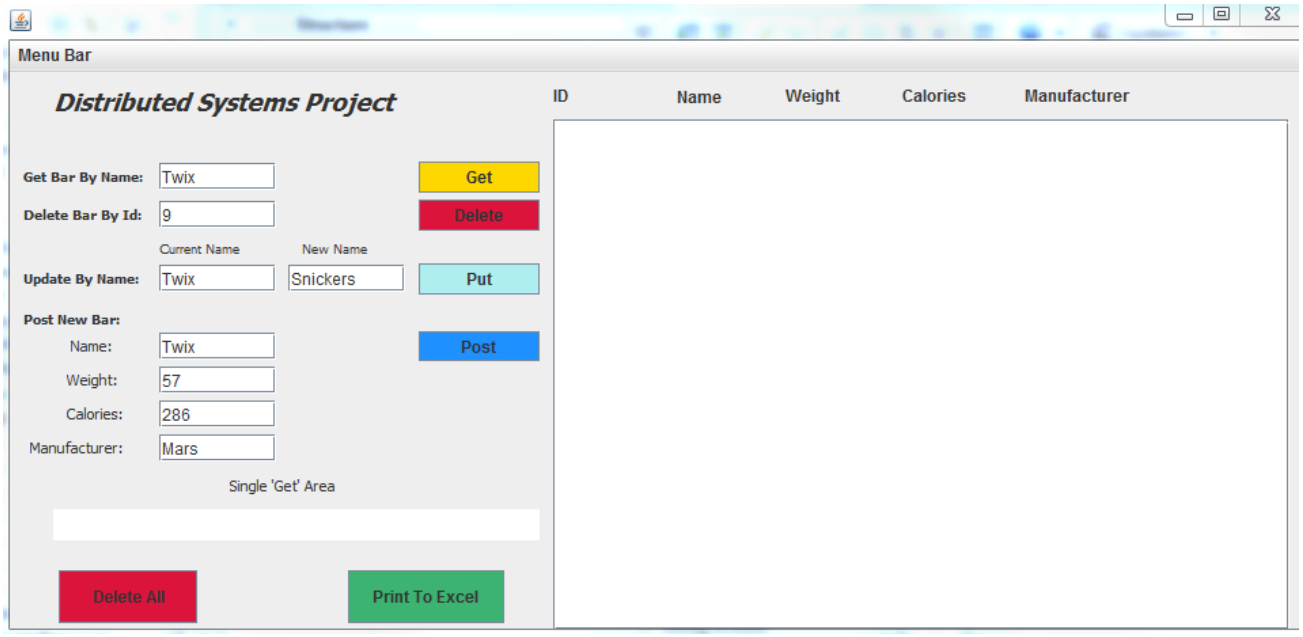
Distributed Systems Assignment 2023

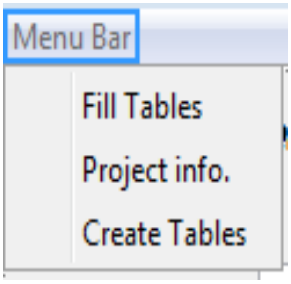
Chris Doe – A1234567

Project Overview

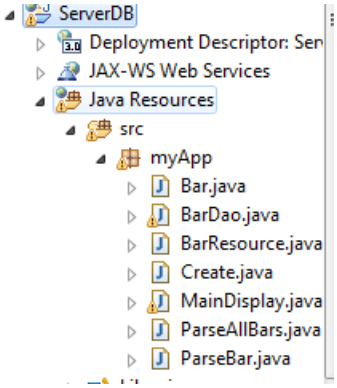
Main Window



Menu Options



Files



This project holds a database for storing and accessing information about chocolate bars and runs on a Tomcat 7.0 server. The information is stored across two HSQLDB tables, one to hold an id (primary key), Name, Weight, and calories and one to hold an id (primary key) and manufacturer. Information is retrieved and processed using an XMLPullParser to identify it using tags. The server uses the JAX:RS api to access/display the data.

Main Functions:

Get - Return the information a single entity by entering a name.

Delete - Enter an Id number and it is deleted from the system.

Put – Enter the current name of a bar and the name that it will be changed to and the table is updated.

Post – Enter the details for a new bar and create it on the server.

Delete All – Removes all data from the tables.

Print to Excel - Prints all the data from both tables to an excel sheet (comma separated values).

Fill Tables (menu bar option) - Creates three entities automatically for the table.

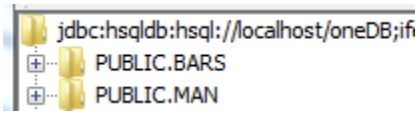
Create Tables (menu bar option) - Runs code to create the two tables (BARS and MAN).

Project Info (menu bar option) - Small popup box with student number and name.

