Project 5 Task 1, 2 Report

Team 108

Task1 : Connection Pooling

- File Path & Line Number

File Path	Line Number
/cs122b-winter19-team-108/project5/src/_dashboardServlet.java	59~72
/cs122b-winter19-team-108/project5/src/addMovieServlet.java	59~72
/cs122b-winter19-team-108/project5/src/addStarServlet.java	54~67
/cs122b-winter19-team-108/project5/src/checkoutServlet.java	61~74
/cs122b-winter19-team-108/project5/src/ConfirmServlet.java	58~71
/cs122b-winter19-team-108/project5/src/LoginServlet.java	75~88
/cs122b-winter19-team-108/project5/src/MainServlet.java	49~62
/cs122b-winter19-team-108/project5/src/metaDataServlet.java	49~62
/cs122b-winter19-team-108/project5/src/metaTableServlet.java	44~57
/cs122b-winter19-team-108/project5/src/MovieServlet.java	136~149
/cs122b-winter19-team-108/project5/src/SingleStarServlet.java	47~60
/cs122b-winter19-team-108/project5/src/SingleMovieServlet.java	45~58

- How is it implemented

First of all, I change jdbc connection URL in the \METAINF\ context.xml. Secondly, I implemented *javax.naming.Context.lookup* to lookup on “jdbc/moviedb” data source. Lastly, I use *Datasource.getConnection() to* connect the dataset. This code is illustrated in the following screenshot.

These a a few lines of code implements connections-reusing when future requests to the database are required. It could enhance the performance by cutting down the time to re-establish the connection.

- Screenshot

TomcatPoolingServlet.java | web.xml | web.xml | context.xml | MovieServlet.java

```
125         }
126         else if(sort.equals("rating_down")) {
127             searchStr="SELECT * FROM "+"("+searchStr+") AS n ORDER BY n.rating ASC";
128         }
129
130         searchStr="SELECT * FROM "+"("+searchStr+") AS n LIMIT ? OFFSET ?";
131         /*
132         System.out.println("Search result");
133         System.out.println(searchStr);*/
134         try {
135             // Get a connection from dataSource
136             Context initCtx = new InitialContext();
137
138             Context envCtx = (Context) initCtx.lookup("java:comp/env");
139             if (envCtx == null)
140                 response.getWriter().println("envCtx is NULL");
141
142             // Look up our data source
143             DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
144             if (ds == null)
145                 response.getWriter().println("ds is null.");
146
147             Connection dbcon = ds.getConnection();
148             if (dbcon == null)
149                 response.getWriter().println("dbcon is null.");
150
```

MovieServlet.java

TomcatPoolingServlet.java | web.xml | web.xml | context.xml | addStarServlet.java | _dashboardServlet.java

```
44          * This example only allows username/password to be anteater/123456
45          * In real world projects, you should talk to the database to verify username/password
46          */
47
48         int loginStatus = 2; // 0: correct, 1: username not match, 2: password not match
49
50         PreparedStatement userNameStr = null;
51         String selectString = "SELECT e.password FROM `employees` e WHERE e.email = ?";
52
53
54
55         try {
56             // the following few lines are for connection pooling
57             // Obtain our environment naming context
58
59             Context initCtx = new InitialContext();
60
61             Context envCtx = (Context) initCtx.lookup("java:comp/env");
62             if (envCtx == null)
63                 response.getWriter().println("envCtx is NULL");
64
65             // Look up our data source
66             DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
67             if (ds == null)
68                 response.getWriter().println("ds is null.");
69
70             Connection dbcon = ds.getConnection();
71             if (dbcon == null)
72                 response.getWriter().println("dbcon is null.");
73
```

_dashBoardServlet.java

TomcatPoolingServlet.java | web.xml | web.xml | context.xml | addStarServlet.java | checkoutServlet.java

```
49          System.out.println(firstName);
50          System.out.println(ccid);
51          System.out.println(expiration);*/
52
53
54          HttpSession session = request.getSession();
55          HashMap<String, Integer> m = (HashMap<String, Integer>) session.getAttribute("itemMap");
56
57
58          int transacionStatus = 0; // 0: correct, 1: username not match, 3: card info not match
59
60          try {
61              Context initCtx = new InitialContext();
62
63              Context envCtx = (Context) initCtx.lookup("java:comp/env");
64              if (envCtx == null)
65                  response.getWriter().println("envCtx is NULL");
66
67              // Look up our data source
68              DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
69              if (ds == null)
70                  response.getWriter().println("ds is null.");
71
72              Connection dbcon = ds.getConnection();
73              if (dbcon == null)
74                  response.getWriter().println("dbcon is null.");
```

checkoutServlet.java

TomcatPoolingServlet.java | web.xml | web.xml | context.xml | addStarServlet.java

