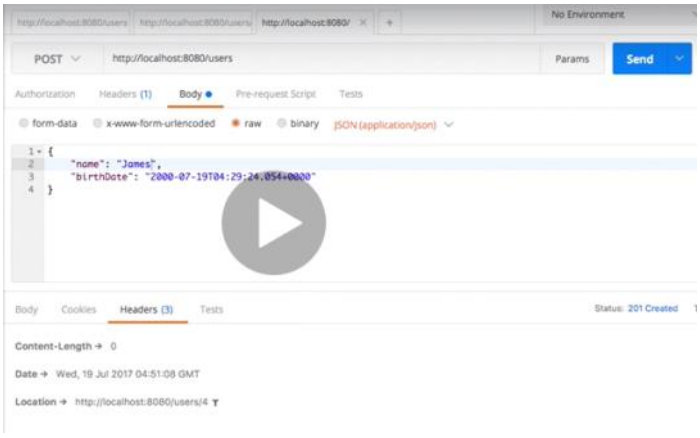
Monday, December 21, 2020 2:33 AM

Reprsentaional state transfer
Maximum use of rest

HTTP ->if we request anything on web browser then it will get the response from server and display in the form of html with all output. It conatins body and header
Swagger->service defination

Post

```
@PostMapping("/users")
ResponseEntity<Object> addvalue(@RequestBody User user) {
    User u = userconfig.addvalue(user);
    URI uri = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(u.getId()).toUri(); // add /users/id for new id and
    //return 201 created status while posting

    return ResponseEntity.created(uri).build();
}
```



New location url also given in header when we are posting with above code

Exception

```
public class UserNotFoundException extends RuntimeException {

    public UserNotFoundException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }

}
```

```
@GetMapping("/users/{id}")
public User retrieveUser(@PathVariable int id) {
    User user = service.findOne(id);
    if(user==null)
        throw new UserNotFoundException("id-"+ id);

    return user;
}
```

```java
@ResponseStatus(HttpStatus.NOT_FOUND)
public class UserNotFoundException extends RuntimeException {
    public UserNotFoundException(String message) {
        super(message);
    }
}
```

```java
ExceptionResponse exceptionResponse = new ExceptionResponse(new Date(),
        ex.getBindingResult().toString());
```

It will add the exception with proper message

@Valid to request body so that proper format has been added or posted

```java
@PostMapping("/users")
ResponseEntity<Object> addvalue(@Valid @RequestBody User user) {
    User u = userconfig.addvalue(user);
    URI uri = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(u.getId()).toUri

    // return 201 created status while posting
    return ResponseEntity.created(uri).build();
}
```

```java
@Component
public class User {

    Integer id;
    @Size(min =2)
    String name;
    String address;
```

Hateos

To provide link in the response

```java
if(user==null)
    throw new UserNotFoundException("id-"+ id);


//"all-users", SERVER_PATH + "/users"
//retrieveAllUsers
Resource<User> resource = new Resource<User>(user);

ControllerLinkBuilder linkTo =
        linkTo(methodOn(this.getClass()).retrieveAllUsers());


//HATEOAS

return user;
```

```
2017-07-19 14
2017-07-19 14
2017-07-19 14
2017-07-19 14
```

```
{
    "id": 1,
    "name": "Adam",
    "birthDate": "2017-07-19T09:26:18.337+0000",
    "_links": {
        "all-users": {
            "href": "http://localhost:8080/users"
        }
    }
}
```

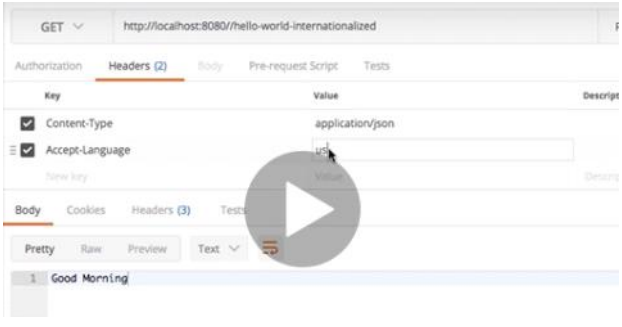
```
//"all-users", SERVER_PATH + "/users"
//retrieveAllUsers
Resource<User> resource = new Resource<User>(user);

ControllerLinkBuilder linkTo =
        linkTo(methodOn(this.getClass()).retrieveAllUsers());

resource.add(linkTo.withRel("all-users"));

//HATEOAS
```

```
#### Internationalization

##### Configuration
- LocaleResolver
   - Default Locale - Locale.US
- ResourceBundleMessageSource

##### Usage
- Autowire MessageSource
- @RequestHeader(value = "Accept-Language", required = false) Locale locale
- messageSource.getMessage("helloWorld.message", null, locale)
```

