**Task 1**

* **How did you use connection pooling?**

We first download the connector file (JDBC Driver jar) and put it into the $CATALINA\_HOME/lib. Then, we create a file called context xml which contains the needed database information like the name of the database, the driverClassName, the username, the password and the URL of the MySQL to be the “connection pool,” and use the way that the professor gave us in the example to connect to this database in all servlets. (look up the name of the database that written in the xml file, and then connect it)

**In project 2:**

* **File name, line numbers as in Github**
* 1. BrowsingServlet.java (119-136)
* **2. CheckoutServlet.java(54-71)**
* 3. EmployeeServlet.java(47-60)
* 4. InsertMovieServlet.java(67-80)(303-320)
* 5. InsertStarServlet.java(61-74)
* 6. MetaDataServlet.java(105-118)
* 7. MovieSuggestion.java(65-78)
* 8. SaxActor.java(73-86)
* 9. SaxHandler.java(46-59)
* 10. SearchServlet.java(270-283)
* 11. ShoppingServlet.java(150-163)
* 12. SingleMServelet.java(162-175)
* 13. SingleSServelet.java(147-160)
* **Snapshots showing use in your code**

1.

A screenshot of a social media post

Description automatically generated

2.

**A screenshot of a social media post

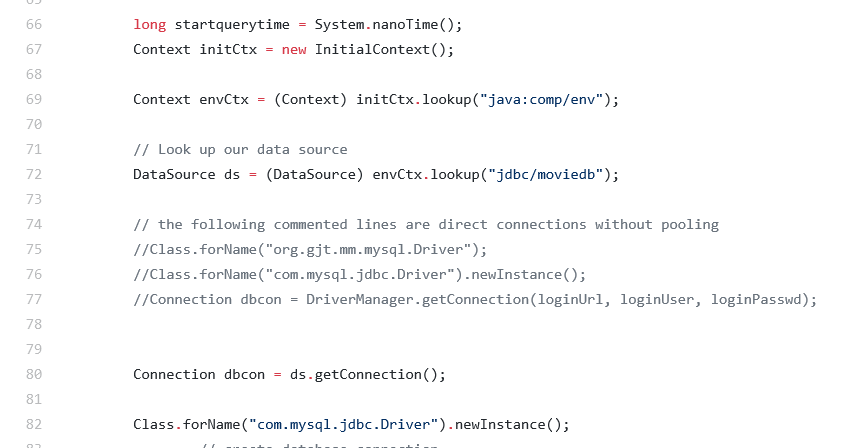
Description automatically generated**

3.

A screenshot of a cell phone

Description automatically generated

4.



A screenshot of a social media post

Description automatically generated

5.



6.

