**GIORNO 02**

**REST API IN SPRING BOOT**

Repo del giorno: https://github.com/nclBaz/epicode-it-u5-d7-2

**Spring Web** ci viene fornito con un Web Server già pronto all’uso in maniera automatica. Un Web Server non è altro che una applicazione che gira all’infinito aspettando che arrivino delle richieste. Arrivata la richiesta, il Web Server la elabora e invia una risposta. Per ogni richiesta viene creato un nuovo thread che si occuperà di gestire quella richiesta. In questo modo possiamo gestire più richieste parallelamente e, in caso di una richiesta più esosa, avremo un netto risparmio sulle prestazioni.

Le API sono insiemi di funzionalità o librerie fornite da un servizio o framework. Le Web API, in particolare, sono funzionalità accessibili tramite chiamate HTTP/HTTPS. Le Web API offrono endpoint, che permettono di eseguire operazioni distinte, ad esempio cercare un utente per ID, modificarlo o eliminarlo. Posso utilizzarle senza sapere il “come” agiscono perché ci ha pensato uno sviluppatore. Questo spiega la INTERFACE di API: io interagisco con una interfaccia di cui conosco la funzionalità, ma non il “come” è costruita. Un esempio è il ristorante: noi siamo il CLIENT a cui viene dato un Menù (FRONT END). Presa l’ordinazione (tramite il Cameriere che rappresenta la Richiesta), questa verrà gestita dalla Cucina (Server-> BACK END) e in seguito l’ordinazione viene gestita, elaborata e riportata al Client (abbiamo la Risposta). Nel contesto delle REST API, il frontend rappresenta il client, che interagisce con l'API tramite richieste HTTP. Il backend è il server che elabora le richieste e restituisce le risposte.

Immagine che contiene testo, Carattere, linea, diagramma

Descrizione generata automaticamenteIn Java esiste il concetto di Applicazione Servlet, cioè Oggetti che servono per gestire tutti i metodi http (GET, POST, ecc…) in maniera multithreaded (per ogni richiesta avremo un thread che gestisce quella richiesta parallelamente ad altri thread, creati per gestire altre richieste). Tutto questo viene eseguito in maniera automatica da Spring Web. Servlet ci permette di aggiungere funzionalità custom al nostro Server ed è auto configurato per noi da Spring Web stesso. Possiamo pensare come ad un portale che accetta tutte le richieste e le smaltisce: è un Front controller.

Il Controller è un insieme di endpoint specifici per ogni Collection che abbiamo da gestire (avremo un Controller per gestire tutte le richieste per gli User, ad esempio, oppure per gestire tutti i Products, ecc…).

Immagine che contiene testo, software, Icona del computer, Pagina Web

Descrizione generata automaticamentePer configurare Spring Web….

…in alternativa possiamo utilizzare spring.initialiazr.

Immagine che contiene testo, schermata, cartone animato

Descrizione generata automaticamente

Per configurare Spring Web, si inseriscono Component (annotati con @Controller) che si occupano di ricevere le richieste e smistarle ai diversi metodi corrispondenti agli endpoint. Il Controller accetta le richieste che arrivano dal FE (esempio applicazione Angular), la gestisce tramite un metodo definito dal Service specifico di una Collection che si interfaccia col Repository. Questo andrà a manipolare il DB e ci restituirà la risposta.

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteNelle REST API, i Model rappresentano le entità che vengono manipolate, come oggetti nel database. In alternativa possiamo dividere tutto in base alle Collection con cui andremo aderagire (esempio potremo avremo il package User al cui interno avremo service,repo,Classe, ecc…)

Immagine che contiene testo, schermata, Carattere, algebra

Descrizione generata automaticamente

Il Servlet instraderà la richiesta che arriva allo specifico Controller che abbiamo creato.

Ogni Controller sarà padrone di un particolar url (/user, /produtc/, ecc…).

Nel Controller avremo tutta una serie di metodi che corrisponde ad un endpoint( avremo dei @GetMapping, @PostMapping, ecc…). In sintesi i Mapping sono padroni di un endpoint specifico(url+metodo):

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

I metodi all'interno del Controller sono annotati con @GetMapping, @PostMapping, ecc., per specificare il tipo di richiesta HTTP che gestiscono.

Immagine che contiene testo, Carattere, schermata, algebra

Descrizione generata automaticamente

La base url sarà comune a tutti:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente<http://localhost(portaDelDB)/nomeDB>. La parte finale viene gestita dal particolare Controller e dai suoi metodi contrassegnati dalla particolare annotazione di Mapping.

Se immaginiamo di avere un Controller per gestire dei Model User, allora ------------------------------------------🡪

…dunque alla base della url si aggiunge un pezzettino, definito dal Mapping particolare.

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Il RestController è una alternativa al Controller e che rende superfluo il @ResponseBody dell’esempio di prima.

Mani sul codice (non lavoriamo sul DB quindi niente Spring Data):

1. Creazione del progetto e importazione in Eclipse:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamenteSpring Boot Dev Tools contiene LiveReload che aggiorna automaticamente il browser ogni volta che apporti modifiche al codice o alle risorse dell'applicazione, rendendo il processo di sviluppo più immediato.

