Sam Alcosser

Dr. Juan Arias

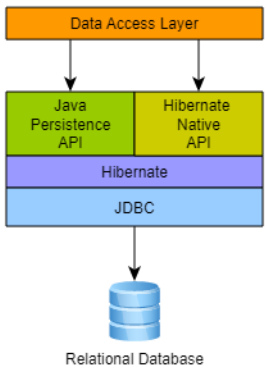
Software Development 1

7 May 2019

Workout Tracker

The goal of this project was to set out and make a free, streamlined, and easy to use workout tracker. That is, if that person is able to use eclipse to compile the project along with running a MySQL server in the correct configuration.

From the start, I knew that my goal was to make something that I could see myself using daily to track workouts. I have tried other apps but they are either too complicated or require entry fees or recurring costs. Through heavy research and hours scouring StackOverflow.com, I was able to come up with a working version of my workout tracker. The features of this tracker, though quite simple, work. Since this is a database project, this project is not as simple as putting the various .java files into a package and running them, there are things like configuration files, third party libraries, and databases that need to be set up correctly before using.



In order to understand how this application works, an understanding of this diagram is necessary. To put it simply, it IS possible to connect directly to an SQL database through java using JDBC, or Java Database Connectivity. Although, if one was to choose this way, that would mean writing out each and every lengthy query in pure SQL. There is an easier way. Referring to the diagram, it is possible to use a third-party library called Hibernate. This may seem overly complex, but in hopes to complete this project the *real* way I decided to utilize it. How hibernate works, is by using a configuration file which is the connection to JDBC, and using mapping classes. Mapping classes are classes which are each

supposed to represent an entity. In database terms, an entity is to a record as a class is to an object. So, one mapping class is necessary for each database table that is desired to be connected to. These mapping classes are used to relate the query results from JDBC, which Hibernate is interacting with, to actual usable data for java. To restate, Hibernate is not a workaround from JDBC, Hibernate just simplifies CRUD by being an in-between for java code and JDBC.

To go into every process that occurs within this program would fill up much more than 8 pages, so for brevity, the general processes will be discussed.

For example, one longer process that is in this program is the process of saving a new user on the sign-up page. Instead of writing out a lengthy SQL query with all of the different fields and all of the different parts of a user entity, we can simply make a new user object with the constructor for the user, and then by simply writing the line "`session.save(newUser);`", followed by "`session.getTransaction.commit();`" we are sending our object to Hibernate, where our mapping class interprets the object and then sends the data to JDBC to be processed. To be more detailed, the sessions are part of objects called session factories. These session factories are the things that are making the actual connection to the right table. For this reason, throughout the source code for this projects, some of the windows may require only one session factory, say to the User database for something like the login page or the sign up page. For more complex windows, such as the tracker window, three separate session factories are required for the connections to the User,exerciseDone, and exercise tables. A session is then created off of these session factories, and each of these sessions has their own transaction. What this means is that when, for example, the newUser object is saved, this means it is saved in memory and not the database. Until the transaction is committed, none of the changes have been made. Also, there is only one transaction per session.

For a better understanding of the various mapping classes and application classes used, please follow the link https://www.lucidchart.com/invitations/accept/003e77ad-e665-4b08-9b2a-bf8a555ae625 to view the UML documentation for all classes. Other than one exception being the injury recording window, the parts of this project can be separated into two main groups, viewing data, and recording data.

Processes that would fall into the category of viewing data would be things like the metrics functions, viewing previous workouts, and viewing account info. These operate quite simply, usually starting with a query to the database for information. Take for example the “my workouts” button. When that is pressed, a window is opened that immediately pulls up all of the workouts associated with the logged in user. It knows the right user because throughout most of the processes in this program, an object called cUser is passed to the constructor of the next window, or frame. This is so that the processes are linked to the right user, and if needed, data can be pulled directly from the object to be displayed on the screen.

