**Task 1**

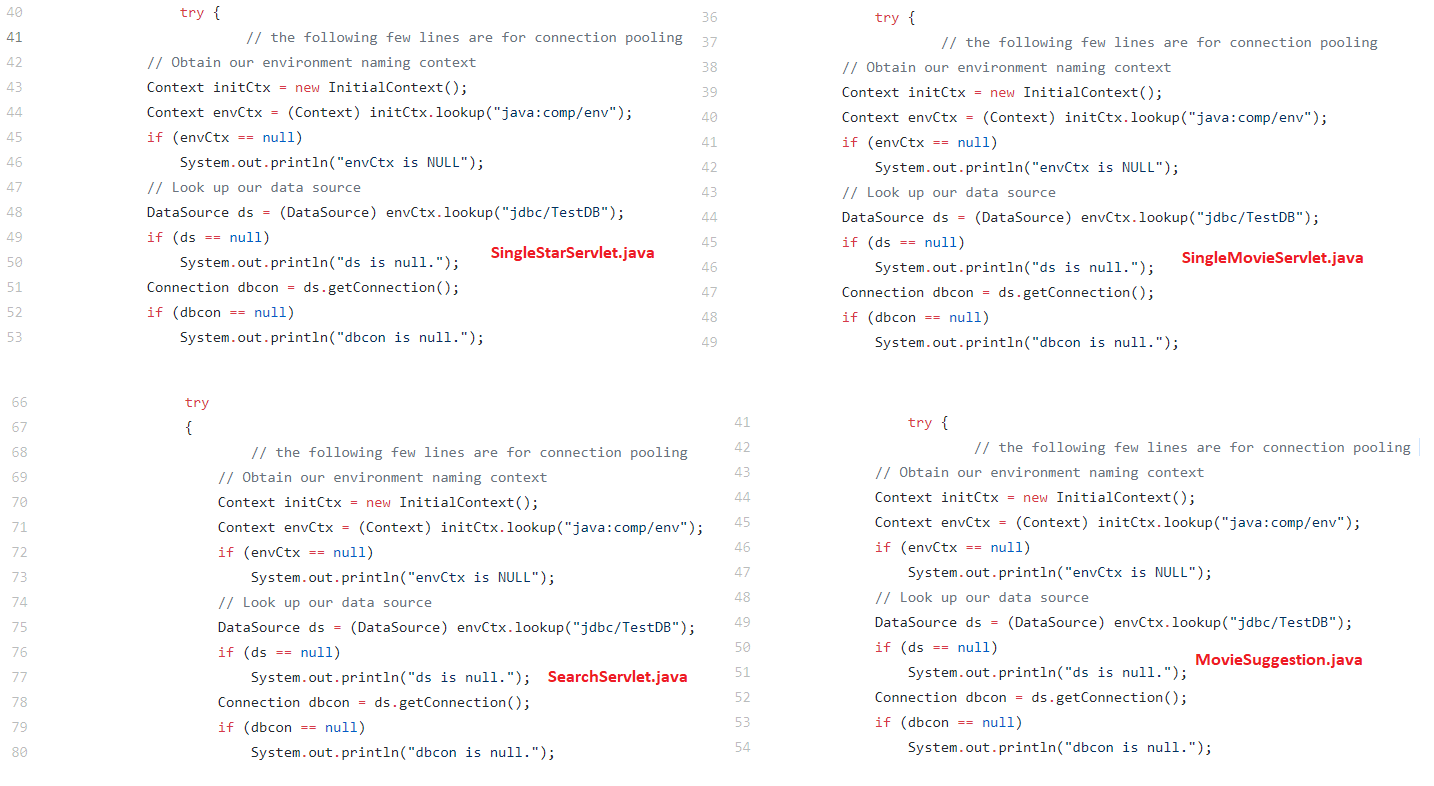
* How did you use connection pooling?

