The report of final task

Introduction

The name of our website is "the Learning of Language Around the World--ZERO TO HERO". This is a learning website that helps people learn various languages of the world from scratch. Students who want to learn multiple languages can get a lot of help on this website. The website so far covers many languages including English, Japanese, Korean, German, French, Russian, Italian, Spanish, Portuguese, Arabic, etc. All languages have established common word libraries to help users remember simple words; each language has a corresponding library of books and movies to help users learn in entertainment. In addition, the website also provides a translation interface. When you don't understand the meaning of words and sentences, you can quickly translate and search; we have also established an independent private library for each user, which can include words and movies; more So many features yet to be discovered...

Data capturing part

Outline

We have ten languages content, in which we have thousands of words, twenty recommend books, twenty recommend movies, some articles. Because the website we build is based on a large amount of language words and some recommendation, the data capturing part is not a easy part. It takes a lot of time and I have to write many python codes files to crawl data of different languages from different websites.

Difficult points

1. How to get the right $xpath$ quickly when analyzing the elements of the website that contains the data we want to crawl?
2. How to use the similar code to crawl different website?
3. How to crawl websites with asynchronous loading?
4. How to process the data crawled from the website to put it into the database?

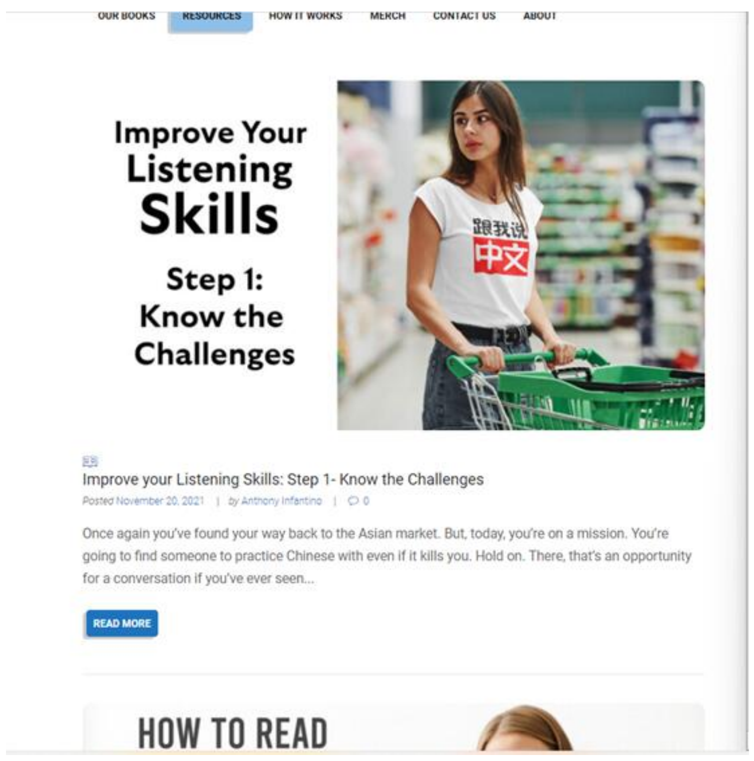
Idea of completing

Search different language tutorial websites on Google, then analyze the elements of the website to know the $xpath$ of data we need.
Write the corresponding python code to crawl the data.
Process the data to eliminate blank space and ‘\n’.
Put the data processed into database.

Solution

Step 1

The first website I crawled is a website contains English essays. It is like this .

