Day 1:

Resful Web API

===============

1. Introduction

================

REST (Representational State Transfer) is an architectural pattern for developing web services.

REST is a popular choice for web services development projects since it is lightweight, stateless, and

therefore easily scalable.

http is stateless protocol

REST uses HTTP verbs (POST, GET, PUT, DELETE) for performing CRUD operations (Create/Read/Update/Delete) on web

resources.

GET

POST

PUT

PATCH

DELETE

OPTIONS

HEAD

Platform supports to develop Restful WEB API

Java

.net

Node JS

PHP

Python

Implementaion

1.Spring Boot annotation

2.JAX-RS Java API for Restful API services

Spring *MVC application*

Model

View c –controller

@Controller

@RequestMapping (“/employees”)

getAllEmployees()

{

}

http://localhost:8080/spring-boot-ems/employees :GET : get all employees

http://localhost:8080/spring-boot-ems/employees/101 :GET : get employee with id 101

http://localhost:8080/spring-boot-ems/employees/101 :DELETE : delete employee with id 101

http://localhost:8080/spring-boot-ems/employees/101 :PUT : update employee with id 101

http://localhost:8080/spring-boot-ems/employees/101 :PATCH : update employee with id 101

http://localhost:8080/spring-boot-ems/employees :POST : add employee

difference is that when you want to update a resource with PUT request, you have to send the full payload as

the request whereas with PATCH, you only send the parameters which you want to update.

Resources

url method mapping new style

=========================================================================================================

/hello GET @RequestMapping(value="/hello",method=RequestMethod.GET) @GetMapping("/hello")

/hello POST @RequestMapping(value="/hello",method=RequestMethod.POST) @PostMapping("/hello")

/hello DELETE @RequestMapping(value="/hello",method=RequestMethod.DELETE) @DeleteMapping("/hello")

/hello PUT @RequestMapping(value="/hello",method=RequestMethod.PUT) @PutMapping("/hello")

/hello PATCH @RequestMapping(value="/hello",method=RequestMethod.PATCH) @PatchMapping("/hello")

==============================================================================

Spring Annotation JAX-RS Annotation

===============================================================================

@RequestMapping(path = "/troopers") @Path("/troopers")

@RequestMapping(method = RequestMethod.POST) @PostMapping @POST

@RequestMapping(method = RequestMethod.GET) @GetMapping @GET

@RequestMapping(method = RequestMethod.DELETE) @DELETE

@RequestMapping(method = RequestMethod.PUT) @PUT

@ResponseBody N/A --

@RequestBody N/A

@PathVariable("/{id}") @PathParam("id")

@RequestParam("xyz") @QueryParam('xyz")

@RequestParam(value="xyz") @FormParam(“xyz”)--through form

@RequestMapping(produces = {"application/json"}) @Produces("application/json")

@RequestMapping(consumes = {"application/json"}) @Consumes("application/json")

@ResponseBody=>Server to CLient=>Java Object to JSON or XML=>Accept=application/json

@RequestBody=>Client to Server=>JSON or XMl to Java Object to =>Content-type=application/json

client----server--request

server----client --response

In spring responce is in JSON . In jersey need to mention

MIME

Content-type;

/getEmployeById

/deleteEmployeeById

/updateEmployeeById

/addEmployee

JAXRS

======

Java API for RESTful Web Services (or JAX-RS) is a Java EE API specification for developing REST-style web

services. The API includes the annotations defined in JSR-311 and fall into the following categories:

URI Mapping for locating resources.

HTTP Methods for manipulating resources.

Data Formats for producing and consuming the textual representations of resources.

Request Parameters for binding parameters to Java types.

Exceptions Mappers for catching application exceptions and returning custom HTTP responses.

There are a few implementations from which to choose when developing JAX-RS web services. These include:

Spring Annotation JAX-RS Annotation

===============================================================================

@RequestMapping(path = "/troopers") @Path("/troopers")

@RequestMapping(method = RequestMethod.POST) @POST

@RequestMapping(method = RequestMethod.GET) @GET

@RequestMapping(method = RequestMethod.DELETE) @DELETE

@RequestMapping(method = RequestMethod.PUT) @PUT

@ResponseBody N/A --

@RequestBody N/A

@PathVariable("/{id}") @PathParam("id")

@RequestParam("xyz") @QueryParam('xyz")

@RequestParam(value="xyz") @FormParam(“xyz”)--through form

@RequestMapping(produces = {"application/json"}) @Produces("application/json")

@RequestMapping(consumes = {"application/json"}) @Consumes("application/json")

RESTEasy

Apache CXF

Jersey

Jersey serves as the JAX-RS reference implementation.

