CSE 445 Project 4 (Assignments 7) 50Points
Summer 2020

Assignment 7 Question 1 Due: Saturday, June 27, 2020, by 11:59pm

Plus a one-day grace period.

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. You can use WebStrar to host your files and services. You can also use your ASU personal Web site space to host your XML, XSD, and XSL files.

Section I 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. Studying the material covered in these questions can help you to prepare for the exam and understand the homework assignment.

4. In this assignment, you can use WebStrar to host your XML files and services. You can also use your ASU personal Web hosting site to host the files. If you have not activated your file service and personal Web hosting site at ASU, you can activate them at: www.asu.edu/selfsub. Choose "Subscribe to additional computing access/services", and then chose "Personal Webpage Hosting". To access your personal Web site, you need to have SSH or PuTTY software to connect to general.asu.edu. You can also search for "Uploading Your Personal Web page" within the ASU search page to find the steps for uploading your files. Notice that the ASU Personal Web site space hosts files only. You can use it to host your .xml, .xsd, and .xsl files. It does not host programs such as Web services and Web applications. IIS are not installed. In order for your files to be accessible from the Web, you must put the files into the WWW directory, it its sub-directories.

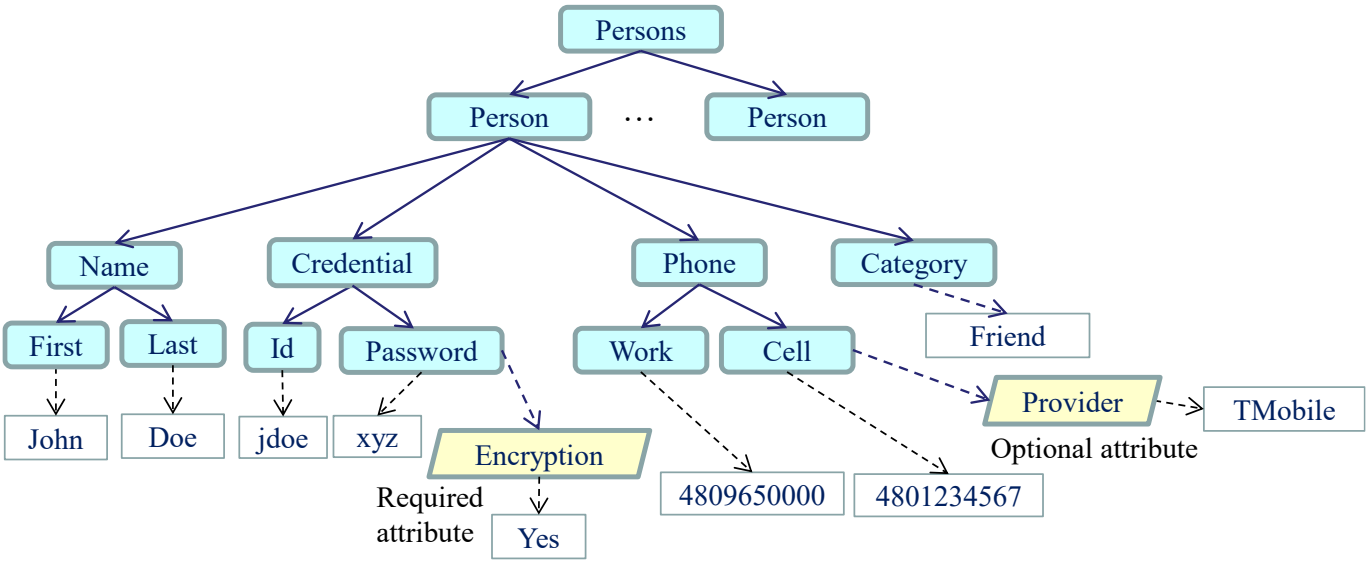
5 You can access the sample XML and XSD files at:

http://neptune.fulton.ad.asu.edu/WSRepository/xml/Courses.xml
http://neptune.fulton.ad.asu.edu/WSRepository/xml/Courses.xsd

Section II Project Questions (Submission Required)

Assignment 7

1. In this question, you will create a directory of persons, which can be represented as an XML tree. The diagram below shows the required structure of the directory of persons in XML tree notation that you will create in this assignment. All the "Person" elements have the same structure. Notice that different shapes of boxes have different meanings. They represent elements, attributes, and text contents/values, respectively. The structure of elements and attributes given in the diagram below are required to implement, while the given text contents/values in the diagram are examples. The Category element can take three content values: Family, Friend, or Work. You can define it as of enumeration type or simply as of string type. The solid arrows show parent-child element relations, and the dotted arrows show the element-content relations. Notice that the optional attribute "Provider" means that the XML instance can have the attribute or not have the attribute, without causing a verification error against its schema. However, it is still required to define the optional attribute in the XML Scheme file.