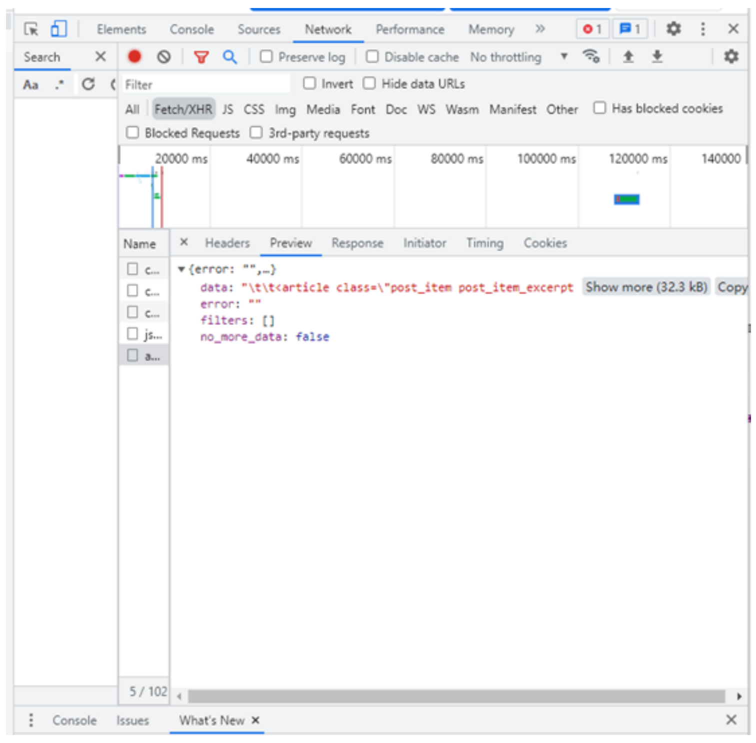


Improve your Listening Skills: Step 1- Know the Challenges

Posted November 20, 2021 | by Anthony Infantino | 0

Once again you've found your way back to the Asian market. But, today, you're on a mission. You're going to find someone to practice Chinese with even if it kills you. Hold on. There, that's an opportunity for a conversation if you've ever seen...

It is a asynchronous loading website, when roll down the mouse, there are other articles cover appear. By analyzing the website, I can get links of these articles easily, but because of the asynchronous loading, the easy action can only get links of the first page.
I know I need to analyze the "$Fetch/XHR$" in Network part, and I indeed find a php file which Request Method is POST, but in the preview of it, there is no useful information about the links in the page. It is this:



```
▾{error: "",…}
  data: "\t\t<article class=\"post_item post_item_excerpt
  error: ""
  filters: []
  no_more_data: false
```

I find a way to solve it by using selenium package. There is a class named webdriver in selenium package, it can actually simulate the real action in your website. I can roll down the website by using the code.

```
browser.execute_script(
      'window.scrollTo(0,document.body.scrollHeight)'
)
```

Use this code some times to get the full pages of the website. Then using xpath can get all links of articles.

Step 2

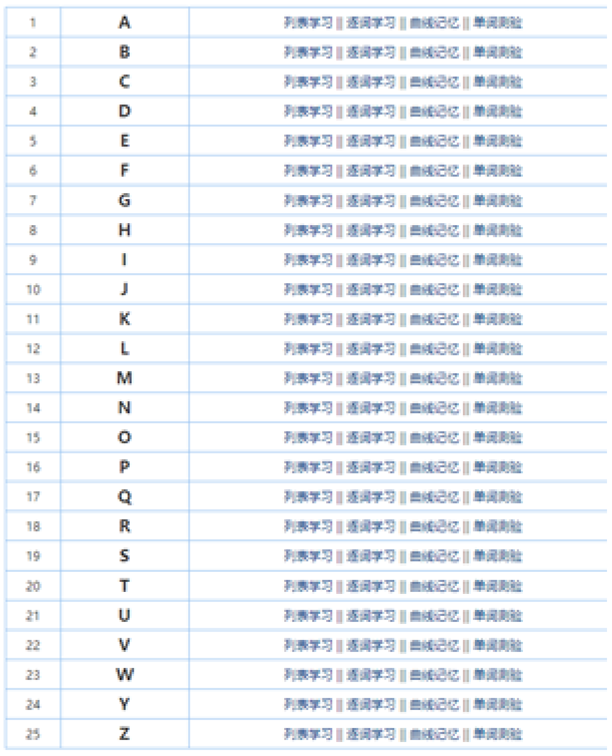
Separate the words and translation in the same string. By searching in Google, I know that the code of Chinese `char >= ‘\u4e00’ and <= ’\u9fff’`. Then by scanning the string linearly and when finding the char in the scope above, split the string into two strings.

