Project 5 Report

## Task 1

1.How did you use connection pooling?

Because we use Intellij as IDE, it doesn’t contain a default META-INF folder, we add a Tomcat Context Descriptor in the Module Setting and the folder and its corresponding context.xml come out. We did the following three modification to allow connection pooling:

1. Add a <Resource> configuration in <Context> to indicate the url, username, password and so on.
2. Add a <resource-ref> node in web.xml to indicate the resources name.
3. get a connection from the pool in each jdbc operation in dao files.

File name, line numbers as in Github

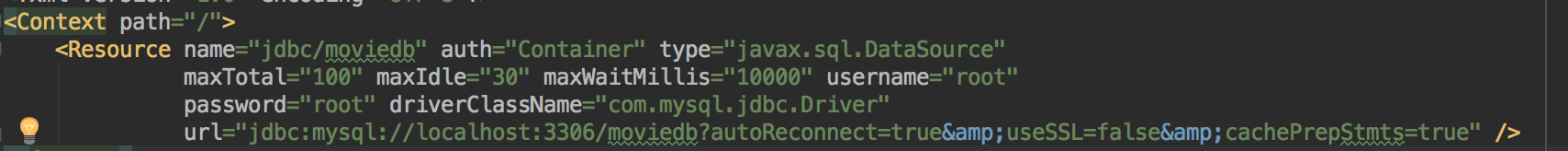
Catsneeze/src/main/webapp/META-INF/context.xml Line 1-11

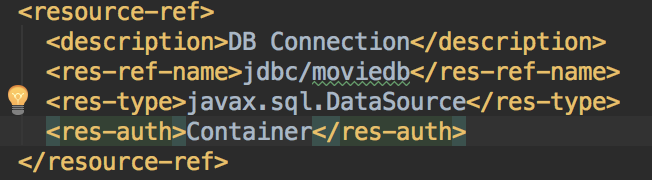
Catsneeze/src/main/webapp/WEB-INF/web.xml Line 11-23

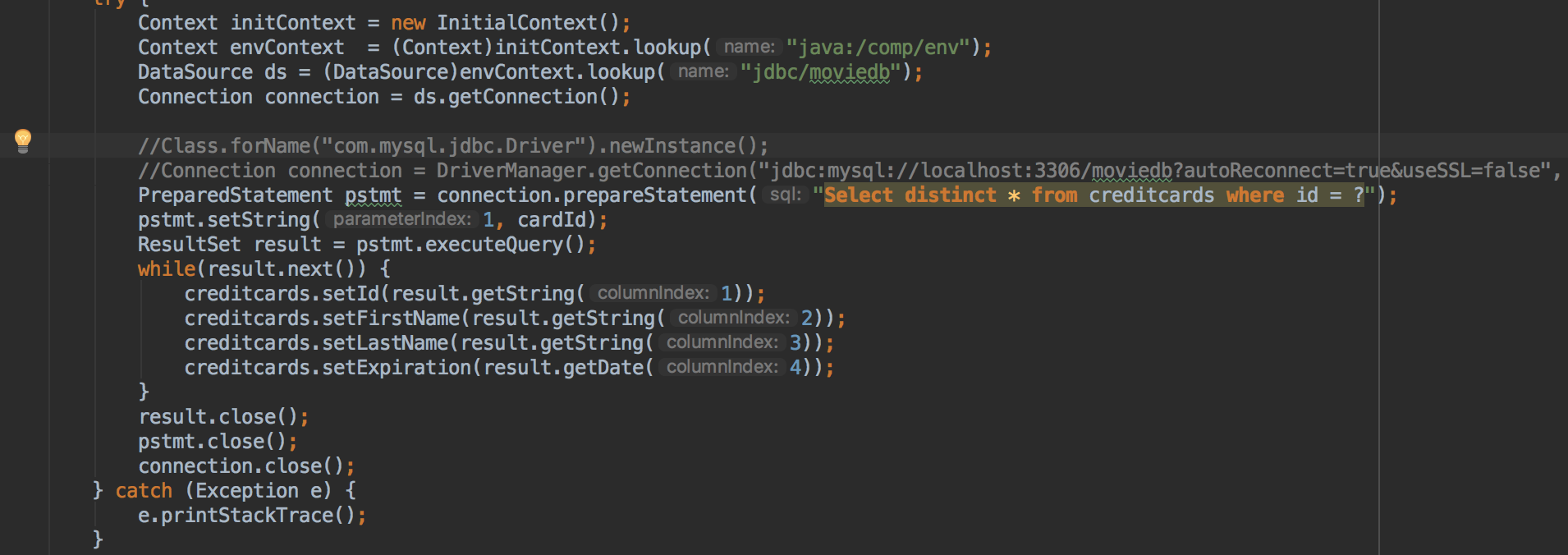
Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/SalesDaoImpl.java Line 30-37

Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/StarsDaoImpl.java Line 131-150

Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/MoviesDaoImpl.java Line 228-231

Snapshots

Adding <Resource> configuration to indicate information

Add <resource-ref> in web.xml to reference the resource.

Using Context.lookup to get data source, and use getConnection factory function to get each connection. Use connection.close() to return the connection to the pool.

2.How did you use Prepared Statements?

We enabled the cache function of prepared statement, and used a ‘?’ style of prepare statement to put variables into sql. In the method, the prepared statement is complied once and cached in memory to speed up the jdbc operation.

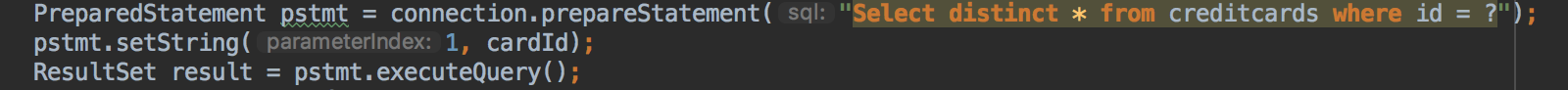
File name, line numbers as in Github

Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/SalesDaoImpl.java Line 37-39

Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/StarsDaoImpl.java Line 138-149

Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/MoviesDaoImpl.java Line 235-268

Snapshots

Enable cache function of prepare statement.

Use “Select distinct \* from creditcards where id = ?” to add variable into sql after creating prepared statement.

## Task 2

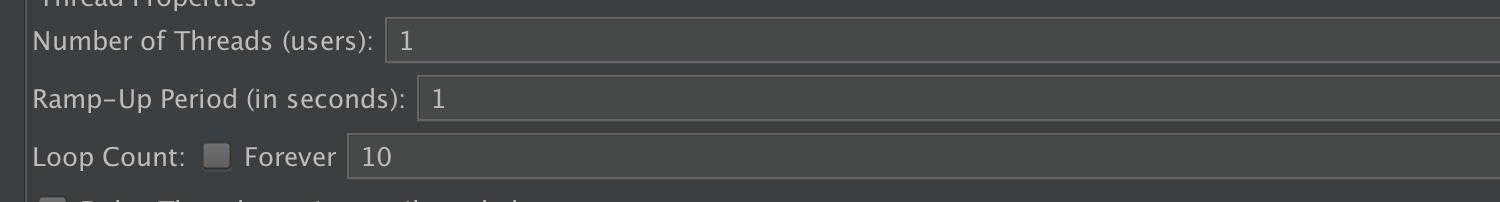
1.Address of AWS and Google instances

Google instance IP: 35.229.25.51, CatSneeze main page is: <http://35.229.25.51/catsneeze/>

