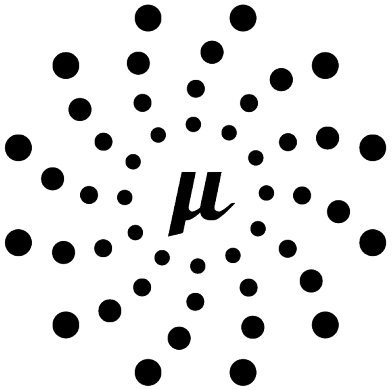
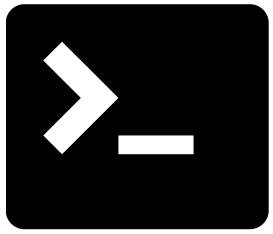
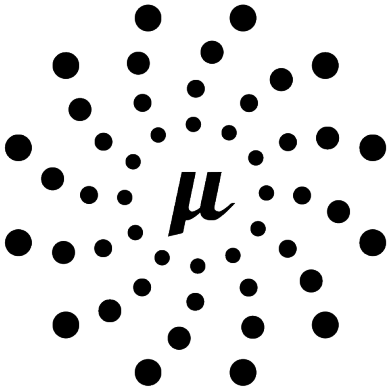


MICRONAUT®

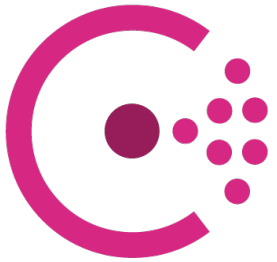
CONFIGURATION

Application Configuration

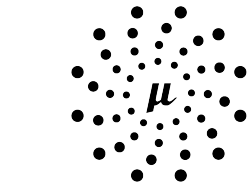




MICRONAUT®



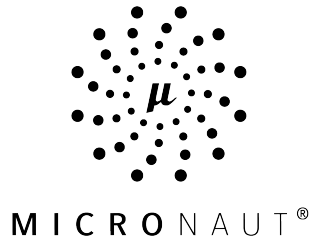
Config Files



- Micronaut Supports Multiple Formats
- Defined In `src/main/resources/`
  - `application.properties`
  - `application.yml`
  - `application.groovy`
  - `application.json`
- Environment specific configuration with -suffix:
  - `application-test.yml`
  - `application-gcp.yml`

https://docs.micronaut.io/latest/api/io/micronaut/context/env/Environment.html

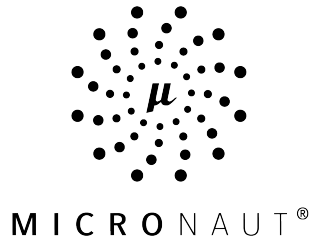
Injecting Config Values



- Config Values May Be Injected Into Property References Using `@Value(…)`

```
import io.micronaut.context.annotation.Value;

@Controller
public class SomeController {

    @Value("${some.config.value}")
    protected String someProperty;

    // ...
}
```

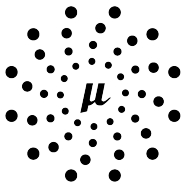
Default Config Values



- `@Value(...)` Supports A Default Value

```
import io.micronaut.context.annotation.Value;

@Controller
public class SomeController {

    @Value("${some.config.value:99}")
    protected String someProperty;

    // ...
}
```

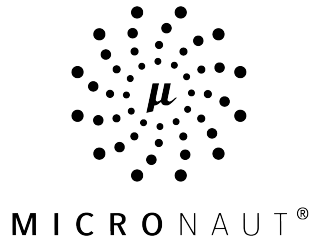
Property Placeholders



- `${value:default}` Syntax can be used in all Micronaut annotations for config injection
- Great for API versioning (but check out `@Version` first!)

```
import io.micronaut.context.annotation.Value;

@Controller("${some.controller.path:/some}")
public class SomeController {

    @Value("${some.config.value:99}")
    protected String someProperty;

    // ...
}
```

