The “news-formatter” application is an implementation of the Jetlore test task on a position of a Java/Scala developer. The application is written in Java with a help of Spring Boot framework (which wasn’t prohibited, and as I’ve been using it heavily for the last several months, the choice was obvious).

# **Model**

The application’s model includes classes Type, Entity and Feed.

## **Type**:

Defines the following properties of an entity type:

* Name;
* Pattern;

There can be any amount of entity types in the system. In the given example, 3 entity types are used:

1. Entity: (name = “entity”, pattern = "<strong>\*</strong>");
2. Twitter: (name = "twitter", pattern = "<a href=\"http://twitter.com/\*\">\*</a>");
3. Link: (name = "link", pattern = "<a href=\"\*\">\*</a>");

## **Entity**:

Defines the exact entity that must be used to process a Facebook feed; has the following properties:

* Start position in the feed;
* End position;
* Entity type name;

For example, an entity can be the following: start = 14, end = 22, type = “entity”.

## **Feed**:

Defines an input object to a FeedController; has the following properties:

* Facebook post as a String;
* List of Entity objects.

# **Controllers**

In Web MVC, “controller” means “web service”. The “news-formatter” application provides 2 controllers – TypeController and FeedController.

## **TypeController**

Provides REST API to work with entity types – POST, GET and DELETE operations on “http://localhost:8080/api/types” endpoint. All these operations are bulk ones – one can create, get or delete an array of “Type” objects. An in-memory H2 database is used to store Type objects.

## **FeedController**

Provides a single POST method that receives a Feed object as a request body, and returns a processed text. Feed and Entity objects are not stored in the DB for the sake of simplicity.

# **Future** **Enhancements**

One can easily add more entity types to the application via the TypeController – just create a new Type object, say, the “hashtag” (name = “hashtag”, pattern = “#\*”), and everything will work as a charm if you provide entities of this type in a Feed object. Adding new types can be done in either of 2 ways:

1. Modify the TypeService.init() method that loads types into the DB at application start.
2. After the application start, call DELETE on “http://localhost:8080/api/types” endpoint, and then “POST” on “http://localhost:8080/api/types” with body

[

{

"id": 1,

"name": "entity",

"pattern": "<strong>\*</strong>"

},

{

"id": 2,

"name": "twitter",

"pattern": "<a href=\"http://twitter.com/\*\">\*</a>"

},

{

"id": 3,

"name": "link",

"pattern": "<a href=\"\*\"></a>"

}

{

"id": 4,

"name": "hashtag",

"pattern": "#\*"></a>"

}

]

Wu la – they are already 4 types in the DB!

# **Build and Deployment**

“news-formatter” is a Maven based application. One must simply run “mvn clean spring-boot:run” to compile it, run all tests and start on <http://localhost:8080> endpoint. REST controllers (TypeController and FeedController) . You can test manually all REST API via Swagger page on <http://localhost:8080/swagger-ui.html> :