It is an open-source production-quality framework that expands on the JAX-RS toolkit.

It is part of the Glassfish JEE server stack but can be deployed on other application servers.

Creating a RESTful web service using Jersey is simple when using Spring Boot as the platform.

The amount of configuration required is small when you use the Spring Boot Jersey starter dependency in your

project, as will be demonstrated in this example.

==============================================================================

Spring Annotation JAX-RS Annotation

===============================================================================

@RequestMapping(path = "/troopers") @Path("/troopers")

@RequestMapping(method = RequestMethod.POST) @POST

@RequestMapping(method = RequestMethod.GET) @GET

@RequestMapping(method = RequestMethod.DELETE) @DELETE

@RequestMapping(method = RequestMethod.PUT) @PUT

@ResponseBody N/A --

@RequestBody N/A

@PathVariable("/{id}") @PathParam("id")

@RequestParam("xyz") @QueryParam('xyz")

@RequestParam(value="xyz") @FormParam(“xyz”)

@RequestMapping(produces = {"application/json"}) @Produces("application/json")

@RequestMapping(consumes = {"application/json"}) @Consumes("application/json")

@PathVariable: It is used to extract the values from the URI. It is most suitable for the RESTful web service,

where the URL contains a path variable. We can define multiple @PathVariable in a method.

@RequestParam: It is used to extract the query parameters form the URL. It is also known as a query parameter.

It is most suitable for web applications. It can specify default values if the query parameter is not present

in the URL.

Client Server

=========

Content-Type=application/json Accept @Consumes("application/json")

Accept =application/json Content-Type @Produces("application/json")

HTTP Status codes

==================

https://www.restapitutorial.com/httpstatuscodes.html

200 : OK

404 : NOT FOUND

500 : INTERNAL SERVER ERROR

403 : NOT AUTHORIZED type=Forbidden, status=403)

401 : NOT AUTHENTICATED type=Unauthorized, status=401

405 : METHOD NOT ALLOWED

415 : Unsupported Media Type

406 : Not Accepted

https://www.infosys.com/digital/insights/documents/restful-web-services.pdf

Rest is an arch style,which provides direction for building distributed and loosely coupled services.

It is not linked to any particular platform or technology. is is an idea to develop services to function

similar to the web.

resources

https://jsonplaceholder.typicode.com/

Resources

JSONPlaceholder comes with a set of 6 common resources:

/posts 100 posts

/comments 500 comments

/albums 100 albums

/photos 5000 photos

/todos 200 todos

/users 10 users

http://localhost:8080/spring-boot-ems/employees :GET : get all employees

http://localhost:8080/spring-boot-ems/employees/101 :GET : get employee with id 101

http://localhost:8080/spring-boot-ems/employees/101 :DELETE : delete employee with id 101

http://localhost:8080/spring-boot-ems/employees/101 :PUT : update employee with id 101

http://localhost:8080/spring-boot-ems/employees/101 :PATCH : update employee with id 101

http://localhost:8080/spring-boot-ems/employees :POST : add employee

url is same based on the method type BL will get implemented

All HTTP methods are supported.

==============================

GET /posts

GET /posts/1

GET /posts/1/comments

GET /comments?postId=1--

GET /posts?userId=1

POST /posts

PUT /posts/1

PATCH /posts/1

DELETE /posts/1

Resources

http://localhost:8080/spring-boot-ems/employees/101 --@Pathvariable supported by GET,POST,DELETE,PATCH

http://localhost:8080/spring-boot-ems/employees?empid=101 --@Requestparam @QueryString is supprted by GET

Representation:

XML or JSON format-- provide with @Produces and @Consumes

jersey dependency

Now we have a JAX-RS resource and we want to access it from spring boot application which include Jersey

dependency. Let’s register this resource as Jersey resource.

restful web api specifications infosys

http://localhost:8080/spring-boot-ems/employees/101==>path variable

http://localhost:8080/spring-boot-ems/employees?employeeid=101=>2RequestParam/ @QueryString

which 1 is recommnded?

