Phase 1 Source – Group 3

University of Maryland Global Campus

Sep 10, 2024

CMSC 495 Capstone

Team Members:

Cook, Sean

Coutros, Peter

Daley, Tasciana

Geisler, Nicholas

Holman, Trevor

Jefferson, Michael

Malixi, Matthew Arvhie

Mussa, Nezifa

Suarez, Karolyn

Introduction

The Calendar/Task Tracker project for CMSC 495 involves developing a Java-based application that enables users to manage tasks and events on a calendar. The system allows the creation, modification, and deletion of tasks and events, as well as tracking their completion. The team is using Spring Boot with Thymeleaf for the front-end and H2 as the database.

Objective

The goal of Phase I is to design the system architecture, set up the environment, and implement basic features like task and event management. Key focuses include:

- Setting up the Spring Boot environment
- Using agile methodologies for project development
- Implementing a task and calendar system
- Initial unit testing with JUnit
- Configuring Thymeleaf for front-end and H2 for data persistence

---

Instructions

1. **Project Setup**
    - **Tools and Dependencies**:
        - Java
        - Spring Web
        - ThymeLeaf
        - Spring Data JPA
        - H2 Database
        - Spring Devtools
        - Lombok
    - **Environment Setup**: The application can be set up using the provided **.zip** file or cloned from the group's GitHub repository (GitHub Repo). Follow the setup guide to configure the environment, import the project as a Maven project, and install dependencies.
    - **IDE**: Eclipse or Spring Tools Suite is recommended.

The project's development environment resides within Eclipse IDE. We are utilizing Eclipse version 4.29.0 (or greater) that has JavaSE-17 (or greater). Additionally, Spring Tools 4 version 4.24.0 is required and can be downloaded/installed from the Eclipse Marketplace (figure 1).

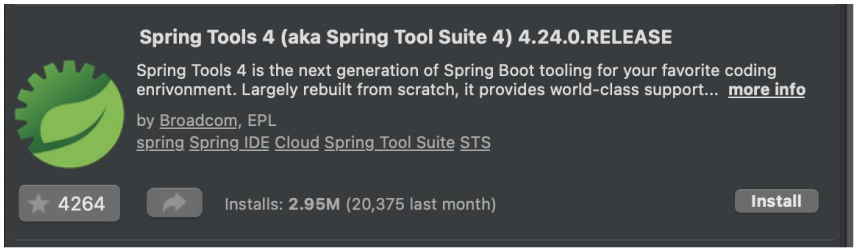

Fig.1

To set up the project within Eclipse, first open Eclipse, select your desired workspace directory, then click "Launch" (figure 2).

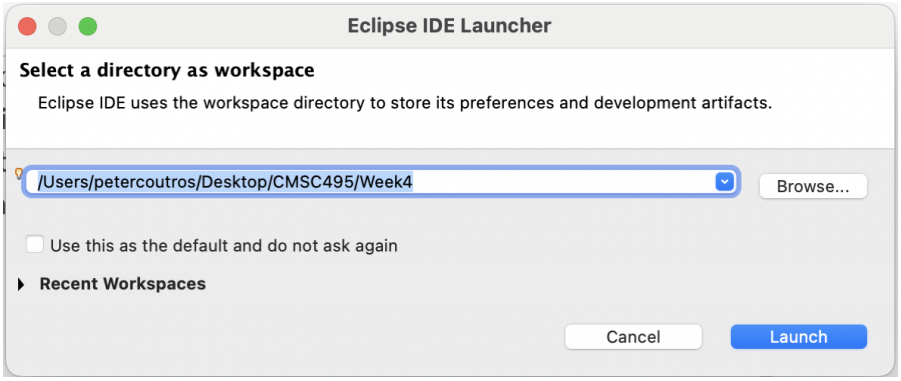

Fig. 2

Import as new Project from a Directory

Navigate to “File” > “Import...” > “General” > “Existing Projects into Workspace” and click “Next” (figure 3).

