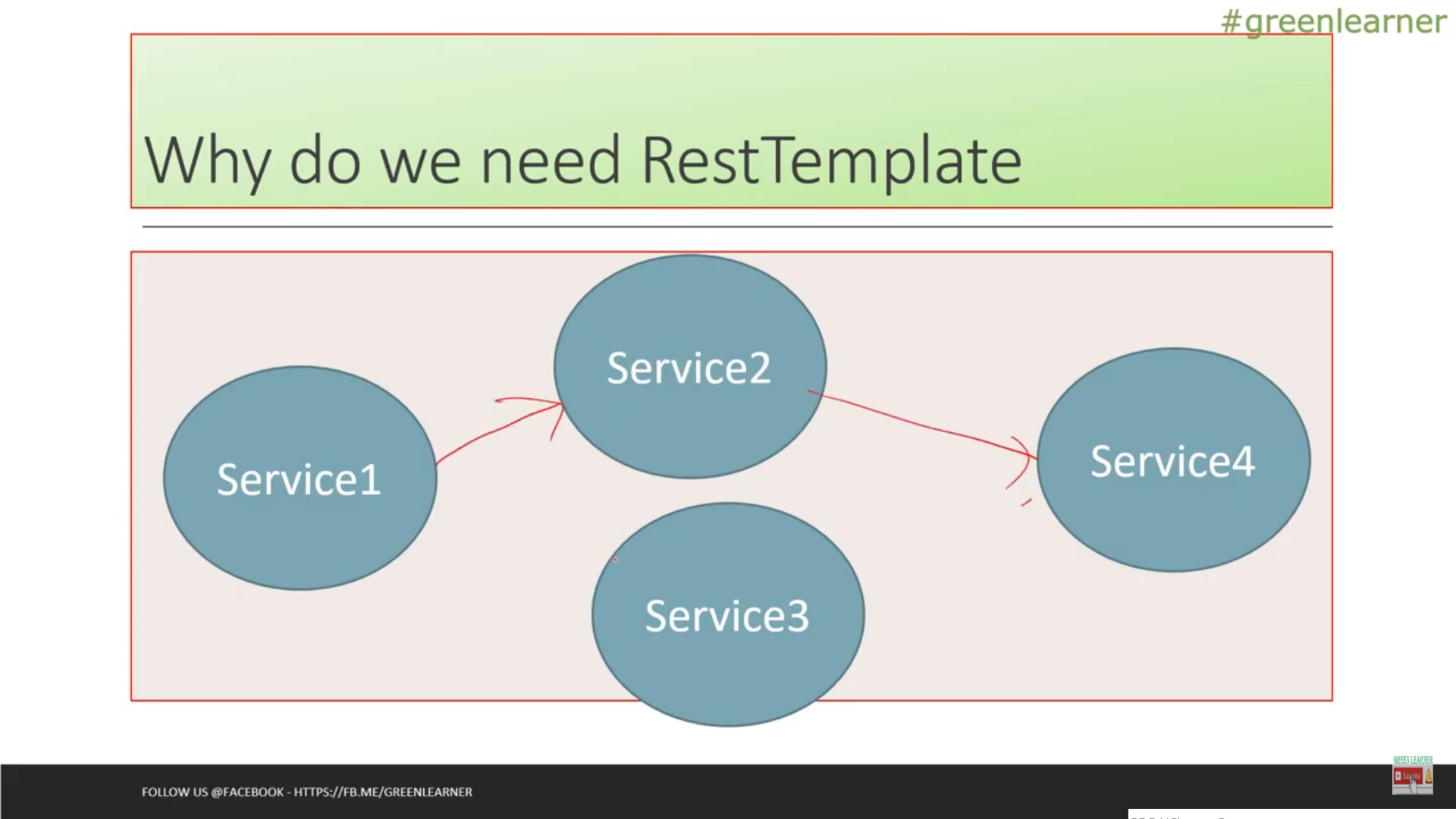
**Rest Template**

# Exchange method of Spring RestTemplate - Part 1 || Calling REST API using RestTemplate

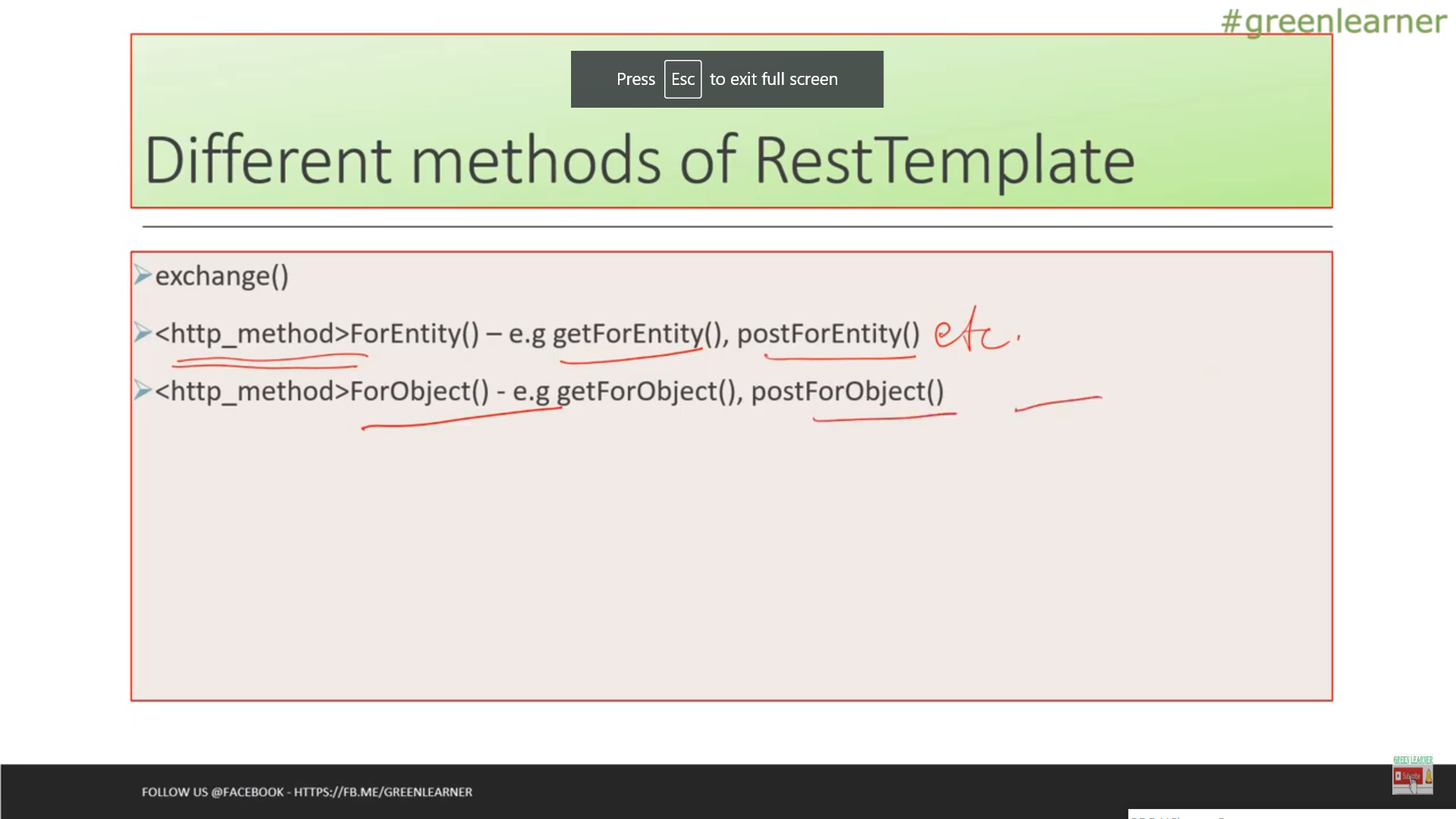
In this video we are going to understand microservice communications @ specifically about Rest Template that is used to enable communication among different services…(swagger UI we have interacted with Spring data and diff api services)…

