| | | |
|---|---|---|
| UNIVERSITY OF LEEDS | School of Computing<br>University of Leeds<br>**Coursework 2 - Report** | **Module Code**<br>**XJCO3211** |

# Web Services Composition

Submission Deadline Date: 30/10/2022

| **Student Name** | | **Username** | **ID** |
|---|---|---|---|
| **1** | Zhu Haining | sc19hz | 201388419 |
| **2** | Sun Jiahong | sc19jhs | 201388935 |

## Composition of Originality (10 marks)

*Describe what the Web services composition does*

*Provide details in the table below*

| Web Service | Own or External | Input | Output | Output Parsing / Extract something of interest |
|---|---|---|---|---|
| **1** | **Own** | **Date and Province where people would like to travel** | **The name of the city that is recommended** | **The name of the city that is recommended** |
| **2** | **Own** | **The name of the city** | **City id** | **City id** |
| **3** | **External** | **City id** | **A set of data returned by the api.** | **The weather, air condition, tips,etc.** |

## 1st Web Service (20 marks)

*Name of student in charge:*

| 1st Web service | *Fill in this table* |
|---|---|
| **Name of service** | **Service0** |
| **SOAP-based or RESTful** | **RESTFUL** |
| **Brief description** | **This service interacts directly with the user, receives the user's data (time, province) and gives recommended cities to travel.** |

*Server design*

Input: date, province

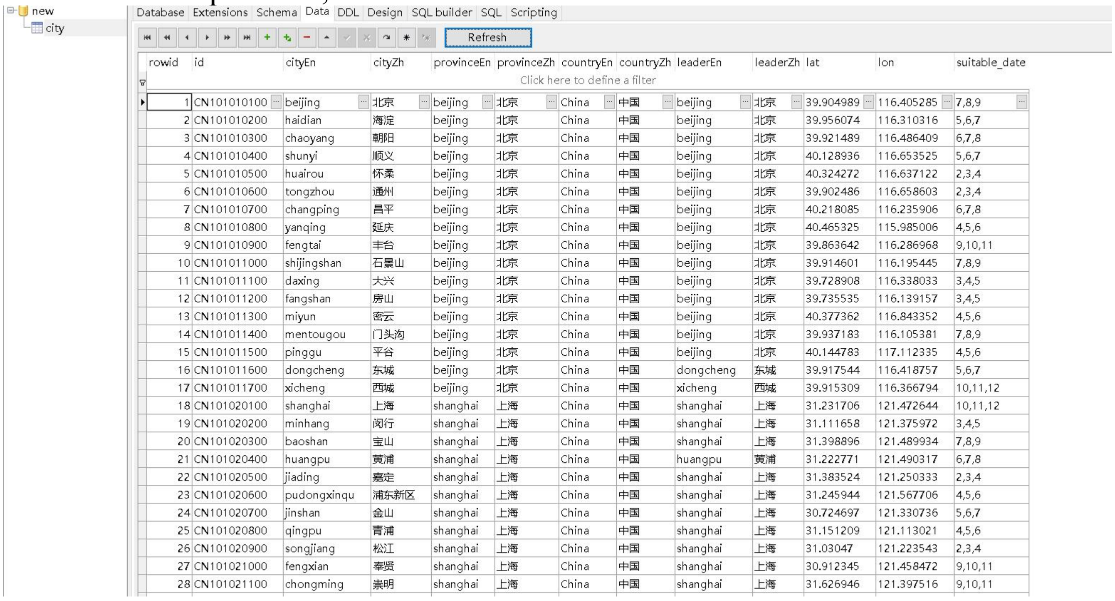
Output: The name of the City

Theory:  Find relevant information from the database

The api receives parameters by url.

*Server implementation*

Firstly, implement a database. The important attributes contain the name of the city, the name of the province, suitable date to travel.



