Task 1

- **How did you use connection pooling?**

In our context.xml, we added these new parameters to our resource for moviedb:

maxTotal="100" maxIdle="30" maxWaitMillis="10000"

And we changed the url in the resource to:

jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrep
Stmts=true

Also, we added a resource reference to this database resource in our web.xml.

All of this setup allows us to use connection pooling in our servlets. In our servlets (shown in screenshots below), we find our environment naming context and look up our data source that we created to actually use connection pooling for each servlet.

- **File name, line numbers as in Github**

context.xml (under cs122b-spring18-team-42/Fablix/WebContent/META-INF)
web.xml (under cs122b-spring18-team-42/Fablix/WebContent/WEB-INF)

***All files under cs122b-spring18-team-42/Fablix/src

AddToCartServlet.java
    Lines: 47-52
AutocompleteSearchServlet.java
    Lines: 34-39
BrowseGenresServlet.java
    Lines: 34-39
BrowseServlet.java
    Lines: 34-39
CheckoutServlet.java
    Lines 41-46
LoginServlet.java
    Lines: 33-38
ShoppingCartServlet.java
    Lines 38-43
SearchServlet.java
    Lines: 66-71

SingleMovieServlet.java

Lines: 39-44

SingleStarServlet.java

Lines: 40-45

UpdateCartServlet.java

Lines: 50-55

_DashboardAddMovie.java

Lines: 55-60

_DashboardAddStar.java

Lines: 50-55

_DashboardLoad.java

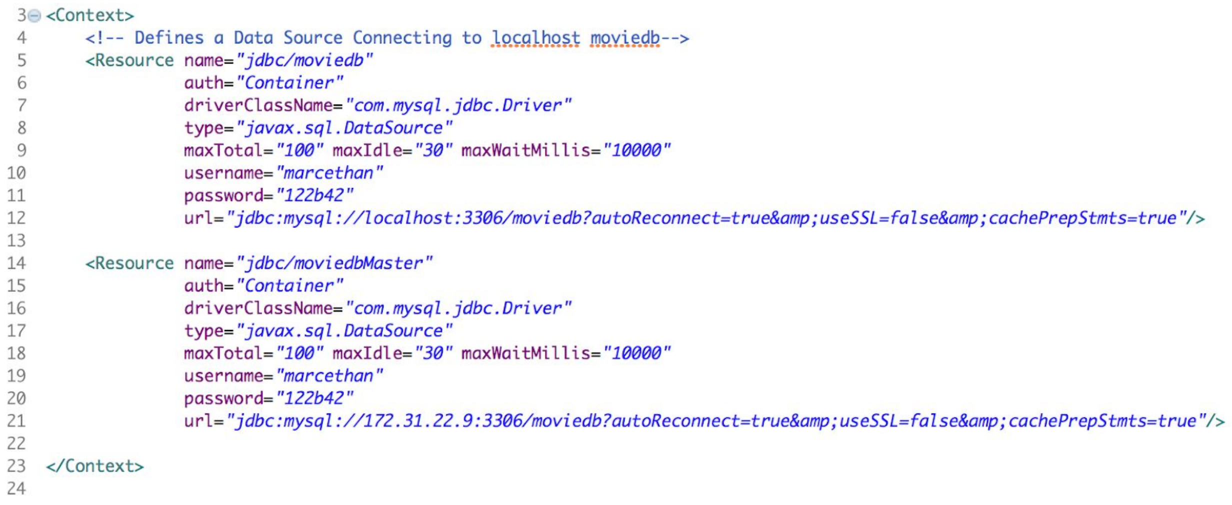
Lines: 35-40

_DashboardLogin.java

Lines: 35-40

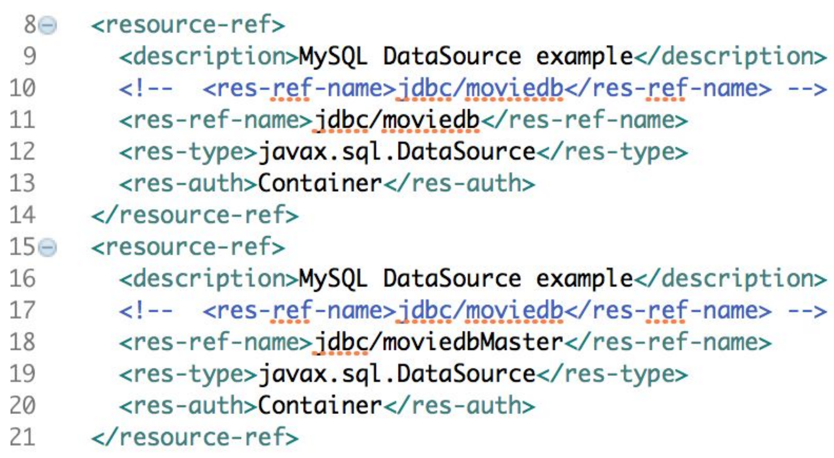
- **Snapshots showing use in your code**

