Web-Based Information Systems

Fall 2004

CMPUT 410: CGI and HTML Forms

Dr. Osmar R. Zaïane

University of Alberta

Outline of Lecture 5

- Introduction
- Poor Man's Animation
- Animation with Java
- Animation with JavaScript
- Sound
- Animation with DHTML

Objectives

•Learn about old and new techniques that allow animation in web pages.

•See some more concepts related to web technologies, server-side and client-side.

Course Content

• Introduction	• Perl & Cookies
• Internet and WWW	• SGML / XML
• Protocols	• CORBA & SOAP
• HTML and beyond	• Web Services
• Animation & WWW	• Search Engines
• CGI & HTML Forms	• Recommender Syst.
• Javascript	• Web Mining
• Databases & WWW	• Security Issues
• Dynamic Pages	• Selected Topics

Web-based Applications

Objectives of Lecture 6

CGI and HTML Forms

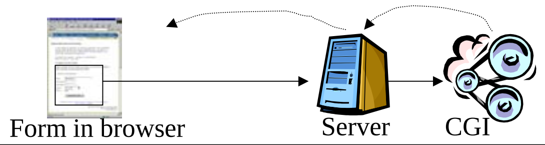
- Learn about another aspect of HTML: building forms to input data.
- Introduce the concept of CGI.

Outline of Lecture 6

- HTML Forms
- CGI programming

Using HTML Forms

- A Web HTML page can gather input and send it to a program on the server for processing.
- The program that receives the input for processing is called a CGI (or Common Gateway Interface).
- A CGI program has a URL like any Web accessible file.
- A Form has a link to the corresponding CGI.

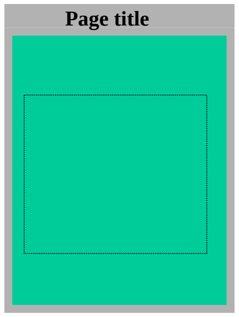


HTML Forms Structure

<FORM ACTION="*URL_CGI*" METHOD="…" …>

…

</FORM>



ACTION specifies the URL of a CGI program or e-mail address (mailto:e-mail@address.there)

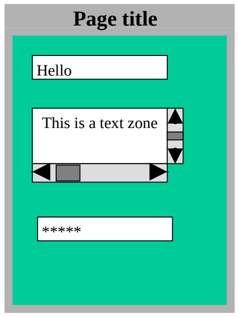
METHOD specifies how the data is transmitted to the server
"GET" : the data is sent appended to the URL
"POST": the data is sent after the HTTP header

ENCTYPE specifies the way in which the data is encoded

You cannot have nested forms. (No forms in forms allowed)

Form Input Elements

Textfields
<INPUT TYPE="TEXT" NAME="var1" **VALUE="Hello" SIZE="20">**
MAXLENGTH would specify the maximum input length



Text areas
<TEXTAREA NAME="var2" COLS="20" ROWS="3">
This is a text zone
</TEXTAREA>
WRAP can either be OFF, HARD or SOFT and specifies how the word wrap is done at the end of line

Password fields
<INPUT TYPE="PASSWORD" NAME="var3">
SIZE
VALUE
MAXLENGTH

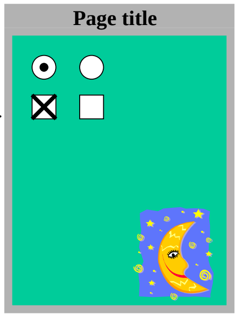
Form Input Elements

Radio buttons (shared names)
<INPUT TYPE="RADIO" NAME="var4" **VALUE="**yes" **CHECKED>**
<INPUT TYPE="RADIO" NAME="var4" **VALUE="**no"**>**

Checkboxes
<INPUT TYPE="CHECKLIST" name="var5" **CHECKED>**
<INPUT TYPE="CHECKLIST" name="var6"**>**