We used connection pooling in our backend servlets. The purpose of connection pooling is to abstract away the user login/password credentials of the backend database. It is dangerous to allow any type of users to have access to the database directly. Thus by using connection pooling it helped solved our security problem. Also another advantage is that certain users who run their server on localhost can easily change all the servlet’s username and password if it differs from their AWS database username and password. The biggest advantage of connection pooling is the ability to recycle connections after it is done being used. Users do not need to open connections every time they want to query the database (which can become very expensive). Thus making the servers response time quicker.

1. File name, line numbers as in Github
   * SingleStarServlet.java
     1. Line Number: 40-53
   * SingleMovieServlet.java
     1. Line Number: 36-49
   * SearchServlet.java
     1. Line Number: 66-80
   * MovieSuggestion.java
     1. Line Number: 41-54
   * MobileSearchServlet.java
     1. Line Number: 36-50
   * LoginServlet.java
     1. Line Number: 49-62

* Snapshots showing use in your code



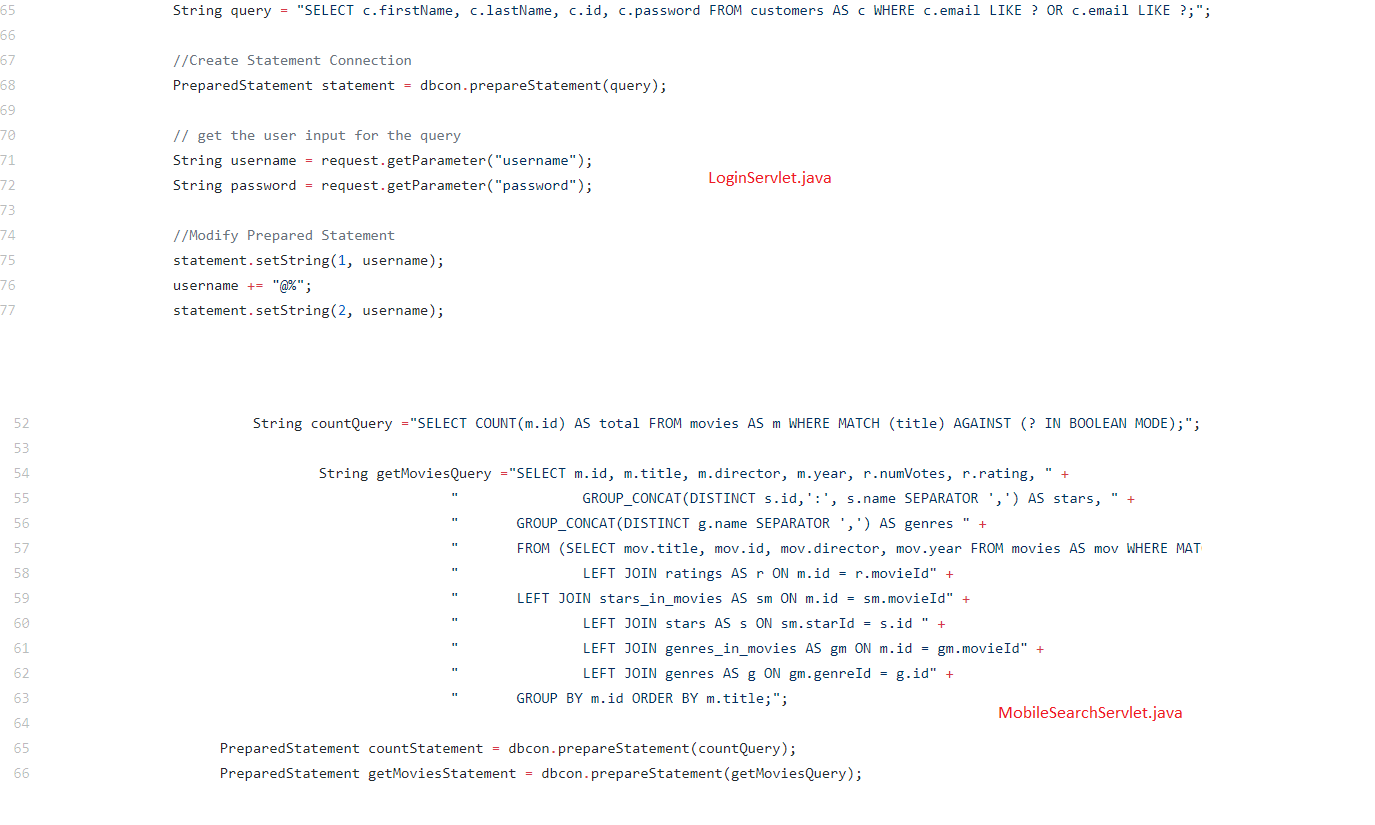


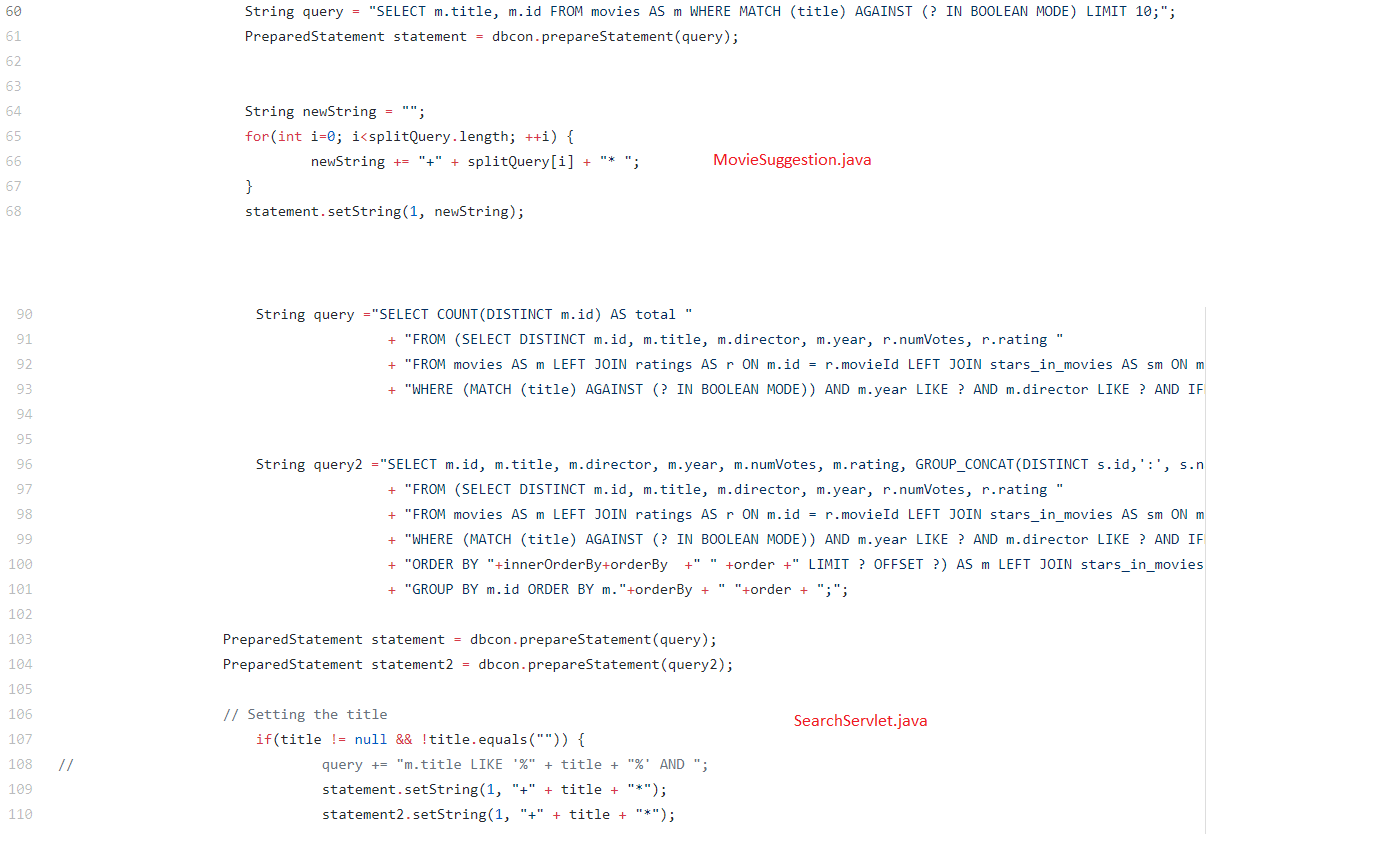
* How did you use Prepared Statements?

