Team We Them Boys' xGame Development Guide

Overview

xGame is a platform for players to face off in head-to-head matches of Legan chess. Players can create an account to create matches with other players and play their matches over time. Matches can be quit and resumed at any time, picking up where they left off. Players will be notified of new match invites, if their match invite has been accepted or rejected, when a move has been made, and when a match has completed. Players can also view their match history and view other player's profile and match history.

Architecture

xGame is a web platform consisting of a SPA React front end, a REST API backend, and a MariaDB database. The full list of technologies is below:

- Node.js
- React
- Java Spring Framework
- Maven
- JPA
- Hibernate
- MariaDB

The Java application is a REST API built in Spring Boot and makes use of N-tier architecture, with defined view, logic, and data access layers. The view layer contains all the REST API endpoint controllers that are used for communication with the front end React client. The logic layer consists of services that make up the business and system logic. Finally, the data access layer contains all the CRUD repositories and is solely responsible for all communications with the backend database. Dependency Injection is used for communication across the various layers of the application.

REST API

User

The User API can be user to search, create, soft delete, and retrieve users.

Endpoints

User Search

Searches for a user by the param. Performs a fuzzy match on the user's nickname and an exact match. Returns a User.

GET /user/search?param=searchtext

Login a User

Checks that user credentials match the given information and returns the found User. Returns a User.

GET /user/login

Request Body

UserCredentials

Register a new User

Creates a new user based on the given UserCredentials. Checks for unique nickname and email. Returns a User.

POST /user/register

Request Body

UserCredentials

Match

The match API can create new matches, get all match invitations, and accept match invitations.

Endpoints

Get match state

Gets state of match with given id. Returns the Match.

GET /match?matchId=int

Create a match

Creates a new PENDING match between the white player and the black player. The black player logically now has an invitation to a match. Returns the newly created Match.

POST /match?whiteId=int&blackId=int

Accept a match invitation

Accepts a match by changing the match status to “INPROGRESS” and setting the turn count to 1. Returns a Match with the updated match state.

PATCH /match/accept?matchId=int

Message

Endpoints

Get all messages

Gets all messages, including notifications and invitations for a single player. Returns a list of generic Message objects that are either a message or invitation. If the object is a message, the id is the messageId. If the object is an invitation, the id is the matchId of the corresponding match.

GET /message?playerId=int

Get match invitations

Gets all match invitations for the player with the given id. Returns a list of MatchInvites.

GET /message/invitation?playerId=int

Objects

Message: {

“id”: int,

“type”: string,

“content”: string,

“timestamp": string

}

Match: {

“id”: int,
“whiteId”: int,
“blackId”: int,
“whiteEmail”: string,
“blackEmail”: string,
“turnCount”: int,
“chessBoard”: string

}

User: {

“id”: int,
“nickname”: string,
“email”: string

}

UserCredentials: {

“nickname”: string,
“email”: string,
“password”: string

}

Environment Setup

Client

Software and Tools

IntelliJ IDEA – 2020-.2.3

IntelliJ is used as the integrated development environment for the client side of the application.

Node.JS

Node.JS if the engine that will allow us to run the JavaScript outside the browser

NPM

NPM is a node package manager that allows us to install packages to use in our code.

Setup Guide

1. Install Intellij IDEA
    a. Follow the steps in the wizard, keeping all settings to default
2. Install Node.JS
3. Install NPM.
4. Open Intellij and create a new project from existing version control.
5. In the Intellij terminal, navigate to the client directory.
6. Run ‘npm install’ to automatically install all the necessary packages.