1.1 Write the Persons.xsd file that defines the XML schema allowing the structure shown in the diagram above. You can use any tool to create/edit the file. [10 points]

1.2 Create an XML file Persons.xml as an instance of your schema file. Define at least five (5) persons in each category: (1) Family, (2) Friend, and (3) Work. Enter the person information into the Persons.xml file. You can use any tool to edit the file. If an element has a Required Attribute, you must provide the attribute for each of the elements. If an element has an Implied (Optional) Attribute, you will provide this attribute for some elements, but not for all elements. [10 points]

1.3 As a test page (TryIt page), create an ASPX Web site application (not a Web service), which takes the URL of an XML file as input (via textbox) and displays the element (tag) names, text contents,

attribute names, and attribute values in the GUI page. You can define the order of data to be displayed. Make your Persons.xml as the default input. [20 points]

2. Develop a Web service (.svc) with **one** of the Web operations listed below. The node mentioned in the sub questions below includes every component (element, content, and attribute) shown in 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 and submitted a service listed below in your assignment 3 as an elective service, you cannot choose the same service here. However, in your Project 5 (next project), you can use the services developed in this assignment in addition to your elective services for the application that you have outlined in project 3. A complete application will be implemented in Project 5.

2.1 Web operation "verification" takes the URL of an XML (.xml) file and the URL of the corresponding XMLS (.xsd) file as inputs 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 created in the previous questions, with and without fault injection, as the test cases. However, your service operation should work for other test cases too. [10 points]

2.2 Web operation "search" takes the URL of an XML (.xml) file and a key (name) as input. It returns the node's content information related to the key: Name, Password, Phone and Provider, and Category. Your program must read the attribute Encryption of the Password element to determine if decryption is necessary. [10 points]

2.3 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. It could be a list of nodes, the content value, etc., depending on the path given. [10 points]

2.4 Web operation "addPerson" modifies the XML by adding a new person into the XML file. You can use multiple parameters or one parameter that contains all the information required for adding a new person into the tree. Your program must determine if encryption is necessary. [10 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.

3. Create an ASP .Net Web application, and add the project into the same "solution" that hosts your web service. The Web application must provide a GUI (TryIt Page), which allows entering the required inputs, such as URLs and keyword, path, or contents, based on the questions that you select. The GUI must have the buttons required to invoke the service operation, depending on the method that you implement in the previous question, for example,

- The button validates the XML file against the schema file.
- The button searches by keyword or by path in the XML file.
- The button adds a new person.

The Web application must use the Web services created in question 2 to perform the required processing, and display the return messages in the GUI. You can use a textbox, a list box, or a label to display the results. You can choose to implement this assignment on localhost (IIS Express) or on

WebStrar.

[20 points]

Deployment: In this assignment, you can choose to use localhost or use WebStrar. In both cases, you must submit the entire project in a single zip file into the Canvas submission site.

The entire project must include all project files of the service and the application, so that the project can be tested by the TA. You also need to submit the files used in the services, such as Persons.xml, Persons.xsd, Persons.xsl, and generated by the services, such as Persons.htm (or Persons.html), and the screenshot of the testing results in a Word or PDF file. Zip all these files in Assignments 7 and 8 into a zip file for submission.

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.

Canvas submission notice: The assignment consists of multiple distributed projects and components. They may be stored in different locations on your computer when you create them. You must copy these projects into a single folder for Canvas submission. To make sure that you have all the files included in the zip file and they work together, you must test them before submission. You must also download your own submission from the Canvas. Unzip the file on a different location or machine, and test your assignment and see if you can run the solution in a different location, because the TA will test your application on a different machine.

Grading and Rubrics

Each sub-question (component) has been assigned certain points. We will grade your programs following these steps:

(1) Compile the code. If it does not compile, 50% of the points given for the code under compilation will be deducted. Then, we will read the code and give points between 50% and 0, as shown in right part of the rubric table.

(2) If the code passes the compilation, we will execute and test the code using test cases. We will assign points based on the left part of the rubric table.

In both cases (passing compilation and failed compilation), we will read your program and give points based on the points allocated to each component (sub-question), 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.

Please notice that we will not debug your program to figure out how big or how small the error is. You may lose 50% of your points for a small error such missing a comma or a space!

We will apply the following rubrics to **each sub-question** listed in the assignment. Assume that points assigned to a sub-question is *pts*:

Rubric Table

Major	Code passed compilation				Code failed compilation		
Points	pts * 100%	pts * 90%	pts * 80%	pts * 70% - 60%	pts * 50% - 40%	pts * 30% - 10%	0
Each sub-question	Meeting all requirements, well commented, and working correctly in all test cases	Working correctly in all test cases. Comments not provided to explain what each part of code does.	Working with minor problem, such as not writing comments, code not working in certain uncommon boundary conditions.	Working in most test cases, but with major problem, such as the code fail a common test case	Failed compilation or not working correctly, but showing serious effort in addressing the problem.	Failed compilation, showing some effort, but the code does not implement the required work.	No attempt

Late submission deduction policy:

- No penalty for late submissions that are received within 24 hours of the given deadline;
- 2% grade deduction for every hour after the first 24 hours! No submission after Monday.