Spring Boot - Microservices

Satyendra Singh
satyendra.singh@capgemini.com

Table of Content

- What you will learn
- Pre-requisite for the course
- Software requirement
- Day wise schedule
- Useful resource link
- Assignments to be solved

What you will learn

- Understand the needs that Spring Boot fulfills
- You will understand how Spring Boot simplify dependency management and configuration. Spring Boot auto-configuration
- How to use Spring Boot starters and start.spring.io to easily create new applications
- Understand Spring Boot's customization of configuration details
- Understand and use Spring Boot's Spring Data / Spring Data JPA capabilities
- Understand and use Spring Boot's Web capabilities (focusing on Spring REST), including embedded servlet containers.
- How to use Spring Security with Spring Boot
- How to use Spring Boot’s Actuator endpoints to monitor and manage applications
- How to use and work with DevTools and Spring CLI
- Introduction of Message-Queueing (RabbitMQ)
- How to install and configure RabbitMQ. Understand architecture of RabbitMQ
- How to implement messaging patterns and applications using the Java client
- Learn how to develop message listeners for specific queues and routings

What you will learn

- How to implement Topics, Queues, Exchanges and Bindings in RabbitMQ
- How to design asynchronous, message-driven systems with RabbitMQ
- Introduction to microservices, Architecture of microservices and 12 factor app
- How to build Spring Boot based microservices.
- Introduction to Spring Cloud
- Understand the major components of Netflix OSS
- How to register services with a Eureka Service and implement “client” load balancing with Ribbon to Eureka managed Services.
- How to create fault-tolerant services with Hystrix callbacks
- How to filter requests to your Microservices using Zuul
- Introduction to Docker. Basic Docker commands.
- How to pull images from a Docker Registery and push images to Docker Hub
- Understand what Docker compose and Swarm is.
- How to containerize a web-based application with a microservice approach and automate it using Dockerfile.
- How to deploy microservices on public cloud (AWS)

Pre-requisite for the course

- Spring Framework
- Web Services (REST)
- Basic Linux commands.

Softwares Required

- JDK 8
- STS (latest)
- Maven 3
- Postman (latest)
- RabbitMQ (latest)
- Docker (latest)
- AWS free trial subscription
- Internet Access

Table of Content

REST Architecture Principles

Agenda

- Introduction to REST
- REST Architecture Principles
- Designing RESTful services

REST Architecture Principles

Synopsis

- REST is an architectural style that defines a set of constraints that, when applied to the architecture of a distributed system, induce desirable properties like loose coupling and horizontal scalability. RESTful web services are the result of applying these constraints to services that utilize web standards such as URIs, HTTP, XML, and JSON. Such services become part of the fabric of the Web and can take advantage of years of web engineering to satisfy their clients' needs.

Objective

- You will learn the concepts and principles of RESTful Architecture

REST Architecture Principles

Reference Material (4 hours 30 minutes)

RESTful Concepts
and Principles

Spring Boot

Agenda

- Introduction
- Using Spring Boot
- Spring Boot Features
- Spring Boot Actuator
- Deploying Spring Boot Applications
- Spring Boot CLI
- Spring Boot Plugins

Spring Boot

Synopsis

- Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run“. It take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration

Objective

- This section goes about how you should use Spring Boot. It covers topics such as build systems, auto-configuration, and how to run your applications. It also cover some Spring Boot best practices.

Spring Boot

Reference Material

Spring Boot

Building RESTful Services using Spring Boot

Agenda

- Building Addressable resources using @RequestMapping and @RestController using Uniform and Constraint Interface
- @GET, @PUT, @POST, @DELETE
- API Naming Convention
- Building a simple CRUD services
- Making service Representation-Oriented
- Building Pagination of data records with HATEOAS
- Building ResponseEntity with response code
- Securing Services

Building RESTful Services using Spring Boot

Synopsis

- Writing RESTful services in Java has been possible for years with the servlet API. If you have written a web application in Java, you are probably already very familiar with servlets. Servlets bring you very close to the HTTP protocol and require a lot of boilerplate code to move information to and from an HTTP request. In 2008, a new specification called JAX-RS was defined to simplify RESTful service implementation.

