Asynchronous, Non-Blocking, Event Driven, Reactive programming With Eclipse Vert.x for Polyglot developers

Java, JavaScript, Groovy, Ruby, Ceylon, Scala **and** Kotlin

Prerequisite:

1. Developer may have knowledge on Java and Java 8 functional Programming concepts and implementation.
2. Developer must have knowledge on JEE Technology
3. Developer must have strong working knowledge on Rest full Web Services.
4. Developer must have idea on build systems such as maven and Gradle
5. Developer must have Knowledge on Docker basics, Kubernetes, Red hat Open Shift

Duration 5 days

Day 01

Java Functional Programming Concepts and Implementation

- What is functional Programming
- Functional Programming concepts
- Functional interfaces
- Lambda functions
- Pure Functions and overview on java streams

Enter into Reactive Programming and Rxjava

- Reactive Programming
- What is Reactive?
- Reactive Programming is Paradigm or Architecture
- Imperative Style
- Declarative Style
- asynchronous
- synchronous
- RxJava Lab
- Observable

- Streams
- Reactive Programming Design Patterns
- Iterator
- Observer
- Styles of Programming
- CREATE
- Easily create event streams or data streams.
- COMBINE
- Compose and transform streams with query-like operators.
- LISTEN
- Subscribe to any observable stream to perform side effects.
- Reactive Systems
- Reactive Programming Standards via Reactive Manifesto
- Properties of Reactive Systems
- Elasticity
- Resilience
- Reactive Streams
- Reactive Streams with NonBlocking Back pressure

Reactive Programming and Asynchronous Programming

- Being distributed and networked is the norm
- Blocking Apis
- Blocking Api waste resource , increase cost
- Asynchronous programming with non-blocking I/O
- Multiplexing event-driven processing: the case of the event loop
- What is a reactive system?
- What else does reactive mean?

Vertx Introduction

- What is Vertx
- Vert.x Project
- Vertx Application

Alternatives to Vert.x

- Java nio
- Netty and Apache Mina
- Spring WebFlux

Preparing Vertx Application

- Setup Vertx
- Introduction to Build Systems
- Maven
- Gradle
- Setup Vertx Application using Gradle

Vertx Projects Overview

- How Vertx Projects are working
- Vertx Core Project
- Vertx Extension Projects

Vertx Core Project

- Vertx Core Project Provides low level Services
- Vertx Core for building basic Non-Blocking Applications
- Non-Blocking ,Async Core Apis
    i. *Writing TCP clients and servers*
    ii. *Writing HTTP clients and servers including support for Web Sockets*
    iii. *The Event bus*
    iv. *Shared data - local maps and clustered distributed maps*
    v. *Periodic and delayed actions*
    vi. *Deploying and undeploying Verticles*
    vii. *Datagram Sockets*
    viii. *DNS client*
    ix. *File system access*
    x. *High availability*
    xi. *Native transports*
    xii. *Clustering*

Day 02

Vertx Core API- io.vertx.core

io.vertx.core.Vertx

The entry point into the Vert.x Core API

- Creating TCP clients and servers

- Creating HTTP clients and servers
- Creating DNS clients
- Creating Datagram sockets
- Setting and cancelling periodic and one-shot timers
- Getting a reference to the event bus API
- Getting a reference to the file system API
- Getting a reference to the shared data API
- Deploying and undeploying verticles

Creating Vertx Object- Factory Apis

- vertx(),
- vertx(io.vertx.core.VertxOptions)
- clusteredVertx (io.vertx.core.VertxOptions, Handler)

Vertx Core Principles

- Are you fluent?.
- Don't call us, we'll call you.
- Don't block me!
- Reactor and Multi-Reactor
- The Golden Rule - Don't Block the Event Loop
- Running blocking code
- Async coordination

Verticles

- Verticles
- Writing Verticles
- Asynchronous Verticle start and stop
- Verticle Types
  - Standard Verticles
  - Worker Verticles
  - Multi-threaded worker verticles
- Deploying verticles programmatically
- Rules for mapping a verticle name to a verticle factory
- How are Verticle Factories located?
- Waiting for deployment to complete
- Undeploying verticle deployments
- Specifying number of verticle instances
- Passing configuration to a verticle
- Accessing environment variables in a Verticle
- Verticle Isolation Groups

- High Availability
- Running Verticles from the command line
- Causing Vert.x to exit
- The Context object
- Executing periodic and delayed actions

The Event Bus

- The Theory
- Addressing
- Handlers
- Publish / subscribe messaging
- Point-to-point and Request-Response messaging
- Types of messages
- The Event Bus API
- Registering Handlers
- Un-registering Handlers
- Publishing messages
- Sending messages
- Setting headers on messages
- Message ordering
- The Message object
- Acknowledging messages / sending replies
- Sending with timeouts
- Send Failures
- Message Codecs
- Clustered Event Bus
- Clustering programmatically
- Configuring the event bus