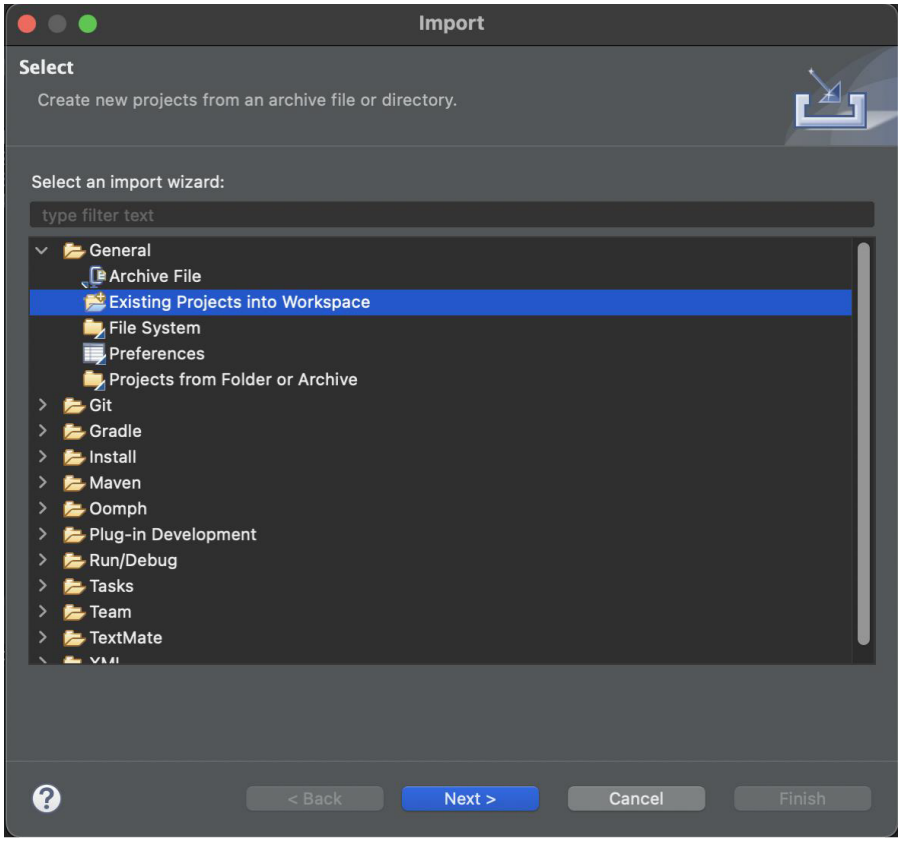

Fig 3.

Press the “Select root directory” radio button and select the folder containing the entire project (figure 4).

