<u>Project 5 Report (Team 65)</u>

<u>Task 1</u>

- How did you use connection pooling?

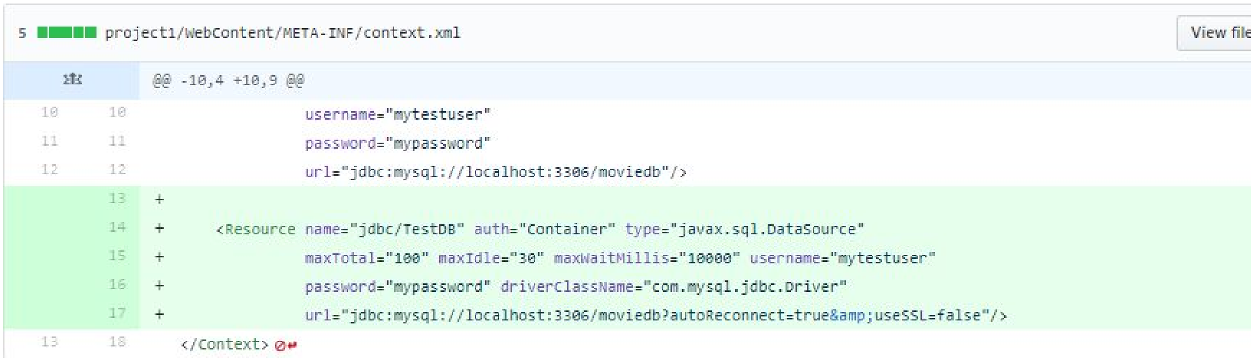
We enabled connection pooling by adding a DataSource in the context.xml file and referencing it in our web.xml file. For each servlet in our Fabflix project, we looked up the DataSource using the Context class to search up the environment naming context and the DataSource.

- File name, line numbers as in Github

File name	Line numbers
context.xml	13-17
web.xml	12-22
AutocompleteSearchServlet.java	62-76
BrowseMenuServlet.java	40-55
BrowseResultsServlet.java	50-64
CheckoutInformationServlet.java	44-58
ConfrimationPageServlet.java	49-63
DashboardServlet.java	81-93
EmployeeLoginServlet.java	66-80
LoginServlet.java	73-85, 87-89
MovieServlet.java	79-94
SingleMovieServlet.java	46-60
SingleStarServlet.java	48-60

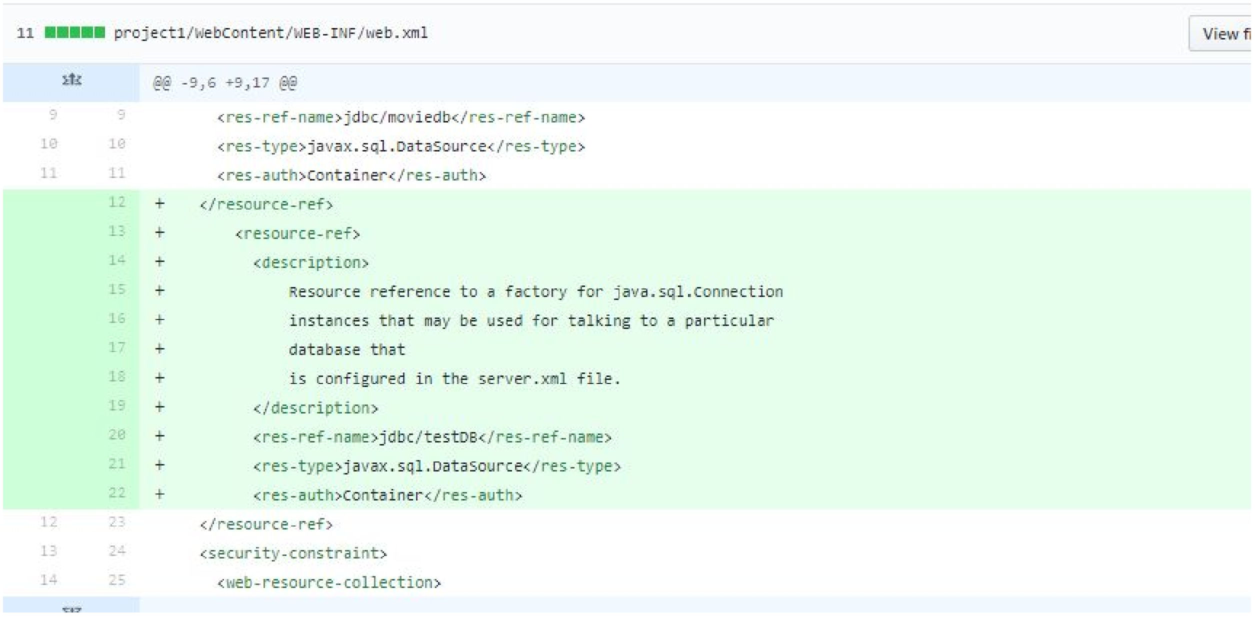
- Snapshots showing use in your code

context.xml:



```
@@ -10,4 +10,9 @@
                      username="mytestuser"
                      password="mypassword"
                      url="jdbc:mysql://localhost:3306/moviedb"/>
+
+     <Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
+               maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
+               password="mypassword" driverClassName="com.mysql.jdbc.Driver"
+               url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false"/>
  </Context>
```

