

# Wired2Perform Coding Assignment

Kavindhya Kurupitage

## ToDo Application

### 1. Used Tools and platforms



ECLIPSE IDE



Spring Boot



JAVA Language



SQL



POSTMAN

### 2. Created a project using **Spring Initializr**. (selected Maven project, JAVA language and Spring Boot version 3.3.3)

#### Project

☐ Gradle - Groovy

☐ Gradle - Kotlin

☒ Maven

#### Language

☒ Java ☐ Kotlin

☐ Groovy

#### Spring Boot

☐ 3.4.0 (SNAPSHOT) ☐ 3.4.0 (M2)

☐ 3.3.4 (SNAPSHOT) ☒ 3.3.3 ☐ 3.2.10 (SNAPSHOT)

☐ 3.2.9

#### Project Metadata

Group com.example

Artifact ToDoApplication

### 3. Used several dependencies

#### Spring Web WEB

Build web, including RESTful, applications using Spring MVC.  
Uses Apache Tomcat as the default embedded container.

#### H2 Database SQL

Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser-based console application.

#### Spring Data JPA SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

#### Spring Security SECURITY

Highly customizable authentication and access-control framework for Spring applications.

#### Spring Boot DevTools DEVELOPER TOOLS

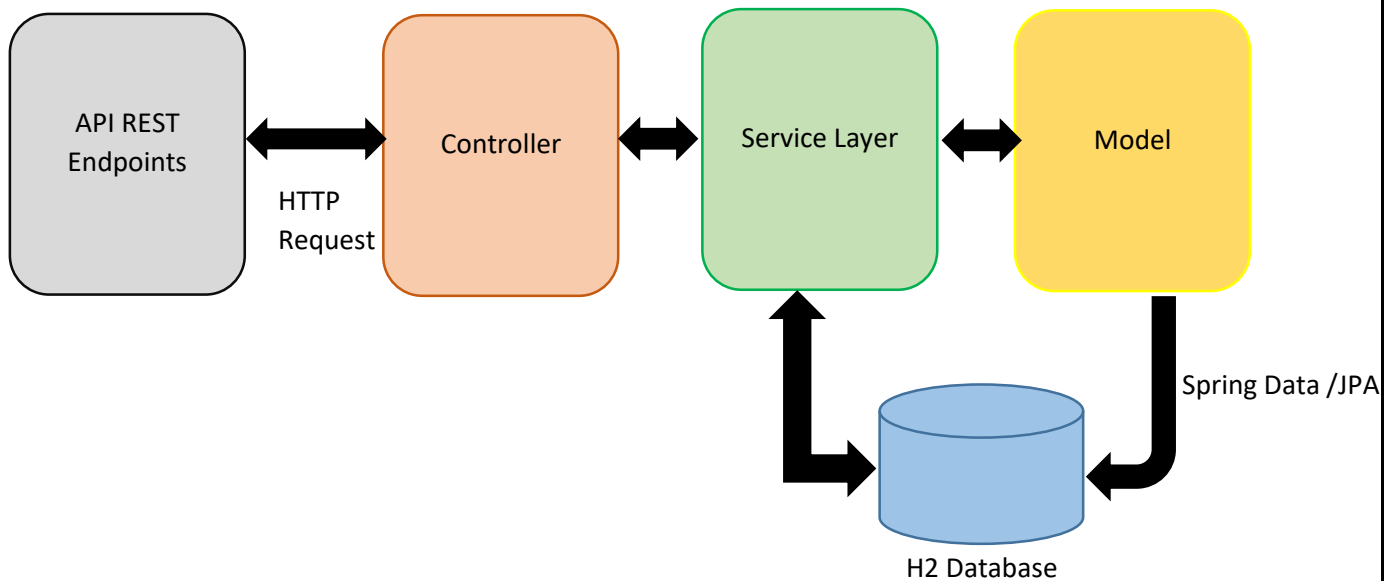
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

- Spring Web is used to build the REST API.
- H2 Database is used for the in-memory database.
- Spring data JPA is used to handle the database side.
- Spring Boot DevTools is used to ease development techniques.
- Used Spring Security for authentications.

### 4. This is my ToDo Application folder structure

```
▼ ToDoApplication
  ▼ src/main/java
    ▼ com.example.demo
      > UserToDoAppApplication.java
    ▼ com.example.demo.config
      > SecurityConfig.java
    ▼ com.example.demo.controller
      > TodoListController.java
      > UserController.java
    ▼ com.example.demo.model
      > TodoList.java
      > User.java
    ▼ com.example.demo.repository
      > TodoListRepository.java
      > UserRepository.java
    ▼ com.example.demo.service
      > TodoListService.java
      > UserService.java
  ▼ src/main/resources
    static
    templates
    application.properties
  > src/test/java
  > JRE System Library [JavaSE-17]
  > Maven Dependencies
```

## 5. My ToDo Application architecture



## 6. After running the UserToDoAppApplication.java file open the H2 database using local host <http://localhost:8080/h2-console>.

English

### Login

Saved Settings:

Setting Name:

---

Driver Class:

JDBC URL:

User Name:

Password:

Test successful

7. Created a data table including users.Used SQL queries for it.

```
Run Run Selected Auto complete Clear SQL statement:
id INT NOT NULL,
name VARCHAR(50) NOT NULL,
email VARCHAR(50) NOT NULL,
telephone BIGINT NOT NULL,
age INT NOT NULL,
password VARCHAR(10) NOT NULL,
PRIMARY KEY(id)
);

CREATE TABLE users (
  id INT NOT NULL,
  name VARCHAR(50) NOT NULL,
  email VARCHAR(50) NOT NULL,
  telephone BIGINT NOT NULL,
  age INT NOT NULL,
  password VARCHAR(10) NOT NULL,
  PRIMARY KEY(id)
);
Update count: 0
(33 ms)
```

8. Tested using Postman

GET localhost:8080/getAllusers

Params Authorization Headers (6) Body Scripts Settings

Query Params

Key	Value	Description
Key	Value	Description

body Cookies Headers (5) Test Results 200 OK

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "name": "Kavindhya",
5     "email": "kavindhya@gmail.com",
6     "telephone": 754493313,
7     "age": 25,
8     "password": "kavindhyaa"
9   },
```

## 9. Tested other CRUD operation using Postman.

```
@RestController
@RequestMapping("/api/todolists")
public class TodoListController {

    @Autowired
    private TodoListService todoListService;

    @PostMapping("/create")
    public ResponseEntity<TodoList> createTodoList(@RequestBody TodoList todoList) {
        TodoList newTodoList = todoListService.createTodoList(todoList);
        return ResponseEntity.ok(newTodoList);
    }

    @GetMapping("/user/{userId}")
    public ResponseEntity<List<TodoList>> getTodoLists(@PathVariable Long userId) {
        List<TodoList> todoLists = todoListService.getTodoListsByUser(userId);
        return ResponseEntity.ok(todoLists);
    }

    @PutMapping("/update")
    public ResponseEntity<TodoList> updateTodoList(@RequestBody TodoList todoList) {
        TodoList updatedTodoList = todoListService.updateTodoList(todoList);
        return ResponseEntity.ok(updatedTodoList);
    }

    @DeleteMapping("/delete/{id}")
    public ResponseEntity<Void> deleteTodoList(@PathVariable Long id) {
        todoListService.deleteTodoList(id);
        return ResponseEntity.ok().build();
    }
}
```