Spring MVC REST

Steps in designing Rest Applications

- **Restful Application design typically has following steps :**
  - ✓ Resource Identification
  - ✓ Resource Representation
  - ✓ Endpoint Identification
  - ✓ Verb/Action Identification

Resource Identification

- **Following are the resources(nouns)**

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

Get /servers implementations

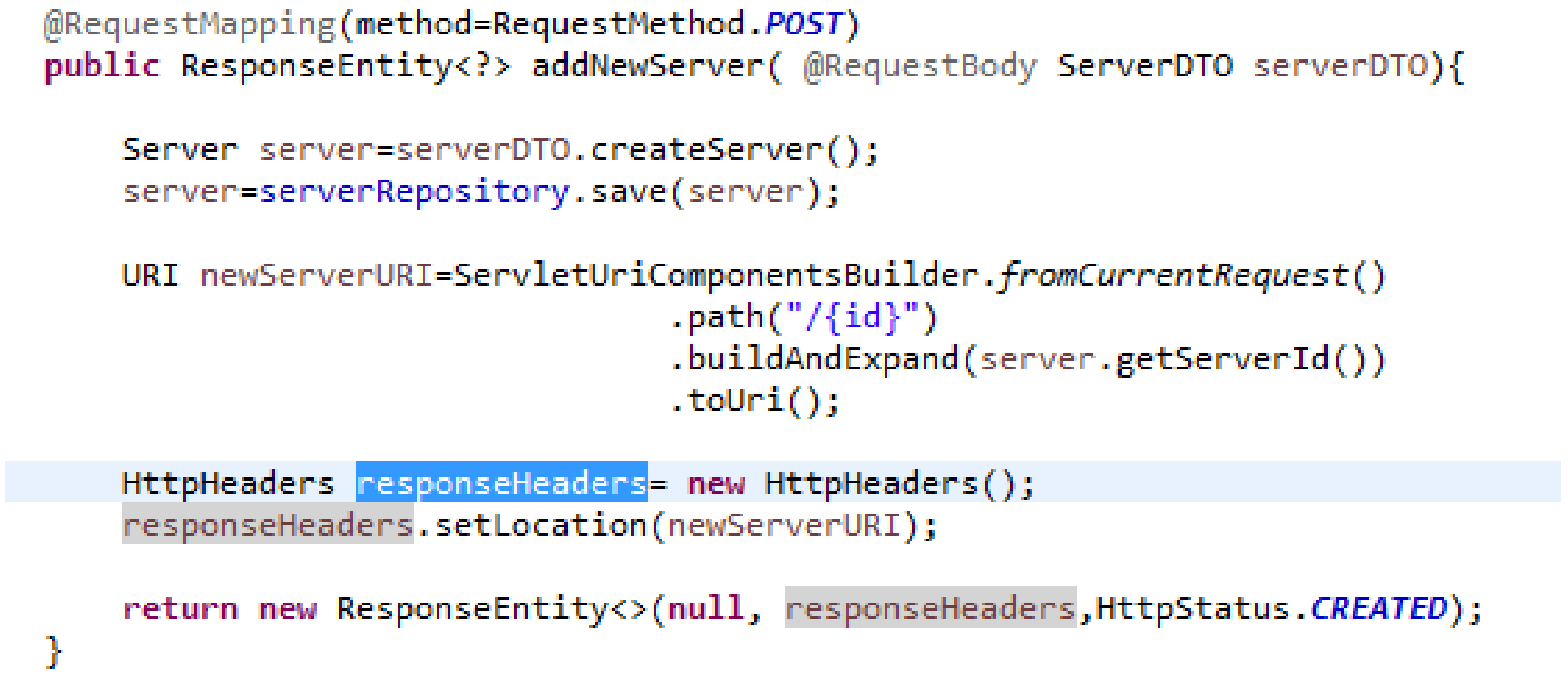
```
@Controller
@RequestMapping("/servers")
public class ServerController {

    @Autowired
    private ServerRepository serverRepository;

    @RequestMapping(method=RequestMethod.GET)
    public @ResponseBody List<ServerDTO> getAllServers(){
        Iterable<Server>  servers=serverRepository.findAll();

        List<ServerDTO> serverDTOs= new ArrayList<ServerDTO>();

        for(Server server:servers){
            serverDTOs.add(ServerDTO.createServerDTO(server));
        }
        return serverDTOs;
    }
```

GET /servers/{id} example

```
@RequestMapping(value="/{serverId}",method=RequestMethod.GET)
public @ResponseBody ServerDTO getServerById(@PathVariable("serverId") Long serverId){
    Server server=serverRepository.findOne(serverId);
    if(server==null){
        throw new ServerNotFoundException("Server with Id "+serverId+" is Not found!!");
    }
    return ServerDTO.createServerDTO(server);
}
```

