CGI

Facilitating interactive web applications

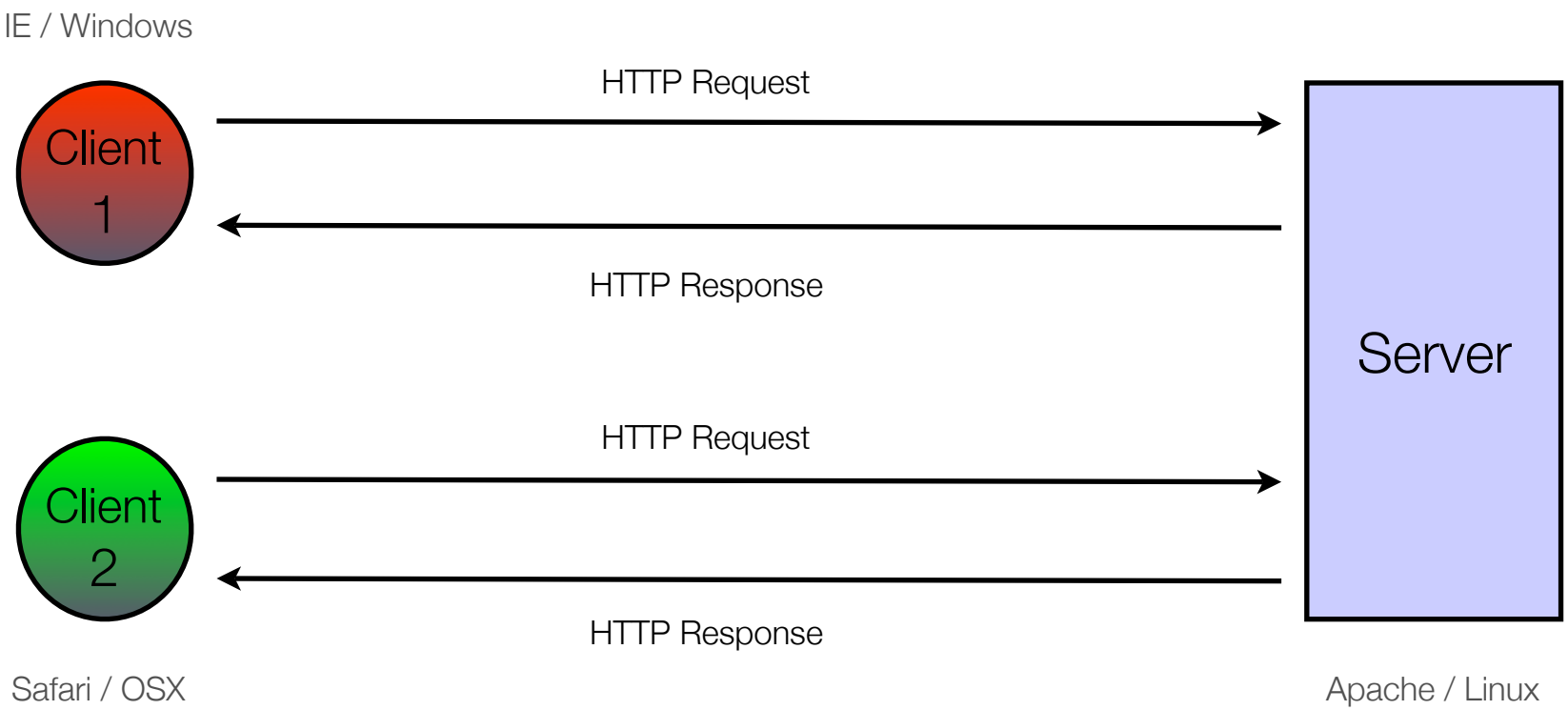
Outline

- In Informatics 1, worksheet 7 says “You will learn more about CGI and forms if you enroll in Informatics 2”.
- Now we make good on that promise.
- First we look at the Hypertext Transfer Protocol (HTTP).
- Then we consider the Common Gateway Interface (CGI).
- In the following lectures we will consider cookies and HTML forms.

