CS 580 Client-Server Programming
Fall Semester, 2002
Doc 13 HTTP & XML-RPC

Contents

Doc 13 HTTP & XML-RPC ..............................................................................................................1
Client-Server Protocols .............................................................................................................2
URI ...........................................................................................................................................3
HTML .........................................................................................................................................4
HTTP .........................................................................................................................................6
  HTTP Message Format ..............................................................................................................7
    Client Request .....................................................................................................................8
    Full-Request ......................................................................................................................9
  Server Response ...................................................................................................................11
    Request Methods ..............................................................................................................19
XML-RPC ....................................................................................................................................22

References


Reading

HTTP/1.0 rfc1945

Copyright ©, All rights reserved. 2002 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license defines the copyright on this document.
Client-Server Protocols

Client-Server Protocol defines

• Syntax & semantics of client-server messages

Must define protocol before building client & server

Too learn about protocols we will look at several examples

Starting with HTTP
URI

URI = Uniform Resource Identifiers
URL = Uniform Resource Locator

gopher://gopher.yoyodyne.com/

news:rec.gardening

http://www.yoyodyne.com/pub/foobar.html


Common Internet Scheme Syntax

URL schemes that involve the direct use of an IP-based protocol to a specified host on the Internet use a common syntax for the scheme-specific data:

//--<user>:@<host>:/<url-path>
HTML
Some Buzz Words

WWW
World Wide Web (or Web, for short)

SGML
Standard Generalized Markup Language
this is a standard for describing markup languages

DTD
Document Type Definition
this is a specific markup language, written using SGML

HTML
HyperText Markup Language
HTML is a SGML DTD.

HTML uses markup tags to tell the Web browser how to display the text

XML
Extensible Markup Language

XHTML
XML + HTML 4.0
What is HTML?

HTML is a language for describing structured documents

HTML does not describe page layout

Web browsers use HTML to render & display a document

HTML is content sent by HTTP

<HTML>
<HEAD>
<TITLE>Sample HTML Document</TITLE>
</HEAD>

<BODY>
This is a document
</BODY>
</HTML>
HTTP

- Stateless (http 1.0)
- Object-oriented protocol

    The typing and negotiation of data representation, allows systems to be built independently of the data being transferred

Assigned port 80

Basic Server-Client Interaction (http 1.0)

Client: Open connection
Server: Accept/Reject connection
Client: Send request
Server: Send response to request
Connection closed
HTTP Message Format

HTTP-message = Simple-Request (HTTP/0.9 messages)
  | Simple-Response
  | Full-Request (HTTP/1.0 messages)
  | Full-Response

Full-Request = Request-Line
  *( General-Header | Request-Header | Entity-Header )
  CRLF
  [ Entity-Body ]

Full-Response = Status-Line
  *( General-Header | Request-Header | Entity-Header )
  CRLF
  [ Entity-Body ]

HTTP-header = field-name ":" [ field-value ] CRLF

Entity-Body = *OCTET
Client Request

Request = Simple-Request | Full-Request
Simple-Request = "GET" SP Request-URI CRLF

Simple-Request Example

rohan 11-> telnet www.eli.sdsu.edu 80
Trying 130.191.226.80...
Connected to www.eli.sdsu.edu.
Escape character is '^]'.
GET /courses/fall00/cs580/index.html
<HTML>
<HEAD>
  <TITLE>CS 580: Course Web Site</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">

<TABLE BORDER=0 WIDTH="100%">
  …stuff removed…
</TABLE>

/</sub>Visitors since 21-Aug-00
</center>
</BODY>
</HTML>
Connection closed by foreign host.
Full-Request

Full-Request = Request-Line
              *( General-Header | Request-Header | Entity-Header )
              CRLF
              [ Entity-Body ]

Request-Line = Method SP URI SP HTTP-Version CRLF

rohan 13-> telnet www.eli.sdsu.edu 80
Trying 130.191.226.80...
Connected to www.eli.sdsu.edu.
Escape character is '^]'.
GET /courses/fall00/cs580/index.html HTTP/1.0

HTTP/1.1 200 OK
Date: Tue, 05 Sep 2000 19:31:14 GMT
Server: Apache/1.3.9 (Unix) PHP/3.0.12
Last-Modified: Mon, 04 Sep 2000 21:03:56 GMT
ETag: "14c199-7e8-39b40e3c"
Accept-Ranges: bytes
Content-Length: 2024
Connection: close
Content-Type: text/html
X-Pad: avoid browser bug

<HTML>
<HEAD>
    <TITLE>CS 580: Course Web Site</TITLE>
</HEAD>
… stuff removed here…
Connection closed by foreign host.

