



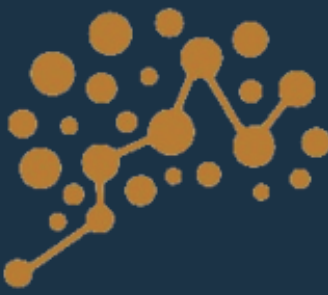
# QUARKUS

A cura di Davide Galati (in arte PsykeDady)

[davide.galati@brainylabs.it](mailto:davide.galati@brainylabs.it)

[psdady@msn.com](mailto:psdady@msn.com)





## Cos'è

*What we mean by "Supersonic Subatomic Java".*

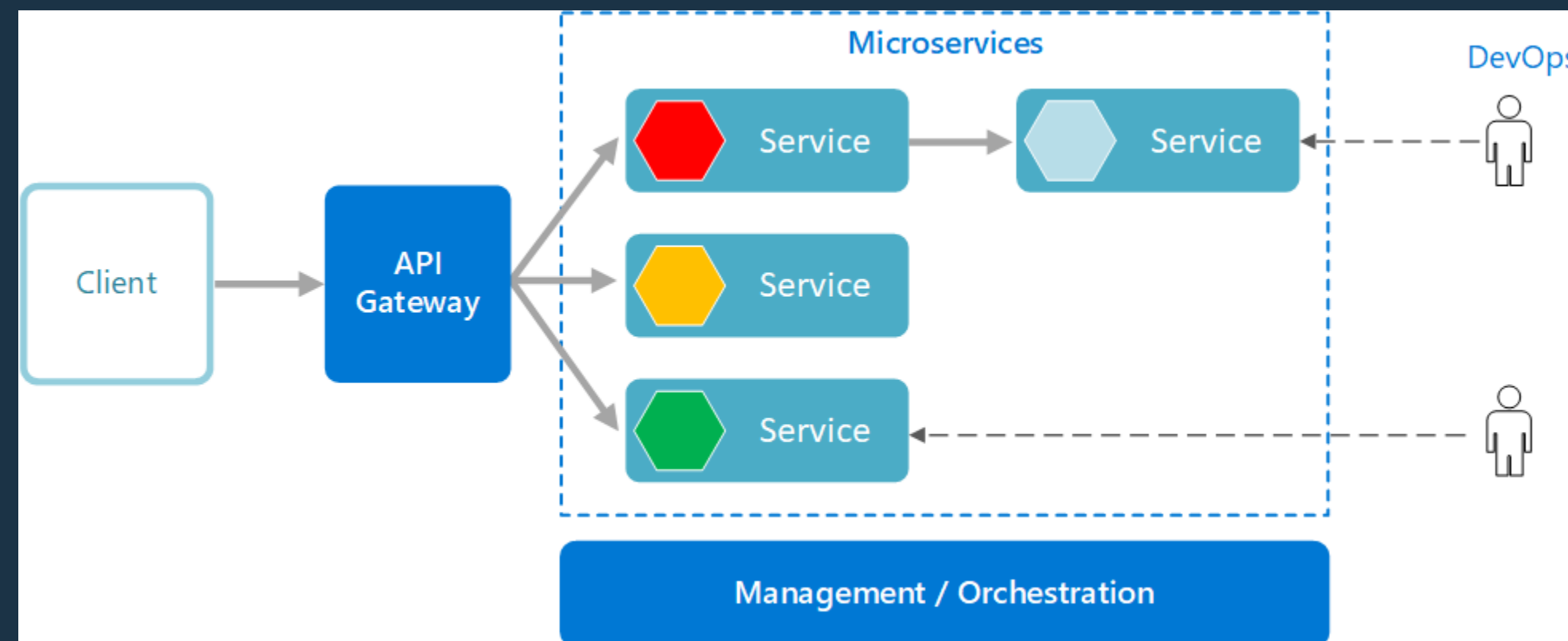
*~Sito quarkus, sezione about*

Mentre l'innovazione tecnologica verte sullo sviluppo di applicazioni a micro servizi, lo stack tecnologico Java tradizionalmente è formato da applicazioni monolitiche in genere molto pesanti





