Accademy Java Full Stack

SpringBoot

Bari, ottobre 2023

SIDEA | GROUP

Sommario

Introduzione a Maven

Gestione progetti Java

Introduzione a Maven

Gestione progetti Java

Cosa è Maven?

Maven è uno strumento di build dei progetti Java con le seguenti finalità:

→ **Semplificare il processo di build**

→ **Fornire un sistema di build uniforme**

→ **Fornire informazioni sulla qualità del progetto**

→ **Incoraggiare le buone pratiche nello sviluppo del software**

Introduzione a Maven

Gestione progetti Java

Semplificare il processo di build.

Sebbene l'utilizzo di Maven non elimini la necessità di conoscere i meccanismi sottostanti, Maven libera gli sviluppatori da molti dettagli, come ad esempio gestione e download automatico delle librerie necessarie al progetto.

Introduzione a Maven

Gestione progetti Java

Fornire un sistema di build uniforme.

Maven crea un progetto utilizzando il *project object model* (**POM**) e un insieme di **plug-in**. Una volta che hai familiarizzato con un progetto Maven, sai come vengono costruiti tutti i progetti Maven. Ciò consente di risparmiare tempo se gestisci molti progetti.

Introduzione a Maven

Gestione progetti Java

Fornire informazioni sulla qualità del progetto.

Maven fornisce informazioni utili sul progetto che sono in parte prese dal POM e in parte generate dal codice sorgente. Ad esempio, Maven può fornire:

- → **Dipendenze utilizzate dal progetto**
- → **Rapporti sui test unitari inclusa la copertura**
- → **Change Log, integrando Maven col sistema di versionamento del codice**

I prodotti di analisi del codice di terze parti forniscono anche plug-in Maven che aggiungono i loro report alle informazioni standard fornite da Maven (ad esempio *SonarScanner*).

Introduzione a Maven

Gestione progetti Java

Incoraggiare le buone pratiche nello sviluppo del software.

Maven mira a raccogliere le best practices per lo sviluppo software e facilitare la gestione del progetto secondo tali principi.

Ad esempio, la specifica, l'esecuzione e il reporting dei test unitari fanno parte del normale ciclo di creazione utilizzando Maven. Le linee guida per i test sonno:

- → **Mantenere il codice sorgente dei test in un albero separato, ma parallelo**
- → **Utilizzo delle convenzioni sui nomi dei test case**
- → **I casi di test configurano il proprio ambiente evitando di personalizzare la build per la preparazione dei test**

Maven suggerisce anche alcune linee guida su come strutturare le directory del progetto. Una volta appreso il layout, è facile navigare in altri progetti che utilizzano Maven.

Introduzione a Maven

Gestione progetti Java

Esercitazione.

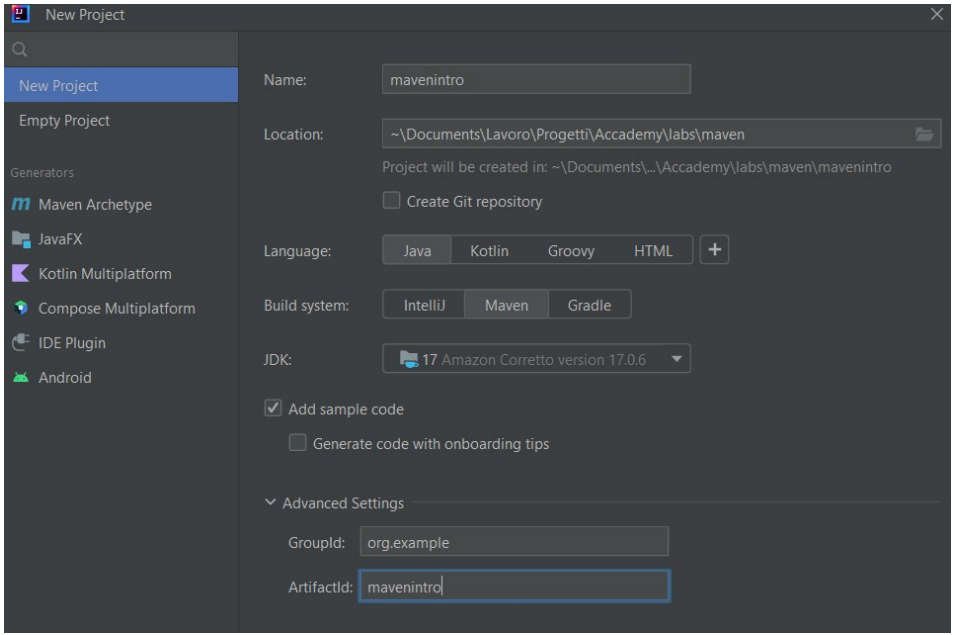
Creazione di un progetto Maven con **IntelliJ**

