



# VALIDAÇÕES NO SPRING

FAPESC – DESENVOLVEDORES PARA TECNOLOGIA DA INFORMAÇÃO

HERCULANO DE BIASI

[herculano.debiasi@unoesc.edu.br](mailto:herculano.debiasi@unoesc.edu.br)



# TÓPICOS

- Tipos de validações
- Especificação Jakarta Bean *Validation*
- Hibernate *validator*
- Onde validar?
- Configuração no Spring Boot
- Principais anotações e atributos
- Spring Validator

# TIPOS DE VALIDAÇÕES

## ■ Sem acesso a dados

### ■ Sintáticas

- Campo não pode estar vazio
- Valor numérico dentro de uma determinada faixa
- Comprimento de uma *string* dentro de uma determinada faixa
- Somente dígitos
- Conformidade com alguma expressão regular, por exemplo, ((##) # ##### - #####

### ■ Outras

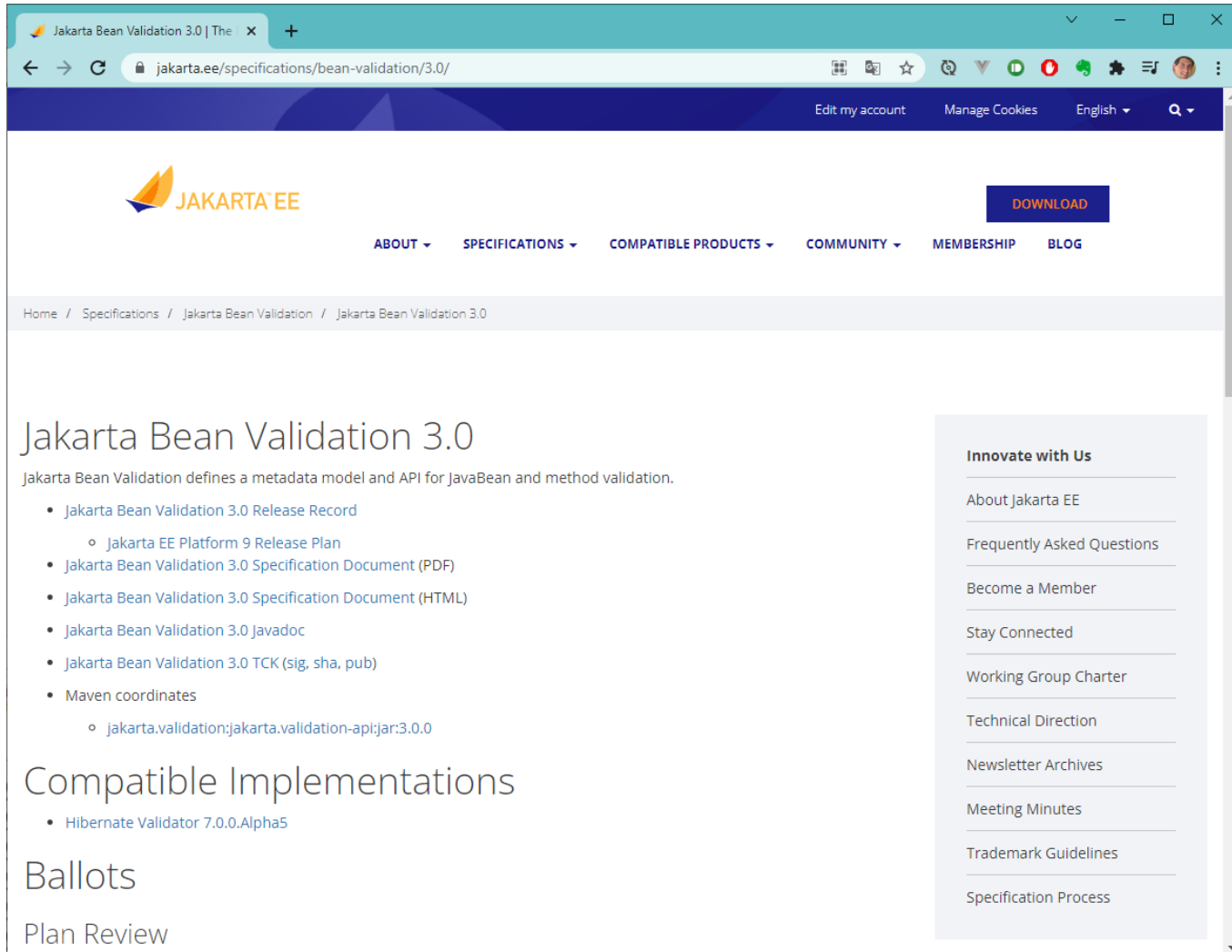
- Data futura / passada
- Confirmação de senha igual à senha

