

Web Systems

John Reekie
University of Technology, Sydney

Terms of Use: Creative Commons Attribution-ShareAlike 2.5
<http://creativecommons.org/licenses/by-sa/2.5/>

What is "the web"?

A collection of cross-linked "websites"

Everyperson's cheap'n'easy client-server computing

A set of "web" technologies, including HTTP, HTML, and everything built around them

The consistent use of URIs to represent resources

HTTP and HTML, as they were

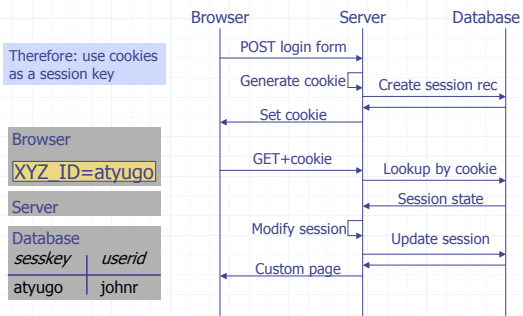
- ◆ A simple markup and formatting language
- ◆ A simple, woefully-inefficient protocol
- ◆ A stroke of genius -- anyone could implement a web client or server
- ◆ "Build it and they will fix it"

```
telnet google.com.au 80
GET /index.html HTTP/1.0
↵
```

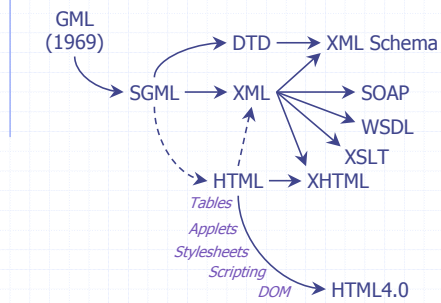
```
HTTP/1.0 200 OK
Content-Type: text/html
Content-length: 2769
Server: GWS/2.1
Date: Wed, 22 Oct 2003 06:48:36
Connection: Keep-Alive
```

```
<html><head>
<title>Google</title>
...
</html>
```

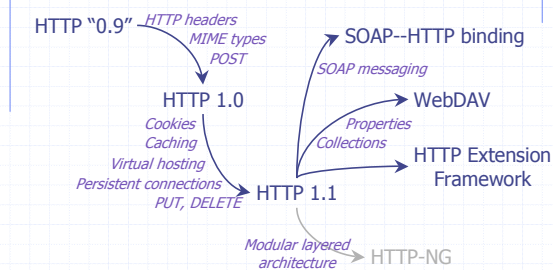
HTTP is stateless



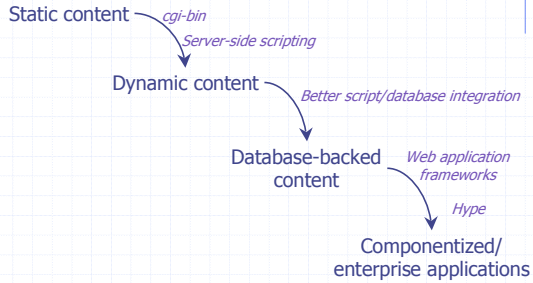
A brief history of the web (1)



A brief history of the web (2)



A brief history of the web (3)



Implementing web applications

- ◆ An HTTP handler on the front end
- ◆ Almost any programming language behind that...
- ◆ And "a" database



A hierarchy of services



Application service provider. Provides specific applications, data management, backup services, security.



Internet service provider (upper). Provides virtual web hosting, network bandwidth, security.



Internet service provider (lower). Provides physical servers, operating system, raw backup services, network bandwidth.



Data center. Provides physical space, physical security, power, network connectivity, basic network security.

Web services

◆ MATYCPASA

- ◆ XML
- ◆ XML-RPC
- ◆ REST
- ◆ SOAP
- ◆ WSDL
- ◆ UDDI



Definitions

""Web Services" is the umbrella term of group of loosely related Web-based resources and components that may be used by other Web applications over HTTP. Those resources could include anything from phone directory data to weather data to sports results. "

<http://www.sitepoint.com/glossary.php?q=W>

"Web services let computers talk to one another over the Internet, allowing computer programs to exchange information by eliminating barriers such as different hardware platforms, software languages, and operating systems that usually make different programs incompatible. Web services make it easier to share information, data, and services, as well as making it cheaper and easier for businesses to work with on-line partners. This technology will help to increase e-business."

<http://www.albertasupernet.ca/general/glossaryp-w.htm>

Three competing (?) approaches

- ◆ REST – Representational State Transfer
 - Requests use HTTP GET and POST
 - Parameters encoded as HTTP query parameters
- ◆ XML-RPC – Remote Procedure Call using XML
 - Requests use HTTP POST
 - Simple XML payload for request and response
- ◆ SOAP – Simple Object Access Protocol
 - Requests can use "any" transport, but HTTP POST is often used as an example
 - Not-so-simple XML payload for request and response

The basic concept

◆ Request

Message body contains an XML request packet instead of parameters

```
POST /my/resource/ HTTP/1.0
Host: myserver.com
↓
<?xml version="1.0"?>
<xmlrequest>
  <xmltag><morestuff>...</morestuff></xmltag>
</xmlrequest>
```

◆ Response

Message body contains an XML response packet instead of HTML

```
HTTP/1.0 200 OK
Content-Type: text/xml
Content-length: 746
↓
<?xml version="1.0"?>
<xmlresponse>
  <xmltag><morestuff>...</morestuff></xmltag>
</xmlresponse>
```

Why this is interesting...

