

# Opc Server with Milo and Spring Boot

---

Documentazione di progetto per la materia Informatica Industriale

- [Introduzione](#)
- [Perché Spring Boot](#)
- [Descrizione del progetto](#)
  - [WeatherService](#)
- [Struttura delle cartelle](#)
  - [Strumenti utilizzati](#)
- [Funzionamento](#)
- [Annex 1: GitHub](#)
  - [Step1: Creare il progetto su github](#)
  - [Step2: Readme and .gitignore](#)
  - [Step3: Repo Download](#)
- [Annex 2: Camel Demo with Spring Boot](#)
  - [Step1: Spring Init](#)
  - [Step2: Importare il progetto Maven dentro Eclipse](#)
  - [Step3: Apache Milo](#)

## Introduzione

La presente documentazione descrive come poter implementare un Server Opc utilizzando come standard SDK per implementare lo stack protocollare [Eclipse Milo](#).


Nell'ambito del corso di Informatica Industriale è stato richiesto di sviluppare un Server OpcUa.

Il server opc deve soddisfare le seguenti caratteristiche:

- Implementare diversi meccanismi di security
- Implementare delle funzionalità interessanti

Al fine di rispettare i requisiti di cui sopra si è scelto di implementare un server con le seguenti caratteristiche:


- Diversi endpoint con il supporto ai certificati
- Implementazione del meccanismo di autenticazione
- Implementazione di un servizio che prenda i dati da [OpenWeater](#)

 [OpenWeater](#) è un servizio on line che permette previa registrazione di utilizzare API atte a fornire informazioni meteo il servizio è gratuito se si rispettano alcuni limiti, per ulteriori informazioni -> [OpenWeater Price](#)


L'idea di questo progetto è quella di implementare il server Opc utilizzando [Spring Boot](#), un framework di sviluppo molto diffuso anche in ambito java enterprise.

## Perché Spring Boot

Spring Boot rappresenta un framework che aiuta lo sviluppatore ad implementare automaticamente dei pattern architetturali durante lo sviluppo del progetto.

 Spring è uno dei framework più utilizzati in ambito enterprise. Uno degli obiettivi di questo progetto è fornire una facile implementazione dello Stack OPC UA utilizzando Eclipse Milo, allo scopo di far vedere come chi utilizzi questo Framework possa aggiungere questo stack alla lista dei servizi esposti.

Capo saldo di Spring Boot e più in generale di Spring framework è il concetto dell'IoC (Inversion of control). Il framework definisce quello che si chiama IoC Container, l'IoC Container rappresenta un'area nella quale vengono istanziati gli oggetti del progetto. Gli oggetti che si trovano all'interno del Container vengono istanziati automaticamente utilizzando di default il pattern Singleton evitando così la proliferazione di oggetti nel heap-space di java. Tutti gli oggetti definiti nel Container sono disponibili per essere utilizzati dagli altri oggetti del sistema, in pratica ogni oggetto potrà utilizzare gli altri senza doverli costruire il tutto facilitato dall'annotazione [@Autowire](#).

Nel corso di questo documento verranno mostrati quali siano i vantaggi nell'usare questo tipo di framework, tali vantaggi verranno evidenziati dall'icona .

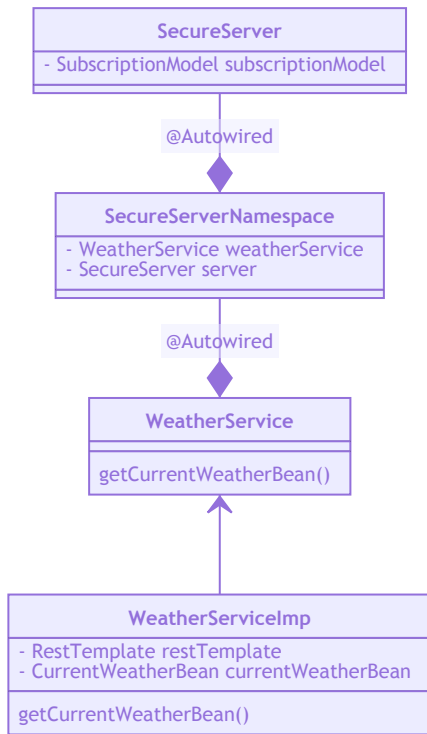
Questa documentazione non ha come obiettivo la descrizione dettagliata del framework, per maggiori informazioni consultare la documentazione ufficiale.