## ■ Com acesso a dados

- Somente clientes novos podem utilizar determinado cupom de promoção
- E-mail não pode ser repetido

# ESPECIFICAÇÃO JAKARTA BEAN VALIDATION


- Jakarta Bean Validation é uma especificação (JSR-380) para um modelo e uma API para validação de métodos e JavaBeans



The screenshot shows a web browser displaying the Jakarta Bean Validation 3.0 specification page. The browser's address bar shows the URL `jakarta.ee/specifications/bean-validation/3.0/`. The page features the Jakarta EE logo and a navigation menu with links for ABOUT, SPECIFICATIONS, COMPATIBLE PRODUCTS, COMMUNITY, MEMBERSHIP, and BLOG. A 'DOWNLOAD' button is also visible. The main content area is titled 'Jakarta Bean Validation 3.0' and includes a description: 'Jakarta Bean Validation defines a metadata model and API for JavaBean and method validation.' Below this, there is a list of links for the release record, specification documents (PDF and HTML), javadoc, TCK, and Maven coordinates. A section for 'Compatible Implementations' lists 'Hibernate Validator 7.0.0.Alpha5'. The page also includes a sidebar with a list of links under the heading 'Innovate with Us'.

Jakarta Bean Validation 3.0 | The Jakarta EE Project

[Edit my account](#) [Manage Cookies](#) [English](#) [Search](#)

 [DOWNLOAD](#)

[ABOUT](#) [SPECIFICATIONS](#) [COMPATIBLE PRODUCTS](#) [COMMUNITY](#) [MEMBERSHIP](#) [BLOG](#)

Home / Specifications / Jakarta Bean Validation / Jakarta Bean Validation 3.0

## Jakarta Bean Validation 3.0

Jakarta Bean Validation defines a metadata model and API for JavaBean and method validation.

- [Jakarta Bean Validation 3.0 Release Record](#)
  - [Jakarta EE Platform 9 Release Plan](#)
- [Jakarta Bean Validation 3.0 Specification Document \(PDF\)](#)
- [Jakarta Bean Validation 3.0 Specification Document \(HTML\)](#)
- [Jakarta Bean Validation 3.0 Javadoc](#)
- [Jakarta Bean Validation 3.0 TCK \(sig, sha, pub\)](#)
- [Maven coordinates](#)
  - `jakarta.validation:jakarta.validation-api:jar:3.0.0`

## Compatible Implementations

- [Hibernate Validator 7.0.0.Alpha5](#)