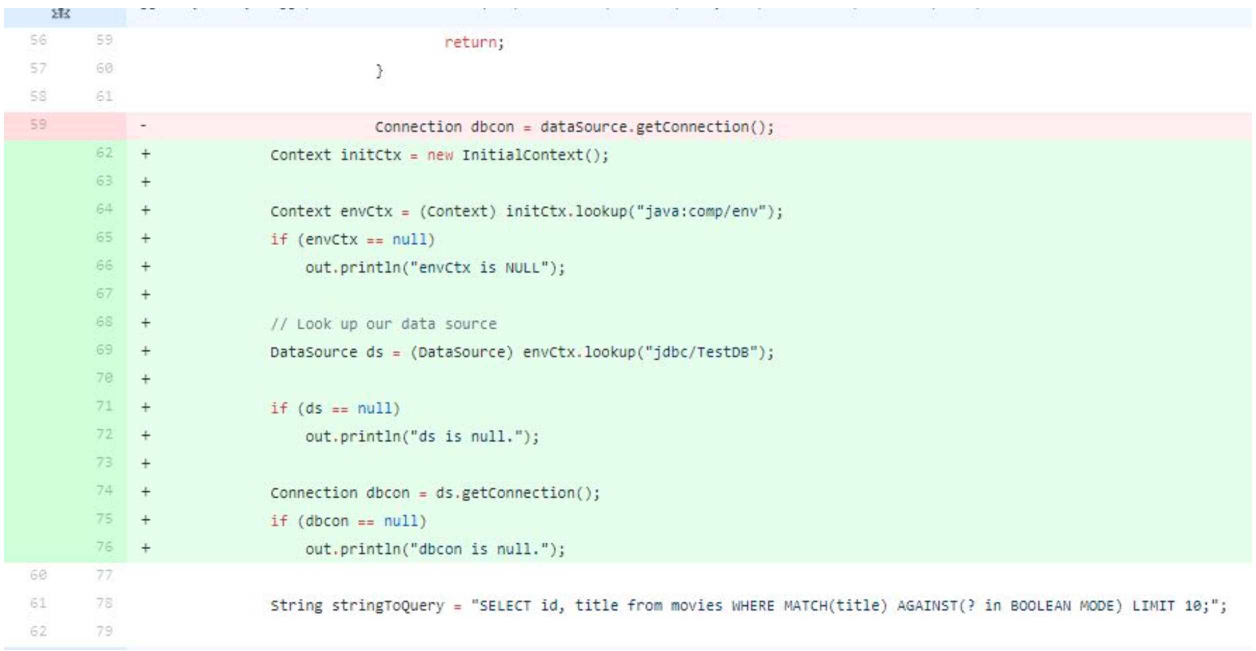
web.xml:



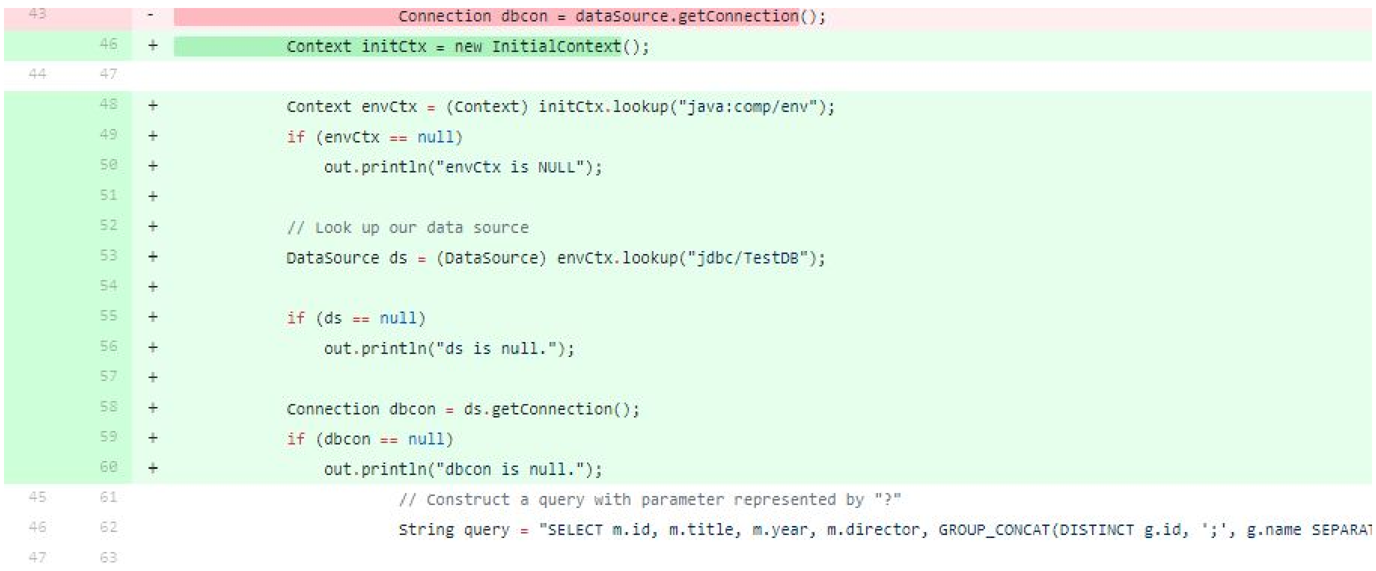
```
@@ -9,6 +9,17 @@
          <res-ref-name>jdbc/moviedb</res-ref-name>
          <res-type>javax.sql.DataSource</res-type>
          <res-auth>Container</res-auth>
+     </resource-ref>
+         <resource-ref>
+           <description>
+               Resource reference to a factory for java.sql.Connection
+               instances that may be used for talking to a particular
+               database that
+               is configured in the server.xml file.
+           </description>
+           <res-ref-name>jdbc/testDB</res-ref-name>
+           <res-type>javax.sql.DataSource</res-type>
+           <res-auth>Container</res-auth>
      </resource-ref>
      <security-constraint>
        <web-resource-collection>
```

For all the servlets listed in the previous table, we were able to enable connection pooling by adding the following code (seen in AutocompleteSearchServlet.java and

SingleMovieServlet.java below)



```
                        return;
                    }

-                   Connection dbcon = dataSource.getConnection();
+           Context initCtx = new InitialContext();
+
+           Context envCtx = (Context) initCtx.lookup("java:comp/env");
+           if (envCtx == null)
+               out.println("envCtx is NULL");
+
+           // Look up our data source
+           DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
+
+           if (ds == null)
+               out.println("ds is null.");
+
+           Connection dbcon = ds.getConnection();
+           if (dbcon == null)
+               out.println("dbcon is null.");

            String stringToQuery = "SELECT id, title from movies WHERE MATCH(title) AGAINST(? in BOOLEAN MODE) LIMIT 10;";

```



```
-                   Connection dbcon = dataSource.getConnection();
+           Context initCtx = new InitialContext();

+           Context envCtx = (Context) initCtx.lookup("java:comp/env");
+           if (envCtx == null)
+               out.println("envCtx is NULL");
+
+           // Look up our data source
+           DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
+
+           if (ds == null)
+               out.println("ds is null.");
+
+           Connection dbcon = ds.getConnection();
+           if (dbcon == null)
+               out.println("dbcon is null.");
                    // Construct a query with parameter represented by "?"
                    String query = "SELECT m.id, m.title, m.year, m.director, GROUP_CONCAT(DISTINCT g.id, ';', g.name SEPARAT

```

- How did you use Prepared Statements?

We used Prepared Statements in most of the servlets to prevent SQL injections, and we used this by declaring the Prepared Statement and binding values to the “?” parameters in the queries that used them.

