Telerik Academy

HTTP and AJAX

WWW, HTTP, AJAX, JSONP

Telerik Software Academy

Learning & Development Team

http://academy.telerik.com

Telerik Academy

Table of Contents

Telerik Academy

WWW and URL

What is WWW? What is URL?

What is WWW?

- WWW = World Wide Web = Web
  - Global distributed information system in Internet
    - A service in Internet (like E-mail, DNS, …)
  - Consists of set of documents (and other resources) located on different Internet servers
    - Accessed through standard protocols like HTTP, HTTPS and FTP by their URL
  - Web servers provide Web content
  - Web browsers display the Web content

WWW Components

- Structural components
  - Internet – provides data transfer channels over the TCP and HTTP protocols
  - Clients (Web browsers) – display Web content
  - Web servers – IIS, Apache, Tomcat, GWS, etc.
- Semantic components
  - Hyper Text Transfer Protocol (HTTP)
  - Hyper Text Markup Language (HTML)
  - Uniform Resource Locator (URL)
    - Uniform Resource Identifiers (URIs)

WWW Infrastructure

- Clients use Web browser application to request resources from the Web servers via HTTP
  - Resources have unique URL address
- Servers send the requested resource as a response
  - Or reply with an error message
- Web pages are resources in WWW
  - HTML text, graphics, animations and other files
- Web sites
  - Web sites are sets of Web pages in WWW

WWW Infrastructure (2)

- Client's browser renders Web pages returned by the Web servers
  - Pages are in HTML (Hyper Text Markup Language)
  - Browsers show the text, graphics, sounds, etc.
  - HTML pages contain hyperlinks to other pages
- The entire WWW system runs over standard networking protocols
  - TCP, DNS, HTTP, FTP, …
- The HTTP protocol is fundamental for WWW

URL

- Uniform Resource Locator (URL)
  - Unique resource location in WWW, e.g.

```
http://yoursite.com:8080/path/index.php?id=27&lang=en
```

- It is just a formatted string, consisting of:
  - Protocol for communicating with the server (e.g., `http`, `ftp`, `https`, …)
  - Name of the server or IP address + optional port (e.g. `www.telerik.com`, `mail.bg:8080`)
  - Path and name of the resource (e.g. `index.php`)
  - Parameters (optional, e.g. `?id=27&lang=en`)

URL Encoding

- URLs are encoded according RFC 1738:

```
"... Only alphanumeric [0-9a-zA-Z], the special
characters $-_.+!*'() and reserved characters used
for their reserved purposes may be used unencoded
within an URL."
```

- All other characters are escaped with the formula:

```
%[character hex code in ISO-Latin character set]
```

  - Example: space has decimal code 32, in hex – 20, so space in URL becomes %20
  - Space can also be encoded as "+"

URL – Examples

- Some valid URLs:

```
http://www.google.bg/search?sourceid=navclient&ie=
UTF-8&rlz=1T4GGLL_enBG369BG369&q=http+get+vs+post
```

```
http://bg.wikipedia.org:80/wiki/%D0%A2%D0%B5%D0%BB
%D0%B5%D1%80%D0%B8%D0%B3
```

- Some invalid URLs:

Should be: ?q=C%23+.NET+4.0

```
http://www.google.bg/search?&q=C# .NET 4.0
```

Should be: ?q=%D0%B1%D0%B8%D1%80%D0%B0

```
http://www.google.bg/search?&q=бира
```

Telerik Academy

HTML, XML, JSON, RSS

Comparing the Common Web Data Formats

HTML

- Hyper Text Markup Language (HTML)
  - Notation for describing formatted text with images and hyperlinks
  - Interpreted and displayed by the Web browsers
- A Web (HTML) page consists of:
  - HTML file
  - CSS stylesheet file (optional)
  - A bunch of images (optional)
  - Other resources (optional)

HTML

- HTML is straight-forward and easy to learn
  - HTML documents are plain text files
    - Easy to add formatting, hyperlinks, bullets, etc.
    - Images can be added as separate files
  - Can be automatically generated by authoring programs
    - Tools to help users creating HTML pages
    - E.g. FrontPage, Dreamweaver, Visual Studio
    - WYSIWYG HTML editors

