Task 1

- How did you use connection pooling?

The connections to MySQL instance is defined in /webapp/META-INF/context.xml for both master and slave. Inside servlets, I user Context.lookup() to find and declare MySQL instance I use(defined in the context.xml), and use datasource.getConnection() method to get the connection from the MySQL and close the connection after request end.

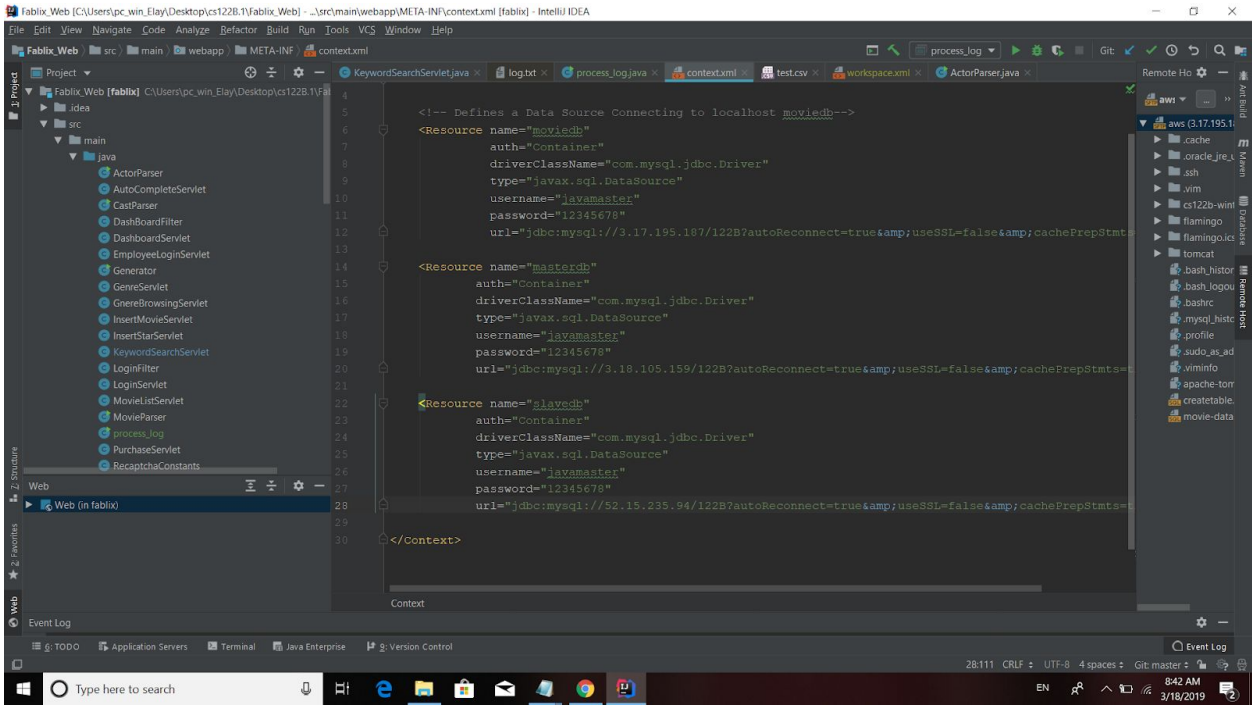
-File name, line numbers as in Github

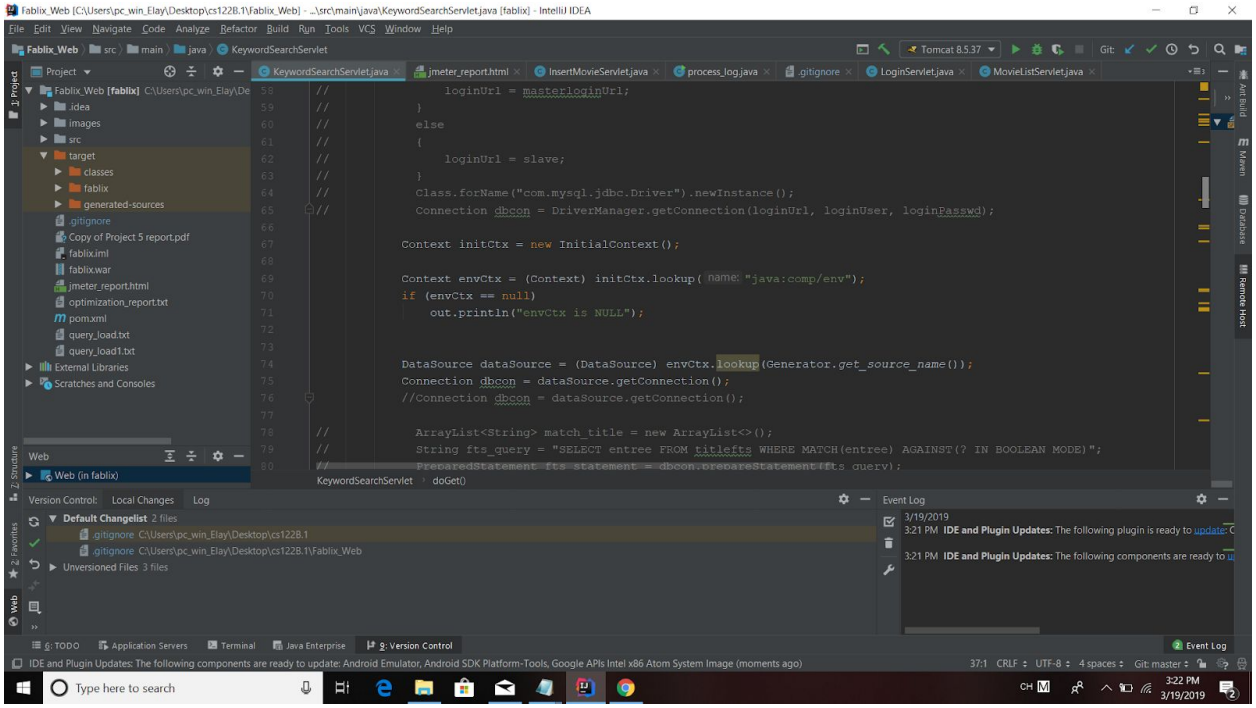
File name: context.xml(/Fablix_Web/src/main/webapp/META-INF/)
Line numbers: main instance: line 6-12; slave instance: line 14-20; master Instance:22-28

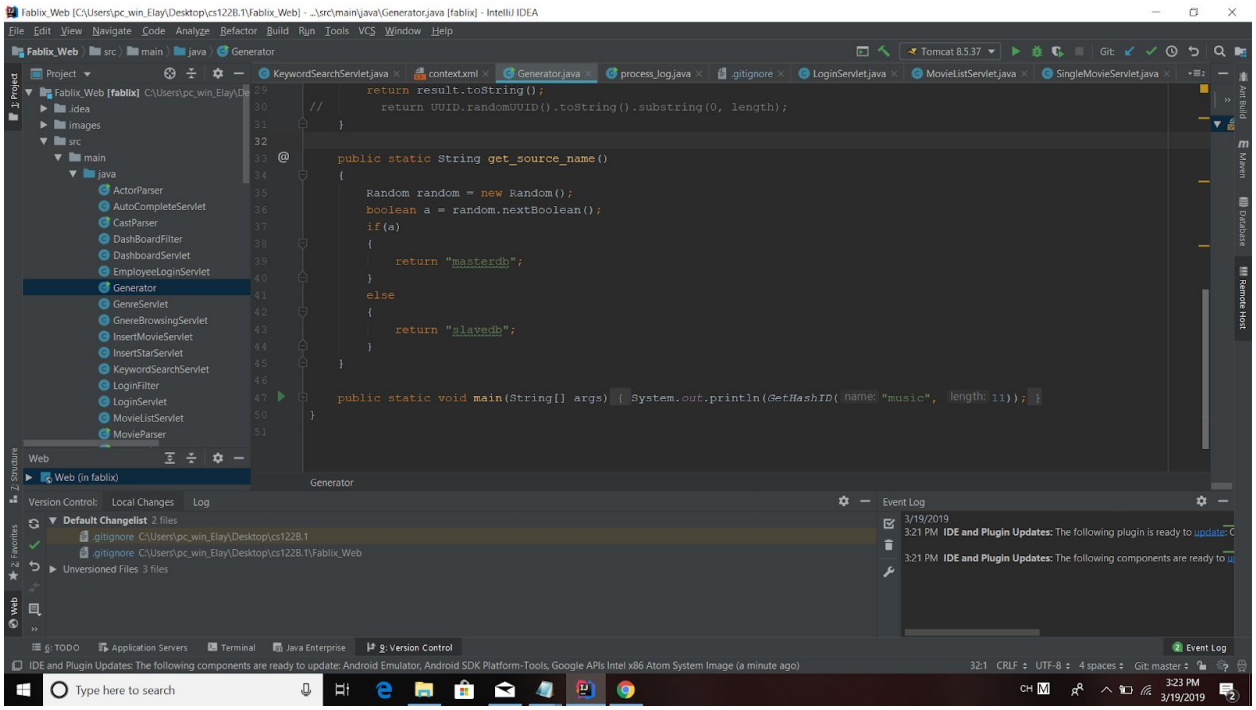
File name: KeywordSearchServlet.java(/Fablix_Web/src/main/java/)
Line numbers: declare the connection to the MySQL instance: line 67-75
(I applied connection pooling to all servlets which have interaction with MySQL instance)

File name: Generator.java(/Fablix_Web/src/main/java/)
Line number: line: 33-45 (randomly return the master/slave MySQL instance.)

- Snapshots showing use in your code







How did you use Prepared Statements?

For all servlets involved in search, I use the prepared statement. I construct basically the complete query and replace all user input parts with placeholders(“?”), and after I receive the user input, I user the setString(the_index_of_placeholder, user_input) to fill the SQL query.

