List of Required Services
CSE445/598 Service Development Assignment
Dr. Yinong Chen

Introduction

Several services to be developed will use a service in ASU Service Repository called Web2String. The service URL is:

http://venus.eas.asu.edu/WsRepository/Services/Web2StringSVC/Service.svc

This service will download the Web page at the given URL and return the text in the page as a string. The operation offered by this service is: `string GetWebContent(string url);`

A test page of the service is available at:

http://venus.eas.asu.edu/wsRepository/Services/Web2StringSVC/tryit.aspx

The service and the test page above are also listed in Table 2 of the ASU Repository at

http://venus.eas.asu.edu/WSRepository/repository.html

Some of the service requirements listed below are related. Combination of the related services can form a coherent application. Some of the service requirements are independent of each other.

Service 1. Top10Words

Description: Analyze the webpage at a given url and return the ten most-frequently occurred words in the webpage. Return the words in the descending order of their appearing frequencies.

Operation: string[] Top10Words(string url)

Input: A webpage url in string.

Output: An array of strings that contains the ten most-frequently occurred words in descending order of their frequencies.

Service 2. WordFilter

Description: Analyze a string of words and filter out the function words (stop words) such as “a”, “an”, “in”, “on”, “the”, “is”, “are”, “am”, as well as the element tag names and attribute names quoted in angle brackets < … >, if the string represents an XML page or HTML source page.

Operation: string WordFilter(string str)

Input: A string.

Output: A string with the stop words removed.

Service 3. Stemming

Description: Analyze a string of words and replace the inflected or derived words to their stem or root word. For example, “information”, “informed”, “informs”, “informative” will be replaced by the tem word “inform”. This service can help find useful keywords or index words in information processing and retrieval.

Operation: string Stemming(string str)

Input: A string.

Output: A string the inflected or derived words replaced by their stem word.

Service 4. Top10ContentWords

Description: Analyze the webpage at a given url and return the ten most-frequently occurred “content” words in the webpage. Return the words in descending order of their appearing frequencies. Content words do not include the stop words and the element tag names and attribute names quoted in angle brackets < … >, if the string represents an XML page or HTML source page, as well as other words that do not represent the content of the webpage. The ranking of the words should be used on the string that have been processed by stemming service.

Operation: string[] Top10ContentWords(string url)

Input: A webpage url in string.

Output: An array of strings that contains the ten most-frequently occurred words in descending order of their frequencies. The required words must not be element tag name or attribute name of XML page or HTML source page.

Service 5. WsdlAddress

Description: Analyze a webpage and return all WSDL addresses in that webpage in an array of strings. The WSDL address can have these formats: xxx.wsdl (e.g. developed Java), or ?wsdl (e.g., .php?wsdl, .svc?wsdl and .asmx?wsdl).

Operation: string[] getWsdlAddress(string url)

Input: A webpage url.

Output: An array of strings that contains WSDL addresses in the given webpage.

Service 6. WsdlDiscovery

Description: Call Google or Bing search engine APIs, using keywords such as .wsdl, .php?wsdl, .svc?wsdl and .asmx?wsdl, to discover the Web pages that contain WSDL addresses, and follow the address to discover WSDL files. You can find

Google search APIs in Google code, and you can find Bing search APIs in MSDN library: Bing SOAP Services: http://msdn.microsoft.com/en-us/library/cc966738.aspx or http://msdn.microsoft.com/en-us/library/dd251056.aspx. For example, when I use these ".wsdl", ".php?wsdl", ".svc?wsdl" ".asmx?wsdl", as search keywords, I find pages with the required wsdl files, including the one below, which has a long list of WSDL files:

http://data.serviceplatform.org/servicefinder/sf-wsdls.list.sorted

Notice that you need to analyze the addresses returned from the search engine and remove those that are not wsdl addresses. The service should return only those addresses that are wsdl addresses. If you read service 7, you can see that the addresses will be given to another service to discover the operations in the wsdl file.

Operation: string[] WsdlDiscovery(string keyWords)

Input: A sting of keywords related wsdl address format

Output: An array of strings that contains WSDL addresses discovered through the keywords search engines.

Service 7. WsOperations

Description: Analyze a WSDL file in XML format and return operation names and their corresponding input parameter and return types.

Operation: string[] getWsOperations(string url)

Input: The url address that points to an WSDL file

Output: An array of strings. Each of the array elements contains return the type, operation name, and parameter types

Service 8. WsHashOperations

Description: Analyze a WSDL file in XML format and return operation names and their corresponding input parameter types and return type in a Hashtable or a similar data structure, for example, Dictionary<object, object>. The format should look like: return type, Input parameter1, parameter2, etc.

Operation: Dictionary<object, object> WsHashOpertions(string wsdlUrl)

Input: The wsdlUrl address that points to an WSDL file

Output: An object of Dictionary<object, object> that contains operation names, input and output parameter types.

Service 9. WsTesting

Description: Given a WSDL address, call all the service operations in the service. The service should first obtain the list of operations and generate their input/output values, for example, calling the service “WsHashOperations”, and then, invoke each operation. The return values must be encoded into an aggregate type, for example, an XML, string, and array types.