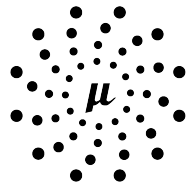
Default Config Values



- Runtime exception thrown if neither value can be found

```
import io.micronaut.context.annotation.Value;

@Controller
public class SomeController {

    @Value("${some.config.value:other.value}")
    protected String someProperty;

    // ...
}
```

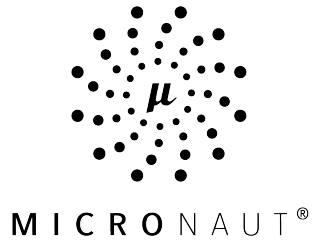
Default Config Values



- Colon is a special character. Escape with back ticks

```
import io.micronaut.context.annotation.Value;

@Controller
public class SomeController {

    @Value("${some.value:`http://localhost`}")
    protected String someProperty;

    // ...
}
```

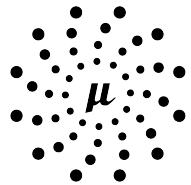
Using @Property



- Useful when you want to simply reference a specific property w/o expression syntax

```
import io.micronaut.context.annotation.Property;

@Controller
public class SomeController {

    @Property(name = “some.value”)
    protected String someProperty;

    // ...
}
```

Random Config Properties

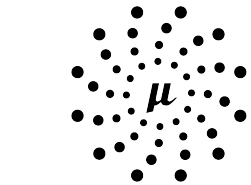


- Micronaut provides a set of randomly assigned properties that you can use within your config

Property	Value
random.port	An available random port number
random.int	Random int
random.integer	Random int
random.long	Random long
random.float	Random float
random.shortuuid	Random UUID of only 10 chars in length
random.uuid	Random UUID with dashes
random.uuid2	Random UUID without dashes

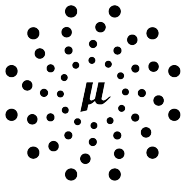
```
micronaut:
  application:
    name: myapplication
    instance:
      id: ${random.shortuuid}
```

Alternate Config Sources



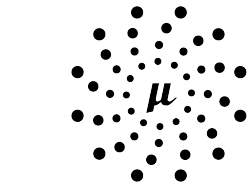
- SPRING_APPLICATION_JSON
- MICRONAUT_APPLICATION_JSON
- System Properties
- OS Environment Variables
- Cloud Configuration
- application-[env].[extension]
  - env = runtime environment

Configuration Annotations



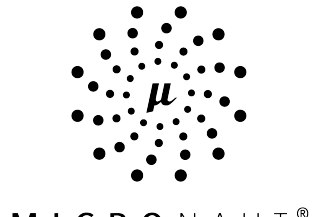
- io.micronaut.context.annotation.ConfigurationProperties
- io.micronaut.context.annotation.EachProperty
- io.micronaut.context.annotation.EachBean

@ConfigurationProperties




```
@ConfigurationProperties("my.engine")
class EngineConfig {

    @NotBlank
    String manufacturer = "Ford"

    @Min(1L)
    int cylinders

    CrankShaft crankShaft = new CrankShaft()

    @ConfigurationProperties("crank-shaft")
    static class CrankShaft {
        Optional<Double> rodLength = Optional.empty()
    }
}
```

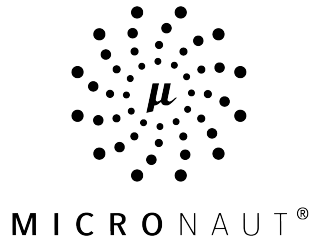
@EachProperty



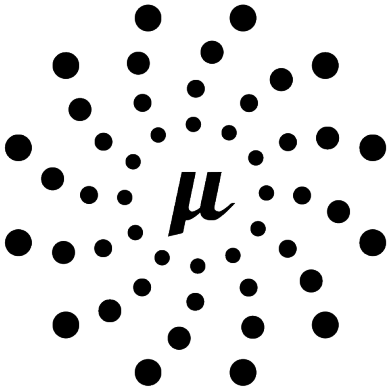

```
@EachProperty("test.datasource")
public class DataSourceConfiguration {

    private final String name;
    private URI url = new URI("localhost");

    public DataSourceConfiguration(@Parameter String name)
            throws URISyntaxException {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public URI getUrl() {
        return url;
    }

    public void setUrl(URI url) {
        this.url = url;
    }
}
```

```
test.datasource.one.url: ”jdbc:mysql://localhost/one"
test.datasource.two.url: "jdbc:mysql://localhost/two"
```

