Stocking

Java EE Project Document

A LIGHTWEIGHT STOCK DATA DISPLAY AND ANALYSIS PLATFORM

JAVA EE, AUTUMN 2017

BY

1552635 胡嘉鑫
1552672 吴可菲
1552673 陈明曦
1552674 李　源

同濟大學
TONGJI UNIVERSITY

Tongji University
School of Software Engineering

Contents

1 Background **3**
1.1 Team Member . . . 4
1.2 Other Information . . . 4

2 Function and Interface **4**
2.1 Brief Introduction . . . 4
2.2 Manual Workflow . . . 5

3 System Framework **9**
3.1 Technologies Used . . . 9
3.1.1 Spring Boot . . . 9
3.1.2 Maven . . . 9
3.1.3 MyBatis . . . 9
3.1.4 Spring RESTful . . . 10
3.1.5 Dependency injection . . . 10
3.1.6 Scrapy . . . 11
3.1.7 Requests . . . 11
3.1.8 LSTM . . . 11
3.2 Entire Framework . . . 12
3.3 Data Acquisition . . . 12
3.4 Data Analysis . . . 12
3.5 Back-end Structure . . . 13
3.5.1 API Description . . . 13
3.6 Database Design . . . 15

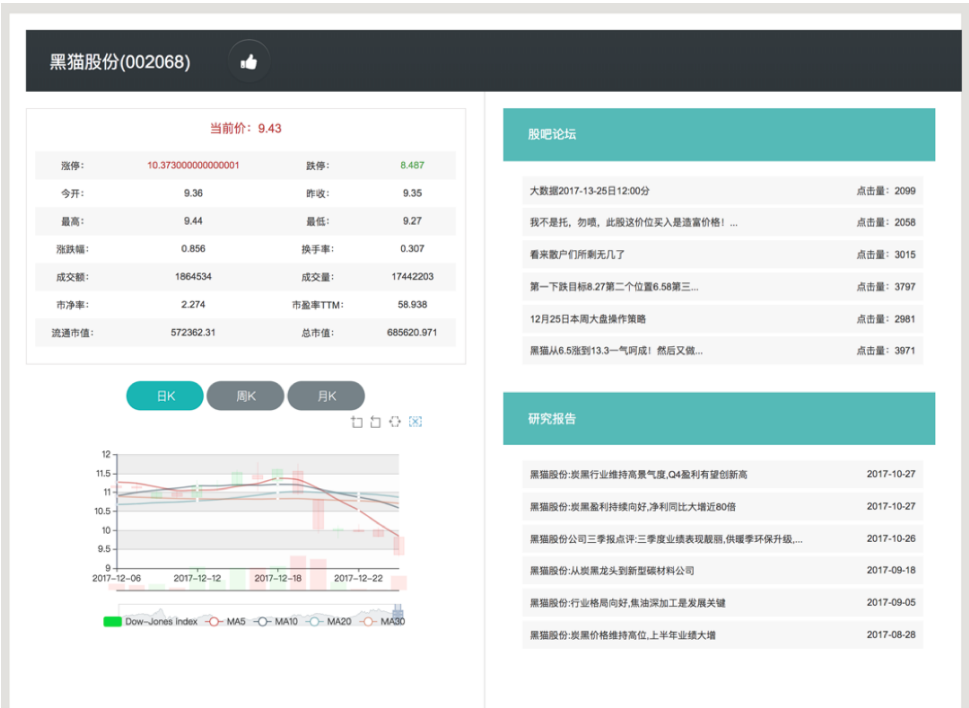
4 Creativity **16**
4.1 Warehouse . . . 16
4.2 Data Mining & Machine Learning . . . 16
4.3 Performance test . . . 17

1 Background

Now there are many different the financial data display and analysis platforms. As you can see, this is the interface of the Sina Finance to view a stock. In fact, not all information is useful to users. For example, some ads, hot topics recommended that users may don't like.

So, we would like to achieve a lightweight stock data display and analysis web platform based on Sina Finance. However, we will discard some redundant information, coupled with some new technology, like Data Mining, Machine Learning and Warehouse.