Step 3

I meet a problem that there is a '\xa0'in string gotten from the website. I find a surprising solution to remove it in string from $stackoverflow$, the `str.replace(u’\xa0’, ‘ ‘)` code.

Step 4

When scrawling the words classified into different first letters. Like this



1	A	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
2	B	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
3	C	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
4	D	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
5	E	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
6	F	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
7	G	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
8	H	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
9	I	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
10	J	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
11	K	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
12	L	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
13	M	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
14	N	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
15	O	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
16	P	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
17	Q	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
18	R	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
19	S	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
20	T	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
21	U	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
22	V	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
23	W	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
24	Y	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验
25	Z	列表学习 ‖ 逐词学习 ‖ 曲线记忆 ‖ 单词测验

There are different pages in the different letters. For example, A has 25 pages and B has 6 pages. It’ s hard to get the pages of different letters from the website elements. Then I try a brute method that I click the letters and record the pages of different letters in list. Then there is a tricky way to analyze the website. The URL is like http://de.qsbdc.com/deword/word_list.php?letter_id=1&&tag=all&&page_id=1 I can get the content by changing the letter_id and page_id.

Step 5

When getting the similar data from the website, trying to code functionally has an advantage. When I crawl the book recommendation of different languages in $douban$, For example, I get English Book and Japanese Book in $douban$, I write functions `get_url(url)` to get different books $urls$ and `get_pic(url)` to get books’ cover pictures and `get_content(url)` to get author and book name and `get_description(url)` to get book’ s introduction. Coding functionally makes it easy to get different data just by changing the url.

Step 6

It's very important to make all the ten language have the same format. For example, every language have three contents, which are words, recommend books, recommend movies, and articles. And every content have it's tags, for example words have name, pronounce and translation, recommend books have $bookCover$, $bookName$, $bookAuthor$, $bookDescription$. To make my teammate who write the $html$ use more easily, it's important to make all these have a similar name and its' tags have the same order. We meet a problem that some languages' recommend books' cover pictures can't work well, by analyzing the element we find the reason is that the order of that language's recommend books' tags is not in order.

Summary

The data capturing part is not a easy part. I write more than ten python files to crawl all the data. It takes me a lot of time to finished it. At the same time, I think I have command the data crawling and things about database very well.

Login System

Outline

We have register and login two buttons in website. I utilize Database for every user to realize these function.

Difficult points

How to storage user's name and passwords?
How to know if this name has been registered?

Idea of completing

When registering, I create Database for every registrant, which name is the user's name. At the same time, I create passwords sheet to storage the passwords and words sheet and articles sheet in the database.(which are used in the next Collect Part.)
When logining, I get all the database out to confirm if he or she indeed has register. If has registered, I will get the passwords in the password sheet of the Database. Then I compare the input passwords with the stored passwords. If they are same, it will login successfully.

Solution

How to get all the database name?

I find a command in website which can show all the database name, `self.cursor.execute('show databases;')` It is very useful to judge if the user has registered.

When registering how to know if the name has been registered before or the user registers successfully?

I let the function `def register(self,str1,str2):` return 0 when succeeding and return 1 when failing meaning the name has been registered before?

When logining how to know if the name hasn't been registered before or the user password is wrong or login successfully?

I let the function `def login(self,str1,str2):` return 1 when there isn't this user_name, return 2 when the password is wrong ,return 0 when logining successfully.

Collect Part

Outline

This part is based on the login system part. I use words and articles sheets of every user's database(which i write above) to put in or get out words or articles.

Difficult point

Get out data from database and put in data from database.

Idea of completing

When fetching data, for example words, connect the database of user by the users'name, then put the words data into the words sheet of the database.
When geting out data, for example words, connecting the database of the user too by the user's name,then get all the data of words sheet out.