| rowid | id | cityEn | cityZh | provinceEn | provinceZh | countryEn | countryZh | leaderEn | leaderZh | lat | lon | suitable_date |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | CN101010100 | beijing | 北京 | beijing | 北京 | China | 中国 | beijing | 北京 | 39.904989 | 116.405285 | 7,8,9 |
| 2 | CN101010200 | haidian | 海淀 | beijing | 北京 | China | 中国 | beijing | 北京 | 39.956074 | 116.310316 | 5,6,7 |
| 3 | CN101010300 | chaoyang | 朝阳 | beijing | 北京 | China | 中国 | beijing | 北京 | 39.921489 | 116.486409 | 6,7,8 |
| 4 | CN101010400 | shunyi | 顺义 | beijing | 北京 | China | 中国 | beijing | 北京 | 40.128936 | 116.653525 | 5,6,7 |
| 5 | CN101010500 | huairou | 怀柔 | beijing | 北京 | China | 中国 | beijing | 北京 | 40.324272 | 116.637122 | 2,3,4 |
| 6 | CN101010600 | tongzhou | 通州 | beijing | 北京 | China | 中国 | beijing | 北京 | 39.902486 | 116.658603 | 2,3,4 |
| 7 | CN101010700 | changping | 昌平 | beijing | 北京 | China | 中国 | beijing | 北京 | 40.218085 | 116.235906 | 6,7,8 |
| 8 | CN101010800 | yanqing | 延庆 | beijing | 北京 | China | 中国 | beijing | 北京 | 40.465325 | 115.985006 | 4,5,6 |
| 9 | CN101010900 | fengtai | 丰台 | beijing | 北京 | China | 中国 | beijing | 北京 | 39.863642 | 116.286968 | 9,10,11 |
| 10 | CN101011000 | shijingshan | 石景山 | beijing | 北京 | China | 中国 | beijing | 北京 | 39.914601 | 116.195445 | 7,8,9 |
| 11 | CN101011100 | daxing | 大兴 | beijing | 北京 | China | 中国 | beijing | 北京 | 39.728908 | 116.338033 | 3,4,5 |
| 12 | CN101011200 | fangshan | 房山 | beijing | 北京 | China | 中国 | beijing | 北京 | 39.735535 | 116.139157 | 3,4,5 |
| 13 | CN101011300 | miyun | 密云 | beijing | 北京 | China | 中国 | beijing | 北京 | 40.377362 | 116.843352 | 4,5,6 |
| 14 | CN101011400 | mentougou | 门头沟 | beijing | 北京 | China | 中国 | beijing | 北京 | 39.937183 | 116.105381 | 7,8,9 |
| 15 | CN101011500 | pinggu | 平谷 | beijing | 北京 | China | 中国 | beijing | 北京 | 40.144783 | 117.112335 | 4,5,6 |
| 16 | CN101011600 | dongcheng | 东城 | beijing | 北京 | China | 中国 | dongcheng | 东城 | 39.917544 | 116.418757 | 5,6,7 |
| 17 | CN101011700 | xicheng | 西城 | beijing | 北京 | China | 中国 | xicheng | 西城 | 39.915309 | 116.366794 | 10,11,12 |
| 18 | CN101020100 | shanghai | 上海 | shanghai | 上海 | China | 中国 | shanghai | 上海 | 31.231706 | 121.472644 | 10,11,12 |
| 19 | CN101020200 | minhang | 闵行 | shanghai | 上海 | China | 中国 | shanghai | 上海 | 31.111658 | 121.375972 | 3,4,5 |
| 20 | CN101020300 | baoshan | 宝山 | shanghai | 上海 | China | 中国 | shanghai | 上海 | 31.398896 | 121.489934 | 7,8,9 |
| 21 | CN101020400 | huangpu | 黄浦 | shanghai | 上海 | China | 中国 | huangpu | 黄浦 | 31.222771 | 121.490317 | 6,7,8 |
| 22 | CN101020500 | jiading | 嘉定 | shanghai | 上海 | China | 中国 | shanghai | 上海 | 31.383524 | 121.250333 | 2,3,4 |
| 23 | CN101020600 | pudongxinqu | 浦东新区 | shanghai | 上海 | China | 中国 | shanghai | 上海 | 31.245944 | 121.567706 | 4,5,6 |
| 24 | CN101020700 | jinshan | 金山 | shanghai | 上海 | China | 中国 | shanghai | 上海 | 30.724697 | 121.330736 | 5,6,7 |
| 25 | CN101020800 | qingpu | 青浦 | shanghai | 上海 | China | 中国 | shanghai | 上海 | 31.151209 | 121.113021 | 4,5,6 |
| 26 | CN101020900 | songjiang | 松江 | shanghai | 上海 | China | 中国 | shanghai | 上海 | 31.03047 | 121.223543 | 2,3,4 |
| 27 | CN101021000 | fengxian | 奉贤 | shanghai | 上海 | China | 中国 | shanghai | 上海 | 30.912345 | 121.458472 | 9,10,11 |
| 28 | CN101021100 | chongming | 崇明 | shanghai | 上海 | China | 中国 | shanghai | 上海 | 31.626946 | 121.397516 | 9,10,11 |

The sql for this search is *SELECT cityZh FROM city where provinceZh=…and suitable_date like …*

Finally, return the first row of the results with method get as a  string type.

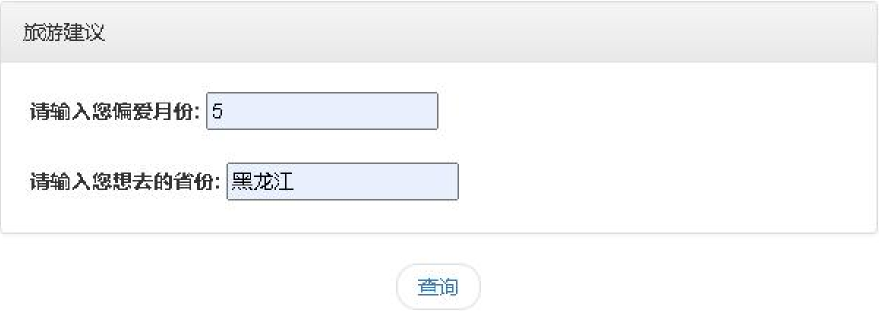
*Explain how the service is invoked. You may include relevant snippet of source code*

The api is invoked by url as: 127.0.0.1:5000/<date>/<province>, and use get to get the return value.

The relevant code is:

```
$(document).ready(function () {
            $("#search1").click(function () {
                var month = $("#month").val();
                var city = $("#city").val();
                var url = "http://localhost:5000/" + month + '/' + city
                $.get(url, function (data, status) {
                    alert("查询结果： " + "成功");
                    if (status == 'success') {
                        $("#first").css({ "display": "none" })
                        $("#second").css({ "display": "block" })
                        $("#service2").val(data)
                    }
                });
            });
```

