Rendszertervezés labor 2
AUT1 mérés
IMSc feladatok dokumentációja

Telbisz Csanád
ESV6L2

2022. 10. 03.

Megoldás ismertetése

A feliratkozásoknak új entitást hoztam létre, melyben tárolom a feliratkozó nevét, email-címét és a feliratkozás tárgyát képező hirdetésre mutató referenciát (az adatbázisban idegen kulcsot). A controller osztályt kibővítettem egy *post* metódussal, aminek segítségével új feliratkozás hozható létre és perzisztálható. Ezen kívül a hirdetést frissítő metódust volt szükséges kibővíteni, hogy sikeres frissítés esetén emailt küldjön minden egyes felhasználónak.

Az email küldés némi konfigurációból és az üzenetet összeállító kódrészletekből áll. Szöveges üzenet esetén a szövegtörzset egyszerű sztring építéssel állítom elő. A thymeleaf-es feladatban pedig egy sablon HTML fájl kerül beolvasásra, amibe a paraméterek helyére a megadott értékek kerülnek behelyettesítésre. Az üzenet elküldését pedig gyakorlatilag a felhasznált könyvtár (a Spring keretrendszer mail csomagja) végzi teljes mértékben: a megfelelő metódus meghívásán kívül más dolgunk nincs.

A megoldás forráskódrészletei

Subscription entitás

```
@Entity
public class Subscription {

   @Id
   @GeneratedValue
   private Long id;

   private String name;

   private String email;

   @ManyToOne
   private Ad ad;

   // getterek és setterek...
}
```

Subscription objektumok perzisztálásának kezelőosztálya:

```
@Repository
public class SubscriptionRepository {

   @PersistenceContext
   private EntityManager em;

   @Transactional
   public Subscription save(Subscription subscription) {
      return em.merge(subscription);
   }

   public List<Subscription> findByAd(Ad ad) {
      return em.createQuery("SELECT s FROM Subscription s WHERE s.ad = ?1",
Subscription.class)
            .setParameter(1, ad)
            .getResultList();
   }
}
```

Feliratkozást kezelő REST endpoint a controllerben:

```
@PostMapping("{id}/subscribe")
public Subscription subscribe(@PathVariable Long id, @RequestBody
Subscription subscription) {
   subscription.setAd(adRepository.findById(id));
   return subscriptionRepository.save(subscription);
}
```

Email küldéséhez szükséges plusz függőségek (Spring mail és thymeleaf):

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
<dependency>
    <groupId>org.thymeleaf</groupId>
    <artifactId>thymeleaf</artifactId>
</dependency>
<dependency>
    <groupId>org.thymeleaf</groupId>
    <artifactId>thymeleaf-spring5</artifactId>
</dependency>
```

Konfiguráció az application.yml-ben:

```
spring:
  mail:
    host: <host> # a konkrét értékeket elrejtettem...
    port: <port>
    username: <user>
    password: <password>
    properties:
      mail:
        smtp:
          auth: true
          starttls:
            enable: true
            required: true
    templates:
      path: template
```

Kódbeli konfigurációs metódusok a thymeleaf sablon motor és parser beállítására:

```
@Primary // primary nélkül összeakad egyes alapértelmezett konfigurációkkal
@Bean
public ITemplateResolver thymeleafTemplateResolver() {
   ClassLoaderTemplateResolver templateResolver = new
ClassLoaderTemplateResolver();
   templateResolver.setPrefix("template/");
   templateResolver.setSuffix(".html");
   templateResolver.setTemplateMode("HTML");
   templateResolver.setCharacterEncoding("UTF-8");
   return templateResolver;
}

@Primary
@Bean
public SpringTemplateEngine thymeleafTemplateEngine(ITemplateResolver
templateResolver) {
   SpringTemplateEngine templateEngine = new SpringTemplateEngine();
   templateEngine.setTemplateResolver(templateResolver);
   templateEngine.setTemplateEngineMessageSource(new
ResourceBundleMessageSource());
   return templateEngine;
}
```

Email küldő metódusok (egyszerű szöveges és HTML alapú emailekhez), plusz a függőség-injektált attribútumok:

```
private final String EMAIL_FROM = "<sender email address>"; // elrejtve...

