01

# Usando nosso CRUD genérico para salvar

### Transcrição

Na aplicação que estamos migrando, a mesma página que listava os convidados também tinha um formulário para convidar mais pessoas para a lista. O formulário era semelhante ao que se segue:

```
<hr>
<form action="convidado">
    <div class="form-group">
        <label for="nome">Nome</label>
        <input type="text" class="form-control" id="nome" name="nome" placeholder="Nome">
    </div>
    <div class="form-group">
        <label for="email">Email</label>
        <input type="email" class="form-control" id="email" name="email" placeholder="Email">
    </div>
    <div class="form-group">
        <label for="telefone">Telefone</label>
        <input type="text" class="form-control" id="telefone" name="telefone" placeholder="Tele
    </div>
    <button type="submit" class="btn btn-success">Convidar</button>
</form>

/...
```

Vamos copiá-lo para a nova aplicação, com algumas pequenas mudanças. Lembre-se que o *Thymeleaf* é bem detalhista, por isso, fecharemos todas as tags corretamente. A segunda alteração é fazer com que o formulário submeta as informações via *POST* e a *action* será `salvar` ao invés de `convidado`. Após o fechamento da tag `<hr>`, o código ficará assim com as refatorações.

```
<form action="salvar" method="post">
    <div class="form-group">
        <label for="nome">Nome</label>
        <input type="text" class="form-control" id="nome" name="nome" placeholder="Nome" />
    </div>
    <div class="form-group">
        <label for="email">Email</label>
        <input type="email" class="form-control" id="email" name="email" placeholder="Email" />
    </div>
    <div class="form-group">
        <label for="telefone">Telefone</label>
        <input type="text" class="form-control" id="telefone" name="telefone" placeholder="Tele
    </div>
    <button type="submit" class="btn btn-success">Convidar</button>
</form>
```

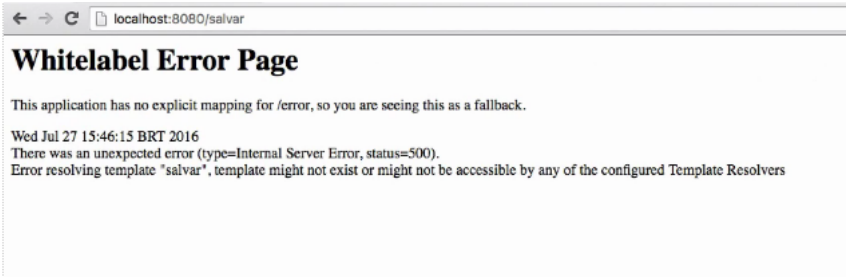
O *controller* é quem recebe as informações enviadas pela *view*. Criaremos então o método `salvar` que recebe estas informações via POST da seguinte forma:

```
@RequestMapping(value= "salvar", method = RequestMethod.POST)
public void salvar(){

}
```

E já podemos testar. O formulário será exibido normalmente, mas ao preencher e submeter o formulário, teremos um erro. Vejamos:


E o erro será:




Apesar da *URL* estar apontando para o endereço correto estamos tendo erros isto por que não estaremos retornando nenhuma resposta ou template. Também não estamos fazendo nenhuma tarefa com as informações.

Primeiro vamos receber todos os dados vindos do formulário, criar um objeto convidado, salva-lo com o objeto `repository` e retornar o *template* de listagem dos convidados.

```
@RequestMapping(value= "salvar", method = RequestMethod.POST)
public String salvar(@RequestParam("nome") String nome, @RequestParam("email") String email,
                     @RequestParam("telefone") String telefone ){

    Convidado novoConvidado = new Convidado(nome, email, telefone);
    repository.save(novoConvidado);

    return "listaconvidados";
}
```

Note que o retorno do método mudou de *void* para *String* para podermos retornar o *template* da página.

O primeiro problema deste código é: Não temos um construtor na classe `Convidado` e por isso teremos erros. Para evita-los, criaremos dois construtores, um vazio, padrão e outro aceitando estes valores. Na classe `Convidado` adicionaremos:

```
public Convidado(){}

public Convidado(String nome, String email, String telefone){
```

```
        this.nome = nome;
        this.email = email;
        this.telefone = telefone;
    }
```

Será que já temos tudo resolvido? Ao testar, um comportamento estranho pode ser identificado. Após salvar, mesmo o convidado tendo sido salvo, não vemos mais a lista de convidados e a URL aponta para a página de salvar.


Isto ocorre porque o método que salva o convidado, não recarrega os convidados para serem listados novamente apesar de usar o mesmo *template*. Vamos solucionar isso fazendo com que o método `salvar` receba o `model` e carregue os convidados para a página novamente. Alteraremos o método salvar para ficar com o seguinte código:

```
@RequestMapping(value= "salvar", method = RequestMethod.POST)
public String salvar(@RequestParam("nome") String nome, @RequestParam("email") String email,
                     @RequestParam("telefone") String telefone, Model model ){

    Convidado novoConvidado = new Convidado(nome, email, telefone);
    repository.save(novoConvidado);

    Iterable<Convidado> convidados = repository.findAll();
    model.addAttribute("convidados", convidados);

    return "listaconvidados";
}
```

Agora ao salvar um convidado, a lista será recarregada mostrando o novo convidado na lista.