


1	 UNIVERSITY OF LEEDS	School of Computing University of Leeds Coursework 2 - Report	Module Code XJCO3211
---	---	--	---

Web Services Composition
Submission Deadline Date: 30/10/2022

Student Name	Username	ID
1 Leshan Wang	sc19l2w@leeds.ac.uk	2019110023
2 Yujia Lei	sc19yl3@leeds.ac.uk	2019110073

Composition of Originality (10 marks)

Describe what the Web services composition does

Web service composition is the mechanism for combining and reusing existing web services to create value-added Web services. It allows different organizations or applications from multiple sources to communicate without the need to share sensitive data or IT infrastructure. Instead, all information is shared through a programmatic interface across a network.

In this particular service, there are one service requester, which is the client; and three service providers, which are three REST server. After sending certain requests, the client will get keywords from one server and passed them on as parameters to the next, and finally achieve valuable results.

The client: get input from user and parsed it with API, then send request to server, receive data that are parsed and displayed.

Server 1: receive request from client, get the parameter (subject) as key to conduct search inside its database, after getting results, return them (book title and book ID) to client.

Server 2: receive request from client, get the parameter (book ID) as key to conduct search inside its database, after getting results, return them (book title and book ISBN) to client.

Server3 (Google Book): receive request from client, get the parameter (book ISBN) as key to conduct search inside its database, after getting results, return detailed book information (publication, author, page count....) to client.

Provide details in the table below

Web Service	Own or External	Input	Output	Output Parsing / Extract something of interest

1	Own	ios	99 iOS Application Security	Book ID is 99 Book title is iOS Application Security
2	Own	99	1593276011	Book ISBN is 1593276011
3	External	1593276011	The detailed information of the book	The Book tile is iOS Application Security Author is David Thiel Page count is 0 Publish date is 2016-02-16

1st Web Service (20 marks)

Name of student in charge:

1st Web service	<i>Fill in this table</i>
Name of service	Server 1 (Book ID Fetcher)
SOAP-based or RESTful	RESTful
Brief description	Call out the ID of the relevant type of book according to the discipline classification entered by the user

Server design

Server 1 contains a simple data base, which includes information about different subjects such as python and iOS, the exact valuable information is book title and its ID. Then a class called 'Book' is created, which defines a function, it takes book_id as parameter, which is the command passed by the client. After getting book_id, it is used to traverse through database to find the result; if nothing is found, the operation will abort, given error code 404 and proper message. Finally, the server is set to run at port 80.

Server implementation

Python Flask RESTful is a Flask extension that adds support for quickly building REST APIs. It is also a lightweight extension that can work with your existing ORM/library. Flask RESTful encourages best practices with minimal settings. It can be used to quickly integrate restful API interface functions. In the background of the system's pure API, this plug-in can save a lot of time. In addition, data in JSON format is returned in each request.

```
parser = reqparse.RequestParser()
parser.add_argument('title', required=True)
parser.add_argument('ID', type=int, required=False)
```

Create a resolution object to resolve the parameters passed in by the server.

```
books = {
    'python': {
        'title': 'Python Crash Course',
        'ID': 97
    },
    'ios': {
        'title': 'iOS Application Security',
        'ID': 99
    },
    'vs': {
        'title': "Visual Studio.Net Developer's Guide",
        'ID': 121
    }
}
```

Server (1) built-in database.

```
def get(self, book_id):
    if book_id not in books:
        abort(404, message=f"Book {book_id} not found!")
    return books[book_id]
```

Define the function get method, accept the input parameter as the key, and traverse in the server database. If successful, return the content to the client. If it fails, the abort function will actively throw a 404 exception.

```
api.add_resource(Book, '/books/<book_id>')
```

Register route.

Explain how the service is invoked. You may include relevant snippet of source code

```
command0 = input("Please enter the subject => ").lower().strip()
api0 = "http://127.0.0.1:80/books/"
#resp0 = urlopen(api0 + command0)
resp0 = requests.get(api0 + command0)
print("The response time is => " + str(resp0.elapsed.microseconds / 1000) +
      " ms")
data0 = json.loads(resp0.text)
id = data0["ID"]
title = data0["title"]
print("The book is: " + str(title) + ", book ID is " + str(id) +
      '\n*****\n')
f = open("./result1.json", "w")
f.write(str(id) + "\n")
f.write(str(title) + "\n")
f.close()
```

Assemble an API instruction at the client, and the instruction is passed to the server. The server parses the instruction and extracts the name of the data subset as the traversal variable. All data exists in the data set of the server: the title and ID of the book. The server sends the data back to the client, and the client parses the data and prints it to the console.

