INDICE DEI CONTENUTI

* INTRODUZIONE
  + DI e IOC
  + Spring boot vs Spring MVC
  + Caso d’uso (recensioni su articoli di giornale)
    - Descrizione della base dati
* CREAZIONE DEL PROGETTO
  + Bootstrapping Spring project Boilerplate (Spring Initilizer)
  + Importazione del progetto in STS
  + @SpringBootApplication
* MODEL LAYER
  + Le entità in Spring
  + Principali annotations
    - @Entity
    - @Table
    - @Column
    - @Enumeration
    - @Temporal
    - @OneToOne
    - @OneToMany
    - @ManyToOne
    - @ManyToMany
  + Il Cascade

Spring boot basic course - Learn by doing approach

Partiamo da un esempio pratico:

Si vuole realizzare un semplice sistema di feedback per una serie di articoli di giornale

Partiamo dello strato di persistenza:

per mantenere il discorso quanto più semplice possibile astraiamo la base dati come segue:

prendiamo in considerazione le seguenti entità: user, article, feedback

USER può astrarre in se sia l'autore dell'articolo che il recensore, e per non complicarci la vita assumiamo che un utente può essere autore o recensore ma non entrambi. diciamo che può essere composto dai seguenti campi:

* id
* type: author | reviewer
* name

ARTICLE rappresenta un articolo di giornale che possiamo rappresentare come:

* id
* author\_id
* type: attualità | cronaca | politica...
* publish\_date
* title
* body

FEEDBACK è abbastanza esplicativo, diciamo che per semplicità può contenere i seguenti campi:

* id
* reviewer\_id
* article\_id
* publish\_date
* vote: [1,5]
* comment

Ora che abbiamo fissato la base dei dati possiamo passare all'implementazione. Partiamo dal modello. in questa fase seguiremo due strade equivalenti una prevede la creazione delle entità in maniera "manuale", l'altra prevede l'utilizzo di hibernate per il forward ed il reverse engineering.

Per lo sviluppo utilizzeremo spring tool suite nella versione 4,

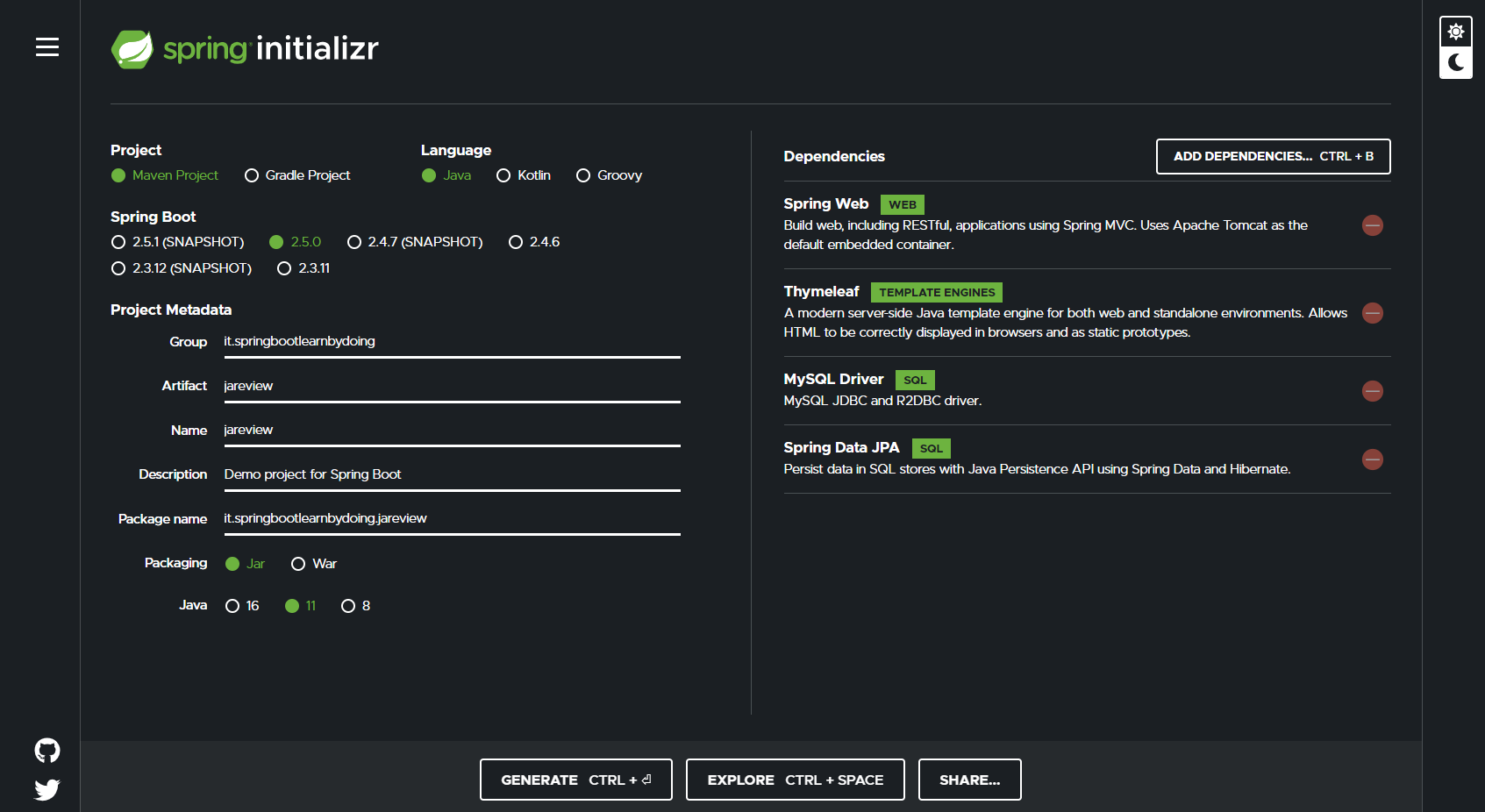
Iniziamo con il creare il progetto spring. Per semplificarci la vita utilizziamo il tool online <https://start.spring.io/>. Questo permette di creare un progetto spring boot boilerplate utilizzando maven o gradle. Permette di scegliettere il linguaggio (nel nostro caso java), la versione di spring boot , il tipo di packaging per la build e la versione di java (nel nostro caso utilizziamo la 11). Inoltre permette di aggiungere già in questa fase le dipendeze di cui avremo bisogno.

