Week 15 – Research

Leiland Tanner

|  |
| --- |
|  |
| Review the following two resources: |
|  | https://www.baeldung.com/spring-data-rest-relationships |
|  | https://hellokoding.com/jpa-many-to-many-relationship-mapping-example-with-spring-boot-maven-and-mysql/ |
|  |  |

Write a detailed explanation of how to set up one-to-one, one-to-many, and many-to-many relationships using spring data.

Just as an initial challenge note: I struggled with this assignment at first because there were a ton of new terms and jargon that I needed to comb through and identify and research a bit before I could really make sense of the progression.

In my ongoing research I identified what I believe is an overarching pattern in setting up these different relationships with Spring Data between objects. After determining the proper relationships needed for a set of objects – their **ENTITIES** must first be created and noted with the proper Spring **annotations** for both the database query functionalities and table relations. Once the Entities are created and proper inverse relational annotations are included (the @ManytoOne v. @OnetoMany in respective classes) –repository **INTERFACES** must then be created which **extend** the Spring standard **CrudRepository**. The resources must first then be created before the proper associations can be given and stored – we’ll do these last steps by sending RESTful requests to the proper endpoints. The **POST** request will create the various resources and the **PUT** and **DELETE** requests will associate and disassociate the entities respectively. The **GET** request at the proper association endpoints will return the joined resources with their attached association URLS

(i.e “href” : http:// localhost:8080/users/1). The Baeldung site outlines a QA practice of building a class with a TestRestTemplate instance that will verify and assure the relationships are set up and working as intended. There are annotation and other slight variations in the above pattern to take into consideration when establishing relationships of one-to-one, one-to-many and many-to-many I note a few here that were outlined in the resources.

One-to-one:

@OneToOne

@JoinColumn/@JoinTable (these apply to other relationships too)

When creating associations with the PUT request a uri-list can be inserted by adding the URI address in the body of the request

EXAMPLE : PUT – http:// localhost:8080/addresses/1

Body = “Conent-Type:text/uri-list” http:// localhost8080/libraries/1/libraryAddress

One-to-many:

@OneToMany

Many-to-many:

@ManyToMany

In the case of many to many relationships – the URIs can be added on the uris.txt or uri-list with each URI occupying a separate line.