Include evidence of its execution through a client, e.g. screen shot

```
PS C:\Users\Shadows\Desktop\cw2> cd c:/Users/Shadows/Desktop/cw2
PS C:\Users\Shadows\Desktop\cw2> & c:/python310/python.exe c:/Users/Shadows/Desktop/cw2/client1.py
Please enter the subject => python
The response time is => 3.204 ms
The book is: Python Crash Course, book ID is 97
*****
```

Measure the service invocation time. You are expected to run the experiments n times (e.g. $n = 5$). A statistical analysis (average, standard deviation) is expected.

Run No.	Service Invocation time
1	2.699

2	2.256
3	2.065
4	2.373
5	2.236
Average	2.326
Standard Deviation	0.210931

Explain how you have obtained these measurements

```
resp0 = requests.get(api0 + command0)
print("The response time is => " + str(resp0.elapsed.microseconds / 1000) +
      " ms")
```

The invocation time of Server1 is implemented in client application, by using requests function: requests.elapsed.microseconds. Each time when Server1 is invoked, the function will get the time cost and displayed on console.

2nd Web Service (20 marks)

Name of student in charge:

2nd Web service	<i>Fill in this table</i>
Name of service	Server 2 (Book ISBN Fetcher)
SOAP-based or RESTful	RESTful
Brief description	Call out the ISBN of the book according to the ID entered by the user

Server design

Server 2 also contains a simple data base, which includes information about different books with their title and ISBN code, the code will be used to fetch detailed information from Google Book Api when connecting to the third REST server.

A class called 'ISBN' is created, which defines a function, it takes isbn_id as parameter, which is the command passed by the client. After getting isbn_id, it is used to traverse through database to find the result; if nothing is found, the operation will abort, given error code 404 and proper message. Finally, the server is set to run at port 8080.

Server implementation

Python Flask RESTful is a Flask extension that adds support for quickly building REST APIs. It is also a lightweight extension that can work with your existing ORM/library. Flask RESTful encourages best practices with minimal settings. It can be used to quickly integrate restful API interface functions. In the background of the system's pure API, this plug-in can save a lot of time. In addition, data in JSON format is returned in each request.

```
parser = reqparse.RequestParser()
parser.add_argument('title', required=True)
parser.add_argument('ID', type=int, required=False)
```

Create a resolution object to resolve the parameters passed in by the server.

```
data = {
    '97': {
        'title': 'Python Crash Course',
        'ISBN': 1593276036
    },
    '99': {
        'title': 'iOS Application Security',
        'ISBN': 1593276011
    },
    '121': {
        'title': "Visual Studio.Net Developer's Guide",
        'ISBN': 1593270231
    }
}
```

Server (2) built-in database.

```
def get(self, isbn_id):
    try:
        return data[isbn_id]
    except:
        abort(404, message=f"ISBN {isbn_id} not found!")
```

Define the function get method, accept the input parameter as the key, and traverse in the server database. If successful, return the content to the client. If it fails, the abort function will actively throw a 404 exception.

```
api.add_resource(ISBN, '/ISBN/<isbn_id>')
```

Register route.

Explain how the service is invoked. You may include relevant snippet of source code

```
command1 = input("Please enter book ID => ").lower().strip()
api1 = "http://127.0.0.1:8080/ISBN/"
resp1 = requests.get(api1 + command1)
print("The response time is => " + str(resp1.elapsed.microseconds / 1000) +
      " ms")
data1 = json.loads(resp1.text)
isbn = data1["ISBN"]
print("The ISBN of this book is: " + str(isbn) + '\n*****\n')
f = open("./result2.json", "w")
f.write(str(isbn) + "\n")
f.close()
```

Assemble an API instruction at the client, and the instruction is passed to the server. The server parses the instruction and extracts the name of the data subset as the traversal variable. All data exists in the data set of the server: the title and ISBN of the book. The server sends the data back to the client, and the client parses the data and prints it to the console.