- File name, line numbers as in Github

File name: KeywordSerachServlet.java(/Fablix_Web/src/main/java/)
Line numbers: line 94-120, line 149-156

File name: SesrchServlet.java(/Fablix_Web/src/main/java/)
Line numbers: line 42-71, line 96-101

File name: AutoCompleteServlet.java(/Fablix_Web/src/main/java)
Line numbers: line 47-56

File name: EmployeeLoginServlet.java(/Fablix_Web/src/main/java/)
Line numbers: line 81-83, line 113-115

File name: GnereServlet.java(/Fablix_Web/src/main/java/)
Line numbers: line 41-43(no placeholder)

File name: GnereBrowsingServlet(/Fablix_Web/src/main/java/)
Line numbers: line 49-64, line 91-95

FIle name: InsertMovieServlet(/Fablix_Web/src/main/java/)
Line numbers: line 42-48

File name: InsertStarServlet(/Fablix_Web/src/main/java/)
Line numbers: line 36-39

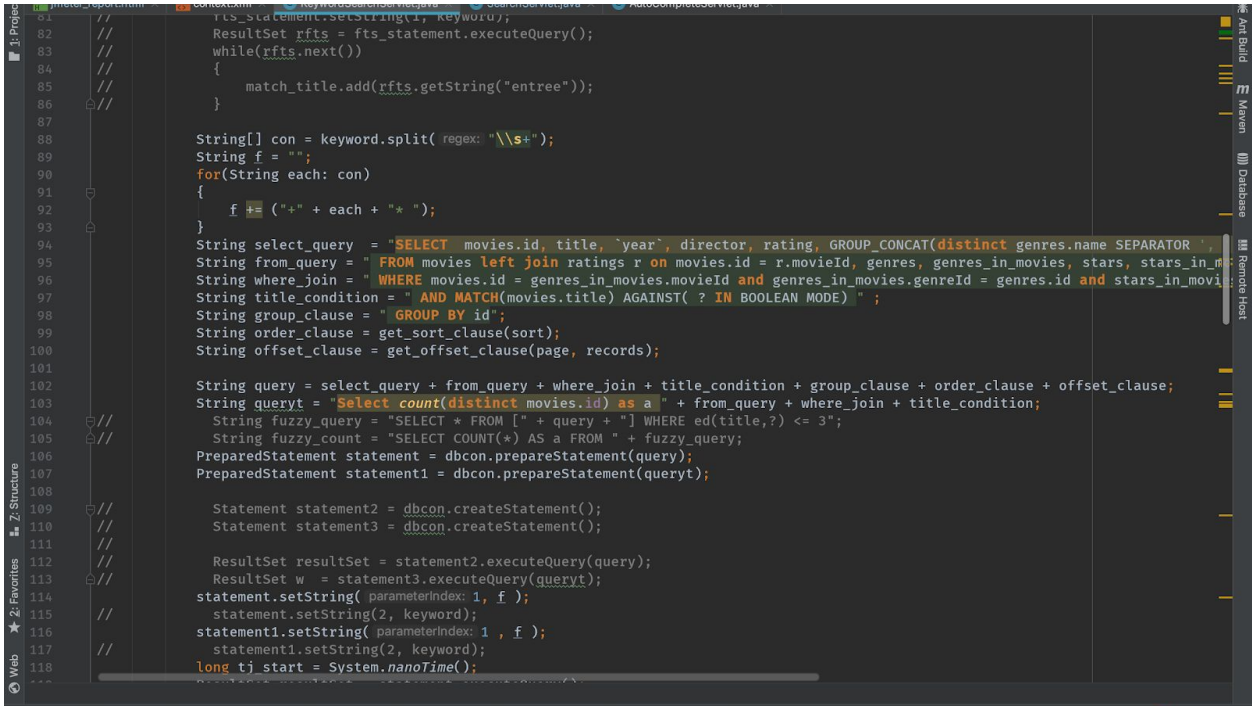
File name: LoginServlet.java(/Fablix_Web/src/main/java/)
Line numbers: line 95-97, 137-139

File name: SingleMovieServlet.java(/Fablix_Web/src/main/java/)
Line numbers: line 44-53, line 71-75

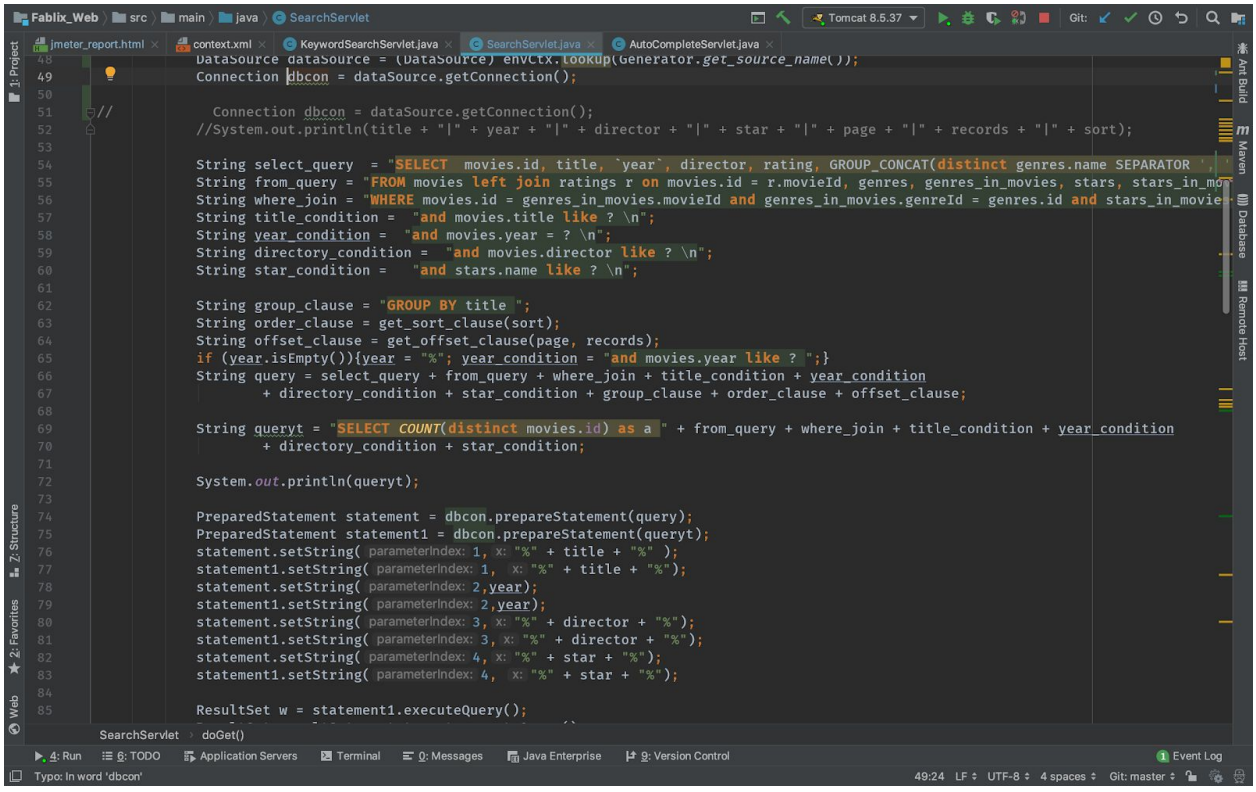
File name: SingleStarServlet.java(/Fablix_Web/src/main/java/)
Line number: line 43-50, line 63-67

File name: TitleBrowsingServlet.java(/Fablix_Web/src/main/java/)
Line number: line 49-62, line 92-96

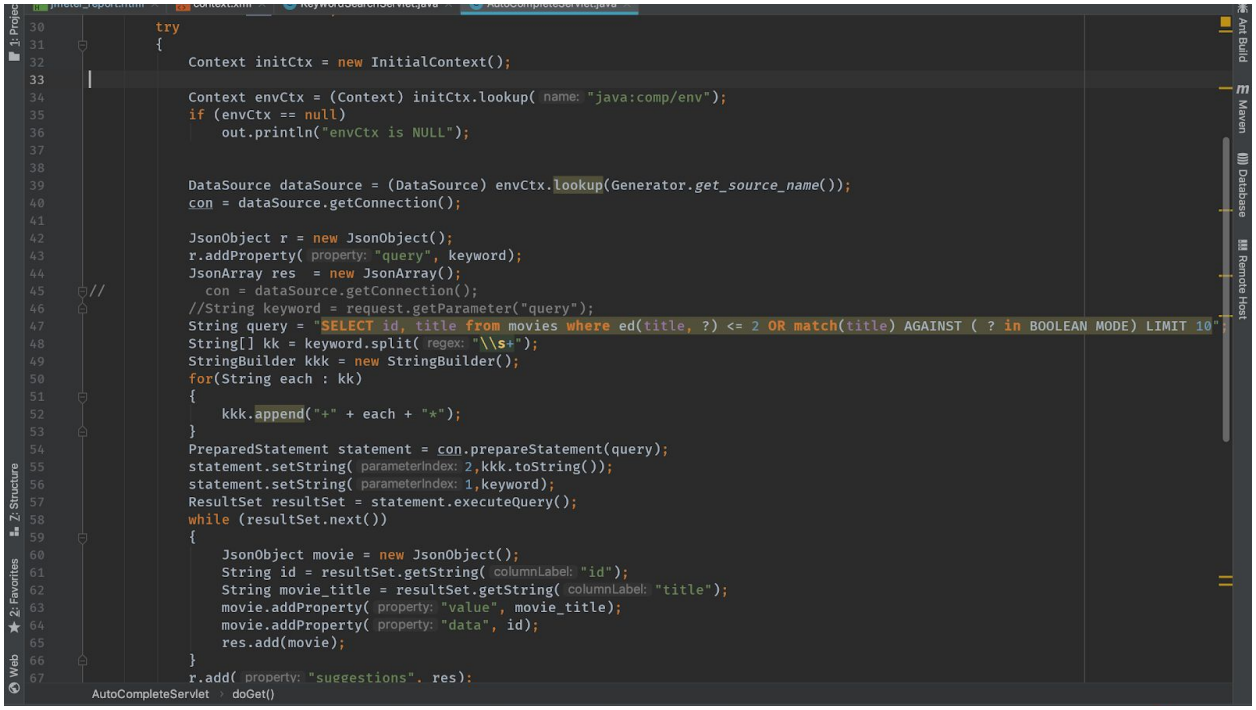
- Snapshots showing use in your code



```
//              rfts_statement.setString(1, keyword);
//              ResultSet rfts = fts_statement.executeQuery();
//              while(rfts.next())
//              {
//                  match_title.add(rfts.getString("entree"));
//              }

                String[] con = keyword.split( regex: "\\s+");
                String f = "";
                for(String each: con)
                {
                    f += ("+" + each + "* ");
                }
                String select_query  = "SELECT  movies.id, title, `year`, director, rating, GROUP_CONCAT(distinct genres.name SEPARATOR ',
                String from_query = " FROM movies left join ratings r on movies.id = r.movieId, genres, genres_in_movies, stars, stars_in_m
                String where_join = " WHERE movies.id = genres_in_movies.movieId and genres_in_movies.genreId = genres.id and stars_in_movi
                String title_condition = " AND MATCH(movies.title) AGAINST( ? IN BOOLEAN MODE) " ;
                String group_clause = " GROUP BY id";
                String order_clause = get_sort_clause(sort);
                String offset_clause = get_offset_clause(page, records);

                String query = select_query + from_query + where_join + title_condition + group_clause + order_clause + offset_clause;
                String queryt = "Select count(distinct movies.id) as a " + from_query + where_join + title_condition;
//              String fuzzy_query = "SELECT * FROM [" + query + "] WHERE ed(title,?) <= 3";
//              String fuzzy_count = "SELECT COUNT(*) AS a FROM " + fuzzy_query;
                PreparedStatement statement = dbcon.prepareStatement(query);
                PreparedStatement statement1 = dbcon.prepareStatement(queryt);

//              Statement statement2 = dbcon.createStatement();
//              Statement statement3 = dbcon.createStatement();
//
//              ResultSet resultSet = statement2.executeQuery(query);
//              ResultSet w  = statement3.executeQuery(queryt);
                statement.setString( parameterIndex: 1, f );
//              statement.setString(2, keyword);
                statement1.setString( parameterIndex: 1 , f );
//              statement1.setString(2, keyword);
                long tj_start = System.nanoTime();
```

