**To-Do List Application with Spring Boot and H2 Database**

* Create a new Spring Boot project using Spring Initializr
* Include the dependencies for "Spring Web" and "Spring Data JPA." , “Spring Boot Devtools” , “H2 database”, “Thymeleaf”
* Set up a simple project structure.

Application.properties

**# DataSource configuration**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

**# H2 Console configuration**

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

**# Hibernate configuration**

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

**# Thymeleaf configuration**

spring.thymeleaf.cache=false

spring.thymeleaf.prefix=file:src/main/resources/templates/

spring.thymeleaf.suffix=.html

**Entity and Repository**

An entity represents a table in a relational database. It is a Java class that is annotated with @Entity to indicate that instances of this class will be persisted to a database. Each instance of the entity class corresponds to a row in the database table.

A repository is an interface that extends the JpaRepository interface provided by Spring Data JPA. This interface defines methods for performing common database operations (CRUD - Create, Read, Update, Delete) on the entity.

**Entity**  
**modal/Task.java**  
package com.todo.app.modal;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

@Entity

public class Task {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String title;

    private String description;

    private boolean completed;

    // Constructors, getters, and setters

    public Task() {

    }

    public Task(String title, String description, boolean completed) {

        this.title = title;

        this.description = description;

        this.completed = completed;

    }

    // Getters and setters

    public Long getId() {

        return id;

    }

    public void setId(Long id) {

        this.id = id;

    }

    public String getTitle() {

        return title;

    }

    public void setTitle(String title) {

        this.title = title;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

    public boolean isCompleted() {

        return completed;

    }

    public void setCompleted(boolean completed) {

        this.completed = completed;

    }

}

**Repository**

repo/TaskRepository.java

package com.todo.app.repo;

import org.springframework.data.jpa.repository.JpaRepository;

import com.todo.app.modal.Task;

public interface TaskRepository extends JpaRepository<Task, Long> {

}

**Controller**

**controller/TaskController.java**

package com.todo.app.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import com.todo.app.modal.Task;

import com.todo.app.repo.TaskRepository;

import java.util.List;

import java.util.Optional;

@Controller

@RequestMapping("/tasks")

public class TaskController {

    @Autowired

    private TaskRepository taskRepository;

    @GetMapping

    public String getAllTasks(Model model) {

        List<Task> tasks = taskRepository.findAll();

        model.addAttribute("tasks", tasks);

        return "tasks";

    }

    @GetMapping("/new")

    public String showTaskForm(Model model) {

        model.addAttribute("task", new Task());

        return "task-form";

    }

    @PostMapping("/new")

    public String addTask(@ModelAttribute Task task) {

        taskRepository.save(task);

        return "redirect:/tasks";

    }

    @GetMapping("/{id}")

    public String editTaskForm(@PathVariable Long id, Model model) {

        Optional<Task> optionalTask = taskRepository.findById(id);

        if (optionalTask.isPresent()) {

            model.addAttribute("task", optionalTask.get());

            return "task-form";

        } else {

            return "redirect:/tasks";

        }

    }

    @PostMapping("/{id}")

    public String updateTask(@PathVariable Long id, @ModelAttribute Task updatedTask) {

        Optional<Task> optionalTask = taskRepository.findById(id);

        if (optionalTask.isPresent()) {

            Task task = optionalTask.get();

            task.setTitle(updatedTask.getTitle());

            task.setDescription(updatedTask.getDescription());

            task.setCompleted(updatedTask.isCompleted());

            taskRepository.save(task);

        }

        return "redirect:/tasks";

    }

    @GetMapping("/delete/{id}")

    public String deleteTask(@PathVariable Long id) {

        taskRepository.deleteById(id);

        return "redirect:/tasks";

    }

}

**Templates**

**tasks.html**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <title>Task List</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL" crossorigin="anonymous"></script>

</head>

<body>

    <div class="container mt-3">

        <h2>Task List</h2>

        <ul class="list-group">

            <li th:each="task : ${tasks}" class="list-group-item">

                <span th:text="${task.title}"></span>

                <span th:text="${task.description}"></span>

                <a th:href="@{'/tasks/' + ${task.id}}" class="btn btn-warning btn-sm">Edit</a>

                <a th:href="@{'/tasks/delete/' + ${task.id}}" class="btn btn-danger btn-sm">Delete</a>

            </li>

        </ul>

        <a th:href="@{'/tasks/new'}" class="btn btn-primary mt-3">Add New Task</a>

    </div>

</body>

</html>

**task-form.html**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <title>Task Form</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL" crossorigin="anonymous"></script>

</head>

<body>

    <div class="container mt-3">

        <form class="col-6 mx-auto card p-3 shadow-lg " th:action="@{/tasks/new}" method="post" th:object="${task}">

             <h2>Task Form</h2>

             <hr>

            <div class="form-group">

                <label for="title">Title:</label>

                <input type="text" id="title" name="title" class="form-control" th:field="\*{title}" required>

            </div>

            <div class="form-group">

                <label for="description">Description:</label>

                <input type="text" id="description" name="description" class="form-control" th:field="\*{description}">

            </div>

            <div class="form-group">

                <div class="form-check">

                    <input type="checkbox" id="completed" name="completed" class="form-check-input" th:field="\*{completed}">

                    <label class="form-check-label" for="completed">Completed</label>

                </div>

            </div>

            <button type="submit" class="btn btn-primary">Save</button>

        </form>

        <a style="margin-left: 350px;" th:href="@{'/tasks'}" class="btn btn-secondary mt-3">Back to Task List</a>

    </div>

</body>

</html>