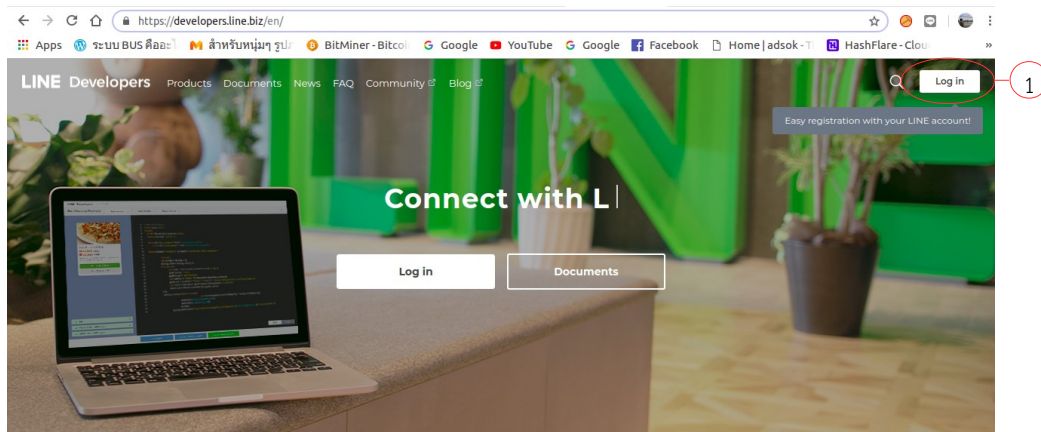


Line Bot

1. การสร้าง LINE Bot

1.1 การสร้าง LINE Bot

การที่จะได้ LINE Bot จำเป็นจะต้องมี ID LINE จากนั้นไปที่เว็บ <https://developers.line.biz/en/> ทำการ Login เพื่อสร้าง LINE Bot

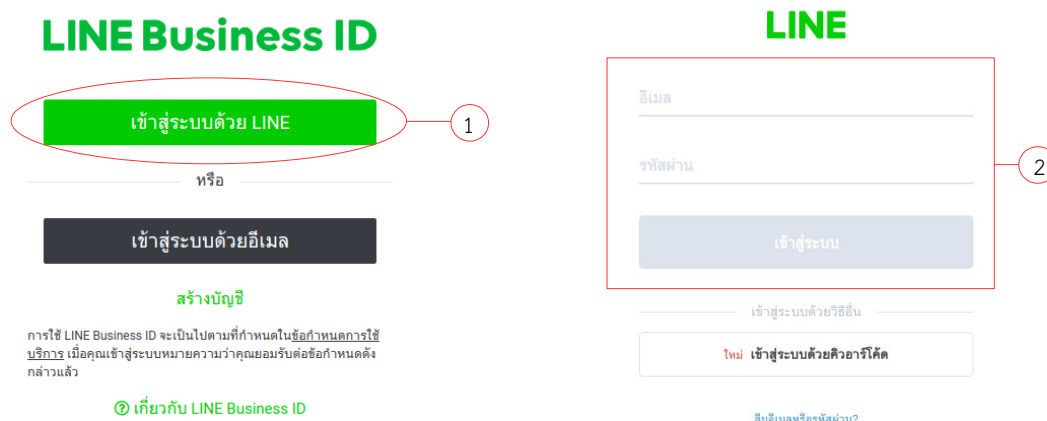


Products

รูปที่ 1.1 แสดงแผนภาพ <https://developers.line.biz/en/>

1.2 เข้าสู่ระบบด้วย ID LINE

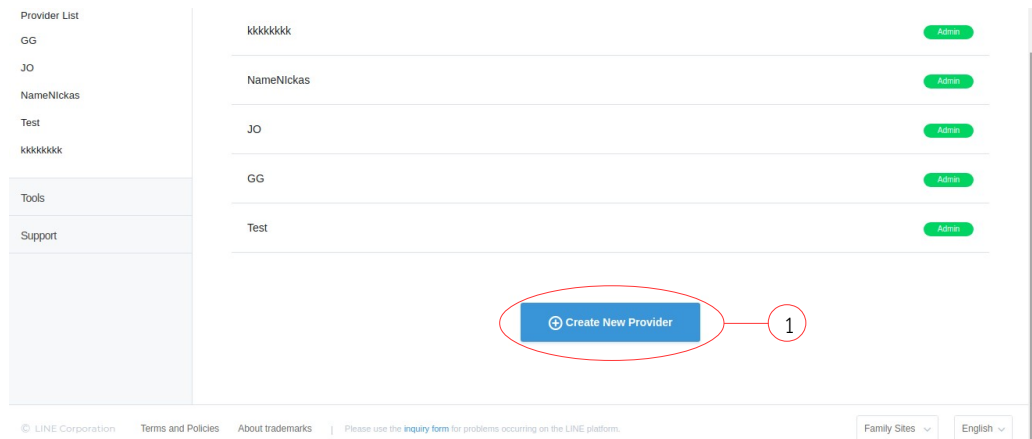
ทำการ Login โดยใช้ ID LINE เพื่อสร้าง LINE Bot



