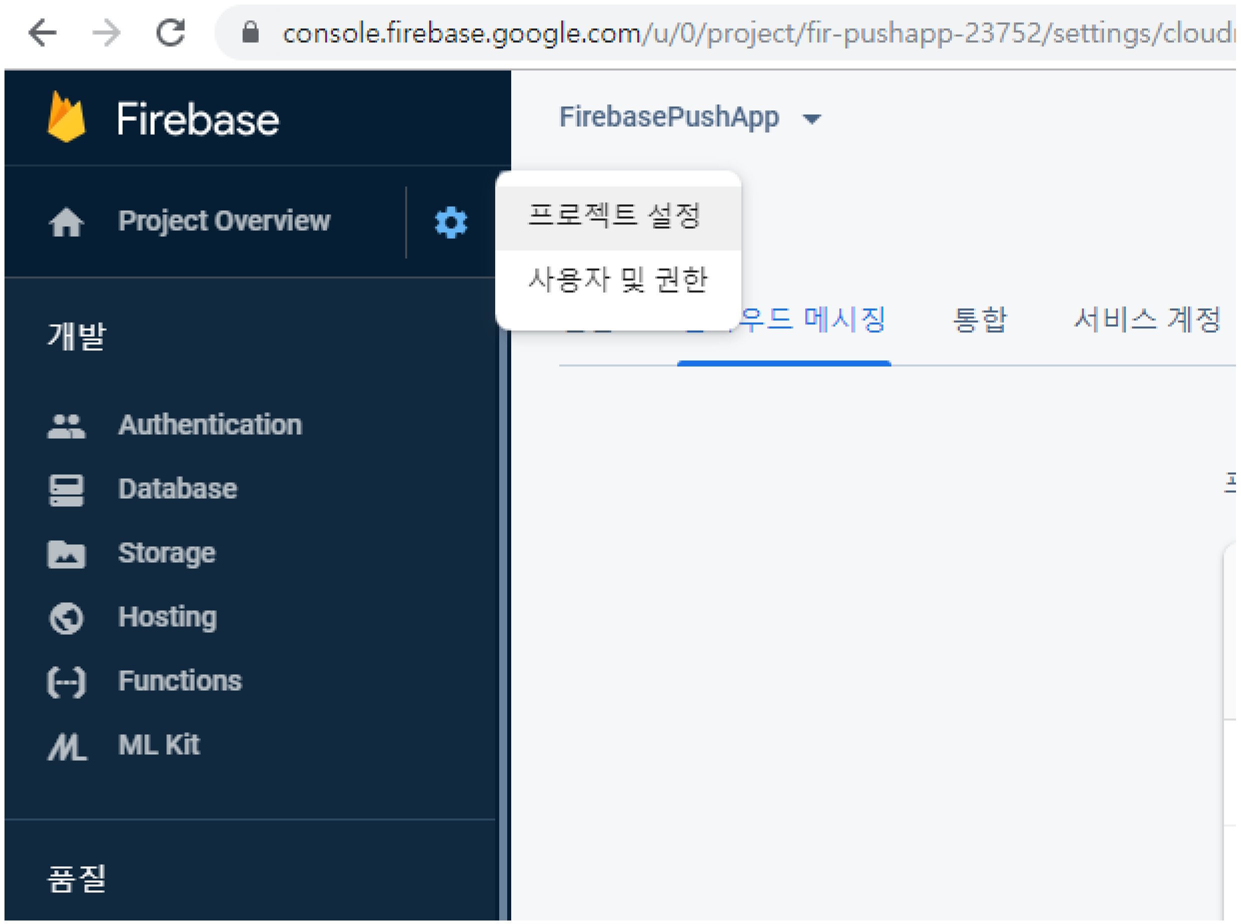
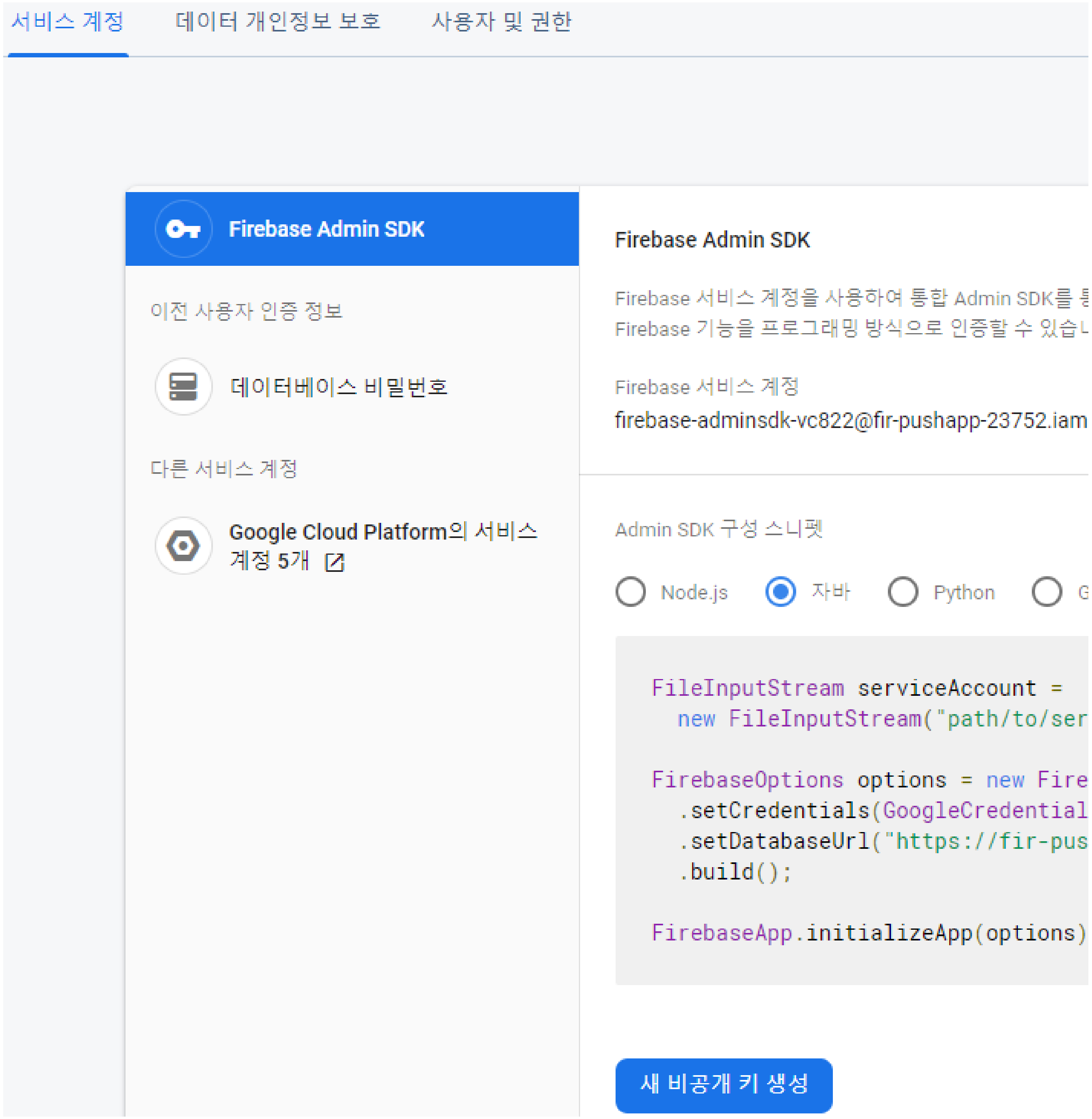
Spring with Firebase Cloud Message

Innova Lee(이상훈)
gcccompil3r@gmail.com


console.firebase.google.com/u/0/project/fir-pushapp-23752/settings/cloud
Firebase
FirebasePushApp
Project Overview
프로젝트 설정
사용자 및 권한
우드 메시징
통합
서비스 계정
개발
Authentication
Database
Storage
Hosting
Functions
ML Kit
품질