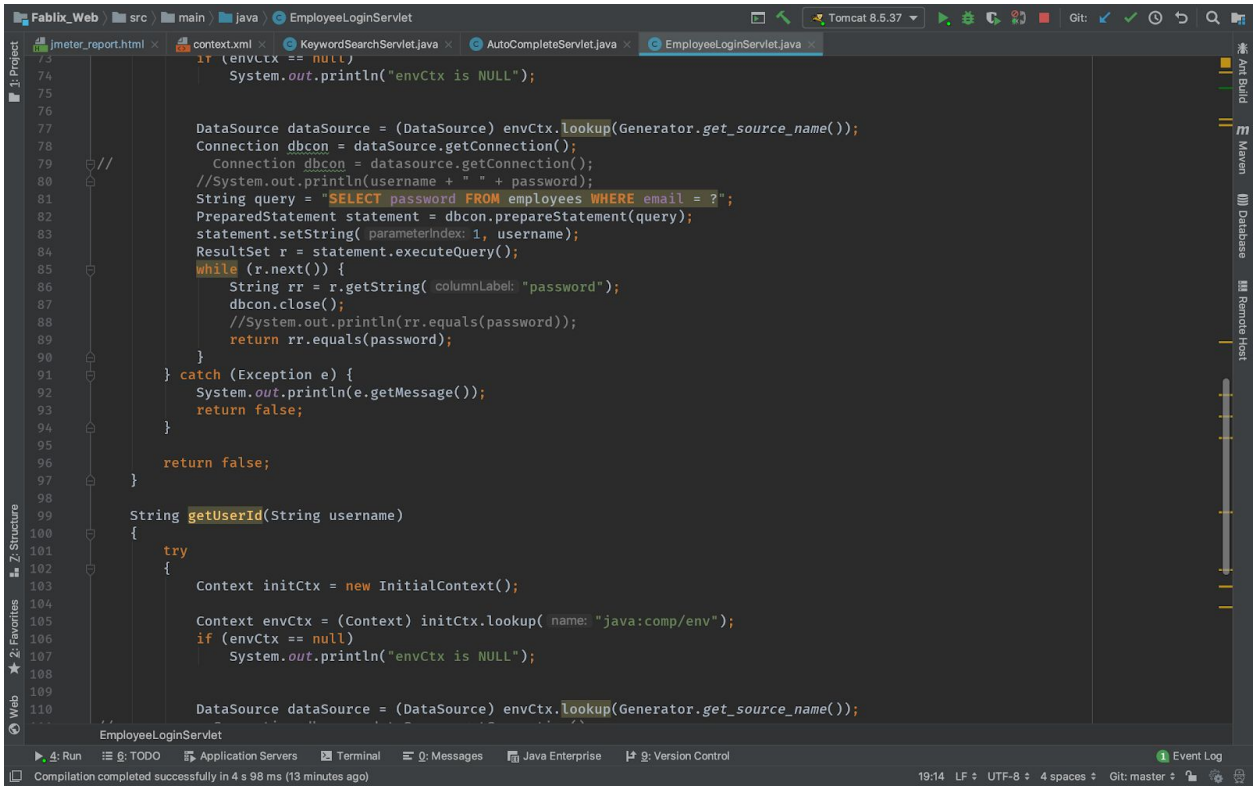


```
                DataSource dataSource = (DataSource) envCtx.lookup(Generator.get_source_name());
                Connection dbcon = dataSource.getConnection();

//              Connection dbcon = dataSource.getConnection();
                //System.out.println(title + "|" + year + "|" + director + "|" + star + "|" + page + "|" + records + "|" + sort);

                String select_query  = "SELECT  movies.id, title, `year`, director, rating, GROUP_CONCAT(distinct genres.name SEPARATOR ',
                String from_query = "FROM movies left join ratings r on movies.id = r.movieId, genres, genres_in_movies, stars, stars_in_mo
                String where_join = "WHERE movies.id = genres_in_movies.movieId and genres_in_movies.genreId = genres.id and stars_in_movie
                String title_condition =  "and movies.title like ? \n";
                String year_condition =  "and movies.year = ? \n";
                String directory_condition =  "and movies.director like ? \n";
                String star_condition =   "and stars.name like ? \n";

                String group_clause = "GROUP BY title ";
                String order_clause = get_sort_clause(sort);
                String offset_clause = get_offset_clause(page, records);
                if (year.isEmpty()){year = "%"; year_condition = "and movies.year like ? ";}
                String query = select_query + from_query + where_join + title_condition + year_condition
                        + directory_condition + star_condition + group_clause + order_clause + offset_clause;

                String queryt = "SELECT COUNT(distinct movies.id) as a " + from_query + where_join + title_condition + year_condition
                        + directory_condition + star_condition;

                System.out.println(queryt);

                PreparedStatement statement = dbcon.prepareStatement(query);
                PreparedStatement statement1 = dbcon.prepareStatement(queryt);
                statement.setString( parameterIndex: 1, x: "%" + title + "%" );
                statement1.setString( parameterIndex: 1,  x: "%" + title + "%");
                statement.setString( parameterIndex: 2,year);
                statement1.setString( parameterIndex: 2,year);
                statement.setString( parameterIndex: 3, x: "%" + director + "%");
                statement1.setString( parameterIndex: 3, x: "%" + director + "%");
                statement.setString( parameterIndex: 4, x: "%" + star + "%");
                statement1.setString( parameterIndex: 4,  x: "%" + star + "%");

                ResultSet w = statement1.executeQuery();
```



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

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


            DataSource dataSource = (DataSource) envCtx.lookup(Generator.get_source_name());
            con = dataSource.getConnection();

