Project Setup

Wednesday, 19 March 2025   14:02

Backend and Frontend to be in separate projects

Backend (REST implementation + db)
New Dynamic Web Project -> name: ProjectBackend
Run Configuration -> Apache Tomcat

On moodle in section REST JAX-RS Jersey
Download:
    Jars
    Servlet definition

On moodle in section 7 - REST CRUD & HSQLDB
Download:
    HSQLDB Files

Unpack jar and hsqldb files

Go to
WEB-INF -> lib -> copy all the jar files and hsqldb two jars from HSQLDB Files

Copy servlet definition and paste it to web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd" id="WebApp_ID"
version="4.0">
<display-name>A00279259TravelPlanner</display-name>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>default.html</welcome-file>
        <welcome-file>default.jsp</welcome-file>
        <welcome-file>default.htm</welcome-file>
    </welcome-file-list>

    <servlet>
        <servlet-name>Jersey</servlet-name>
        <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
        <init-param>
        <param-name>com.sun.jersey.config.property.packages</param-name>
        <!-- The below param-value needs to be changed to the package name -->
        <param-value>myApp</param-value>
        </init-param>
        <init-param>
        <param-name>com.sun.jersey.api.json.POJOMappingFeature</param-name>
        <param-value>true</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
        </servlet>
        <servlet-mapping>
        <servlet-name>Jersey</servlet-name>
        <url-pattern>/rest/*</url-pattern>
    </servlet-mapping>
</web-app>
```

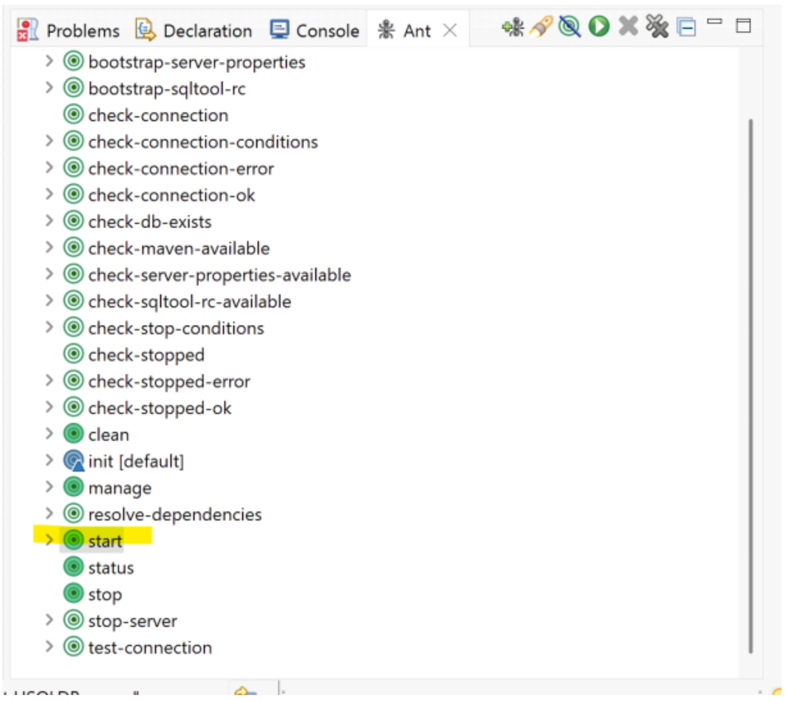
After that's done, copy the param-value (as this has to be the package name) and create package with that name. It can be anything, eg. your studentID

Click right click on project
Create new folder "db"
Drag and drop managedDB_v3 into "db" folder (from HSQLDB Files)

Add ANT
Window -> Show View -> Other -> Ant

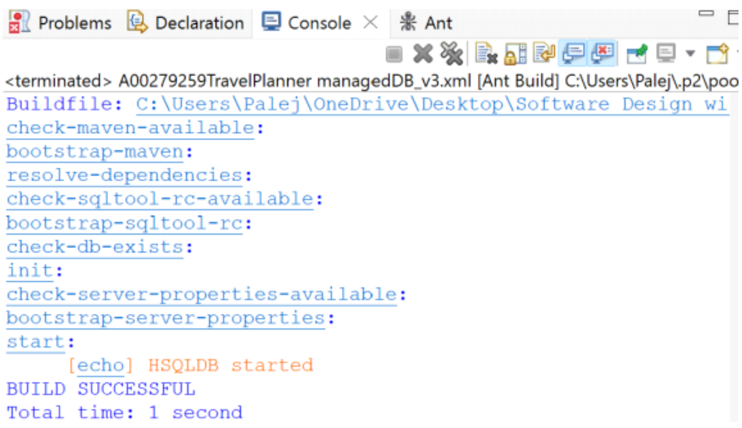
Drag and drop managedDB_v3 into the Ant view

Open up Ant and click "start"



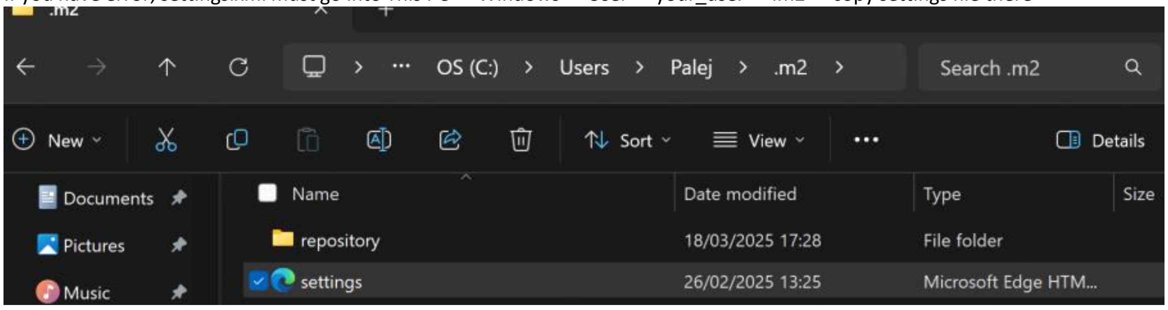
```
bootstrap-server-properties
bootstrap-sqltool-rc
check-connection
check-connection-conditions
check-connection-error
check-connection-ok
check-db-exists
check-maven-available
check-server-properties-available
check-sqltool-rc-available
check-stop-conditions
check-stopped
check-stopped-error
check-stopped-ok
clean
init [default]
manage
resolve-dependencies
start
status
stop
stop-server
test-connection
```

If all Ok, you will get confirmation that HSQLDB started, just like below



