

CS 312

Internet Concepts

Developing Web Content: CGI Forms

Dr. Michele Weigle

Department of Computer Science

Old Dominion University

mweigle@cs.odu.edu

<http://www.cs.odu.edu/~mweigle/CS312-F10/>

1

Overview

- ◆ Specifying HTML Forms
- ◆ Communicating with Scripts using URLs
- ◆ CGI Server-Side Actions
- ◆ Writing CGI Scripts

2

HTML Forms

- ◆ Form elements are elements that allow the user to enter information
 - » text fields
 - » radio buttons
 - » checkboxes
 - » buttons
 - » drop-down menus
 - » textareas
- ◆ The definition and layout of a form is HTML, but CGI is needed to process the data provided to the forms.

3

What is CGI?

- ◆ Common Gateway Interface
- ◆ It's not a language, but a protocol
 - » common to refer to a program that uses CGI as "a CGI program"
- ◆ CGI program can be written in almost any programming language
 - » C, C++, Perl (most popular for CGI), Visual Basic
- ◆ Most typically used for processing form data and returning results in HTML format

4

Forms

- ◆ Defined with the <form> tag:

<form action="">

<label> <input /> </label>

...


<label> <input /> </label>

</form>

- ◆ action attribute is required by XHTML.
- ◆ User input fields are defined by the <input /> tag
 - » attributes: type (type of input), name (used for referencing)
- ◆ Text label associated with an input field is defined with the <label> tag

5

Form Example



<form action=" ../cgi-bin/maillist.pl" method="post">

<h2>Join our mailing list</h2>

<label>Name: <input type="text" name="realname" /> </label>

<label>E-mail: <input type="text" name="email" /></label>

<p><input type="submit" value="Submit" /></p>

</form>

6

Forms and Actions



Join our mailing list

Name:

E-mail:

- ◆ The main attribute of a form tag is *action*
 - » ex: `<form action="/cgi-bin/maillist.pl">`
 - » action is required by XHTML
 - ❖ if no action to be taken, then `action=""` will validate
- ◆ *action* tells the browser where to send the data for processing
- ◆ `input type="submit"` creates the submit button
 - » when pressed, the data is sent to the action defined

7

`<form action="../cgi-bin/maillist.pl" method="post">`

Forms and Methods

- ◆ If the method is *get*
 - » query string of the arguments is tacked onto the end of the URL (of action attribute)
 - ❖ `name=value`
 - ❖ `"?"` is separator between data-value pairs
 - » URL is sent to the web server
 - » should only be used when doing a search or requesting data
- ◆ If the method is *post*
 - » client sends the query string directly to the server, separately from the URL
 - » should be used when updating data on the server, for example, in a database

8

Form Processing

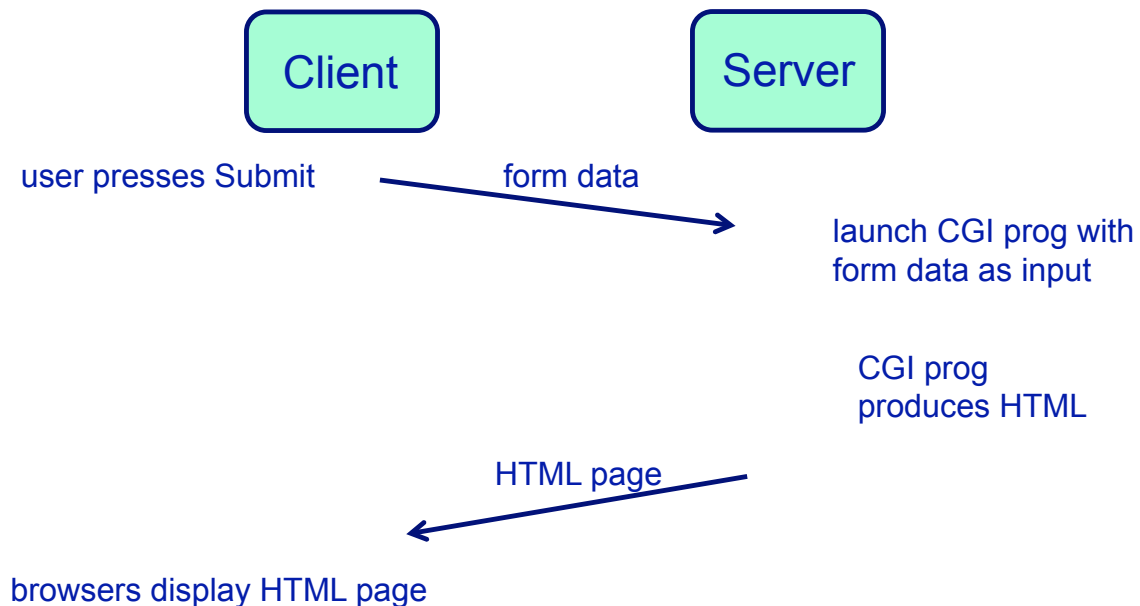
What Happens When Submit is Pressed?