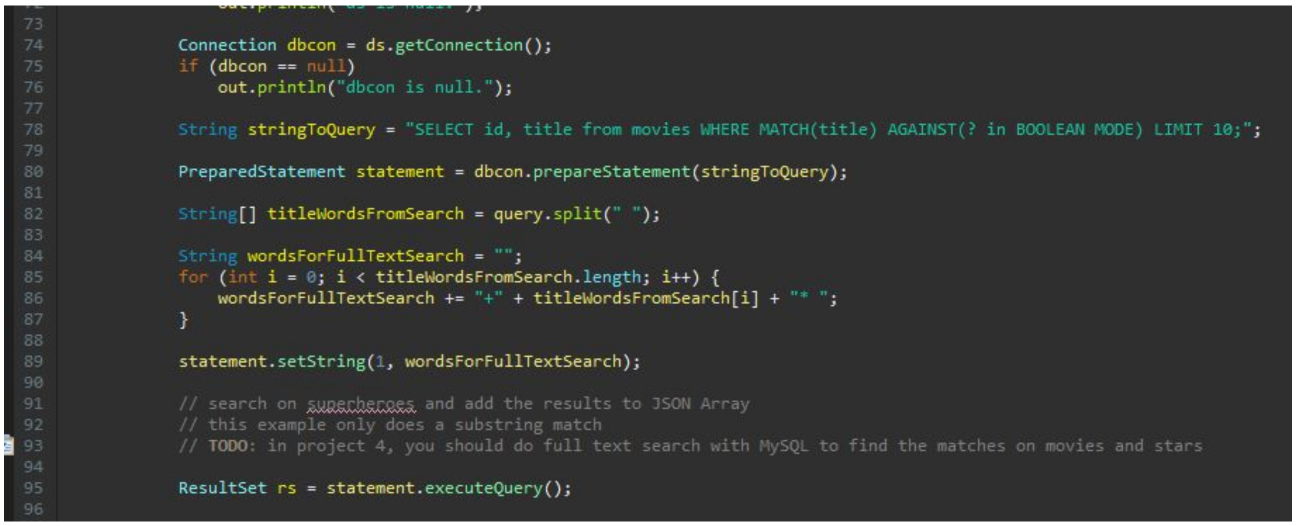
- File name, line numbers as in Github

File name	Line numbers

AutocompleteSearchServlet.java	80
BrowseResultsServlet.java	100
CheckoutInformationServlet.java	62
ConfrimationPageServlet.java	68
DashboardServlet.java	98
EmployeeLoginServlet.java	90
LoginServlet.java	90
MovieServlet.java	168
SingleMovieServlet.java	65
SingleStarServlet.java	65

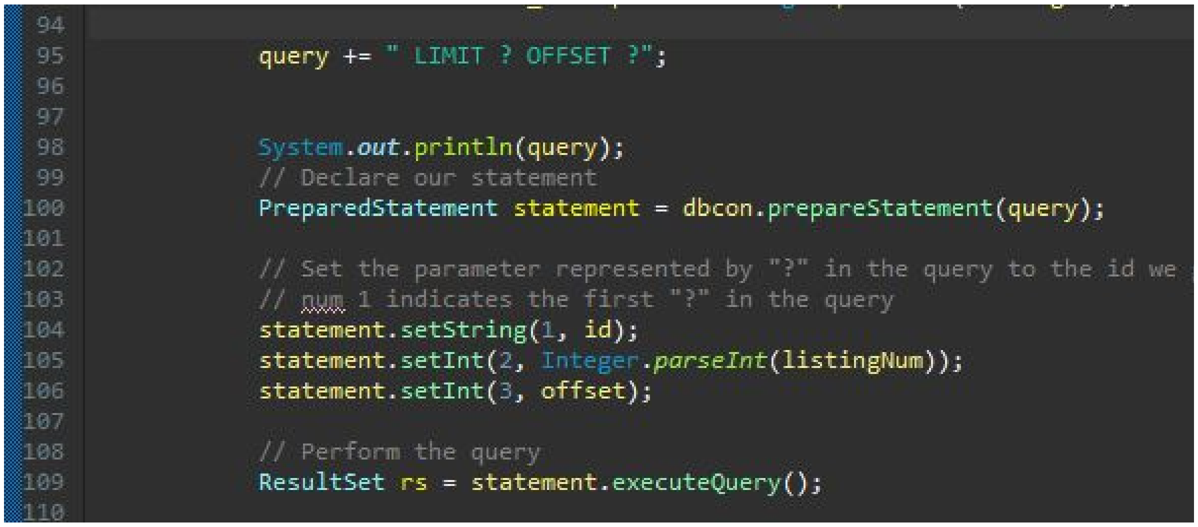
- Snapshots showing use in your code

AutocompleteSearchServlet.java



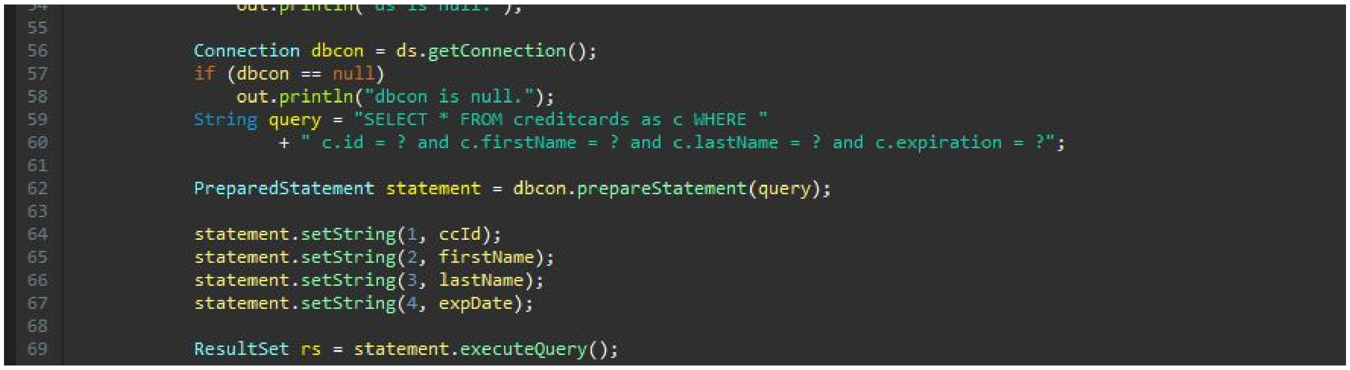
```
            Connection dbcon = ds.getConnection();
            if (dbcon == null)
                out.println("dbcon is null.");

            String stringToQuery = "SELECT id, title from movies WHERE MATCH(title) AGAINST(? in BOOLEAN MODE) LIMIT 10;";

            PreparedStatement statement = dbcon.prepareStatement(stringToQuery);

            String[] titleWordsFromSearch = query.split(" ");

            String wordsForFullTextSearch = "";
            for (int i = 0; i < titleWordsFromSearch.length; i++) {
                wordsForFullTextSearch += "+" + titleWordsFromSearch[i] + "* ";
            }

            statement.setString(1, wordsForFullTextSearch);

            // search on superheroes and add the results to JSON Array
            // this example only does a substring match
            // TODO: in project 4, you should do full text search with MySQL to find the matches on movies and stars

            ResultSet rs = statement.executeQuery();
```

