Web Services Basics for Non-Programmers

NobleProg

Web Services Basics for Non-Programmers

Contents

Service Oriented Architecture

- Service Oriented Architecture (SOA) is a software architectural style
    - It uses services that are available in a network such as the web to build applications
    - Services are implementations of units of well defined business functionality
    - Web Services are a particular implementation of an SOA architecture
        - Web Services use open standards such as
            - TCP/IP - Transmission Control Protocol and Internet Protocol
            - HTTP - HyperText Transfer Protocol, the foundation for communication on the World Wide Web
            - XML - Extensible Markup Language, a language for defining data
            - SOAP - Simple Object Access Protocol, a specification for exchanging data

References

Oracle on Service Oriented Architechure (SOA) and Web Services

SOA Governance

- Generally it is necessary to exercise control over web services
    - The functions of each service
    - prevention of duplicate services
    - Approval of new Services
    - Documentation of the services available beyond the simple WSDL description
        - Written documentation describing the services in detail
        - Policies regarding SOA (remembering that web services are part of SOA)

        - Service level agreements between groups
        - Responsibilities of managers and development groups
    - Prevention of the proliferation of unused or unusable web services
    - Standards for web services
    - Decision making authorities
    - There are software application built to help with these tasks
    - In many organizations these processes are ad-hoc and instantiate as the need occurs
        - It is useful to define the processes purposefully if there are going to be many web services

References

Wikipedia on SOA Governance
IBM on SOA Governance

Introduction to Web Services

- Web Services are used to exchange data between web applications

    - World Wide Web Consortium (W3C) the international open standard organization defines a web service as:

A software system designed to support interoperable machine-to-machine interaction over a network

- Generally the data exchange is focused on a particular need, small and contained
- The returned data is usually standardized as either XML or JSON
    - XML and JSON are well known formats for exchanging data
    - However, other formats can be used and could be proprietary
- A Web Server generally has several components such as
    - Some logic on a web server that returns information and may manipulate that information before returning it
        - Examples would be
            - Currency calculating components that return various currency values
            - Interest calculating components that return interest rates or dollar amounts
            - Components that return savings or checking values
            - Components that check logging in user information
    - An interface that describes how to put a request to the service and what is expected as the result
    - A method of connecting to the service to make a request and receive a response
        - In general this requires client and server side programming of components

Why Do We Need Web Services

- Organizations usually have multiple software systems
    - Web Services allow the sharing of data between these systems
- Larger systems are distributed, data exists in many locations
- A markup language is need to pass data so it can be understood
- Standardized protocols are used making it less expensive to implement sharing
- Low cost since it uses the normal web protocol (HTTP) for network transfers

References
Wikipedia on Simple Object Access Protocol (SOAP)
Wikipedia on Internet Protocol (IP)
Wikipedia on Transmission Control Protocol (TCP)

TCP/IP

Internet Protocol (IP)

- Internet Protocol (IP) is the principal communications protocol for relaying data across a network
- IP consists of
    - Addressing Mechanisms - identifying a computer with an IP address
    - Routing functions - how to deliver a packet to a specific destination from a specific source
    - Packets formats - the structure of a packet of data being routed
    - Location services - find the location of a computer on the network

Addressing

- An IP address looks like 123.456.789.012 - four groups of three numbers separated by dots
- Domain names - IP addresses are complex looking so Domain Names are used instead
    - They are what we are used to seeing i.e. google.com or microsoft.com
    - Human readable
    - Domain Name Services (DNS) used to lookup IP addresses given a domain name
    - Users are allowed to buy and register domain names of their choice (if not already in use)
    - Addressing uses a host IP (or domain name) and a port
        - Multiple applications may connect to the same computer, i.e. FTP, Web Browsers, Database connections, etc.
        - All applications use the same IP (or domain name)
        - They use different ports making them distinct