@Autowired
private JavaMailSender emailSender;

@Autowired
private SpringTemplateEngine thymeleafTemplateEngine;


public void sendSimpleMessage(String to, String subject, String text) {
   try {
      SimpleMailMessage message = new SimpleMailMessage();
      message.setFrom(EMAIL_FROM);
      message.setTo(to);
      message.setSubject(subject);
      message.setText(text);
      emailSender.send(message);
   } catch (MailException exception) {
      exception.printStackTrace();
   }
}

public void sendMessageUsingThymeleafTemplate(String to, String subject,
Map<String, Object> templateModel) {
   Context thymeleafContext = new Context();
   thymeleafContext.setVariables(templateModel);
   String htmlBody = thymeleafTemplateEngine.process("email.html",
thymeleafContext);
   sendHtmlMessage(to, subject, htmlBody);
}
```

```
private void sendHtmlMessage(String to, String subject, String htmlBody) {
   MimeMessage message = emailSender.createMimeMessage();
   try {
      MimeMessageHelper helper = new MimeMessageHelper(message, true, "UTF-
8");
      helper.setFrom(EMAIL_FROM);
      helper.setTo(to);
      helper.setSubject(subject);
      helper.setText(htmlBody, true);
      emailSender.send(message);
   } catch (MessagingException e) {
      throw new RuntimeException(e);
   }
}
```

Az alkalmazott HTML thymeleaf sablon:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>

<body>
<p>Dear <span th:text="${recipient}"></span></p>
<p>Details of <i>Ad#<a th:text="${adId}"></a></i> has changed.</p>
<p>
    <b>Title:</b> <i th:text="${adTitle}"></i><br>
    <b>Description:</b> <a th:text="${adDescription}"></a><br>
    <b>Tags:</b> <code th:text="${adTags}"></code>
</p>
<p>
    Best regards,<br>
    <em>Ad Service</em>
</p>
<small>This is a HTML message.</small>
</body>
</html>
```

Egyszerű szöveges email küldése a controller reklámfrissítést végző metódusának végén:

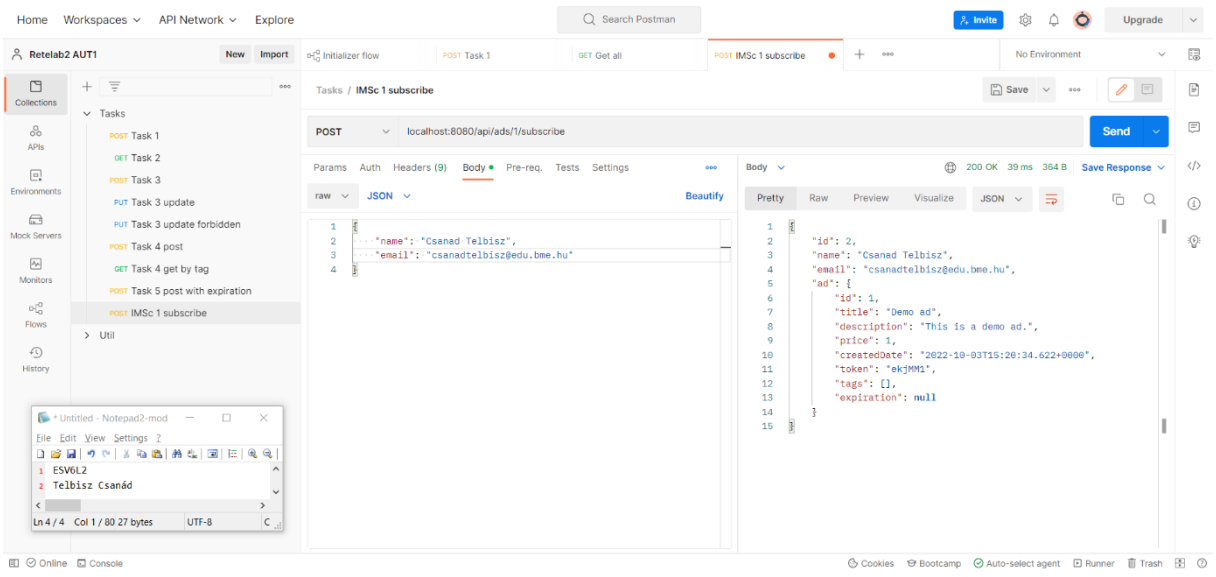
```
List<Subscription> subscriptions= subscriptionRepository.findByAd(savedAd);
Ad finalSavedAd = savedAd;
subscriptions.forEach(subscription -> {
   emailService.sendSimpleMessage(
      subscription.getEmail(),
      "Ad details changed",
      "Dear " + subscription.getName() + ",\n\n" +
         "Details of Ad#" + finalSavedAd.getId() + " has changed.\n\n" +
         "Title: " + finalSavedAd.getTitle() + "\n" +
         "Description: " + finalSavedAd.getDescription() + "\n" +
         "Tags: " + (finalSavedAd.getTags() == null ? "-" : String.join(
                  ", ", finalSavedAd.getTags())) + "\n\n" +
         "Best regards\nAd Service"
   );
});
```

