01

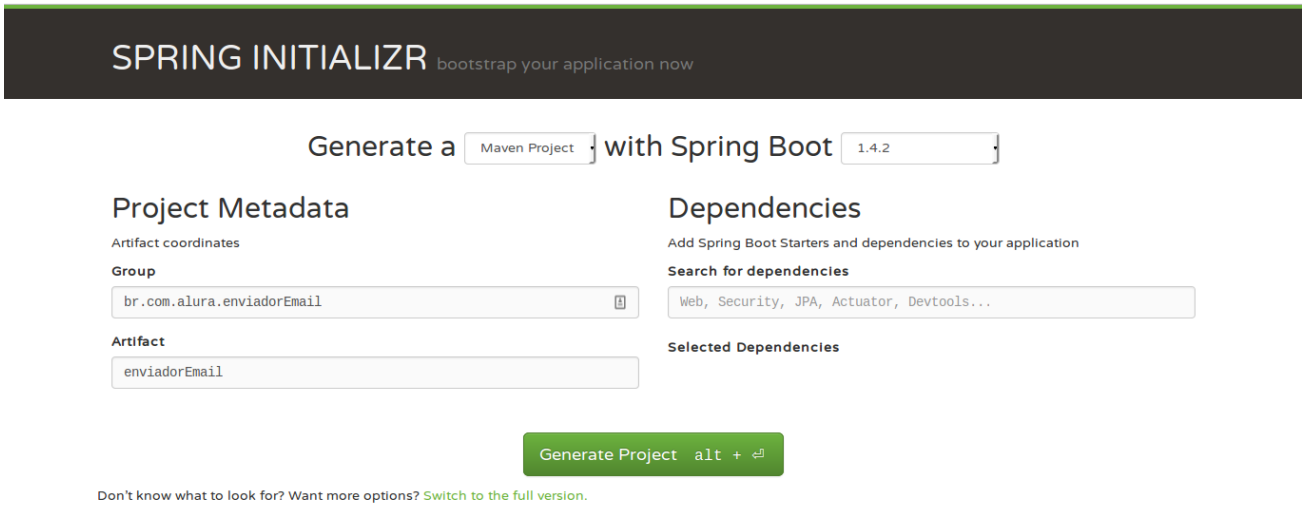
Criando aplicação com o Spring Initializr

Transcrição

Em nossa aplicação está faltando uma regra de negócio importante que havia na aplicação antiga: O envio de emails. Os convidados ao serem adicionados à lista VIP devem ser notificados via email, indicando que estes foram convidados.

Nosso cliente, aproveitando do momento de modernização da aplicação nos solicitou que o envio de emails fosse realizado em um serviço diferente, para que este pudesse ser reaproveitado em outros de seus projetos.

Resolveremos este problema criando uma outra aplicação que será responsável apenas pelo envio de emails. Poderíamos iniciar este projeto manualmente, mas por questões de praticidade, utilizaremos o *Spring Initializr*, disponível em start.spring.io (http://start.spring.io/).



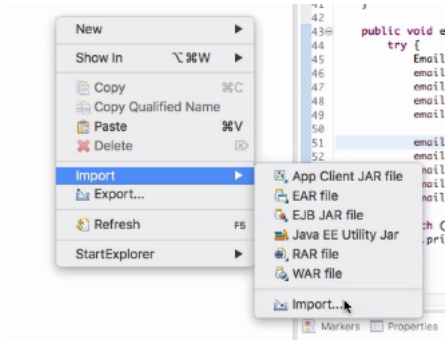
O *Spring Initializr*, permite que inicializemos rapidamente uma aplicação com *Spring Boot* utilizando de uma página web onde podemos adicionar uma série de dependências apenas clicando em alguns botões.

> Experimente clicar no link **Switch to the full version.** para verificar todas as possíveis opções que podemos deixar pré-configuradas no projeto.