So in previous applications we learned how to create different microservice using different features and what tools are available.



But in real world there will be a multiple microservices in a single application and all services should talk or interact with each other.

**Rest Template is one way which is used to enable communication…**



**exchange() ->** using exchange u can pass different types of http methods like Get, Put and Post and interact with the services…

package com.always.learning.UserManagementClient;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.http.\*;  
import org.springframework.web.client.RestTemplate;  
  
@SpringBootApplication  
public class UserManagementClientApplication {  
  
 static RestTemplate *restTemplate*=new RestTemplate();  
 static String *baseUrl*= "http://localhost:8083/springDataDemo/userdata/2";  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(UserManagementClientApplication.class, args);  
  
 *useExchangeMethodOfRestTemplate*();  
 }  
  
 private static void useExchangeMethodOfRestTemplate()  
 {  
 HttpHeaders httpHeaders=new HttpHeaders();  
 httpHeaders.setContentType(MediaType.*APPLICATION\_JSON*);  
 HttpEntity<Object> httpEntity=new HttpEntity<Object>(httpHeaders);  
  
 ResponseEntity<String> responseEntity=*restTemplate*.exchange(*baseUrl*,  
 HttpMethod.*GET*,  
 httpEntity,  
 String.class);  
 HttpStatus statusCode=responseEntity.getStatusCode();  
 System.*out*.println("status code="+statusCode);  
  
 String users=responseEntity.getBody();  
 System.*out*.println("response body="+users);  
  
 HttpHeaders headers=responseEntity.getHeaders();  
 System.*out*.println("response header="+users);  
  
 }  
  
}

server.port=8089

2020-07-10 18:19:09.755 INFO 31340 --- [ main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'

2020-07-10 18:19:11.404 INFO 31340 --- [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8089 (http) with context path ''

2020-07-10 18:19:11.600 INFO 31340 --- [ main] c.a.l.U.UserManagementClientApplication : Started UserManagementClientApplication in 9.091 seconds (JVM running for 12.62)

status code=200 OK

response body={"id":2,"name":"Raju","address":"Maldives"}

response header={"id":2,"name":"Raju","address":"Maldives"}

**If in Http we are mentioning the application name then in that application on what we are trying to get access resource file…..need to add in application.yaml below content**

server:  
 port: 8083  
 servlet:  
 context-path: /springDataDemo  
spring:  
 application:  
 name: spring-data-demo  
 datasource:  
 driver-class-name: com.mysql.cj.jdbc.Driver  
 password: Guidewire@2019  
 url: jdbc:mysql://localhost:3306/alwayslearning  
 username: root

**need to add server.servlet.context-path : /springDataDemo….**

**Otherwise we can access by**

<http://localhost:8083/userdata/2>

package com.always.learning.UserManagementClient;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.http.\*;  
import org.springframework.web.client.RestTemplate;  
  
@SpringBootApplication  
public class UserManagementClientApplication {  
  
 static RestTemplate *restTemplate*=new RestTemplate();  
 static String *baseUrl*= "http://localhost:8083/springDataDemo/userdata/2";  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(UserManagementClientApplication.class, args);  
  
 *useExchangeMethodOfRestTemplate*();  
 }  
  
 private static void useExchangeMethodOfRestTemplate()  
 {  
 HttpHeaders httpHeaders=new HttpHeaders();  
 httpHeaders.setContentType(MediaType.*APPLICATION\_JSON*);  
 HttpEntity<Object> httpEntity=new HttpEntity<Object>(httpHeaders);  
  
 ResponseEntity<String> responseEntity=*restTemplate*.exchange(*baseUrl*,  
 HttpMethod.*GET*,  
 httpEntity,  
 String.class);  
 HttpStatus statusCode=responseEntity.getStatusCode();  
 System.*out*.println("status code="+statusCode);  
  
 String users=responseEntity.getBody();  
 System.*out*.println("response body="+users);  
  
 HttpHeaders headers=responseEntity.getHeaders();  
 System.*out*.println("response header="+headers);  
  