7. From here the project is ready for development
    a. To test the project type ‘npm start’ this will open a browser in development mode so you can see the changes you make in the code live
    b. To build the project for deployment type ‘ npm run build’ this will create a build folder that will hold the files ready for deployment
    c. To run unit test type ‘npm test’ this will find all of the test code in the project and run it

Server

Software and Tools

Eclipse IDE for Java Developers – 2020-09

Eclipse is used as the integrated development environment for the server side of the application. Required extensions include:

- Spring Tools 4
- ObjectAid UML Explorer

Maria DB 10.3.25

MariaDB is a database management system. The server application uses MariaDB to store users, messages, and chess match information. It is necessary to stand up and instance for the development environment. A direct download link can be found below:

https://mariadb.org/download/

HeidiSQL

HeidiSQL is a database administration tool that we will use to manage and maintain the application database as necessary. HeidiSQL comes bundled with MariaDB and is highly recommended. Although any database administration tool that works with MariaDB or MySQL can be used.

GitHub Desktop (Optional)

GitHub Desktop is a tool for working with our GitHub repository. With it, you can pull down code, create branches, merge branches, and create pull requests. It is not required, and other tools can do the same job. A direct download link can be found below:

https://central.github.com/deployments/desktop/desktop/latest/win32

Postman (Optional)

Postman is a tool for testing REST API functionality. It is recommended to check that HTTP calls to the server are functioning properly. A direct download link can be found below:

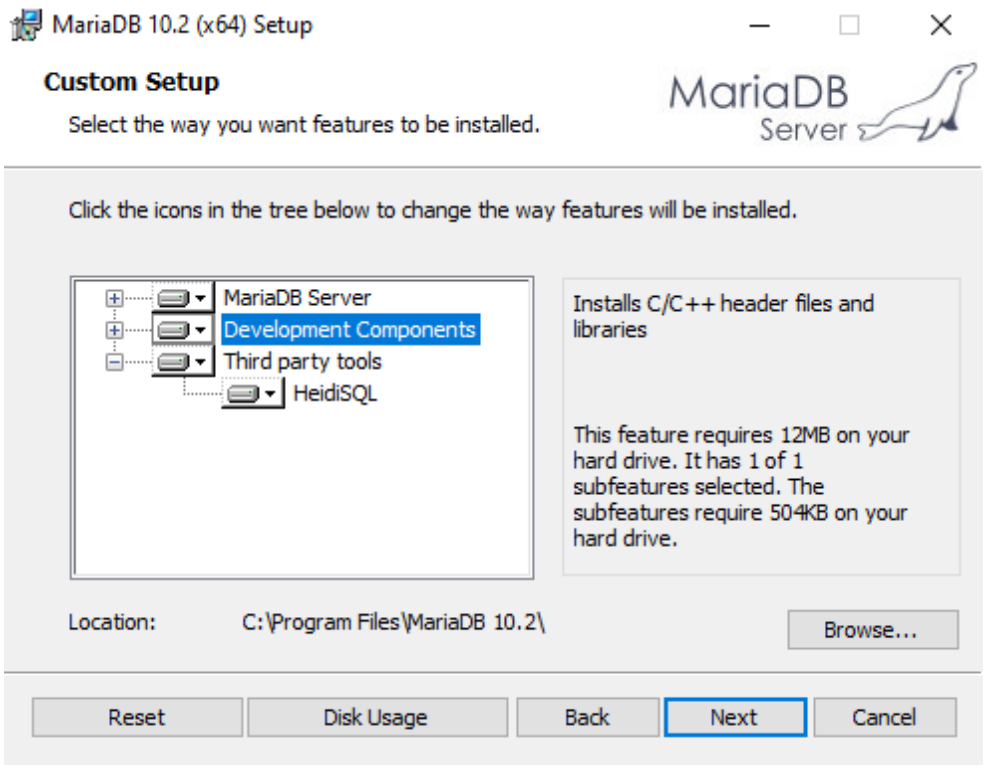
https://dl.pstmn.io/download/latest/win64