BrowseResultsServlet.java



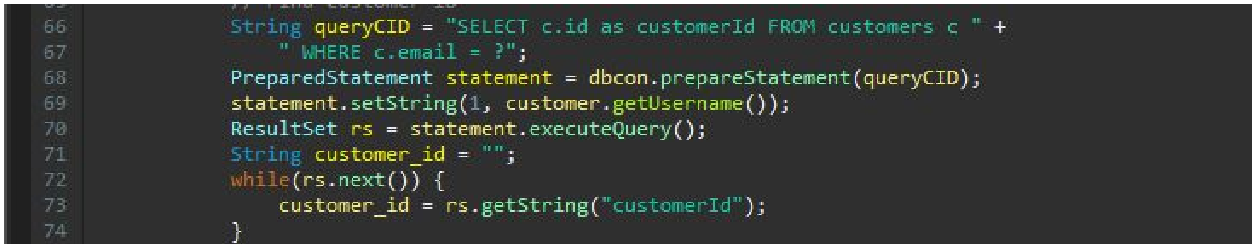
```java
        query += " LIMIT ? OFFSET ?";


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

        // Set the parameter represented by "?" in the query to the id we
        // num 1 indicates the first "?" in the query
        statement.setString(1, id);
        statement.setInt(2, Integer.parseInt(listingNum));
        statement.setInt(3, offset);

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

```

CheckoutInformationServlet.java



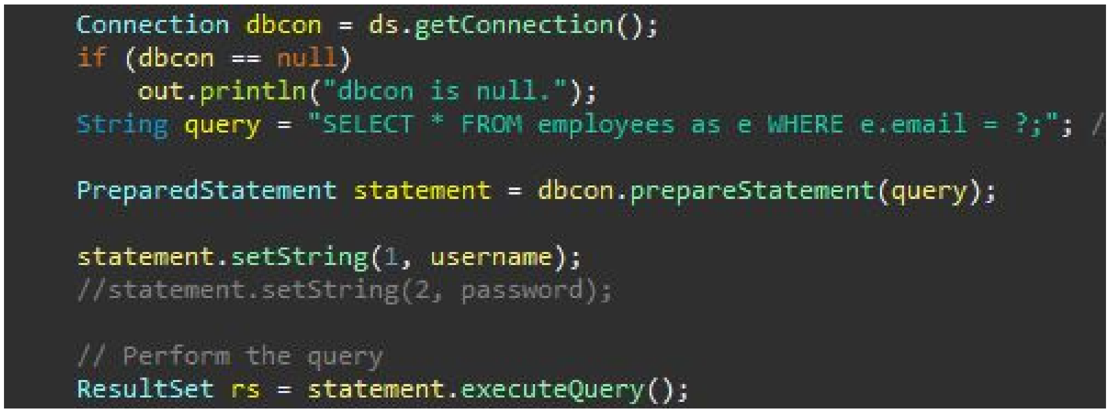
```java
            out.println("ds is null.");

        Connection dbcon = ds.getConnection();
        if (dbcon == null)
            out.println("dbcon is null.");
        String query = "SELECT * FROM creditcards as c WHERE "
                + " c.id = ? and c.firstName = ? and c.lastName = ? and c.expiration = ?";

        PreparedStatement statement = dbcon.prepareStatement(query);

        statement.setString(1, ccId);
        statement.setString(2, firstName);
        statement.setString(3, lastName);
        statement.setString(4, expDate);

        ResultSet rs = statement.executeQuery();
```