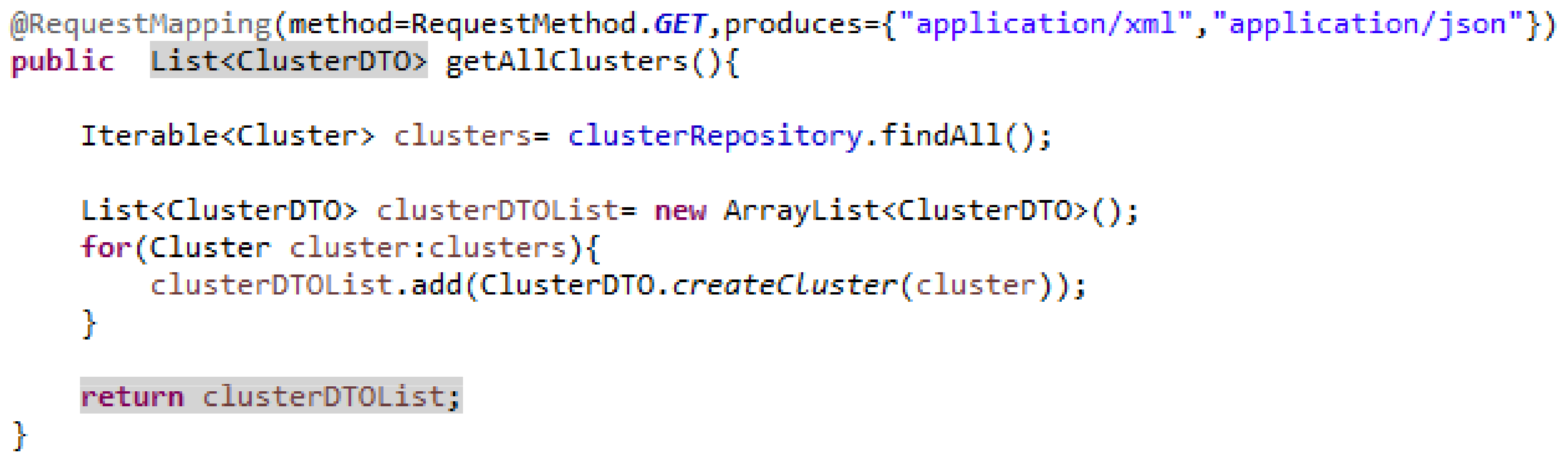
WAY2LEARN

POST /servers



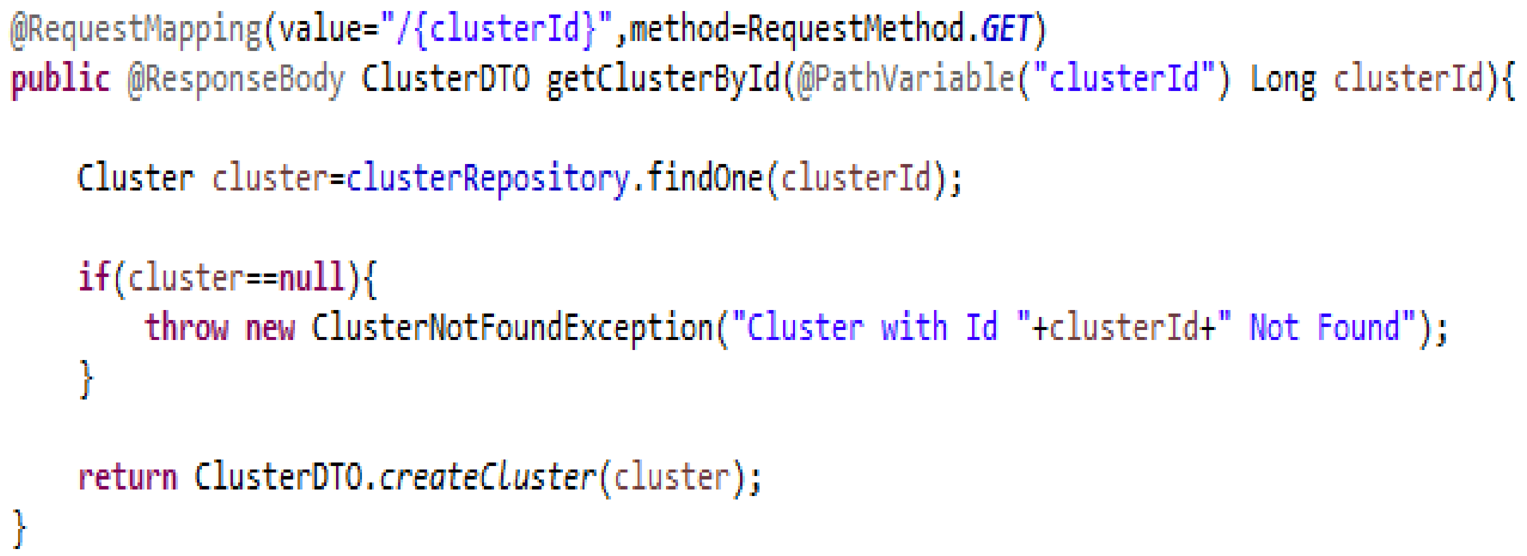
```
@RequestMapping(method=RequestMethod.POST)
public ResponseEntity<?> addNewServer( @RequestBody ServerDTO serverDTO){

    Server server=serverDTO.createServer();
    server=serverRepository.save(server);

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

    HttpHeaders responseHeaders= new HttpHeaders();
    responseHeaders.setLocation(newServerURI);

    return new ResponseEntity<>(null, responseHeaders,HttpStatus.CREATED);
}
```