 ResponseEntity<Users> responseEntity1=*restTemplate*.exchange(*baseUrl*,  
 HttpMethod.*GET*,  
 httpEntity,  
 Users.class);  
 Users users1=responseEntity1.getBody();  
 System.*out*.println("response body="+users1);  
  
  
 }  
  
}

**response body={"id":2,"name":"Raju","address":"Maldives"}**

**response header=[Content-Type:"application/json", Transfer-Encoding:"chunked", Date:"Fri, 10 Jul 2020 18:02:36 GMT", Keep-Alive:"timeout=60", Connection:"keep-alive"]**

**response body=Users{id='2', name='Raju', address='Maldives'}**

**first user as an json format and user as an object…**

# Exchange method of Spring RestTemplate- Part 2 || Calling REST API using RestTemplate

**For POST req with header need to pass Body also with HttpEntity…**

package com.always.learning.UserManagementClient;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.http.\*;  
import org.springframework.web.client.RestTemplate;  
  
@SpringBootApplication  
public class UserManagementClientApplication {  
  
 static RestTemplate *restTemplate*=new RestTemplate();  
 static String *baseUrl*= "http://localhost:8083/springDataDemo/";  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(UserManagementClientApplication.class, args);  
 HttpHeaders httpHeaders=new HttpHeaders();  
 httpHeaders.setContentType(MediaType.*APPLICATION\_JSON*);  
 HttpEntity<Object> httpEntity=new HttpEntity<Object>(httpHeaders);  
 System.*out*.println("=============Retrieve a single user based on id===========");  
 *getSingleUserByExchangeMethod*(httpEntity);  
 System.*out*.println("============Retrieve a List of Users ============");  
 *getListUserByExchangeMethod*(httpEntity);  
 System.*out*.println("============Adding the User ============");  
 Users sysUser=new Users();  
 sysUser.setName("kalpana");  
 sysUser.setAddress("us");  
 HttpEntity<Object> httpEnt=new HttpEntity<Object>(sysUser,httpHeaders);  
 *addUserByExchangeMethod*(httpEnt);  
 System.*out*.println("============update address based on Id of a User ============");  
 *updateUserByExchangeMethod*(httpEntity);  
 System.*out*.println("============delete specific based on Id of a User ============");  
 *deleteUserByExchangeMethod*(httpEntity);  
  
 }  
  
 private static void getSingleUserByExchangeMethod(HttpEntity<Object> httpEntity)  
 {  
 ResponseEntity<String> responseEntity=*restTemplate*.exchange(*baseUrl*+"userdata/2",  
 HttpMethod.*GET*,  
 httpEntity,  
 String.class);  
 HttpStatus statusCode=responseEntity.getStatusCode();  
 System.*out*.println("status code="+statusCode);  
  
 String users=responseEntity.getBody();  
 System.*out*.println("response body="+users);  
  
 HttpHeaders headers=responseEntity.getHeaders();  
 System.*out*.println("response header="+headers);  
  
// ResponseEntity<Users> responseEntity1=restTemplate.exchange(baseUrl,  
// HttpMethod.GET,  
// httpEntity,  
// Users.class);  
// Users users1=responseEntity1.getBody();  
// System.out.println("response body="+users1);  
 }  
  
 private static void getListUserByExchangeMethod(HttpEntity<Object> httpEntity)  
 {  
 ResponseEntity<String> responseEntity=*restTemplate*.exchange(*baseUrl*+"users\_data",  
 HttpMethod.*GET*,  
 httpEntity,  
 String.class);  
 HttpStatus statusCode=responseEntity.getStatusCode();  
 System.*out*.println("status code="+statusCode);  
  
 String users=responseEntity.getBody();  
 System.*out*.println("response body="+users);  
  
 HttpHeaders headers=responseEntity.getHeaders();  
 System.*out*.println("response header="+headers);  
  
 }  
  
 private static void addUserByExchangeMethod(HttpEntity<Object> httpEntity)  
 {  
 ResponseEntity<String> responseEntity=*restTemplate*.exchange(*baseUrl*+"user\_data",  
 HttpMethod.*POST*,  
 httpEntity,  
 String.class);  
 HttpStatus statusCode=responseEntity.getStatusCode();  
 System.*out*.println("status code="+statusCode);  
  
 String users=responseEntity.getBody();  
 System.*out*.println("response body="+users);  
  
 HttpHeaders headers=responseEntity.getHeaders();  
 System.*out*.println("response header="+headers);  
 }  
  
 private static void updateUserByExchangeMethod(HttpEntity<Object> httpEntity)  
 {  
 ResponseEntity<String> responseEntity=*restTemplate*.exchange(*baseUrl*+"user\_data/12/delhi",  
 HttpMethod.*PUT*,  
 httpEntity,  
 String.class);  
 HttpStatus statusCode=responseEntity.getStatusCode();  
 System.*out*.println("status code="+statusCode);  
  
 String users=responseEntity.getBody();  
 System.*out*.println("response body="+users);  
  
 HttpHeaders headers=responseEntity.getHeaders();  
 System.*out*.println("response header="+headers);  
  
 }  
  
 private static void deleteUserByExchangeMethod(HttpEntity<Object> httpEntity)  
 {  
 ResponseEntity<String> responseEntity=*restTemplate*.exchange(*baseUrl*+"user\_data/12",  
 HttpMethod.*DELETE*,  
 httpEntity,  
 String.class);  
 HttpStatus statusCode=responseEntity.getStatusCode();  
 System.*out*.println("status code="+statusCode);  
  
 String users=responseEntity.getBody();  
 System.*out*.println("response body="+users);  
  
 HttpHeaders headers=responseEntity.getHeaders();  
 System.*out*.println("response header="+headers);  
  
 }  
  
}

**17:37:39: Executing task 'UserManagementClientApplication.main()'...**

**> Task :compileJava**

**> Task :processResources UP-TO-DATE**

**> Task :classes**

**> Task :UserManagementClientApplication.main()**

**. \_\_\_\_ \_ \_\_ \_ \_**

**/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \**

**( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \**

**\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )**

**' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /**

**=========|\_|==============|\_\_\_/=/\_/\_/\_/**

**:: Spring Boot :: (v2.3.1.RELEASE)**

**2020-07-11 17:37:51.790 INFO 26040 --- [ main] c.a.l.U.UserManagementClientApplication : Starting UserManagementClientApplication on IN2393357W1 with PID 26040 (C:\GIT\_REPO\UserManagementClient\build\classes\java\main started by Kaushal.Kumar1 in C:\GIT\_REPO\usermanagementapp)**