? -- supported only get method where as path variable is supported by get,put,delete,patch

use nouns for resources and not verbs

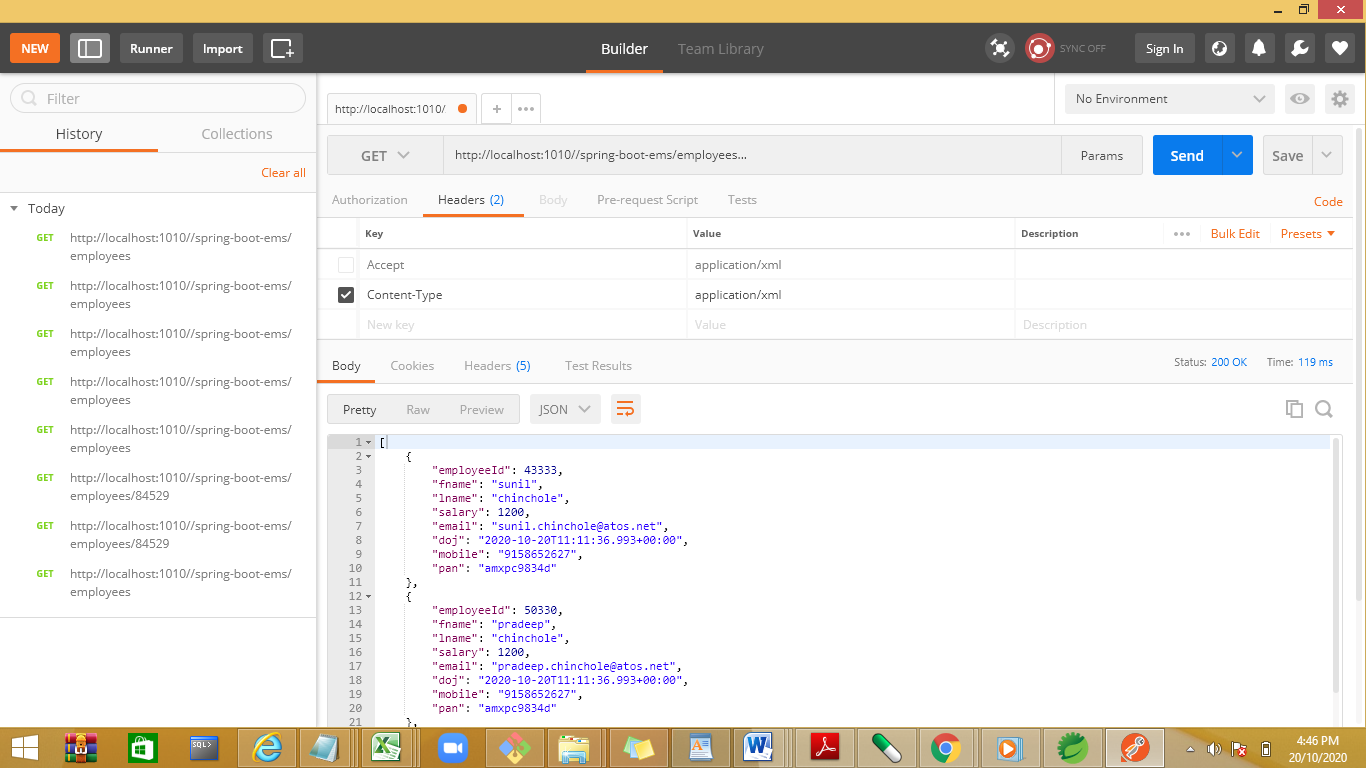
conventions

services any BL invoked through the http urls

Demos1: spring Demo with json and with xml s data type

