Web forms and CGI scripts

Dr. Andrew C.R. Martin
andrew.martin@ucl.ac.uk
http://www.bioinf.org.uk/

Aims and objectives

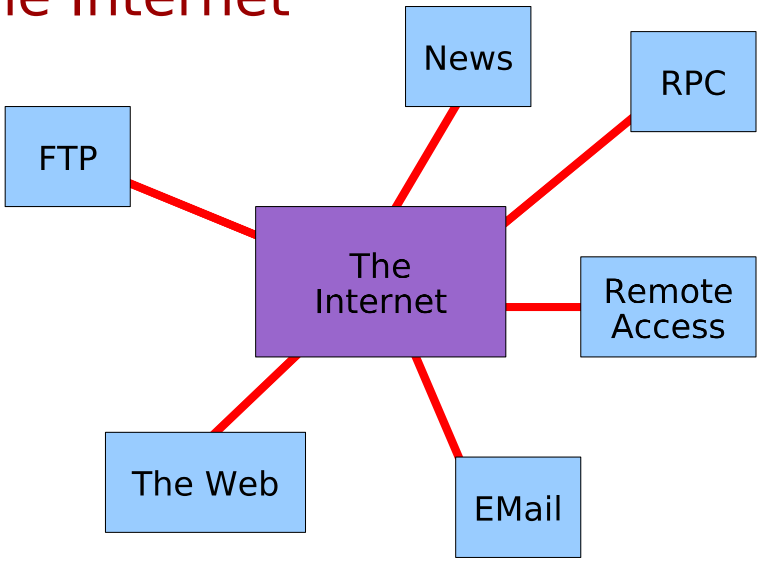
- Understand **how the web works**
- Be able to **create forms** on HTML pages
- Understand how CGI scripts can **create pages** and **obtain data** from forms
- Be able to **write a CGI script** to process data from a form

Internet and Web

Internet ≠ The Web

The Web is just one application of the Internet

The Internet



Names and addresses

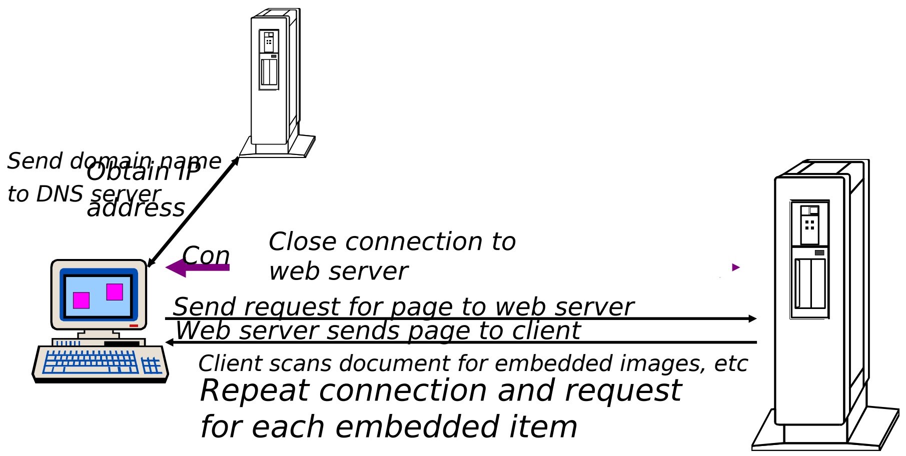
- Each computer has a unique **IP address**
  - e.g. **128.40.46.27**
- Numbers are **difficult to remember**!
- Hierarchical **domain name** scheme
  - e.g. **www.biochem.ucl.ac.uk**

- Names are mapped to numbers using **Domain Name Service** (**DNS**)

How does the web work?

- The World Wide Web was developed to share text via hyperlinks between documents on the same or different servers.

How does the web work?