Routing

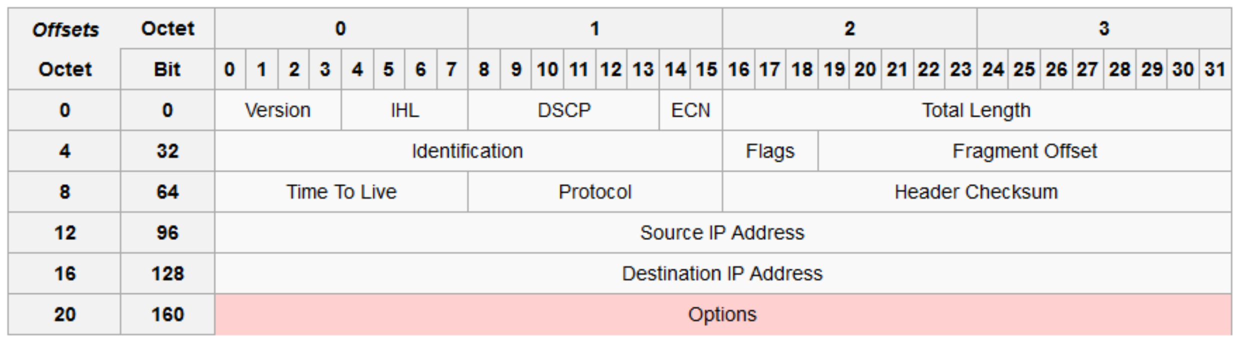
- IP is connectionless - does not depend on a direct connection to the destination
    - It just sends off packets to a router with no concern for whether they arrive at the destination
    - The network routers direct the packets towards the destination
        - A router will send to other routers depending on volume, delays, errors, etc.
        - The packets make up the communication between source and destination
        - The packets may go individually along different routes, the routers choose
        - The packets are reassembled into the total communication at the destination (but not by IP)

Packet Formats

- Communications are split into a set of packets
- The packets are transmitted over the network
- The packets have a format
    - Tutorials Point discussion of Packet format

Packet Header Format From Wikipedia

- Data follows the Header
- Header is actually in Binary
    - Shown below so that it can be understood
- Note the source and destination addresses (IP addresses)


IPv4 Header Format

Offsets	Octet	0								1								2								3							
Octet	**Bit**	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	**0**	Version				IHL				DSCP						ECN		Total Length															
4	**32**	Identification																Flags			Fragment Offset												
8	**64**	Time To Live								Protocol								Header Checksum															
12	**96**	Source IP Address																															
16	**128**	Destination IP Address																															
20	**160**	Options																															

References

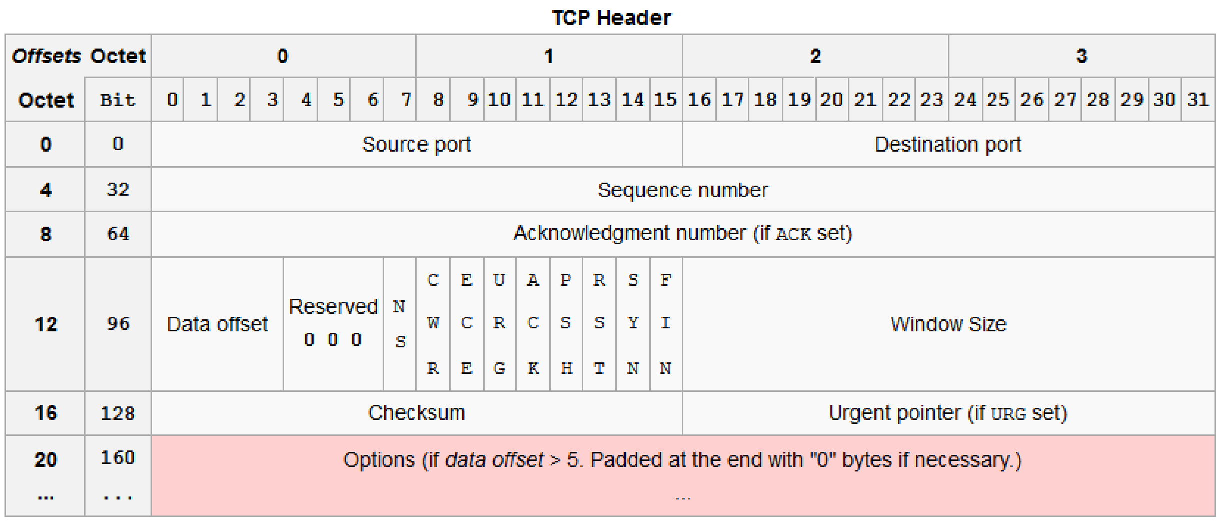
Packet format
Wikipedia on Internet Protocol
Wikipedia on IPv4 Packet Structure