Setup Guide

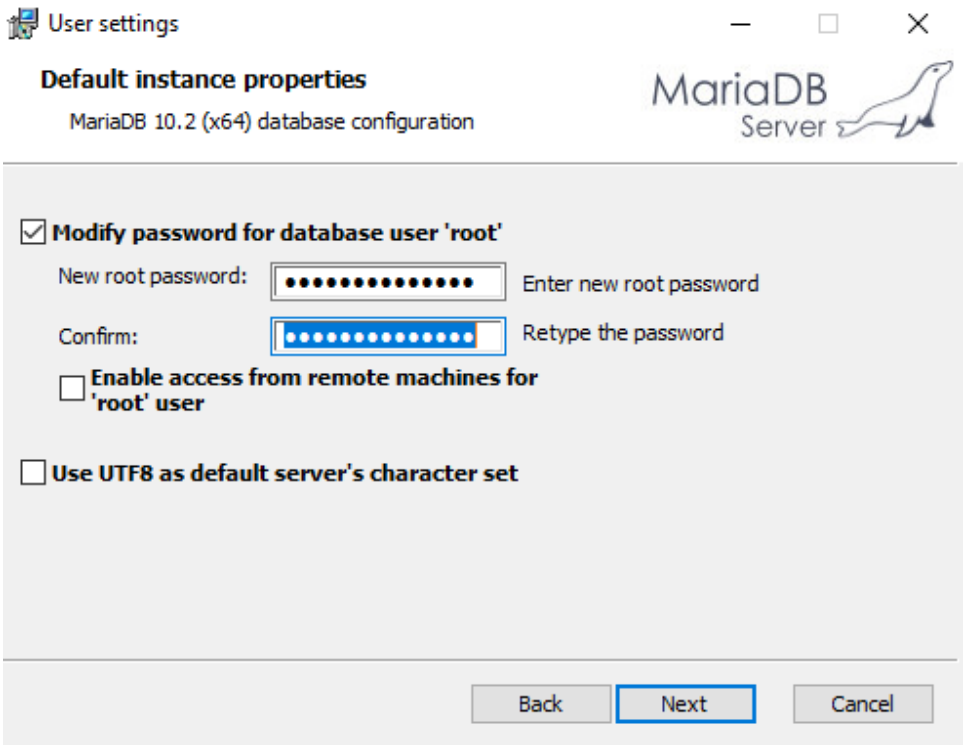
Development Database

The web application relies heavily on a database, so a local development database will need to be created on your machine.

1. Download and install MariaDB 10.3.25. Keep the configuration settings at their default values. This will also install HeidiSQL.