From context.xml



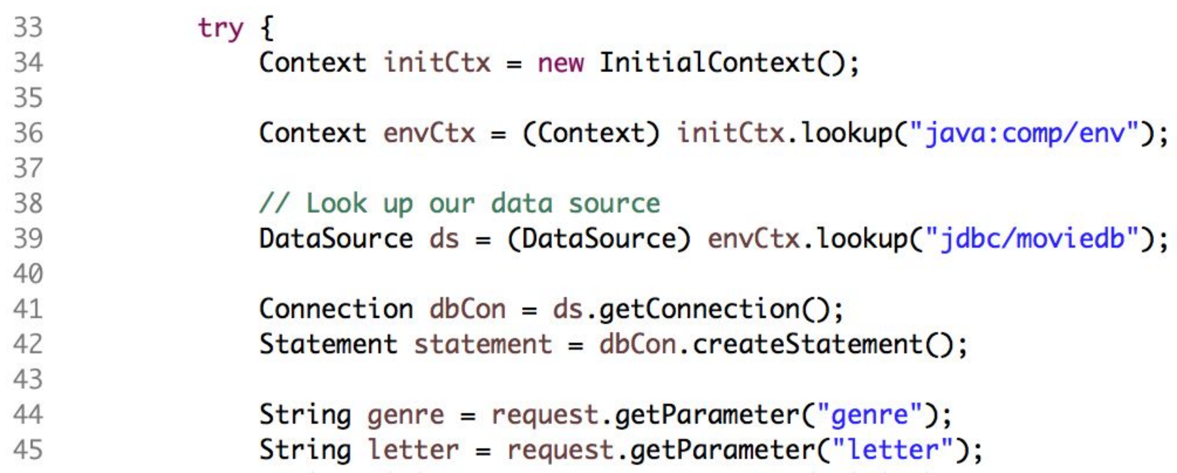
```
<Context>
    <!-- Defines a Data Source Connecting to localhost moviedb-->
    <Resource name="jdbc/moviedb"
              auth="Container"
              driverClassName="com.mysql.jdbc.Driver"
              type="javax.sql.DataSource"
              maxTotal="100" maxIdle="30" maxWaitMillis="10000"
              username="marcethan"
              password="122b42"
              url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>

    <Resource name="jdbc/moviedbMaster"
              auth="Container"
              driverClassName="com.mysql.jdbc.Driver"
              type="javax.sql.DataSource"
              maxTotal="100" maxIdle="30" maxWaitMillis="10000"
              username="marcethan"
              password="122b42"
              url="jdbc:mysql://172.31.22.9:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>

</Context>
```

From web.xml



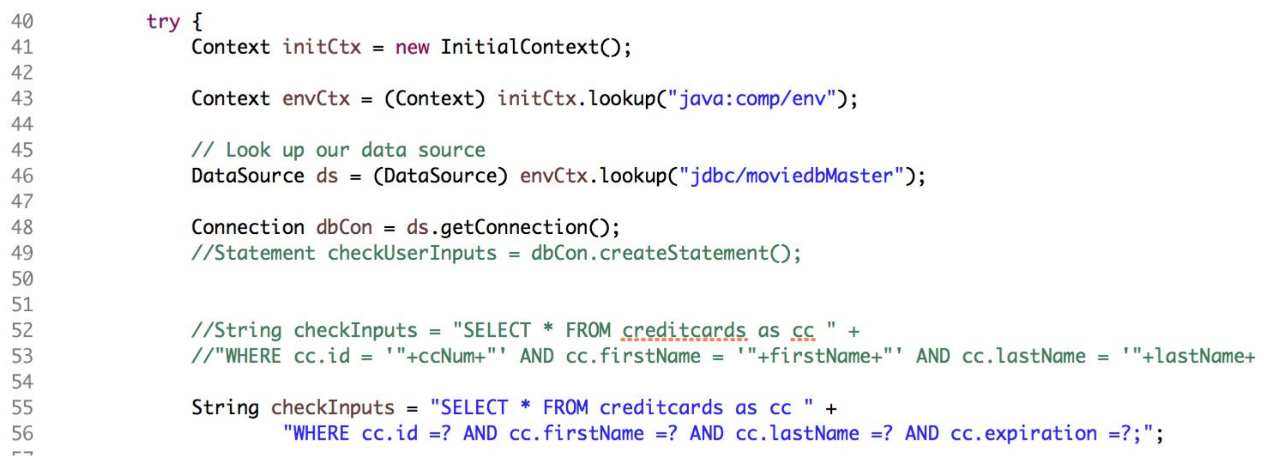
```
<resource-ref>
  <description>MySQL DataSource example</description>
  <!--  <res-ref-name>jdbc/moviedb</res-ref-name> -->
  <res-ref-name>jdbc/moviedb</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
<resource-ref>
  <description>MySQL DataSource example</description>
  <!--  <res-ref-name>jdbc/moviedb</res-ref-name> -->
  <res-ref-name>jdbc/moviedbMaster</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

From BrowseServlet.java



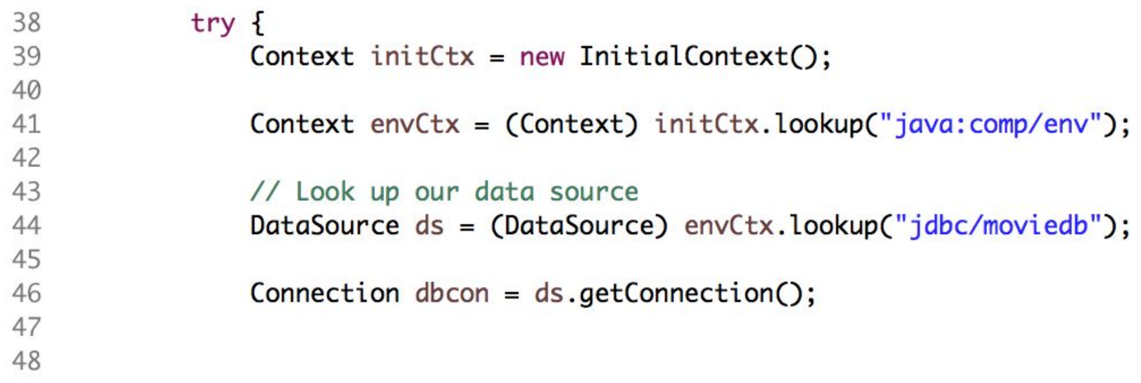
```
        try {
            Context initCtx = new InitialContext();

