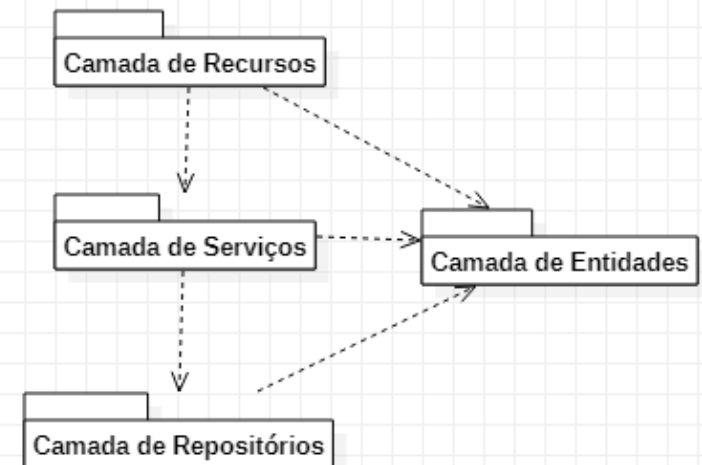


# Camada de Serviço



# Introdução

- Até o momento implementamos as classes A (camada de entidades), RecursoA (camada de recursos) e a interface RepositorioA (camada de repositórios).
- Nessa aula iremos implementar a classe ServicoA que será o representante da camada de serviços.
- A separação entre recursos e serviços é interessante para colocar algumas regras de negócio na camada de serviço (fluxo de casos de uso).
- A Camada de Recursos é a camada “porta de entrada” do sistema, nela colocamos como será feito acesso as funcionalidades (camada de controle de interação).
- A Camada de Serviços trata do fluxo de casos de uso (camada de gerência de tarefas).
- A Camada de Repositórios trata da persistência de dados (camada de gerência de dados).
- A Camada de Entidades trata das classes de domínio do problema (camada de domínio do problema).



# ServicoA e obterTodos

- O uso de **@Service** é para registrar essa classe no Spring (sem isso o Spring não consegue instanciar essa classe lá no RecursoA).
- O **@Autowired** de repositório irá garantir a instanciação desse objeto.
- Na sequência temos a definição de um método que retorna a lista de todos os “As” usando o objeto de repositório (que é um objeto padrão que fará a busca dos “As”)

```
import java.util.List;

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

import com.servicos.abcde.entidades.A;
import com.servicos.abcde.repositorios.RepositorioA;

@Service
public class ServicoA {

    // Fazendo a Injeção de Dependência
    @Autowired
    private RepositorioA repositorio;

    public List<A> obterTodos(){
        return repositorio.findAll();
    }
}
```



# Modificando o RecursoA

```
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.servicos.abcde.entidades.A;
import com.servicos.abcde.servicos.ServicoA;

@RestController
@RequestMapping(value = "/as")
public class RecursoA {

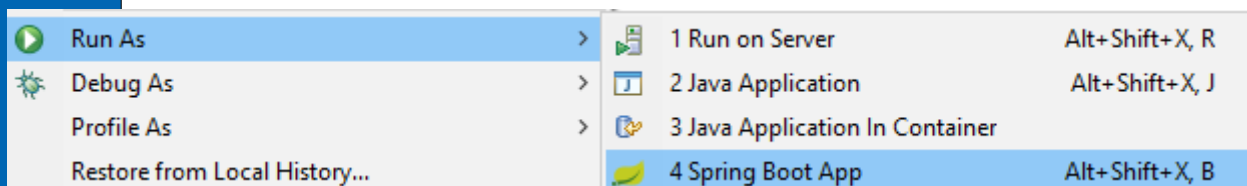
    @Autowired
    private ServicoA servico;

    @GetMapping
    public ResponseEntity<List<A>> obterTodos(){
        List<A> lista = servico.obterTodos();
        return ResponseEntity.ok().body(lista);
    }
}
```

- **RecursoA** agora precisa instanciar o serviço, por isso o uso de **@Autowired** (antes estávamos gerando um único objeto da classe A “na mão” e retornando no método obterTodos).
- Notar a chamada de obterTodos do objeto de serviço instanciado.
- Foi necessário modificar o retorno do método pois ao invés de retornar um A passamos a retornar uma lista de A.



