Design Specifications for Cisco Log Parser Project

Log Files

The logs in the log files have the structure of:
TIMESTAMP | TYPE | THREAD | | CLASS | MESSAGE |
in two log files named "spf-device-manager" and "spf-service-manager"

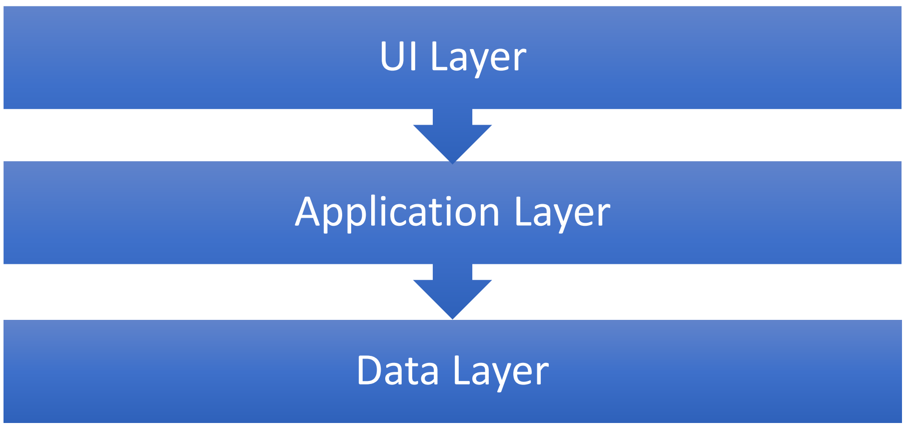
Database

The data base in this project has one table called 'info' with the following structure:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
time	datetime	YES		NULL	
ms	int(11)	YES		NULL	
type	varchar(30)	YES		NULL	
threadname	varchar(100)	YES		NULL	
classname	varchar(100)	YES		NULL	
message	longtext	YES		NULL	
exception	text	YES		NULL	
servicename	varchar(50)	YES		NULL	