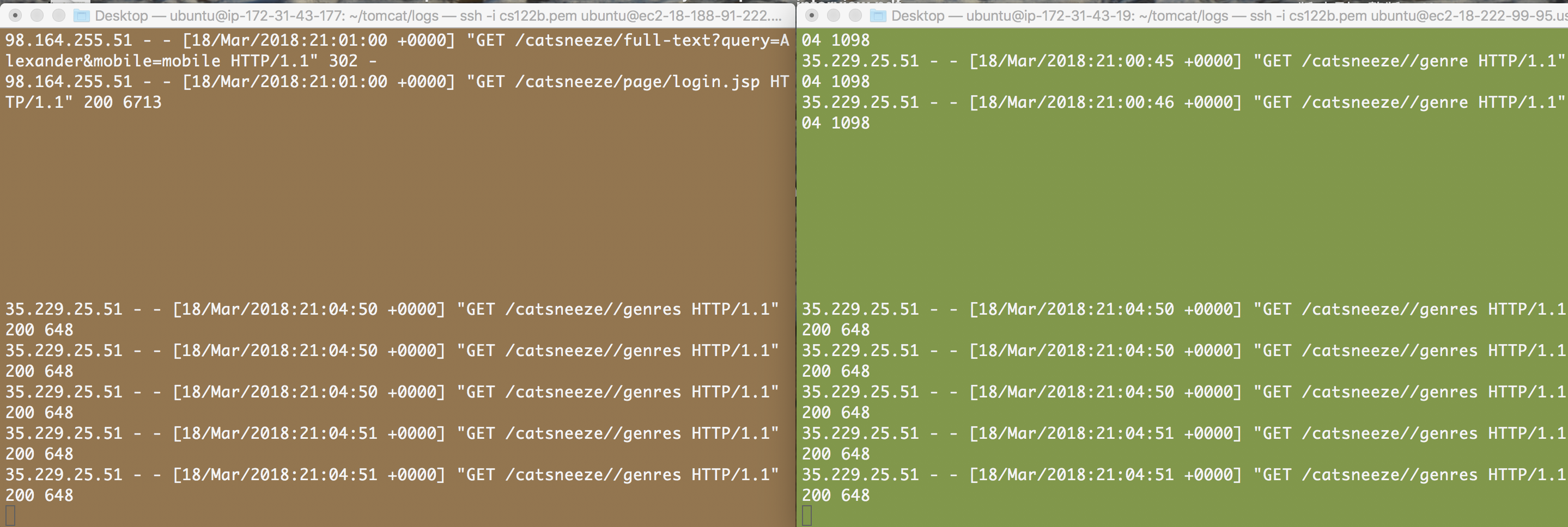
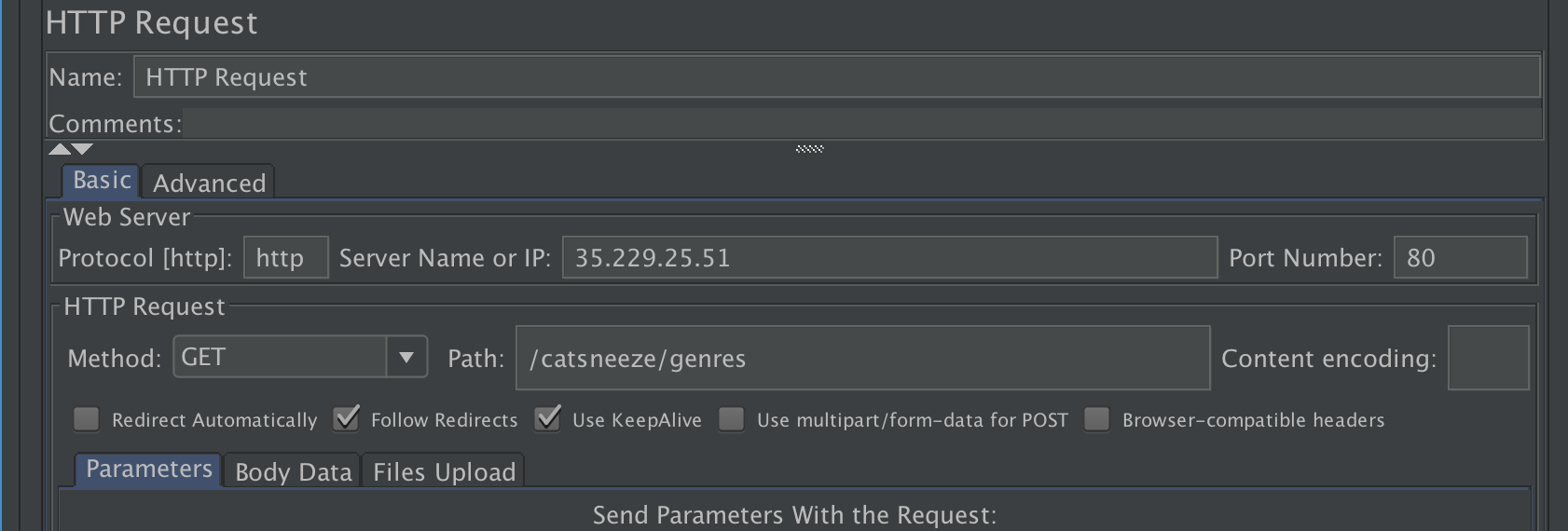
AWS Master IP: 18.188.91.222