            Context envCtx = (Context) initCtx.lookup("java:comp/env");

            // Look up our data source
            DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");

            Connection dbCon = ds.getConnection();
            Statement statement = dbCon.createStatement();

            String genre = request.getParameter("genre");
            String letter = request.getParameter("letter");
```

From CheckoutServlet.java



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

            Context envCtx = (Context) initCtx.lookup("java:comp/env");

            // Look up our data source
            DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedbMaster");

            Connection dbCon = ds.getConnection();
            //Statement checkUserInputs = dbCon.createStatement();


            //String checkInputs = "SELECT * FROM creditcards as cc " +
            //"WHERE cc.id = '"+ccNum+"' AND cc.firstName = '"+firstName+"' AND cc.lastName = '"+lastName+

            String checkInputs = "SELECT * FROM creditcards as cc " +
                    "WHERE cc.id =? AND cc.firstName =? AND cc.lastName =? AND cc.expiration =?;";
```

From SingleMovieServlet.java



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

            Context envCtx = (Context) initCtx.lookup("java:comp/env");

            // Look up our data source
            DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");

            Connection dbcon = ds.getConnection();


```

- **How did you use Prepared Statements?**

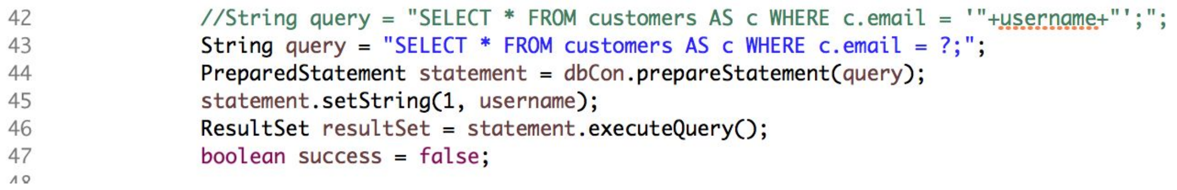
We added prepared statements to any servlet that dealt with asking the user for input (i.e. from a text box), which includes any that dealt with search.