GET /clusters



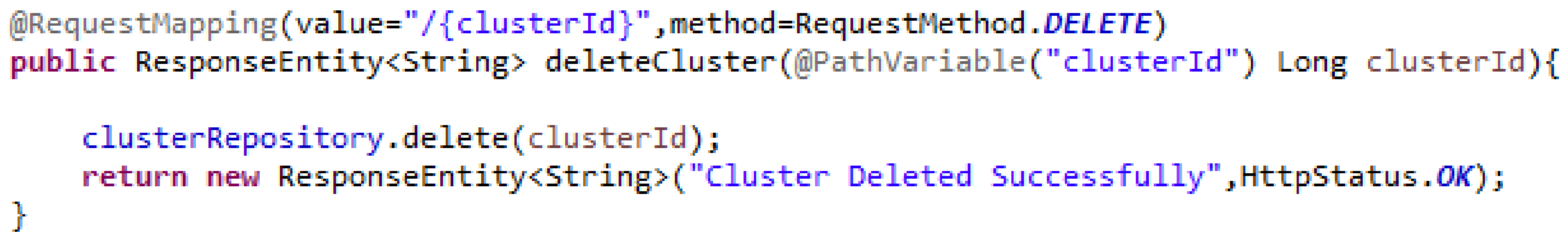
```
@RequestMapping(method=RequestMethod.GET,produces={"application/xml","application/json"})
public  List<ClusterDTO> getAllClusters(){

    Iterable<Cluster> clusters= clusterRepository.findAll();

    List<ClusterDTO> clusterDTOList= new ArrayList<ClusterDTO>();
    for(Cluster cluster:clusters){
        clusterDTOList.add(ClusterDTO.createCluster(cluster));
    }

    return clusterDTOList;
}
```

GET /clusters/{id}



```
@RequestMapping(value="/{clusterId}",method=RequestMethod.GET)
public @ResponseBody ClusterDTO getClusterById(@PathVariable("clusterId") Long clusterId){

    Cluster cluster=clusterRepository.findOne(clusterId);

    if(cluster==null){
        throw new ClusterNotFoundException("Cluster with Id "+clusterId+" Not Found");
    }

    return ClusterDTO.createCluster(cluster);
}
```

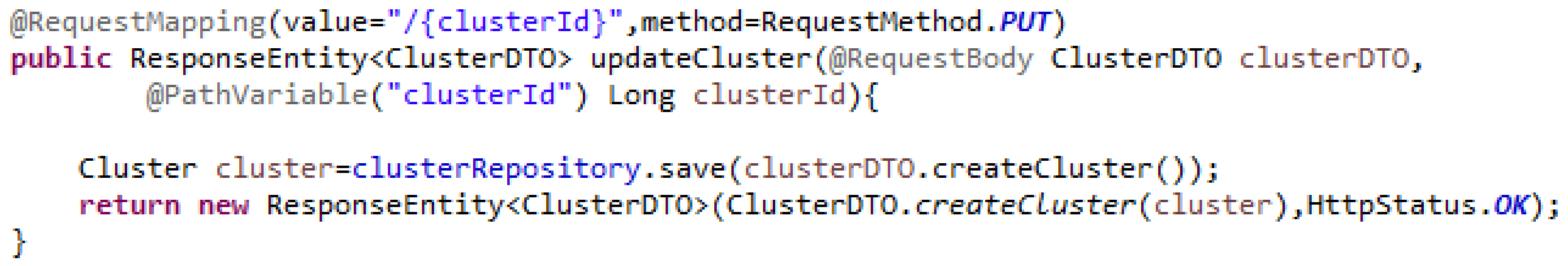
DELETE /clusters/{id}



```
@RequestMapping(value="/{clusterId}",method=RequestMethod.DELETE)
public ResponseEntity<String> deleteCluster(@PathVariable("clusterId") Long clusterId){

    clusterRepository.delete(clusterId);
    return new ResponseEntity<String>("Cluster Deleted Successfully",HttpStatus.OK);
}
```

Sivaprasad.valluru@gmail.com

PUT /clusters/{id}



```java
@RequestMapping(value="/{clusterId}",method=RequestMethod.PUT)
public ResponseEntity<ClusterDTO> updateCluster(@RequestBody ClusterDTO clusterDTO,
        @PathVariable("clusterId") Long clusterId){

    Cluster cluster=clusterRepository.save(clusterDTO.createCluster());
    return new ResponseEntity<ClusterDTO>(ClusterDTO.createCluster(cluster),HttpStatus.OK);
}
```

