**REPORT**

Group 3

group leader:杨宇博、陈泓屹

crawler and database:尹晟、吴淏悦

text information retrieval and search engine:孙瑞冬、张人豪

website development:陈硕、杨冠柔

visualization:浦金、孙涵

**The theme of our website system: Warship**

CONTENTS

Crawler and Database……………………………………………………………3

Crawler……………………………………………………………………………………………………………3

Database…………………………………………………………………………………………………………9

Summary………………………………………………………………………………………………………11

Text Information Retrieval and Search Engine……………………12

Introduction…………………………………………………………………………………………………12

Results Display………………………………………………………………………………………………16

Adding data…………………………………………………………………………………………………18

Problems and Reflection………………………………………………………………………………19

Website Development…………………………………………………………21

Homepage……………………………………………………………………………………………………21

Country webpage…………………………………………………………………………………………22

The structure and detail page………………………………………………………………………24

Using php to Connect Elasticsearch and Do Retrieval………………………………30

Visualization………………………………………………………………………35

GRAPH 1………………………………………………………………………………………………………35

GRAPH 2………………………………………………………………………………………………………37

GRAPH 3、4…………………………………………………………………………………………………39

**Crawler and Database**

**1** **Crawler**

①. Process

**We divided the whole process of crawler into 3 parts--** **preparation for initialization, getting every warship’s website and some parts of its information and getting the rest information of each warship through the website.**

**A)** **preparation for initialization**

******

In this step, it is mainly a preparation for the crawler. We set up an array to store information about the ship and two lists based on the categories we need to use in the third step. we noticed that the url we crawled didn't have the preceding part, so we created a function that mimicked what the teacher did so that the url is complete and can be accessed.

**B) getting every warship’s website and some parts of its information**

First, through observation, we found out that we could get a certain warboat’s url, name, category and active time which represent in the initial website.

According to the teacher's explanation, we respectively found the Xpath address of crawling URL and the remaining three information in the original webpage source code and complete my relative codes.

* original webpage source*

**

codes of crawling url



But here because the two properties of classification and active time are together, they are seperated by ‘ | ‘, so we used a function to separate the two properties.

Code of crawling other information

**C) getting the rest information of each warship through the website**

This step can be roughly divided into two parts—crawling the information in the left and then in the right.



**The process on the left** is similar to the previous one. The idea is to find the xpath address for each piece of information and then crawl it. Since some information of the warship was not available when the database was imported, we made a judgment when storing it.

**The number of properties on the right is obviously much higher, so we thought it would be easier to find the pattern of their Xpath addresses and crawl them.**

However, the attributes on the right can also be divided into four different categories. *The first category is the weapon information of the warship*, which is different from other attributes and needs to be crawled separately.

*The second and third types of information can be crawled through two different Xpath addresses,* respectively, as represented by *tmp1* and *tmp2* above. But since *some of the information that the third type crawls contains are properties that we ultimately don't want—the fouth type*, we created two lists in the beginning to sort the properties so that we could filter through them. we finally crawled down the required properties by creating a list of l1,l2, and l3 and using functions.

At this point we have all the information we need to crawl.

② **difficulties and how WE solved them**

**We think there are roughly 2 difficulties in the process of crawling.**

**The first one** **is how to separate the properties of classification and active time as they are not individual parts.**

**Solution:**

Use the spilt function in Python to separate the two ( the detailed codes are above)

**The second one is how to find out the rules of different attributes and sort them for crawling in the third step.**

**Solution:**



In fact, this process is difficult and complex. The main method we used is to find out every property’s xpath address, their order and whether it was needed by our team and then we created two lists to divide the rest information into 3 categories and finally made use of functions to crawl their information.(The concrete codes are above.)

**2.** **Database**

**1) Process**

**In this part, WE think there are two main steps, one is to create the database and the other is to store the data we got earlier.**

**A create the database**

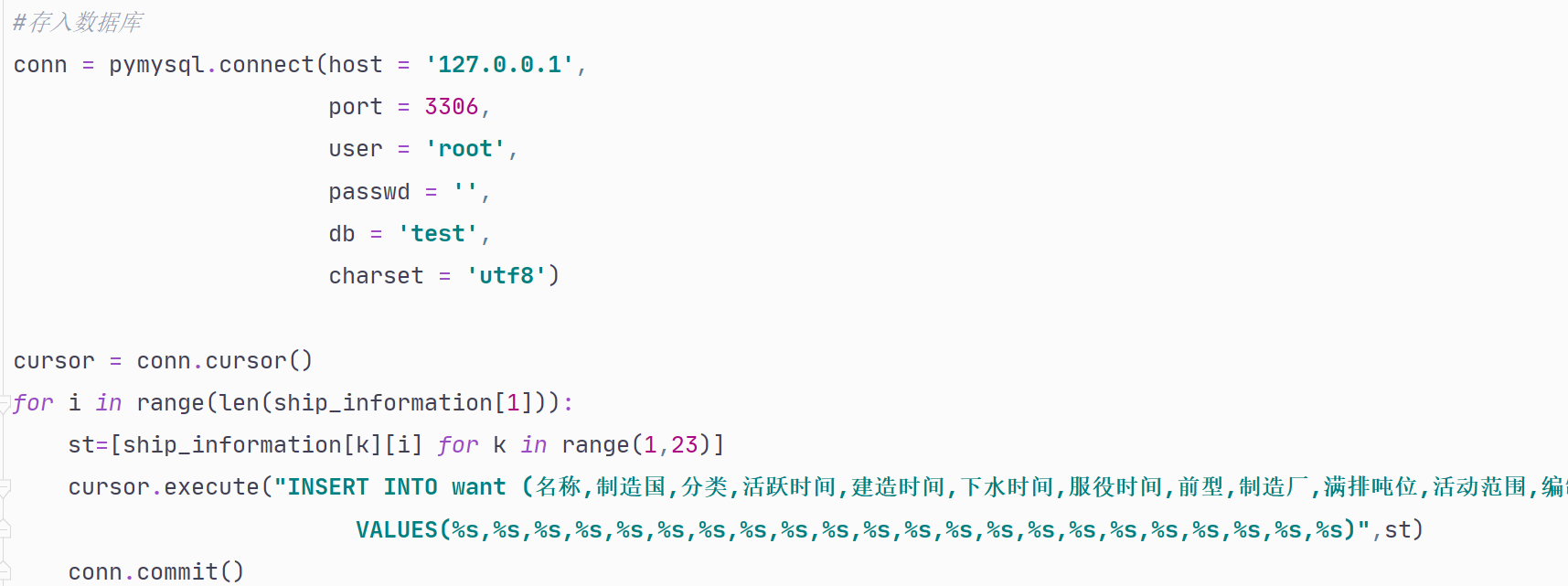
This process was relatively simple. We created a database named “want” in my own mysql by imitating the teacher's explanation in class, and added the name of each column in the database according to the requirements of our group.



**B store the data we got earlier**

The general idea for this step is to mimic the teacher's stored procedure.

But given the amount of information that needs to be stored, I'm using a *for* loop here.



**2) difficulty and how to sovle it**

In my first few attempts to save failed, We asked the teacher and printed out the list of stored information. Then we found out that *in the process of data crawling, due to different data formats, the information of all ships in some properties is put in a list, but in other properties, the information of a ship is a list, which leads to the failure of my for loop.*

So we changed the crawler code for those properties where a piece of information of a warship is in a separate list by adding [0] which means that the information string was stored not the list. After that, we succeeded.



**3. summary**

The above is the whole process that we crawled the data and stored it in the database. We think we learned how to find the key to the problem through observation and how to find the rules of things in this big homework. In a word, we benefited a lot from this activity!

**Text Information Retrieval and Search Engine**

**Introduction**

Our target is to build a website related to warships, and the data we get is from the same website. We need to build a tokenizer to realize text retrieval and make the results as accurate as possible.