# Cos'è



# Cos'è

Quarkus è un framework full REST creato per consentire agli sviluppatori Java di creare applicazioni orientate al mondo cloud





java  
source



quarkus  
cli



kubernetes

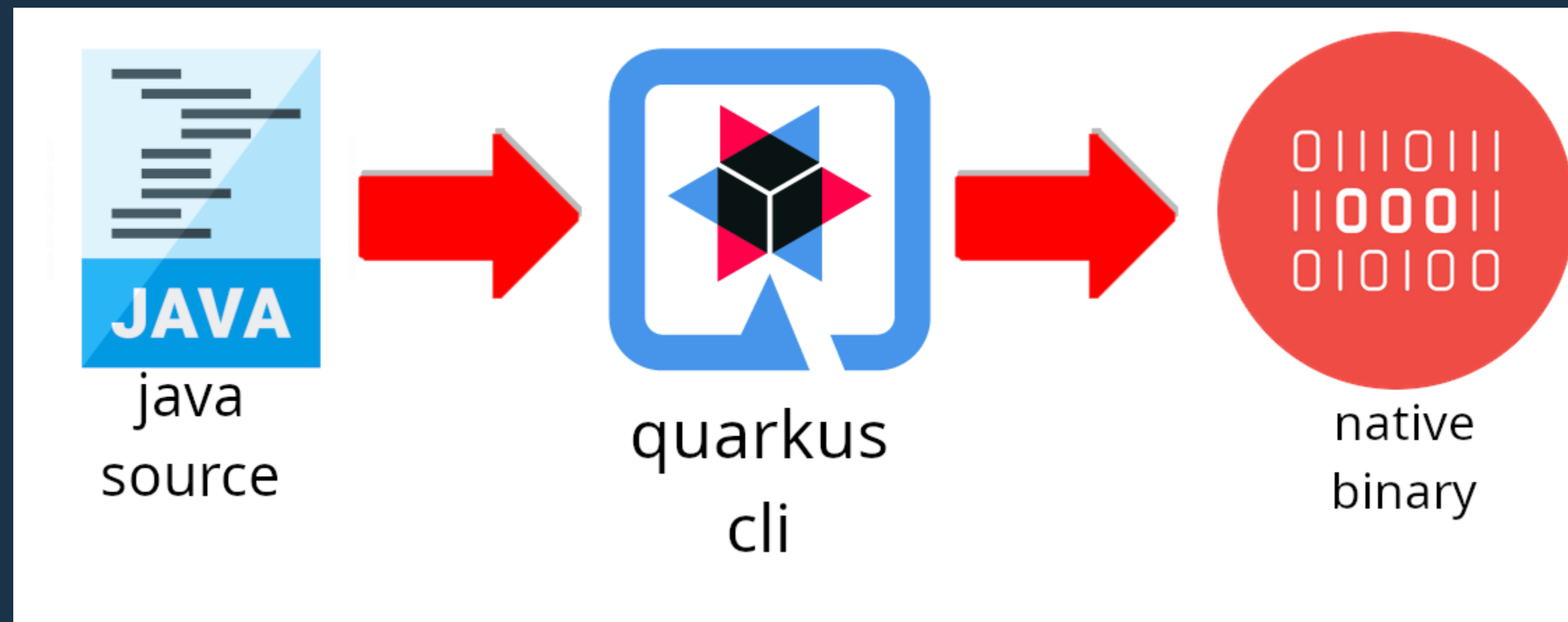


cloud computing  
generico



## Cos'è

Nel farlo crea, a partire da un sorgente Java, un binario nativo per il sistema con le librerie incluse





## Howto

Per creare un progetto di quarkus si può seguire [il tutorial ufficiale](#).  
I prerequisiti sono fondamentalmente:

- Installare Java GraalVM (seguire istruzioni da sito)
- Installare Quarkus cli
- Docker (per testare il caso più verosimilmente possibile)





# Howto - Installare Quarkus cli

Da terminale UNIX eseguire:

```
curl -Ls https://sh.jbang.dev | bash -s - trust add https://repo1.maven.org/maven2/io/quarkus/quarkus-cli/
```

