Introduction to MicroServices with Spring Boot

Service Oriented Architecture (SOA) defines data sharing between the applications in language and platform independent applications as services.
SOAP Web services support communication and data sharing across the applications in a language neutral way suing xml as content language.
REST architecture is the new way of building service oriented applications on the internet, which becomes platform independent and enable the data sharing in defined formats. REST services work faster and flexible compared to SOAP services.

MicroServices is a variant of the service-oriented architecture (SOA) architectural style that defines the application as a composition of loosely coupled services.
In MicroServices architecture the big application is decomposed into different smaller and isolated service units which improves modularity and makes the application easier to understand, develop and test. It enables isolated parallel development by small distributed teams in develop, deploy and scaling the respective service units independently. MicroServices are flexible in scaling and load balancing in the deployments.
Varieties of patterns are specified for MicroServices which further simplifies MicroServices deployments and data sharing.

Spring Boot is a higher level Spring environment which simplifies build and deployments of Spring based applications and service applications.
Spring Boot supports Spring Data, JPA, Web, Web Services, JDBC, ORM Integration and security flavors to Spring application with minimum programming and configuration and supports application health monitoring, diagnosis and a host of other features that make building, deploying, and running the Spring applications much easier.

Netflix has a framework and tools open-sourced to support the MicroServices and are implemented as Spring-based tools under the umbrella of the Spring Cloud project. These tools address the concerns related to MicroServices.
Deploying the MicroServices in Docker containers further simplifies the deployment and flexibility in scaling and load balancing the services.
Managing Docker containers with Kubernetes cluster simplifies the container orchestration and management.

Course Objectives

This course imparts practical introduction to software developers on MicroService applications with Spring Boot framework, Netflix tools and Docker containers with Kubernetes environment.

The attendees:

- Understand the basics of MicroServices architecture
- Understand the basics of Spring Boot Framework for MicroServices.
- Understand using the Spring Boot Command Line Interface (CLI)
- Build and deploy MicroService applications with Spring Boot
- Utilize the Netflix components in MicroService deployments.
- Understand the MicroService deployments in Docker containers with Kubernetes cluster environment.

Prerequisites

The participants must be well versed in java EE web application development and basic understanding about Apache Maven, Spring Framework, NO-SQL Database, and JPA with exposure to Soap and REST services is mandatory.
The participants lacking these skills reduce the pace of class and impact others learning speed.
Maximum 20 participants matching the pre-requisites as above should only attend this course.

Training Methodology

Each topic is supplemented with practical demonstrations and exercises for the participants.
The theoretical topics are discussed interactively and technical details are discussed with practical demonstrations and followed by the participants. The participants work on the hands on exercises which strengthen the concepts learned.
The training material including documents, presentations with demo exercises and sample case studies are shared with the participants in soft format.

Software Installations

Instead of different environments and network settings with each participant machine, it is advised to have common environment for all of them with virtual machines having pre-defined setup as below.
The Intel dual core compatible CPU with minimum 8GB RAM and 500 GB HDD with **Linux (Centos/Ubuntu) 64** bit Adobe Acrobat reader with, JDK1.8 64 bit, 7Zip, MongoDB server and client latest versions, latest version installed. The Zip distributions   for Eclipse-Jee-Photon 64 bit, Apache Maven 3.6, Spring Tool Suite 4.4 and Apache Tomcat 8.x Web server 64 bit to be made available.

The participants must have admin rights on their systems.
Live internet connection with reasonable speed and download permissions is required to download the dependencies and plug-ins.

Training duration: Five days.

Instructor: Prakash Badhe.

Course Plan

Day1
SOA and Web services Review

- SOA overview.
- The need of SOA in the Enterprise
- Applications as services
- Soap Web services with XML
- SOAP,WSDL and dependencies
- SOAP services limitations.
    - Interoperability issues
    - Heavyweight architecture
    - Performance and integration issues.
    - Porting/migration issues
- REST Services overview
- Soap vs. REST

Spring Boot Introduction

- Spring Boot high level features
- The Spring Boot CLI
- Spring Initializr configuration
- The application properties configuration
- Create and design applications with Spring Boot
- Build and deploy with Apache Maven

The Spring Boot REST services are introduced interactively and practically demonstrated with create, build, deploy, consume and monitoring stages.

