1.264 Lecture 22

XML
Web services

Next class: No readings. Exercise due after class
Use same Web site today that we started in Lecture 21
Also open SQL Server Management Studio

Recap of XML Basics

- XML documents hold self-describing data
  - Hierarchies, objects, database tables can be sent
  - Extensible, flexible, decided by industry groups, partners
- XSLT documents can transform or style XML documents
  - CSS isn't enough because we want to display tags here
- DTD files can validate XML documents
  - URL has the DTD that server or client can use
  - DTDs are limited: can't define data types, etc.
- XSchema (XSD) files preferred to validate XML documents
  - More structured, more extensible than DTDs
- Databases can read and write XML
  - Web servers can send and receive XML as payload in HTTP, much like HTML pages
  - Microsoft has made XML the markup language in Office
  - Putting a disruptive technology in place to automate commerce?

Exercise 1: XmlDataSource: BookMIT.aspx

- File-> New File -> Web Form: BookMIT.aspx
- Drag XmlDataSource control on page, configure:
  - Browse for Books.xml
  - XPath: //booklist/book [Forward slashes, not backward]
- Drag DataList onto XmlDataSource
  - Choose XmlDataSource
  - (View in browser—MS tools only show attributes, not elements)
- Go to source view, delete all text in <ItemTemplate> and replace with:

```
<p> <%# XPath("author/first-name")%>
    <%# XPath("author/last-name")%>
    <%# XPath("title") %>
    <%# XPath("price") %>
    <%# XPath("@topic") %>
    <%# XPath("@publicationdate") %>
    <%# XPath("@ISBN") %>
</p>
```

- Save and test (BookMIT2.aspx has nicer format—see download)

XSLT: XML Stylesheet Language/Transformation

- XSLT is a language to transform and style XML documents from one form to another
  - Use for display, especially to show tags, which CSS does not do
  - Use to convert an XML document from one format to another, in passing it from one organization to another
- Example on next page transforms Books.xml into Authors.xml by selecting just the author tags
  - XSLT has loops, if statements and is a mini-programming language to select and transform data
  - You don't need to understand the details of XSLT
  - BookAuthorXSLT.aspx applies Books.xslt to Books.xml
    - See download and example run in class

Exercise 2: XSLT: Books.xslt

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   version="1.0">
    <xsl:template match="/">
        <xsl:element name="Authors">
            <xsl:apply-templates select="//book"/>
        </xsl:element>
    </xsl:template>
    <xsl:template match="book">
        <xsl:element name="Author">
            <xsl:value-of select="author/first-name"/>
            <xsl:text> </xsl:text>
            <xsl:value-of select="author/last-name"/>
        </xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

Run BookAuthorXSLT.aspx to apply Books.xslt to Books.xml

Exercise 3: Simple XSLT

- File-> New File -> Web Form: BookMIT3.aspx
- Drag Xml control on page from the Standard tools
  - Not an XMLDataSource control
- Configure it by typing two properties in Source view, or by using the Properties window to select them:
  - DocumentSource="Books.xml"
  - TransformSource="Books.xslt"
- The XML control will display the authors from the XML file
  - This is a simple illustration of transforms
- All of these exercises are illustrative. XML still requires some programming, often simple, which is beyond the scope of this class.

Exercise 4: Blogs are XML: Blog.aspx

```
<%@ Page Language="C#“ (details omitted) %>
<!DOCTYPE html … (details omitted)>
<html xmlns="http://www.w3.org/1999/xhtml" >
    <head id="Head1" runat="server">
        <title>XmlDataSource Blog</title>
    </head>
    <body>
    <form id="form1" runat="server">
        <asp:DataList ID="D1" Runat="server“ DataSourceID="XmlData1">
            <ItemTemplate>
                <%# XPath("title") %><br />
                <%# XPath("pubDate") %><br />
                <%# XPath("description") %>
            </ItemTemplate>
        </asp:DataList>
        <asp:XmlDataSource ID="XmlData1" Runat="server"
            DataFile="http://www.hanselman.com/blog/feed"
            XPath="rss/channel/item">
        </asp:XmlDataSource>
    </form>
    </body>
</html>
```

Execute Blog.aspx. Compare with www.hanselman.com in browser.

