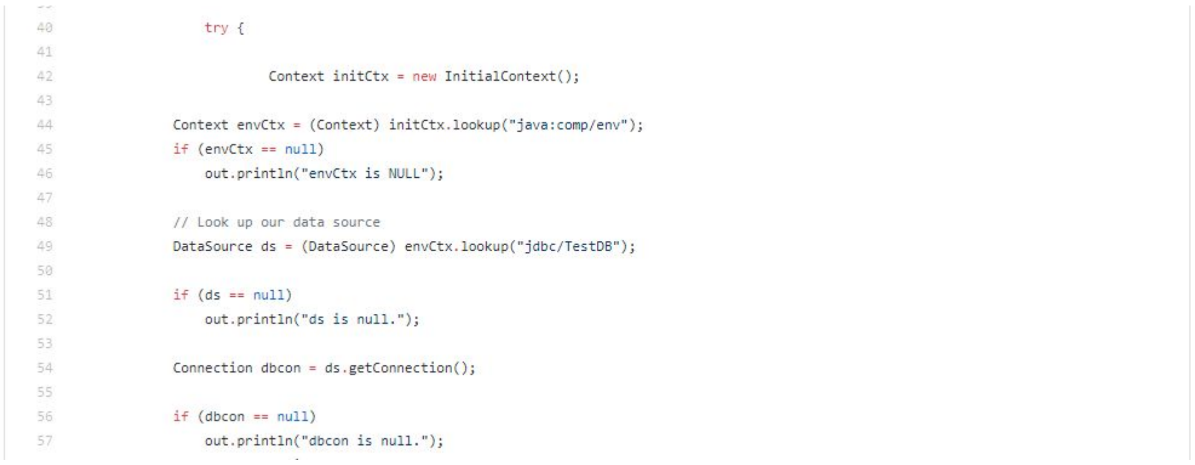
<u>Task 1</u>
How did you use connection pooling?

We used connection pooling to increase performance because creating a network connection to a database server is pretty expensive. By using a connection pool, wecan reuse existing connections/prepared statements, avoiding the cost of initiating a connection, which increases performance and uses less resources.

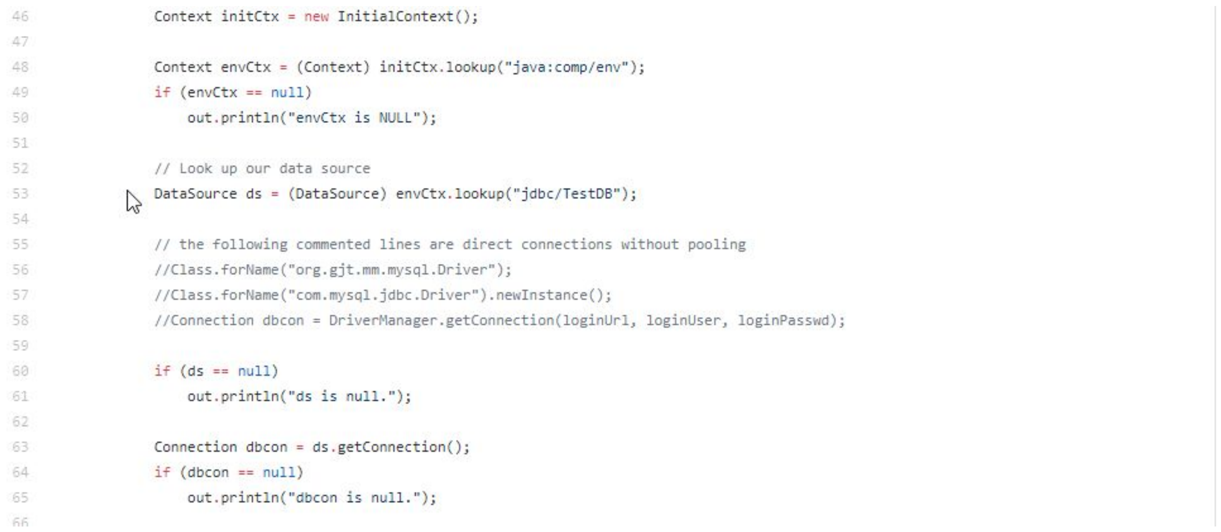
File name, line numbers as in Github:

1. project2/src/AutocompleteServlet.java from lines 44 to 58
2. project2/src/CheckOutServlet.java from lines 42 to 57
3. project2/src/DashBoard.java from lines 49 to 68
4. project2/src/EmployeeLogin.java from lines 43 to 62
5. project2/src/FulltextServlet.java from lines 51 to 70
6. project2/src/LoginServlet.java from lines 59 to 73
7. project2/src/SingleMovieServlet.java from lines 44 to 62
8. project2/src/SingleStarServlet.java from lines 44 to 62

Snapshots showing use in your code:



```
40                try {
41
42                        Context initCtx = new InitialContext();
43
44            Context envCtx = (Context) initCtx.lookup("java:comp/env");
45            if (envCtx == null)
46                out.println("envCtx is NULL");
47
48            // Look up our data source
49            DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
50
51            if (ds == null)
52                out.println("ds is null.");
53
54            Connection dbcon = ds.getConnection();
55
56            if (dbcon == null)
57                out.println("dbcon is null.");
```

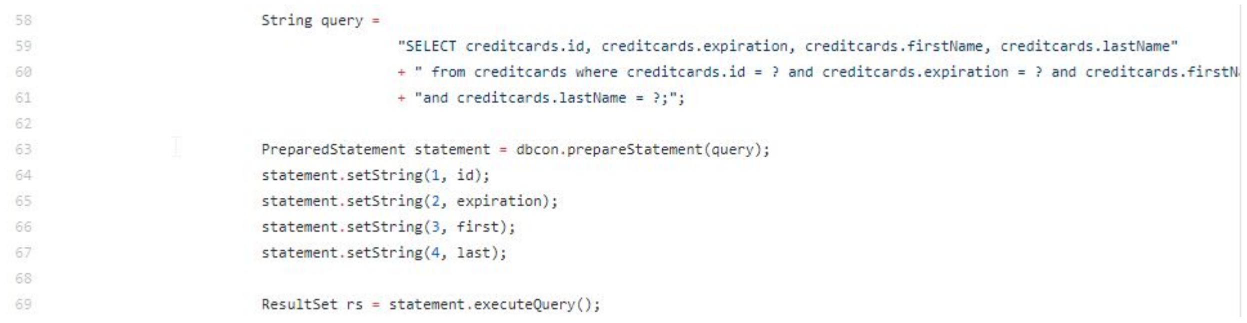


```
46              Context initCtx = new InitialContext();
47
48              Context envCtx = (Context) initCtx.lookup("java:comp/env");
49              if (envCtx == null)
50                  out.println("envCtx is NULL");
51
52              // Look up our data source
53              DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
54
55              // the following commented lines are direct connections without pooling
56              //Class.forName("org.gjt.mm.mysql.Driver");
57              //Class.forName("com.mysql.jdbc.Driver").newInstance();
58              //Connection dbcon = DriverManager.getConnection(loginUrl, loginUser, loginPasswd);
59
60              if (ds == null)
61                  out.println("ds is null.");
62
63              Connection dbcon = ds.getConnection();
64              if (dbcon == null)
65                  out.println("dbcon is null.");
66
```