HTML – Example

```
<html>
  <head><title>HTML Example</title></head>
  <body>
    <h1>Heading 1</h1>
    <h2>Sub heading 2</h2>
    <h3>Sub heading 3</h3>
    <p>This is my first paragraph</p>
    <p>This is my second paragraph</p>
    <div align="center"
      style="background:skyblue">
      This is a div</div>
  </body>
</html>
```

XML

- XML is markup-language for encoding documents in machine-readable form
  - Text-based format
  - Consists of tags, attributes and content
  - Provide data and meta-data in the same time

```
<?xml version="1.0"?>
<library>
  <book><title>HTML 5</title><author>Bay Ivan</author></book>
  <book><title>WPF 4</title><author>Microsoft</author></book>
  <book><title>WCF 4</title><author>Kaka Mara</author></book>
  <book><title>UML 2.0</title><author>Bay Ali</author></book>
</library>
```

RSS

- RSS (Really Simple Syndication)
  - Family of Web feed formats for publishing frequently updated works
    - E.g. blog entries, news headlines, videos, etc.
  - Based on XML, with standardized XSD schema
- RSS documents (feeds) are list of items
  - Each containing title, author, publish date, summarized text, and metadata
- Atom protocol aimed to enhance / replace RSS

RSS – Example

```
<?xml version="1.0" encoding="utf-8" ?>
<rss version="2.0">
<channel>
  <title>W3Schools Home Page</title>
  <link>http://www.w3schools.com</link>
  <description>Free web building tutorials</description>
  <item>
    <title>RSS Tutorial</title>
    <link>http://www.w3schools.com/rss</link>
    <description>New RSS tutorial on W3Schools</description>
  </item>
  <item>
    <title>XML Tutorial</title>
    <link>http://www.w3schools.com/xml</link>
    <description>New XML tutorial on W3Schools</description>
  </item>
</channel>
</rss>
```

JSON

- JSON (JavaScript Object Notation)
  - Standard for representing simple data structures and associative arrays
  - Lightweight text-based open standard
  - Derived from the JavaScript language


```
{
  "firstName": "John", "lastName": "Smith", "age": 25,
  "address": { "streetAddress": "33 Alex. Malinov Blvd.",
     "city": "Sofia", "postalCode": "10021" },
  "phoneNumber": [{ "type": "home", "number": "212 555-1234"},
    { "type": "fax", "number": "646 555-4567" }]
},
{ "firstName": "Bay", "lastName": "Ivan", "age": 79 }
```

Telerik Academy

The HTTP Protocol

How HTTP Works?

HTTP

- Hyper Text Transfer Protocol (HTTP)
  - Client-server protocol for transferring Web resources (HTML files, images, styles, etc.)
- Important properties of HTTP
  - Request-response model
  - Text-based format
  - Relies on unique resource URLs
  - Provides resource metadata (e.g. encoding)
  - Stateless (cookies can overcome this)

Telerik Academy

HTTP: Request-Response Protocol

- Client program
  - Running on end host
  - E.g. Web browser
  - Requests a resource
- Server program
  - Running at the server
  - E.g. Web server
  - Provides resources

Telerik Academy

Example: Hyper Text Transfer Protocol

- HTTP request:

```
GET /academy/about.aspx HTTP/1.1
Host: www.telerik.com
User-Agent: Mozilla/5.0
<CRLF>
```

The empty line denotes the end of the request header

- HTTP response:

```
HTTP/1.1 200 OK
Date: Mon, 5 Jul 2010 13:09:03 GMT
Server: Microsoft-HTTPAPI/2.0
Last-Modified: Mon, 12 Jul 2010 15:33:23 GMT
Content-Length: 54
<CRLF>
<html><title>Hello</title>
Welcome to our site</html>
```

The empty line denotes the end of the response header

HTTP Request Message

- Request message sent by a client consists of
  - Request line – request method (GET, POST, HEAD, …), resource URI, and protocol version
  - Request headers – additional parameters
  - Body – optional data
    - E.g. posted form data, files, etc.

```
<request method> <resource> HTTP/<version>
<headers>
<empty line>
<body>
```