Some reflections

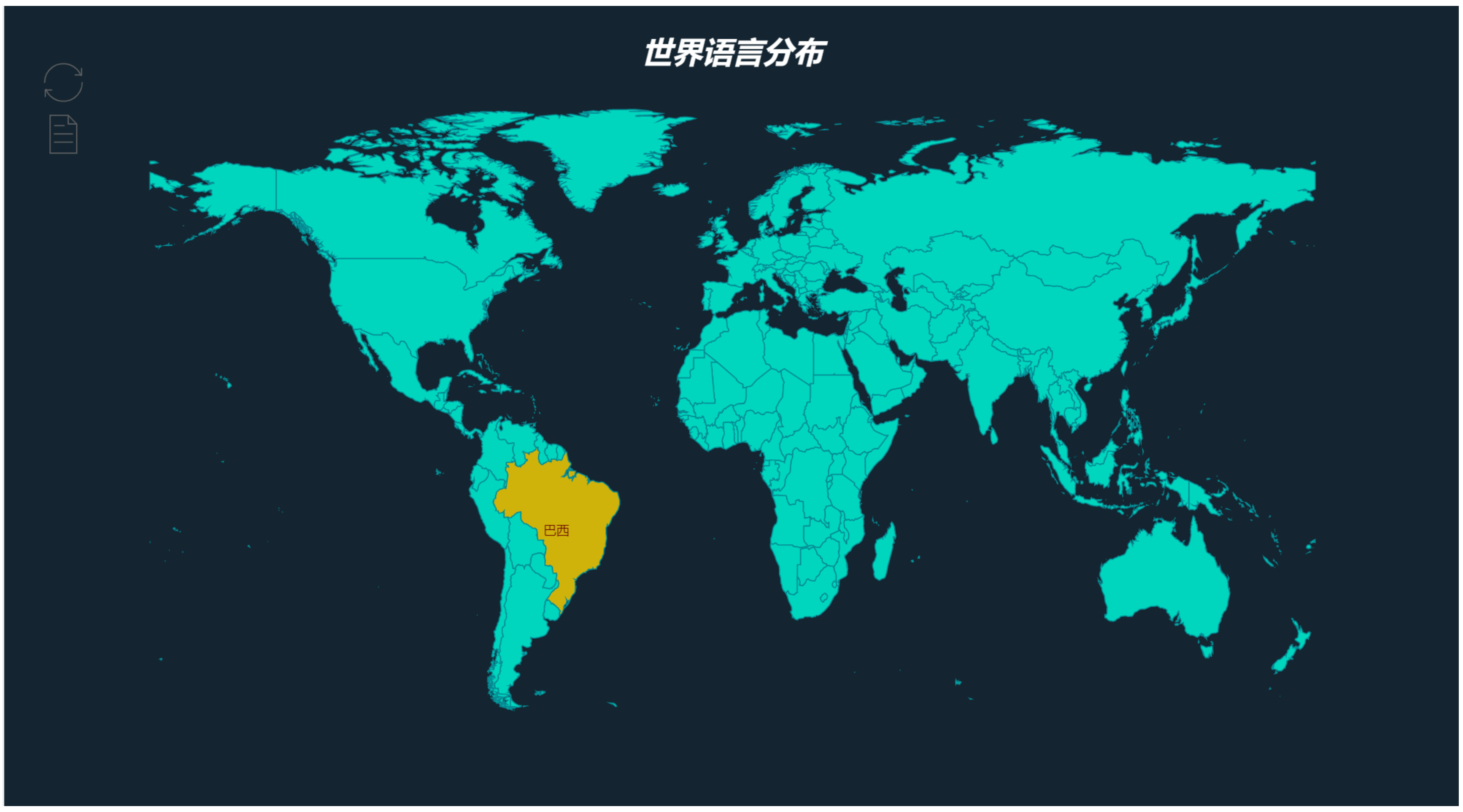
Honestly speaking, the two part is not a very difficult work,but i still spend hours to finish the two part.I think the idea of creating a login system and collection function is wonderful.
I command the database better and when i am coding , i try to write classes to make my teammate use easier. I write the `FetchData` `InsertData` `Login` three classes for getting out data from database and inserting data to database and

the login system.
At all, I still remember the surprising feeling to finish the login system and collection part for zero to perfect.