Sivaprasad.valluru@gmail.com

GET all server belonging to a cluster

```
@RequestMapping(value="/{clusterId}/servers",method=RequestMethod.GET)
public @ResponseBody List<ServerDTO> getAllServers(@PathVariable("clusterId") Long clusterId){

    Iterable<Server> servers=serverRepository.findServersByClusterId(clusterId);

    List<ServerDTO> serverDTOs= new ArrayList<ServerDTO>();

    for(Server server:servers){
        serverDTOs.add(ServerDTO.createServerDTO(server));
    }
    return serverDTOs;

}
```

```java
@RequestMapping(value="/{clusterId}/servers",method=RequestMethod.POST)
public ResponseEntity<?> addServerToCluster(@RequestBody ServerDTO server,
                                @PathVariable("clusterId") Long clusterId){
    Cluster cluster=clusterRepository.findOne(clusterId);

    cluster.addServer(server.createServer());

    clusterRepository.save(cluster);
    URI newserverURI=ServletUriComponentsBuilder.fromCurrentRequest()
            .path("/{id}")
            .buildAndExpand(server.getServerId())
            .toUri();

    HttpHeaders responseHeaders= new HttpHeaders();
    responseHeaders.setLocation(newserverURI);

    return new ResponseEntity<>(null, responseHeaders,HttpStatus.CREATED);
}
```

CLIENT USING REST TEMPLATE

Get request with out response headers

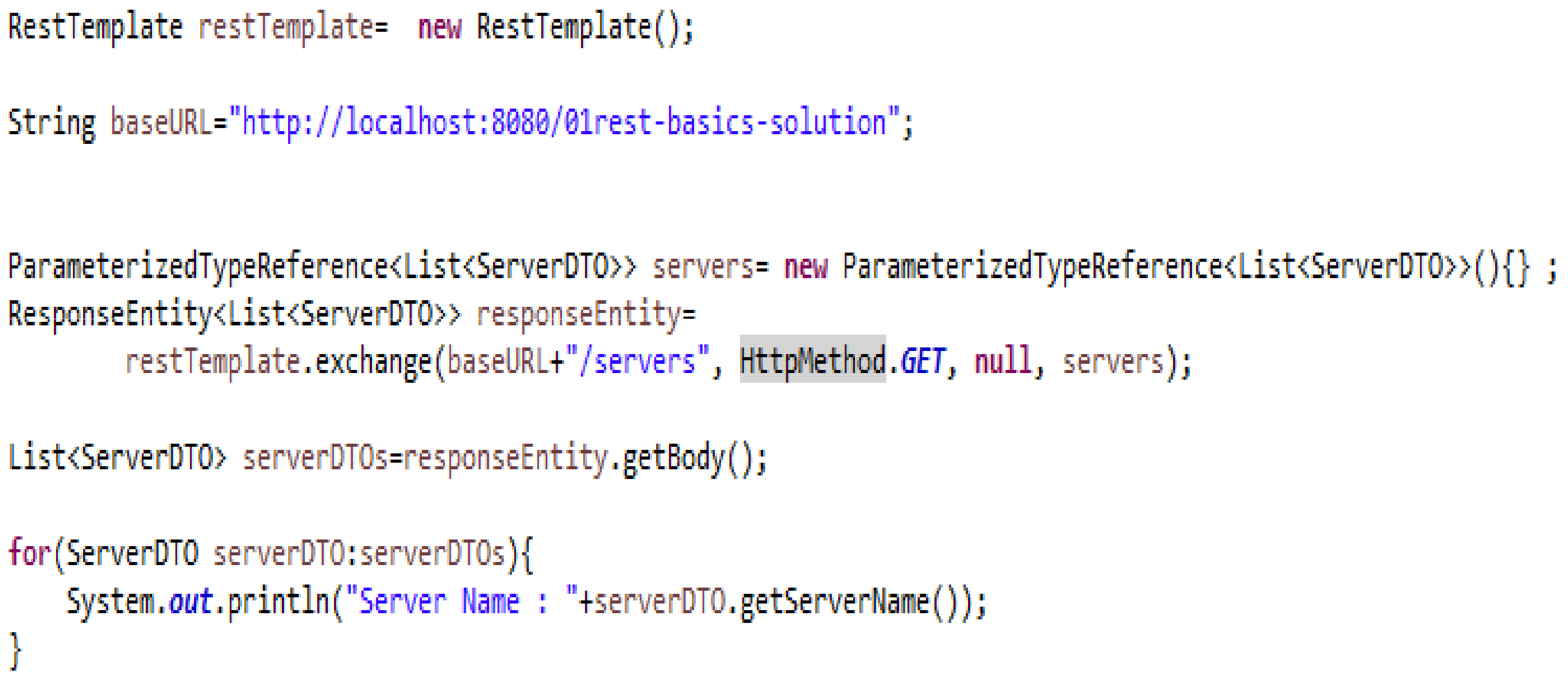
```
RestTemplate restTemplate=  new RestTemplate();

String baseURL="http://localhost:8080/01rest-basics-solution";


String getServerByIdURL=baseURL+"/servers/{id}";
ServerDTO serverDTO=restTemplate.getForObject(getServerByIdURL, ServerDTO.class,1L);

System.out.println("Server Name :"+serverDTO.getServerId());
```