รูปที่ 1.2 แสดงแผนภาพการเข้าสู่ระบบด้วย ID LINE

1.3 ทำการสร้าง LINE Bot

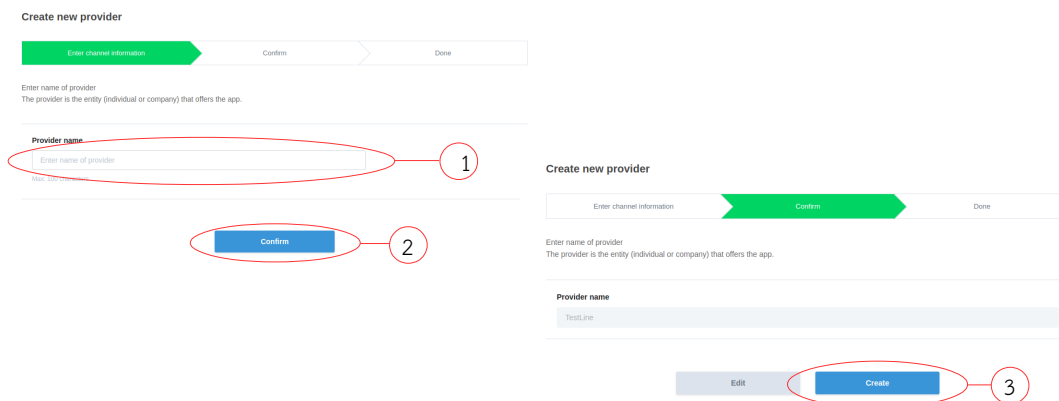
กดที่ Create New Provider สร้างผู้ให้บริการ



รูปที่ 1.3 แสดงแผนภาพ สร้าง LINE Bot

1.4 ทำการสร้าง LINE Bot

ตั้งชื่อผู้ให้บริการ จากนั้นกดที่ Confirm จะให้ดูอีกครั้งเพื่อยืนยันความถูกต้อง จากนั้นกด Create



รูปที่ 1.4 แสดงแผนภาพ สร้าง LINE Bot

1.5 ทำการสร้าง LINE Bot

เมื่อสร้างผู้ให้บริการเสร็จสิ้น จะได้บริการ LINE Login, Messaging API และ Clova Skill ซึ่งที่ใช้ทำ LINE Bot คือ Messaging API จากนั้นเลือกประเภท และ กรอก Email กด Confirm

TestLine

A provider has been created!
Next, let's create a new channel

Please select a channel from the following and create a new channel.

LINE Login
Create Channel

Messaging API
Create Channel

Clova Skill
Create Channel

Plan
Developer Trial
A trial plan which lets you create a bot that can send push messages and have up to 50 friends.
Note: You cannot upgrade or buy a premium ID for a Developer Trial plan.

Category
เว็บไซต์บล็อก

Subcategory
เว็บไซต์บล็อก(อื่นๆ)

Email address
example@line.me
Max: 100 characters

Previous page

Confirm

รูปที่ 1.5 แสดงแผนภาพ สร้าง LINE Bot

1.6 ทำการสร้าง LINE Bot

เมื่อกด Confirm แล้วให้ติ๊กข้อตกลงทั้ง 2 จากนั้นกดไปที่ Create

Category
เว็บไซต์บล็อก

Subcategory
เว็บไซต์บล็อก(อื่นๆ)

Email address
nictheripper@gmail.com

1

☒ LINE® Terms of Use ⓘ I have read and agree to the Terms of Use.

☒ Messaging API (Developer Trial plan) Terms of Use: ⓘ I have read and agree to the Terms of Use.