## Descrizione del progetto

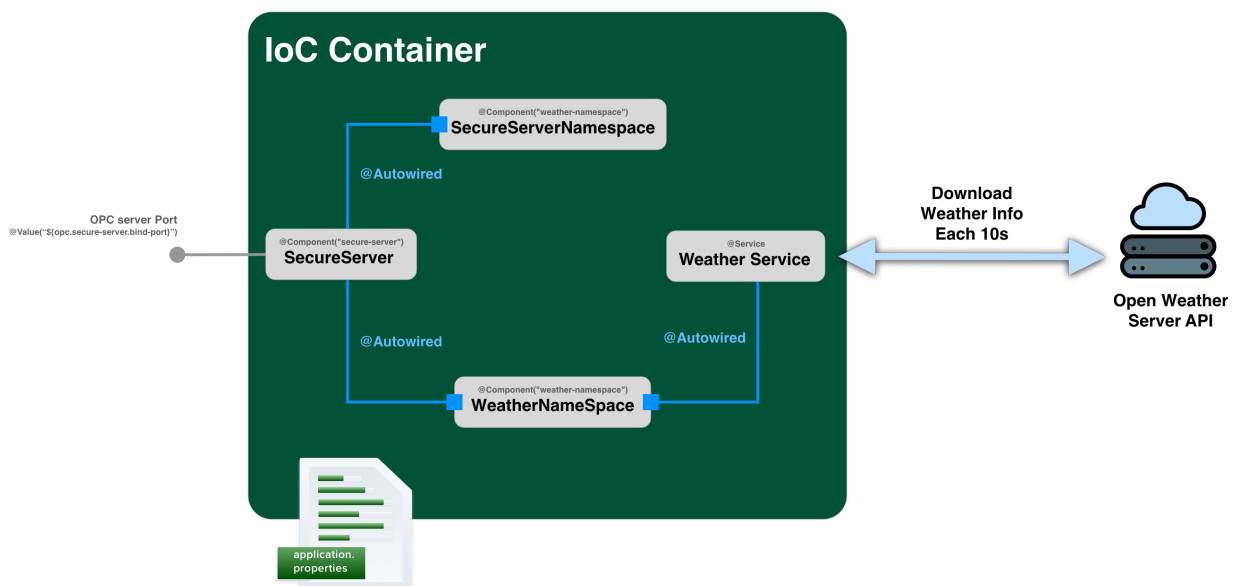
Da un punto di vista logico il progetto si compone da un insieme di oggetti che vengono creati all'interno dell'loC container di Spring. Gli oggetti che vengono istanziati sono gestiti *automaticamente* da Spring.

✓ Spring Istanza automaticamente tutto gli oggetti utilizzando il pattern **SINGLETON** evitando così la proliferazione incontrollata degli oggetti

I principali oggetti che verranno messi nell'loC container sono:



Il sistema quindi si compone dai seguenti elementi:



L'immagine sopra mostra la struttura del sistema. Come si vede dall'immagine i componenti atti a gestire il sistema sono

- WeatherService
- SecureServer
- SecureServerNamespace
- WeatherNamespace

Nel seguito dettaglieremo tutti questi componenti.

## WeatherService

## Struttura delle cartelle

Il progetto si struttura in diverse cartelle in accordo alla seguente struttura:

```
spring-camel-milo-server
├── dist
├── doc
└── opccserver
```

dove:

- dist: contiene l'eseguibile di questo progetto, dentro la cartella è presente una descrizione per far partire il progetto.
- doc: contiene i sorgenti della documentazione.
- opccamelservice: contiene un server Opc di esempio utilizzando Apache Camel come descritto nell'[Annex 2](#).
- opccserver: contenente il progetto java oggetto della presente documentazione.