- File name, line numbers as in Github

CheckoutServlet.java
    Lines: 58,80,103,114,136
SearchServlet.java
    Lines: 84, 180-284
ShoppingCartServlet.java
    Lines: 60
_DashboardAddMovie.java
    Lines: 65,83,104,113,140,149,182
_DashboardAddStar.java
    Lines: 60,71
LoginServlet.java
    Lines: 44
AutocompleteSearchServlet.java
    Lines: 63

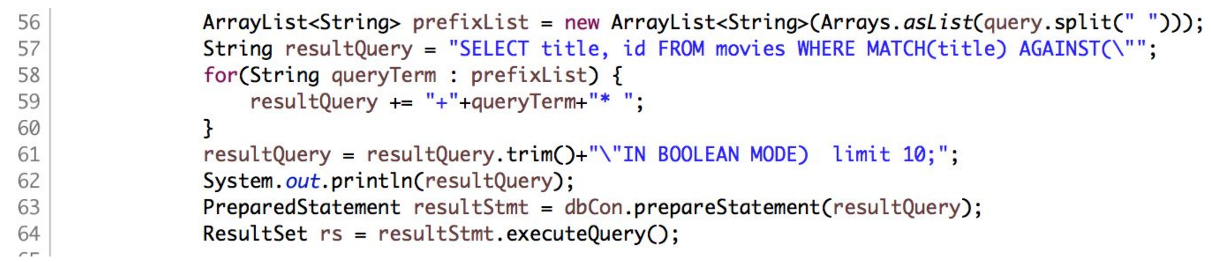
- Snapshots showing use in your code

From LoginServlet.java



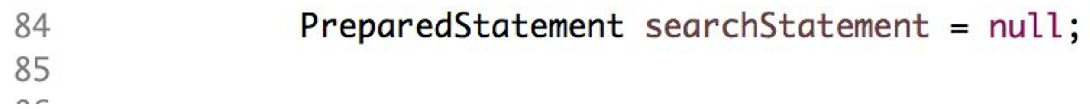
```java
42            //String query = "SELECT * FROM customers AS c WHERE c.email = '"+username+"';";
43            String query = "SELECT * FROM customers AS c WHERE c.email = ?;";
44            PreparedStatement statement = dbCon.prepareStatement(query);
45            statement.setString(1, username);
46            ResultSet resultSet = statement.executeQuery();
47            boolean success = false;
```

From AutocompleteSearchServlet.java

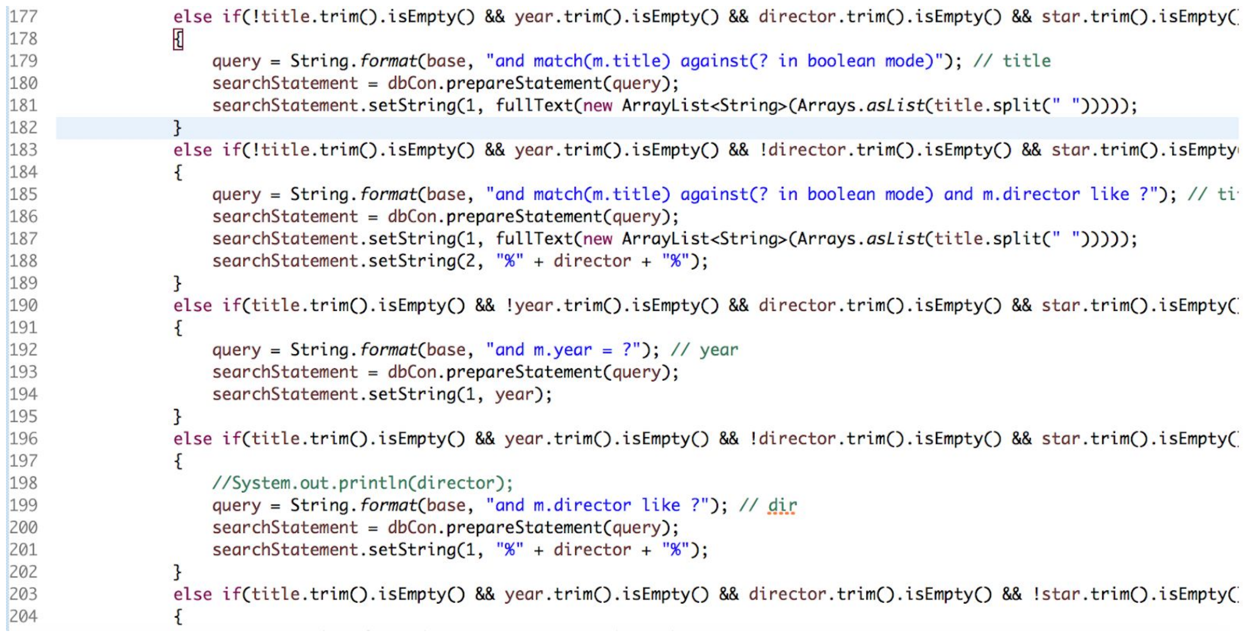


```
            ArrayList<String> prefixList = new ArrayList<String>(Arrays.asList(query.split(" ")));
            String resultQuery = "SELECT title, id FROM movies WHERE MATCH(title) AGAINST(\"";
            for(String queryTerm : prefixList) {
                resultQuery += "+"+queryTerm+"* ";
            }
            resultQuery = resultQuery.trim()+"\"IN BOOLEAN MODE)  limit 10;";
            System.out.println(resultQuery);
            PreparedStatement resultStmt = dbCon.prepareStatement(resultQuery);
            ResultSet rs = resultStmt.executeQuery();
```

From SearchServlet.java (large if-else block has multiple prepared statement assignments)