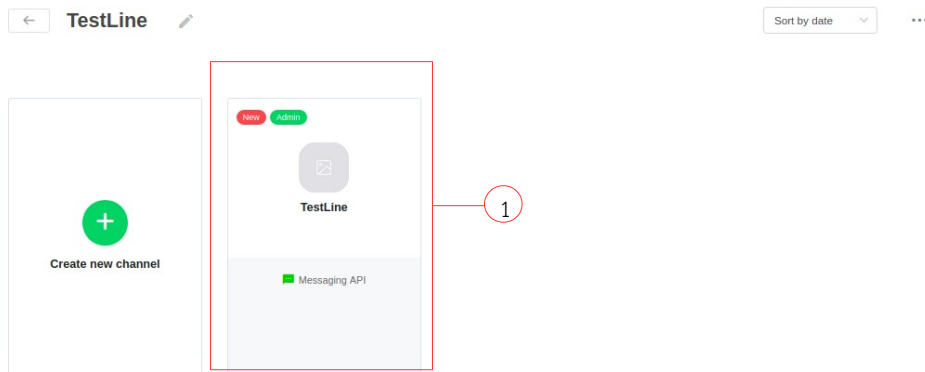
Edit

Create

2

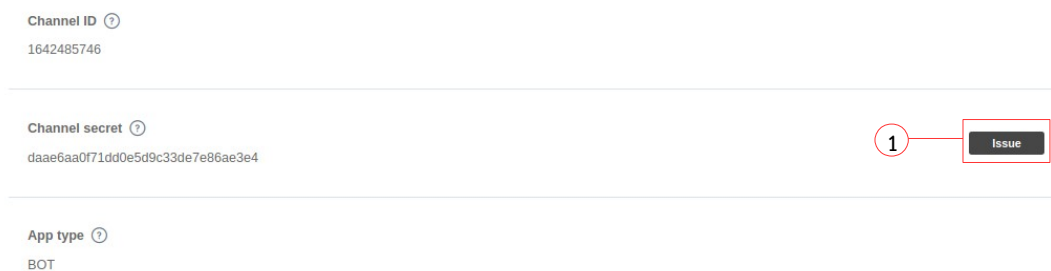
รูปที่ 1.6 แสดงแผนภาพ สร้าง LINE Bot

1.7 เสร็จสิ้นการสร้าง LINE Bot จากนั้นจะได้ Messaging API

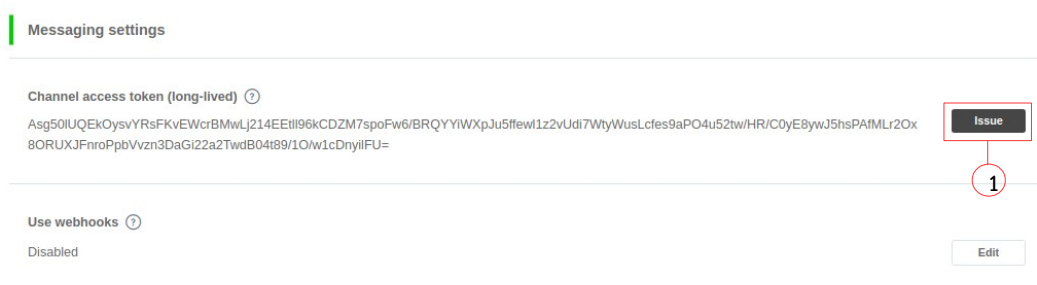


รูปที่ 1.7 แสดงแผนภาพ สร้าง LINE Bot