Note 2 CRLF are needed to end the full request
HTTP 1.1 Example

rohan 14-> telnet www.eli.sdsu.edu 80
Trying 130.191.226.80...
Connected to www.eli.sdsu.edu.
Escape character is '\]'.
GET /courses/fall00/cs580/index.html HTTP/1.1
Connection: close
Host: www.eli.sdsu.edu

HTTP/1.1 200 OK
Date: Tue, 05 Sep 2000 22:41:26 GMT
Server: Apache/1.3.9 (Unix) PHP/3.0.12
Last-Modified: Mon, 04 Sep 2000 21:03:56 GMT
ETag: "14c199-7e8-39b40e3c"
Accept-Ranges: bytes
Content-Length: 2024
Connection: close
Content-Type: text/html
X-Pad: avoid browser bug

<HTML>
<HEAD>
    <TITLE>CS 580: Course Web Site</TITLE>
</HEAD>
…stuff removed here…
</BODY>
</HTML>
Connection closed by foreign host.
Server Response

Example Full-response

HTTP/1.0 200 Document follows
MIME-Version: 1.0
Server: CERN/3.0
Date: Thursday, 21-Mar-96 17:00:45 GMT
Content-Type: text/html
Content-Length: 2686
Last-Modified: Tuesday, 27-Feb-96 05:34:12 GMT

<table>
<thead>
<tr>
<th>field-name</th>
<th>field-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIME-Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Server:</td>
<td>CERN/3.0</td>
</tr>
<tr>
<td>Date:</td>
<td>Thursday, 21-Mar-96 17:00:45 GMT</td>
</tr>
<tr>
<td>Content-Type:</td>
<td>text/html</td>
</tr>
<tr>
<td>Content-Length:</td>
<td>2686</td>
</tr>
<tr>
<td>Last-Modified:</td>
<td>Tuesday, 27-Feb-96 05:34:12 GMT</td>
</tr>
</tbody>
</table>
What is the big Deal?

Name-Value Pairs

What are the data fields in this?

1.0; CERN/3.0; Thursday, 21-Mar-96 17:00:45 GMT;
text/html; 2686; Tuesday, 27-Feb-96 05:34:12 GMT

What are the data fields in this?

MIME-Version: 1.0
Server: CERN/3.0
Date: Thursday, 21-Mar-96 17:00:45 GMT
Content-Type: text/html
Content-Length: 2686
Last-Modified: Tuesday, 27-Feb-96 05:34:12 GMT

Which is Safer?

Which is Easier to Parse?
Name -Value Pairs are Good

Does Order Matter?

MIME-Version: 1.0
Server: CERN/3.0
Date: Thursday, 21-Mar-96 17:00:45 GMT
Content-Type: text/html
Content-Length: 2686
Last-Modified: Tuesday, 27-Feb-96 05:34:12 GMT

Server: CERN/3.0
Content-Type: text/html
MIME-Version: 1.0
Content-Length: 2686
Last-Modified: Tuesday, 27-Feb-96 05:34:12 GMT
Date: Thursday, 21-Mar-96 17:00:45 GMT
Extending Protocols

MIME-Version: 1.0
Server: CERN/3.0
Date: Thursday, 21-Mar-96 17:00:45 GMT
Content-Type: text/html
Forwaded: by http://rohan.sdsu.edu/ for cs.sdsu.edu
Content-Length: 2686
WhitneyInfo: Hi Mom
Last-Modified: Tuesday, 27-Feb-96 05:34:12 GMT

Cookies were added to HTTP by adding a new name-value pair

Clients/servers that were not programmed for cookies ignored the new name-value pair
Name -Value Pairs are Everywhere

Data Files
Which is easier for a program to parse?
Which is safer

<table>
<thead>
<tr>
<th>name</th>
<th>course</th>
<th>hwork</th>
<th>exam1</th>
<th>exam2</th>
<th>final</th>
<th>as1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allen, Sally</td>
<td>87</td>
<td>92</td>
<td>85</td>
<td>55</td>
<td>74</td>
<td>10</td>
</tr>
<tr>
<td>Battista, Joe</td>
<td>92</td>
<td>98</td>
<td>98</td>
<td>55</td>
<td>78</td>
<td>10</td>
</tr>
<tr>
<td>Biag, Sam</td>
<td>83</td>
<td>91</td>
<td>78</td>
<td>51</td>
<td>72</td>
<td>8</td>
</tr>
<tr>
<td>Chen, Pete</td>
<td>89</td>
<td>92</td>
<td>89</td>
<td>57</td>
<td>79</td>
<td>10</td>
</tr>
<tr>
<td>Chen, Roger</td>
<td>74</td>
<td>68</td>
<td>59</td>
<td>61</td>
<td>55</td>
<td>10</td>
</tr>
</tbody>
</table>

lastName:Allen, lastName:Sally, course:87, hwork:92, exam1:85, exam2:55, final:74, as1:10
lastName:Battista, lastName:Joe, course:92, hwork:98, exam1:98, exam2:55, final:78, as1:10
lastName:Baig, lastName:Sam, course:83, hwork:91, exam1:78, exam2:51, final:72, as1:8
lastName:Chen, lastName:Pete, course:89, hwork:92, exam1:89, exam2:57, final:79, as1:10
lastName:Chen, lastName:Roger, course:74, hwork:68, exam1:59, exam2:61, final:55, as1:10
Name-Value Pairs and Parameters

