

PAPER • OPEN ACCESS

## Optimization and Design Based on Data Visualization of Enterprise-class Crawler Systems for Shipping

To cite this article: Hao Wu 2021 *J. Phys.: Conf. Ser.* **1746** 012078

View the [article online](#) for updates and enhancements.



**IOP | ebooks™**

Bringing together innovative digital publishing with leading authors from the global scientific community.

Start exploring the collection—download the first chapter of every title for free.

# Optimization and Design Based on Data Visualization of Enterprise-class Crawler Systems for Shipping

Hao Wu\*

School of Computer Science, JLUZH, Zhuhai, China

\*Corresponding author email: Wuhao@jluzh.edu.cn

**Abstract.** The designed crawler scheduling system is a JEE application based on Quartz scheduling framework that sends crawler tasks to the crawler control system automatically or manually at regular intervals. The crawler control system visually manages and controls the shipping crawlers. The HttpCilent-centric crawler system crawls the shipping website. The message queue ActiveMQ is used as a message middleware for the crawler control system and the crawler system to achieve asynchronous communication and reduce coupling. In this system, front-end frameworks such as Bootstrap, AngularJS, and SweetAlert are used to implement a visual scheduling system and data visualization system. Change the crawler deployment environment by deploying containers through Docker. Add Selenium dynamic page processing for shipping website. Adding message queues and MongoDB database data storage. By adopting a series of techniques and designing, testing and quality assurance of the system, we aim to make the crawler more stable and efficient, and to achieve security, stability and ease of maintenance of the system.

## 1. Introduction

In today's big data, enterprises are always concerned about data-centric change, and constantly around big data to the development and innovation of enterprises. For the ship operation industry, the reptiles are used rationally to obtain valuable data and analyze the data on this basis to cope with changes in the market situation.

The crawler is an application that continuously crawls various information from the Internet by following appropriate rules. However, for enterprises, the need level of data is high and there are higher requirements for crawling stability and success rate, making the design technique complex. Shipping crawlers primarily crawl the websites of specific shipping companies to extract their shipping schedule information and for data collection.

In this paper, it explore and research on shipping and enterprise crawler, the crawling of shipping websites, the use of JMS in enterprise applications, how to build web applications through front-end technology, etc. The design of the enterprise shipping crawler has high availability, visual implementation, and user-friendly interface. The design of the enterprise shipping crawler has high availability, visualization, user-friendly interface, and enjoy the powerful experience of human control.

## 2. Relevant Technologies

### 2.1. MongoDB

This system is a single-page application, the transmission of data is json format, and MongoDB is based on the Bson format, a good match with the front and rear data transmission, no cumbersome data format conversion problems.



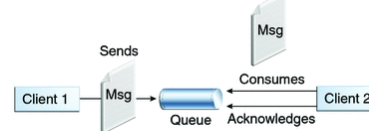
Database design 8 document sets: the User Set, the Scheduling Job Set, the Proxy Set, the Task set, the Task Log Set, the Crawler Configure Set, the Crawler Results Set.

## 2.2. ActiveMQ Message Domain

2.2.1. *ActiveMQ*. Apache ActiveMQ quickly supports multiple cross-language clients and protocols.

2.2.2. *The message domain*. JMS regulates each domain, while defining domain compliance.

Point-to-point (PTP) is built on top of the message queue senders and receivers. The queues hold all messages sent to them until they are consumed or expire.



**Figure 1.** PTP message domain.

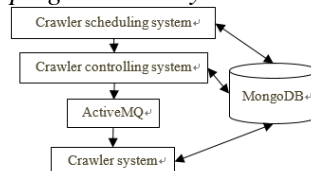
2.2.3. *Helminth*. Since the crawler needs to not miss any and every crawler task to be received 100% successfully, PTP ensures that every message is handled successfully.

## 2.3. Quartz Scheduling Framework

Quartz's core concepts include Scheduler, Job, Trigger, JobDetail. There is an abstract method execute in the Job interface. JobDetail contains task details through which to get Job's name and description information; trigger Trigger for when rules are triggered; CronTigger for time-critical situations, such as a few hours a month, and a Simple Trigger for execution several times a day; and finally, a registration for JobDetail and Trigger to Scheduler.