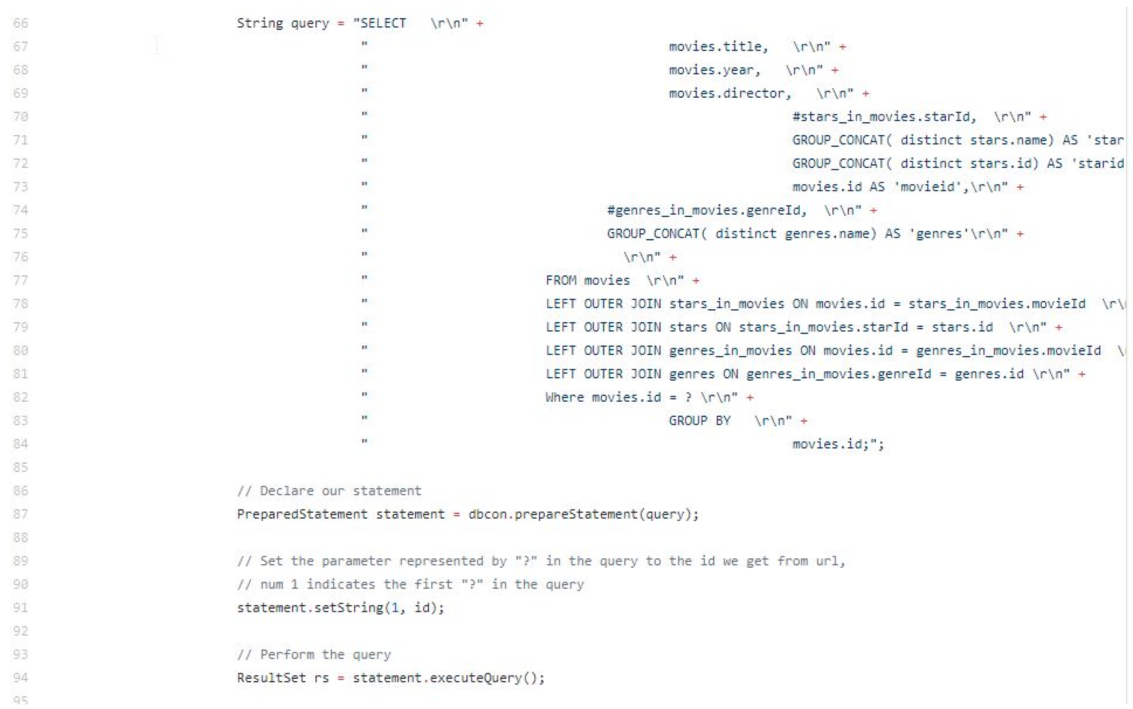
How did you use Prepared Statements?
Precompilation and DB-side caching of the SQL statement leads to overall faster execution and the ability to reuse the same SQL statement in batches. Automatic prevention of SQL injection attacks by builtin escaping of quotes and other special characters

File name, line numbers as in Github:
1. project2/src/AutocompleteServlet.java from lines 58 to 65
2. project2/src/CheckOutServlet.java from lines 58 to 69
3. project2/src/DashBoard.java from lines 69 to 79
4. project2/src/EmployeeLogin.java from lines 66 to 70
5. project2/src/FulltextServlet.java from lines 72 to 76
6. project2/src/LoginServlet.java from lines 77 to 80
7. project2/src/SingleMovieServlet.java from lines 66 to 94
8. project2/src/SingleStarServlet.java from lines 75 to 109
9. project2/src/ItemsServlet.java from lines 69 to 72