The interface we finally realized can be see as follow:



1.1 Team Member

- 1552635 胡嘉鑫: Back-end & Database, 25%
- 1552672 吴可菲: Back-end & Front-end, 25%
- 1552673 陈明曦: Data Analyze & Front-end, 25%
- 1552674 李　源: Data Analyze & Database, 25%

1.2 Other Information

- Front-end: HTML5 & Angular.js
- Back-end: Back-end: Spring Boot & Mybatis
- Database: MySQL 5.6.25
- Data Source: tushare & Sina Finance
- Data Analyze: LSTM
- Github: https://github.com/FoxerLee/Financial-Data-Analyze

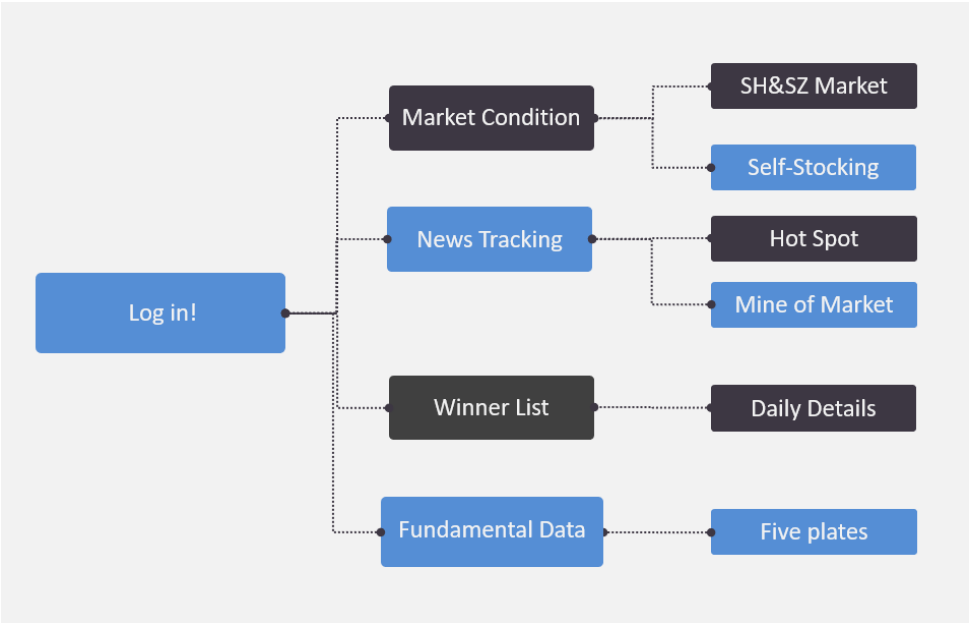
2 Function and Interface

2.1 Brief Introduction

Our program mainly have four part:

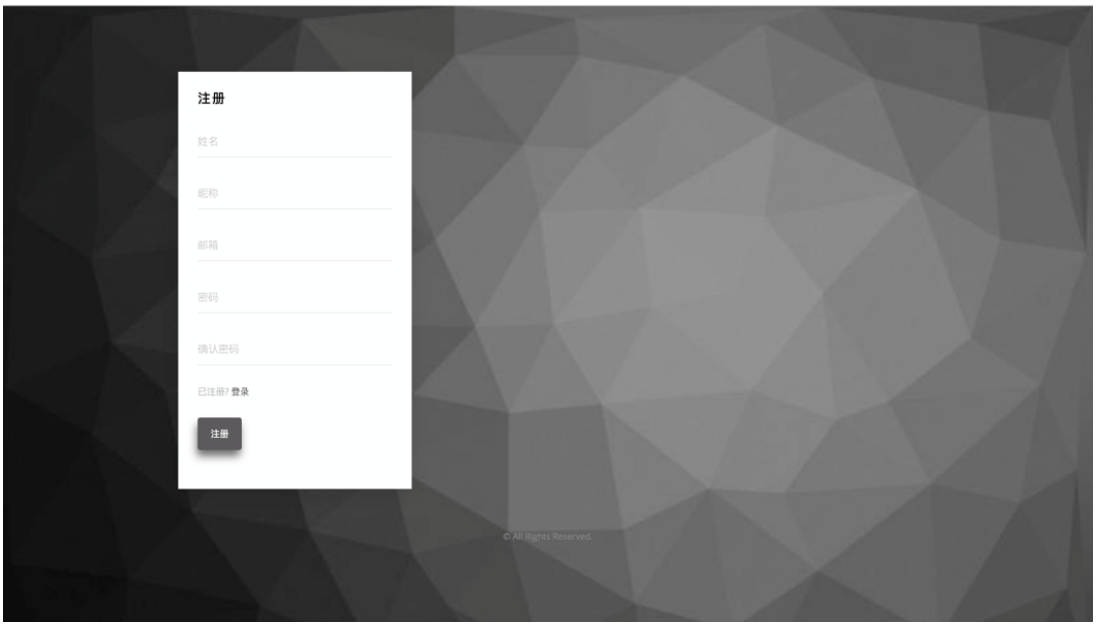
1. Market condition, which consist of stocks of SH and SZ market, and Self-stocking as well.
2. News tracking part take in charge of some hot spot news and mine of market.
3. Winner List shows daily detail of the stock market.
4. Fundamental Data shows five plates.