Spring Boot REST Service applications

- Spring Boot REST service application components and dependencies
- Integration with Spring Data JPA and No-SQL database MongoDB
- Configure Persistent Entities
- Understand request and response processing
- CRUD operations.
- Build and deploy war
- Consume the REST API from client side
- Integrate with Actuator for health monitoring
- Integrate with Swagger for documenting REST API
- Service deployment and monitoring
- Implement Security with O-Auth

Day2

MicroService Architecture

The participants are introduced with MicroServices architecture and case study implementation is demonstrated with Spring Boot framework.

MicroServices Introduction

- Monolith SOA applications pros and cons
- MicroService Architecture
- MicroService pros and cons
- Monolith vs. MicroServices
- Demo-Decomposing big fat application into smaller service units
- Features
  - Loose coupling
  - Isolated development and isolated testing
  - Performance enhancement and maintainability
  - Scaling the services with ease

The MicroServices patterns of implementation are demonstrated with examples.

Micro-Services Patterns and implementation

- Decompose the monolith into MicroServices
  - Decompose by Business Capabilities
  - Decompose by Domain
- Session façade as front end service
- Aggregation of services
- Split into services
- Shared Database
- The observable
- CQRS : Command Query Responsibility Segregation
- UI Compositor pattern

Day3

The Twelve-Factor App methodology

- One codebase
- Dependencies isolated

- Backing services
- Externalizing configuration with environment

Deployment Scenarios overview

- Scaling the services
- Service instance per host
- Multiple service instances per host
- Shared Database
- Load balancing
- Deployment in container
- Deployments in Cloud

Deployment patterns with Netflix and Spring Cloud

- API Gateway : Zuul
    - Proxy with load balancing
    - Security
    - Routing
- Load balancing with Ribbon
- Circuit Breaker with Hystrix
    - Protect from fault
    - Fallback
- Service Discovery and registration
    - Loose coupling of service Node
    - Using Eureka service discovery with registration
- Tracing with Sleuth and Zipkin
- Saga :Failure management and Transactions
- Externalizing configuration : Spring Cloud Config Server
- Blue Green Pattern of deployment
- Deploy with Consul service Mesh
- Lagom MicroService framework overview

Day4

Micro-Services communication

- Messaging protocols and usage
- Message brokers
- Point to Point
- Publish – Subscribe
- Synchronous vs. Asynchronous
- Event Sourcing and collaboration

- Implement messaging with Apache Kafka

MicroServices in practice
- Best Practices for MicroServices
- When to use (Not to Use) MicroServices
- Common Mistakes with respect to MicroServices

Day5

The participants are introduced to Docker container architecture and case study deployment and scaling is demonstrated for Spring Boot MicroServices in the Docker containers managed with Kubernetes cluster environment.

Container architecture
- Deployment in Virtual Machine environment
- The VM limitations
- Why Containers?
  - Development and Deployment platform
  - Manage deployment across platforms
  - Application dependencies and scaling
  - Load balancing and high availability
  - Containerized applications
  - Container vs. VM

Introduction to Docker
- Docker container architecture
- Docker commands overview
- Create and run the first Container instance
- Pulling images from Docker Hub
- Define your own image with Dockerfile for MicroService application
- Build and push the image to Docker Hub for sharing
- Docker networking
- Shared Volume for state management
- Docker compose usage

Kubernetes Introduction

- Container orchestration
- Kubernetes architecture
- Kubernetes with Docker Containers
- Pod,Label,Replication,Node and Cluster
- Controller,Service,Deployments and Namespace
- Service discovery

- Start the Cluster with Minikube/DinD
- Kubectl commands
- Deploy MicroServices in Kubernetes cluster
- Manage and monitor with Dashboard
- Monitoring with Logs
- Kubernetes as service from Amazon EC2(EKS) overview

****