// default local=us

We can pass as @Request Header accept-language

```
@Autowired
MessageSource messagesource;

@GetMapping("/hellovalue")
String message(@RequestHeader(name="Accept-Language", required = false) Locale locale) {  //
    // as locale needs to be passed using header on based accept-language we need to pass locale in header
    return messagesource.getMessage("good.morning.message", null, locale);
}

@Bean
ResourceBundleMessageSource source() {
    ResourceBundleMessageSource source = new ResourceBundleMessageSource();
    source.setBasename("messages");
    return source;
}
```




```
@Bean
public LocaleResolver localeResolver() {
    AcceptHeaderLocaleResolver localeResolver = new AcceptHeaderLocaleResolver();

    localeResolver.setDefaultLocale(Locale.US);

    return localeResolver;
}
```

Swagger->

```java
@Configuration
@EnableSwagger2
public class Swagger {

    @Bean
    Docket docket()
    {
        return new Docket(DocumentationType.SWAGGER_12);
    }

}
```

```
{
    swagger: "2.0",
  + info: {…},
    host: "localhost:8080",
    basePath: "/",
  + tags: […],
  - paths: {
      + /error: {…},
      + /hello-world: {…},
      + /hello-world-bean: {…},
      + /hello-world-internationalized: {…},
      + /hello-world/path-variable/{name}: {…},
      + /users: {…},
      + /users/{id}: {…}
    },
  + definitions: {…}
}
```

```java
@ApiModel(description="All details about the user. ")
public class User {

    private Integer id;

    @Size(min=2, message="Name should have atleast 2 charac
    @ApiModelProperty(notes="Name should have atleast 2 cha
    private String name;

    @Past
    @ApiModelProperty(notes="Birth date should be in the past")
    private Date birthDate;

    protected User() {

    }
```

Filter out some of the contents and don't want to show much to end user

```java
public class SomeBean {

    private String field1;

    private String field2;

    @JsonIgnore
    private String field3;

@JsonIgnoreProperties(value={"field1","field2"})
public class SomeBean {
```

Dynamic filter

```java
@JsonFilter("SomeBeanFilter")
public class SomeBean {

    private String field1;

    private String field2;

    private String field3;

    public SomeBean(String field1
```

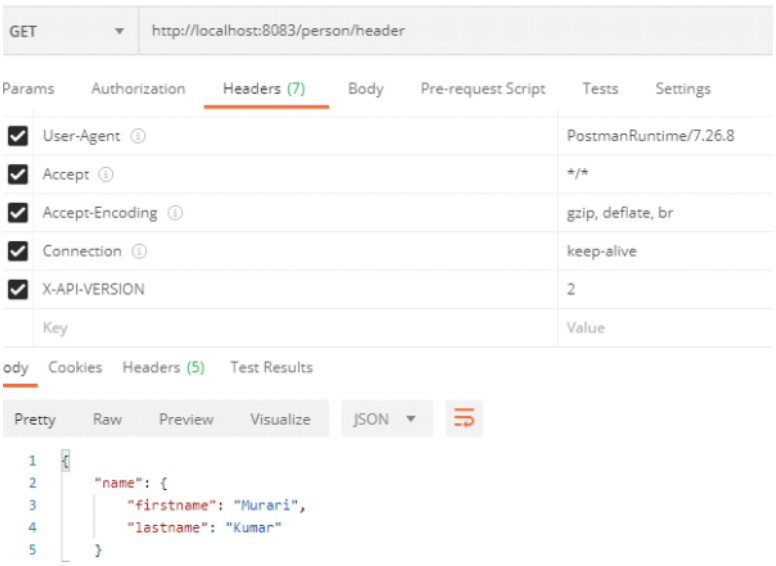
```java
@GetMapping("/filtering")
public MappingJacksonValue retrieveSomeBean(){
    SomeBean someBean = new SomeBean("value1","value2","value3");

    SimpleBeanPropertyFilter filter = SimpleBeanPropertyFilter.
            filterOutAllExcept("field1","field2");

    FilterProvider filters = new SimpleFilterProvider().addFilter("SomeBeanFilt

    MappingJacksonValue mapping = new MappingJacksonValue(someBean);

    mapping.setFilters(filters);

    return mapping;
```

```java
@RestController
public class VersioningController {

    @GetMapping("v1/person")  // url versioning
    PersonV1 personev1()
    {
        return new PersonV1("Murari Kumar");
    }

    @GetMapping("v2/person")
    PersonV2 personev2()
    {
        return new PersonV2(new Name("Murari", "Kumar"));
    }

}
```

