CPSC 476 Problemset 2
Daniel Jordan
daniel_jordan@csu.fullerton.edu

1. Using the database from Problem Set 1, write a program using JDBC to list questions and answers in the format shown here.Connect using DriverManager.getConnection() and a JDBC URL as shown in The Java SE Tutorials.

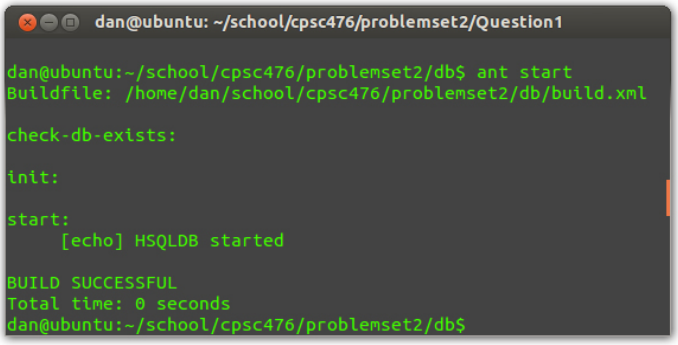
Connection to the database was made with DriverManager.getConnection(), with the following parameters:
driver: org.hsqldb.jdbcDriver
url: jdbc:hsqldb:hsql://localhost/cpsc476;ifexists=true
user: SA
pass: Passw0rd

Dependencies are added to my ant buildfile, to download the necessary files for HSQLDB 2.3.0

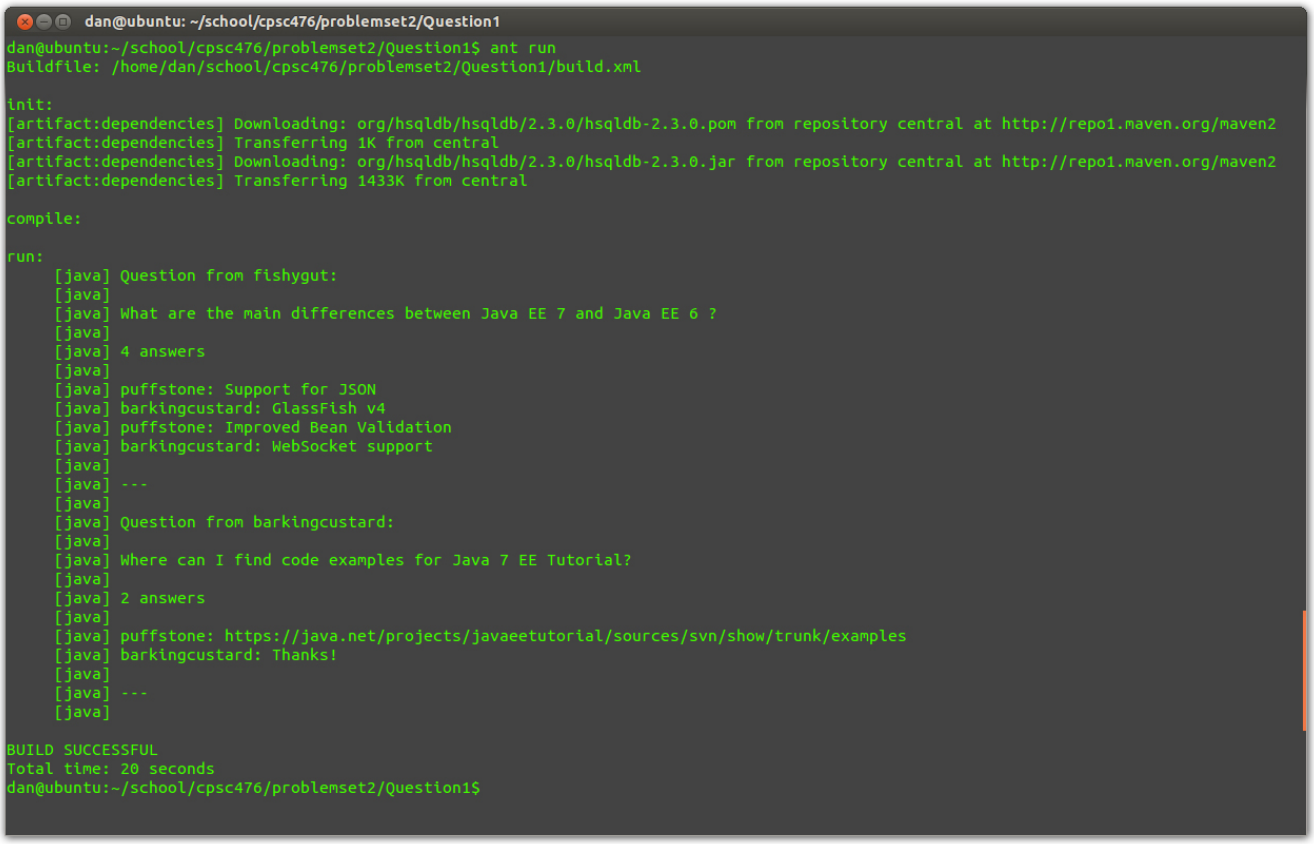
A java.sql.Statement object is created on the database connection. This statement object then carries out the SQL queries on the HSQL database.

