Distributed Systems Assignment 2025

A00292349

***Main Window***

**Project Files**

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

**File menu options**

A screen shot of a computer

AI-generated content may be incorrect.

This project has a database containing information on pirates(ID, name, crew, bounty, position, age, gender). It runs on a Tomcat 8.5 server. The information is stored on a HSQL table. Information is retrieved and processed using an XMLPullParser to identfiy it using tags. The server uses the JAX:RS api to access/display the data.

**Main Functions:**

**Post:** Upload a new pirate to the server through entering its’ details

**Delete:** Enter an ID number and delete a pirate from the system

**Put:** Using the corresponding ID edit the details of the pirate (barring the ID)

**Get:** Retrieve the specific information of pirate

**Clear:** Clears the input boxes

**Show All:** Displays all pirates in the table

**ListAllBountys(Export Option):** Exports a List of all the pirates in the table

**Exit(File Menu Bar):** closes the application

**Project Info(File Menu Bar):** Makes a small popup that gives a short description of what’s contained within the database

**Fill Table(File Menu Bar):** Fills the table with 3 entities

**Clear Table(File Menu Bar):** Wipes all information contained in the database

**HSQL Database**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Note:** Bounty is the table being used in the operations of this system. The book table is irrelevant.

**Create the Table**

CREATE TABLE Bounty (

    id INT GENERATED BY DEFAULT AS IDENTITY (START WITH 1, INCREMENT BY 1) PRIMARY KEY,

    name VARCHAR(255) NOT NULL,

    crew VARCHAR(255),

    bounty INT,

    position VARCHAR(255),

    age INT,

    gender VARCHAR(50)

);

**Insert Information Into the Table**

INSERT INTO Bounty (name, crew, bounty, position, age, gender)

VALUES ('Monkey D. Luffy', 'Straw Hat Pirates', 300000000, 'Captain', 19, 'Male');

INSERT INTO Bounty (name, crew, bounty, position, age, gender)

VALUES ('Roronoa Zoro', 'Straw Hat Pirates', 100000000, 'Swordsman', 21, 'Male');

INSERT INTO Bounty (name, crew, bounty, position, age, gender)

VALUES ('Usopp', 'Straw Hat Pirates', 500000, 'Sniper, 19, 'Male');

**Show the contents of the table**

Select \* from bounty;

**A screenshot of a computer

AI-generated content may be incorrect.**

**Steps for running the application**

1. **Run the Tomcat Server**
2. **Run Ant**
3. **File Menu-> Fill Table**

When the project is first run the tables are already set up, they may need to press fill tables in order to get the premade bounties to fill into the database.

The Get/Select, Update/Put, Delete and insert/Post functions are called from the BountDAO(data access object) files and connect to the HSQLB server with a direct connection. The DAO acts as the middle man between the client and the HSQLB server, allowing for functions to be called without oversharing information about the server.

*Please see the BountyDao java file for the implementation of the SQL commands*

**Parsing and Output to GUI**

*“The client application will parses the response using XMLPullParser and outputs to the GUI" + "A tomcat server that responds to all of the HTTP requests GET/PUT/POST/DELETE"”*

**Get:** There are two different @GET JAX:RS commands within the system. There is one get which returns all the pirates stored and another which just returns a single one specified by its’ id

A screenshot of a computer

AI-generated content may be incorrect.

Figure : A single pirate bounty being returned

To move the xml format to the table within the gui, the information was parsed from the xml format as can be seen in the parsebountys file. The information is taken from the xml format through a series of if statements to see which tag of the XML the parser is currently in. As the parser moves through a specific bounty it will get the information of the bounty from between the specific tags that correspond to that info. At the closing tag when the end of the information for that specific bounty is met, the bounty is returned.

Figure 1 is the information of the full Get and Figure 2 is of the single get on a specific bounty

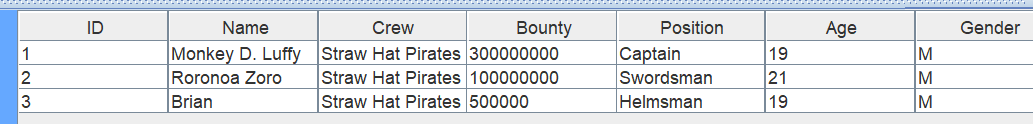


Figure : Get all items in the table

A screenshot of a computer

AI-generated content may be incorrect.

Figure : Get a single item from the table by it's id

**Put:** This is used to update the information of a pirate through the specification of it’s id.

**Post:** Takes in the info of a pirate then passes on to the BountyResource object and then the BarDao creates the object in the sql

**Delete:** There are two deletes one for a selected single object by id and a another for clearing the whole table.

**GUI**

The GUI has already been previously displayed with key functionality described. Displayed here are the extra functionalities added on with screenshots for reference.

**Alerts**

If the user misses out on in putting key info when trying to perform an operation they will be given an alert.

**A screenshot of a computer error

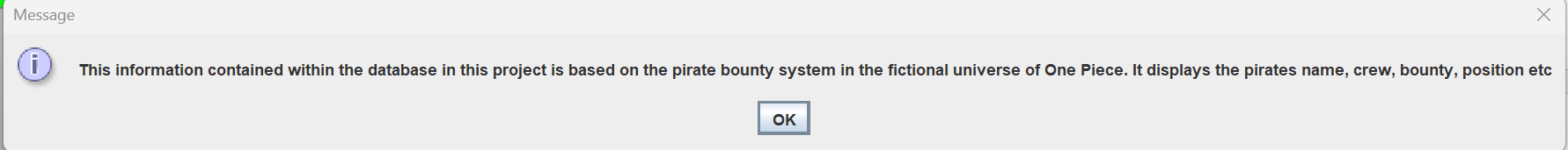
AI-generated content may be incorrect.A screenshot of a computer error message

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

**Project Info**

Similar in function to the alerts when the project info button in the File Menu is pressed this alert pops up

****

**A screenshot of a computer

AI-generated content may be incorrect.Print to Excel**

When the ListAllBountys button in the export data field is pressed an Excel sheet is generated containing the information displayed within the table.

**A screenshot of a computer

AI-generated content may be incorrect.**

**Scroll Bar**

**A screenshot of a computer

AI-generated content may be incorrect.**If a large amount of items are added to the table a scroll bar appears allowing for easy navigation of the table