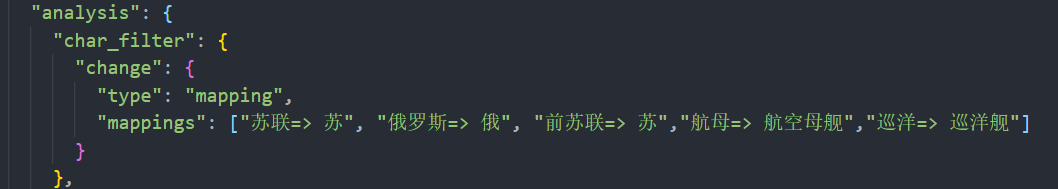
**Requirements for this assignment**

Determine the type of each keyword and build the index & mapping.

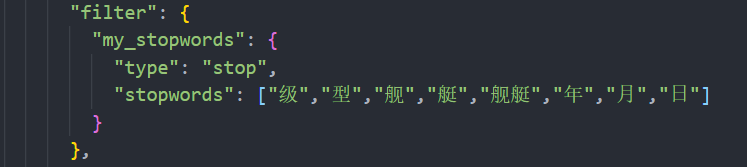
**Tokenizer**

First, since our data are Chinese, so we use standard, Chinese ik\_max\_word and ik\_smart basically, and which one to use is to be decided.

Second, we consider some useful but not included in the tokenizer words and build the char\_filter, and the details are as follows.

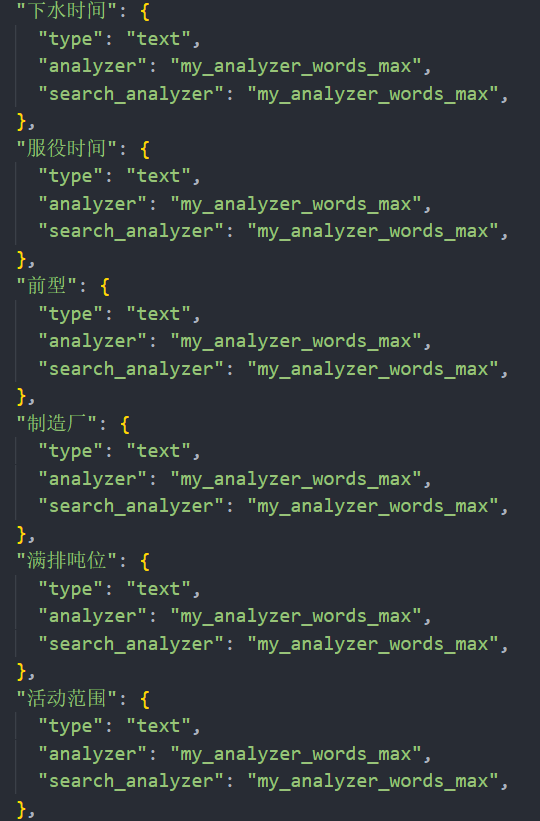
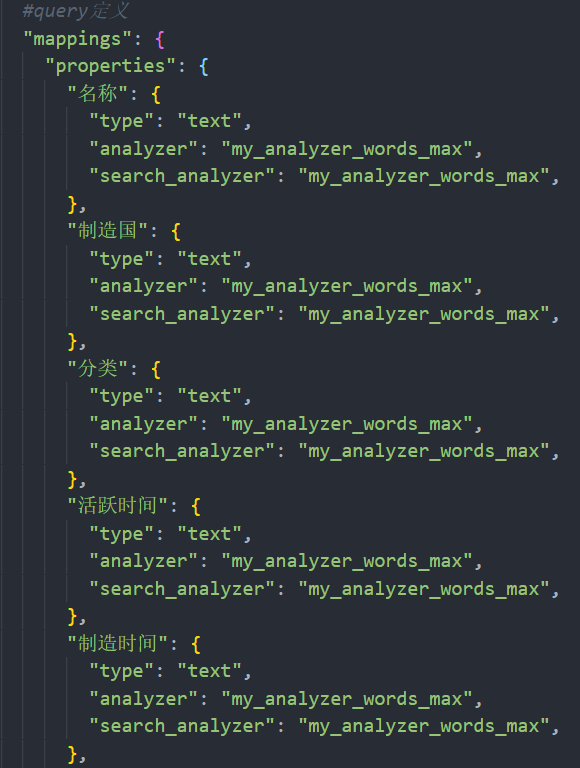


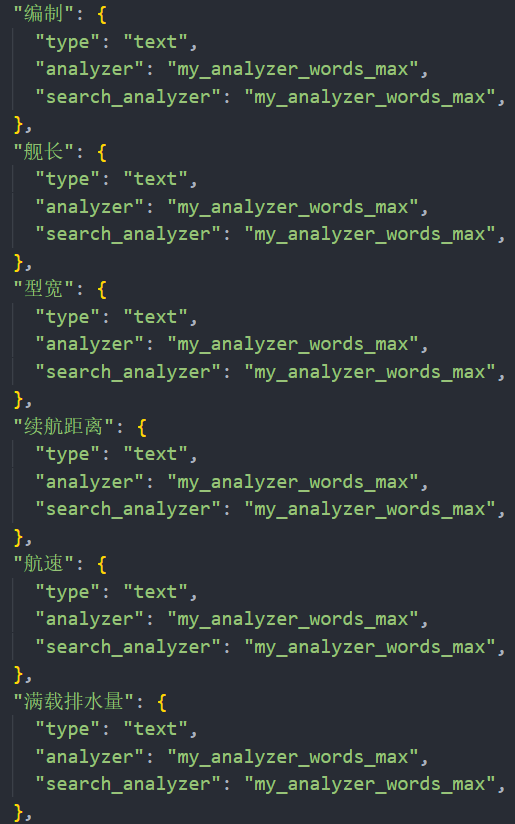
Third, since our thesis is warship, so some characters and words are meaningless, and we set them as stopwords to increase accuracy. The details are as follows.

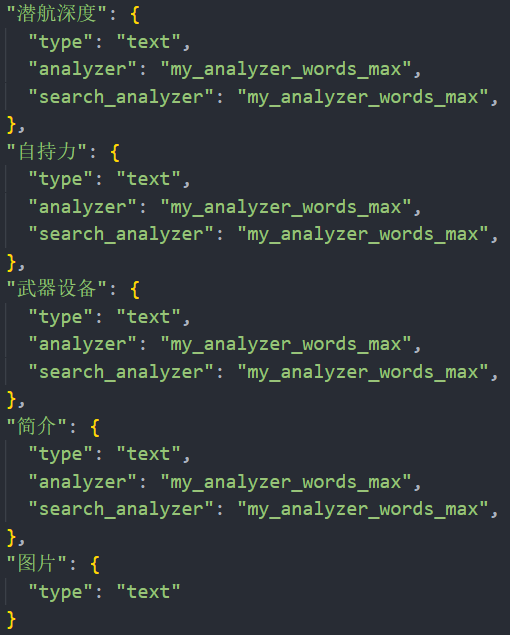


Fourth, we finish the properties. Our data is mainly Chinese words, even birth date and some other time also have “年”“月”“日”, so we all consider them as “text”. After detailed test, we decide to apply the combination of our own tokenizer and ik\_max\_word as it works better than ik\_smart in our test in this field.

Our tokenizer and mapping are as follows:





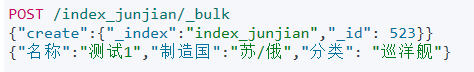


**Results Display**

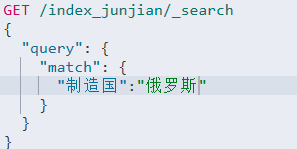
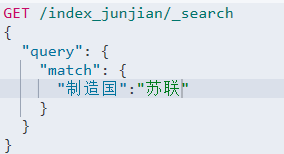
**Test for “char\_filter”**

Because we thought that there might be users searching for the former Soviet union, we designed the word splitter above, and now we will try it out by using Kibana.