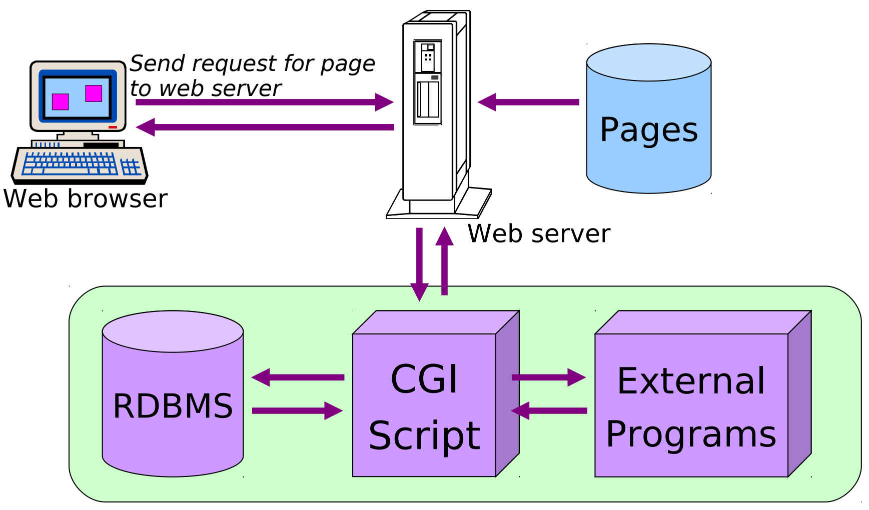

Send domain name
to DNS server
Obtain IP
address
Con
Close connection to
web server
Send request for page to web server
Web server sends page to client
Client scans document for embedded images, etc
Repeat connection and request
for each embedded item

Enter a URL

How does the web work?

- 'Forms' allow information to be sent **to the web server** for processing.
- The web server collects the information and dispatches it to a **CGI script** for processing.
- The program outputs information to **create a new web page** which is returned to the web server and thence to your web browser.

How does the web work?



CGI can extract parameters sent with the page request

What is CGI?

- **Common Gateway Interface**
  - the standard method for a web server to interact with external programs
- Can be implemented in any language
- Scripts (e.g. In Python or Perl) written to interact with a web server are often called **CGI scripts**
- The script's **standard output** is returned by the web server to your web browser

A simple CGI script

```
#!/usr/bin/python

print (“Content-Type: text/html\n”)

print ('''
<html>
<head>
   <title>Hello World!</title>
</head>
<body>
<h1>Hello World!</h1>
</body>
</html>
''')
```

What is CGI?

- First standard way to handle CGI from scripts was in Perl using the **CGI.pm** Perl module.
  - written by a Bioinformatician, Lincoln Stein
- Python equivalent is
  import cgi
- Makes it easy to interact with the web server

Creating a form

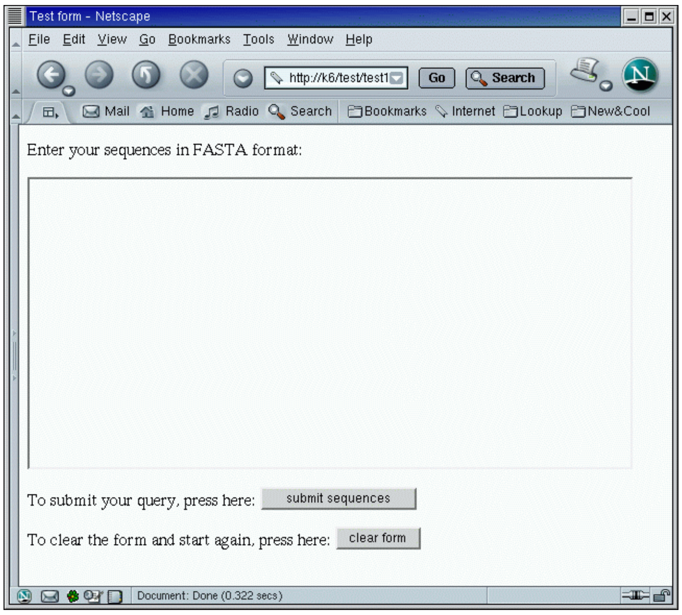
A simple form:

```
<form method="post" action="/cgi-bin/clustalw/clustalw.py">

<p>Enter your sequences in FASTA format:</p>
<p                s" r                          rea></p>
<p                   r
<i                alue
</

<p>To clear the form and start again, press here:
<input type="reset" value="clear form" />
</p>
</form>
```

