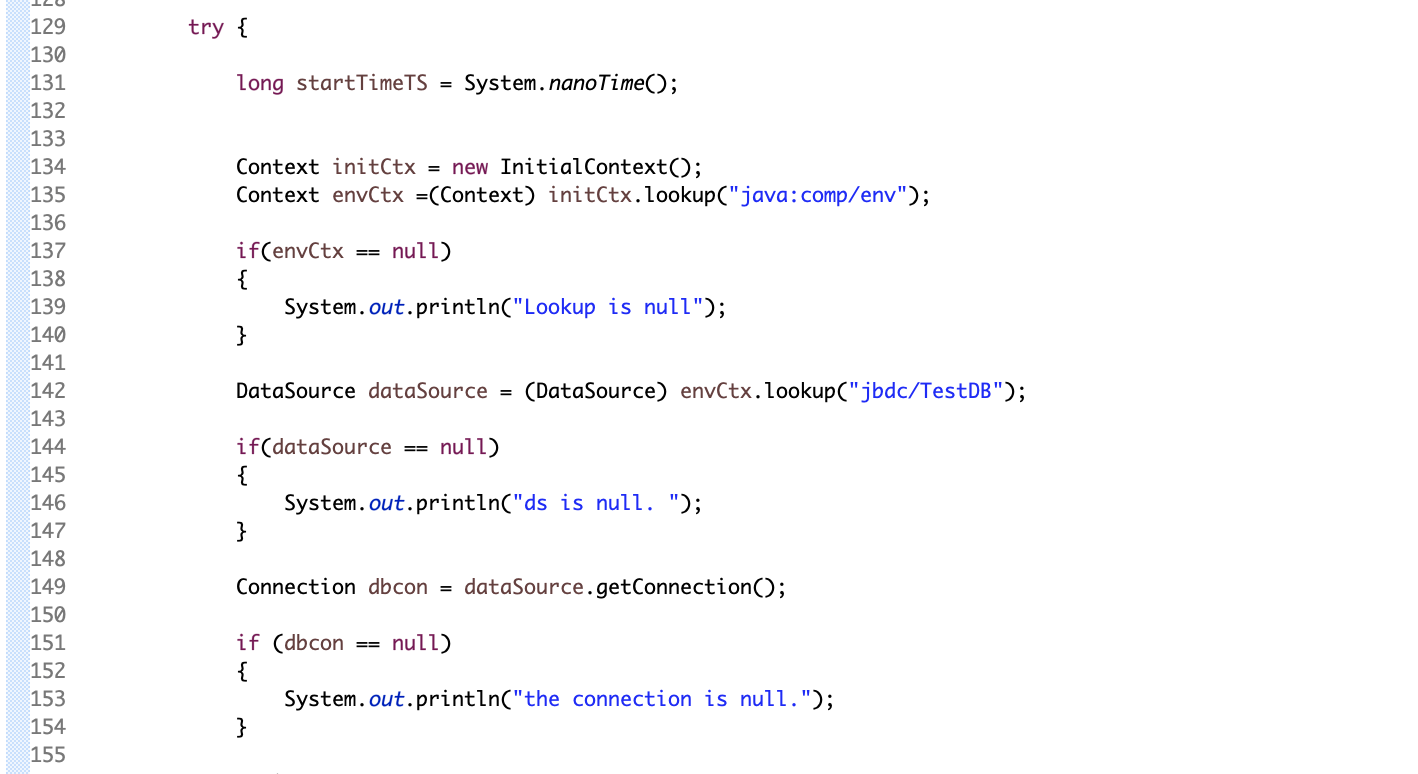
**Task 1**

* How did you use connection pooling?

I used connection pooling in my backend search servlet. There are a few reasons to choose connection pooling as a solution, but the main purpose is in order to have faster connection by granting the ability to not have to use multiple connections. In fact, connection pooling allows the user to recycle connections and therefore speeds up the process and also allows for a smaller amount of connections that need to be opened and closed.