```
            PreparedStatement searchStatement = null;
```



```
            else if(!title.trim().isEmpty() && year.trim().isEmpty() && director.trim().isEmpty() && star.trim().isEmpty(
            {
                query = String.format(base, "and match(m.title) against(? in boolean mode)"); // title
                searchStatement = dbCon.prepareStatement(query);
                searchStatement.setString(1, fullText(new ArrayList<String>(Arrays.asList(title.split(" ")))));
            }
            else if(!title.trim().isEmpty() && year.trim().isEmpty() && !director.trim().isEmpty() && star.trim().isEmpty
            {
                query = String.format(base, "and match(m.title) against(? in boolean mode) and m.director like ?"); // ti
                searchStatement = dbCon.prepareStatement(query);
                searchStatement.setString(1, fullText(new ArrayList<String>(Arrays.asList(title.split(" ")))));
                searchStatement.setString(2, "%" + director + "%");
            }
            else if(title.trim().isEmpty() && !year.trim().isEmpty() && director.trim().isEmpty() && star.trim().isEmpty(
            {
                query = String.format(base, "and m.year = ?"); // year
                searchStatement = dbCon.prepareStatement(query);
                searchStatement.setString(1, year);
            }
            else if(title.trim().isEmpty() && year.trim().isEmpty() && !director.trim().isEmpty() && star.trim().isEmpty(
            {
                //System.out.println(director);
                query = String.format(base, "and m.director like ?"); // dir
                searchStatement = dbCon.prepareStatement(query);
                searchStatement.setString(1, "%" + director + "%");
            }
            else if(title.trim().isEmpty() && year.trim().isEmpty() && director.trim().isEmpty() && !star.trim().isEmpty(
            {
```

<u>Task 2</u>

- **Address of AWS and Google instances**

Instance 1: 18.219.25.24
Instance 2: 13.58.89.221