## Strumenti utilizzati

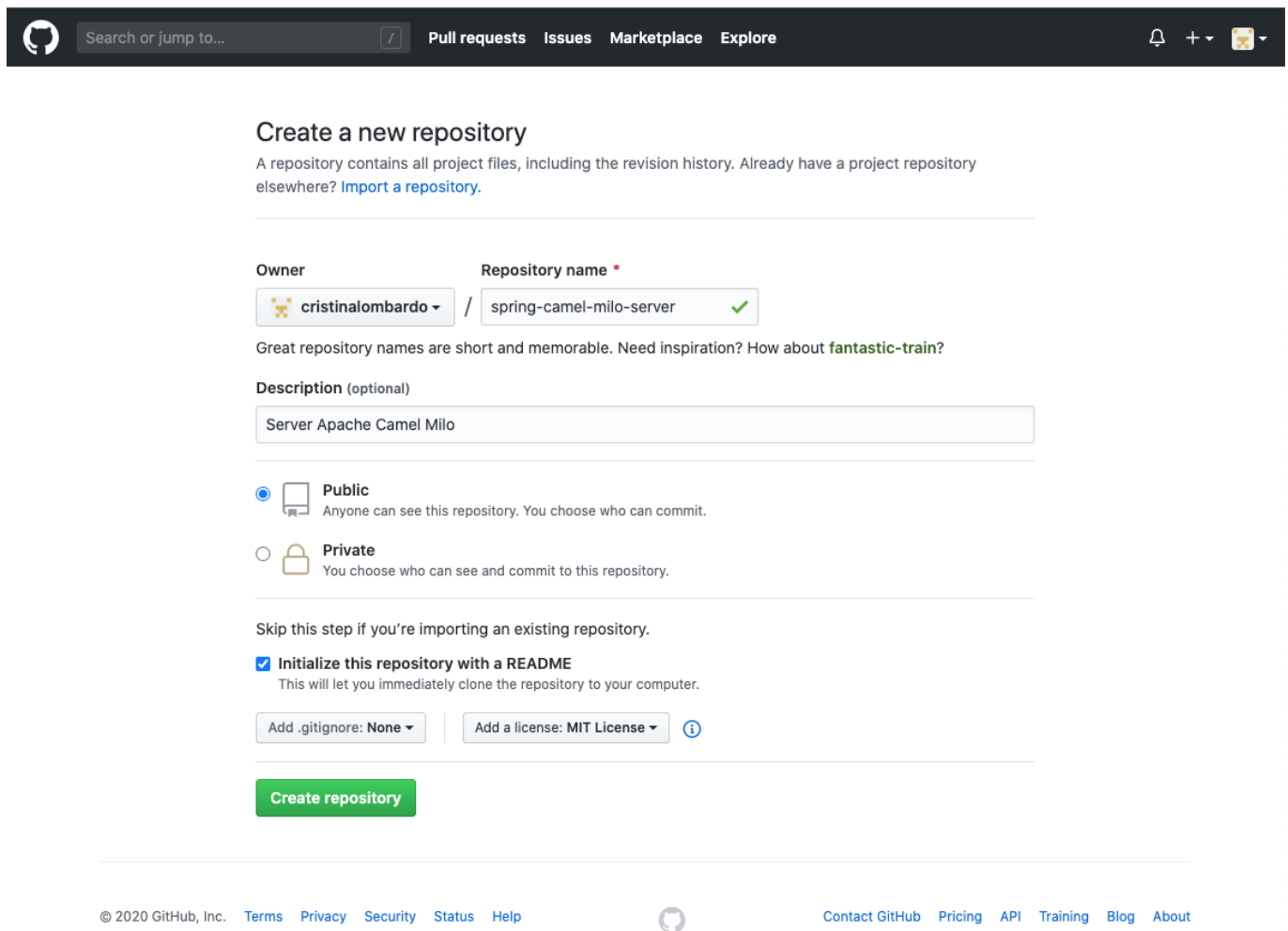
- Java SE 9
- Maven
- Eclipse with Spring tool suite
- [Spring initializr](#)

## Funzionamento

//TODO

## Annex 1: GitHub

### Step1: Creare il progetto su github





**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---

**Owner** **Repository name \***


 cristinalombardo / spring-camel-milo-server 


Great repository names are short and memorable. Need inspiration? How about **fantastic-train**?