**2020-07-11 17:37:51.800 INFO 26040 --- [ main] c.a.l.U.UserManagementClientApplication : No active profile set, falling back to default profiles: default**

**2020-07-11 17:37:56.010 INFO 26040 --- [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8089 (http)**

**2020-07-11 17:37:56.043 INFO 26040 --- [ main] o.apache.catalina.core.StandardService : Starting service [Tomcat]**

**2020-07-11 17:37:56.043 INFO 26040 --- [ main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.36]**

**2020-07-11 17:37:56.491 INFO 26040 --- [ main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext**

**2020-07-11 17:37:56.492 INFO 26040 --- [ main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 4560 ms**

**2020-07-11 17:37:57.330 INFO 26040 --- [ main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'**

**2020-07-11 17:37:58.067 INFO 26040 --- [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8089 (http) with context path ''**

**2020-07-11 17:37:58.105 INFO 26040 --- [ main] c.a.l.U.UserManagementClientApplication : Started UserManagementClientApplication in 7.498 seconds (JVM running for 11.279)**

**=============Retrieve a single user based on id===========**

**status code=200 OK**

**response body={"id":2,"name":"Raju","address":"Maldives"}**

**response header=[Content-Type:"application/json", Transfer-Encoding:"chunked", Date:"Sat, 11 Jul 2020 12:07:58 GMT", Keep-Alive:"timeout=60", Connection:"keep-alive"]**

**============Retrieve a List of Users ============**

**status code=200 OK**

**response body=[{"id":1,"name":"Bharat","address":"Bangalore"},{"id":2,"name":"Raju","address":"Maldives"},{"id":3,"name":"kanishk","address":"patna"},{"id":5,"name":"Raju","address":"Mumbai"},{"id":6,"name":"asedrtyui","address":"yui"},{"id":7,"name":"yui","address":"asd"},{"id":8,"name":"yui","address":"asd"},{"id":9,"name":"zxcv","address":"yui"}]**

**response header=[Content-Type:"application/json", Transfer-Encoding:"chunked", Date:"Sat, 11 Jul 2020 12:07:58 GMT", Keep-Alive:"timeout=60", Connection:"keep-alive"]**

**============Adding the User ============**

**status code=201 CREATED**

**response body=null**

**response header=[Location:"http://localhost:8083/springDataDemo/user\_data/12", Content-Length:"0", Date:"Sat, 11 Jul 2020 12:07:58 GMT", Keep-Alive:"timeout=60", Connection:"keep-alive"]**

**============update address based on Id of a User ============**

**status code=200 OK**

**response body=null**

**response header=[Content-Length:"0", Date:"Sat, 11 Jul 2020 12:07:58 GMT", Keep-Alive:"timeout=60", Connection:"keep-alive"]**