Transmission Control Protocol (TCP)

- The TCP header and data go inside an IP Packet as its data
- TCP provides
    - Reliable communication between source and destination
        - At source sequence numbers are assigned to packets
        - Expects a positive acknowledgement (ACK) from destination for each packet
        - If ACK not received by timeout then retransmits
    - Flow Control
        - Destination sends back to source the number of bytes it can still receive without overflowing buffer
    - Order assembly of the packets received at the destination
        - Sequence number is used to order the packets at destination
    - Re-request of packets that go missing (IP does not guarantee delivery)
        - Source re-sends packets if ACK not received in timeout
    - Multiplexing
        - Multiple applications can communicate between sources and the destination computer
        - The port numbers are used, separate ones for various applications
    - Error checking of the data

Format of a TCP header from Wikipedia

- Data follows the header
- Note the source and destination Ports are here
- This TCP packet would actually be encoded in binary
    - Shown here for in a form to make it understandable



TCP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	128	Checksum																Urgent pointer (if URG set)															
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

References

Wikipedia on Transmission Control Protocol

Programming Sockets

- Sockets are the underlying programming construct for TCP/IP communication
  - They allow a computer application (the destination) to connect to a source server
  - The host domain name or IP address is specified
  - The port number on the host is specified
  - The socket is opened
  - Communication occurs between the server (source) and the application (destination)
  - The socket is closed
- Everything else is built on top of this foundation
- HTML is sent over a socket from a web server
- SOAP is sent over a socket from a web service
- WSDL is retrieved over a socket from a web service
- UDDI is enabled over a socket from a web service directory

Example of the Client Code for a Socket

- Example taken from TutorialsPoint and changed

```
import java.net.*;
import java.io.*;

public class SampleClient
{
   public static void main(String [] args)
   {
      String serverName = someServer.com; // the assigned name of a server
      int port = 5678;  // ports go up to 65000
      try
      {
         System.out.println("Connecting to " + serverName
                            + " on port " + port);
         Socket clientSocket = new Socket(serverName, port);
         System.out.println("Just connected to "
                     + client.getRemoteSocketAddress());
         OutputStream outToServer = clientSocket.getOutputStream();
         DataOutputStream out =
                      new DataOutputStream(outToServer);

         out.writeUTF("Hello from "
                     + clientSocket.getLocalSocketAddress());
         InputStream inFromServer = clientSocket.getInputStream();
         DataInputStream in =
                       new DataInputStream(inFromServer);
         System.out.println("Server says " + in.readUTF());
         clientSocket.close();
      }catch(IOException e)
      {
         e.printStackTrace();
      }
   }
}
```

Example of the Client Code for a Socket

- Example taken from TutorialsPoint and changed

```
import java.net.*;
import java.io.*;

public class SampleServer extends Thread
{
   private ServerSocket serverSocket;

   public SampleServer(int port) throws IOException
   {
      serverSocket = new ServerSocket(port);
      serverSocket.setSoTimeout(10000);
   }

   public void run()
   {
      while(true)
      {
         try
         {
            System.out.println("Waiting for client on port " +
            serverSocket.getLocalPort() + "...");
            Socket server = serverSocket.accept();
            System.out.println("Just connected to "
                  + server.getRemoteSocketAddress());
            DataInputStream in =
                  new DataInputStream(server.getInputStream());
            System.out.println(in.readUTF());
            DataOutputStream out =
                 new DataOutputStream(server.getOutputStream());
            out.writeUTF("Thank you for connecting to "
              + server.getLocalSocketAddress() + "\nGoodbye!");
            server.close();
         }catch(SocketTimeoutException s)
         {
            System.out.println("Socket timed out!");
            break;
         }catch(IOException e)
         {
            e.printStackTrace();
            break;
         }
      }
```

```
    }
    public static void main(String [] args)
    {
       int port = 7890;
       try
       {
          Thread t = new SampleServer(port);
          t.start();
       }catch(IOException e)
       {
          e.printStackTrace();
       }
    }
}
```

