

CSE 135 Server Side Web Languages Lecture # 2

HTTP Overview

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

It's all about the network...

- If you want to really do Web programming right you will need to know the ins and outs of HTTP
 - If the network has problems you/users have problems much more than you are probably aware
- Sadly most don't know as much as they think they do
 - easily demoted by perf and security problems
 - A few examples
 - URLs - case sensitive? Length? Format
 - GET vs POST?
 - Cookies
 - "Layer 8" Error Correction - the meat layer

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

HTTP Intro

- HTTP (Hyper Text Transfer Protocol)
 - It's an *application layer protocols* similar to SMTP, POP, IMAP, NNTP, FTP, etc.
 - Simple protocol that defines the standard way that clients request data from Web servers and how these server respond
 - Typically it is running on top of TCP/IP
- Three versions have been used (0.9,1.0,1.1) and two are still commonly used
 - RFC 1945 HTTP 1.0 (1996)
 - RFC 2616 HTTP 1.1 (1999)

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

HTTP and TCP/IP

HTTP sits atop the TCP/IP Protocol Stack

Application Layer

HTTP

Transport Layer

TCP

Network Layer

IP

Data Link Layer

Network Interfaces

Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

HTTP and TCP/IP, contd.

- IP provides *packets* that are *routed* based on source and destination IP addresses
- TCP provides *segments* that ride inside the IP packets and add connection information based on source and destination *ports*
 - The ports let TCP carry multiple protocols that connect services running on default ports
 - HTTP on port 80
 - HTTP with SSL (HTTPS) on port 443
 - FTP on port 21
 - SMTP on port 25
 - SSH on port 22

Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

HTTP and TCP/IP, contd.

- TCP also provides mechanisms to make the connection a *reliable* bit pipe
 - 3-way handshake, sequence numbers, checksums, control flags
 - A data stream is chopped up into chunks that are reassembled, complete and in correct order on the other endpoint of the connection
- TCP segments, riding inside IP packets, carry the chunks of data
 - When HTTP is the Application Layer protocol on top of the stack, these chunks of data are the contents of the HTTP Message

Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

HTTP over TCP/IP Examples

GET /index.html HTTP/1.1<CRLF>
Host: www.hostname.com Con...

HTTP Message's data stream is chopped up into chunks small enough to fit in a TCP segment

The chunks ride inside TCP segments used to reassemble them correctly on the other end of the connection

The segments are shipped to the right destination inside IP datagrams

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

HTTP over TCP/IP Issues?

- HTTP/1.0 opens and closes a new TCP connection for each operation. Since most Web objects are small, this practice means a high fraction of packets are simply TCP
- Add the previous point to TCP's slow start congestion control mechanism and you find HTTP/1.0 operations use TCP at its least efficient.
- HTTP 1.1 addresses these concerns with persistent connections using Keep-Alive
- See <http://www.w3.org/Protocols/HTTP/Performance> for papers and information on HTTP performance issues

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Basic HTTP Request/Response Cycle