서비스 계정 데이터 개인정보 보호 사용자 및 권한

Firebase Admin SDK

이전 사용자 인증 정보

데이터베이스 비밀번호

다른 서비스 계정

Google Cloud Platform의 서비스 계정 5개

Firebase Admin SDK

Firebase 서비스 계정을 사용하여 통합 Admin SDK를
Firebase 기능을 프로그래밍 방식으로 인증할 수 있습니

Firebase 서비스 계정
firebase-adminsdk-vc822@fir-pushapp-23752.iam

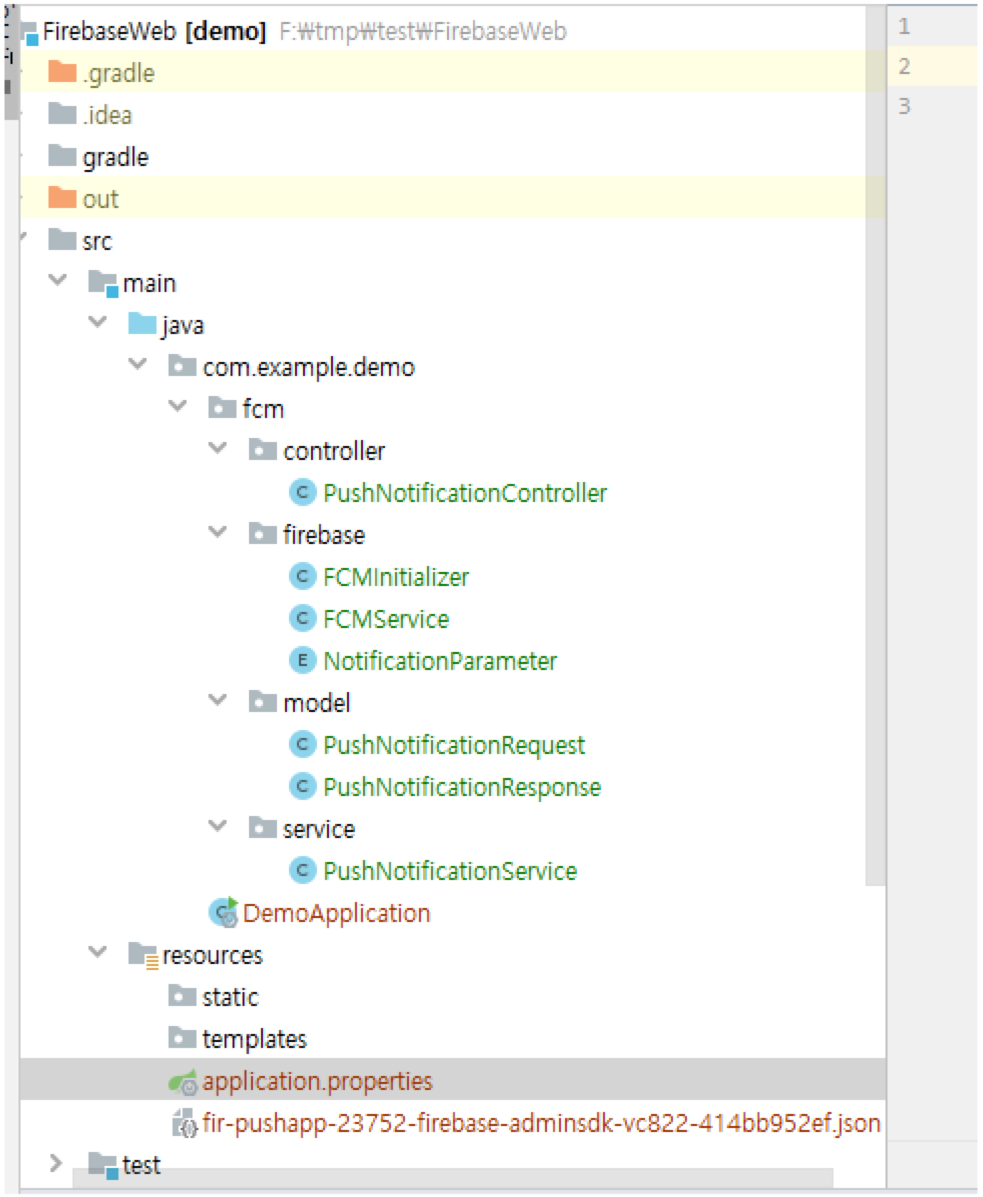
Admin SDK 구성 스니펫

Node.js [✓ 자바] Python G

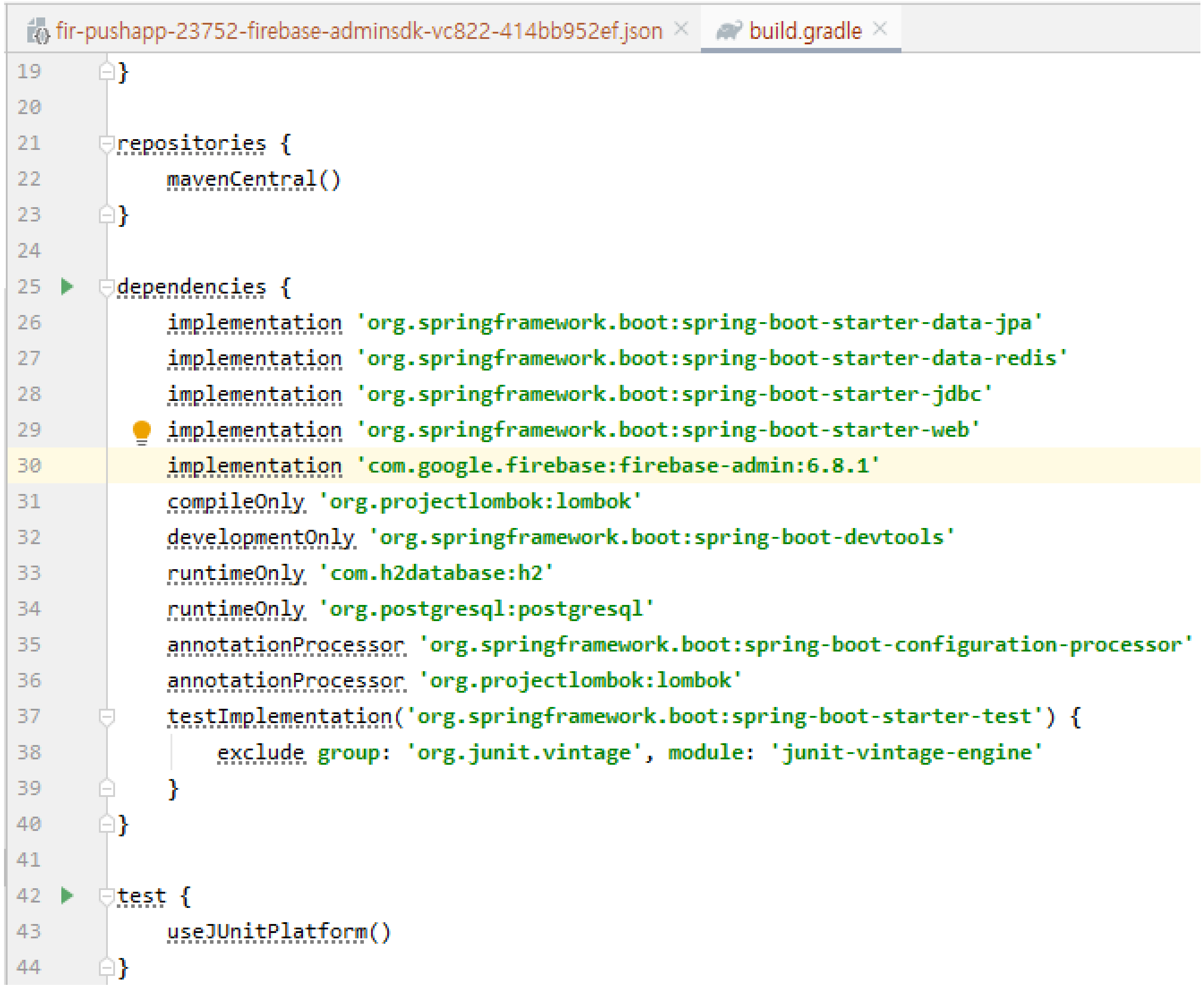
```
FileInputStream serviceAccount =
  new FileInputStream("path/to/ser

FirebaseOptions options = new Fire
  .setCredentials(GoogleCredential
  .setDatabaseUrl("https://fir-pus
  .build();

FirebaseApp.initializeApp(options)
```

새 비공개 키 생성

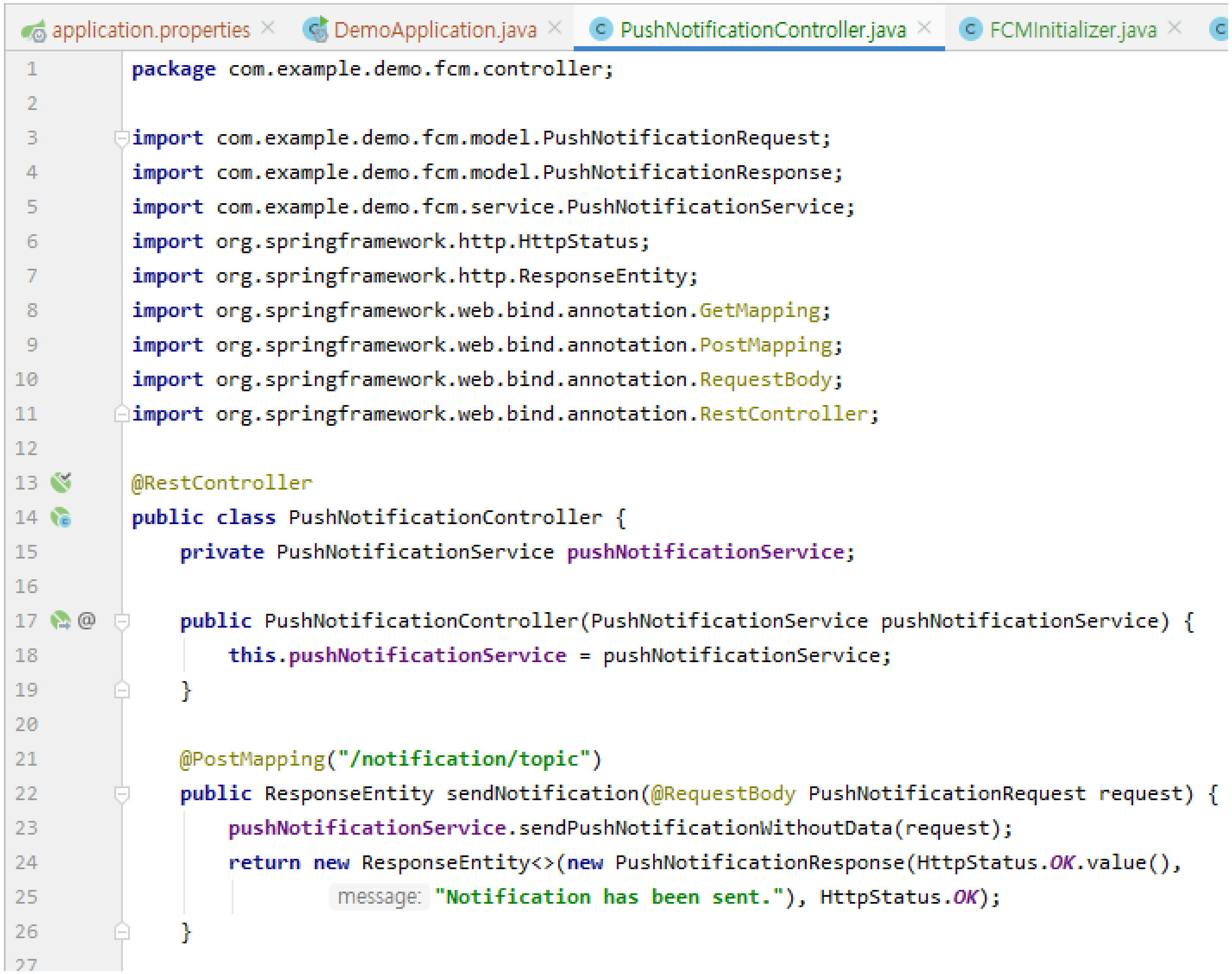
IJ
IntelliJ IDEA
ULTIMATE 2019.2
JET BRAINS



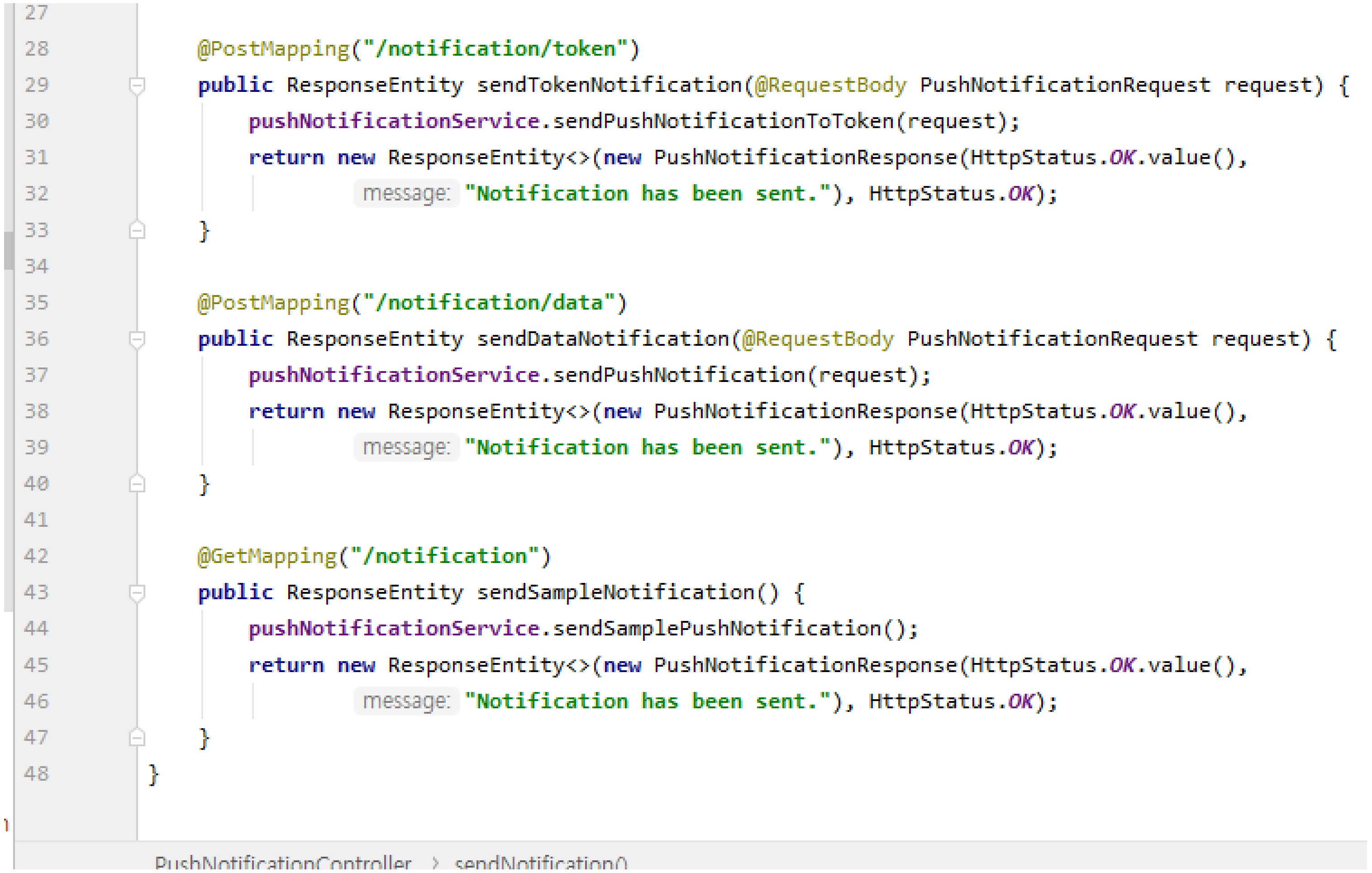
```
FirebaseWeb [demo]  F:₩tmp₩test₩FirebaseWeb
  .gradle
  .idea
  gradle
  out
  src
    main
      java
        com.example.demo
          fcm
            controller
              PushNotificationController
            firebase
              FCMInitializer
              FCMService
              NotificationParameter
            model
              PushNotificationRequest
              PushNotificationResponse
            service
              PushNotificationService
          DemoApplication
      resources
        static
        templates
        application.properties
        fir-pushapp-23752-firebase-adminsdk-vc822-414bb952ef.json
    test
```