References [TutorialsPoint on Sockets]

HTTP and XML - What is the whole buzz about

- HTTP defines the basic protocols for the World Wide Web
    - It defines the request format for web page requests
    - It defines the response format for pages coming back to browsers (Firefox, Chrome, IE, etc.)
    - One part of the defined information is the status code such as 404 File Not Found, 200 successful, etc
- XML defines a basic markup language for building documents that are human and machine readable

References

Wikipedia on Extensible Markup Language (XML)
Wikipedia on HyperText Transfer Protocol (HTTP)

HTTP - HyperText Transfer Protocol

- Created at CERN by Tim Berners-Lee and his team
    - This was the original concept of the web server and a browser
    - They also defined the markup language HTML which was very similar to XML
    - The original protocol define only the operation GET which would get a page from a server

Example of an HTTP request

The request sent to a web server would be similar to the following:

```
GET /index.html HTTP/1.1
Host: www.example.com
```

Example of an HTTP response

The response from the server would be similar to the following:

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
ETag: "3f80f-1b6-3e1cb03b"
Content-Type: text/html; charset=UTF-8
Content-Length: 138
Accept-Ranges: bytes
Connection: close

<html>
<head>
  <title>An Example Page</title>
</head>
<body>
  Hello World, this is a very simple HTML document.
</body>
</html>
```

- Looking at the access log for a web server such as Tomcat
    - The requests can be seen

```
127.0.0.1 - - [20/Feb/2015:13:31:14 -0500] "GET /Web_Project_1/userForm.jsp
HTTP/1.1" 200 592
127.0.0.1 - - [20/Feb/2015:13:31:14 -0500] "POST
/Web_Project_1/actionPage.jsp HTTP/1.1" 500 2340
127.0.0.1 - - [20/Feb/2015:13:31:14 -0500] "GET
/Web_Project_1/anotherPage.jsp HTTP/1.1" 200 400
```

eXtensible Markup Language

- A markup language that defines a set of rules for encoding documents in human and machine readable format
- A free open standard
- It can define arbitrary data structures
- based upon the concept of elements (tags) and parent child relationships between elements
- It is widely used because of its simplicity, readability, and ability to represent most data

Example of XML

Define an XML document for a book reference (simplified)

```
<?xml version="1.0" encoding="UTF-8"?>
<book>
   <author>Leroy Jones</author>
   <published>August 4, 2014</published>
   <title>Trees of America</title>
</book>
```

XML Explanation

- XML declaration - declares some information about the xml for example

```
<?xml version="1.0" encoding="UTF-8"?>
```

- Markup - generally in XML markup begins with a < and ends with a >
- Content - content are strings of characters that are not markup
- Tags - Markup that begins with a < and ends with > are called tags
    - There are three types of tags in general
        - Start Tags - for example <book>
        - End Tags - for example </book>
        - Empty Element Tags - for example <line-break/>
- Elements - constructs that start with a Start Tag and end with an End Tag
    - For example <author>Susan Jones</author> is the author element of the document
    - Elements can have sub-elements
        - These are said to be in a parent child relationship
- Attributes - Start Tags and Empty Element Tags can have attributes
    - Attributes are name value pairs separated by an equal sign (=)
    - There can be several attributes on one tag
    - An example of an Empty Element Tag with attributes

```
<img src="redwood.jpg" alt="A Redwood Along Highway 101" />
```

Simple Object Access Protocol (SOAP)

- SOAP defines the XML elements that are used to communicate between the client and server
- The SOAP XML moves on the network between the client and server
- SOAP requests and responses move over HTTP
    - This makes it easier to move messaging through firewalls
    - The HTTP port is normally open in the firewall
    - HTTP servers to receive the messages are commonplace
    - Programming languages have Application Programming Interfaces (API) the work with HTTP
    - SOAP messages are not Operating System or programming language dependent
- SOAP contains Header, Envelope, and Body

Examples of SOAP Request and Response

The Soap Request

```
Header: POST /Hello_Service/services/Hello HTTP/1.0

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<helloName xmlns="http://nobleprog.com">
<name>Thomas</name>
</helloName>
</soapenv:Body>
</soapenv:Envelope>
```

The Soap Response