            JsonObject r = new JsonObject();
            r.addProperty( property: "query", keyword);
            JsonArray res  = new JsonArray();
//            con = dataSource.getConnection();
            //String keyword = request.getParameter("query");
            String query = "SELECT id, title from movies where ed(title, ?) <= 2 OR match(title) AGAINST ( ? in BOOLEAN MODE) LIMIT 10";
            String[] kk = keyword.split( regex: "\\s+");
            StringBuilder kkk = new StringBuilder();
            for(String each : kk)
            {
                kkk.append("+" + each + "*");
            }
            PreparedStatement statement = con.prepareStatement(query);
            statement.setString( parameterIndex: 2,kkk.toString());
            statement.setString( parameterIndex: 1,keyword);
            ResultSet resultSet = statement.executeQuery();
            while (resultSet.next())
            {
                JsonObject movie = new JsonObject();
                String id = resultSet.getString( columnLabel: "id");
                String movie_title = resultSet.getString( columnLabel: "title");
                movie.addProperty( property: "value", movie_title);
                movie.addProperty( property: "data", id);
                res.add(movie);
            }
            r.add( property: "suggestions", res);
```

AutoCompleteServlet › doGet()

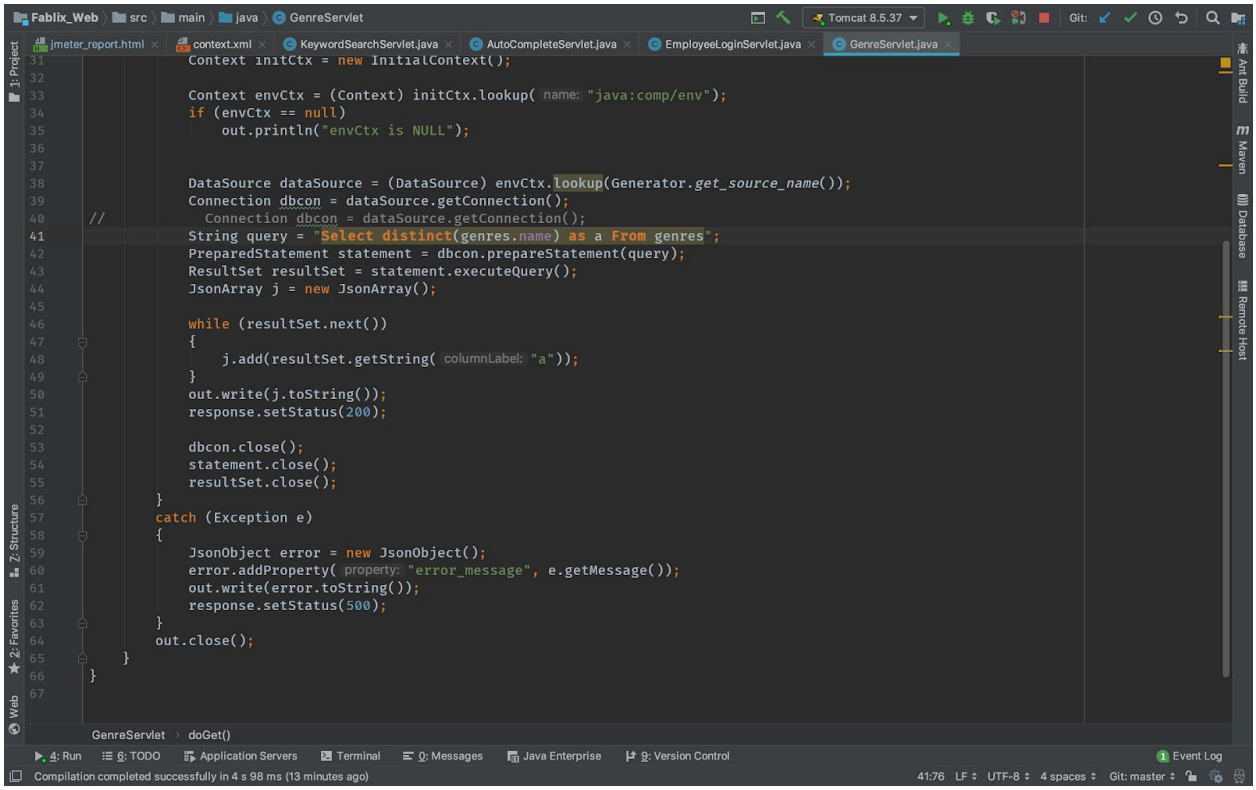


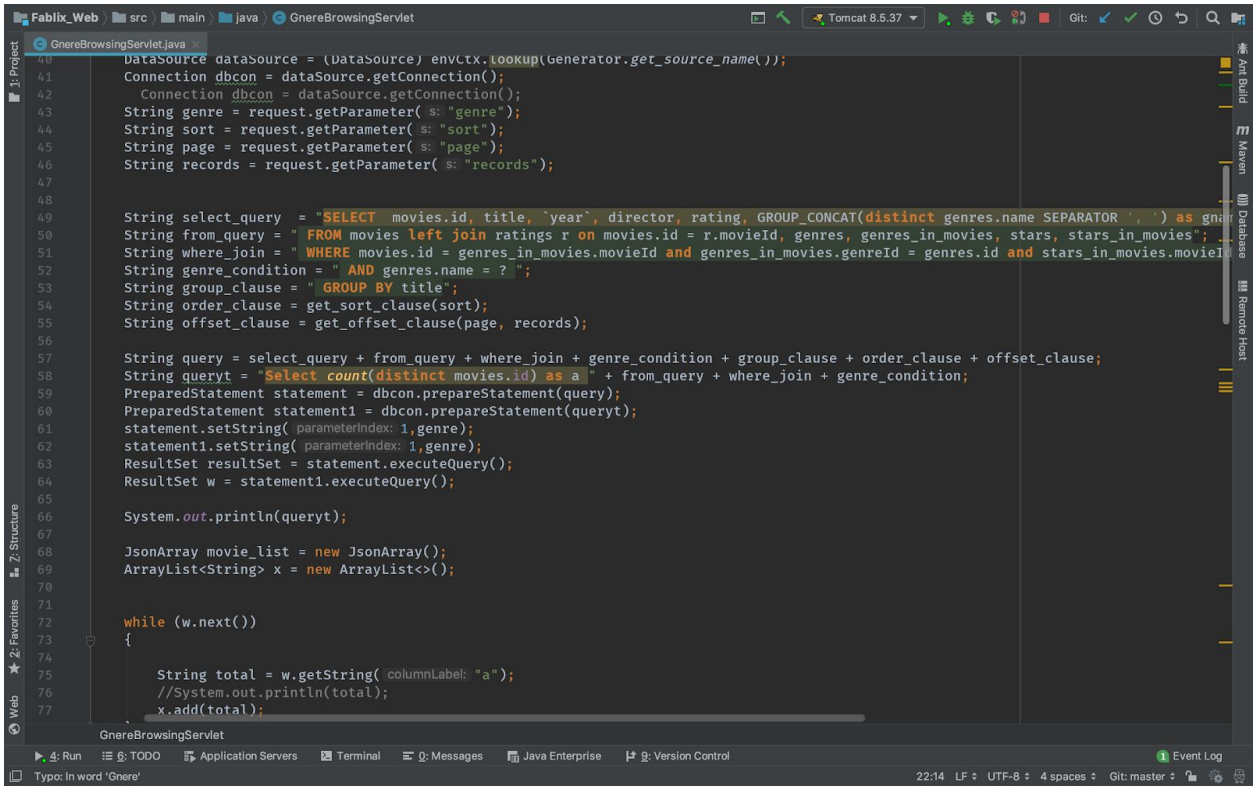
```
            if (envCtx == null)
                System.out.println("envCtx is NULL");


            DataSource dataSource = (DataSource) envCtx.lookup(Generator.get_source_name());
            Connection dbcon = dataSource.getConnection();
//            Connection dbcon = datasource.getConnection();
            //System.out.println(username + " " + password);
            String query = "SELECT password FROM employees WHERE email = ?";
            PreparedStatement statement = dbcon.prepareStatement(query);
            statement.setString( parameterIndex: 1, username);
            ResultSet r = statement.executeQuery();
            while (r.next()) {
                String rr = r.getString( columnLabel: "password");
                dbcon.close();
                //System.out.println(rr.equals(password));
                return rr.equals(password);
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
            return false;
        }

        return false;
    }

    String getUserId(String username)
    {
        try
        {
            Context initCtx = new InitialContext();