Objective

- You will be able to implement JAX-RS standards with Spring Boot to build enterprise grade application using RESTful Services.

Building RESTful Services using Spring Boot

Reference Material

- https://spring.io/guides/gs/rest-service/
- https://www.youtube.com/watch?v=msXL2oDexqw&list=PLqq-6Pq4lTTbx8p2oCgcAQGQyqN8XeA1x
- https://dzone.com/articles/securing-rest-services-with-oauth2-in-springboot-1

- Basic Demos and OAuth2.0 with Social Media Account
  - https://github.com/satyensingh/SpringBootDemosVer2

- API Naming Convention

API Naming Convention

Assignment

Messaging

Agenda

- Introduction to Messaging
- Messaging Types
- Role of Messaging in Service Communication
- RabbitMQ Introduction
- Exchanges
- Message flow in RabbitMQ
- Types of Exchanges
- RabbitMQ and Server Concepts
- Queue and Topic in RabbitMQ
- Building a Example using RabbitMQ and Spring Boot

Messaging

Synopsis

- RabbitMQ is a message-queueing software called a message broker or queue manager. Simply said; It is a software where queues can be defined, applications may connect to the queue and transfer a message onto it.

Objective

- You will learn how to send and receive messages to message broker queue which helps services to communicated with each other even when they are not available temporarily.

Messaging

Messaging Using RabbitMQ

https://www.cloudamqp.com/blog/2015-05-18-part1-rabbitmq-for-beginners-what-is-rabbitmq.html

https://spring.io/guides/gs/messaging-rabbitmq/

https://sivalabs.in/2018/02/springboot-messaging-rabbitmq/

Async Messaging

Basic Spring Boot Course

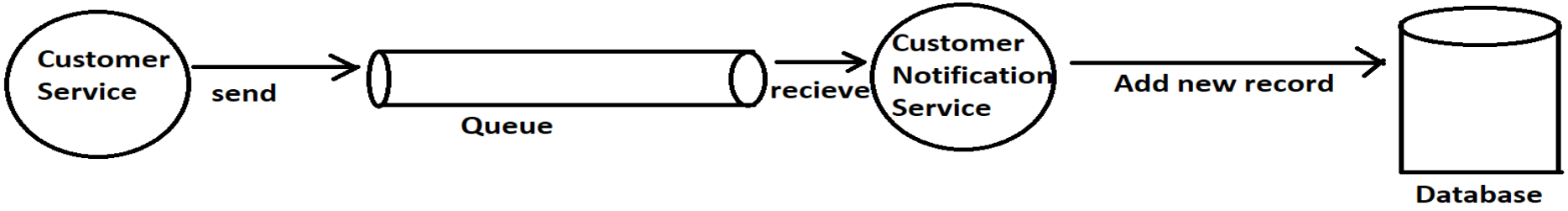
Topics	URLs
Spring Initializer	https://www.baeldung.com/spring-boot https://spring.io/guides https://www.springboottutorial.com/ https://github.com/spring-projects/spring-boot/tree/master/spring-boot-samples https://howtodoinjava.com/spring-boot2/ https://github.com/in28minutes/learn/blob/master/readme.md https://www.vogella.com/tutorials/SpringBoot/article.html https://www.javainuse.com/spring/sprboot https://www.youtube.com/user/koushks/playlists?view=1&flow=grid https://www.youtube.com/watch?v=jDchAEHIht0 https://www.youtube.com/watch?v=s9vt6UJiHg4
Spring Boot Starters	
MVC with Spring Boot	
Spring Boot Annotations	
Spring Boot Auto Configurations	
Spring Boot Actuator	
Properties File handling	
Testing with Spring Boot	
Spring Data JPA	
While Label Error Page	
Spring Boot Tomcat	
Spring Boot JDBC	
Basic Authentication in Spring Boot	
Controller Advice in Spring Boot	