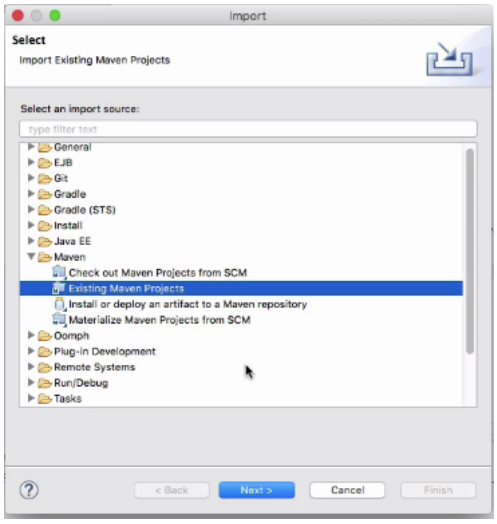
Apenas selecionamos a versão mais recente do *Spring Boot*, configuramos o `GroupId` com o valor `br.com.alura.enviadorEmail` e o `Artifact` como `enviadorEmail` e clicamos no botão: `Generate Project`.

Após clicar no botão para gerar o projeto. Um arquivo zip será baixado e este é o projeto com *Spring Boot* configurado, com todas as dependências mais básicas e outras caso tenha selecionado mais. Precisamos apenas importar no *Eclipse* e permitir que o *Maven* baixe todas as dependências.

1 - Selecione a opção de importar:



2 - Selecione a opção importar projeto Maven existente:



3 - A próxima etapa é apenas selecionar a pasta do projeto e clicar em ok. **Lembre-se de descompactar o arquivo gerado pelo Spring Initializr**.

> Experimente explorar como o *Spring Initializr* criou o projeto. Quais classes já estão prontas, quais dependências estão no `pom.xml`. **Explore!**

Precisamos também adicionar uma nova dependência em nosso projeto. Esta se trata de uma biblioteca que fará o envio dos emails. Usaremos então a `Apache Commons Email`. No `pom.xml`, adicionaremos:

```
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-email</artifactId>
    <version>1.4</version>
</dependency>
```

O próximo passo é criar a classe responsável por enviar os emails e configura-la como um serviço. Chamaremos esta de `EmailService`, ela será anotada com `@Service` e terá um método `enviar` que não terá retorno e receberá como parâmetro o nome e o email do convidado.

```
@Service
public class EmailService {
    public void enviar(String nome, String emailConvidado){

    }
}
```

O código presente na aplicação antiga é bem funcional. Só precisamos adaptá-lo ao nosso novo serviço. Assim não teremos que codificar tudo novamente. Vejamos o código da aplicação antiga.

```
public void enviarEmail(Convidado convidado) {
        try {
            Email email = new SimpleEmail();
            email.setHostName("smtp.googlemail.com");
            email.setSmtpPort(465);
            email.setAuthenticator(new DefaultAuthenticator("email", "password"));
            email.setSSLOnConnect(true);

            email.setFrom("email");
            email.setSubject("Você foi convidado pelo ListaVIP");
            email.setMsg("Olá " + convidado.getNome() + ". Você acaba de ser convidado pelo List
            email.addTo(convidado.getEmail());
            email.send();

        } catch (EmailException e) {
            e.printStackTrace();
        }
}
```

Precisamos apenas copiar o bloco `try/catch` alterando os lugares onde se usa o objeto `convidado` para utilizar o nome e o email recebidos por parâmetro.

```
@Service
public class EmailService {
    public void enviar(String nome, String emailConvidado){
        try {
            Email email = new SimpleEmail();
            email.setHostName("smtp.googlemail.com");
            email.setSmtpPort(465);
            email.setAuthenticator(new DefaultAuthenticator("email", "password"));
            email.setSSLOnConnect(true);

            email.setFrom("email");
            email.setSubject("Você foi convidado pelo ListaVIP");
            email.setMsg("Olá " + nome + ". Você acaba de ser convidado pelo ListaVIP.");
            email.addTo(emailConvidado);
            email.send();

        } catch (EmailException e) {
            e.printStackTrace();
        }
    }
}
```

Agora precisamos gerar o artefato deste projeto com o *Maven* para que possamos utilizar a biblioteca nos outros projetos, como o Lista VIPs inclusive. Com a ajuda do terminal, vá até a pasta do projeto `enviadorEmail` e use o comando `mvn install` .

Após o comando, algumas tarefas serão realizadas pelo *Maven* e o **.jar** do projeto `enviadorEmail` estará disponível na pasta `target` do projeto.

Lembre-se de trocar os dados de autenticação e uso do servidor de email para garantir o funcionamento com seu servidor de emails.