## 3. Implementation of Enterprise-class Shipping Crawler Systems

### 3.1. Structure of Enterprise-class Shipping Crawler Systems



**Figure 2.** The Crawler scheduling system.

3.1.1. *Quartz Listener*. The scheduler implements the ServletContextListener interface when the application is started, and calls the contextInitialized method once. To start inherits the JobSchedule class, which is a subclass of Thread.

3.1.2. *Job scheduling class*. By execute method, obSchedule inherits Thread class, put all status-opened jobs into newJobsMap, code currentJobsMap=newJobsMap, iterate currentJobsMap in the middle, determine key value with newJobsMap, remove closed jobs from iterator, and call unsheduleJob method to stop jobs. Iterate through the newJobsMap, add and re-update jobs, and determine if the Cron expression is legal, then call scheduleJob and reshceduleJob to add jobs and reschedule jobs.

3.1.3. *Custom job classes*. Implementation of the Quartz Job interface, the implementation of the execute method calls Post to the controller, set JobContent to Task class fields, and sent to the controller ReceiveTaskServlet, response success set the status of the Task to NEW.

### 3.2. The Reality of the Crawler Control System

**3.2.1. Crawler Task Producer.** Initialize the queue connection, create a session and a crawler task message producer object, while loop to determine whether the queue size is smaller than the queue length, query the database, get the first corresponding ship company crawler task, get the details of the Crawler task, and call producer's send message to the queue. Update the status of the current task to USE and last updated.

**3.2.2. Crawler mission timeout.** Get the creation time of the task before entering the queue, determine whether the creation time is within the time range configured by the Crawler, enter the time-out logic if not, update the crawler task status to TIMEOUT and last updated time, and jump out of the current while loop and do not execute the logic of entering the queue.

**3.2.3. Limit Crawler tasks.** Access the first access time within the time range, and increase the number of times. Determine whether the number of visits in the effective time and quantity limit. If valid, the number of v perform the limiter logic according to the front crawler limit on state isits is updated, otherwise not to enter the queue. If the timeout is re-recorded this time and reset the number of times, through the continuous cycle can achieve a certain amount of time to limit the number of crawler messages into the queue.

**3.2.4. Crawler configuration interface.**

- Crawler placement

Table's first th placed input tag, type for checkbox, provides full selection function and check all list auto-check function; ng-click triggers the user's click event, which triggers the change of sorting style and the corresponding descending ascending function;

```
<th ng-click="sortBy('queueSize')">队列大小 <span
  class="sortorder" ng-show="propertyName === 'queueSize'"
  ng-class="{reverse: reverse}"></span>
</th>
```

**Figure 3.** Figure html header.

There is only one tr in the tbody label, ng-repeat is to provide each item in the profiles (collection) will copy the current element once, for the tr element;

```
<tr
  ng-repeat="profile in profiles | filter: property | orderBy:propertyName:reverse"
  ng-class="{bg-selected:profile.isChecked}"
  ng-click="profile.isChecked=!profile.isChecked"
  ng-dblclick=openConfigModal(profile)>
```

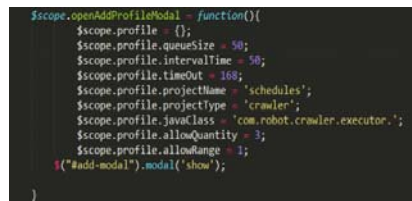
**Figure 4.** Figure html table row.

The tr sub-label is a td tag that displays data, such as the s profile. queueSize, which is a double parenthesis expression that binds the queueSize property of profile.

- The Crawler configuration pop-up box.

Pop-ups use Bootstrap's modal box, covering child windows in the parent interface. You can pop up the add-on crawler configuration detail box for add-modal with \$("#add-modal").

