# SpringREST

This repository will contain a step by step approach to create RestFulWebServices using Spring, Spring Security and other related modules.

## Summary:

This is a basic example to make all the configurations using java annotations instead of configuring in xml.

We have used Spring 4 and many annotations like @Configuration , @EnableWebMvc, @ComponentScan for the configuration class. In the controller we have used @Controller , @RequestMapping , @RequestBody and @ResponseBody.

While configuring in the java, default http message converters are not used instead Xstream library is used to marshall and unmarshall message to and fro the REST service.

This Xstream library wrapped under the Http Message Converter is used to convert XML, JSON to Java and Java to XML and JSON.

All these message converters are configured in the Class with @Configuration annotation. The is configuration class is mentioned in the web.xml. In the "context-param" with param name "contextConfigLocation" which contains the package where the configuration class is present. Spring will first check the Class name then search for all the classes in the mentioned package line /org/balaji/config/\*\*/\*.class

This work is done by the AnnotationConfigWebApplicationContext class. And the ContextLoaderListener triggers the start.

Rest all are same as normal invocation, @RequestMapping will specify the method type GET/PUT/POST and the URL , indicating the url to be invoked to reach the method.

@ResponseBody will convert the response using the Http Message Converters to the format specified in the "Accept" header

@RequestBody will convert the incoming message to the appropriate Java object using the Http Message Converter based on "Content-Type". And @PathVariable is used to fetch the parameter from the HTTP URL.''

Also RestTemplate is used for testing the API's using different methods like exchange(), getForObject() etc. since this will directly hit the Service , Message Converters need to be configured and part of RestTemplate. In the test class we create a Foo object which gets converted to the XML/JSON using the HttpMessageConverters present in the RestTemplate.

Jmeter is also used to test the URL's and inspect the responses.

## How to test:

You can use postman app from the Google Chrome browser/CURL/Jmeter to test tha same.

## URL's to be used:

1. GET http://localhost:8080/spring-rest/foo/1

2. PUT http://localhost:8080/spring-rest/foo/1

## You can also use CURL also to test:

curl -i -H "Accept: application/json" -H "Content-Type: application/json" -X GET http://localhost:8080/spring-rest-new/foos/1

curl -H "Accept: application/xml" -H "Content-Type: application/xml" -X GET http://localhost:8080/spring-rest-new/foos/1

## Versions used:

|  |  |
| --- | --- |
| Spring Framework | 4.1.6.RELEASE |
| Java | 1.8 - 64 bit |
| Spring sts | Spring Tool Suite Version: 3.6.4.RELEASE - 64 bit |
| Tomcat | 8.0 - 64 bit |
| Servlet | 3.1.0 |