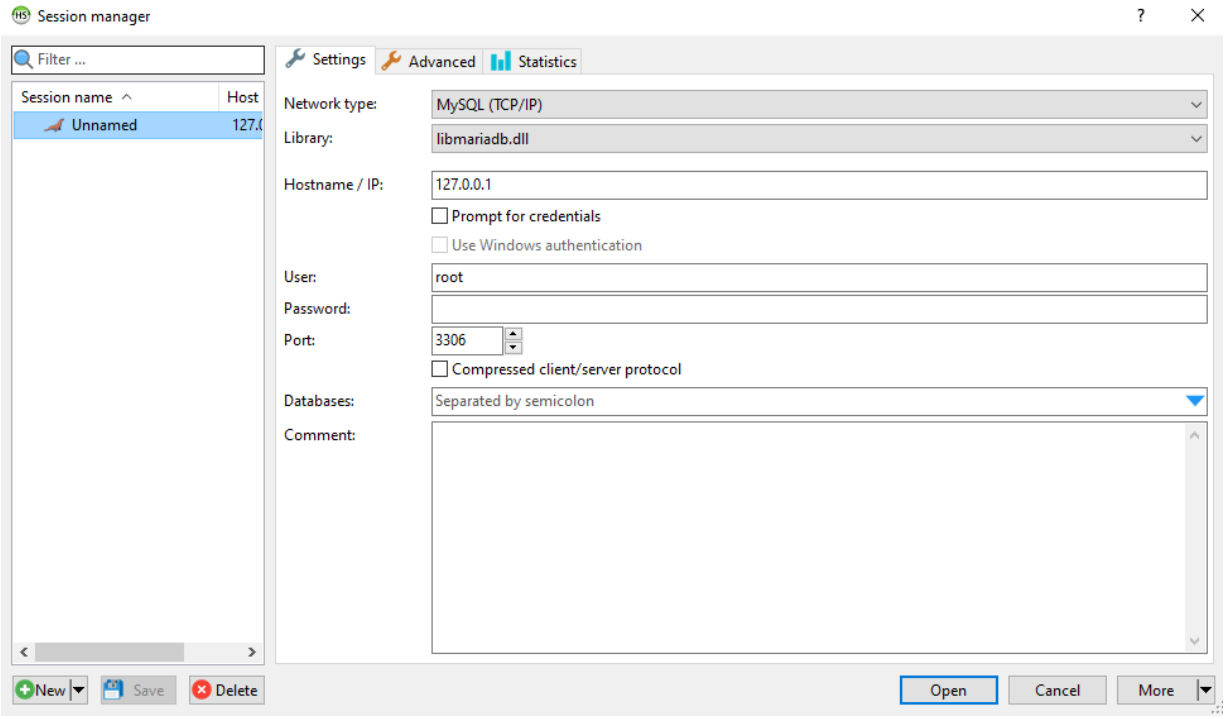


2. Modify the root password to: “teamwethemboys”

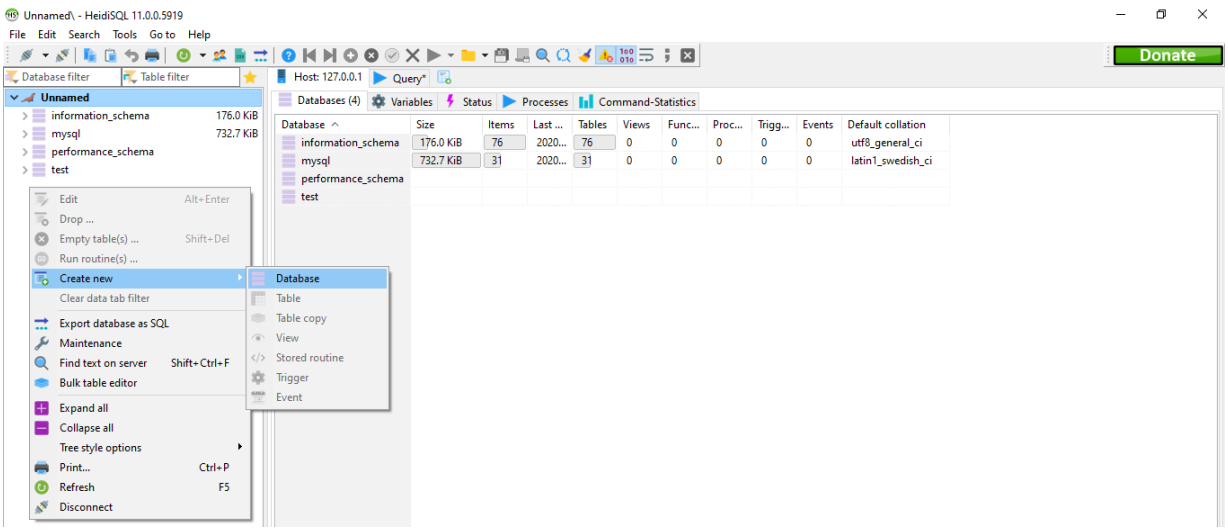


3. Continue through each remaining setup screen, keeping the default values. MariaDB and HeidiSQL should now be installed on your machine.

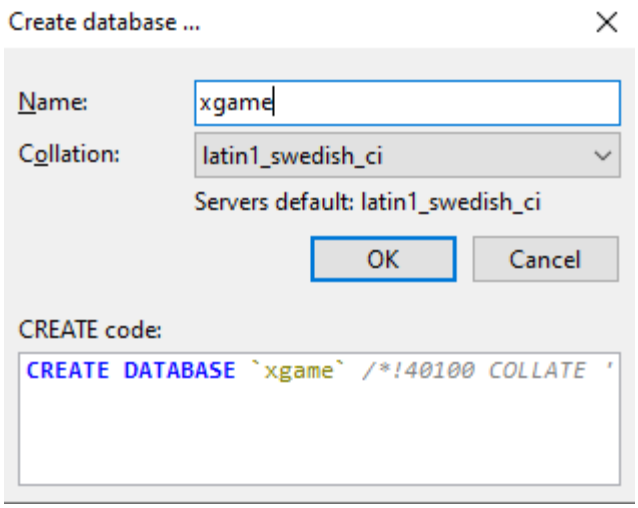
4. Open HeidiSQL. If the default connection to localhost:3306 is not there, create a new connection to the database by selecting the ‘New’ button in the bottom left corner of the screen. Enter “teamwethemboys” as the password, then click the ‘Open’ button.