# Rodando a aplicação



```
Console
ibcde - AbcdeApplication [Spring Boot App]

:: Spring Boot :: (v2.5.0)

2021-06-15 20:04:31.664 INFO 10040 --- [main] com.servicos.abcde.AbcdeApplication : Starting AbcdeApplication using Java 15.0.2 on DESKTOP-JKB61SJ with PID 1004
2021-06-15 20:04:31.672 INFO 10040 --- [main] com.servicos.abcde.AbcdeApplication : The following profiles are active: test
2021-06-15 20:04:33.440 INFO 10040 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-06-15 20:04:33.595 INFO 10040 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 132 ms. Found 1 JPA repository i
2021-06-15 20:04:35.271 INFO 10040 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-06-15 20:04:35.296 INFO 10040 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-06-15 20:04:35.297 INFO 10040 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.46]
2021-06-15 20:04:35.828 INFO 10040 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2021-06-15 20:04:35.828 INFO 10040 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 4012 ms
2021-06-15 20:04:35.959 INFO 10040 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2021-06-15 20:04:36.595 INFO 10040 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2021-06-15 20:04:36.613 INFO 10040 --- [main] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:ab
2021-06-15 20:04:36.994 INFO 10040 --- [main] org.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2021-06-15 20:04:37.127 INFO 10040 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 5.4.31.Final
2021-06-15 20:04:37.592 INFO 10040 --- [main] org.hibernate.annotations.common.Version : HCNAN000001: Hibernate Commons Annotations {5.1.2.Final}
2021-06-15 20:04:38.018 INFO 10040 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.H2Dialect

hibernate: drop table if exists tb_a CASCADE
hibernate: create table tb_a (id bigint generated by default as identity, atributo1 varchar(255), atributo2 double, atributo3 timestamp, atributo4 integer, primary key (id))
2021-06-15 20:04:39.409 INFO 10040 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transacti
2021-06-15 20:04:39.432 INFO 10040 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2021-06-15 20:04:40.850 INFO 10040 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-06-15 20:04:40.880 INFO 10040 --- [main] com.servicos.abcde.AbcdeApplication : Started AbcdeApplication in 10.194 seconds (JVM running for 12.246)
2021-06-15 20:04:40.883 INFO 10040 --- [main] o.s.b.a.ApplicationAvailabilityBean : Application availability state LivenessState changed to CORRECT
2021-06-15 20:04:41.064 INFO 10040 --- [main] o.s.b.a.ApplicationAvailabilityBean : Application availability state ReadinessState changed to ACCEPTING_TRAFFIC
2021-06-15 20:04:58.563 INFO 10040 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2021-06-15 20:04:58.564 INFO 10040 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-06-15 20:04:58.566 INFO 10040 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms
hibernate: select a0_.id as id1_0_, a0_.atributo1 as atributo2_0_, a0_.atributo2 as atributo3_0_, a0_.atributo3 as atributo4_0_, a0_.atributo4 as atributo5_0_ from tb_a a0_
```



# Testando a aplicação

localhost:8080/as

localhost:8080/as

[[{"id":1,"atributo1":"Atributo 11","atributo2":7.77,"atributo3":"2021-06-15T23:04:40Z","atributo4":7},{ "id":2,"atributo1":"Atributo 12","atributo2":2.77,"atributo3":"2020-07-20T19:51:09Z","atributo4":2}]]

http://localhost:8080/as

GET http://localhost:8080/as

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 51 ms Size: 367 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 1,
4     "atributo1": "Atributo 11",
5     "atributo2": 7.77,
6     "atributo3": "2021-06-15T23:04:40Z",
7     "atributo4": 7
8   },
9   {
10    "id": 2,
11    "atributo1": "Atributo 12",
12    "atributo2": 2.77,
13    "atributo3": "2020-07-20T19:51:09Z",
14    "atributo4": 2
15  }
16 }
```

