RESTful Web Services

What are REST Services?

RESTful Web Services

- Data-structure oriented
  - Implied behavior through HTTP METHOD types
  - Not “strongly typed”; assumes client understands data
  - Data may be sent in XML, JSON, or other forms
- REST interactions are performed using URLs
  - Following general HTTP request structures (headers, parameters, cookies, etc.)
  - Focused on invoking / accessing resources
  - REST WS should be cacheable

Composition of HTTP

- Basic format of HTTP request / response
  - *Initial Line*
  - *Headers*
  - Blank Line
  - *Message Body*
  - Blank line

Initial Request Line

- Single line containing space delimited fields
- Used to describe request
  - HTTP method
  - Resource
    - Translated into path starting from *server root* (/)
    - May contain a *query string* depending on Method type
  - Protocol version
    - HTTP 1.0
    - HTTP 1.1

HTTP Methods

- GET
  - Retrieve information specified in URI
  - Server returns Headers and Message Body
  - Responses are cacheable
  - Typically result of an `<A href>`
  - Extra URI information passed as a query string
- POST
  - “Send” information to specified URI
  - Asks server to accept information as subordinate of specified URI
  - Information is “handled” by specified URI
  - Responses are not cacheable
  - Typically result of `<FORM>`
  - Post data sent as entity

HTTP Methods (cont.)

- PUT
  - “Send” information to specified URI
  - Server stores information as a resource under specified URI
  - Can overwrite existing content
- DELETE
  - Requests specified resource be deleted
  - Typically generates a success or a fail response

- Proposed: PATCH
  - Currently under discussion
  - Variation of PUT intended to modify only part of the resource
    - PUT overwrites the entire resource
    - PATCH modifies the elements included in the entity

HTTP Methods (cont.)

- HEAD
  - Like GET; retrieves information
  - Retrieves meta-information; server returns only Headers
- OPTIONS
  - Retrieve information about the communication options available
  - May be used by browser to determine best interaction mechanism
- TRACE
  - Similar to trace route
  - Used to determine what information is received along request chain

REST Interactions with HTTP

- HTTP methods create database like access
  - POST - Creates resource
  - GET - Reads resource
  - PUT - Updates resource
  - DELETE - Deletes resource

Example Request Initial Lines

- Accessing top-level domain
  - **http://www.host.com**
  - `GET / HTTP/1.1`
- Accessing sub-resource
  - **http://www.host.com/LEARN/j2se_overview.pdf**
  - `GET /LEARN/j2se_overview.pdf HTTP/1.1`
- Accessing dynamic resource using GET
  - **http://www.google.com/search?q=rest**
  - `GET /search?q=rest HTTP/1.1`

Initial Response Line

- Single line containing space-delimited fields
- Used to describe response
  - Protocol version
    - HTTP 1.0
    - HTTP 1.1
  - Response status (*status code*)
    - Numeric value
    - Standard or proprietary
  - Description
    - Human readable description of status code
- Five main categories of responses
  - *Information*
  - *Success*
  - *Redirection*
  - *Client Error*
  - *Server Error*

Informational Responses

- Provisional response; typically used for handshaking or negotiating
- Common status codes
  - `100` - Tells client to continue with request
  - `101` - Tells client server is trying to adopt client-suggested protocol
- Consists of
  - Initial line (*status line*)
  - Headers
  - Empty line

Successful Responses

- Signifies client request was received, understood, and accepted by server
- Common status codes
  - `200` - Tells client request succeed; response body is sent depending on type of method
  - `202` - Tells client server accepted request; but processing of request has not been completed
- Consists of
  - Status line
  - Headers
  - Empty line
  - Response body*

Redirection Responses

- Used to notify client further action is required to fulfill request
- Typically used to notify client to access resource in a different location
- Common status code
  - `301` - Resource permanently relocated
  - `307` - Temporary redirect of resource; also known as client-side redirect
- Consists of
  - Status line
  - Headers
  - Empty line