```
Header: HTTP/1.1 200 OK

<?xml version="1.0" encoding="utf-8" standalone="no"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<helloNameResponse xmlns="http://nobleprog.com">
<helloNameReturn>Hello there Thomas</helloNameReturn>
</helloNameResponse>
</soapenv:Body>
</soapenv:Envelope>
```

- Note the XML Namespace xmlns:soapenv it gives the definition of the SOAP XML
  - That schema can be viewed in a browser by going to the URL
    - Then do a view source for the page in the browser

References

W3Schools on SOAP

Web Services Definition Language (WSDL)

- An XML based protocol that defines information exchanges between a server and a client
- WSDL Describes a web service and the operations of that service
- Used to locate web services
- An open standard
- It usually can be retrieved and viewed to determine how to access and use a web service
  - A client application can be written using the WSDL
  - For an organization using internal web services

- The WSDL is a part of the documentation of a web service
- There may be further documentation given to client side developers

Elements of a WSDL

The main elements of a WSDL are:

- definitions - the container for the other major elements
- types - the data type definitions
- message - a typed definition of the data being communicated
- portType - a set of operations supported by the endpoint
- binding - the protocol and data format for a port type
- service - the service definition such as service URL

WSDL Example

An Example of a WSDL XML Document

Developed in Eclipse IDE using Eclipse Web Services Tutorial

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://nobleprog.com"
    xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://nobleprog.com"
    xmlns:intf="http://nobleprog.com"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <!--WSDL created by Apache Axis version: 1.4 Built on Apr 22, 2006
(06:55:48
        PDT) -->
    <wsdl:types>
        <schema elementFormDefault="qualified"
targetNamespace="http://nobleprog.com"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <element name="helloName">
                <complexType>
                    <sequence>
                        <element name="name" type="xsd:string" />
                    </sequence>
                </complexType>
            </element>
            <element name="helloNameResponse">
                <complexType>
                    <sequence>
                        <element name="helloNameReturn" type="xsd:string" />
                    </sequence>
                </complexType>
            </element>
        </schema>
    </wsdl:types>
    <wsdl:message name="helloNameResponse">
        <wsdl:part element="impl:helloNameResponse" name="parameters">
        </wsdl:part>
    </wsdl:message>

    <wsdl:message name="helloNameRequest">
        <wsdl:part element="impl:helloName" name="parameters">
        </wsdl:part>
    </wsdl:message>

    <wsdl:portType name="Hello">
        <wsdl:operation name="helloName">
            <wsdl:input message="impl:helloNameRequest"
name="helloNameRequest">
            </wsdl:input>
            <wsdl:output message="impl:helloNameResponse"
                name="helloNameResponse">
            </wsdl:output>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name="HelloSoapBinding" type="impl:Hello">
        <wsdlsoap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http" />
        <wsdl:operation name="helloName">
            <wsdlsoap:operation soapAction="" />
            <wsdl:input name="helloNameRequest">
                <wsdlsoap:body use="literal" />
            </wsdl:input>
            <wsdl:output name="helloNameResponse">
                <wsdlsoap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="HelloService">
        <wsdl:port binding="impl:HelloSoapBinding" name="Hello">
            <wsdlsoap:address
                location="http://localhost:8080/Hello_Service/services/Hello"
/>
        </wsdl:port>
    </wsdl:service>

</wsdl:definitions>
```

Universal Description, Discovery, and Integration (UDDI)

- UDDI is intended to be a directory system where web services can be looked up
- Artifacts of UDDI

    - Business (White Pages) - a description of a business and the UDDI key assign to a business
    - Service (Yellow Pages) - a web service provided for/by the business
        - There can be many services for a business
    - binding - location and access information for a web service
    - tModel (Green Pages) - descriptions and links to external descriptions
- UDDI does not seem to have been as successful as first envisioned
- It is used at some companies

Hands on Exercise

- Look up functioning UDDI software that can be purchased or obtained through open source organizations.
- Look up functioning public UDDI directories.
- Look up or think about alternatives to UDDI
- Discuss UDDI after looking up the above topics.