Get Request with response headers

```java
RestTemplate restTemplate=  new RestTemplate();

String baseURL="http://localhost:8080/01rest-basics-solution";

String getServerByIdURL=baseURL+"/servers/{id}";
ResponseEntity<ServerDTO> responseEntity=restTemplate.getForEntity(getServerByIdURL, ServerDTO.class,1L);

HttpHeaders httpHeaders=responseEntity.getHeaders();

System.out.println("Headers: "+httpHeaders);

ServerDTO serverDTO=responseEntity.getBody();

System.out.println("Server Name :"+serverDTO.getServerName());
```

Sivaprasad.valluru@gmail.com

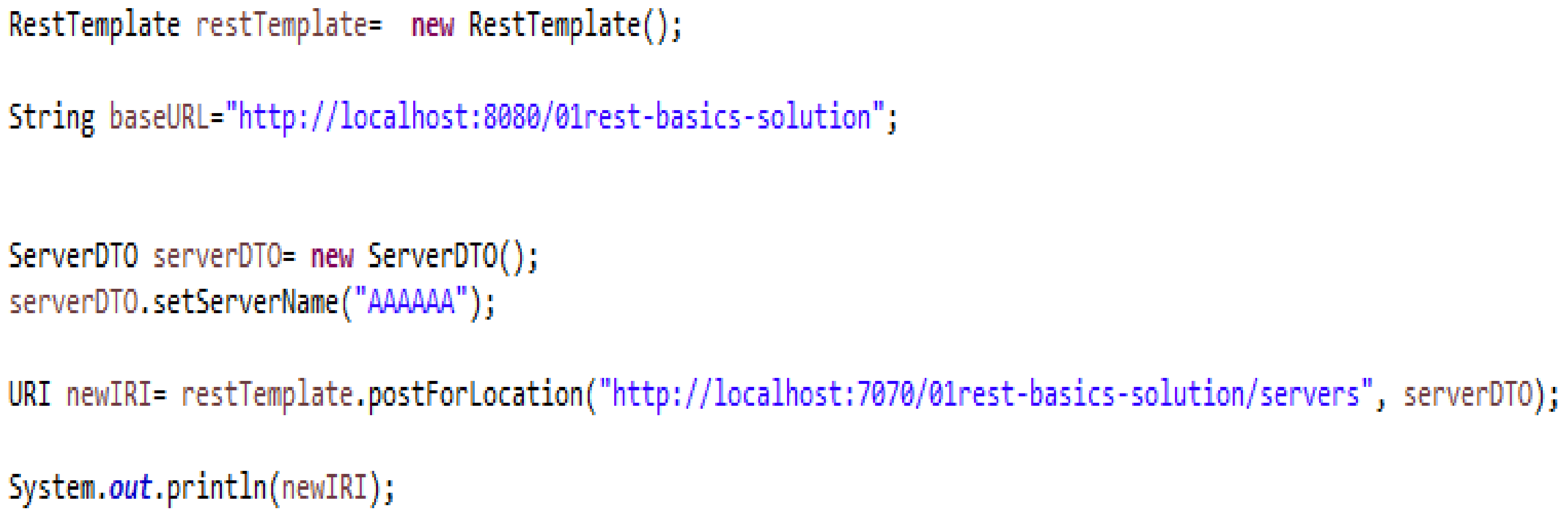
Using ParameterizedTypeReference



```java
RestTemplate restTemplate=  new RestTemplate();

String baseURL="http://localhost:8080/01rest-basics-solution";


ParameterizedTypeReference<List<ServerDTO>> servers= new ParameterizedTypeReference<List<ServerDTO>>(){} ;
ResponseEntity<List<ServerDTO>> responseEntity=
        restTemplate.exchange(baseURL+"/servers", HttpMethod.GET, null, servers);

List<ServerDTO> serverDTOs=responseEntity.getBody();

for(ServerDTO serverDTO:serverDTOs){
    System.out.println("Server Name : "+serverDTO.getServerName());
}
```