```
@RestController
@RequestMapping(value = "/as")
public class RecursoA {

    @Autowired
    private ServicoA servico;

    @GetMapping
    public ResponseEntity<List<A>> obterTodos(){
        List<A> lista = servico.obterTodos();
        return ResponseEntity.ok().body(lista);
    }
}
```



# obterPorId

```
@Service
public class ServicoA {

    // Fazendo a Injeção de Dependência
    @Autowired
    private RepositorioA repositorio;

    public List<A> obterTodos(){
        return repositorio.findAll();
    }

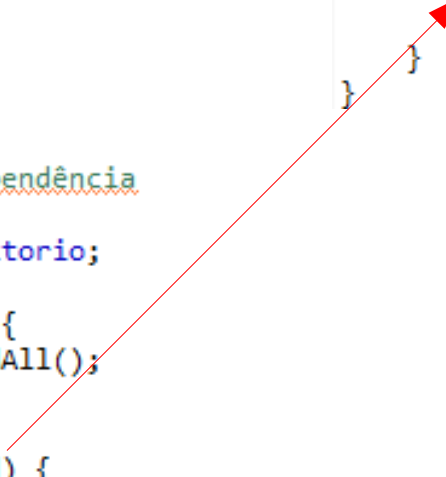
    public A obterPorId(Long id) {
        Optional<A> obj = repositorio.findById(id);
        return obj.get();
    }
}
```

```
@RestController
@RequestMapping(value = "/as")
public class RecursoA {

    @Autowired
    private ServicoA servico;

    @GetMapping
    public ResponseEntity<List<A>> obterTodos(){
        List<A> lista = servico.obterTodos();
        return ResponseEntity.ok().body(lista);
    }

    @GetMapping(value =("/{id}")
    public ResponseEntity<A> obterPorId(@PathVariable Long id){
        A a = servico.obterPorId(id);
        return ResponseEntity.ok().body(a);
    }
}
```



# Testando a aplicação

The screenshot displays the Postman interface for testing a REST API. At the top, a browser-like address bar shows `localhost:8080/as/1`. Below it, a JSON response is visible: `{"id":1,"atributo1":"Atributo 11","atributo2":7.77,"atributo3":"2021-06-15T23:33:09Z","atributo4":7}`.

The main section shows a GET request to `http://localhost:8080/as/2`. To the right of the request, the corresponding Java controller method is shown:

```
@GetMapping(value =("/{id}")
public ResponseEntity<A> obterPorId(@PathVariable Long id){
    A a = servico.obterPorId(id);
    return ResponseEntity.ok().body(a);
}
```

Below the request, the 'Query Params' table is empty. The 'Body' tab is selected, showing a JSON body:

```
1 {
2   "id": 2,
3   "atributo1": "Atributo 12",
4   "atributo2": 2.77,
5   "atributo3": "2020-07-20T19:51:09Z",
6   "atributo4": 2
7 }
```

On the right, the 'Test Results' tab shows the status: `Status: 200 OK Time: 42 ms Size: 264 B`. Below this, the Java test class is shown:

```
@Configuration
@Profile("test")
public class SemeadoraDadosTeste implements CommandLineRunner{

    @Autowired
    private RepositorioA repositorioA;