Versioning using request param

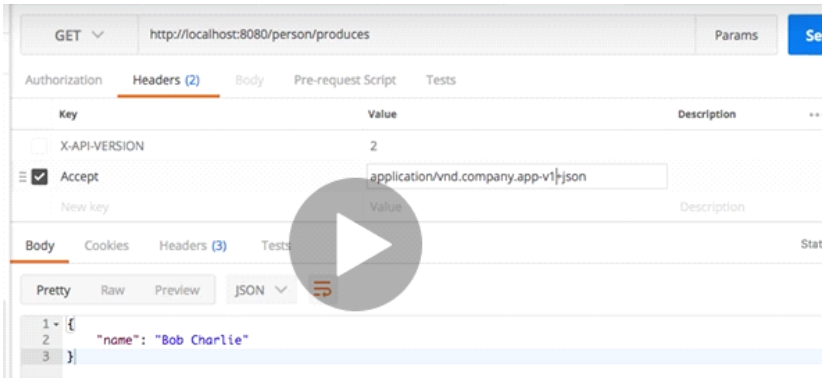
```java
@GetMapping(value = "v1/person", params = "version=1") // versioning using request param
PersonV1 paramv1() {
    return new PersonV1("Murari Kumar");
}

@GetMapping(value = "v2/person", params = "version=2")
PersonV2 paramv2() {
    return new PersonV2(new Name("Murari", "Kumar"));
}
```

```java
@GetMapping(value = "/person/header", headers = "X-API-VERSION=1") // versioning using request header
PersonV1 headerv1() {
    return new PersonV1("Murari Kumar");
}

@GetMapping(value = "/person/header", headers = "X-API-VERSION=2")
PersonV2 headerv2() {
    return new PersonV2(new Name("Murari", "Kumar"));
}
```



Versioning using produces

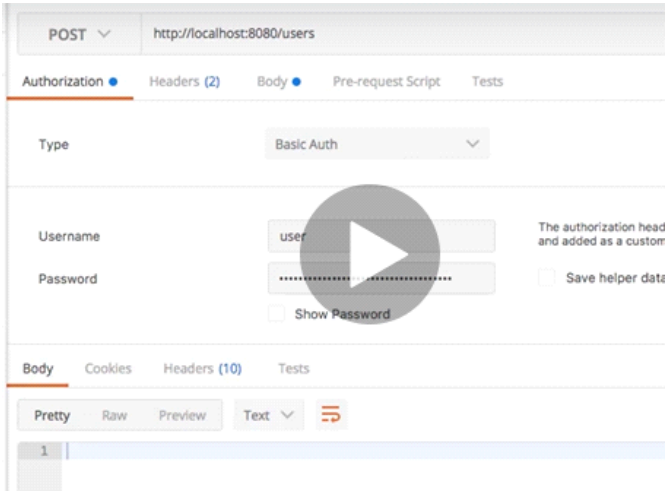
```
@GetMapping(value = "/person/produces", produces = "application/vnd.company.app-v1+json") // versioning using produces
PersonV1 producesv1() {
    return new PersonV1("Murari Kumar");
}

@GetMapping(value = "/person/produces", produces = "application/vnd.company.app-v2+json")
PersonV2 producesv2() {
    return new PersonV2(new Name("Murari", "Kumar"));
}
```



Header request problem in caching difficult // cannot use from browser// swagger and documentation difficulty