Most languages use positional matching for parameters

“The cat in the hat came back”.substring(2, 6);

Smalltalk uses name-value pairs

‘The cat in the hat came back’ copyFrom: 2 to: 6

copyFrom:to: is on method

<table>
<thead>
<tr>
<th>Keyword (name)</th>
<th>Parameter (value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>copyFrom:</td>
<td>2</td>
</tr>
<tr>
<td>to:</td>
<td>6</td>
</tr>
</tbody>
</table>
Name -Value Pairs are Your Friends

Don't program without them
Big Issue: In-line data

If we send binary data or data of unknown format how does receiver know when the data ends?

**POP solution**

Use termination sequence

Insure that termination sequence does not occur in data

**HTTP Solution**

Full-Response = Status-Line
*General-Header
*Response-Header
*Entity-Header
CRLF
[ Entity-Body ]

Send length of data to be sent in header
Request Methods

Method = "GET" | "HEAD" | "PUT" | "POST"

| "DELETE" | "LINK" | "UNLINK"
| extension-method

All HTTP/1.0 servers must support GET and HEAD

Servers should return the Status-Code

"501 Not Implemented"

if the method is unknown.

GET

Retrieves whatever item is identified by the URI.

The URI can refer to a data-producing process, or a script

The produced data which shall be returned as the Entity-Body

HEAD

Identical to GET except that the server must not return any Entity-Body in the response
POST

Request that the origin server accept the item enclosed in the request as a new subordinate of the resource identified by the URI

Allows a uniform function to:

- Annotation of existing documents;
- Posting a message to a bulletin board topic, newsgroup, mailing list, or similar group of articles;
- Providing a block of data (usually a form) to a data-handling process, or a script, which can be run by such a process;
- Extending a document during authorship
These are not always supported

Why?

**PUT**

The enclosed item in the request is to be stored under the supplied URI

**DELETE**

Requests that the server delete the resource identified by the given URI

**LINK**

Establishes one or more Link relationships between the existing resource identified by the URI and other existing resources

**UNLINK**

UNLINK method removes one or more Link relationships from the existing resource identified by the URI
XML-RPC Requests

Send via HTTP post

User-Agent & Host must be specified

Content-Type is text/xml

Content-Length must be specified and be correct in bytes

Content is in XML and contains single <methodCall>

Example

POST /cs580 HTTP/1.1
Host: rugby.sdsu.edu:8008
Content-length: 158
Content-type: text/xml;charset=iso-8859-1
User-Agent: Smalltalk XMLRPC version 0.5 (VisualWorksÆ NonCommercial, Release 7 of June 14, 2002)
Connection: keep-alive

<?xml
version="1.0"?><methodCall><methodName>officeDataFor</methodName><params>
  <param><value><string>Whitney</string></value></param>
</params></methodCall>
XML in Requests

```xml
<methodCall>
  <methodName>[Character]*</methodName>
  <params>
    <param><value>[Legal type]</value></param>*
  </params>
</methodCall>
```

Characters – string of characters

**Legal types**

- `<i4>` or `<int>`
- `<boolean>`
- `<string>`
- `<double>`
- `<dateTime.iso86021>`
- `<base64>`

```xml
<struct>
  <member>
    <name>[Character]*</name>
    <value>[Legal type]</value>
  </member>*
</struct>
```

```xml
<array>
  <data>
    <value>[Legal type]</value>
  </data>*
</array>
```
Response

Unless lower-level error always return 200 OK

Content type is text/xml

Content-length must be present and be correct

Body contains on XML structure <methodResponse>

Example

HTTP/1.0 200 OK
Date: Tue, 8 Oct 2002 23:27:43 -0700
Server: VisualWave TinyHTTP/1.0
Mime-version: 1.0
Content-type: text/xml;
Content-length: 252

<?xml version="1.0"?><methodResponse><params><param><value><array><data>
  <value><string>Whitney</string></value>
  <value><string>P-243</string></value>
  <value><string>594-3535</string></value>
</data></array></value></param></params></methodResponse>
Response XML

Response contains one `<methodResponse>` tag

The `<methodResponse>` tag can contain either
- `<params>`
- `<fault>`

`<params>` tag can only contain one `<param>`

**methodResponse Structure**

```xml
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>[Legal type]</value>
    </param>
  </params>
</methodResponse>

<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value>[Legal type]</value>
  </fault>
</methodResponse>
```