Design Strategy

The Design we implemented was the three-tier approach

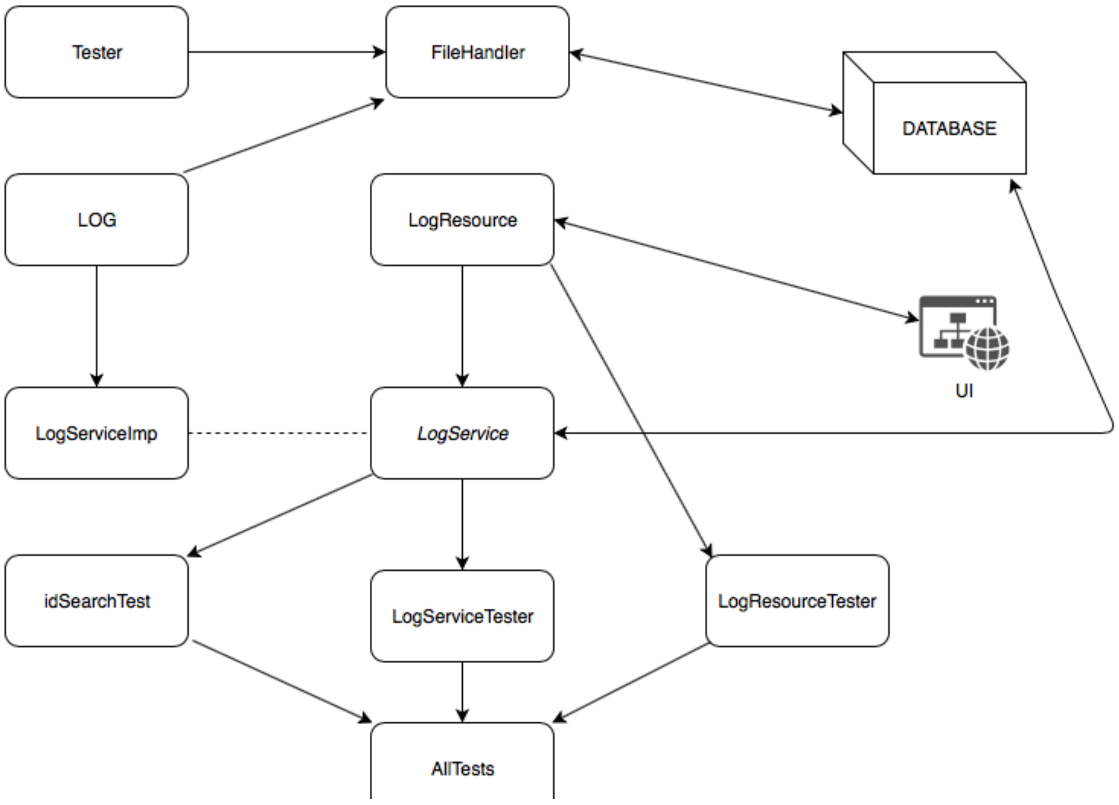


Where the UI layer was implemented as a simple website for