

## Solution Overview

This bookstore solution is built with Spring Boot and secure it with Spring Security and JWT.

We'll rely on the following technology stack to write the necessary code required to build and secure our Employee REST API:

- Spring Boot
- Spring Security
- H2 database
- Jjwt
- Maven

## Security

The solution is security with Spring Security and JWT. There are 3 roles provided.

Role	Operation
MANAGER	Retrieve, Create, Update, Delete
ADMIN	Retrieve, Create, Update
GUEST	Retrieve information only

There are 3 preconfigure user is provided for testing purposes.

- 1) Manager
- 2) Admin
- 3) Guest

## Rest API

In general, the API will expose the following endpoints:

GET /api/find-all-books	Get all Books available.
GET /api/find-books	Retrieve books by Book Title or Author Name.
POST /api/create-book-record	Create a new book record

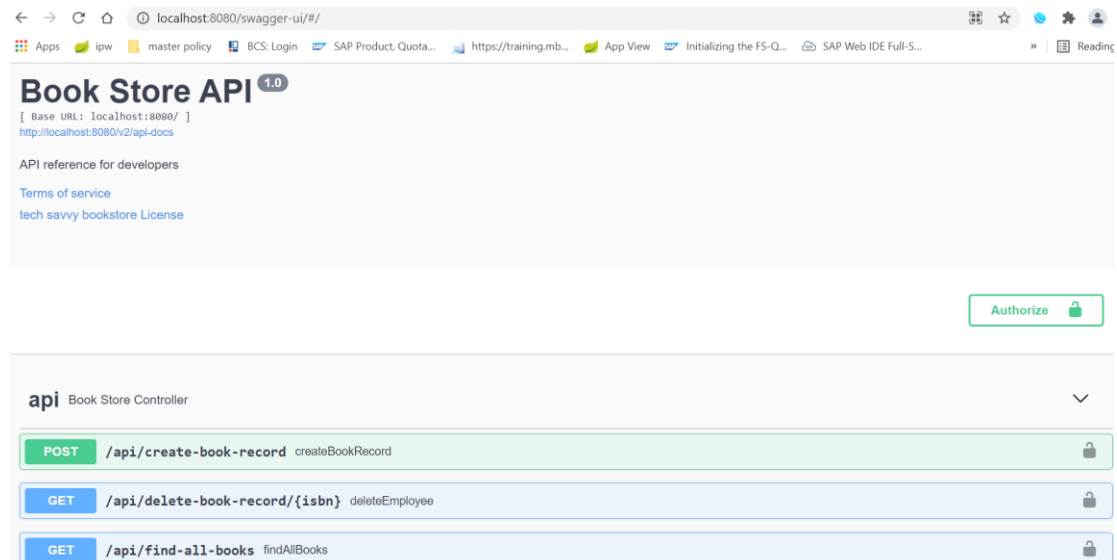
POST /api/update-book-record	Update book record base on isbn.
GET /api/delete-book-record/{id}	delete a particular book record based on isbn.
POST /auth/signin	Sign in using username and password which will return a JWT token to access API resources.

## Starting the application.

- 1) Download the jar file **bookstore-0.0.1-SNAPSHOT.jar**
- 2) Open a command prompt and navigate to the folder which contains the jar file.
- 3) Issue the cmd **java -jar bookstore-0.0.1-SNAPSHOT.jar**
- 4) When you see this out put at console. The application has been started.

```
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
main] tech.savvy.bookstore.Application : Started Application in 11.192 seconds (JVM running for 11.732)
```

- 5) Open your browser and enter the url <http://localhost:8080/swagger-ui/>
- 6) You should be able to view the details of the api in the swagger ui.



- 7) Open your browser and enter the url <http://localhost:8080/index.html>

8) You will see a page which you can generate access token to use when testing the end point.

The screenshot shows a web browser at localhost:8080/index.html. The page has a title 'Generate Token' and a table with three columns: 'Login User', 'Action', and 'Token'. The 'Login User' column lists 'Manager', 'admin', and 'guest'. The 'Action' column has a 'Get Token' button for each user. The 'Token' column shows a long alphanumeric string for the 'Manager' user and empty text boxes for 'admin' and 'guest'.

Login User	Action	Token
Manager	Get Token	<code>{"token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJtYW5hZ2VYIiwiaWF0IjoxNjM0NDc0NjE0LCJleHAiOiJlZWZ0NzUyMTR9.4hiLe9um6ySYFKNcH1FrUssR1aXe_cZ_DzRSNmxxkdZBMU_WhXPH178d1_GFeTHsTj3xnq5KOT-M2gHqzuiQgIg"}</code>
admin	Get Token	
guest	Get Token	

## Testing the endpoint

You can test using curl or post man. It would be easier to use a post man for this purposes.

- 1) Open postman application.
- 2) Choose the action type and enter the endpoint. (for example)

The screenshot shows the Postman application interface. The 'GET' method is selected, and the endpoint is 'http://localhost:8080/api/find-all-books'. The 'Send' button is visible.

- 3) Get an access token from a user in the index.html page (such as manager) and create a header with the token. (take note it has an expiry time of 10 min, a new token is required after 10 min)

The screenshot shows the Postman application interface with the 'Headers' tab selected. A table with 'Key' and 'Value' columns is visible. The 'Authorization' header is added with the value 'Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJtYW5hZ2VYIiwiaWF0IjoxNjM0NDc0NjE0LCJleHAiOiJlZWZ0NzUyMTR9.4hiLe9um6ySYFKNcH1FrUssR1aXe\_cZ\_DzRSNmxxkdZBMU\_WhXPH178d1\_GFeTHsTj3xnq5KOT-M2gHqzuiQgIg'. The 'Status' is 200 OK and the 'Time' is 41 ms.

Key	Value	Description
<input checked="" type="checkbox"/> Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJtYW5hZ2VYIiwiaWF0IjoxNjM0NDc0NjE0LCJleHAiOiJlZWZ0NzUyMTR9.4hiLe9um6ySYFKNcH1FrUssR1aXe_cZ_DzRSNmxxkdZBMU_WhXPH178d1_GFeTHsTj3xnq5KOT-M2gHqzuiQgIg	

- 4) Click on the send button and you should be able to see the result.

The screenshot displays a REST client interface with the following components:

- Request Bar:** Method: GET, URL: `http://localhost:8080/api/find-all-books`, Params: empty.
- Buttons:** Send (blue), Save (grey).
- Tabs:** Authorization, Headers (1), Body, Pre-request Script, Tests.
- Headers Tab:** Contains one header: Authorization (checked) with value: Bearer eyJhbGciOiJIUzUxMiI9.eyJzdWIiOiJtYW5hZ2VyiWiaWF0Ijox... and a new key/value pair below it.
- Body Tab:** Active tab showing the response body. It includes a status bar: Status: 200 OK, Time: 41 ms.
- Response Body:** A JSON object representing a book and its author, highlighted with a red rectangle:

```
[
  {
    "isbn": "1234567890123",
    "title": "Home alone",
    "year": 1999,
    "price": 30,
    "genre": "adventure",
    "authors": [
      {
        "name": "Jones",
        "birthday": "30-12-1979"
      }
    ]
  }
],
```
- Format Selectors:** Pretty, Raw, Preview, JSON (selected), and a copy icon.

- 5) Similar step can be use for others endpoint. Take note of the parameter and body needed. The information can be found the swagger ui page.