**============delete specific based on Id of a User ============**

**status code=200 OK**

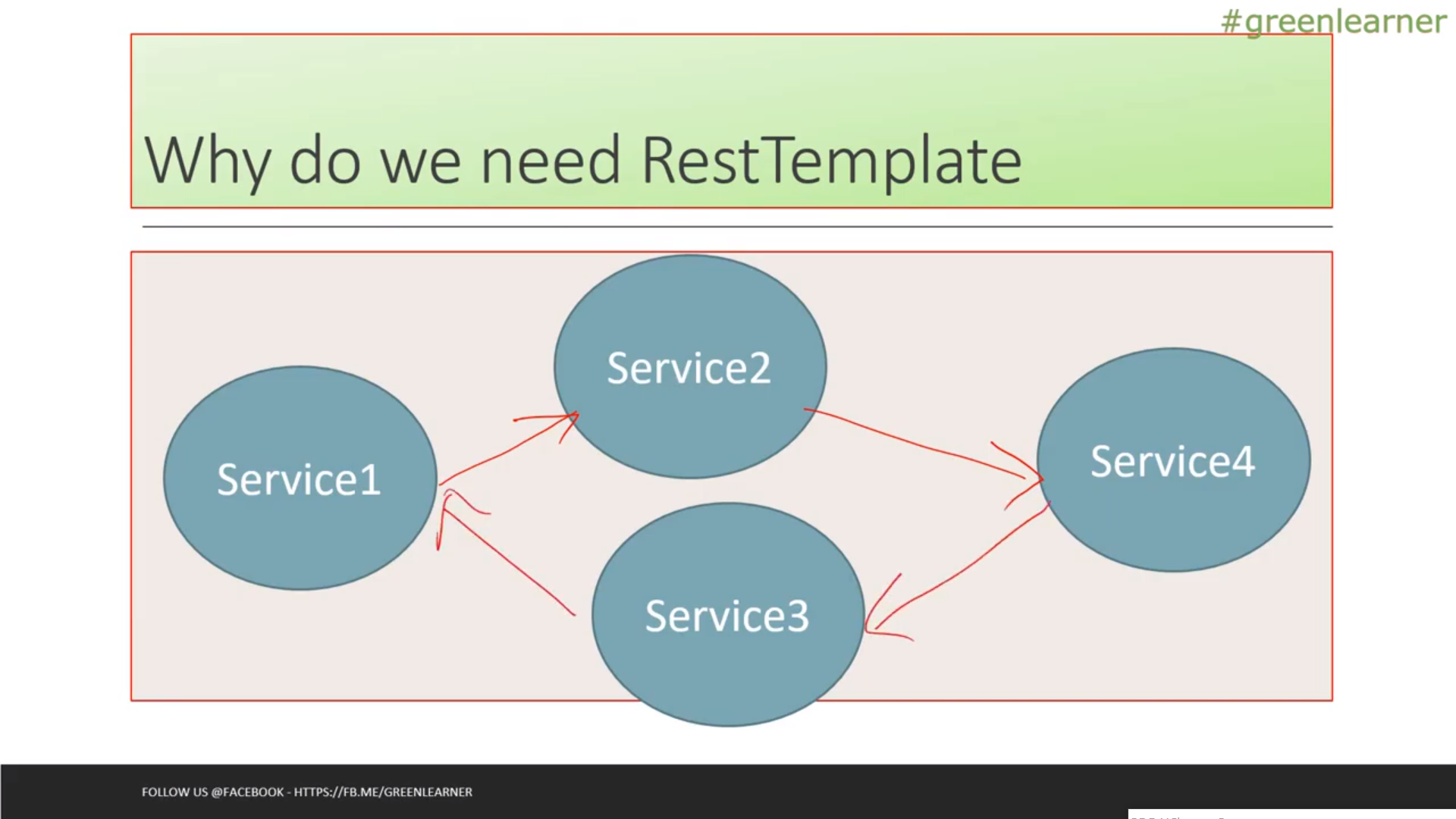
**response body=null**

**response header=[Content-Length:"0", Date:"Sat, 11 Jul 2020 12:07:58 GMT", Keep-Alive:"timeout=60", Connection:"keep-alive"]**

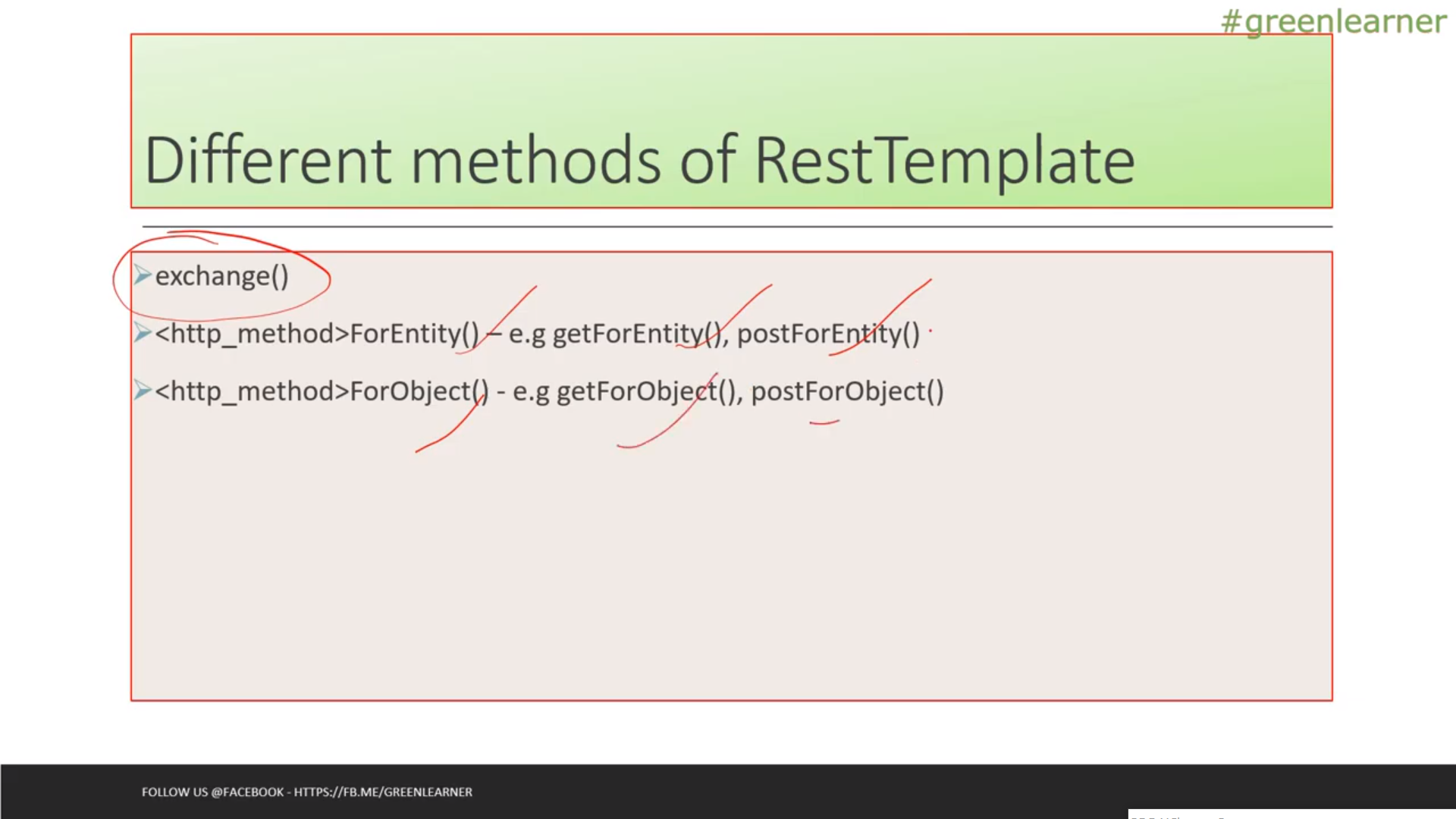
package com.always.learning.UserManagementClient;  
  
public class Users  
{  
 private String id;  
 private String name;  
 private String address;  
  
 public String getId() {  
 return id;  
 }  
  
 public void setId(String id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 @Override  
 public String toString() {  
 return "Users{" +  
 "id='" + id + '\'' +  
 ", name='" + name + '\'' +  
 ", address='" + address + '\'' +  
 '}';  
 }  
}