References

Stack Overflow on Non-use of UDDI
Wikipedia on UDDI

WS-* Profiles - what are these

- Web Services Interoperability (WS-I) profile
- Expose web services through common protocols
    - So that they are interoperable
    - For example are the following protocols interoperable
        - WSDL XML elements
        - SOAP XML elements
        - UDDI (if used) XML elements

- It is more a question of using commercial software that is WS_I compliant
    - Some examples of WS-I Compliant software
        - Oracle WebLogic Server
        - IBM Websphere
        - Apache CXF
        - GlassFish Metro
        - ASP.NET 2.0
        - JBossWS
- For example a business uses both JBoss and WebSphere as web servers
    - Are the web services on each interoperable

References

Wikipedia on WS-I Basic Profile
IBM on WS-I
Why is it Important to be WS-I Compliant?

Representational State Transfer (REST)

- REST is quickly overtaking SOAP and WSDL as the method of choice for web services
- It focuses upon resources and the retrieval and manipulations of them
- Much simplier
- UDDI is not used
- Should be documented well using other methods, i.e. Wiki, Word Docs, JavaDocs or other coding documentation, etc.

four basic design principles

- See IBM article in References below
    - Use HTTP methods explicitly.
    - Be stateless.
    - Expose directory structure-like URIs.
    - Transfer request using XML, JavaScript Object Notation (JSON), or both.

Use HTTP Methods Explicitly

- HTTP has all of the methods needed to retrieve and manipulate resources
    - Get - retrieve a resource
    - Post - create a resource
    - Delete - delete a resource
    - Put - update or modify a resource

- The HTTP methods should be used correctly
    - An example of using the methods correctly

Bad way to add a user

- There are two problems here
    -
        - Using GET to add a resource
        - Using query strings to specify data

```
GET /adduser?name=Robert HTTP/1.1
```

Better way to add a user

- The solution now uses
    - PUT to add a resource
    - XML to specify the data
        - JSON could have been used rather than XML

```
POST /users HTTP/1.1
Host: myserver
Content-Type: application/xml
<?xml version="1.0"?>
<user>
  <name>Robert</name>
</user>
```

Be Stateless

- The web service server should not have to save any state to fulfill consecutive requests from the same client
- The server can send requests on to other servers without worrying about state
- As an example view or explain JSESSIONID (Can be viewed in JMeter)
    - JSESSIONID is how a server built using Java tracks the user and creates state
- The server may set options so that responses are not cached by the client
- Clients send requests that are complete and independent and therefore do not rely on state at the server

Expose Directory Structure-like URIs

- Commonly REST tries to make simple URIs (URL)
- The URIs are intuitive and easy to guess as to their meaning
- They look like a folder path rather than having a query string
- The folder like path is hierarchical (in its nature)
    - Each succeeding sub-folder is a sub-characteristic of its parent
    - There is a constant pattern that allows similar URIs to be built
- They refer to resources
    - For example http://someServer.com/employee/19876
        - Rather than http://someServer.com/getEmployee?id=19876

Transfer request using XML, JavaScript Object Notation (JSON), or both

- Try not to use query strings to transfer requests or data
- Transfer data in XML, JSON, or HTML
- The data should be human readable and simple to read

References Very Useful Discussion of REST by IBM Dr. Dobbs on RESTful Web Services

JavaScript Object Notation (JSON)

- JSON and XML are:
  - Human readable
  - Can be transmitted over a web request or response
  - Hierarchical representation of data
  - Can be parsed to get field names and data

- JSON advantages
  - Shorter
  - Doesn't have end tags
  - easier to read by humans

- JSON data is in name value pairs
  - Name gives a variable name in the code
  - Value gives the data for the variable in the code
  - Examples
    - "nameOfString": "string value"
    - "nameOfFloatNumber": 1234.67
    - "nameOfBoolean": true

- JSON mimics JavaScript objects
  - For example a JavaScript object can be declared as:

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

JSON representation of a resource.

```
{
    "id": "123",
    "Firstname": "John",
    "Lastname": "Smith",
    "Email": "j.smith@mycompany.com",
    "Country": "England"
}
```

XML representation of a resource.

```
<Employee>
   <id>123</id>
   <Firstname>John</Firstname>
   <Lastname>Smith</Lastname>
   <Email>j.smith@mycompany.com</Email>
   <Country>England</Country>
</Employee>
```