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

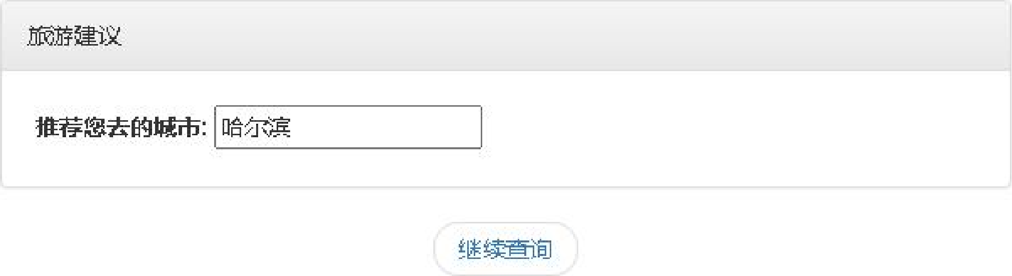


旅游建议

请输入您偏爱月份: 5

请输入您想去的省份: 黑龙江

↓



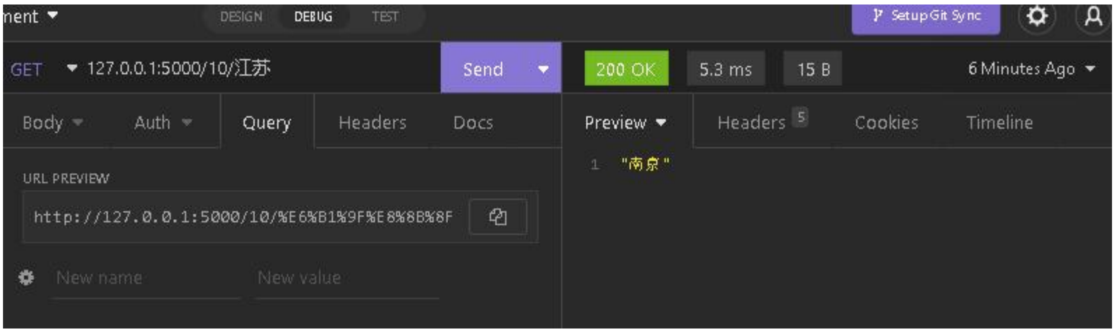
旅游建议

推荐您去的城市: 哈尔滨

*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** | 7.51ms |
| **2** | 5.58ms |
| **3** | 15.7ms |
| **4** | 16.6ms |
| **5** | 5.3ms |
| **Average** | 10.138ms |
| **Standard Deviation** | 4.97557 |

*Explain how you have obtained these measurements*
I used a tool called Insomnia to query the api with a url and get the output.



GET 127.0.0.1:5000/10/江苏

200 OK 5.3 ms 15 B 6 Minutes Ago

URL PREVIEW

```
http://127.0.0.1:5000/10/%E6%B1%9F%E8%8B%8F
```

```
1 "南京"
```

## 2nd Web Service (20 marks)

*Name of student in charge:*

| **2nd Web service** | *Fill in this table* |
|---|---|
| **Name of service** | Service1 |
| **SOAP-based or RESTful** | Restful |
| **Brief description** | This service takes a city name as parameter and return the city id for next service. |

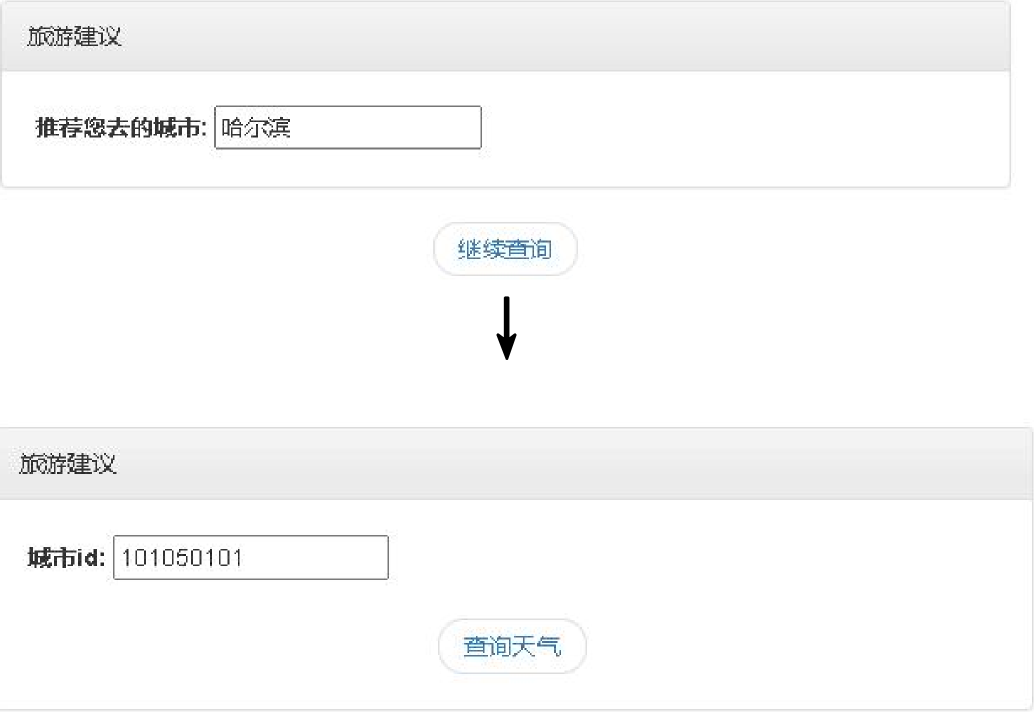
*Server design*
The theory is just like service0, but this time is to use the city name to search the city id in database.

*Server implementation*
The sql for the search is: *SELECT id FROM city where cityZh=…*
The result of the search should be a string type, and then return it in get method.

*Explain how the service is invoked. You may include relevant snippet of source code*
The api is invoked by url as: 127.0.0.1:5001/cityname, and use get to get the return value.
The relevant code is:

```
$("#search2").click(function () {
        var serviceArgs = $("#service2").val();
        var url = "http://localhost:5001/" + serviceArgs
        $.get(url, function (data, status) {
            alert("查询结果: " + "成功");
            if (status == 'success') {
                $("#second").css({ "display": "none" })
                $("#third").css({ "display": "block" })
                $("#service3").val(data)
            }
        });
      });
```

