

Spring data rest

Features

- Provides rest api for domain model using HAL.
- Supports pagination.
- Allows to define projections.

What is HATEOAS?

HATEOAS is a concept of *application architecture*. It defines the way in which application clients interact with the server, by navigating hypermedia links they find inside resource models returned by the server.

A core principle of HATEOAS is that resources should be discoverable through the publication of links that point to the available resources.

What is HAL?

HAL = Hypertext Application Language.

HAL is a generic media type with which Web APIs can be developed and exposed as series of links.

MediaType = "application/hal+json"

What is HAL?

GET /orders/523 HTTP/1.1

Host: example.org

Accept: application/hal+json

HTTP/1.1 200 OK

Content-Type: application/hal+json

```
{
  "_embedded" : {
    "transactions" : [ { ... } ]
  },
  "_links" : {
    "first" : {
      "href" : "http://localhost:8080/api/transactions?page=0&size=20"
    }, ...
  },
  "page" : {
    "size" : 20,
    "totalElements" : 26,
    "totalPages" : 2,
    "number" : 0
  }
}
```

Dependency

```
<dependency>
```

```
  <groupId>org.springframework.boot </groupId>
```

```
  <artifactId>spring-boot-starter-data-rest </artifactId>
```

```
</dependency>
```

RepositoryRestConfigurer

```
@Bean

public RepositoryRestConfigurer repositoryRestConfigurer() {

    return new RepositoryRestConfigurerAdapter() {

        @Override

        public void configureRepositoryRestConfiguration(RepositoryRestConfiguration config) {

            config.setBasePath( "/api" );

        }

    };

}
```

RepositoryRestConfigurer properties

- basePath
- defaultPageSize
- maxPageSize
- pageParamName
- limitParamName
- sortParamName
- defaultMediaType

e.t.c.

Example

```
@RepositoryRestResource
```

```
public interface UserRepository extends JpaRepository<User, Long> {}
```

Supported http methods

- GET
- POST
- PUT
- DELETE
- HEAD
- OPTIONS

User

@Entity

```
public class User {
```

```
    @Id @GeneratedValue
```

```
    private long userId;
```

```
    private String username;
```

```
    private int age;
```

```
...getters/setters
```

```
}
```

How to access repository?

The path is derived from the uncapitalized, pluralized, simple class name of the domain class being managed.

User -> .../users

.../users/3

GET request

.../users // find all users using default pagination. Returns first page

.../users?page=1 // find all users using default pagination. Returns second page

.../users?size=2 // find all users using pagination by 2. returns first page.

POST request

.../users

body:

```
{  
  "username" : "test",  
  "age" : "1"  
}
```

PUT request

```
.../users/1
```

```
body:
```

```
{
```

```
  "username" : "test",
```

```
  "age" : "1"
```

```
}
```

DELETE request

.../users/1

Custom database query

```
@RepositoryRestResource
```

```
public interface UserRepository extends JpaRepository<User, Long> {  
    Collection<User> findByAge(@Param("age") int age);  
}
```

Custom database query call

.../users/search/findByAge?age=10