ConfrimationPageServlet.java



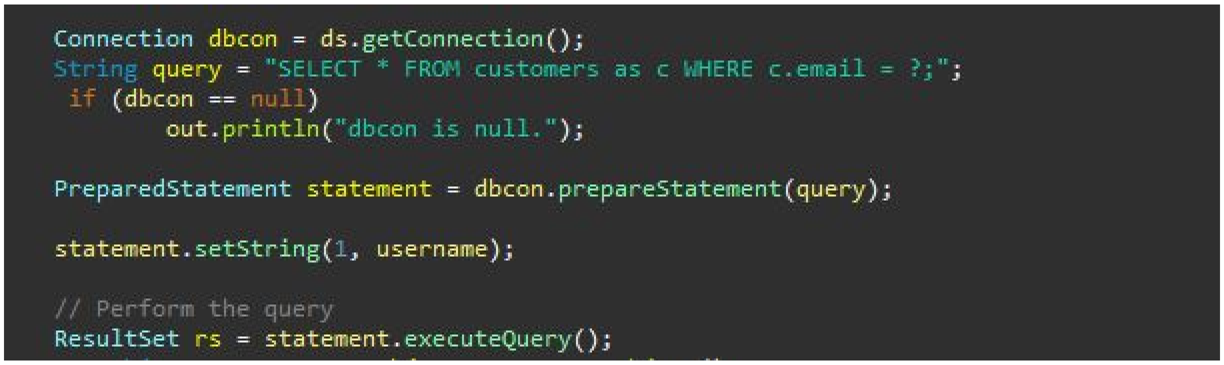
```java
        String queryCID = "SELECT c.id as customerId FROM customers c " +
            " WHERE c.email = ?";
        PreparedStatement statement = dbcon.prepareStatement(queryCID);
        statement.setString(1, customer.getUsername());
        ResultSet rs = statement.executeQuery();
        String customer_id = "";
        while(rs.next()) {
            customer_id = rs.getString("customerId");
        }
```

EmployeeLoginServlet.java



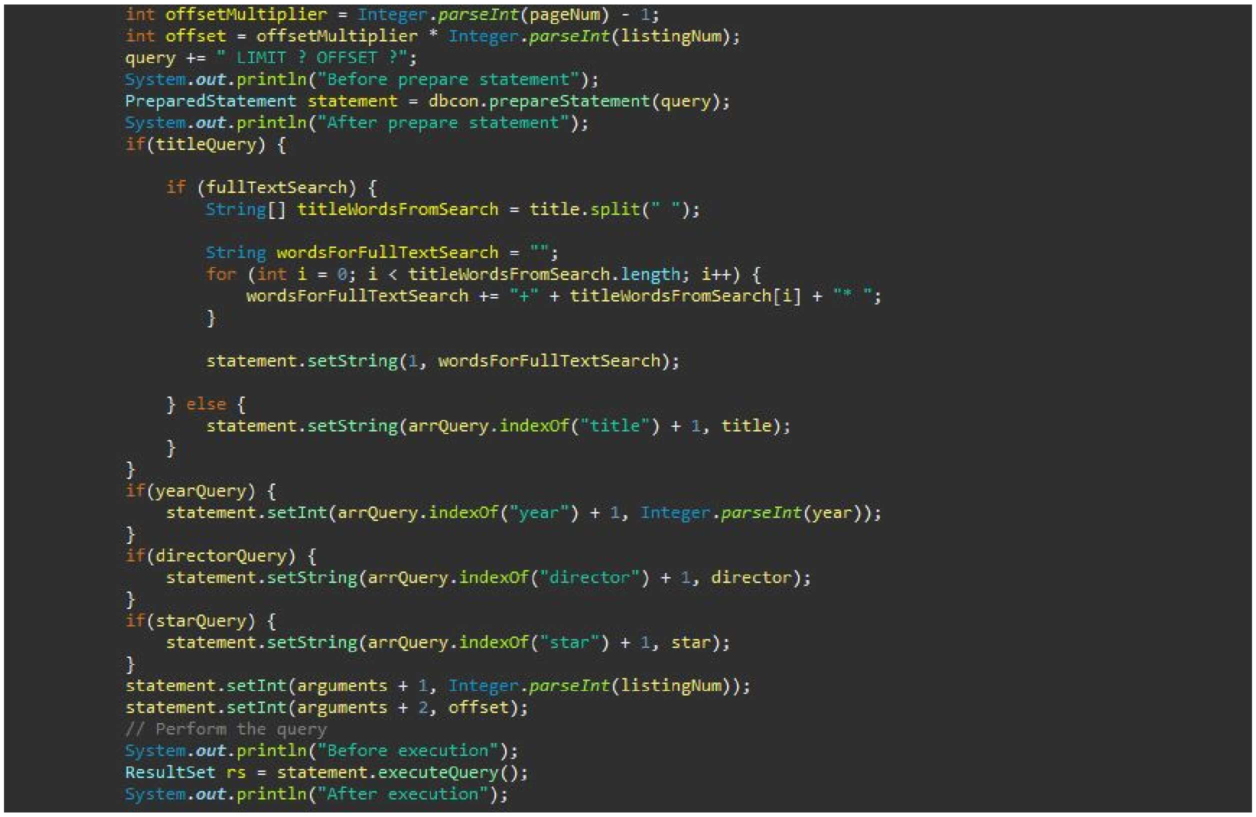
```
    Connection dbcon = ds.getConnection();
    if (dbcon == null)
        out.println("dbcon is null.");
    String query = "SELECT * FROM employees as e WHERE e.email = ?;"; /

    PreparedStatement statement = dbcon.prepareStatement(query);

    statement.setString(1, username);
    //statement.setString(2, password);

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

LoginServlet.java



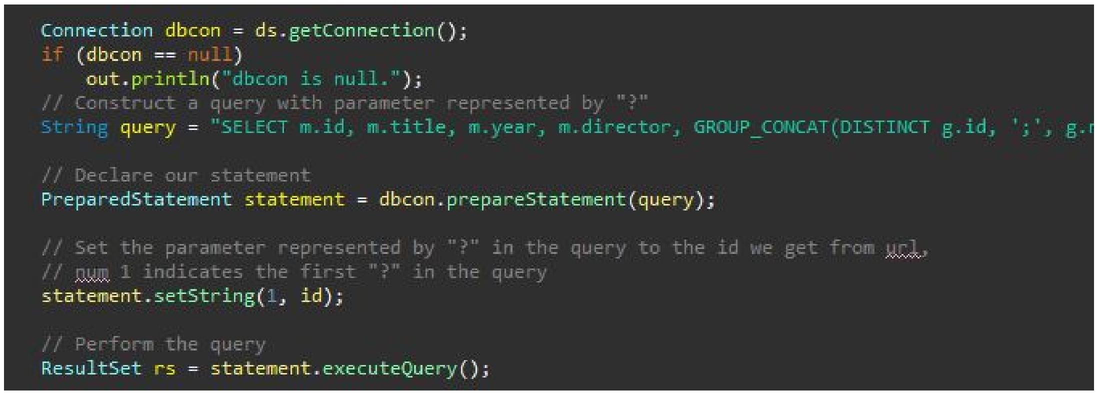
```
  Connection dbcon = ds.getConnection();
  String query = "SELECT * FROM customers as c WHERE c.email = ?;";
   if (dbcon == null)
        out.println("dbcon is null.");