5. On the left pane, right-click and select Create new > Database.



6. Enter “xgame” as the name of the new database, then click “ok”. We have just created a development database. Spring Boot will initialize the schema once we run the Java project.



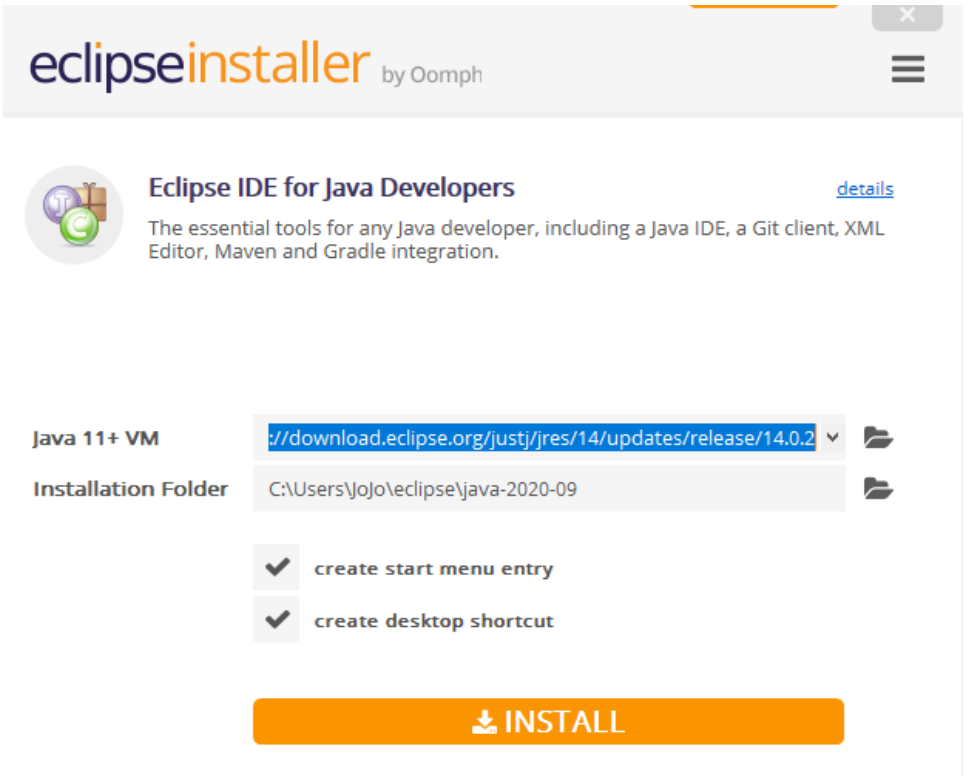
Integrated Development Environment

1. Download and install Eclipse using the following link:

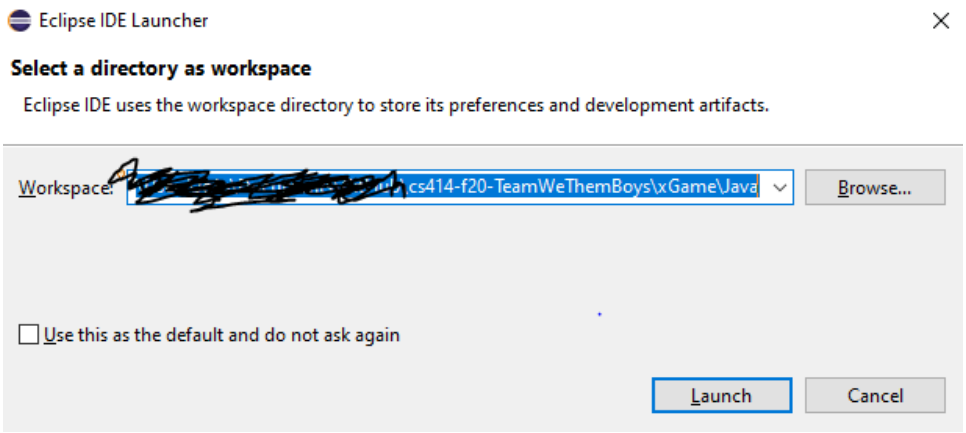
   https://www.eclipse.org/downloads/download.php?file=/oomph/epp/2020-09/R/eclipse-inst-jre-win64.exe&mirror_id=518

   Run the Eclipse Installer. Select “Eclipse IDE for Java Developers”.

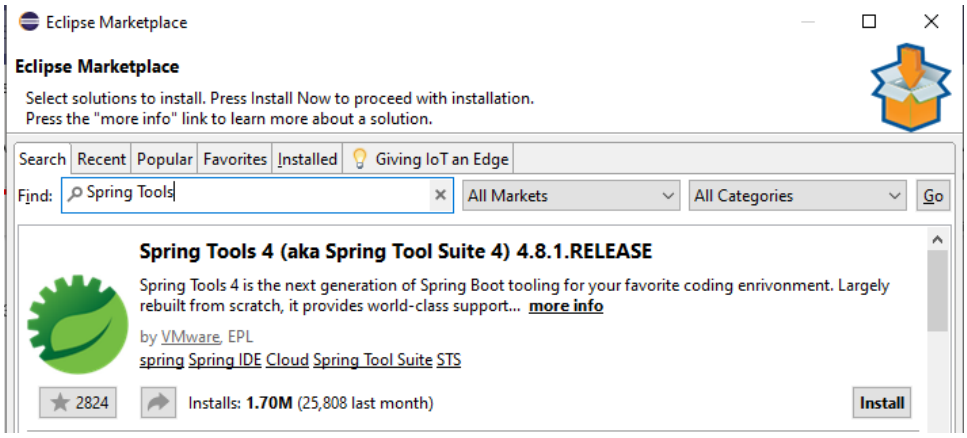