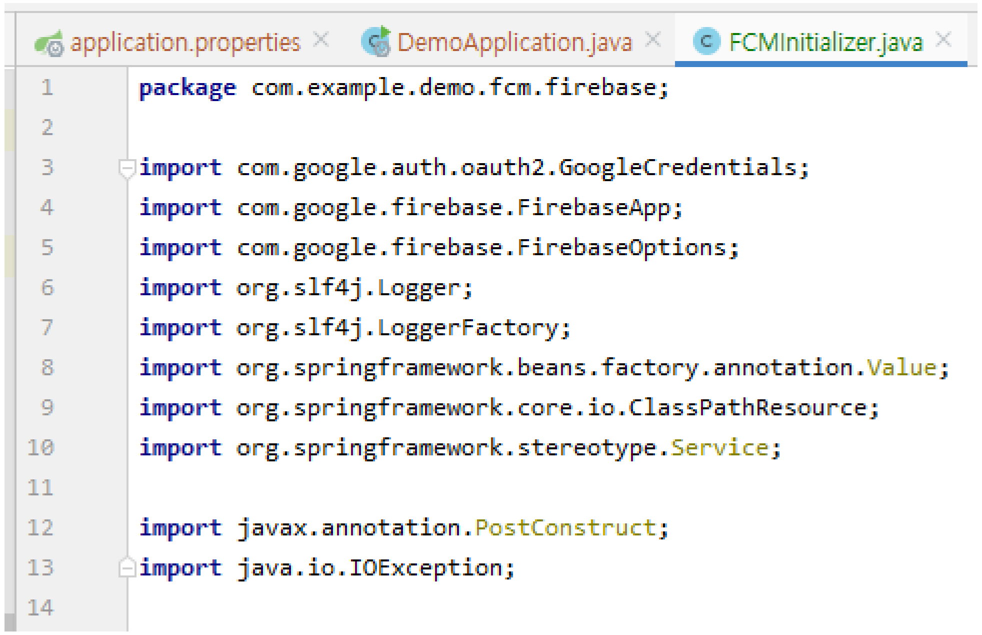
```
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'
    implementation 'org.springframework.boot:spring-boot-starter-jdbc'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'com.google.firebase:firebase-admin:6.8.1'
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'com.h2database:h2'
    runtimeOnly 'org.postgresql:postgresql'
    annotationProcessor 'org.springframework.boot:spring-boot-configuration-processor'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation('org.springframework.boot:spring-boot-starter-test') {
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
    }
}

test {
    useJUnitPlatform()
}
```


application.properties DemoApplication.java PushNotificationController.java FCMInitializer.java

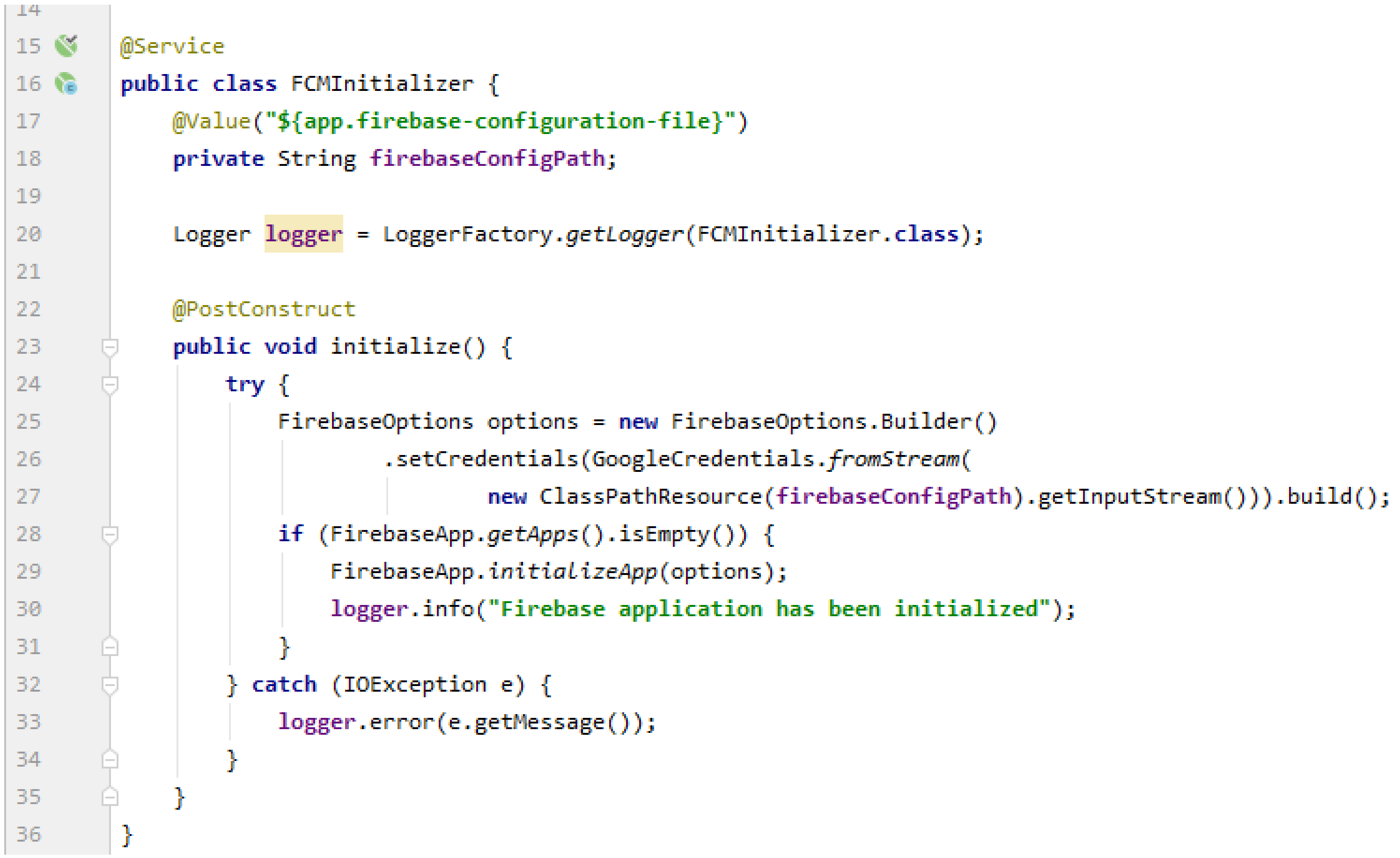
```java
package com.example.demo.fcm.controller;

import com.example.demo.fcm.model.PushNotificationRequest;
import com.example.demo.fcm.model.PushNotificationResponse;
import com.example.demo.fcm.service.PushNotificationService;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class PushNotificationController {
    private PushNotificationService pushNotificationService;

    public PushNotificationController(PushNotificationService pushNotificationService) {
        this.pushNotificationService = pushNotificationService;
    }

    @PostMapping("/notification/topic")
    public ResponseEntity sendNotification(@RequestBody PushNotificationRequest request) {
        pushNotificationService.sendPushNotificationWithoutData(request);
        return new ResponseEntity<>(new PushNotificationResponse(HttpStatus.OK.value(),
                message: "Notification has been sent."), HttpStatus.OK);
    }
```