Sivaprasad.valluru@gmail.com

POST with out headers



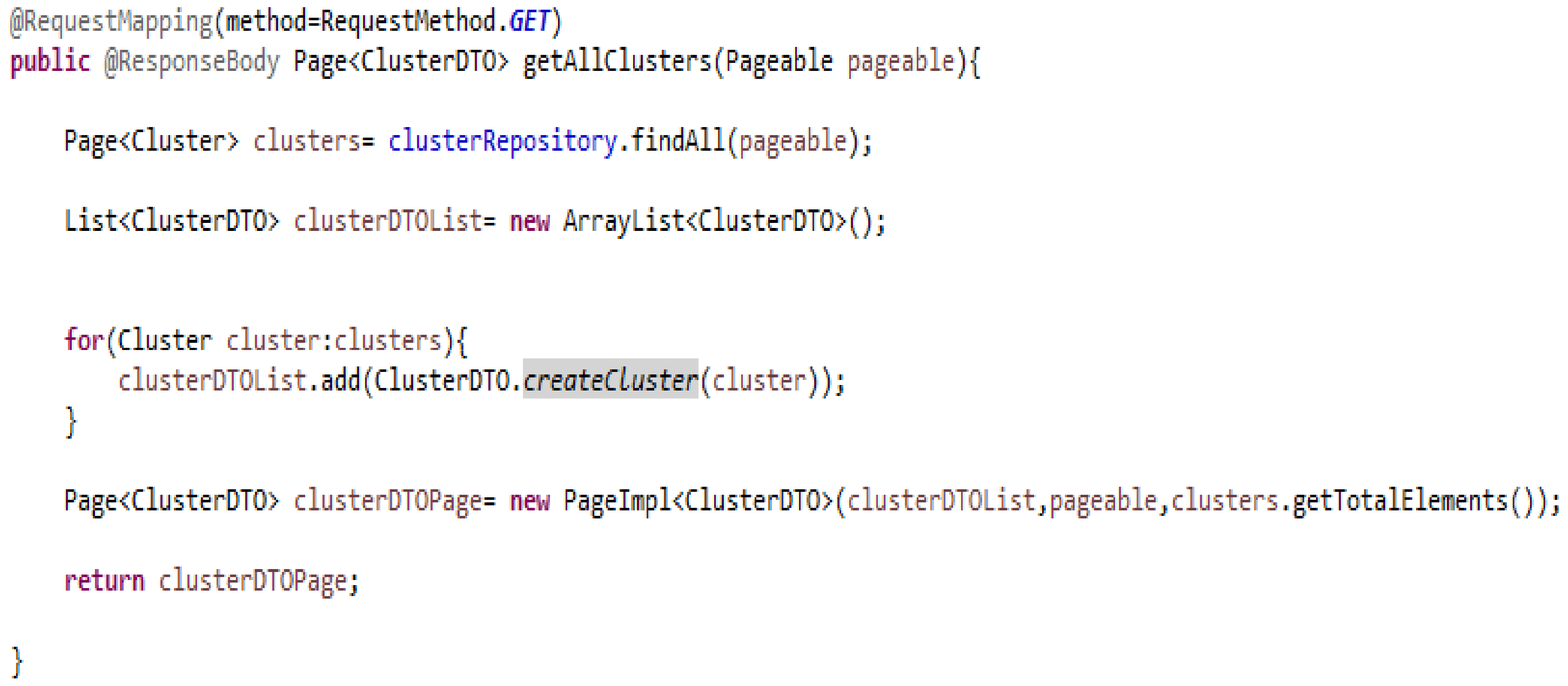
```
RestTemplate restTemplate=  new RestTemplate();

String baseURL="http://localhost:8080/01rest-basics-solution";


ServerDTO serverDTO= new ServerDTO();
serverDTO.setServerName("AAAAAA");

URI newIRI= restTemplate.postForLocation("http://localhost:7070/01rest-basics-solution/servers", serverDTO);

System.out.println(newIRI);
```

POST with headers

```
RestTemplate restTemplate=  new RestTemplate();

String baseURL="http://localhost:8080/01rest-basics-solution";

ServerDTO serverDTO= new ServerDTO();
serverDTO.setServerName("AAAAAA");

ResponseEntity<Object> responseEntity=restTemplate.postForEntity(baseURL+"/servers", serverDTO, null);
HttpHeaders httpHeaders=responseEntity.getHeaders();
System.out.println(httpHeaders);
System.out.println(responseEntity.getBody());
```

PAGING

Implementing Paging



```java
@RequestMapping(method=RequestMethod.GET)
public @ResponseBody Page<ClusterDTO> getAllClusters(Pageable pageable){

    Page<Cluster> clusters= clusterRepository.findAll(pageable);

    List<ClusterDTO> clusterDTOList= new ArrayList<ClusterDTO>();


    for(Cluster cluster:clusters){
        clusterDTOList.add(ClusterDTO.createCluster(cluster));
    }

    Page<ClusterDTO> clusterDTOPage= new PageImpl<ClusterDTO>(clusterDTOList,pageable,clusters.getTotalElements());

    return clusterDTOPage;

}
```

