Abstract

This document documents a developed Task Management Application using Spring Boot that offers the functionality of task management via a RESTful API, mainly providing basic CRUD operations. Basic Authentication is used to secure access to the API and a HashMap for the storage of task data in memory. The systems are designed to perform tasks such as creating, retrieving, updating, and deletion of the tasks. Each task has a unique ID, a title, a description, a due date, and a status.

The system is designed as a proof-of-concept, illustrating basic API functionality, secure access, and simple management of tasks. Design decisions and assumptions formed throughout the development process are described in detail; this discussion is focused on structure and potential for future extension of the system.

INDEX

Table of Contents

Introduction

RESTful APIs are the most extensively used exposure mechanisms for functionalities and data in modern web development. This is a very simple application of a Task Management API, designed to exemplify the concept of task management by a web service. The base CRUD operations have been implemented using Spring Boot for the purpose of managing tasks (Create, Read, Update, and Delete).

The application is designed to handle task data with fields such as title, description, due date, and status, offering a straightforward mechanism for the creation and management of tasks. The API is also secured using Basic Authentication to ensure that only authorized users have access to it.

Aim

- The main objective of this project is to show how to:
- Implement a RESTful API using Spring Boot.
- Integrate Basic Authentication for securing the API.
- Design and manage CRUD operations for tasks.
- Provide a simple and scalable in-memory task management system.
- Test the API using JUnit and demonstrate how to interact with it using tools like Postman or cURL.

Scope of Project

This project encapsulates the following:

Task Management API:

- **Basic operations**: Create, Read, Update, Delete tasks.
- Secure access via Basic Authentication.

Task Representation:

- Each task would then have an ID, title, description, due date, and status.

In-Memory Storage:

- Tasks are stored in a HashMap for simplicity.

Testing:

- Basic API tests using JUnit and RestTemplate.

Authentication:

- **Basic Authentication** is implemented with a single user (admin).

This system is actually meant to be a prototype or proof of concept, so though it is working, it's not ready for production without further development.

Design Decisions

Choice of Framework: Spring Boot:

- **Reason:** Spring Boot offers quick turnaround on development and easy configuration settings. Much boilerplate code required in Java web applications is mostly removed, and built-in support for REST APIs, embedded servers, and auto-configuration provides ease of development.
- **Impact:** It was rather helpful since Spring Boot automatically configures in many areas, like database and security settings, to quickly assemble a working API.

Task Structure:

- **Reason:** All these activities incorporate must-have fields like title, description, due_date, and status. This provides for the basic structure of the activity management while keeping flexibility for potential later extensions.
- **Impact:** The structure is simple and sufficient for this application but could be expanded to include additional fields (e.g., priority, categories, or user assignment) in a more complex application.

Task Storage: In-Memory HashMap:

- **Reason:** For simplicity of implementation and demonstration of the API functionality, tasks are persisted in an in-memory HashMap. The structure allows for really fast lookups as well as easy manipulation of task data.
- **Impact:** Since the storage is chosen as in-memory, data has no persistence to it after an application restart. For persistence in a production environment, a database would be required.

Authentication: Basic Authentication:

- **Reason:** Implementing Basic Authentication for the API presented certain access control. However, it is simple and does not require additional infrastructure to be installed or implemented.
- **Impact:** Basic Authentication is suitable for small scale or internal applications; however, it is less secure when using token-based authentication like JWT and should be used only in secure environments.

Implementation and Testing

API Endpoints
The application has the following RESTful API endpoints:

1. **POST /task: Create new task**
    - **Input**: JSON containing details about a task (title, description, due_date, status).
    - **Output:** HTTP status 201 Created with a message confirming that the task has been added.

2. **GET /task: Returns all tasks.**
    - **Output:** JSON array with all tasks in the system.

3. **GET /task/{taskId}: Returns a single task by its ID.**
    - Output: A JSON object that represents this task, or a 404 Not Found if such task does not exist.

4. **PUT /task/{taskId}: Modify an existing task.**
    - **Input:** JSON with updated details of the task.
    - **Output:** HTTP status 200 OK, or 400 Bad Request if the task ID is not found.

4. **DELETE /task/{taskId}: Deletes a task by its ID**

- **Output:** HTTP status 204 No Content, or 404 Not Found.

Testing Approach:

Testing of the API was done using **JUnit** and **RestTemplate**. A simple test case (`ControllerTest`) was created to:

- Send a POST request to add a task.
- Assert that the response is correct (HTTP status `201 Created` and the message "Task added").

Assumptions

No Persistence Layer:

- The system assumes that data is not stored between an application restart. If persistence is expected, database integration such as MySQL or MongoDB would be added.

Single User Authentication:

- The system assumes a single user of the application, typically an administrator. For multi-user applications, user administration and role-based access control would be required.

Basic Input Validation:

- The application assumes that the input is correct and does not do complex validation on the fields (e.g., checks that due_date is indeed a date).

No Complex Error Handling:

- The system relies on basic error responses (e.g., 404 Not Found, 400 Bad Request), but

there are no more complex error cases or validations.

Security Considerations:

- Basic authentication is used, which is suitable for simple applications. For production systems, more secure authentication mechanisms, such as OAuth2 or JWT, shall be used instead.

Conclusion:

- This Task Management Application is a prototype in the management of tasks by simple RESTful API. The features provided include the most basic kinds of creating, retrieving, updating, and deletion, all securely accessible through Basic Authentication.

- It's designed for simplicity and ease of use, but there are plenty of scopes for improvement. For instance, persistent storage, more robust error handling, more advanced input validation, and support for multiple users should be developed. Future enhancements could include the integration of a database, some more complex authentication methods, and an extension of the task model in order to allow more fields and perhaps their respective business logic.