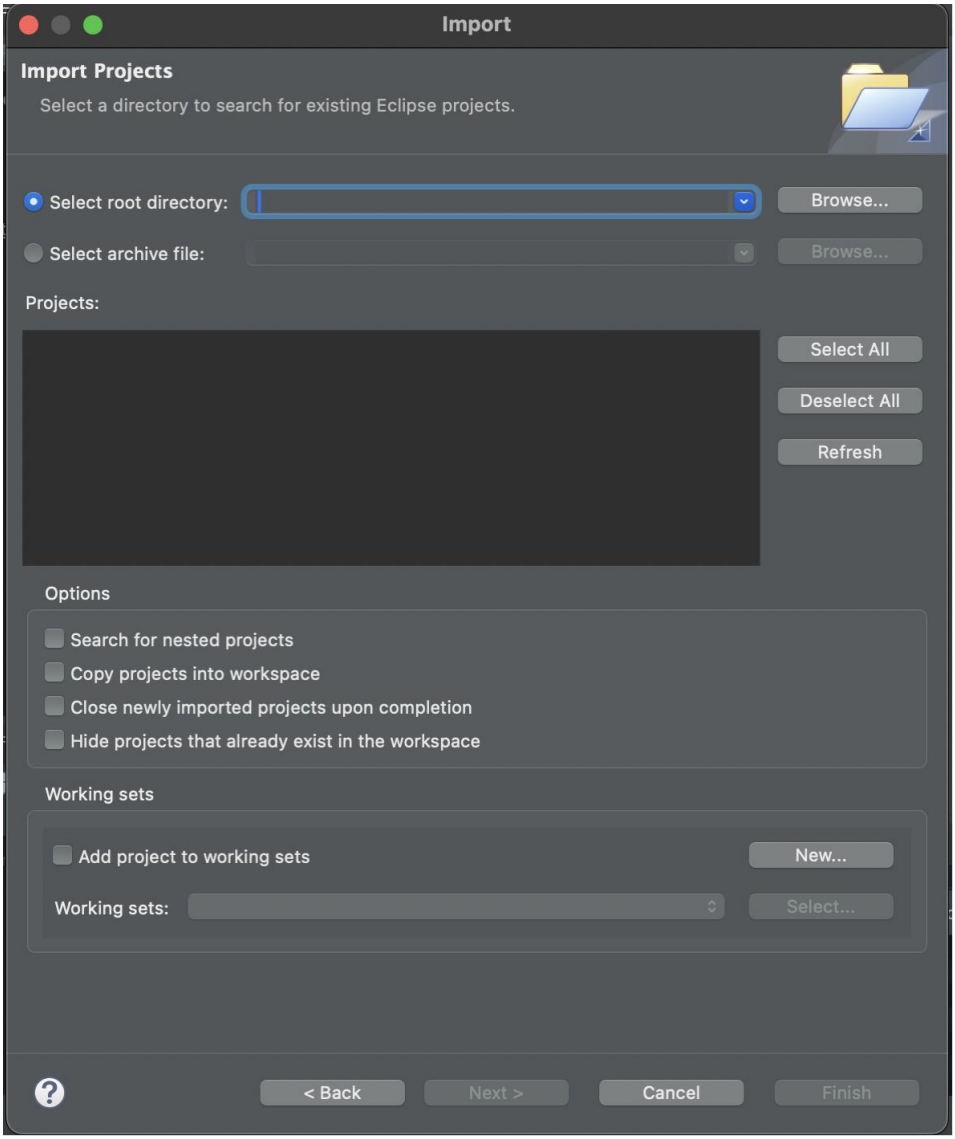

Fig. 4

Ensure that the folder containing the project is checked in the “Projects:” section of the import and click “Finish” (figure 5).