Visual world map part

Outline

1. Display the language usage of countries around the world in the form of world map
2. Click the country to display the language usage and briefly introduce the language
3. Click the national language link to jump to the independent page of the corresponding language



Difficult points

1. How to import the country's language usage as data into the map
2. For the official world of echarts JS operation
3. For various plug-in operations in the echarts prompt box, insert hyperlinks and language introduction into the prompt box
4. Optimize the overall map, such as drag and drop and restore the initial size

Solution

1. **Use the official map of $echarts\ word\ JS$ library for map rendering**
   Introduce $echarts.js$, $jquery.js$, $world.JS$, $three.JS$ files for map rendering.

```
<div id="main">
    </div>
    <script src="jquery.js"></script>
    <script src="echarts.js"></script>
    <script src="world.js"></script>
    <div class="container">
        <!--为echarts准备一个dom容器-->
        <div id="myEcharts"></div>
    </div>
```

2. **Chinese translation of country names**
   Because the official rendered map of $ecarts$, the names of countries are in English, so it needs to be transformed into Chinese.

```
    // Draw a chart
    let worldChart = echarts.init(document.getElementById('worldMap'));
    // Comparison of English and Chinese country names
    let nameComparison = {
        'Canada': '加拿大',
        'Russia': '俄罗斯',
        'China': '中国',
        'United States': '美国',
        'Singapore Rep.': '新加坡',
        'Dominican Rep.': '多米尼加',
        'Palestine': '巴勒斯坦',
        'Bahamas': '巴哈马',
        'Timor-Leste': '东帝汶',
        'Afghanistan': '阿富汗',
        'Guinea-Bissau': '几内亚比绍',
        "Côted'Ivoire": '科特迪瓦',
        'Siachen Glacier': '锡亚琴冰川',
        "Br. Indian Ocean Ter.": '英属印度洋领土',
        'Angola': '安哥拉',
        'Albania': '阿尔巴尼亚',
        'United Arab Emirates': '阿联酋',

    };
```

3. **Data storage**

    Since a language may be used by multiple countries, a des list is created to store the brief introduction of the ten languages we will introduce. If the language used in a country is too rare, we set it as not introduced for the time being.

    Create a list datamap. The list stores the relevant information dictionary of each country. The name keyword corresponds to the name element of the country, the value keyword corresponds to the list of languages used in the country, the number keyword is used to store the number of languages used in the country, and the last des keyword corresponds to the introduction information of the mainstream language used in the country.

4. Map operation

    Set the overall background color of the map to dark blue.

```
backgroundColor: "#142531"
```

    The title of the display map is "world language distribution", and the color is white, centered and italicized.

```
title: {    //地图显示标题
                show: true,
                text: '世界语言分布',
                top: "30px",
                left: 'center',
                textStyle: {
                    color: '#fff',
                    fontSize: 28,
                    fontStyle: 'italic'


                }
            },
```

    In order to better serve users, a $toolbar$option is set. Because the map can be zoomed in and out with the mouse pulley and dragged with the left button, in order to facilitate the restoration of the original size, the $toolbar$is set to realize the restoration function and the function of a data view.

    Introduce the toolbar, set the item size to 35, and set the relative position top: 50, left: 30
    Two basic functions: restore restore map size; The $DataView$ data view makes it easy to view language data.