→ **POM**

→ **Modifica classe Main**

→ **Inserimento dipendenza**

→ **goal package**



Introduzione a Tomcat e HTTP

Introduzione a Tomcat e HTTP

Cosa è tomcat?

In poche parole, Apache Tomcat è un **server web** e **servlet container** utilizzato per distribuire e servire applicazioni Web scritte in Java.

Tomcat è una implementazione open source delle specifiche:

- → **Jakarta Servlet**
- → **Jakarta Server Pages**
- → **Jakarta Expression Language**
- → **Jakarta WebSocket**
- → **Jakarta Annotations**
- → **Jakarta Authentication**

Una servlet è una classe che gestisce le richieste HTTP, le elabora e invia una risposta al chiamante.

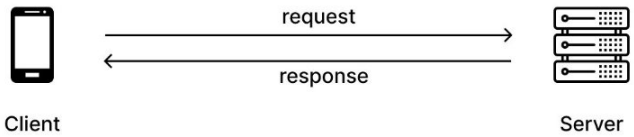
Il software Apache Tomcat viene utilizzato in numerose applicazioni web mission-critical su larga scala in una vasta gamma di settori e organizzazioni.

Introduzione a Tomcat e HTTP

Cosa è HTTP?

Senza HTTP (Hypertext Transfer Protocol), il World Wide Web come lo conosciamo oggi non esisterebbe. HTTP è il protocollo che consente il trasferimento di dati su Internet, permettendo agli utenti di accedere a siti Web e ad altre risorse online.

In HTTP la comunicazione è basata sul **ciclo request-response**:



Per creare una richiesta HTTP valida occorre:

→ **URL**

→ **Metodo HTTP**

→ **Una lista di headers**

→ **Il corpo della richiesta (request body)**

Per esempio:

```
GET /watch?v=8PoQpnlBXD0 HTTP/1.1
Host: www.youtube.com
Cookie: GPS=1; VISITOR_INFO1_LIVE=kOe2UTUyPmw; YSC=Jt6s9YVWMd4
```

Introduzione a Tomcat e HTTP

Cosa è HTTP?

Il metodo HTTP (detto anche *verbo*) dice al server che tipo di azione il client vuole che il server intraprenda. I metodi più comuni sono:

HTTP METHODS	DEFINITION
HEAD	Asks the server for status (size, availability) of a resource.
GET	Asks the server to retrieve a resource.
POST	Asks the server to create a new resource.
PUT	Asks the server to edit/update an existing resource.
DELETE	Asks the server to delete a resource.

La **response** inviata dal server a fronte di una request è composta da un **codice di stato**, **headers** e **body**. Per esempio:

```
HTTP/1.1 200 OK
Date: Sun, 28 Mar 2023 10:15:00 GMT
Content-Type: application/json
Server: Apache/2.4.39 (Unix) OpenSSL/1.1.1c PHP/7.3.6
Content-Length: 1024

{
    "name": "John Doe",
    "email": "johndoe@example.com",
    "age": 30,
    "address": {
        "street": "123 Main St",
        "city": "Anytown",
        "state": "CA",
        "zip": "12345"
    }
}
```

Introduzione a Tomcat e HTTP

Cosa è HTTP?