How data are sent to the server

The CGI script to be run on the server

Buttons and text boxes


Test form - Netscape
File Edit View Go Bookmarks Tools Window Help
http://k6/test/test1
Go
Search
Mail Home Radio Search Bookmarks Internet Lookup New&Cool
Document: Done (0.322 secs)

Enter your sequences in FASTA format:

To submit your query, press here: submit sequences

To clear the form and start again, press here: clear form

CGI Scripts

```
<form method="post" action="/cgi-bin/clustalw/clustalw.py">
```

- Note the **location of the script**
  - the CGI script will reside on the same machine as the web page
  - can also use a full URL

Post and get

‘get’

- Used where small amounts of data are to be sent
- Data are sent as part of the URL

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

```
http://www.bioinf.org.uk/cgi-bin/foo.py?seqid=P00001&format=xml
```

Post and get

‘post’

- Used where larger amounts of data are to be sent
- Data sent separately from the URL

```
<form method="post" action="/cgi-bin/clustalw/clustalw.py">
```

Form elements

Submit and **reset**

```
<input type="submit"    value="submit sequences" />
<input type="reset"     value="clear form" />
```

Submit: submit form
reset: clear form

Text rendered
on buttons

Form elements

<textarea>

```
<textarea name="seqs" rows="20" cols="80"></textarea>
```

- A (large) box for text entry
- **rows=** and **cols=** attributes specify size of box
- **name=** gives a name for the CGI script to use for the data
- Must have a **</textarea>**
  - Any text in between will appear as default text in the box

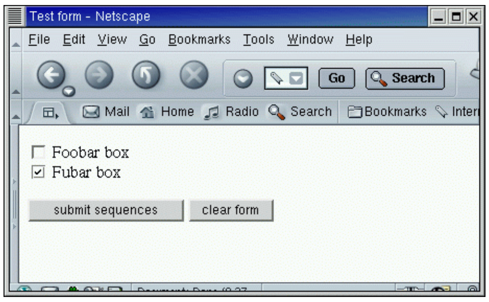
Form elements

Checkbox

```
<input type='checkbox' name='foo' value='bar' />
```

Creates a tick box

- If the checkbox is clicked, the name/value pair is sent to the server



☐ Foobar box
☑ Fubar box

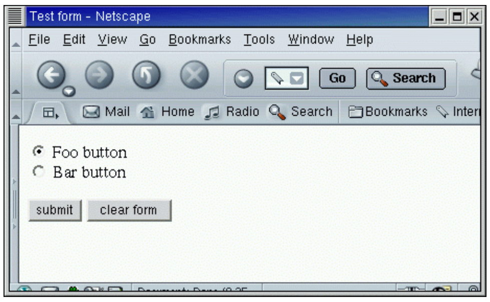
checked='1' pre-ticks a box

Form elements

Radio buttons

```
<input type='radio’ name=‘foo’ value=‘bar’ />
```

- Radio buttons are grouped by name
- One name/value pair sent to server


Test form - Netscape
File Edit View Go Bookmarks Tools Window Help
Go Search
Mail Home Radio Search Bookmarks Inter
Foo button
Bar button
submit clear form

checked=‘1’ pre-ticks a box

Form elements

Text boxes

```
<input type=‘text’ name=‘foo’ value=‘bar’
size=‘x’ maxlength=‘y’/>
```

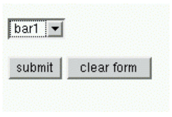
- Creates a one-line text box
- All attributes other than **name=** are optional

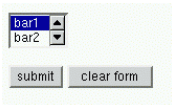
Form elements

Pull-down menus

```
<select name='foo' size='1'>
<option>bar1</option>
<option>bar2</option>
</select>
```

- **size='n'** gives scrolling list
- **multiple='1'** allows multiple selections





Form elements

- Various other form elements available
- All support several other attributes.