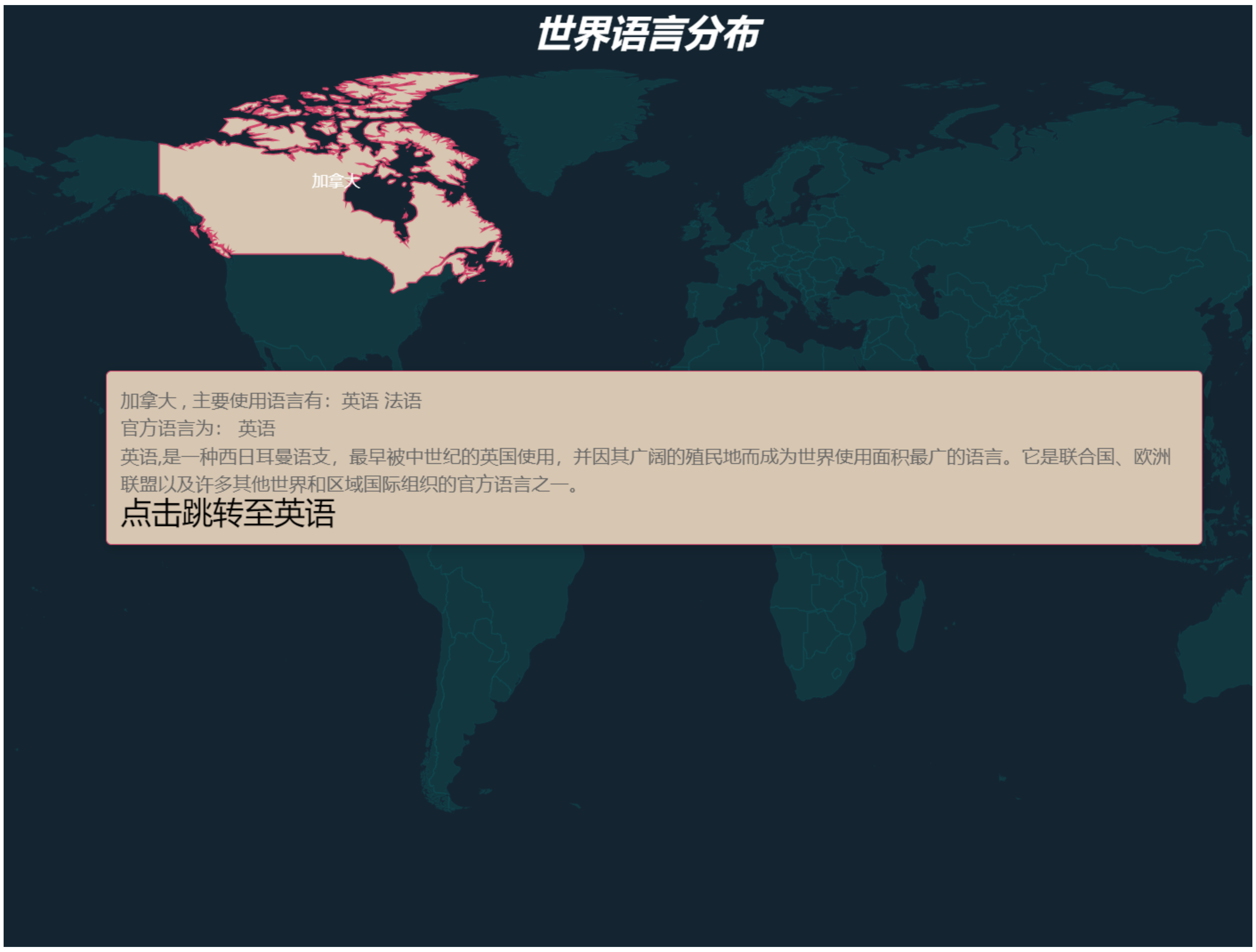
```
toolbox: {  //工具栏
```

```
                show: true,
                itemSize: 35,
                orient: 'vertical',
                top: 50,
                left: 30,

                feature: {
                    restore: {
                        show: true,
                        title: "恢复地图最初大小",
                        textStyle: { color: '#fff' }
                    },
                    dataView: {},
                },
            },
```

Set ecarts to map visualization and maptype to world map mode
Bind data to the map and set roam so that the map can be dragged.
Data binding data of countries in datamap.
$Namemap$ binds the Chinese names of countries in $namecomparis$. Because there are too many countries in this map, the display of labels is disabled. When the mouse moves, the corresponding labels will be displayed.

Use the prompt box component to introduce the language of the displayed country and realize the jump function



加拿大，主要使用语言有：英语 法语
官方语言为： 英语
英语,是一种西日耳曼语支，最早被中世纪的英国使用，并因其广阔的殖民地而成为世界使用面积最广的语言。它是联合国、欧洲联盟以及许多其他世界和区域国际组织的官方语言之一。
点击跳转至英语

`Configure: true` limit the prompt box to the scope of the chart
`Trigger on: 'click'` set the trigger effect of the prompt box to mouse click
`Enterable: true` set the mouse to move into the prompt box for viewing