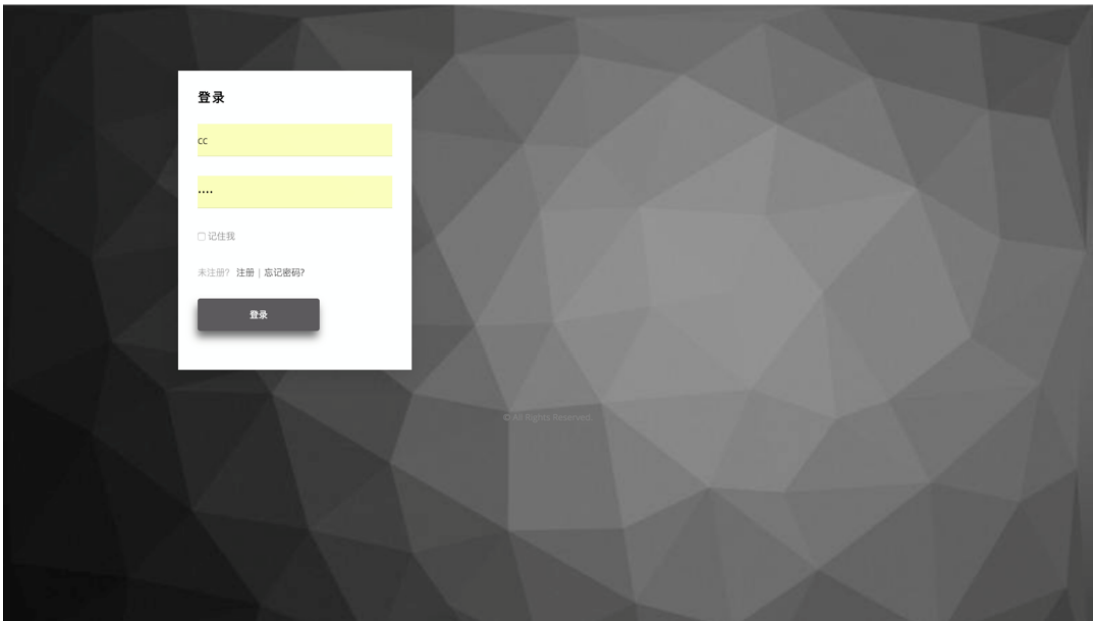
And the whole functionalities are shown below:



2.2 Manual Workflow

This website focus much on individualization so firstly the user is required to register and log in, then jump to the big data.





This page uses heat map to provided users with comprehensive understanding of the stock market. In the above part, 49 sectors of A-share is ordered by turnover and the background color presents the change percent, red—greater than zero, green—less than zero the color is darker when the absolute value is larger. You can move cursor to one sector block to see corresponding turnover, change and change percent, if you click the block the below part will show you at most 50 corresponding stocks ordered by turnover too. You can also click the stock block to go to corresponding detail page and see more information. There is a search box on the sidebar, enter stock code and this is another way to go to detail page.

In detail page, click the above like button, you can put this stock into the list of self selected stocks. On the left side, it shows some basic information and the K line chart. You can click these three button to choose day K, week K or month K chart. On the right side, there are some posts of forum and reports. All of them can be clicked and jump to the origin page just like this and this

The news page will show news according to stocks selected by yourself. Click the magnifying glass icon, then jump to the origin news page.

The winner list page will show different stocks with different reasons such as their daily amplitude have reached 15% and they are all real-time changing

The fundamental data page shows individual company information of different dimensions such as profit ability operating capacity and cash flow you can click different button to choose different dimension.

The personal page shows users own information and the list of self selected stocks. Click the stock' s name you can see corresponding detail page.