Iniziamo con i primi settaggi:

* Project -> Maven Project
* Language -> Java
* Spring Boot -> 2.5.0
* Packaging -> War
* Java -> 11
* Group: it.springbootlernbydoing
* Artifact = Name: jareview
* Description: Va più che bene quella di default
* Package name: it.springbootlearnbydoing.jareview

Come dipendenze scegliamo:

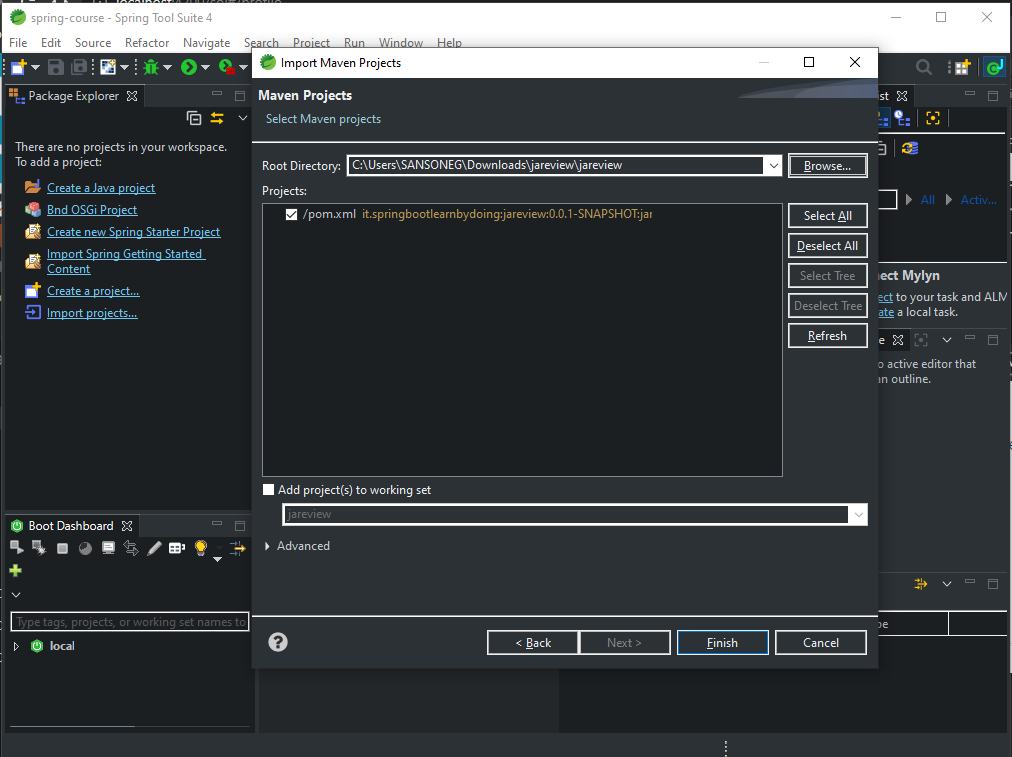
* Spring web -> aggiunge il modulo rest, Spring MVC
* Thymaleaf -> è un template engine server side (faremo qualche esempio di utilizzo)
* MySQL Driver -> mi sembra abbastanza autoesplicativo
* Spring Data JPA -> per lo strato di repository