2. Keep the default install locations for the JVM and installation folder unless you have a good reason to put them somewhere else. Click ‘Install’ and accept the following agreement.



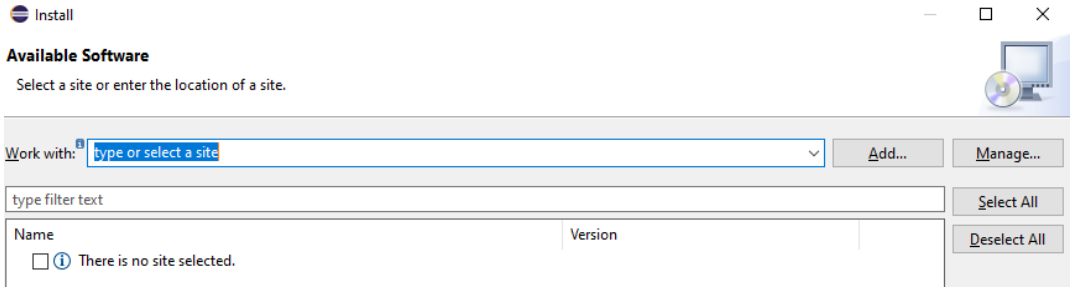
3. If you haven't already, use your favorite Git tool to pull down the repository code. Launch Eclipse and set the current workspace to the location of the code. Specifically target the "Java" folder of the repository. If eclipse does not automatically recognize the project, you may have to refresh the workspace, or select "File > Open Projects from File System..." and select the working directory.