accessing and filtering logs.

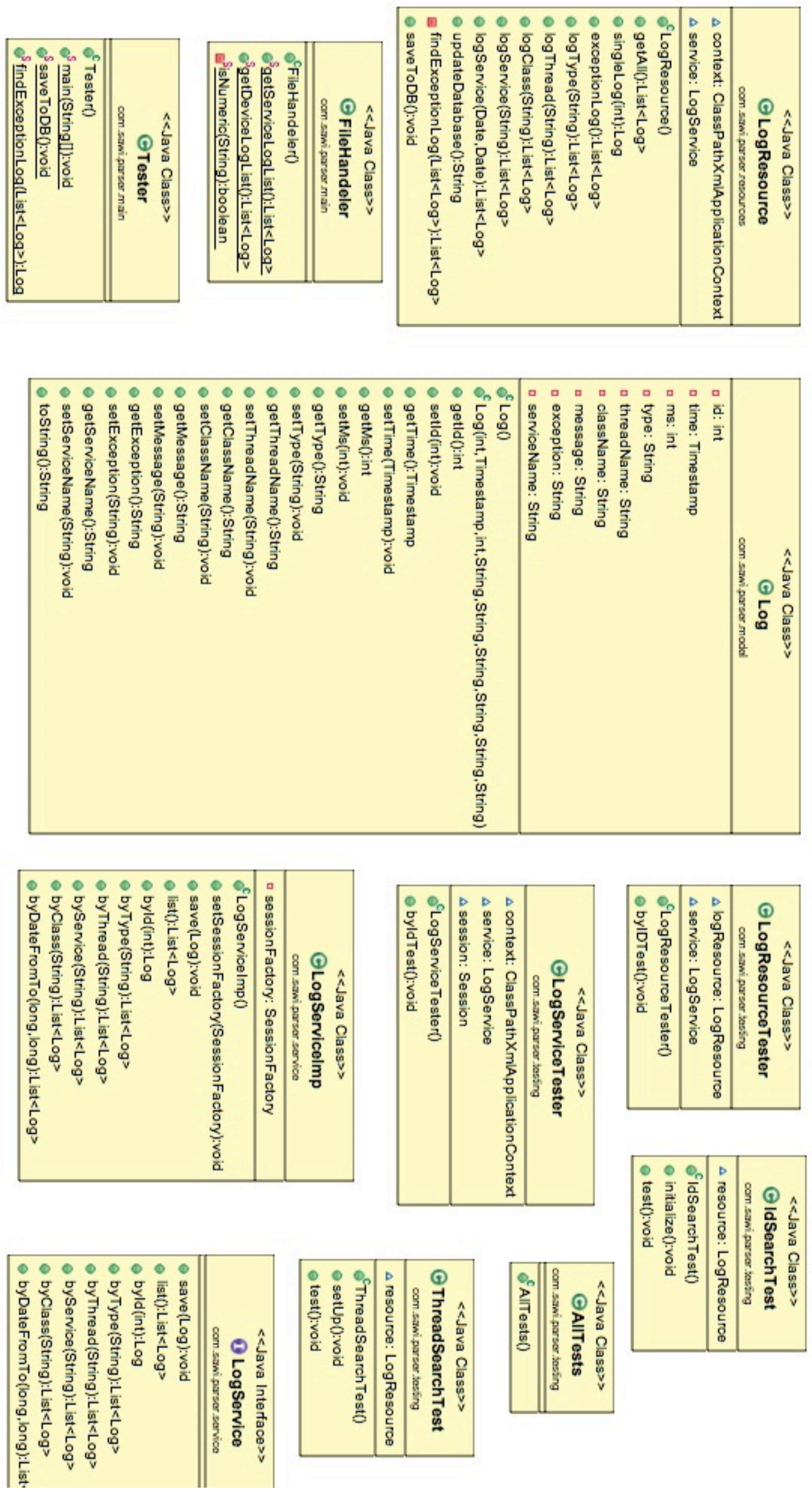
Application layer is the backend of the project that connects to the database and performs the parsing