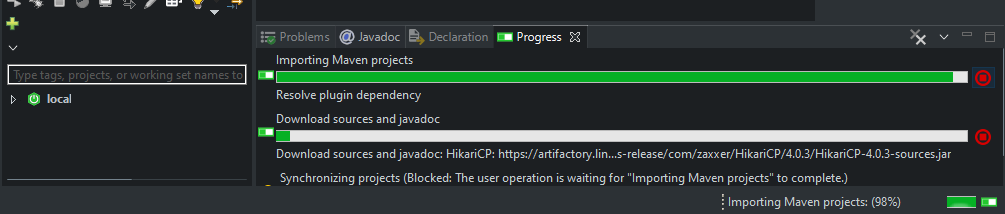
A questo punto clicchiamo "GENERATE" ed otteniamo un zip contenente il nostro progetto boilerplate.

Possiamo iniziare ad importare il nostro progetto all'interno dell'ide. Apriamo STS è creiamo un nuovo Workspace.

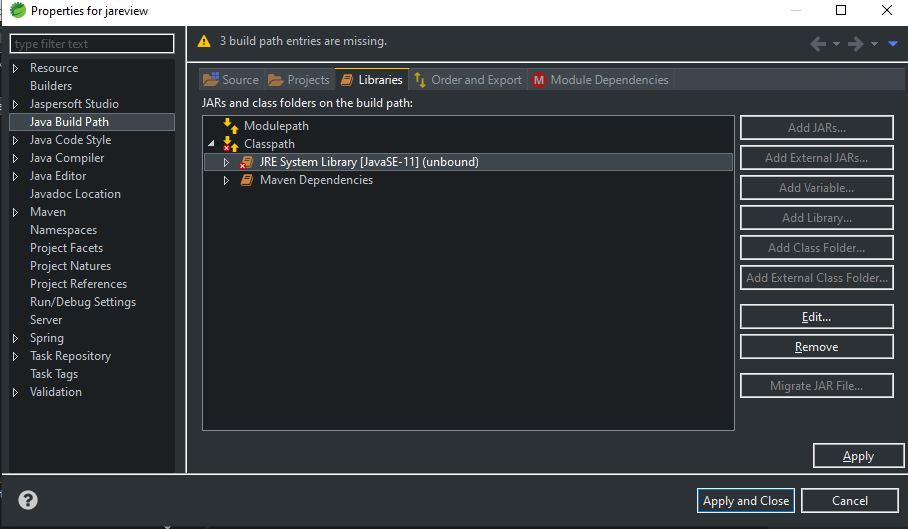
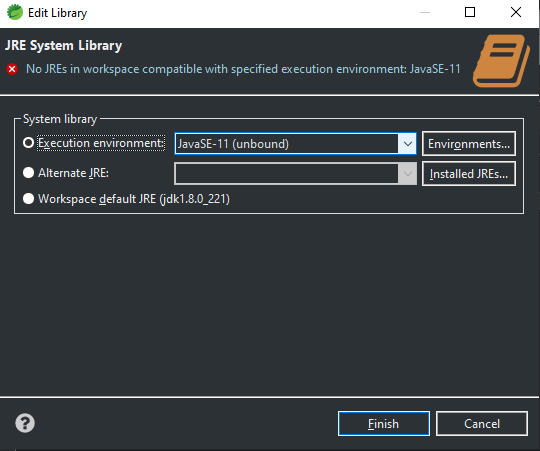
Scompattiamo lo zip precendentemente scaricato ed importiamolo su STS come "Existing Maven Projects"