- ◆ User presses "Submit" button
- ◆ Browser sends form data to web server
 - » specifically, to CGI program defined in <form action>
- ◆ Web server launches the CGI program
- ◆ CGI program executes taking the data from the form as input
- ◆ CGI program typically will generate a web page using HTML
- ◆ CGI program passes the HTML page back to the web server
- ◆ Web server passes the HTML page back to the browser

9

Form Processing

What Happens When Submit is Pressed?



10

Forms and CGI

- ◆ For now, we'll use the HTML Code Tutorial's (<http://www.htmlcodetutorial.com>) mycgi.pl script
- ◆ Displays name=value pairs that are sent to it
- ◆ We'll look at writing our own CGI programs later

11

Form Input Types

- | | |
|-----------------|----------------------------------------|
| ◆ Text | ◆ Non-Input Types |
| ◆ Submit Button | » select (scrolling or drop-down list) |
| ◆ Reset Button | » textarea |
| ◆ Password | |
| ◆ Radio Button | |
| ◆ Checkbox | |

12


Text Type

- ◆ A one-line text entry field

- `<input type="text" name="user" value="Donald Smith" size="30" />`

- ◆ Attributes:

- » `type`
 - » name of this parameter
 - » value (optional) – default input value
 - » size (optional) – field width
 - » maxlength (optional) – limit the number of characters the user can enter

A screenshot of a web browser showing a single-line text input field. The field is rectangular with a thin border and contains the text "Donald Smith". The background of the browser window is a light yellowish-green.

- ◆ When form is submitted, the information will be passed as

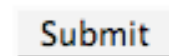
- » `user=Donald+Smith`

13

Submit Type

- ◆ A submit button

- » `<input type="submit" value="Submit" />`



- ◆ Value indicates the text that will be placed on the button

- » if nothing given, default is "Submit Query"

- ◆ When pressed, the form data is submitted to the script specified the form's *action* attribute

14

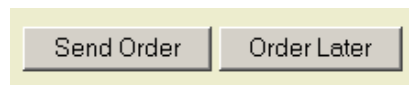
Submit Type

Multiple Submit Buttons

- ◆ Can have multiple submit buttons in the same form
- ◆ We can have multiple submit buttons, e.g., two labeled Send Order and Order Later, respectively.

```
<input type="submit" name="action" value="Send Order" />
```

```
<input type="submit" name="action" value="Order Later" />
```



15

Submit Type

Multiple Submit Buttons

- ◆ When multiple submit buttons are used in a single form, they should have the *same name* but *different values*
- ◆ Only one submit button can be clicked/effective.
 - » If the user clicked on the button labeled Send Order, then the corresponding part of the query string will be action=Send+Order

<http://www.cs.odu.edu/~mweigle/cs312/forms/form.html>

16

Image as Submit Button

- ◆ An image can also be used as a submit button, typically the image is some icon.

```
<input type="image" name="lion" src="../odulion.gif" />
```

- ◆ The type is *image*, not submit
- ◆ Must specify the source URL for src
- ◆ When the image is clicked, the corresponding part of the query string will be:
 - » `lion.x=xvalue&lion.y=yvalue`
- ◆ *xvalue*, *yvalue* are location in pixels where the mouse was clicked on the image
 - » a bit like image map

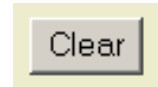


<http://www.cs.odu.edu/~mweigle/cs312/forms/form.html>

17

Reset Type

- ◆ A reset button
- ```
<input type="reset" value="Clear" />
```
- ◆ Value indicates the text that will be placed on the button
    - » if nothing given, default is "Reset"
  - ◆ When pressed, the all field data and selections in the form are reset back to their original, default values



<http://www.cs.odu.edu/~mweigle/cs312/forms/form.html>

18

## Hidden Field

---

- ◆ Used to pass some value, not given in any current input fields, to the called procedure.

`<input type="hidden" name="to" value="weigle" />`

- ◆ In the query string, this field and value pair are passed as

» `to=weigle`

- ◆ But, nothing is shown in the document text or form

<http://www.cs.odu.edu/~mweigle/cs312/forms/form.html>

19

## Password Type

---

- ◆ A one-line password entry field



`<input type="password" name="passwd" value="xyzy" size="10" />`

- ◆ All characters, default or user input, in the password field are shown as asterisks or dots.