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



推荐您去的城市: 哈尔滨

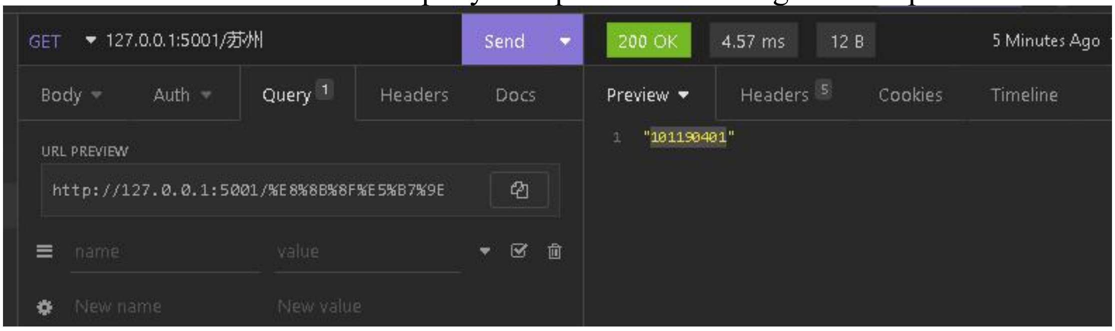
城市id: 101050101

*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** | 6.33ms |
| **2** | 4.47ms |
| **3** | 4.6ms |
| **4** | 16ms |
| **5** | 4.57ms |
| **Average** | 7.194ms |
| **Standard Deviation** | 4.45705 |

*Explain how you have obtained these measurements*

I used a tool called Insomnia to query the api with a url and get the output.



GET 127.0.0.1:5001/苏州

200 OK 4.57 ms 12 B 5 Minutes Ago

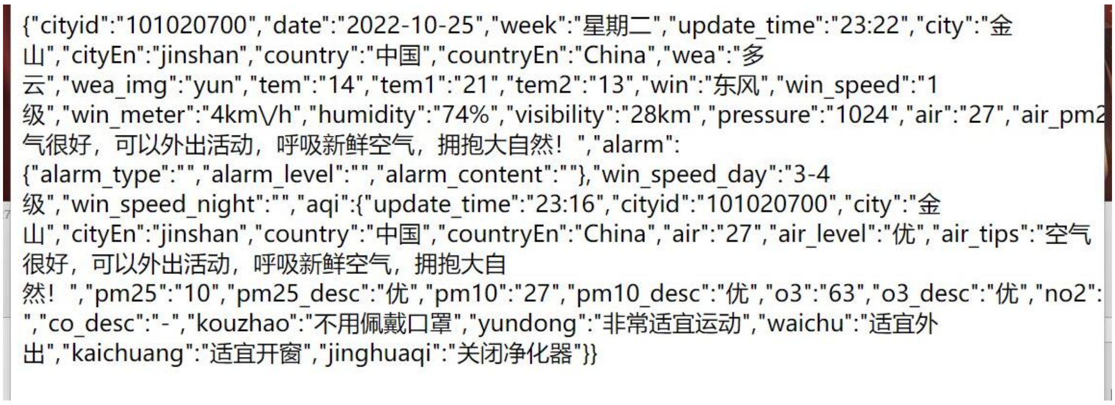
URL PREVIEW
```
http://127.0.0.1:5001/%E8%8B%8F%E5%B7%9E
```

```
"101190401"
```

## 3rd Web Service - External Service (15 marks)

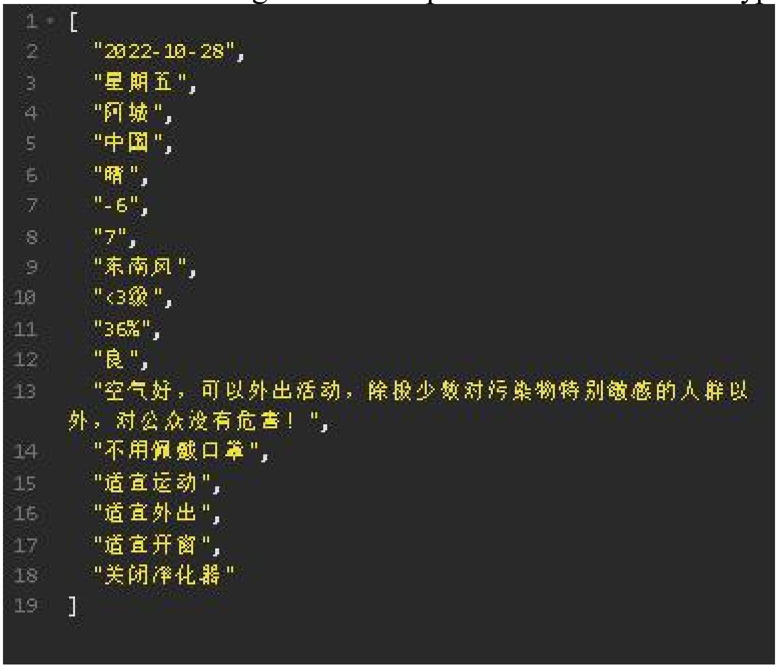
| **3rd Web service** | *Fill in this table* |
|---|---|
| **Name** | Service2 |
| **SOAP-based or RESTful** | Restful |
| **Name of publisher, e.g. Google, Twitter …** | YeKeYun weather |
| **Brief description** | This service invokes an external api with the parameter of city id and get the details of thew city. |
| **URL** | https://v0.yiketianqi.com/api |

This service has two parts: the first part calls the external api to get the weather information of the city, and the returned result is probably like this:



```
{"cityid":"101020700","date":"2022-10-25","week":"星期二","update_time":"23:22","city":"金
山","cityEn":"jinshan","country":"中国","countryEn":"China","wea":"多
云","wea_img":"yun","tem":"14","tem1":"21","tem2":"13","win":"东风","win_speed":"1
级","win_meter":"4km\/h","humidity":"74%","visibility":"28km","pressure":"1024","air":"27","air_pm2
气很好，可以外出活动，呼吸新鲜空气，拥抱大自然！","alarm":
{"alarm_type":"","alarm_level":"","alarm_content":""},"win_speed_day":"3-4
级","win_speed_night":"","aqi":{"update_time":"23:16","cityid":"101020700","city":"金
山","cityEn":"jinshan","country":"中国","countryEn":"China","air":"27","air_level":"优","air_tips":"空气
很好，可以外出活动，呼吸新鲜空气，拥抱大自
然！","pm25":"10","pm25_desc":"优","pm10":"27","pm10_desc":"优","o3":"63","o3_desc":"优","no2":
","co_desc":"-","kouzhao":"不用佩戴口罩","yundong":"非常适宜运动","waichu":"适宜外
出","kaichuang":"适宜开窗","jinghuaqi":"关闭净化器"}}
```