```
<terminated> A00279259TravelPlanner managedDB_v3.xml [Ant Build] C:\Users\Palej\.p2\poo
Buildfile: C:\Users\Palej\OneDrive\Desktop\Software Design wi
check-maven-available:
bootstrap-maven:
resolve-dependencies:
check-sqltool-rc-available:
bootstrap-sqltool-rc:
check-db-exists:
init:
check-server-properties-available:
bootstrap-server-properties:
start:
     [echo] HSQLDB started
BUILD SUCCESSFUL
Total time: 1 second
```

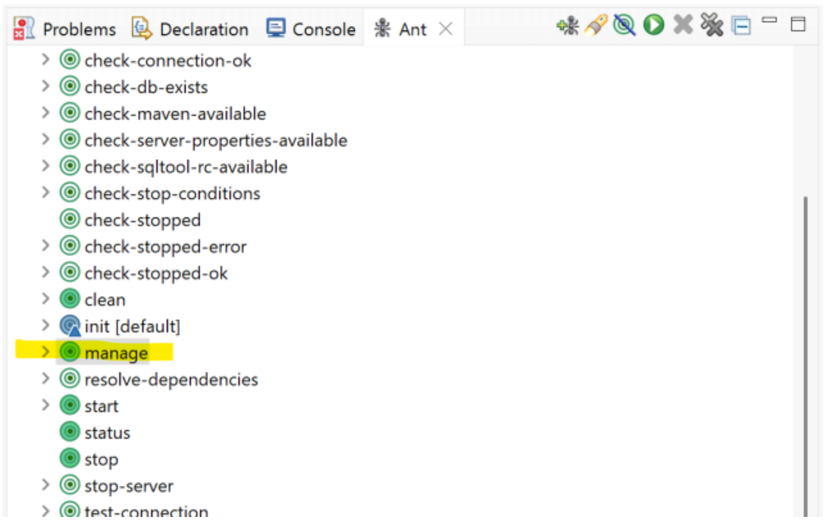
If you have error, settings.xml must go into This PC -> Windows -> User -> your_user -> .m2 -> copy settings file there



Name	Date modified	Type	Size
repository	18/03/2025 17:28	File folder	
settings	26/02/2025 13:25	Microsoft Edge HTM...	

Re-run the ant (click start again on db) and it should be ok

Click "manage" on Ant console