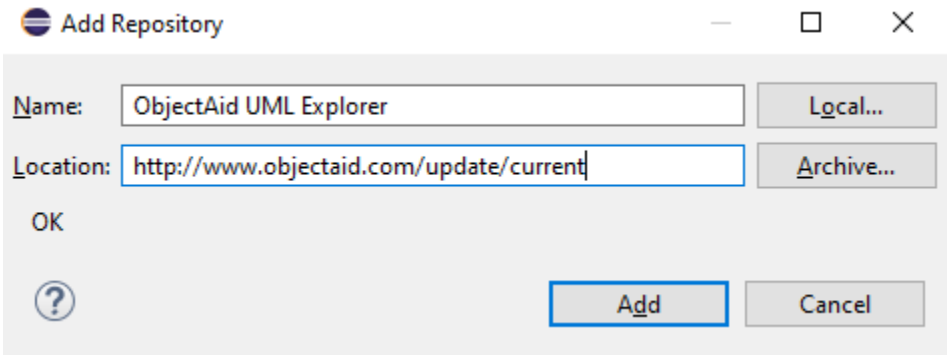
4. Now we need to install our extensions. The first will be Spring Tools 4. Select "Help > Marketplace", then search "Spring Tools 4". Click "Install", install all features, and accept the terms of agreement. You will also have to restart Eclipse.



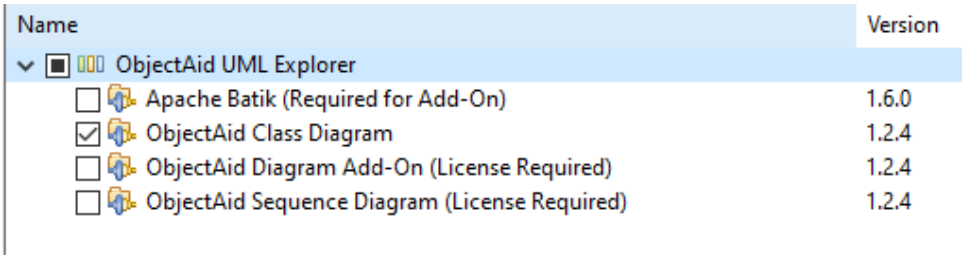
5. To install ObjectAid UML Explorer select “Help > Install New Software...”. Select “Add...” in the wizard.



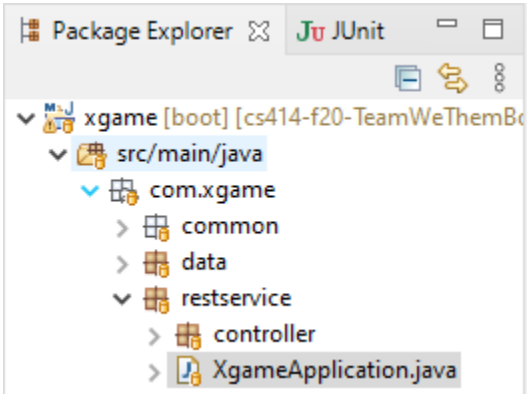
6. Enter the following as shown below, then click the “Add” button.



7. Select “ObjectAid Class Diagram”. We don’t need any of the other packages. Then click the “Next” button and agree to the terms. You may see a warning about untrusted add-ons. If you do, click “OK”. You will have to restart Eclipse once installation is complete.



8. In the Package Explorer, open “src/main/java/com.xgame.restservice.XgameApplication.java. This contains the main method for the application. Run it.



9. In the Console, you should see something like below. The final message “Started XgameApplication...” means that the web application is now running locally on port 8080.