Quindi scrivere:

```
curl -Ls https://sh.jbang.dev | bash -s - app install --fresh --force quarkus@quarkusio
```







## Howto - Installare Quarkus cli (Windows)

Per installare il tool su Windows è consigliato uno dei seguenti software per emulare l'ambiente UNIX:

- WSL
- Cygwin
- MinGW





# Howto - Installare Quarkus cli (Windows Powershell)

Si può eventualmente utilizzare powershell scrivendo:

```
iex "& { $(iwr https://ps.jbang.dev) } trust add https://repo1.maven.org/maven2/io/quarkus/quarkus-cli/"  
iex "& { $(iwr https://ps.jbang.dev) } app install --fresh --force quarkus@quarkusio"
```





## Strumenti/IDE

- Quarkus è un framework relativamente nuovo (~**2019**), ma comunque supportato da alcuni IDE.
- È possibile consultare sul sito stesso [la lista degli IDE e delle relative estensioni](#).

*Nota del presentatore:*

*Personalmente utilizzo ed ho testato solo il plugin di Visual Studio Code.  
Nella presentazione sarà quindi mostrato quello.*



# Visual Studio Code - estensioni



- È possibile installare l'estensione per quarkus di Visual Studio Code direttamente dall'IDE utilizzando l'apposito menù



1

2

3

4

ESTENSIONI: MARKETPLACE

quarkus

Quarkus

Quarkus Tools for Visual S...

Red Hat

Installa

Quarkus snip...

Quarkus snippets

offrey GREBERT

Installa

Quarkus Dark The...

Quarkus dark theme for Vi...

Tihomir Surdilovic

Installa

Quarkus debug sn...

Snippets for debugging wi...

Matthiasbrat

Installa

Advanced Java + ...

This collection includes all ...

Arthur Gulate Br...

Installa

Language Sup...

Provides completion, valid...

Red Hat

Installa

AtlasMap-Dat...

Opens a browser for Atlas...

Red Hat

Installa

Quarkus

v1.17.0

Anteprima

Red Hat

redhat.com

251.343

★★★★★(6)

Quarkus Tools for Visual Studio Code

Installa

DETTAGLI

FUNZIONALITÀ

LOG DELLE MODIFICHE

DIPENDENZE

VS MARKETPLACE

V1.17.0

INSTALLS

251K

BUILD

PASSING

LICENSE

APACHE-2.0

Description

This Visual Studio Code extension provides support for :

- Quarkus and MicroProfile development by extending Visual Studio Code extension for MicroProfile with Quarkus features.

[Extension Development Host] - application.properties — using-vertx

EXPLORER

OPEN EDITORS

application.properties ... 2

USING-VERTX

OUTLINE

{ } %dev

{ } myapp

{ } schema

create true

{ } quarkus

{ } datasource

password quarkus\_test

url vertx-reactive:postgres...

username quarkus\_test

{ } http

port 8080

{ } ssl

native true

application.properties

src > main > resources > application.properties > ...

1 quarkus.ssl.native=true

2 quarkus.datasource.url=vertx-reactive:postgresql://localhost:5432/quarkus\_test

3 quarkus.datasource.username=quarkus\_test

4 quarkus.datasource.username=quarkus\_test

5 quarkus.datasource.password=quarkus\_test

6 %dev.myapp.schema.create=true

7

8 quarkus.http.port

9 The HTTP port

10 • Type: int

11 • Default: 8080

12 • Phase: runtime

13 • Extension: quarkus-undertow

14

15 quarkus.http.port=8080

PROBLEMS 2

OUTPUT

DEBUG CONSOLE

TERMINAL

Filter. E.g.: text, \*\*/...

application.properties src/main/resources 2

⚠ Duplicate property 'quarkus.datasource.username' quarkus(duplicate) [3, 1]

⚠ Duplicate property 'quarkus.datasource.username' quarkus(duplicate) [4, 1]

Categorie

Programming Languages

Risorse

Marketplace

Problemi

Repository

Licenza

Red Hat

Altre info

Publicato

2019-09-20, 19:36:18

Ultimo rilascio

2024-04-12, 10:20:14

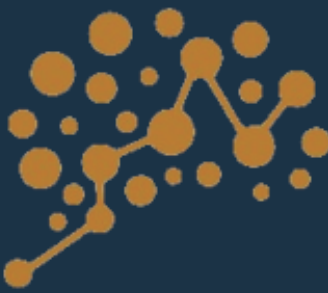
Identificatore

redhat.vscode-quarkus

❗ Installare estensione 'Extension Pack for Java' di Microsoft consigliata/e per questo repository?

