

Task 1

- How did you use connection pooling?

```
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");
Connection connection = ds.getConnection();
```

I firstly copy the JDBC connector's jar file into the tomcat's library. Then, by adding <Resource> in the WebContent/META-INF/context.xml and using the code above to look up specific connection resource in all servlets that need to connect to database. Specifically, I used connection pooling in my AutoServlet, Browsing Servlet, dashboard servlet, Login Servlet, Movie Servlet, singleMovie Servlet, Single Star servlet with looking up "TestDB" and used connection pooling in my add_movie, completeCheckout with using look up "Write".

- File name, line numbers as in Github

```
/project5/src/MovieServlet.java 81-89
/project5/src/AutoServlet.java 43-49
/project5/src/BrowsingServlet.java 56-64
/project5/src/LoginServlet.java 63-69
/project5/src/add_movie.java 49-56
/project5/src/completecheckoutServlet.java 68-74
/project5/src/dashboard.java 45-51
/project5/src/makechange.java 50-55
/project5/src/singleMovieServlet.java 43 -50
/project5/src/singleStarServlet.java 45-53
```

- Snapshots showing use in your code
MovieServlet.java 81-89

```
//
//
Connection connection = DriverManager.getConnection(loginUrl, loginUser, loginPasswd);
Context initCtx = new InitialContext();
long TimeJ = System.nanoTime();
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");
Connection connection = ds.getConnection();
```

AutoServlet.java 43-49

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

BrowsingServlet.java 56-64

```
55
56         Context initCtx = new InitialContext();
57
58         Context envCtx = (Context) initCtx.lookup("java:comp/env");
59         if (envCtx == null)
60             out.println("envCtx is NULL");
61
62         // Look up our data source
63         DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
64         Connection connection = ds.getConnection();
```

LoginServlet.java 63-69

```
59         try {
60
61             Context initCtx = new InitialContext();
62
63             Context envCtx = (Context) initCtx.lookup("java:comp/env");
64             if (envCtx == null)
65                 System.out.println("envCtx is NULL");
66
67             // Look up our data source
68             DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
69             Connection connection = ds.getConnection();
```

add_movie.java 49-56

```
47         try {
48
49             Context initCtx = new InitialContext();
50
51             Context envCtx = (Context) initCtx.lookup("java:comp/env");
52             if (envCtx == null)
53                 System.out.println("envCtx is NULL");
54
55             // Look up our data source
56             DataSource ds = (DataSource) envCtx.lookup("jdbc/write");
57             Connection connection = ds.getConnection();
```

completecheckoutServlet.java 68-74

```
65         try {
66             Context initCtx = new InitialContext();
67
68             Context envCtx = (Context) initCtx.lookup("java:comp/env");
69             if (envCtx == null)
70                 System.out.println("envCtx is NULL");
71
72             // Look up our data source
73             DataSource ds = (DataSource) envCtx.lookup("jdbc/write");
74             Connection connection = ds.getConnection();
```

dashboard.java 45-51

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

makechange.java 50-55

```
..
47         try {
48             Context initCtx = new InitialContext();
49
50             Context envCtx = (Context) initCtx.lookup("java:comp/env");
51             if (envCtx == null)
52                 System.out.println("envCtx is NULL");
53
54             // Look up our data source
55             DataSource ds = (DataSource) envCtx.lookup("jdbc/write");
56             Connection connection = ds.getConnection();
```

singleMovieServlet.java 43 -50

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

singleStarServlet.java 45-53

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

- How did you use Prepared Statements?

For all plain statement, we changed it to prepared statement and then execute it after we inserted proper variable; So, after execution, It could give us proper data in return.

- File name, line numbers as in Github
AutoServlet.java line: 58 ~ 62
BrowsingServlet.java: 66 ~ 79
CartServlet.java: 98 ~103
LoginServlet.java: 71~78
MovieServlet.java: 118~187
MovieServlet.java: 260~295
add_movie.java: 62~113
completecheckoutServlet.java: 76 ~ 80
completecheckoutServlet.java: 115~133
dashboard.java: 54 ~ 60
makechange.java: 67~75
singleMovieServlet.java: 55~115
singleStarServlet.java: 58~109

- Snapshots showing use in your code
AutoServlet.java line: 58 ~ 62

```

57         query += " IN BOOLEAN MODE) LIMIT 10 ;
58         PreparedStatement statement = connection.prepareStatement(query);
59         for(int i=0;i<wordList.length;i++) {
60             statement.setString(i+1,""+wordList[i]+"*");
61         }
62         ResultSet rs = statement.executeQuery();
63         while(rs.next()) {
64             JsonObject movieObject = new JsonObject();
65             String movieId = rs.getString("id");
66             String movieTitle = rs.getString("title");
67             movieObject.addProperty("movieId", movieId);