Il codice di stato comunica l'esito della richiesta. I codici più comuni sono:

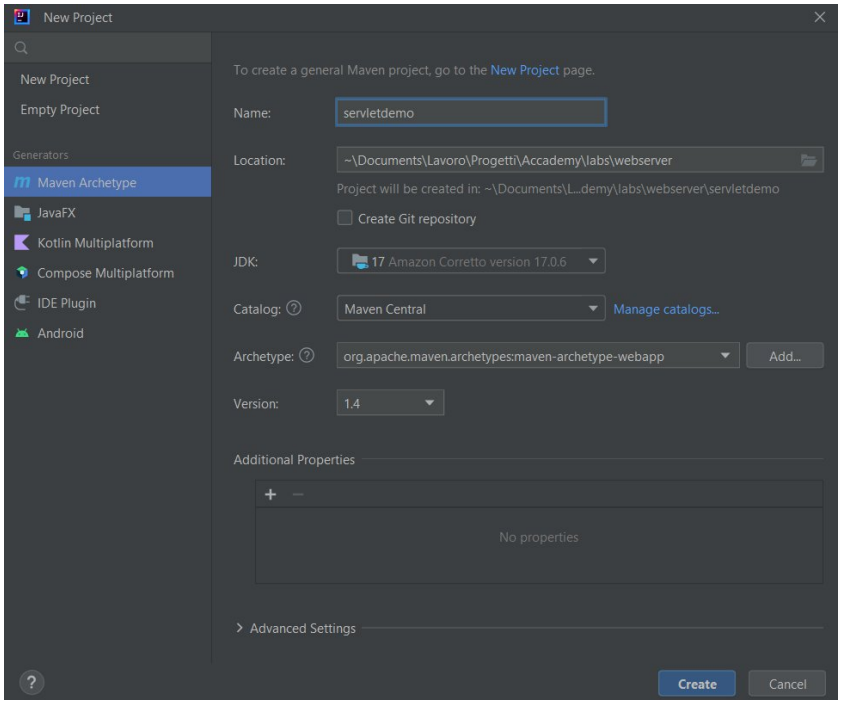
CODE	MEANING
100	Continue
101	Switching Protocols
200	OK
201	Created
202	Accepted
203	Non-Authoritative Information
404	Not Found : The requested resource was not found on the server.
500	Internal Server Error : The server encountered an error while processing the request.
301	Moved Permanently: The requested resource has been permanently moved to a new URL.

Introduzione a Tomcat e HTTP

Esercitazione.

Creazione di un progetto Maven con **IntelliJ**

- → **Modifica POM**
- → **Creazione servlet**
- → **Download tomcat**
- → **modifica conf/tomcat-users.xml**
- → **Deploy del progetto**



Introduzione a SpringBoot

SpringBoot in 2 parole

Introduzione a SpringBoot

SpringBoot in 2 parole

Cosa è Spring?

Framework per lo sviluppo di applicazioni Java, molto usato per creare applicazioni web. Include molti moduli che forniscono una varietà di servizi; ad esempio:

→ **Spring Core**: modulo di base che contiene il container **IoC/DI**

→ **Spring Security:** modulo per autenticazione e autorizzazione

→ **Spring Data**: modulo per l'accesso ai database

→ **Spring MVC**: modulo per la creazione di applicazioni web e servizi RESTful

Inversion of Control (IoC) e Dependency Injection (DI).

Inversion of Control è un principio dell'ingegneria del software dove il framework assume il controllo del flusso del programma e invoca il nostro codice custom. Questo approccio è alternativo a quello tradizionale dove è il nostro codice custom che invoca le librerie.

Dependency injection è il pattern utilizzato da Spring per implementare IoC dove ciò che viene invertito è l'inserimento delle dipendenze in un oggetto.

In Spring, l'interfaccia **ApplicationContext** rappresenta il container IoC che è responsabile della creazione, della configurazione e dell'assemblaggio di oggetti noti come **bean**, nonché della gestione del loro ciclo di vita.

L'inserimento delle dipendenze in Spring può essere eseguito tramite costruttori, setter o campi.

Introduzione a SpringBoot

SpringBoot in 2 parole

Cosa è SpringBoot?