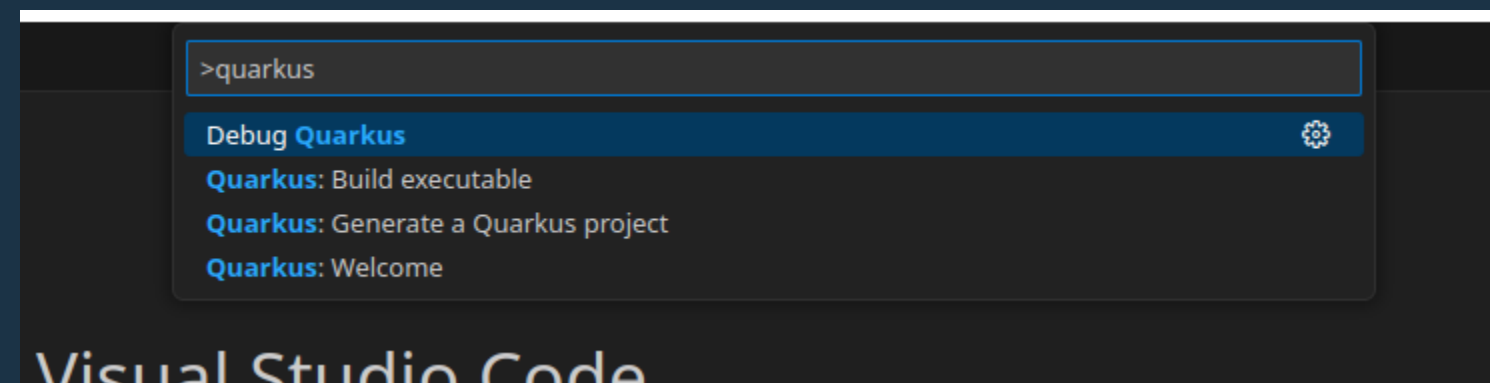
Installa

Mostra elementi consigliati



## Visual Studio Code - estensioni

- Trattandosi di un framework scritto in java è consigliabile comunque utilizzare tutto il set di estensioni di base di RedHat per Java.
- Una volta installata saranno possibili diverse opzioni.
- Per visualizzare aprire il control panel (CTRL+Shift+P su Win/Linux, Cmd+Shift+P su Mac).
- Una volta aperto cercare "Quarkus" per vedere le azioni disponibili





## Visual Studio Code - estensioni

- In generale, l'opzione più utile è sicuramente quella per fare il DEBUG e l'esecuzione del progetto.





## Creare un progetto Quarkus

- Una volta installato il progetto è necessario semplicemente scrivere:

```
quarkus create
```

- Il tool creerà in maniera totalmente autonoma il progetto, dentro cui sarà poi possibile navigare.







## Creare un progetto Quarkus

- A fine esecuzione si potrà leggere:

```
-----  
applying codestarts...  
📄 java  
🔨 maven  
📦 quarkus  
📄 config-properties  
🔧 tooling-dockerfiles  
🔧 tooling-maven-wrapper  
🚀 resteasy-reactive-codestart  
  
-----  
[SUCCESS] ✅ quarkus project has been successfully generated in:  
--> /home/SharedFiles/code-with-quarkus  
-----  
Navigate into this directory and get started: quarkus dev
```

- Nell'ultima riga si può leggere l'istruzione per far partire il progetto.



# Un'occhiata alla struttura del progetto...



- Aprendo il progetto la cosa sicuramente più evidente è una grande pulizia nella struttura stessa.