Telerik Academy

HTTP GET Request – Example

- Example of HTTP GET request:

```
GET /academy/winter-2009-2010.aspx HTTP/1.1
Host: www.telerik.com
Accept: */*
Accept-Language: bg
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0(compatible;MSIE 6.0;
Windows NT 5.0)
Connection: Keep-Alive
Cache-Control: no-cache
<CRLF>
```

HTTP request line

HTTP headers

The request body is empty

Telerik Academy

HTTP POST Request – Example

- Example of HTTP POST request:

```
POST /webmail/login.phtml HTTP/1.1
Host: www.abv.bg
Accept: */*
Accept-Language: bg
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0(compatible;MSIE 6.0;
Windows NT 5.0)
Connection: Keep-Alive
Cache-Control: no-cache
Content-Length: 59
<CRLF>
LOGIN_USER=mente
DOMAIN_NAME=abv.bg
LOGIN_PASS=top*secret!
<CRLF>
```

HTTP request line

HTTP headers

The request body contains the submitted form data

Telerik Academy

Conditional HTTP GET – Example

- Example of HTTP conditional GET request:

```
GET /academy/join.aspx HTTP/1.1
Host: www.telerik.com
User-Agent: Gecko/20100115 Firefox/3.6
If-Modified-Since: Tue, 9 Mar 2010 11:12:23 GMT
<CRLF>
```

- Fetches the resource only if it has been changed at the server
  - Server replies with "`304 Not Modified`" if the resource has not been changed
  - Or "`200 OK`" with the latest version otherwise

HTTP Response Message

- Response message sent by the server
  - Status line – protocol version, status code, status phrase
  - Response headers – provide meta data
  - Body – the contents of the response (the requested resource)

```
HTTP/<version> <status code> <status text>
<headers>
<CRLF>
<response body – the requested resource>
```

HTTP Response – Example

- Example of HTTP response from the Web server:

```
HTTP/1.1 200 OK
Date: Fri, 17 Jul 2010 16:09:18 GMT+2
Server: Apache/2.2.14 (Linux)
Accept-Ranges: bytes
Content-Length: 84
Content-Type: text/html
<CRLF>
<html>
  <head><title>Test</title></head>
  <body>Test HTML page.</body>
</html>
```

HTTP response status line

HTTP response headers

The HTTP response body

HTTP Response – Example

- Example of HTTP response with error result:

```
HTTP/1.1 404 Not Found
Date: Fri, 17 Jul 2010 16:09:18 GMT+2
Server: Apache/2.2.14 (Linux)
Connection: close
Content-Type: text/html
<CRLF>
<HTML><HEAD>
<TITLE>404 Not Found</TITLE>
</HEAD><BODY>
<H1>Not Found</H1>
The requested URL /img/telerik-logo.gif was not
found on this server.<P>
<HR><ADDRESS>Apache/2.2.14 Server at Port
80</ADDRESS>
</BODY></HTML>
```

Response status line

HTTP response headers

The HTTP response body

Telerik Academy

Content-Type and Disposition

- The Content-Type header at the server specifies how the output should be processed
- Examples:

UTF-8 encoded HTML page.
Will be shown in the browser.

```
Content-Type: text/html; charset=utf-8
```

```
Content-Type: application/pdf
Content-Disposition: attachment;
  filename="Financial-Report-April-2010.pdf"
```

This will download a PDF file named
`Financial-Report-April-2010.pdf`

HTTP Request Methods

- HTTP request methods:
  - GET
    - Return the specified resource, run a program at the server, or just download file, …
  - HEAD
    - Return the meta-data associated with a resource (headers only)
  - POST
    - Update a resource, provide input data for processing at the server, …

HTTP Response Codes

- HTTP response code classes
  - 1xx: informational (e.g., "100 Continue")
  - 2xx: success (e.g., "200 OK")
  - 3xx: redirection (e.g., "304 Not Modified", "302 Found")
  - 4xx: client error (e.g., "404 Not Found")
  - 5xx: server error (e.g., "503 Service Unavailable")
- "302 Found" is used for redirecting the Web browser to another URL

