

CGI 2

Form -- Fields, Submit, Bindings

HTML form -- presents GUI text fields, lists, checkboxes, ...

The user can enter data and "submit" the form.

Submit takes up the state the user has put into the various fields, and sends it to a CGI on the server. The data is encoded as a series of name=value "bindings"

Bindings

A name=value syntax used to encode what the user has entered in the form.

Syntax: var1=value1&var2=value2&var3=value3

CGI Environment

Recall Environment variables -- combination of client and server info for the CGI too use.

QUERY_STRING-- holds the bindings with the GET method

Forms

An HTML form accepts user input (text fields, checkboxes...) on the client side, and sends the data to a CGI on the server side when the "submit" button is selected.

```
<form action = cgi-url method = get/post>...</form>
```

On submission, sends data to server script named in the action. If the action is omitted, sends data to the URL which produced the form itself.

Form Data Encoding

The data is coded as attribute/value pairs (bindings) with other characters "URL encoded": space changed to a '+', bindings separated by '&', and non alphanumeric characters encoded with the "%xx" form where xx is the hex code for the character.

```
person=Chuck+Jones&age=35&topping=anchovies
```

Each form element has a name which is used to identify it in the bindings such as "person" and "age" in the above example.

JavaScript can be used to make a form a little more reactive before the submit -- we may study JavaScript a little at the end of the quarter.

The MIME type for the above (+, =, &) encoding is application/x-www-form-urlencoded. Some browsers also support an ENCTYPE=multipart/form-data submission where the form is encoded like a MIME mail enclosure and then submitted by the Post method.

GET Method

GET method.

Attribute/value pairs attached to URL after the ?. This is the simple approach - ok for small amounts of data. Has the advantage that CGI operations can be bookmarked. Easier to debug, since you can see the data.

`http://blah.com/cgi-bin/foo.pl?person=Chuck+Jones&age=35`

POST Method

POST method.

The bindings are sent from the client to the server on the connection socket. The client sends the usual request stuff but with POST instead of GET, followed by a blank line, followed by the encoded data. The server, in turn, sends the data to the CGI on the CGI's standard input (instead of on the QUERY_STRING env var).

dumpform.pl

Returns to you the name=value bindings from a form submission -- use for debugging.

Accepts both GET and POST submissions

`http://www-cs-faculty.stanford.edu/cgi-bin/nick/dumpform.pl?a=b`

We'll look at the source code for dumpform.pl later

Input Fields -- Name and Type

Generally, each input field has a **name** and a **type**

The **type** is the sort of control

The **name** is how the data will be identified in the bindings.

type=text

```
<p>
<input type=text name=insult size = 40 value="This sucks">
name
```

`name=insult` defines the name used to return the data binding, not shown in client HI.

mapping

Maps to `insult=whatever-they-type-in-the-field` when sent to server.

size

Width in characters of the input box

maxLength

max length in characters of the entered string

`value="This sucks"`

to sets default text

type=password

similar but doesn't show characters on screen

type=checkbox

On-or-off

```
<input type=checkbox name=rude checked>Rude Mode
```

mapping

Maps to `rude=on` if checkbox is checked, otherwise there is no binding.

There's no automatic title around the checkbox, so you need to include your own title as HTML near the checkbox.

value=

If there's a value binding, that is used instead of the string "on"

type=radio

One-of

All have the same name and different values.