- ◆ When form is submitted, the information will be passed as

» `passwd=xyzy`

» no encryption is performed (plain-text)

<http://www.cs.odu.edu/~mweigle/cs312/forms/form.html>

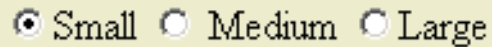
20

## Radio Button Type

---

- ◆ A group of radio buttons

- » Similar to checkboxes, but the user can select only one out of a group



☒ Small ☐ Medium ☐ Large

```
<input type="radio" name="size" value="small" checked="checked" />Small
```

```
<input type="radio" name="size" value="medium" /> Medium
```

```
<input type="radio" name="size" value="large" />Large
```

<http://www.cs.odu.edu/~mweigle/cs312/forms/form.html>

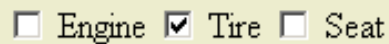
21

## Checkbox Type

---

- ◆ A group of checkboxes

- » Used to select multiple items.



☐ Engine ☒ Tire ☐ Seat

```
<input type="checkbox" name="items" value="engine" />Engine
```

```
<input type="checkbox" name="items" value="tire" checked="checked" /> Tire
```

```
<input type="checkbox" name="items" value="seat" /> Seat
```

<http://www.cs.odu.edu/~mweigle/cs312/forms/form.html>

22

## Button Type

---

- ◆ A push button
  - `<input type="button" value="Press Me!" />`
- ◆ Used to implement *client-side* scripts
  - » e.g., JavaScript
  - » *nothing is sent to the server*
- ◆ Example with simple JavaScript
  - `<input type="button" value="Click!"  
onclick="javascript:alert('Clicked!');" />`

<http://www.cs.odu.edu/~mweigle/cs312/forms/form.html>

23

## Select Tag

---

- ◆ A drop-down or scrolling list
  - `<select name="cars">` *drop-down*
  - `<select name="favorites" size="4" multiple="multiple">` *scrolling*
- ◆ Each option in the list is surrounded by `<option>...</option>` tags
  - » ex: `<option>jogging</option>`
  - » for default selection, use *selected* attribute on option tag
    - ❖ ex: `<option selected="selected">swimming</option>`



<http://www.cs.odu.edu/~mweigle/cs312/forms/form.html>

24

## Textarea Tag

---

- ◆ textarea tag, not an input tag
  - » For defining a large input text area, *not just a field of a single line*, use textarea tag.

```
<textarea name="longtext" rows="5" cols="60">
</textarea>
```



<http://www.cs.odu.edu/~mweigle/cs312/forms/form.html>

25

## Overview

---

- ◆ Specifying HTML Forms
- ◆ Communicating with Scripts using URLs
- ◆ CGI Server-Side Actions
- ◆ Writing CGI Scripts

26

## Communicating with Scripts Via URLs

- ◆ Scripts may or may not require arguments from users.
- ◆ The arguments are called a *query string* and may be appended at the end of a URL with the question mark "?" leading it.
- ◆ If argument has blank space
  - » use "+" or "%20"
- ◆ If there are two or more name/value pairs
  - » use "&" to delimit

27

## Communicating with Scripts Via URLs

### Examples

- ◆ One argument with parameter and value
  - » <http://www.google.com/search?q=titanic>
- ◆ Argument value has blank space
  - » <http://www.google.com/search?q=john+smith>
- ◆ Two or more parameters, using '&' to link pairs
  - » <http://finance.yahoo.com/q/bc?s=AAPL&t=2y>