Browser Redirection

- HTTP browser redirection example
  - HTTP GET requesting a moved URL:

```
GET / HTTP/1.1
Host: academy.telerik.com
User-Agent: Gecko/20100115 Firefox/3.6
<CRLF>
```

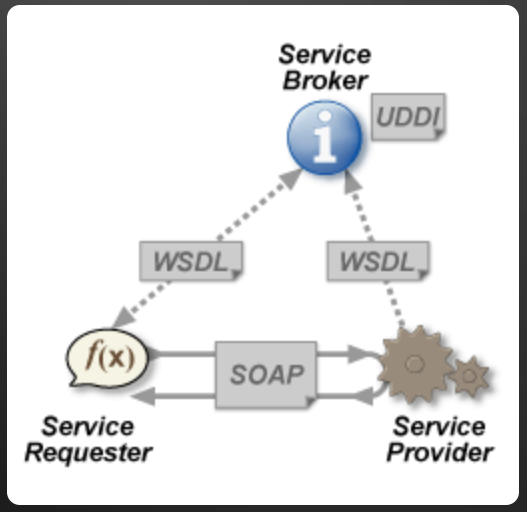
  - The HTTP response says the browser should request another URL:

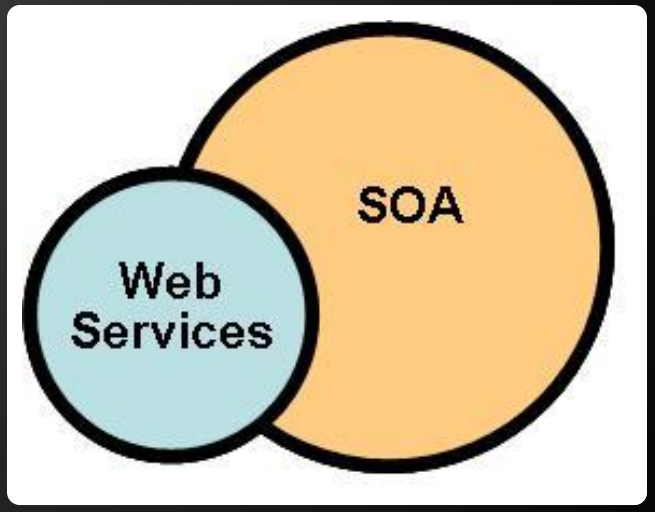
```
HTTP/1.1 301 Moved Permanently
Location: http://www.telerik.com/academy/
…
```

