DBC
DIGITAL BUSINESS COMPANY®

VEM SER
DBC

Spring Web

Aula 5 – Lombok e ObjectMapper

Sumário

- Lombok
- Logs
- Data Transfer Object (DTO)
- Object Mapper

Lombok

- O **Lombok** é um framework para Java que permite escrever código eliminando a verbosidade, o que permite ganhar tempo de desenvolvimento para o que realmente é importante. Seu uso permite gerar em tempo de compilação os métodos **getters e setters**, métodos **construtores**, padrão **builder e muito mais.**

- https://projectlombok.org/features/all

```
<dependency>
     <groupId>org.projectlombok</groupId>
     <artifactId>lombok</artifactId>
     <version>1.18.22</version>
     <scope>provided</scope>
</dependency>
```

Lombok

- Principais anotações:

- **@AllArgsConstructor**: Construtor com todos os argumentos
- **@NoArgsConstructor**: Construtor sem argumentos
- **@RequiredArgsConstructor**: Construtor com os campos final
- **@ToString**: Gera um método ToString com todos os campos
- **@Getter**: Gera o método getter
- **@Setter**: Gera o método setter
- **@Builder**: padrão de projetos Builder, para quando seu objeto é imutável
- **@Data**: junção de todas elas (Get, Set, toString, eqhash, requiredArgs)

Logs

Example:

```
@Slf4j
public class LogExample {
}
```

will generate:

```
public class LogExample {
    private static final org.slf4j.Logger log = org.slf4j.LoggerFactory.getLogger(LogExample.class);
}
```

```
@PostMapping
public ResponseEntity<PessoaDTO> create(@RequestBody @Valid PessoaD
    log.info("criando pessoa");
    PessoaDTO pessoaCriado = pessoaService.create(pessoa);
    return new ResponseEntity<>(pessoaCriado, HttpStatus.CREATED);
}
```

Vamos praticar!

Exercício #1

- Alterar as classes Pessoa, Contato e Endereço para utilizar Lombok.
- Remover Getters/Setters.
- Remover Construtores.
- Colocar Logs nos Controllers e em todos os métodos PUT, POST e DELETE.
- Testar app.

DTOs

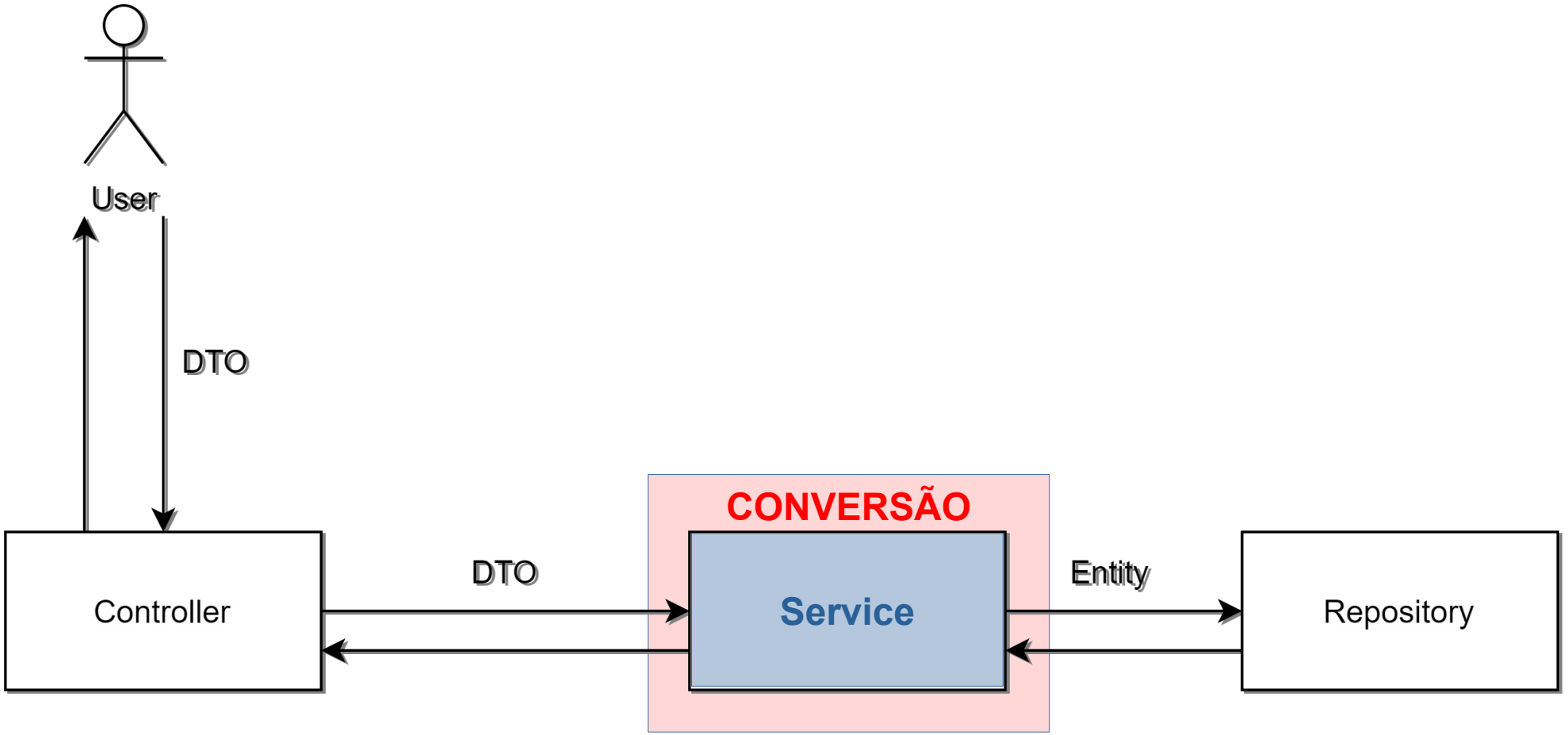
- **DTO**: Data Transfer Object.
- Podem ser chamados de VO (virtual object).