1.8 การขอ Channel secret และ Channel access token



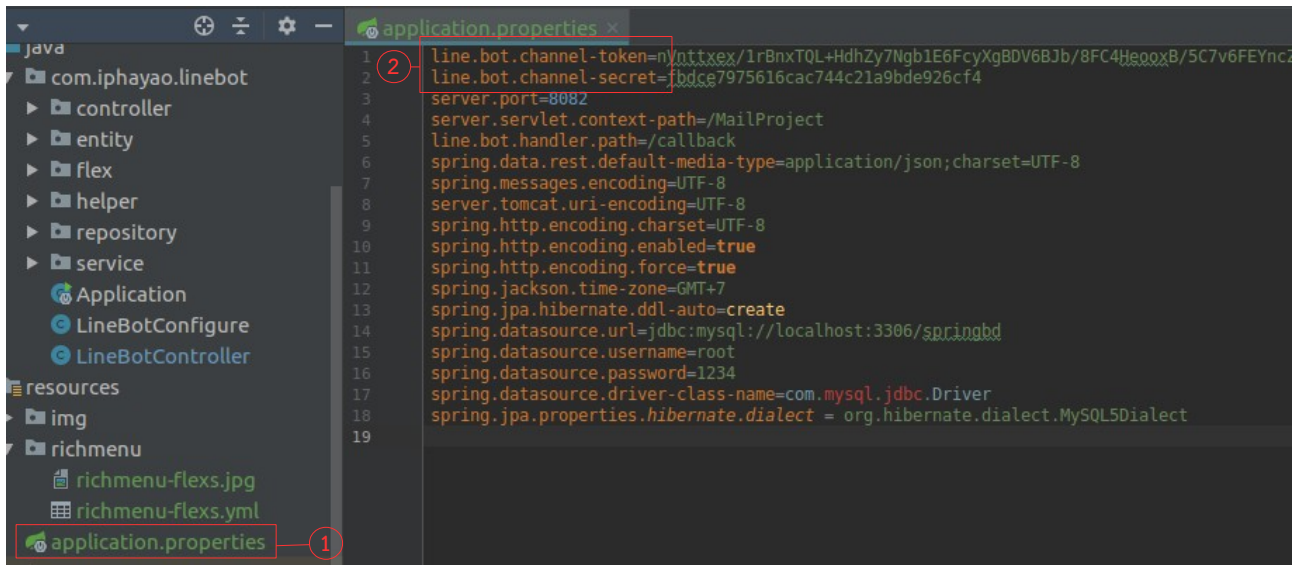
รูปที่ 1.8 แสดงแผนภาพ ขอ Channel secret



รูปที่ 1.9 แสดงแผนภาพ ขอ Channel access token

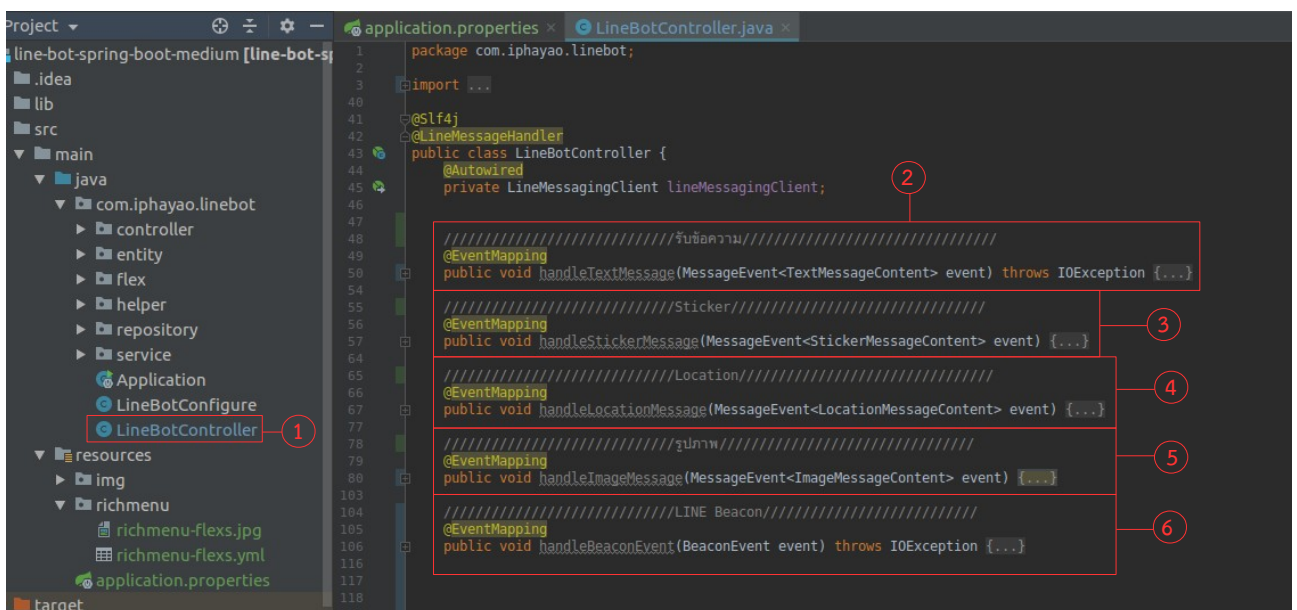
2. โหลดโปรแกรม git clone <https://github.com/epsonthenic/JAVA-LINEBot.git>