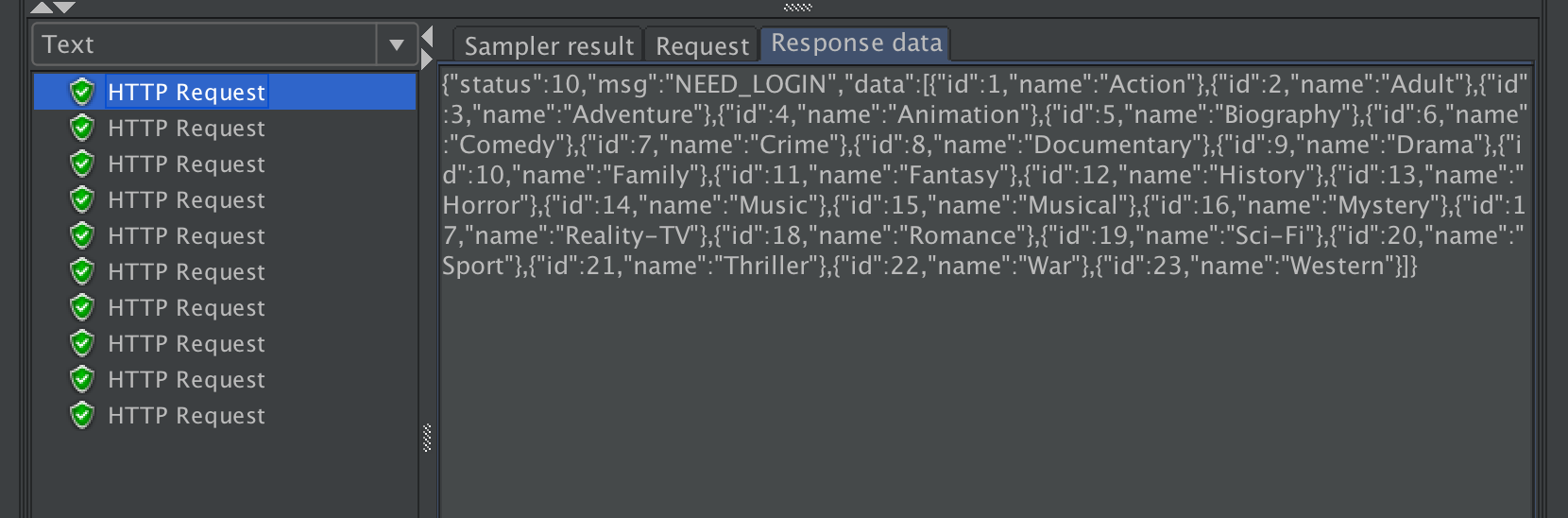
AWS Slave IP: 18.222.99.95

When no login (there’s no session), we use JMeter to request 10 times to a “/genres” service to Load Balancer in Google Cloud.

JMeter configuration:

One thread requests 10 times.

