Validação de objetos do Body – Bean Validation

Objetos complexos são passados pelo body de uma requisição, mas isso não siginifica que eles não podem ou não deveriam ser validados, afinal de contas esses objetos se tornam entidades.

Pensando nisso existe o projeto de **Bean Validation**, simples de se usar como todo projeto do ecossistema Spring:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

A ideia mais simples de uso do projeto é **anotar os POJOS** que vão ser validados com suas respectivas condições, em 90% dos casos as anotações existentes servem seu propósito:

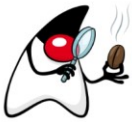
Uma vez que os POJOS estejam anotados é preciso **dizer para os métodos que os recebem como parâmetro** para que **ocorra a validação:**

```
@PostMapping
public ResponseEntity<EntityModel<PedidoDTOResponse>> criarPedido(@RequestBody @Valid PedidoDTOPost pedidoPostDTO){...}
```

Validações customizadas – Bean Validation

Por mais que as validações existentes cubram a maioria dos casos de uso, você pode criar anotações próprias de validação para serem usadas em casos específicos. Uma anotação que segue a

especificação é mais ou menos assim:



Anotações para criar uma anotação:
-> Onde ela vai ser usada
-> Vai ser "avaliada" em tempo de execução
-> A classe que contém a lógica da validação

```java
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
@Constraint(validatedBy = ClasseConcretaDaValidacao.class )
public @interface AnotacaoCustomizada {

    String message() default "Alguma mensagem default";

    Class<?> [] groups () default {};
    public abstract Class<? extends Payload> [] payload() default {};
}
```

Se quiser que sua anotação seja validada pelo Bean validation tem que seguir detalhes da implementação proposta

A classe concreta da validação pode conter qualquer lógica, desde que retorne um valor booleano:

Extente a interface de constraintValidator, especifica a anotação e o tipo em que sua lógica vai agir.

```java
public class ClasseConcretaDaValidacao implements ConstraintValidator<AnotacaoCustomizada, QualquerTipoParaValidar> {

    @Override
    public boolean isValid(QualquerTipoParaValidar value, ConstraintValidatorContext context) {
        boolean maiorQue0 = value.podeSerInteiro > 0;
        boolean isNull = value.podeSerOutroObjeto != null;
        boolean existeEssaFrase = value.podeSerString.contains("PoderiaSerUmaRegex");

        return maiorQue0 && isNull && existeEssaFrase;
    }

}
```

Agora é só usar onde você quiser:

```java
public class PedidoDTOPost {

    @NotNull @NotEmpty
    List<@Valid ItemPedidoDTO> itens;

    @AnotacaoCustomizada(message = "Posso sobrescrever a mensagem")
    private QualquerTipoParaValidar qualquerCoisa;
```