Telerik Academy

Web Services

Web Services and SOA Architecture

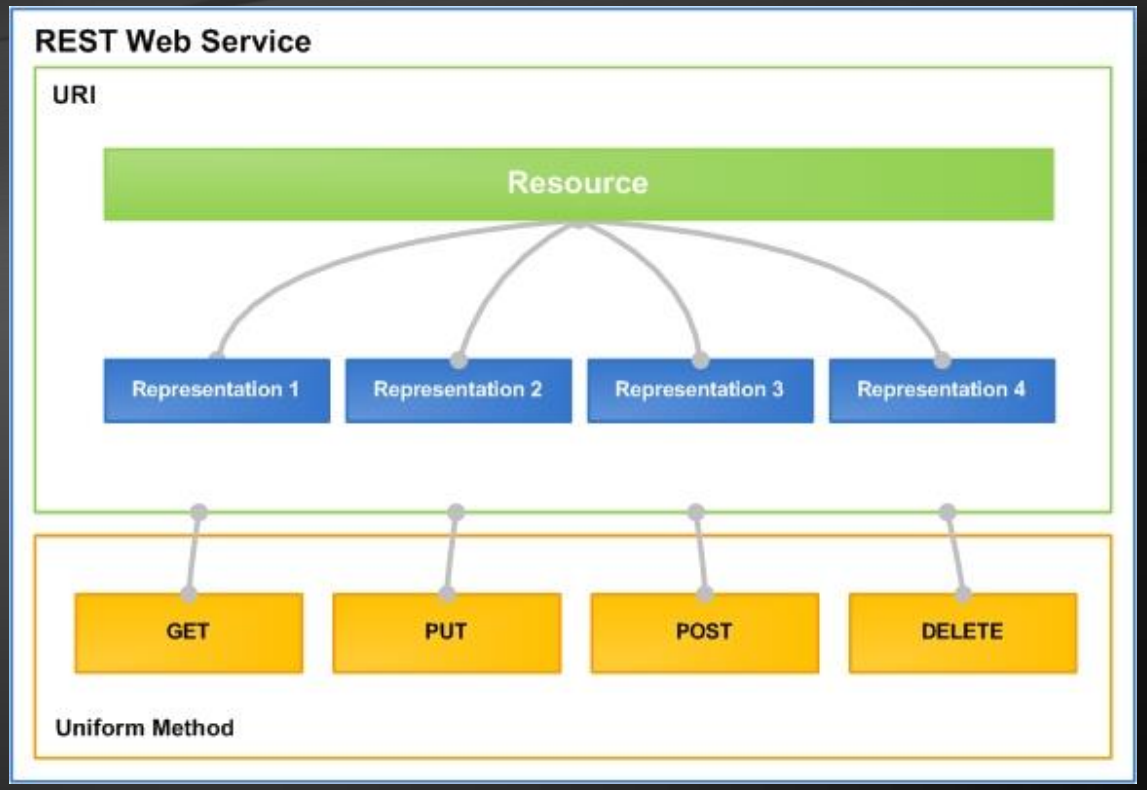




Web Services

- A web service is a method of communication between two devices in WWW
  - The server device exposes services
  - The client consumes these services
- Web services are a main part of the SOA architecture
  - Database and Business logic on the server
    - The server exposes public services
  - UI logic on the client
    - Consumes these services

Telerik Academy



RESTful Web Services

Lightweight Architecture for Web Services

What is REST?

> "Representational state transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web."

http://en.wikipedia.org/wiki/Representational_State_Transfer

- Application state and functionality are resources
  - Resources are used as common data files
- Every resource has an URI
- All resources share a uniform interface
- This natively maps to the HTTP protocol

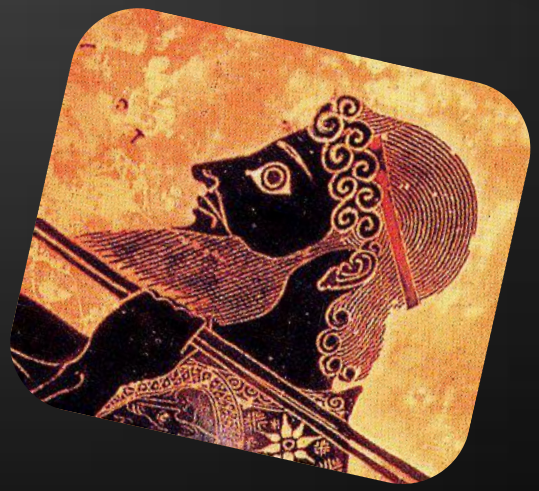
RESTful Services

- One URI for a resource, multiple operations
  - Add a new document "RestTalk" in category "Code"
    - PUT http://mysite.com/docs/Code/RestTalk
  - Get the document / some page
    - GET http://mysite.com/docs/Code/RestTalk
    - GET http://mysite.com/docs/Code/RestTalk/pages/3
  - Remove the document
    - DELETE http://mysite.com/docs/Code/RestTalk
  - Retrieve metadata
    - HEAD http://mysite.com/docs/Code/RestTalk

Telerik Academy

AJAX

Asynchronous JavaScript and XML



AJAX

- AJAX is acronym of Asynchronous JavaScript and XML
  - Technique for background loading of dynamic content and data from the server side
  - Allows dynamic client-side changes
- Two types of AJAX
  - Partial page rendering – loading of HTML fragment and showing it in a `<div>` (AHAH)
  - JSON service – loading JSON object and client-side processing it with JavaScript / jQuery

AJAX

- Technically, AJAX is a group of technologies working together
  - HTML & CSS for presentation
  - The DOM for data display & interaction
  - XML (or JSON) for data interchange
  - XMLHttpRequest for async communication
  - JavaScript to use the above