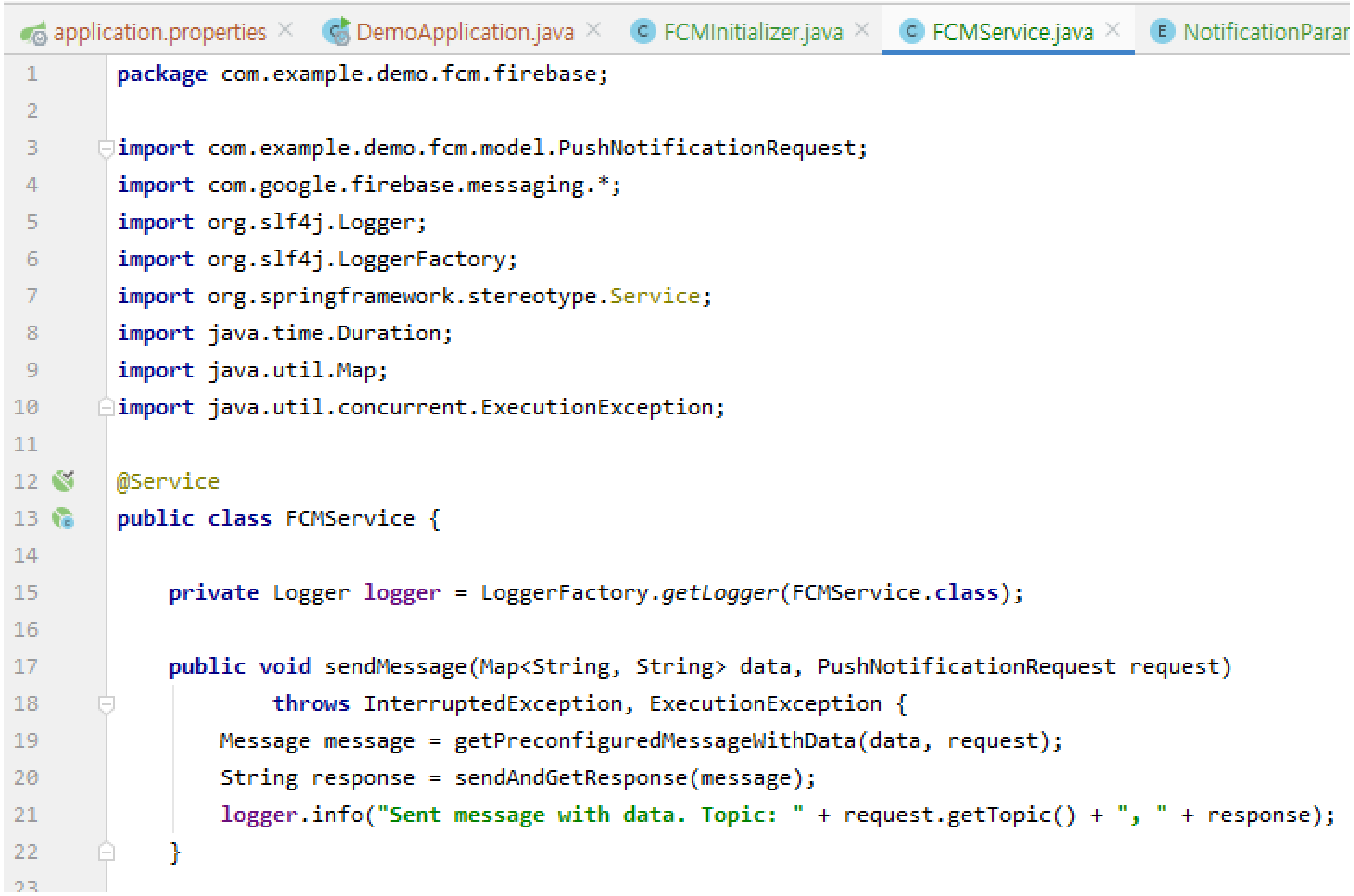
```java

    @PostMapping("/notification/token")
    public ResponseEntity sendTokenNotification(@RequestBody PushNotificationRequest request) {
        pushNotificationService.sendPushNotificationToToken(request);
        return new ResponseEntity<>(new PushNotificationResponse(HttpStatus.OK.value(),
                message: "Notification has been sent."), HttpStatus.OK);
    }

    @PostMapping("/notification/data")
    public ResponseEntity sendDataNotification(@RequestBody PushNotificationRequest request) {
        pushNotificationService.sendPushNotification(request);
        return new ResponseEntity<>(new PushNotificationResponse(HttpStatus.OK.value(),
                message: "Notification has been sent."), HttpStatus.OK);
    }

    @GetMapping("/notification")
    public ResponseEntity sendSampleNotification() {
        pushNotificationService.sendSamplePushNotification();
        return new ResponseEntity<>(new PushNotificationResponse(HttpStatus.OK.value(),
                message: "Notification has been sent."), HttpStatus.OK);
    }
}
```



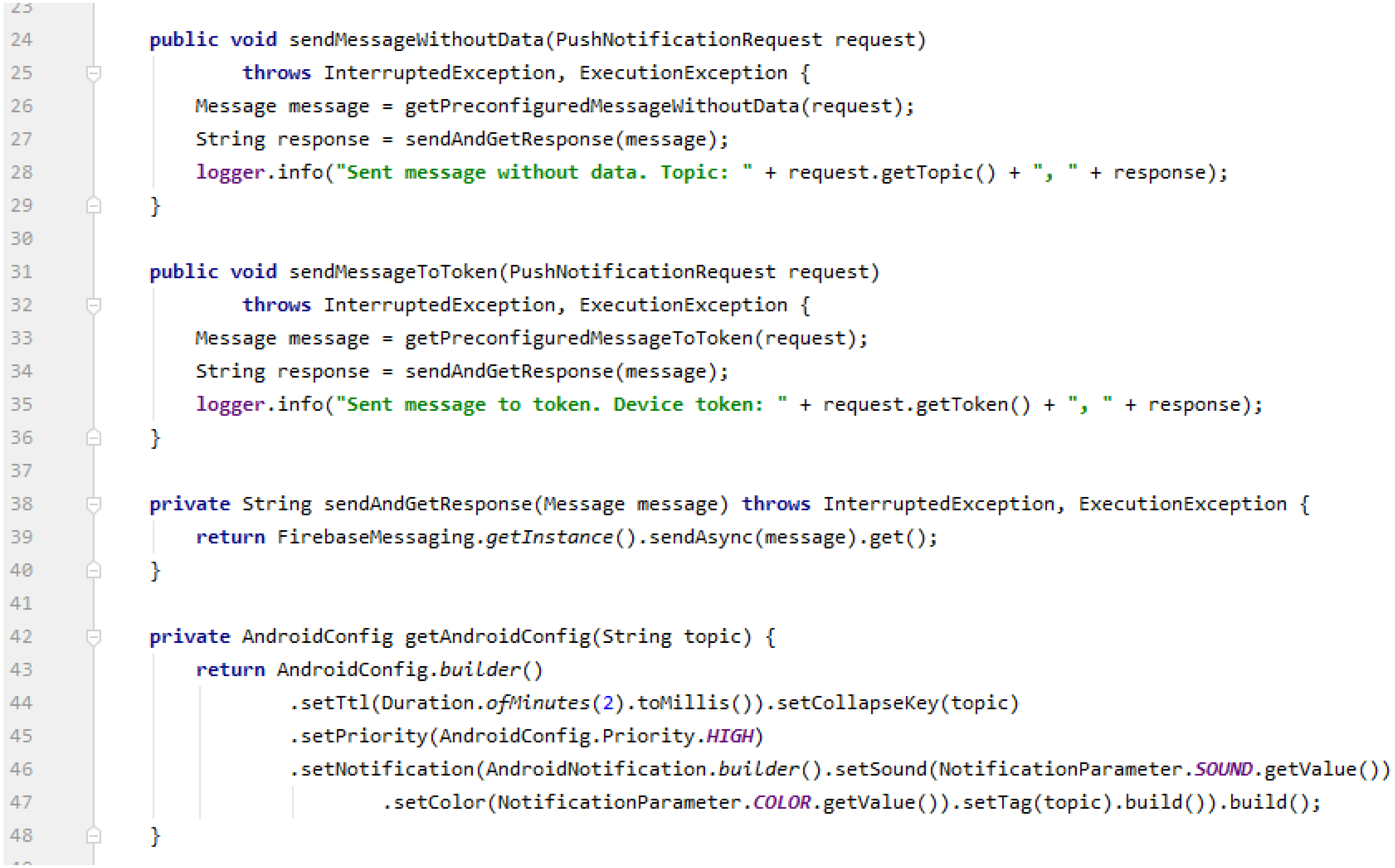
```java
package com.example.demo.fcm.firebase;

import com.google.auth.oauth2.GoogleCredentials;
import com.google.firebase.FirebaseApp;
import com.google.firebase.FirebaseOptions;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.io.ClassPathResource;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import java.io.IOException;

```



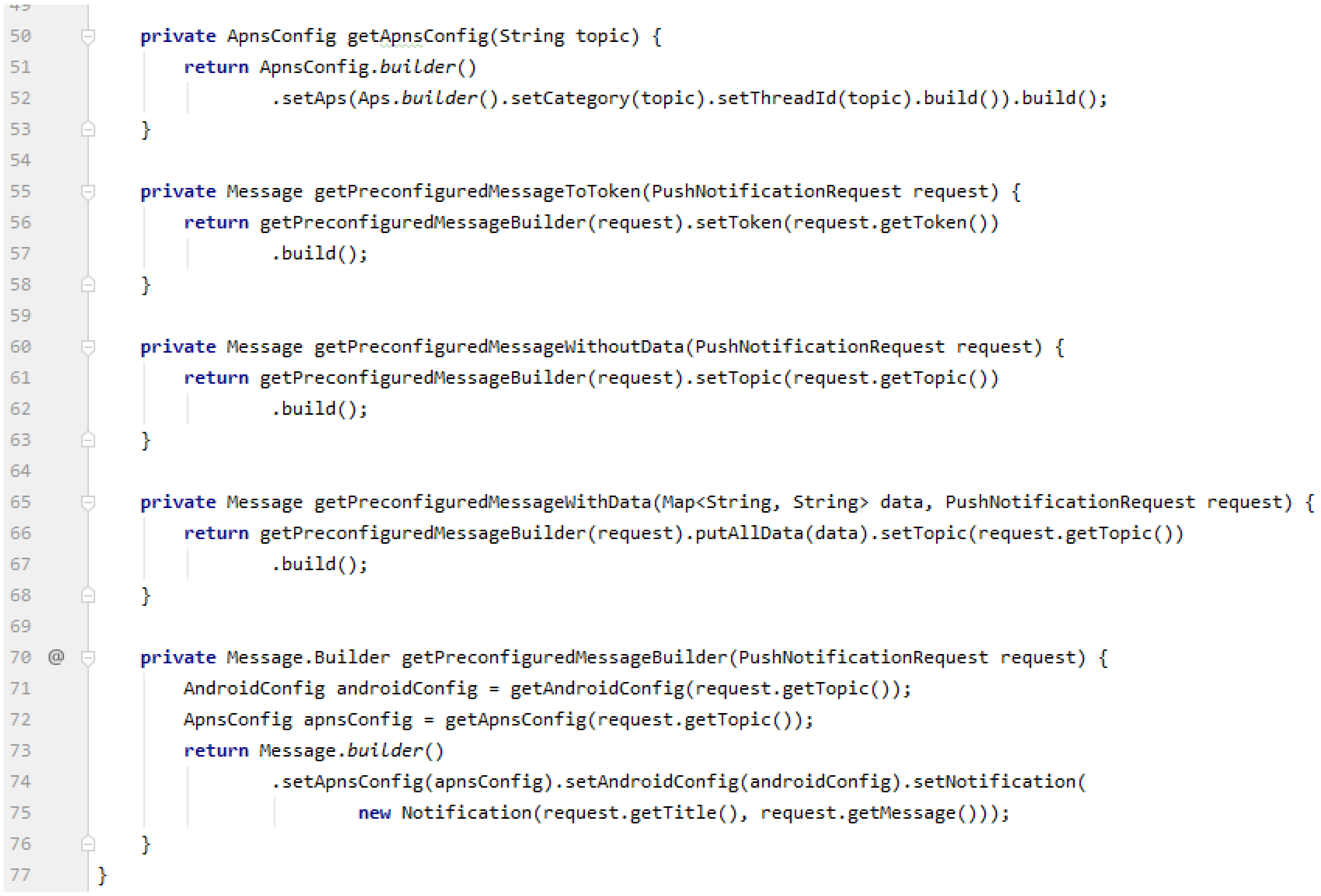
```
@Service
public class FCMInitializer {
    @Value("${app.firebase-configuration-file}")
    private String firebaseConfigPath;

    Logger logger = LoggerFactory.getLogger(FCMInitializer.class);

    @PostConstruct
    public void initialize() {
        try {
            FirebaseOptions options = new FirebaseOptions.Builder()
                    .setCredentials(GoogleCredentials.fromStream(
                            new ClassPathResource(firebaseConfigPath).getInputStream())).build();
            if (FirebaseApp.getApps().isEmpty()) {
                FirebaseApp.initializeApp(options);
                logger.info("Firebase application has been initialized");
            }
        } catch (IOException e) {
            logger.error(e.getMessage());
        }
    }
}
```