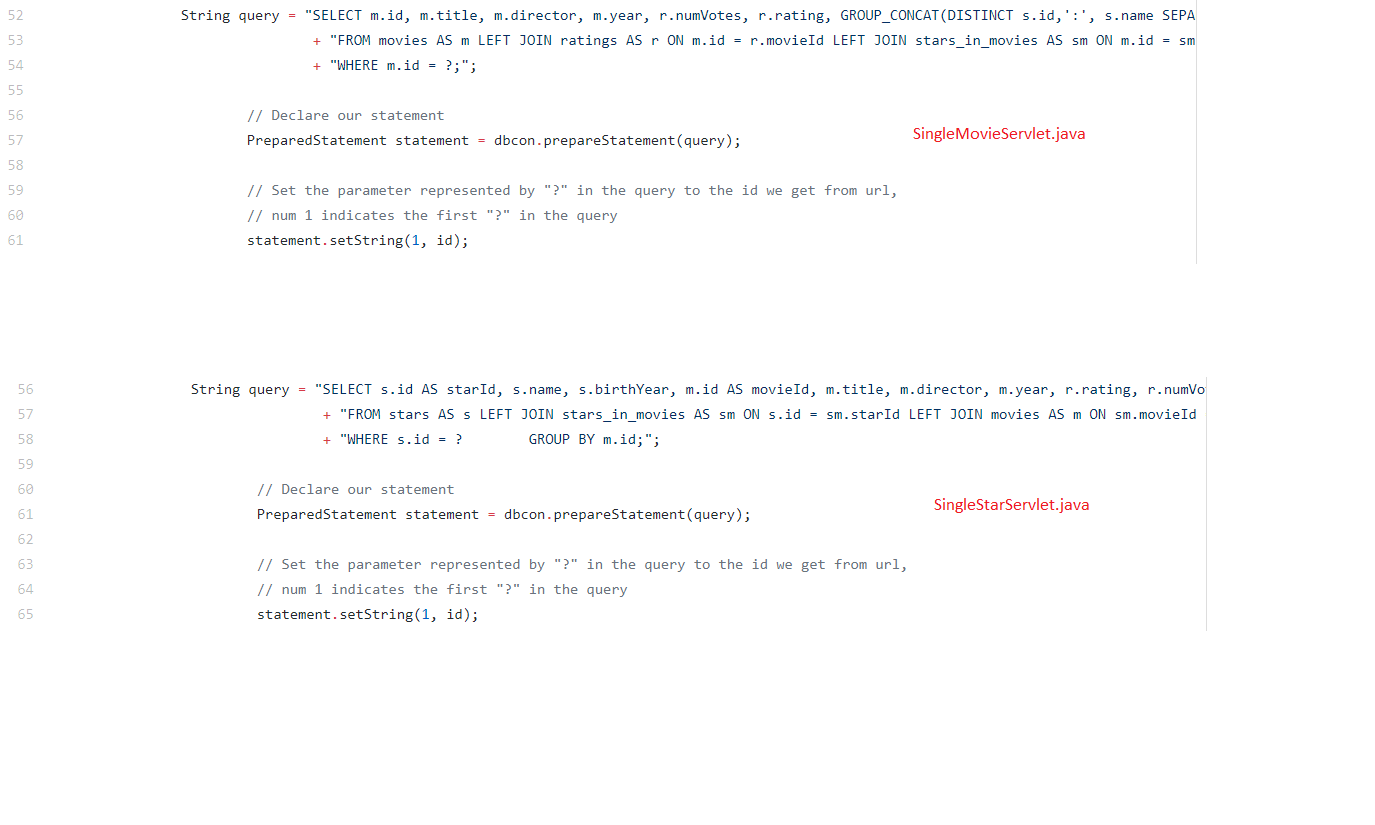
We used prepared statements to query the database based on the users input. This was much easier to use than concatenating strings together to create the query we need to search the database. With prepared statements it was much simpler and we only needed to fill in the index of the (?) to create the query.