Hidden variables
<INPUT TYPE="HIDDEN" NAME="var7" **VALUE=**23**>**

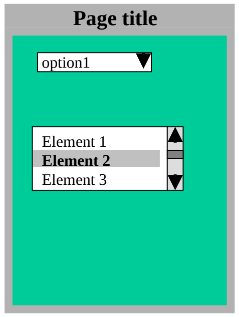
Image maps (server-side)
<INPUT TYPE="IMAGE" NAME="var8" SRC="image.gif" ALIGN="RIGHT">
Clicking on the image would submit the form along with two variables: name.x and name.y (in this case var8.x and var8.y)



Form Input Elements

Combo boxes and List boxes
<SELECT NAME="var9"**>**
  <OPTION VALUE="A"> option 1**</OPTION>**
  <OPTION VALUE="B"> option 2**</OPTION>**
</SELECT>

<SELECT NAME="var10" **SIZE="**3" **MULTIPLE>**
  <OPTION>Element 1**</OPTION>**
  <OPTION SELECTED>Element 2**</OPTION>**
  <OPTION>Element 3**</OPTION>**
  <OPTION>Element 4**</OPTION>**
</SELECT>



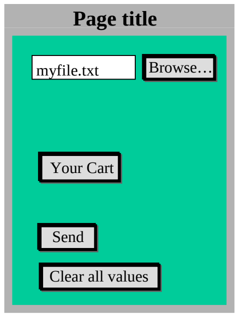
Form Input Elements

Attached file
<INPUT TYPE="FILE" NAME="var11" VALUE="myfile.txt"**>**
  SIZE, MAXLENGTH used as for textfields
  ACCEPT restricts the files to certain MIME types

JavaScript button
<INPUT TYPE="BUTTON" NAME="var12"
 VALUE="Your Cart">
  no submission but can be attached to a JavaScript

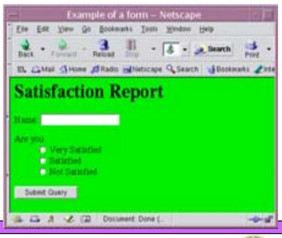
Submit button
<INPUT TYPE="SUBMIT" VALUE="Send">
  submits the form (pairs of variable-value)

Reset button
<INPUT TYPE="RESET" VALUE="Clear all values">
  Resets all variables in the form to their original values



Example of a Form

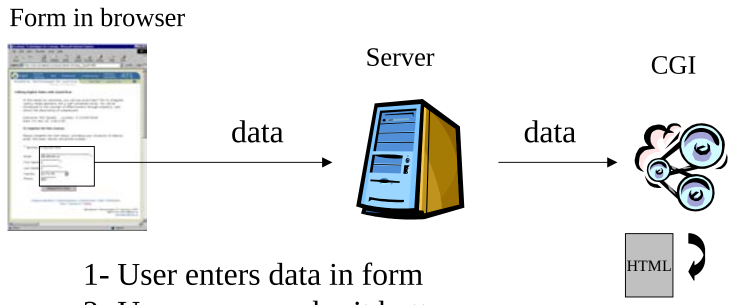
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD><TITLE>Example of a form**</TITLE></HEAD>**
  <BODY BGCOLOR="#00FF00">
  <H1> Satisfaction Report</H1>
  <FORM ACTION ="mailto:me@university.ca" METHOD=POST>
    Name: <INPUT TYPE=text name="name"**><BR>**
    <DL><DT>Are you
        <DD><INPUT TYPE=radio name="satisf" **Value="**1"**>** Very Satisfied
        <DD><INPUT TYPE=radio name="satisf" **Value="**2"**>** Satisfied
        <DD><INPUT TYPE=radio name="satisf" **Value="**3"**>** Not Satisfied
    </DL>
    <INPUT TYPE=submit>
  </FORM>
  </BODY>
</HTML>



Outline of Lecture 6

- HTML Forms
- CGI programming

Common Gateway Interface



1- User enters data in form
2- User presses submit button
3- data sent to web server with URL of CGI
4- Server starts a CGI and passes data
5- CGI processes data and generates HTML page

CGI Programming

- Any programming language that allows reading Standard Input and writing to the Standard Output can be used for CGI programming.
- Perl is commonly used for CGI, but also C/C^{++}, Python, Unix shell script, AppleScript, Visual Basic, Java, etc.