Day 03

Message Formats

JSON objects

- Creating JSON objects
- Putting entries into a JSON object
- Putting entries into a JSON object
- Mapping between JSON objects and Java objects
- Encoding a JSON object to a String

- JSON arrays
- Creating JSON arrays
- Adding entries into a JSON array
- Getting values from a JSON array

Buffers

- Creating buffers
- Creating buffers
- Appending to a Buffer
- Random access buffer writes
- Reading from a Buffer

Non-Blocking, Async Network Programming

Writing TCP servers and clients

- Creating a TCP server
- Configuring a TCP server
- Start the Server Listening
- Listening on a random port
- Getting notified of incoming connections
- Reading data from the socket
- Reading data from the socket

Writing HTTP servers and clients

- Creating an HTTP Server
- Configuring an HTTP server
- Configuring an HTTP/2 server
- Logging network server activity
- Start the Server Listening
- Getting notified of incoming requests
- Handling requests
- Reading Data from the Request Body
- Sending back responses
- Setting status code and message
- Closing the underlying connection

Vert.x Modules-Web

- **Vert.x Web Sub Modules**

- Web Core
- Web Client
- Web Api Contract
- Web Service Api

Some of the key features of Vert.x-Web include:

- Routing (based on method, path, etc)
- Regular expression pattern matching for paths
- Extraction of parameters from paths
- Content negotiation
- Request body handling
- Body size limits
- Cookie parsing and handling
- Multipart forms
- Multipart file uploads
- Sub routers
- Session support - both local (for sticky sessions) and clustered (for non sticky)
- CORS (Cross Origin Resource Sharing) support
- Error page handler
- Basic Authentication
- Redirect based authentication
- Authorisation handlers
- JWT based authorization
- User/role/permission authorisation
- Favicon handling
- Template support for server side rendering, including support for the following template engines out of the box:
    1. *Handlebars*
    2. *Jade,*
    3. *MVEL*
    4. *Thymeleaf*
    5. *Apache FreeMarker*
    6. *Pebble*
- Response time handler
- Static file serving, including caching logic and directory listing.
- Request timeout support
- SockJS support
- Event-bus bridge
- CSRF Cross Site Request Forgery
- VirtualHost

Day 04

Vert.x Modules-Data Access-Java

- **Vert.x Data Access Sub Modules**
  - Mongo Client
  - JDBC Client
  - SQL Common
  - Redis Client
  - MySQL / PostgreSQL client
  -
- **JDBC Client**
  - Creating a the client
  - Closing the client
  - Getting a connection
  - Configuration
  - JDBC Drivers
  - Data types
- **MonogoDB**
  - Using Vert.x MonogoDB Client
  - Creating a client
  - Using the API
  - Configuring the client
  - RxJava 2 API

Day 05

Micro services

Vert.x offers various component to build micro service-based applications.

Vert.x Service Discovery

This component lets you publish, lookup and bind to any type of services.

- Using the service discovery
- Overall concepts
- Creating a service discovery instance
- Publishing services

- Withdrawing services
- Looking for services
- Retrieving a service reference
- Types of services
- Listening for service arrivals and departures
- Listening for service usage
- Service discovery bridges
- Additional bridges
- Additional backends
- This component provides an infrastructure to publish and discover various resources, such as service proxies, HTTP endpoints, data

Vert.x Circuit Breaker

- Vert.x Circuit Breaker
- Using the vert.x circuit breaker
- Using the circuit breaker
- Retries
- Callbacks
- Event bus notification
- The half-open state
- Reported exceptions
- Pushing circuit breaker metrics to the Hystrix Dashboard
- Using Netflix Hystrix

Vert.x Config

- Concepts
- Using the Config Retriever
- Overloading rules
- Using the retrieve configuration
- Available configuration stores
- Listening for configuration changes
- Retrieving the last retrieved configuration
- Reading configuration as a stream
- Processing the configuration
- Retrieving the configuration as a Future
- Extending the Config Retriever
- Additional formats
- Additional stores

Vertx unit

- Asynchronous polyglot unit testing.
- Introduction
- Writing a test suite
- Asserting
- Asynchronous testing
- Asynchronous assertions
- Repeating test
- Sharing objects
- Running
- Reporting
- Vertx integration
- Junit integration
- Java language integration

Vertx deployments on docker, Kubernetes, open shift

- Dockizing vertx applications
- Dockerfile creations for vertx
- Creating a Image
- Docker compose
- Docker Machine
- Deployment on kubernetes and red hat open shift