- Implement the basic atributes to DTO
    - Basic Attributes
    - Associations (Instantiate collections)
    - Constructors
    - Getters & Setters (collections: only get)
    - Serializable

nt.java | ClientResource.java | ClientRepository.java | ClientService.java | application-test.properties | application.properties | import.sql | *ClientDTO.java

```java
package com.restcrudbasics.client.dto;

import java.time.Instant;

public class ClientDTO {

    private Long id;
    private String name;
    private String cpf;
    private Double income;
    private Instant birthDate;
    private Integer children;

    public ClientDTO() {
    }

    public ClientDTO(Long id, String name, String cpf, Double income, Instant birthDate, Integer children) {
        this.id = id;
        this.name = name;
        this.cpf = cpf;
        this.income = income;
        this.birthDate = birthDate;
        this.children = children;
    }
```

```java
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public Double getIncome() {
        return income;
    }

    public void setIncome(Double income) {
        this.income = income;
    }

    public Instant getBirthDate() {
        return birthDate;
    }
```

- Now, make a constructor that receive the entity Client

```java
public ClientDTO(Client client) {
    this.id = client.getId();
    this.name = client.getName();
    this.cpf = client.getCpf();
    this.income = client.getIncome();
    this.birthDate = client.getBirthDate();
    this.children = client.getChildren();
}
```

- Now, change the Service
- The repository, work just with Entity, not DTO. You have to modif the code to convert Entity to DTO

```java
@Transactional(readOnly = true)
public List<CategoryDTO> findAll() {
    List<Category> list = repository.findAll();

    List<CategoryDTO> listDto = new ArrayList<>();
    for (Category cat : list) {
        listDto.add(new CategoryDTO(cat));
    }

    return listDto;
}
```

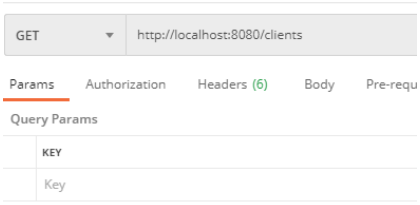
- We can use a lambda expression.

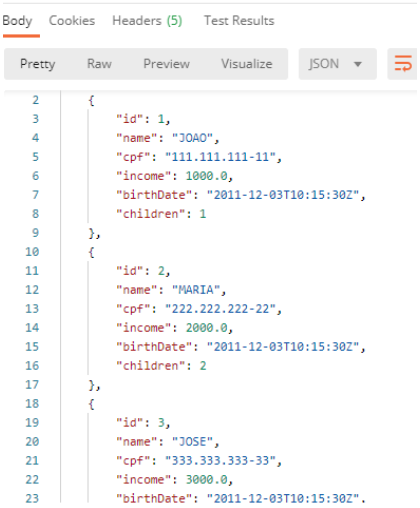
```java
package com.restcrudbasics.client.services;

import java.util.List;
import java.util.stream.Collectors;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import com.restcrudbasics.client.dto.ClientDTO;
import com.restcrudbasics.client.repositories.ClientRepository;
import com.restcrudbasics.client.entities.Client;

@Service // this annotation will register the class ClientService as a dependency injection component and will be management by spring
public class ClientService {

    @Autowired //This annotation will make this instance auto injected (Remember the @Repository annotation in ClientRepository class)
    private ClientRepository repository;

    @Transactional(readOnly = true) // Transactional close when make the REST request. Good Practice in Program
    public List<ClientDTO> findAll(){
        List<Client> list = repository.findAll(); // We have to convert this Client list to ClientDTO list.
        return list.stream().map(x -> new ClientDTO(x)).collect(Collectors.toList()); // Use Functional program
    }

}
```

- Now, change the resource

```java
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.restcrudbasics.client.dto.ClientDTO;
import com.restcrudbasics.client.services.ClientService;

@RestController
@RequestMapping(value ="/clients") //the REST route when you make the requisition. Use the plural form of the entity
public class ClientResource {

    @Autowired //This annotation will make this instance auto injected (Remember the @Service annotation in ClientService class)
    private ClientService service;

    @GetMapping //endpoint GET
    public ResponseEntity<List<ClientDTO>> findAll(){ //This is a endpoint to find a client by id
        List<ClientDTO> list = service.findAll();
        return ResponseEntity.ok().body(list);
    }

}
```

- Make a test





- Now, we implement all the web service architecture