Task 1:

**Connection Pooling:**

Connection Pooling is used to enhance the performance of executing commands on database and make the web applications perform much more efficiently. By reusing the connections through the caching of the connections, we are able to perform the database connections much faster and much more efficiently.

To use the Connection Pooling, we simply overrode the Connection made in the earlier projects. Also, we added the context.xml in the /WebContent/META-INF/context.xml file to add the following line:

<Context path="/project3">

<br/>

<lt;Resource name="jdbc/moviedb" auth="Container" type="javax.sql.DataSource"

initialSize="40" maxActive="100" maxIdle="80" minIdle="40” minEvictableIdleTimeMillis="55000" timeBetweenEvictionRunsMillis="34000"

testOnBorrow="true" validationInterval="60000" validationQuery="SELECT 1"

removeAbandoned="true" removeAbandonedTimeout="55" maxWait="10000"

username="mytestuser" password="mypassword” driverClassName="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=

false&amp;cachePrepStmts=true" />

<br/>

</Context>

We also had to add the following code into the /WebContent/WEB-INF/web.xml before the servlet declarations:

<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/moviedb</res-ref-name>

<res-type>javax.sql.DataSource</res-type>

<res-auth>Container</res-auth>

</resource-ref>

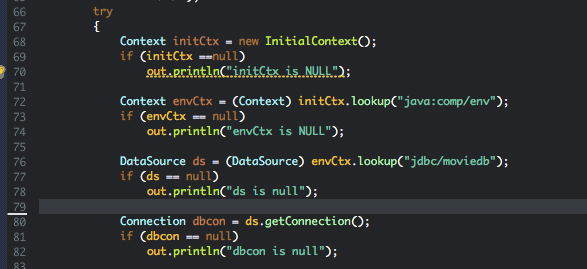
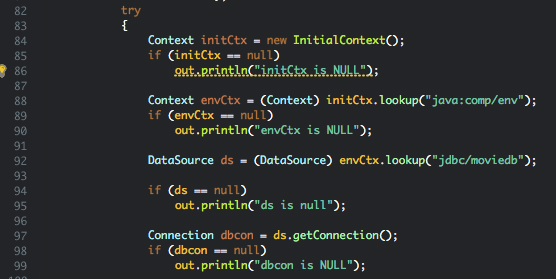
In addition, we had to import the following packages from the Java Library to allow the Connection Pooling in each of our Servlets:

import javax.naming.InitialContext;

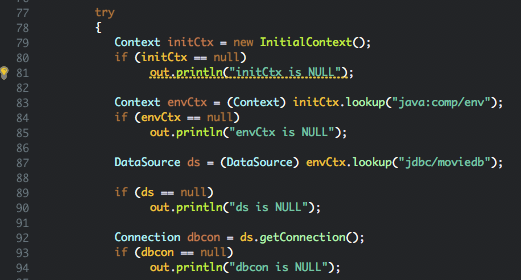
import javax.naming.Context;

import javax.sql.DataSource;

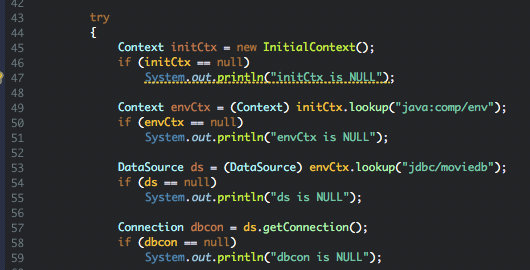
add\_movie.java: addstar.java

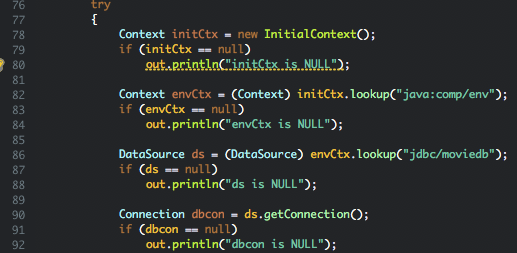
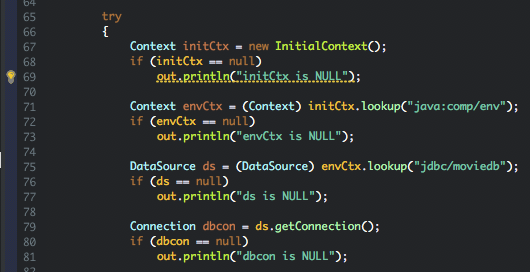
browsegenre.java checkccinfo.java

