Team 20 Xinyue Gong Yishan Ma
#91045960 #61483923

Task 1

- How did you use connection pooling?
- Firstly, modify the web.xml in WEB-INF: add resource reference name, and resource type.
- Then, create context.xml in META-INF: create a Resource element, and put name, type, user name and password in it.
- Thirdly, obtain environment naming service, use context.loopup() to retrieve the name of the object, in this case, it is “java:comp/env”.
- Finally, look up the data source, and get the connection from data source.
- File name, line numbers as in Github

File Name	Line Numbers
/WebContent/WEB-INF/web.xml	13-26
/WebContent/META-INF/context.xml	1-8
/src/cs122B/FabFlix/Servlet/Browsing.java	90-93
/src/cs122B/FabFlix/Servlet/SearchQ.java	55-58
/src/cs122B/FabFlix/Servlet/SearchingPage.java	58-61
/src/cs122B/FabFlix/Servlet/SingleMovie.java	69-83
/src/cs122B/FabFlix/Servlet/SingeStar.java	68-71
/src/cs122B/FabFlix/Servlet/TestLog.java	58-61

- Snapshots
  (Since the connection in servlets are the same, only put the snapshot of Browsing.java)

```
<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/TestDB
      </res-ref-name>
  <res-type>
                    javax.sql.DataSource
      </res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

Figure 1 web.xml snippets

```
<Context path="/TomcatTest">

        <Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
                maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
                password="1234" driverClassName="com.mysql.jdbc.Driver"
                url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true

</Context>
```

Figure 2 context.xml snippets

```
                PrintWriter out = response.getWriter();
                try {
                        //conn = JDBCUtil.getConn();
                        //
                        Context initCtx = new InitialContext();
            Context envCtx = (Context) initCtx.lookup("java:comp/env");
            DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
            conn = ds.getConnection();

                } catch (Exception e1) {
                        e1.printStackTrace();
                }
```

Figure 3 Browsing.java snippets

- How did you use Prepared Statements?
- Firstly, modify context.xml in META-INF: set cachePrepStmts to true.
- Then, create a PreparedStatement Object.
- Then, Supplying Values for PreparedStatement Parameters.
- Finally, Executing PreparedStatement Objects.
- File name, line numbers as in Github

File Name	Line Numbers
/WebContent/META-INF/context.xml	6
/src/cs122B/FabFlix/Servlet/Browsing.java	151-186

- Snapshots

```
<Context path="/TomcatTest">

        <Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
                maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
                password="1234" driverClassName="com.mysql.jdbc.Driver"
                url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true" />

</Context>
```

Figure 4 context.xml snippets(2)