    @Override
    public void run(String... args) throws Exception {
        A a1 = new A(null, "Atributo 11", 7.77, Instant.now(), 7);
        A a2 = new A(null, "Atributo 12", 2.77, Instant.parse("2020-07-20T19:51:09Z"), 2);

        repositorioA.saveAll(Arrays.asList(a1, a2));
    }
}
```

Red arrows indicate the flow of data: one arrow points from the `id` field in the request body to the `id` parameter in the controller method; another arrow points from the `atributo1` field in the request body to the `"Atributo 11"` value in the test data; a third arrow points from the `atributo2` field in the request body to the `2.77` value in the test data.





# Inserindo objetos da classe A

```
@Service
public class ServicoA {

    // Fazendo a Injeção de Dependência
    @Autowired
    private RepositorioA repositorio;

    public List<A> obterTodos(){
        return repositorio.findAll();
    }

    public A obterPorId(Long id) {
        Optional<A> obj = repositorio.findById(id);
        return obj.get();
    }

    public A inserir(A a) {
        return repositorio.save(a);
    }
}
```

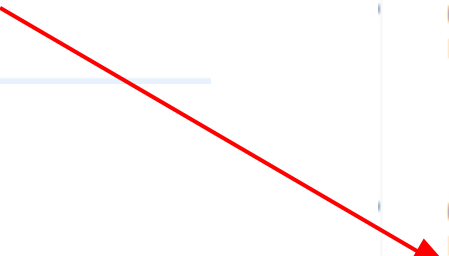
```
@RestController
@RequestMapping(value = "/as")
public class RecursoA {

    @Autowired
    private ServicoA servico;

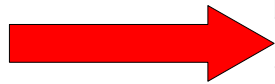
    @GetMapping
    public ResponseEntity<List<A>> obterTodos(){
        List<A> lista = servico.obterTodos();
        return ResponseEntity.ok().body(lista);
    }

    @GetMapping(value =("/{id}")
    public ResponseEntity<A> obterPorId(@PathVariable Long id){
        A a = servico.obterPorId(id);
        return ResponseEntity.ok().body(a);
    }

    @PostMapping
    public ResponseEntity<A> inserir(@RequestBody A a){
        a = servico.inserir(a);
        URI uri = ServletUriComponentsBuilder.fromCurrentRequest().
            path("/{id}").buildAndExpand(a.getId()).toUri();
        return ResponseEntity.created(uri).body(a);
    }
}
```



# Testando a aplicação



GET http://localhost:8080/as

http://localhost:8080/as

GET http://localhost:8080/as

Params Authorization Headers (9) Body Pre-request Script Tests Settings

<input checked="" type="checkbox"/>	Content-Length	<calculated when request is sent>
<input checked="" type="checkbox"/>	Host	<calculated when request is sent>
<input checked="" type="checkbox"/>	User-Agent	PostmanRuntime/7.28.0
<input checked="" type="checkbox"/>	Accept	*/*
<input checked="" type="checkbox"/>	Accept-Encoding	gzip, deflate, br
<input checked="" type="checkbox"/>	Connection	keep-alive
<input checked="" type="checkbox"/>	Content-Type	application/json

Key Value Description

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 1,
4     "atributo1": "Atributo 11",
5     "atributo2": 7.77,
6     "atributo3": "2021-06-16T00:08:04Z",
7     "atributo4": 7
8   },
9   {
10    "id": 2,
11    "atributo1": "Atributo 12",
12    "atributo2": 2.77,
13    "atributo3": "2020-07-20T19:51:09Z",
14    "atributo4": 2
15  }
16 }
```

Status: 200 OK Time: 636 ms Size: 367 B Save Response



# Testando a aplicação

The screenshot shows a REST client interface with a POST request to `http://localhost:8080/as`. The request body is a JSON object with four attributes. The response status is 201 Created. Red arrows highlight the 'Send' button and the response status.