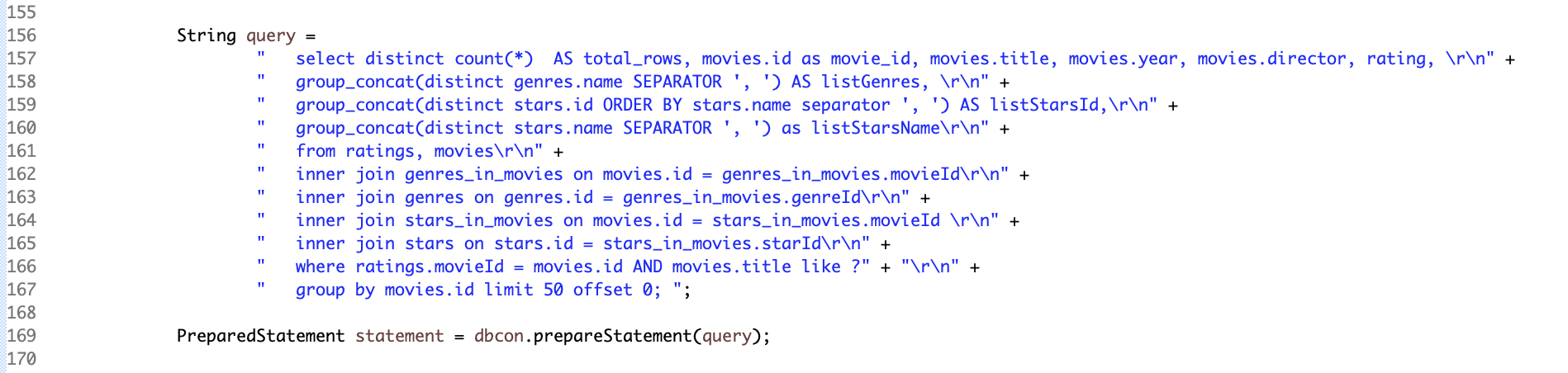
* File name, line numbers as in Github
  + searchResult.java
    - Line number: 129 - 155
* Snapshots showing use in your code



* How did you use Prepared Statements?

In my searchResult servlet I initially coded it so that no prepared statements would be involved. However, prepared statements are required in order to use JMeter for testing. In my code I was able to squeeze in a ‘?’ in the small section of the query that user input is given. In this case, we are accepting title input from the user or from JMeter which is why we use setString to insert into the ‘?’ after the connection has been made.

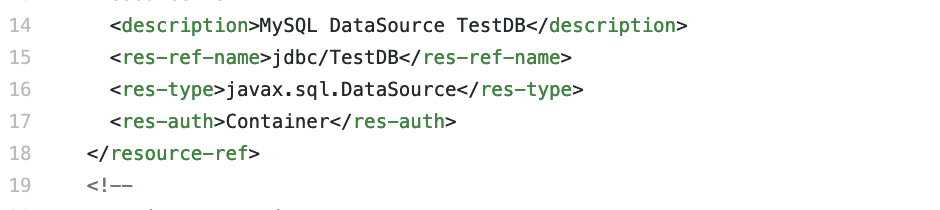
* File name, line numbers as in Github
  + searchResult.java
    - Line number: 156 - 169
* Snapshots showing use in your code



**Task 2**

* Address of AWS and Google instances
  + Google Instace: [*35.243.128.4*](http://35.243.128.4/)
  + Original Instance: 3.18.102.16
  + Master Instance: 3.17.67.202
  + Slave Instance: 3.18.110.167
* Have you verified that they are accessible? Does Fablix site get opened both on Google’s 80 port and AWS’ 8080 port?
  + The fabflix site is accessible through all the AWS 8080 port.
* Explain how connection pooling works with two backend SQL (in your code)?
  + The way that I did connection pooling between two backend instances was I specified in the context.xml which resource the instance is using. The master instance uses its own database for both reading and writing whereas the slave will use its own spare database to read, and then write queries to the master database.
  + File name, line numbers as in Github
    - Context.xml: line 14-17
    - Web.xml: line 14 - 18
  + Snapshots



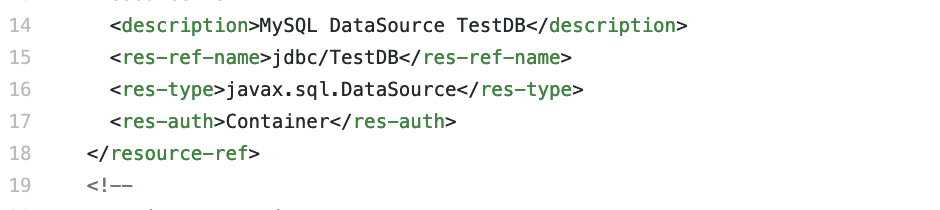


* How read/write requests were routed?

The way that I routed between read and write commands is in the context.xml. A master instance is able to read and write to itself while the slave needs a new resource that allows it to write to the master.

* + File name, line numbers as in Github
    - Context.xml: line 14-17
    - Web.xml: line 14 - 18
* Snapshots





**Task 3**

* Have you uploaded the log files to Github? Where is it located?
  + Single instance Logs: /Users/dennisbui/Desktop/cs122b-winter19-team-125/src/tester
  + Scaled instance Logs:

/Users/dennisbui/Desktop/cs122b-winter19-team-125/src/tester

* Have you uploaded the HTML file (with all sections including analysis, written up) to Github? Where is it located?
  + /Users/dennisbui/Desktop/cs122b-winter19-team-125/WebContent/jmeter\_report.html
* Have you uploaded the script to Github? Where is it located?
  + /Users/dennisbui/Desktop/cs122b-winter19-team-125/src/tester/script.py
* Have you uploaded the WAR file and README to Github? Where is it located?
  + /Users/dennisbui/Desktop/cs122b-winter19-team-125/fabflix.war