WA2892 Booz Allen Hamilton Tech Excellence Modern Software Development Program - Phase 1

Student Labs

LabGuide 2

(Spring Project)

Web Age Solutions Inc.

Table of Contents

Lab 1 - Build a Skeleton API

In this lab you will design a simple RESTful Web Service API to replace the function of the command line Stock Tracker.

Next you will implement the API in a skeleton Spring MVC service. This service will not initially use the existing Stock framework (the next project lab will connect the skeleton Web service to the Stock framework).

Finally, you will use Postman to call the skeleton Web service.

Part 1 - API Requirements

- The API must allow users to create an account
- The API must allow users to buy stock and dividend stock
- The API must allow users to sell stock and dividend stock
- The API must allow users to retrieve account name
- The API must allow users to retrieve account balance
- The API must allow users to retrieve a list of held stocks
- The API must allow users to retrieve all account information

Part 2 - RESTful Service Design Hints

For this project we will only allow one account to be created. Use JSON for both request and response bodies.

The create, buy and sell operations should use the HTTP POST method. The retrieve operations should use the HTTP GET method.

Make sure you set the appropriate HTTP status codes in your responses.

Part 3 - Examine the Starter Application

There is a starter application for this lab at:

C:\LabFiles\project-skeleton-api.zip

You may import this application as a Maven project into your IDE.

This application provides an implementation stub for:

Creating an account:	POST /stocks/account (uses AccountInfo value-object)
Getting account:	GET /stocks/account

Part 4 - Build and Deploy the Starter Application

In this part you will build the starter application WAR file, rename it and copy it into your server's *deployment* directory.

__1. Build the starter application with the Maven *package* goal (this will generate the WAR file in the *target* directory.

__2. Rename the WAR file from *StockService-0.0.1-SNAPSHOT.war* to *stockservice.war*

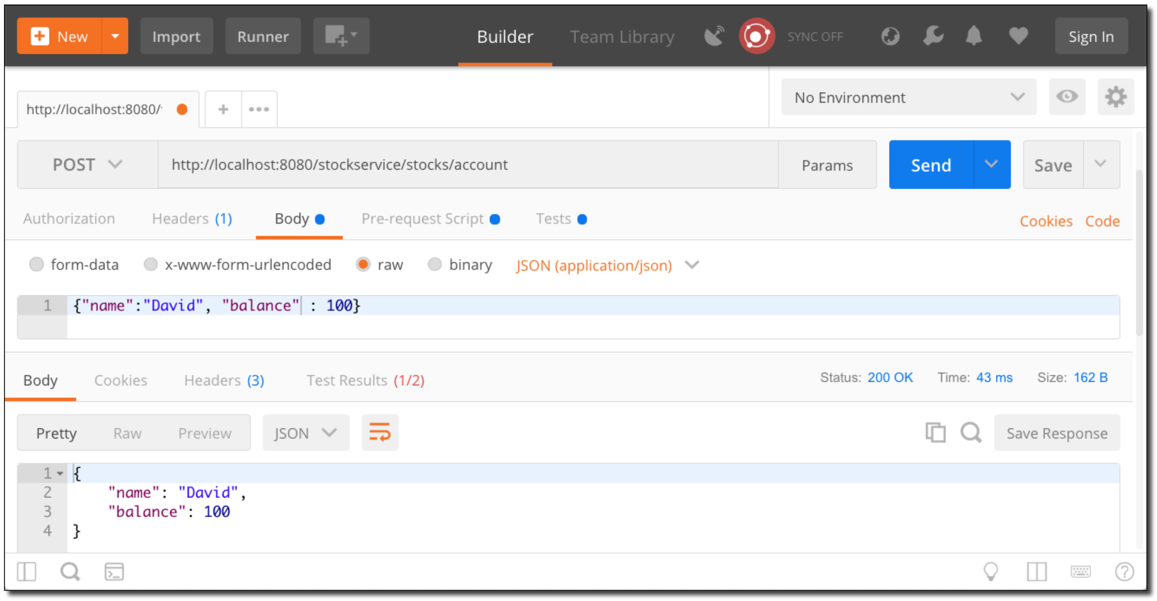
__3. Copy *stockservice.war* into the deployment directory of your server (*webapps* directory on Tomcat)

__4. Make sure your server is running.

Part 5 - Test the Starter Application with Postman

In this part you will use Postman to test the application. Try calling the create account and get account services.

Create URL	http://localhost:8080/stockservice/stocks/account (POST)
Get URL	http://localhost:8080/stockservice/stocks/account (GET)



POST http://localhost:8080/stockservice/stocks/account

JSON (application/json)

```
{"name":"David", "balance" : 100}
```

Status: 200 OK Time: 43 ms Size: 162 B

JSON

```
{
    "name": "David",
    "balance": 100
}
```

Part 6 - Complete the Skeleton Application

In this part you will add skeleton implementations for the other service operations. Use Postman to test each of your service skeletons. At this point you can make all the service return strings. The goal is simply to build the skeleton and make sure you can call all of the operations.

Part 7 - Review

In this lab we designed a Web Service API for the Stock Trader Application, we then created a skeleton web service and tested the operations in Postman.

Lab 2 - Complete the RESTful Stock Service

In this lab you will complete the project by hooking the Web Service you created up to the Stock Trader classes.

Part 1 - Copy the Stock Trader files into the Stock Service

In this part you will copy the source files (and directories) from the Stock Trader application into the Stock Service project.

If you put the Stock Trader and Stock Service files in the same Java package, you may want to refactor and put them in separate packages.

Note: in this method of code sharing/re-use (copying files) should not be used in production. Instead we would package up the files that do the *back-end* stock work into a JAR file and share it that way. Ideally through an enterprise Artifactory or Nexus repository.

Part 2 - Link the Service Skeleton to the Stock Trader Files

In this part you will take the Web Service stub operations that you created in the StockService application and in each method add calls to the Stock Trader back end. The Web service code you wrote in the last section my need to be modified to support the back end.

Be sure to add HTTP Status codes for error handling. To simplify things you can just return `HttpServletResponse.SC_INTERNAL_SERVER_ERROR` for all errors.

Part 3 - Test the Completed Service

In this part you will use Postman to test the completed service.

Try creating an account, buying and selling stocks and viewing account information.

Part 4 - Review

In this lab you completed the project by connecting the *skeleton* Web service to the Stock Trader backend. You then tested the service in Postman to verify that it worked correctly.