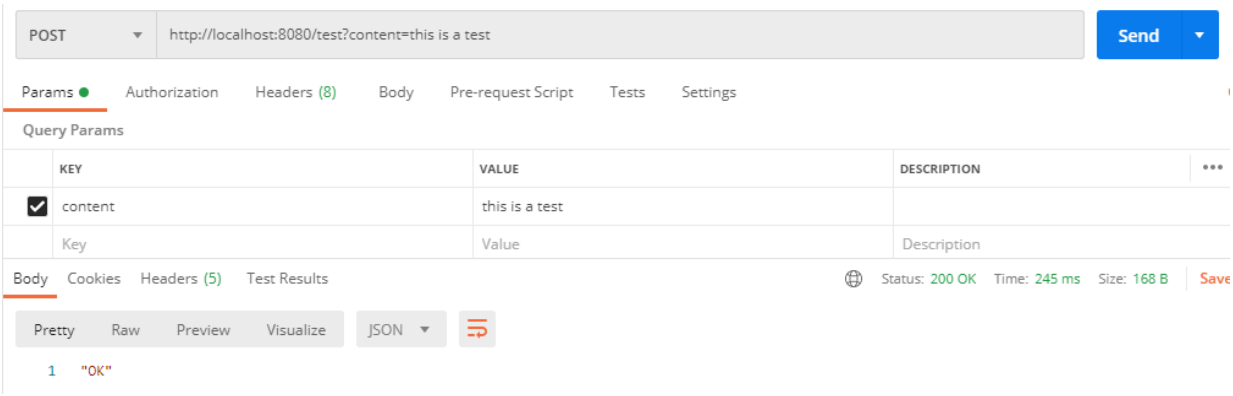
Checking Database Schema Creation

Running the Java project should have created the schema in the xgame database. To confirm:

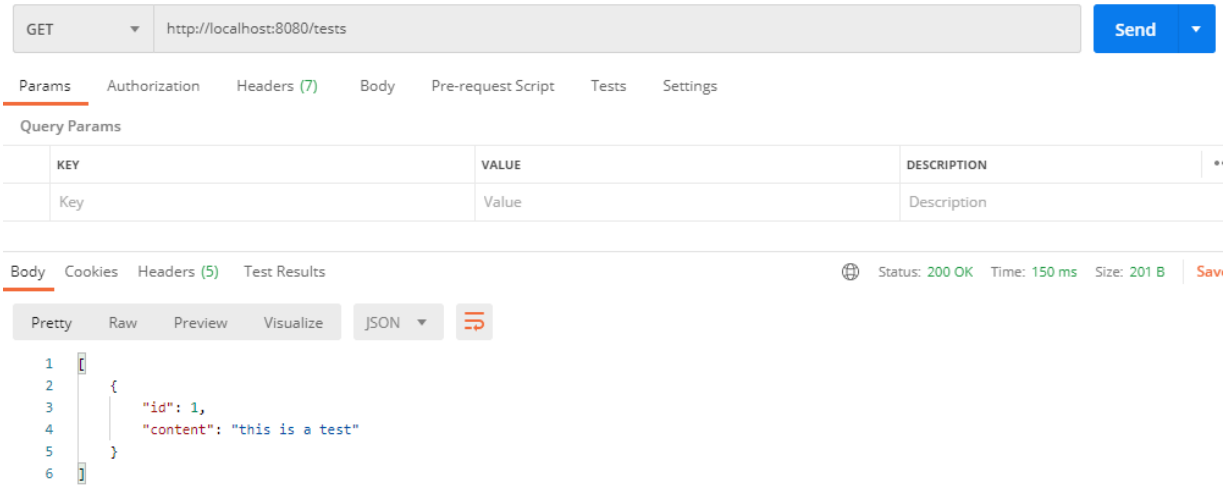
1. Open up HeidiSQL and reconnect to 127.0.0.1.
2. Select the xgame database
3. You should now see a list of tables in the database. Feel free to explore the database.

Testing the Environment

1. To test the application properly, ensure that Postman is installed.
2. Fire up Postman and create a POST request with the url “localhost:8080/test” and set some value for the parameter “content”. If you get a 200 OK response, then the API call was successful and a new test message has been saved to the database.



3. Now try a GET request on the same URL with no parameters to retrieve all of the test messages from the database. You should see the test message you just created in the list of messages.



4. Since we got a message back, we know that the application can successfully receive HTTP requests and communicate with our local database. To confirm there is a message in the database, check back in with HeidiSQL.