2.1 เมื่อได้ Channel secret และ Channel access token นำมาใส่ในโปรแกรม



รูปที่ 2.1 แสดงแผนภาพ ตั้งค่าใน Application.properties

2.2 @EventManager ของ LINE Bot จะมีดังนี้



รูปที่ 2.2 แสดงแผนภาพ Class LineBotController

2.2.1 เหตุการณ์ของข้อความ

2.2.2 เหตุการณ์ของสติ๊กเกอร์ (Sticker)

2.2.3 เหตุการณ์ของที่ตั้ง (Location)

2.2.4 เหตุการณ์ของรูปภาพ

2.2.5 เหตุการณ์ของ Beacon

2.3 เหตุการณ์ของข้อความ

```
////////////////////////////////////////รับข้อความ////////////////////////////////////////
@EventListener
public void handleMessage(MessageEvent<TextMessageContent> event) throws IOException {
    log.info(event.toString());
    handleTextContent(event.getReplyToken(), event, event.getMessage());
}
```

รูปที่ 2.3 แสดงแผนภาพ เหตุการณ์ของข้อความ

```
121
122 @ 1 private void handleTextContent(String replyToken, Event event, TextMessageContent content) throws IOException {
123     String text = content.getText();
124     String userId = event.getSource().getUserId();
125
126     switch (text) {
127         case "Flex": {
128             String pathImageFlex = new ClassPathResource("richmenu/richmenu-flexs.jpg").getFile().getAbsolutePath();
129             String pathConfigFlex = new ClassPathResource("richmenu/richmenu-flexs.yml").getFile().getAbsolutePath();
130             RichMenuHelper.createRichMenu(lineMessagingClient, pathConfigFlex, pathImageFlex, userId);
131             break;
132         }
133         case "Flex Back": {
134             RichMenuHelper.deleteRichMenu(lineMessagingClient, userId);
135             break;
136         }
137         case "Flex Restaurant": {
138             this.reply(replyToken, new RestaurantFlexMessageSupplier().get());
139             break;
140         }
141         case "Flex Menu": {
142             this.reply(replyToken, new RestaurantMenuFlexMessageSupplier().get());
143             break;
144         }
145         case "Flex Receipt": {
146             this.reply(replyToken, new ReceiptFlexMessageSupplier().get());
147             break;
148         }
149         case "Flex News": {
150             this.reply(replyToken, new NewsFlexMessageSupplier().get());
151             break;
152         }
153         case "Flex Ticket": {
154             this.reply(replyToken, new TicketFlexMessageSupplier().get());
155             break;
156         }
157     }
}
```

รูปที่ 2.4 แสดงแผนภาพ แบบ switch case

2.3.1 get ข้อความที่ได้จาก LINE เก็บไว้ที่ text และ get ID LINE มาเก็บไว้ที่ userId

2.3.2 เป็นเงื่อนไข ถ้า text เท่ากับค่าใน case ใดให้ทำอันนั้น เช่น ถ้า text เท่ากับ Flex Menu