@EachBean



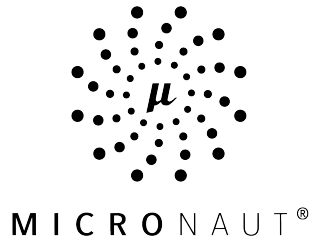

```
@Factory
public class DataSourceFactory {

    @EachBean(DataSourceConfiguration.class)
    DataSource dataSource(DataSourceConfiguration configuration) {
        URI url = configuration.getUrl();
        return new DataSource(url);
    }
}
```



MICRONAUT®

STATIC RESOURCES

Serving Static Resources

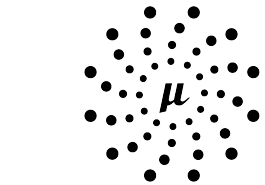


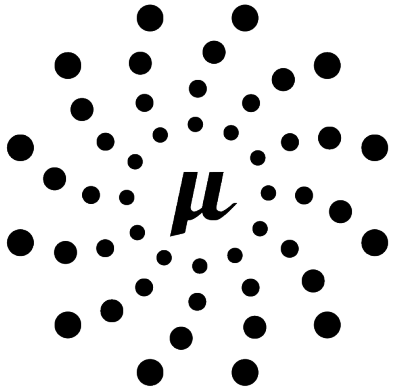
- Disabled By Default

```
micronaut:
    router:
        static-resources:
            default:
                enabled: true
                mapping: /static/**
                paths: classpath:public
```

Will resolve resources defined in `src/main/resources/public/`.

http://localhost:8080/static/someFile.txt

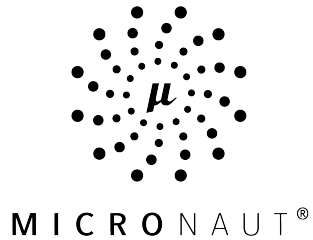




MICRONAUT®

POLYGLOT MICRONAUT

Application class



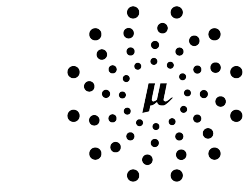

```java
package com.example;

import io.micronaut.runtime.Micronaut;

public class Application {

    public static void main(String[] args) {
        Micronaut.run(Application.class, args);
    }
}
```

Application class

MICRONAUT®

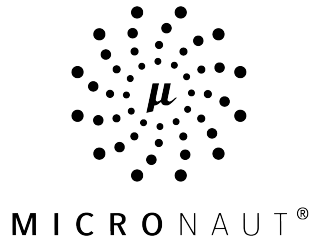
```
package com.example

import io.micronaut.runtime.Micronaut
import groovy.transform.CompileStatic

@CompileStatic
class Application {

    static void main(String[] args) {
        Micronaut.run(Application, args)
    }
}
```

Application class




```
package com.example

import io.micronaut.runtime.Micronaut.*

fun main(args: Array<String>) {
    build()
        .args(*args)
        .packages("com.example")
        .start()
}
```

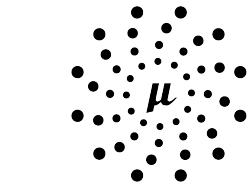
Polyglot Micronaut



"In computing, a polyglot is a computer program or script written in a valid form of multiple programming languages, which performs the same operations or output independent of the programming language used to compile or interpret it."

