Today's Topic

• Sending form data to the webserver
  1. GET method
  2. POST method

• There are also other methods to communicate information depending on the kind of webserver we run:
  – e.g. Apache Tomcat (for Java)
  – e.g. Websocket interface (for bidirectional data passing)
  – etc.
Sending information: GET

First: [input field] Last: [input field] Submit

• HTML form:
  1. `<form action="http://localhost/cgi-bin/get.cgi" method="GET">`
  2. First: `<input type="text" name="first" size=12>`
  3. Last: `<input type="text" name="last" size=12>`
  4. `<input type="submit">`
  5. `</form>`

http://localhost/cgi-bin/get.cgi?first=Sandiway&last=Fong

• Information encoded using alphanumeric characters: why?
• URLs are restricted to alphanumeric characters only
• **bash** accesses the URL-encoded string via the environment variable QUERY_STRING
Sending information: GET

- **get.cgi:**
  1. `#!/bin/bash`
  2. `echo "Content-Type: text/plain"`
  3. `echo`
  4. `#echo $QUERY_STRING`
  5. `origIFS=$IFS`
  6. `IFS='=&'`
  7. `set -- $QUERY_STRING`
  8. `IFS=$origIFS`
  9. `echo "1:<$1> 2:<$2> 3:<$3> 4:<$4>"`

http://localhost/cgi-bin/get.cgi?first=Sandiway&last=Fong

**In bash:**
- `IFS` = **internal field separator** (for arguments)
- default: space newline tab
- `set -- String`
- `--` option: positional parameters `$1`, `$2`,..etc. are set after splitting `String`
Sending information: GET

```
#!/bin/bash

echo "Content-Type: text/plain"

echo

# echo $QUERY_STRING

origIFS=$IFS

IFS='=&

set -- $QUERY_STRING

IFS=$origIFS

echo "1:<$1> 2:<$2> 3:<$3> 4:<$4>"
```

• **get.cgi:**

```bash
#!/bin/bash
echo "Content-Type: text/plain"
echo
# echo $QUERY_STRING
origIFS=$IFS
IFS='=&
set -- $QUERY_STRING
IFS=$origIFS
echo "1:<$1> 2:<$2> 3:<$3> 4:<$4>"
```
Sending information: GET

OS X
- /Library/WebServer/CGI-Executables/
- $ls -l get.cgi
- -rwxr-xr-x 1 root wheel 161 Oct 16 2014 get.cgi
- sudo chmod 755 get.cgi

Ubuntu
- /usr/lib/cgi-bin/
Sending information: POST

• HTML form:
  1. `<form action="http://localhost/cgi-bin/read.cgi" method="POST">`
  2. First: `<input type="text" name="first" size=12>`
  3. Last:  `<input type="text" name="last" size=12>`
  4. `<input type="submit">`
  5. `</form>`

• **bash** accesses the URL-encoded string on standard input via read
Sending information: POST

- **bash** accesses the URL-encoded string on standard input via `read`

- **read.cgi:**
  1. `#!/bin/bash`
  2. `echo "Content-Type: text/plain"
  3. `echo`
  4. `read input`
  5. `origIFS=$IFS`
  6. `IFS='=&'`
  7. `set -- $input`
  8. `IFS=$origIFS`
  9. `echo "<$2><$4>"`

**read**
Read a line from standard input
Syntax
```
read [-ers] [-a aname] [-p prompt] [-t timeout] [-n nchars] [-d delim] [name...]
```
Sending information: POST

```html
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head>
<title>CGI POST Example</title>
</head>
<body>
<h1>CGI POST Example</h1>
<form action="http://localhost/cgi-bin/read.cgi" method="POST">
  First: <input type="text" name="first" size=12>
  Last:  <input type="text" name="last" size=12>
  <input type="submit">
</form>
</body>
</html>
```
Limitations of positional parameters

- **GET values:**

  ```bash
  origIFS=$IFS
  IFS='=&'
  set -- $QUERY_STRING
  IFS=$origIFS
  echo "1:<$1> 2:<$2> 3:<$3> 4:<$4>"
  ```

A positional parameter is a parameter denoted by one or more digits, other than the single digit 0. Positional parameters are assigned from the shell’s arguments when it is invoked, and may be reassigned using the `set` builtin command. Positional parameter `n` may be referenced as `$n`, or as `$N` when `n` consists of a single digit. Positional parameters may not be assigned to with assignment statements. The `set` and `shift` builtins are used to set and unset
Another Example

• Suppose we want to maintain a list of names on the webserver stored in a file names.txt
  – MySQL would be a more sophisticated approach (hint!)
• Design a webpage to add names to this file and print out the table of names
Homework 9

• Use `cgi-bin` to implement a BMI bash shell script on the webserver
  – i.e. you should modify your BMI program to accept data from a html form and compute the value on the server side

• Use both GET and POST methods to communicate the weight, height and units.

• Submit your .html and .cgi files

• Submit screenshots to show it working in both cases