



Desenvolvimento para Servidores-II Spring Email

Neste tópico abordaremos o Spring Email mostrando como enviar e-mails através de um serviço de e-mail do Google.

Prof. Ciro Cirne Trindade

Introdução

- O Spring Email é uma biblioteca de utilitários útil para enviar e-mail
- Deixa transparente para usuário as especificações do sistema de correspondência subjacente
- Responsável pelo gerenciamento de recursos de baixo nível em nome do cliente

E-mail Simples

- O pacote `org.springframework.mail` é o pacote de alto nível do suporte a e-mail do Spring
 - Interface `MailSender`: interface que provê funcionalidades básicas para enviar e-mails simples
 - Seu método `send` espera um `SimpleEmailMessage` como parâmetro
 - Classe `SimpleEmailMessage`: representa uma mensagem de e-mail simples, incluindo campos como *from*, *to*, *cc*, *subject* e *text*

Dependência do Spring Email

- Dependência no pom.xml

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

Configurações do Spring Email

- No arquivo application-dev.properties inclua essas propriedades de configuração do Spring Email para o **Gmail**

```
spring.mail.host=smtp.gmail.com
spring.mail.username=username
spring.mail.password=password
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.socketFactory.port=465
spring.mail.properties.mail.smtp.socketFactory.class=
    javax.net.ssl.SSLSocketFactory
spring.mail.properties.mail.smtp.socketFactory.fallback = false
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.ssl.enable=true
```

Senha de app

Configuração do Gmail

- Verificação de duas etapas
 - <https://support.google.com/accounts/answer/185839>

Ativar a verificação em duas etapas

Com a verificação em duas etapas, também conhecida como autenticação de dois fatores, você adiciona uma camada a mais de segurança à sua conta para o caso de a senha ser roubada. Depois de configurar a verificação em duas etapas, você fará login na conta usando:

- algo que você sabe, como sua senha;
- algo que você possui, como seu smartphone.

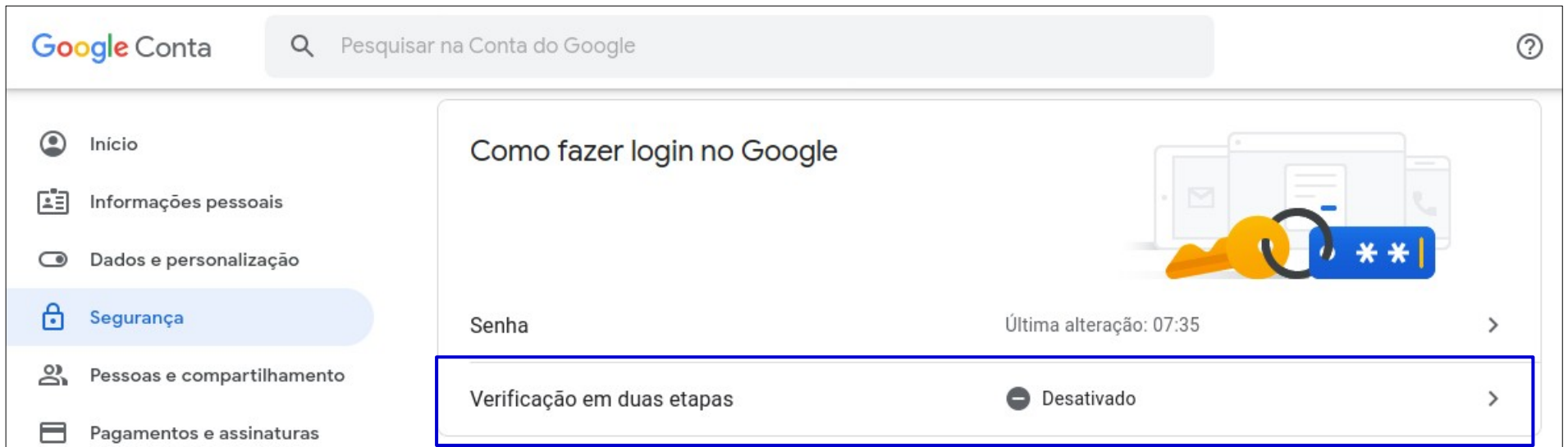
Ativar a verificação em duas etapas

1. Abra sua [Conta do Google](#).
2. No painel de navegação, selecione **Segurança**.
3. Em "Como fazer login no Google", selecione **Verificação em duas etapas** > **Primeiros passos**.
4. Siga as etapas exibidas na tela.