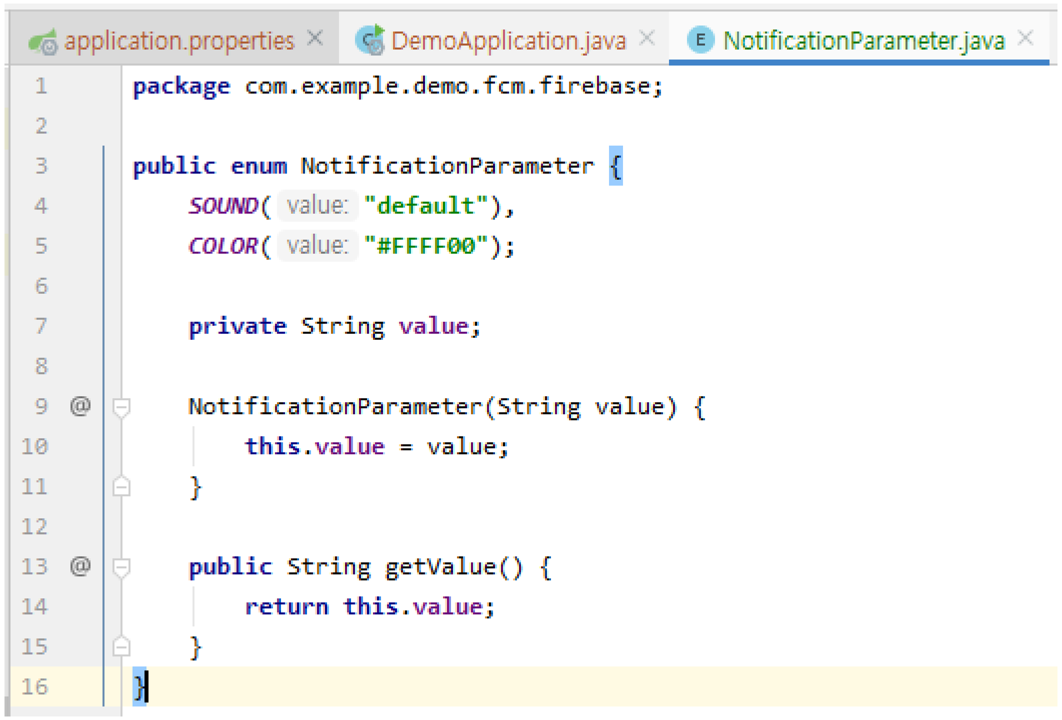
```java
package com.example.demo.fcm.firebase;

import com.example.demo.fcm.model.PushNotificationRequest;
import com.google.firebase.messaging.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Service;
import java.time.Duration;
import java.util.Map;
import java.util.concurrent.ExecutionException;

@Service
public class FCMService {

    private Logger logger = LoggerFactory.getLogger(FCMService.class);

    public void sendMessage(Map<String, String> data, PushNotificationRequest request)
            throws InterruptedException, ExecutionException {
        Message message = getPreconfiguredMessageWithData(data, request);
        String response = sendAndGetResponse(message);
        logger.info("Sent message with data. Topic: " + request.getTopic() + ", " + response);
    }
```



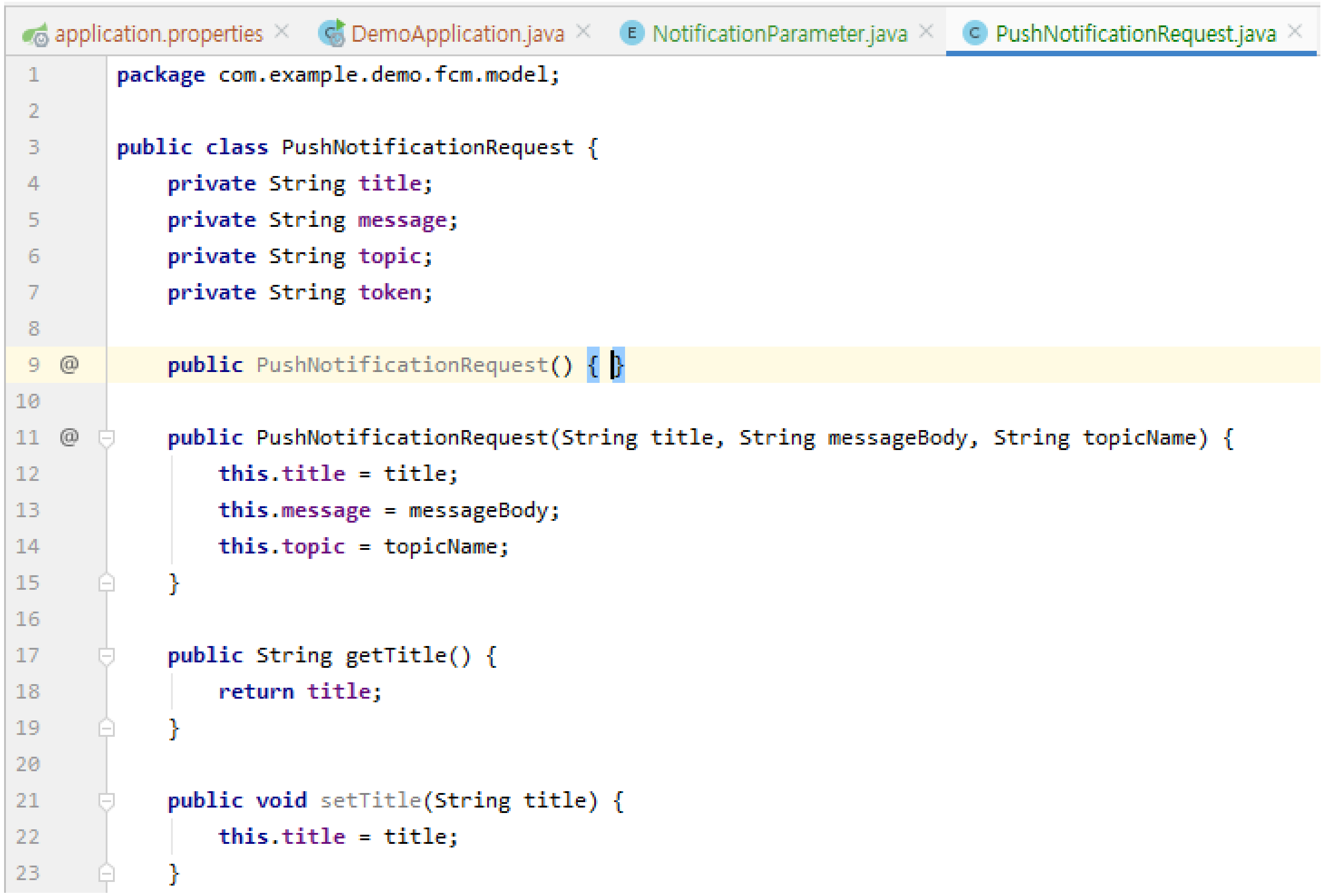
```java
    public void sendMessageWithoutData(PushNotificationRequest request)
            throws InterruptedException, ExecutionException {
        Message message = getPreconfiguredMessageWithoutData(request);
        String response = sendAndGetResponse(message);
        logger.info("Sent message without data. Topic: " + request.getTopic() + ", " + response);
    }

    public void sendMessageToToken(PushNotificationRequest request)
            throws InterruptedException, ExecutionException {
        Message message = getPreconfiguredMessageToToken(request);
        String response = sendAndGetResponse(message);
        logger.info("Sent message to token. Device token: " + request.getToken() + ", " + response);
    }

    private String sendAndGetResponse(Message message) throws InterruptedException, ExecutionException {
        return FirebaseMessaging.getInstance().sendAsync(message).get();
    }

    private AndroidConfig getAndroidConfig(String topic) {
        return AndroidConfig.builder()
                .setTtl(Duration.ofMinutes(2).toMillis()).setCollapseKey(topic)
                .setPriority(AndroidConfig.Priority.HIGH)
                .setNotification(AndroidNotification.builder().setSound(NotificationParameter.SOUND.getValue())
                        .setColor(NotificationParameter.COLOR.getValue()).setTag(topic).build()).build();
    }
```



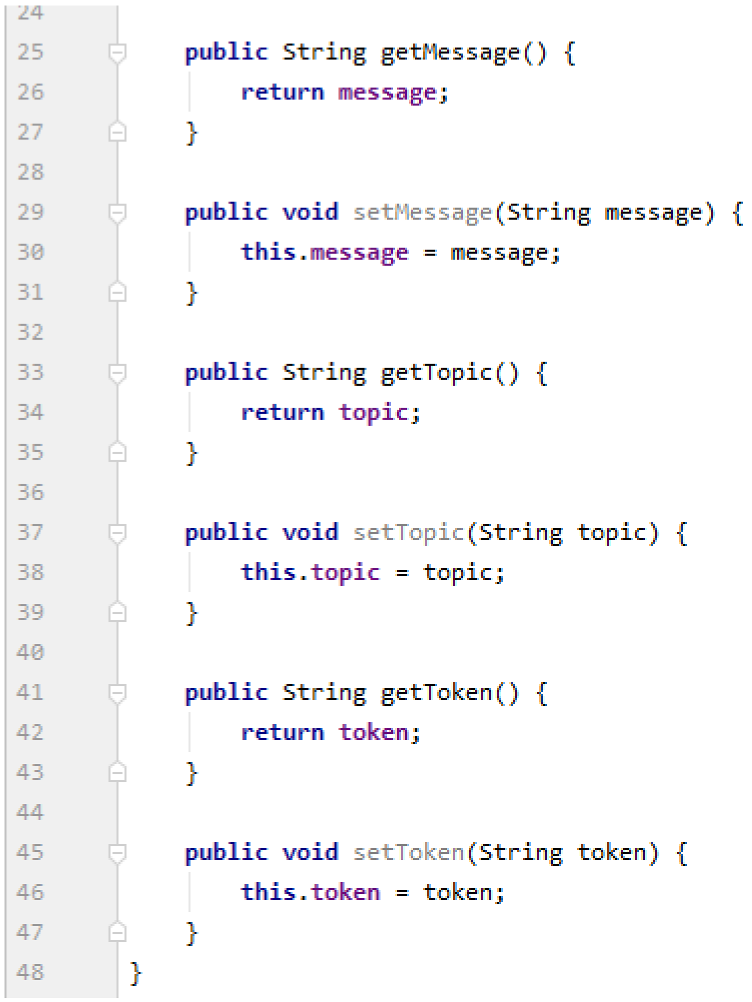
```java
    private ApnsConfig getApnsConfig(String topic) {
        return ApnsConfig.builder()
                .setAps(Aps.builder().setCategory(topic).setThreadId(topic).build()).build();
    }

    private Message getPreconfiguredMessageToToken(PushNotificationRequest request) {
        return getPreconfiguredMessageBuilder(request).setToken(request.getToken())
                .build();
    }

    private Message getPreconfiguredMessageWithoutData(PushNotificationRequest request) {
        return getPreconfiguredMessageBuilder(request).setTopic(request.getTopic())
                .build();
    }

    private Message getPreconfiguredMessageWithData(Map<String, String> data, PushNotificationRequest request) {
        return getPreconfiguredMessageBuilder(request).putAllData(data).setTopic(request.getTopic())
                .build();
    }

    private Message.Builder getPreconfiguredMessageBuilder(PushNotificationRequest request) {
        AndroidConfig androidConfig = getAndroidConfig(request.getTopic());
        ApnsConfig apnsConfig = getApnsConfig(request.getTopic());
        return Message.builder()
                .setApnsConfig(apnsConfig).setAndroidConfig(androidConfig).setNotification(
                        new Notification(request.getTitle(), request.getMessage()));
    }
}
```