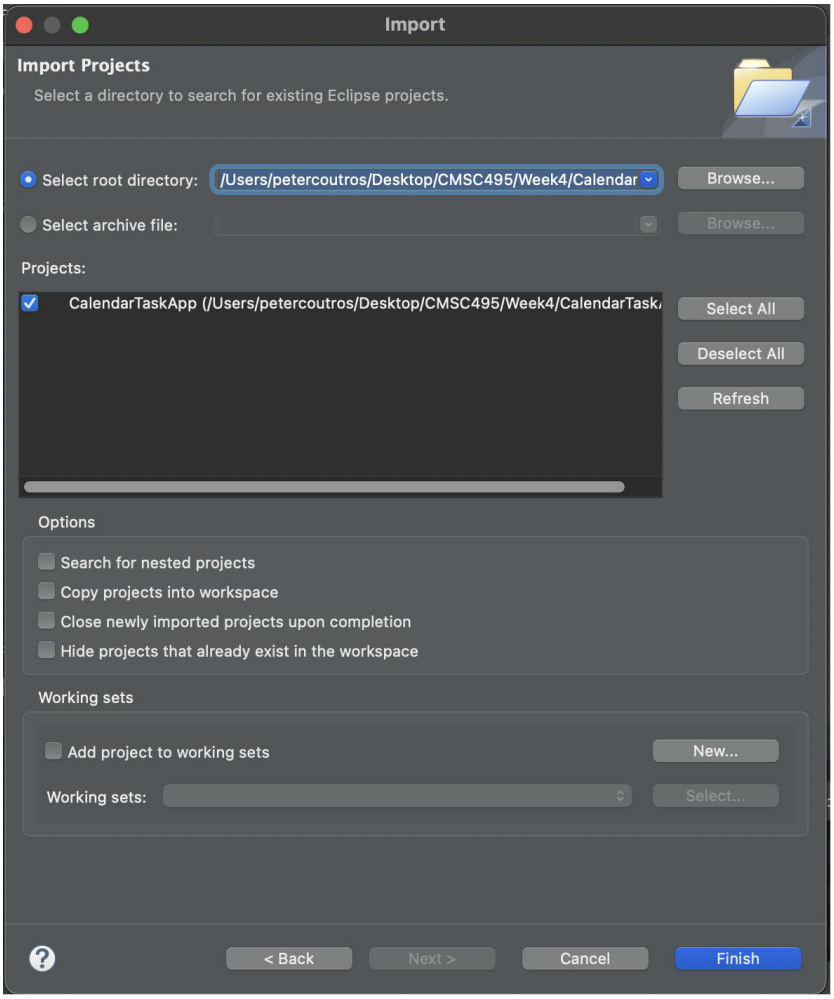

Fig. 5

With the project now open, navigate to the file containing the main method (currently housed in CalendarTaskAppApplication.java) and run as “Spring Boot App” (figure 6).

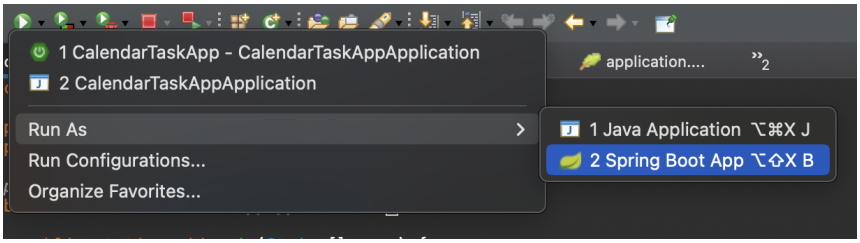

Fig. 6

Navigate to your browser of choice and enter http://localhost:8080/ into the navigation bar to interact with the application.

Import as Existing Maven Project

If the above did not recognize the project try importing as an existing maven project (this will be necessary if cloning from GitHub).

Navigate to “File” > “Import...” > “Maven” > “Existing Maven Project” and click “Next” (figure 7).