The second part is the processing of the data, using the json load method to dict the mixed data, and extracting key information according to the dictionary, putting the information in a list and returning it. An example of the returned data type:



```
[
  "2022-10-28",
  "星期五",
  "阿城",
  "中国",
  "晴",
  "-6",
  "7",
  "东南风",
  "<3级",
  "36%",
  "良",
  "空气好，可以外出活动，除极少数对污染物特别敏感的人群以
外，对公众没有危害！",
  "不用佩戴口罩",
  "适宜运动",
  "适宜外出",
  "适宜开窗",
  "关闭净化器"
]
```

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

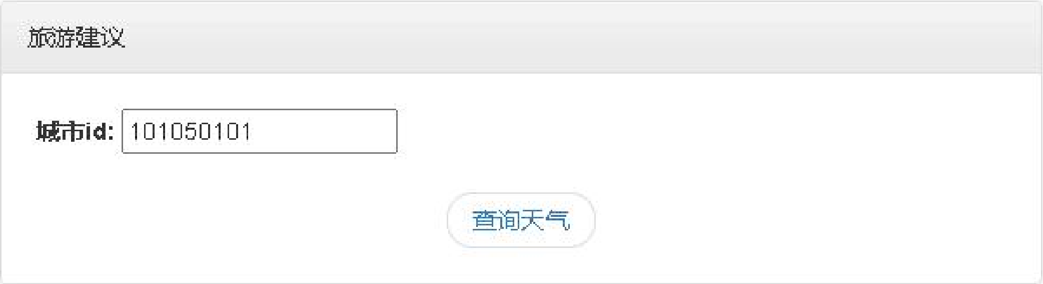
The external API makes requests in the format specified by the third party.  First  get the weather interface, then according to the requirements of the API, define the corresponding Content-Type

```python
def get(self, id):
        # """Get the weather interface"""
        host = 'https://v0.yiketianqi.com/api'
        querys = 'unescape=1&version=v61&appid=68185384%20&appsecret=4i9jWmpe&cityid='
        url = host + '?' + querys + id
        request = urllib.request.Request(url)
        # According to the requirements of the API, define the corresponding Content-Type
        request.add_header('Content-Type', 'application/json; charset=UTF-8')
        response = urllib.request.urlopen(request)
        content = response.read().decode(encoding='UTF-8', errors='ignore')
        city_dict = json.loads(content)
```

The front-end invocation method is as follows:

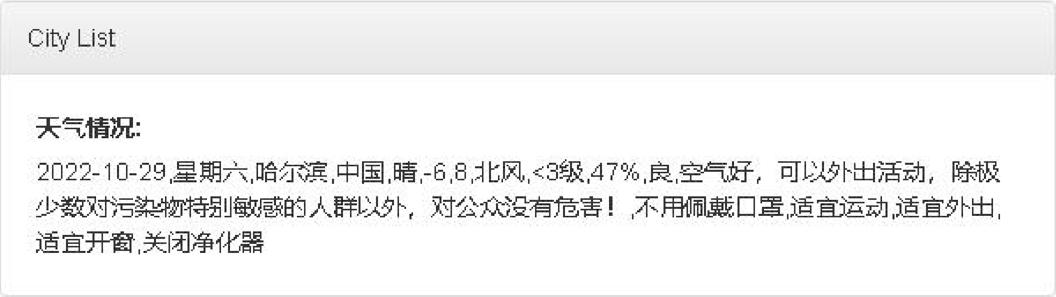
```javascript
$("#search3").click(function () {
                var cityId = $("#service3").val();
                var url = "http://localhost:5002/" + cityId
                $.get(url, function (data, status) {
                    alert("查询结果: " + "成功");
                    if (status == 'success') {
                        $("#third").css({ "display": "none" })
                        $("#fourth").css({ "display": "block" })
                        $("#service4").text(data)
                    }
```

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



旅游建议

城市id: 101050101



City List

天气情况:

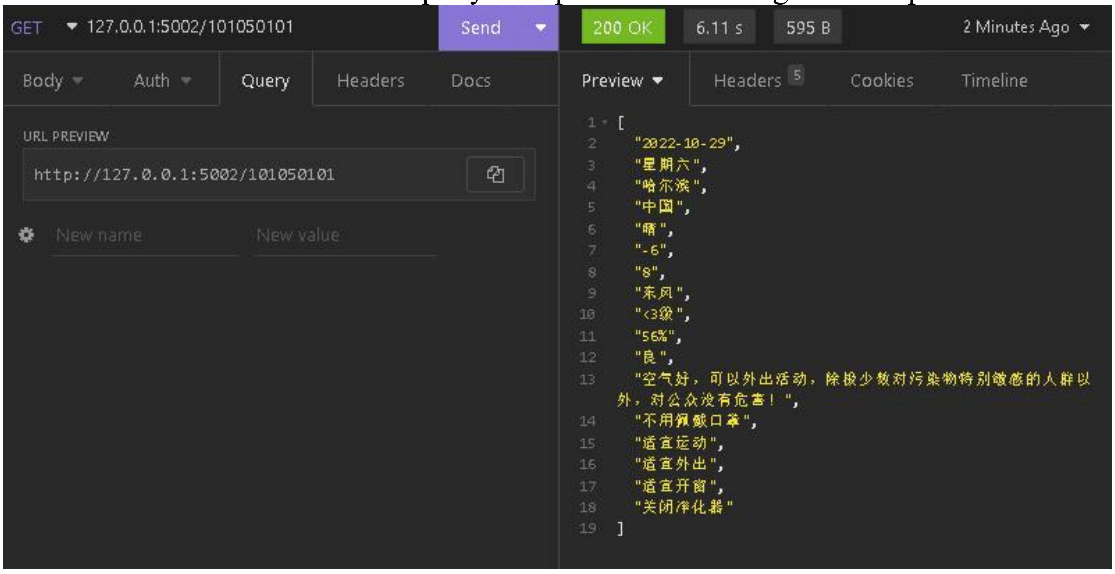
2022-10-29,星期六,哈尔滨,中国,晴,-6,8,北风,<3级,47%,良,空气好，可以外出活动，除极少数对污染物特别敏感的人群以外，对公众没有危害！,不用佩戴口罩,适宜运动,适宜外出,适宜开窗,关闭净化器