The CGI Interaction Process

- There are 4 basic steps in a CGI program:
  1. Read the data (input parameters)
  2. Process the data
  3. Output an HTTP response header
  4. Generate a document
- The CGI should send a blank line to separate the HTTP header from the generated document.
- Reading the data is different depending upon the method used to send the data (GET, POST)

Reading the GET Data

- With the GET method, data is sent with the CGI URL: ***myprogram.cgi?var1=abc&var2=123***
- Data is appended to the URL with "?"
- Variable-value pairs are separated by "&"
- A Variable and a Value are separated by "="
- When a web server receives a GET with a CGI URL it puts the data in an environment variable **QUERY_STRING** before calling the CGI program.

Java and QUERY_STRING

- QUERY_STRING="***var1=abc&var2=123***"
- Java has no method to directly read environment variables (the concept doesn't exist with all OS).
- We need to use an intermediary script to pass the variable along to Java

Example with Unix shell script

```
#!/bin/sh
/usr/local/JDK/bin/java myJavaCgi "$QUERY_STRING"
```

Reading the POST Data

- No data is attached to the CGI URL
- The data is sent like a document after an HTTP request header (in a single line).
- The header would contain information about the data such as *Content-Length*, etc
- The data is available to the program as standard input.

Pros and Cons of Get and Post

- Since data is appended to the URL with GET, the size of data is limited by the browser's URL maximum size (truncated).
- There is no size limit for data sent with POST.
- We can activate a CGI without using an HTML form if we use GET:

  http://server/path/cgiprogram.cgi?parameters
- POST can be used to send private information.

CGI Environment Variables

- In Addition to QUERY_STRING, there are many standard environment variables available to CGI programs:
- CONTENT_LENGTH
- CONTENT_TYPE
- HTTP_COOKIE
- HTTP_REFERER
- HTTP_USER_AGENT
- REMOTE_ADDR and REMOTE_HOST
- REMOTE_USER
- REQUEST_METHOD
- SCRIPT_NAME
- SERVER_NAME
- SERVER_PORT
- SERVER_PROTOCOL
- etc.

Example of parsing CGI input with Perl

```
#-----------------------------------------------------------------------
#######       Getting the input from STDIN or command line
#-----------------------------------------------------------------------
$my_input =   ($ENV{REQUEST_METHOD} eq "POST") ?
        <STDIN> : $ENV{QUERY_STRING};
 #-----------------------------------------------------------------------
#######       Splitting input by parameter and value
#-----------------------------------------------------------------------
@my_QUERY_LIST = split( /&/, $my_input);          # Splitting all pairs
foreach $item (@my_QUERY_LIST) {
     ($my_param, $my_value) = split( /=/, $item);      # Splitting variables and values
     $my_value =~ s/\+/ /g;                            # Change +'s to spaces
     $my_value =~ s/\s*$//;                            # eliminate spaces at the end
     $my_value =~ s/\%0D\%0A/\n/g;
     $my_value =~ s/%(..)/pack('C',hex($1))/ge;
     if ($my_in{$my_param}) {
          $my_in{$my_param} .= ' ';
          $my_in{$my_param} .= $my_value;
      } else {
           $my_in{$my_param} = $my_value;
      }
}    $firstName=$my_in{"fname"}; $lastName=$my_in{"lname"}; # accessing parameters
```

Another example accessing parameters in a CGI with Perl

```
#-----------------------------------------------------------------------
#######       Accessing parameters using the CGI module
#             and importing standard functions.
#-----------------------------------------------------------------------
use CGI qw( :standard );

$dtd = " -//W3C//DTD HTML 4.01 Transitional//EN ";
print( header());
print( start_html({dtd=>$dtd, title => " My page title "}));
$firstName=param(" fname");
$lastName=param(" lname ");
…
```

Returns: `Content-type: text/html\n\n`

Deals with tags in the header `<HTML><HEAD><TITLE>` up to `<BODY>`

No need to build the data structure and do all the parsing