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


            DataSource dataSource = (DataSource) envCtx.lookup(Generator.get_source_name());
```

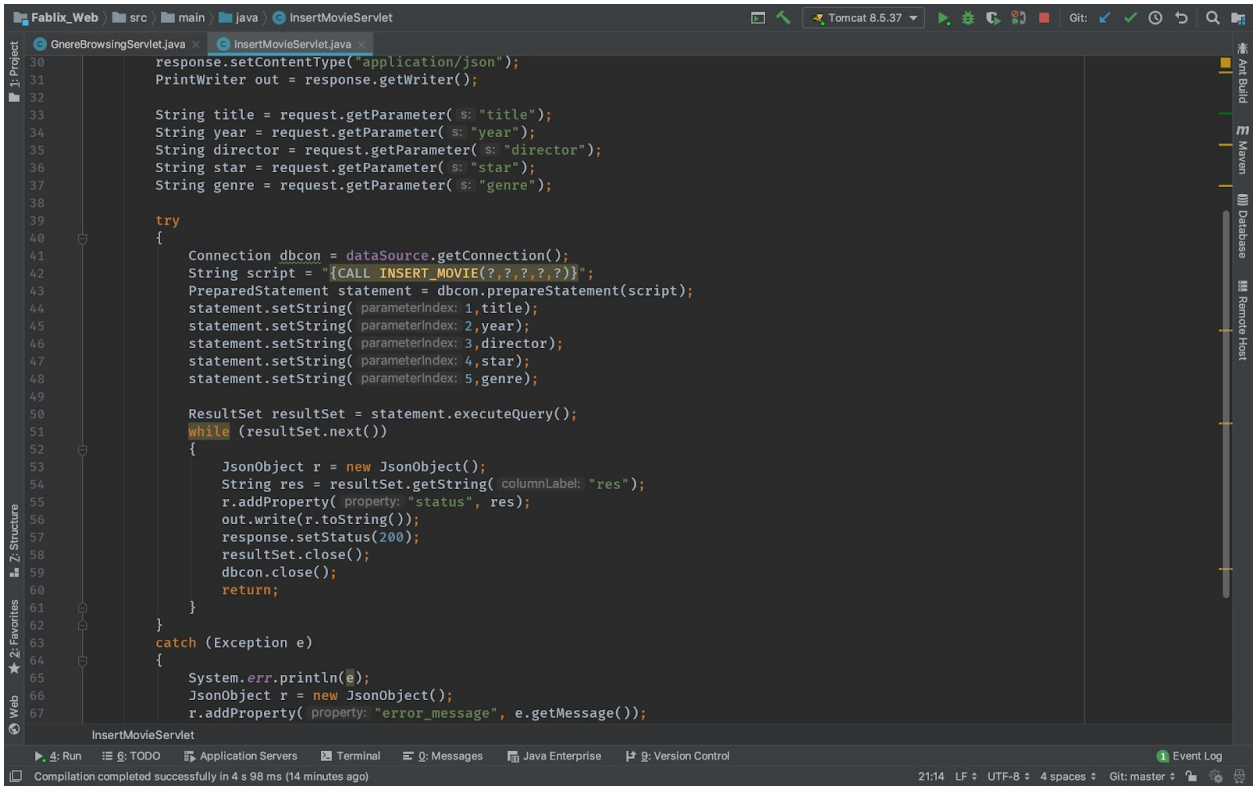
EmployeeLoginServlet

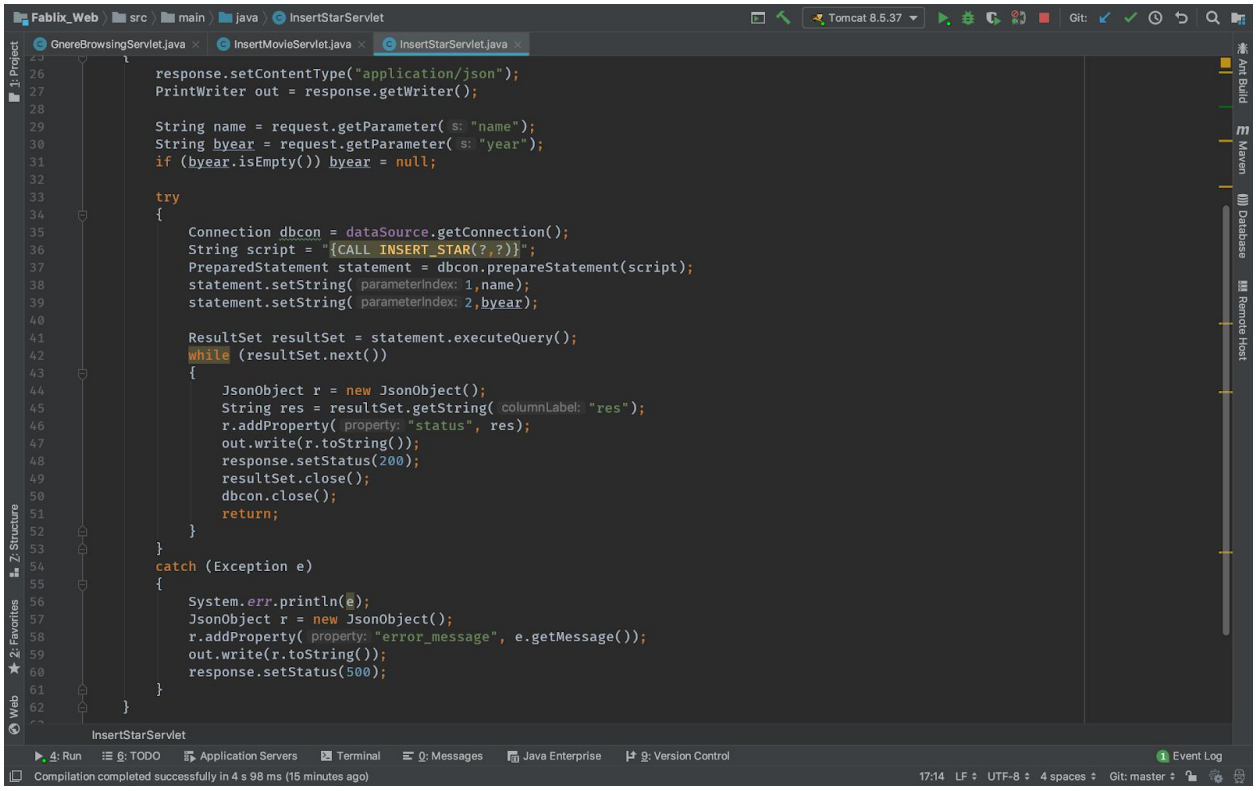
4: Run | 6: TODO | Application Servers | Terminal | 0: Messages | Java Enterprise | 9: Version Control | 1 Event Log

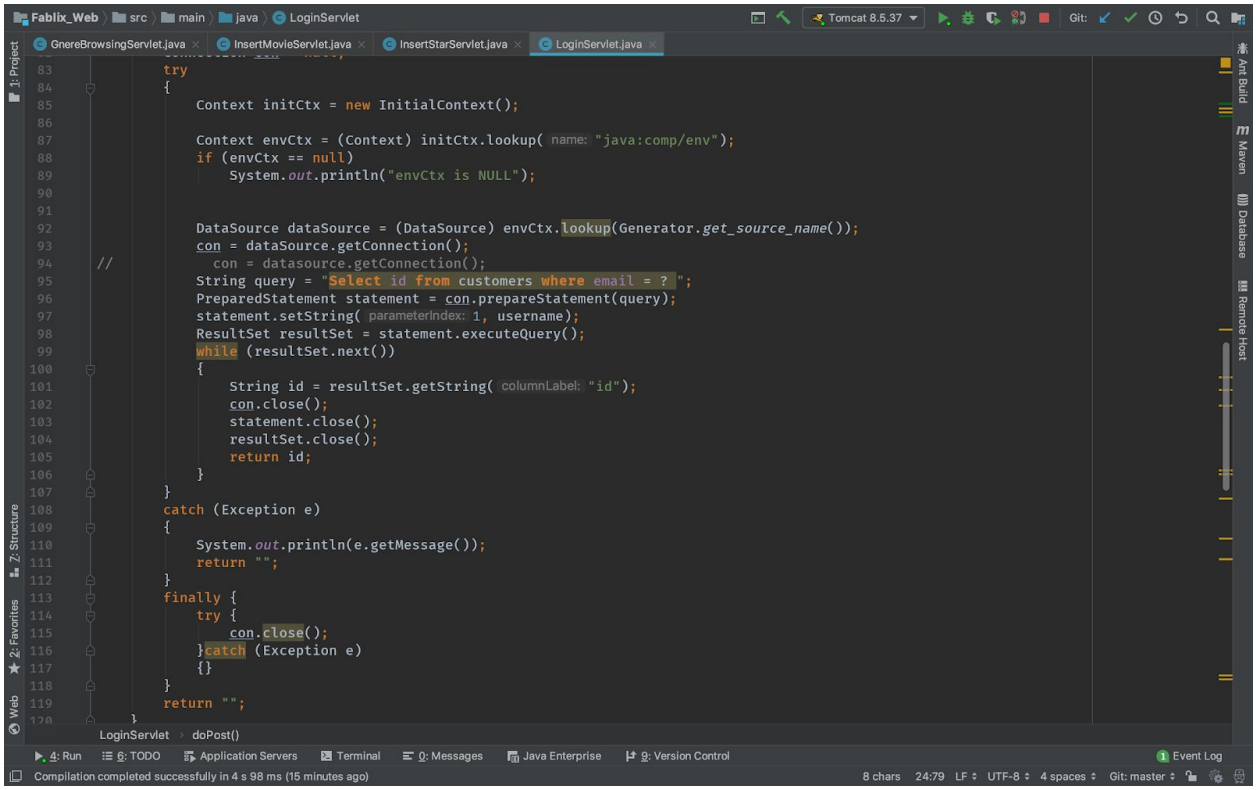
Compilation completed successfully in 4 s 98 ms (13 minutes ago) 19:14 LF UTF-8 4 spaces Git: master

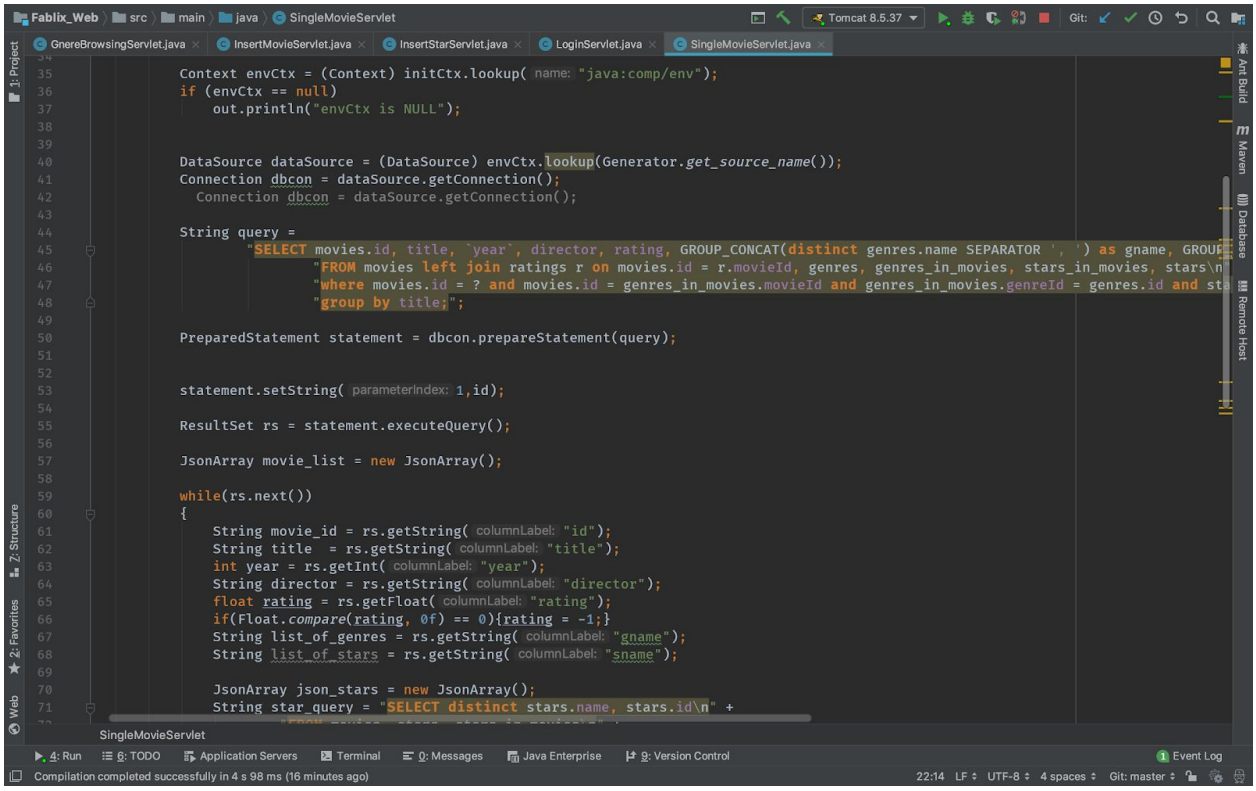

Fablix_Web
src
main
java
GenreServlet
Tomcat 8.5.37
Git:
jmeter_report.html
context.xml
KeywordSearchServlet.java
AutoCompleteServlet.java
EmployeeLoginServlet.java
GenreServlet.java
Context initCtx = new InitialContext();
Context envCtx = (Context) initCtx.lookup(name: "java:comp/env");
if (envCtx == null)
out.println("envCtx is NULL");
DataSource dataSource = (DataSource) envCtx.lookup(Generator.get_source_name());
Connection dbcon = dataSource.getConnection();
// Connection dbcon = dataSource.getConnection();
String query = "Select distinct(genres.name) as a From genres";
PreparedStatement statement = dbcon.prepareStatement(query);
ResultSet resultSet = statement.executeQuery();
JsonArray j = new JsonArray();
while (resultSet.next())
{
j.add(resultSet.getString(columnLabel: "a"));
}
out.write(j.toString());
response.setStatus(200);
dbcon.close();
statement.close();
resultSet.close();
}
catch (Exception e)
{
JsonObject error = new JsonObject();
error.addProperty(property: "error_message", e.getMessage());
out.write(error.toString());
response.setStatus(500);
}
out.close();
}
}
GenreServlet
doGet()
4: Run
6: TODO
Application Servers
Terminal
0: Messages
Java Enterprise
9: Version Control
Event Log
Compilation completed successfully in 4 s 98 ms (13 minutes ago)
41:76
LF
UTF-8
4 spaces
Git: master

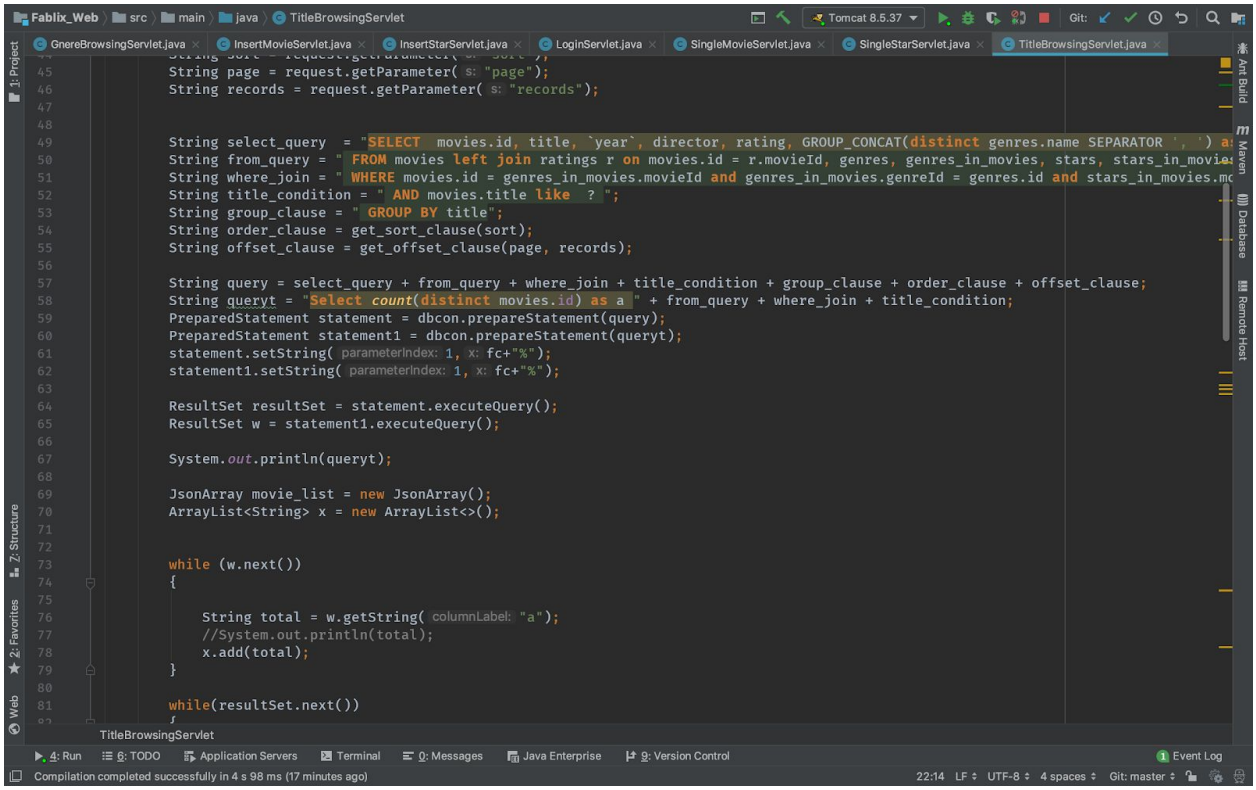

Fablix_Web
src
main
java
GnereBrowsingServlet
Tomcat 8.5.37
Git:
GnereBrowsingServlet.java
DataSource dataSource = (DataSource) envCtx.lookup(Generator.get_source_name());
Connection dbcon = dataSource.getConnection();
Connection dbcon = dataSource.getConnection();
String genre = request.getParameter(s: "genre");
String sort = request.getParameter(s: "sort");
String page = request.getParameter(s: "page");
String records = request.getParameter(s: "records");
String select_query = "SELECT movies.id, title, `year`, director, rating, GROUP_CONCAT(distinct genres.name SEPARATOR ', ') as gna
String from_query = " FROM movies left join ratings r on movies.id = r.movieId, genres, genres_in_movies, stars, stars_in_movies";
String where_join = " WHERE movies.id = genres_in_movies.movieId and genres_in_movies.genreId = genres.id and stars_in_movies.movieI
String genre_condition = " AND genres.name = ? ";
String group_clause = " GROUP BY title";
String order_clause = get_sort_clause(sort);
String offset_clause = get_offset_clause(page, records);
String query = select_query + from_query + where_join + genre_condition + group_clause + order_clause + offset_clause;
String queryt = "Select count(distinct movies.id) as a " + from_query + where_join + genre_condition;
PreparedStatement statement = dbcon.prepareStatement(query);
PreparedStatement statement1 = dbcon.prepareStatement(queryt);
statement.setString(parameterIndex: 1,genre);
statement1.setString(parameterIndex: 1,genre);
ResultSet resultSet = statement.executeQuery();
ResultSet w = statement1.executeQuery();
System.out.println(queryt);
JsonArray movie_list = new JsonArray();
ArrayList<String> x = new ArrayList<>();
while (w.next())
{
String total = w.getString(columnLabel: "a");
//System.out.println(total);
x.add(total);
GnereBrowsingServlet
4: Run
6: TODO
Application Servers
Terminal
0: Messages
Java Enterprise
9: Version Control
Event Log
Typo: In word 'Gnere'
22:14
LF
UTF-8
4 spaces
Git: master


Fablix_Web
src
main
java
InsertMovieServlet
Tomcat 8.5.37
Git:
GnereBrowsingServlet.java
InsertMovieServlet.java
response.setContentType("application/json");
PrintWriter out = response.getWriter();
String title = request.getParameter(s: "title");
String year = request.getParameter(s: "year");
String director = request.getParameter(s: "director");
String star = request.getParameter(s: "star");
String genre = request.getParameter(s: "genre");
try
{
Connection dbcon = dataSource.getConnection();
String script = "{CALL INSERT_MOVIE(?,?,?,?,?)}";
PreparedStatement statement = dbcon.prepareStatement(script);
statement.setString(parameterIndex: 1,title);
statement.setString(parameterIndex: 2,year);
statement.setString(parameterIndex: 3,director);
statement.setString(parameterIndex: 4,star);
statement.setString(parameterIndex: 5,genre);
ResultSet resultSet = statement.executeQuery();
while (resultSet.next())
{
JsonObject r = new JsonObject();
String res = resultSet.getString(columnLabel: "res");
r.addProperty(property: "status", res);
out.write(r.toString());