  PreparedStatement statement = dbcon.prepareStatement(query);

  statement.setString(1, username);

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

MovieServlet.java



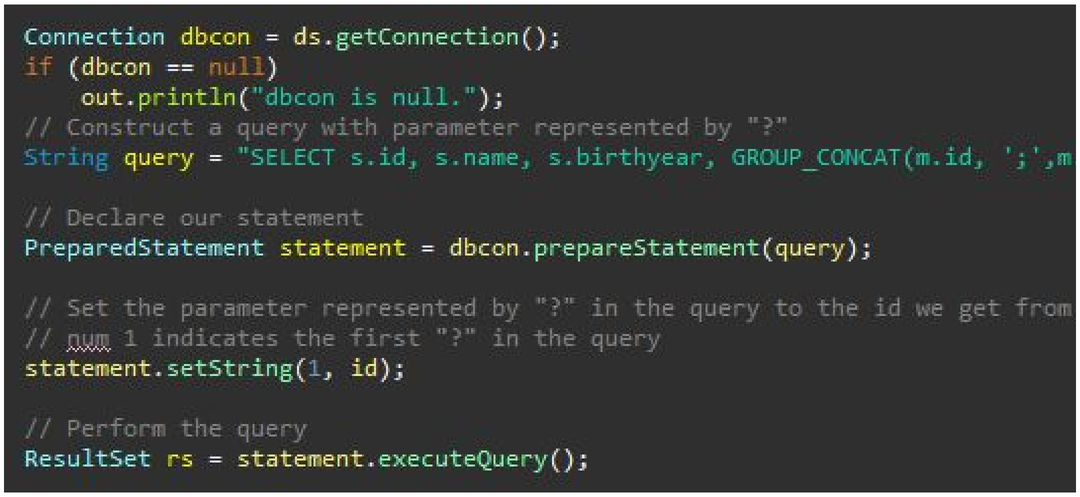
```java
int offsetMultiplier = Integer.parseInt(pageNum) - 1;
int offset = offsetMultiplier * Integer.parseInt(listingNum);
query += " LIMIT ? OFFSET ?";
System.out.println("Before prepare statement");
PreparedStatement statement = dbcon.prepareStatement(query);
System.out.println("After prepare statement");
if(titleQuery) {

    if (fullTextSearch) {
        String[] titleWordsFromSearch = title.split(" ");

        String wordsForFullTextSearch = "";
        for (int i = 0; i < titleWordsFromSearch.length; i++) {
            wordsForFullTextSearch += "+" + titleWordsFromSearch[i] + "* ";
        }

        statement.setString(1, wordsForFullTextSearch);

    } else {
        statement.setString(arrQuery.indexOf("title") + 1, title);
    }
}
if(yearQuery) {
    statement.setInt(arrQuery.indexOf("year") + 1, Integer.parseInt(year));
}
if(directorQuery) {
    statement.setString(arrQuery.indexOf("director") + 1, director);
}
if(starQuery) {
    statement.setString(arrQuery.indexOf("star") + 1, star);
}
statement.setInt(arguments + 1, Integer.parseInt(listingNum));
statement.setInt(arguments + 2, offset);
// Perform the query
System.out.println("Before execution");
ResultSet rs = statement.executeQuery();
System.out.println("After execution");
```