จะแสดง Menu กลับไปหาผู้ใช้ LINE

2.4 เพื่อให้ครอบคลุมข้อความมากขึ้นโดยจะเช็คจากข้างในข้อความ

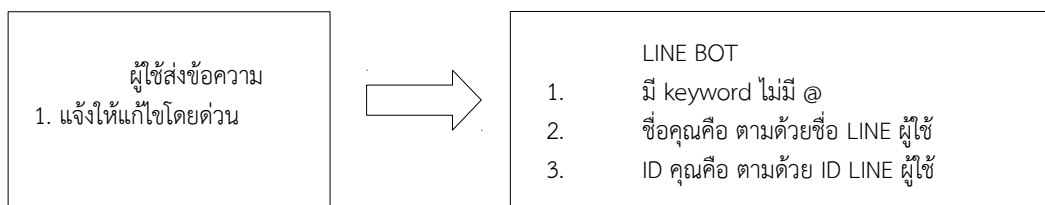
```
161 default:
162     String pathImageFlex = new ClassPathResource("richmenu/richmenu-flexs.jpg").getFile().getAbsolutePath();
163     String pathConfigFlex = new ClassPathResource("richmenu/richmenu-flexs.yml").getFile().getAbsolutePath();
164     RichMenuHelper.createRichMenu(lineMessagingClient, pathConfigFlex, pathImageFlex, userId);
165     boolean hasText = text.contains("@");
166     boolean hasText3 = text.contains("ขอเข้ากลุ่ม");
167
168     ① if (hasText3 == true) {
169         ② reply(replyToken, Arrays.asList(
170             new TextMessage("https://line.me/R/ti/g/vYYHUCuMG_")
171         ));
172     } else if ((AppMailServiceImp.checkTextMatches(text) == true) && (hasText == false)) {
173         ③ if (userId != null) {
174             lineMessagingClient.getProfile(userId)
175                 .whenComplete((profile, throwable) -> {
176                     if (throwable != null) {
177                         this.replyText(replyToken, throwable.getMessage());
178                         return;
179                     }
180                     this.reply(replyToken, Arrays.asList(
181                         new TextMessage("มี keyword ไม่มี @"),
182                         new TextMessage("ชื่อคุณคือ: " + profile.getDisplayName()),
183                         new TextMessage("ID คุณคือ: " + profile.getUserId())
184                     ));
185                 });
186         }
187     } else if ((hasText == true) && (AppMailServiceImp.checkTextMatches(text) == true)) {
188         ④ if (userId != null) {
189             reply(replyToken, Arrays.asList(
190                 new TextMessage("มี keyword และมี @"),
191                 new TextMessage(text)
192             ));
193         }
194     }
195 }
```

รูปที่ 2.4 แสดงแผนภาพ แบบ switch case

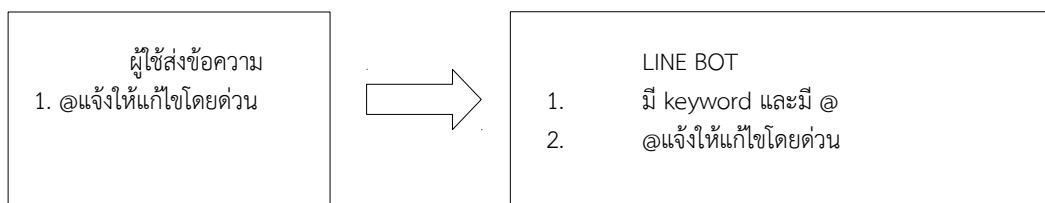
2.4.1 เรียกใช้ Rich Menu และ กำหนดข้อความว่ามี @ อยู่ใน text มั้ยถ้ามี hasText จะเท่ากับ true แต่ถ้าไม่มี จะเท่ากับ false และ กำหนดข้อความว่ามี ขอเข้ากลุ่ม อยู่ใน text มั้ยถ้ามี hasText3 จะเท่ากับ true แต่ถ้าไม่มี จะเท่ากับ false

2.4.2 เป็นเงื่อนไข ถ้าใน text มีคำว่าขอเข้ากลุ่ม ให้ LINE Bot แสดงกลุ่มออกไปให้กับผู้ใช้

2.4.3 เป็นเงื่อนไข ถ้าใน text มีคำที่มีใน Keyword และไม่มี @ ให้ LINE Bot แสดงข้อความดังนี้

