>PASSWORD NOT UPDATED</blink></font>"
         << endl;
}
else {
    ifstream newIfstr("temp.txt");
    ofstream newOfstr("grades.txt");

    string line;
    while (getline(newIfstr, line)) {
        newOfstr << line << endl;
    }
    newIfstr.close();
    newOfstr.close();
}

cout << "</body></html>" << endl;

return 0;
}
```

Password interface

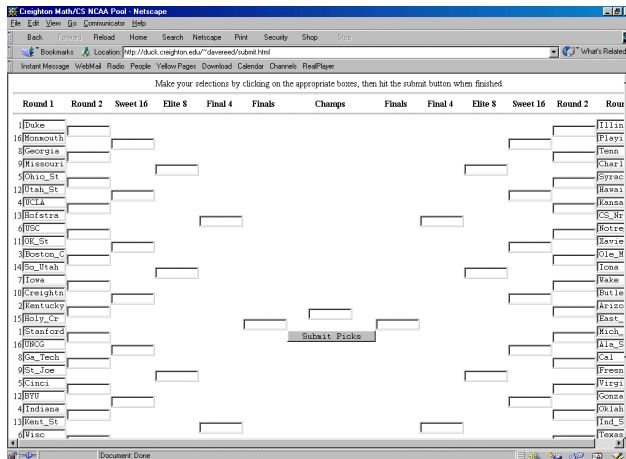
in separate window, allow user to enter login ID, old and new passwords

submit using POST

```
<html>
<head>
  <title>CS 551 Password Change</title>
</head>

<body>
  <form action="http://duck.creighton.edu/cgi-bin/passwd.cgi" method="post">
  <center>
  <p>
  <table>
    <tr><th align="middle" colspan=2>Change your password
    <tr><td colspan=2><HR>
    <tr><td align="right">login: <td><input type="text" name="login">
    <tr><td align="right">old password: <td><input type="password" name="old">
    <tr><td align="right">new password: <td><input type="password" name="new1">
    <tr><td align="right">new password (again): <td><input type="password" name="new2">
  </table>
  <p>
  <input type="submit" value="click to change password">
  </center>
  </form>
</body>
</html>
```

NCAA tourney example



text boxes are arranged in a table

user selects teams by clicking on text boxes: ONCLICK event-handler copies box contents to next round box

when user clicks submit button, prompted for name (stored as hidden), then all data is sent to CGI program

NCAA implementation

CGI program parses input and saves in a file

- gets submitter's name, stores in a file by that name

player=Dave+Reed → Entries/Dave_Reed

- if a file with that name already exists, disallow entry
- could have CGI program generate HTML page for each entry

the generation of individual HTML pages & scoring is handled by UNIX scripts

```
...
int main()
{
    CGIinput cgi;

    string name = cgi.ElementValue("player");
    for (int j = 0; j < name.length(); j++) {
        if (name[j] == ' ') {
            name[j] = '_';
        }
    }
    string fname = "Entries/" + name;

    cout << "Content-Type: text/html" << endl << endl; // PRINT HEADER
    cout << "<html><head><title>NCAA Pool Submission</title></head><body>" << endl;

    ifstream ifstr(fname.c_str());
    if (ifstr) {
        ifstr.close();
        cout << "Submission failed: there is already a submission under that name." << endl;
    }
    else {
        ifstr.close();
        ofstream ofstr(fname.c_str());
        for (int i = 0; i < cgi.NumElements(); i++) {
            ofstr << cgi.Element(i) << " " << cgi.Value(i) << endl;
        }
        ofstr.close();
        cout << "Submission accepted. Good luck" << endl;
    }
    cout << "</body></html>" << endl;
    return 0;
}
```

NCAA CGI program

Server-side includes

CGI programs can be called

- directly using the full URL (as in hello, fortune)
- via form submission (as in helloEcho, emailDB, grades, NCAA)
- using embedded server-side includes

```
<!-- #exec cgi="URL" -->
```

must store page with extension .shtml:

tells server to scan for #exec command, calls CGI program, replaces with output
Web server must be configured to allow server-side includes