Hypertext Transfer Protocol (HTTP)

- One of many protocols for the transfer of data on the Internet.
- Emerged in the early 1990s, and has gone through a couple revisions.
- Version 1.1 appears to the be the most popular version in use today.
- Originally intended for transmission of “hypertext” documents (text with links).
- Now used (or abused) for a wide variety of media (video, audio, web apps).
- Supplanted the *gopher* protocol, and reduced the significance of others, such as *ftp* (file transfer) and *nntp* (news groups).

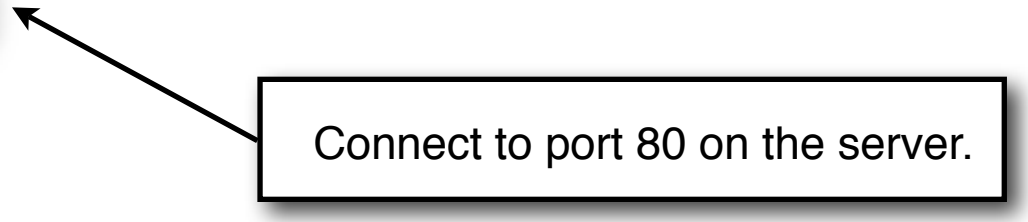
Client-Server Architecture



Demonstration of HTTP using the Unix command “telnet”.
Black text is typed input from the user.
Red text is output from the telnet program.
Green text is output from the web server.

```
telnet www.theage.com.au 80
Trying 203.26.51.71...
Connected to theage.com.au.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.1 302 Found
Date: Sat, 06 Sep 2008 04:35:48 GMT
Server: Apache
Location: http://www.fairfax.com.au
Content-Length: 209
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.fairfax.com.au">here</a>.</p>
</body></html>
Connection closed by foreign host.
```

```
unix> telnet www.theage.com.au 80
Trying 203.26.51.71...
Connected to theage.com.au.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.1 302 Found
Date: Sat, 06 Sep 2008 04:35:48 GMT
Server: Apache
Location: http://www.fairfax.com.au
Content-Length: 209
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.fairfax.com.au">here</a>.</p>
</body></html>
Connection closed by foreign host.
```



```
unix> telnet www.theage.com.au 80
Trying 203.26.51.71...
Connected to theage.com.au.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.1 302 Found
Date: Sat, 06 Sep 2008 04:35:48 GMT
Server: Apache
Location: http://www.fairfax.com.au
Content-Length: 209
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.fairfax.com.au">here</a>.</p>
</body></html>
Connection closed by foreign host.
```

Request a document from the server, using version 1.0 of HTTP protocol.

```
unix> telnet www.theage.com.au 80
Trying 203.26.51.71...
Connected to theage.com.au.
Escape character is '^]'.
GET /index.html HTTP/1.0
```

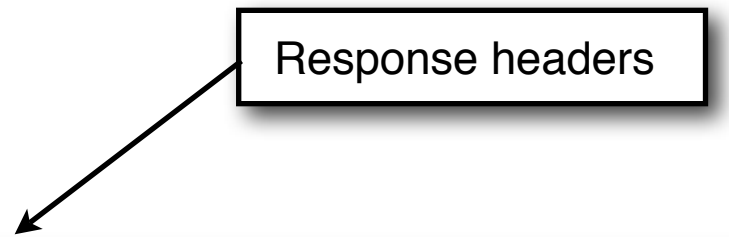
Response from the server, using version 1.1 of the HTTP protocol.

```
HTTP/1.1 302 Found
Date: Sat, 06 Sep 2008 04:35:48 GMT
Server: Apache
Location: http://www.fairfax.com.au
Content-Length: 209
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.fairfax.com.au">here</a>.</p>
</body></html>
Connection closed by foreign host.
```

```
unix> telnet www.theage.com.au 80
Trying 203.26.51.71...
Connected to theage.com.au.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.1 302 Found
Date: Sat, 06 Sep 2008 04:35:48 GMT
Server: Apache
Location: http://www.fairfax.com.au
Content-Length: 209
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.fairfax.com.au">here</a>.</p>
</body></html>
Connection closed by foreign host.
```

Status code.
302 means “redirect” to another URI (slight abuse of the standard).