- São classes/objetos **intermediários** que criamos para a recepção e o envio dos dados na **entrada/saída API.**

- Geralmente **não mandamos/recebemos** as classes de **Entidades** nos serviços (controllers).

- **Sempre** enviamos e recebemos um **DTO** (mesmo que seja com os mesmos dados).

DTOs


User
DTO
CONVERSÃO
DTO
Entity
Controller
Service
Repository

Vamos praticar!

ObjectMapper

- A classe **ObjectMapper (com.fasterxml.jackson.databind.ObjectMapper)** é a maneira mais simples de analisar **JSON**.
- O ObjectMapper pode analisar JSON de uma string, stream ou arquivo e criar um objeto Java que representa o JSON analisado.
- Classe responsável por **converter** um objeto em outro.

```
@Autowired
private ObjectMapper objectMapper;
```

```
public PessoaDTO create(PessoaCreateDTO pessoa) throws Exception {
    Pessoa pessoaEntity = objectMapper.convertValue(pessoa, Pessoa.class);
```

Vamos praticar!

Exercício #2

- Criar pacote dto e criar **DTOs** para Pessoa (**PessoaDTO** e **PessoaCreateDTO** (para entradas)).
- Remover todas as **entitys** da **PessoaController** e usar somente **DTO** (conforme padrão).
- Utilizando a biblioteca **ObjectMapper** para **converter** um objeto para o outro na **Service.**

Task

- Criar pacote **dto** e criar **DTOs** para **Enderecos** e **Contatos.**
- Remover todas as **entitys** das **Controllers** e usar somente **DTO** (conforme padrão).
- Utilizando a biblioteca **ObjectMapper** para **converter** um objeto para o outro no **Service.**

- **Revisitar os exercícios** e atualizar ao menos o cadastro de **Pessoa** com os itens vistos como:
  - PropertiesReader;
  - Pacote de exception (Handlers);
  - RegraDeNegocioException nos throws;
  - ResponseEntity;

Task #2 Grupo (opcional)

- Utilizar o conceito de **DTO** e **não** expor nenhuma **Entity** no Controller;
- Utilizar **Lombok** e **ObjectMapper**;

Obrigado!

DBC

DIGITAL BUSINESS COMPANY®

/dbc.company /dbccompany /dbccompany.com.br /company/dbc-company