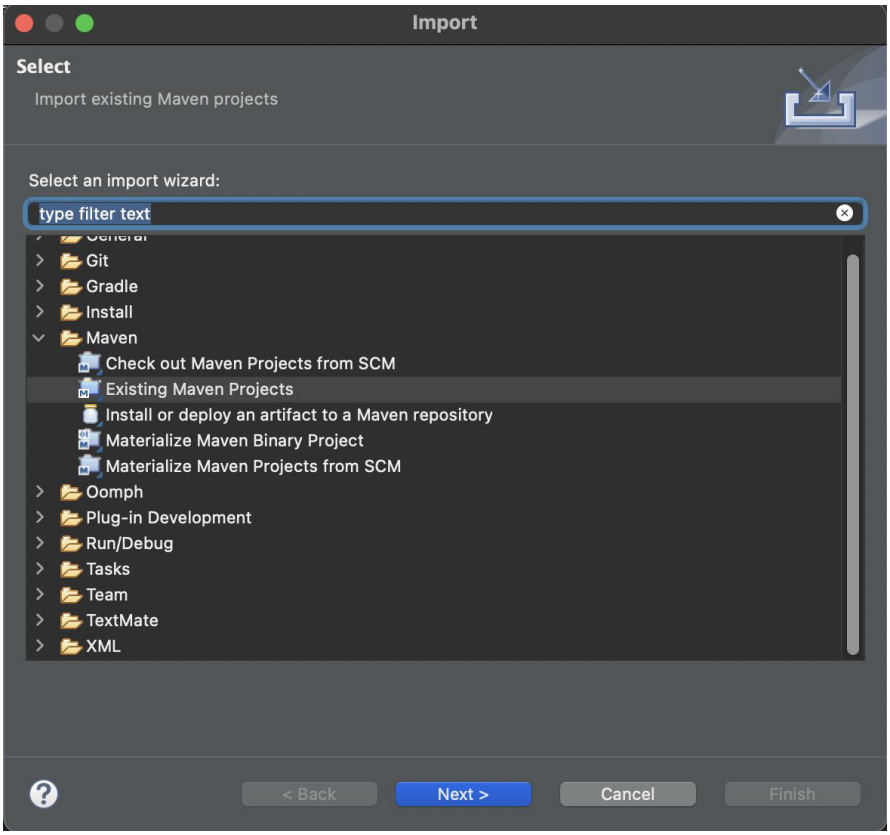

Fig. 7

Browse and select the root directory housing the project/cloned repo and ensure that the project populates in the “Projects:” section and that its box is checked and click “Finish” (figure 8).

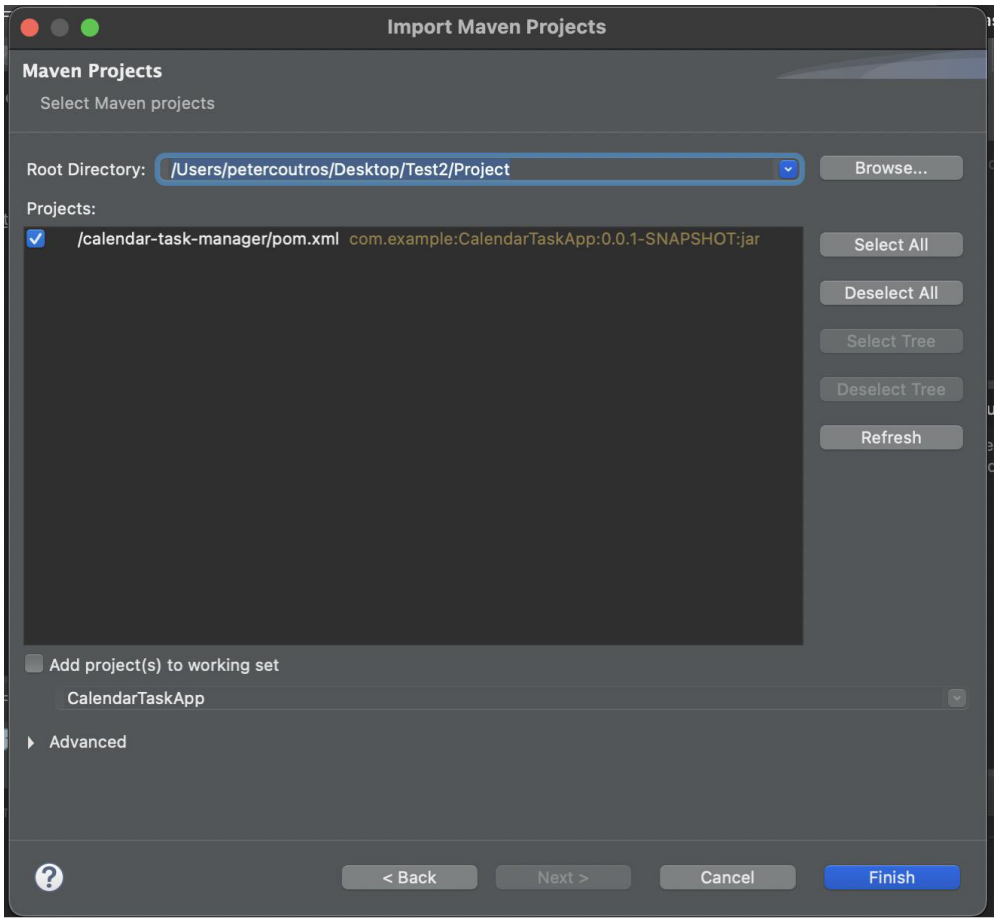

Fig. 8

With the project now open, navigate to the file containing the main method (currently housed in CalendarTaskAppApplication.java) and run as “Spring Boot App” (figure 9).

