PROJECT 5 REPORT

TASK1: JDBC connection pooling

DataBase Connection Configuration

- Prepared Statement: It is implemented in every servlet. The locations are mention below in the database connection pooling, the places are the same. In the project, preparedStatement is used to pre-compile the SQL sentences and process it afterwards. This will increase the efficiency when when the SQL is reused to fill some potential placeholders.
- Database connection is configured in the context.xml, see the link below.
- Database connection pooling is used to reuse the connection resources in a pool, this will save the cost of creating connections. It is configured in the context.xml, the link is listed below.

Places where database connection pooling is used

- context.xml:
    - line 18
    - line 23

https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/WebContent/META-INF/context.xml

```
    <!-- Previous resource without tomcat pooling -->
<!--     <Resource name="jdbc/moviedb"
              auth="Container"
              driverClassName="com.mysql.jdbc.Driver"
              type="javax.sql.DataSource"
              username="mytestuser"
              password="mypassword"
              url="jdbc:mysql://localhost:3306/moviedb"/> -->

     <!-- Resource definition with tomcat pooling -->

    <Resource name="jdbc/moviedb" auth="Container" type="javax.sql.DataSource"
              maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
              password="123456" driverClassName="com.mysql.jdbc.Driver"
              url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>

    <Resource name="jdbc/writedb" auth="Container" type="javax.sql.DataSource"
              maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
              password="mypassword" driverClassName="com.mysql.jdbc.Driver"
              url="jdbc:mysql://13.57.248.101:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
```

How is it used?

```
18  public class CardDaoImpl implements CardDao {
19
20          @Override
21          public CreditCard selectCardById(String cardId) {
22                  CreditCard creditCard = new CreditCard();
23                  try {
24                          Context initContext = new InitialContext();
25                          Context envContext = (Context) initContext.lookup("java:comp/env");
26                          DataSource ds = (DataSource) envContext.lookup("jdbc/moviedb");
27                          Connection dbcon = ds.getConnection();
28
29                          String findCreditCard = "Select * from creditcards where id = ?";
30
31                          PreparedStatement pstmt = dbcon.prepareStatement(findCreditCard);
32                          pstmt.setString(1, cardId);
33                          ResultSet result = pstmt.executeQuery();
34
35              while(result.next()) {
36                  creditCard.setId(result.getString(1));
37                  creditCard.setFirstName(result.getString(2));
38                  creditCard.setLastName(result.getString(3));
39                  creditCard.setExpiration(result.getDate(4));
40              }
41              result.close();
42              pstmt.close();
43              dbcon.close();
44                  } catch (Exception e) {
45                          e.printStackTrace();
46                  }
47                  return creditCard;
48          }
49
50  }
```

We search for a data source to search for an available connection. The general format of access the JDBC connection pool is the same.

Where tomcat pooling is used?

1.[CardDaoImpl.java](https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/src/com/cs122b/fablix/dao/Impl/CardDaoImpl.java)
- line 24

2.[CustomerDaoImpl.java](https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/src/com/cs122b/fablix/dao/Impl/CustomerDaoImpl.java)
- line 29
- line 57

3.[EmployeeDaoImpl](https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/src/com/cs122b/fablix/dao/Impl/EmployeeDaoImpl.java)
- line 25
- line 53

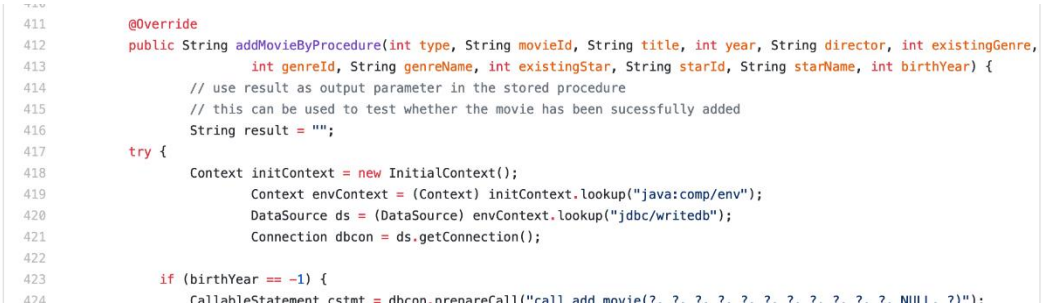
4.[FuzzySearchDaoImpl.java](https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/src/com/cs122b/fablix/dao/Impl/FuzzySearchDaoImpl.java)
- line 23

5.[GenreDaoImpl.java](https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/src/com/cs122b/fablix/dao/Impl/GenreDaoImpl.java)
- line 26
- line 59
- line 86
- line 117
- line 148

6.[MetaDaoImpl.java](https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/src/com/cs122b/fablix/dao/Impl/MetadataImpl.java)
- line 23

7.[MovieDaoImpl.java](https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/src/com/cs122b/fablix/dao/Impl/MovieDaoImpl.java)
- line 28
- line 62
- line 96
- line 134
- line 165
- line 213
- line 286
- line 329
- line 360
- line 392
- line 418
- line 473
- line 502