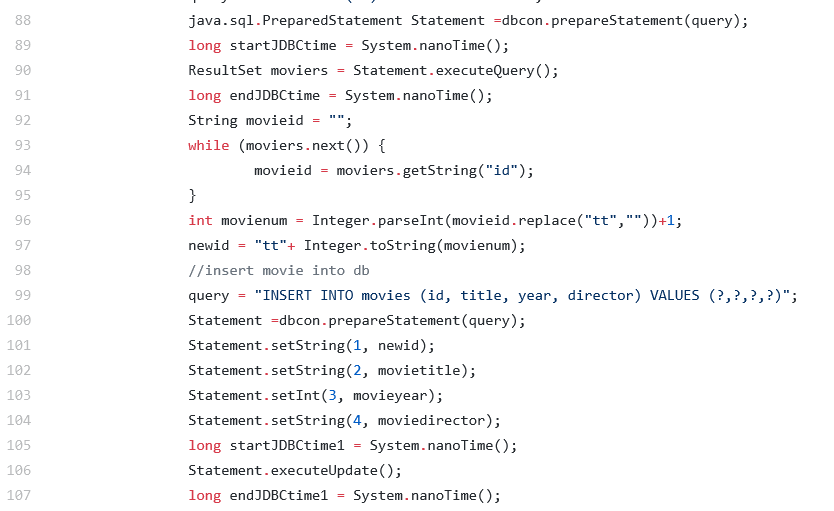
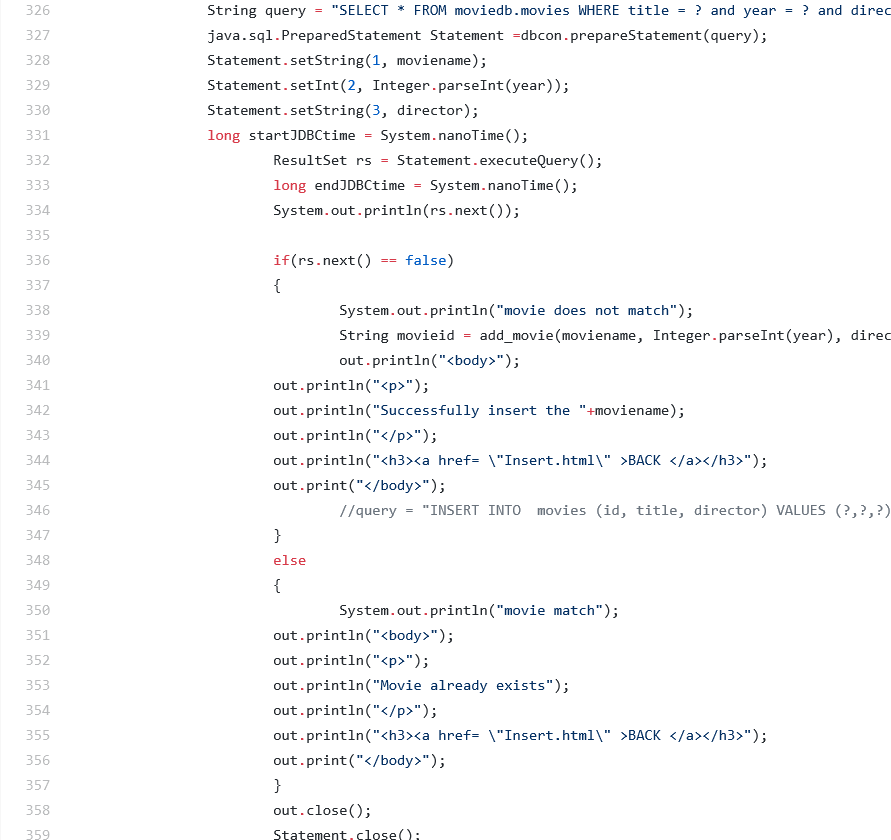
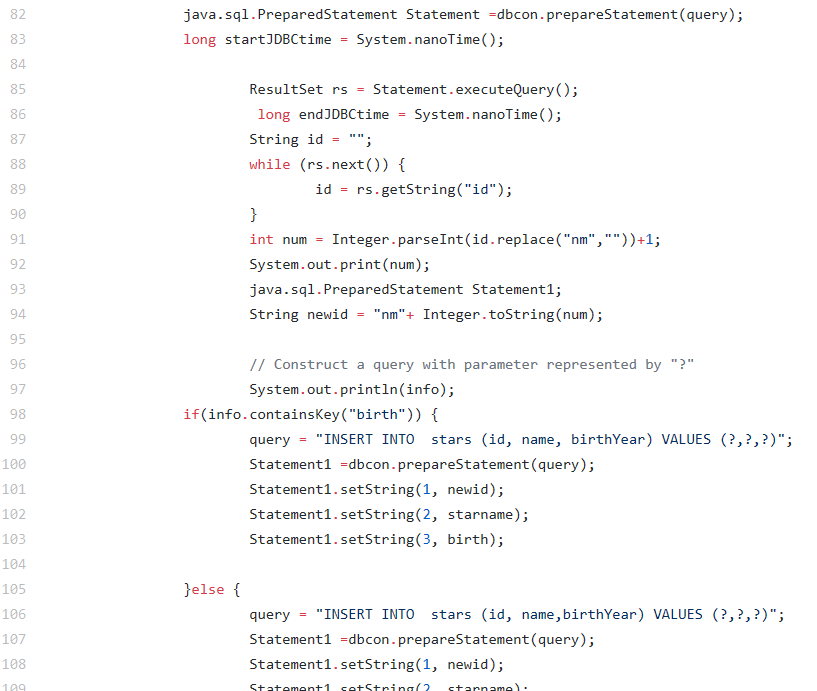
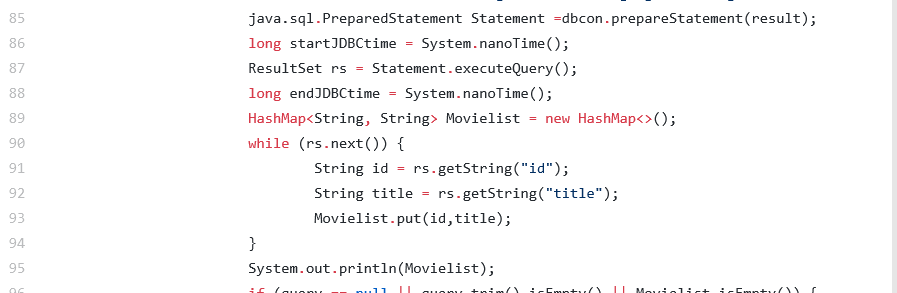
A screenshot of a social media post

Description automatically generated

* **How did you use Prepared Statements?**

As the tutorial said, we first create the java.sql.PreparedStatement to create the prepared statement object. Then, we create the ResultSet object which is eaqul to the statement.executeQuery() to get the result. After that, we use the while loop to get the result.