**Request:**

```
POST http://localhost:8080/as
```

**Body (JSON):**

```
{  "atributo1": "Atributo 13",  "atributo2": 3.77,  "atributo3": "2020-03-20T19:51:09Z",  "atributo4": 3}
```

**Response:**

```
{  "id": 3,  "atributo1": "Atributo 13",  "atributo2": 3.77,  "atributo3": "2020-03-20T19:51:09Z",  "atributo4": 3}
```

**Status:** 201 Created

Código HTTP: 201 – Created

Específico para inserções de novos Itens.



# Testando a aplicação

Registro inserido no slide anterior

GET http://localhost:8080/as

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "atributo1": "Atributo 13",
3   "atributo2": 3.77,
4   "atributo3": "2020-03-20T19:51:09Z",
5   "atributo4": 3
6 }
```

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 330 ms Size: 468 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 1,
4     "atributo1": "Atributo 11",
5     "atributo2": 7.77,
6     "atributo3": "2021-06-16T00:13:24Z",
7     "atributo4": 7
8   },
9   {
10    "id": 2,
11    "atributo1": "Atributo 12",
12    "atributo2": 2.77,
13    "atributo3": "2020-07-20T19:51:09Z",
14    "atributo4": 2
15  },
16  {
17    "id": 3,
18    "atributo1": "Atributo 13",
19    "atributo2": 3.77,
20    "atributo3": "2020-03-20T19:51:09Z",
21    "atributo4": 3
22  }
23 }
```



# Excluindo objetos da classe A

```
@Service
public class ServicoA {

    // Fazendo a Injeção de Dependência
    @Autowired
    private RepositorioA repositorio;

    public List<A> obterTodos(){
    }

    public A obterPorId(Long id) {
    }

    public A inserir(A a) {
    }

    public void excluir(Long id) {
        repositorio.deleteById(id);
    }
}
```

```
@RestController
@RequestMapping(value = "/as")
public class RecursoA {

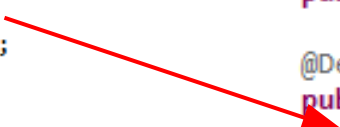
    private ServicoA servico;

    public ResponseEntity<List<A>> obterTodos(){
    }

    public ResponseEntity<A> obterPorId(@PathVariable Long id){
    }

    public ResponseEntity<A> inserir(@RequestBody A a){
    }

    @DeleteMapping(value =("/{id}")
    public ResponseEntity<Void> excluir(@PathVariable Long id){
        servico.excluir(id);
        return ResponseEntity.noContent().build();
    }
}
```