*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** | 5.03s |
| **2** | 16s |
| **3** | 12.3s |
| **4** | 5.67s |
| **5** | 6.11s |
| **Average** | 9.022s |
| **Standard Deviation** | 4.36095 |

*Explain how you have obtained these measurements*

I used a tool called Insomnia to query the api with a url and get the output.

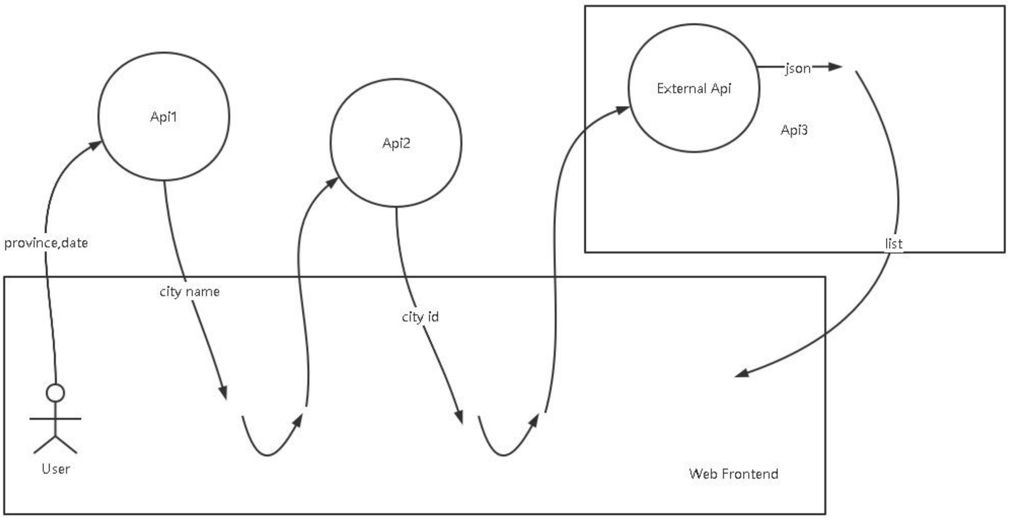


```
http://127.0.0.1:5002/101050101
```

```
[
  "2022-10-29",
  "星期六",
  "哈尔滨",
  "中国",
  "晴",
  "-6",
  "8",
  "东风",
  "<3级",
  "56%",
  "良",
  "空气好，可以外出活动，除极少数对污染物特别敏感的人群以外，对公众没有危害！",
  "不用佩戴口罩",
  "适宜运动",
  "适宜外出",
  "适宜开窗",
  "关闭净化器"
]
```

## Web Services Integration (10 marks)

*Provide details of the Web services integration*



This is the simple frame of the web application.
All services are apis , and the role of the client is to directly interact with the user and call the three APIs in turn .

## Web User Interface (10 marks) – if attempted

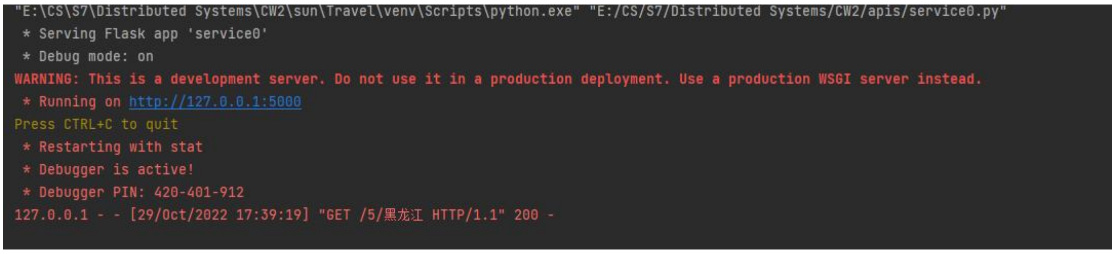
*Provide details of your Web-based application (Servlets/JSP/Other Frameworks)*

## Successful Execution (10 marks)

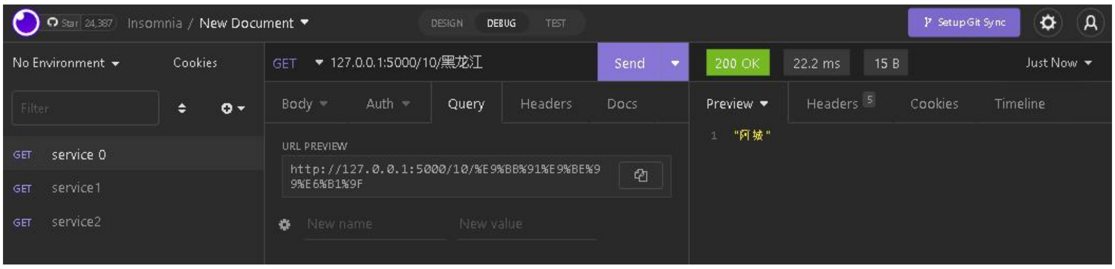
*Include evidence of the Web services integration execution, e.g. screen shot*

