Spring Boot Training Running Notes
---

**Trainer Contact:** Pradeep Kumar K
https://www.linkedin.com/in/prashdeep/
**Reference Projects:** We can clone topic wise projects from trainer's Github public projects.
Filter contribution activity day wise to get the relevant projects from the below link:
https://github.com/prashdeep
**IDE:** Intellij / Eclipse(with STS plugin)

Day 1 (21-Nov-2019):

1. Spring Boot Overview-AOP, IOC, DI, Annotations
2. Spring vs Spring Boot Framework
3. Spring versions and features
4. Declarative & Imperative programming
5. Spring- Bill of Materials(BOM)
6. Spring MVC practical
7. Design pattern- Builder, Interceptor

Day 2 (22-Nov-2019):

8. Spring Starter Dependencies
9. Custom Auto Configuration
10. Spring Boot MVC
11. Spring Boot REST with WebMVC
12. Spring JPA
13. Security
14. Actuator
15. Config Properties
16. Profiling
17. Miscellaneous Topics(Lombok Plugin, Exception Handling)

Day 3 (25-Nov-2019):

Spring security
---

Refer: https://github.com/prashdeep/toshiba-spring-mvc-app
Reference PDF:
https://files.meetup.com/6015342/Spring%20Toronto%20-%20Joe%20Grandja.pdf
-Oauth2 authentication (token authentication) to be discussed later
-Common errors in security: 401-unauthorised, 403-access denied
-Security types--->
    1. in memory authentication
    https://www.baeldung.com/spring-security-multiple-auth-providers
    2. dao or db backed authentication
    https://www.baeldung.com/spring-security-authentication-with-a-database
here this can be done using annotation or xml config, better to go with Annotation

-AuthenticationManagerBuilder-userDetailsService, passwoedEncoder
-password encoder:
    1.Bcrpyt
(strong)-https://www.baeldung.com/spring-security-registration-password-encoding-bcrypt
https://stackoverflow.com/questions/53516952/spring-boot-how-to-check-if-encoded-password-from-db-is-matching-with-password-f
    2.md5(weak)
-Java8: Stream, map, collect
-Spring cloud and Sping microservcies--> more about scalability and resiliency, business logic implementation is less

Microservices
---
Reference Book:
https://www.manning.com/books/spring-microservices-in-action?query=springg

-based on domain, service is split into microservices
-CAP: consistency, Availability,Partition
-micro services has eventual consistency
-vertical and horizontal scalability
-micro services are distributed, partition tolerant
-Murphy law
-adv: scale particular services on demand, issue in one service alone(Eg-payment good , customer service down, delivery good), test particular service

-No particular rule to break service to micro service(depends on architect , developer and dev ops view)
-Spring cloud has all necessary design patterns from microservices
-Netflix first used microservices
-every microservice should have github repo
-separate config
-create instance on the fly(we use config server(preconfigured)), kill any time
-main service should not talk to microservice through hard coded url or end point instead they need to talk using managed config(use Eureka). Eureka- all micro services are called eureka servers
-Eureka
 -brings resiliency on failure of any microservice. It maintains
 -It maintains cache
 -based on annotation microservice binds with eureka server
-Resiliency Patterns: on failure is it fallback or something else(what it needs to do on failure)???? This pattern has pre configured things for us to use
-Zuul : 1 use case: we can rate-limit of rest endpoint hits(load balancing)

Zuul gateway service proxy – It would be again a spring boot based, which will basically intercept all the traffic of student service and apply series of request filter and then route to the underlying service and again at the time of response serving, it will apply some response filtering. Since it is a gateway, we can literally take many interesting and useful action using the filters effectively.

-Sleuth:
-Zipkin
-No standard pattern for transaction
-pattern -CQRS, event pattern , event streaming further these events can be sent to Active MQ/AI/ML and analyze further
-Spring cloud has parent spring cloud starter

reference-https://github.com/prashdeep/configuration-service
-Hoxton, dependency management, spring cloud, spring configuration processor
-"{ciper}encrypted_password"

Imp:

-Config Service (very important, used in AxisRooms for storing dev, prod url and encrypted password, we were expecting key from client and comparing with this encrypted password. we have custom password generator)
-Microservices and Spring Cloud Config Server
-Spring Boot - Cloud Configuration Server
-Spring Boot - Cloud Configuration Client
Automate refreshing microservices on any prop change from main appl
-value(${})
-@RefreshScope use in main appl
https://www.tutorialspoint.com/spring_boot/spring_boot_cloud_configuration_client.htm

Check all springBoot concepts here-
https://www.tutorialspoint.com/spring_boot/index.htm

Eureka
---
-To manage all clients: Eureka Server- captures client's heart beat
Eureka Server is an application that holds the information about all client-service applications. Every Micro service will register into the Eureka server and Eureka server knows all the client applications running on each port and IP address. Eureka Server is also known as Discovery Server.

```
eureka:
  client:
    registerWithEureka: false //here eureka server itself cant be a client , so mark both properties as false
    fetchRegistry: false
  server:
    port: 8761
```

-Read about eureka server and client
-@enableeurekaclient(usual) vs @enablediscoveryclient(use this when u r working with Spring Cloud)

Tomo plan---> Hystrix, Zuul
Prerequisite → Install docker, Apache kafka

Day-4 (26-Nov-2019)

-Coming up with many instances of Eureka Server and find how they talk among themselves
https://cloud.spring.io/spring-cloud-static/Greenwich.SR4/single/spring-cloud.html
-Ribbon Client, feign client, discoveryClient
-@LoadBalanced
https://howtodoinjava.com/spring-cloud/spring-boot-ribbon-eureka/
-Swagger working, @enableSwaggerConfig
-Spring Rest Docs(better than Swagger)
-Git Markdown: **GitHub** combines a syntax for formatting text called **GitHub** Flavored **Markdown** with a few unique writing features. Better than Google docs
-Spring cloud starter Netflix Hystrix
-Circuit Breaker Design pattern
-Bulk Head Resiliency
-Linked Blocking queue
-Zuul

Docker→
---
Alpine-based on open JDK 8
JDK
Write one Docker file
Refer Spring boot docker
Spotify Plugin
Use Docker compose- when using more than 1 app with db dependency

Docker run -p (port) -d(detach)
Learning
---
- Reactive programming, webflux(learn these Spring -5 topics, relate them with previous startup OTA integration project)

Day-5 (27-Nov-2019)

-Zuul
-https://www.baeldung.com/spring-security-prefilter-postfilter

OAuth 2.0 specification defines 4 types of authorization flows:

- Authorization Code
- Resource Owner Password Credentials
- Implicit
- Client Credentials

- Oauth2: authorization code grant flow(discussed)
https://medium.com/google-cloud/understanding-oauth2-and-building-a-basic-authorization-server-of-your-own-a-beginners-guide-cf7451a16f66
-Commonly used standard Oauth- okta
refer-https://www.zoho.com/books/api/v3/#oauth
-Spring Cloud Stream → used for real time updates
-Kafta working