The results of the query are gathered in a ResultSet object, which is looped through to display the correct data

Starting the database



Running the program



3. Write an HttpServlet at the URL /list to list questions and answers. Use a format similar to the output of Problems 1 and 2, but with appropriate HTML tags. Deploy your servlet to the Jetty servlet container at the context root /questions so that the page can be viewed at http://localhost:8080/questions/list.

I used the same code to connect to and query the database as in question 1. Spring was used to create a decoupled database reader. I used your GreetServlet.java file to deploy the app to /list.

A StringBuilder object is used to hold all of the output. All of the corret html tags are inserted in place. The string is then returned, and output when an IReader object is called by the servlet.

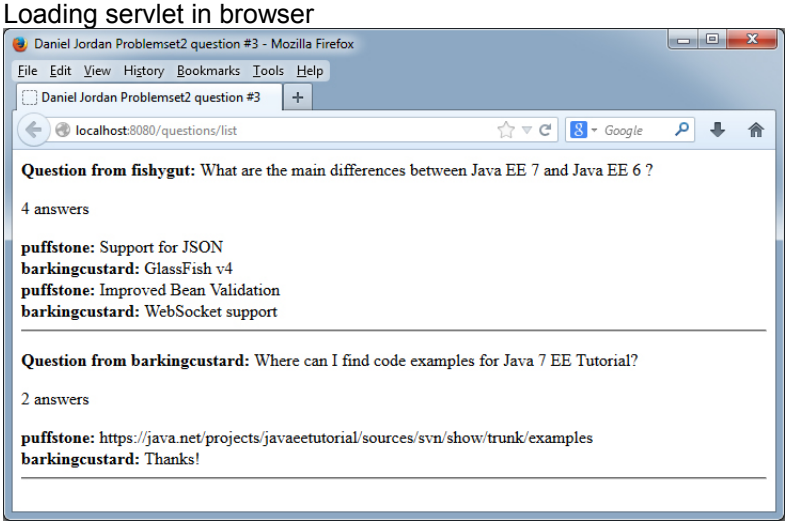
My buildfile contains dependencies for the javax servlet api 3.0.1, spring 3.2.4, and hsqldb 2.3.0. After compiling, a questions.war file is created. To deploy, I copy the questions.war file to webapps.demo inside my jetty folder.

After running the command java -jar start.jar from within my jetty folder, the server is started, and http://localhost:8080/questions/list loads the servlet.

Creating war file

```
dan@ubuntu: ~/school/cpsc476/problemset2/Question3
dan@ubuntu:~/school/cpsc476/problemset2/Question3$ ant dist
Buildfile: /home/dan/school/cpsc476/problemset2/Question3/build.xml

init:
    [mkdir] Created dir: /home/dan/school/cpsc476/problemset2/Question3/classes
    [mkdir] Created dir: /home/dan/school/cpsc476/problemset2/Question3/dist

compile:
    [javac] Compiling 3 source files to /home/dan/school/cpsc476/problemset2/Question3/classes

dist:
      [war] Building war: /home/dan/school/cpsc476/problemset2/Question3/dist/questions.war

BUILD SUCCESSFUL
Total time: 1 second
dan@ubuntu:~/school/cpsc476/problemset2/Question3$
```

Loading servlet in browser



4. Read Chapter 1 of the ObjectDB Developer's Guide, then create JPA Entities and Relationships for users, questions, and answers.Write a Java program to populate an ObjectDB database with the same questions and answers as insert-qanda-data.sql.

JPA entity classes are created for the User, Question, and Answer tables created in the HSQL db from question 1. A one-to-many relationship is created from User->Answer, and User->Question, to allow users to ask multiple questions, and submit multiple answers per question. The correct fields are created, and setter/getter functions are created. @Id annotations are used to denote primary key, and @OneToMany annotations used for 1:N relationships.

CreateEntityManagerFactory() is called, to create a questions.odb file within the db folder. An entity manager is created, and the db transaction begins.

The constructor sets the fields for each entity class, so on inserting data, a new entity object is created, and then persisted to the database. After persisting all of the data, the transaction is committed.

My buildfile contains a remote repository, to connect to ObjectDB's maven repository. ObjectDB 2.4.0 is then added as a dependency.