3 System Framework

3.1 Technologies Used

3.1.1 Spring Boot

Spring Boot framework is very popular currently. Compared with traditional Spring MVC, Spring Boot has less complex xml configuration files and configuration classes such as WebConfig, instead, it encapsulates them into annotations that allow programmers to easily call annotations to configure filtering effects, such as @RestController It encapsulates the original @Controller so that it can be scanned by the main function as a Component. In addition, the return value of all the functions in this class is automatically converted into a Json format string, greatly reducing the Json data in the framework of separation between the front and back ends Transfer workload. In addition, another very handy place of Spring boot is that it encapsulates the DispatcherServlet class and Configuration class, which are replaced by annotations and a very simple main function that can execute the project; In addition, it has a built-in Tomcat which can add dependencies automatically so you don't have to build Tomcat by yourself (of course built-in Tomcat is convenient for programming and testing, using an external Tomcat container is essential for real deployment). The Spring Boot framework gives us a lot of convenience, which I'll cover in the rest of the documentation.

3.1.2 Maven

In order to unify the way of management, we adopted Maven package dependency management. Maven is a very common package-dependent management tools. Just searching the Internet for the deployment you want and add the package name and version number in the pom.xml, then the packages on Maven basically meet the needs of users. So using Maven in projects is a good choice.

3.1.3 MyBatis

MyBatis is evolved from Spring's traditional Hibernate technology to prevent users from writing lengthy JDBC statements for data connection. Specific steps are listed as follows:

1. Add the URL of the database in the .properties file,including user name, password, and Driver class corresponding to the database.

2.Create a Model class (Model layer) in the project, add the property as a container class after access to the database (due to the name of Mybatis requested data and Model class should be identically the same , each property of the class Should be consistent with the name of the data in the database)

3. Write an interface to add annotations @ Mapper that this class is an interface to provide access to the upper service data, and Mybatis has been our package for this type of implementation. What we need to do is to add each method name as the interface Annotations (@Select, @Update, @Insert ....) and add the sql statement as value. then the sql statements will be automatically connect to the database and the obtained data is filled in the method's return class. These interfaces make up the project's DAO layer.

4. Add the @Autowired annotation to the Service class or Controller class and add an interface property that automatically binds an instance object (which has been done by Mybatis so it can be automatically bound), so just use the interface directly in the method Get the data.

Generally, MyBatis is easy to use, and the Model class can be correspondingly mapped to the database tables, which is pretty convenient for querying simple data. But when you want to join different tables, this technology will require user to create a new Model to accept the data returned by sql. If the join query is relatively large, this method is not a good choice. Fortunately, our project is rarely involved in multi-table join operation, so we choose to use Mybatis as the Model layer technology.

3.1.4 Spring RESTful

In this project we use the separation of front and back-end, which is currently very popular within RESTful programming rules. Here are three reasons for why we use it:

1.The front and back completely separated, remove the clear View layer and Controller layer.

2.All data from the backend to the front end will be delivered in JSON format. The benefits of the unified transport approach are programming specifications and flexibility to use, avoiding the limitations and unknown errors that many MVC front-end templates transmit data.

3.In the last semester's database major project, the vast majority of students using the ASP.NET MVC framework as the back-end framework. Due to the initial understanding of the MVC model (and front-end Razer templates and Ajax will always fight the painful memories) we want to add new ones in this project.

Of course, RESTful programming is still challenging for us:

1.We are not familiar with RESTful rules, backend design URL do not confront to the standard requirement.

2.The back end has less Model and View packaging, causing the front end not only have to design the interface but also to parse the data, which is an increase for front-end workload.

3.Cross-domain problems caused by the separation of front-end and back-end

Although there were a lot of problems when getting started, most of them on the front-end and back-end data interactions, the project hierarchy was very clear and easy to change, so we decided to write it using RESTful rules.

3.1.5 Dependency injection

Dependency injection is one of the core elements of Spring. It is proposed to dispense with the process of instantiating an interface (that is, the process of new). As long as the interface and a class that implements it are provided, both can be assembled into a bean in the configuration class. In Spring Boot, the interaction between the Controller and Service, Service, and DAO layers is achieved by configuring the principles of beans.

3.1.6 Scrapy

Scrapy is a free and open source web crawling framework, written in Python. Originally designed for web scraping, it can also be used to extract data using APIs or as a general purpose web crawler.

Scrapy project architecture is built around ‘spiders’, which are self-contained crawlers which are given a set of instructions. Following the spirit of other don’t repeat yourself frameworks, such as Django, it makes it easier to build and scale large crawling projects by allowing developers to re-use their code. Scrapy also provides a web crawling shell which can be used by developers to test their assumptions on a site’s behavior.