```
sql = "select Idmain as id, title, year, director, rating, group_concat(distinct genres.name separator '; ') as genres, group_concat(di
                "from (select movies.id as Idmain, movies.title, movies.year, movies.director, r.rating "
                + "from movies , (select movieId, rating "
                + " from ratings)as r "
                + " where movies.id = r.movieId and (MATCH (title) AGAINST (? IN BOOLEAN MODE)) and (movies.year = ? or ? = '') and (mo
                + " order by " +sort+" desc "
                + " limit "+lowerBound+", "+upperBound+") as tempmovies, genres_in_movies, genres, stars_in_movies, stars "
                + "where genres_in_movies.movieId = Idmain and genres.id = genres_in_movies.genreId and stars_in_movies.movieId = Idmai
                + "group by  Idmain, title, year, director, rating "
                + " order by "+sort+" desc";
//search by detail
try {
        pst = (PreparedStatement) conn.prepareStatement(sql);
        if(movieTitle.contains(" ")) {
                String[] curr = movieTitle.split(" ");
                String newMovie = "";
                for(String temp:curr) {
                        if(temp.equals("of")||temp.equals("the")||temp.equals("an"))
                                continue;
                        newMovie += "+"+temp+"* ";
                }
                pst.setString(1, newMovie);
        }else
                pst.setString(1, movieTitle);
        pst.setString(2, search_year);
        pst.setString(3, search_year);
        pst.setString(4, search_director);
        pst.setString(5, search_director);
        pst.setString(6, search_star);
        pst.setString(7, search_star);
    System.out.println(pst.toString());
        rs = pst.executeQuery();
} catch (SQLException e) {
        e.printStackTrace();
}
```

Figure 5 Browsing.java for PreparedStatement

Task 2

- Address of AWS and Google instances
- Google: 35.196.36.46
- AWS: Master: 13.59.13.137
  Slave: 18.219.104.46
  Original: 18.216.105.36

- Have you verified that they are accessible? Does Fablix site get opened both on Google's 80 port and AWS' 8080 port?
- Have verified accessibility.
- The site opens for all the ports. (See snapshots below)

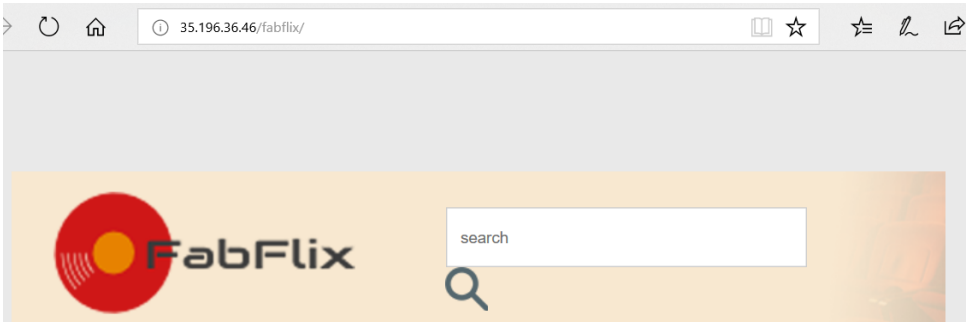

Figure 6 Google Cloud Instance

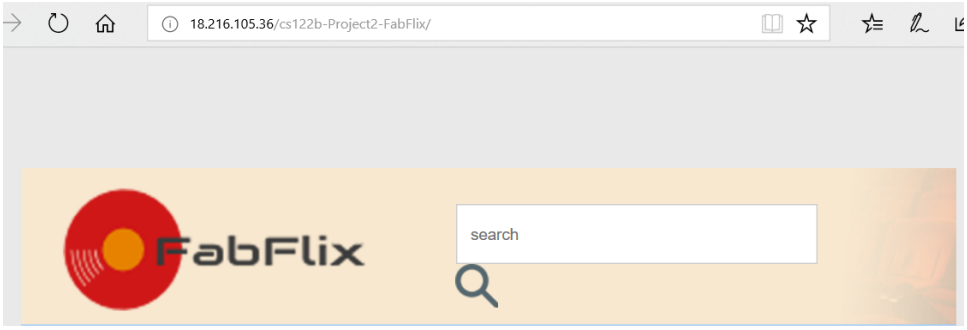

Figure 7 AWS Instance

- How connection pooling works with two backend SQL?
- Modify the 000-default.conf file, in particular, the setting for LoadBalancer.
- In ProxySet, enable the sticky session.
- In Proxy, name it Fabflix_balancer, and use the IP for instance 2 and 3 to route. And set sticky session routeId.
- Also set the ProxyPass and ProxyPassReserve accordingly.
- File name, line numbers as in Github

File Name	Line Numbers
/WebContent/WEB-INF/web.xml	13-26
/WebContent/META-INF/context.xml	1-8
/src/cs122B/FabFlix/Servlet/Browsing.java	90-93
/src/cs122B/FabFlix/Servlet/SearchQ.java	55-58
/src/cs122B/FabFlix/Servlet/SearchingPage.java	58-61
/src/cs122B/FabFlix/Servlet/SingleMovie.java	69-83
/src/cs122B/FabFlix/Servlet/SingeStar.java	68-71
/src/cs122B/FabFlix/Servlet/TestLog.java	58-61

- Snapshots

- Since we cannot upload the 000-default.conf to github, the snapshot is given below. The connection polling code snapshot is already given in task 1.