Advanced Spring Boot Course

Topics	URLs
SpringBootServlet Initializer	https://www.baeldung.com/spring-boot https://spring.io/guides https://www.springboottutorial.com/ https://github.com/spring-projects/spring-boot/tree/master/spring-boot-samples https://howtodoinjava.com/spring-boot2/ https://github.com/in28minutes/learn/blob/master/readme.md https://www.vogella.com/tutorials/SpringBoot/article.html https://www.javainuse.com/spring/sprboot https://www.youtube.com/user/koushks/playlists?view=1&flow=grid https://developer.okta.com/blog/2019/03/12/oauth2-spring-security-guide https://developer.okta.com/blog/2019/05/02/spring-boot-single-sign-on-oauth-2
SpringBoot Filters	
Migrating From Spring To Spring boot	
Spring Boot CLI	
Custom Logging in Spring Boot	
Caching (Hazelcast/Redis) with Spring Boot	
Spring Boot + Rabbit MQ	
Spring Boot + Swagger UI	
Spring Boot Profiles	
Spring Boot Batch and Task Scheduler	
Internationalization with Spring Boot	
Serialization and Deserialization in Spring Boot	
Spring Boot Custom Starters	
Spring Boot Custom Auto Configurations	
Spring Boot Security Auto Configurations	
Spring Boot OAuth	
Spring Boot OAuth 2with OKTA	
Spring boot Admin UI	
Spring Boot Dev Tools	
Spring Boot Gradle Plugin	
Spring Boot Application as Service	
Dockerizing Spring Boot App	

Assignment

Create two services, “Customer” Service and “CustomerNotification” Service. Customer service will expose API for adding and viewing customers. CustomerNotification will add the newly added customer to database. Write an Application with these two services, whenever a new customer is added, customer service will send the message through queue to customernotification service, which will add the customer data in the database.



Microservice Architecture

Agenda

- Problem With Monolithic Architecture
- Microservice Concept – A Solution
- Microservice Architecture Principle
- 12 Factor App

Microservice Architecture

Synopsis

- Microservices is an architecture style and an approach for software development to satisfy modern business demands. Microservices are not invented, it is more of an evolution from the previous architecture styles.
- Microservices is one of the increasingly popular architecture patterns next to Service Oriented Architecture (SOA), complemented by DevOps and Cloud. Its evolution has been greatly influenced by the disruptive digital innovation trends in modern business and the evolution of technology in the last few years.

Objective

- You will learn the concepts and Principles of Microservice pattern

Microservice Architecture

Reference Material

- https://martinfowler.com/articles/microservices.html
- https://www.youtube.com/watch?v=wgdBVIX9ifA
- https://microservices.io/
- https://12factor.net/

Building Microservices Using Spring Boot and Spring Cloud

Agenda

- Spring Cloud Introduction
- Tools/Projects/Products Under Spring Cloud
- Service Discovery and Registration
- Managing Configuration
- Handling Failure and Latency
- API Gateway

Building Microservices Using Spring Boot and Spring Cloud

Synopsis

- The Spring Cloud project is an umbrella project from the Spring team, which implements a set of common patterns required by distributed systems as a set of easy-to use Java Spring libraries. Despite its name, Spring Cloud by itself is not a cloud solution. Rather, it provides a number of capabilities, which are essential when developing applications targeting cloud deployments that adhere to the Twelve-Factor Application principles. By using Spring Cloud, developers just need to focus on building business capabilities using Spring Boot, and leverage the distributed, fault-tolerant and self-healing capabilities available out-of-the-box from Spring Cloud.

Objective

- This chapter will provide a deep insight into the various components of the Spring Cloud project, such as Eureka, Zuul, Ribbon, Hystrix and Spring Config, by positioning them against the microservices capability model.

Building Microservices Using Spring Boot and Spring Cloud

Reference Material

- http://spring.io/projects/spring-cloud

Cloud Native

Spring Cloud Course

Topics	URLs
Spring Cloud – Config Server	https://www.baeldung.com/spring-cloud-tutorial https://www.baeldung.com/category/spring/spring-cloud/ https://www.javainuse.com/spring/springcloud https://howtodoinjava.com/spring-cloud/ https://spring.io/projects/spring-cloud
Spring Cloud – Config Server Using GIT	
Spring Cloud – Netflix Service Discovery	
Spring Cloud – Consul Service Discovery	
Spring Cloud – Hystrix Circuit Breaker	
Spring Cloud – Netflix Feign	
Spring Cloud – Zuul API Gateway	
Spring Cloud – Zipkin and Sleuth	
Spring Cloud – Ribbon	