Because of the way that Hibernate works, much of the data is compiled at runtime so that the data is up to date. For example, there is the functionality behind viewing previous workouts. This is done mainly through querying the workout and exerciseDone tables, while referencing the exercise table. Say for example that a user wants to view their workouts. When they press the my workouts button, a query is sent to the databases, through Hibernate, for all workouts with the ID of the cUser. This functionality is possible because the workouts are all marked to the user through taking the PK of that user and using it as an FK to mark the workout. This way, it is easy to find the workouts that correspond to each user. Once those workouts are found, they are displayed in a list for the user to select. Here, yet another window is open where a query is sent to find all of the exercises done under the scope of the selected workout. Again, the PK of the

workouts are used to mark the exercisesDone with an FK so that they can be traced back to the right workout, and with some work, back to the user. All of this functionality is expedited by Hibernate because it allows for the programmer to use HQL, which is a simplified form of SQL designed for interacting with java. This can make some queries that would otherwise be tens of lines down to no more than four or five lines of code, in the case of create statements.

The view account info button is quite simple. It doesn't even interact with Hibernate. Due to the fact that most of the windows are passed the cUser object, the window just pulls the data about the user using the getter methods built into the mapping class.

The metrics functions are more complex. The window starts out quite similar to any other, querying the tables for data about the exercises done within a workout. Although, the processing is quite complex. Using mildly complex algorithms, the program finds all of the distinct exercises that the user has done, then multiplies the reps and sets of each exercise done of that exercise to find the "best exercise" that was done. These exercises done that have the maximum reps are then sent to a list as a set of strings to then be displayed. Also, as a side function, there is a dropdown list where the user can select any exercise and see what their max weight used was. This was one of the many places that a try catch statement was used. In this case, to prevent an error occurring when a user tries to see the max weight used for an exercise they have never done. One very small function that may go overlooked is embedded into the sign-up page. In this page, I have added a function that lets the user see if their desired username has been taken. This does a simple HQL query to see if that username exists in the User table.

Now to move onto the recording functions in this program. These consist of the tracker, and the sign up page. As the functionality behind the sign up page has already been discussed, I will cover how the tracker works. The processing of the tracker begins even before the tracker

window is initialized. When the user clicks the button on the homepage to track a workout, a new workout object is created and sent to the workout table. This function is essential to the processing of the tracker. The workout object sent just consists of the user id and the timestamp of when the button was pressed. The reason why this is so essential is because all of the exercises that the user will track in the next page need to be linked to a workout to have any meaning. Because of this the workout needs to be created beforehand. Now, once the user is into the tracker window, they choose the exercise done, the reps, sets, and weights if applicable. Also as a side note, the user has the opportunity to add new exercises to the database through the home screen. The exercises are then packaged into exerciseDone objects, to be later processed by a mapping class. When the user presses add exercise, they are not saving to the database, but rather to memory. The exercises are only committed to the database once the save workout button is pressed.

One thing that was glanced over that deserves more focus is how the mapping class works for the exercise done class. Relationaly, these exerciseDone objects have a many to one relation to the workouts. For this, special notation is necessary. In the mapping class, the workout is not saved by the workout id, and within the notation hibernate is told which piece of the workout to take as the foreign key.

To briefly mention the MySQL side, there is simply four tables, exercises, exercisedone, workouts, and users. Within these, there are primary keys and foreign keys mapping just as seen in the java code.

The problem that this project was addressing was making a simple workout tracker that was easy to use. To achieve this, that meant having a way to track workouts with detail, be able to view these workouts, and see basic statistics or metrics about the data. This

program achieves this by doing exactly that. Allowing for users to track workouts, view metrics about them, and see their best over time. The one part that was left out in the final version was the ability to view progress over time, or goal tracking. The reasoning for this is purley because coming down to the end of the wire, I decided between more functionality, or more time testing and debugging for the final project. As a whole, I stand by my decision as it has created a well-functioning, fairly stable end product.