```
<html>
<!-- fortune.shtml -->

<head>
<title>embedded CGI call</title>
</head>

<body>
<table border=1 align="center">
<tr><td>
<!-- #exec cgi="/cgi-bin/fortune.cgi" -->
</td>
</tr>
</table>
</body>
</html>
```

Server-specific technologies

Netscape and Microsoft both have established platforms for Web development and Web programming

- Netscape ONE
 - group of technologies packaged for *crossware support*, including
 - Enterprise server, Netscape Communicator
 - Java and JavaScript client-side programming
 - Internet Foundation Classes, now part of Java Foundation Classes
 - LiveWire (server-side JavaScript) for server-side programming
 - Dynamic HTML with Cascading Style Sheets
 - component communication using CORBA & JavaBeans model
- Microsoft DNA (Distributed interNet Applications architecture)
 - group of Microsoft-specific technologies, including
 - Internet Information Server (IIS), Internet Explorer
 - Java, JScript & Visual Basic for client-side programming
 - Application Framework Classes
 - Active Server Pages for server-side programming
 - Dynamic HTML with Cascading Style Sheets
 - component communication using COM & ActiveX

Server-side JavaScript via Netscape's LiveWire

can place JavaScript code within `<SERVER></SERVER>` tags

- will be processed by the Web server before downloading to client
- can be any JavaScript code, but no interactive routines

```
<html>
<head>
  <title>Server-side Fortune</title>
</head>

<server>
  list = ["Live long and prosper",
         "Save for tomorrow",
         "You will meet someone"];
  fortune = list[Math.floor(Math.random()*list.length)];
</server>

<body>
<table border=1 align="center">
<tr><td>
  <server> write(fortune); </server>
</td>
</tr>
</table>
</body>
</html>
```

to access the page

- must compile the page using the Netscape compiler
fortune.html → fortune.web
- must add the application to the Web server Application Manager

Server-side scripts via Microsoft's ASP

unlike server-side JavaScript, Active Server Pages do not require any special compilation or installation

- .asp file extension tells server to process scripts before downloading page

```
<%@ language=javascript %>
<html>
<head>
  <title>Server-side Fortune</title>
</head>

<%
  list = ["Live long and prosper",
         "Save for tomorrow",
         "You will meet someone"];
  fortune = list[Math.floor(Math.random()*list.length)];
%>

<body>
<table border=1 align="center">
<tr><td>
  <% = fortune %>
</td>
</tr>
</table>
</body>
</html>
```

must specify JavaScript as the scripting language (VBScript is the default)

code is embedded in `<% %>` tag

script values can be accessed using `<% = %>`

Java servlets

servlets are the server-side counterparts of applets

alternative to CGI

- servlet API defines input/output behavior, similar to CGI
- unlike CGI, servlet is loaded and executed as part of the Web server doesn't need to spawn a new process, so more efficient
- unlike LiveWire & ASP, servlets can be used with different servers

in order to execute, must

- compile servlet (e.g., with Java Servlet Development Kit)
- place in special servlet directory, similar to cgi-bin
- unlike CGI, servlet is loaded & initialized only once
each subsequent call spawns a new thread under the same process
note: all threads share same memory space, raises synchronization issues

HelloWorld servlet

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

public class HelloWorld extends HttpServlet {
    public void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        String userName = request.getParameter("yourName");

        response.setContentType("text/html");

        ServletOutputStream out = response.getOutputStream();
        out.println("<html>");
        out.println("<head><title>Hello World</title></head>");
        out.println("<body>");
        out.println("<h1>Hello " + userName + "!</h1>");
        out.println("</body></html>");
    }

    public String getServletInfo()
    {
        return "A HelloWorld Servlet";
    }
}
```

servlet communicates with the browser via request & response objects

service method handles GET & POST requests

getParameter method of request object extracts a field value

Web development tools

Web publishing tools such as Microsoft FrontPage & Adobe PageMill aid in the development of HTML documents with hooks for plug-ins & simple code

Server-side Web development tools such as Cold Fusion & WebObjects also facilitate software engineering

- integration with database systems
- hooks for including code developed using CGI, Java, JavaScript, ...
- supports code reuse via file inclusion and remote procedure calls

- Cold Fusion: supported on Windows NT/95 & Solaris extends HTML tags with some of the functionality of CGI e.g., routines for form validation are predefined

```
<input type="text" name="yourName"
      required="Yes" message="Please enter your name">
```

- WebObjects: high-end development tool, supported on more platforms provides drag-and-drop GUI interface for creating pages object-oriented architecture for integrating software technology

Next week...

Emerging and alternate Web technologies

- Dynamic HTML
- ActiveX
- XML

Course overview

read Chapters 33 & 34

as always, be prepared for a quiz on

- today's lecture (moderately thorough)
- the reading (superficial)