Contents

[Demystifying Microservices: 2](#_Toc456941302)

[Principles of micro services: 2](#_Toc456941303)

[Single responsibility per service: 2](#_Toc456941304)

# Demystifying Microservices:

Microservices are not invented; they are more of an evolution from the previous architecture styles.

Business demand as a catalyst for microservices evolution

# Principles of micro services:

## Single responsibility per service:

The single responsibility principle is one of the principles defined as part of the SOLID design pattern. It states that a unit should only have one responsibility.

This implies that a unit, either a class, a function, or a service, should have only one responsibility. At no point should two units share one responsibility or one unit have more than one responsibility. A unit with more than one responsibility indicates tight coupling.

# Profiles and Yaml File:

## Approach 1:

Maintaining all the profiles in a properties/yaml file.

-Dspring.profiles.active=prod

This picks up the “**prod**” profiles that is mentioned in the yaml file.

spring:

profiles: development

server:

port: 9090

---

spring:

profiles: test

server:

port: 8080

bootrest: Hello from test environment Yaml file

---

spring:

profiles: prod

server:

port: 8080

bootrest: Hello from prod environment Yaml file

### Running from Command line:

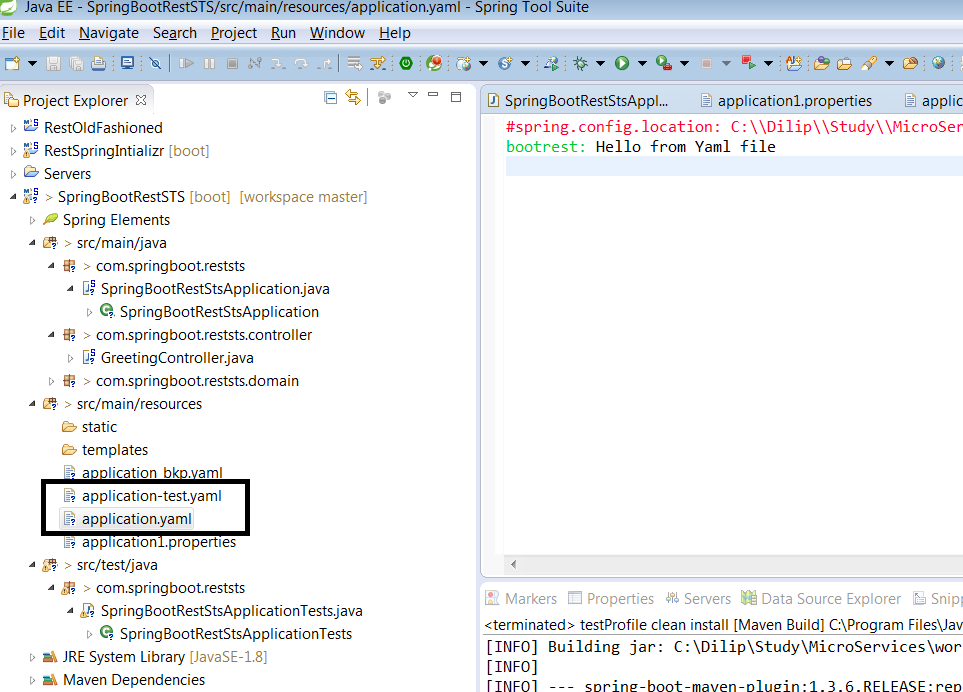
java -jar -Dspring.profiles.active=test target/demo-0.0.1-SNAPSHOT.jar

## Approach 2:

Maintaining different profiles in separate files.

For “test” profile:

We need to have the **“application-test.yaml”** file in the classpath. So when we run the application with -"Dspring.profiles.active=test” VM argument then the application it will pick up the “application-test.yaml” file.



# Securing Micro Services with Basic Security:

Set the user name and password in the application.properties file.