## Ballots

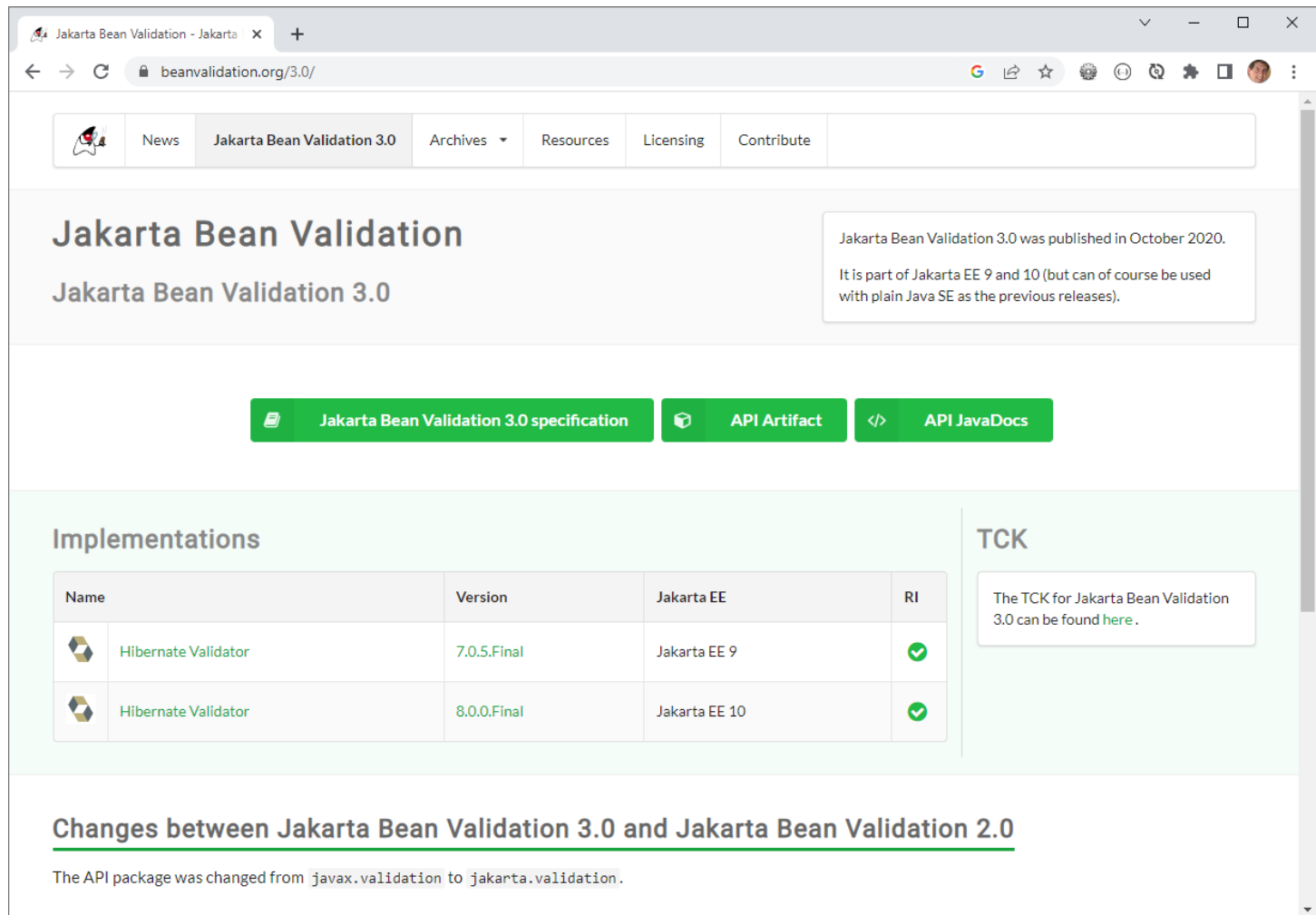
## Plan Review

### Innovate with Us



- [About Jakarta EE](#)
- [Frequently Asked Questions](#)
- [Become a Member](#)
- [Stay Connected](#)
- [Working Group Charter](#)
- [Technical Direction](#)
- [Newsletter Archives](#)
- [Meeting Minutes](#)
- [Trademark Guidelines](#)
- [Specification Process](#)

# HIBERNATE *Validator*

- O [Hibernate Validator](#), iniciando na versão 7.0, é uma implementação da especificação Jakarta Bean Validation 3.0