Demo2:Jersey Demo

Demo 1 Spring Boot App with JSON



Demo2 with XML

Add the dependencies

</dependency>

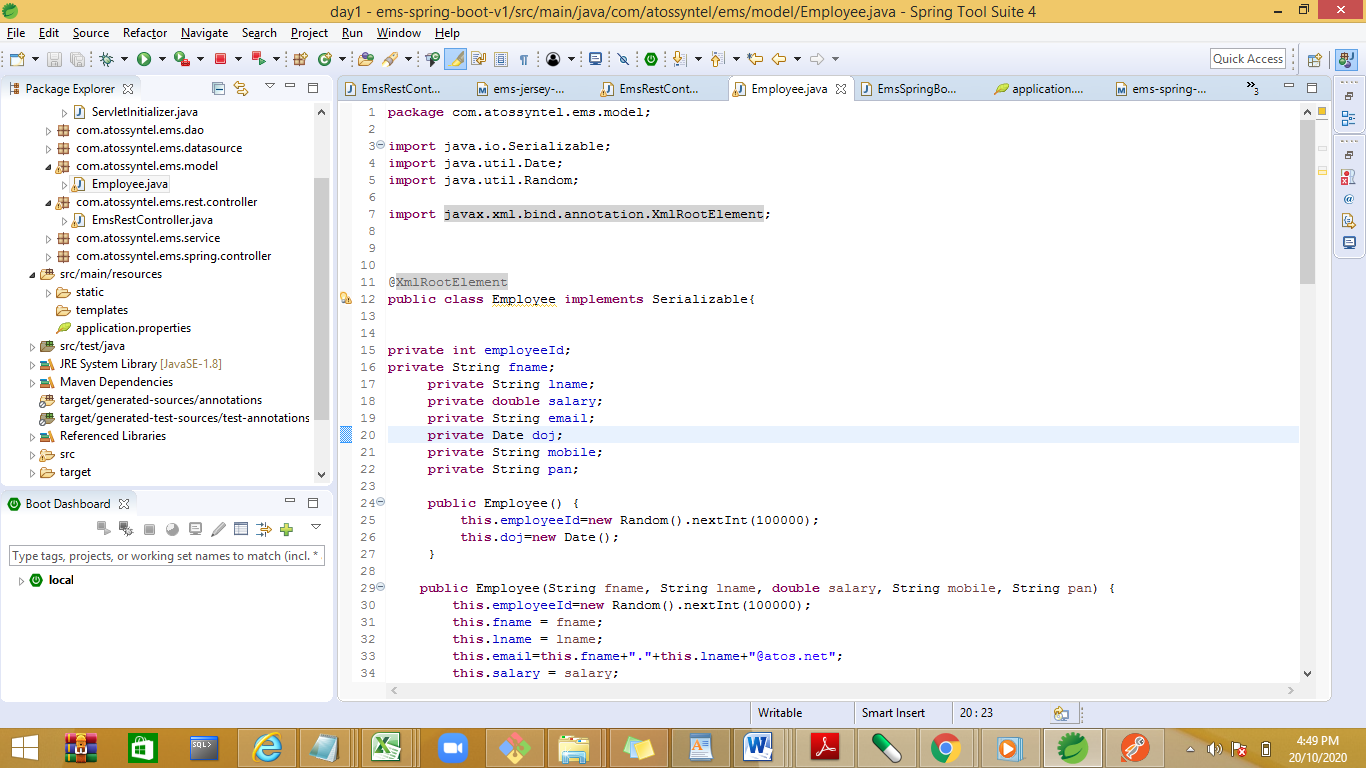
<dependency>

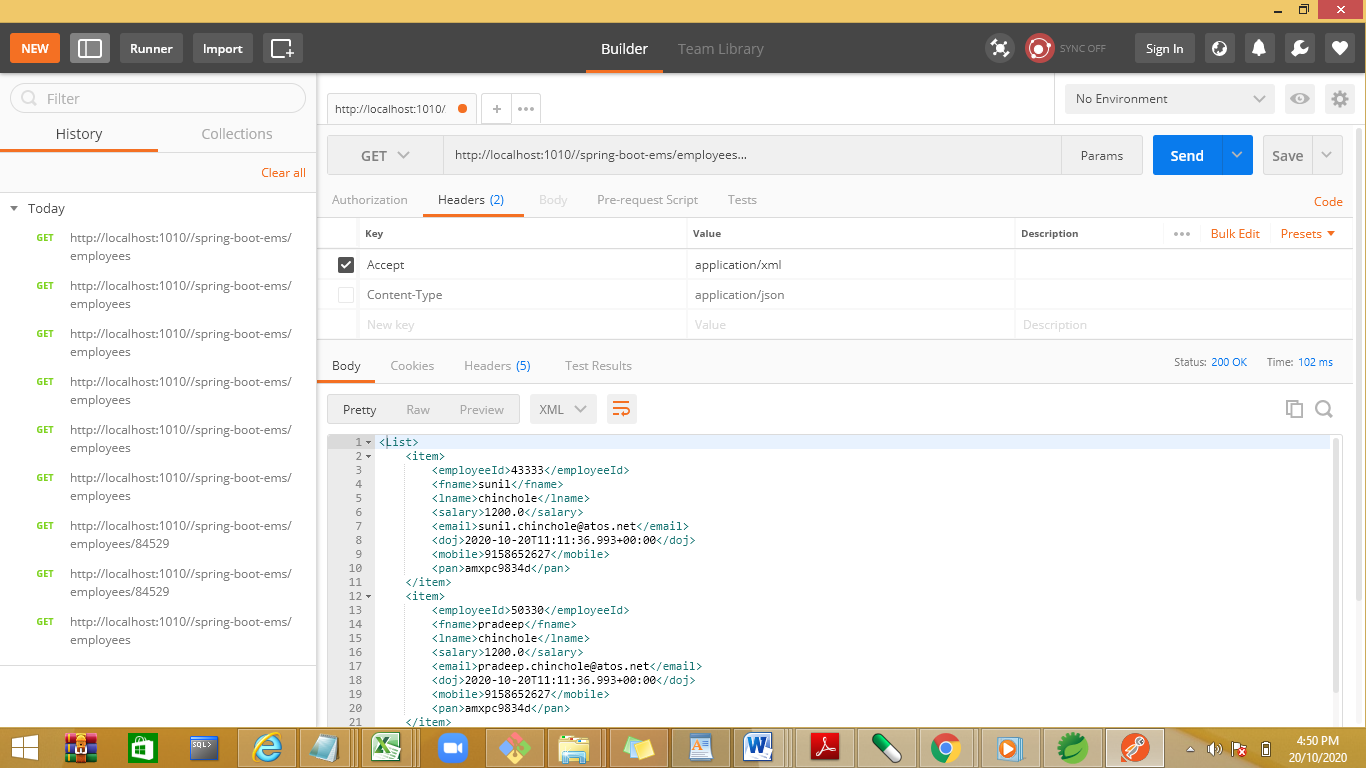
<groupId>com.fasterxml.jackson.dataformat</groupId>