When we search for “前苏联”,“苏联”or“俄罗斯”, we can successfully search for relevant content.



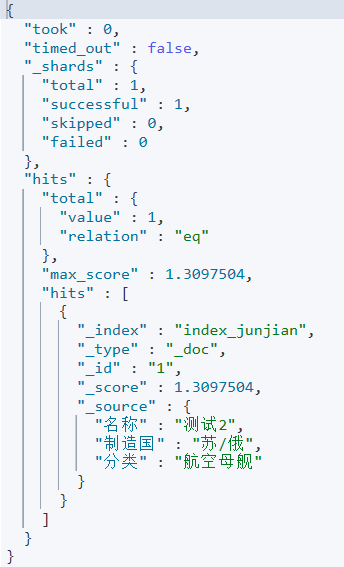
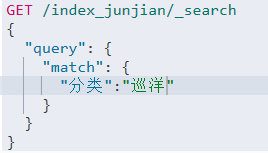
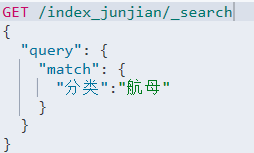
(this is the text for the test)



And the results are as follows.



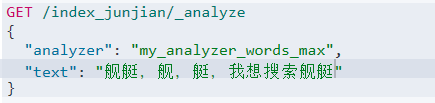
What’s more, we’re used to calling “航空母舰” “航母”, so we ran a test for it, and it turned out to be correct. And we also noticed that the word “巡洋” is separated in the IK, but the word “巡洋舰” is combined, then we solved this problem and we also ran a test for this part.



**Test for “filter”**

To avoid unnecessary searches, we use several words as stopwords, such as “舰”,“舰艇”

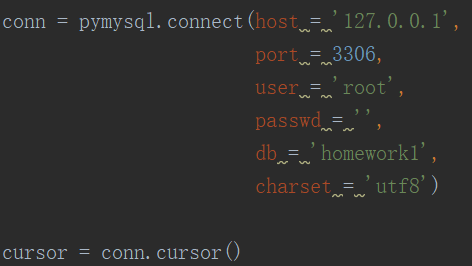
“艇” and so on. And we also ran a test for it by using Kibana. It turned out pretty good.



The results above roughly illustrate shows the uniqueness of our word splitter. Other searches can be done by depending on the IK word splitter, so we thought that we didn’t need to show what the IK word splitter can do.

**Adding data**

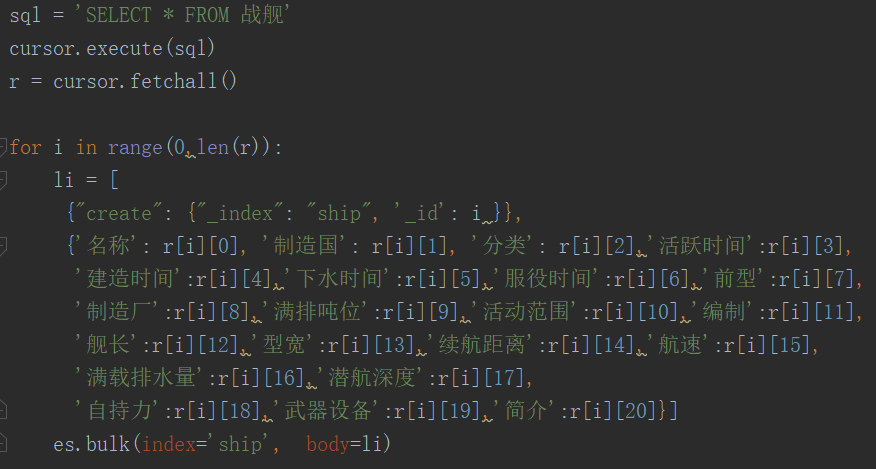
After that, we need to build our index using the data in our database. Firstly, we import pymysql to connect to our database and create a cursor.



And we create a final index.

1592031666(1)

Then, we use the cursor to fetch all data in our database and use circulate to add documents into our index named ‘ship’.



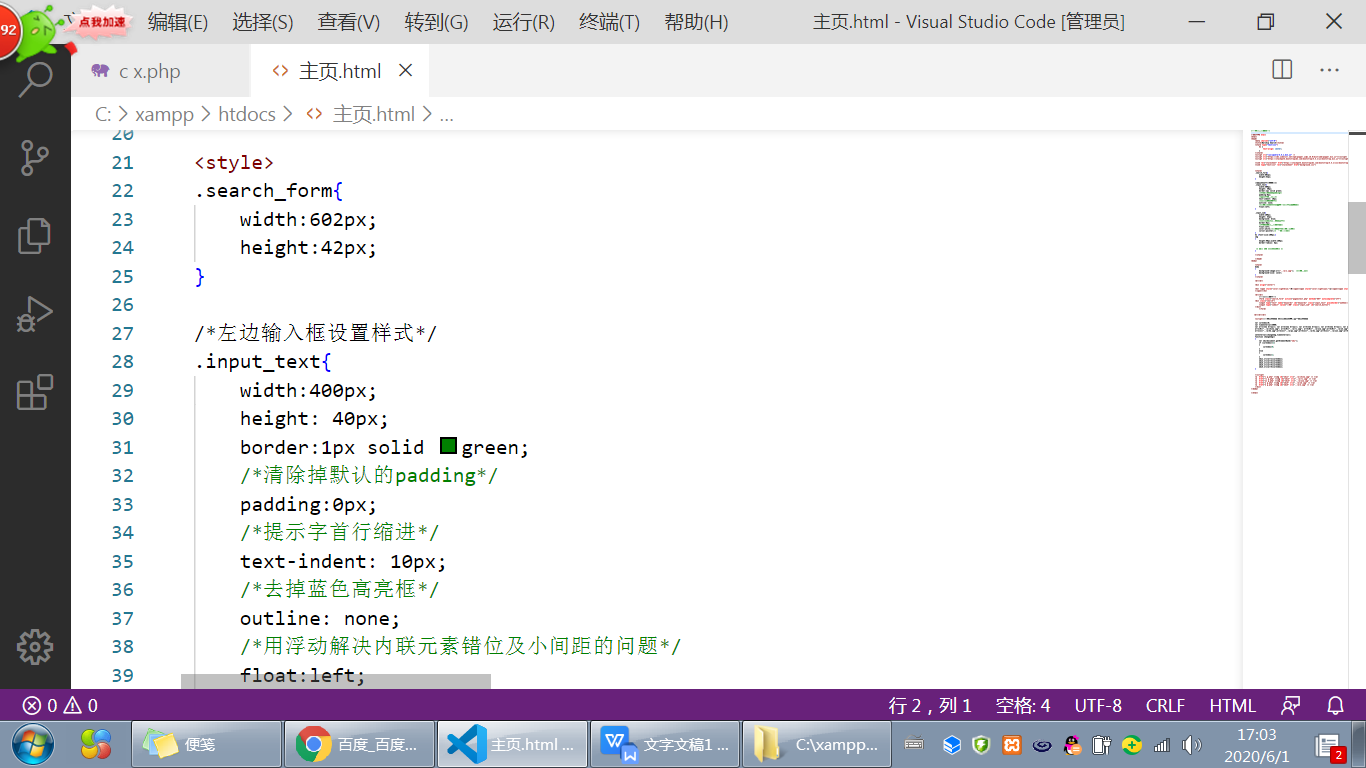
**Problems and Reflection**

1. In our data, there are mixed languages, so we apply both standard and Chinese ik tokenizer, but there is also little French, Spanish, Russian and Japanese, with limited tokenizers, we can’t achieve the same effect. We tried some advice online, but considering we can’t read these languages, so we can’t test or promise the accuracy. And our target is Chinese, so we finally give up the idea.
2. We originally wanted to achieve full text retrieval to optimize the experience, but the accuracy is unavoidably low and not that ideal, so we also abandon the idea.
3. Later we want to achieve picture retrieval, this may be a little difficult as it is related to machine learning or some other fields, so we intend to try it in this summer maybe.