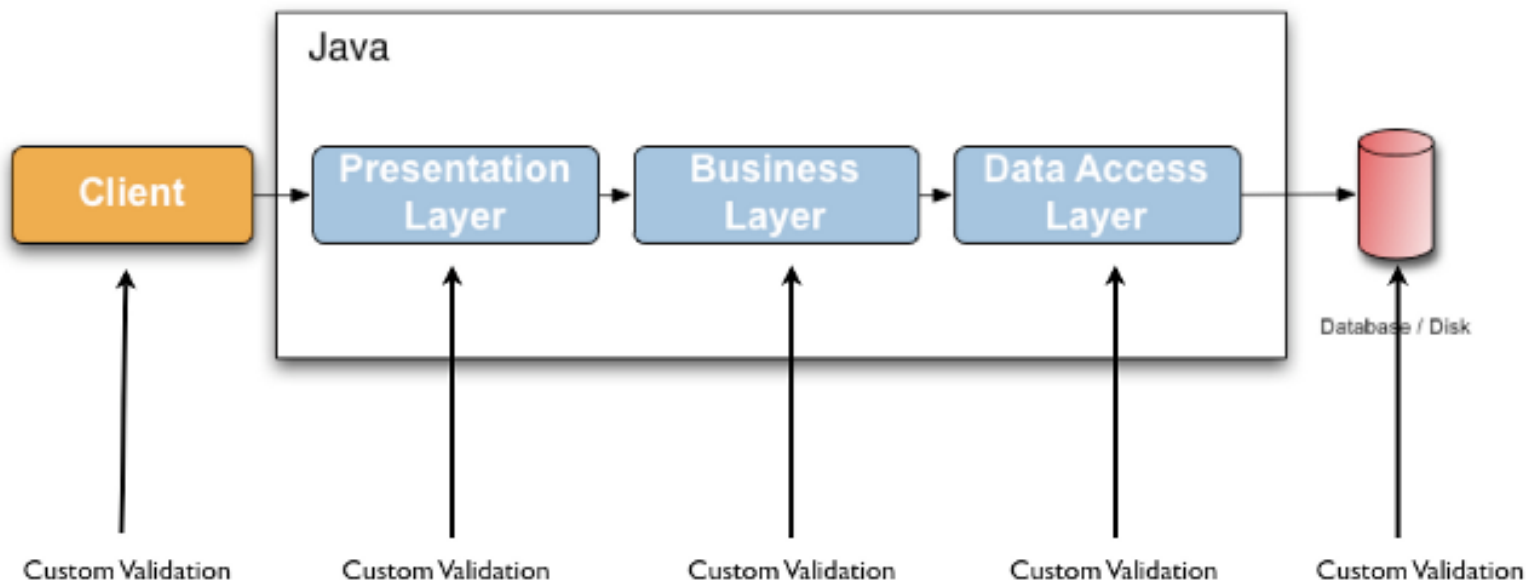
The screenshot shows the official website for Jakarta Bean Validation 3.0. The page has a navigation bar with links for News, Jakarta Bean Validation 3.0, Archives, Resources, Licensing, and Contribute. The main heading is "Jakarta Bean Validation" with a sub-heading "Jakarta Bean Validation 3.0". A text box states that the specification was published in October 2020 and is part of Jakarta EE 9 and 10. Below this are three green buttons: "Jakarta Bean Validation 3.0 specification", "API Artifact", and "API JavaDocs". The "Implementations" section features a table with two entries for Hibernate Validator, showing versions 7.0.5.Final and 8.0.0.Final, both compatible with Jakarta EE 9 and 10 respectively. A "TCK" section mentions that the TCK for Jakarta Bean Validation 3.0 can be found [here](#). At the bottom, a section titled "Changes between Jakarta Bean Validation 3.0 and Jakarta Bean Validation 2.0" notes that the API package was changed from `javax.validation` to `jakarta.validation`.

Name	Version	Jakarta EE	RI
 <a href="#">Hibernate Validator</a>	7.0.5.Final	Jakarta EE 9	✓
 <a href="#">Hibernate Validator</a>	8.0.0.Final	Jakarta EE 10	✓