ESPLORA RISORSE

EDITOR APERTI

CODE-WITH-QUARKUS

java/org/acme

GreetingResource.java

resources

test

.dockerignore

.gitignore

mvnw

mvnw.cmd

pom.xml

README.md

STRUTTURA

SEQUENZA TEMPORALE

MAVEN

SONARLINT ISSUE LOCATIONS

JAVA PROJECTS

pom.xml

pom.xml

<?xml version="1.0"?>

<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd" xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<modelVersion>4.0.0</modelVersion>

<groupId>org.acme</groupId>

<artifactId>code-with-quarkus</artifactId>

<version>1.0.0-SNAPSHOT</version>

<properties>

<compiler-plugin.version>3.12.1</compiler-plugin.version>

<maven.compiler.release>17</maven.compiler.release>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<quarkus.platform.artifact-id>quarkus-bom</quarkus.platform.artifact-id>

<quarkus.platform.group-id>io.quarkus.platform</quarkus.platform.group-id>

<quarkus.platform.version>3.9.3</quarkus.platform.version>

<skipITs>true</skipITs>

<surefire-plugin.version>3.2.5</surefire-plugin.version>

</properties>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>\${quarkus.platform.group-id}</groupId>

<artifactId>\${quarkus.platform.artifact-id}</artifactId>

<version>\${quarkus.platform.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<dependencies>

<dependency>

<groupId>io.quarkus</groupId>

<artifactId>quarkus-arc</artifactId>

</dependency>

<dependency>

<groupId>io.quarkus</groupId>

<artifactId>quarkus-rest</artifactId>

</dependency>

<dependency>

<groupId>io.quarkus</groupId>

<artifactId>quarkus-junit5</artifactId>

<scope>test</scope>

</dependency>

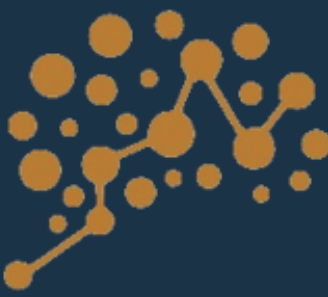
<dependency>

<groupId>io.rest-assured</groupId>

<artifactId>rest-assured</artifactId>

Maven dependency explorer

Opening Java Projects: [check details](#)



## Un'occhiata alla struttura del progetto...

- I punti di interesse maggiori sono sicuramente:
  - `pom.xml`
  - `src/main/java/org/acme/GreetingResource.java` che rappresenta un primo endpoint di esempio
- Il pom fa riferimento al groupId `org.acme` e artifactID `code-with-quarkus`.
- Questi valori ovviamente possono essere cambiati sostituendo quelli della propria organizzazione nonché del nome del progetto
- Di conseguenza può essere cambiato/creato anche il package per la classe Java