Operation: UserType WsTesting(string wsdlUrl)

Input: The wsdlUrl address that points to an WSDL file.

Output: UserType that wraps the return result.

Service 10. GroupTesting

Description: Given an array of WSDL addresses, call the service operations of each service. These services are supposed to implement the same function. The testing service will validate whether the opeartions indeed generate the same output for the same input.

Operation: boolean GroupTesting(string[] wsdlUrl,)

Input: The wsdlUrl addresses that point to an WSDL files.

Output: a true or false value indicating whether the services give the same output or not.

Service 11. NewsFocus

Description: Find news about specific topics, for example, find all (as many as possible) news articles about ASU (Arizona State University).

Operation: string[] NewsFocus(string[] topics)

Input: a list of topics or key words

Output: A list of URLs in which the given topics are reported.

Service 12. Loyal Service

Description: Explore and call the services available at http://loyals.azurewebsites.net/Info/api to provide an additional service. The service page is created by Joshua Claxton, a formal CSE445 student, in the framework of his Honors Thesis 2013.

The services that you will call are

int getHighestPoints(): it returns the highest loyalty score in the system;

in getUserCountByPoints(int score): It returns the number of users who have the given score.

Your service will add a function to the existing services.

Operation: string[] loyaltyDistribution(int step);

It calls the getUserCountByPoints(int score) in a loop: for (i = 0; i<=HighestPoints; i=i++) { … }

Input: a positive integer

Output: A string that lists the scores in the interval between [0, step-1], [step, 2*step-1], [2*step, 3*step-1], …, [n*step, HighestPoints]

Service 13. Weather Service

Description: Create a 5-day weather forecast service of zipcode location based on the national weather service at: http://graphical.weather.gov/xml/SOAP_server/ndfdXMLserver.php?wsdl.

Operation: string[] Weather5day(string zipcode)

Input: a U.S. zipcode

Output: An array (or list) of strings, storing 5-day weather forecast for the given zipcode location.

Hint, the given national weather service has an operation: `LatLonListZipCode(zipcode);` It returns an XML file containing the latitude and longitude of the zipcode location.
For example, `LatLonListZipCode(85281)` will return

```
"<?xml version='1.0'?>
<dwml version='1.0' xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:noNamespaceSchemaLocation='http://graphical.weather.gov/xml/DWMLgen/schema/DWM
L.xsd'><latLonList>33.4357,-111.917</latLonList></dwml>"
```

Then, you can use latitude and longitude to call the operation to obtain the 5-day forecast. The available operations in this service is listed and explained at:
http://graphical.weather.gov/xml/SOAP_server/ndfdXMLserver.php

Service 14. Solar Energy Service

Description: Create a service that returns the annual average sunshine index of a given position (latitude, longitude). This service can be used for deciding if installing solar energy device is effective at the location.

Operation: decimal SolarIntensity(decimal latitude, decimal longitude)

Input: latitude and longitude

Output: a number reflecting the annual average solar intensity at the location.

Service 15. Wind Energy Service

Description: Create a service that returns the annual average wind index of a given position (latitude, longitude). This service can be used for deciding if installing windmill device is effective at the location.

Operation: decimal WindIntensity(decimal latitude, decimal longitude)

Input: latitude and longitude

Output: a number reflecting the annual average wind intensity at the location.

Service 16. Natural Hazards Service

Description: Create a service that returns the natural hazards (Tsunamis, earthquake, volcanoes) index of a given position (latitude, longitude). This service can be used for building decision and insurance premium.

Operation: decimal NaturalHazards(decimal latitude, decimal longitude)

Input: latitude and longitude

Output: a number reflecting the natural hazards at the location.

Hint: getting data from NOAA (National Geophysical Data Center): http://www.nesdis.noaa.gov/, http://maps.ngdc.noaa.gov/, and http://gosic.org/

In the site, you can search keyword API.

Service 17. Number2Words

Description: Convert a number, e.g., a phone number, into an easy-to-remember character/digit string. All valid words, e.g., helloClass are easy to remember. Other commonly used character/digit strings, such as CSE100, CSE445, and SCIDSE, are easy to remember strings too.

Operation: string Number2Words(string number)

Input: a string of digits (0, 1, 2, …, 9).

Output: a character/digit string, which are easy-to-remember and each character corresponds to the input digits by the definition of a standard phone touch key system:
0 = none
1 = none
2 = ABC
3 = DEF
4 = GHI
5 = JKL
6 = MNO
7 = P(Q)RS
8 = TUV
9 = WXYZ

If the input contains 0 or 1, these two characters will remain 0 and 1. All other digits must be translated into letter, unless the digits make better sense, e.g., CSE445.

In order to implement the service, you must have a set of easy to remember words of your own, such as CSE445598, ASU, SCIDSE. You also need a dictionary. You could use the Web2String service to read a set of words from a public dictionary service.

Service 18. Find the Nearest Store

Description: Use an existing online service or API to find the provided storeName closest to the zipcode and return the address. If no store is found, return an error message. (Optional: if the store is further than 20 miles, from the zipcode, return a “no stores within 20 miles” message). .

Operation: Operation: string findNearestStore(string zipcode, string storeName)

Input: two strings

Output: string message