Ugyanitt a HTML alapú level elküldése:

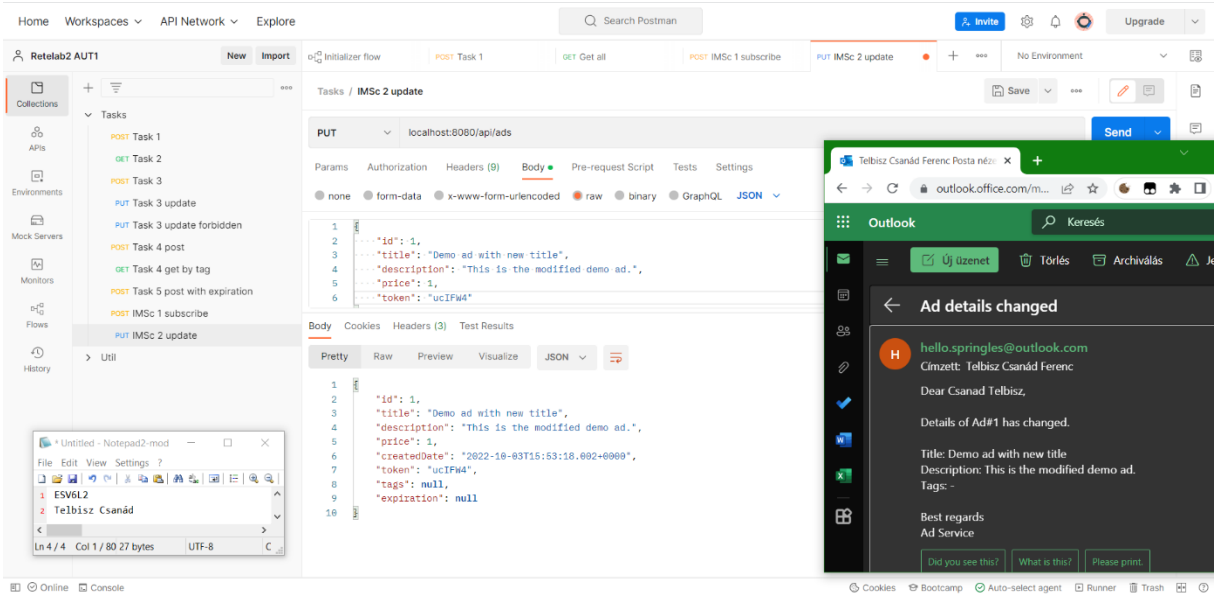
```
List<Subscription> subscriptions= subscriptionRepository.findByAd(savedAd);
Ad finalSavedAd = savedAd;
subscriptions.forEach(subscription -> {
    Map<String, Object> parameters = new HashMap<>();
    parameters.put("recipient", subscription.getName());
    parameters.put("adId", finalSavedAd.getId());
    parameters.put("adTitle", finalSavedAd.getTitle());
    parameters.put("adDescription", finalSavedAd.getDescription());
    parameters.put("adTags", finalSavedAd.getTags() == null ? "-" :
.           String.join(", ", finalSavedAd.getTags()));
    emailService.sendMessageUsingThymeleafTemplate(subscription.getEmail(),
.           "Ad details changed", parameters);
});
```

A működést igazoló képernyőképek

Feliratkozás művelete:



Egyszerű szöveges üzenet küldése és az eredmény ellenőrzése:



HTML alapú üzenet küldése és az eredmény ellenőrzése (látható a formázás eredménye):