1. Requirement

"Multiple Tables and/or tables created using code at runtime plus code to select, insert, update and delete data"

Completion



SELECT * FROM BARS JOIN MAN ON BARS.ID=MAN.ID

ID	NAME	WEIGHT	CALS	ID	MANUFACTURER
0	Crunchie	40g	187	0	Cadbury
1	Twix	57g	286	1	Mars
2	Mars Bar	58g	120	2	Mars
3	Crunchie	40g	187	3	Cadbury
4	Twix	57g	286	4	Mars
5	Mars Bar	58g	120	5	Mars

Creation SQL:

CREATE TABLE BARS(Id INTEGER IDENTITY PRIMARY KEY, Name varchar(30) NOT NULL, Weight varchar(30) NOT NULL, Cals int NOT NULL);

CREATE TABLE MAN(Id INTEGER IDENTITY PRIMARY KEY, Manufacturer varchar(30));

Steps for First Time Use:

1. **Run Tomcat Server.**
2. **Run Ant.**
3. **Menu Bar -> Create Tables.**
4. **Menu Bar -> Fill Tables.**

The first time the project is run the user must click the Create Tables option of the menu bar. This will run the create objects init() function to create the two tables. Once the tables are set up this does not need to be run again. This is probably not the best way to create the tables as it requires the user to connect directly to the server but the feature seemed more important to include than to leave out altogether. Create is the only function that works like this, every other command uses Jax:Rs and the BookDao.

The select, update, delete and insert functions are called from the BarDao (data access object) files and connect to the HSQLDB server with a direct connection. A data access object acts like an interface between the client and the HSQLDB server, allowing for functions to be called without giving away too much information about the server. This is an example of the select command for getting a single bar by its name:

```
String cmd = "SELECT * FROM BARS JOIN MAN ON BARS.ID = MAN.ID WHERE BARS.name = '"+name + "';";
ResultSet rs =stmt.executeQuery(cmd);
```

As code is not required in this write up please see the BarDao java file for more implementations of SQL commands.

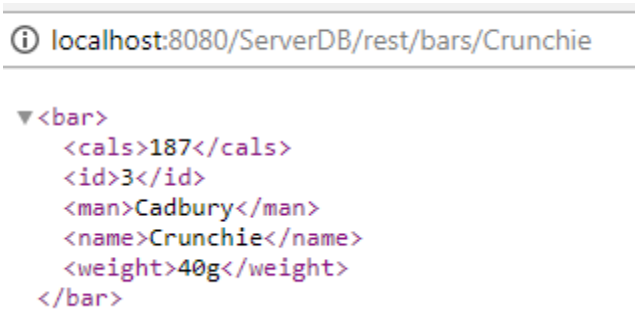
2. Requirement

"A Jax-Rs/Jersey client that sends all of the HTTP requests GET/PUT/POST/DELETE, parses the response XMLPullParser and outputs to the GUI" + "A tomcat server that responds to all of the HTTP requests GET/PUT/POST/DELETE"

Completion

Get: 2 different @GET JAX:RS commands implemented, one which simply returns all the bars stored on the server and one for returning a specific bar by name. The single get works by supplying a bar name as part of the path.

Example of a single bar returned:



As part of the get command the XMLPullParser was used to separate the information returned by its tags which can be seen in the ParseBar file. When the information is returned it is pulled apart using a series of if statements to examine which tag of the XML the parser is currently in. E.g. as the parser iterates through the specific bar it will meet the start of a tag such as <name> then it will set the text between that and the closing tag of </name> as the name for a temporary bar. The bar will be returned. When the </bar> tag is met as that is the end of information about that specific bar.

The information returned is the displayed in the Single Get Area:

The getAll works similarly but uses the ParseAllBars file. This returns a list of all the bars and is used to write to the main print screen in the same way that the single get was.

ID	Name	Weight	Calories	Manufacturer
0	Crunchie	40g	187	Cadbury
1	Twix	57g	286	Mars
2	Mars Bar	58g	120	Mars

Put: The @put command is used to update the name for a bar to a new name. It takes in the current name as a parameter and the new name that it will be changed to. These are parsed by adding them as BasicNameValuePairs to a nameValuePairs list and set as the URLEncodedFormEntity for the httpPut request.

The old and new names being read in as @FormPararm so they can be then sent as parameters to the BarDao:

```
ıpdateName(@FormParam("oldName") String oldName, @FormParam("newName") String newName)
```

Post: Takes in 4 parameters from the user for a new par and then passes to the BarResource object in a similar way to the PUT (I.e. with parameters) and then on the BarDao to create the object in SQL.

When the filltable menu item is selected it simply calls the post command 3 times with 3 set bars.

DELETE: There are two versions of the @DELETE command implemented, one that takes a name provided by the user as a parameter and one which just deletes everything in both tables. The single delete works very similarly to the @GET, except it calls a delete command in sql based on the name provided. When the Delete all command is called it will also reset the auto incrementation of the bar and man ids back to 0.

Every time a command is called other than the single get then the getAll() function is called to display the newly updated tables on the main text area screen. This was done to make sure the user gets some feedback when they make a change to the tables.