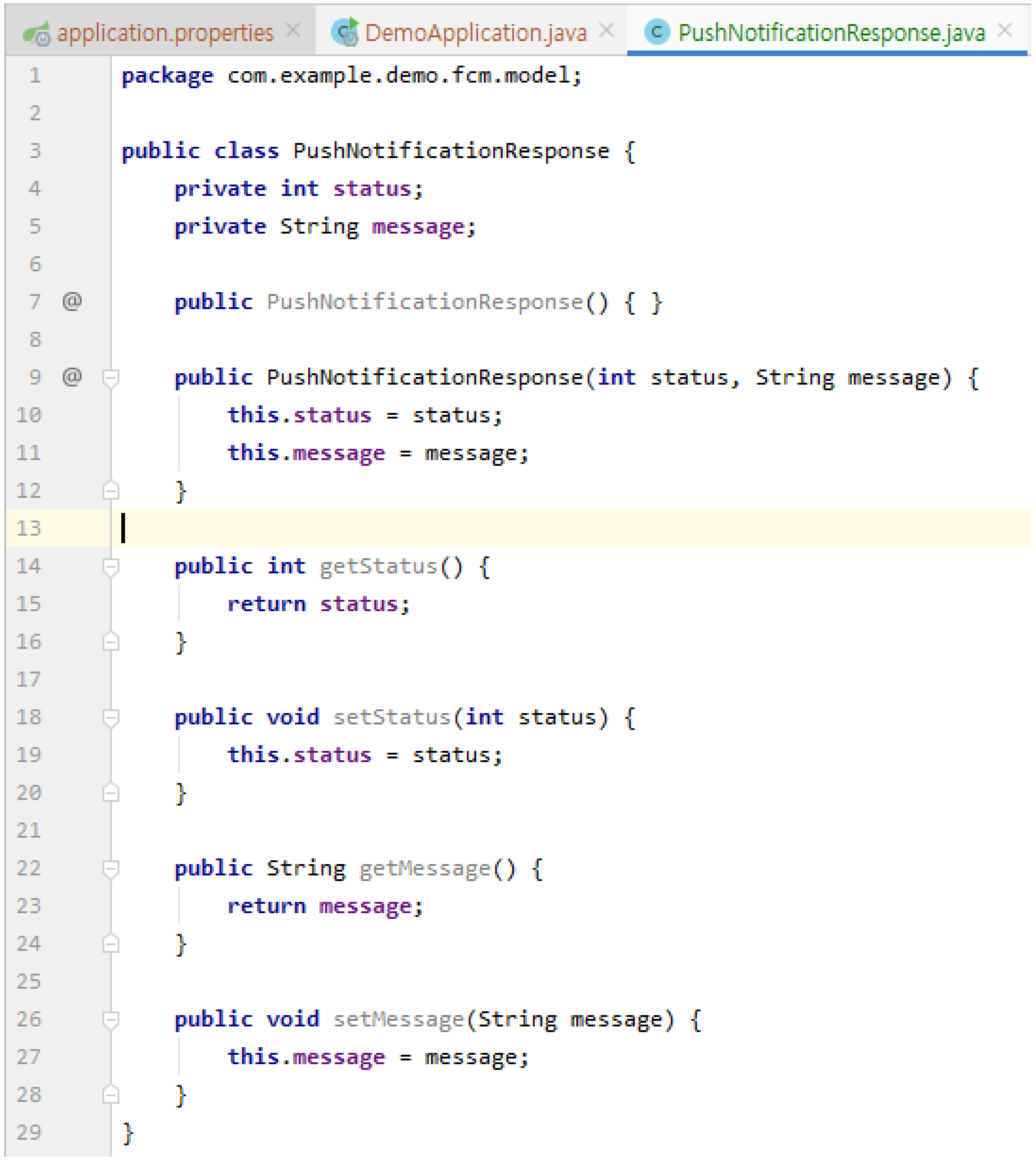
```java
package com.example.demo.fcm.firebase;

public enum NotificationParameter {
    SOUND( value: "default"),
    COLOR( value: "#FFFF00");

    private String value;

    NotificationParameter(String value) {
        this.value = value;
    }

    public String getValue() {
        return this.value;
    }
}
```



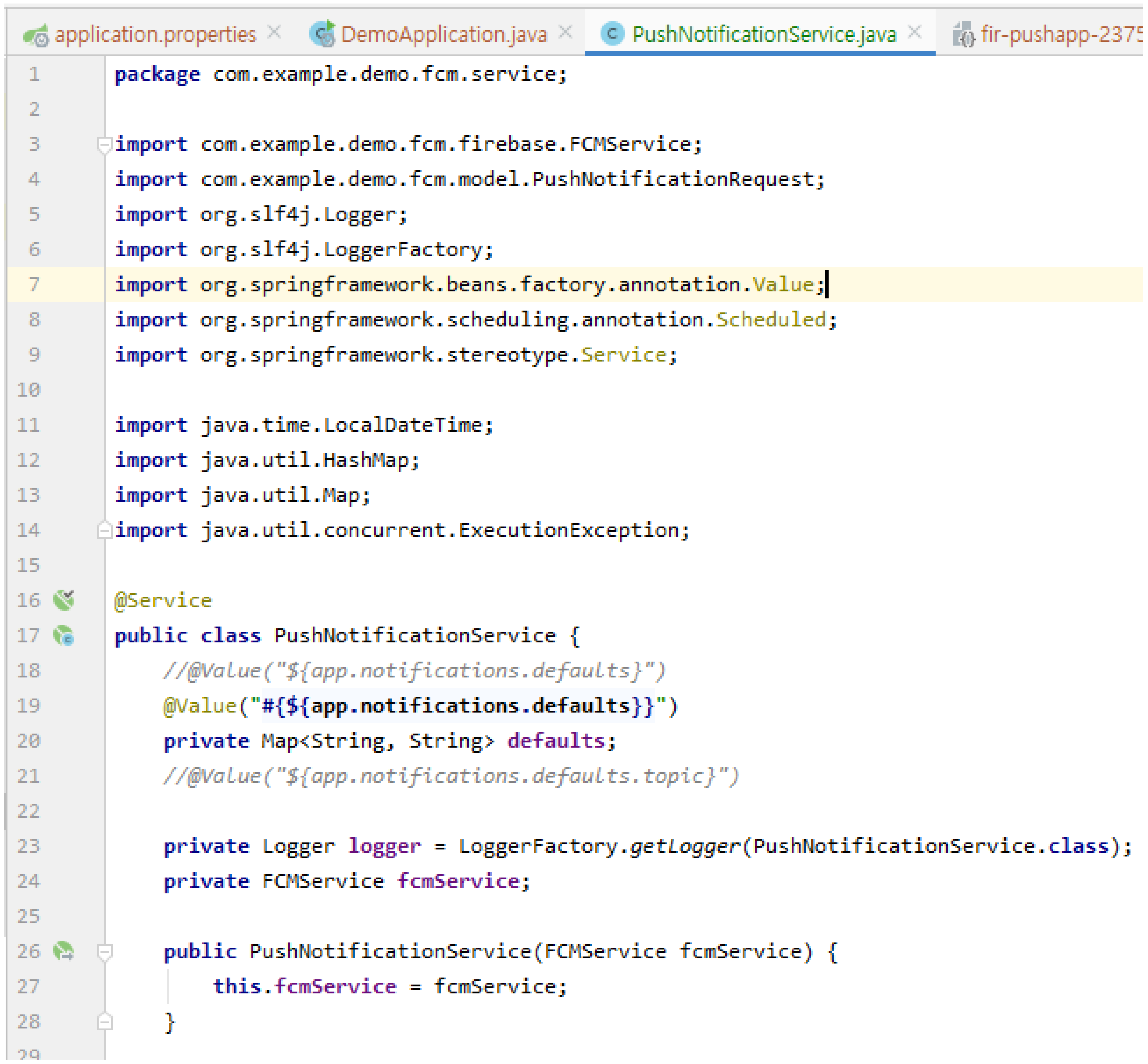
```java
package com.example.demo.fcm.model;

public class PushNotificationRequest {
    private String title;
    private String message;
    private String topic;
    private String token;

    public PushNotificationRequest() { }

    public PushNotificationRequest(String title, String messageBody, String topicName) {
        this.title = title;
        this.message = messageBody;
        this.topic = topicName;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }
```



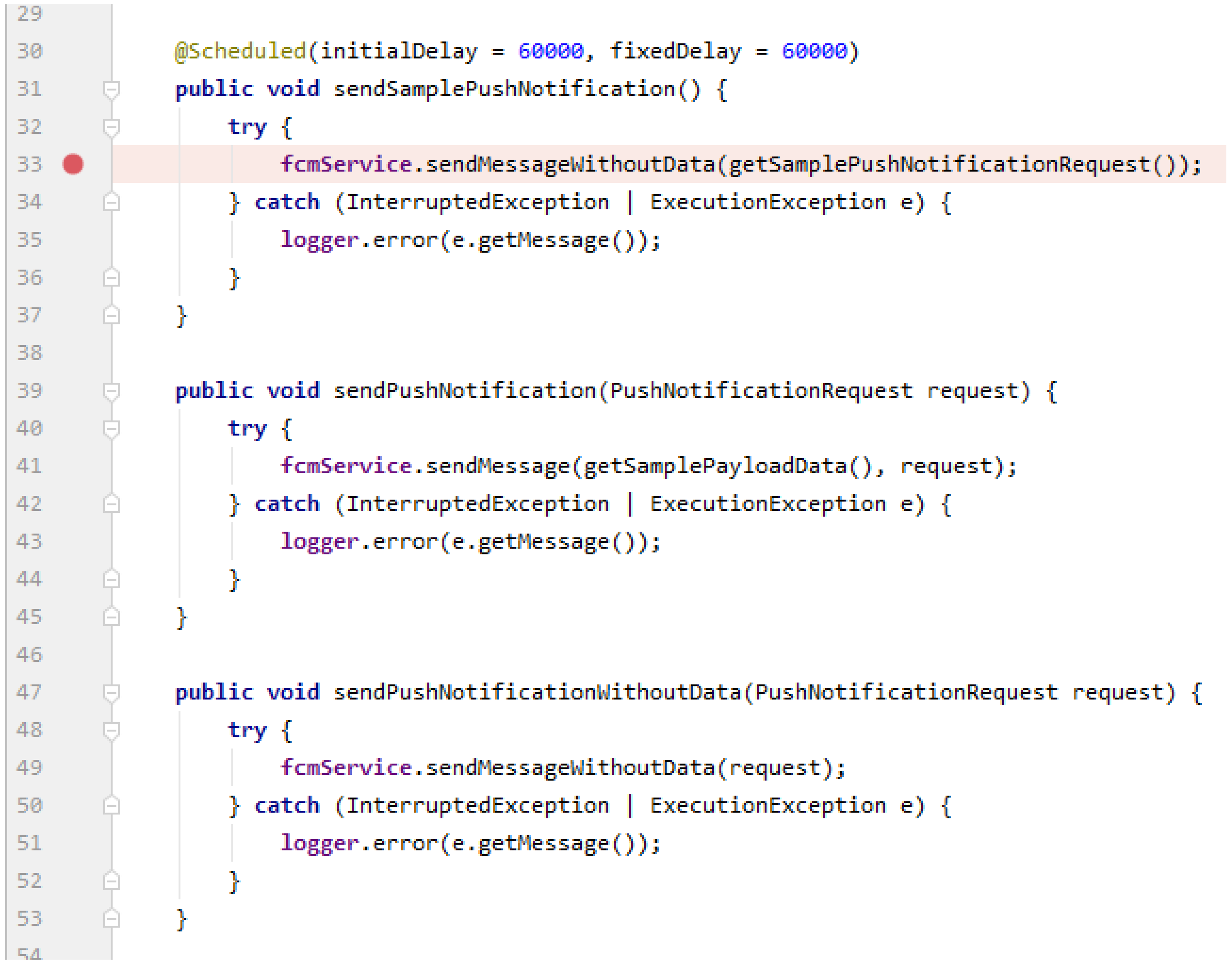
```java

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public String getTopic() {
        return topic;
    }

    public void setTopic(String topic) {
        this.topic = topic;
    }

    public String getToken() {
        return token;
    }

    public void setToken(String token) {
        this.token = token;
    }
}
```