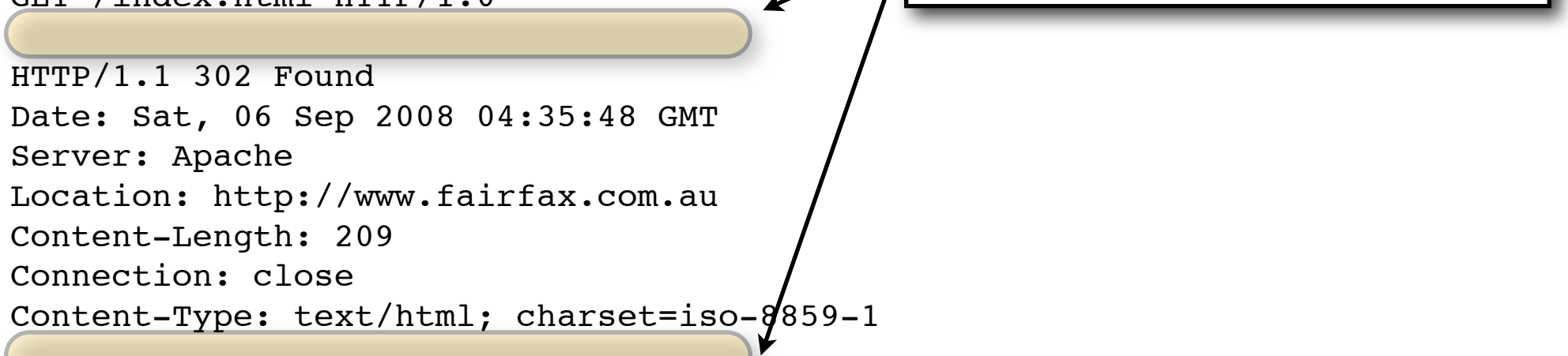
```
unix> telnet www.theage.com.au 80
Trying 203.26.51.71...
Connected to theage.com.au.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.1 302 Found
Date: Sat, 06 Sep 2008 04:35:48 GMT
Server: Apache
Location: http://www.fairfax.com.au
Content-Length: 209
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.fairfax.com.au">here</a>.</p>
</body></html>
Connection closed by foreign host.
```



```
unix> telnet www.theage.com.au 80
Trying 203.26.51.71...
Connected to theage.com.au.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.1 302 Found
Date: Sat, 06 Sep 2008 04:35:48 GMT
Server: Apache
Location: http://www.fairfax.com.au
Content-Length: 209
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.fairfax.com.au">here</a>.</p>
</body></html>
Connection closed by foreign host.
```

MIME type.
Tells the client what kind of entity appears in the body of the response.

```
unix> telnet www.theage.com.au 80
Trying 203.26.51.71...
Connected to theage.com.au.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.1 302 Found
Date: Sat, 06 Sep 2008 04:35:48 GMT
Server: Apache
Location: http://www.fairfax.com.au
Content-Length: 209
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.fairfax.com.au">here</a>.</p>
</body></html>
Connection closed by foreign host.
```



```
unix> telnet www.theage.com.au 80
Trying 203.26.51.71...
Connected to theage.com.au.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.1 302 Found
Date: Sat, 06 Sep 2008 04:35:48 GMT
Server: Apache
Location: http://www.fairfax.com.au
Content-Length: 209
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

Response body (HTML).

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.fairfax.com.au">here</a>.</p>
</body></html>
Connection closed by foreign host.
```

```
unix> telnet www.theage.com.au 80
Trying 203.26.51.71...
Connected to theage.com.au.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.1 302 Found
Date: Sat, 06 Sep 2008 04:35:48 GMT
Server: Apache
Location: http://www.fairfax.com.au
Content-Length: 209
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.fairfax.com.au">here</a>.</p>
</body></html>
Connection closed by foreign host.
```

Connection to server closed.

