CSE 445/598 Assignment 4 (100 Points)
Fall 2013

Part 1: Questions 1, 2, and 3 Due Date: Saturday, November 2, 2013, by 11:59pm (Arizona Time)
Part 2: Questions 4, 5, and 6 Due Date: Saturday, November 9, 2013, by 11:59pm (Arizona Time)

Introduction

The aim of this assignment is to make sure that you understand and are familiar with the concepts covered in the lectures, including XML elements, attributes, statements, XML schema, XML validation, XML transformation (XSL), and their classes in .Net FCL. By the end of the assignment, you should have applied these concepts and techniques in creating an XML file, its schema, its style sheet, and have written Web services and an SOA application to process these files.

This is an **individual assignment**. Each student must complete and submit independent work. No cooperation is allowed, even among the team members for assignment 3. Do not use WebStrar or V-Lab to host your files and services that are shared among your team members. You can use your ASU personal Web site space to host your XML, XSD, and XSL files, see Part 0 question 4 and use localhost to host the services in Part 2.

Part 0 Practice Exercises (No submission required)

No submission is required for this part of exercises. However, doing these exercises can help you better understand the concepts and thus help you in quizzes or exams.

1. Reading: Textbook Chapter 4.

2. Answer the multiple choice questions 1.1 through 1.16 of the text Section 4.8. Study the material covered in these questions can help you to prepare for the class exercises, quizzes, and the exam.

3. Study for the questions 2 through 8 in text Section 4.8. Make sure that you understand these questions and can briefly answer these questions. Study the material covered in these questions can help you to prepare for the exam and understand the homework assignment.

4. If you have not activated your file service and personal Web hosting site at ASU, you can activate them at: https://selfsub.asu.edu/apps/WebObjects/ASURITEActivation. Then, you can search for Uploading Your Personal Web page within ASU search page to find the steps of uploading your file. Notice that ASU Personal Web site space hosts files only. It does not host programs such as Web services and Web applications. IIS are not installed.

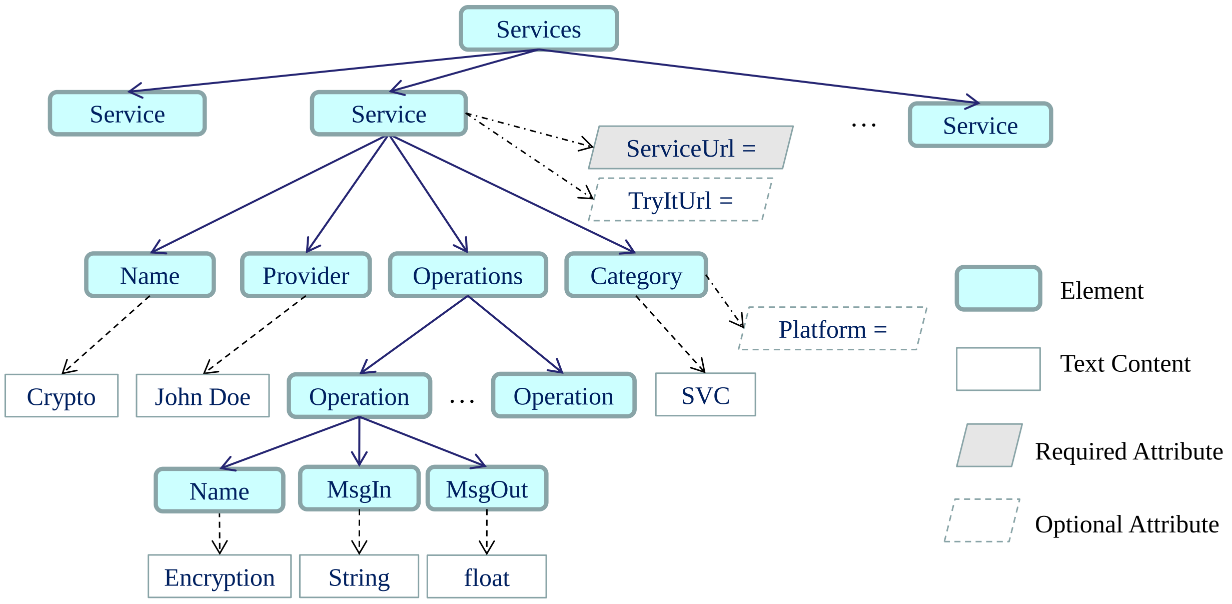
Part 1 Creating XML, XSD, and XSL Files (30 points)

Part 1 (Questions 1, 2, and 3) Due Date: Saturday, November 2, 2013, by 11:59pm (Arizona Time).

This part has 30 points. The grade will be based on your Pat 1 submission. You will use these files in your Part 2 questions, and you must include them in Part 2. You may modify these files when you resubmit them in Part 2. However, we will not grade these files in Part 2. We will grade the Web service questions only in Part 2.

A service directory can be represented as a table, as we have seen in ASU Service and Application repository and in Xmethods.net Website. However, it is more efficient to represent the directory as a tree in XML format.

The diagram below shows a simplified structure of a service directory in XML format. All the "Service" elements have the same structure. Notice that different shapes of boxes have different meanings. They represent elements, attributes, and text contents, respectively. The structure of elements and attributes are required to implement, while the given text contents in the diagram are examples.



