Trainer’s name: [Banu] [Prakash]

|  |
| --- |
| **Build Secure RESTful Webservices using Spring Boot** |

**Program objective:**

**Pre-requisite:**

|  |
| --- |
| **Participants should have good knowledge of Java, JPA and Spring** |

|  |
| --- |
| **3 Days** |

**Duration of program**

**H/w – S/w required for Lab Setup**

|  |  |
| --- | --- |
| Type of hardware | List of software |
| **NA** | **Operating System: Windows/MAC/ Linux/ Unix**  **Web Browsers: Chrome**  **Editor: Eclipse for enterprise edition – 2020**  **Docker**  **MySQL 5 Community edition** |

**Target Audience**

|  |
| --- |
| **Use Spring boot to build Secure RESTful web service with relational database and build micro services, then this course covers them all.** |

* REST
* What is REST
* HTTP Methods and Status Codes
  + GET, POST, PUT, PATCH, DELETE
  + PUT vs PATCH vs JSON PATCH
* Richardson Maturity Model
* Resources and URI
  + REST Resource naming conventions
  + Use Nouns to represent resources
    - Document
    - Collection
    - Store [ client managed resource repository]
    - Controller [ Procedural concepts like checkout, play]
  + Nested Resources
    - Why?
    - Why not?
  + Good practices in resource naming conventions
    - Use forward slash to indicate hierarchical relationships
    - Do not use trailing slash in URIs
    - Use hyphens (-) to improve readability of URIs
    - Avoid using underscore (\_)
    - Use lower case letters in URIs
    - Do not use CRUD functions in URIs
    - Use query component to filter URI collection
  + Fine grained CRUD resources versus Coarse Grained resources
  + Pagination, Filtering and Sorting
  + Versioning
* Guiding Principles of REST
  + Client-server
  + Stateless
  + Cacheable
  + Uniform Interface
  + Layered System
* **Spring Boot introduction**
  + Introduction to Spring Boot
  + Bootstrapping Spring boot application
  + Starting Annotations of Spring Boot
    - @SpringBootApplication
    - @Repository
    - @Service
    - @Configuration
    - @Bean
  + Spring Profiles
* **Spring Data JPA**
  + - Spring data JPA CRUD
      * Repository, CrudRepository and JpaRepository
      * Query methods
        + Query generation using method name

find…By, read…By, query…By, count…By, and get…By.

* + - * + Using @Query annotation
        + Named Queries
      * Modifying queries
      * Query Hints
      * Using SpEL Expressions
        + Using #{#entityName}
        + Accessing arguments
        + Using wildcard
        + Sanitizing input values
      * Projections
        + Closed and open projections
        + Projection interface using default method for custom logic
        + Dynamic projections
      * Pagination and Sorting
        + PagingAndSortingRepository
        + Pageable, Page, Sort and PageRequest
      * Passing method parameters to Query methods
      * Specifications
        + JpaSpecificationExecutor
      * Query by Example
        + Example
        + Example Matchers
      * JPA Entity graphs
        + Benefits of using Entity Graphs
        + @EntityGraph
        + @NamedEntityGraph
      * Spring Data — Envers
        + @Audited
        + @AuditTable
        + RevisionRepository
        + Enable EnversRevisionRepositoryFactoryBean
        + Querying data
    - Unit testing
      * Spring data jpa
  + Caching
    - Register cache engine with spring boot
    - Enable caching on spring managed beans
    - @EnableChaching
    - @Cachable
    - @Caching
      * @CacheEvict
* Spring Boot Actuator
  + - Monitoring
    - Built-in production ready endpoints
    - Monitoring with Prometheus and Grafana
* RESTful Web Services using Spring BOOT
  + Creating REST controllers
    - @RestController
    - @RequestMapping
    - CRUD Mapping: @GetMapping(), @PostMapping(), @PutMapping(), etc
    - @ResponseBody and @RequestBody
    - @ResponseStatus
    - Exception Handling
      * Using Own exception: customize exception handling
      * @ControllerAdvice and @ExceptionHandler
  + Using POSTMAN
  + RestTemplate vs WebClient
  + Content Negotiation
    - Content Negotiation: XML/JSON or your own content type
    - Custom Message Convertor
    - Creating your own content-type
  + Using Swagger to document RESTful web services

* Spring Boot: Unit Testing Rest Services
  + Unit testing the GET Rest Service
    - @RunsWith: using SpringJUnit4ClassRunner to launch Spring TestContext Framework
    - @WebMvcTest: Unit testing Spring MVC Framework
    - MockMvc and MockBean: adding mocks to Spring ApplicationContext
    - Mockito: Mocking behaviour
* Spring HATEOAS
  + Building Hypermedia driven RESTful web service
    - RepresentationModel
    - Creating links using Link object
    - WebMvcLinkBuilder – which simplifies building URIs
      * The linkTo() method
      * The slash() method
      * The withSelfMethod()
    - Relations
    - HAL
      * Hyperlink between resources in API
      * HAL Browser
* Spring Data REST
  + - Build hypermedia-driven REST web services with spring data repository
* Spring Security
  + Spring Security Architecture
  + Getting Spring Security
  + Web Security Java Configuration
    - @EnableWebSecurity
    - WebSecurityConfigurerAdapter
    - UserDetails and UserDetailService
    - InMemoryUserDetailsManager
    - AuthenticationProvider
      * DaoAuthenticationProvider
        + Custom UserDetails, UserDetailService and UserRepository
        + NoOpPasswordEncoder and BCryptPasswordEncoder
    - AuthenticationManagerBuilder
      * inMemoryAuthentication
      * jdbcAuthentication
    - HttpSecurity
      * CSRF
      * formLogin(), loginPage()
      * Authorize Requests
        + antMatchers() , anyRequest()
        + permitAll() and authenticated()
        + hasRole()
      * logout
        + Invalidate HttpSession and Clear authentication info
        + logoutRequestMatcher [ AntPathRequestMatcher ]
        + logoutSuccessUrl
      * SPRING\_SECURITY\_LAST\_EXCEPTION
    - Method Security
      * EnableGlobalMethodSecurity
        + @EnableGlobalMethodSecurity(prePostEnabled = true)
        + @PreAuthorize and @PostAuthorize
      * GlobalMethodSecurityConfiguration
        + MethodSecurityExpressionHander
        + Custom Permission Evaluator
        + JWT
* Building Microservices with Spring Boot **Introduction** with example
  + Spring Cloud and Netflix OSS
    - Service Discovery using Eureka
      * Registering with Discovery Service
    - HTTP clients with Feign
    - Api Gateway
* Protocol Buffers
  + What and Why?
  + Define message structures with .proto files
  + Protocol buffers with Java
  + Protobuf in Spring REST API