HTTP is *very* simple

- GET - retrieve an entity (document) from the server.
- POST - submit data to the server (usually via an HTML form) for processing.
- PUT - send an entity to be stored on the server.
- DELETE - request the removal of an entity from the server.
- OPTIONS - ask the server to reveal its capabilities.
- HEAD - same as GET but only return HTTP headers and not the document.
- http://www.ietf.org/rfc/rfc2616.txt for more details.

The Common Gateway Interface (CGI)

- A standard method which allows the server to construct a dynamic web page (or other entity) in response to a request from a client.
- The dynamic content is created by a program running on the server, often called a *CGI script*.
- The URI for the requested content identifies the name of the CGI script to use.
- The server locates the CGI script, executes it (possibly with some additional input sent from the client), and sends the output of the script back to the client.
- The script can be implemented in any programming language (even compiled languages).

Input to CGI scripts

- CGI scripts are executed on behalf of the web server.
- The server initialises special “environment variables” to communicate information to the CGI script.
- Environment variables are provided by the operating system.
- “Post” information is given to the script via “the standard input device (stdin)”.
- This design reflects the Unix roots of CGI (and the web in general).

```
#!/opt/local/bin/python2.5
# A Python CGI program which prints out its environment variables,
# and the contents of stdin.

import os
import sys

print 'Content-Type: text/plain'
print

for var in os.environ:
    print '%s = %s' % (var, os.environ[var])

print 'stdin = %s' % sys.stdin.read()
```

```
#!/opt/local/bin/python2.5
# A Python CGI program which prints out its environment variables,
# and the contents of stdin.

import os
import sys

print 'Content-Type: text/plain'
print

for var in os.environ:
    print '%s = %s' % (var, os.environ[var])

print 'stdin = %s' % sys.stdin.read()
```

Tells the operating system how to execute this script. System dependent. Not needed on IVLE.

```
#!/opt/local/bin/python2.5
# A Python CGI program which prints out its environment variables,
# and the contents of stdin.

import os
import sys

print 'Content-Type: text/plain'
print

for var in os.environ:
    print '%s = %s' % (var, os.environ[var])

print 'stdin = %s' % sys.stdin.read()
```

Partial HTTP response header (just the MIME type). Server will fill in the rest for us. Note the blank line to indicate the end of the header.

```
#!/opt/local/bin/python2.5
# A Python CGI program which prints out its environment variables,
# and the contents of stdin.

import os
import sys

print 'Content-Type: text/plain'
print

for var in os.environ:
    print '%s = %s' % (var, os.environ[var])

print 'stdin = %s' % sys.stdin.read()
```

Generate dynamic (text) output which is sent back to the client as response body.

Print the contents of the standard input for the script.

Print the value of each environment variable.

CGI environment variables on IVLE

- The previous Python script is served on IVLE at the address:

```
http://students.informatics.unimelb.edu.au/~bjpope/info2/mywork/lectures/cgi/env.py
```

- We can examine its output by loading it in a web browser (say Firefox).

- This will show us the values of all the environment variables available to the script when it is run by the web server on IVLE.