28

# Overview

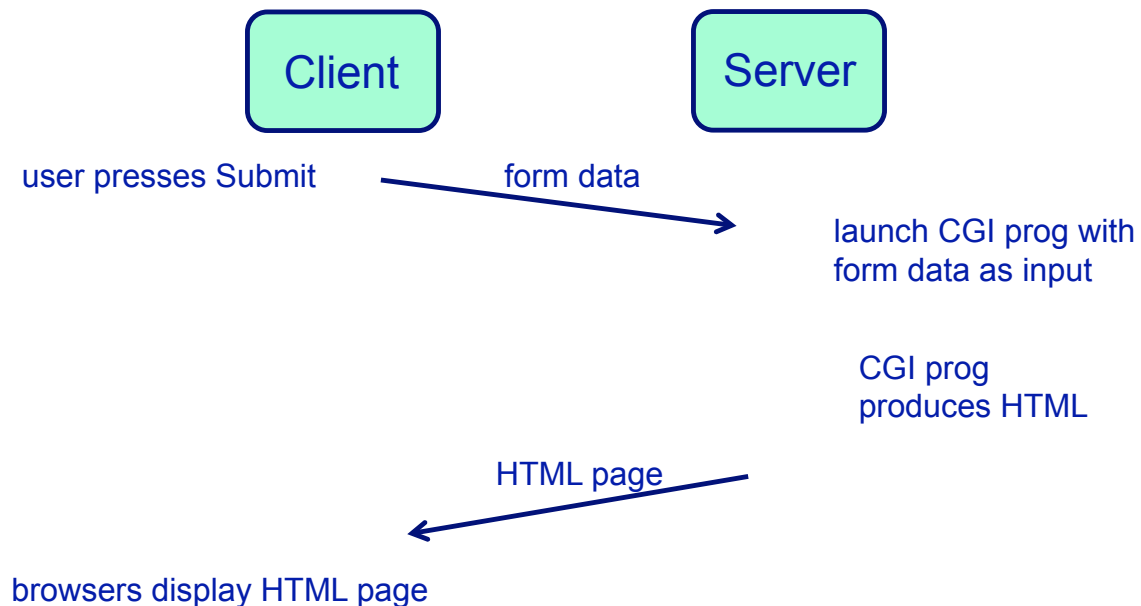
---

- ◆ Specifying HTML Forms
- ◆ Communicating with Scripts using URLs
- ◆ CGI Server-Side Actions
- ◆ Writing CGI Scripts

29

## CGI Server-Side Actions

---



30

# CGI Server-Side Actions

## Details

---

*When a web server receives a CGI request:*

- ◆ It creates a set of environment variables containing information about
  - » the server itself
  - » the remote browser
  - » the current request, including QUERY\_STRING
- ◆ It calls the corresponding script with any arguments in the environment variable QUERY\_STRING.

31

# CGI Server-Side Actions

## Details

---

- ◆ The script picks up any information it wants from the environment variables, particularly the arguments from QUERY\_STRING
  - » *i.e.* the parameters with corresponding values
  - » many programming languages provide tools for easy picking of parameter values by procedures
- ◆ The script then executes its own instructions
- ◆ The output by the script, typically a HTML page, is sent back to the client by the server

32



## Overview

---

- ◆ Specifying HTML Forms
- ◆ Communicating with Scripts using URLs
- ◆ CGI Server-Side Actions
- ◆ Writing CGI Scripts

33

## Writing CGI Scripts

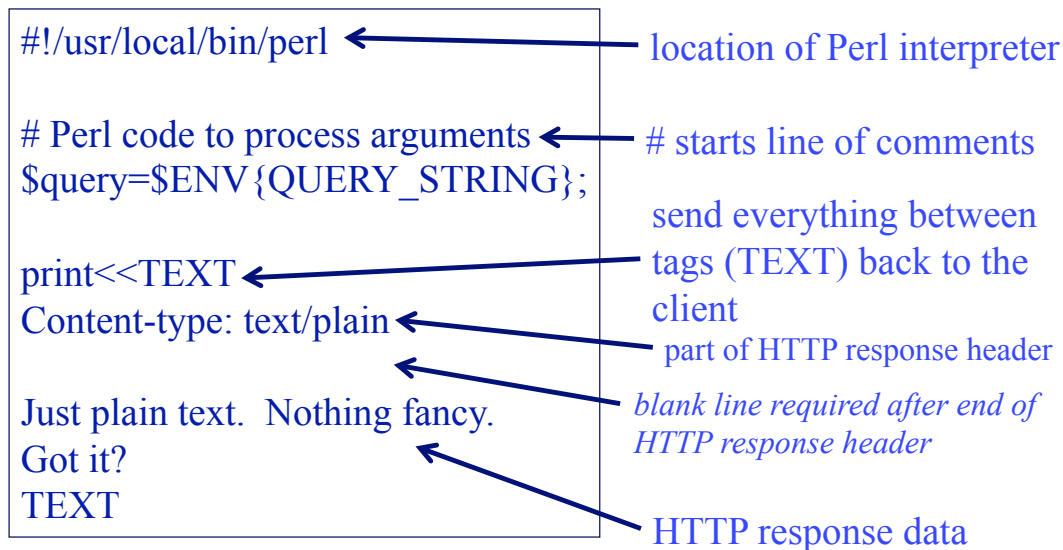
### ODU-CS CGI Implementation

---

- ◆ CGI scripts must be stored under `~/public_html`
- ◆ CGI scripts are best, but not required, to be stored under `~/public_html/cgi-bin/`
- ◆ The `cgi-bin` directory and all programs in the directory should have access mode `755` so that they can be executed by the web server.

