What is spring-data-rest?

[spring-data-rest](http://www.springsource.org/spring-data/rest) is a framework that helps you expose your entities directly as RESTful webservice endpoints, by default in [HAL](http://stateless.co/hal_specification.html)+JSON format and many more like JSON+ALPS, JSON+Schema.

1. Expose by default all your repositories on base url in HAL+JSON format
2. If we don’t want to expose some repositories just add annotation

@RepositoryRestResource(exported=false) on the Repository interface

1. By default Spring data rest creates the path to repository by pluralizing the entity name and adding it to the base URL Ex: base url: <http://localhost:8080/api>

Entity name “Model” final path: <http://localhost:8080/api/models>

If you want to change the path you can add some attributes to the @RepositoryRestResource(path = "mfgs", collectionResourceRel = "mfgs")

1. If we’d like to change the JSON payload returned by api we can add projections. For example, if we want to get all data about the model we have to define an interface with annotation @Projection(name = "modelDetail", types = { Model.class }) and define properties which you want to retrieve. Mandatory thing is that projection should be in domain model package or sub-package and projection defined getters should be identical with entity getters. If we want to customize our JSON returned, we can write virtual projections and we are not tied to the entity fields name. To have by default projections applied we add excerptProjection = ModelDetailView.class attribute on @RepositoryRestResource annotation. It works only for collection returned not for single item.
2. If we want to enable basic authentication you have to add spring security library to the pom and another configuration class with annotations:

@Configuration

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled = true)

On repository interface we have to add

@PreAuthorize("hasRole('ROLE\_ADMIN')")

By default it is securing custom spring data queries like: List<Manufacturer> findByActiveTrue();

To secure default JPARepository methods we have to override them and add

@PreAuthorize("hasRole('ROLE\_ADMIN')”) annotation.

1. We can add also some validation on entity fields and it validation fails then return error code: 400,"Bad Request" message will be returned. Also we can create our custom validations. All we need is to define an EventHandler with following annotations on the class level: @RepositoryEventHandler(Manufacturer.class) and @PreAuthorize("hasRole('ROLE\_ADMIN')")

@HandleBeforeCreate on method level.