As previously mentioned in the milestone writeup, there are mountains of workout trackers out there, but most involve paying a price premium up front or a recurring cost. One that was described was called *HeavySet.* This product is an iOS app that involvs paying a $10 fee just to even unlock the capacity to track a custom workout. In my opinion, this is base functionality. The argument can be made that you are paying for those statistics and pretty graphics, but to me, this was not necessary. The problem I was facing was just to have a simple way to track my workouts, and in my opinion, paying nothing for something that fully achieves the requirements is better than paying a price to get mildly better features.

User Manual:

To begin, press the create user button on the login screen. Heed the warning, you must have an active connection to MySQL running for any of the functionality of this program to work. From here you will be redirected to a form where you can create your user. Also, as a side function, feel free to check if your desired username is valid. From here, you will be taken to the homepage. Some simple functions involve adding exercises, viewing account info or viewing your bests. More complicated ones would be tracking a workout or viewing past workouts. To view a past workout, press the my workouts button and then select the date of the workout you would like to view. To track a workout, press the track workout button. Here input each separate

exercise to the tracker along with the related reps and sets, or even weight in lbs. If an exercise is not listed, please go back to the home screen and add the exercise. By doing this, you allow any other user to use that exercise in their workouts as well. Now that you have filled out one exercise, press add exercise. Repeat this process until each exercise is added. At this point, press save workout. Now, you have the ability to see the workout in the my workouts page, or even see it in the my best. The my best page offers the ability to see your best exercises. Feel free to select your desired exercise from the dropdown list to see what your max weight ever used was. Once you are done, feel free to close the window. Now that you have signed up, you can login normally next time.

The goals accomplished by this system are to give people a purposefully stripped-down workout tracker that simplifies complex processes to be able to track and see past workouts. As simple as it may seem, I was able to achieve most of what I set out to. To be able to save workouts, call back on them to see them, but I have lacked a little on the metrics. Yes, there are some metrics functions like most reps and most weight, but I was forced to choose between goal tracking and better stability. In the long run, I am happy that I chose to spend the time on stability, as I can say that I am much prouder of the end result.

Works Cited

"A Visual Guide to Swing Components." *A Visual Guide to Swing Components (from: The Java™ Tutorials > Graphical User Interfaces > Swing Features)*, MIT, web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html.

Baeldung. "Hibernate One to Many Annotation Tutorial." *Baeldung*, Baeldung, 24 Dec. 2018, www.baeldung.com/hibernate-one-to-many.

"Hibernate – Cascade Example (Save, Update, Delete and Delete-Orphan)." *Hibernate – Cascade Example (Save, Update, Delete and Delete-Orphan) – Mkyong.com*, MKYong, www.mkyong.com/hibernate/hibernate-cascade-example-save-update-delete-and-delete-orphan/.

Janssen, Thorben. "Best Practices for Many-To-One and One-To-Many Association Mappings." *Thoughts on Java*, Thoughts On Java, 16 Jan. 2019, thoughts-on-java.org/best-practices-many-one-one-many-associations-mappings/.

luv2code. "Hibernate Tutorial #1 - Hibernate Overview." *YouTube*, YouTube, 1 Apr. 2017, www.youtube.com/watch?v=0hm3QidFwYY.

Mankotia, Pankaj, and Dinesh Rajput. "Many to One Mapping in Hibernate Example." *Dinesh on Java*, Dinesh on Java, www.dineshonjava.com/hibernate/hibernate-many-to-one-mapping-tutorial/.

"SQL Quick Reference from W3Schools." *SQL Quick Reference*, w3schools, www.w3schools.com/sql/sql_quickref.asp.

Tutorialspoint.com. "Hibernate Query Language." *Www.tutorialspoint.com*, Tutorialspoint, www.tutorialspoint.com/hibernate/hibernate_query_language.htm.