project5/src/addStarServlet.java

```
44          String birth = request.getParameter("birth");
45
46          System.out.println(starName);
47          System.out.println(birth);
48
49
50          HttpSession session = request.getSession();
51
52
53          try {
54              Context initCtx = new InitialContext();
55
56              Context envCtx = (Context) initCtx.lookup("java:comp/env");
57              if (envCtx == null)
58                  response.getWriter().println("envCtx is NULL");
59
60              // Look up our data source
61              DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
62              if (ds == null)
63                  response.getWriter().println("ds is null.");
64
65              Connection dbcon = ds.getConnection();
66              if (dbcon == null)
67                  response.getWriter().println("dbcon is null.");
```

addStarServlet.java

Task1 : Prepared Statements

- File Path & Line Number

File Path	Line Number
/cs122b-winter19-team-108/project5/src/_dashboardServlet.java	74~79
/cs122b-winter19-team-108/project5/src/addStarServlet.java	72~76
/cs122b-winter19-team-108/project5/src/checkoutServlet.java	77~87
/cs122b-winter19-team-108/project5/src/ConfirmServlet.java	73~86
/cs122b-winter19-team-108/project5/src/LoginServlet.java	90~95
/cs122b-winter19-team-108/project5/src/MainServlet.java	63~74
/cs122b-winter19-team-108/project5/src/metaDataServlet.java	64~72
/cs122b-winter19-team-108/project5/src/metaTableServlet.java	61~71
/cs122b-winter19-team-108/project5/src/MovieServlet.java	151~153, 215,216, 248,249
/cs122b-winter19-team-108/project5/src/SingleStarServlet.java	66~73
/cs122b-winter19-team-108/project5/src/SingleMovieServlet.java	66~76

- How is it implemented

First of all, I declare *PreparedStatement* object. Secondly, I disable auto commit to manually control the commit execution. Next, I initiate *PreparedStatement* object by taking string of query as inputs. If there are some unknown parameters in the query, I would set the designated parameters by respective methods (ex: setString, setInt). Then, I would execute the prepared statements to retrieve result set or int. Lastly, I would commit to make all changes. This code is illustrated in the following screenshot.

These a a few lines of code implements statements precompiling. This methods could be used to efficiently execute this statement multiple times, and avoid SQL injection attacks.

- Screenshot

TomcatPoolingServlet.java | web.xml | web.xml | context.xml | MovieServlet.java | metaTableServlet

```
            Context envCtx = (Context) initCtx.lookup("java:comp/env");
            if (envCtx == null)
                response.getWriter().println("envCtx is NULL");

            // Look up our data source
            DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
            if (ds == null)
                response.getWriter().println("ds is null.");

            Connection dbcon = ds.getConnection();
            if (dbcon == null)
                response.getWriter().println("dbcon is null.");

            String showTableStr = "SHOW COLUMNS FROM " + tableName;

            dbcon.setAutoCommit(false);
            showStatement = dbcon.prepareStatement(showTableStr);


            System.out.println(showStatement);

            JsonArray jsonArray = new JsonArray();

            // Perform the query
            ResultSet rs = showStatement.executeQuery();
            dbcon.commit();
```

metaTableServlet.java

TomcatPoolingServlet.java | web.xml | web.xml | context.xml | addStarServlet.java

```
        try {
            Context initCtx = new InitialContext();

            Context envCtx = (Context) initCtx.lookup("java:comp/env");
            if (envCtx == null)
                response.getWriter().println("envCtx is NULL");

            // Look up our data source
            DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
            if (ds == null)
                response.getWriter().println("ds is null.");

            Connection dbcon = ds.getConnection();
            if (dbcon == null)
                response.getWriter().println("dbcon is null.");

            String query = "SELECT max(id) AS mx FROM stars";


            dbcon.setAutoCommit(false);
            PreparedStatement statement = dbcon.prepareStatement(query);

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

addStarServlet.java

TomcatPoolingServlet.java | web.xml | web.xml | context.xml | addStarServlet.java | _dashboardServlet.java

```
    try {
        // the following few lines are for connection pooling
        // Obtain our environment naming context

        Context initCtx = new InitialContext();

        Context envCtx = (Context) initCtx.lookup("java:comp/env");
        if (envCtx == null)
            response.getWriter().println("envCtx is NULL");

        // Look up our data source
        DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
        if (ds == null)
            response.getWriter().println("ds is null.");

        Connection dbcon = ds.getConnection();
        if (dbcon == null)
            response.getWriter().println("dbcon is null.");

        dbcon.setAutoCommit(false);
        userNameStr = dbcon.prepareStatement(selectString);
        userNameStr.setString(1, username);

        ResultSet rs = userNameStr.executeQuery();
        dbcon.commit();
```

_dashBoardServlet.java

TomcatPoolingServlet.java | web.xml | web.xml | context.xml | addStarServlet.java | checkoutServlet.java