Still cache using uri

Security password sending in authorization



H2 DATABASES FECTHING DATA

```
@Entity
public class UserDTO {

    @Id
    @GeneratedValue
    int id;
    String name;
    String address;
    Long phone;
```

```
server.port=8083
spring.jpa.show-sql=true
 #this start logging jpa
spring.h2.console.enabled=true
```

```
@RestController
public class UserJPAResource {

    @Autowired
    UserRepositroy user;

    @GetMapping("/findall")
    List<User> findAll() {
        return user.findAll();
    }

    @GetMapping("/findId/{id}")
    Optional<User> findById(@PathVariable int id
        return user.findById(id);
    }

    @PostMapping("/users")
    void addUser(@RequestBody User u) {
        user.save(u);
    }

    @DeleteMapping("/users/{id}")
    void deleteuser(@PathVariable int id) {
        user.deleteById(id);
```

```
@Repository
public interface UserRepositroy  extends JpaRepository<User, Integer>{

}
```

Many to one relationship

User can have many post
Post will have same user id for all post

```java
@Entity
public class Post {

    @Id
    @GeneratedValue
    private Integer id;
    private String description;

    @ManyToOne(fetch=FetchType.LAZY)
    @JsonIgnore
    private User user;

    public Integer getId() {
        return id;
    }
```

To get all his post

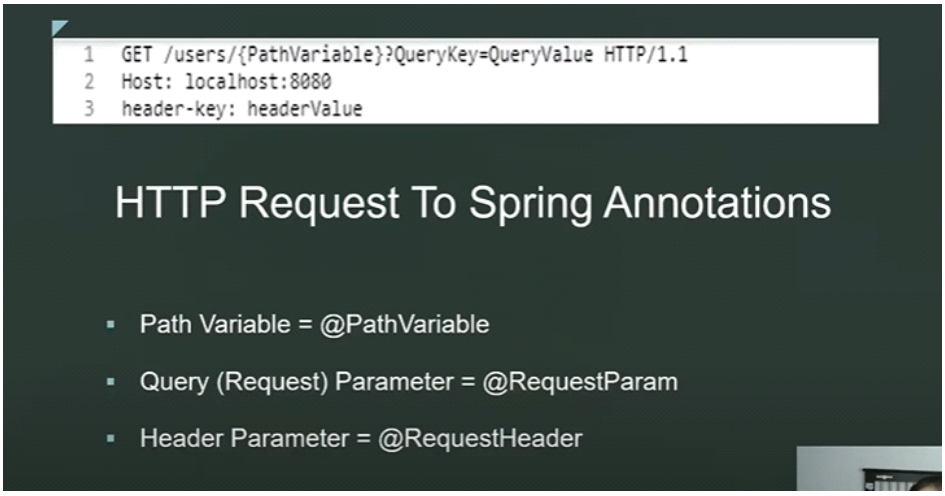
```java
@Entity
public class User {

    @Id
    @GeneratedValue
    int id;
    String name;
    String address;
    int phone;
    @OneToMany(mappedBy = "user")
    List<Post> post;

    public User() {
        // TODO Auto-generated constructor stub
    }
```