**Unit test:**

**Service 0:**



```
"E:\CS\S7\Distributed Systems\CW2\sun\Travel\venv\Scripts\python.exe" "E:/CS/S7/Distributed Systems/CW2/apis/service0.py"
 * Serving Flask app 'service0'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 420-401-912
127.0.0.1 - - [29/Oct/2022 17:39:19] "GET /5/黑龙江 HTTP/1.1" 200 -
```



GET 127.0.0.1:5000/10/黑龙江

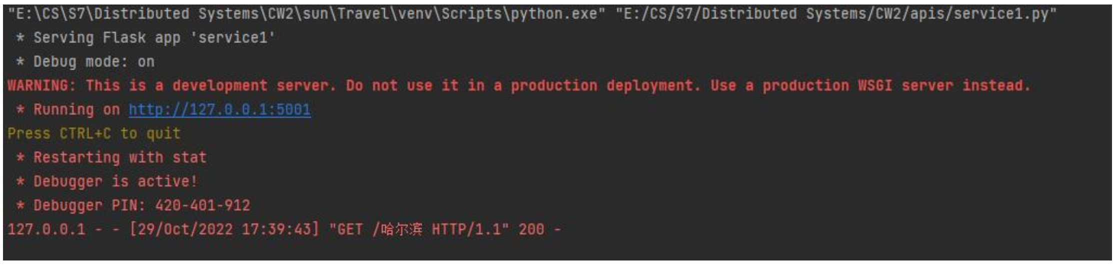
200 OK 22.2 ms 15 B

URL PREVIEW

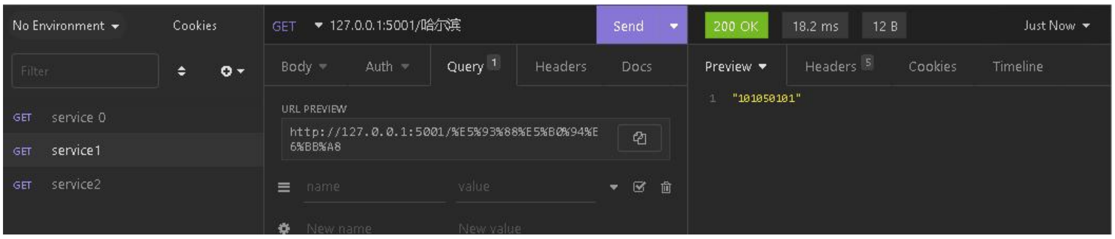
```
http://127.0.0.1:5000/10/%E9%BB%91%E9%BE%99%E6%B1%9F
```

```
1 "阿城"
```

## Service 1:



```
"E:\CS\S7\Distributed Systems\CW2\sun\Travel\venv\Scripts\python.exe" "E:/CS/S7/Distributed Systems/CW2/apis/service1.py"
 * Serving Flask app 'service1'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 420-401-912
127.0.0.1 - - [29/Oct/2022 17:39:43] "GET /哈尔滨 HTTP/1.1" 200 -
```

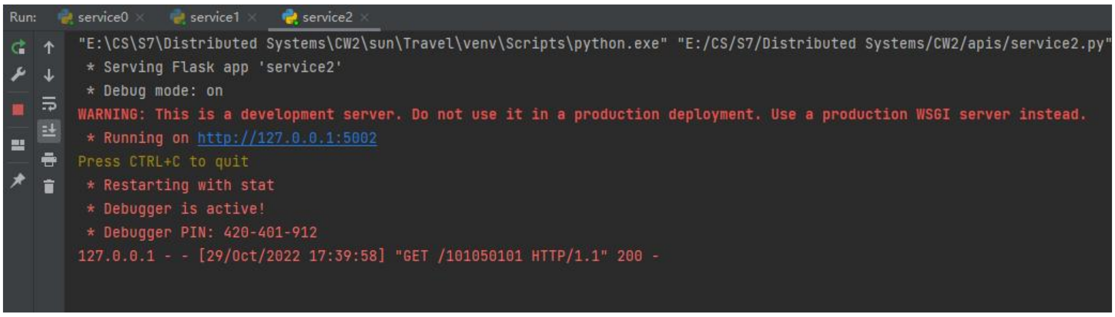


GET 127.0.0.1:5001/哈尔滨

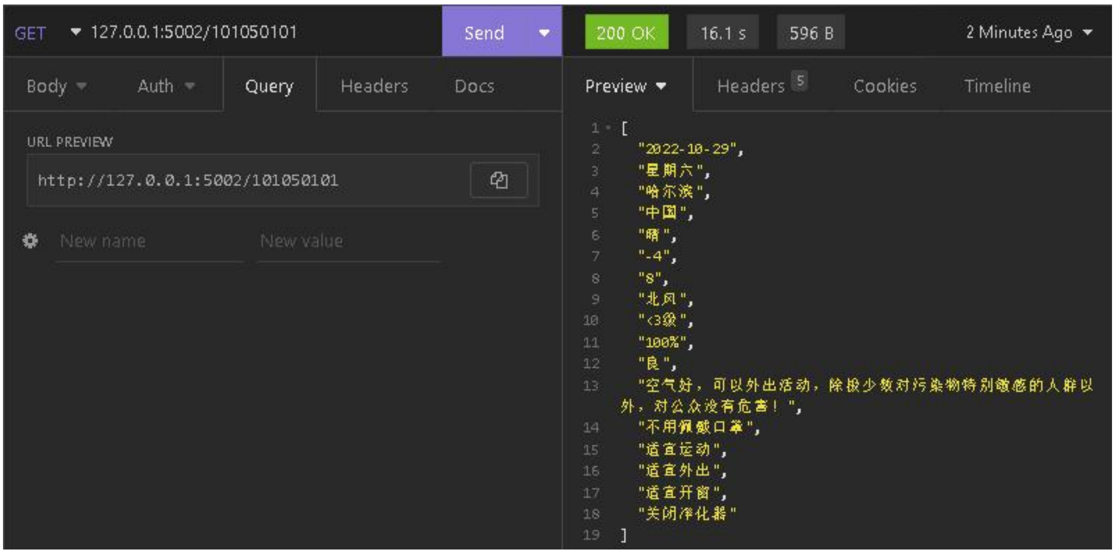
```
http://127.0.0.1:5001/%E5%93%88%E5%B0%94%E6%BB%A8
```

```
"101050101"
```