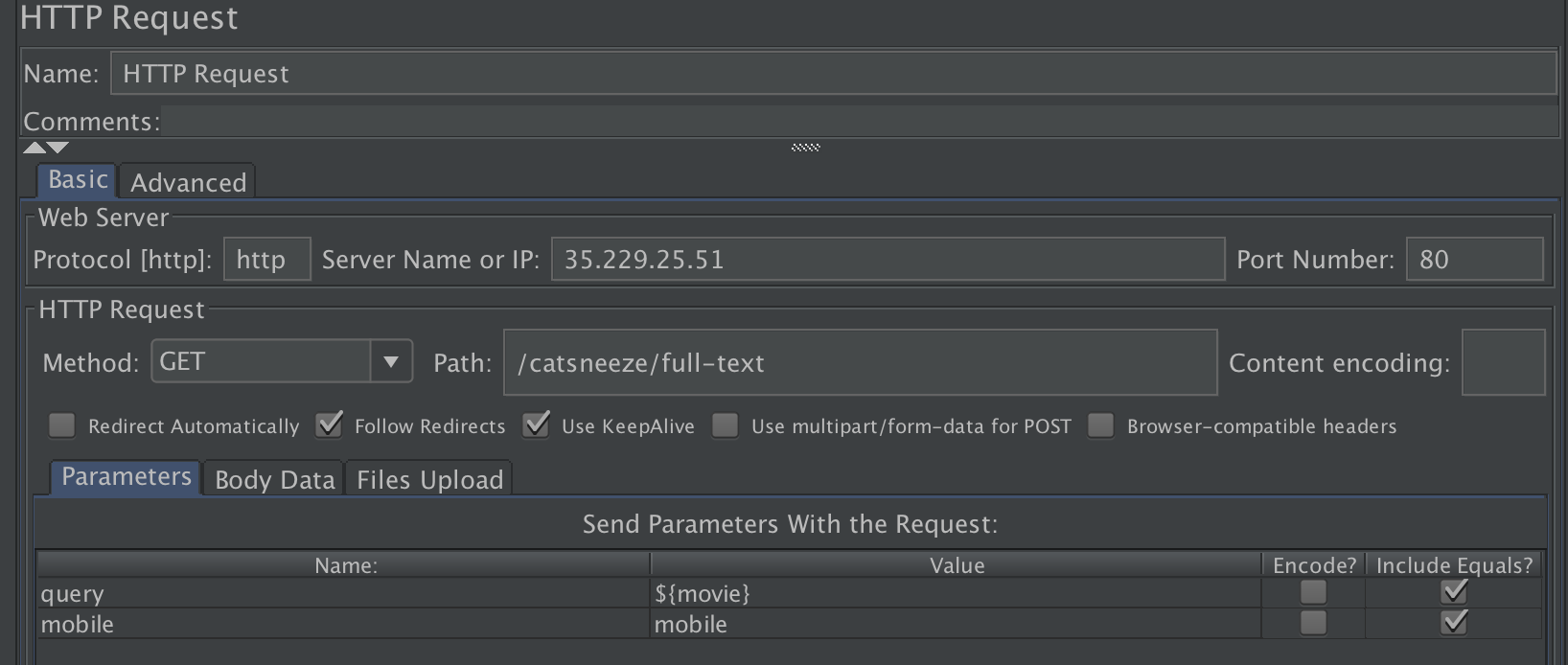
Request configuration.

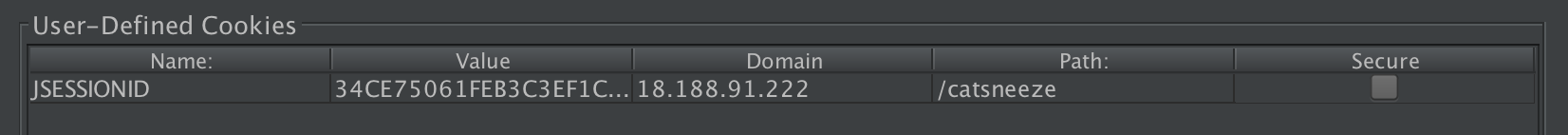
And each instance in AWS got 5 requests. It turns out the round robin way is success.

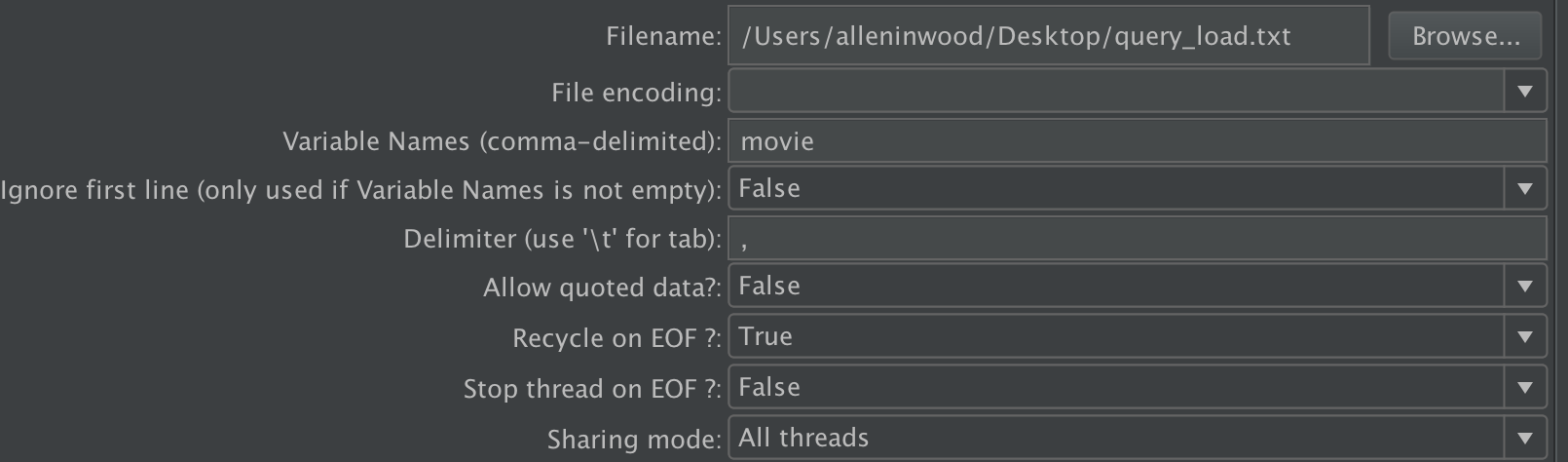
And the response is success and got the corresponding Json information.