Soluzione per creare in maniera più rapida e semplice applicazioni Spring, evitando di specificare manualmente le configurazioni dei moduli utilizzati nel progetto (**convention over configuration**) e sfruttando gli **starter** per impostare le dipendenze necessarie al progetto.

→ **Convention over configuration**: meno configurazioni da impostare manualmente, già settate di default

→ **Starter:** aggiungere e gestire dipendenze in maniera semplice

Introduzione a SpringBoot

SpringBoot in 2 parole

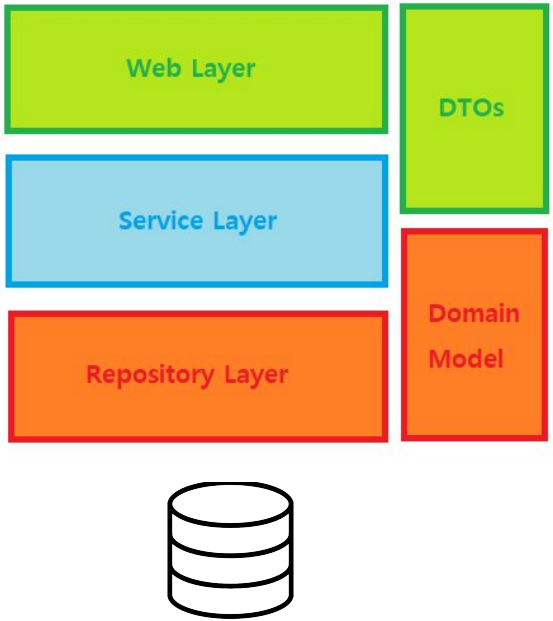
Struttura di una applicazione SpringBoot

L'applicazione è composta da livelli; ogni livello ha la propria responsabilità e comunica con i livelli sottostanti tramite interfacce ben definite. I livelli sono disaccoppiati.

Il **Web Layer** espone al mondo esterno le funzionalità dell'applicazione tramite i controllers. Controllers e services comunicano utilizzando i **DTO**s (**D**ata **T**ransfer **O**bject).

Il **Service Layer** implementa la logica di business. Comunica con il livello sottostante utilizzando gli oggetti del modello di dominio.

Il **Repository Layer** offre i servizi per l'accesso al database sottostante.



REST API

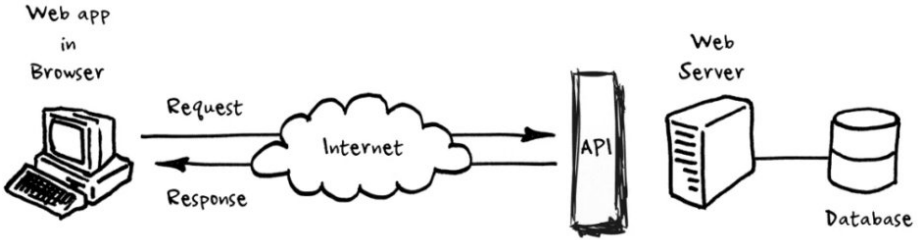
REST API

Cosa sono le API?

Le **API** (acronimo di **Application Programming Interface**, ovvero Interfaccia di programmazione delle applicazioni) sono un insieme di regole e protocolli con i quali vengono realizzati e integrati software applicativi. Possono essere considerate come un contratto tra un fornitore di informazioni e l'utente destinatario di tali dati. Ne è un esempio la struttura di un'API di un servizio meteorologico, in cui l'utente invia una richiesta contenente un codice postale al quale il produttore fornisce una risposta contenente la temperatura massima e la temperatura minima.

Una **REST API**, nota anche come **RESTful** API, è una API conforme allo stile architetturale **REST** (**RE**presentational **S**tate **T**ransfer).

REST è uno stile architetturale - presentato da Roy Fielding nella tesi di dottorato *Architectural Styles and the Design of Network-based Software Architectures* - ampiamente utilizzato per lo sviluppo di Web services.