Snapshots showing use in your code:



```
58                      String query =
59                                      "SELECT creditcards.id, creditcards.expiration, creditcards.firstName, creditcards.lastName"
60                                      + " from creditcards where creditcards.id = ? and creditcards.expiration = ? and creditcards.firstN
61                                      + "and creditcards.lastName = ?;";
62
63                      PreparedStatement statement = dbcon.prepareStatement(query);
64                      statement.setString(1, id);
65                      statement.setString(2, expiration);
66                      statement.setString(3, first);
67                      statement.setString(4, last);
68
69                      ResultSet rs = statement.executeQuery();
```



```
String query = "SELECT   \r\n" +
                "                                       movies.title,   \r\n" +
                "                                       movies.year,   \r\n" +
                "                                       movies.director,   \r\n" +
                "                                                       #stars_in_movies.starId,  \r\n" +
                "                                                       GROUP_CONCAT( distinct stars.name) AS 'star
                "                                                       GROUP_CONCAT( distinct stars.id) AS 'starid
                "                                                       movies.id AS 'movieid',\r\n" +
                "                               #genres_in_movies.genreId,  \r\n" +
                "                               GROUP_CONCAT( distinct genres.name) AS 'genres'\r\n" +
                "                                 \r\n" +
                "                       FROM movies  \r\n" +
                "                       LEFT OUTER JOIN stars_in_movies ON movies.id = stars_in_movies.movieId  \r\
                "                       LEFT OUTER JOIN stars ON stars_in_movies.starId = stars.id  \r\n" +
                "                       LEFT OUTER JOIN genres_in_movies ON movies.id = genres_in_movies.movieId  \
                "                       LEFT OUTER JOIN genres ON genres_in_movies.genreId = genres.id \r\n" +
                "                       Where movies.id = ? \r\n" +
                "                                       GROUP BY   \r\n" +
                "                                                       movies.id;";

// Declare our statement
PreparedStatement statement = dbcon.prepareStatement(query);

// Set the parameter represented by "?" in the query to the id we get from url,
// num 1 indicates the first "?" in the query
statement.setString(1, id);

// Perform the query
ResultSet rs = statement.executeQuery();
```

Task 2

Address of AWS and Google instances:
AWS instance 1 apache2: 18.237.248.44
AWS instance 2 master: 34.209.244.33
AWS instance 3 slave: 34.212.202.163
Google: 35.230.48.212

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.

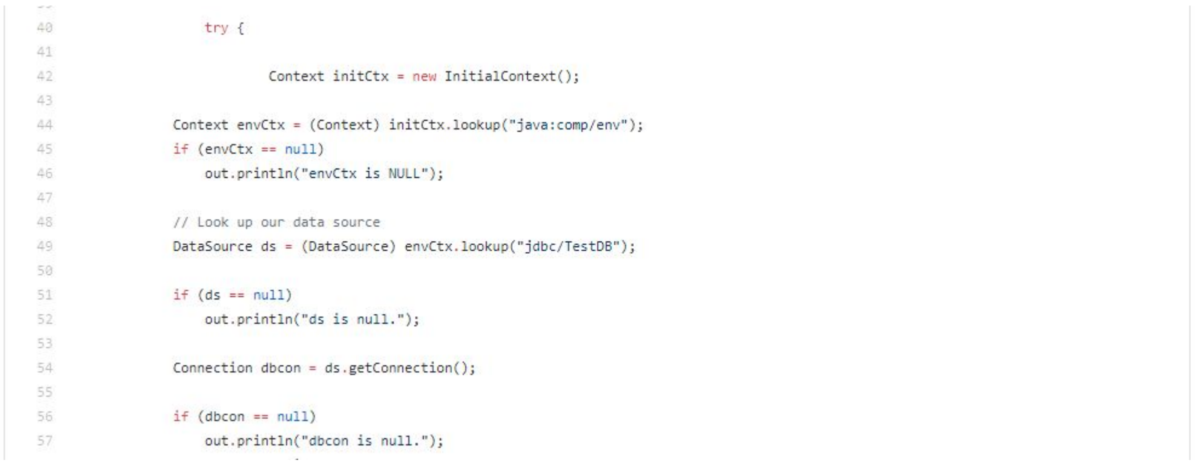
Explain how connection pooling works with two backend SQL (in your code)?
Connection pooling leads to sharing of connections among threads on a single instance. It doesn't share connection across instances. So connection pooling would not share the same pool across instances in our case.