When there’s a session attached with master ip, it will use a sticky session characteristic.

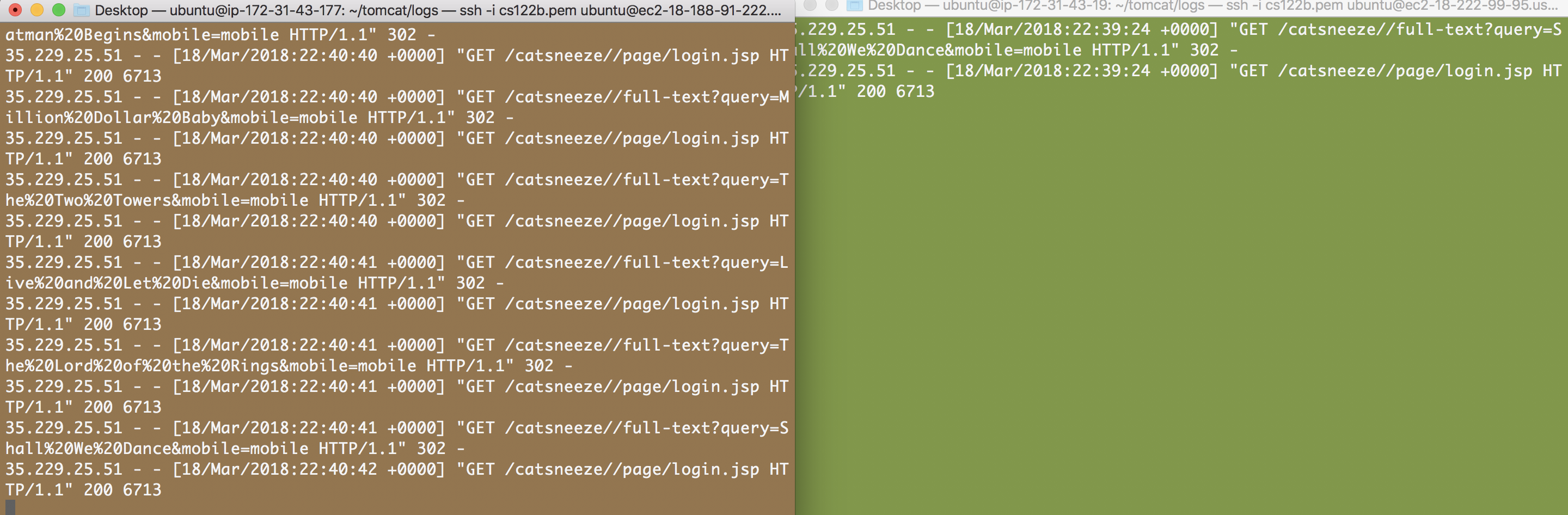
We use the query\_load to request the full-text search service:

The JMeter configuration:

This is the request configuration.

Generated cookie attached to 18.188.91.222 (master cookie)

CSV request attributes.

And the log files in tomcat of two instances are like the following:

Only the master instance got those 10 requests. So the sticky session function is successfully.

2.Have you verified that they are accessible? Does Fablix site get opened both on Google’s 80 port and AWS’ 8080 port?