```
        int transacionStatus = 0; // 0: correct, 1: username not match, 3: card info not match

        try {
            Context initCtx = new InitialContext();

            Context envCtx = (Context) initCtx.lookup("java:comp/env");
            if (envCtx == null)
                response.getWriter().println("envCtx is NULL");

            // Look up our data source
            DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
            if (ds == null)
                response.getWriter().println("ds is null.");

            Connection dbcon = ds.getConnection();
            if (dbcon == null)
                response.getWriter().println("dbcon is null.");

            // prepare string and statement
            PreparedStatement statement = null;
            String query = "SELECT * FROM (SELECT c.firstName, c.lastName, c.ccId,  cc.expiration FROM

            dbcon.setAutoCommit(false);
            statement = dbcon.prepareStatement(query);
            statement.setString(1, firstName);
            statement.setString(2, lastName);

            // execute the statement
            ResultSet rs = statement.executeQuery();
            dbcon.commit();
```

checkoutServlet.java

Task2 : How connection pooling works with 2 MySQL

If we don't consider read/write separation, load balancer would choose 1 instance for Fabflix functionality, and it would perform connections-reusing with a specific mysql database from all the instances.

The following 2 screenshot are examples of using connection pooling to connect to local mysql database. The first one specifies "localhost" at url attribute since it wants to connect to local mysql database. It indicates that instance 2 would connect to MySQL database on instance 2 while instance 3 would connect to MySQL database on instance 3. As for the second screenshot from *loginservlet.java,* we perform connection pooling with this datasource "*moviedb*".

```
<Context>

    <!-- Defines a Data Source Connecting to localhost moviedb-->
    <Resource name="jdbc/moviedb" 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&amp;cachePrepStmts=true"/>
```

```
try {
    // the following few lines are for connection pooling
    // Obtain our environment naming context

    Context initCtx = new InitialContext();

    Context envCtx = (Context) initCtx.lookup("java:comp/env");
    if (envCtx == null)
        response.getWriter().println("envCtx is NULL");

    // Look up our data source
    DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
    if (ds == null)
        response.getWriter().println("ds is null.");

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

Of course, we could specify 1 instance such that it performs connection pooling with multiple MySQL database in different cases. It would be discussed in the next part.

- File Path & Line Number

  I only specify *context.xml* since the other location of codes are same as connection pooling in task 1.

File Path	Line Number
/cs122b-winter19-team-108/project5/Webcontent/META-INF/context.xml	6~10

Task2 : How to direct read/write requests

To direct read/write requests, we need another connection source in *context.xml.* The first one deal with read requests and connect to local MySQL database while the second one deal with write requests and connect to database at master instance. The following figure is a screenshot that how I configure these 2 connections. The obvious different is that I specify public IP of master instance in the second data source to

connect to master instance on write request. As for the first connection source, it only connects to local database.

```
<?xml version="1.0" encoding="UTF-8"?>

<Context>

    <!-- Defines a Data Source Connecting to localhost moviedb-->
    <Resource name="jdbc/moviedb" 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&amp;cachePrepStmts=true"/>


    <Resource name="jdbc/write_moviedb" auth="Container" type="javax.sql.DataSource"
              maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
              password="mypassword" driverClassName="com.mysql.jdbc.Driver"
              url="jdbc:mysql://18.221.18.233:3306/moviedb?autoReconnect=true&amp;cachePrepStmts=true"/>

</Context>
```

The next question is how could we implement in our code. What we need to do is retrieving different naming object by looking up different data source name at write requests. The following figure is an example on *addMovieServlet.java.* I look up the second data source *"write_moviedb"* instead of "*moviedb*" since I want to insert movie informations into database of master instance.

```
try {
    // the following few lines are for connection pooling
    // Obtain our environment naming context

    Context initCtx = new InitialContext();

    Context envCtx = (Context) initCtx.lookup("java:comp/env");
    if (envCtx == null)
        response.getWriter().println("envCtx is NULL");

    // Look up our data source
    DataSource ds = (DataSource) envCtx.lookup("jdbc/write_moviedb");
    if (ds == null)
        response.getWriter().println("ds is null.");

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

- File Path & Line Number

I specify *context.xml* and web.xml where I change the configuration for two connections. As for codes, only three servlets perform write

requests and these are: *ConfirmServlet.java, addStarServlet.java, addMovieServlet.java*

File Path	Line Number
/cs122b-winter19-team-108/project5/Webcontent/META-INF/context.xml	6~16
/cs122b-winter19-team-108/project5/Webcontent/WEB-INF/web.xml	19~24
/cs122b-winter19-team-108/project5/src/addMovieServlet.java	59~72
/cs122b-winter19-team-108/project5/src/addStarServlet.java	54~67
/cs122b-winter19-team-108/project5/src/ConfirmServlet.java	58~71