# Calling REST API using Spring RestTemplate using {HttpMethod}forEntity() || All about RestTemplate

# ( Microservice Communication )



**Using RestTemplate microservices can communicate with each others…**

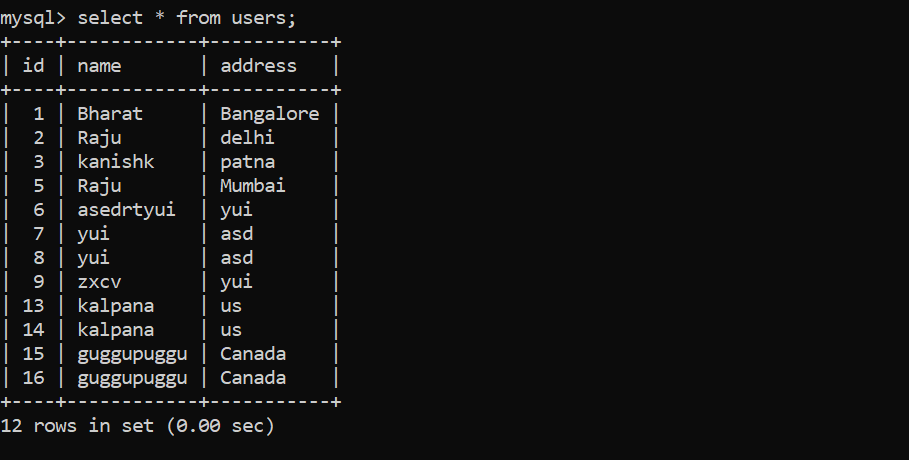


**Difference between getForEntity() and getForObject()…**

getForEntity() always returns the ResponseEntity from where we can get Header, Body and status details…but getForObject() always returns the User object or String it donot return the header details .

getForEntity() always returns the ResponseEntity but getForObject() always returns the String…

Before Performing operation database:-



Example for **ForEntityMethodOfRestTemplateDemo:-**

@SpringBootApplication  
public class UserManagementClientApplication {  
 static RestTemplate *restTemplate*=new RestTemplate();  
 static String *baseUrl*= "http://localhost:8083/springDataDemo/";  
  
 public static void main(String[] args)  
 {  
 SpringApplication.*run*(UserManagementClientApplication.class, args);  
 //useExchangeMethodOfRestTemplate();  
 ForEntityMethodOfRestTemplateDemo forEntityMethodOfRestTemplateDemo=new ForEntityMethodOfRestTemplateDemo();  
 forEntityMethodOfRestTemplateDemo.driverMethod();  
 //forObjectMethodOfRestTemplateDemo.driverMethod();  
  
 }

}

package com.always.learning.UserManagementClient;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.stereotype.Component;  
import org.springframework.web.client.RestTemplate;  
import java.net.URI;  
import java.util.List;  
  
@Component  
public class ForEntityMethodOfRestTemplateDemo  
{  
 static String *baseUrl*= "http://localhost:8083/springDataDemo/";  
 RestTemplate restTemplate=new RestTemplate();  
  
 public void driverMethod()  
 {  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*forEntity Methods demo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 getSingleObject();  
 getListObject();  
 //addUser();  
 updateUser();  
 deleteUser();  
 }  
 private void getSingleObject()  
 {  
 String url=*baseUrl*+"userdata/2";  
 ResponseEntity<String> user=restTemplate.getForEntity(url,String.class);  
 HttpStatus statusCode=user.getStatusCode();  
 System.*out*.println("status code="+statusCode);  
 String userDetails=user.getBody();  
 System.*out*.println("response body="+userDetails);  
 HttpHeaders headers=user.getHeaders();  
 System.*out*.println("response header="+headers);  
 }  
  
 private void getListObject()  
 {  
 String url=*baseUrl*+"users\_data";  
 ResponseEntity<List> user=restTemplate.getForEntity(url, List.class);  
 HttpStatus statusCode=user.getStatusCode();  
 System.*out*.println("status code="+statusCode);  
 List<Object> userDetails=user.getBody();  
 System.*out*.println("response body="+userDetails);  
 HttpHeaders headers=user.getHeaders();  
 System.*out*.println("response header="+headers);  
 }  
  
 private void addUser()  
 {  
 String url=*baseUrl*+"user\_data";  
 Users users=new Users();  
 users.setName("guggupuggu");  
 users.setAddress("Canada");  
 ResponseEntity<String> responseEntity=restTemplate.postForEntity(url,users,String.class);  
 HttpStatus statusCode=responseEntity.getStatusCode();  
 System.*out*.println("status code="+statusCode);  
 String body=responseEntity.getBody();  
 System.*out*.println("response body="+body);  
 HttpHeaders headers=responseEntity.getHeaders();  
 System.*out*.println("response header="+headers);  
// URI uri=restTemplate.postForLocation(url,String.class);  
// System.out.println("Location URI="+uri);  
  
 }  
 public void updateUser()  
 {  
 String url=*baseUrl*+"user\_data/2/chicago";  
 restTemplate.put(url,String.class);  
 System.*out*.println("User Updated !!!");  
 }  
 public void deleteUser()  
 {  
 String url=*baseUrl*+"user\_remove\_data/3";  
 restTemplate.delete(url,String.class);  
 System.*out*.println("User Deleted !!!");  
 }  
  
  
}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*forEntity Methods demo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

status code=200 OK

response body={"id":2,"name":"Raju","address":"chicago"}

response header=[Content-Type:"application/json", Transfer-Encoding:"chunked", Date:"Sun, 12 Jul 2020 13:26:04 GMT", Keep-Alive:"timeout=60", Connection:"keep-alive"]

status code=200 OK

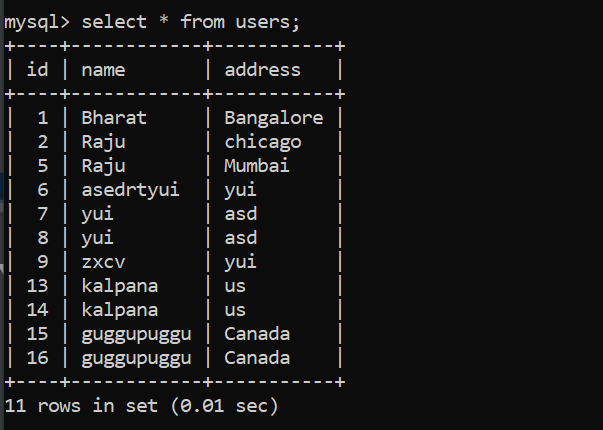
response body=[{id=1, name=Bharat, address=Bangalore}, {id=2, name=Raju, address=chicago}, {id=3, name=kanishk, address=patna}, {id=5, name=Raju, address=Mumbai}, {id=6, name=asedrtyui, address=yui}, {id=7, name=yui, address=asd}, {id=8, name=yui, address=asd}, {id=9, name=zxcv, address=yui}, {id=13, name=kalpana, address=us}, {id=14, name=kalpana, address=us}, {id=15, name=guggupuggu, address=Canada}, {id=16, name=guggupuggu, address=Canada}]

response header=[Content-Type:"application/json", Transfer-Encoding:"chunked", Date:"Sun, 12 Jul 2020 13:26:04 GMT", Keep-Alive:"timeout=60", Connection:"keep-alive"]

User Updated !!!

