

## ARTIGO: 11367

# Usando a API de Validações

## Introdução

Nesse artigo iremos ver como usar a API de validações fornecida pelo Veloster Framework

## Anotações

A API de validações do Veloster contém um conjunto de anotações que são processadas antes de uma entidade ser inseridas ou alteradas no banco de dados. São elas:

Annotation	Description
DateValidation	Validação para data.
Length	Validação para tamanho de String
NotNull	Validação para valor nulo
NumberValidation	Validação para números
Regex	Validação com aplicação de expressão regular

## Mensagens de validação

Após uma validação ser processada e algum valor ser identificado como inválido, uma exceção do tipo:

Java Code

```
br.com.mobilemind.veloster.exceptions.EntityValidatorException
```

é lançada. Em sua mensagem, será apresentado um valor recuperado no arquivo *message\_validates.properties*, que deve estar presente na pasta de recursos da aplicação. Os mensagens padrão são:

Java Code

```
br.com.mobilemind.date=date range should be between {0} and {1} for field {2}
br.com.mobilemind.notnull=value cannot be null for field {0}
br.com.mobilemind.length=length range should be between {0} and {1} for field {2}
br.com.mobilemind.number=number range should be between {0} and {1} for field {2}
br.com.mobilemind.regex=invalid value. required format {0} for field {1}
```

Existe uma mensagem para cada tipo de validação. Além disso, uma mensagem personalizada pode ser configurada para o atributo *message* da anotação, ou ainda pode ser configurada uma nova chave de validação, no atributo *messageKey* da anotação, que deverá ser inserida no arquivo *message\_validates.properties*.

Java Code

```
@Length(min=5, max=200, message="Texto deve ter entre 2 a 200 caracteres")
```

Ou

Java Code

```
//default key is br.com.mobilemind.length
@Length(min=5, max=200, messageKey="br.com.mobilemind.custonLength")

//in message_validates.properties
br.com.mobilemind.custonLength=Texto deve ter entre 2 a 200 caracteres
```

## Processamento de uma validação

As validações são processadas através de uma implementação da interface:

Java Code

```
interface br.com.mobilemind.veloster.orm.EntityValidator<T>
```

A implementação padrão disponível para essa interface é:

Java Code

```
br.com.mobilemind.veloster.orm.core.EntityValidatorImpl<T extends Entity>
```

Para executar o validador em uma entidade, devemos executar:

Java Code

```
EntityValidator<Person> validator = VelosterRepository.getORMControll(Person.class).getEntityValidator();
Person person = new Person();
try {
```

```
    validator.validate(person);  
} catch (EntityValidatorException e) {  
    message = e.getMessage();  
    say(message);  
}
```