Sua conta, usuario@gmail.com, está associada ao seu trabalho ou à sua escola. Se não for possível configurar a verificação em duas etapas, [entre em contato com seu administrador](#).

Configuração do Gmail


- Verificação de duas etapas
 - Conta do Google → Segurança → Como fazer login no Google → Verificação e duas etapas



Configuração do Gmail


- Verificação de duas etapas
 - Siga as instruções

← Verificação em duas etapas



Vamos configurar seu smartphone

Qual número de telefone você quer usar?


 ▼ _____

O Google só usará este número para fins de segurança da conta.
Não use um número do Google Voice.
Podem ser cobradas tarifas padrão para o envio de mensagens e dados.

Como deseja receber os códigos?

☒ Mensagem de texto ☐ Chamada telefônica

← Verificação em duas etapas



Confirmar se ele funciona

O Google acaba de enviar uma mensagem de texto com um código de verificação para (13) 99-11-8833

[Digite o código](#)

Não recebeu? [Reenviar](#)

[VOLTAR](#) Etapa 2 de 3 [PRÓXIMA](#)

Configuração do Gmail

- Login com senha de app
 - <https://support.google.com/accounts/answer/185833?hl=pt-BR>

Fazer login com senhas de app

Dica: as senhas de app não são recomendadas e são desnecessárias na maioria dos casos. Para ajudar a manter sua conta segura, use o recurso "Fazer login com o Google" para conectar apps à sua Conta do Google.

As senhas de app têm 16 dígitos e permitem que um dispositivo ou app menos seguro acesse sua Conta do Google. As senhas de app podem ser usadas apenas em contas que tenham a [verificação em duas etapas](#) ativada.

Quando usar senhas de app

Dica: as senhas de app não são necessárias em iPhones e iPads com iOS 11 ou versões mais recentes. Em vez disso, use o recurso "Fazer login com o Google".

Se o app não oferecer essa opção, você poderá:

- usar senhas de app;
- usar um app ou dispositivo mais seguro.