# Testando a aplicação

GET localhost:8080/as

localhost:8080/as

GET localhost:8080/as

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 698 ms Size: 367 B

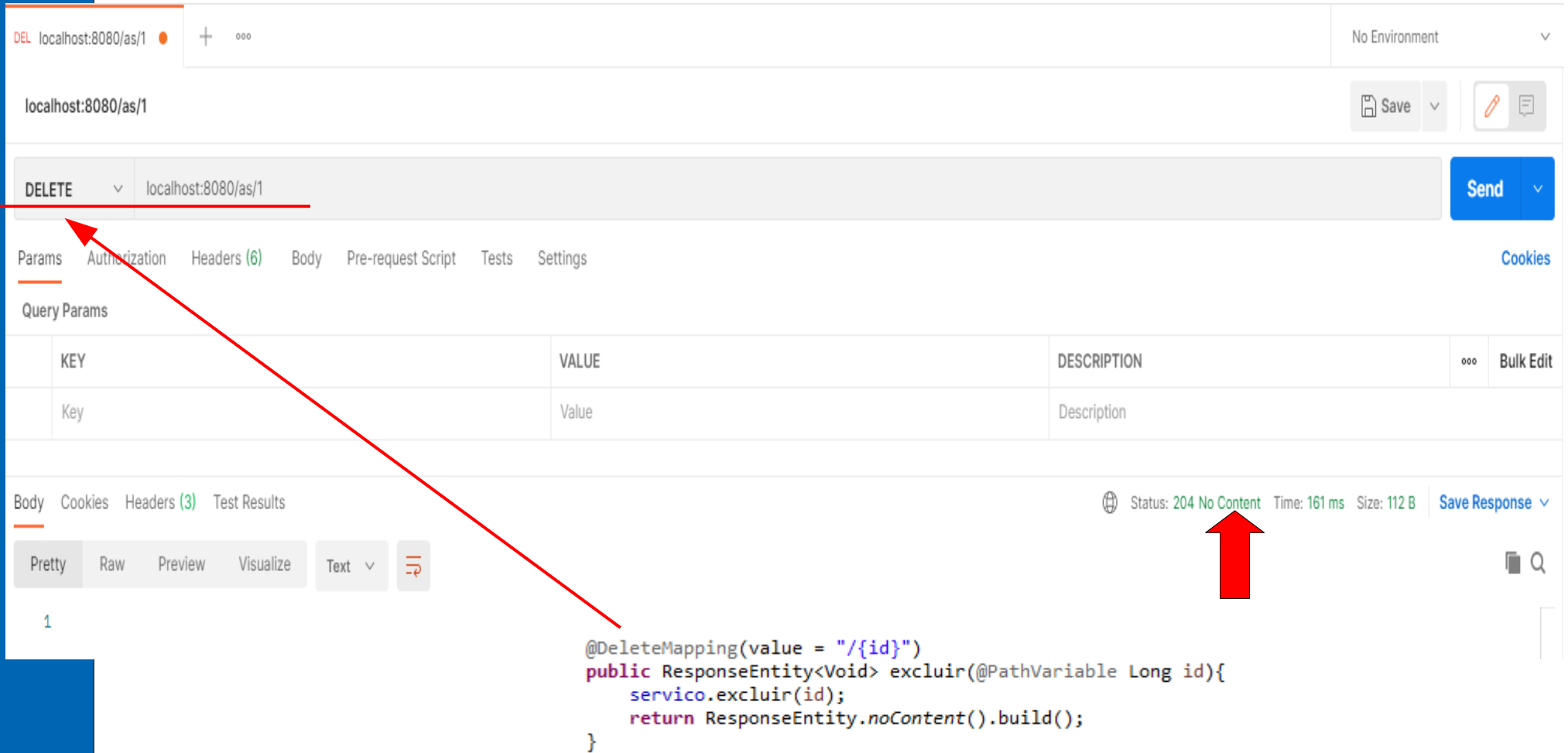
Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 1,
4     "atributo1": "Atributo 11",
5     "atributo2": 7.77,
6     "atributo3": "2021-06-17T12:03:10Z",
7     "atributo4": 7
8   },
9   {
10    "id": 2,
11    "atributo1": "Atributo 12",
12    "atributo2": 2.77,
13    "atributo3": "2020-07-20T19:51:09Z",
14    "atributo4": 2
15  }
16 }
```

Vamos excluir o registro de id = 1



# Testando a aplicação



DEL localhost:8080/as/1

localhost:8080/as/1

DELETE localhost:8080/as/1

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize Text

Status: 204 No Content Time: 161 ms Size: 112 B

```
@DeleteMapping(value =("/{id}")  
public ResponseEntity<Void> excluir(@PathVariable Long id){  
    servico.excluir(id);  
    return ResponseEntity.noContent().build();  
}
```



# Testando a aplicação

GET localhost:8080/as/ + ... No Environment ▼

localhost:8080/as/ Save ▼ ✎ 💬

GET ▼ localhost:8080/as/ Send ▼

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 23 ms Size: 266 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ ↺

```
1 {  
2     
3   "id": 2,  
4   "atributo1": "Atributo 12",  
5   "atributo2": 2.77,  
6   "atributo3": "2020-07-20T19:51:09Z",  
7   "atributo4": 2  
8 }  
9
```

Registro de id = 1 excluído





# Atualizando objetos da classe A