34

## Layout of Perl CGI Script



35

## Simple Examples of Perl CGI Scripts

### ◆ Generating a plain text page, in perl on Unix

```
#!/usr/local/bin/perl

print<<PLAIN
Content-type: text/plain
```

```
Just plain text. Nothing fancy.
Got it?
PLAIN
```

*You must have the blank line in between the Content-type:text/plain (end of HTTP header) and the data part.*

<http://www.cs.odu.edu/~mweigle/cgi-bin/plainText.cgi>

36

## Simple Examples of Perl CGI Scripts

### ◆ The perl script mapCoord.cgi

```
#!/usr/local/bin/perl
```

```
$queryString = $ENV{QUERY_STRING};
```

```
print <<END;
```

```
Content-type: text/html
```

```
<html>
```

```
<p>The coordinates where the mouse was clicked were:</p>
```

```
<p>${queryString}</p>
```

```
</html>
```

```
END
```

<http://www.cs.odu.edu/~mweigle/cgi-bin/mapCoord.cgi>

*How can we use this to  
get coordinates clicked in  
an image map?*

37

## Determining Image Map Coordinates

### ◆ Use a form with image submit button

```
<form action="../cgi-bin/mapCoord.cgi" method="get">
```

```
<input type="image" name="coordinate" src="shapes.jpg" />
```

```
</form>
```

<http://www.cs.odu.edu/~mweigle/cs312/forms/finding-coord.html>

38

## Improving CGI efficiency

- ◆ CGI technology generally requires a fresh copy of the program to be executed for every CGI request
  - » The interpreter and the script may need to be reloaded each time
  - » The workload may overwhelm the web server when interpreting scripts is needed
- ◆ Integrating script interpreters directly into web servers
  - » `mod_perl` embeds Perl interpreter into the Apache server
- ◆ Caching compiled versions of the scripts in system location so that further requests for the file are automatically directed to the compiled code

39

## Other Efficient Approaches

- ◆ Active Server Pages, ASP
  - » A programming language, Microsoft's server-side technology for Internet Information Service, IIS
  - » An add-on to Internet Information Services (IIS)
  - » Using various built-in objects, each of which corresponds to a group of frequently-used functionality useful for creating dynamic web pages
  - » Can be mixed with HTML

40

## ASP Example

---

```
<html>
 Today's date is: <%response.write(date())%>.

 The server's local time is: <%response.write(time())%>.
</html>
```

*Would produce something like (does **not** work on our Apache):*

Today's date is: 15.03.2006.

The server's local time is: 10:17:18.

*The syntax is simply <% XXXXX %> where XXXXX is just the script language function calls.*

41

## Apache::ASP

---

- ◆ An Active Server Pages port to the Apache Web Server with Perl scripting only

- ◆ Apache::ASP syntax:

<%xxx%>

where xxx is any valid perl code.

Reference, Apache::ASP: <http://www.apache-asp.org/>

42

# PHP (Hypertext Preprocessor)

- ◆ An open-source, scripted programming language
- ◆ Allows interaction with a large number of relational databases
- ◆ Interacts with many major Web servers
- ◆ Can be embedded into HTML

Reference, a PHP tutorial:

<http://www.w3schools.com/php/default.asp>

43

## Simple PHP Example

- ◆ A HTML document calling a php script

```
<html>
<form action="1action.php" method="post">

<p>Give your name please: <input type="text" name="name" /> </p>

<p>Give your age please: <input type="text" name="age" /></p>

<p><input type="submit" /></p>
</form>
</html>
```

*ASP and PHP scripts can be called in forms.*

44

# PHP Script

---

We'll do more PHP later

- ◆ The php script (1action.php) called in the previous form

```
Hi <?php echo $_POST['name']; ?> ! <p>
You are <?php echo $_POST['age']; ?> years old.
```

*PHP can be mixed with HTML, just use: <?php XXXX ?>*

<http://www.cs.odu.edu/~mweigle/cs312/forms/php.html>

45

## Comparisons of ASP & PHP

---

- ◆ Both are languages used to build Dynamic Web sites that can interact with Databases and exchange information
- ◆ ASP programs require IIS on Windows, DB connection is to MS-SQL, both not free
- ◆ PHP programs run on Linux with Apache server, DB connection to MySQL, all free
- ◆ PHP also runs on many other platforms and can connect to many other databases, is faster than ASP, and has many free, open source software

46

# Overview

---

- ◆ Specifying HTML Forms
- ◆ Communicating with Scripts using URLs
- ◆ CGI Server-Side Actions
- ◆ Writing CGI Scripts