◆ Arbitrarily complex requests and responses can be layered over HTTP

- XML is a computer-readable data format
- This is just as much distributed computing as CORBA/RMI/DCOM

◆ Existing HTTP servers can handle this traffic

- Provision of vast amounts of data in "computer-readable" form (XML) as well as "human-readable" form (HTML)
- Integration of open services with existing (Web) infrastructure

REST

◆ "Representational State Transfer"

◆ Coined by Roy Fielding in about 2000

Refers to the fact that HTTP is fundamentally stateless --- every transaction carries its state token with it

```
GET /my/resource/?user=JohnR HTTP/1.0
Host: myserver.com
↓
```

```
HTTP/1.0 200 OK
Content-Type: text/xml
Content-length: 746
↓
<?xml version="1.0"?>
<userdata>
  <username>JohnR</username>
  <address>Factory of Engineering</address>
</userdata>
```

REST (2)

- ◆ Services are accessed as resources (URIs)
- ◆ GET reads data only
- ◆ POST modifies data only
- ◆ REST has no standardized XML format, although specific forms are appearing (RSS, Atom)

```
POST /my/resource HTTP/1.0
Host: myserver.com
user=JohnR&address=Factory%20of%20IT
```

```
HTTP/1.0 200 OK
Content-Type: text/xml
Content-length: 254
```

```
<?xml version="1.0"?>
<response status="updateok">
  <username>JohnR</username>
</response>
```

XML-RPC

"Remote Procedure Call"

```
POST /api/RPC2 HTTP/1.0
Host: plant.blogger.com
Content-Type: text/xml
Content-length: 515

<?xml version="1.0"?>
<methodCall>
  <methodName>blogger.newPost</methodName>
  <params>
    <param><value><string>744145</string></value></param>
    <param><value><string>ewilliams</string></value></param>
  </params>
</methodCall>
```

Examples from www.livejournal.com

- ◆ A URI is simply an entry point to a set of methods provided by the service
- ◆ Request body conforms to a simple XML format
- ◆ Parameters are passed in order

XML-RPC (2)

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 125
Content-Type: text/xml
```

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param><value><string>4515151</string></value></param>
  </params>
</methodResponse>
```

- ◆ Response body also conforms to a simple standardized XML format

Examples from www.livejournal.com

SOAP

- ◆ A complex set of specifications (still evolving)
- ◆ HTTP is only a "transport"
- ◆ Emphasis on typing, service definition
- ◆ Commonly generated by higher-level tools (eg .NET Web Services)

```
POST /objectURI HTTP/1.1
Host: www.flickr.com
SOAPMethodName: urn:flickr#flickr.echo
Content-Type: text/xml
Content-Length: 777

<s:Envelope
  xmlns:s=http://www.w3.org/2003/05/soap-envelope
  xmlns:xsi=http://www.w3.org/1999/XMLSchema-instance
  xmlns:xsd="http://www.w3.org/1999/XMLSchema" >
  <s:Body>
    <x:FlickrRequest xmlns:x="urn:flickr">
      <method>flickr.echo</method>
      <name>value</name>
    </x:FlickrRequest>
  </s:Body>
</s:Envelope>
```

Examples from
www.flickr.com

SOAP (2)

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 944
Content-Type: text/xml
```

```
<s:Envelope>
<s:Body>
  <x:FlickrResponse>
    <method>flickr.test.echo</method>
    <format>soap</format>
    <foo>bar</foo>
    <api_key>36f59f1f07d65be29a85196cf3260c1c</api_key>
  </x:FlickrResponse>
</s:Body>
</s:Envelope>
```

- ◆ WSDL describes services (like an IDL for SOAP)
- ◆ UDDI supports registration and discovery of services

Examples from
www.flickr.com

JohnR's crystal ball

- ◆ The field is evolving fast
- ◆ A number of large companies are providing web services APIs
 - Ebay
 - Amazon
 - Yahoo
 - Google
- ◆ In practice, it *seems* that REST is going to prove more popular for open systems, whereas SOAP will probably prove more popular at an enterprise level (alternative to CORBA and DCOM)



Online resources

- ◆ "An overview of HTTP"
<http://icm.mcs.kent.edu/reports/2000/ICM-200009-0006.pdf>
- ◆ "What's This All About, Then?" (HTTP tutorial)
<http://apache-server.com/Podium/show.php?p=HTTP>
- ◆ HTTP 0.9
<http://www.w3.org/Protocols/HTTP/AsImplemented.html>
- ◆ HTTP 1.0
<http://www.w3.org/Protocols/HTTP/HTTP2.html>
- ◆ HTTP 1.1
<http://asq.web.cmu.edu/rfc/rfc2616.html>
- ◆ Overview of REST
http://en.wikipedia.org/wiki/Representational_State_Transfer
- ◆ XML-RPC
<http://www.xmlrpc.com/>
- ◆ SOAP tutorials
<http://www.soaprpc.com/tutorials/>

Concluding remarks

- ◆ Web computing is the most spectacularly successful set of client-server technologies ever, and will stay that way for a while.
- ◆ A wide range of application complexities is supported by current web architecture
- ◆ Emerging web protocols will likely supplant binary protocols such as CORBA and RMI for the majority of large-scale distributed applications

That's all folks!

- ◆ Questions or comments?