3.1.7 Requests

Python’s standard urllib2 module provides most of the HTTP capabilities you need, but the API is thoroughly broken. It was built for a different time — and a different web. It requires an enormous amount of work (even method overrides) to perform the simplest of tasks.

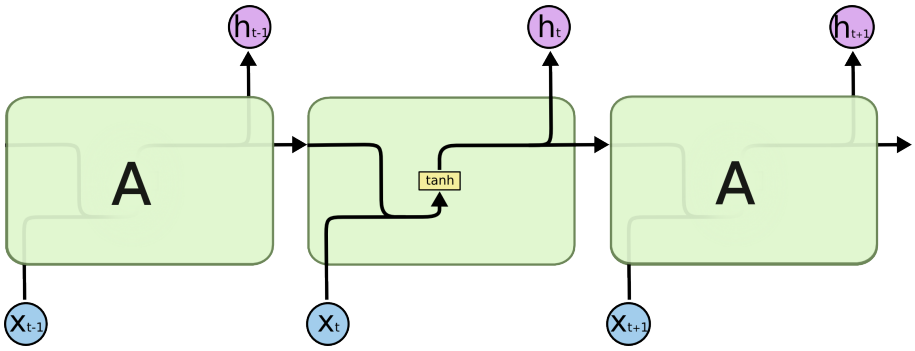
So Requests came out. Requests is a Python HTTP library, released under the Apache2 License. The goal of the project is to make HTTP requests simpler and more human-friendly.

3.1.8 LSTM

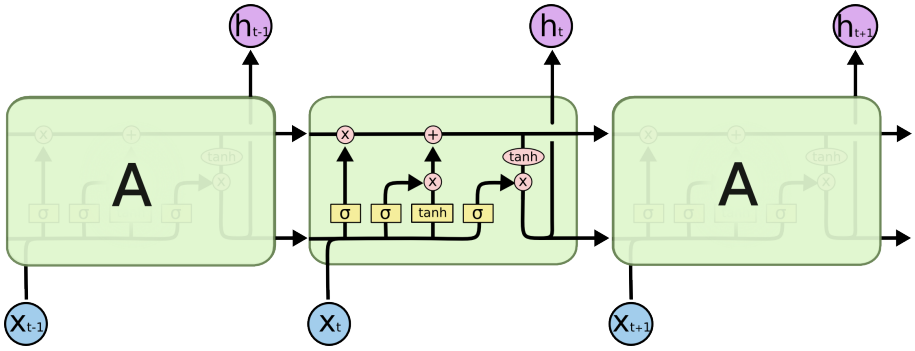
Long Short Term Memory network usually just called “LSTM” is a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work. It work tremendously well on a large variety of problems, and is now widely used.

LSTM is explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

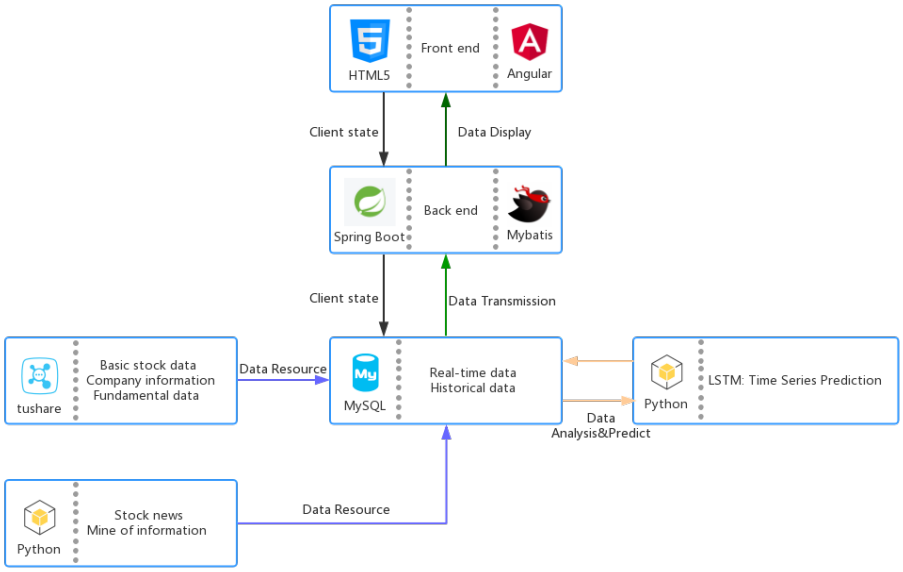
All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.



