Web Services Part I

Instructor: Dr. Wei Ding

Fall 2009

XML Web Services

- "XML Web Services" = "Web Services"
- A Web service is a different kind of Web application.
- It does not have a user interface as done a traditional Web application
- It exposes callable API functions, known as *Web methods*, over the Internet.
- It is not designed to serve end users as traditional Web applications are.
- It is designed to provide services to other applications.

Examples of services that Web services provide

- Real-time stock quotes to interested parties
- Validate credit cards
- Provide current information about the weather
- Up to the implementer…

- The goal of Web services—an Internet that does not just serve end users, but one that allows servers to communicate with each other and applications to be freed from the bonds of the platforms on which they run.

Web services are an industry standard

- Web services are not the property of Microsoft.
- They are an industry standard built on open protocols such as HTTP and the Simple Object Access Protocol (SOAP).
- Many of the Web services in operation today run on UNIX servers
- You do not need the .NET Framework to write Web services or Web service client

Web Services is an application

- Runs on a Web server
- Exposes Web methods to interested callers
- Listens for HTTP requests representing commands to invoke Web methods
- Executes Web methods and returns the results

- Most Web services expect their Web methods to be invoked using HTTP requests containing SOAP messages. SOAP is an XML-based vocabulary for performing remote procedure calls using HTTP and other protocols.

HTTP

Instructor: Wei Ding

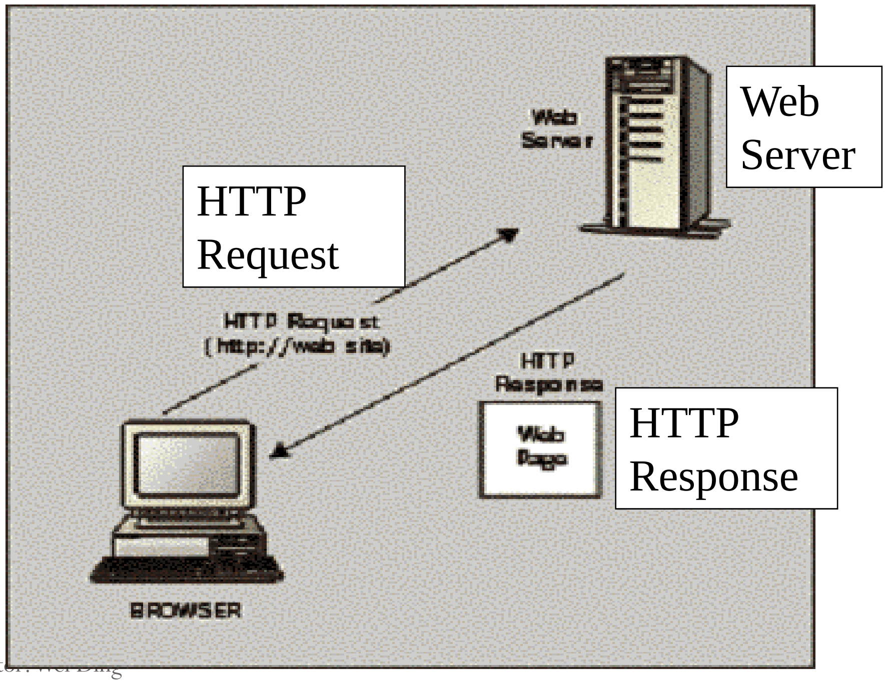
The Hypertext Transfer Protocol (HTTP)

- Hypertext Transfer Protocol or HTTP is the language web clients and servers use to communicate with each other.
- HTTP is essentially the backbone of the World Wide Web. When you see a URL prefixed with http://, you know that the Internet protocol being used is HTTP, as HTTP is the default protocol used by web browsers. This means if we type www.cnn.com, the browser will automatically use the HTTP protocol and search for http://www.cnn.com.

The Hypertext Transfer Protocol (HTTP)

- HTTP only defines what the browser and web server say to each other, not how they communicate. The actual work of moving bits and bytes back and forth across the network is done by TCP and IP.

How a link works?



HTTP Transactions

- Each client request and server response has three parts:
  - The request or response line
  - A header section
  - And the entity body

The client initiates a transaction – HTTP Request

1. The client contacts the server by specifying an HTTP command, a document address, and an HTTP version number.

   GET /index.html HTTP/1.1

2. Next, the client sends optional header information to inform the server of its configuration and the document formats it will accept.

   User-Agent: Mozilla/4.05(WinNT; I)

   Accept: image/gif, image/jpeg, */*

3. The client sends a blank line to end the header.
4. After sending the request and headers, the client may send additional data. This data is mostly used by the POST method.