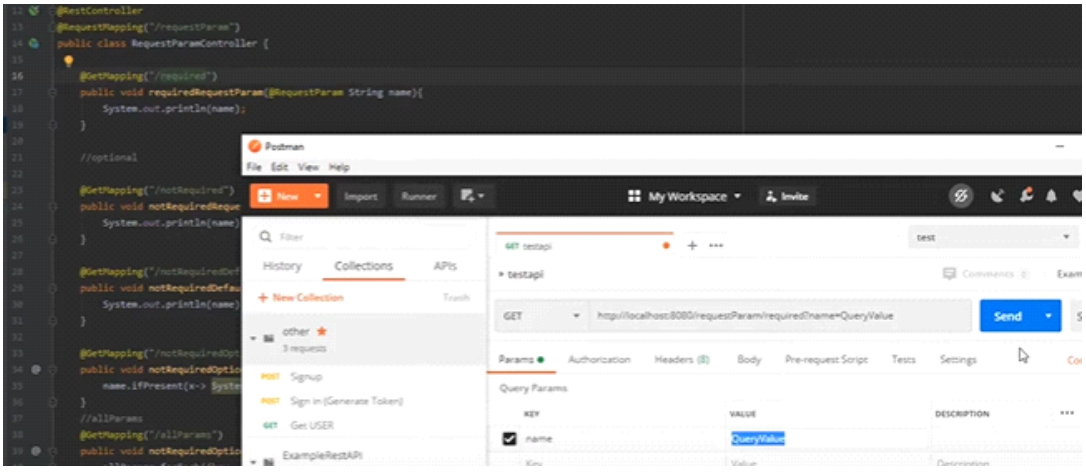
```java
@GetMapping("/jpa/users/{id}/posts")
public List<Post> retrieveAllUsers(@PathVariable int id) {
    Optional<User> userOptional = userRepository.findById(id);

    if(!userOptional.isPresent()) {
        throw new UserNotFoundException("id-" + id);
    }

    return userOptional.get().getPosts();
}
```

```java
@Entity
public class Post {
    @Id
    @GeneratedValue
    int id;
    String descrription;
    @ManyToOne(fetch = FetchType.LAZY)
    @JsonIgnore  // infinite loop as both will start calling each other so we put ignore
    User user;

    public int getId() {
```

```java
@PostMapping("/jpa/users/{id}/posts")
public ResponseEntity<Object> createUser(@PathVariable int id, @RequestBody Post post)
{
Optional<User> userOptional= userRepository.findById(id);
if(!userOptional.isPresent())
{
throw new UserNotFoundException("id: "+ id);
}
User user=userOptional.get();
//map the user to the post
post.setUser(user);
//save post to the database
postRepository.save(post);
//getting the path of the post and append id of the post to the URI
URI location=ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(post.getId()).toUri();
//returns the location of the created post
return ResponseEntity.created(location).build();
}
```

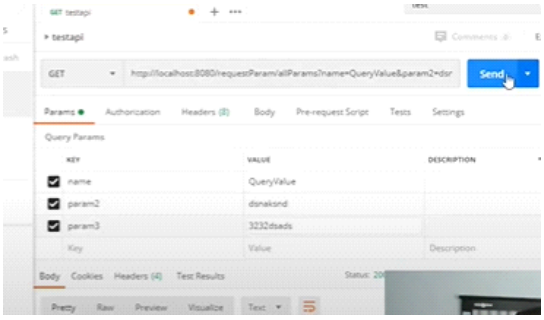
When posting post from postman while in list post from user

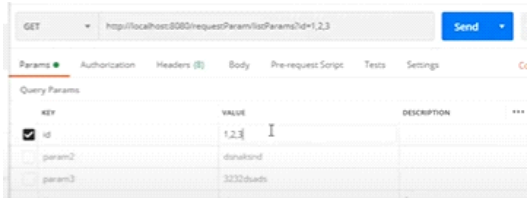




Query parammater using map @RequestParam

```
//allParams
@GetMapping("/allParams")
public void notRequiredOptional(@RequestParam Map<String,String> allParams ){
    allParams.forEach((key, value) -> {
        System.out.println(String.format("Request Param '%s' = %s", key, value));
    });
}

@GetMapping("/listParams")
public void paramList(@RequestParam List<Integer> id ){
    id.forEach(eachId-> System.out.println(eachId));
}
```





List param

When we want parse api json

```
@GetMapping("/restore")
List<BitCoin> storevalue() {
    RestTemplate rest = new RestTemplate();
    ResponseEntity<List<BitCoin>> rateResponse =
            rest.exchange("https://bitpay.com/api/rates",
                    HttpMethod.GET, null, new ParameterizedTypeReference<List<BitCoin>>() {
                });

    return reposiotry.save(rateResponse.getBody());
```

```
@SpringBootApplication
//@EnableDiscoveryClient
@ComponentScan(basePackages = "com.*")
@EntityScan("com.*")
public class CurrencyExchangeServiceApplication {
```