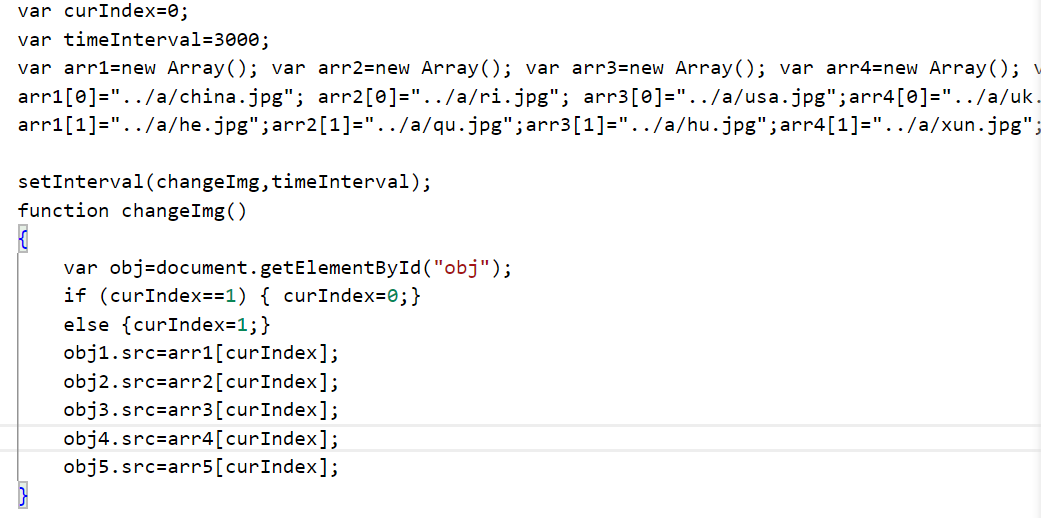
**Website Development**

**Homepage**

The homepage contains a title,a search box,and a group of pictures.The search box’s style imitates the style of Baidu.Partial codes are as follows. And we use “get” method to transfer parameters.



The group of pictures will switch from five national flags to five classical warships in every three seconds.We realize this function by using “Window setInterval()”.We create several arrays to store the address of images.In every 3 seconds,we change the src of the pictures which are being displayed.Partial codes are as follows.



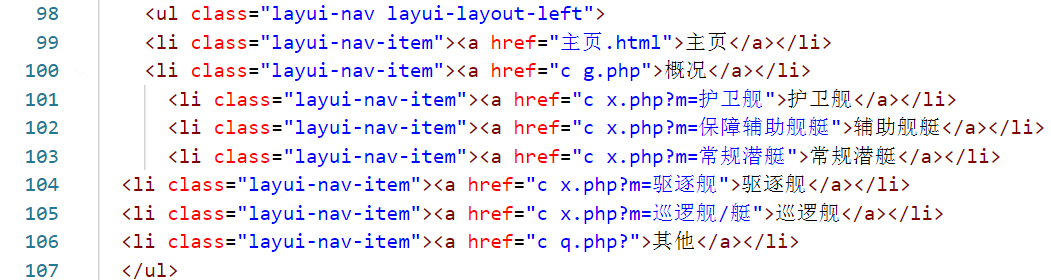
Each picture is a link to the corresponding country webpage,which will show the information of the warships of the country in detail.

**Country webpage**

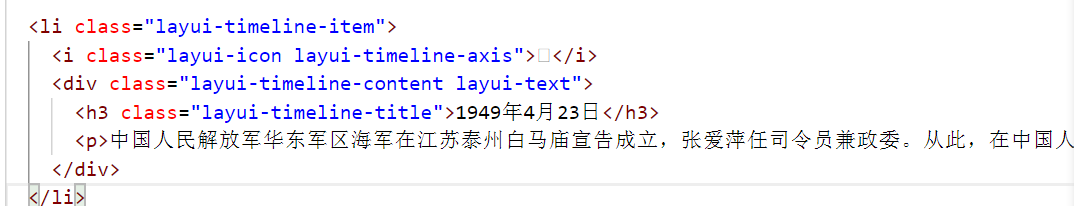
Each country webpage contains the outline page and several pages showing the detailed information of a specific type of warship.

1. **Outline webpage**

At the top of the outline page is a navigation bar.Each item is linked to a corresponding page.We use Layui front frame to create a nice-looking navigation bar.Partial codes are as follows.



In the centre of the page is a brief introduction of the navy and its historical development.We use the timeline module of the Layui front frame to display the history timeline clearly.Partial codes are as follows.

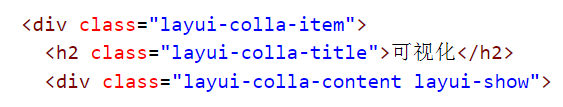


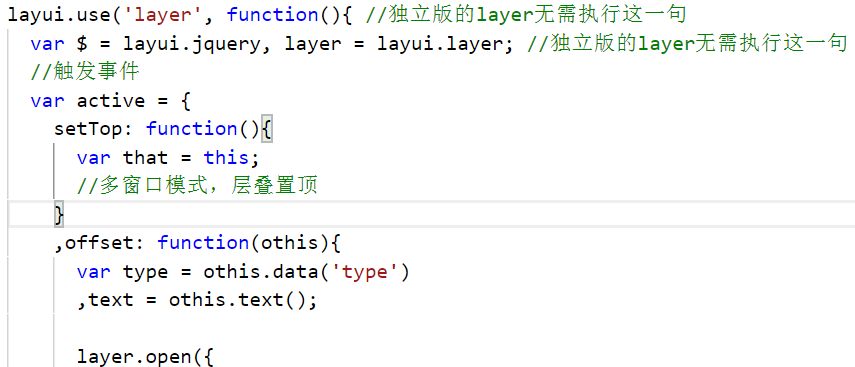
At the bottom of the page is a rose diagram which shows the number of each type of warship.

1. **Type webpage**

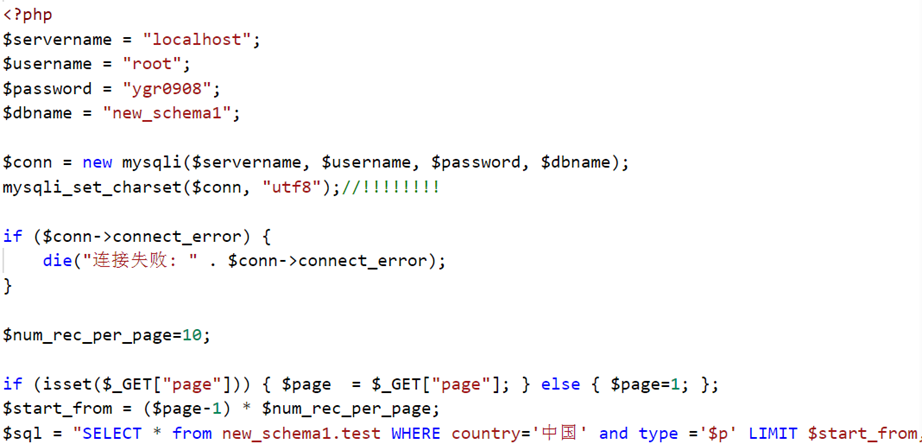
At the top of the type page is a navigation bar.At the left of the page is a Accordion which contains two parts.The first part is a brief introduction of the warship.The second part contains a mip-lightbox,which contains four histograms that show the change of properties of the warship over time.

Both the Accordion and the mip-lightbox are created by using the module in Layui front frame.Partial codes are as follows.