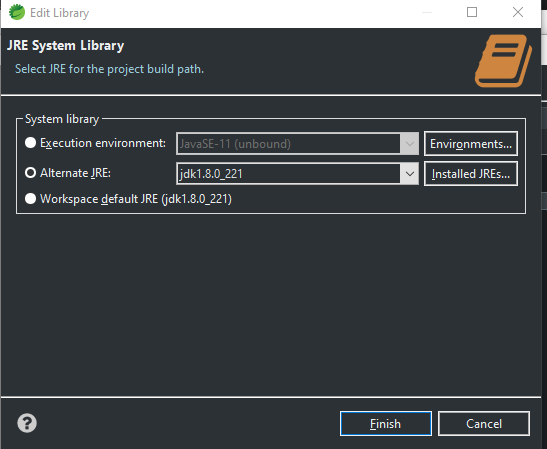


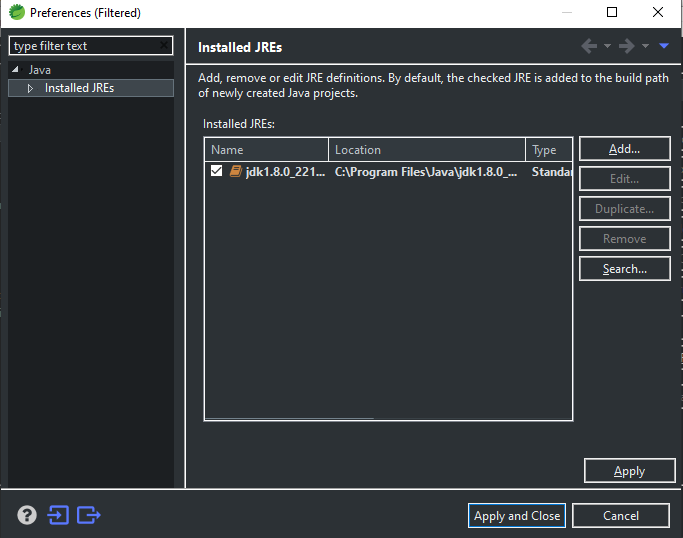
A questo punto aspettiamo che STS importi il progetto e si scarichi tutte le sue dipendenze

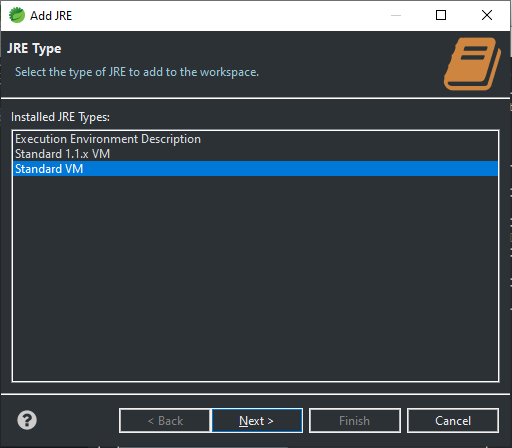


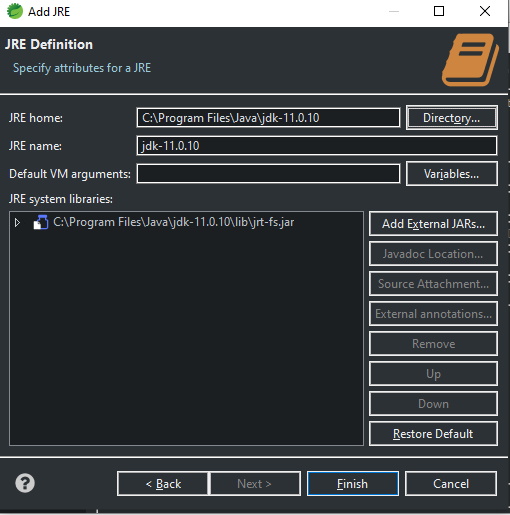
Abbiamo detto che per questo progetto useremo JAVA 11 per cui accertiamoci che eclipse utilizzi il corretto buildPath

