

Spring Rest

Rest stands for Representational State Transfer.

Rest API calls over HTTP. It's a lightweight approach for communication between applications. Rest is language independent i.e. server can use any programming language and client can use any programming language.

Data format used in REST can be Json (Java Script object notation) or XML.

Data Binding or Serialization or mapping or marshaling is a process of converting Json data to java POJO. It is done in the background by Jackson. Please read

 Jackson Data Binding to understand the process.

In controller class we will use `@RestController` which is an extension of `@Controller`.

Most common use of Rest is over HTTP: -

- Get to get entity
- Post to add entity
- Put to update entity
- Delete to delete entity

Dependencies required for a REST project-


- war package
- spring-tx and spring-orm
- Mysql-connector-java
- C3p0
- spring-webmvc
- jackson-databind
- javax.servlet-api
- javax.servlet.jsp-api
- hibernate-core

To add new data use `@GetMapping` before method and `@PathVariable` as parameter.

To delete data use `@DeleteMapping` before method and `@PathVariable` as parameter.

To Add data use `@PostMapping` before method and `@RequestBody` as parameter.

To Update data use `@PutMapping` before method and `RequestBody` as parameter.

1. Import as maven project and add dependencies in pom.xml.
2. Create some basic packages named as entity (for entity classes), dao (for dao classes), service (for service classes), rest (for controller classes), config (for Dispatcher Servlet classes).
3. In src/main/java/webapp create a jsp file named index.jsp to add a link to display data in the UI.
4. Add application.properties with properties in this  application.properties .

Spring Rest

5. Create entity class in entity package and write the required code with annotations, getters and setters, toString() and constructors.
6. Create a Dao Interface declaring the necessary methods and define those methods in DaoImpl class. In DaoImpl class @Autowired sessionFactory with its object, in every method you need to get the current hibernate session, create a query, execute query and return. Basically, Dao is used to connect the project with a database.
7. Create an interface in the service package with the name service.java and declare the necessary methods for your project and then create a class with name ServiceImpl in the same package with the extension of service interface we just created to define the methods declared in service class. Use @Service before ServiceImpl class name, @Autowired Dao with an object. Now one by one write codes for different methods like you want them to function in your application. Basically, in ServiceImpl all you need is to return the methods with the Dao.
8. Now create a controller class in the rest package to control your serviceImpl class functions. First define @RestController and @RequestMapping("/api") before controller class name, inject service Impl with its object, @Autowired a constructor injection and define serviceImpl with tempImplObject in the parenthesis and initialize the main ImplObject with tempImplObject in the constructor body.
9. Now create a controller class in the rest package to control your serviceImpl class functions. First define @RestController and @RequestMapping("/api") before controller class name, inject service Impl with its object, @Autowired a constructor injection and define serviceImpl with tempImplObject in the parenthesis and initialize the main ImplObject with tempImplObject in the constructor body.
10. Map with annotations before all the functions. There are several annotations for mapping like
 - @GetMapping to get the data
 - @PostMapping to add the data
 - @PutMapping to update the data
 - @DeleteMapping to delete the data

After doing the necessary coding in the above classes, run the project as an application and after it is up in IDE, check localhost:8080/ in Browser.

Do CRUD operations in postman with the same URL from the browser and verify from the database.

If you type any specific id which is not in the database, the compiler will throw a 404 error. To avoid that error or get a custom message use exception handling.

Refer [Exception Handling](#) to understand exception handling in Spring Rest.

Find the code, postman collections and SQL query here:

<https://github.com/Leaf-Co-Kb/CRM-Spring-Rest>.