File name, line numbers as in Github:

1. project2/src/AutocompleteServlet.java from lines 44 to 58
2. project2/src/CheckOutServlet.java from lines 42 to 57
3. project2/src/DashBoard.java from lines 49 to 68
4. project2/src/EmployeeLogin.java from lines 43 to 62
5. project2/src/FulltextServlet.java from lines 51 to 70

6. project2/src/LoginServlet.java from lines 59 to 73
7. project2/src/SingleMovieServlet.java from lines 44 to 62
8. project2/src/SingleStarServlet.java from lines 44 to 62

Snapshots:



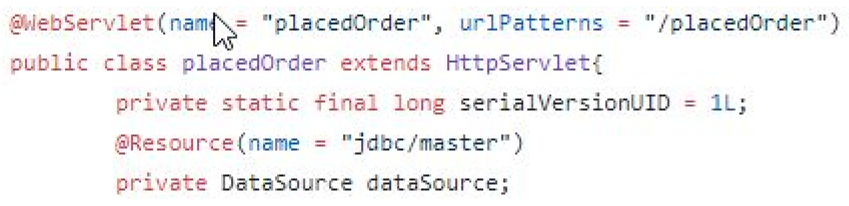
```
40              try {
41
42                      Context initCtx = new InitialContext();
43
44          Context envCtx = (Context) initCtx.lookup("java:comp/env");
45          if (envCtx == null)
46              out.println("envCtx is NULL");
47
48          // Look up our data source
49          DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
50
51          if (ds == null)
52              out.println("ds is null.");
53
54          Connection dbcon = ds.getConnection();
55
56          if (dbcon == null)
57              out.println("dbcon is null.");
```

How read/write requests were routed?
Read Requests were routed to either the master or slave instance because it doesn't matter where each read request is routed. However, write requests are routed only to the master instance. We created another datasource that url has the master instance ip for jdbc.

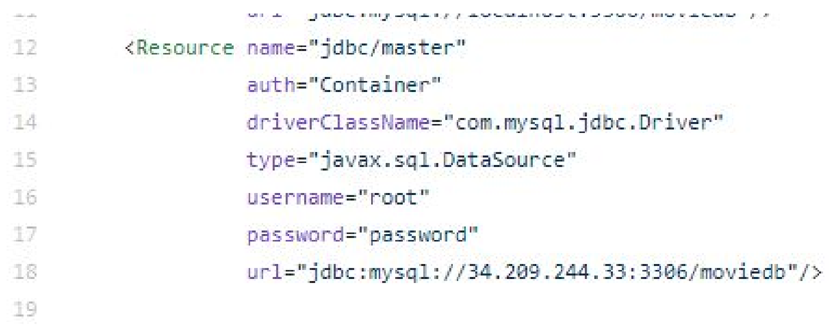
File name, line numbers as in Github:

project2/src/DashBoard.java	Lines 26 to 27
project2/src/InsertStar.java	Lines 24 to 25
project2/src/placedOrder.java	Lines 24 to 25
project2/WebContent/META-INF/context.xml	Lines 12 to 18

Snapshots:



```
@WebServlet(name = "placedOrder", urlPatterns = "/placedOrder")
public class placedOrder extends HttpServlet{
        private static final long serialVersionUID = 1L;
        @Resource(name = "jdbc/master")
        private DataSource dataSource;
```



```
                url="jdbc:mysql://localhost:3306/moviedb"/>
      <Resource name="jdbc/master"
                auth="Container"
                driverClassName="com.mysql.jdbc.Driver"
                type="javax.sql.DataSource"
                username="root"
                password="password"
                url="jdbc:mysql://34.209.244.33:3306/moviedb"/>
```

Task 3

Have you uploaded the log file to Github? Where is it located?
yes , master/122bLogFile.txt

Have you uploaded the HTML file to Github? Where is it located?
Yes, master/html_report/jmeter_report

Have you uploaded the script to Github? Where is it located?
Yes, /master/my_script.py

Have you uploaded the WAR file and README to Github? Where is it located?
Yes /master/readme.md, war file is /master/project2.war