



LAB- Spring Mvc - Rest Basics

In this lab you will be understanding how to expose and consume Restful Webservices using Spring Mvc

STEP -1

In this step, you will be working on **01rest-basics-start** project

Firstly, let us try to understand what we want to achieve.

Assume the we are going to write some services which are used to **manage** a cluster of servers.

We have some servers on our network. we want to perform management of the servers like start/stop the servers, deploying/undeploying applications on servers.

We want to group servers so that we can manage whole group at once . i.e , if, i want to deploy/undeploy an application to all the servers in a group, i should be able to do it just on the group level.

We should be able to create a cluster out of all servers and should be able to manage clusters.

User interface is not designed for this application . We are concentrating only on exposing services.

When we are exposing restful services, the first thing to be done is to identify which resources we need to expose for CRUD operations .i.e , we have to identify nouns

We identified below resources :

- ✓ Clusters
- ✓ Cluster
- ✓ ServerGroups
- ✓ ServerGroup
- ✓ Servers
- ✓ Server
- ✓ Deployments
- ✓ Deployment



Next we have to identify interface for these resources . Since we use http, every resource has a uniform interface.

We identified following operations on each resource

GET /servers --> to retrieve information about all the servers

POST /servers --> to add a new server

PUT /servers --> NOT applicable as we dont want to update all the servers at once

DELETE /servers --> We dont want o delete all the servers at once

GET /servers/{serverId} --> To retrieve information about a server identified by serverId

POST /servers/{serverId} --> NOT supported

PUT /servers/{serverId} --> to update a server identified by serverId

DELETE /servers/{serverId} -- > to delete a server from our management application.

GET /servers/{serverID}/deployments -- >

POST /servers/{serverID}/deployments -- >

PUT /servers/{serverID}/deployments -- >

DELETE /servers/{serverID}/deployments -- >

GET /clusters -- >to retrieve information about all clusters

POST /clusters --> to create a new cluster

PUT /clusters --> NOT Supported

DELETE /clusters -- > NOT Suppoerted

GET /clusters/{clusterID } --> TO retrieve all the information about a cluster identified by ID

POST /clusters/{clusterID} --> NOT supported

PUT /clusters /{clusterID} --> update a cluster identified by clusterID

DELETE /clusters/{clusterID} --> to delete a cluster identified by clusterID

GET /clusters/{clusterID }/deployments

POST /clusters/{clusterID }/deployments

PUT /clusters/{clusterID }/deployments

DELETE /clusters/{clusterID }/deployments

GET /clusters/{clusterID}/servers --> to retrieve all the servers of a cluster identified by clusterID

POST /clusters/{clusterID}/servers --> to add a new server to the existing servers of cluster identified by clusterID

PUT /clusters/{clusterID}/servers --> NOT supported

DELETE /clusters/{clusterID}/servers -->Delete all the servers belonging to a cluster identified by clusterID .(This is just to delete the server from cluster. Server will not be deleted)

Can you come up with the meanings of the below URLS and decided which operations makes sense for each URL



```
GET /clusters/{clusterID}/servers/{serverID}
POST /clusters/{clusterID}/servers/{serverID}
PUT /clusters/{clusterID}/servers/{serverID}
DELETE /clusters/{clusterID}/servers/{serverID}
```

```
GET/deployments
```

```
GET/deployments/{deploymentID}
PUT /deployments/{deploymentID}
DELETE /deployments/{deploymentID}
```

```
GET /servergroups
POST /servergroups
PUT /servergroups
DELETE /servergroups
```

```
GET /servergroups/{serverGroupID}
POST /servergroups/{serverGroupID}
PUT /servergroups/{serverGroupID}
DELETE /servergroups/{serverGroupID}
```

```
GET /servergroups/{serverGroupID}/servers
POST /servergroups/{serverGroupID}/servers
PUT /servergroups/{serverGroupID}/servers
DELETE /servergroups/{serverGroupID}/servers
```

Next we have to decide which representations we want to support. We will support on XML and Json representations only

Let us start implementing it.

1) Open Server.java, ServerGroup.java, Cluster.java and Deployment.java in com.way2learnonline.model package and observe how they are mapped to tables using JPA .

Take some time to understand the mappings and database structure

2) We want to use Spring data repositories. So, we have created interfaces for our entities in package com.way2learnonline.repository . Open ServerRepository.java, ServerGroupRepository.java, ClusterRepository.java and DeploymentRepository.java and observe them

3) We don't want to use same data model classes in our presentation layer. We want presentation layer to evolve independently. So, we have create DTOs for our entities.



Open ServerDTO.java, ServerGroupDTO.java ,ClusterDTO.java and ClusterGroupDTO.java and observe the code

4) Open ServerController.java and observe that `ServerRepository` are autowired.

complete TODO -1 by mapping `ServerController` to `/servers`.

Complete TODO-2 on `getAllServers()` method

Make sure to annotate the return value with appropriate annotation so that a `HttpMessageConverter` will be used to convert return value into a required representation

Observe the code inside the method `getAllServers()` and understand how `Server` is converted to `ServerDTO`

5) Complete TODO-3 . Observe that `getServerById` method is throwing `ServerNotFoundException` id server with the given `serverID` is not found.

How do we map the Exception to status code ?

Open `ServerNotFoundException.java` and complete TODO-4

6) Complete TODO -5 inside `ServerController`

If the resource is created successfully after a POST request, the response should contain a `Location` header with the value of newly created resource URL.

So, complete TODO-6 by using below code :

```
ServletUriComponentsBuilder.fromCurrentRequest()
    .path("/{id}")
    .buildAndExpand(server.getServerId())
    .toUri();
```

Observe how `ResponseEntity` is created with status code and headers.

7) We want to populate some dummy data for testing the application.

So, Open `TestDataPopulator.java` and Observe the code.

Run it and make sure that data is populated in database. Go to database ad check data.



8) We are done with implementing the ServerController as of now.

what is the starting point when i deploy this application ?

Open WebInitializer.java and observe the configuration.

9) Deploy the application .

Open **PostMan** and give request to /servers. Add Accept header as application/json

Did you observe the expected result?

Similarly test for GET /servers/1 and also POST /servers

Make sure that Content-Type header is set as application/json or when making POST request .

Send server representation in json format similar to the one u got when u gave GET for /servers/1

Writing Client using RestTemplate

1) Open Client.java and complete all the TODOs

Optional

1) Complete TODO-7 to TODO-14 inside ClusterController.java



Congratulations !! You know how to expose and consume rest services using spring mvc