Client Error Responses

- Used to notify client that the request contained an error
- Error could be
  - `400` - Bad request; malformed request
  - `401` - Unauthorized access; the request was missing the authorization header
  - `403` - Attempted forbidden; no resolution
  - `404` - Resource specified in URI was not found
  - `405` - Method not supported
- Typically consists of
  - Status line
  - Headers
  - Empty line

Server Error Responses

- Used to notify client that server encountered an error in processing the request
- Server errors could be
  - `500` - Internal, irresolvable, unexpected problem
  - `503` - Service unavailable typically as a result of load issues
  - `505` - HTTP version not supported
- Consists of
  - Status line
  - Headers
  - Empty line

Example Response Initial Lines

- Accessing top-level domain
  - **http:// www.host.com**
  - `HTTP/1.1 200 OK`
- Accessing missing sub-resource
  - **http:// www.host.com/LEARN/dot-net_overview.pdf**
  - `HTTP/1.1 404 NOT FOUND`
- Accessing redirected resource
  - **http://www.javasoft.com**
  - `HTTP/1.1 301 Moved Permanently`

HTTP Headers

- Used to describe meta-information
  - Describe information about client capabilities
  - Describe information about server response
- Used for both requests and responses
- Represented as name/value pairs
- Follows this syntax:
  - *HeaderName: HeaderValue*
  - *HeaderValue* typically contains text data
- May use custom headers

Request Headers

- `Accept`
  - Notifies server type of data client can handle
  - Can be repeated
  - Contains comma separate list of mime-types; may use wildcards
- `Accept-Encoding`
  - Notifies server type of data encoding client can handle
  - Used for things like Zip compression
- `Accept-Language` - Desired response language
- `Host` - Specifies Internet host and port of requested resource
- `User-Agent`
  - Describes client software to server
  - Used for tracking purposes; commonly written in HTTP access logs
  - Used to generate browser specific HTML
- `Referrer` - Notifies server of where the request originated

Response Headers

- `Age`
  - Designation (estimate) of time since response was generated
  - Used with proxies
- `Server` - Describes server software used
- `Location` - Describes location client should use to fulfill request

General Purpose Headers

- Applicable to both request and response
- Secondary meta-information about interaction
- `Cache-Control`
  - Describes caching directives
  - Used by clients and proxies
- `Date` - General purpose date
- `Upgrade`
  - Used by client on request to tell server what protocol it would like to use
  - Used by server to notify of which protocols are changing

Entity Headers

- Headers used to describe entity (*message body*)
- Can be used in request and / or response
- `Allow` - Specifies which HTTP methods are supported
- `Content-Language` - Natural language of content
- `Content-Length` - Length of content
- `Content-Type`
  - Describes type of content
  - Usually MIME type representation
- `Expires` - When content is considered out of date
- `Last-Modified` - Last modification date of content

Client Request Example

- Example of client HTTP request
  - Initial Line
  - Headers
- URI used for request **http:// www.host.com**

```
GET / HTTP/1.1
Host:  www.host.com
Accept-Encoding: gzip
Accept: */*
Accept-Language: en-us, ja;q=0.62, de-de;q=0.93, de;q=0.90, fr-
fr;q=0.86, fr;q=0.83, nl-nl;q=0.79, nl;q=0.76, it-it;q=0.72,
it;q=0.69, ja-jp;q=0.66, en;q=0.97, es-es;q=0.59, es;q=0.55, da-
dk;q=0.52, da;q=0.48, fi-fi;q=0.45, fi;q=0.41, ko-kr;q=0.38
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en-us)
AppleWebKit/125.5.5 (KHTML, like Gecko) Safari/125.11 Web-Sniffer/
1.0.17
```

Server Response Example

- Example of HTTP response
  - Initial Line
  - Headers
- Sent from http:// www.host.com

```
HTTP/1.1 200 OK
Date: Tue, 16 Nov 2004 03:16:22 GMT
Server: Apache/1.3.28 (Unix) mod_watch/3.12 PHP/4.3.2 mod_ssl/2.8.15
OpenSSL/0.9.7b
X-Powered-By: PHP/4.3.2
Content-Type: text/html
```