**3.2.5. The AngularJS interaction of the Crawler configuration module.** Open the new crawler configuration pop-up box and set the configured parameters, such as the queue size of 50, scope is the glue between the model data, view and view model, which provides the binding of data or methods, very quick response, here binding HTML double parenthesis values, when you open the new pop-up box, these values are set in the corresponding input box.



```

$scope.openAddProfileModal = function(){
    $scope.profile = {};
    $scope.profile.queueSize = 50;
    $scope.profile.intervalTime = 50;
    $scope.profile.timeOut = 168;
    $scope.profile.projectName = 'schedules';
    $scope.profile.projectType = 'crawler';
    $scope.profile.javaClass = 'com.robot.crawler.executor.';
    $scope.profile.allowQuantity = 3;
    $scope.profile.allowRange = 1;
    $('#add-modal').modal('show');
}

```

**Figure5.** Figure Angular JS.

### 3.3. The Implementation of the Crawler System

**3.3.1. Crawler task consumers.** Initialize queue connection, create session, Queue object, create crawler task message consumer object; consumer set OnMessageHandler, and bind listener, if there is a message, then actively receive the message; when there is a corresponding ship company's crawler task message, it is implemented The OnMessageHandler class of the MessageListener interface calls the onMessage method; gets the details of the crawler task, and calls the corresponding crawler class to execute the crawler through the reflection mechanism; after crawling, stores the data into the database, and updates the state of the current task to CAPTURED and the last update time; calls the onMessage method. acknowledge() method to confirm that this message has been consumed.

#### 3.3.2. Climb the ship's static page.

- Crawlers.

To write a crawler request to a browser, through Java's native HTTP request library or httpClient under Apache, view the process of network request through the browser developer tool mode, view each request method POST or GET, request address, SENT parameters sent, request header, etc., and finally write the crawler's request logic based on this information. The simplest GET request requires only one request address and a call to http's GET method. However, the ZIM Shipping website, where the page content is loaded by JavaScript or Dynamic AJAX, can simulate JS, piece together its parameters and parse the Johnson returned by AJAX.

- Climb the MAERSK website.

View Network requests via the chrome developer tool, with the first request spanning the most hours, so view the details of the first request. View the results of this request, the address is <https://my.maerskline.com/schedules/pointtopointresults>, the protocol is https (important), the request method is POST In this way, there are post parameters, by comparing the request parameters and search criteria, you can find that some parameter values are fixed, such as b.lines.iso:42G1 and b.lines.0.quantity:1. Other parameters need further analysis; Looking back and re-entering the From and To forms, there are the following requests, and the values of the post parameters b.from.geoId and b.to.0.geoId that can be obtained from the 3-step request are obtained from this request.

Finally analyze the other parameters of the 3 steps, found that b.numberOfWeeks and b.date are also regular, only authenticityToken, try to search for this keyword from HTML text, the result is id is autoenticityToken this input element, extract value can be. But this HTML is the result of a search, so re-browser address re-request <https://my.maerskline.com/schedules>, and then query the HTML text of this GET request also has an input element idauthenticityToken.

#### 3.3.3. Climb the ship's shipping season dynamic web page.

- Selenium

Selenium drives browser automation. WebDriver is designed to provide a simpler, cleaner programming interface, in addition to addressing some of the limitations in the Selenium-RC API. Selenium-WebDriver better supports dynamic Web pages, pages where page elements change without reloading.

- Climb the ZIM website.

Identify the search terms From, To, Start Date, Weeks Ahead; view the Network request via the chrome developer tool, the fourth request spans the most time, and view the details of that request. Viewing Network requests via the chrome developer tool, the fourth request spans the most time.

By looking at the response of this request, determine the information required, then view the form has paging, click paging, view the next page of the request, find that the table has been updated, and the entire page is not refreshed, so view the following request;

After viewing, the two requests are xhr, the first Web service form returns js source code, analyzed js has an onload event, which happens to be the performance of the second request, and the second request header parameter data comes from js, for unfamiliar js is quite difficult, and the parameters are complex, processing will be time-consuming, so we choose Selenium automation way;

Selenium allows us to operate the browser like a real user, and opens up a multilingual, easy-to-understand api, which of course includes support for Java, where WebDriver is more powerful, directly using the browser's native support to operate the browser, it allows you to access a url, and then through the mouse keyboard and other page elements to carry out various operations, while supporting the call to js and so on;

### 3.3.4. Extraction of shipping information.

- Jsoup

Jsoup parses HTML, and Jsoup documents HTML.

- Extract MAERSK shipping.

