Reactive Micro Services 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

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

Day 02

Vertx Projects Overview

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

Futures

Vertx Core Project

Vertx Core Project Provides low level Services
Vertx Core for building basic Non-Blocking Applications Non-Blocking ,Async Core Apis
The Event bus
Shared data - local maps and clustered distributed maps Periodic and delayed actions
Deploying and undeploying Verticles
File system access

Vertx Core API- io.vertx.core

io.vertx.core.Vertx

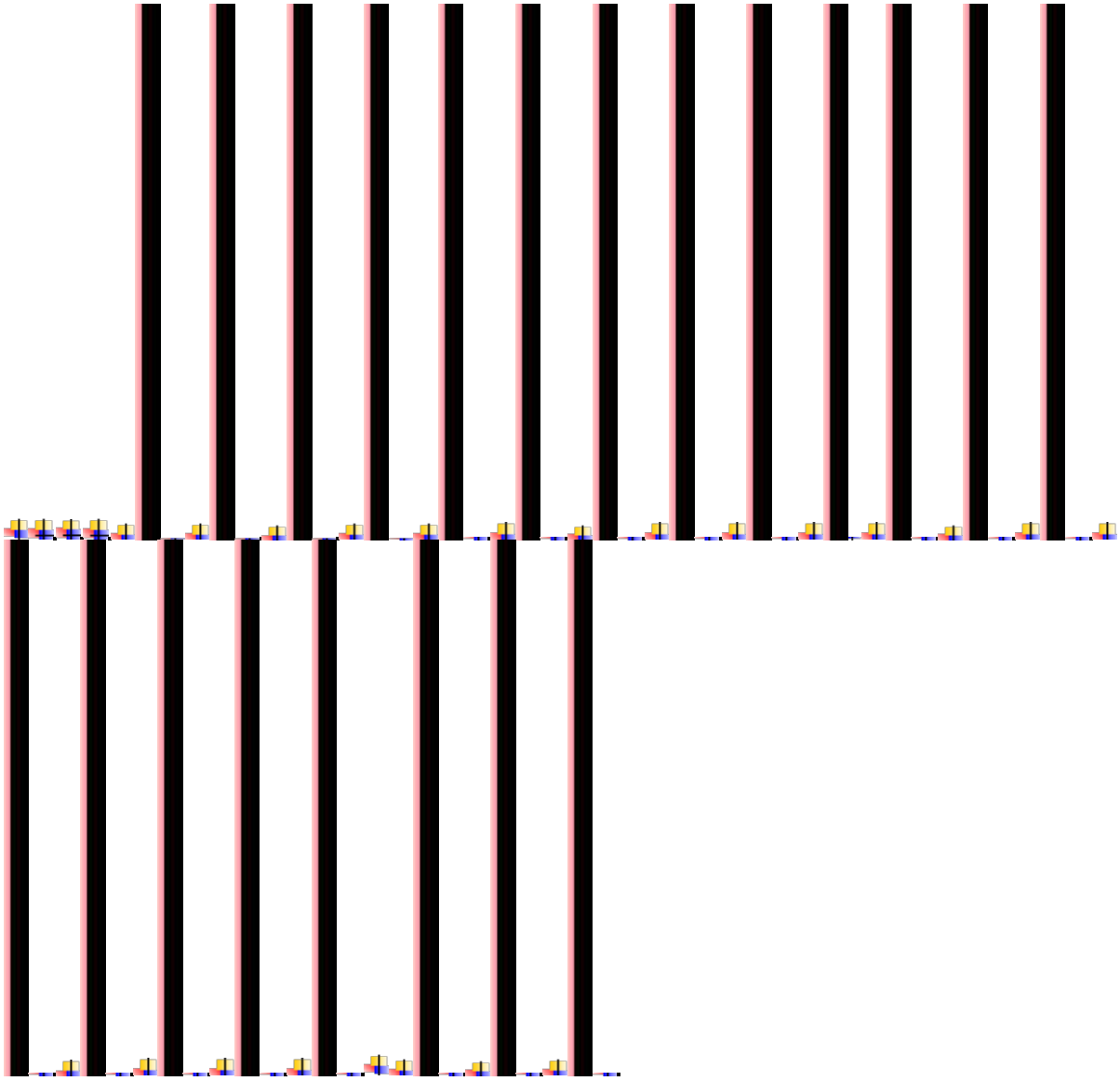
Creating Vertx Object- Factory Apis

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

o Standard Verticles
o Worker Verticles
o Multi-threaded worker verticles

Deploying verticles programmatically
Rules for mapping a verticle name to a verticle factory Waiting for deployment to complete
Undeploying verticle deployments
Specifying number of verticle instances

Passing configuration to a verticle
High Availability
Causing Vert.x to exit

Day 03

Micro Services implementation using Vertx

Microservices Architecture Overview
Patterns
Why Vertx is choice for Microservices
Reactive vs Future style Microservice Implementations

Non-Blocking, Async Network Programming

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

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
Routing (based on method, path, etc)
Regular expression pattern matching for paths

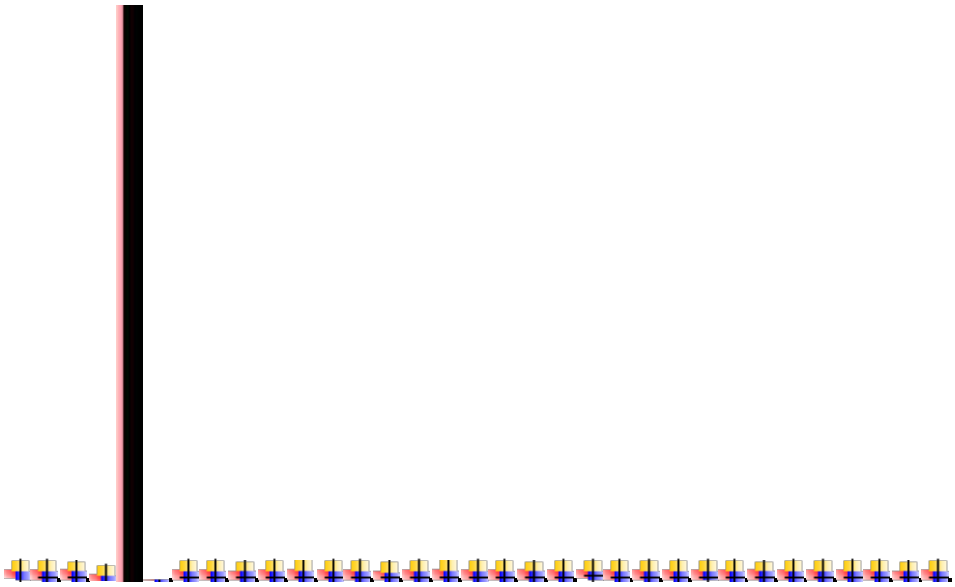
Content negotiation
Request body handling
Body size limits
Multipart file uploads
Sub routers
Error page handler
Favicon handling
Template support for server side rendering, including support for the following template engines out of the box:

Response time handler
Static file serving, including caching logic and directory listing.

The Event Bus -Service Communications Pattern through Event Sourcing Pattern

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



Event bus communication over distributed systems

How to send messages over tcp bridge Sending messages to browsers via sockjs

Buffers

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

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

Day 04 & 05

Micro services Modules

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

Reactive Programming using Rxjava && Reactive Microservice implementation

Reactive Programming using RxJava Observables
Streams
Operators

Hot and Cold Streams
Backpressure
Vertx with Reactive apis
Micro services implementations using Rxjava.