1. Avviando l’applicazione notiamo che ci viene fornito il Web Server Apache Tomcat ed una porta (8080) che potremo configurare.
2. Immagine che contiene testo, schermata, software, Software multimediale

   Descrizione generata automaticamenteIn Postman facendo una GET con <http://localhost:8080> riceviamo già una risposta di errore:
3. Immagine che contiene testo, Carattere, schermata, design

   Descrizione generata automaticamenteConfiguriamo la porta nell’apllication.properties e assicurarsi che non sia escluso:
4. Immagine che contiene testo, schermata, software, Software multimediale

   Descrizione generata automaticamenteCreiamo un ExamplesController annotata con RestController che serve per definire una “collezione” di enpoints. Definiamo anche tutti i metodi di Mapping, e per ognuno avremo un endpoint specifico (url+metodo).

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Tra le parentesi nei vari Mapping specifico la parte finale dell’url.

Se voglio che tutte gli enpoints di un Controller abbiano un prefisso comune, allora uso @RequestMapping:

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Analogamente possiamo definire un Post

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente@PostMapping può avere la stessa parte finale che abbiamo già definito (in particolare abbiamo definito “index” per GetMapping), purchè sia un metodo diverso altrimenti avremmo uno stesso endpoint definito in maniera differente nei due metodi.

Immagine che contiene testo, Carattere, ricevuta, schermata

Descrizione generata automaticamenteGli endpoint hanno bisogno di Param. I parametri possono essere specificati con il Body, queryParams e pathParams.

Per i QueryParams c’è una annotazione specifica:

Immagine che contiene testo, schermata, Carattere, algebra

Descrizione generata automaticamenteGli specifici parametri che mi serviranno vengono contrassegnati con @requestParam all’interno dei parametri accettati dal metodo.

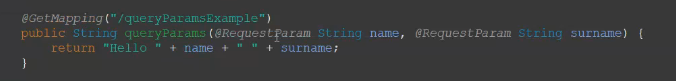


Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamentePossiamo definire meglio il metodo attraverso dei controlli specifici che controllano che ci sia un name e un surname.

Eventualmente un Param specifico può anche non essere specifico (required=false).

PathParams

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Definire i params in questo modo è conveniete perché potrò usare, invece di paramer, un id per un particolare oggetto di una particolare Collection (esempio “users”) e farmelo ritornare semplicemente chiamando quel metodo e passandogli l’id come paramentro (int, long, UUID, ecc….).

Definiamo una Classe User:

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata

Descrizione generata automaticamentee nella Classe ExamplesController definiamo un metodo Post che si aspetta un body specifico di tipo User:

Immagine che contiene testo, software, Software multimediale, schermata

Descrizione generata automaticamente

In PostMan dovrò definire un oggetto User di tipo Json (l’id viene creato da solo) che definisce il Body della richiesta POST definita dal metodo payload. In seguito all’interno di questo metodo posso andare a definire i metodi Dao per interagire col DB.

Immagine che contiene testo, Carattere, algebra, schermata

Descrizione generata automaticamentePossiamo definire gli Status Code delle risposte: @ResponseStatus

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamenteHttpStatus è un Enum di riferimento con cui posso accedere a tutti gli status code. Con Created mi ritorna un 201

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamenteGli status code si dividono in 3 famiglie:

200 - tutto ok

400 – errore per colpa del Client

500 – errore per colpa del Server

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

ResponsEntity permette di customizzare le risposte e di farci ritornare un json.

Immagine che contiene testo, schermata, Carattere, algebra

Descrizione generata automaticamenteIn questo caso ci ritorna una String, ma potremmo farci ritornare un User.

Immagine che contiene testo, schermata, Carattere, Software multimediale

Descrizione generata automaticamente

In questo caso possiamo farci ritornare una new ResponseEntity che contiene un body, header e status code.

Immagine che contiene testo, schermata

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteNel Controller potremo richiamare un Service grazie all’Autowired.

Creiamo ora la Classe UserService con i metodi CRUD da utilizzare nel DB (simula la presenza di un DB con una List users) …diversamente avremmo dovuto definire un UserRepo e creare un DB.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamenteCreiamo un UsersController con RequestMapping(“/users”)-> SEMPRE AL PLURALE!

Questa Classe avrà tutti i metodi Mapping che avranno propri endpoints specifici.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Tutti gli endpoints hanno già “users” perché fornito a monte da RequestMapping. Questi metodi hanno bisogno di Autowired dell’UserService dove abbiamo definito i metodi CRUD

Immagine che contiene testo, software, Software multimediale, schermata

Descrizione generata automaticamente

Testiamo:

Immagine che contiene schermata, software, Software multimediale, testo

Descrizione generata automaticamente

Definiamo il metodo 2 che ci ritorna una List di User filtrata grazie alla query params:

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

Definiamo il metodo 3:

Immagine che contiene testo, schermata, Carattere

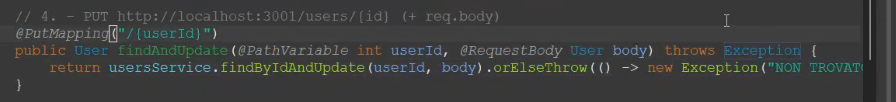
Descrizione generata automaticamente

…e testiamo

Immagine che contiene schermata, software, testo, Software multimediale

Descrizione generata automaticamente

Definiamo metodo 4



Definiamo metodo 5

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Con Delete non ci facciamo ritornare nulla.

Tutto ci riassume nell’aver 5 endpoint per i metodi CRUD.

TEST

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

Testare un’applicazione Web significa eseguire dei metodi che facciano delle richieste http testandoli.

Spring mi da una Classe di Test avviabile. Con Autowired chiamo MockMvc.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Mi aspetto che la risposta sia Ok, cioè 200.