Creating CGI scripts

- HTML **<form>** tag specifies the script that should be run in the **action=** attribute
- CGI scripts live in a specific directory. Typically
  - **/var/www/cgi-bin**
  - **/home/httpd/cgi-bin**
- Web server can be configured to allow CGI scripts in the same directory as HTML pages or elsewhere

Creating CGI scripts

- Must **extract data** sent with GET or POST methods
- This involves quite **complex** unpacking and decoding
- All handled (in Python) by **import cgi**

Using cgi in Python

- Read the form

```
form = cgi.FieldStorage()
```

- The fields are then accessed with

```
value = form[“key”].value
-or-
value = form.getvalue(“key”)
```

Using cgi in Python

- Printing a header

```
print (“Content-Type: text/html”)
print (“”)
```

OR

```
print (“Content-Type: text/html\n”)
```

- This tells the browser what sort of data are being delivered – HTML, plain text, files to be saved, images, audio, etc.

A simple CGI script

```
#!/usr/bin/python

print (“Content-Type: text/html\n”)

print ('''
<html>
<head>
   <title>Hello World!</title>
</head>
<body>
<h1>Hello World!</h1>
</body>
</html>
''')
```

A simple CGI script

- The CGI module was not actually used here!
- We simply printed HTML to standard output.

Another CGI script

```
#!/usr/bin/env python3
import cgi
form = cgi.FieldStorage()
val  = form.getvalue(“id”)
print (“Content-Type: text/html\n”)

html  = “<html>\n”
html += “<head>\n”
html += “<title>Print ID Parameter</title>\n”
html += “</head>\n”
html += “<body>\n”
html += “<p>ID parameter was: “ + val + “</p>\n”
html += “</body>\n</html>”

print (html)
```

Names and values

- Normally only **one value** for each form element (name=)
  - You use a different **name=** attribute for each item
- Selection lists can return **multiple name/value pairs** for the same name.
  - We use `form.getlist()`

Obtaining multiple values

```
#!/usr/bin/env python3
import cgi
form = cgi.FieldStorage()
values = form.getlist(“id”)
print (“Content-Type: text/html\n”)
html = “<html>\n<head>\n”
html += “<title>Print ID Parameter</title>\n”
html += “</head>\n<body>\n”
html += “<p>ID had parameters:</p>\n<ul>\n”
for val in values:
    html += “<li>” + val + “</li>\n”
html += “</ul>\n</body>\n</html>”

print (html)
```

Accessing external programs

- Often need to access another program (e.g. BLAST) from your CGI script
- Run a program from a Python script:

```
import subprocess

result = subprocess.check_output(“prog args”, shell=True)
result = str(result, 'utf-8')
```

Note:

- In Perl you can just do:

  $retval = `cmd args`;

When you don't need output…

```
import os
os.system(“cmd args”)
```
✔ Shell commands, redirection
✘ Escape special chars
✘ Deprecated

```
import subprocess
subprocess.call(“cmd args”, shell=True)
subprocess.call([“cmd”,”arg”])
```
✔ Lots of flexibility – recommended way to do it!

When you need the output…

```
import os
stream = os.popen(“cmd args”)
```
✔ As os.system() but stream is a file handle that can be used in the usual way

---

```
import subprocess
retval=subprocess.check_output(“cmd args”, shell=True)
retval=subprocess.check_output([“cmd”,”arg”])
retval=str(retval, 'utf-8') # Convert from byte string
```
✔ Lots of flexibility – recommended way to do it!
✘ Python >= 2.7

```
import commands
(status,retval) = commands.getstatusoutput(“cmd args”)
```
✔ Simple!
✘ Unix only
✘ Deprecated in Python 3

Accessing external programs

- CGI scripts and the programs they spawn run as the '**nobody**' user.
  - Search path and environment variables may well not be what you expect!

Accessing external programs

- Set any **environment variables** you need in your CGI script:

```
import os
os.environ[“varname”]=”value”
```

- Use the **full path** to any external programs
  - (possible exception of standard Unix-like commands)

Temporary files

- Often need to create **temporary** working files
- Must ensure that the filename is unique
  - More than one person could hit your web server at the same time!
- Use the **process ID** to ensure a unique filename

```
import os
filename = “/tmp/cgifile_” + os.getpid
```

Temporary files

- May need to create a temporary file to **return to the user**
- Most web servers provide a directory in which such files can be written

Summary

- **Forms** are used to send data to the web server
- **GET** and **POST** methods for transferring data
- CGI scripts can **simply serve web pages**
  - no data obtained from a form
- CGI scripts can **obtain data from a page** and run external programs