**Description (optional)**

Server Apache Camel Milo

---


☒  **Public**  
Anyone can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

---

Skip this step if you're importing an existing repository.


☒ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** | Add a license: **MIT License** 

---

**Create repository**

---

© 2020 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#)  [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

### Step2: Readme and .gitignore

Aggiungere Readme e .gitignore

Come template del .gitignore è stato scelto Maven.

Skip this step if you're importing an existing repository.

☒ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▼

Add a license: **MIT License** ▼



.gitignore

Mave

Maven

2020 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#)



[Contat](#)

### Step3: Repo Download

Creare il repository e clonarlo in locale con il programma preferito di gestione del git.

## Annex 2: Camel Demo with Spring Boot

Prerequisiti:

- Java 9+
- Eclipse per j2ee con Spring Tools Suite Plugin installato

### Step1: Spring Init

Per inizializzare il progetto utilizzeremo Spring Initializr

[Go to Spring initializr](#)

Inizializzare il progetto in accordo alla seguente immagine

The screenshot shows the Spring Initializr web interface. On the left, there is a sidebar with a hamburger menu icon and a Twitter icon. The main content area is divided into two columns. The left column contains the 'Project' and 'Language' sections. The 'Project' section has radio buttons for 'Maven Project' (selected) and 'Gradle Project'. The 'Language' section has radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'. Below this is the 'Spring Boot' section with radio buttons for versions: '2.3.1 (SNAPSHOT)', '2.3.0' (selected), '2.2.8 (SNAPSHOT)', '2.2.7', '2.1.15 (SNAPSHOT)', and '2.1.14'. The 'Project Metadata' section includes fields for 'Group' (com.github.cristinalombardo), 'Artifact' (opcsrver), 'Name' (Spring Boot OPC Server), 'Description' (OPC Server using Apache Camel Milo into Spring Boot), and 'Package name' (com.github.cristinalombardo.opcsrver). The 'Packaging' section has radio buttons for 'Jar' (selected) and 'War'. The 'Java' section has radio buttons for '14', '11' (selected), and '8'. The right column contains the 'Dependencies' section with a button 'ADD DEPENDENCIES... 8 + 8'. Below this is the 'Spring Boot DevTools' section, which is highlighted with a green 'DEVELOPER TOOLS' tag. It includes a description: 'Provides fast application restarts, LiveReload, and configurations for enhanced development experience.' At the bottom of the form, there are three buttons: 'GENERATE ⌘ + ↵', 'EXPLORE CTRL + SPACE', and 'SHARE...'. The top right corner of the form has a settings icon and a dark mode toggle icon.

Per inizializzare il progetto abbiamo utilizzato le seguenti configurazioni:

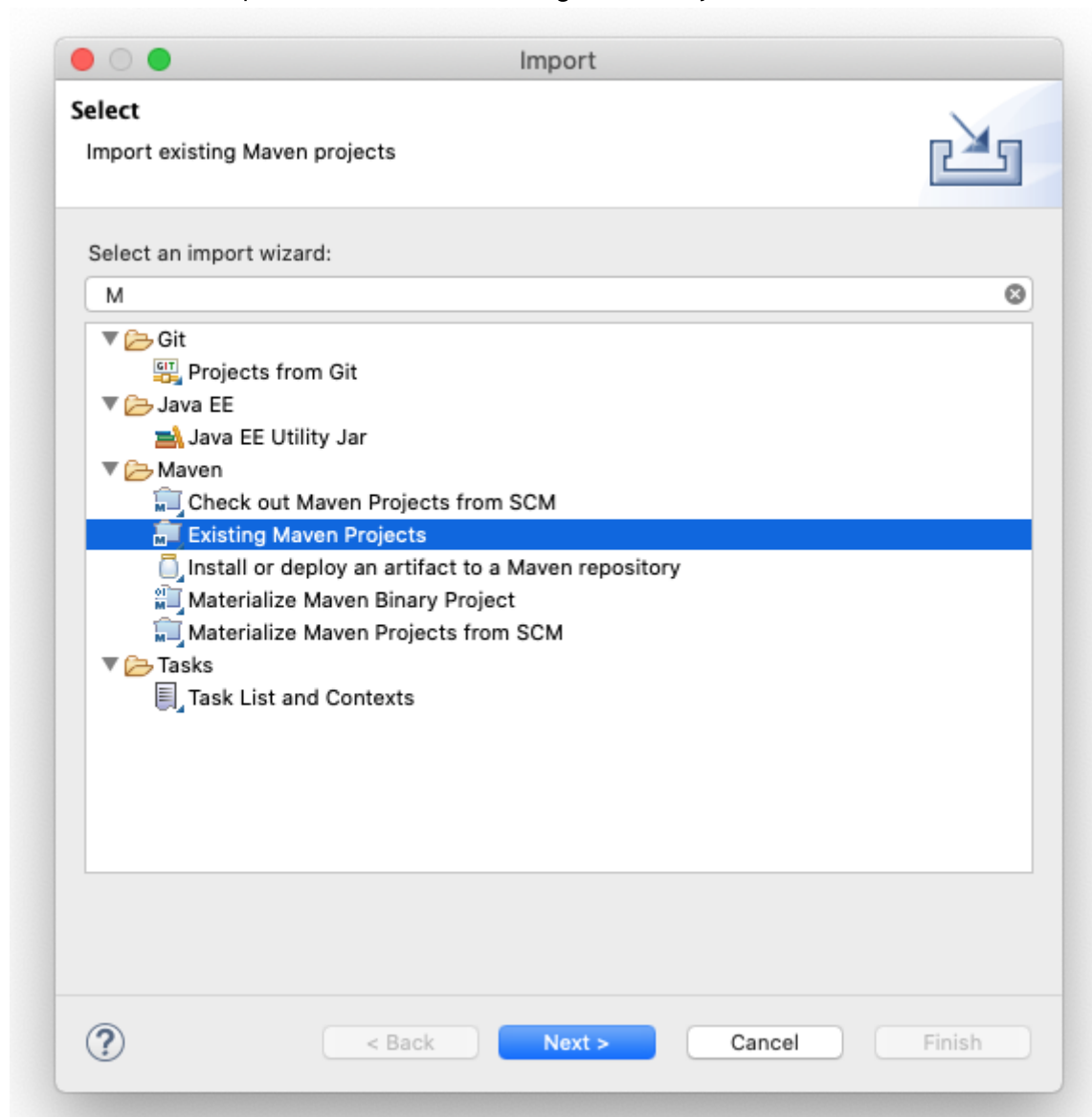
- Maven: Utilizzeremo Maven per compilare il progetto
- Spring Boot 2.3.0: Versione Stabile di Spring Boot al momento di questo tutorial
- Packaging Jar: Si vuole realizzare un'applicazione standalone
- Java 11: Apache Camel Milo richiede Java 9+
- Dependencies Spring Boot DevTools: dipendenza utile durante lo sviluppo

A questo punto è possibile scaricare il progetto cliccando sul tasto GENERATE.  
Una volta scaricato il progetto è stato copiato nella root del repository e decompattato.

## Step2: Importare il progetto Maven dentro Eclipse

Per importare il progetto i passi sono:

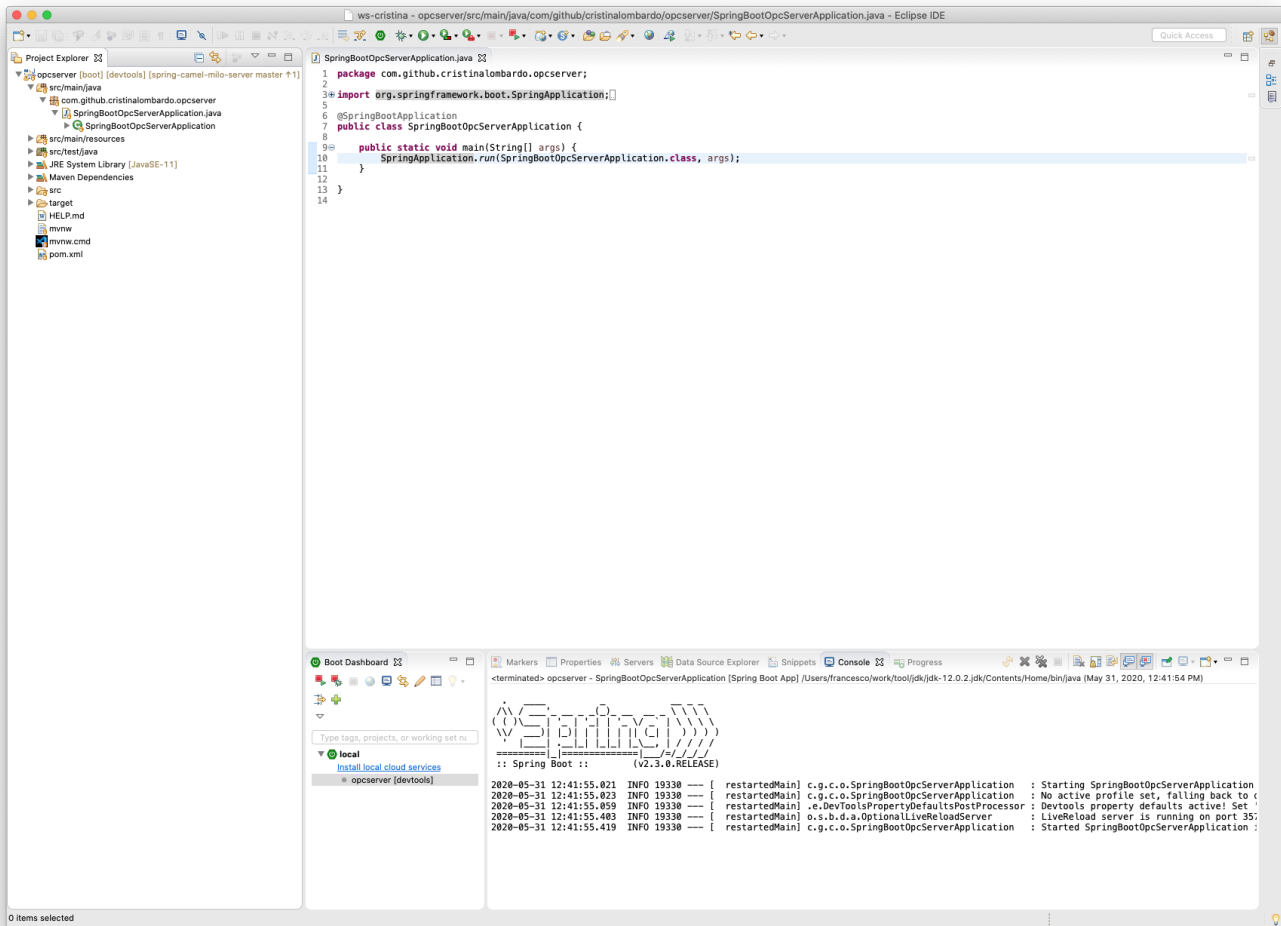
1. Aprire Eclipse
2. Cliccare su File->Import... e selezionare Existing Maven Project



3. Selezionare la cartella dove risiede il progetto e cliccare su Finish

A questo punto il progetto dovrebbe presentarsi nel seguente modo:





## Step3: Apache Milo

In questo step aggiungeremo le librerie Apache Camel Milo.

1. Apriamo il file `pom.xml`
2. Aggiungiamo la seguente dipendenza al file

```

<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-milo-starter</artifactId>
  <version>${camel.milo.version}</version>
  <!-- use the same version as your Camel core version -->
</dependency>

```

La versione di Camel viene inserita nella sezione `<properties>` del `pom.xml`

```

<camel.milo.version>3.0.0</camel.milo.version>

```