[Bài đọc] Trình khởi động Mail Starter

Quản lý sự phụ thuộc(dependency) là một khía cạnh quan trọng của bất kỳ dự án phức tạp nào. Và làm điều này theo cách thủ công là ít lý tưởng hơn; bạn càng dành nhiều thời gian cho nó thì bạn càng có ít thời gian hơn cho các khía cạnh quan trọng khác của dự án.

Bộ khởi động Spring Boot(Spring Boot starters) được xây dựng để giải quyết chính xác vấn đề này. Starter POMs là một tập hợp các bộ mô tả phụ thuộc thuận tiện mà bạn có thể đưa vào ứng dụng của mình. Bạn nhận được một điểm dừng duy nhất cho tất cả Spring và công nghệ liên quan mà bạn cần, mà không cần phải tìm kiếm mã mẫu và sao chép-dán vô số bộ mô tả phụ thuộc(dependency descriptors).

Chúng ta có sẵn hơn 30 trình khởi động Boot - hãy xem một trong số chúng:

Mail Starter

Một nhiệm vụ rất phổ biến trong phát triển doanh nghiệp là gửi email và xử lý trực tiếp với Java Mail API thường có thể khó khăn.

Bộ khởi động Spring Boot che giấu sự phức tạp này - các phụ thuộc mail có thể được chỉ định theo cách sau:

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-mail</artifactId>

</dependency>

Bây giờ chúng ta có thể trực tiếp sử dụng JavaMailSender, vì vậy hãy viết một số test.

Đối với mục đích test, chúng ta cần một máy chủ SMTP đơn giản. Trong ví dụ này, chúng ta sẽ sử dụng Wiser. Đây là cách chúng ta có thể đưa nó vào POM:

<dependency>

<groupId>org.subethamail</groupId>

<artifactId>subethasmtp</artifactId>

<version>3.1.7</version>

<scope>test</scope>

</dependency>

Phiên bản mới nhất của Wiser có thể được tìm thấy trên kho lưu trữ trung tâm Maven ([Maven central repository](https://search.maven.org/classic/#search%7Cga%7C1%7Csubethasmtp)).

Đây là mã nguồn cho test:

@RunWith(SpringJUnit4ClassRunner.class)

@SpringApplicationConfiguration(classes = Application.class)

public class SpringBootMailTest {

@Autowired

private JavaMailSender javaMailSender;

private Wiser wiser;

private String userTo = "user2@localhost";

private String userFrom = "user1@localhost";

private String subject = "Test subject";

private String textMail = "Text subject mail";

@Before

public void setUp() throws Exception {

final int TEST\_PORT = 25;

wiser = new Wiser(TEST\_PORT);

wiser.start();

}

@After

public void tearDown() throws Exception {

wiser.stop();

}

@Test

public void givenMail\_whenSendAndReceived\_thenCorrect() throws Exception {

SimpleMailMessage message = composeEmailMessage();

javaMailSender.send(message);

List<WiserMessage> messages = wiser.getMessages();

assertThat(messages, hasSize(1));

WiserMessage wiserMessage = messages.get(0);

assertEquals(userFrom, wiserMessage.getEnvelopeSender());

assertEquals(userTo, wiserMessage.getEnvelopeReceiver());

assertEquals(subject, getSubject(wiserMessage));

assertEquals(textMail, getMessage(wiserMessage));

}

private String getMessage(WiserMessage wiserMessage)

throws MessagingException, IOException {

return wiserMessage.getMimeMessage().getContent().toString().trim();

}

private String getSubject(WiserMessage wiserMessage) throws MessagingException {

return wiserMessage.getMimeMessage().getSubject();

}

private SimpleMailMessage composeEmailMessage() {

SimpleMailMessage mailMessage = new SimpleMailMessage();

mailMessage.setTo(userTo);

mailMessage.setReplyTo(userFrom);

mailMessage.setFrom(userFrom);

mailMessage.setSubject(subject);

mailMessage.setText(textMail);

return mailMessage;

}

}

Phương thức @Before và @After chịu trách nhiệm khởi động và dừng mail server.

Cũng giống như bất kỳ cài đặt mặc định nào khác trong Boot, cài đặt email cho JavaMailSender có thể được tùy chỉnh trong application.properties:

spring.mail.host=localhost

spring.mail.port=25

spring.mail.properties.mail.smtp.auth=false

Vì vậy, chúng ta đã định cấu hình mail server trên localhost: 25 và chúng ta không yêu cầu xác thực.

Last modified: Wednesday, 18 August 2021, 12:58 AM