**Representational state Transfer(REST)**

State is transferred in some format like json/xml , json is preferred nowadays as xml is verbose .

State as we know is represented by object in object oriented language

Json (Javascript object notation) : lightweight data interchange format

1)Data can be represented as object (syntax similar to javascript object)

Foreg {“id”:1,”name”:”gautam”}

1. Data can be represented as an array of objects

(syntax similar to array of javascript objects)

**[**{“id”:1,”name”:”gautam”}, {“id”:2,”name”:”saran”}**]**

**Advantage of REST**

1. Clear separation of frontend and business logic which simplifies code due to which both can be developed individually at a same time

1. Reduces number of requests because the whole object is returned as state and object can contain a lot of different fields which can be objects themselves.
2. Frontend/Client part is simpilfied since there is no complex business logic if there is a need frontend/client part can make requests to fetch data from more than one services

foreg.

Employee employee=restTemplate.getForObject(**"theserver/employees/8"**,Employee.**class**);

String dptId=employee.getDepartmentId();

Department dpt=restTemplate.getForObject(**"theserverdepartments/"+dptId**, Department.**class**);

**Popular http methods in REST**

**1)GET:** retrieve/fetch resource, this doesnt change state of resource, the request is considered idempotent which means making request again for the same url will return the same result

**2)POST:** Create new resource at server , not idempotent

**3)PUT:** Update existing resource at server. Of course if the resource is immutable(can’t be changed), it can’t be updated,not idempotent

**4)DELETE:**Delete the resource at server , is idempotent

**5)OPTIONS:** used for fetching the options available to work on a resource like the result can be the http methods does this resource supports .

**Popular Http status codes**

**200:** OK

**400:** Bad request

**403:** forbidden

**404:** resource not found

**Remember Error codes start from 400**

**GET Urls used in REST can either follow path variable strategy or request parameter strategy**

1. employees/8
2. **employees?id=8 or even employees?8 , second version assumes 8 is for some default attribute like empid**
3. **Uri is “get” only(let’s assume rest service is simplistic and will give us the default object employee or whatever )**

**Popular convention in REST is plural words should be used in url**

For eg. Url with employees/8 is preferred over employee/8

means from employees not employee give one employee with 8 as some attribute like id

Similarly for post employees/add is preferred over employee/add

means add an employee to into the available store of employees

In that sense employee/add is not right

**Various ways of fetching objects using RestTemplate**

1. using method **getPostForObject()**

Employee employee=restTemplate.getForObject(**"theserver/employees/8"**,Employee.**class**);

1. Using method **getForEntity()**

ResponseEntity<Employee> responseEntity = template.getForEntity(**"theserver/employees/8"**, Employee.**class**);

Employee employee = responseEntity.getBody();

HttpStatus status=responseEntity.getStatusCode();// status can be used too

1. Using method **exchange()**

String url = **"theserver/employees/8"**;  
URI uri=URI.*create*(url);  
RequestEntity<Employee>requestEntity=**new** RequestEntity<>(HttpMethod.***GET***,uri);  
ResponseEntity<Employee> responseEntity = template.exchange(url,HttpMethod.***GET***,requestEntity ,Employee.**class**);

Employee employee = responseEntity.getBody();

**Returning the body and response in rest in one of two ways**

1. Returning response entity object which will wrap the state object and the status code  
   @GetMapping(value = **"/employees/findbyid/{id}"**)  
   **public** ResponseEntity<Employee> findEmployee(@PathVariable(**"id"**) **int** id) {  
    Employee employee = **employeeService**.findEmployeeById(id);  
    ResponseEntity<Employee> responseEntity = **new** ResponseEntity<>(employee, HttpStatus.***OK***);  
    **return** responseEntity;  
   }
2. Returning the state object itself in the method and also using @ResponseStatus annotation to mention httpstatus that will be returned  
   @GetMapping(value = **"/employees/findbyid/{id}"**)  
   @ResponseStatus(HttpStatus.***OK***)  
   **public** Employee findEmployee(@PathVariable(**"id"**) **int** id) {  
    Employee employee = **employeeService**.findEmployeeById(id);  
    **return** employee;  
   }

**Different request mapping options in spring rest**

1. @GetMapping(value = **"/employees/findbyid/{id}"**)

**Or**

@RequestMapping(method = RequestMethod.***GET***,value = **"/employees/findbyid/{id}"**)

@GetMapping works as shortcut for @RequestMapping

Same for other http methods like Post, Put, Delete etc

@PostMapping(value = **"/employee/add"**)

**Or**

@RequestMapping(method = RequestMethod.POST,value = **"/employees/add"**)

**@RestController is a type of Controller when put on top of a Class** provides this information that there are methods defined in this class which will return the response body for a request mapping done with @GetMapping/@RequestMapping etc.