HTTP Client
Asks for resource by its URL:
`http://www.foo.com/page.html`

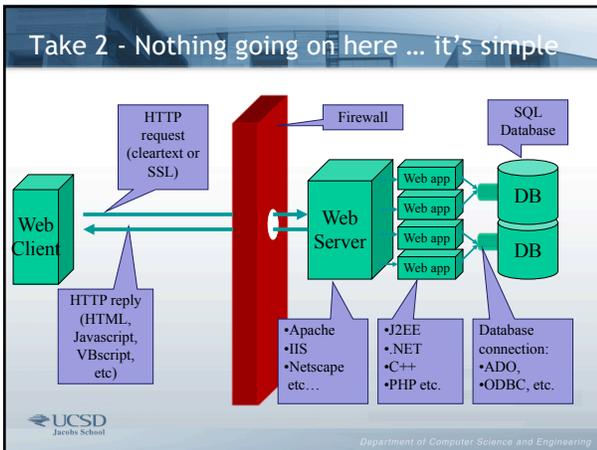
HTTP Request

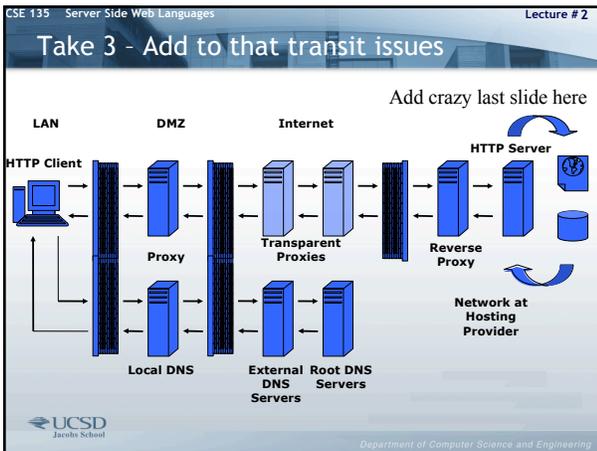
HTTP Response

HTTP Server
`www.foo.com`

Resource
`/page.html`

UCSD Jacobs School Department of Computer Science and Engineering





- ### Types and Uses of Proxy Servers
- Proxies are HTTP Intermediaries
 - All act as both clients and servers
 - Friend and foe to Web devs
 - Major types of proxies can be distinguished by where they live and how they get traffic
 - Explicit
 - Transparent/Intercepting
 - Reverse/Surrogate
 - Three primary uses for proxies
 1. Security
 2. Performance
 3. Content Filtering
- UCSD Jacobs School
Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

HTTP Requests

- HTTP requests and responses are both types of Internet Messages (RFC 822) , and share a general format:
 - A Start Line, followed by a CRLF
 - Request Line for requests
 - Status Line for responses
 - Zero or more Message Headers
 - field-name ":" [field-value] CRLF
 - An empty line
 - Two CRLFs mark the end of the Headers
 - An optional Message Body if there is a payload
 - All or part of the "Entity Body" or "Entity"

UCSD Jacobs School
Department of Computer Science and Engineering

The Law of Three

- Consider the form of HTTP requests and responses there are three pieces to each
- As a server-side program you understand you can only get data from the three pieces of the request (so consider those your input and watch them carefully!)
- Furthermore you can only output via the three pieces of the response (mostly the message body)
- This is it, there is nothing more to the bedrock of the Web server side wise (save addressing statefulness)

UCSD Jacobs School
Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Making a simple HTTP request

- You can do the last example with a tool or just use telnet to access the default HTTP port (80)
 - C:\>telnet www.google.com 80
- Ask for a resource using a minimal request syntax:
 - GET / HTTP/1.1 <CRLF>
 - Host: www.google.com <CRLF><CRLF>

Note: A Host header is required for HTTP 1.1 connections, though not for HTTP 1.0

UCSD Jacobs School
Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

A Closer Look at the Request Methods

- GET
 - By far most common method
 - Retrieves a resource from the server
 - Supports passing of query string arguments
- HEAD
 - Retrieves only the Headers associated with a resource but not the entity itself
 - Highly useful for protocol analysis, diagnostics
- POST
 - Allows passing of data in entity rather than URL
 - Can transmit of far larger arguments that GET
 - Arguments not displayed on the URL

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

More Request Methods

- OPTIONS
 - Shows methods available for use on the resource (if given a path) or the host (if given a "*")
- TRACE
 - Diagnostic method for assessing the impact of proxies along the request-response chain
- PUT, DELETE
 - Used in HTTP publishing (e.g., WebDav)
- CONNECT
 - A common extension method for Tunneling other protocols through HTTP
- There's even more methods if you look at WebDav

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Why do I care?

- Well if you are doing doing Web programming you may have to form raw requests with headers ourselves.
 - Example in JavaScript using Ajax you will have to form raw HTTP requests using GET and POST (or even HEAD if you like) to transmit your data

```
xmlhttp = xmlhttp();
xmlhttp.open("POST", url, true);
xmlhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xmlhttp.send("ret=" + escape(param));
```
- Also in HTML forms when you set the action attribute <form action="GET|POST" > you are specifying the HTTP method to transmit the data

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

A Closer Look at the Request URI

- Absolute URI vs. Absolute Path
 - Explicit Proxies Require Absolute URIs
 - Client is connected directly to the proxy
 - Protocol and host name needed to resolve request
 - You might need full URLs too esp. for Web services
 - Watch out for www vs. no www issues
 - Grammar of the Absolute Path
 - Like Absolute URI minus the "http://hostname"
 - Initial "/" equivalent of the host's document root
 - In HTTP 1.1 with *name-based virtual hosting* Host header directs request to appropriate document root