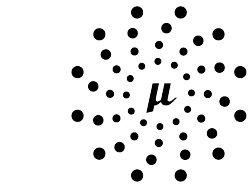
— https://en.wikipedia.org/wiki/Polyglot_(computing)

Polyglot Micronaut



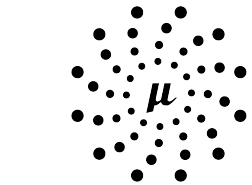
- Micronaut Is A Polyglot Framework
- Any JVM Language May Be Used
- Explicit Support For Defining Micronaut Artifacts
  - Java
  - Groovy
  - Kotlin

Polyglot AOT Compilation



- Kotlin And Java
  - Compile-Time Annotation Processors
  - kapt For Kotlin
- Groovy
  - AST Transformations

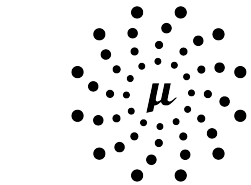
Java Dependencies




```
mn create-app appname
```

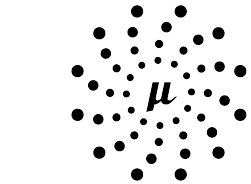
```
dependencies {
    implementation("io.micronaut:micronaut-validation")
    implementation("io.micronaut:micronaut-runtime")
    implementation("io.micronaut:micronaut-http-client")
    runtimeOnly("ch.qos.logback:logback-classic")
}
```

Java Controller




```java
package polyglotdemo;

import io.micronaut.http.annotation.Controller;
import io.micronaut.http.annotation.Get;

@Controller("/java")
public class JavaHelloController {

    @Get("/hello/{name}")
    public String hello(String name) {
        return "Hello " + name + " From Java";
    }
}
```

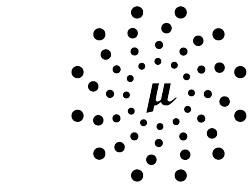
Groovy Dependencies




```
mn create-app -l groovy appname
```

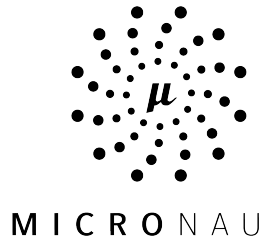
```
dependencies {
    implementation("io.micronaut:micronaut-validation")
    implementation("io.micronaut.groovy:micronaut-runtime-groovy")
    implementation("io.micronaut:micronaut-http-client")
    runtimeOnly("ch.qos.logback:logback-classic")
}
```

Groovy Controller




```groovy
package polyglotdemo

import io.micronaut.http.annotation.Controller
import io.micronaut.http.annotation.Get

@Controller('/groovy')
class GroovyHelloController {

    @Get('/hello/{name}')
    String hello(String name) {
        "Hello $name From Groovy"
    }
}
```

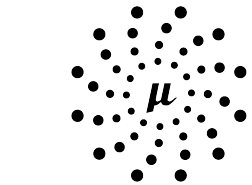
Kotlin Dependencies




```
mn create-app –l kotlin appname
```

```
dependencies {
    implementation("io.micronaut:micronaut-validation")
    implementation("org.jetbrains.kotlin:kotlin-stdlib-jdk8:${kotlinVersion}")
    implementation("org.jetbrains.kotlin:kotlin-reflect:${kotlinVersion}")
    implementation("io.micronaut.kotlin:micronaut-kotlin-runtime")
    implementation("io.micronaut:micronaut-runtime")
    implementation("io.micronaut:micronaut-http-client")
    runtimeOnly("ch.qos.logback:logback-classic")
    runtimeOnly("com.fasterxml.jackson.module:jackson-module-kotlin")
}
```

Kotlin Controller




```kotlin
package polyglotdemo

import io.micronaut.http.annotation.Controller
import io.micronaut.http.annotation.Get

@Controller("/kotlin")
class KotlinHelloController {

    @Get("/hello/{name}")
    fun hello(name: String): String {
        return "Hello $name From Kotlin"
    }
}
```