LSTM also has this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.



3.2 Entire Framework



This picture shows our entire framework.

The overall framework of our project can be divided into four parts: Data Acquisition, Data Storage, Data analysis, and Data display (front end, back end). The following sections of the document will describe each part in detail.

3.3 Data Acquisition

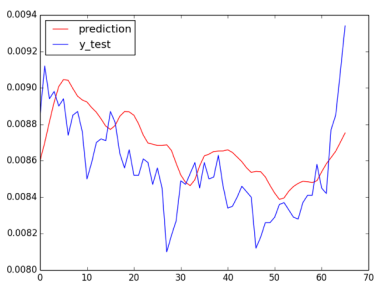
Our data is mainly two sources, the first is a third-party data API for Sina Finance — tushare. We get basic stock quotes, company information and fundamentals data.

The second is that we use two python spider framework to get hot topics new of interest to users, announcements and other content.

We implemented a total of 11 scripts, including 3 spiders to get news and research information. Some data migration scripts are also included.

3.4 Data Analysis

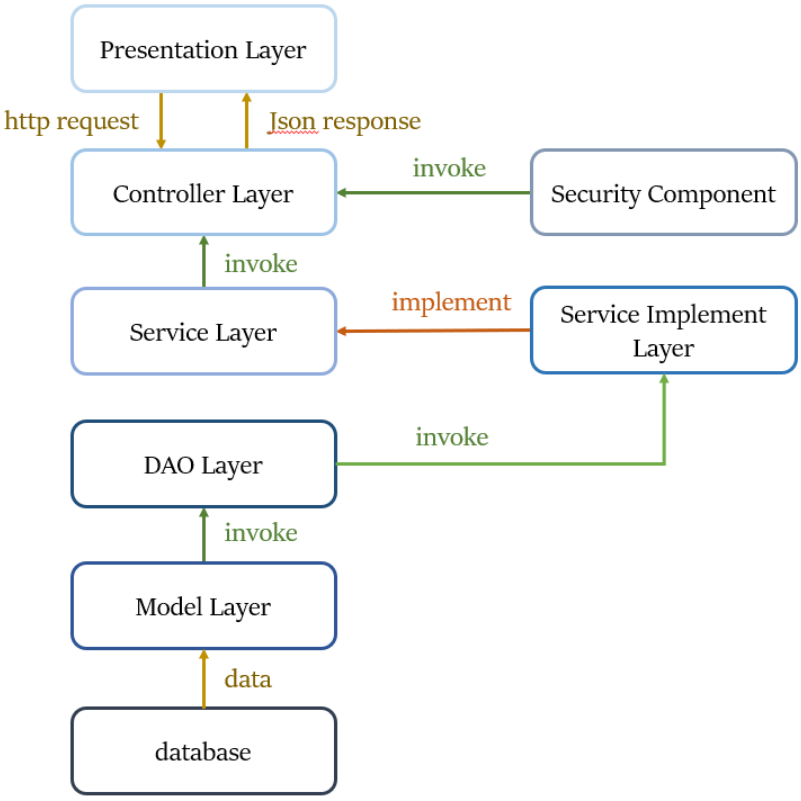
Due to the characteristics of stock data: a large amount of data. We try to use the LSTM algorithm in machine learning to analyze and excavate some useful information in the historical stock market, and on this basis, we give a prediction of the stock's ups and downs in the coming week. This picture shows the specific internal structure of the LSTM algorithm.



This chart is the stock trend we predicted. However, this chart contains many data of the historical market data, so the user can't get a specific prediction about the stock in the coming week. Therefore, we choose the prediction of the nearest week, and we weigh the data according to the time. Finally, we get the up and down prediction of the stock in the coming week and display it right above the detail page.

3.5 Back-end Structure

Presentation Layer: Mainly consisting of .html, .css and .js files to generate interface.
Controller Layer: Provided with API interface.
Service Layer: All the transactions are handled in this layer.
DAO Layer: Provided us with the interface to get data.
Model Layer: Each of which represents one of the tables in database, which is used to store data.
Here I draw a graph to show the structure of our java backend structure:



3.5.1 API Description

localhost: 8080/index

This class provides a homepage access method.

@ Controller annotation indicates that this class is the Spring Controller class, so this class can be configured by using java bean configuration package

@etMapping annotation accepts GET request with path / index. Returned is a static html resource, you can access the index.html file in the static directory and displayed in the browser.