User Deleted !!!

After performing operation database:-



Example for **ForObjectMethodOfRestTemplateDemo:-**

@SpringBootApplication  
public class UserManagementClientApplication {  
 static RestTemplate *restTemplate*=new RestTemplate();  
 static String *baseUrl*= "http://localhost:8083/springDataDemo/";  
  
 public static void main(String[] args)  
 {  
 SpringApplication.*run*(UserManagementClientApplication.class, args);  
 //useExchangeMethodOfRestTemplate();  
 //ForEntityMethodOfRestTemplateDemo forEntityMethodOfRestTemplateDemo=new ForEntityMethodOfRestTemplateDemo();  
 //forEntityMethodOfRestTemplateDemo.driverMethod();  
 ForObjectMethodOfRestTemplateDemo forObjectMethodOfRestTemplateDemo=new ForObjectMethodOfRestTemplateDemo();  
 forObjectMethodOfRestTemplateDemo.driverMethod();  
  
 }

}

package com.always.learning.UserManagementClient;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.stereotype.Component;  
import org.springframework.web.client.RestTemplate;  
import java.net.URI;  
import java.util.List;  
  
@Component  
public class ForObjectMethodOfRestTemplateDemo  
{  
 Logger logger=LoggerFactory.*getLogger*(ForObjectMethodOfRestTemplateDemo.class);  
 static String *baseUrl*= "http://localhost:8083/springDataDemo/";  
 RestTemplate restTemplate=new RestTemplate();  
  
 public void driverMethod()  
 {  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*forObject Methods demo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 getSingleObject();  
 getListObject();  
 addUser();  
 updateUser();  
 deleteUser();  
 }  
 private void getSingleObject()  
 {  
 String url=*baseUrl*+"userdata/2";  
 String user=restTemplate.getForObject(url,String.class);  
 logger.info("User - {}",user);  
 }  
  
 private void getListObject()  
 {  
 String url=*baseUrl*+"users\_data";  
 List listOfUser=restTemplate.getForObject(url, List.class);  
 logger.info("List of User - {}",listOfUser);  
 }  
  
 private void addUser()  
 {  
 String url=*baseUrl*+"user\_data";  
 Users users=new Users();  
 users.setName("temp1");  
 users.setAddress("Canada");  
 String response=restTemplate.postForObject(url,users,String.class);  
 logger.info("response - {}",response);  
 }  
 public void updateUser()  
 {  
 String url=*baseUrl*+"user\_data/2/temp";  
 restTemplate.put(url,String.class);  
 System.*out*.println("User Updated !!!");  
 }  
 public void deleteUser()  
 {  
 String url=*baseUrl*+"user\_remove\_data/5";  
 restTemplate.delete(url,String.class);  
 System.*out*.println("User Deleted !!!");  
 }  
}

package com.always.learner.springDataDemo.controller;  
import com.always.learner.springDataDemo.dto.Users;  
import com.always.learner.springDataDemo.service.UserServices;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.PageRequest;  
import org.springframework.data.domain.Slice;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;  
import java.net.URI;  
import java.util.List;  
  
@RestController  
public class UserControllers  
{  
 Logger logger= LoggerFactory.*getLogger*(UserControllers.class);  
 @Autowired  
 UserServices userService;  
  
 @GetMapping("/users\_data")  
 public List<Users> getAllUser()  
 {  
 logger.info("access all user");  
 return userService.getAllUser();  
 }  
  
 @GetMapping("/userdata/{id}")  
 public Users getUser(@PathVariable("id") Long id)  
 {  
 logger.info("access user based on id");  
 return userService.getUser(id);  
 }  
  
 @PostMapping("/user\_data")  
 public ResponseEntity<Object> addUser(@RequestBody Users user)  
 {  
 userService.addUser(user);  
 URI path=ServletUriComponentsBuilder.*fromCurrentRequest*()  
 .path("/{id}")  
 .buildAndExpand(user.getId())  
 .toUri();  
 return ResponseEntity.*created*(path).build();  
 }  
  
 @PutMapping("/user\_data/{id}")  
 public String updateUser(@PathVariable Long id,@RequestParam String address)  
 {  
 userService.updateUser(id,address);  
 return "address updated for "+id;  
 }  
  
 @DeleteMapping("/user\_remove\_data/{id}")  
 public String deleteUser(@PathVariable Long id)  
 {  
 userService.deleteUser(id);  
 return "User with Id: "+id+" Deleted !!";  
 }  
  
 @GetMapping("/users/{name}")  
 public List<Users> getAllUsersByNames(@PathVariable String name)  
 {  
 return userService.getAllUsersByNames(name);  
 }  
  
 @GetMapping("/user/{address}")  
 public List<Users> getUsersByAddress(@PathVariable String address)  
 {  
 return userService.getUsersByAddress(address);  
 }  
  
 @PutMapping("/user\_data/{id}/{address}")  
 public void changeAddress(@PathVariable Long id,@PathVariable String address)  
 {  
 userService.changeAddress(id,address);  
 }  
  
 @GetMapping("/sort\_user/{sortedParam}")  
 public List<Users> findAllSortedUsers(@PathVariable String sortedParam)  
 {  
 return userService.findAllSortedUsers(sortedParam);  
 }  
  