dLogin.java finalcheckout.java

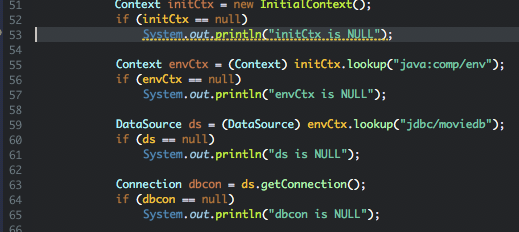
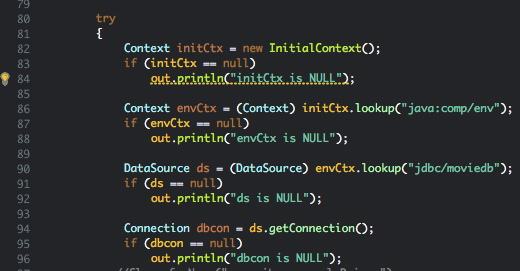
genresearch.java login.java

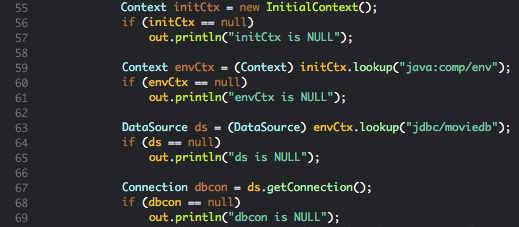
loginapp.java metadata.java

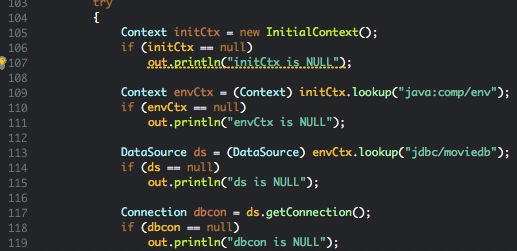
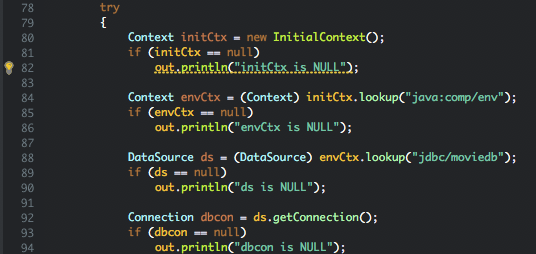
movie\_suggestion.java movieinfo.java

search\_page\_app.java searchpage.java

shoppingcart.java starinfo.java

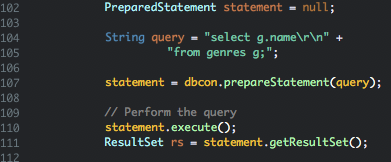
 

**Prepared Statements:**

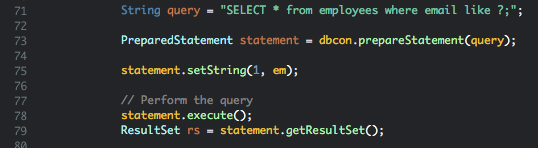
To reduce the execution time of the SQL queries every time we log into the MySQL Server, we used the Prepared Statements. The main benefit of the PreparedStatement is that it sends the query to the DBMS right away where it is compiled.

We implemented this by adding the “import java.sql.PreparedStatement;” package to all the servlets that use the search method and implemented the following lines:

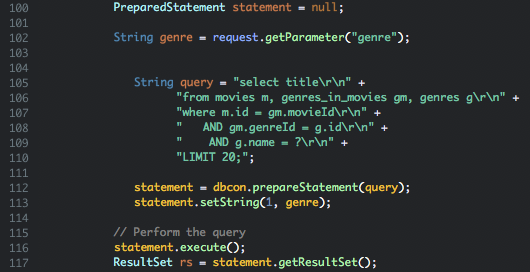
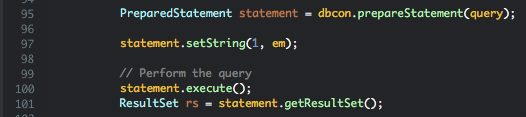
browsegenre.java checkccinfo.java

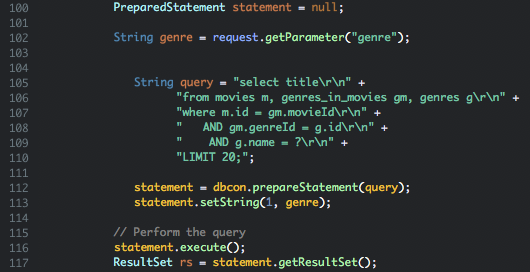
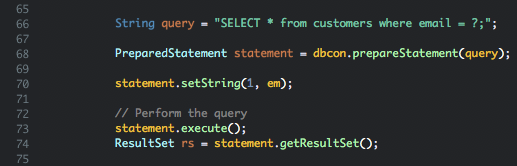
dLogin.java finalcheckout.java