https://www.foreach.be/blog/spring-cache-annotations-some-tips-tricks

Spring cache to search using ehcache
Spring Cache Example using EhCache in Spring Boot | Tech Primers

```
<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="ehcache.xsd"
         updateCheck="true"
         monitoring="autodetect"
         dynamicConfig="true">

    <diskStore path="java.io.tmpdir" />

    <cache name="usersCache"
           maxEntriesLocalHeap="10000"
           maxEntriesLocalDisk="1000"
           eternal="false"
           diskSpoolBufferSizeMB="20"
           timeToIdleSeconds="300" timeToLiveSeconds="600"
           memoryStoreEvictionPolicy="LFU"
           transactionalMode="off">
        <persistence strategy="localTempSwap" />
    </cache>
```

```java
@EnableJpaRepositories(basePackages = { "com.techprimers.cache.repository", "com.techprimers.cache" })
@EnableCaching
@Configuration
public class EhCacheConfiguration {

    @Bean
    public CacheManager cacheManager() {
        return new EhCacheCacheManager(cacheMangerFactory().getObject());
    }

    @Bean
    public EhCacheManagerFactoryBean cacheMangerFactory() {
        EhCacheManagerFactoryBean bean = new EhCacheManagerFactoryBean();
        bean.setConfigLocation(new ClassPathResource("ehcache.xml"));
        bean.setShared(true);
        return bean;
    }
}
```

```java
@Component
public class UsersCache {

    @Autowired
    UsersRepository usersRepository;

    @Cacheable(value = "usersCache", key = "#name")
    public Users getUser(String name) {
        System.out.println("Retrieving from Database for name: " + name);
        return usersRepository.findByName(name);
    }
}
```

```
timeToIdleSeconds="300"
timeToLiveSeconds="600">
```

If it has not been used for 300 seconds
Or after stays in cache for 600 seconds
When echache exceed from its configured size ehcache removed expired elements
If that doesn't clear enough space it cleasr object from ehcache
By default it remove LRU elements

Second level caching is we put on entity or whole object

```java
@Configuration
@EnableCaching
public class CacheConfig {

    @Bean
    public CacheManager cacheManager() {
        SimpleCacheManager cacheManager = new SimpleCacheManager();
        List<Cache> cacheList = new ArrayList<Cache>();
        cacheList.add(new ConcurrentMapCache("userCache"));
        cacheList.add(new ConcurrentMapCache("addressCache"));
        cacheManager.setCaches(cacheList);
        return cacheManager;
    }
}
```

If we don’t want to configure in ehache.xml

Evict cache -> If we now want to clear cache just evict it

```
@RequestMapping(value = "/greetings")
@Cacheable("greetings")
public Collection<Greeting> getGreetings() throws InterruptedException {
    Thread.sleep(5000);
    return greetingMap.values();
}

@RequestMapping(value = "/greetings", method=RequestMethod.POST)
public String createGreeting(@RequestBody Greeting g) {
    greetingMap.put(g.getId(), g);
    return "Greeting is saved successfully";
}

@RequestMapping(value="/evictcache")
public String evictCache() {
    return "Cache is cleared successfully";
}
```

```
@RestController
 public class CachingController {
@Autowired
 CachingService cachingService;
@GetMapping("clearAllCaches")
public void clearAllCaches() {
 cachingService.evictAllCaches(); } }
```

From <https://www.baeldung.com/spring-boot-evict-cache>

3.4. Sending HTTP Headers using RestTemplate

```
private static void getEmployees()
{
    final String uri = "http://localhost:8080/springrestexample/employees";
    RestTemplate restTemplate = new RestTemplate();

    HttpHeaders headers = new HttpHeaders();
    headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON));
    headers.set("X-COM-PERSIST", "NO");
    headers.set("X-COM-LOCATION", "USA");

    HttpEntity<String> entity = new HttpEntity<String>(headers);

    ResponseEntity<String> response = restTemplate.exchange(uri, HttpMethod.GET, entity, String.class);

    //Use the response.getBody()
}
```

3.5. Sending URL Parameters using RestTemplate

```
private static void getEmployeeById()
{
    final String uri = "http://localhost:8080/springrestexample/employees/{id}";
    RestTemplate restTemplate = new RestTemplate();

    Map<String, String> params = new HashMap<String, String>();
    params.put("id", "1");

    EmployeeVO result = restTemplate.getForObject(uri, EmployeeVO.class, params);

    //Use the result
}
```

From <https://howtodoinjava.com/spring-boot2/resttemplate/spring-restful-client-resttemplate-example/>