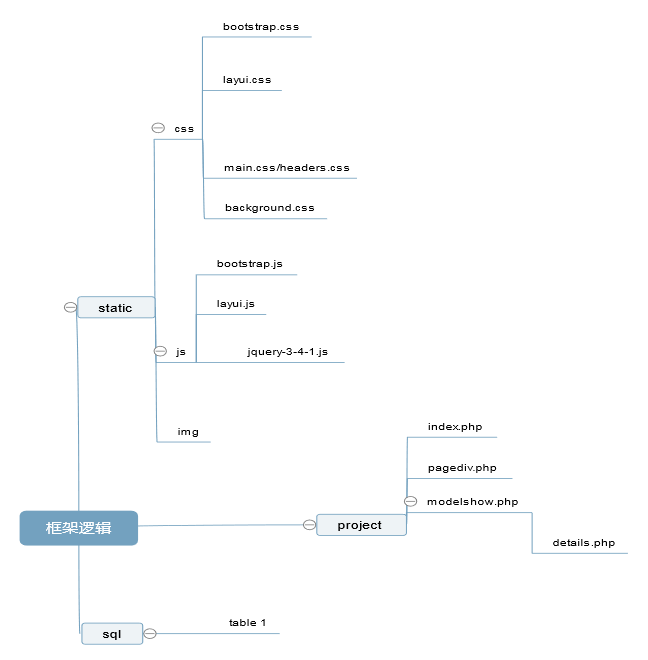
At the right of the page is a list of warships containing images,names and manufacture time.We get these information by connecting Mysql.Partial codes are as follows.The warship’s name is linked to the detail page.



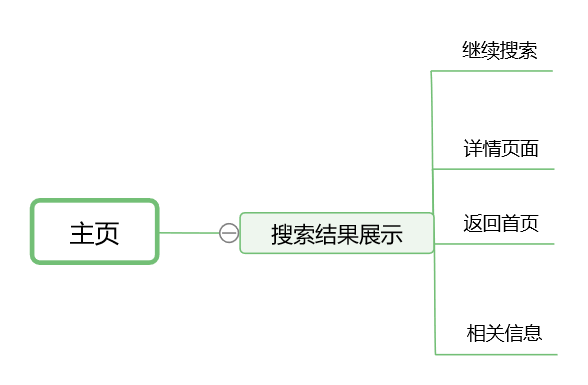
**The structure and detail page**

**Design idea flow chart**:

If the MVC development model is adopted for small projects, it will increase the complexity of the structure, so it is not selected.



**The logical order of the responsible parts:**

****

**html part:**

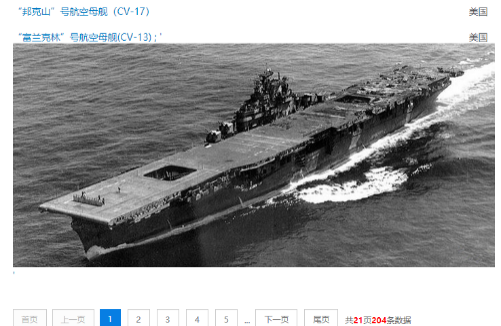
**Headers:**

Basic css rendering calls layui.css and additional headers.css and the common.css file of the material website.

Then because the main pagination layout still needs to be rendered, so continue to add in the style sheet.

**Body part:**

Use div tags and box models. It is mainly divided into three parts, the first is the head package, which is specifically explained on the details page; followed by the container part, mainly using the echo in php to pass the php variable to the front-end html, where the a tag uses target = '\_ blank' Click to jump to a new window to increase user experience. In this part, first determine whether the image value is empty. If it is not empty, display the content of the address pointed to by the image link. This effect is as follows:



This enriches the content of the page while increasing the degree of discrimination. The user can not only click the picture to jump to the relevant detailed page, but also click the text to jump to the corresponding page. Among them, in order to display part of the information of the warship, considering that the attribute value of the belonging country field is not empty, so the belonging country is also displayed after the link.

Finally, the page number is displayed in this part. Since the jump function is already encapsulated when the page class is defined above, it is equivalent to an a tag without using jquery to bind js events.

The third part is the **footer**, which is still described on the detailed page.

The parameters from pagediv.php are sent to the modelmore.php file. It is mainly to deploy the database file to obtain the parameters, which can be displayed directly on the details page without actually being needed.

The detail page is mainly divided into four parts, **header, container, visualization** and **footer.**

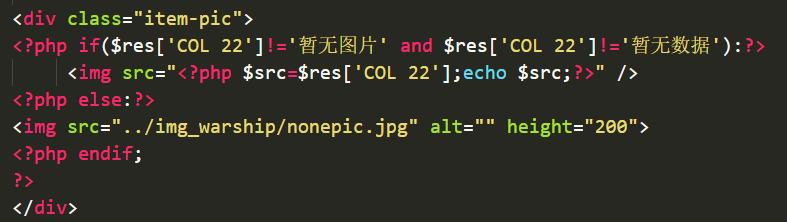
Because it is necessary to judge whether the field value is temporarily no data later, considering the user experience, the missing data is not displayed, so a **strcomp function** is defined in advance. Organize the main CSS rendering together to separate the static files from the front-end web pages.

The first is the head **div class = "header"**, designing an icon style that can return to the home page, followed by a search box, so that you can continue to search for other content on the details page. After that, there is a prompt to return to the page and the About tab, where the About page is the group member information.

Then **div class = ”container”**, design item container tag.

The first is a welcome display. The name of the warship is displayed in this section. Use nested tags in **php** and **html** multiple times.

Next is the display of the country of manufacture of the warship and various time information. Because there is no data in this part of the information corresponding to each record in the database, use the strcomp function mentioned above to make a judgment first.



Then show pictures of warships

Some of the picture links are also invalid, so the judgment will be made first, and the missing ones will be displayed with the local temporary pictures. Use the img tag in the mixed PHP and HTML language mentioned above to link to the network resource and display it on the web page.

This is followed by all the information displayed by the general ship. Considering the type of information and the display effect, choose to use the form of li tag and bind the click event of layui.js. The content under each li tag is first judged whether it is empty before deciding whether to output it in the front end, and is rendered using a layui-tab-item style table. Because the last part of the data in the category display is missing but the content is still to be displayed on the front end, choose to replace with a specific string.

Immediately after writing **JavaScript** code, define the use of layui, set the bomb layer and element operations. In order to test whether it is successfully loaded, use layui.msg to display hello world first in the front end. Let the user clearly switch the li tag, so set the element to listen to the tab switching function and show how many times it has switched. Finally hide the bottom information.

Then comes the code of the **visualization** team. I have seen that this part of the code overlaps with the rendered style and the page cannot be loaded. Set the position parameter to the default to solve it.

After the visual display, the relevant recommendation function was added. Since it is an unpublished webpage, the recommendation based on search terms is relevant. Directly connect to the database and randomly display 3 related data records according to country, warship type and active time. In order to make the page layout look more beautiful, so the relevant recommendations are packed into a table, each table has only the header without the second line, and then the three tables are put together, the inspiration is Baidu's related recommended layout.

Finally, the bottom of the information display, showing the final project time, showing the team information, the effect is similar to about.

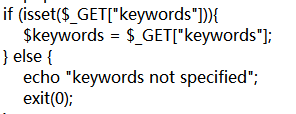
The following mainly analyzes the content of the **headers.css** file. The header of the website is designed to float. For the content of each wrapped div box, set the left and right margins from the top to the bottom. The border of the search box is rendered so that its color is significantly lightened. Set the border properties of the **ul / li** tag and the internal font size and color style. For the main **container** part of the web page, adjust the height of the **item** tags one by one. Finally set the bottom background and select the regular mode.

**Use the frame of the detail page** to display information about the group members on the about page. With the overall framework of the details page, it is effortless to implement.