**In project 2:**

* **File name, line numbers as in Github**
* 1. BrowsingServlet.java (148-176)
* **2. CheckoutServlet.java(81-105)**
* 3. EmployeeServlet.java(66-85)
* **4. InsertMovieServlet.java(87-233)(326-360)**
* 5. InsertStarServlet.java(81-130)
* 6. **MovieSuggestion.java(84-94)**
* 7. SaxActor.java(91-164)
* 8. SaxHandler.java(63-114)(203-236)
* 9. SearchServlet.java(374-390)
* 10. ShoppingServlet.java(178-190)
* 11. SingleMServelet.java(199-260)
* 12. SingleSServelet.java(170-228)
* **Snapshots showing use in your code**
* **1.**
* 
* **2.**
* 
* 3. 
* 4. (part)
* 
* 
* 5.(Part)
* 
* 6. 

**Task 2**

* **Address of AWS and Google instances**
* Address of AWS: <3.18.108.153>

- Address of Google instance: 35.235.64.233

* **Have you verified that they are accessible? Does Fablix site get opened both on Google’s 80 port and AWS’ 8080 port?**
* Yes, they are accessible. Fablix site can be opened on both on Google’s 80 port and AWS’ 8080 port.
* **Explain how connection pooling works with two backend SQL (in your code)?**

When request for a SQL is made, depending on if it is read or write, either the master or the slave SQL will be pull out of the pooling to complete the request. When it is finished, it will return to the pool for reuse.

* + **File name, line numbers as in Github**

**context.xml(1-19)**

* + **Snapshots**



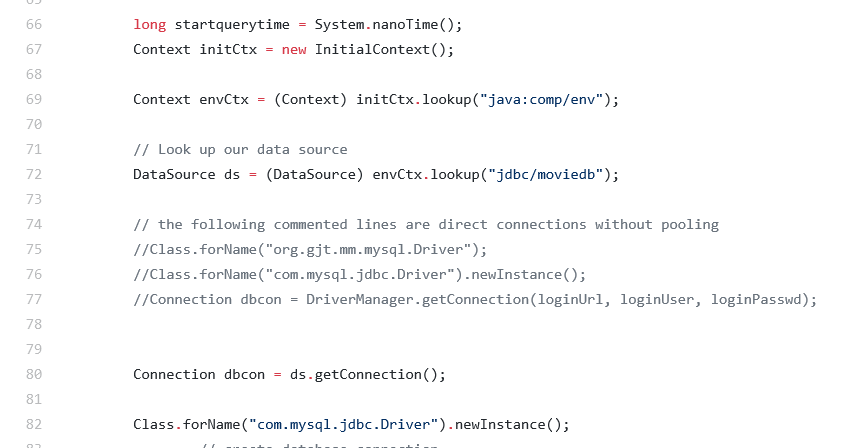
* **How read/write requests were routed?**
* Usually the MySQL will go through the slave SQL to read the file, but since the slave SQL can’t insert, so for the insert part, we use the master SQL instead. To make sure they can work separately, we look up different names while using connection pooling. For example, we look up the “jdbc/TestDB” for the BrowingServlet, and look up the “jdbc/moviedb” for InsertMovieServlet.
  + **File name, line numbers as in Github**
  + context.xml(1-19)
  + 1. BrowsingServlet.java (119-136)
  + **2. CheckoutServlet.java(54-71)**
  + 3. EmployeeServlet.java(47-60)
  + 4. InsertMovieServlet.java(67-80)(303-320)
  + 5. InsertStarServlet.java(61-74)
  + 6. MetaDataServlet.java(105-118)
  + 7. MovieSuggestion.java(65-78)
  + 8. SaxActor.java(73-86)
  + 9. SaxHandler.java(46-59)
  + 10. SearchServlet.java(270-283)
  + 11. ShoppingServlet.java(150-163)
  + 12. SingleMServelet.java(162-175)
  + 13. SingleSServelet.java(147-160)

**Snapshots:**

* 1.
* A screenshot of a social media post

  Description automatically generated
* 2.
* **A screenshot of a social media post

  Description automatically generated**
* 3.
* A screenshot of a cell phone

  Description automatically generated
* 4.
* 
* A screenshot of a social media post

  Description automatically generated
* 5.
* 
* 6.
* A screenshot of a social media post

  Description automatically generated

**Task 3**

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

Yes, the log file is located at the cs122b-winter19-team-78/Desktop/project\_2/WebContent/log.txt.

* **Have you uploaded the HTML file (with all sections including analysis, written up) to Github? Where is it located?**

Yes, the HTML file is located at cs122b-winter19-team-78/Desktop/project\_2/WebContent/jmeter\_report.html.

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

Yes, the script is located at the cs122b-winter19-team-78/Desktop/project\_2/WebContent/calculate.py.

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

Yes, the WAR file is located at the cs122b-winter19-team-78/Desktop/project\_2/WebContent/project\_2.jar

* and README is located at cs122b-winter19-team-78/Desktop.