Example request: localhost: 8080 / index

Example returns: Home interface

localhost:8080/news/user?code=...

The @RestController annotation identifies the current class as a Controller class and automatically wraps the return value as a Json-formatted string.

The @etMapping annotation accepts a GET request with the path / news / user.

@Param (value = "code") Remarks Gets the value of the code in the GET request and automatically assigns it to the String code variable.

Verification.verify () is an authentication-enabled function that parses cookies in HttpRequest and looks for the cookie whose name is "fnan" to see if it has the same value as the attribute whose name is "name" in the session. If the same is considered the current visit is the user visited before, validation returns true, otherwise returns false.

The function finally returns "400" or code for the stock code news news,

localhost: 8080/news/code?code=...

The function returns code for the stock code-related news of the simple information used to put information on the various stocks to do a brief display of the interface

localhost:8080/profile/stock/check

The function first authenticates, after passing through the cookie to identify the user id, the final return of

the information held by the user's stock

Possible return value:

1. 400: Not logged in
2. null: Did not find the stock information or anomalies
3. The stock information held by the user

localhost: 8080/stock/add?code=...

This function can add corresponding stock to this user according to code code

localhost: 8080/stock/delete?code=...

@Delete annotation accepts the DELETE request whose path is "stock / delete ..." and returns true or false, indicating whether the deletion succeeded or not

This function deletes the user's particular favorite according to the code

localhost: 8080/profile/check

This function returns the user's name, nickname and email address

localhost:8080/research/code?code= ...

The API returns the stock corresponding to the stock research report summary information

localhost: 8080/research/personal

The API returns access to the user-selected stocks corresponding to the research report

localhost:8080/stock/all

The API is passed to the back-end form, the stock code or stock name, returns the stock the day before the general information.

@PostMapping annotation to accept the post request.

The @RequestBody annotation assigns the returned information to the content variable.

localhost:8080/stock/one

The API is passed to the back-end form, the stock code or stock name, returns the stock the day before the general information

@PostMapping annotation to accept the post request

The @RequestBody annotation assigns the returned information to the content variable

localhost:8080/stock/industry?name=...

This API returns a summary of all stocks in an industry

localhost:8080/stock/brief?code

Back to specific stock brief information

localhost:8080/stock/history/days?code=...

(As well as localhost:8080/stock/history/weeks,localhost:8080/stock/history/months)

Returns the historical information of a particular stock and is used to draw a map of Day k / Week k / Month k graph

localhost:8080/user/login,form:name:...&password=...

The form is transmitted using the x-www-form format

This type of incoming login information back to the back-end verification graph

localhost:8080/logout

Log out user login status

localhost:8080/signup,form:name:...&password=...&nickname=...&mailbox=...

Same as the first API, use for registered users

localhost:8080/winner/all

Get the list all the information

localhost:8080/special/cashflow

Get all the cash flow information.

localhost:8080/special/debt

Get all the debt paying information.

localhost:8080/special/growth

Get all the stock growth information.