response.setStatus(200);
resultSet.close();
dbcon.close();
return;
}
}
catch (Exception e)
{
System.err.println(e);
JsonObject r = new JsonObject();
r.addProperty(property: "error_message", e.getMessage());
InsertMovieServlet
4: Run
6: TODO
Application Servers
Terminal
0: Messages
Java Enterprise
9: Version Control
Event Log
Compilation completed successfully in 4 s 98 ms (14 minutes ago)
21:14
LF
UTF-8
4 spaces
Git: master

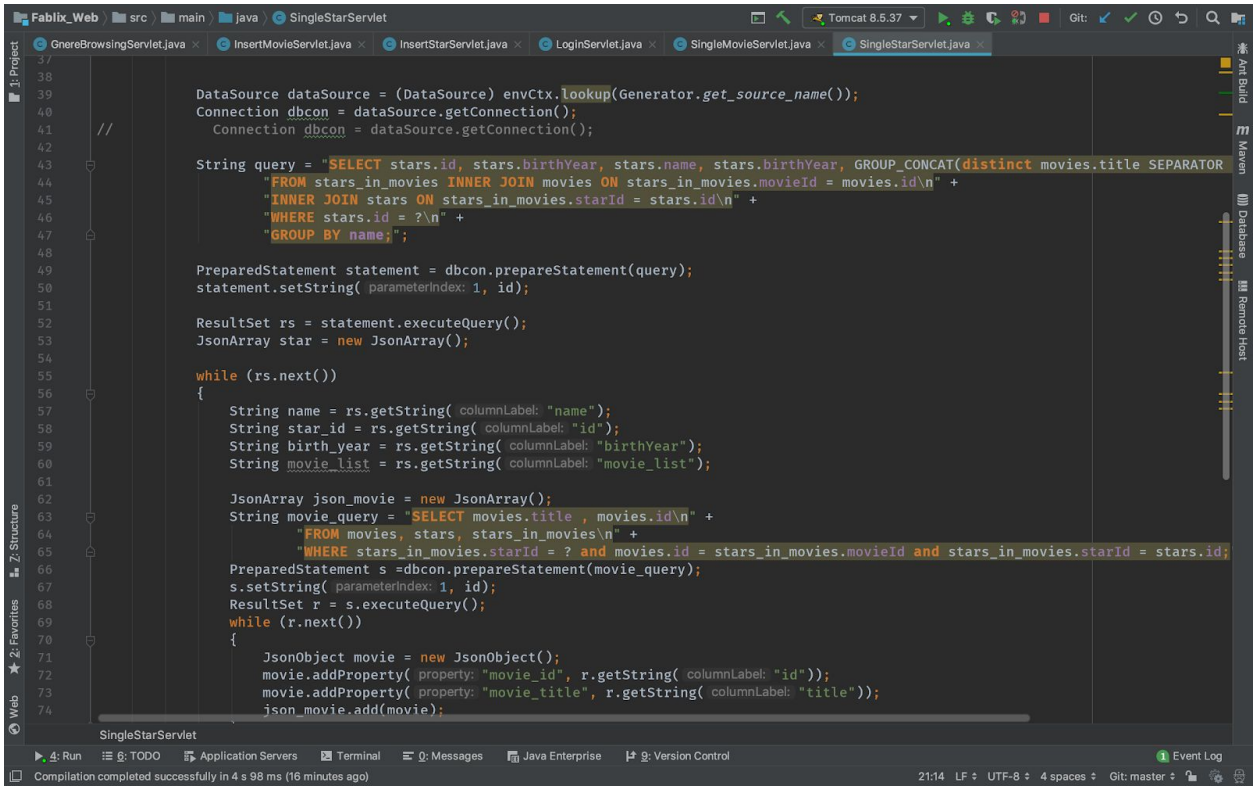

Fablix_Web
src
main
java
InsertStarServlet
Tomcat 8.5.37
Git:
GnereBrowsingServlet.java
InsertMovieServlet.java
InsertStarServlet.java
response.setContentType("application/json");
PrintWriter out = response.getWriter();
String name = request.getParameter(s: "name");
String byear = request.getParameter(s: "year");
if (byear.isEmpty()) byear = null;
try
{
Connection dbcon = dataSource.getConnection();
String script = "{CALL INSERT_STAR(?,?)}";
PreparedStatement statement = dbcon.prepareStatement(script);
statement.setString(parameterIndex: 1,name);
statement.setString(parameterIndex: 2,byear);
ResultSet resultSet = statement.executeQuery();
while (resultSet.next())
{
JsonObject r = new JsonObject();
String res = resultSet.getString(columnLabel: "res");
r.addProperty(property: "status", res);
out.write(r.toString());
response.setStatus(200);
resultSet.close();
dbcon.close();
return;
}
}
catch (Exception e)
{
System.err.println(e);
JsonObject r = new JsonObject();
r.addProperty(property: "error_message", e.getMessage());
out.write(r.toString());
response.setStatus(500);
}
}
InsertStarServlet
4: Run
6: TODO
Application Servers
Terminal
0: Messages
Java Enterprise
9: Version Control
Event Log
Compilation completed successfully in 4 s 98 ms (15 minutes ago)
17:14
LF
UTF-8
4 spaces
Git: master


Fablix_Web › src › main › java › LoginServlet
Tomcat 8.5.37
Git:
GnereBrowsingServlet.java
InsertMovieServlet.java
InsertStarServlet.java
LoginServlet.java
try
{
Context initCtx = new InitialContext();
Context envCtx = (Context) initCtx.lookup(name: "java:comp/env");
if (envCtx == null)
System.out.println("envCtx is NULL");
DataSource dataSource = (DataSource) envCtx.lookup(Generator.get_source_name());
con = dataSource.getConnection();
// con = datasource.getConnection();
String query = "Select id from customers where email = ? ";
PreparedStatement statement = con.prepareStatement(query);
statement.setString(parameterIndex: 1, username);
ResultSet resultSet = statement.executeQuery();
while (resultSet.next())
{
String id = resultSet.getString(columnLabel: "id");
con.close();
statement.close();
resultSet.close();
return id;
}
}
catch (Exception e)
{
System.out.println(e.getMessage());
return "";
}
finally {
try {
con.close();
}catch (Exception e)
{}
}
return "";
}
LoginServlet › doPost()
4: Run
6: TODO
Application Servers
Terminal
0: Messages
Java Enterprise
9: Version Control
1 Event Log
Compilation completed successfully in 4 s 98 ms (15 minutes ago)
8 chars
24:79
LF
UTF-8
4 spaces
Git: master
1: Project
Z: Structure
2: Favorites
Web
Ant Build
Maven
Database
Remote Host


Fablix_Web › src › main › java › SingleMovieServlet
Tomcat 8.5.37
Git:
GnereBrowsingServlet.java
InsertMovieServlet.java
InsertStarServlet.java
LoginServlet.java
SingleMovieServlet.java
Context envCtx = (Context) initCtx.lookup(name: "java:comp/env");
if (envCtx == null)
out.println("envCtx is NULL");
DataSource dataSource = (DataSource) envCtx.lookup(Generator.get_source_name());
Connection dbcon = dataSource.getConnection();
Connection dbcon = dataSource.getConnection();
String query =
"SELECT movies.id, title, `year`, director, rating, GROUP_CONCAT(distinct genres.name SEPARATOR ', ') as gname, GROUP
"FROM movies left join ratings r on movies.id = r.movieId, genres, genres_in_movies, stars_in_movies, stars\n
"where movies.id = ? and movies.id = genres_in_movies.movieId and genres_in_movies.genreId = genres.id and sta
"group by title;";
PreparedStatement statement = dbcon.prepareStatement(query);
statement.setString(parameterIndex: 1,id);
ResultSet rs = statement.executeQuery();
JsonArray movie_list = new JsonArray();
while(rs.next())
{
String movie_id = rs.getString(columnLabel: "id");
String title = rs.getString(columnLabel: "title");
int year = rs.getInt(columnLabel: "year");
String director = rs.getString(columnLabel: "director");
float rating = rs.getFloat(columnLabel: "rating");
if(Float.compare(rating, 0f) == 0){rating = -1;}
String list_of_genres = rs.getString(columnLabel: "gname");
String list_of_stars = rs.getString(columnLabel: "sname");
JsonArray json_stars = new JsonArray();
String star_query = "SELECT distinct stars.name, stars.id\n" +
SingleMovieServlet
4: Run
6: TODO
Application Servers
Terminal
0: Messages
Java Enterprise
9: Version Control
1 Event Log
Compilation completed successfully in 4 s 98 ms (16 minutes ago)
22:14
LF
UTF-8
4 spaces
Git: master
1: Project
Z: Structure
2: Favorites
Web
Ant Build
Maven
Database
Remote Host