The Server responds to the client's request – HTTP Response

1. The server replies with a status line containing three fields: HTTP version, status code, and description.

   200 OK HTTP/1.1

2. After the status line, the server sends header information to the clients about itself and the requested document.

   content-length: 14250
   content-location: http://dcm.cl.uh.edu/Default.htm
   server: Microsoft-IIS/5.0
   last-modified: Thu, 02 Jun 2005 01:41:39 GMT
   date: Wed, 04 Jan 2006 17:34:50 GMT
   content-type: text/html

3. A blank line ends the header.
4. If the client's request is successful, the requested data is sent.

HTTP Request

HTTP Request Format

```
[METH] [REQUEST-URI] HTTP/[VER]
[fieldname1]: [field-value1]
[fieldname2]: [field-value2]

[request body, if any]
```

HTTP Request Methods

- The most commonly used HTTP request Methods:
  - GET	Return the contents of the specified document
  - HEAD	Return the header information for the specified document
  - POST	Execute the specified document, using the enclosed data
  - PUT	Replace the specified document with the enclosed data
  - DELETE	Delete the specified document

GET vs. POST

- **GET vs. POST**: In both techniques, the form data is coded into a text string called *query string* when the user presses the Submit button.
  - The GET method can also be used to pass parameters to the server when forms are not involved (this cannot be done with POST). The main disadvantage of the GET method is that some servers place a limit on the length of the URL string and truncate any characters past the limit. So, if the form has more than a few controls, GET is not a good choice. Another potential problem with GET is that the query string is vulnerable to illegal or inappropriate access because of its appearance with the URL, because URLs are easy to find by network sniffers.
  - When the POST method is used, the query string is passed through HTTP request body. The length of the query string is passed through the environment variable CONTENT_LENGTH. There is no length limitation for the query string with the POST method, so it is obviously the choice when there are more than a few controls in the form.

Examples

```
GET /cgi-bin/guestbook.pl?firstname=Joe&lastname=Smith HTTP/1.0
```

```
POST /cgi-bin/guestbook.pl HTTP/1.0
CONTENT-LENGTH=28
… [More headers here]

firstname=Joe&lastname=Smith
```

Query String Format

- For each control in the form that has value, the control’s name and that value are coded as the assignment statement and included in the query string. A control’s value is always a character string.

HTML: <INPUT type = "radio" name = "payment" value = "discover">

Query string: Payment=discover

- If the form has more than one control, the assignments that code their values in the query string are separated by ampersands (&):

Caramel=7&payment=discover

Query String Format

- Percent Sign (%): If there are special characters in the value of a control, they are coded as a percent sign followed by a two-character hexadecimal number that is the ASCII code for that character. E.g. ! is %21, a white space is %20.

Query String Format

- Suppose that you had a form whose value was visa and a text control whose name was saying and whose value was "Eat your fruit!". The query string for this form would be as follows:

payment=visa&saying=Eat%20your%20fruit%21

Or (some browsers replace spaces with +)

payment=visa&saying=Eat+your+fruit%21

HTTP Request Header Fields

- The format of a Header field is the field name, followed by a colon and the value of the field.
- The Accept field is often included in a request; it specifies preference of the browser for the MIME type of the requested document.

  Accept: text/plain or Accept: image/gif or

  Accept: text/* (if any kind of text is acceptable)

HTTP Request Header Fields

- If the request method is POST, the content-length field must be included in the request header. It specifies the number of bytes in the body of the request data. why?
- The header of a request must be followed by a blank line, which is used to separate the header from the data.

HTTP Response

HTTP Response Format

[CODE] [TEXT] HTTP/[VER]

Field1: Value1

Field2: Value2 …

Document content here…

HTTP: The Response Phase

The general form of an HTTP response is as follows:

- Status line includes the HTTP version used, a three-digit status code for the response, and a short textual explanation of the status code.

  200 ok HTTP/1.1

- The status codes begin with 1,2,3,4, or 5. The general meanings of the five categories specified by these first digits are:
  1. Informational
  2. Success
  3. Redirection (the information required has now been moved)
  4. Client error (the request was either incomplete, incorrect or impossible)
  5. Server error (the request appeared to be valid, but that the server failed to carry it out)

HTTP: The Response Phase

- Response header fields follow the status line. The only essential field of the header is **content-type**, which tells the browser how to display the response data.
- Blank line: The response header must be followed by a blank line, as is the case for request headers.
- Response body follows the blank line.

HTTP is a stateless protocol

- HTTP is known as a stateless protocol. This is because it doesn't know whether the request that has been made is part of an ongoing correspondence or just a single message, just the same way your postman won't know whether your letter is the first asking your local hi-fi company for a refund, or the fifteenth.