SingleMovieServlet.java



```java
Connection dbcon = ds.getConnection();
if (dbcon == null)
    out.println("dbcon is null.");
// Construct a query with parameter represented by "?"
String query = "SELECT m.id, m.title, m.year, m.director, GROUP_CONCAT(DISTINCT g.id, ';', g.

// 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();
```

SingleStarServlet.java


```
Connection dbcon = ds.getConnection();
if (dbcon == null)
    out.println("dbcon is null.");
// Construct a query with parameter represented by "?"
String query = "SELECT s.id, s.name, s.birthyear, GROUP_CONCAT(m.id, ';',m

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

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

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

<u>Task 2</u>

- Address of AWS and Google instances

This is subject to change because of the limited hours on AWS

Instance	IP address	URL to Fablix
1	18.224.21.244	http://18.224.21.244:80/project1/
2 (master)	3.17.207.9	http://3.17.207.9:8080/project1/
3 (slave)	18.218.103.37	http://18.218.103.37:8080/project1/
Google Instance	35.229.117.140	http://35.229.117.140/project1/

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

Yes, we verified that they are accessible, and the Fablix site gets opened on Google's 80 port and AWS' 8080 port

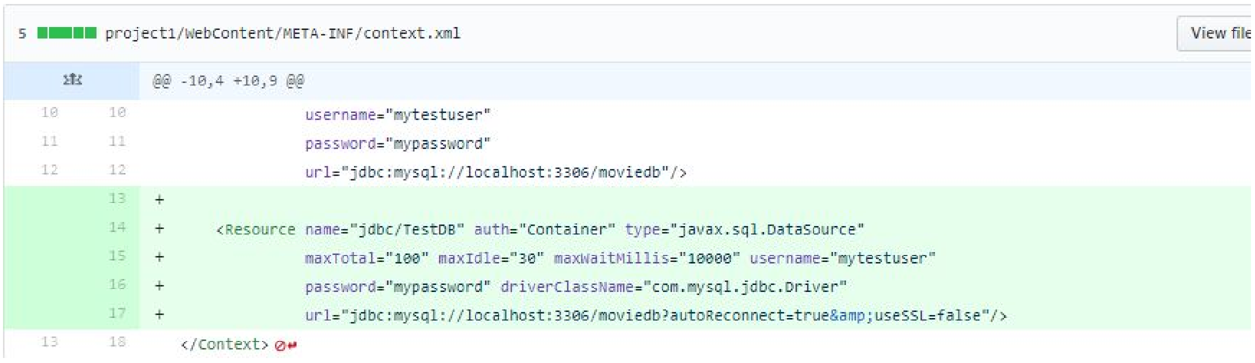
- Explain how connection pooling works with two backend SQL (in your code)?

We created 2 URLs in context.xml that linked to the backend SQL in the localhost (instance 1) and master instances

    - File name, line numbers as in Github

Context.xml - lines 13-17

- Snapshots



```
5 project1/WebContent/META-INF/context.xml                                    View file
      @@ -10,4 +10,9 @@
10  10                      username="mytestuser"
11  11                      password="mypassword"
12  12                      url="jdbc:mysql://localhost:3306/moviedb"/>
    13  +
    14  +       <Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
    15  +                   maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
    16  +                   password="mypassword" driverClassName="com.mysql.jdbc.Driver"
    17  +                   url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false"/>
13  18      </Context>
```

- How read/write requests were routed?

Using the 2 resources from the context.xml file, for each servlet, we reference which resource we want to use for the servlet depending on if it's a read or write request. For jdbc/moviedb, that was for any read requests; for jdbc/testdb, that was for any write requests.

- File name, line numbers as in Github

  Same as Task 1 (first table)

- Snapshots

  Same as Task 1 (first set of snapshots)

Task 3

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

N/A

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

N/A

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

N/A

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

N/A