```
@Service
public class ServicoA {

    // Fazendo a Injeção de Dependência
    @Autowired
    private RepositorioA repositorio;

    public List<A> obterTodos(){

    }

    public A obterPorId(Long id) {

    }

    public A inserir(A a) {

    }

    public void excluir(Long id) {

    }

    public A update(Long id, A objeto_alterado) {
        A a = repositorio.getById(id);
        atualizarDados(a, objeto_alterado);
        return repositorio.save(a);
    }

    public void atualizarDados(A destino, A origem) {
        destino.setAtributo1(origem.getAtributo1());
        destino.setAtributo2(origem.getAtributo2());
        destino.setAtributo3(origem.getAtributo3());
        destino.setAtributo4(origem.getAtributo4());
    }
}
```

```
@RestController
@RequestMapping(value = "/as")
public class RecursoA {

    private ServicoA servico;

    public ResponseEntity<List<A>> obterTodos(){

    }

    public ResponseEntity<A> obterPorId(@PathVariable Long id){

    }

    public ResponseEntity<A> inserir(@RequestBody A a){

    }

    public ResponseEntity<Void> excluir(@PathVariable Long id){

    }

    @PutMapping(value =("/{id}")
    public ResponseEntity<A> update(@PathVariable Long id, @RequestBody A obj){
        obj = servico.update(id, obj);
        return ResponseEntity.ok().body(obj);
    }
}
```



# Testando a aplicação

The screenshot shows a REST client interface with a GET request to `localhost:8080/as/`. The response is a JSON array of two objects. The second object is highlighted with a red box and labeled "Registro que será alterado" (Record that will be changed).

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body

Status: 200 OK Time: 592 ms Size: 367 B Save Response

JSON

```
1 {
2   {
3     "id": 1,
4     "atributo1": "Atributo 11",
5     "atributo2": 7.77,
6     "atributo3": "2021-06-17T12:39:12Z",
7     "atributo4": 7
8   },
9   {
10    "id": 2,
11    "atributo1": "Atributo 12",
12    "atributo2": 2.77,
13    "atributo3": "2020-07-20T19:51:09Z",
14    "atributo4": 2
15  }
16 }
```



# Testando a aplicação

```
@PutMapping(value =("/{id}")  
public ResponseEntity<A> update(@PathVariable Long id, @RequestBody A obj){  
    obj = servico.update(id, obj);  
    return ResponseEntity.ok().body(obj);  
}
```

The screenshot shows a REST client interface with the following details:

- Method:** PUT (indicated by a red arrow from the title)
- URL:** localhost:8080/as/2
- Body:** A JSON object with the following structure:

```
{  
  "id": 2,  
  "atributo1": "Atributo 8",  
  "atributo2": 8.77,  
  "atributo3": "2028-07-20T19:51:09Z",  
  "atributo4": 8  
}
```
- Headers:** 8 headers are listed.
- Pre-request Script:** Empty.
- Tests:** Empty.
- Settings:** Empty.
- Status:** 200 OK, Time: 177 ms, Size: 263 B.
- Response Body:** A JSON object with the following structure:

```
{  
  "id": 2,  
  "atributo1": "Atributo 8",  
  "atributo2": 8.77,  
  "atributo3": "2028-07-20T19:51:09Z",  
  "atributo4": 8  
}
```



# Testando a aplicação

GET localhost:8080/as/ PUT localhost:8080/as/2 + ... No Environment

localhost:8080/as/ Save Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 19 ms Size: 366 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 1,
4     "atributo1": "Atributo 11",
5     "atributo2": 7.77,
6     "atributo3": "2021-06-17T12:39:12Z",
7     "atributo4": 7
8   },
9   {
10    "id": 2,
11    "atributo1": "Atributo 8",
12    "atributo2": 8.77,
13    "atributo3": "2028-07-20T19:51:09Z",
14    "atributo4": 8
15  }
16 }
```

Registro alterado



# Dúvidas?