- You would see similar behaviour on other web servers, but some of the values of the environment variables will be different.

```
SERVER_SOFTWARE = IVLE/0.1
SCRIPT_NAME = /~bjpope/info2/mywork/lectures/cgi/env.py
SERVER_SIGNATURE = <address>Apache/2.2.8 (Ubuntu) DAV/2 SVN/1.4.6 mod_python/3.3.1 ...
REQUEST_METHOD = GET
HTTP_KEEP_ALIVE = 300
SERVER_PROTOCOL = HTTP/1.1
QUERY_STRING =
HOME = /home/bjpope
HTTP_ACCEPT_CHARSET = ISO-8859-1,utf-8;q=0.7,*;q=0.7
HTTP_USER_AGENT = Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-GB; rv:1.9.0.1)
Gecko/2008070206 Firefox/3.0.1
HTTP_CONNECTION = keep-alive
SERVER_NAME = students.informatics.unimelb.edu.au
REMOTE_ADDR = 128.250.190.211
PATH_TRANSLATED = /home/bjpope/info2/cgi-lecture/env.py
SERVER_PORT = 80
SERVER_ADDR = 128.250.190.210
SERVER_ADMIN = ivle-sysadmin@informatics.unimelb.edu.au
HTTP_HOST = students.informatics.unimelb.edu.au
REQUEST_URI = /serve/bjpope/info2/cgi-lecture/env.py
HTTP_ACCEPT = text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
GATEWAY_INTERFACE = CGI/1.1
HTTP_X_FORWARDED_FOR = 59.167.64.206
REMOTE_PORT = 60609
HTTP_ACCEPT_LANGUAGE = en-gb,en;q=0.5
REMOTE_HOST = 128.250.190.211
HTTP_ACCEPT_ENCODING = gzip,deflate
PATH_INFO =
stdin =
```

```
SERVER_SOFTWARE = IVLE/0.1
SCRIPT_NAME = /~bjpope/info2/mywork/lectures/cgi/env.py
SERVER_SIGNATURE = <address>Apache/2.2.8
REQUEST_METHOD = GET
HTTP_KEEP_ALIVE = 300
SERVER_PROTOCOL = HTTP/1.1
QUERY_STRING =
HOME = /home/bjpope
HTTP_ACCEPT_CHARSET = ISO-8859-1,utf-8;q=
HTTP_USER_AGENT = Mozilla/5.0 (Macintosh;
Gecko/2008070206 Firefox/3.0.1
HTTP_CONNECTION = keep-alive
SERVER_NAME = students.informatics.unimelb.edu.au
REMOTE_ADDR = 128.250.190.211
PATH_TRANSLATED = /home/bjpope/info2/cgi-lecture/env.py
SERVER_PORT = 80
SERVER_ADDR = 128.250.190.210
SERVER_ADMIN = ivle-sysadmin@informatics.unimelb.edu.au
HTTP_HOST = students.informatics.unimelb.edu.au
REQUEST_URI = /serve/bjpope/info2/cgi-lecture/env.py
HTTP_ACCEPT = text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
GATEWAY_INTERFACE = CGI/1.1
HTTP_X_FORWARDED_FOR = 59.167.64.206
REMOTE_PORT = 60609
HTTP_ACCEPT_LANGUAGE = en-gb,en;q=0.5
REMOTE_HOST = 128.250.190.211
HTTP_ACCEPT_ENCODING = gzip,deflate
PATH_INFO =
stdin =
```

The QUERY_STRING variable, and the standard input are the two key places where the script receives input from the client (via the server).

In this example they are empty.

Sending input via the query string

- A query string is everything in a URL to the right of the question mark character:

```
http://foo.bar.com/somescript.py?name=James+Bond&mission=top+secret
```