Why a stateless protocol?

- The reason HTTP is stateless is that it was only intended to retrieve a single web page for display. Its purpose was to handle simple transactions, where a user requests a web page, the browser connects to the requisite web server, retrieves that web page, and then shuts down the connection.
- The Internet would be very slow and might even collapse if permanent connections needed to be maintained between browsers and servers as people moved from one page to another. Think about the extra work HTTP would have to do if it had to worry about whether you had been connected for one minute or whether you had been idle for an hour, and needed disconnecting. Then multiply that by a million for all the other users. Instead, HTTP makes the connection and delivers the request, and then returns the response and disconnects.

The downside of being a stateless protocol

- However, the downside is that HTTP can't distinguish between different requests, and can't assign different priorities, so it won't be able to tell whether a particular HTTP Request is the request of a user, or the request of a virus infected machine, that might have been set up, for instance, to hit a government web server 1,000 times an minute. It will treat all requests equally, as there are no ways for HTTP to determine where the request originated.

XML

Instructor: Wei Ding

What is XML?

- XML stands for E**X**tensible **M**arkup **L**anguage.
- XML is a **markup language** much like HTML.
- XML was designed to **describe data**. HTML was designed to display data and to focus on how data looks. XML is not a replacement for HTML.
- XML tags are not predefined in XML. You must **define your own tags**.
- XML uses a DTD (**Document Type Definition**) to describe the data.
- XML with a DTD is designed to be **self-descriptive**.

Portable Data

- Using plain text files, XML is a cross-platform, software and hardware independent **tool for transmitting information**.
- With XML, data can be exchanged between incompatible systems.

Example

```
<?xml version="1.0" encoding="ISO-8859-1”?>
<note>
 <to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Study the first example

- It is just pure information wrapped in XML tags.
- Someone must write a piece of software to send it, receive it or display it.
- <note> describes the root element of the document (like it was saying: “this document is a note”).
- <to>Tove</to> <from>Jani</from> <heading>Reminder</heading> <body>Don't forget me this weekend!</body>, the 4 lines describe 4 child elements of the root (to, from, heading, and body)
- </note>, and finally the last line defines the end of the root element.

Study the first example

- <?xml version="1.0" encoding="ISO-8859-1"?> is the XML declaration. It is not an XML element.
- XML declaration defines the XML version and the character encoding used in the document. In this case the document conforms to the 1.0 specification of XML and used the ISO-8859-1 (Latin-1/West European) character set.
- An XML document does not have to begin with an XML declaration. But once it does, the declaration must be the first thing in the document. It must not be preceded by comments, whitespace, or processing instructions.

XML Tags

- **XML tags are not predefined.** You must "invent" your own XML tags and document structure.
- **XML tags are case sensitive.**
  - Opening and closing tags must therefore be written with the same case:
    - ~~<Message>This is incorrect</message>~~
    - <message>This is correct</message>
- **With XML, it is illegal to omit the closing tag.**
  - In XML all elements must have a closing tag like this:

    <title>XML in a Nutshell</title>

    <emptyElement></emptyElement>

    <emptyElement />

    ~~<ThisIsIncorrect>~~

Elements have Content

- An XML element is everything from (including) the element's start tag to (including) the element's end tag.

  ```
  <person>
        Tom Smith
  </person>
  ```

  A single element whose type is person. Tom Smith is the element content.
- An element can have element content, mixed content, simple content, or empty content.
- An element can also have attributes. In XML the attribute value must be enclosed in single or double quotation.