## Service 2:



```
"E:\CS\S7\Distributed Systems\CW2\sun\Travel\venv\Scripts\python.exe" "E:/CS/S7/Distributed Systems/CW2/apis/service2.py"
 * Serving Flask app 'service2'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5002
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 420-401-912
127.0.0.1 - - [29/Oct/2022 17:39:58] "GET /101050101 HTTP/1.1" 200 -
```

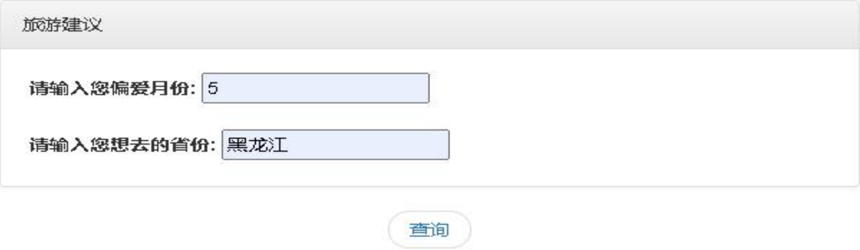


GET 127.0.0.1:5002/101050101

```
http://127.0.0.1:5002/101050101
```

```
[
  "2022-10-29",
  "星期六",
  "哈尔滨",
  "中国",
  "晴",
  "-4",
  "8",
  "北风",
  "<3级",
  "100%",
  "良",
  "空气好，可以外出活动，除极少数对污染物特别敏感的人群以外，对公众没有危害！",
  "不用佩戴口罩",
  "适宜运动",
  "适宜外出",
  "适宜开窗",
  "关闭净化器"
]
```

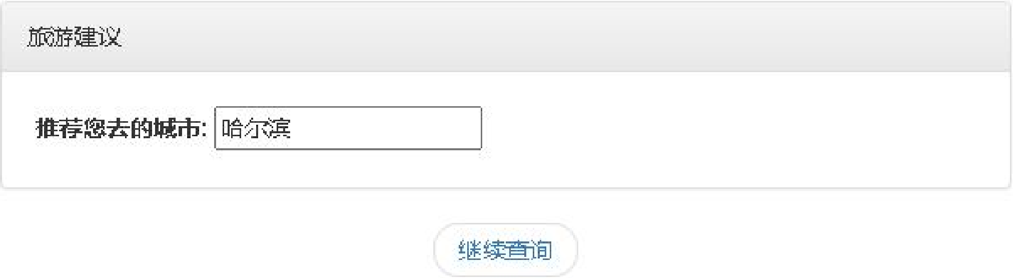
## Holistic test



旅游建议

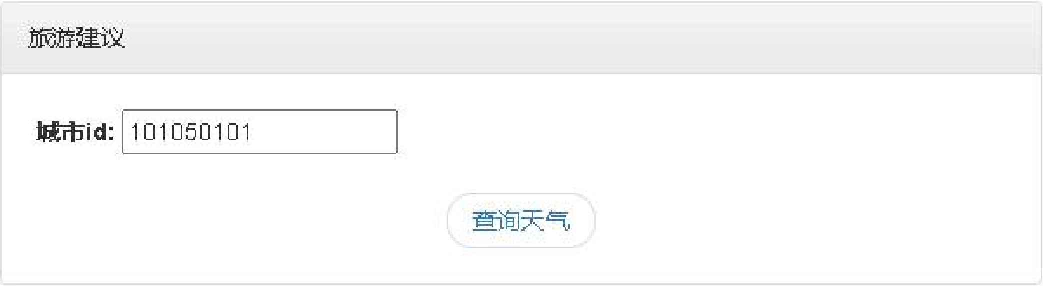
请输入您偏爱月份: 5

请输入您想去的省份: 黑龙江



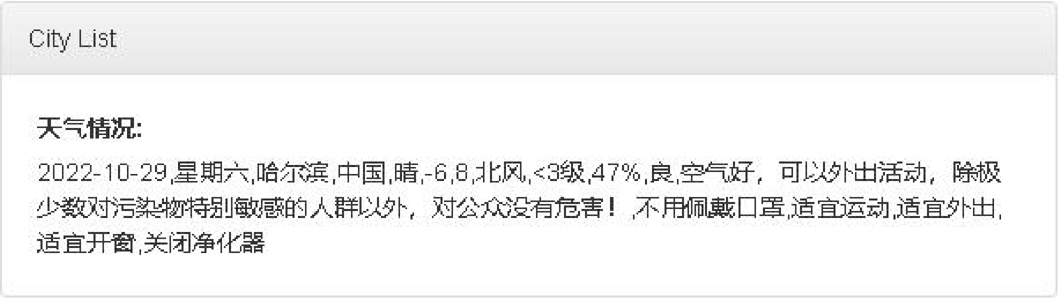
旅游建议

**推荐您去的城市:** 哈尔滨



旅游建议

**城市id:** 101050101



City List

**天气情况:**

2022-10-29,星期六,哈尔滨,中国,晴,-6,8,北风,<3级,47%,良,空气好，可以外出活动，除极少数对污染物特别敏感的人群以外，对公众没有危害！,不用佩戴口罩,适宜运动,适宜外出,适宜开窗,关闭净化器

## Other Comments

This web application is based on flask and flask_restful. All apis are invoked by url.
*Include any details you feel are relevant.*