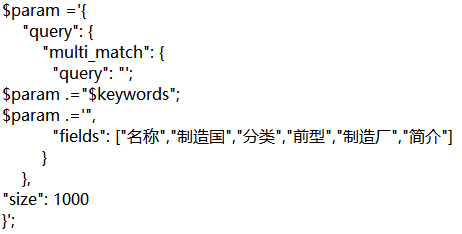
**Using php to Connect Elasticsearch and Do Retrieval**

This part is about how we use php to connect elasticsearch and use it to do retrieval.

Firstly, we use GET to get the value that client input in the Search Box.



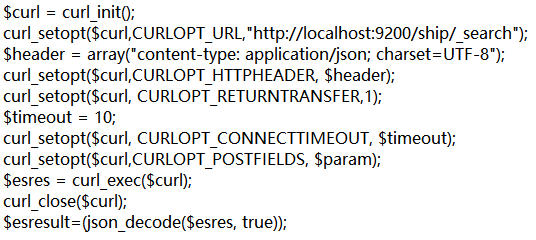
We have already built our index which has been referred to in the previous part. So we set a variable named ‘$param’ in php and assign its value as follow:



We can use this $param to do retrieval of six different fields in our index and they are "名称","制造国","分类","前型","制造厂","简介". And they are retrieved by order. That means, for example, “名称” has priority over the other five fields in the retrieval. And if the content you search is only be included in some ship’s “简介” field, It will be listed behind the returned ships with the contents you search in the first five fields.

We use ‘.=’ to add a new sub string to the string $param because if we write all the sub strings together, there will be a grammar mistake. That’s because the $keyword which we use GET to assign is a variable, and it can only be recognized as a variable in “”. But in the json form variable ,the $keyword must be in a ‘’, so we have to use ‘.=’ to add sub string to assign the variable $param.

Then, we use some codes to connect elasticsearch:

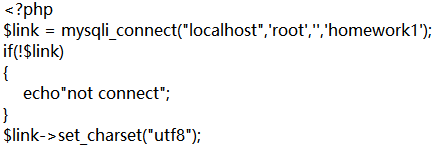


The variable $esres is in json form, and we use decode to change it into a array variable $esresult. And then we can use array index to get the elements in the array.

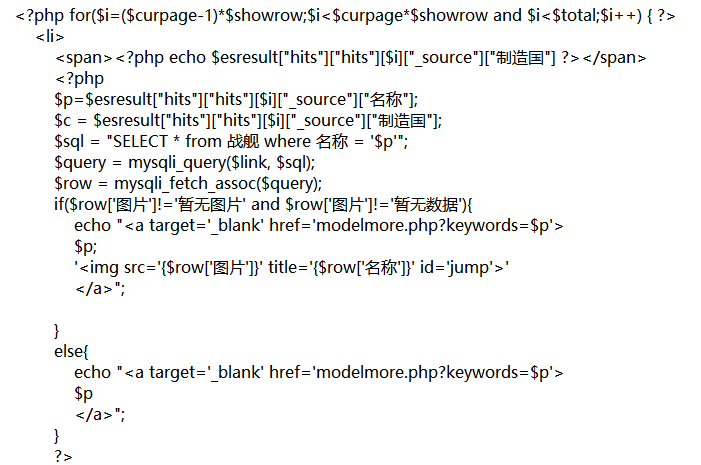
1592212831(1)

The variable $total is the number of documents that we get by elasticsearch.

Except connecting elasticsearch, we should also connect mysql because the page we get from inputting things in Search Box will include the picture of the ships and we don’t have the links of the pictures in our index. The codes connecting mysql are as follows.



And then, we would like to show the codes that present the returning list we get from inputting in Search Box.



In the first row of the codes, the variable $curpage is the page we are in now. For example, like the follow picture, we are in the second page.

1592213536(1)

And the variable $showrow is ten, which means the number of searching results we can see per page is ten. We use for calculate to show every results per page in the returning order.

Then, we use the array $esresult to set $p as the name of the ship, $c as the country of the ship. And we use the sql grammar to find the picture link in the mysql database with the ship’s name $p;

Because some of the ships in our database don’t have picture and such ships’ picture link will be sorted as “暂无数据” or “暂无图片” in our database so we need use ‘if’ to judge. If it has a picture, we firstly echo its name and use href=’modelmore.php’ to make it can jump to the detailed information page of the ship, and then we show the picture of it.

The result is like this:



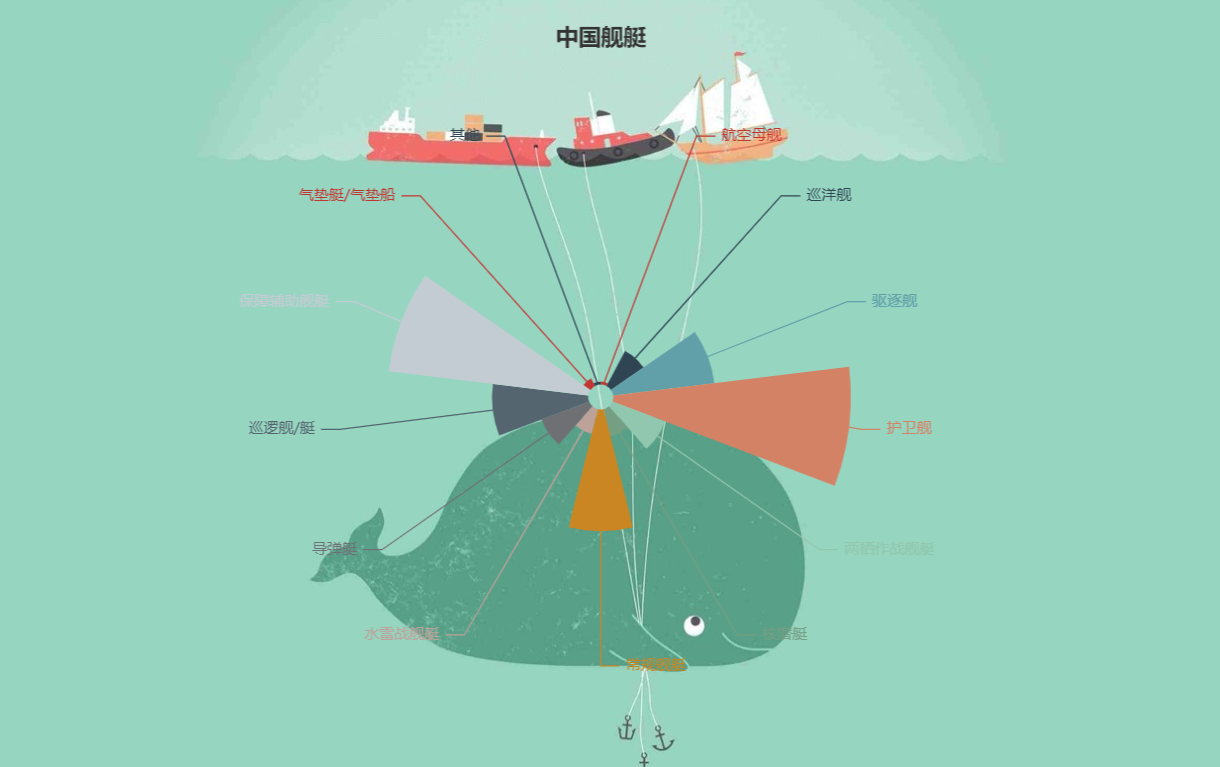
If the ship don’t have picture, we just show its name and make it can jump to the detailed page of the ship.

Above are all about how we use elasticsearch in php.

**Visualization**

In this part, we use echarts to build visualization.

**GRAPH 1:**



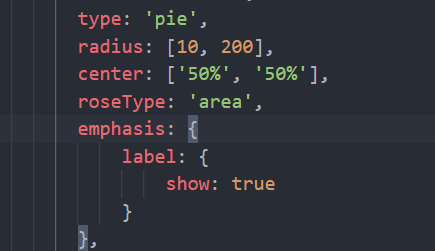
This chart is to show the number of warships in each country. So we choose to use the way of rosette, so that the figure becomes intuitive and beautiful.