Exercise 5: Databases and XML

- Open SQL Server Mgt Studio, use MIT1264 database
- Type:
  - SELECT * FROM Customers FOR XML AUTO
    - Generates XML with data as attributes
  - SELECT * FROM Customers FOR XML AUTO, ELEMENTS
    - Generates XML with data as elements
  - Place a root tag (e.g., <Customerlist>) and end tag around the SQL result set, and we have a valid XML document
    - Can generate the root and end tags with SQL, C#, Java, ….
  - Can also INSERT, UPDATE, DELETE with XML

XML database insert example

```
CREATE PROCEDURE xmlOrderInsert @order ntext AS
DECLARE @docHandle int, @OID int
EXEC sp_xml_preparedocument @docHandle OUTPUT, @order
BEGIN TRANSACTION
INSERT INTO Orders( CustomerID, EmployeeID, OrderDate, RequiredDate )
  SELECT CustomerID, EmployeeID, OrderDate, RequiredDate
  FROM Openxml( @docHandle, '/Order', 3) WITH ( CustomerID nchar(5),
  EmployeeID int,   OrderDate datetime, RequiredDate datetime   )
IF @@ERROR<>0 BEGIN ROLLBACK TRANSACTION RETURN -100 END
SET @OID = SCOPE_IDENTITY()
INSERT INTO [Order Details] ( OrderID, ProductID, UnitPrice, Quantity,
Discount )
SELECT @OID AS [PO ID], ProductID, UnitPrice, Quantity, Discount
 FROM OpenXml( @docHandle, '/Order/OrderDetails', 1)   WITH
  ( ProductID int, UnitPrice money, Quantity smallint, Discount real
)
  IF @@ERROR<>0 BEGIN ROLLBACK TRANSACTION RETURN -101 END
COMMIT TRANSACTION
EXEC sp_xml_removedocument @docHandle SELECT @OID AS [Order ID]
GO
```

From codeproject.com

Exercise 6: More databases and XML

- You can have XML columns in a database
  - You can query, update and index them
- One at a time, run:
  - xml1.sql
  - xml2.sql
  - xml3.sql
  - xml4.sql
- What does each do?
- This is a nice way to store highly variable information where standard columns don't work

XML1-4 code from Kiessig, Ultra-Fast ASP.NET

Solution: Exercise 6

- Xml1: Creates XML schema and Products table
  - Used by Products table to validate XML column
- Xml2:
  - Select all columns from all rows
  - Select id, name, props where width exists
  - Select id, name, width where color of top part is black
  - Select id, name, first color where color of legs is chrome
- Xml3:
  - Change leg color to silver
  - Add arm with color white
  - Remove first color (arm)
- Xml4: creates index on XML column, path, properties and values
  - See indexes on Products table in SSMS explorer

Midpoint summary

- XML document: snippet of database being sent from one Web server to another
  - Contains tags defined by business, hyperlinks, etc.
  - Grew from HTML but defines content, not appearance
- DTD and/or XSD: contains business rules (1-1, 1-many, null/not null, etc) to validate XML document
- XSLT: transforms and styles and XML document
  - Make change to sent or received XML to meet business needs
- XPath: query language used for XML documents
  - Same role as SQL

Simple Object Access Protocol: SOAP

- SOAP is XML and HTTP
  - Intent is to use no extra technology for distributed computing
  - SOAP adds some headers to HTTP
  - SOAP adds new MIME type: text/xml
  - Adds agreed definitions of data types, mandatory values, etc.
  - HTTP POST and XML replace complex programs
  - SOAP is text rather than binary, so it's much easier to interoperate across machines and debug it
    - Binary transmissions are not human-readable
    - SOAP can send binary objects like pictures (that are human readable)
  - SOAP is sufficiently efficient for most machine-machine communication
- SOAP is the key component in Web services, or Service Oriented Architectures (SOA)

Web Services

- Web service provides information in an XML format to be used by a client
  - It does not display the data in a user interface
  - It is not tied to a specific Web page or document
- Web services in .NET are on pages with a .asmx extension (not .aspx)
  - VSW and .NET make writing simple Web services easy
  - SQL is generally used to generate the XML payload
  - Web services accept parameters, such as customer number or order number, to return only data required
    - Parameters are used in the WHERE clause in SQL
- Web Services Description Language (WSDL) describes a Web service
  - Automatically created by VSW