```
String page = request.getParameter( s: "page");
String records = request.getParameter( s: "records");


String select_query  = "SELECT  movies.id, title, `year`, director, rating, GROUP_CONCAT(distinct genres.name SEPARATOR ', ') a
String from_query = " FROM movies left join ratings r on movies.id = r.movieId, genres, genres_in_movies, stars, stars_in_movie
String where_join = " WHERE movies.id = genres_in_movies.movieId and genres_in_movies.genreId = genres.id and stars_in_movies.m
String title_condition = " AND movies.title like  ? ";
String group_clause = " GROUP BY title";
String order_clause = get_sort_clause(sort);
String offset_clause = get_offset_clause(page, records);

String query = select_query + from_query + where_join + title_condition + group_clause + order_clause + offset_clause;
String queryt = "Select count(distinct movies.id) as a " + from_query + where_join + title_condition;
PreparedStatement statement = dbcon.prepareStatement(query);
PreparedStatement statement1 = dbcon.prepareStatement(queryt);
statement.setString( parameterIndex: 1, x: fc+"%");
statement1.setString( parameterIndex: 1, x: fc+"%");

ResultSet resultSet = statement.executeQuery();
ResultSet w = statement1.executeQuery();

System.out.println(queryt);

JsonArray movie_list = new JsonArray();
ArrayList<String> x = new ArrayList<>();


while (w.next())
{

    String total = w.getString( columnLabel: "a");
    //System.out.println(total);
    x.add(total);
}

while(resultSet.next())
```



```
DataSource dataSource = (DataSource) envCtx.lookup(Generator.get_source_name());
Connection dbcon = dataSource.getConnection();
//            Connection dbcon = dataSource.getConnection();

String query = "SELECT stars.id, stars.birthYear, stars.name, stars.birthYear, GROUP_CONCAT(distinct movies.title SEPARATOR
       "FROM stars_in_movies INNER JOIN movies ON stars_in_movies.movieId = movies.id\n" +
       "INNER JOIN stars ON stars_in_movies.starId = stars.id\n" +
       "WHERE stars.id = ?\n" +
       "GROUP BY name;";

PreparedStatement statement = dbcon.prepareStatement(query);
statement.setString( parameterIndex: 1, id);

ResultSet rs = statement.executeQuery();
JsonArray star = new JsonArray();

while (rs.next())
{
    String name = rs.getString( columnLabel: "name");
    String star_id = rs.getString( columnLabel: "id");
    String birth_year = rs.getString( columnLabel: "birthYear");
    String movie_list = rs.getString( columnLabel: "movie_list");