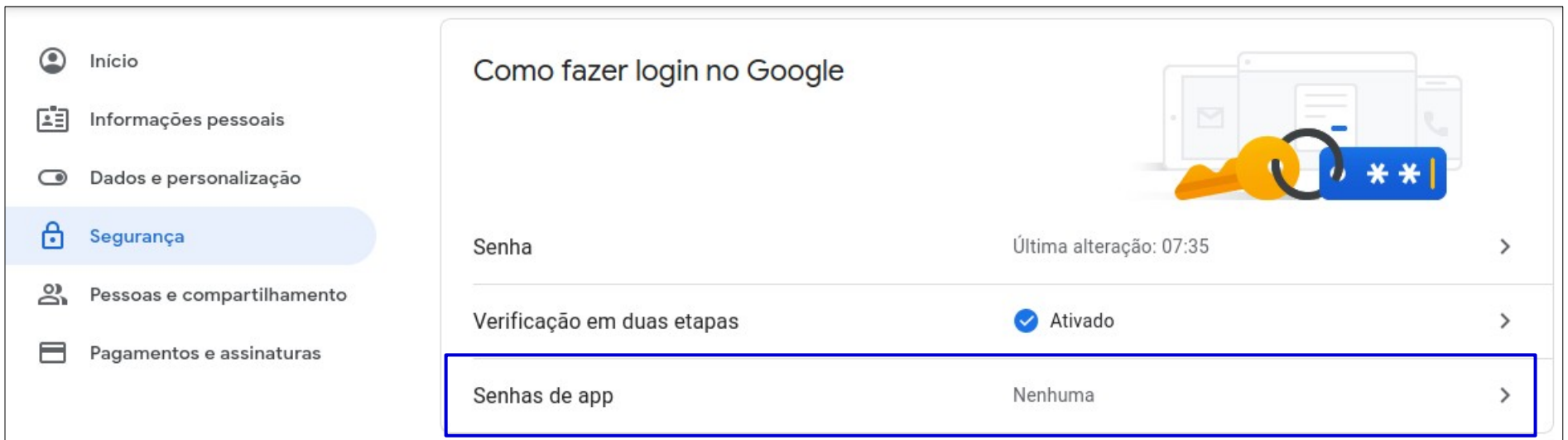
Criar e usar senhas de app

Se você usar a [verificação em duas etapas](#) e receber um erro de "senha incorreta" ao fazer login, tente usar uma senha de app.

1. Acesse sua [Conta do Google](#) .
2. Selecione **Segurança**.
3. Em "Como fazer login no Google", selecione **Senhas de app**. Talvez seja necessário fazer login. Se você não tiver essa opção, pode ser porque:


Configuração do Gmail

- Login com senha de app
 - Conta do Google → Segurança → Como fazer login no Google → Senhas de app



Configuração do Gmail

- Login com senha de app
 - Conta do Google → Segurança → Senhas de app



← Senhas de app

Senhas de app permitem que você faça login na sua Conta do Google a partir de apps em dispositivos que não sejam compatíveis com a verificação em duas etapas. Como só será necessário informar a senha uma vez, você não precisa memorizá-la. [Saiba mais](#)

Você não tem nenhuma senha de app.

Selecione o app e o dispositivo para o qual você quer gerar a senha de app.

E-mail ▼ Computador Windows ▼

GERAR

Selecione o app
E-mail e informe
o dispositivo

Clique em
Gerar

Configuração do Gmail

- Login com senha de app
 - Senha gerada

Senha de app gerada

Sua senha de app para computador Windows

Como usar

1. Abra o app "Mail".
2. Abra o menu "Configurações".
3. Selecione "Contas" e selecione sua Conta do Google.
4. Substitua sua senha pela senha de 16 caracteres mostrada acima.

Assim como sua senha normal, esta senha de app concede acesso total à sua Conta do Google. Não é necessário memorizá-la, por isso não a anote ou a compartilhe com outras pessoas.

[Saiba mais](#)

CONCLUÍDO

Valor da propriedade
`spring.mail.password`

Classe EmailDTO.java

```
@Getter
@Setter
@NoArgsConstructor
public class EmailDTO implements Serializable {
    private static final long serialVersionUID = 1L;

    private String to;
    private String subject;
    private String text;
}
```

Classe EmailService.java (1/2)

```
@Service
public class EmailService {
    @Autowired
    private MailSender mailSender;

    @Value("${spring.mail.username}")
    private String sender;

    public void sendEmail(EmailDTO email) {
        SimpleMailMessage message =
            prepareSimpleMailMessage(email);
        mailSender.send(message);
    }
}
```

Classe EmailService.java (2/2)

```
private SimpleMailMessage  
    prepareSimpleMailMessage(EmailDTO email) {  
        SimpleMailMessage mailMessage =  
            new SimpleMailMessage();  
        mailMessage.setTo(email.getTo());  
        mailMessage.setFrom(sender);  
        mailMessage.setSubject(email.getSubject());  
        mailMessage.setSentDate(  
            new Date(System.currentTimeMillis()));  
        mailMessage.setText(email.getText());  
        return mailMessage;  
    }  
}
```

Classe EmailController.java

```
@RestController
@RequestMapping("/email")
public class EmailController {
    @Autowired
    private EmailService service;

    @PostMapping("/simples")
    public ResponseEntity<?> enviarEmail(
        @RequestBody EmailDTO email) {
        try {
            service.sendEmail(email);
            return ResponseEntity.ok("E-mail simples enviado");
        } catch (MailException e) {
            return ResponseEntity.badRequest()
                .body(e.getMessage());
        }
    }
}
```