- Look what happens when we open this (long) URL in Firefox:

```
http://students.informatics.unimelb.edu.au/~bjpope/info2/mywork/lectures/cgi/env.py?
name=James+Bond&mission=top+secret
```

```
SERVER_SOFTWARE = IVLE/0.1
SCRIPT_NAME = /~bjpope/info2/mywork/lectures/cgi/env.py
SERVER_SIGNATURE = <address>Apache/2.2.8 (Ubuntu) DAV/2 SVN/1.4.6 mod_python/3.3.1 ...
REQUEST_METHOD = GET
HTTP_KEEP_ALIVE = 300
SERVER_PROTOCOL = HTTP/1.1
QUERY_STRING = name=James+Bond&mission=top+secret
HOME = /home/bjpope
HTTP_ACCEPT_CHARSET = ISO-8859-1,utf-8;q=0.7,*;q=0.7
HTTP_USER_AGENT = Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-GB; rv:1.9.0.1)
Gecko/2008070206 Firefox/3.0.1
HTTP_CONNECTION = keep-alive
SERVER_NAME = students.informatics.unimelb.edu.au
REMOTE_ADDR = 128.250.190.211
PATH_TRANSLATED = /home/bjpope/info2/cgi-lecture/env.py
SERVER_PORT = 80
SERVER_ADDR = 128.250.190.210
SERVER_ADMIN = ivle-sysadmin@informatics.unimelb.edu.au
HTTP_HOST = students.informatics.unimelb.edu.au
REQUEST_URI = /serve/bjpope/info2/cgi-lecture/env.py
HTTP_ACCEPT = text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
GATEWAY_INTERFACE = CGI/1.1
HTTP_X_FORWARDED_FOR = 59.167.64.206
REMOTE_PORT = 60609
HTTP_ACCEPT_LANGUAGE = en-gb,en;q=0.5
REMOTE_HOST = 128.250.190.211
HTTP_ACCEPT_ENCODING = gzip,deflate
PATH_INFO =
stdin =
```

Special characters in the query string

- The query string can contain characters which have special meaning, e.g.

```
? & : + %    (and whitespace characters).
```

- To include such characters in the query string, they must be encoded.

- The usual way of encoding special characters is to use a hexadecimal number representing the code of the character (though whitespace is encoded as '+').

- For instance: '?' is encoded as '%3F'. The percent sign tells us that the next two characters are hexadecimal encoding numbers.

- 3F is hexadecimal for sixty three, which is the ASCII code for '?'

Sending input via stdin

- The CGI script receives input via stdin (the standard input device) when the client sends a POST request (instead of a GET request).
- The POST method sends the data in the body of the request.
- The GET method sends the data in the header of the request, as part of the URL (as the query string).
- There are pros and cons of both approaches.
- We will get back to the POST request when we see HTML forms in the next lecture.

Accessing the CGI input in a systematic way

- We could just look inside the QUERY_STRING variable and stdin.
- Troublesome to have to look in both places all the time.
- If we are receiving input from a HTML form, then the input data is structured like so:

  name1=value1&name2=value2&name3=value3

  where name1, name2, name3 represent input items in the form.
- It is tedious to deal with the unstructured representation as an encoded string.

```
#!/opt/local/bin/python2.5
# A Python CGI program which demonstrates the use of FieldStorage

import cgi

print 'Content-Type: text/plain'
print

store = cgi.FieldStorage()

for var in store:
    print "%s = %s" % (var, store.getvalue(var))
```

```
#!/opt/local/bin/python2.5
# A Python CGI program which demonstrates the use of FieldStorage

import cgi

print 'Content-Type: text/plain'
print

store = cgi.FieldStorage()

for var in store:
    print "%s = %s" % (var, store.getvalue(var))
```

This function reads the QUERY_STRING and stdin, decodes them, and parses each ‘name=value’ entry. It builds a data structure which behaves like a dictionary, mapping each ‘name’ to its corresponding ‘value’.

cgi.FieldStorage in Python

- The previous Python script is served on IVLE at the address:

```
http://students.informatics.unimelb.edu.au/~bjpope/info2/mywork/lectures/cgi/
fieldstore.py
```

- Consider what happens when we request that URL with the query string `name=James+Bond&mission=top+secret`

- We get the output:

```
name = James Bond

mission = top secret
```