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

The HTTP Response

- Response status line consists of three major parts
 - The HTTP Version followed by a space
 - Status Code followed by a SP
 - 5 groups of 3 digit integers indicating the result of the attempt to satisfy the request
 - 1xx are informational
 - 2xx are success codes
 - 3xx are for alternate resource locations (redirects)
 - 4xx indicate client side errors
 - 5xx indicate server side errors
 - The "Reason Phrase" followed by the CRLF

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Observation - One Way Requests and 204s

- There are many details to HTTP that people don't consider but are highly useful one example is 204 responses which send back no data
- Observe Google using this in its search results page to send what I dub a "flare gun" request to see what exactly the user clicked on
 - The purpose of this is for improving search quality and defeating those folks who reverse engineer the Google algorithm - "The human filter if you like"

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

HTTP Headers

- Headers come in four major types, some for requests, some for responses, some for both:
 - General Headers
 - Provide info about messages of both kinds
 - Request Headers
 - Provide request-specific info
 - Response Headers
 - Provide response-specific info
 - Entity Headers
 - Provide info about request and response entities
 - Extension headers are also possible

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

A Closer Look at General Headers

- Connection - lets clients and servers manage connection state
 - Connection: Keep-Alive (HTTP 1.0)
 - Connection: close (HTTP 1.1)
- Date - when the message was created
 - Date: Sat, 31-May-03 15:00:00 GMT
- Via - shows proxies that handled message
 - Via: 1.1 www.myproxy.com (Squid/1.4)
- Cache-Control - Among the most complex of headers, enables caching directives
 - Cache-Control: no-cache

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Why do I care? - Unfriendly Caches

- If you are issuing a GET request and you do it again the browser will not bother to talk to the server (depending on browser settings including defaults), but instead will pull the data from cache. This can cause lots of screw-ups
 - Example - someone looking at stale content
 - Example - Problems with Ajax style apps never waking up because browser using cached data
- Obvious solution to "stale caches" is to add cache control headers (or to change resource names) but then again that does defeat the value
 - Better to know about caching and do it properly
 - Consider typical Web pages what would you want to cache?

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

A Closer Look at Request Headers

- Host - The hostname (and optionally port) of server to which request is being sent
 - Required for name-based virtual hosting
 - Host: www.port80software.com
- Referer - The URL of the resource from which the current request URI came
 - Misspelled in the specification
 - Referer: http://www.host.com/login.asp
- User-Agent - Name of the requesting application, used in browser sensing
 - User-Agent: Mozilla/4.0 (Compatible; MSIE 6.0)

 Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Some More Request Headers

- Accept and its variants - Inform servers of client's capabilities and preferences
 - Enables *content negotiation*
 - Accept: image/gif, image/jpeg;q=0.5
 - Accept- variants for Language, Encoding, Charset
- If-Modified-Since and other *conditionals*
 - Frequently used by browsers to manage caches
 - If-Modified-Since: Sat, 31-May-03 15:00:00 GMT
- Cookie - How clients pass cookies back to the servers that set them
 - Cookie: id=23432;level=3

 Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Using Request Headers: Browser Sniffing

- User-agent is often used in browser detection to serve different type of page to different type of accessing agent
 - Similarity problem
 - Everything looks like old "Mozilla"
 - Spoofing or removing problem
- Better approach is to take this and add in an injected script or program that profiles the device.
- In the long run as device diversity grows the concept of browser will evolve significantly

 Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Using Request Headers: Anti-Leeching

- Often times people may leech your bandwidth with direct hotlinking to your object (GIF, Flash, etc.) without fetching the other related objects
 - This certainly would be bad if your biz model was about people seeing the related ads around the 'stolen' object
- Since the referer header is sent from the base page a simple form of anti-leeching is to check for it before sending a dependent object
- Of course the bad guy now moves to forge the header
- Class Question: can you think of other countermeasures?

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Using Request Headers: Content Negotiation

