SOURCESENSE	**Redatto da:** Valerio Bondì
	14 Settembre 2022
Oggetto	**Stato**
Documentazione progetto SourcesenseAssignment	Final Draft

Contents

Introduzione

Il progetto **SourcesenseAssignment** consiste in una semplice applicazione Spring Boot che permette l'aggregazione di notizie da 2 sorgenti in particolare: **Hacker News** e **New York Times**.

Al fine di facilitare la lettura e la gestione dei contenuti, le news verranno filtrate ritornando semplicemente 3 informazioni:

- Titolo;
- Data;
- URL.

Struttura

La struttura del progetto è organizzata nei seguenti *package*:

- **Controller:** i controller dell'applicazione, dove sono definite le varie API REST di interfaccia verso l'esterno;
- **Converter:** classi che consentono di convertire i dati in ingresso e in uscita, in modo tale da non avere un singolo modello per tutto;
- **Exception:** gestione delle eccezioni dell'applicazione;
- **Mapper:** i mapper che consentono il parsing delle informazioni provenienti dagli URL di Hacker News e New York Times;
- **Model:** le classi che modellano i dati del dominio dell'applicazione;
- **Service:** le classi che modellano la logica di business.

Inoltre il progetto possiede anche:

- Un file di **application.properties** utile per salvare le proprietà globali del progetto (sostanzialmente gli URL delle sorgenti);
- Un package di **test** che contiene i vari **Unit Test** delle classi.

Nei successivi paragrafi, verranno spiegate nel dettaglio le varie classi che compongono la **logica di business** dell'applicazione.

Mapper

La classe **NewsMapper** si occupa di connettersi agli URL di riferimento per Hacker News o New York Times ed effettua il parsing mediante la libreria *Jackson*.

E' organizzata sostanzialmente in 2 metodi:

- Il *costruttore* che registra un modulo di Jackson per decodificare la data in formato Java;
- Il metodo *parse(String URL, Class<T> clazz)* che si occupa di prendere URL da chiamare e il tipo di classe sulla quale effettuare il parsing della risposta dell'URL.
  In caso ci sia un errore nella formazione dell'URL, verrà generata una *MalformedURLException* e in caso di un JSON invalido o errore IO, un *IOException.* Le due eccezioni verranno rimappate in una *ResponseException* con una BAD_REQUEST o BAD_JSON.

Converter

Le classi **HackerNewsConverter** e **NYTConverter** si occupano semplicemente di convertire gli oggetti in risposta dal parser in un oggetto **News**, contenente solo le informazioni necessarie.

Viene fatto nel primo caso un adattamento da timestamp *long* a *ZonedDateTime*.

Service

Le classi che contengono la principale logica di business.

- **NewsService**: il principale servizio che si occupa di recuperare la lista di news dalle due sorgenti. Esso ha 4 metodi:
    - *getAggregatedNews(Integer limit, String section):* si occupa di prendere le news sia da Hacker News che da New York Times, e unisce le news sotto un'unica lista. Vengono usati i parametri *limit* per limitare le chiamate API verso Hacker News (dato che per ogni elemento effettua una chiamata http diversa) e *section* per prelevare le top stories da una sezione specifica del New York Times.
    - *getNewsFromHackerNews(Integer limit):* preleva le notizie da Hacker News con *limit* il numero di news da prelevare. Effettua anche tramite il converter la conversione dalla risposta del parser all'oggetto *News* finale;
    - *getNewsFromNewYorkTimes(String section):* preleva le notizie da New York Times con *section* la sezione dove prelevare le notizie. Effettua anche tramite il converter la conversione dalla risposta del parser all'oggetto *News* finale;
    - *sortNewsByDate(List<News> newsList):* ordina una lista di *News* per data dalla più recente alla meno recente.

Controller

La classe **NewsController** si occupa di definire le API che verranno richiamate dall'esterno per interagire con l'applicazione.

Esse sono 3, tutte di tipo *GET*:

- *getAggregatedNews(Integer limit, String section)*: preleva le notizie sia da Hacker News che da New York Times, specificando un limite per le notizie della prima sorgente e una sezione per la seconda;
- *getNewsFromHackerNews(Integer limit):* preleva le notizie soltanto da Hacker News con *limit* il numero di news da prelevare.
- *getNewsFromNewYorkTimes(String section):* preleva le notizie soltanto da New York Times con *section* la sezione dove prelevare le notizie.

Nella classe controller sono inoltre definite diverse annotazioni **OpenAPI 3.0** in modo tale da lanciare una schermata **Swagger** per poter visionare una documentazione e testare le varie API.

Gestione delle eccezioni

Il package **exception** comprende 4 classi per la gestione delle eccezioni:

- **ErrorResponse:** classe che modella l'errore mediante 3 informazioni: il messaggio, il tipo di codice di errore http e il timestamp dell'errore;
- **ResponseException:** eccezione per gestire le eccezioni in uscita, tramite l'enum *ResponseErrorEnum*;
- **ResponseErrorEnum:** classe enum che contiene gli errori possibili. Essi sono:
  - *BAD_REQUEST("Bad request", 400)*
  - *BAD_JSON("Bad request", 400)*
- **GlobalControllerExceptionHandler:** Handler globale delle eccezioni. Viene registrata solo l'eccezione *ResponseException*, ma potrebbe essere utilizzato anche per altri tipi di eccezioni come quelle di validazione ecc;

Test

Gli Unit Test dell'applicazione sono stati eseguiti mediante JUnit 5 e Mockito. Vi sono anche 2 classi astratte per i test relativi al controller e tutti gli altri test.

I test raggiungono una **code coverage** del 96%.

Documentazione e Repository

Tramite l'importazione della dipendenza OpenAPI 3.0, è possibile visionare la struttura delle API all'indirizzo (una volta avviata l'applicazione):

http://localhost:8080/swagger-ui/index.html

è inoltre possibile trovare un repository GitHub (se si volesse clonare il progetto da lì) direttamente all'indirizzo:

valerio65xz/sourcesenseassignment (github.com)