3. Requirement

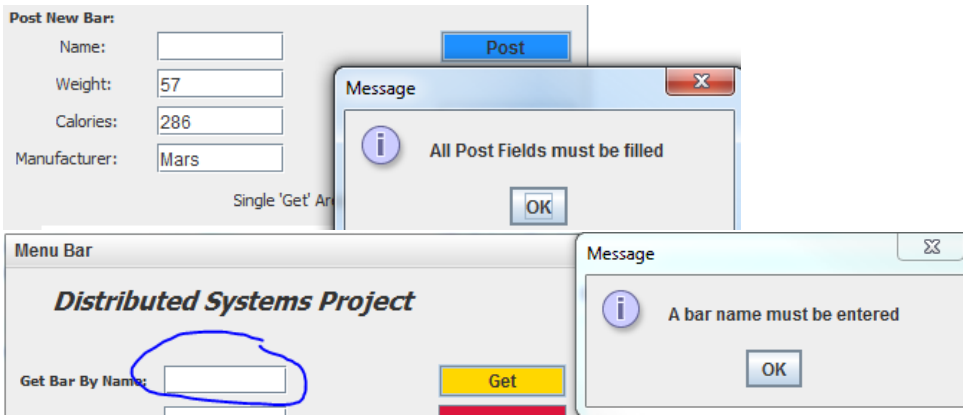
"GUI that handles the GET/PUT/POST/DELETE actions"

Completion

Most screenshots of the GUI have already been included but there are some extra features that will be highlighted here.

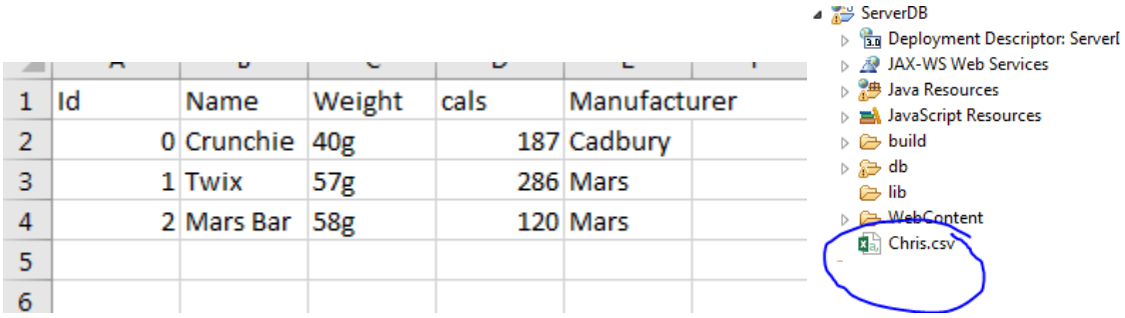
Alerts

If a user tries to call a command without entering all the inputs they need to a popup box is displayed and no call is made:



Print to Excel

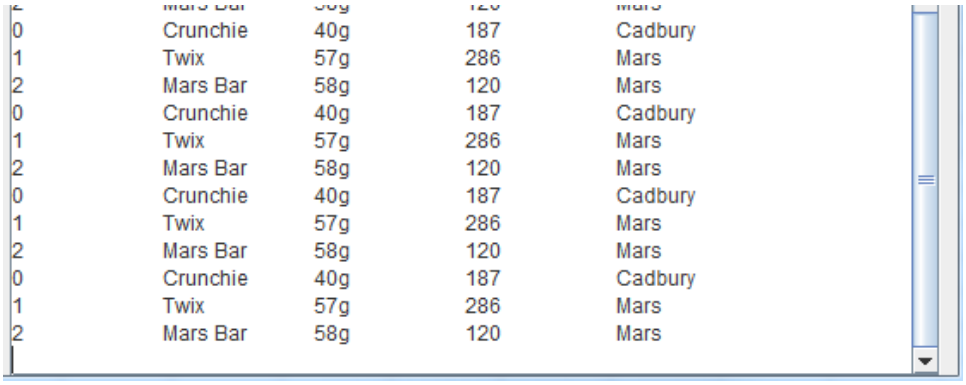
All data from both files printed to an excel file. This works the same way as the get all command to return all the bars and their info and uses the XMLPullParser to extract the data to be printed. This was added as an additional feature to increase interactivity.



	Id	Name	Weight	cals	Manufacturer
1					
2	0	Crunchie	40g	187	Cadbury
3	1	Twix	57g	286	Mars
4	2	Mars Bar	58g	120	Mars
5					
6					

Scroll Bar

If a lot of bars are added to the main text display, then a scroll bar will appear to let the user scroll through their entries.



0	Crunchie	40g	187	Cadbury
1	Twix	57g	286	Mars
2	Mars Bar	58g	120	Mars
0	Crunchie	40g	187	Cadbury
1	Twix	57g	286	Mars
2	Mars Bar	58g	120	Mars
0	Crunchie	40g	187	Cadbury
1	Twix	57g	286	Mars
2	Mars Bar	58g	120	Mars
0	Crunchie	40g	187	Cadbury
1	Twix	57g	286	Mars
2	Mars Bar	58g	120	Mars