AJAX

- AJAX uses HTTP
  - Requests have headers – GET, POST, HEAD, etc.
  - Requests have bodies – XML, JSON or plain text
  - The request must target a resource with a URI
  - The resource must understand the request
    - Server-side logic
  - Requests get a HTTP Response
    - Header with a body

The Same Origin Policy

i.e. Don't talk to strangers

Same Origin Policy

- Security concept for browser-side programming languages
- Scripts running on pages from the same site
  - i.e. the same origin
  - Can access each other without restriction
- Scripts cannot access pages on different sites
- This also applies to XMLHttpRequest
  - Sent only between pages with same origin

Origin Determination Rules

- Origin is defined using
  - Domain name (e.g. example.com)
  - Application layer protocol (e.g. http)
  - Port number (not all browsers!)
  - Two resources are of the same origin if all of the above match

Telerik Academy

Origin Determination Example

- Say we have a resource at http://www.example.com/dir/page.html
- The following table shows outcomes of origin checks with resources at similar URLs

Compared URL	Outcome	Reason
http://www.example.com/dir/page2.html	Success	Same protocol and host
http://www.example.com/dir2/other.html	Success	Same protocol and host
http://www.example.com:**81**/dir/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir/other.html	Failure	Different protocol
http://**en.example.com**/dir/other.html	Failure	Different host
http://**example.com**/dir/other.html	Failure	Different host (exact match required)
http://**v2.www.example.com**/dir/other.html	Failure	Different host (exact match required)
http://www.example.com:**80**/dir/other.html	Don't use	Port explicit. Depends on implementation in browser.

Relaxing Same Origin Policy

- Same origin policy is sometimes too restrictive
  - Large sites with lots of subdomains
  - Accessing web services
- Ways of "relaxing"
  - document.domain – can be set to a superdomain when in proper subdomain
  - Cross document messaging – HTML5, postMessage() to page in <iframe>
  - Cross Origin Resource Sharing
- Workaround – JSONP

JSONP

i.e. not listening to mommy

JSONP

- JSON with padding (also JSON prefix)
- Same origin policy denies cross-origin requests
  - But not for the <script> tag
  - The <script> tag can be exploited
- Retrieve JS code and the browser executes it
  - In the case of a service, we get a JSON object
  - The script tag can have a callback

```
<script type="text/javascript"
src="http://server2.example.com/Users/1234?
jsonp=parseResponse"></script>
```

  - We receive parseResponse(…)

JSONP – How it Works

- After the script URL, we add a query parameter ?jsonp= (or ?callback=)
  - This parameter tells the server what to return
  - The server wraps its return value in the specified callback
  - Example
    - server returns a JSON object {"age":"5"}
    - the query parameter is ?callback=parseResponse
    - The browser executes the following JS code:

```
parseResponse({"age":"5"})
```

JSONP – the "Padding"

- The callback function in the example is called the "padding"
- Typically its used to pass the JSON to a function, which acts as a handler, but it can be anything else
  - Variable assignment, if statement, etc.
  - What we receive is JS code, not JSON
    - This is a security concern

Telerik Academy

Web Developer Tools

Simulating and Tracking Web Traffic

Web Developer Tools

- Firebug plug-in for Firefox
  - A must have for Web developers
  - The ultimate tool for monitoring, editing and debugging HTTP, HTML, CSS, JavaScript, etc.
  - Free, open-source – www.getfirebug.com
- Fiddler – HTTP proxy
  - Intercepts the HTTP traffic
  - Analyzes the HTTP conversation
  - Free tool (by Telerik) – http://www.telerik.com/fiddler

Web Developer Tools (2)

- Web Developer Tools
  - Built-in in Google Chrome and Opera
  - Network requests logging
  - Code execution timeline
- Postman
  - Google app
  - Perform HTTP requests
  - Get from Chrome web store

Telerik Academy

HTTP and AJAX

http://schoolacademy.telerik.com

Free Trainings @ Telerik Academy

- "C# Programming @ Telerik Academy
  - csharpfundamentals.telerik.com

- Telerik Software Academy
  - academy.telerik.com

- Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy

- Telerik Software Academy Forums
  - forums.academy.telerik.com