One can be checked

```
<p>
```

```
Insult Size:
```

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

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

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

mapping

Maps to `size=medium`

<select> Pop-Up

```
Color:<br>
```

```
<select name = "color">
```

```
<option>red
```

```
<option selected>blue
```

```
<option>green
```

```
<option>purple
```

```
<option>pink
```

```
</select>
```

<option>

Contains option tags followed by text

size=10

How large should the list UI appear (different from the number of items)

multiple

`<select multiple ...>` allows multiple selections which map to
`animal=Stoat&animal=Goat`

value=

Can have `value=` within an option, otherwise uses text as value

"selected" pre-selects one

Mapping

Maps to `color=blue`

<textarea>

Big text area for typing...

```
<textarea name = "comments" rows = rows cols = columns> ...</textarea>
<br><textarea name=comments rows=4 cols=60 wrap>
```

type=submit

Submit the data to the action url. If it is given a name (optional), the clicked button is given a binding in the submission — handy if you have multiple buttons and you need to know which was clicked.

```
<input type=submit name=sub value="Submit Insult Request">
```

Trick: if there's a single text field in the form, hitting return can also do a submit -- this feature varies among browsers.

Can multiple submit buttons, all with the same name, and distinguish which got clicked by the value. Or, could give each submit button a unique name.

type=image

Variant on the submit button -- add a src= def in the tag

Notice, it shows up with a light blue outline -- giving a visual cue that clicking it does something. Visual feedback is a key part of good UI.

Can add width=xxx or height=yyy to scale the image (as with a tag

Maps to name.x=27 name.y=67 in the bindings (it would be nice to also get a sort of name=1 binding, but that's not part of the spec)

```
<input type=image name= jeffbutton src=jeffsad.jpg>
```

type=hidden

```
<input type = "hidden" name = "secret" value ="blorg">
```

Maps to secret=blorg in data, but doesn't show in client HI — a good way to store information in the form for later reading by your CGI.

type=reset

Clear all fields in client HI

I have never in my life gotten any use out of this button

```
<input type = "reset" value = "button-title">
```

HTML Form Example

```
<html>
```

```
<head>
```

```
<title>Sample Form</title>
```

```
</head>
```

```
<body bgcolor=white>
```

```
<!-- form tag -->
```

```
<!-- action=url to send data, method=get or post -->
```

```
<form
```

```
  action=http://www-cs-faculty.stanford.edu/cgi-bin/nick/dumpform.pl
```

```
  method=get>
```

```

<h1>Insult Generator</h1>

<!-- text input -->
<!-- maps to person="Bob" -->
<p>
First name:
<input type=text name=person size = 40 value="Bob">

<!-- checkbox -->
<!-- maps to size=on if checked (not present if not checked) -->
<p>
<input type=checkbox name=rude>Rude mode

<!-- radio -->
<!-- maps to size=medium -->
<p>
Insult Size:
<br><input type=radio name=size value = "small">Small
<br><input type=radio name=size value = "medium" checked>Medium
<br><input type=radio name=size value = "large">Large

<!-- selection pop-up list -->
<!-- maps to animal=Goat -->
<p>
Preferred Animal:
<select name = "animal">
<option>Stoat
<option selected>Goat
<option>Weasel
</select>

<!-- selection scrollable list -->
<!-- maps to color=pink -->
<p>
Color:<br>
<select name = "color" size=8>
<option>red
<option selected>blue
<option>green
<option>purple
<option>pink
<option>yellow
<option>striped
<option>brown
<option>mauve
<option>dotted
<option>speckled
<option>puce
<option>orange
<option>tan
<option>black
<option>white

```

```
<option>gray
</select>

<!-- hidden field -->
<input type=hidden name=secretcode value="Keyser Soze">

<!-- text area -->
<p>
Comments:
<br><textarea name=comments rows=4 cols=60 wrap>
</textarea>

<!-- submit button -->
<!-- maps to sub="Submit Insult Request" if clicked -->
<p>
<input type=submit name=sub value="Submit Insult Request">

<!-- image submit -->
<!-- maps to im-submit.x=xxx for click location if clicked -->
<p>
<input type=image name=im-submit src=jeffsad.jpg width=100>

</form>

</body>
</html>
```

Netscape: Sample Form

Back Forward Reload Home Search Netscape Images Print Security Shop

Location: file:///Null/Classes/193i-01-3/CGI%20example/form.html

Insult Generator

First name:

Rude mode

Insult Size:

Small
 Medium
 Large


Preferred Animal:

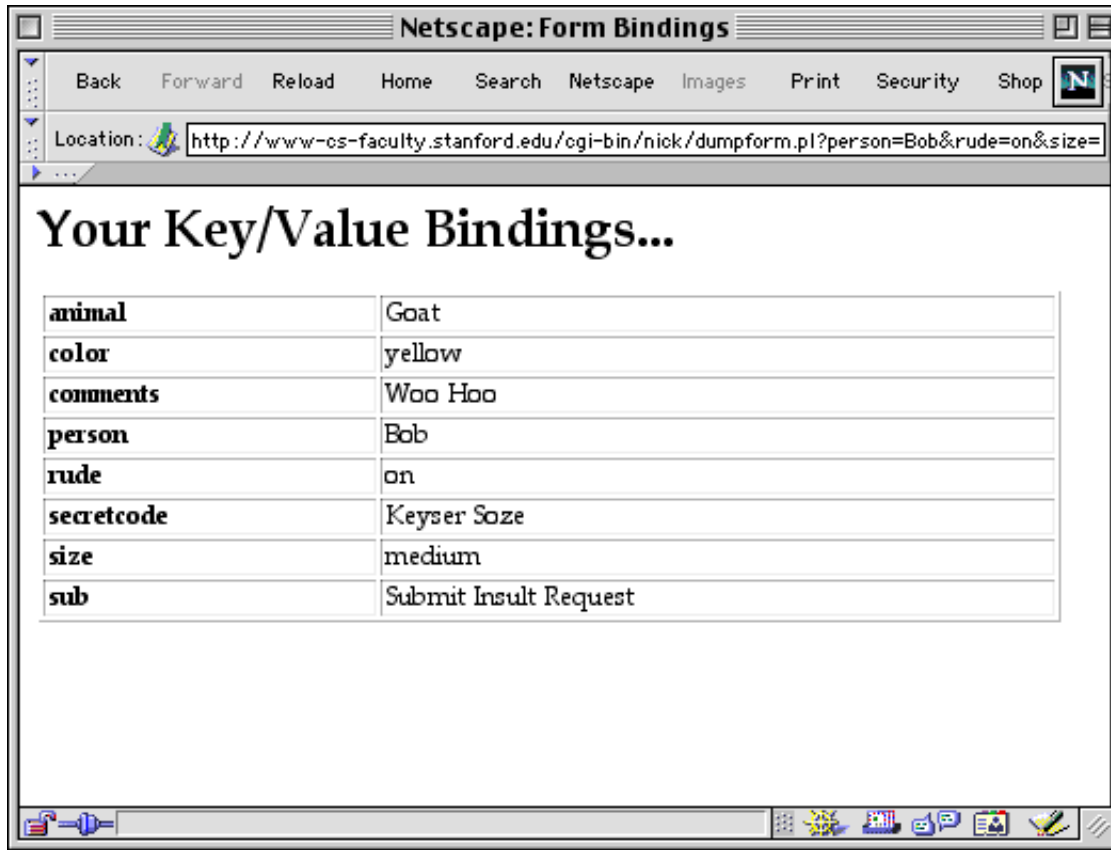
Color:

- red
- blue
- green
- purple
- pink
- yellow
- striped
- brown

Comments:

Woo Hoq





CGI.pm

It has lots of features, but for now we'll just use it to extract bindings. Modern Perl installations have CGI.pm installed already by default

use CGI;

\$q = new CGI;

\$param = \$q->param('param-name');

Will be undef if there is no such binding.

Use the defined() operator to check it

<<EOT; Perl Syntax

The following syntax puts the raw text into the \$string...

```
$string = <<EOT;
```

This here can be any old thing.

Including \$variable interpolation.

```
*** yeee haaa ***
```

```
EOT
```