Include evidence of its execution through a client, e.g. screen shot

```
Please enter book ID => 97
The response time is => 2.872 ms
The ISBN of this book is: 1593276036
*****
```

Measure the service invocation time. You are expected to run the experiments n times (e.g. $n = 5$). A statistical analysis (average, standard deviation) is expected.

Run No.	Service Invocation time
1	2.285

2	2.047
3	2.330
4	2.090
5	2.147
Average	2.180
Standard Deviation	0.109912

Explain how you have obtained these measurements

```
resp1 = requests.get(api1 + command1)
print("The response time is => " + str(resp1.elapsed.microseconds / 1000) +
      " ms")
```

The invocation time of Server2 is implemented in client application, by using requests function: requests.elapsed.microseconds. Each time when Server1 is invoked, the function will get the time cost and displayed on console.

3rd Web Service - External Service (15 marks)

3 rd Web service	Fill in this table
Name	Google Books API
SOAP-based or RESTful	RESTful
Name of publisher, e.g. Google, Twitter ...	Google
Brief description	With the Google Books API, we can programmatically perform most of the operations that can be performed interactively on the Google Books website, such as searching and browsing the list of books matching a given query, and viewing information about books, including metadata, inventory status, and price.
URL	https://www.googleapis.com/books/v1/volumes?q=isbn: Add ISBN after:, such as 1593276036

Explain how it is invoked. You may include relevant snippet of source code

Assemble an API directive on the client that contains a link and an ISBN, and the directive is passed to the Google server, which sends the JSON data back to the client. The client parses the JSON data and extracts information according to different categories, such as author, publication date, and so on. In RESTful systems, resources are stored in the data store; Clients send requests that require the server to perform a specific action, such as creating, retrieving, updating, or deleting a resource, and the server performs that action and sends a response, usually in the form of a specified resource.

```
api = "https://www.googleapis.com/books/v1/volumes?q=isbn:"
isbn = input("Enter 10 digit ISBN => ").strip()
startTime = datetime.now()
resp = urlopen(api + isbn)
endTime = datetime.now()
span = (endTime - startTime).total_seconds()
print("The response time is => " + str(span * 1000) + " ms")

book_data = json.load(resp)
volume_info = book_data["items"][0]["volumeInfo"]
author = volume_info["authors"]
prettyfy_author = author if len(author) > 1 else author[0]
```

```
api0 = "http://127.0.0.1:80/books/"
#resp0 = urlopen(api0 + command0)
resp0 = requests.get(api0 + command0)
```

Command0 is the user input, which is spliced with api0 as the final api. The requests.get function calls the server data, and the original data is stored in resp0.

```
resp0.elapsed.microseconds / 1000
```

The method attached to Requests is used to measure the invocation time.

```
data0 = json.loads(resp0.text)
id = data0["ID"]
title = data0["title"]
```

Convert the original data returned by the server to the json format, and parse the content according to different index keys.

```
f = open("./result1.json", "w")
f.write(str(id) + "\n")
f.write(str(title) + "\n")
f.close()
```

Write the parsed data to the local backup.

```
resp = urlopen(api + isbn)
```

Because Google Server uses proxy calls, the requests function will make errors, so use the Uropenfunction to call.

Include evidence of its execution through a client, e.g. screen shot

```
Enter 10 digit ISBN => 1593276036
The response time is => 785.74 ms

Title: Python Crash Course
Author: Eric Matthes
Page Count: 564
Publication Date: 2015-11-20

*****

Would you like to conduct another search? y or n => |
```

Measure the service invocation time. You are expected to run the experiments n times (e.g. $n = 5$). A statistical analysis (average, standard deviation) is expected.

Run No.	Service Invocation time
1	776.639
2	2180.680
3	674.759
4	694.427
5	656.740
Average	996.649
Standard Deviation	593.434621

Explain how you have obtained these measurements

```
startTime = datetime.now()
resp = urlopen(api + isbn)
endTime = datetime.now()
span = (endTime - startTime).total_seconds()
print("The response time is => " + str(span * 1000) + " ms")
```

The invocation time of Google Server is implemented client application, different from previous ones, due to using proxy to get data from Google, the requests.elapsed function generates incompatible failure with python version 3.8. Therefore, we use the difference between current time of the start of invocation, and current time of the completion of invocation as the estimated time elapsed. This involved using function datetime.now().

