05

Criando um template de email simples

Agora que sabemos enviar um email, precisamos criar um template com as informações que serão enviadas, criando uma classe que estende o **Email** do *Play!*. Precisamos receber um token de cadastro no construtor e definir parte do conteúdo que será inserido no email.

```
package models;
public class EmailDeCadastro extends Email {
    private static final String REMETENTE = "Caelum <no-reply@caelum.com.br>";
    private static final String ASSUNTO = "Confirmação de cadastro na API de produtos!";
    private static final String CORPO_FORMAT = "Olá, %s! Por favor clique no link a seguir para
    public EmailDeCadastro(TokenDeCadastro token) { }
}
```

Como isso é um template, o conteúdo precisa ser imutável, constante. Repare que a gente cria o formato do corpo do email, mas ainda não passa o nome do usuário nem o endereço do link de confirmação. Faremos isso no construtor, já que é um conteúdo dinâmico.

```
public EmailDeCadastro(TokenDeCadastro token) {
    Usuario usuario = token.getUsuario();
    String destinatario = String.format("%s <%s>", usuario.getNome(), usuario.getEmail());
    String link = String.format("http://localhost:9000/usuario/confirma/%s/%s", usuario.getEmai
    String corpo = String.format(CORPO_FORMAT, usuario.getNome(), link);

    this.addTo(destinatario);
    this.setFrom(REMETENTE);
    this.setSubject(ASSUNTO);
    this.setBodyHtml(corpo);
}
```

Primeiro pegamos o usuário do token e usamos ele pra criar o destinatário, formatado de acordo com os padrões de email. Em seguida, geramos o link de cadastro, na rota `/usuario/confirma/email/codigo`, trocando **email** pelo email do usuário e **codigo** pelo código de segurança do token de cadastro. Depois, geramos o corpo usando a constante e inserindo o nome do usuário e o link que acabamos de criar. Enfim, terminamos por configurar o objeto **Email** usando os métodos herdados da classe do *Play!*.

Agora podemos alterar a lógica no controller e enviar um email já completo para o usuário, terminando por mostrar uma mensagem de sucesso e

```
public Result salvaNovoUsuario() {
    //...
    token.save();
    enviador.send(new EmailDeCadastro(token));
    flash("success", "Um email foi enviado para que você confirme seu cadastro!");
    return redirect("/login"); // TODO rota
}
```

Sucesso! Se tentarmos cadastrar um usuário agora, podemos ver algo deste gênero no log do servidor.

```
[info] application - mock implementation, send email
[info] application - subject: Confirmação de cadastro na API de produtos!
[info] application - from: Caelum <no-reply@caelum.com.br>
[info] application - bodyHtml: Olá Marco Salles! Por favor clique no link a seguir para confirma
[info] application - to: Marco Salles <marco.salles@caelum.com.br>
```