  ```
  <note date="12/11/99" />
  <person name='Tom Smith' />
  ```

Element Naming Rules

- Names can contain letters, numbers, and other characters.
- Names must not start with a number or punctuation character (dollar signs, carets, percent symbols, semicolon, etc.)
- Names cannot contain spaces.
- The ":" should not be used in element names because it is reserved to be used for something called namespaces

Tips

- Make names descriptive. For example, names with an underscore separator. <first_name>.
- Names should be short and simple.

XML Trees

Description of a book:

Book Title: My First XML

Chapter 1: Introduction to XML

- What is HTML
- What is XML

Chapter 2: XML Syntax

- Elements
- Tags

XML Elements have Relationships

- XML documents are trees.
- book is the root element.
- title, prod, and chapter are child elements of book.
- book is the parent element of title, prod, and chapter.
- title, prod and chapter are siblings (or sister elements) because they have the same parent.

Elements have Content

- In the last Book example,
  - book has **element content**, because it contains other elements.
  - Chapter has **mixed content** because it contains both text and other elements.
  - Para has **simple content** (or **text content**) because it contains only text.
  - Prod has **empty content**, because it carries no information.

XML Trees

- Improper nesting of tags makes no sense to XML. In XML all elements must be properly nested within each other like this:

<name><first_name>Tom</first_name>

</name>

- All XML documents must contain exact one tag pair to define the root element. All other elements must be nested within the root element.

Comments in XML

- The syntax for writing comments in XML is similar to that of HTML.
- <!-- This is a comment -->

XML Elements are Extensible

- XML documents can be extended to carry more information.
- Look at the following XML example
- <note> <to>Tove</to> <from>Jani</from> <body>Don't forget me this weekend!</body> </note>
- Let's imagine that we created an application that extracted the <to>, <from>, and <body> elements for the XML document to produce this output:

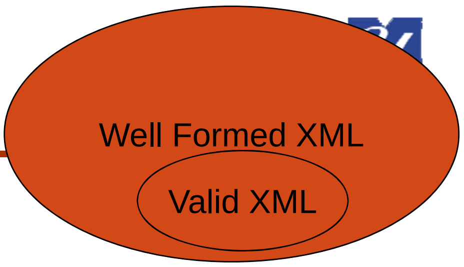
XML Elements are Extensible

- **MESSAGE To:** Tove
  From: Jani
  Don't forget me this weekend!
- Imagine that the author of the XML document added some extra information to it:
- <note> <date>1999-08-01</date> <to>Tove</to> <from>Jani</from> <heading>Reminder</heading> <body>Don't forget me this weekend!</body> </note>

XML Elements are Extensible

- Should the application break or crash?
- No. The application should still be able to find the <to>, <from>, and <body> elements in the XML document and produce the same output.
- XML documents are extensible.

XML Validation



- XML with correct syntax is **Well Formed XML.**
  - A "Well Formed" XML document is a document that conforms to the XML syntax rules that we have learned.
- XML validated against a DTD is **Valid XML**.
  - A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a Document Type Definition (DTD).

XML DTD

- A DTD defines the legal elements of an XML document.
- The purpose of a DTD is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements.
- A DTD can be declared inline in your XML document, or as an external reference.

Internal DOCTYPE declaration

- If the DTD is included in your XML source file, it should be wrapped in a DOCTYPE definition with the following syntax:
- <!DOCTYPE **root-element** [element-declarations]>

Example: note_in_dtd.xml

- In this example, we have

```
<!DOCTYPE note [
  <!ELEMENT note    (to,from,heading,body)>
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>]>
```

Example: note_in_dtd.xml

- **!DOCTYPE note** (in line 2) defines that this is a document of the type **note.**
- **!ELEMENT note** (in line 3) defines the **note** element as having four elements: "to,from,heading,body".
- **!ELEMENT to** (in line 4) defines the **to** element to be of the type "#PCDATA".
- **!ELEMENT from** (in line 5) defines the **from** element to be of the type "#PCDATA"
  and so on.....

External DOCTYPE declaration

- If the DTD is external to your XML source file, it should be wrapped in a DOCTYPE definition with the following syntax:
- <!DOCTYPE root-element SYSTEM "filename">

Example: note_ex_dtd.xml note.dtd

- This is the same XML document as note_in_dtd.xml, but with an external DTD.
- External file note.dtd contains the DTD.

Why use a DTD?

- With DTD, each of your XML files can carry a description of its own format with it.
- With a DTD, independent groups of people can agree to use a common DTD for interchanging data.
- Your application can use a standard DTD to verify that the data you receive from the outside world is valid.
- You can also use a DTD to verify your own data.

DTD – XML building blocks

- Seen from a DTD point of view, all XML documents are made up by the following simple building blocks:
  - Elements
  - Tags
  - Attributes
  - Entities (&amp; &lt; …)
  - PCDATA: means parsed character data. It is text that will be parsed by a parser. Tags inside the text will be treated as markup and entities will be expanded.
  - CDATA: means character data. CDATA is text that will NOT be parsed by a parser. Tags inside the text will NOT be treated as markup and entities will not be expanded.

Introduction to XSL

- XSL stands for eXtensible Stylesheet Language
- XSL is a language that can **transform** XML into HTML, a language that can **filter and sort** XML data, a language that can **address parts** of an XML document, a language that can **format** XML data based on the data value, like displaying negative numbers in red, and a language that can **output** XML data to different devices, like screen, paper or voice.

Example: Displaying XML with XSL

- XSL is far more sophisticated than CSS.
- The example just illustrates **one** way to use XSL to transform XML into HTML before it is displayed by the browser.

Acknowledgement

- www.w3schools.com
- Elliotte Rusty Harold & W. Scott Means, XML IN A NUTSHELL, O'REILLY