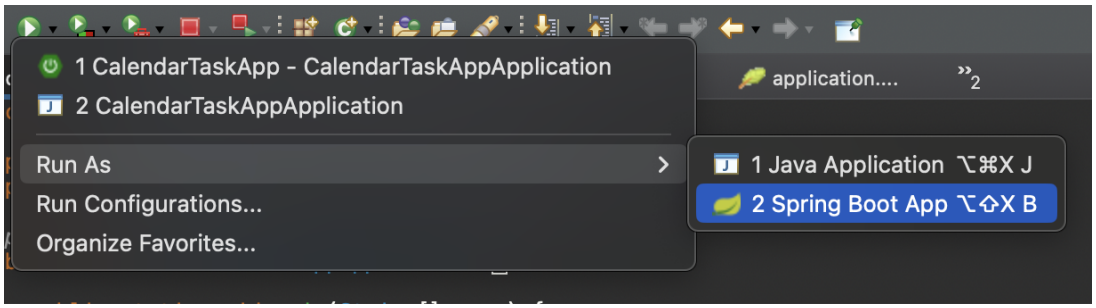

Fig. 9

Navigate to your browser of choice and enter http://localhost:8080/ into the navigation bar to interact with the application.

2. **Design the Software Architecture**
    - **Components**: The main components of the system include task management, event scheduling, and the integration of the Thymeleaf front-end with the back-end.
    - **Interactions**: The user interacts with the calendar through a web-based interface, using forms to create and edit tasks or events.
    - **Database**: H2 is used for data storage. The system supports basic CRUD operations.
3. **Core Functionality**
    - **Core Features**: The current implementation includes a basic home page and task manager logic. The Thymeleaf front-end is integrated, and the back-end supports task creation and retrieval.
    - **Spring Boot Configurations**: Key configurations like setting **spring.thymeleaf.prefix** and **suffix** are necessary for Thymeleaf to work properly. The web application runs on port **8080**.
4. **Unit Testing:**
    - **Unit Testing**: JUnit is used for unit testing. Tests focus on task creation, deletion, and retrieval. Further integration testing is needed once the front-end and back-end communication is fully functional.
    - **Error Handling**: Issues like the "404 not found" error have been encountered, requiring adjustments in routing and Thymeleaf configuration. Debugging is ongoing to resolve these issues.

Code Example

Here's a sample code snippet for the **Task Manager** component:

```java
package com.example.calendartaskapp.service;


import java.util.List;

import org.springframework.stereotype.Service;

import com.example.calendartaskapp.form.TaskForm;
import com.example.calendartaskapp.repository.TaskFormRepository;

/**
 * This is a Spring Service class that provides business logic for TaskForm entities.
 *
 * It is annotated with @Service, which marks it as a Spring Bean.
 */
@Service
public class TaskService {
    /**
     * The TaskFormRepository instance, which is used to interact with the database.
     *
     * It is injected through the constructor, which is a good practice for dependency injection.
     */
    private final TaskFormRepository taskFormRepository;

    /**
     * The constructor, which takes a TaskFormRepository instance as a parameter.
     *
     * It is used to initialize the taskFormRepository field.
     */
    public TaskService(TaskFormRepository taskFormRepository) {
        this.taskFormRepository = taskFormRepository;
    }

    /**
     * A method to retrieve all TaskForm entities from the database.
     *
     * It uses the findAll() method of the TaskFormRepository to fetch all tasks.
     */
    public List<TaskForm> getTasks() {
        return taskFormRepository.findAll();
    }

    /**
```

Conclusion

Phase I of the Calendar/Task Tracker project involves setting up the project environment, implementing core features, and conducting initial tests. The team is collaborating via GitHub to ensure version control and address any issues like the 404 error. Future phases will focus on expanding functionality and integrating more complex features, including user authentication and advanced scheduling options.