Detail:

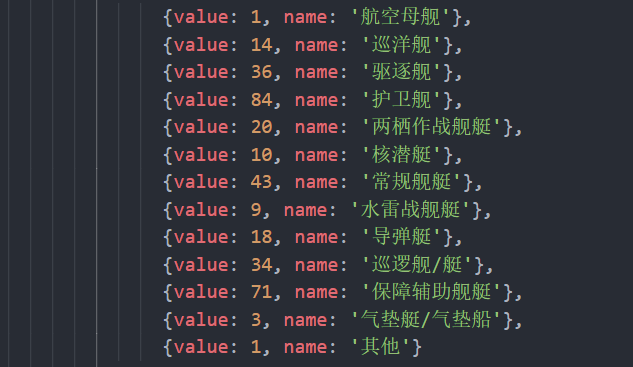
First we determine the location of the title and the format of the data.



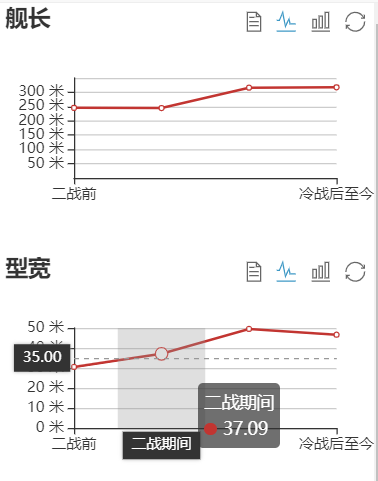
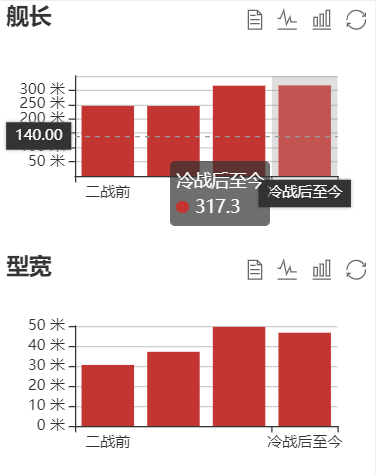
Then we define the chart as a rose chart and define its size and center, and use the size of each sector to determine the size of each category.



Finally we use the data attribute to fill in the data and the name.



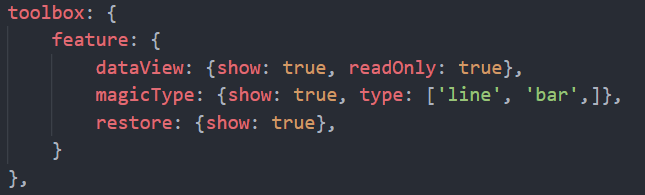
**GRAPH 2:**



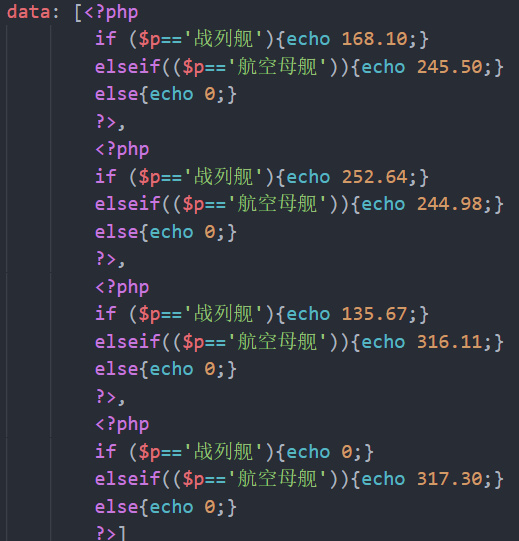
This chart can reflect the change in the country's ship type attributes over time. In order to make the graph intuitive and beautiful, we designed the function that supports the switch between bar graph and line graph.

Detail:

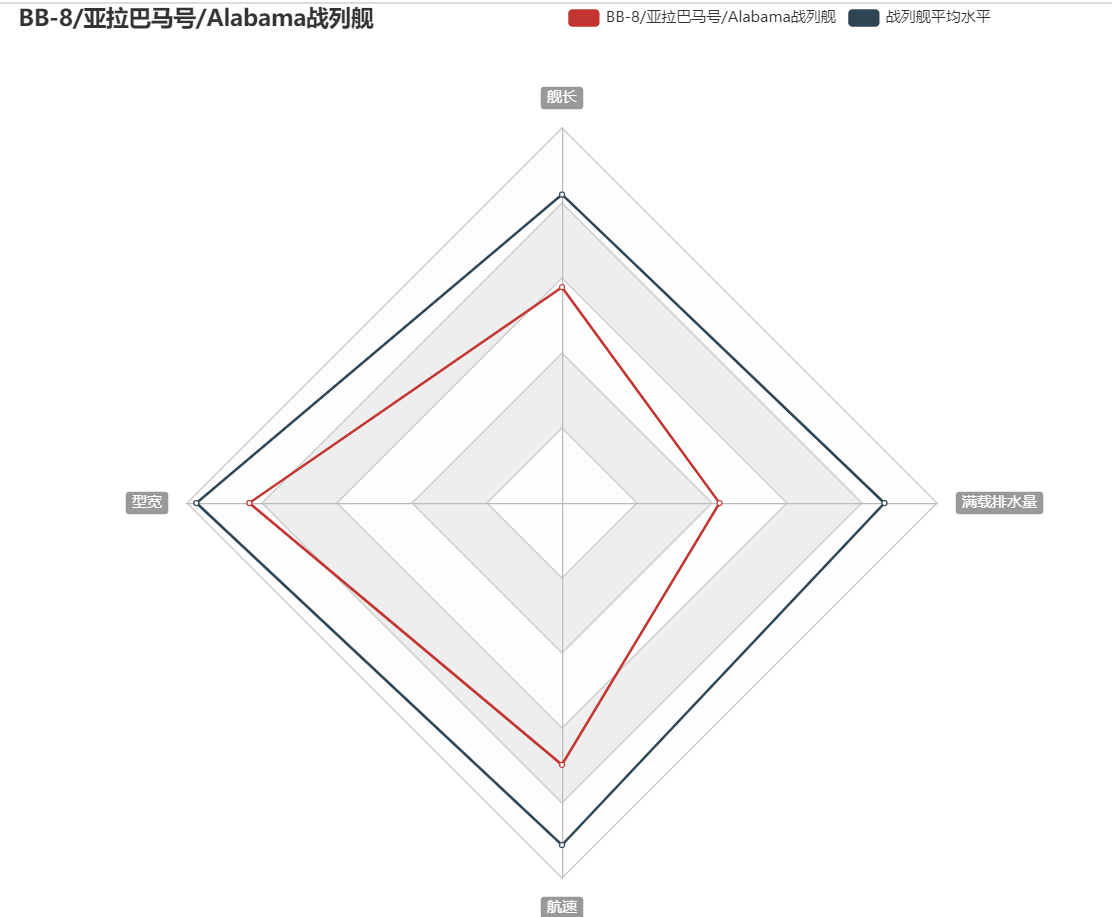
We use the toolbox property to enable us to switch between bar and line graphs.