- User-agent sends accept header indicating type of content it can handle

```
GET /images/HF_servermask HTTP/1.1
Host: www.port80software.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.6) Gecko/20040206 Firefox/0.8
Accept: image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Using Request Headers: Content Negotiation

- A "q-rating" can indicate the preference the user agent has for the data requested
- Content negotiation allows us to ask for something like "logo" and then get the appropriate image (PNG, JPG, etc.) based upon what the device can handle.
 - This leads to extensionless URLs which aids in long term maintainability
 - We'll see the file extensions don't mean much really
- Content negotiation can also allow language to be automatically negotiated.

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Using Request Headers: HTTP Compression

- Compressed HTTP is enabled via Accept headers
- User agent sends header indicating compression acceptance (gzip or deflate). Server using mod_gzip, httpZip, etc. sends compressed content or not.
- Works only on text (HTML, CSS, JS) but with compression up to 70% or more
- Time to first byte increased so high speed connections may not see as much benefit, though bandwidth is saved. Low speed clearly sees benefit.

UCSD Jacobs School
Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

HTTP Compression Example

The screenshot shows two side-by-side windows of the PipeBoost Header Retrieval tool. Both windows are set to download from <http://www.google.com/> using Mozilla/4.0 (compatible; MSIE 6.0) as the user agent. The left window has 'Compression' set to 'Disabled' and shows a 'Response Size' of 2,393. The right window has 'Compression' set to 'Enabled' and 'Compression Acceptance' checked, showing a 'Response Size' of 1,062. Both windows show the same cookies and HTTP headers, including 'Cache-Control: private', 'Content-Type: text/html', and 'Set-Cookie: PREF=ID=08649e7c3846c234:TM=1088728;'. The UCSD logo and department name are visible at the bottom.

CSE 135 Server Side Web Languages Lecture # 2

Compression Considerations

- Increased origin server CPU - or wasted cycles?
- TTFB vs TTLB consideration and LANs
- Decompress times
- Nasty little bugs
 - <http://support.microsoft.com/default.aspx?scid=kb;en-us;823386&Product=ie600>
 - In Internet Explorer, ... The bytes that remain to be decoded in the buffer may be small (8 bytes or less) and the data contained in the buffer decompresses to 0 bytes. ... When Mshtml receives 0 bytes, it thinks that all the data is read and closes the data stream. As a result, the HTML page sometimes appears truncated. Typically, if it is for a referenced file such as a .js or a .css file type, the HTTP connection stops responding.*
 - Most solved by commercial server-side implementations
- People in the know use this stuff

UCSD Jacobs School
Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Response Headers

- Server - The server's name and version
 - Server: Microsoft-IIS/5.0
 - Can be problematic for security reasons
 - Security by obscurity?
- Vary - Tells client & proxy caches which headers were used for content negotiation
 - Vary: User-Agent, Accept
- Set-Cookie - This is how a server sets a cookie on a client
 - Set-Cookie: id=234; path=/shop; expires=Sat, 31-May-03 15:00:00 GMT; secure

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

A Closer Look at Entity Headers

- Allow - Lists the request methods that can be used on the entity
 - Allow: GET, HEAD, POST
- Location - Gives the alternate or new location of the entity
 - Used with 3xx response codes (redirects)
 - Location: <http://www.ibm.com/us/>
- Content-Encoding - specifies encoding performed on the body of the response
 - Corresponds to Accept-Encoding request header
 - Content-Encoding: gzip

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

More Entity Headers

- Content-Length - The size of the entity body in bytes
 - Value shrinks when compression is applied
 - Content-Length: 24000
- Content-Location - The actual URL of the resource if different than its request URL
 - Often used to show the index or default page
 - Content-Location: <http://www.foo.com/home.html>

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

More Entity Headers

- Content-Type - specifies Media (MIME) type of the entity body
 - Corresponds to Accept header
 - Content-Type: image/png
- This is the most important header to the browser. The data in this header tells the browser what it is receiving. Now it should make sense why file extensions don't really matter and are arbitrary.
 - Server: file extension -> Mime type
 - Browser: Mime type -> Action (display, download, etc.)
- *Note: Without HTTP browser relies on file extension - example loading a file off local disk.*

UCSD Jacobs School
Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Why do I care?

- Because sometimes you need to stamp outgoing data on the server-side with the appropriate MIME type