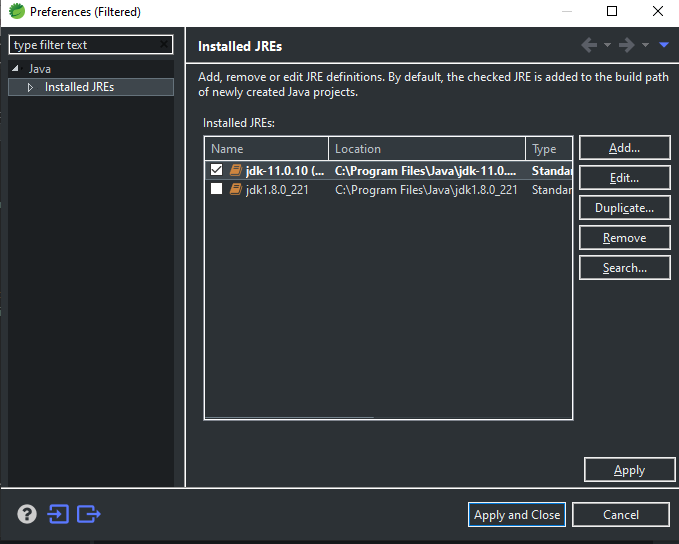


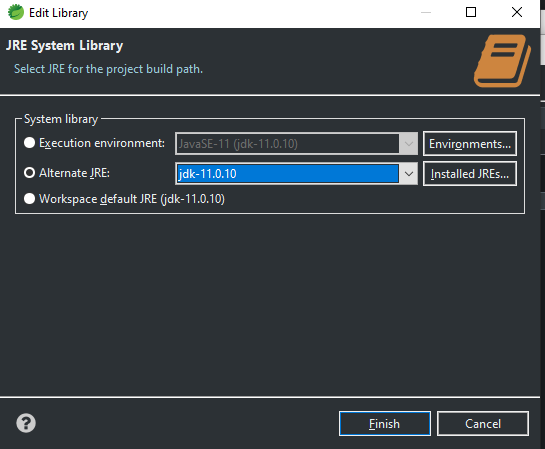








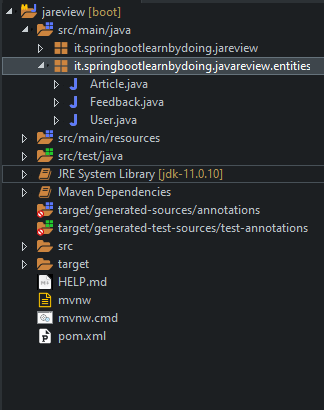




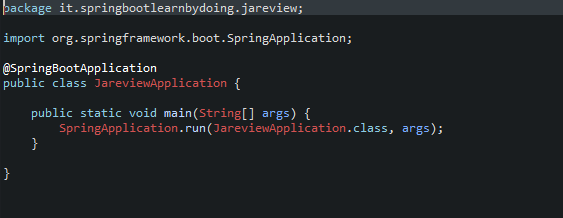
Diamo il tempo a STS di ribuildare il progetto

Iniziamo a creare lo scaffolding:

Creaiamo il package "it.springbootlearnbydoing.javareview.entities" che conterrà il nostro strato di model, quindi il mapping delle entità User, Article, Feedback; andiamo quindi a creare queste 3 classi all'interno del nuovo package.



Il package base "it.springbootlearnbydoing.javareview" conterrà la classe che rappresenta l'entry point per l'applicazione spring boot



La classe è annotata come @SpringBootApplication, ciò sta ad indicare che la classe in questione contiene il metodo main per l'applicazione Spring boot. Ogni classe che mappa un'entità della base dei dati deve essere annotata con @Entity ed implementare l'interfaccia Serializable. Ad esempio, un possibile mapping per User potrebbe essere:



In questo momento non è necessario implementare l’interfaccia Serializable ma sarà comunque propedeutica per la parte REST.

L’annotation @Table indica il nome della tabella a cui è associata l’entità, non è necessaria se la tabella ha lo stesso nome della classe che la mappa.

Per quanto riguarda i campi con @Id si indica l’attributo che corrisponde alla chiave primaria; mentre con @GeneratedValue(strategy = GenerationType.AUTO) stiamo dicendo che quel campo viene generato automaticamente. Questo sarà necessario per l’operazione di creazione, in quanto non sarà necessario settare l’id a livello di entità (lo vedremo più avanti).

Con @Column effettuiamo la mappatura di colonna della tabella: name è il nome della colonna, length è la lunghezza massima del valore, nullable se il campo può essere un valore nullo o meno.

Con @Enumerated diciamo che quell’attributo deve essere mappato appunto come un Enum a valori di tipo stringa.

È necessario specificare sempre un costruttore di default vuoto e senza parametri per le vecchie versioni di Java.