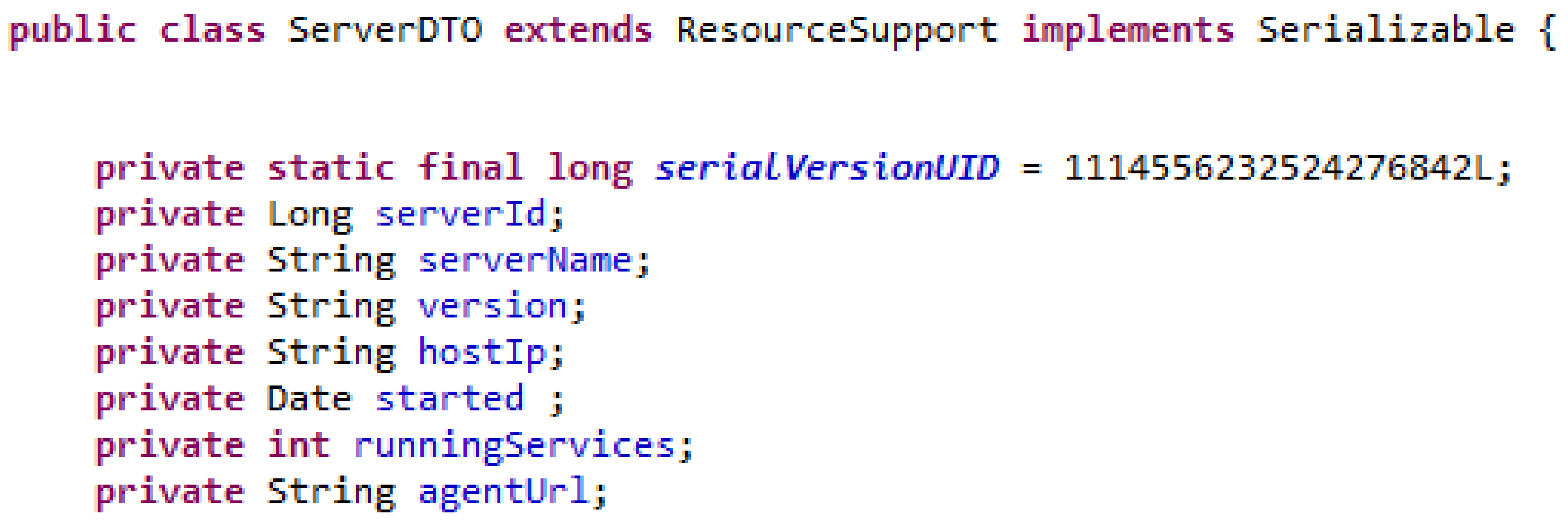
Customizing paging

```java
@Configuration
@EnableAutoConfiguration
@ComponentScan
public class RestConfig  extends WebMvcConfigurerAdapter{

    @Override
    public void addArgumentResolvers(List<HandlerMethodArgumentResolver> argumentResolvers) {
        PageableHandlerMethodArgumentResolver phmar = new
                PageableHandlerMethodArgumentResolver();
                // Set the default size to 5
                phmar.setFallbackPageable(new PageRequest(0, 2));
                argumentResolvers.add(phmar);
        super.addArgumentResolvers(argumentResolvers);
    }

}
```

SPRING-HATEOS

ResourceSupport

- **Make all your Resource classes to extend ResourceSupport**



```java
public class ServerDTO extends ResourceSupport implements Serializable {

    private static final long serialVersionUID = 1114556232524276842L;
    private Long serverId;
    private String serverName;
    private String version;
    private String hostIp;
    private Date started ;
    private int runningServices;
    private String agentUrl;
```

Creating Links

```
Link link = linkTo(PersonController.class).slash(person.getId()).withSelfRel();
```

- **If your domain class implements the Identifiable interface the slash(…) method will rather invoke getId() on the given object instead of toString()**

```
class Person implements Identifiable<Long> {
  public Long getId() { … }
}

Link link = linkTo(PersonController.class).slash(person).withSelfRel();
```

Link to URI

- **The builder also allows creating URI instances to build up e.g. response header values:**

```
HttpHeaders headers = new HttpHeaders();
headers.setLocation(linkTo(PersonController.class).slash(person).toUri());
return new ResponseEntity<PersonResource>(headers, HttpStatus.CREATED);
```

```java
Link link = linkTo(methodOn(PersonController.class).show(2L)).withSelfRel();
assertThat(link.getHref(), is("/people/2")));
```