1. Write the Services.xsd file that defines the XML schema allows the structure shown in the diagram above. You can use any tool to edit the file. [10 points]

2. Create an XML file Services.xml for storing services information. Use the ASU Service Repository and other service directories to find at least five (5) services in each category of services: (1) SVC services, (2) RESTful services, and (3) other type of services like ASMX services, php services, and java services). For each service, at least two operations must be listed. Enter the service information into the Services.xml file. You can use any tool to edit the file. If an element has a Required Attribute, you must provide this attribute for this element. If an element has an Optional Attribute, you will provide this attribute for some elements and not provide this attribute for some other elements. [10 points]

3. Write the Services.xsl file for defining the HTML style for displaying of the services stored in Services.xml in a formatted table. You can use any tool to edit the file. [10 points]

Part 2 Creating Web Services to Process XML, XSD, and XSL Files (70 points)

Part 2 (Questions 4, 5, and 6) Due Date: Saturday, November 9, 2013, by 11:59pm (Arizona Time).

4. Develop a Web service (.svc) with two of the Web operations below. The node mentioned in the sub questions below include every component (element, content, and attribute) showing the XML tree above. You need to choose two operations to implement. If you implement more than two operations, we will grade the first two operations only. If you have implemented a service list below in your assignment 3, you cannot choose this service.

4.1 Web operation "verification" takes the URL of an XML (.xml) file and the URL of the corresponding XMLS (.xsd) file as input and validates the XML file against the corresponding XMLS (XSD) file. The Web method returns "No Error" or an error message showing the available information at the error point. You must use files that you create in the previous questions as the test case. However, your service operation should work for other test cases too. [20 points]

4.2 Web operation "transformation" takes the URL of an XML (.xml) file and the URL of the XSL file as inputs and generates the HTML file based on the XML and XSL files. The generated HTML file can be stored in a text file or in an .htm (or .html) file. You can also display the file in the GUI of your Web application (question 5). You must use files that you create in the previous questions as the test case. However, your service operation should work for other test cases too. [20 points]

4.3 Web operation "search" takes the URL of an XML (.xml) file and a keyword as input. It returns the node's information related to the keyword: [20 points]
(1) If the node has content, return the first content found. For example, if keyword = Name, return Crypto;
(2) If the node has no content, return all the child node's element names of the first element. For example, if keyword = Service, return Name, Provider, Operations, Category, WsdlUrl, and TryItUrl. Note that the attributes are considered child nodes too.
(3) If the node is the leaf node, return the parent node name. For example, if keyword = Encryption, return Name.

4.4 Web operation "XPathSearch" takes the URL of an XML (.xml) file and a path expression as input. It returns the path expression value of the given path. [20 points]

4.5 Web operation "searchWsdl" takes the URL of wsdl file. It reads the WSDL file as an xml file and returns the list operation names in the wsdl file. [20 points]

Notice that, for all the questions above, do not place the XSD file as a namespace in your XML file. It may cause an exception to some library classes. The absence of the XSD namespace will not *cause a problem with the schema validation, as your validation method will take two parameters as input: the XML file and the schema file.*

5. Create a Web site application (ASPX), and add the project into the same "solution" that hosts your web service. The Web site application must provide a GUI, which allows entering the required inputs, such as URLs and keyword, and path, based on the questions that you select. The GUI must have the buttons required to invoke the service operations, depending on the methods that you implement in the previous question, for example,

   - The button validates of the XML file against the schema file;
   - The button generates the HTML file;
   - The button searches by keyword or by path in the XML file.
   - The button searches the WSDL file

   The application must use the Web service created in question 4 to perform the required processing, and display the return message in the GUI. You can use a textbox, a list box, or a label to display the html file, or display it in a separate page. This assignment will be implemented on localhost, and you must use static URL for the services to ensure that the application and the services are still linked when the assignment is graded on a different computer. [20 points]

6. Testing: Based on your selection of the sub questions in question 4, provide test inputs and test results. For example, place the three files: Services.xml, Services.xsd, and Services.xsl, into a Web site and use them to test your program on your .Net development server. Inject an error in your XML file and make sure the validation service can detect the error. For question 6, you must submit the test results (inputs and outputs) in screenshots. For example [10 points]

   (1) A screenshot of the GUI, including the output of the XML validation displayed in the GUI, with no error and with error message;

   (2) A screenshot of the GUI, including the input and output for keyword search;

   (3) The Services.htm file (or Services.html) file generated.

   These three files above are not programs and can be deployed to any web location, such as the personal web space provided by ASU. See exercise 4 in Part 0. Do not use the server for assignment 3 in this assignment.

   After testing, make sure you remove these files from the Web site: Make sure the students in the same class will not see your assignment before the submission due date. When the TAs grade the assignment, they will place the files in a different location.

Submission Requirement

All submissions must be electronically submitted to the assignment folder where you downloaded the assignment paper. All files must be zipped into a single file.

Submission list: The complete solution folder with all project files for the service and the application, Services.xml, Services.xsd, Services.xsl, Services.htm (or Services.html) file generated, and the screenshot of the testing results in a Word or PDF file. Zip all these file into a zip file for submission.

Grading

The TA will grade your program following these steps:

(1) The TA will read your program and give points based on the points allocated to each component, the readability of your code (organization of the code and comments), logic, inclusion of the required functions, and correctness of the implementations of each function.

(2) Compile the code. If it does not compile, 40% of the points given in (1) will be deducted. For example, if you are given 20 points in step (1), your points will become 12 if the program fails to compile.

(3) If the code passes the compilation, the TA will execute and test the code. If, for any reason, the program gives an incorrect output or crashes for any input, 20% of the points given in (1) will be deducted.

Please notice that the TA will not debug your program to figure out how big or how small the error is. You may lose 40% or 20% of your points for a small error such missing a comma or a space!

Late submission deduction guidelines:

- For the first part of the submission, missing the deadline will result in a deduction of 5 points. No late submission for this part will be accepted. The submission folder will disappear after the deadline.
- For the second part, no penalty for late submissions that are received within 24 hours after the given deadline; and 1% grade deduction for every hour after the first 24 hours.