1. File name, line numbers as in Github
   * LoginServlet.java
     1. Line Number: 65-72
   * MobileSearchServlet.java
     1. Line Number: 52-66
   * MovieSuggestion.java
     1. Line Number: 60-68
   * SearchServlet.java
     1. Line Number: 90-140
   * SingleMovieServlet.java
     1. Line Number: 52-61
   * SingleStarServlet.java
     1. Line Number: 56-65

* Snapshots showing use in your code

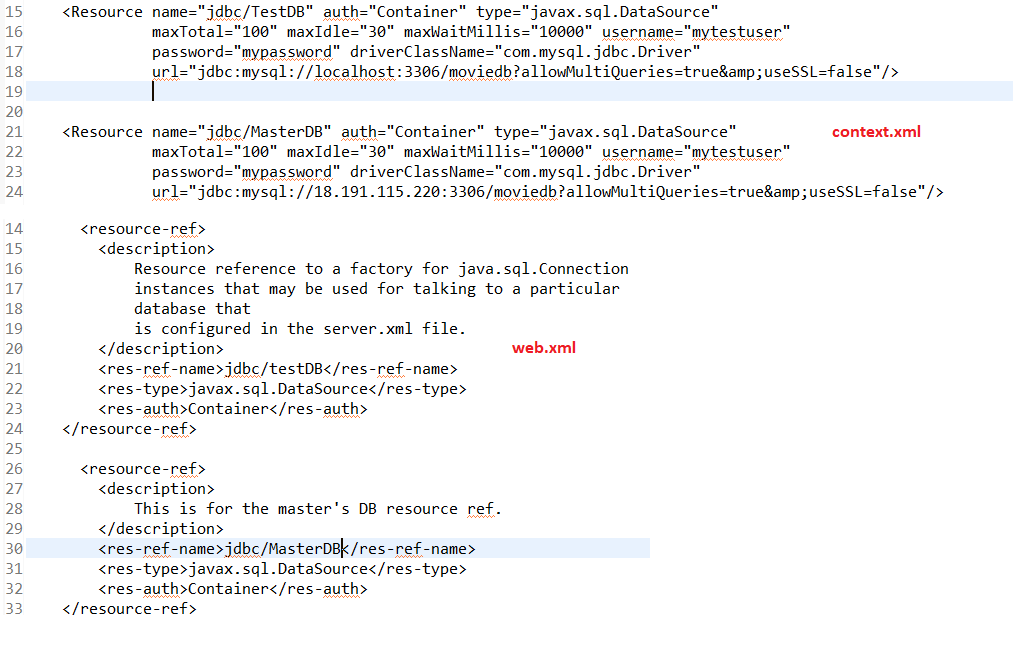




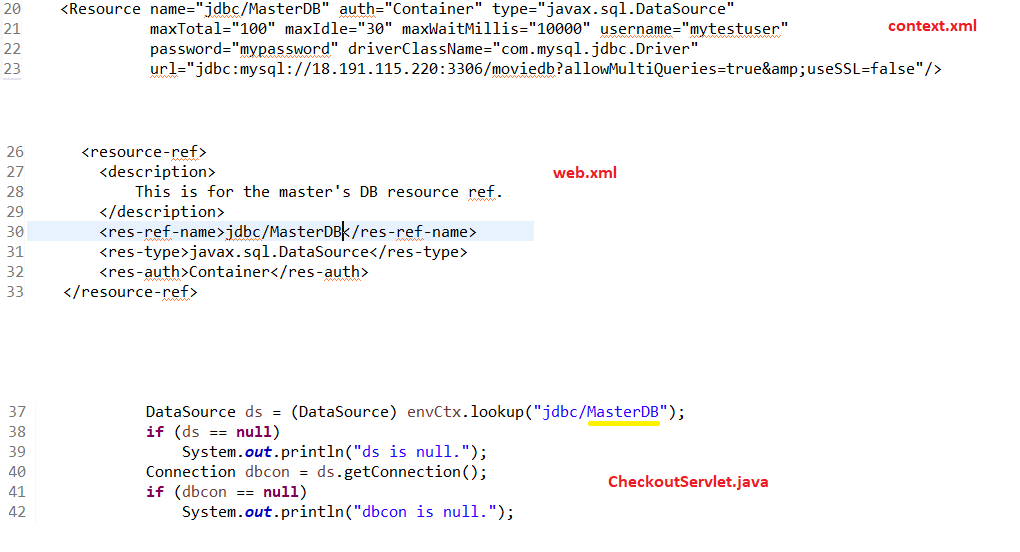


**Task 2**

* Address of AWS and Google instances
  + LOAD BALANCER INSTANCE: 18.217.170.114
  + GOOGLE INSTANCE : 104.154.196.92
  + MASTER INSTANCE: 18.191.115.220
  + SLAVE INSTANCE: 18.221.32.80
* Have you verified that they are accessible? Does Fablix site get opened both on Google’s 80 port and AWS’ 8080 port?
  + Yes, Fabflix is accessible for Google and the Load Balancer (AWS Instance). We added a security group entries that allowed port 80 and port 8080 for the Google’s instance and AWS instance respectively (by allowing their ip’s).
* Explain how connection pooling works with two backend SQL (in your code)?
  + Connection pooling between two backend database works by specifying the context.html to use localhost (their own) database. So if we are in the master instance it would use their own database for writing and reading. For the slave instance it would use its own database to read, and when there are write queries the slave queries the master database.