First convert html to jsoup's Document object, locate the table tag, select all the tr tags that start with the row, traverse all the tr tags, extract depart, arrival, vessel, and so on. As shown in Figure 6.

```
Document doc = Jsoup.parse(html);
Elements trEles = doc.select("table[class='table table-striped cols20 schedule-table']" ).select("tr[id^=row]");
for (int i = 0; i < trEles.size(); i++) {
    Element trEle = trEles.get(i);
    String departure = trEle.select("a").get(0).text();
    String arrival = trEle.select("a").get(1).text();
    String vessel = trEle.select("a").get(2).text();
    String departureTime = trEle.getElementsByTag("strong").get(0).text();
    String arrivalTime = trEle.getElementsByTag("strong").get(1).text();
    String transitTime = trEle.select("td").get(2).text();
}
```

**Figure 6.** jsoup code diagram.

## 4. Enterprise-class Shipping Crawler System Testing

### 4.1. Crawler System Unit Test

Through the setting of the Crawler instructions for each shipping site, if all use cases pass, the various fields of its shipping results are empty, such as ship name, route, number of days, etc. are not empty, which indicates that the unit test was successful.

### 4.2. User Interface Testing

After manual testing of the Web user interface of the Crawler dispatch system and the Crawler control system, the following conclusions are drawn, including whether it is easy to operate, whether the font is appropriate, whether the layout is reasonable, etc.

**Table 1.** User interface test results table.

Test requirements	Test results
The size, color, and layout of the icon are reasonable.	Passed
Button, the size of the interface, and the coordination of the two.	Passed
The prompt information fonts are consistently passed.	Passed
Information displays visibility through.	Passed
Each domain confirms that there is a problem with the appropriate prompt information through.	Passed
The interface is easy to operate and easy to use.	Passed
The interface is friendly through.	Passed



## 5. Enterprise-class Shipping Crawlers are Deployed and Operated

### 5.1. Docker

Docker automates the setup and configuration of repetitive tasks for the development environment.

### 5.2. Mirrors and Containers

The image is the file system and parameters to be used at runtime. It has no state and never changes. The use of containers allows everything a piece of software needs to run to be encapsulated in an isolated container. Containers don't bundle the full operating system, just the libraries and settings, regardless of its deployment.

### 5.3. Homemade Mirroring

```
FROM robot/tomcat:latest
COPY ./scheduler.war /usr/tomcat8/webapps
RUN mkdir /workdir
WORKDIR /workdir
ADD ./start.sh ./
RUN chmod +x ./start.sh
CMD ./start.sh
```

**Figure 7.** Dockerfile.

### 5.4. Executing the Build Command

Put Dockerfile, web application war package, and start.sh in the same directory, switch to that directory and type `docker build -t scheduler`.

### 5.5. Compose Runs Multi-container Applications

Compose can run multiple containers. The Compose file configuration app becomes a service. Use commands to create and start all services. Write `docker-compose.yml` and execute the command `docker-compose up -d` directly to run the app.

## 6. Conclusion

Enterprise-class shipping Crawlers, set scheduling Crawlers, control Crawlers and Crawlers function, through the realization of visual scheduling system, data visualization system, shipping Crawler dynamic page processing and Docker container technology deployment, message middleware ActiveMQ, Quartz scheduling framework, AngularJS and other front-end framework technology design and adoption, so that users can easily manage and control Crawlers, but also make the system more secure and easy to maintain.

The structural coupling of the front-end technology can be reduced to conform to MVVM, thereby reducing the use of duplicate components and increasing the reusability of the code. The structured design of Crawler data requires a change in field type to make it more reasonable.

## References

- [1] Luo Gang, Wang Zhendong. Write your own web crawler [M]. Beijing: Tsinghua University Press, October 2010.
- [2] Shao Yizhi, Web research and application based on Ajax schema [D], Wuhan University of Technology, 2008.
- [3] Xie Xiren, Computer Networks, People's Posts and Telecommunications Press, 2008-1.
- [4] Liu Shitao, etc., a brief analysis of the search strategy of internet crawlers in search engines, the journal of Puyang Normal College. (Natural Science Edition), 2016 (09):P60-63.
- [5] Goknin et al., a high-performance spider program that supports Web information classification, a small microcomputer family. 1309-1312: P 2016 (07).
- [6] Luo Gang Wang Zhendong. Write your own web crawlers. Beijing: Tsinghua University Press, October 2016.
- [7] Wisenut. WiseNut Search Engine White Paper .M. Beijing: China Electric Power Press, 2011.
- [8] [8] Fang Jun, Design and Application of Ajax Engine, Computer and Information Technology, No. 03 of 2016.