```

4 header("Content-Type: text/xml");
5
6 $datemsg = "Hello World to " . rawurlencode($name) . " at " . date("h:i:s A");
7
8 echo <<< END_OF_FILE
9 <hello>
10 <message id="date">{$datemsg}</message>
11 </hello>
12 END OF FILE

```

UCSD Jacobs School
Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

More Entity Headers : Caching Related

- Expires - Gives expiration for the instance of the resource for use in caching
 - Expires: Sat, 31-May-03 19:00:00 GMT
- Last-Modified - Date/time the entity was last changed (or created)
 - Last-Modified: Fri 30-May-03 09:00:00 GMT

UCSD Jacobs School
Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

More Entity Headers : Caching Related

- Etag - Uniquely identifies a particular instance of a given resource
 - Used with conditional request headers to validate cached instances of the resource
 - If-Match, If-None-Match
 - Etag: adkskdashjgk07563AF

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Why do I care?

- Well you could go beyond basic cache-control and pragma headers and do Expires and other forms of cache hints.
- Ultimately you may be forced to use a query string or alternate file name to force misconfigured caches to stop causing you problems

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Sending data via HTTP

- Data can be sent to a server-application in two primary ways:
 1. Query String sent via a GET request
 2. Data body sent via a POST request
- In both cases the data is encoded in a special manner called **x-www-form-urlencoded** which replaces spaces with + symbols, special characters with %hex values equivalent to the particular special character being "escaped" and separates individual arguments to be passed with ampersands (&) characters.
- *Note: Data may be sent via HTTP headers mostly in the form of cookie based data. Though other HTTP headers such as user-agent, referrer, etc. can be tapped, but this is generally not user supplied but instead constitutes the environment in which the Web transaction takes place.*

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Sending Data with GET

- In the case of GET we see submitted data (often from a fill-in form though may be hard coded in links) with the actual request URL
- The passed data is called the query string and follows a ? Character in the URL
 - Example: `http://www.pint.com/cgi-bin/doi.pl?name=Thomas`
 - Example: `http://www.pint.com/cgi-bin/toit.pl?x=5&y=7`
- These "dirty URLs" have potential downsides including:
 - Technology exposure - "visual reconnaissance"
 - Easy fiddling of parameters
 - Poor usability (may be a good thing as well)
 - Lack of long term maintainability
 - Size limit is dependent on URL size limits
- However, GET string based URLs are portable - you can bookmark them, send to friends, etc.

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Sending Data with GET Contd.

- The GET based data can be submitted in one of two ways
 - Hard-coded into a link


```
<a href="http://www.google.com/search?q=Web+server+software">Run query</a>
```
 - As a result of a form submission


```
<form action="http://www.google.com/search" method="get">
<label>Query: <input type="text" name="q"></label>
<input type="submit" value="Submit">
</form>
```

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Sending Data with GET Contd.

- Now it should start to make sense what query strings mean and how they are formed



UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Sending Data with GET Contd.

- Behind the scenes you see that indeed the data is transmitted in the request method itself

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Sending Data with Post

- In the case of POST you always generate the request either programmatically or more likely with a form

```
<form action="http://www.example.com/programs/submit-query" method="post">
  Query: <input type="text" name="query">
  <input type="submit" value="Submit">
</form>
```

- The POST request sends the data in the message body but does so in **x-www-form-urlencoded** as well so we might have a message body like

```
Name=A1+Smith&Age=30&Sex=male
```

- No size limit, but issues with browsers have to address lack of redos "Repost form data?"

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Sending Data with Post Contd.

- The network trace shows the difference between POST and GET

```
http://www.pint.com/
POST / HTTP/1.1
Host: www.pint.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.6) Gecko/20040206 Firefox/0.8
Accept: application/x-shockwave-flash;text/xml,application/xml,application/xhtml+xml;text/html;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: cookies=true; CFID=3441521; CFTOKEN=37652414
Content-Type: application/x-www-form-urlencoded
Content-Length: 33
  username=Thomas+Powell&number=345
```

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

Why do I care?

- GET and POST have different uses
- GET used when request is idempotent - meaning multiple requests return same result. POST should be used when you change the state of the server
- Lots of folks will often use GET for state changes because of ease of coding
 - Downsides - inadvertent state changes by spiders, browsers, etc.

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 2

HTTP Considerations

- HTTP is a stateless protocol
 - No "memory" from one request to the next
- Question: How can you keep track of information from one page to the next?
- Answer:
 - Hidden Form fields that are posted backed to the server
 - E.g. Microsoft's VIEWSTATE value in .NET
 - Data posted in dirty URL strings
 - Cookies
 - Two types - memory or "session cookies" and persistent or "disk cookies"
- Many programming environments go to significant ends to make provide for easy state management - more on this later!

UCSD Jacobs School Department of Computer Science and Engineering

HTTP Futures

- It is clear (particularly when we look later at performance and security) that HTTP is a bit too simple for our modern Web needs.
- What should we do?
 - HTTP 2 or SPDY - yep that's working right now
 - Tunnel over SSL, it's really the only way and it works today!
 - Reality Check - watch out!
 - Parallel protocol for app part
 - See Web Sockets
 - If you could own both ends you could do some rather interesting things ... I'm no fortuneteller but, ...

UCSD Jacobs School Department of Computer Science and Engineering