Instance 3: 18.220.237.143
Google: 104.196.151.243

http://IP:port/Fablix/ for all four IP addresses

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

Yes both ports are open on both.

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

We created another data resource called moviedbMaster which is the resource we use for connecting to the master instance's database. Our original data source (moviedb) had the url that included localhost which could connect to either the master or slave instance. We also created another resource reference to this data resource (movedbMaster) in our web.xml.

- **File name, line numbers as in Github**

context.xml (under cs122b-spring18-team-42/Fablix/WebContent/META-INF)
web.xml (under cs122b-spring18-team-42/Fablix/WebContent/WEB-INF)

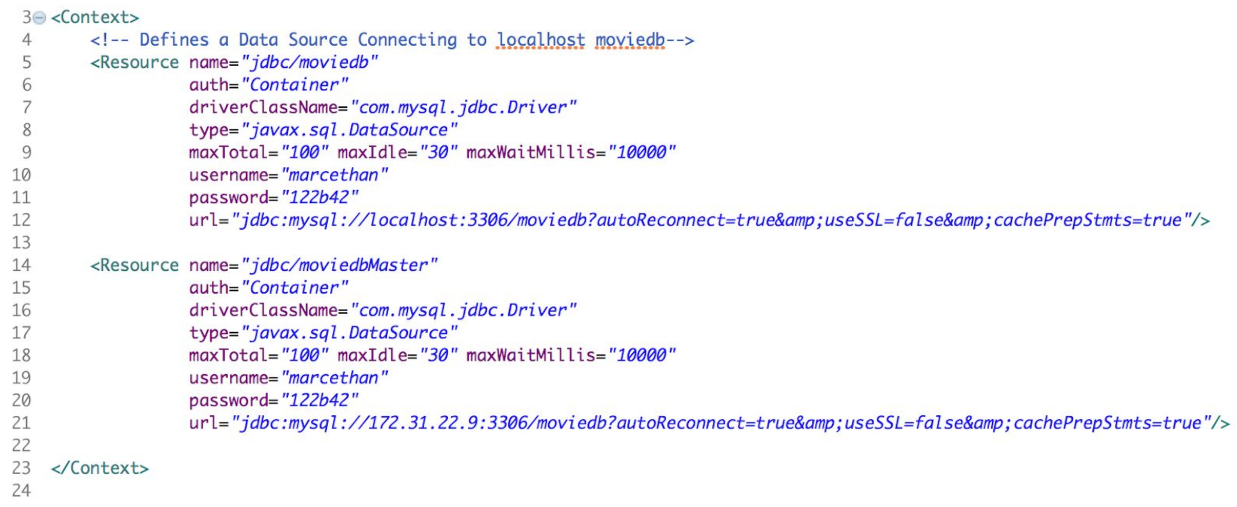
***All files under cs122b-spring18-team-42/Fablix

AddToCartServlet.java
    Lines: 47-52
AutocompleteSearchServlet.java
    Lines: 34-39
BrowseGenresServlet.java
    Lines: 34-39
BrowseServlet.java
    Lines: 34-39
CheckoutServlet.java
    Lines 41-46
LoginServlet.java
    Lines: 33-38
ShoppingCartServlet.java
    Lines 38-43
SearchServlet.java
    Lines: 66-71
SingleMovieServlet.java
    Lines: 39-44
SingleStarServlet.java
    Lines: 40-45
UpdateCartServlet.java
    Lines: 50-55
_DashboardAddMovie.java
    Lines: 55-60
_DashboardAddStar.java
    Lines: 50-55

_DashboardLoad.java
Lines: 35-40
_DashboardLogin.java
Lines: 35-40

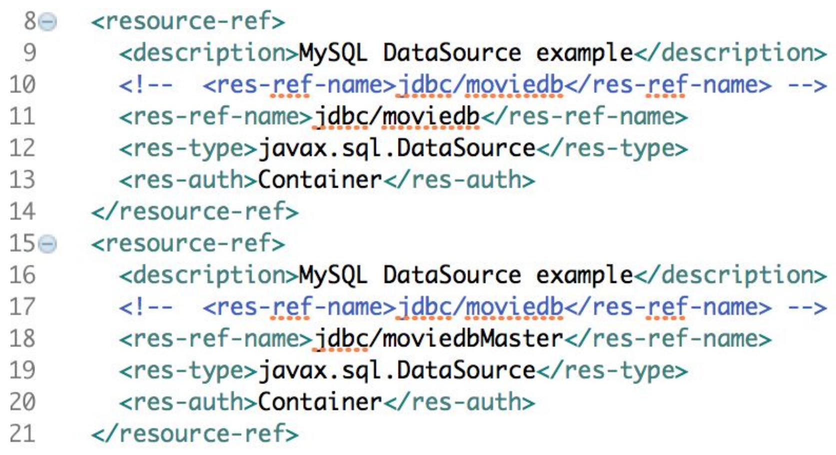
- Snapshots

From context.xml



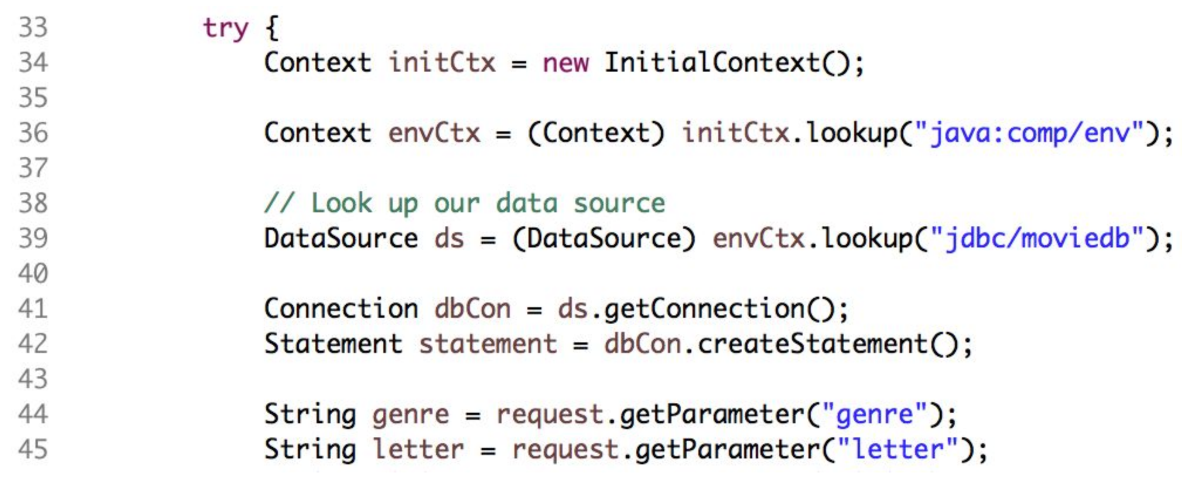
```
<Context>
    <!-- Defines a Data Source Connecting to localhost moviedb-->
    <Resource name="jdbc/moviedb"
              auth="Container"
              driverClassName="com.mysql.jdbc.Driver"
              type="javax.sql.DataSource"
              maxTotal="100" maxIdle="30" maxWaitMillis="10000"
              username="marcethan"
              password="122b42"
              url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>