  + File name, line numbers as in Github
    - Context.xml: lines 20-23
    - Web.xml: lines 26-33
    - **(REFER TO TASK 1 LINES/Snapshots)**
  + Snapshots



* How read/write requests were routed?
  + For our read request we routed the query to any of the database from the Slave or Master it is fine. For writing the Master can write to its own database but it was not the same for the slave. In the slave instance it needed to write to the master’s database as well. We solved this issue by creating a new Data Source that was routed to the master’s database. We only used this in our checkout servlet, it is the only time we need to update the database.
  + File name, line numbers as in Github
    - Context.xml: lines 20-23
    - Web.xml: lines 26-33
    - CheckoutServlet.java: lines 37-42
  + Snapshots



**Task 3**

* Have you uploaded the log file to Github? Where is it located?
* Single Instance Logs located:
  + cs122b-spring18-team-55/project1/WebContent/imgs/single\_case\_imgs/
* Scaled Instance Logs Located:
* Have you uploaded the HTML file to Github? Where is it located?
* HTML File Located:
  + cs122b-spring18-team-55/project1/WebContent/Report.html
* Have you uploaded the script to Github? Where is it located?
* Script Uploaded at:
  + cs122b-spring18-team-55/project1/src/SimpleAverageScript.java
* Have you uploaded the WAR file and README to Github? Where is it located?