- line 555



```
411         @Override
412         public String addMovieByProcedure(int type, String movieId, String title, int year, String director, int existingGenre,
413                         int genreId, String genreName, int existingStar, String starId, String starName, int birthYear) {
414                 // use result as output parameter in the stored procedure
415                 // this can be used to test whether the movie has been sucessfully added
416                 String result = "";
417         try {
418                 Context initContext = new InitialContext();
419                         Context envContext = (Context) initContext.lookup("java:comp/env");
420                         DataSource ds = (DataSource) envContext.lookup("jdbc/writedb");
421                         Connection dbcon = ds.getConnection();
422
423             if (birthYear == -1) {
424                 CallableStatement cstmt = dbcon.prepareCall("call add_movie(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, NULL, ?)");
```

8.[RatingDaoImpl](https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/src/com/cs122b/fablix/dao/Impl/RatingDaoImpl.java)
- line 21
- line 47

9.[SalesDaoImpl.java](https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/src/com/cs122b/fablix/dao/Impl/SalesDaoImpl.java)
- line 29

10.[StarDaoImpl.java](https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/src/com/cs122b/fablix/dao/Impl/StarDaoImpl.java)
- line 26
- line 55
- line 81
- line 106
- line 126
- line 155

TASK 2:

Google cloud platform ip: http://35.236.56.154/cs122b-project/page/login.html
AWS platform ip: http://52.53.158.173/cs122b-project/page/login.html
instance2: http://13.57.248.101:8080/cs122b-project/page/login.html
instance3: http://54.153.23.28:8080/cs122b-project/page/login.html

Verified the instance: Right, Fablix site get opened both on Google's 80 port and AWS' 8080 port.

Connection pooling with two backend servers
- context.xml:
    - line 18
    - line 23

https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/WebContent/META-INF/context.xml

How read/write requests were routed?

- The load balancer will be evenly distributed the requests to two backend servers. When a read request hit a server, it will use the local database connection to request data. If it is a write request, it will be send to master instance(instance 2).

```
    <!-- Previous resource without tomcat pooling -->
<!--    <Resource name="jdbc/moviedb"
            auth="Container"
            driverClassName="com.mysql.jdbc.Driver"
            type="javax.sql.DataSource"
            username="mytestuser"
            password="mypassword"
            url="jdbc:mysql://localhost:3306/moviedb"/> -->

     <!-- Resource definition with tomcat pooling -->

    <Resource name="jdbc/moviedb" auth="Container" type="javax.sql.DataSource"
            maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
            password="123456" driverClassName="com.mysql.jdbc.Driver"
            url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>

    <Resource name="jdbc/writedb" auth="Container" type="javax.sql.DataSource"
            maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
            password="mypassword" driverClassName="com.mysql.jdbc.Driver"
            url="jdbc:mysql://13.57.248.101:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
```

- Read requests

```
public class CardDaoImpl implements CardDao {

        @Override
        public CreditCard selectCardById(String cardId) {
                CreditCard creditCard = new CreditCard();
                try {
                        Context initContext = new InitialContext();
                        Context envContext = (Context) initContext.lookup("java:comp/env");
                        DataSource ds = (DataSource) envContext.lookup("jdbc/moviedb");
                        Connection dbcon = ds.getConnection();

                        String findCreditCard = "Select * from creditcards where id = ?";

                        PreparedStatement pstmt = dbcon.prepareStatement(findCreditCard);
                        pstmt.setString(1, cardId);
                        ResultSet result = pstmt.executeQuery();

            while(result.next()) {
                creditCard.setId(result.getString(1));
                creditCard.setFirstName(result.getString(2));
                creditCard.setLastName(result.getString(3));
                creditCard.setExpiration(result.getDate(4));
            }
            result.close();
            pstmt.close();
            dbcon.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
                return creditCard;
        }

}
```

- Write request example

- https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/src/com/cs122b/fablix/dao/Impl/SalesDaoImpl.java: line 31
- https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/src/com/cs122b/fablix/dao/Impl/MovieDaoImpl.java: line 418

```
        @Override
        public String addMovieByProcedure(int type, String movieId, String title, int year, String director, int existingGenre,
                        int genreId, String genreName, int existingStar, String starId, String starName, int birthYear) {
                // use result as output parameter in the stored procedure
                // this can be used to test whether the movie has been sucessfully added
                String result = "";
        try {
                Context initContext = new InitialContext();
                        Context envContext = (Context) initContext.lookup("java:comp/env");
                        DataSource ds = (DataSource) envContext.lookup("jdbc/writedb");
                        Connection dbcon = ds.getConnection();

            if (birthYear == -1) {
                CallableStatement cstmt = dbcon.prepareCall("call add_movie(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, NULL, ?)");
```

- https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/cs122b-project/src/com/cs122b/fablix/dao/Impl/StarDaoImpl.java: line126

TASK 3:
log files:
https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/tree/master/Jmeter_Test_Result/LogFiles

HTML report of the Jmeter test:
https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/Jmeter_Test_Result/JmeterReport/jmeter_report.html

You can download the folder to view the report :
https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/tree/master/Jmeter_Test_Result/JmeterReport

Script for processing the log file:
https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/LogProcessor/src/LogProcessor.java

WAR files and README explanation for WAR files
- https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/tree/master/WAR_File
- https://github.com/UCI-Chenli-teaching/cs122b-winter19-team-10/blob/master/WAR_File/README.md