    <Resource name="jdbc/moviedbMaster"
              auth="Container"
              driverClassName="com.mysql.jdbc.Driver"
              type="javax.sql.DataSource"
              maxTotal="100" maxIdle="30" maxWaitMillis="10000"
              username="marcethan"
              password="122b42"
              url="jdbc:mysql://172.31.22.9:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>

</Context>

```

From web.xml



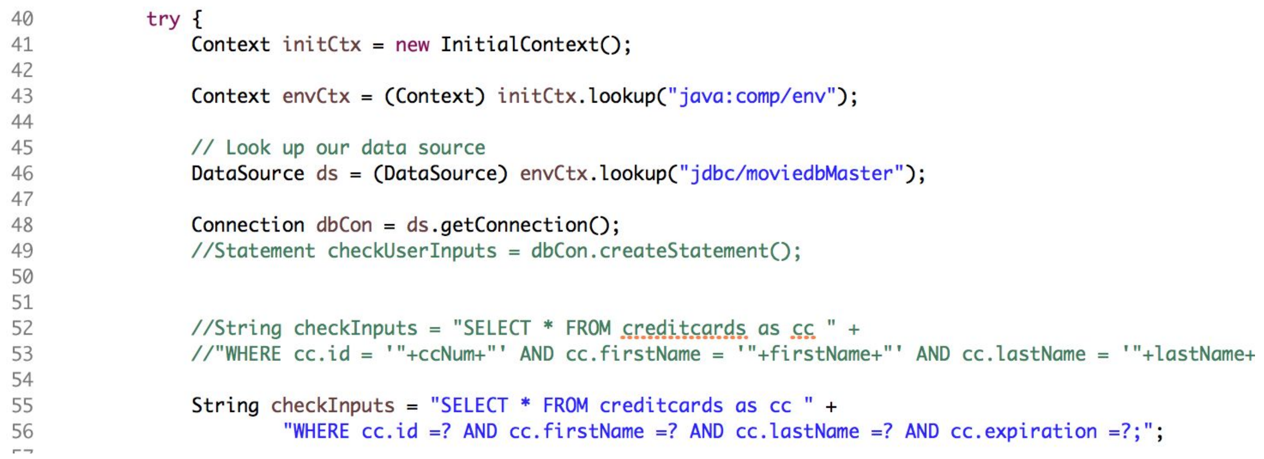
```
<resource-ref>
  <description>MySQL DataSource example</description>
  <!--  <res-ref-name>jdbc/moviedb</res-ref-name> -->
  <res-ref-name>jdbc/moviedb</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
<resource-ref>
  <description>MySQL DataSource example</description>
  <!--  <res-ref-name>jdbc/moviedb</res-ref-name> -->
  <res-ref-name>jdbc/moviedbMaster</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

From BrowseServlet.java



```
33          try {
34              Context initCtx = new InitialContext();
35
36              Context envCtx = (Context) initCtx.lookup("java:comp/env");
37
38              // Look up our data source
39              DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
40
41              Connection dbCon = ds.getConnection();
42              Statement statement = dbCon.createStatement();
43
44              String genre = request.getParameter("genre");
45              String letter = request.getParameter("letter");
```

From CheckoutServlet.java



```
40          try {
41              Context initCtx = new InitialContext();
42
43              Context envCtx = (Context) initCtx.lookup("java:comp/env");
44
45              // Look up our data source
46              DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedbMaster");
47
48              Connection dbCon = ds.getConnection();
49              //Statement checkUserInputs = dbCon.createStatement();
50
51
52              //String checkInputs = "SELECT * FROM creditcards as cc " +
53              //"WHERE cc.id = '"+ccNum+"' AND cc.firstName = '"+firstName+"' AND cc.lastName = '"+lastName+
54
55              String checkInputs = "SELECT * FROM creditcards as cc " +
56                      "WHERE cc.id =? AND cc.firstName =? AND cc.lastName =? AND cc.expiration =?;";
```

- **How read/write requests were routed?**

We send any write requests to master (our master resource used the private ip of our master instance) while we send any read requests to either master or slave (we use the resource that has localhost in the url for this one, which does the redirection for us).

- **File name, line numbers as in Github**

***Marked which ones used the master resource and which ones used the localhost resource

AddToCartServlet.java (master)
Lines: 47-52
AutocompleteSearchServlet.java (localhost)
Lines: 34-39
BrowseGenresServlet.java (localhost)

Lines: 34-39

BrowseServlet.java (localhost)

Lines: 34-39

CheckoutServlet.java (master)

Lines 41-46

LoginServlet.java (localhost)

Lines: 33-38

ShoppingCartServlet.java (localhost)

Lines 38-43

SearchServlet.java (localhost)

Lines: 66-71

SingleMovieServlet.java (localhost)

Lines: 39-44

SingleStarServlet.java (localhost)

Lines: 40-45

UpdateCartServlet.java (master)

Lines: 50-55

_DashboardAddMovie.java (master)

Lines: 55-60

_DashboardAddStar.java (master)

Lines: 50-55

_DashboardLoad.java (localhost)

