

MICRONAUT

Aleck BILOUNGA



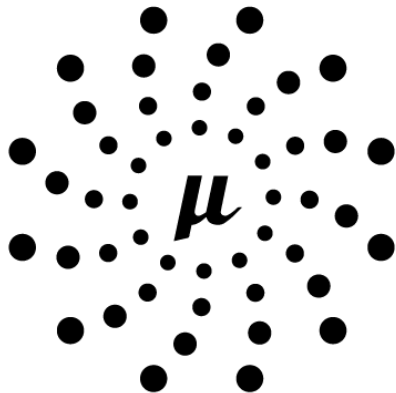


Plan

- I. PRÉSENTATION
- II. FONCTIONNEMENT
- III. FONCTIONNALITÉS
- IV. CAS D'UTILISATIONS
- V. AVANTAGES ET INCONVÉNIENTS



Présentation



M I C R O N A U T



OBJECT
COMPUTING

FONCTIONNEMENT

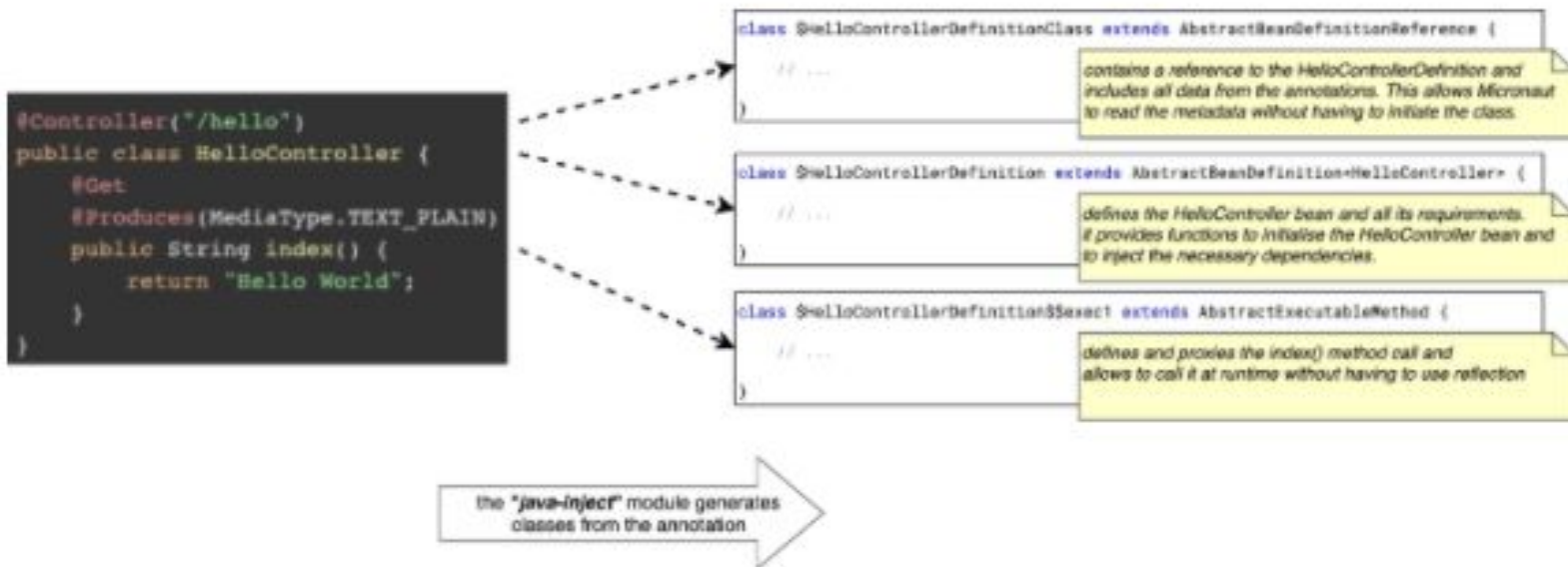


Figure: Hello World Example: classes generated during compilation

FONCTIONNEMENT

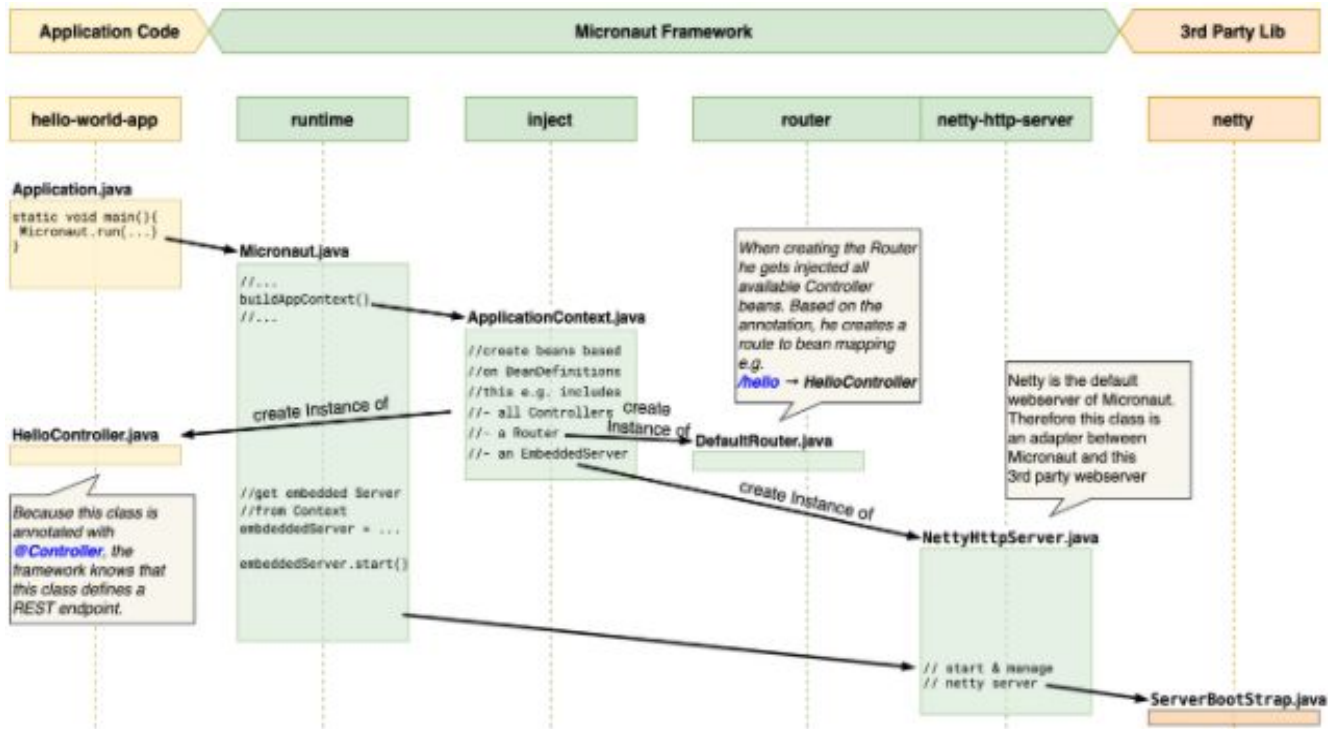


Figure: Startup phase in the Hello World example



FONCTIONNALITÉS

- Injection de dépendance et inversion de contrôle(IOC):
- Aspect Oriented Programming(AOP):
- Sensible Defaults and Auto-Configuration: Fast Access Configuration
- Micronaut propose un modèle asynchrone et non bloquant et le rend donc indiqué pour développer des applications réactives. Pour cela la couche réseau est basée sur le serveur Netty qui apporte la gestion de l'évent loop .
- Configuration d'accès des données avec les bases de données mongoDB, Neo4j, Postgres, Cassandra, Redis...

Résilience et tolérance aux fautes

- Mécaniques intégrées de retry et circuit breaker

FONCTIONNALITÉS

Cloud

- Configuration distribuée (HashiCorp Consul, AWS Parameter Store etc)
- Service Discovery (Consul, Eureka, Kubernetes)
- Load Balancing côté client (Netflix Ribbon)
- Distributed Tracing
- fonctions Serverless


NATIVELY CLOUD NATIVE

Micronaut's cloud support is built right in, including support for common discovery services, distributed tracing tools, and cloud runtimes.







FONCTIONNALITÉS

Les projets Micronaut peuvent également être générés à l'aide d'un générateur en ligne



MICRONAUT™
L A U N C H



Application Type
Application

Java Version
11

Base Package
com.example

Name
demo

Micronaut Version