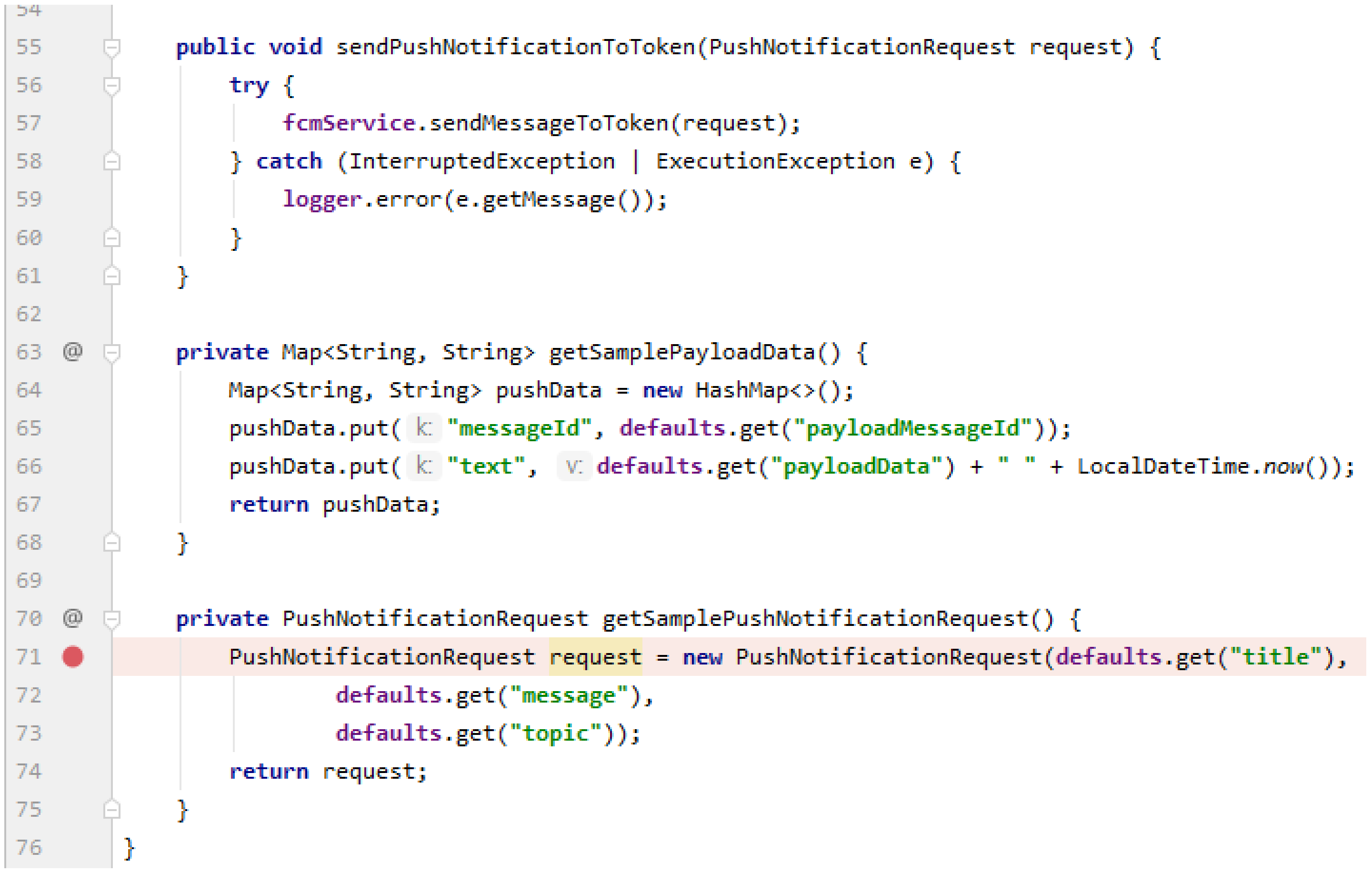
```java
package com.example.demo.fcm.model;

public class PushNotificationResponse {
    private int status;
    private String message;

    public PushNotificationResponse() { }

    public PushNotificationResponse(int status, String message) {
        this.status = status;
        this.message = message;
    }

    public int getStatus() {
        return status;
    }

    public void setStatus(int status) {
        this.status = status;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```


application.properties × DemoApplication.java × PushNotificationService.java × fir-pushapp-2375

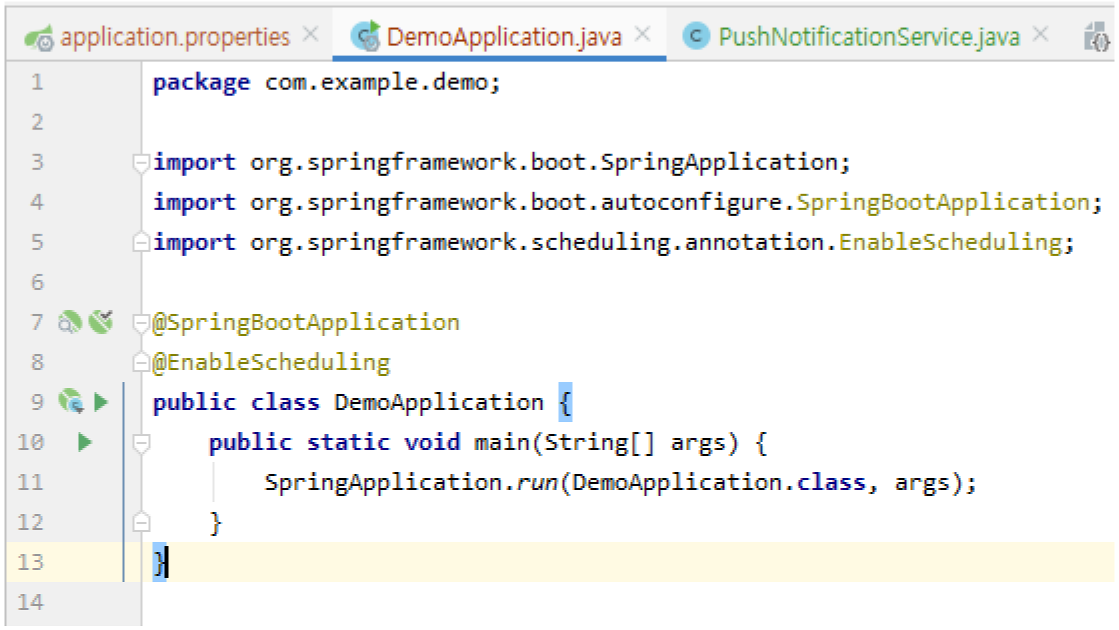
```java
package com.example.demo.fcm.service;

import com.example.demo.fcm.firebase.FCMService;
import com.example.demo.fcm.model.PushNotificationRequest;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Service;

import java.time.LocalDateTime;
import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.ExecutionException;

@Service
public class PushNotificationService {
    //@Value("${app.notifications.defaults}")
    @Value("#{${app.notifications.defaults}}")
    private Map<String, String> defaults;
    //@Value("${app.notifications.defaults.topic}")

    private Logger logger = LoggerFactory.getLogger(PushNotificationService.class);
    private FCMService fcmService;

    public PushNotificationService(FCMService fcmService) {
        this.fcmService = fcmService;
    }

```



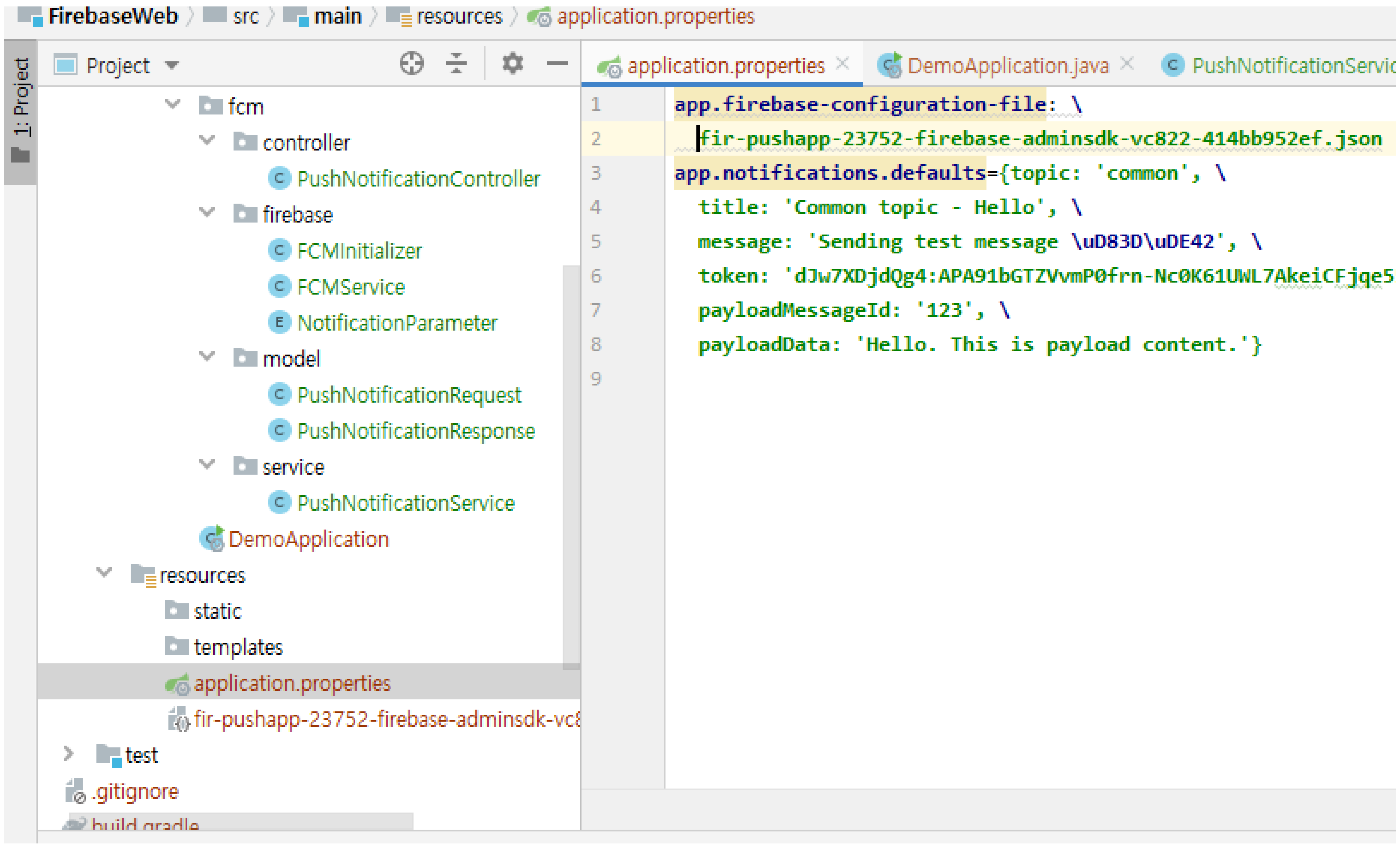
```java