# ONDE VALIDAR?

## ■ Onde validar?

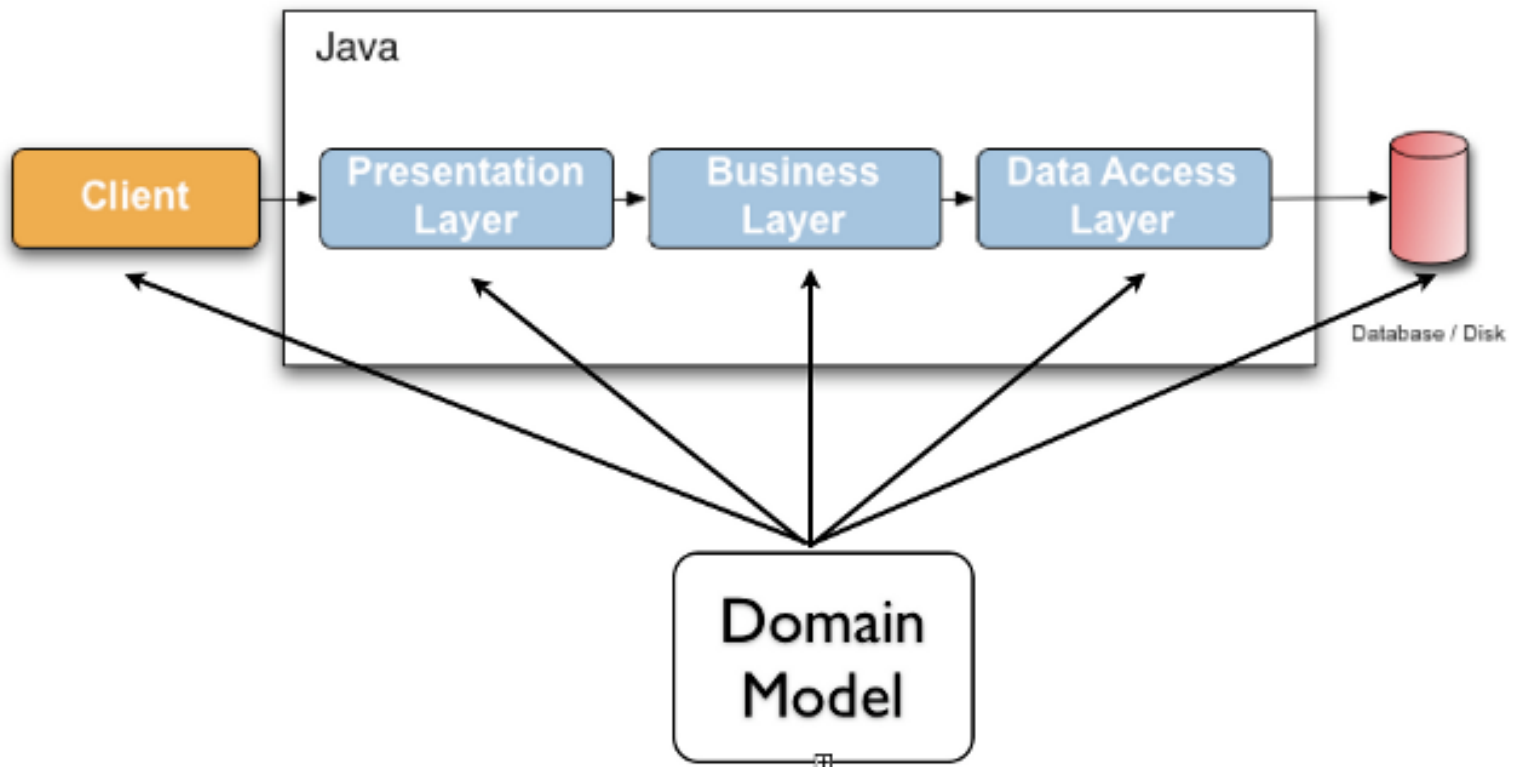
- Frequentemente as tarefas de validação são implementadas em cada camada do sistema, o que consome tempo e é suscetível a erros



# ONDE VALIDAR?

## ■ Onde validar?

- Alternativa: Para evitar duplicação dessas validações, desenvolvedores frequentemente agrupam a lógica de validação diretamente no modelo de domínio
- As classes de domínio são anotadas com código de validação que na verdade são metadados sobre a própria classe



# CONFIGURAÇÃO NO SPRING BOOT

■ No Spring Boot este recurso pode ser incluído através de um *starter*

**New Spring Starter Project Dependencies**

Service URL:

Spring Boot Version:

Frequently Used:

<input type="checkbox"/> Flyway Migration	<input type="checkbox"/> H2 Database	<input type="checkbox"/> Lombok
<input type="checkbox"/> PostgreSQL Driver	<input type="checkbox"/> Spring Boot DevTools	<input type="checkbox"/> Spring Data JPA
<input type="checkbox"/> Spring Web	<input type="checkbox"/> Thymeleaf	<input checked="" type="checkbox"/> Validation