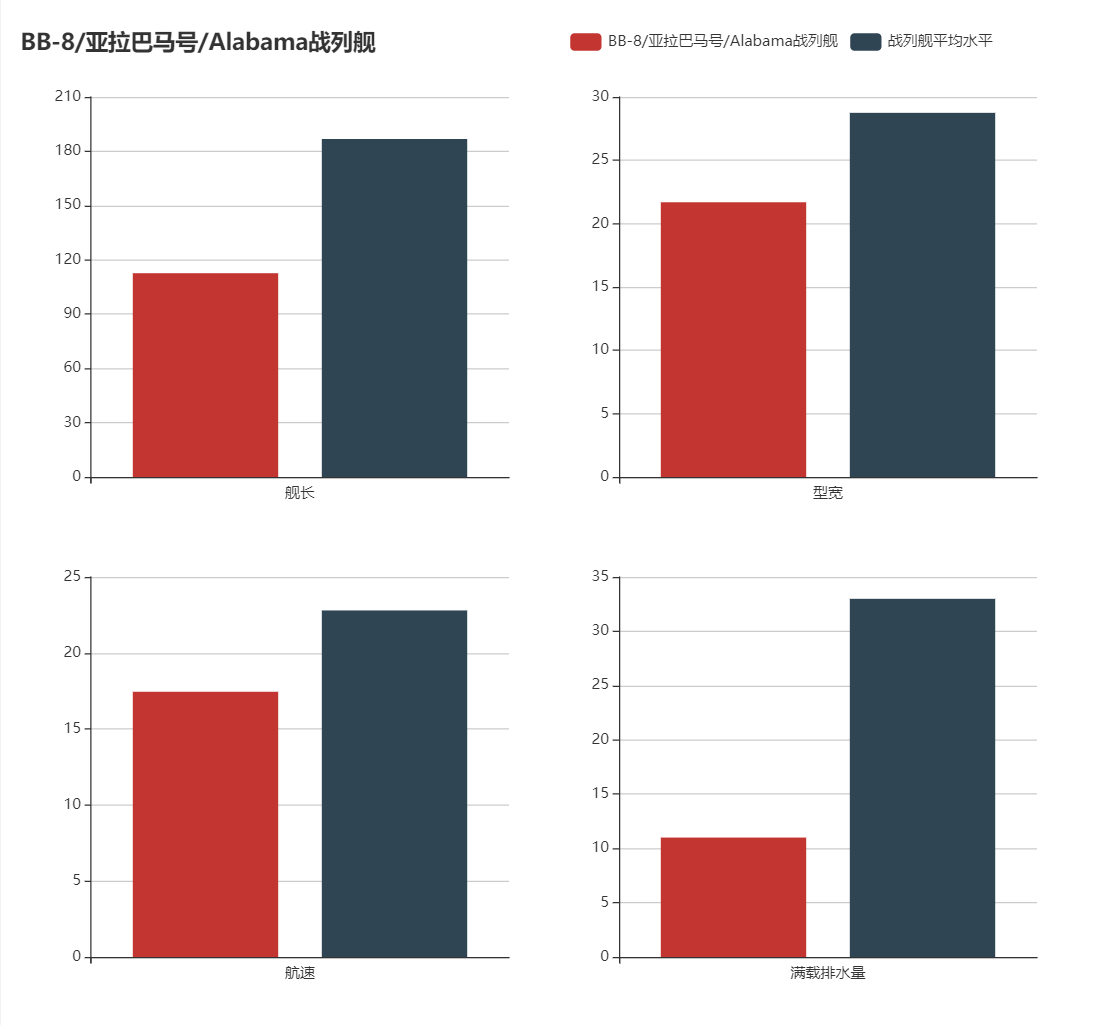


While filling in the data, we call the data stored in PHP directly, and this allows the graph to change dynamically as PHP changes.



**GRAPH 3、4:**





In each ship page, in order to be able to get a more intuitive level of each ship performance, we designed a comparison of the data per vessel and the average of the data for that type of vessel. To make the comparison more intuitive and scientific, we used a radar chart and a bar chart.

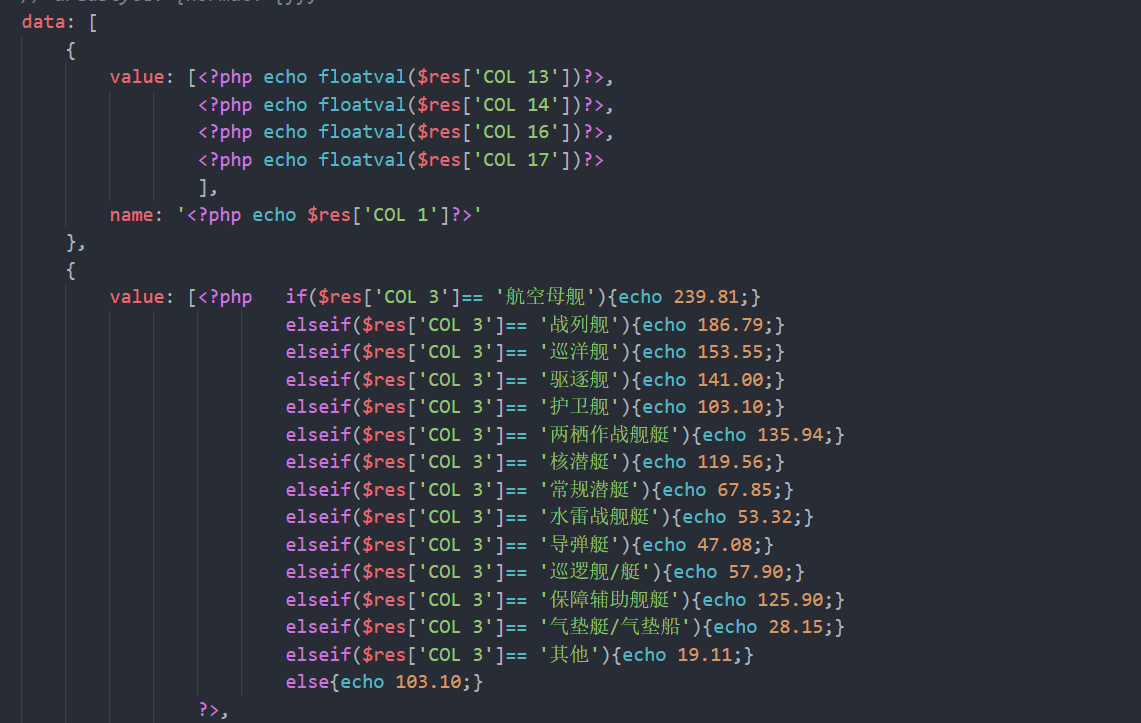
Detail:

①radar chart:

First we defined a radar chart and its four sides.

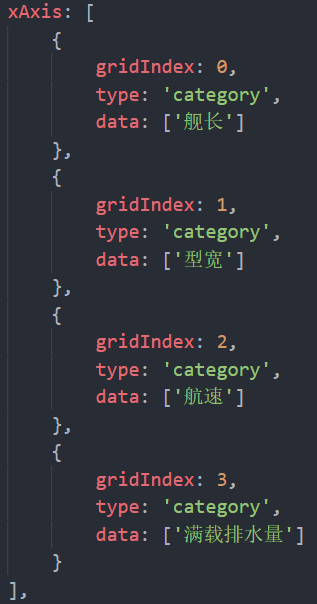
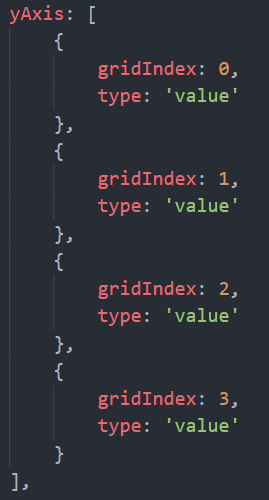


We then populate the radar graph with the data through calls to the data in PHP.



②bar chart

First we define four axes in a bar chart.

Then we finished populating the data.



**Summary**

With the help of TAs, we used tools like pycharm, mysql, elasticsearch, apache and echarts and so on to build a website system. We learned much during this course, not only the way to use these tools, but also the way to make a team work together and every team member do their part efficiently. Our website has some basic functions, but I think we still have a long way to go to make it better.