Project 5 Task 1: Explanation

**When you are done,** write a brief explanation of how/where (file path and line numbers as seen on Github, and snapshots of usage in code) you use the JDBC connection pooling and prepared statements in your code. You should submit this report to Github.

I was able to enable JDBC connection pooling by:

1) Creating a initial Context Object) ‘Context’ initCtx = new InitialContext();

2) Lookup connection in the connection Pool) Context envCtx = (Context)initCtx.lookup("java:comp/env"); - a Context will look at all the pool connections.

3) Look up our Data Source) DataSource dataSource = (DataSource) envCtx.lookup(“jdbc/TestDB”); - This statement will create a dataSource to connect to our mysql database.

Below are two tables where it explains how and where I used JDBC connection pooling and prepared statements in my code.

JDBC Connection Pooling Explanation: - I only included 1 explanation and 1 snapshot because they are all basically a repetition of the same task and will look/have the same purpose.

|  |  |  |  |
| --- | --- | --- | --- |
| File Path: | Line Number | How | Snapshot of Usage |
| /AddMovieServlet.java | 52-59 | 1.Created Context Object  2.used Context object to do a lookup in the connection pool.  3.used DataSource to get connection and execute queries. |  |
| /AndroidLoginServlet.java | 56-76 |  |  |
| /CheckoutServlet..java | 62-84 |  |  |
| /EmployeeLoginServlet.java | 47-67 |  |  |
| /GenreSearchServlet.java | 49-67 |  |  |
| /InsertStarServlet.java | 52-59 |  |  |
| /LoginServlet.java | 46-64 |  |  |
| /MovieSuggestion.java | 43-57 |  |  |
| /Results.java | 44-57 |  |  |
| /SearchServlet.java | 43-67 |  |  |
| /SingleMovieServlet.java | 40-58 |  |  |
| /SingleStarServlet.java | 40-55 |  |  |
| /TitleSearchServlet.java | 49-67 |  |  |

Prepared Statement Explanation:

1. I stored the query in a string variable. Any place that requires user input will be replaced with a ‘?’
2. I created a ‘PreparedStatement’ object and initialized it a Conection Object where I use my string query as a argument. The Connection object will connect my servlet to my mysql instance.
3. I set the value of any ‘?’ in my PreparedStatement with the statement ‘statement.set\*’.
4. I execute the query and store the result into a ‘ResultSet’ object.

|  |  |  |  |
| --- | --- | --- | --- |
| File Path: | Line Number | How | Snapshot of Usage |
| /AddMovieServlet.java | 102-107 | I used a prepare statement to execute a query to add a movie to the database. I used a prepare statement to improve performance and increase security for prevention against mysql injection attacks. |  |
| /GenreSearchServlet.java | 85-102 |  |  |
| /MovieSuggestion.java | 84-105 |  |  |
| /SearchServlet.java | 141-158 |  |  |
| /SingleMovieServlet | 61-73 |  |  |
| /SingleStarServlet.java | 66-73 |  |  |
| /TitleSearchServlet.java | 85-102 |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Task 2**

* Address of AWS and Google instances

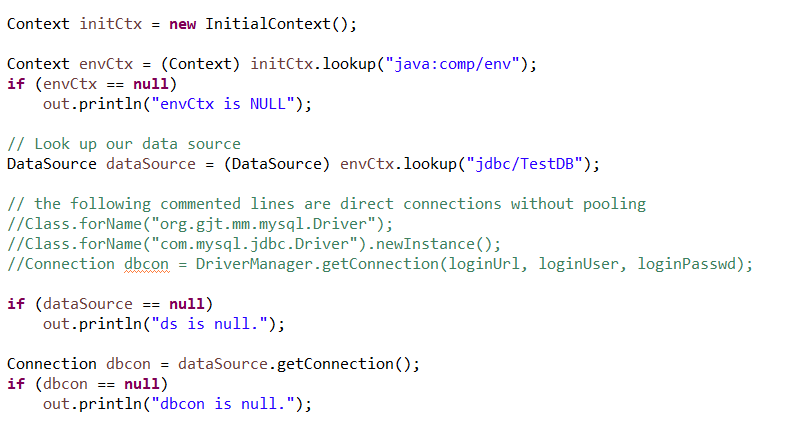
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Google ip | AWS 1 ip | AWS 2 IP | AWS 3 IP | Project url |
| 104.198.102.203 | 13.57.250.62 | 54.183.189.126 | 54.183.219.83 | http://IP:port/project1/ |

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

The Fabflix website successfully loads on both the Google 80 port and AWS 8080 port. When accessing the Google 80 port, the user is sent to either AWS 2 or AWS 3 on port 8080 with Apache load balancing.

* Explain how connection pooling works with two backend SQL (in your code)?

Since the master and slave SQL databases are both synchronized, whenever a new connection is requested, the connection pool cache is checked before establishing a new connection to either the master or slave database. This was done by using a context object to lookup a connection in the connection pool cache. (lines 52-59)



* How read/write requests were routed?

The architecture was setup so that the master database could be both read and written to, while the slave could only be read from. This was done by their respective tomcat instance (2 and 3). In the Fabflix project, the context.xml file was modified (lines 14-17) to allow a connection through a testDB which would route read/write requests through the master database, which is synchronized with the slave. This synchronization was done on each of the instances by connecting the master and slave MySQL database with a replication user and master log file with a given starting position.



**TASK 3**

The C script to calculate the average times is in:

cs122b-projects/project1/calcAverage.c

The log files are in: (same dir as this report)

*\*\*I did not log the cases removing prepared statements or pooling.\*\**

cs122b-projects/scaled\_1thread

cs122b-projects/scaled\_10thread

cs122b-projects/single\_1thread

cs122b-projects/ single\_10thread

cs122b-projects/single\_https10thread

The HTML file is in:

cs122b-projects/jmeter\_report.html (same dir as this report)

WAR and readme is in:

cs122b-projects/project1.war

cs122b-projects/README.md