REST API

Regole per lo sviluppo di una **“buona”** REST API

1

Seguire una Naming Conventions per l’URI

Convenzione	Esempio
Usare lettere minuscole e nomi plurali. Non usare verbi	/users /orders
Usare il trattino per separare le parti del nome	/managed-devices/{device_id}
Usare l’underscore per separare le parti del nome nelle query strings	/users?sort_by=name /orders?filter_by_status=completed
Utilizzare una gerarchia di risorse per le relazioni. Limitare la profondità (max 3 livelli)	/users/{user_id}/orders

REST API

Regole per lo sviluppo di una **“buona”** REST API

2

Utilizzare i metodi HTTP per comunicare l'intento

La tabella a fianco mostra i metodi più utilizzati.

Come si può vedere, le operazioni REST API sono molto simili alle operazioni **CRUD** dei database.

Metodo HTTP	Scopo
GET	Recupera una risorsa
POST	Crea una risorsa; può anche essere utilizzato per operazioni di ricerca
PUT	Aggiorna una risorsa
PATCH	Aggiorna parzialmente una risorsa
DELETE	Cancella una risorsa

REST API

Regole per lo sviluppo di una **“buona”** REST API

3

Ove possibile, usare il formato JSON per inviare e ricevere i dati

```
GET /articles?include=author HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/vnd.api+json

{
  "data": [{
    "type": "articles",
    "id": "1",
    "attributes": {
      "title": "JSON:API paints my bikeshed!",
      "body": "The shortest article. Ever.",
      "created": "2015-05-22T14:56:29.000Z",
      "updated": "2015-05-22T14:56:28.000Z"
    },
    "relationships": {
      "author": {
        "data": {"id": "42", "type": "people"}
      }
    }
  }],
  "included": [
    {
      "type": "people",
      "id": "42",
      "attributes": {
        "name": "John",
        "age": 80,
        "gender": "male"
      }
    }
  ]
}
```

REST API

Regole per lo sviluppo di una **“buona”** REST API

4

Seguire una Naming Conventions per i campi JSON

Le prime due convenzioni sono quelle più usate.

Convenzione	Esempio
snake_case	user_name
camelCase	userName
PascalCase	UserName
kebab-case	user-name

REST API

Regole per lo sviluppo di una **“buona”** REST API

5

Usare i codici di risposta previsti da HTTP

La tabella a fianco mostra i codici più comuni.

HTTP Status Code	Descrizione
200 OK	La richiesta ha avuto esito positivo e la risposta contiene i dati richiesti
201 Created	La richiesta ha avuto successo ed è stata creata una nuova risorsa
204 No Content	La richiesta è andata a buon fine ma non vi è alcuna rappresentanza da restituire
400 Bad Request	La richiesta non è valida e non può essere soddisfatta
401 Unauthorized	La richiesta richiede l'autenticazione e l'utente non ha fornito credenziali valide
403 Forbidden	Il server comprende la richiesta, ma rifiuta di autorizzarla
404 Not Found	Non è stata trovata la risorsa richiesta
500 Internal Server Error	Errore generico non meglio specificato

REST API

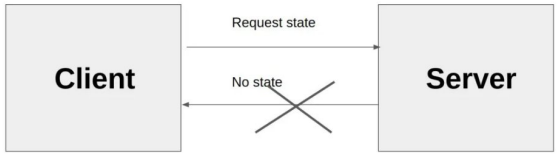
Regole per lo sviluppo di una **“buona”** REST API

6

Condizione di stateless

Le REST API sono stateless, il che vuol dire che ogni richiesta deve includere tutte le informazioni necessarie per elaborarla. Il server non conserva alcuna informazione sulle richieste precedenti del client e non memorizza alcun dato di sessione o cronologia relativa al client.

Questo è importante perché consente all'API di essere cache-abile, scalabile e disaccoppiata dal client.



REST API

Regole per lo sviluppo di una **“buona”** REST API

7

Versionare le APIs

Il controllo delle versioni dell'API è importante perché consente la compatibilità con le versioni precedenti, consente l'introduzione di nuove funzionalità senza interrompere i client esistenti e consente l'implementazione graduale delle modifiche.

