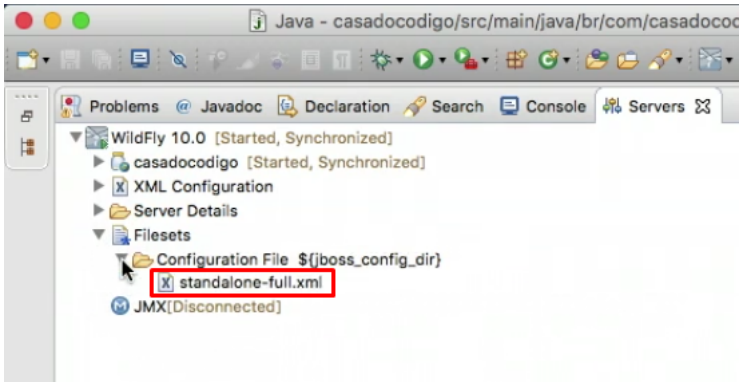
03

Testando o envio de email

Transcrição

O próximo passo é ter a Classe `Session` funcionando. Fizemos o `@Resource(mappedName = "java:/jboss/mail")` . Porém de onde vem o `java:/jboss/mail` ? Toda vez que se trata de Java EE, algumas configurações devem ser feitas. Nesse caso será no servidor. Na aba "Servers" selecionamos "Wildfly > Filesets > Configuration File", dentro dessa pasta achamos o arquivo `standalone-full.xml` :



Nós já trabalhamos anteriormente com esse arquivo, nos cursos passados, para configurar os *datasources*. Neste caso vamos mexer no `<mailsession>` :

```
<subsystem xmlns="urn:jboss:domain:mail:2.0">
    <mail-session name="default" jndi-name="java:jboss/mail/Default">
        <smtp-server outbound-socket-binding-ref="mail-smtp"/>
    </mail-session>
</subsystem>
```

Verifique se o `xmlns` é o mesmo que o apresentado acima. O `<mail-session>` padrão é apenas um exemplo, e perceba que ele tem um `jndi` diferente daquele com o qual estamos trabalhando. Vamos criar algo parecido:

```
<mail-session name="gmail" jndi-name="java:jboss/mail/gmail">

</mail-session>
```

Vamos dar o nome e o identificador de `gmail` (mudamos isso no resource também) pois ele será nosso servidor de envio de e-mail. Ele é o mais fácil e direto de ser usado, mas isso vai depender da produção e se você tiver um servidor próprio. Para o `smtp-server` fazemos

```
<smtp-server outbound-socket-biding-ref="mail-smtp-gmail" ssl="true" username="alura.springmvc@g
```

A grande maioria dos servidores de e-mail utilizam *ssl*, então acrescentamos isso, além do `username` e da senha (ambos retirados do nosso curso de Spring MVC) para fazer esse processo de envio. No fim teremos:

```
<mail-session name="gmail" jndi-name="java:jboss/mail/gmail">
    <smtp-server outbound-socket-biding-ref="mail-smtp-gmail" ssl="true" username="alura.springr
</mail-session>
```

Perceba que no `smtp-server` usamos o `outbound-socket-biding-ref` . Se formos para o final do arquivo, por volta da linha 465, podemos verificar isso:

```
<outbound-socket-binding name="mail-smtp">
    <remote-destination host="localhost" port="25"/>
</outbound-socket-binding>
```

Precisaremos de um outro desse para o Gmail configurando o host e uma porta segura:

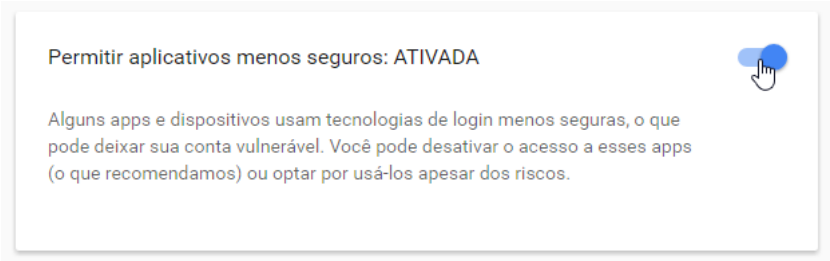
```
<outbound-socket-binding name="mail-smtp-gmail">
    <remote-destination host="smtp.gmail.com" port="465"/>
</outbound-socket-binding>
```

Temos, assim, uma configuração base do nosso `standalone-full.xml` . E um dos passos mais importantes dela é a referência ao JNDI no `@Resource` :

```
@Resource(mappedName = "java:/jboss/mail/gmail")
```

Vocês podem estar se perguntando porque não utilizamos o `@Inject` no lugar do `@Resource` . Sempre que utilizamos um recurso que é captado pelo JNDI, o padrão do Java EE é o `@Resource` que já injeta o objeto para conseguirmos fazer o envio do e-mail através dos recursos injetados. O `@Inject` ainda não tem essa integração.

No seu e-mail do Gmail devemos fazer uma última configuração. Acessamos a [página de segurança (https://myaccount.google.com/security#connectedapps)](https://myaccount.google.com/security#connectedapps) e em "Aplicativos e sites conectados" selecionamos a opção "Permitir aplicações menos seguras":



No `MailSender.java` , no `catch` fazemos:

```
catch (MessagingException e) {
    throw new RuntimeException(e);
}
```

Dessa forma temos certeza que a Exception será vista por alguém. Já no `PagamentoService.java` , o `context.getContextPath()` está dentro do `executor.submit()` . Dessa maneira ele não irá funcionar pois ele já se perdeu e libera a thread. O que temos que fazer é extraí-lo para uma variável fora do `executor` :

```
String contextPath = context.getContextPath();

executor.submit(() -> {
    try {
        //...

        URI responseUri = UriBuilder.fromPath("http://localhost:8080" + contextPath + "/index.xl
        //...
    }
});
```

Para testar reiniciamos obrigatoriamente o servidor. Adicionamos um livro ao Carrinho de Compras e finalizamos o pedido colocando o próprio e-mail configurado, por motivos didáticos. E receberemos o seguinte na caixa de entrada:

Funcionou! Agora veremos o problema do tempo que demorou entre finalizar a compra e voltar para a tela inicial.