```

BrowsingServlet.java: 66 ~ 79

```

66         PreparedStatement statement = connection.prepareStatement(query);
67         ResultSet genreSet = statement.executeQuery();
68
69
70
71         JSONArray genreArray = new JSONArray();
72
73
74         // add a row for every star result
75         while (genreSet.next()) {
76
77             String genreName = genreSet.getString("name");
78
79             // Add genre name to array

```

CartServlet.java: 98 ~103

```

97         String query = "SELECT M.title from movies M where M.id = ?";
98         PreparedStatement statement = connection.prepareStatement(query);
99         statement.setString(1, id);
100         ResultSet movieSet = statement.executeQuery();
101         while(movieSet.next())
102         {
103             title = movieSet.getString("title");
104         }
105     } catch (Exception e) {
106         // TODO Auto-generated catch block
107         e.printStackTrace();
108     }

```

LoginServlet.java: 71~78

```

71         PreparedStatement statement = connection.prepareStatement(query);
72         statement.setString(1, username);
73         ResultSet rs = statement.executeQuery();
74         while(rs.next())
75         {
76             email = rs.getString("email");
77             pwd = rs.getString("password");
78         }
79     }

```

MovieServlet.java: 118~187

```

118 PreparedStatement countStatement = connection.prepareStatement(countQuery);
119 PreparedStatement statement = connection.prepareStatement(tableQuery);
120
121
122 if(year.length()>0)
123 {
124     statement.setInt(1,Integer.parseInt(year));
125     statement.setInt(2,Integer.parseInt(year));
126     countStatement.setInt(1,Integer.parseInt(year));
127     countStatement.setInt(2,Integer.parseInt(year));
128 }
129 else {
130     statement.setInt(1,1000);
131     statement.setInt(2,9999);
132     countStatement.setInt(1,1000);
133     countStatement.setInt(2,9999);
134 }
135 if(director.length()>0)
136 {
137     statement.setString(3,"%"+director+"%");
138     countStatement.setString(3,"%"+director+"%");
139 }
140 else {
141     statement.setString(3,"%");
142     countStatement.setString(3,"%");
143 }
144 if(star.length()>0)
145 {
146     statement.setString(4,"%"+star+"%");
147     countStatement.setString(4,"%"+star+"%");
148 }
149 else {

```

MovieServlet.java: 260~295

```

264         PreparedStatement genreStatement = connection.prepareStatement(genreQuery);
265         genreStatement.setString(1, movieId);
266         ResultSet genreSet = genreStatement.executeQuery();
267
268         JSONArray genreArray = new JSONArray();
269         while(genreSet.next())
270         {
271             String name = genreSet.getString("name");
272             JSONObject genreObject = new JSONObject();
273             genreObject.addProperty("name",name);
274             genreArray.add(genreObject);
275
276         }
277         movieObject.add("genre", genreArray);
278         movieObject.addProperty("urlTitle",title);
279         movieObject.addProperty("urlYear",year);
280         movieObject.addProperty("urlDirector",director);
281         movieObject.addProperty("urlStar",star);
282         movieObject.addProperty("count",count);
283         movieObject.addProperty("npp",npp);
284         movieObject.addProperty("sorting",sorting);
285         movieObject.addProperty("page",page);
286         movieObject.addProperty("totalPage",totalPage);
287         movieObject.addProperty("genreWord",genre);
288         movieObject.addProperty("st",st);
289         genreSet.close();
290         genreStatement.close();
291         movieArray.add(movieObject);
292
293     }
294
295

```

add_movie.java: 62~113

```

62         PreparedStatement statement = connection.prepareStatement(query);
63
64         ResultSet rs = statement.executeQuery();
65
66         JSONArray tableArray = new JSONArray();
67         // //JSONArray genreArray = new JSONArray();
68
69         while (rs.next()) {
70             String tableName = rs.getString("TABLE_NAME");
71
72             String queryNext = "Select COLUMN_NAME, DATA_TYPE,IS_NULLABLE " +
73                 "From INFORMATION_SCHEMA.COLUMNS " +
74                 "Where TABLE_NAME Like ?";
75
76             PreparedStatement statementNext = connection.prepareStatement(queryNext);
77
78             statementNext.setString(1, tableName);
79
80             ResultSet result = statementNext.executeQuery();
81
82
83             JSONObject tableObject = new JSONObject();
84             tableObject.addProperty("name",tableName);
85             JSONArray columArray = new JSONArray();
86             while(result.next()) {
87                 JSONObject columnObject = new JSONObject();
88                 columnObject.addProperty("cname", result.getString("COLUMN_NAME"));
89                 columnObject.addProperty("ctype", result.getString("DATA_TYPE"));
90                 columArray.add(columnObject);
91
92             }
93             tableObject.add("column" columArray);