La tabella a fianco mostra varie tecniche di versionamento.

Il primo metodo è quello più utilizzato.

Metodo versionamento	Esempio
URL path	/api/v1/users
URL query parameter	/api?version=1
Custom header nella richiesta	X-API-Version: 1
Utilizzo dell’header Accept	Accept: application/vnd.sidea.api-v1+json

REST API

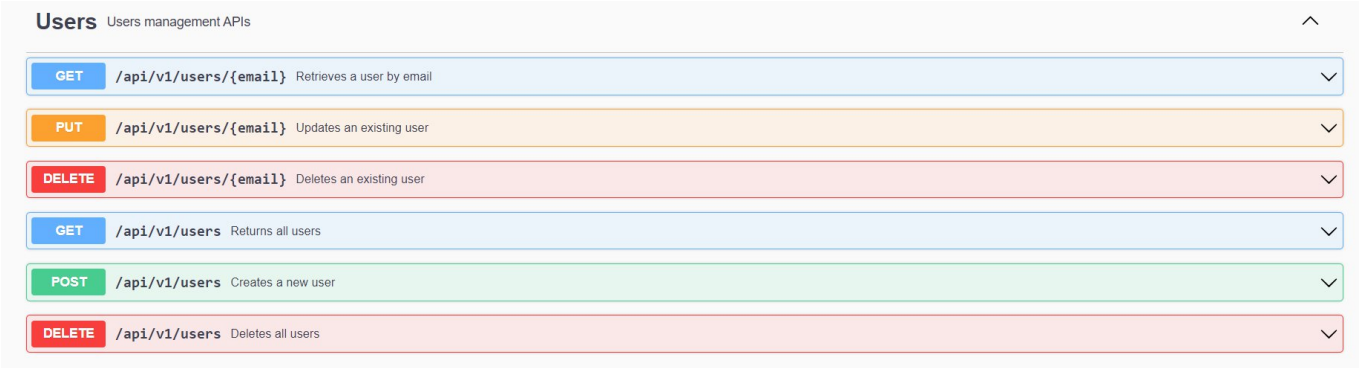
Regole per lo sviluppo di una **“buona”** REST API

8

Documentare le APIs

Si usa lo standard

OpenAPI Specification



REST API

Regole per lo sviluppo di una **“buona”** REST API

9

Usare messaggi di errore coerenti

Il codice di errore spesso non è sufficiente per comunicare al client il motivo del fallimento. Sarebbe opportuno fornire un JSON di risposta con ulteriori dettagli.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errorCode": "INVALID_INPUT",
    "message": "The input provided is invalid.",
    "details": [
        {
            "field": "firstName",
            "issue": "required field is missing"
        },
        {
            "field": "lastName",
            "issue": "required field is missing"
        }
    ],
    "request_id": "5124129901",
    "timestamp": "2022-12-10T13:46:20.857Z"
}
```

REST API

Regole per lo sviluppo di una **“buona”** REST API

10

Usare filtri, ordinamento e paginazione per recuperare i dati richiesti

A volte, il database di un'API può diventare incredibilmente grande. Se ciò accade, il recupero dei dati da un database di questo tipo potrebbe essere molto lento. I filtri, l'ordinamento e la paginazione sono tutte azioni che permettono di recuperare, ordinare e organizzare i dati in pagine in modo che il server non sia troppo occupato dalle richieste.