    @Scheduled(initialDelay = 60000, fixedDelay = 60000)
    public void sendSamplePushNotification() {
        try {
            fcmService.sendMessageWithoutData(getSamplePushNotificationRequest());
        } catch (InterruptedException | ExecutionException e) {
            logger.error(e.getMessage());
        }
    }

    public void sendPushNotification(PushNotificationRequest request) {
        try {
            fcmService.sendMessage(getSamplePayloadData(), request);
        } catch (InterruptedException | ExecutionException e) {
            logger.error(e.getMessage());
        }
    }

    public void sendPushNotificationWithoutData(PushNotificationRequest request) {
        try {
            fcmService.sendMessageWithoutData(request);
        } catch (InterruptedException | ExecutionException e) {
            logger.error(e.getMessage());
        }
    }

```



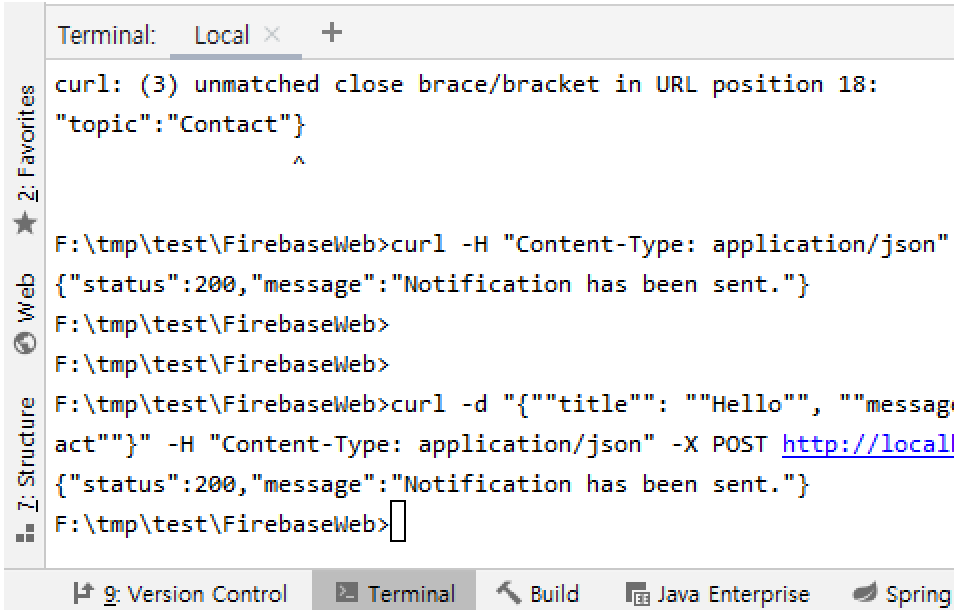
```java
    public void sendPushNotificationToToken(PushNotificationRequest request) {
        try {
            fcmService.sendMessageToToken(request);
        } catch (InterruptedException | ExecutionException e) {
            logger.error(e.getMessage());
        }
    }

    private Map<String, String> getSamplePayloadData() {
        Map<String, String> pushData = new HashMap<>();
        pushData.put( k: "messageId", defaults.get("payloadMessageId"));
        pushData.put( k: "text",  v: defaults.get("payloadData") + " " + LocalDateTime.now());
        return pushData;
    }

    private PushNotificationRequest getSamplePushNotificationRequest() {
        PushNotificationRequest request = new PushNotificationRequest(defaults.get("title"),
                defaults.get("message"),
                defaults.get("topic"));
        return request;
    }
}
```



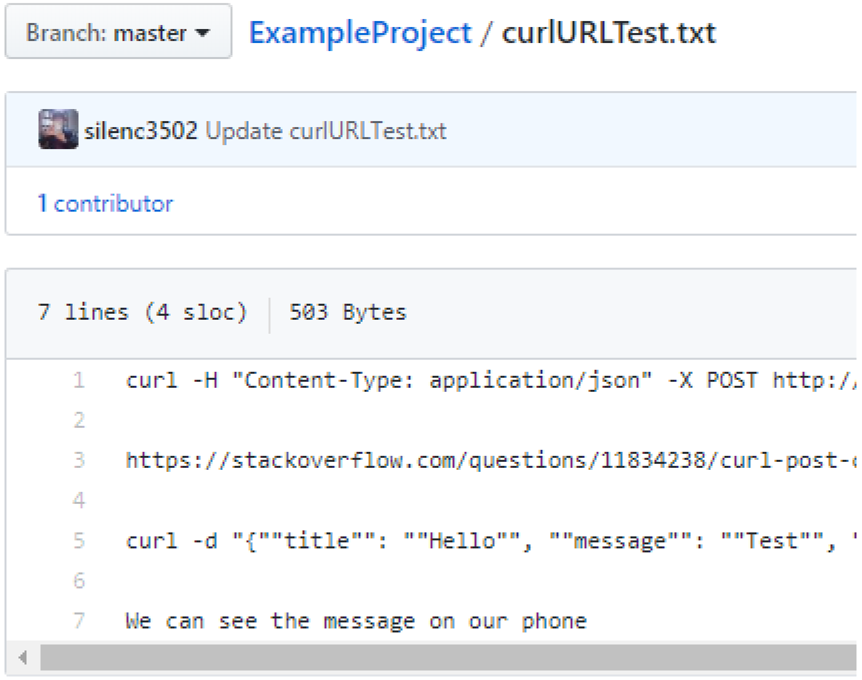
```java
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication
@EnableScheduling
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}

```



```
app.firebase-configuration-file: \
  fir-pushapp-23752-firebase-adminsdk-vc822-414bb952ef.json
app.notifications.defaults={topic: 'common', \
  title: 'Common topic - Hello', \
  message: 'Sending test message \uD83D\uDE42', \
  token: 'dJw7XDjdQg4:APA91bGTZVvmP0frn-Nc0K61UWL7AkeiCFjqe5
  payloadMessageId: '123', \
  payloadData: 'Hello. This is payload content.'}

```



```
curl: (3) unmatched close brace/bracket in URL position 18:
"topic":"Contact"}
                 ^


F:\tmp\test\FirebaseWeb>curl -H "Content-Type: application/json"
{"status":200,"message":"Notification has been sent."}
F:\tmp\test\FirebaseWeb>
F:\tmp\test\FirebaseWeb>
F:\tmp\test\FirebaseWeb>curl -d "{""title"": ""Hello"", ""messag
act""}" -H "Content-Type: application/json" -X POST http://locall
{"status":200,"message":"Notification has been sent."}
F:\tmp\test\FirebaseWeb>
```



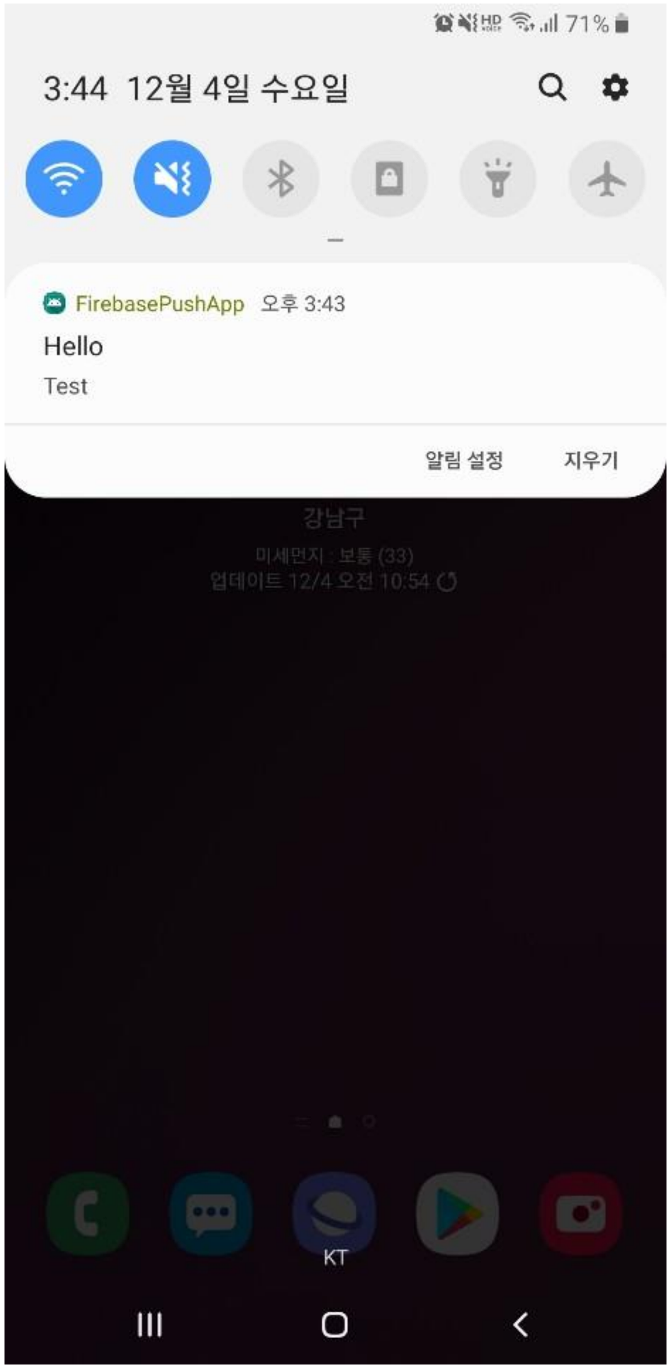
Branch: master ▾ ExampleProject / curlURLTest.txt

silenc3502 Update curlURLTest.txt

1 contributor

7 lines (4 sloc) | 503 Bytes

```
curl -H "Content-Type: application/json" -X POST http:/

https://stackoverflow.com/questions/11834238/curl-post-

curl -d "{""title"": ""Hello"", ""message"": ""Test"", '

We can see the message on our phone
```



3:44 12월 4일 수요일

FirebasePushApp 오후 3:43

Hello

Test