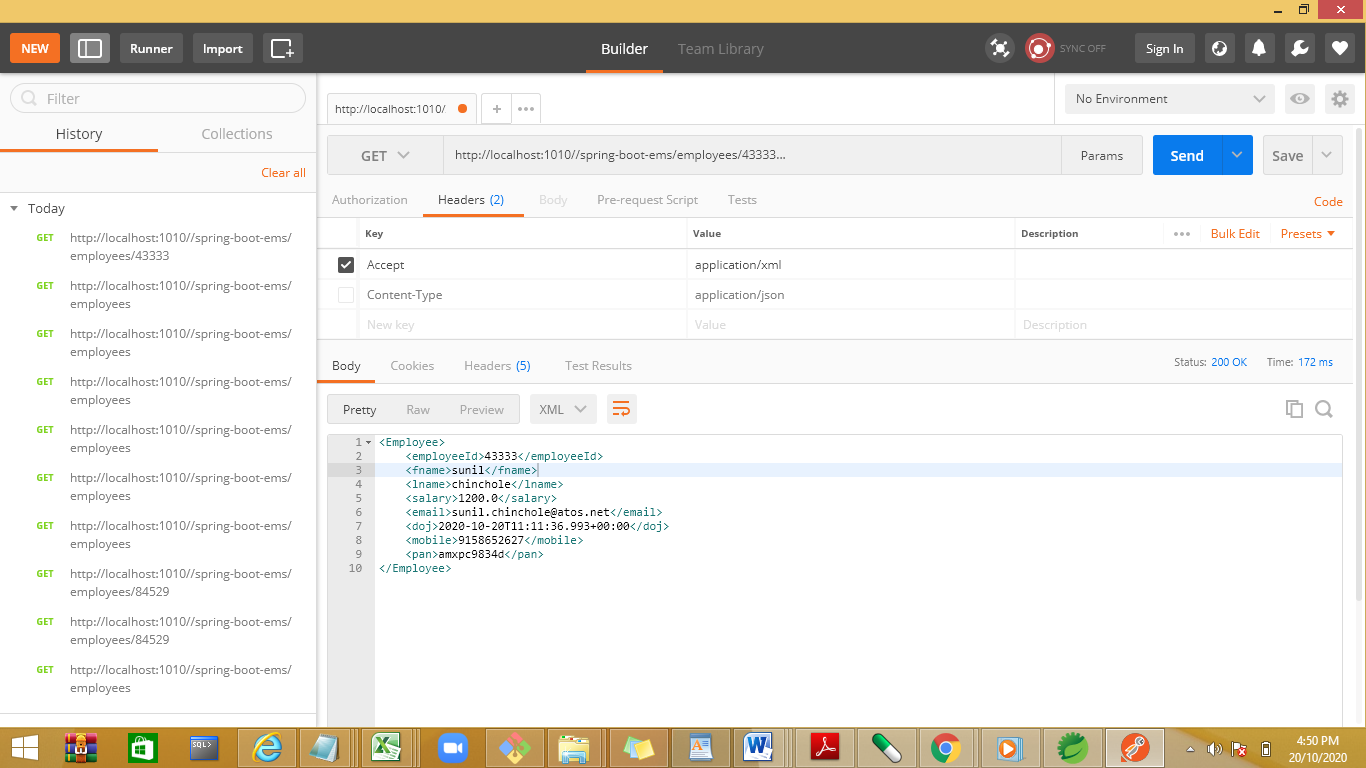
<artifactId>jackson-dataformat-xml</artifactId>

</dependency>

In Model add @XMLRootElement annotation







Demo :Jersy –API with JSON

Demo2: Jerset API with XML

Add the dependencies and 2XMLRootElement