```
https://api.twitter.com/2/users/2244994945/tweets?tweet.fields=created_at&max_results=100&start_time=2019-01-01T17:00:00Z&end_time=2020-12-12T01:00:00Z
```

REST API

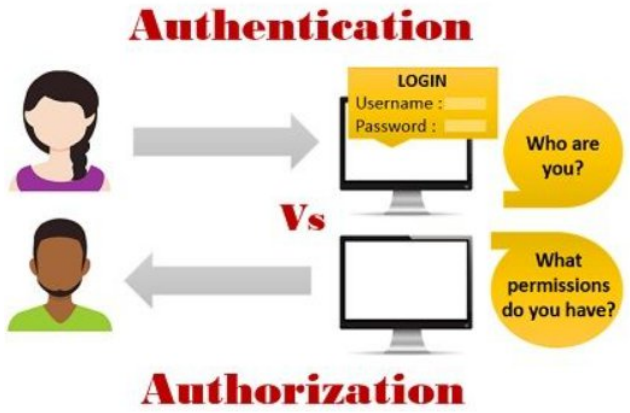
Regole per lo sviluppo di una **“buona”** REST API

11

Sicurezza: TLS + Autenticazione/Autorizzazione

Transport Layer Security (**TLS**) e il suo predecessore **Secure Sockets Layer** (**SSL**) sono dei protocolli crittografici che permettono una comunicazione sicura dalla sorgente al destinatario.

Inoltre deve essere utilizzato un meccanismo di **autenticazione/autorizzazione**.



SpringBoot in pratica

SpringBoot in pratica

Controller & Service

Esercitazione.

In questa esercitazione creeremo un progetto SpringBoot e approfondiremo i concetti di controller, service, inserimento delle dipendenze, disaccoppiamento e divisione delle responsabilità.

Project

○ Gradle - Groovy
○ Gradle - Kotlin ● Maven

Language

● Java ○ Kotlin ○ Groovy

Spring Boot

○ 3.2.0 (SNAPSHOT) ○ 3.2.0 (M3) ○ 3.1.5 (SNAPSHOT) ● 3.1.4
○ 3.0.12 (SNAPSHOT) ○ 3.0.11 ○ 2.7.17 (SNAPSHOT) ○ 2.7.16

Project Metadata

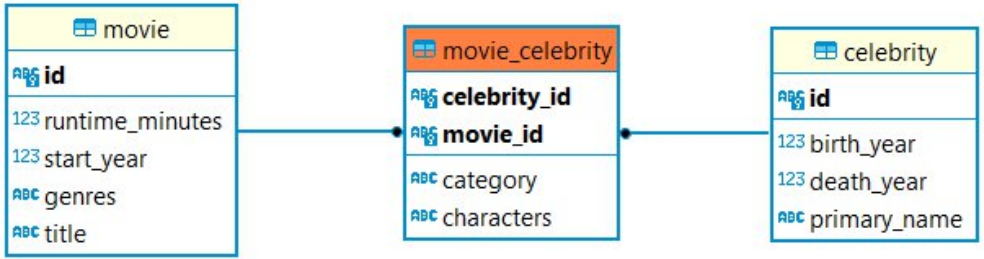
Group	com.sideagroup
Artifact	imdbclone
Name	imdbclone
Description	Learn Spring Boot
Package name	com.sideagroup.accademy
Packaging	● Jar ○ War
Java	○ 21 ● 17 ○ 11 ○ 8

SpringBoot in pratica

Spring Data JPA

Esercitazione.

In questa esercitazione permetteremo alla nostra applicazione di accedere al database tramite il modulo Spring Data JPA. La figura a lato mostra il modello dati che utilizzeremo.



SpringBoot in pratica

Documentare le API

Esercitazione.

In questa esercitazione vedremo come documentare le nostre API secondo le specifiche OpenApi.

Riferimenti

Per approfondire...

Riferimenti

Per approfondire...

- → https://maven.apache.org/index.html
- → https://www.jetbrains.com/idea/
- → https://en.wikipedia.org/wiki/Spring_Framework
- → https://tomcat.apache.org/ 
- → https://www.freecodecamp.org/news/what-is-http/
- → https://it.wikipedia.org/wiki/Roy_Fielding
- → https://spring.io/tools 
- → https://www.postman.com/ 
- → https://docs.spring.io/spring-boot/docs/current/reference/html/
- → https://docs.spring.io/spring-data/jpa/docs/current/reference/html/
- → https://www.youtube.com/@lesstheoryacademy6051
- → https://developer.imdb.com/non-commercial-datasets/
- → https://www.openapis.org/

Grazie

Tel 080 4391385
E-mail info@sideagroup.com
Pec sideagroup@pec.it

sideagroup.com