Load Balancer is accessible to CatSneeze Main Page, but the url should be <http://35.229.25.51/catsneeze/>. Don’t forget the last ‘/’.

Yes, CatSneeze is opened both on Google’s 80 port and AWS’ 8080 port.

3.How connection pooling works with two backend SQL?

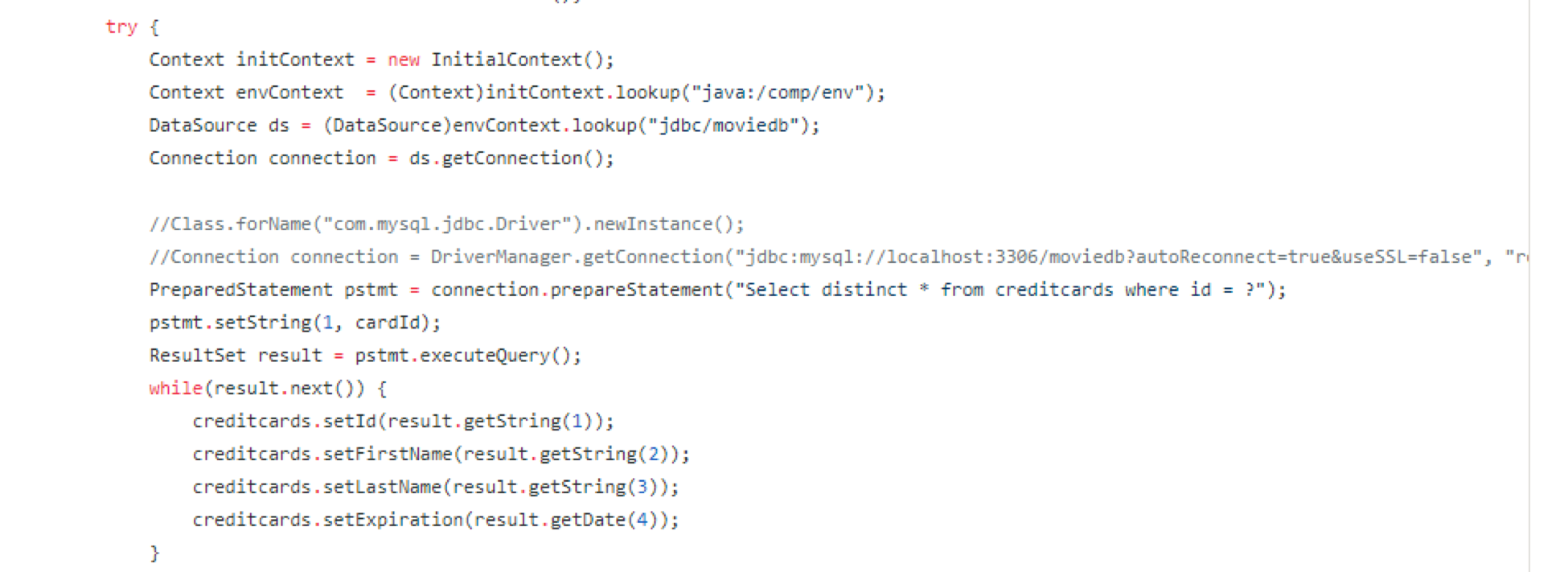
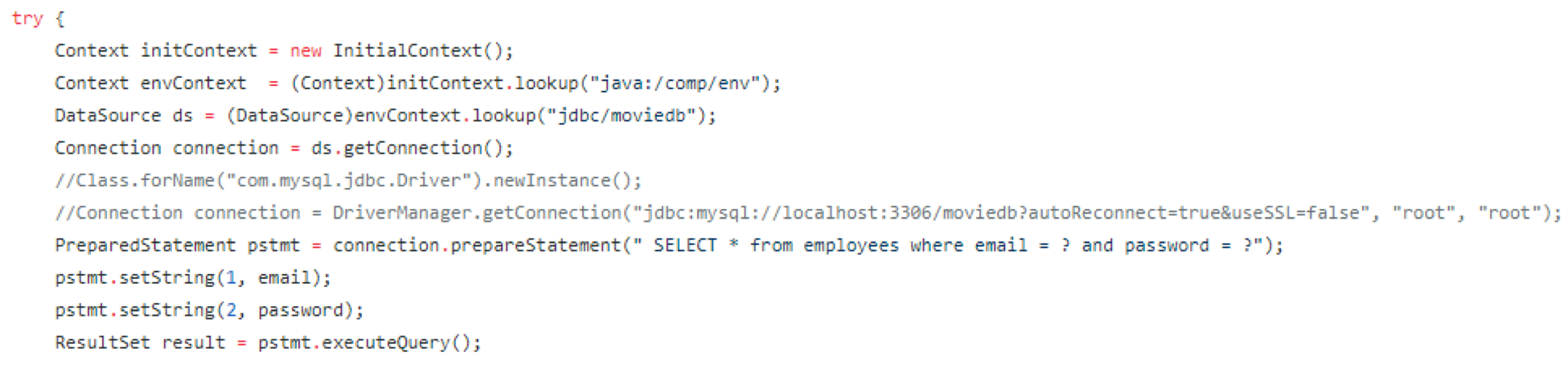
­File name, line numbers as in Github

Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/SalesDaoImpl.java Line 30-37

Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/CardDaoImpl.java Line 22-31

Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/CustomerDaoImpl.java Line 19-25

Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/EymployeeDaoImpl.java Line 15-20

­Snapshots

4.­How read/write requests were routed?

Some read operations were sent to the master while some are routed to the slave

All the write operations are sent to the master.

­File name, line numbers as in Github

Read:

Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/CardDaoImpl.java Line 22-31

Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/CustomerDaoImpl.java Line 19-25

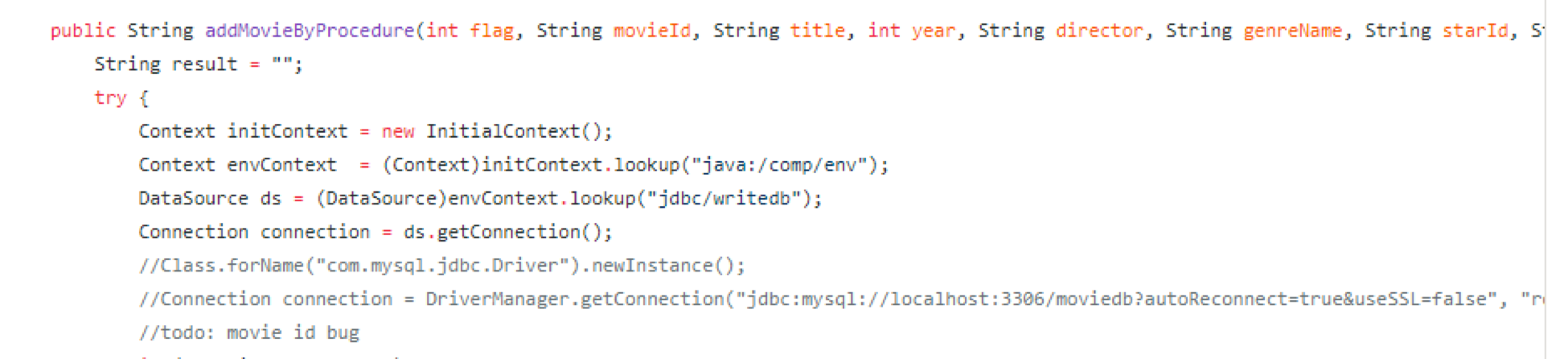
Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/EymployeeDaoImpl.java Line 15-20

Write:

Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/SalesDaoImpl.java Line 30-37

Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/MoviesDaoImpl.java Line 225-235

Catsneeze/src/main/java/com.cs122b.catsneeze/dao/impl/StarsDaoImpl.java Line 128-135

Snapshots



## Task3

1.Have you uploaded the log file to Github? Where is it located?

The Log file logs the TS TJ time in full text search request, and write them into timeLog.txt in WEB-INF folder.

2.Have you uploaded the HTML file to Github? Where is it located?

The HTML is in the Catsneeze/src/main/webapp/jmeter/jmeter\_report.html

3.Have you uploaded the script to Github? Where is it located?

The Python script in WEB-INF folder is to calculate the average TS TJ time in timeLog.txt by using:

Python calAverageTJTS.py

4.Have you uploaded the WAR file and README to Github? Where is it located?

The WAR file and the README file is in the root folder.