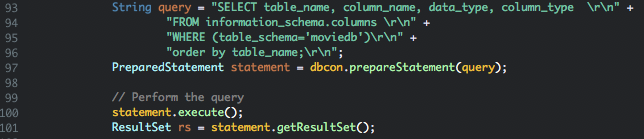
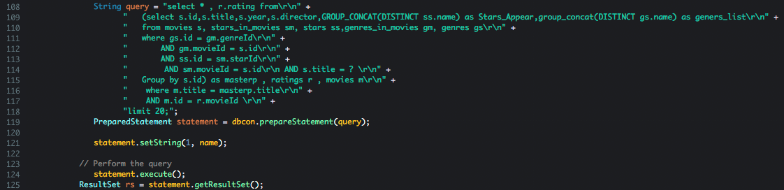
genresearch.java login.java

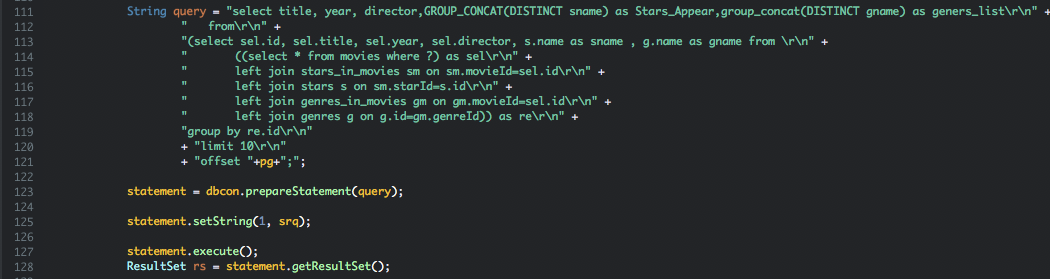
loginapp.java metadata.java

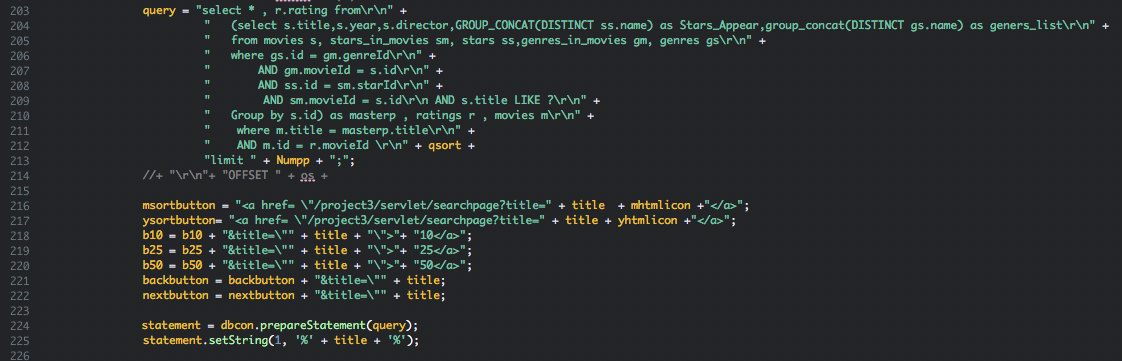
 

movieinfo.java search\_page\_app.java

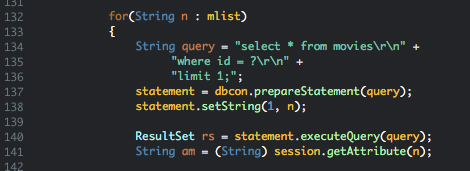
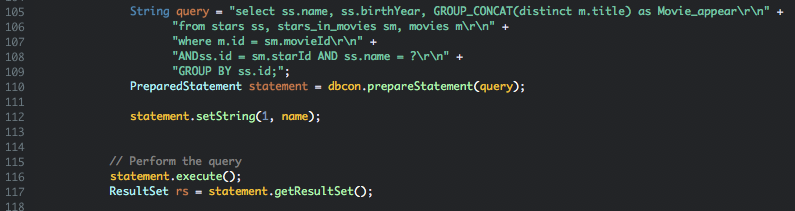
 

searchpage.java





shoppingcart.java starinfo.java

Task 2

Address of AWS and Google instances

AWS address: 52.53.153.231/project3/servlet/welcome

Google address: 35.230.118.114/project3/servlet/welcome

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

Yes, I had verified that they are accessible.

How connection pooling works with two backend SQL?

The java code has 2 pooling resources that are the master and slave server. The pooling for these two connections allows the servlet to choice which connection is open when doing query operation.

File name, line numbers as in Github

Context.xml in the project3/WebContent/META-INF

Line 4 - 13

Snapshots



How read/write requests were routed?

File name, line numbers as in Github

Snapshots

Task 3

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

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

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

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