Data layer is the MySQL database we filled with the logs recorded in the log files

Class Diagram

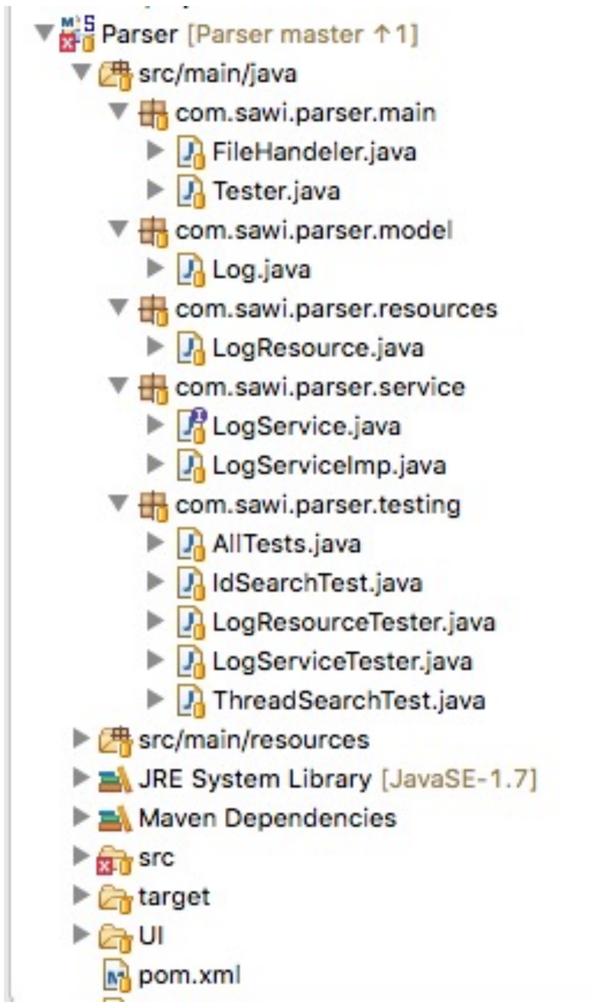


UML



<<Java Class>>
LogResource
com.sawi.parser.resources
context: ClassPathXmlApplicationContext
service: LogService
LogResource()
getAll():List<Log>
singleLog(int):Log
exceptionLog():List<Log>
logType(String):List<Log>
logThread(String):List<Log>
logClass(String):List<Log>
logService(String):List<Log>
logService(Date,Date):List<Log>
updateDatabase():String
findExceptionLog(List<Log>):List<Log>
saveToDB():void
<<Java Class>>
FileHandeler
com.sawi.parser.main
FileHandeler()
getServiceLogList():List<Log>
getDeviceLogList():List<Log>
isNumeric(String):boolean
<<Java Class>>
Tester
com.sawi.parser.main
Tester()
main(String[]):void
saveToDB():void
findExceptionLog(List<Log>):Log
<<Java Class>>
Log
com.sawi.parser.model
id: int
time: Timestamp
ms: int
type: String
threadName: String
className: String
message: String
exception: String
serviceName: String
Log()
Log(int,Timestamp,int,String,String,String,String,String)
getId():int
setId(int):void
getTime():Timestamp
setTime(Timestamp):void
getMs():int
setMs(int):void
getType():String
setType(String):void
getThreadName():String
setThreadName(String):void
getClassName():String
setClassName(String):void
getMessage():String
setMessage(String):void
getException():String
setException(String):void
getServiceName():String
setServiceName(String):void
toString():String
<<Java Class>>
LogResourceTester
com.sawi.parser.testing
logResource: LogResource
service: LogService
LogResourceTester()
byIDTest():void
<<Java Class>>
LogServiceTester
com.sawi.parser.testing
context: ClassPathXmlApplicationContext
service: LogService
session: Session
LogServiceTester()
byIdTest():void
<<Java Class>>
LogServiceImp
com.sawi.parser.service
sessionFactory: SessionFactory
LogServiceImp()
setSessionFactory(SessionFactory):void
save(Log):void
list():List<Log>
byId(int):Log
byType(String):List<Log>
byThread(String):List<Log>
byService(String):List<Log>
byClass(String):List<Log>
byDateFromTo(long,long):List<Log>
<<Java Class>>
IdSearchTest
com.sawi.parser.testing
resource: LogResource
IdSearchTest()
initialize():void
test():void
<<Java Class>>
AllTests
com.sawi.parser.testing
AllTests()
<<Java Class>>
ThreadSearchTest
com.sawi.parser.testing
resource: LogResource
ThreadSearchTest()
setUp():void
test():void
<<Java Interface>>
LogService
com.sawi.parser.service
save(Log):void
list():List<Log>
byId(int):Log
byType(String):List<Log>
byThread(String):List<Log>
byService(String):List<Log>
byClass(String):List<Log>
byDateFromTo(long,long):List<

Project Structure



```
▼ Parser [Parser master ↑1]
  ▼ src/main/java
    ▼ com.sawi.parser.main
      ▶ FileHandeler.java
      ▶ Tester.java
    ▼ com.sawi.parser.model
      ▶ Log.java
    ▼ com.sawi.parser.resources
      ▶ LogResource.java
    ▼ com.sawi.parser.service
      ▶ LogService.java
      ▶ LogServiceImp.java
    ▼ com.sawi.parser.testing
      ▶ AllTests.java
      ▶ IdSearchTest.java
      ▶ LogResourceTester.java
      ▶ LogServiceTester.java
      ▶ ThreadSearchTest.java
  ▶ src/main/resources
  ▶ JRE System Library [JavaSE-1.7]
  ▶ Maven Dependencies
  ▶ src
  ▶ target
  ▶ UI
    pom.xml
```

Components are split into 5 packages:

- Main has the runnable java applications mainly for testing while developing. And the file handling class
- Model has the basic LOG java class
- Resources has the REST api implementation
- Service has the hibernate ORM implementation to access the database
- Testing houses the Junit test cases.

And the pom file has the maven dependencies.

Implementation specifications

- The project was developed in the Eclipse IDE environment. using a maven project structure.

- Hibernate ORM was used to do the connection and operations with the database.

- Also the project implemented Spring framework for the fetching of the service beans that run the Hibernate ORM.

- Structure of the project contained DAO for the logs which for simplicity was just left as the POJO of the LOG class.

- RESTful API was implemented using the Jersey framework to achieve the backend aspect. the API uses http GET requests and returns JSONs containing required log objects.
- The API allows for multiple searches among the recorded logs, including: search by ID, by Class, Thread, Type, filtering logs for those with exception, and filtering logs between certian dates.