Available:

- Developer Tools
- Google Cloud Platform
- I/O
- Messaging
- Microsoft Azure
- NoSQL
- Observability
- Ops
- SQL
- Security
- Spring Cloud
- Spring Cloud Circuit Breaker
- Spring Cloud Config
- Spring Cloud Discovery
- Spring Cloud Messaging

Selected: X Validation

Make Default Clear Selection

< Back Next > Finish Cancel

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



# PRINCIPAIS ANOTAÇÕES E ATRIBUTOS

## ■ Principais anotações de validação

- @NotNull: Campo não pode ser nulo
- @NotEmpty: Um campo de lista não pode estar vazio (mas pode ter somente espaços em branco)
- @NotBlank: Um campo *string* não pode estar vazio
- @Min e @Max : Um campo numérico é válido somente se seu valor estiver dentro de uma determinada faixa de valores
- @Pattern: Um campo *string* somente é válido se estiver de acordo com uma determinada expressão regular
- @Email: O campo só é válido se contiver um endereço de *e-mail* válido
- @PastOrPresent: Data passada ou atual
- @Positive / @PositiveOrZero: Somente valores positivos / 0 ou positivos

## ■ Atributos

- min e max: Tamanhos mínimo e máximo
- message: Mensagem de exibição

# SPRING VALIDATOR

- O Spring Validator é um recurso nativo do Spring e não tem vínculo com a especificação Bean Validation
    - Para fazer uso desse recurso é necessário criar uma classe que implemente a *interface* `Validator` do pacote `org.springframework.validation`
    - A interface exige que sejam implementados dois métodos: `supports()` e `validate()`
  - `validate()` tem como objetivo verificar que o objeto enviado pelo formulário é do mesmo tipo esperado pela classe de validação
    - Para isso, basta incluir um teste simples de verificação como corpo do método
- ```
@Override
public boolean supports(Class<?> clazz) {
    return Funcionario.class.equals(clazz);
}
```

# SPRING VALIDATOR

- O método `validate()` é onde a lógica de validação dos campos serão realizadas – para isso ele utiliza dois argumentos
- O primeiro argumento é do tipo `Object` que contém a variável enviada pelo formulário
- O outro é do tipo `Errors`, e é usado para lançar as mensagens de validação nos respectivos campos

## ■ Exemplo

```
@Override  
public void validate(Object object, Errors errors) {  
    Funcionario f = (Funcionario) object;  
  
    if (f.getNome() == null) {  
        errors.rejectValue("nome", "O nome é obrigatório");  
    }  
}
```

# SPRING VALIDATOR

## ■ Funcionamento

- No método `validate()` do *slide* anterior, a variável `Funcionario` é criada partir da variável `object` por meio de um *cast*
- Agora é possível criar qualquer lógica de validação sobre os campos de `Funcionario`
- No exemplo apresentado há um teste para verificar se o campo nome enviado pelo formulário está nulo
- Caso esteja, usa-se a variável `errors` para criar a mensagem de validação e avisar ao *controller* que um erro ocorreu
- Isso é feito pelo método `rejectValue()` o qual como primeiro parâmetro recebe o nome do campo validado e como segundo parâmetro a mensagem de validação
- Desta forma pode-se criar qualquer tipo de teste ou lógica para validar qualquer campo do formulário

# SPRING VALIDATOR

## ■ Funcionamento

- Para finalizar o processo com o Spring Validator é preciso dizer ao *controller* que ele deve usar a classe que implementou a *interface* `Validator` como validador do formulário
- Para isso, deve-se inserir no *controller* o método `initBinder`
- Esta anotação vai instruir o Spring a executar este método como o primeiro da classe
- Assim, a classe `FuncionarioValidator` vai ser executada antes da requisição chegar ao método `salvar()`
- Quando então a requisição chegar ao `salvar()` o Spring Validator já terá validado o formulário e em caso de algum campo não ter passado na validação o teste condicional do `hasErrors()` retornará `true`

```
@InitBinder  
  
public void initBinder(WebDataBinder binder) {  
    binder.addValidators(new FuncionarioValidator());  
}
```