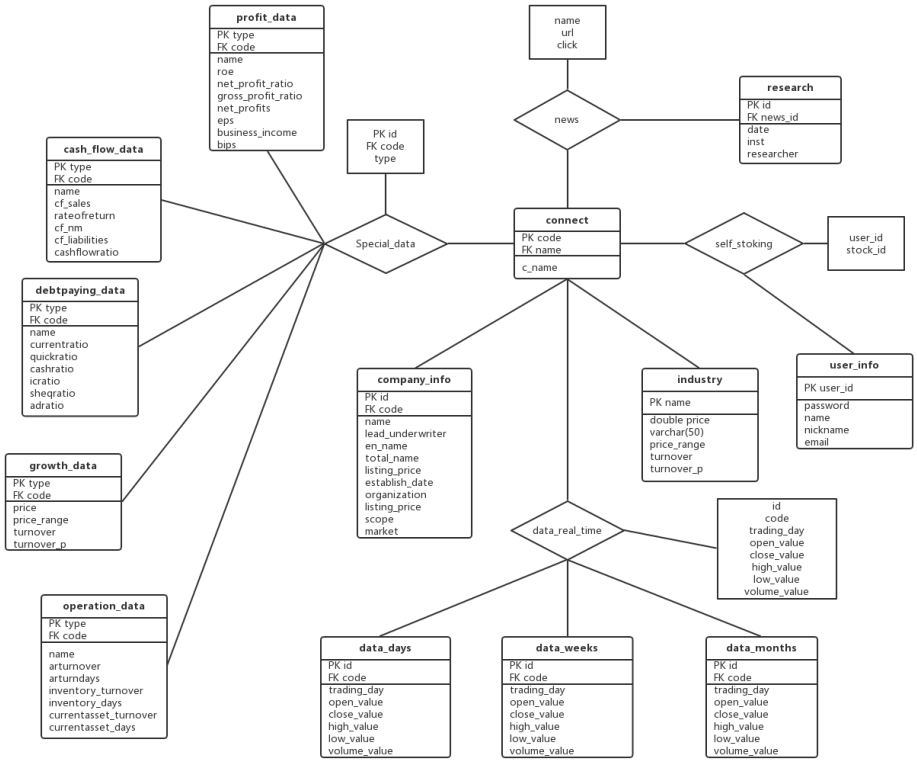
localhost:8080/special/profile

Get all the profile information.

localhost:8080/special/operation

Get all the

3.6 Database Design



Our database design is shown as before. As you can see, the main subject of our project is stock, so the core part of our diagram is 'connect' entity which have code, name and the criteria name of the code. Some related entity such as the stock's company's information, code's industry information and so on are all dependent on the connect entity.

To draw a day-K /week-k / month-k diagram, we should know every stock's information each time, so we create a data_real_time relationship set and store the information of every stock per minute. According

to the data_real_time relation set, the data_days, data_months and data_weeks can change as a result.

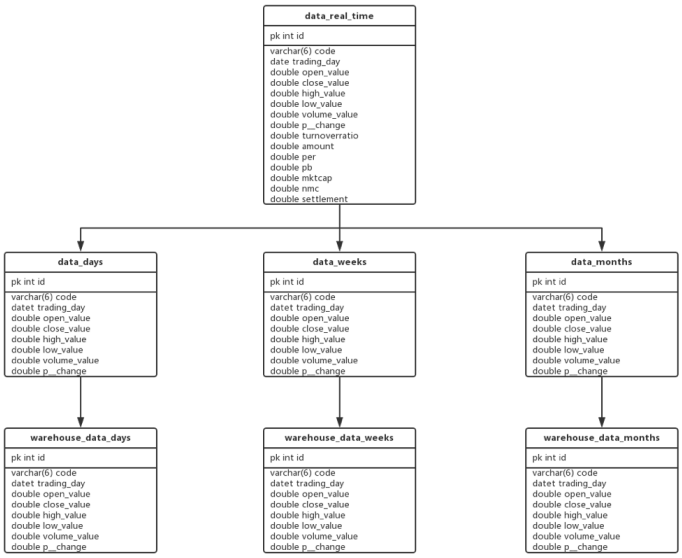
As for user, self_stocking relation set make stock_id and user_id as primary key, because a user may concern many stocks, and one stock may be concerned by different users.

4 Creativity

Due to the characteristics of stock data: a large amount of data, we try to use some new technology, like Data Mining, Machine learning and Warehouse in our project.

4.1 Warehouse

Due to the stock market record can be divided into the real-time market which will constantly updated, and the historical market which will never change, we storage market data respectively into two tables, as it‘s shown.



For each stock, we get the market data every minute, stored in the data_real_time table, and then update the historical data_days data_weeks data_months tables with this data. But each of these tables has about 1.5 million pieces of data. If we need to select a particular piece data to update, the time it takes is bound to be very long. Therefore, we split the three tables into the data_ and warehouse_data_ tables, where the data_ table data will be updated every minute, with only one for each stock, totally 2,700. The remaining 1.5 million data stored separately in warehouse_data_ tables. When the stock market is closed, we will move the data in the data warehouse_data_ tables.

4.2 Data Mining & Machine Learning

Due to the characteristics of stock data: a large amount of data. We try to use the LSTM algorithm in machine learning to analyze and excavate some useful information in the historical stock market, and on this basis, we give a prediction of the stock's ups and downs in the coming week.

The detailed description can be viewed in the previous section.

4.3 Performance test

We tried to use the locust framework for our project performance testing, locust is an open source web project performance testing python package, the following is some of our test results.

Type	Name	# requests	# fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Content Size	# reqs/sec
get	/detail.html?code=000001	218	0	7	12	3	320	28316	2.8
get	/fundamentals.html	213	0	7	12	3	136	15445	1.7
get	/ranking.html	226	0	7	10	3	153	5856	1.8
	Total	**657**	**0**	**7**	**11**	**3**	**320**	**16417**	**6.3**