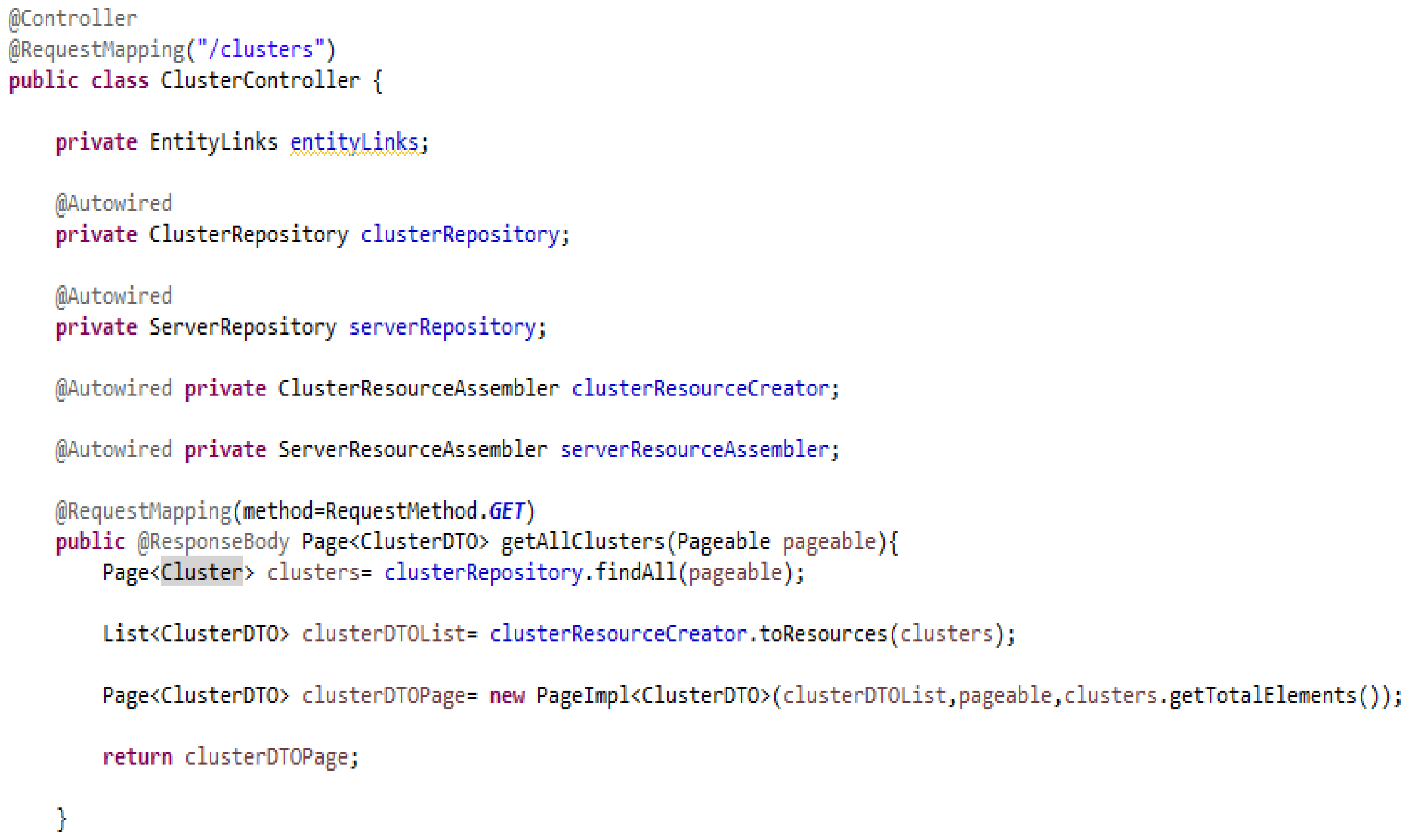
```
public static ClusterDTO toResource(Cluster cluster){

    ClusterDTO clusterDTO=ClusterDTO.createCluster(cluster);


    Link selfLink=ControllerLinkBuilder.linkTo(
            ControllerLinkBuilder.methodOn(ClusterController.class).getClusterById(clusterDTO.getClusterId()))
            .withSelfRel();
    clusterDTO.add(selfLink);

    Link serversLink=ControllerLinkBuilder.linkTo(
                            ControllerLinkBuilder.methodOn(ClusterController.class)
                                                    .getAllServers(clusterDTO.getClusterId())
                                            ).withRel("servers");
        clusterDTO.add(serversLink);

        return clusterDTO;
}
```

```java
public static List<ClusterDTO> toResources(Iterable<Cluster> clusters){
    List<ClusterDTO> clusterDTOs= new ArrayList<ClusterDTO>();

    for(Cluster cluster:clusters){
        clusterDTOs.add(toResource(cluster));
    }
    return clusterDTOs;
}
```

RESOURCE ASSEMBLER

```java
@Component
public class ClusterResourceAssembler extends ResourceAssemblerSupport<Cluster, ClusterDTO> {


    public ClusterResourceAssembler() {
        super(ClusterController.class, ClusterDTO.class);
    }

    @Override
    public ClusterDTO toResource(Cluster cluster) {
        ClusterDTO clusterDTO=  ClusterDTO.createCluster(cluster);

        Link serversLink=ControllerLinkBuilder.linkTo(
                ControllerLinkBuilder.methodOn(ClusterController.class)
                                    .getAllServers(clusterDTO.getClusterId())
                        ).withRel("servers");

        clusterDTO.add(serversLink);

        return clusterDTO;
    }
}
```

Using Resource Assembler



```
@Controller
@RequestMapping("/clusters")
public class ClusterController {

    private EntityLinks entityLinks;

    @Autowired
    private ClusterRepository clusterRepository;

    @Autowired
    private ServerRepository serverRepository;

    @Autowired private ClusterResourceAssembler clusterResourceCreator;

    @Autowired private ServerResourceAssembler serverResourceAssembler;

    @RequestMapping(method=RequestMethod.GET)
    public @ResponseBody Page<ClusterDTO> getAllClusters(Pageable pageable){
        Page<Cluster> clusters= clusterRepository.findAll(pageable);

        List<ClusterDTO> clusterDTOList= clusterResourceCreator.toResources(clusters);

        Page<ClusterDTO> clusterDTOPage= new PageImpl<ClusterDTO>(clusterDTOList,pageable,clusters.getTotalElements());

        return clusterDTOPage;

    }
```

```java
@RequestMapping(value="/{clusterId}",method=RequestMethod.GET)
public @ResponseBody ClusterDTO getClusterById(@PathVariable("clusterId") Long clusterId){

    Cluster cluster=clusterRepository.findOne(clusterId);

    if(cluster==null){
        throw new ClusterNotFoundException("Cluster with Id "+clusterId+" Not Found");
    }

    return clusterResourceCreator.toResource(cluster);
}
```

SPRING DATA REST