Web Services Integration (10 marks)

Provide details of the Web services integration

Website integration is when your website sends or receives information from another website or application. It means that rather than your website being an isolated system that doesn't talk to anything else, it can make friends, connecting with other key systems to save time and give other benefits.

//todo

Integrated Client runs in the console, Web Service1 accepts the Input 1 (Subject) assembly API and the title and ID of the Output 1 book to the server, The title and ID are returned to the client as data, the client extracts and returns data to print in the console as Input 2 (ID) of Web Service 2, also assembles the API and the title and ISBN of the Output 2 book to the server, the title and ISBN are returned to the client as data, the client extracts and returns data to print in the console as Input 3 (ISBN) of the Web Service 2, Also assemble the same API and Output 3 final result of composition to the server, final result of composition is returned to the client as data and the client extracts and returns the data to print in the console.

The three specific integrated servers are located in the client. They are in turn: acquiring input (instructions), assembling apis, acquiring data, parsing data, printing data, and then acquiring input for api request data.

Web User Interface (10 marks) – if attempted

Provide details of your Web-based application (Servlets/JSP/Other Frameworks)

Console

Console interfaces save computer system resources than GUI, do not stop responding because threads are blocked, and operate faster.

```
PS C:\Users\Shadows\Desktop\cw2> cd c:/Users/Shadows/Desktop/cw2
PS C:\Users\Shadows\Desktop\cw2> cd c:/Users/Shadows/Desktop/cw2
PS C:\Users\Shadows\Desktop\cw2> & c:/python310/python.exe c:/Users/Shadows/Desktop/cw2/client1.py
Please enter the subject => ios
The response time is => 2.644 ms
The book is: i05 Application Security, book ID is 99
*****

Please enter book ID => 99
The response time is => 2.09 ms
The ISBN of this book is: 1593276011
*****

Enter 10 digit ISBN => 1593276011
The response time is => 694.427 ms

Title: i05 Application Security
Author: David Thiel
Page Count: 0
Publication Date: 2016-02-16

Enter 10 digit ISBN => 1593276036
The response time is => 785.74 ms

Title: Python Crash Course
Author: Eric Matthes
Page Count: 564
Publication Date: 2015-11-20
*****

Would you like to conduct another search? y or n => 
```

Successful Execution (10 marks)

Include evidence of the Web services integration execution, e.g. screen shot

```
PS C:\Users\Shadows\Desktop\cwk2> c:
PS C:\Users\Shadows\Desktop\cwk2> cd c:/Users/Shadows/Desktop/cwk2
PS C:\Users\Shadows\Desktop\cwk2> & c:/python310/python.exe c:/Users/Shadows/Desktop/cwk2/client1.py
Please enter the subject => ios
The response time is => 2.644 ms
The book is: iOS Application Security, book ID is 99
*****

Please enter book ID => 99
The response time is => 2.09 ms
The ISBN of this book is: 1593276011
*****

Enter 10 digit ISBN => 1593276011
The response time is => 694.427 ms

Title: iOS Application Security
Author: David Thiel
Page Count: 0
Publication Date: 2016-02-16

Enter 10 digit ISBN => 1593276036
The response time is => 785.74 ms

Title: Python Crash Course
Author: Eric Matthes
Page Count: 564
Publication Date: 2015-11-20

*****

Would you like to conduct another search? y or n => 
```

Other Comments

Include any details you feel are relevant.

Http is currently the most used protocol on the Internet. It is a traditional API interface. REST is resource oriented, and resources are exposed through URIs. The URI is designed to expose resources in a reasonable way. The operation of a resource is independent of it. The operation is embodied by HTTP verbs. When REST exposes a resource through a URI, it will emphasize not to use verbs in the URI. The REST API is based on HTTP. REST makes good use of some features of HTTP itself, such as HTTP verbs, HTTP status codes, HTTP headers, and so on. REST is a standard and a specification. Following the REST style can make the developed interface universal and facilitate the caller to understand the role of the interface. Because it restricts URIs to only define resources. It looks easy to understand. It is easy to understand the addition, deletion, modification, and query of simple objects. Flask RESTful is a leader in Flask extensions. It adds support for rapid construction of RESTful APIs, encapsulates Flask, and makes it easier, faster, and more convenient to develop RESTful APIs.