E-mail HTML

- Para enviar e-mails no formato HTML ou com anexos, o Spring Email provê:
 - Interface `JavaMailMessage`: estende a interface `MailSender` permitindo o envio de e-mails HTML e com anexo
 - Classe `MimeMessage`: representa um mensagem MIME

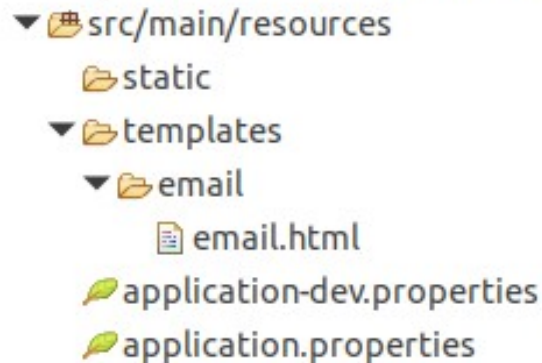


- Para criar um e-mail no formato HTML, vamos usar o Thymeleaf
- Thymeleaf é um framework de template Java para HTML
- Acrescentar a dependência do Thymeleaf no pom.xml

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

Template do Thymeleaf

- Crie uma pasta email no diretório src/main/resource/templates
- Dentro desta pasta crie o arquivo email.html



```
▼ src/main/resources
  static
  ▼ templates
    ▼ email
      email.html
    application-dev.properties
    application.properties
```

email.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <meta charset="UTF-8">
    <title th:remove="all">E-mail</title>
  </head>
  <body>
    <div>
      <h1>
        <span th:text="${email.text}"></span>
      </h1>
    </div>
  </body>
</html>
```

Namespace
do Thymeleaf

Remove a tag
e seus filhos

Avalia sua expressão e define
o resultado dessa avaliação
como o corpo da tag

Atualizando a classe EmailService.java (1/2)

```
public class EmailService {  
    ...  
    @Autowired  
    private TemplateEngine templateEngine;  
  
    @Autowired  
    private JavaMailSender javaMailSender;  
    ...  
  
    private String htmlFromTemplate(EmailDTO email) {  
        Context context = new Context();  
        context.setVariable("email", email);  
        return templateEngine  
            .process("email/email", context);  
    }  
}
```

Classe principal para
a execução de templates
do Thymeleaf

Classe do
Thymeleaf

Define o nome da variável
usada no template

Processa o template

Atualizando a classe EmailService.java (2/2)

```
public void sendHtmlEmail(EmailDTO email) throws  
    MessagingException {  
    MimeMessage message = prepareHtmlEmailMessage(email);  
    javaMailSender.send(message);  
}
```

```
private MimeMessage prepareHtmlEmailMessage(  
    EmailDTO email) throws MessagingException {  
    MimeMessage msg = javaMailSender.createMimeMessage();  
    MimeMessageHelper helper = new MimeMessageHelper(  
        msg, true);  
    helper.setTo(email.getTo());  
    helper.setFrom(sender);  
    helper.setSubject(email.getSubject());  
    helper.setSentDate(  
        new Date(System.currentTimeMillis()));  
    helper.setText(htmlFromTemplate(email), true);  
    return msg;  
}
```

Helper class
para popular um
MimeMessage

Atualizando a classe EmailController.java

```
public class EmailController {  
    ...  
  
    @PostMapping("/html")  
    public ResponseEntity<?> sendHtmlEmail(  
        @RequestBody EmailDTO email) {  
        try {  
            service.sendHtmlEmail(email);  
            return ResponseEntity.ok("E-mail HTML enviado");  
        } catch (MailException e) {  
            return ResponseEntity.badRequest()  
                .body(e.getMessage());  
        } catch (MessagingException e) {  
            return ResponseEntity.status(HttpStatus  
                .INTERNAL_SERVER_ERROR).body(e.getMessage());  
        }  
    }  
}
```

Referências

- Spring Email. Disponível em:
<https://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mail.html>
- Thymeleaf. Disponível em:
<https://www.thymeleaf.org/>