```
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED

<Proxy "balancer://FabFlix_balancer" >
    BalancerMember "http://172.31.33.47:8080/cs122b-Project2-FabFlix" route=1
    BalancerMember "http://172.31.43.86:8080/cs122b-Project2-FabFlix" route=2
ProxySet stickysession=ROUTEID
</Proxy>

<Proxy "balancer://Session_balancer">
    BalancerMember "http://172.31.33.47:8080/Session" route=1
    BalancerMember "http://172.31.43.86:8080/Session" route=2
ProxySet stickysession=ROUTEID
</Proxy>

<Proxy "balancer://TomcatTest_balancer">
    BalancerMember "http://172.31.33.47:8080/TomcatTest/"
    BalancerMember "http://172.31.43.86:8080/TomcatTest/"
</Proxy>

<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ProxyPass /TomcatTest http://172.31.33.47:8080/TomcatTest/
        ProxyPassReverse /TomcatTest http://172.31.33.47:8080/TomcatTest/

        ProxyPass /Session balancer://Session_balancer
        ProxyPassReverse /Session balancer://Session_balancer

        ProxyPass /cs122b-Project2-FabFlix balancer://FabFlix_balancer
        ProxyPassReverse /cs122b-Project2-FabFlix balancer://FabFlix_balancer


        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
```

Figure 8 000-default.conf

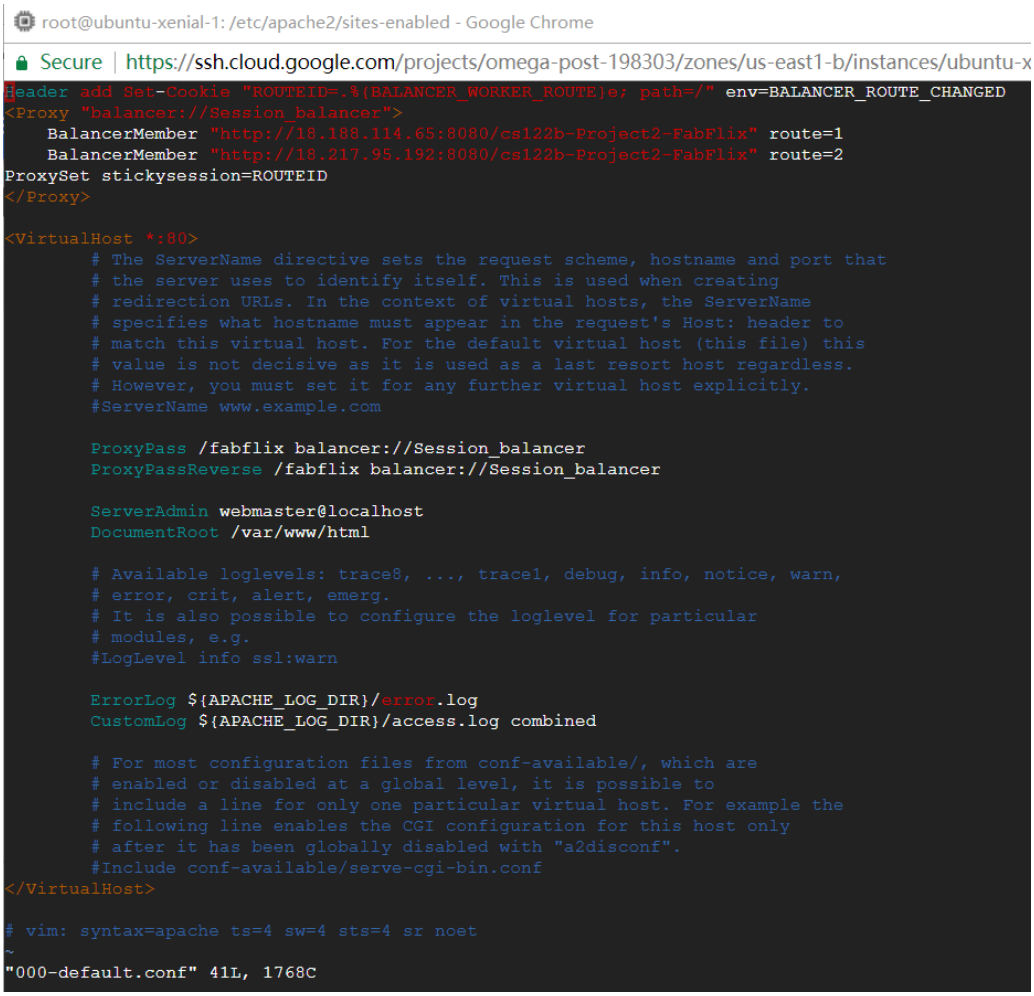

Figure 9 000-default.conf for Google Cloud

- How read/write requests were routed?
- The operation of scaling FabFlix requires a load balancer to distribute the request from user to two-backend instances. The two mysql databases in the two instances has master-slave relationship between them, in which the write operation in the Master mysql will propagate and repeat in the Slave mysql through a log file. The Slave will not propagate the writing operation to the master. Write in slave will break the synchronization between master and slave.
- The extra connection pooling resource provide a direct connection for both instances to connect to the master mysql database. When there is a writing operation in slave's servlet, the servlet will directly connect with Master's mysql and perform the writing.
- Both instance2 and instance3 has the same version of scaled FabFlix deployed, each with three connection pooling resources. The servlet will choose which connection pooling to use based on type of request.
- File name, line numbers as in Github

File Name	Line Numbers
/src/cs122B/FabFlix/Servlet/Main_Employee.java	66-69
/WebContent/META-INF/context.xml	8-11
/WebContent/WEB-INF/web.xml	20-25

- Snapshots