☒ 2.1.3
☐ 2.2.0.BUILD-SNAPSHOT

Language
☒ Java
☐ Kotlin
☐ Groovy

Build
☒ Gradle
☐ Maven

Test Framework
☒ JUnit
☐ Spock
☐ KoTest


+ FEATURES

↔ DIFF

🔍 PREVIEW

⬇️ GENERATE PROJECT

Included Features (0)



OBJECT
COMPUTING
HOME TO MICRONAUT™



CAS D'UTILISATIONS

Avec Micronaut, on peut construire des applications de type :

- Microservices
- Serverless
- Message-Driven : axés sur les messages avec Kafka/Rabbit
- CLI : interface en ligne de commandes
- Android



AVANTAGES

- Temps de démarrage rapide
- Réduction de l'empreinte mémoire
- Utilisation minimale de la réflexion
- Utilisation minimale des proxys
- Pas de génération d'environnement d'exécution du bytecode
- Robuste, Scalable:
- Test unitaire facile

The logo for GraalVM, featuring the word "Graal" in blue and "VM" in orange, with a small trademark symbol (TM) to the right.



AVANTAGES

Source: [Red Hat](#)

	Memory - MB	Start-up time - (ms)	Execution time -(ms)
Micronaut app with Native-Image	61	18	3
Micronaut app on GraalVM JIT	278	1,100	10
SpringBoot + HotSpotVM JIT	338	1,400	15



INCONVÉNIENTS

- Certaines dépendances ne supportent pas (encore) GraalVM :
- Micronaut n'offre pas autant de fonctionnalités ou d'intégrations comme Spring Boot
- il n'existe actuellement aucune prise en charge des vues côté serveur ou des fonctionnalités typiques d'un framework MVC côté serveur traditionnel.



DÉMONSTRATION



MERCI POUR VOTRE ATTENTION



CRÉDITS

- <https://dzone.com/articles/micronaut-build-for-next-wave-of-microservices-amp>
- <https://docs.micronaut.io/1.0.0/guide/index.html#serverlessFunctions>
- <https://desosa.nl/projects/micronaut/2020/03/17/from-vision-to-architecture.html#sec-runtime-view>
- <https://micronaut.io/launch/>