Set the variable n to store the number of languages in the country, and the string w to store the text information of the prompt box. Write the language used in the country, the official language and the introduction of the official language through circular operation. Finally, add a hyperlink to jump to the interface in the official language.

```
formatter: function (data) {
```

```
                    var n = data.data.number
                    var i = 0
                    var w = '<a>' + data.data.name + ' ' + ',' + ' ' + '主要使用语言有: '
                    var flag=data.data.des
                    while (i < n) {
                        w += data.data.value[i] + ' '
                        i += 1
                    }
                    w += '</a>' + "<br>" + '<a>' + "官方语言为:  " + data.data.value[0] + '</a>'+"<br>"
                    w += data.data.des+"<br>"
                    w +='</a>'
                    if(flag!=des[10])
                        {w+='<a href="#" style="font-size:23px;color:black;text-decoration: none;">'+"点击
跳转至"+data.data.value[0]+'</a>'   }
                    return w
                }
```

Website Design Part

Outline

Our website can be divided into Four parts . The introduction part , the language part , the map part and the search part . Each part has its own page , and can be skipped to through home page . Their details are as follows .

Introduction Part

This part actually has two pages . Page 1 displays our website's intention and the function .To polish it , we add the animation and the background-music in movie Star War . Page 2 displays the group members' name and division of work .

Language Part

Up to now , we have ten different languages , each has the same layout . The language part can be subdivided into tree parts , word&vocabulary part , movie&article part , translation part.

Word&Vocabulary Part

This part's function is to offer all the words of a language so that the user can learn them , also , the user should be able to add the words they don't know to the vocabulary book . So first , we get all the words of a language ,and offer two areas for the user to check. First area is a word card , the user can get the words's information one by one.

The vocabulary part is simple , we connect to the user's database , then display his/her name and all the words in the database .

Movie&Article Part

The word vocabulary's implement method is as follows . First , we have a login interface , the user only needs to enter his user_name , then we get this name and connect to his own database . Then we attach a form with different values to each word , when the user click the add button on the word , it will be submitted . We will add this word to the user's database .

Translation Part

Using the $Baidu\ API$. By sending a "post" request to a url, we can get the result of the translation.

Search part

Outline

From the beginning, I decided to use ES as the final search engine. However, there were some kinds of problems. My computer crashed once and I had to reinstall the system. So I lost the original code which was kind of a disaster for me. Fortunately, my companion had stored the data. Thus I came to realize that although ES was convenient but it was always a problem to open two documents.

So I search the Internet to find some useful and handy ways to search the data table directly.

The search code is as followed:

```
@app.route('/')
def search():
    conn = pymysql.connect(user='root', host='localhost', port=3306,passwd='11111111', db='web')
    cur = conn.cursor()
```

function: connect to the database.

```
S=request.args.get('name','')

    sql = "select * from database where name like '%" + S + "%'"
    cur.execute(sql)
    datas = cur.fetchall()
```

function: add a method, get the name user input and store it using a variable ‘S’. Execute the command to search accurately if there are some articles includes exactly the word S and fetch the data and store it in data which is used for transfer data.

```
    if(S==''):
        items="输入你想搜索的文章关键词"
    else:
        if(datas==()):
        items="没有搜索到"
        else:
            items=datas[0][1]
    return render_template('search.html', items=items)
```

If the data can’t be found in our database, result is that we can not find it. However, if we manage to find the data, the data will be fetched in the form of tuple but we just need the title of our article so store it in the variable ‘items’. And of course we need to return to our $search.html$ and transfer the items.

Besides that, we need a front end to be shown to our users. Thus we have the $search.html$ as followed:

```
<p class="description">description:</p>
<div class="body">

  <div class="wrapper">
    <form name='searchcontent' >
    <input class="search" type="text" id="search" name="name"/>
    <button class="submit"  value="Submit"  ></button>
    </form>
  </div>
</div>
{% if items %}
<div id="box1" class="box blurred-bg tinted">
    <p class="content">{{items}}</p>
  </div>
{% endif %}
```

The items being transferred collects the information of the description of each language. And we use an if sentence to change our $html$ like the first homework. Of course we need to using some more beautiful class to beautify the searching page so we include some interesting js document.

And after that, the searching page is accomplished.