SQUER

Micronaut - Quid est?

Brainfood Friday, 01 March 2024

Agenda

- **Introduction to Micronaut**

  What, Why and How

- **Key Features and Differentiators**

  High performance, natively cloud-native, GraalVM support

- **Ecosystem and Community**

  Documentation, community support, library and tool landscape

- **Hands-on Session**

01

Introduction to Micronaut

INTRODUCTION TO MICRONAUT

What is Micronaut?

A MODERN, JVM-BASED, FULL-STACK FRAMEWORK FOR BUILDING MODULAR, EASILY TESTABLE MICROSERVICE AND SERVERLESS APPLICATIONS

MICRONAUT®

- Modern JVM Framework

  Built for microservices and serverless applications

- Built for Speed and Efficiency

  Enables fast startup times, reduces memory footprint, improves runtime efficiency (compared to traditional JVM frameworks)

- Tailored for cloud-native Development

  Built-in support for common discovery services, distributed tracing tools and cloud runtimes

SQUER

INTRODUCTION TO MICRONAUT

What sets Micronaut apart?

- AOT Compilation

  To reduce startup times and memory footprint

- Reflection-free DI and AOP

  To reduce performance overhead and memory usage. Dynamic features require the JVM to perform additional processing and reflective method calls are slower than direct calls. Memory usage is increased as well as more classes and metadata are loaded and retained in memory.

- Natively cloud-native

  To simplify integration with cloud providers and tooling like service discovery, load-balancing and distributed tracing

- First-class reactive Support (RxJava, Reactor)

  To improve resource efficiency

MICRONAUT®

SQUER

02

Key Features and Differentiators

SQUER

KEY FEATURES AND DIFFERENTIATORS

JIT vs. AOT Compilation

JIT

With JIT compilation, the JVM can optimize performance based on actual real-time metrics, which allows it to adapt to actual user behavior. Since JIT happens in the JVM the write-once, run-anywhere promise is kept.

JIT compilation comes with some warm-up time, leading to poor performance on startup. It can also lead to inconsistent performance and generally increased memory consumption.

AOT

With AOT compilation, the code has already been compiled to native code, which leads to faster start-up times and reduced memory consumption. It also comes with predictable performance.

AOT compilation comes with higher upfront compilation costs (build-times) and less effective optimizations. AOT compiled code can also complicate the deployment across different platforms.

KEY FEATURES AND DIFFERENTIATORS

Micronaut and GraalVM

GraalVM is a universal VM that supports a polyglot runtime environment and the ability to compile Java applications to native machine code

Micronaut integration

Micronaut applications are well suited for use with GraalVM, because of the AOT compilation model.

Benefits of Native Images

Using GraalVM native images offers reduced startup times, lower runtime overhead and consistent performance.

This makes native images ideal for usage in microservice architectures (rapid scaling) or FaaS settings.

Considerations

Building a native image can be time- and resource intensive, potentially affecting the development and deployment cycles. Although core features of Micronaut don't rely on dynamic features like reflection, some other libraries might not be as compatible and require additional configuration effort.

SQUER

KEY FEATURES AND DIFFERENTIATORS

Natively cloud-native

MICRONAUT®

Micronaut comes with built-in cloud support including discovery services, distributed tracing and cloud runtimes.

Distributed Configuration

Micronaut features a robust system for externalizing and adapting configuration, even in distributed scenarios, by providing specific APIs. This allows easy integration with AWS Parameter Store, Hashicorp Vault or Kubernetes ConfigMaps and Secrets.

Service Discovery

Micronaut makes service discovery easy by integrating with tools and services like Eureka, AWS Route 53 and Kubernetes.

Client-Side Load Balancing

When using service discovery, Micronaut by default performs Round Robin client-side load balancing using the servers available after service discovery. The strategy used can be replaced and customized, e.g. using Netflix Ribbon.

SQUER

KEY FEATURES AND DIFFERENTIATORS

Natively cloud-native

Micronaut comes with built-in cloud support including discovery services, distributed tracing and cloud runtimes.

Distributed Tracing

Micronaut provides a tracing annotations library which easily integrates with tracing systems like Jaeger and Zipkin.

Serverless Functions

Traditional JVM frameworks are not well suited for usage with serverless functions. With its optimizations for fast startup times and low memory consumptions, Micronaut is a good candidate for FaaS systems like AWS Lambda and Google Cloud Functions and offers native support.

SQUER

03

Ecosystem and Community

SQUER

ECOSYSTEM AND COMMUNITY

Documentation and Community Support

Official Documentation

Micronaut provides official documentation and guides for all basic and advanced features, including sub-projects like Micronaut Data and Micronaut Security. This documentation gives a good introduction and starting point, but lacks depth, which will become annoying for more complex problems.

Community Support

Compared to well-established frameworks like Spring, the community is a lot smaller, so the number of community resources is still limited, which makes it a lot harder to find a solution when facing issues.

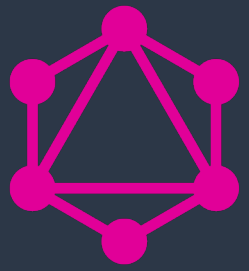
MICRONAUT®

ECOSYSTEM AND COMMUNITY

Library and Tool Landscape

Library and Integration Support

Micronaut already provides support for essential features and integrations and libraries for the most common use-cases are available. For less common or specialized use-cases, this support might still be lacking breadth and maturity.



Tooling and IDE Support

Micronaut integrates well with common IDEs but this is also still maturing. Some tools or libraries are still in early stages and don't offer the robustness known from more established frameworks like Spring.

SQUER

04

Hands-on Session

SQUER

HANDS-ON SESSION

Let's see it in Action!

Explore Essential Features

In this session we will start from scratch and build a Micronaut application that touches some of the common features. This includes project setup, exposing HTTP endpoints, accessing a relational database, introducing caching and integrating with Kafka.

Performance Benchmarking

Of course we don't trust the benchmarks online, so we want to crunch some numbers ourselves. In this session we will define the requirements for an application suitable for such benchmarks, implement this (simple) application using Micronaut and another framework of choice and compare some metrics.

To the Cloud

In this session we will test the *natively cloud-native* promise of Micronaut and experiment with service discovery, distributed configuration and serverless functions.

https://github.com/squer-solutions/micronaut-bfs

SQUER

Discussion

Thanks!

SQUER