It's not asked to do so in this question, but I output the raw data, just to verify it was entered correctly into the database.

```
dan@ubuntu: ~/school/cpsc476/problemset2/Question4
dan@ubuntu:~/school/cpsc476/problemset2/Question4$ ant run
Buildfile: /home/dan/school/cpsc476/problemset2/Question4/build.xml

init:
[artifact:dependencies] Downloading: com/objectdb/objectdb/2.4.0/objectdb-2.4.0.pom from repository objectdb at http://m2.objectdb.com
[artifact:dependencies] Transferring 1K from objectdb
[artifact:dependencies] Downloading: com/objectdb/objectdb/2.4.0/objectdb-2.4.0.jar from repository objectdb at http://m2.objectdb.com
[artifact:dependencies] Transferring 1360K from objectdb

compile:

run:
     [java] Inserting data......
     [java]
     [java]
     [java] Verifying data......
     [java]
     [java]
     [java] Users
     [java] Id: 1, Username: fishygut, Password: password
     [java] Id: 2, Username: puffstone, Password: qwerty
     [java] Id: 3, Username: barkingcustard, Password: welcome
     [java]
     [java] Questions
     [java] ID: 1, User ID: 1, Text: What are the main differences between Java EE 7 and Java EE 6 ?
     [java] ID: 2, User ID: 3, Text: Where can I find code examples for Java 7 EE Tutorial?
     [java]
     [java] Answers
     [java] User ID: 2, Question ID: 1, Text: Support for JSON
     [java] User ID: 3, Question ID: 1, Text: GlassFish v4
     [java] User ID: 2, Question ID: 1, Text: Improved Bean Validation
     [java] User ID: 3, Question ID: 1, Text: WebSocket support
     [java] User ID: 2, Question ID: 2, Text: https://java.net/projects/javaeetutorial/sources/svn/show/trunk/examples
     [java] User ID: 3, Question ID: 2, Text: Thanks!
     [java]
     [java] Closing database connection......
     [java]
     [java]

BUILD SUCCESSFUL
Total time: 4 seconds
```

5. Generate the same question report as Problems 1 and 2 using JPA instead of JDBC or Spring.

For question 5, I used the same entity classes used in question 4. Just like question 4, an EntityManagerFactory object is created, to create an entity manager factory on the file “db/questions.odb”.

To be safe, in case you ran the problems out of order, a copy of the object db file created in question 4 was placed in the question5 folder for this program.

The program begins by querying all of the question objects in the database. The results are put into a list of type “Question”, and are iterated through. On each question iteration, a query is ran to select the username for the user who asked the question. A single result query is used to do this, since a question can only be asked by a single user. I opted to use JPQL instead of the Criteria API, so dynamic “WHERE” clauses are inserted after creating the query, by calling setParameter on the query object. This allows me to retrieve the username for each question, using one query.

Using the same method as the previous query, a count is done on the questions who's question_id matches that of the current question being looped through. Since this is a single integer, a single result query is executed.

To retrieve the answers, a TypedQuery object of type “Answers” is created, to store a list of Answer objects returned from the database. The query returns all of the answers who's question_id matches that of the current question being looped through.

The answers are then looped through, which a single result query being executed to find the user who's id matches the user_id returned by the getUser_id() method of the current Answer object being iterated through.

Once each answer is looped through, the Question loop repeats and queries the next question.

The same buildfile used for question4 is used for this question. The same objectdb repository and dependency is used.

```
dan@ubuntu: ~/school/cpsc476/problemset2/Question5
dan@ubuntu:~/school/cpsc476/problemset2/Question4$ cp -r db ../Question5
dan@ubuntu:~/school/cpsc476/problemset2/Question4$ cd ../Question5
dan@ubuntu:~/school/cpsc476/problemset2/Question5$ ant run
Buildfile: /home/dan/school/cpsc476/problemset2/Question5/build.xml

init:

compile:
    [javac] Compiling 3 source files to /home/dan/school/cpsc476/problemset2/Question5/classes

run:
     [java] Question from: fishygut
     [java]
     [java] What are the main differences between Java EE 7 and Java EE 6 ?
     [java]
     [java] 4 answers
     [java]
     [java] puffstone: Support for JSON
     [java] barkingcustard: GlassFish v4
     [java] puffstone: Improved Bean Validation
     [java] barkingcustard: WebSocket support
     [java]
     [java]
     [java] Question from: barkingcustard
     [java]
     [java] Where can I find code examples for Java 7 EE Tutorial?
     [java]
     [java] 2 answers
     [java]
     [java] puffstone: https://java.net/projects/javaeetutorial/sources/svn/show/trunk/examples
     [java] barkingcustard: Thanks!
     [java]
     [java]

BUILD SUCCESSFUL
Total time: 1 second
```