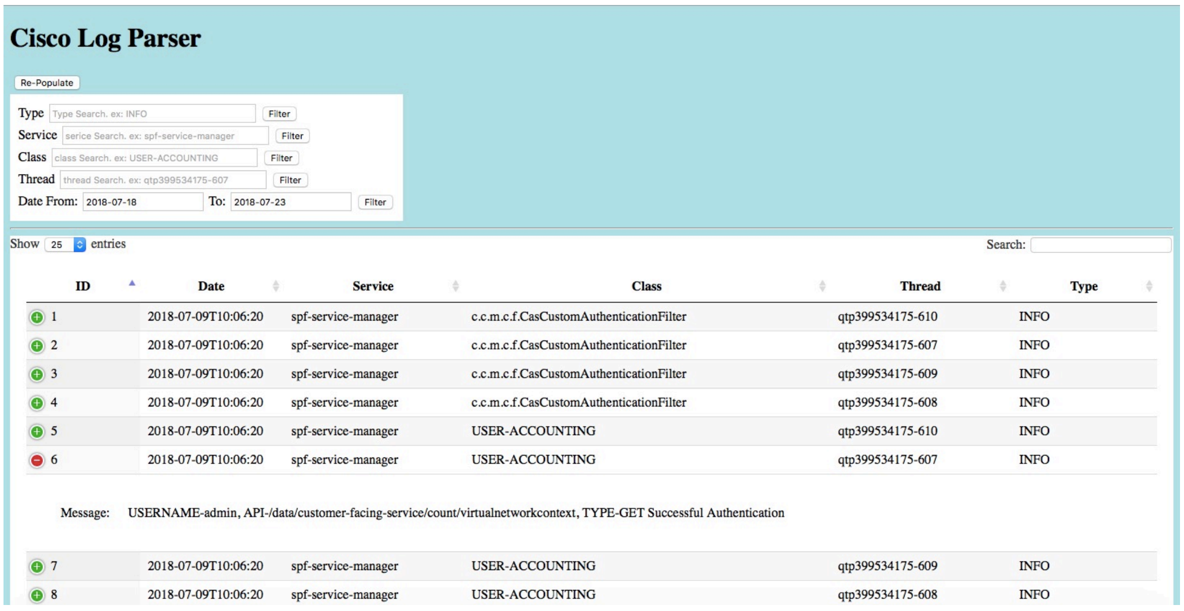
- UI was implemented mainly using the JQuery widget "DataTables" and some filtering feilds made manually using JavaScript.

- Unit testing was implemented using the JUnit testing framework with Mockito for testing the logic of various services in the project.

- Continuous integration was also implemented using Jenkins and Git.

UI

using HTML, CSS, and JS we created a simple website to view and interact with the logs



Cisco Log Parser

Type	Type Search. ex: INFO	Filter
Service	serice Search. ex: spf-service-manager	Filter
Class	class Search. ex: USER-ACCOUNTING	Filter
Thread	thread Search. ex: qtp399534175-607	Filter
Date From: 2018-07-18	To: 2018-07-23	Filter

Show 25 entries   Search:

ID	Date	Service	Class	Thread	Type
1	2018-07-09T10:06:20	spf-service-manager	c.c.m.c.f.CasCustomAuthenticationFilter	qtp399534175-610	INFO
2	2018-07-09T10:06:20	spf-service-manager	c.c.m.c.f.CasCustomAuthenticationFilter	qtp399534175-607	INFO
3	2018-07-09T10:06:20	spf-service-manager	c.c.m.c.f.CasCustomAuthenticationFilter	qtp399534175-609	INFO
4	2018-07-09T10:06:20	spf-service-manager	c.c.m.c.f.CasCustomAuthenticationFilter	qtp399534175-608	INFO
5	2018-07-09T10:06:20	spf-service-manager	USER-ACCOUNTING	qtp399534175-610	INFO
6	2018-07-09T10:06:20	spf-service-manager	USER-ACCOUNTING	qtp399534175-607	INFO
Message:	USERNAME-admin, API-/data/customer-facing-service/count/virtualnetworkcontext, TYPE-GET Successful Authentication				
7	2018-07-09T10:06:20	spf-service-manager	USER-ACCOUNTING	qtp399534175-609	INFO
8	2018-07-09T10:06:20	spf-service-manager	USER-ACCOUNTING	qtp399534175-608	INFO