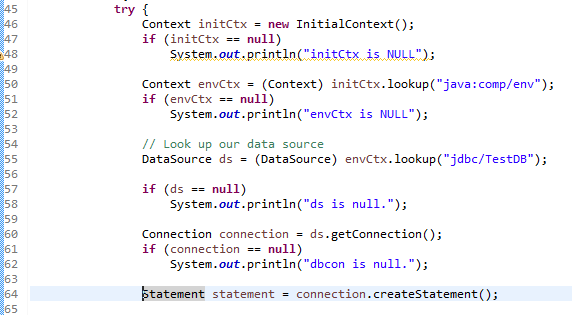
**Task 1**

* How did you use connection pooling?

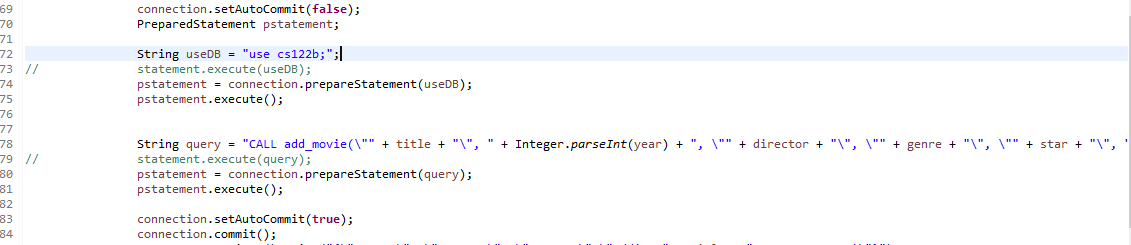
I added JDBC connection pooling to each of my servlets involving queries or transactions so that each future request wouldn’t have to create a completely new connection each time it wanted to query or change the database, and instead uses the existing connections in the pool to improve performance.

* File name, line numbers as in Github
  + AddMovieServlet: 46-62
  + AddStarServlet: 56-72
  + BrowseServlet: 35-51
  + FullTextServlet: 50-66
  + LoginServlet: 60-75
  + MetaDataServlet: 36-52
  + SearchServlet: 75-93
  + StarServlet: 32-48
* Snapshots

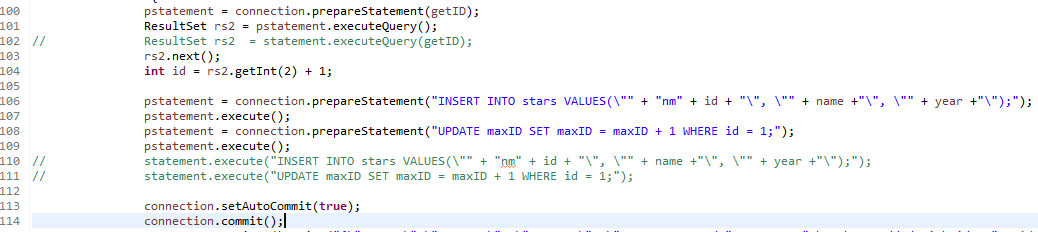


* How did you use Prepared Statements?

For all servlets that would be likely to require large amounts of statements (adding movies, stars, and searching) I replaced all jdbc statements with prepared statements to take advantage of the performance improvement of having precompiled SQL statements. Since I was already custom constructing the SQL statements myself, I didn’t make use of the Prepared Statements’ ability to take parameters to use the same statement while putting different values.

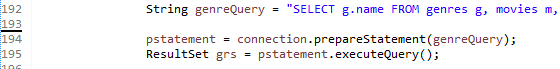
* File name, line numbers as in Github
  + AddMovieServlet:69-85
  + AddStarServlet: 84-94, 100-114
  + SearchServlet: 108-112, 162-163, 195-196, 218-219
* Snapshots

AddMovieServlet:



AddStarServlet:

SearchServlet: (using prepared statements to retrieve genres for each movie result)



**Task 2**

* Address of AWS and Google instances
  + (subject to change, written in excel sheet)
  + AWS: 52.14.51.248
  + Google: 130.211.135.24
* Have you verified that they are accessible? Does Fablix site get opened both on Google’s 80 port and AWS’ 8080 port?

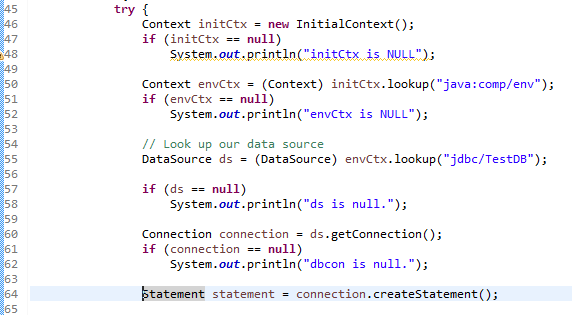
Both accessible at their respective ports

* How connection pooling works with two backend SQL?

Since these two backend SQL servers have their own database, they open their own connections with their database and therefore have their own connection pool, rather than sharing connections to the same database.

* + File name, line numbers as in Github

The code used for connection pooling with the two backend SQL servers is the same as the code in the original instance:

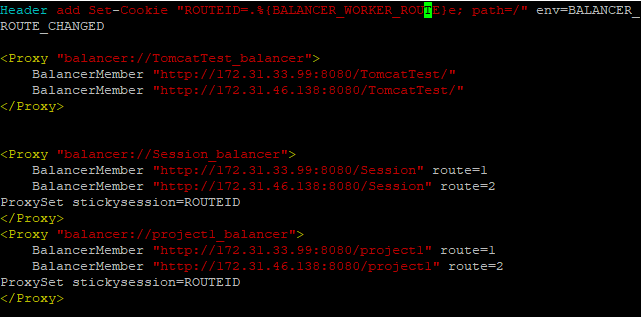
* + AddMovieServlet: 46-62
  + AddStarServlet: 56-72
  + BrowseServlet: 35-51
  + FullTextServlet: 50-66
  + LoginServlet: 60-75
  + MetaDataServlet: 36-52
  + SearchServlet: 75-93
  + StarServlet: 32-48
  + Snapshots
  + 
* How read/write requests were routed?

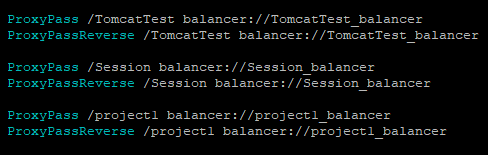
Read/write requests were routed by establishing a master/slave relationship between the two backend SQL servers. By providing one server with the other server’s ip, user username/password, log file, and log number, it becomes a slave to the other server by connecting to it. Once this happens the other server establishes itself as a master, showing a connected slave host. From here, our original instance will send its requests to the master, which will automatically route read either to itself or the slave, and write requests only to itself.

* + File name, line numbers as in Github

/etc/apache2/sites-enabled/000-default.conf: 1-24, 77-84

* Snapshots





**Task 3**

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

Located in Project Logs folder (has Single, and Scaled)

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

Located in the root folder as jmeter\_report.html

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

Located in the Project\_Logs folder as log\_script

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

Located in the root folder as project1.war and README.txt