```
check-connection-ok
check-db-exists
check-maven-available
check-server-properties-available
check-sqltool-rc-available
check-stop-conditions
check-stopped
check-stopped-error
check-stopped-ok
clean
init [default]
manage
resolve-dependencies
start
status
stop
stop-server
test-connection
```

This is where you do db queries, create table etc

The queries are slightly different than normal sql, so make sure that you lookup queries by adding "hsql" in your google sear ch

Examples to create table

```
CREATE TABLE IF NOT EXISTS trips (
        tripId INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
        destination VARCHAR(100),
        startDate DATE,
        endDate DATE,
        budget DECIMAL(10,2),
        notes VARCHAR(255)
);

CREATE TABLE IF NOT EXISTS activities (
    activityId INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    tripId INTEGER NOT NULL,
    name VARCHAR(100) NOT NULL,
    time DATE NOT NULL,
    location VARCHAR(100) NOT NULL,
    cost DECIMAL(10,2) DEFAULT 0.00,
    FOREIGN KEY (tripId) REFERENCES trips(tripId) ON DELETE CASCADE
);

DROP TABLE IF EXISTS activities;
DROP TABLE IF EXISTS trips;
```

BACKEND SETUP DONE.

Frontend SETUP (have to have GUI)

New Project -> Standard Java Application -> Eg. Student_id_Frontend
Right Click -> Configure -> Convert To Maven

Add dependencies for libraries

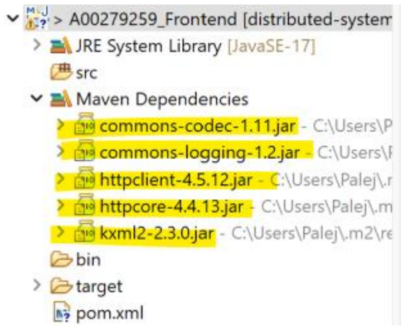
Go back to moodle -> REST Clients -> dependencies

Copy and paste the dependencies after </build> end tag in pom.xml

```
<dependencies>
        <!-- https://mvnrepository.com/artifact/org.apache.httpcomponents/httpcore -->
        <dependency>
                <groupId>org.apache.httpcomponents</groupId>
                <artifactId>httpcore</artifactId>
                <version>4.4.13</version>
        </dependency>
<!-- https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient -->
        <dependency>
                <groupId>org.apache.httpcomponents</groupId>
                <artifactId>httpclient</artifactId>
                <version>4.5.12</version>
        </dependency>
```

```
<!-- https://mvnrepository.com/artifact/commons-logging/commons-logging -->
        <dependency>
                <groupId>commons-logging</groupId>
                <artifactId>commons-logging</artifactId>
                <version>1.2</version>
        </dependency>
<!-- https://mvnrepository.com/artifact/net.sf.kxml/kxml2 -->
        <dependency>
                <groupId>net.sf.kxml</groupId>
                <artifactId>kxml2</artifactId>
                <version>2.3.0</version>
        </dependency>
</dependencies>
```

To check if it works, go into Maven Dependencies - if you have 5 dependencies there, it means it works fine



```
> A00279259_Frontend [distributed-system
  JRE System Library [JavaSE-17]
  src
  Maven Dependencies
    commons-codec-1.11.jar - C:\Users\P
    commons-logging-1.2.jar - C:\Users\
    httpclient-4.5.12.jar - C:\Users\Palej\.r
    httpcore-4.4.13.jar - C:\Users\Palej\.m
    kxml2-2.3.0.jar - C:\Users\Palej\.m2\re
  bin
  target
  pom.xml
```

FRONTEND setup done.
Now you need to define the interactions and classes