 @GetMapping("/user\_per\_page/{pagenumber}/{numberOfElementsPerPage}")  
 public Page<Users> getAllUserByPages(@PathVariable String pagenumber, @PathVariable String numberOfElementsPerPage)  
 {  
 return userService.getAllUserByPages(Integer.*parseInt*(pagenumber),Integer.*parseInt*(numberOfElementsPerPage));  
  
 }  
  
 @GetMapping("/user\_per\_slice/{pagenumber}/{numberOfElementsPerSlice}")  
 public Slice<Users> getAllUserBySlice(@PathVariable String pagenumber, @PathVariable String numberOfElementsPerSlice) {  
 return userService.getAllUserBySlice(Integer.*parseInt*(pagenumber),Integer.*parseInt*(numberOfElementsPerSlice));  
 }  
  
 @GetMapping("/native/{address}")  
 public List<Users> getAllUsersById(@PathVariable String address)  
 {  
 return userService.getAllUsersByAddress(address);  
 }  
  
 @GetMapping("/logs\_data")  
 public String logs()  
 {  
 logger.trace("A TRACE MESSAGE");  
 logger.info("An INFO Message");  
 logger.debug("A DEBUG Message");  
 logger.error("An ERROR Message");  
 logger.warn("A WARN Message");  
 return "Hey buddy see the console for different level of logs";  
 }  
}

package com.always.learner.springDataDemo.controller;  
import com.always.learner.springDataDemo.dto.Users;  
import com.always.learner.springDataDemo.service.UserServices;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.PageRequest;  
import org.springframework.data.domain.Slice;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;  
import java.net.URI;  
import java.util.List;  
  
@RestController  
public class UserControllers  
{  
 Logger logger= LoggerFactory.*getLogger*(UserControllers.class);  
 @Autowired  
 UserServices userService;  
  
 @GetMapping("/users\_data")  
 public List<Users> getAllUser()  
 {  
 logger.info("access all user");  
 return userService.getAllUser();  
 }  
  
 @GetMapping("/userdata/{id}")  
 public Users getUser(@PathVariable("id") Long id)  
 {  
 logger.info("access user based on id");  
 return userService.getUser(id);  
 }  
  
 @PostMapping("/user\_data")  
 public ResponseEntity<Object> addUser(@RequestBody Users user)  
 {  
 userService.addUser(user);  
 URI path=ServletUriComponentsBuilder.*fromCurrentRequest*()  
 .path("/{id}")  
 .buildAndExpand(user.getId())  
 .toUri();  
 return ResponseEntity.*created*(path).build();  
 }  
  
 @PutMapping("/user\_data/{id}")  
 public String updateUser(@PathVariable Long id,@RequestParam String address)  
 {  
 userService.updateUser(id,address);  
 return "address updated for "+id;  
 }  
  
 @DeleteMapping("/user\_remove\_data/{id}")  
 public String deleteUser(@PathVariable Long id)  
 {  
 userService.deleteUser(id);  
 return "User with Id: "+id+" Deleted !!";  
 }  
  
 @GetMapping("/users/{name}")  
 public List<Users> getAllUsersByNames(@PathVariable String name)  
 {  
 return userService.getAllUsersByNames(name);  
 }  
  
 @GetMapping("/user/{address}")  
 public List<Users> getUsersByAddress(@PathVariable String address)  
 {  
 return userService.getUsersByAddress(address);  
 }  
  
 @PutMapping("/user\_data/{id}/{address}")  
 public void changeAddress(@PathVariable Long id,@PathVariable String address)  
 {  
 userService.changeAddress(id,address);  
 }  
  
 @GetMapping("/sort\_user/{sortedParam}")  
 public List<Users> findAllSortedUsers(@PathVariable String sortedParam)  
 {  
 return userService.findAllSortedUsers(sortedParam);  
 }  
  
 @GetMapping("/user\_per\_page/{pagenumber}/{numberOfElementsPerPage}")  
 public Page<Users> getAllUserByPages(@PathVariable String pagenumber, @PathVariable String numberOfElementsPerPage)  
 {  
 return userService.getAllUserByPages(Integer.*parseInt*(pagenumber),Integer.*parseInt*(numberOfElementsPerPage));  
  
 }  
  
 @GetMapping("/user\_per\_slice/{pagenumber}/{numberOfElementsPerSlice}")  
 public Slice<Users> getAllUserBySlice(@PathVariable String pagenumber, @PathVariable String numberOfElementsPerSlice) {  
 return userService.getAllUserBySlice(Integer.*parseInt*(pagenumber),Integer.*parseInt*(numberOfElementsPerSlice));  
 }  
  
 @GetMapping("/native/{address}")  
 public List<Users> getAllUsersById(@PathVariable String address)  
 {  
 return userService.getAllUsersByAddress(address);  
 }  
  
 @GetMapping("/logs\_data")  
 public String logs()  
 {  
 logger.trace("A TRACE MESSAGE");  
 logger.info("An INFO Message");  
 logger.debug("A DEBUG Message");  
 logger.error("An ERROR Message");  
 logger.warn("A WARN Message");  
 return "Hey buddy see the console for different level of logs";  
 }  
}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*forObject Methods demo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

2020-07-12 20:12:20.741 INFO 44332 --- [ main] .a.l.U.ForObjectMethodOfRestTemplateDemo : User - {"id":2,"name":"Raju","address":"temp"}

2020-07-12 20:12:20.850 INFO 44332 --- [ main] .a.l.U.ForObjectMethodOfRestTemplateDemo : List of User - [{id=1, name=Bharat, address=Bangalore}, {id=2, name=Raju, address=temp}, {id=5, name=Raju, address=Mumbai}, {id=6, name=asedrtyui, address=yui}, {id=7, name=yui, address=asd}, {id=8, name=yui, address=asd}, {id=9, name=zxcv, address=yui}, {id=13, name=kalpana, address=us}, {id=14, name=kalpana, address=us}, {id=15, name=guggupuggu, address=Canada}, {id=16, name=guggupuggu, address=Canada}, {id=17, name=temp, address=Canada}]

2020-07-12 20:12:20.939 INFO 44332 --- [ main] .a.l.U.ForObjectMethodOfRestTemplateDemo : response - null

User Updated !!!

User Deleted !!!