References

W3Schools JSON Tutorial

The XML Technology

- As shown above XML is
  - A set of elements and attributes
  - Hierarchical
  - Is human readable
  - Is more complex than JSON
- XML also
  - Maps to Objects well in Object Oriented languages
  - Can be further defined using a DTD or Schema
  - Can be translated to and from objects by well known programming APIs

DTDs

- For DTDs XML Documents are made up from
  - Elements - the main building blocks
    - i.e. in XHTML

      ```
      <body> </body> and <table> </table>
      ```

  - Attributes - extra information about elements placed within the opening tag
    - i.e. in XHTML

      ```
      <a href="www.google.com>Google Website</a> href is an
      attribute
      ```

  - Entities - constructs such as

    ```
    < for the less than sign
    ```

  - PCDATA - parsed character data, data that will be parsed by a parser
    - a parser would find special characters and represent them correctly
  - CDATA - character data
    - Tags and entity data within the character data will not be used as markup or entities

Declaring Elements

```
<!ELEMENT element-name(child1, child2)>
```

- Examples

```
<!ELEMENT book (title, author, isbn)>
<!ELEMENT title (#PCDATA)>
```

Declaring Attributes for Elements

```
<!ATTLIST element-name attribute-name attribute-type attribute-value>
```

- element-name matches the name of a defined element in the DTD
- Example

```
<!ATTLIST book pubYear CDATA #REQUIRED>
```

- Data for an element can be child elements or attributes

```
<book title="The book's title as an attribute"> </book>

<book>
   <title>The Book's title as an element</title>
</book>
```

References

W3Schools DTD Tutorial

Schemas

- XML Schemas are an alternative to DTDs
- XML Schemas are more powerful in defining an XML document than DTDs
- Schemas are themselves XML documents and have a defining DTD
- They have a more complex system of types, such as integers, strings and floats
- Schemas define
  - Elements
  - Attributes
  - child elements
  - order of child elements
  - number of child elements
  - empty elements
  - Data types for elements and attributes
  - Default and fixed values for elements and attributes

The following page at W3Schools show the differences between DTDs and Schemas fairly well

Schemas vs DTDs

The following page at W3Schools shows how to reference the schema in your XML to allow validation

How to reference the Schema in the XML

- note that the schema would be placed on a web server so that it can be retrieved

Hands on Exercise

- Using the following XML write a DTD and a Schema that defines the XML
- Show how to reference the DTD or Schema within the XML for validation purposes

```
<book isbn="123456">
   <title>Some Title</title>
   <author>Author Name</author>
   <publisher>Publisher Name</publisher>
   <pages>234</pages>
</book>
```

References

W3Schools on Schemas
Differences between Schema and DTD
Another on Differences between Schema and DTD

eXtensible Stylesheet Language Transformations (XSLT)

- Used to transform XML documents into other forms.
    - For example transform XML to XHTML
- XSLT uses XPath to locate elements in the XML
- A transformation template is then applied to the element

The following W3Schools page shows how the transform would work fairly well

XSLT Example Transformation

References

W3Schools XSLT tutorial

XML Processing/Parsing in the Code

- It is convenient to move between XML and objects
    - In an object oriented programming language
- It is also possible to call event handlers as elements in XML documents are discovered
- There are numerous code libraries that can be used
    - Java Architecture for XML Binding (JAXB)
    - Java Architecture for XML Processing (JAXP)
        - SAX uses a set of callback event handlers
            - They are called as the document is parsed and elements found
        - DOM builds a object model of the document elements
- There are XML parsers for most programming languages

References

XML Parsers for various languages

Alternatives to SOAP and WSDL

- HTTP with XML
- HTTP with JSON
- RMI - Remote Method Invocation in Java
- CORBA - from way back in the 1990s
- RESTful - Discussed above

Hands on Exercise

- Lookup alternatives to WSDL and SOAP and Discuss
- Note that these are language dependent (i.e. Java, C++, C#, Python, etc.)

References

Wikipedia on Web Service Protocols

References

Wikipedia on Web Services
W3 Schools on Web Services
Webopedia on Web Services
TutorialsPoint on Web Services
IBM Web Services Tutorial