## Un "ping" bring test

- Per iniziare con un attività semplice e vedere il funzionamento del proprio del progetto si può personalizzare la classe `GreetingResouce`.
- Ad esempio modificando package e contenuto dell'Hello World.
- Si può notare una struttura simile agli applicativi Spring, ma ancora di più ai server JAX-RS
  - Ad esempio il path del Controller è dato dall'annotazione `@Path`
  - La tipologia di metodo HTTP utilizzato è dato dall'annotazione `@GET`
  - Il tipo di risultato restituito è specificato dal tipo di ritorno `String`, ma anche dall'annotazione `@Produces(MediaType.TEXT_PLAIN)`
- Cambiamo anche nome del metodo e path API del controller:

```
@Path("/bring")
public class BringTest {

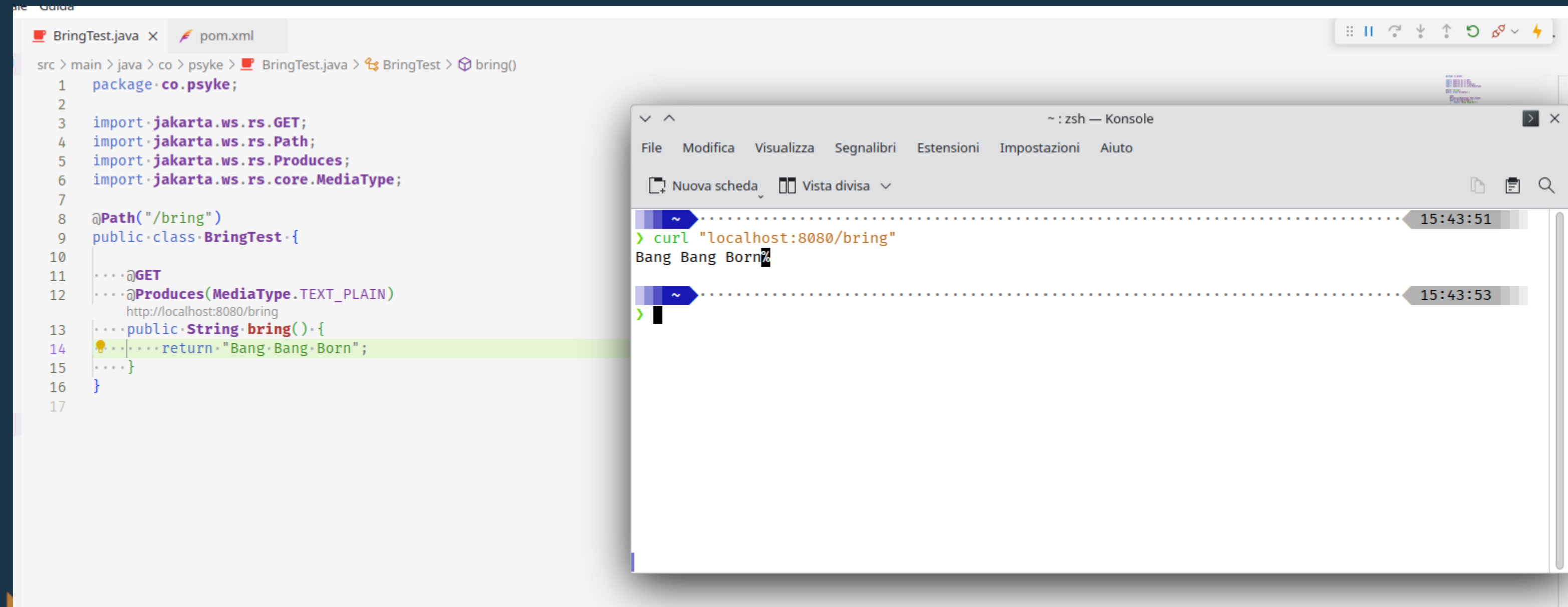
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String bring() {
        return "Bang Bang Born";
    }
}
```





# Un "ping" bring test

Ecco il risultato:



The screenshot shows an IDE with two windows. The left window displays the source code of a Java class named `BringTest`. The right window is a terminal titled `~ : zsh — Konsole` showing the execution of a `curl` command.

```
src > main > java > co > psyke > BringTest.java > BringTest > bring()
1 package co.psyke;
2
3 import jakarta.ws.rs.GET;
4 import jakarta.ws.rs.Path;
5 import jakarta.ws.rs.Produces;
6 import jakarta.ws.rs.core.MediaType;
7
8 @Path("/bring")
9 public class BringTest {
10
11     @GET
12     @Produces(MediaType.TEXT_PLAIN)
13     http://localhost:8080/bring
14     public String bring() {
15         return "Bang Bang Born";
16     }
17 }
```

```
~ : zsh — Konsole
File Modifica Visualizza Segnalibri Estensioni Impostazioni Aiuto
Nuova scheda Vista divisa
> curl "localhost:8080/bring"
Bang Bang Born%
15:43:51
15:43:53
```





**Compilare ed avviare il progetto**





**Compilare ed avviare il progetto**







**Build nativa**





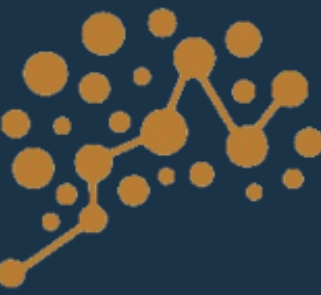
**Build nativa (Windows)**





**Build nativa**





# The End



(T.hanks)



**Slide semplice**





**Slide innestata1**





**Slide innestata 2**