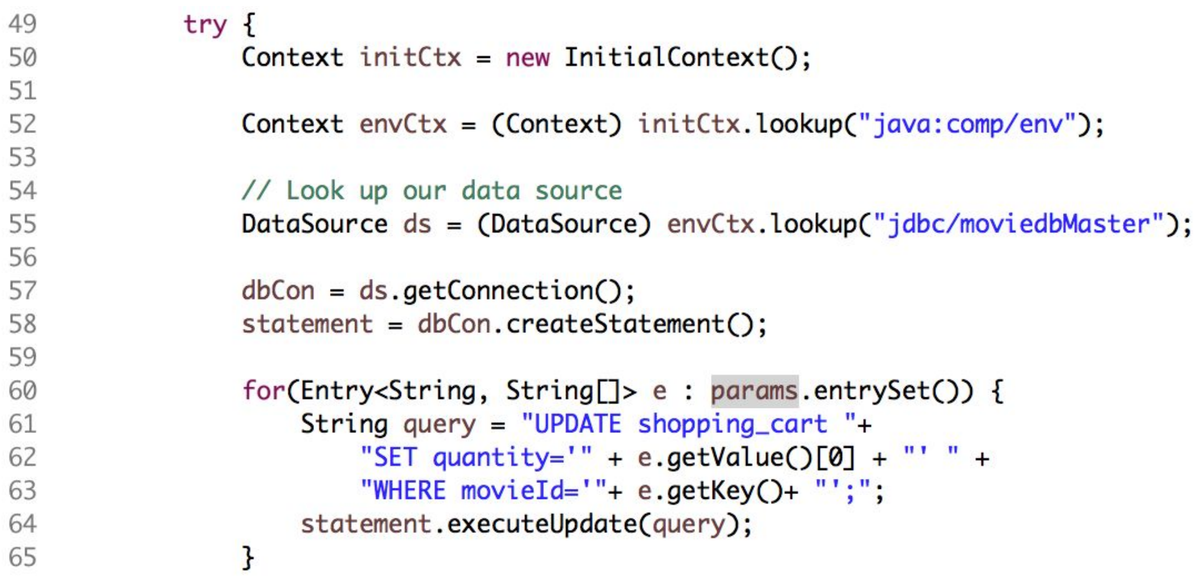
Lines: 35-40

_DashboardLogin.java (localhost)

Lines: 35-40

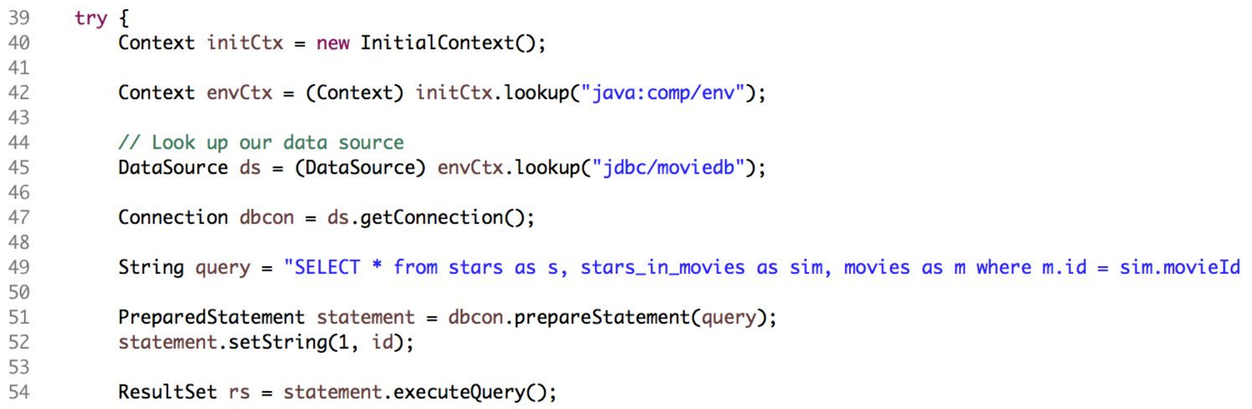
- **Snapshots**

From UpdateCartServlet.java (redirect write request example)



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

            Context envCtx = (Context) initCtx.lookup("java:comp/env");

            // Look up our data source
            DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedbMaster");

            dbCon = ds.getConnection();
            statement = dbCon.createStatement();

            for(Entry<String, String[]> e : params.entrySet()) {
                String query = "UPDATE shopping_cart "+
                    "SET quantity='" + e.getValue()[0] + "' " +
                    "WHERE movieId='"+ e.getKey()+ "';";
                statement.executeUpdate(query);
            }
```

From SingleMovieServlet.java (redirect read request example)



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

    Context envCtx = (Context) initCtx.lookup("java:comp/env");

    // Look up our data source
    DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");

    Connection dbcon = ds.getConnection();

    String query = "SELECT * from stars as s, stars_in_movies as sim, movies as m where m.id = sim.movieId

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

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

Task 3

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

Yes. It is located under cs122b-spring18-team-42/Fablix Logs

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

Yes. It is located under cs122b-spring18-team-42/Fablix Logs

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

Yes. It is located under cs122b-spring18-team-42/parse.py (root of our git repo)

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

Yes. WAR file is located under cs122b-spring18-team-42/Fablix.war
README is located under cs122b-spring18-team-42/Fablix