Some Advanced Topics to Explore

Scalable Microservices with Kubernetes - https://www.udacity.com/course/scalable-microservices-with-kubernetes--ud615

Build Spring Microservices and Dockerize Them for Production - https://developer.okta.com/blog/2019/02/28/spring-microservices-docker

Spring Cloud Stream With Kafka - https://dzone.com/articles/spring-cloud-stream-with-kafka

Kafka Streams and Spring Cloud Stream - https://spring.io/blog/2018/04/19/kafka-streams-and-spring-cloud-stream

Getting Started with GraphQL and Spring Boot - https://www.baeldung.com/spring-graphql

A Quick Guide to Using Keycloak with Spring Boot - https://www.baeldung.com/spring-boot-keycloak

CI/CD for SpringBoot applications using Travis-CI - https://sivalabs.in/2018/01/ci-cd-springboot-applications-using-travis-ci

Build CD Pipeline With Gitlab + GitLab CI for Spring Cloud Microservices - http://stytex.de/blog/2016/09/22/build-cd-pipeline-with-gitlab-gitlab-ci

Assignment

Order Processing System

Containerizing Microservices

Agenda

- What are Containers?
- Difference between VMs and Containers
- Difference between VMs and Containers
- Microservices and Containers
- Introduction to Docker
- Deploying Microservices into Docker
- Running RabbitMQ on Docker
- Using Docker registry
- Publishing Microservices to Docker Hub

Containerizing Microservices

Synopsis

- Containers are not revolutionary groundbreaking concepts. It has been in action for quite a while. However, the world is witnessing the reentry of containers, mainly due to the wide adoption of cloud computing.
- In the context of microservices, containerized deployment is like the icing on the cake. It helps microservices to be further autonomous by self-containing the underlying infrastructure, thereby making microservices cloud-neutral.

Objective

- You will learn the concepts and relevance of virtual machine images and containerized deployments of microservices on Container registry like Docker Hub.

Containerizing Microservices

Reference Material

https://app.pluralsight.com/library/courses/docker-getting-started/table-of-contents

https://docs.docker.com/get-started/

https://docker-curriculum.com/

https://github.com/wsargent/docker-cheat-sheet

https://github.com/fabric8io/docker-maven-plugin
https://www.youtube.com/playlist?list=PLsyeobzWxl7pYWSoIhLHFKPzEky7DiGoN

Containers

Assignment

Docker
Assignments

Microservice Deployment on Public Cloud (AWS)

Agenda

- Cloud Computing Introduction
- Cloud Types
- Different Cloud Model
- Installing Docker on AWS EC2 instance
- Running services on EC2

Microservice Deployment on Public Cloud (AWS)

Objective

- You will understand how to deploy your services on EC2 instance on AWS and running it from the same.

Microservice Deployment on Public Cloud (AWS)

Reference material

- https://aws.amazon.com/getting-started/tutorials/deploy-docker-containers/
- https://www.youtube.com/playlist?list=PLsyeobzWxl7pYWSoIhLHFKPzEky7DiGoN

Accomplish the below things while developing final Project/Case Study

- You have to build UI of an application by using Angular.
- For business functionalities, you have to create spring boot microservices. Each microservice should expose REST/JSON endpoints for accessing business capabilities.
- Each microservice stores its own persistent data using database (select the database as per the use case/requirement).
- Design appropriate database schema (if any).
- You have to write JUnit test cases for testing all functionalities of each microservice with 70-80% of code coverage.
- You have to write test cases for every component, service and pipe of your Angular application by using Jasmine and Karma testing frameworks.
- You have to register all your microservices with Eureka service.
- You should implement Zull API gateway who's API is exposed to Angular application.
- You have to use RabbitMQ for external messaging.
- You have to use of SonarQube for continuous code quality inspection.
- You have to use Jenkins for continuous integration.
- Deploy your microservices by using Docker.

Project : Case Study (choose one)

Flight Booking System

Shopping Cart

Online Railway Reservation System

Online Hotel Management System

Thank you for your Participation

kindly reach satyendra.singh@capgemini.com for queries.