```

completecheckoutServlet.java: 76 ~ 80

```
76         PreparedStatement statement = connection.prepareStatement(query);
77         statement.setString(1, card_num);
78         statement.setString(2, first);
79         statement.setString(3, last);
80         statement.setDate(4, sDate);
81
82
83         ResultSet resultSet = statement.executeQuery();
84
85
```

completecheckoutServlet.java: 115~133

```
115         PreparedStatement statement12 = connection.prepareStatement(query);
116         statement12.setString(1, email);
117         ResultSet CresultSet = statement12.executeQuery();
118         // add a row for every star result
119         while (CresultSet.next()) {
120
121             userId = CresultSet.getInt("id");
122
123         }
124         for(Map.Entry<String, Integer> entry:l.entrySet()) {
125             String movieId = entry.getKey();
126             int num = entry.getValue();
127             for(int i=0;i<num;i++) {
128                 String query1 = "insert into sales(customerId,movieId,saleDate) values(?, ?,curdate());";
129                 PreparedStatement statement1 = connection.prepareStatement(query1);
130                 statement1.setInt(1, userId);
131                 statement1.setString(2, movieId);
132                 statement1.executeUpdate();
133
```

dashboard.java: 54 ~ 60

```
54         PreparedStatement statement = connection.prepareStatement(query);
55         statement.setString(1, username);
56         ResultSet rs = statement.executeQuery();
57         while(rs.next())
58         {
59             email = rs.getString("email");
60             pwd = rs.getString("password");
61         }
```

makechange.java: 67~75


```

61         PreparedStatement statement = connection.prepareStatement(query);
62
63         ResultSet rs = statement.executeQuery();
64
65         JSONArray tableArray = new JSONArray();
66         // JSONArray genreArray = new JSONArray();
67
68         while (rs.next()) {
69             String tableName = rs.getString("TABLE_NAME");
70
71             String queryNext = "Select COLUMN_NAME, DATA_TYPE, IS_NULLABLE " +
72                               "From INFORMATION_SCHEMA.COLUMNS " +
73                               "Where TABLE_NAME Like ?;";
74
75             PreparedStatement statementNext = connection.prepareStatement(queryNext);
76
77             statementNext.setString(1, tableName);
78
79             ResultSet result = statementNext.executeQuery();

```

singleMovieServlet.java: 55~115

```

55         PreparedStatement statement = connection.prepareStatement(query);
56         PreparedStatement genreStatement = connection.prepareStatement(genreQuery);
57
58
59         statement.setString(1, id);
60         genreStatement.setString(1, id);
61
62         ResultSet rs = statement.executeQuery();
63         ResultSet gs = genreStatement.executeQuery();
64         JSONArray starArray = new JSONArray();
65         JSONArray genreArray = new JSONArray();
66         String title = request.getParameter("title");
67         String year = request.getParameter("year");
68         String director = request.getParameter("director");
69         String star = request.getParameter("star");
70         String sorting = request.getParameter("sorting");
71         String npp = request.getParameter("npp");
72         String page = request.getParameter("page");
73         String genre = request.getParameter("genre");
74         String st = request.getParameter("st");
75
76
77         while (gs.next()) {
78             JSONObject g = new JSONObject();
79             String name = gs.getString("name");
80             String gid = gs.getString("genreId");
81             g.addProperty("name", name);
82             g.addProperty("id", gid);
83             genreArray.add(g);
84         }
85         while (rs.next()) {
86

```

singleStarServlet.java: 58~109

```

58         PreparedStatement statement = connection.prepareStatement(query);
59
60
61         statement.setString(1, id);
62
63
64         ResultSet rs = statement.executeQuery();
65
66         JSONArray starArray = new JSONArray();
67         String title = request.getParameter("title");
68         String year = request.getParameter("year");
69         String director = request.getParameter("director");
70         String star = request.getParameter("star");
71         String sorting = request.getParameter("sorting");
72         String npp = request.getParameter("npp");
73         String page = request.getParameter("page");
74         String genre = request.getParameter("genre");
75         String st = request.getParameter("st");
76
77
78         while (rs.next()) {
79
80             String starId = rs.getString("starId");
81             String starName = rs.getString("name");
82             String starDob = rs.getString("birthYear");
83
84             String movieId = rs.getString("movieId");
85             String movieTitle = rs.getString("title");
86             String movieYear = rs.getString("year");
87             String movieDirector = rs.getString("director");
88

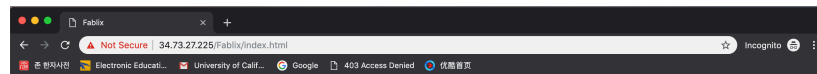
```

Task 2

- Address of AWS and Google instances
 AWS Instance 1: 52.53.239.70
 AWS Instance 2(Master): 52.53.210.87
 AWS Instance 3(Slave): 54.219.150.85
 GCP: 34.73.27.225

- Have you verified that they are accessible? Does Fablix site get opened both on Google's 80 port and AWS' 8080 port?
 - Yes, the port 80 for GCP and 80 for AWS Instance1 and 8080 for all 3 AWS instances are open and accessible as in the snapshot. And the name of the website is Fablix.