HTTP Entity

- Considered body or payload of
  - Request
  - Response
- Entity contents may or may not exist
  - Depends on Method
  - Depends on type of response
- Content headers assist in
  - Determining length of entity
  - Type of entity
  - Encoding of entity
- Server may
  - Take action against entity (process it)
  - Apply it (store it)
  - Ignore it
- Client may
  - Take action against entity (render it)
  - Apply it (store it)
  - Ignore it

GET Request - Entity Example

- Client requesting **http://www.google.com/search?q=rest**
- Entity is empty

```
GET /search?q=rest HTTP/1.1
Host: www.google.com
Accept-Encoding: gzip
Accept: */*
Accept-Language: en-us, ja;q=0.62, de-de;q=0.93, de;q=0.90, fr-
   fr;q=0.86, fr;q=0.83, nl-nl;q=0.79, nl;q=0.76, it-it;q=0.72,
   it;q=0.69, ja-jp;q=0.66, en;q=0.97, es-es;q=0.59, es;q=0.55,
   da-dk;q=0.52, da;q=0.48, fi-fi;q=0.45, fi;q=0.41, ko-kr;q=0.38
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en-us)
   AppleWebKit/125.5.5 (KHTML, like Gecko) Safari/125.11 Web-
   Sniffer/1.0.17
Content-type: application/x-www-form-urlencoded
Content-length: 7
```

<entity is empty>

Post Request - Entity Example

- Client requested **http://www.google.com/search?q=rest**
- Entity contains `q=john`

```
POST /search HTTP/1.1
Host: www.google.com
Accept-Encoding: gzip
Accept: */*
Accept-Language: en-us, ja;q=0.62, de-de;q=0.93, de;q=0.90, fr-
   fr;q=0.86, fr;q=0.83, nl-nl;q=0.79, nl;q=0.76, it-it;q=0.72,
   it;q=0.69, ja-jp;q=0.66, en;q=0.97, es-es;q=0.59, es;q=0.55,
   da-dk;q=0.52, da;q=0.48, fi-fi;q=0.45, fi;q=0.41, ko-kr;q=0.38
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en-us)
   AppleWebKit/125.5.5 (KHTML, like Gecko) Safari/125.11 Web-
   Sniffer/1.0.17
Content-type: application/x-www-form-urlencoded
Content-length: 7

q=john
```

Response - Entity Example

- Client requested **http://www.google.com/search?q=rest**

```
HTTP/1.0 200 OK
Content-Type: text/html
Server: GWS/2.1
Date: Wed, 17 Nov 2004 04:09:10 GMT
Connection: Close


<html><head><meta HTTP-EQUIV="content-type" CONTENT="text/html; charset=ISO-8859-1"><title>Google Search: rest </title><style><
!--
body,td,div,.p,a{font-family:arial,sans-serif }
div,td{color:#000}
.f,.fl:link{color:#6f6f6f}
a:link,.w,a.w:link,.w a:link{color:#00c}
a:visited,.fl:visited{color:#551a8b}
a:active,.fl:active{color:#f00}
.t a:link,.t a:active,.t a:visited,.t{color:#000}
.t{background-color:#e5ecf9}
.k{background-color:#36c}
.j{width:34em}
.h{color:#36c}
.i,.i:link{color:#a90a08}
.a,.a:link{color:#008000}
.z{display:none}
div.n {margin-top: 1ex}
.n a{font-size:10pt; color:#000}
.n .i{font-size:10pt; font-weight:bold}
.q a:visited,.q a:link,.q a:active,.q {color: #00c; }
.b{font-size: 12pt; color:#00c; font-weight:bold}
.ch{cursor:pointer;cursor:hand}
.e{margin-top: .75em; margin-bottom: .75em}
.g{margin-top: 1em; margin-bottom: 1em}
//--><script>
<!--
function ss(w){window.status=w;return true;}
function cs(){window.status='';}
function ga(o,e) {return true;}
//-->
</script>
```