Exercise 7: Web Services in VSW

- We will create a Web service in a regular Web site
  - Normally we create a separate project/site for Web services
- In VSW, open previous Web site, or create new one
  - File -> New File -> Web Service: TemperatureService.asmx
  - Uncheck 'place code in separate file'. No master page.
- Type the following code:

```
[WebMethod]
    public double toCelsius(double tf)
    {
        return (5.0 / 9.0) * (tf - 32.0);
    }
```

- Save and view in browser: a WSDL page is shown

Web service example: request

```
POST /TemperatureService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: 300

SOAPAction: "http://tempuri.org/ToCelsius" <?xml version="1.0"
   encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
   instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
        <ToCelsius xmlns="http://tempuri.org/">
               <TF> 32.0</TF>
        </ToCelsius>
   </soap:Body>
</soap:Envelope>
```

Web service example: response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 250

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
   instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
        <ToCelsiusResponse xmlns="http://tempuri.org/">
               <ToCelsiusResult> 0.0</ToCelsiusResult>
        </ToCelsiusResponse>
   </soap:Body>
</soap:Envelope>
```

Web service example 2: database query

- We wish to query the Parts table for all parts of a given vendor
  - The vendor will be a parameter to the Web service
  - The Web service will use a database connection to execute the SQL query
  - It will return an XML document as its result
- The code is in your Lecture 22 download as ProductService.asmx

Web service example 2: request

```
POST /mit1264lecture22/ProductsService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/GetProducts"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
   instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetProducts xmlns="http://tempuri.org/">
      <Vendor>string</Vendor>
    </GetProducts>
  </soap:Body>
</soap:Envelope>
```

ProductService.asmx

```
<%@ WebService Language="C#" Class="ProductService" %>
using System;  // and other 'using' directives
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]

public class ProductService  : System.Web.Services.WebService {
    [WebMethod]
    public DataSet GetProducts(string Vendor) {
        SqlConnection con1264= new SqlConnection(    // Connect to db
        "Data Source=.\\SQLEXPRESS; Password=xxx;
         User ID=yyy;Initial Catalog=MIT1264");

        SqlDataAdapter dad1264= new SqlDataAdapter("SELECT * FROM
         Parts WHERE Vendor= @Vendor", con1264);     // Create query

        dad1264.SelectCommand.Parameters.Add(        // Add parameter
         new SqlParameter("@Vendor", Vendor));

        DataSet dstProducts= new DataSet();   // Create output object
        dad1264.Fill(dstProducts, "Products");       // Run query
        return dstProducts;
    }
}
```

Web service example 2: response

```
<?xml version="1.0" encoding="utf-8" ?>
- <DataSet xmlns="http://tempuri.org/">
  - <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    - <xs:element name="NewDataSet" msdata:IsDataSet="true">
      - <xs:complexType>
        - <xs:choice maxOccurs="unbounded">
          - <xs:element name="Products">
            - <xs:complexType>
              - <xs:sequence>
                  <xs:element name="PartID" type="xs:string" minOccurs="0" />
                  <xs:element name="Vendor" type="xs:string" minOccurs="0" />
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  - <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
    - <NewDataSet xmlns="">
      - <Products diffgr:id="Products1" msdata:rowOrder="0">
          <PartID>123</PartID>
          <Vendor>A</Vendor>
        </Products>
      - <Products diffgr:id="Products2" msdata:rowOrder="1">
          <PartID>234</PartID>
          <Vendor>A</Vendor>
        </Products>
      - <Products diffgr:id="Products3" msdata:rowOrder="2">
          <PartID>3464</PartID>
          <Vendor>A</Vendor>
        </Products>
      - <Products diffgr:id="Products4" msdata:rowOrder="3">
          <PartID>362</PartID>
          <Vendor>A</Vendor>
        </Products>
      </NewDataSet>
    </diffgr:diffgram>
  </DataSet>
```

Exercise 8: Web service

- Change the connection string in ProductService as needed:

```
Data Source=.\\SQLEXPRESS; Password=xxx;
User ID=yyy;Initial Catalog=MIT1264
```

- Modify the Web service to return parts and vendors only if the part ID is 1000 or greater
  - Modify the connection string to log into your database

Solution

- "SELECT * FROM Parts WHERE Vendor= @Vendor AND partID > 1000"

1.264J / ESD.264J Database, Internet, and Systems Integration Technologies
Fall 2013