    JsonArray json_movie = new JsonArray();
    String movie_query = "SELECT movies.title , movies.id\n" +
           "FROM movies, stars, stars_in_movies\n" +
           "WHERE stars_in_movies.starId = ? and movies.id = stars_in_movies.movieId and stars_in_movies.starId = stars.id;
    PreparedStatement s =dbcon.prepareStatement(movie_query);
    s.setString( parameterIndex: 1, id);
    ResultSet r = s.executeQuery();
    while (r.next())
    {
        JsonObject movie = new JsonObject();
        movie.addProperty( property: "movie_id", r.getString( columnLabel: "id"));
        movie.addProperty( property: "movie_title", r.getString( columnLabel: "title"));
        json_movie.add(movie);
```

Task 2

- Address of AWS and Google instances

  AWS Balancer: 3.17.195.187/fablix/
  AWS Master: 52.15.235.94
  AWS Slave: 3.18.105.159
  GCP Balancer: 34.73.131.140/fablix/

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

  Yes
  AWS Single: 3.17.195.187:8080/fablix

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

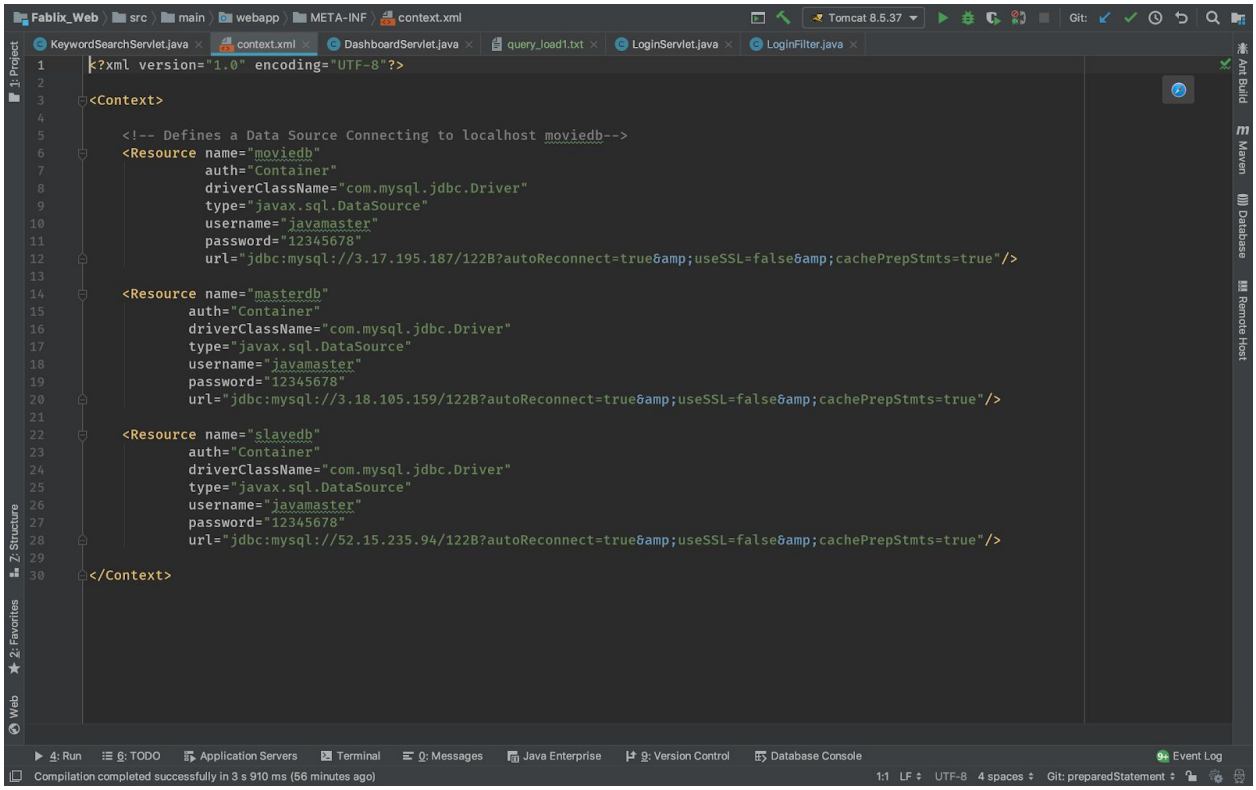
`

There are two database resources written in context.xml,
For the servlet which only read from database it connect to master/slave randomly(read only), for the servlet which need to write into database it connect to the master(read/write)

- File name, line numbers as in Github

  Fablix_Web/src/main/webapp/META-INF/context.xml
  Line 14-28

- Snapshots



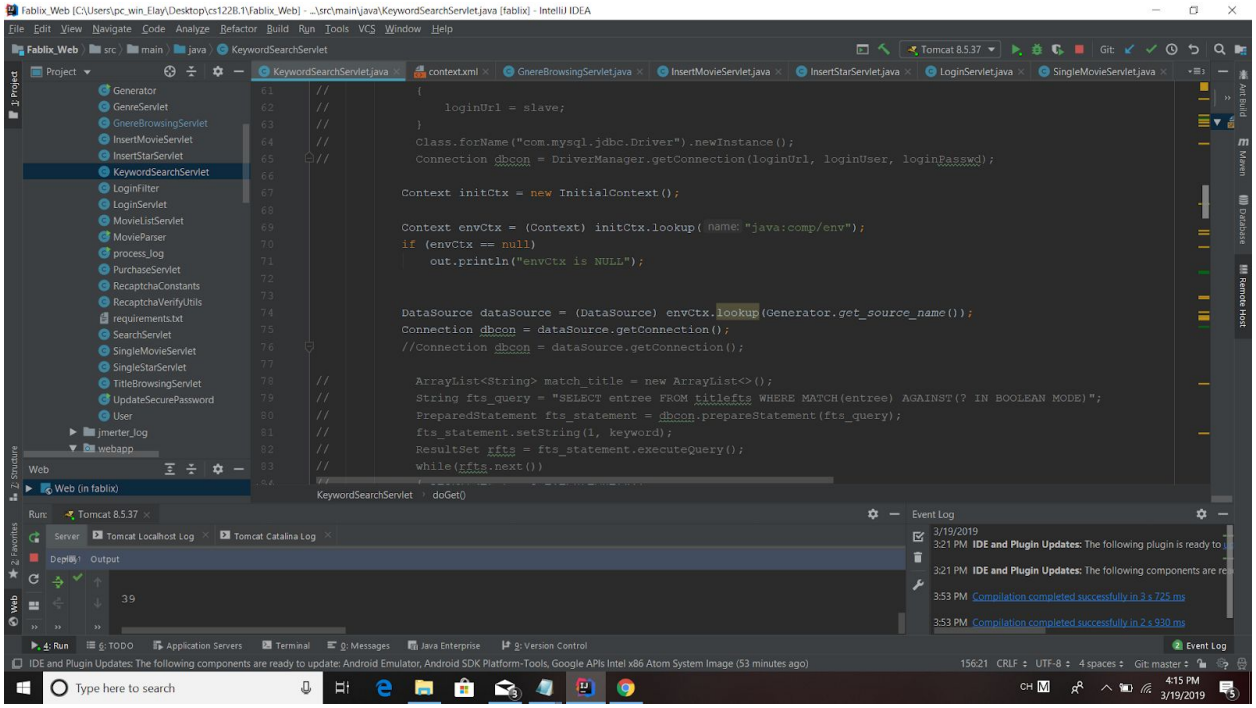
- How read/write requests were routed?

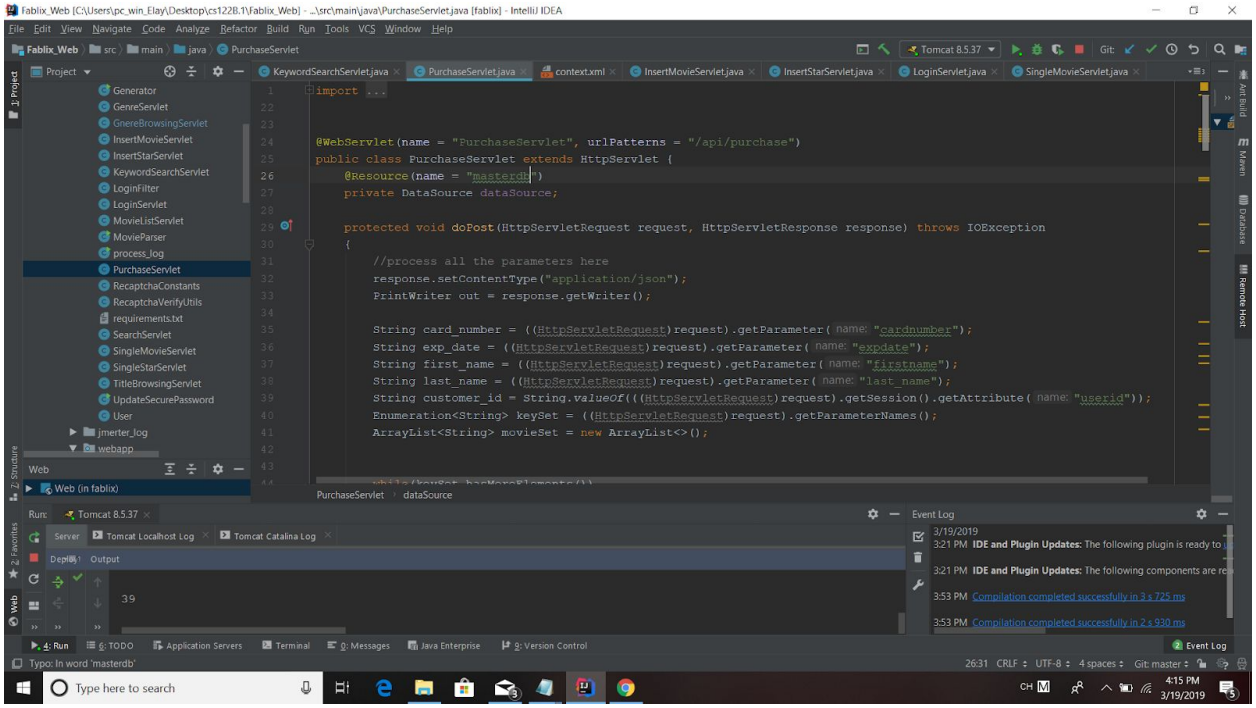
We randomly choose which db(master or slave) will be used in the servlet that only reads the db. We choose the master db in the servlet that writes to db

- File name, line numbers as in Github

All ***Servlet.java uses random db except for PurchaseServlet.java, InsertMovieServlet.java and InsertStarServlet.java, which are using master db only.

- Snapshots





Task 3

- Have you uploaded the log files to Github? Where is it located?
  All log files have been uploaded to the github.
  Path: /Fablix_Wed/src/main/jmeter_log/

- Have you uploaded the HTML file (with all sections including analysis, written up) to Github? Where is it located?
  Yes the file have been uploaded to the project.
  Path: /Fablix_Web/jmeter_report.html

- Have you uploaded the script to Github? Where is it located?
  Yes:
  Path: /Fablix_Web/src/main/java/process_log.java

- Have you uploaded the WAR file and README to Github? Where is it located?
  Path to the war file: is /Fablix_Web/fablix.war
  Path to the README file: /Fablix_Web/README.md