Javaの開発を始める

```
$ docker compose exec tomcat /bin/bash
```

Javaの開発

1. hello.javaを作成する
2. hello.javaをコンパイル
3. 実行

hello.javaを作成する

- hello.java.txtをsrc/hello.javaに設置

```
public class hello {

    public static void main(String[] args) {
        System.out.println("Hello, world.");
    }

}
```

コンパイル

```
$ javac hello.java
$ ll hello.class
```

=> hello.javaからhello.classが作成されます

実行

```
$ java hello
Hello, world.
```

Servletの開発

1. hello_servlet.javaを作成する
2. hello_servlet.javaをコンパイル
3. Tomcatに配置する

hello_servlet.javaの作成

- hello_servlet.java.txtをsrc/hello_servlet.javaに設置

```
import java.io.*;
import jakarta.servlet.*; // tomcat10以降、javax.servletではなく
jakarta.servletを使用する
import jakarta.servlet.http.*; // 同上
import jakarta.servlet.annotation.WebServlet; // web.xmlを使わずannotationを
使用する場合に指定

@WebServlet("/")
public class hello_servlet extends HttpServlet {
        protected void doGet(HttpServletRequest
request,HttpServletResponse response) throws ServletException,IOException
{
                PrintWriter resout = response.getWriter();
                resout.println("Servlet test program.");
        }
}
```

コンパイル（失敗）

```
$ javac hello_servlet.java
```

=> エラーが発生

コンパイル(classpath付き)

```
$ javac -classpath /usr/local/tomcat/lib/servlet-api.jar
hello_servlet.java
$ ll hello_servlet.class
```

Tomcatに配置する

webapps/hello/WEB-INF/classes ディレクトリの作成

```
$ mkdir -p /app/webapps/hello/WEB-INF/classes
$ mv hello_servlet.class /app/webapps/hello/WEB-INF/classes
```

=> http://localhost:8888/hello/ の表示を確認

動かない時「webappsディレクトリにあるhelloディレクトリをhogeなどに変更

例）hogeにした場合は

=> http://localhost:8888/hoge/ の表示を確認

ライブラリのダウンロード

※ 事前実行済

pom.xmlの作成

mysql https://mvnrepository.com/artifact/mysql/mysql-connector-java/8.0.31

Jackson https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind/2.13.4

Servlet-API https://mvnrepository.com/artifact/jakarta.servlet/jakarta.servlet-api/6.0.0

maven経由でインストール

```
$ mvn dependency:copy-dependencies
=> target/dependencyにファイルが入る

$ ls ./target/dependency
```

MySQLに接続出来るかを確認する

確認のためのdbtest.javaを作成する

- dbtest.java.txtをsrc/dbtest.javaに設置

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class dbtest {
    public static void main(String[] args) {
        ...
        // データベース接続
        conn = DriverManager.getConnection(
                // ホスト名、データベース名
                "jdbc:mysql://mysql:3306/study",
                // ユーザー名
                "system",
                // パスワード
                "system");
    }
}
```

コンパイル&実行

```

```

```
# コンパイル
$ javac -d bin dbtest.java

# 実行
$ java -classpath "./bin:./target/dependency/mysql-connector-j-8.0.31.jar"
dbtest
```

ダミーデーターを取り込む

確認のためのdbtest.javaを作成する

- insert_db.java.txtをsrc/insert_db.javaに設置
- insert_dbはJSONの読み込みが必要

必要なjarが増えて大変

1. クラスパスのワイルドカードを利用する
2. 環境変数CLASSPATHを使う

コンパイル&実行

```
# 環境変数の設定
export CLASSPATH=.:./bin:./target/dependency/*

# コンパイル
javac -d bin insert_db.java

# 実行
java insert_db
```

IDEを使わないとエラー表示になる

VSCodeを使ってJavaのビルドが出来るようにする

- Extension Pack for Java
- 設定「java.project.referencedLibraries」

```
{
    "java.project.referencedLibraries": ["src/target/dependency/*.jar"]
}
```

シンプルなJSONを返却するServlet

- api_list01.txtをsrc/api_list.javaに配置

```
export CLASSPATH=.:./bin/./target/dependency/*:/usr/local/tomcat/lib/*
$ javac -d bin api/api_list.java
=> src/bin/api/api_list.classを確認

$ mkdir -p bin/api_test/WEB-INF/classes
$ cp -r bin/api bin/api_test/WEB-INF/classes
$ mv bin/api_test /app/webapps
```

http://127.0.0.1:8888/api_test/ を開いて確認する

Jacksonを使ってJSONをアウトプット

- APIResponseBean01.txt => src/api/APIResponseBean.java
- api_list02.txt => src/api/api_list.java

```
$ rm -rf /app/webapps/api_test
=> 少し待機

$ export CLASSPATH=.:./bin/./target/dependency/*:/usr/local/tomcat/lib/*
$ javac -d bin api/*
$ mkdir -p bin/api_test/WEB-INF/classes
$ mkdir -p bin/api_test/WEB-INF/lib
$ cp -r bin/api bin/api_test/WEB-INF/classes
$ mv bin/api_test /app/webapps
```

http://127.0.0.1:8888/api_test/ => エラーになる

理由はTomcatにJarを渡せていないから

```
$ rm -rf /app/webapps/api_test
少し待機

$ rm -rf bin/api_test
$ mkdir -p bin/api_test/WEB-INF/classes
$ mkdir -p bin/api_test/WEB-INF/lib
$ cp -rf bin/api bin/api_test/WEB-INF/classes
$ cp -rf target/dependency/*.jar bin/api_test/WEB-INF/lib
$ mv bin/api_test /app/webapps
```

http://127.0.0.1:8888/api_test/ を開いて確認する

作業が難しくなってきたので、シェルスクリプトにまとめる

- deploy_test_api.txt => src/deploy_test_api.sh

```

```

```
$ chmod +x deploy_test_api.sh
$ ./deploy_test_api.sh
```

顧客情報を渡すようにする

- APIResponseBean02.txt
- api_list03.txt
- CustomerInfoBean01.txt

```
$ ./deploy_test_api.sh
```

DBから取得するようにする

- api_list04.txt

http://127.0.0.1:8888/api_test/ を開いて確認する

```
$ ./deploy_test_api.sh
```

NextJSからアクセスする

```
//通信
const loadCustomers = () => {

    fetch('http://127.0.0.1:8888/api_test/')
        .then((response) => response.json())
        .then((data) => {
            console.log(data)
        });

};
```

=> エラーが発生する

CORS対策を追加したServlet

```
//CORS対策
response.setHeader("Access-Control-Allow-Origin","*");
response.setHeader("Access-Control-Allow-Headers", "Content-Type");
```

=> 動作確認