- google: url: <http://34.73.27.225/Fablix/index.html>



Welcome to Fablix!

[Cart](#)

[Search Page](#)

[Browsing Page](#)

[Dashboard Page](#)

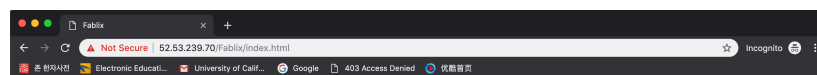
-
-
-
-
-
-
-
-
-

<input type="text" value="knight"/>	<input type="button" value="Search"/>
Knight Club	
Don Quixote , Knight Errant	
The Red Knight	
The Dark Knight	

AWS

url: <http://52.53.239.70/Fablix/index.html>

Instance 1: 80



Welcome to Fablix!

[Cart](#)

[Search Page](#)

[Browsing Page](#)

[Dashboard Page](#)

-
-
-
-
-
-
-
-
-
-
-
-
-
-

<input type="text" value="long"/>	<input type="button" value="Search"/>
Long Lost Love	
Longshot	
The Long Run	
Lives No Longer Ours	

AWS Instance 1: 8080 url: <http://52.53.239.70:8080/Fablix/index.html>

-

-
-

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

There are 2 resources added into context.xml, one is called testDB and the other one is called write, the testDB has the ip address of localhost, which means whenever the backend is connected and need to read from the database, it will always connect to its own database, but when a backend is connected and when it needs to write something into database, it will always connect to master's database and in order to do this, I also did grant the user on slave to access the database on master.

- File name, line numbers as in Github
project5/WebContent/META-INF/context.xml 15-23
project5/WebContent/WEB-INF/web.xml 21-40

- Snapshots
- context.xml

```
<Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
    maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="CS122B"
    password="CS122B" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&useSSL=false&cachePrepStmts=true"/>

<Resource name="jdbc/write" auth="Container" type="javax.sql.DataSource"
    maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="CS122B"
    password="CS122B" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://172.31.0.184:3306/moviedb?autoReconnect=true&useSSL=false&cachePrepStmts=true"/>
```

- web.xml

```
<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>

<resource-ref>
<description>
    write through master
</description>
<res-ref-name>jdbc/write</res-ref-name>
<res-type>javax.sql.DataSource</res-type>
<res-auth>Container</res-auth>
</resource-ref>
```

- How read/write requests were routed?
 - When a servlet needs to do a write, it will always look up the "write" connection in the context.xml to connect to database only on master. When a servlet needs to

do a read, it will look up the "testDB" in the context.xml to connect to database on either master or slave

- File name, line numbers as in Github
/project5/src/makechange.java 50-55
/project5/src/add_movie.java 49-56
/project5/src/completecheckoutServlet.java 68-74
- Snapshots
- makechange.java 50-55

```
47         try {
48             Context initCtx = new InitialContext();
49
50             Context envCtx = (Context) initCtx.lookup("java:comp/env");
51             if (envCtx == null)
52                 System.out.println("envCtx is NULL");
53
54             // Look up our data source
55             DataSource ds = (DataSource) envCtx.lookup("jdbc/write");
56             Connection connection = ds.getConnection();
```

add_movie.java 49-56

```
47         try {
48
49             Context initCtx = new InitialContext();
50
51             Context envCtx = (Context) initCtx.lookup("java:comp/env");
52             if (envCtx == null)
53                 System.out.println("envCtx is NULL");
54
55             // Look up our data source
56             DataSource ds = (DataSource) envCtx.lookup("jdbc/write");
57             Connection connection = ds.getConnection();
```

completecheckoutServlet.java 68-74

```
65         try {
66             Context initCtx = new InitialContext();
67
68             Context envCtx = (Context) initCtx.lookup("java:comp/env");
69             if (envCtx == null)
70                 System.out.println("envCtx is NULL");
71
72             // Look up our data source
73             DataSource ds = (DataSource) envCtx.lookup("jdbc/write");
74             Connection connection = ds.getConnection();
```

Task 3

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

Yes, it is in the Time measure folder, and the log file for every test is in its own folder

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

Yes, it is in the Time measure folder

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

Yes, it is in the cs122b-winter19-team-2/Time measure/calTime.py

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

yes, the war file is in the root of our team repo, called Fablix.war