```
65
66                                  Context initCtx = new InitialContext();
67                          Context envCtx = (Context) initCtx.lookup("java:comp/env");
68                          DataSource ds = (DataSource) envCtx.lookup("jdbc/Master");
69                          Connection conn = (Connection) ds.getConnection();
```

Figure 10 Main_Employee.java snippets

```
1  <Context path="/TomcatTest">
2
3          <Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
4                  maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
5                  password="1234" driverClassName="com.mysql.jdbc.Driver"
6                  url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true" />
7
8          <Resource name="jdbc/Master" auth="Container" type="javax.sql.DataSource"
9                  maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
10                 password="1234" driverClassName="com.mysql.jdbc.Driver"
11                 url="jdbc:mysql://13.59.13.137:3306/moviedb?autoReconnect=true&amp;useSSL=false" />
12
13         <Resource name="jdbc/EmpDB" auth="Container" type="javax.sql.DataSource"
14                 maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
15                 password="1234" driverClassName="com.mysql.jdbc.Driver"
16                 url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false" />
17
18 </Context>
```

Figure 11 context.xml snippets

```
12
13     <resource-ref>
14       <description>TestDB</description>
15       <res-ref-name>jdbc/TestDB</res-ref-name>
16       <res-type>javax.sql.DataSource</res-type>
17       <res-auth>Container</res-auth>
18     </resource-ref>
19
20      <resource-ref>
21         <description>MASTERDB Connection</description>
22         <res-ref-name>jdbc/Master</res-ref-name>
23         <res-type>javax.sql.DataSource</res-type>
24         <res-auth>Container</res-auth>
25      </resource-ref>
```

Figure 12 web.xml snippets

Task 3

- Have you uploaded the log file to Github? Where is it located?
- cs122b-winter18-team-20/log_file/tj.txt
- cs122b-winter18-team-20/log_file/ts.txt

- Have you uploaded the HTML file to Github? Where is it located?
- cs122b-winter18-team-20/jmeter/**jmeter_report.html**

- Have you uploaded the script to Github? Where is it located?
- cs122b-winter18-team-20/cal_avg.py

- Have you uploaded the WAR file and README to Github? Where is it located?
- cs122b-winter18-team-20/cs122b-Project2-FabFlix.war
- cs122b-winter18-team-20/README.md