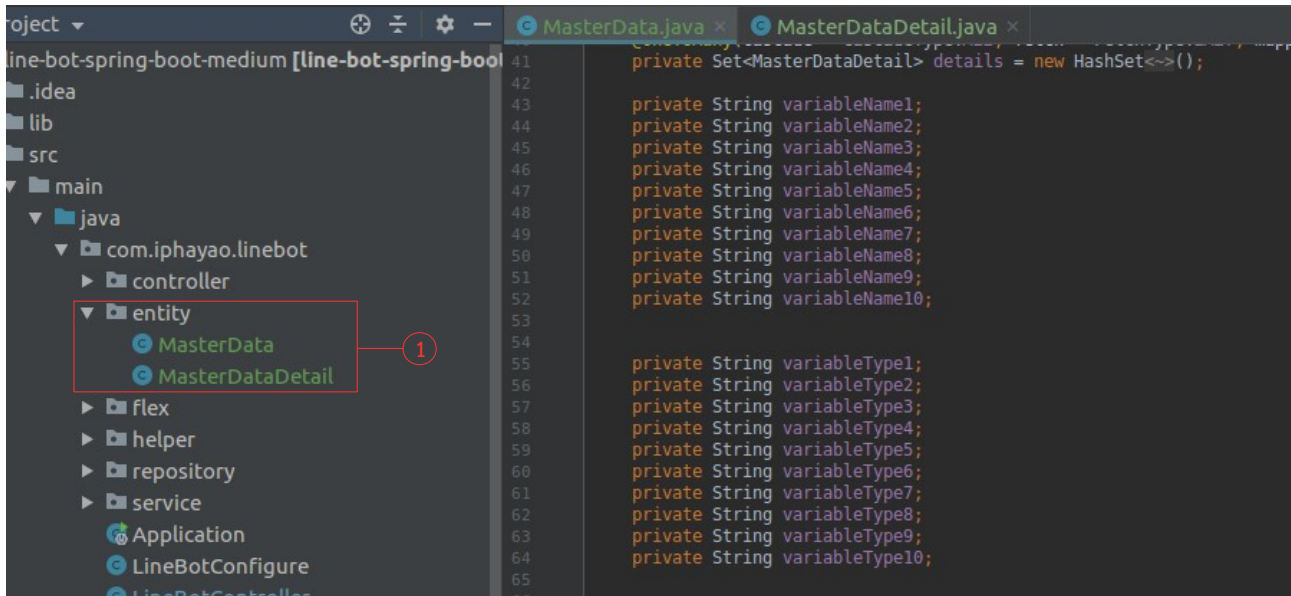


2.4.4 เป็นเงื่อนไข ถ้า text มีคำที่มีใน Keyword และมี @ ให้ LINE Bot แสดงข้อความดังนี้



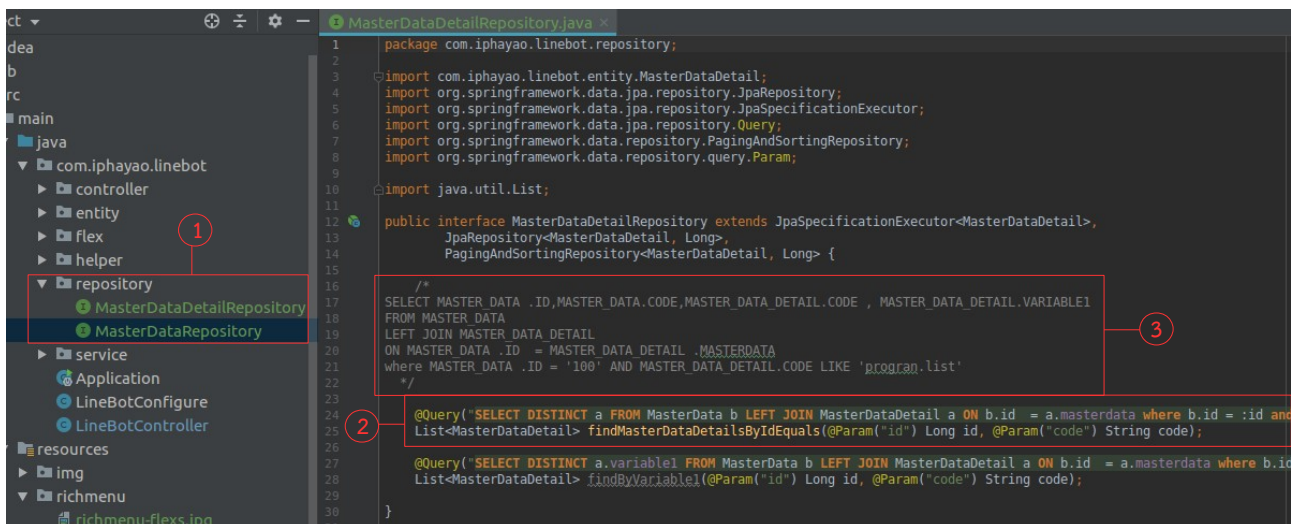
2.5 การทำ Keyword

สร้างตาราง MasterData และ MasterDataDetail แบบ OneToMany



รูปที่ 2.5 แสดงแผนภาพ MasterData และ MasterDataDetail แบบ OneToMany

2.6 สร้าง Repository ของ MasterData และ MasterDataDetail โดย MasterDataDetail ทำฟังก์ชัน Query Keyword



รูปที่ 2.6 แสดงแผนภาพ MasterData และ MasterDataDetail แบบ OneToMany

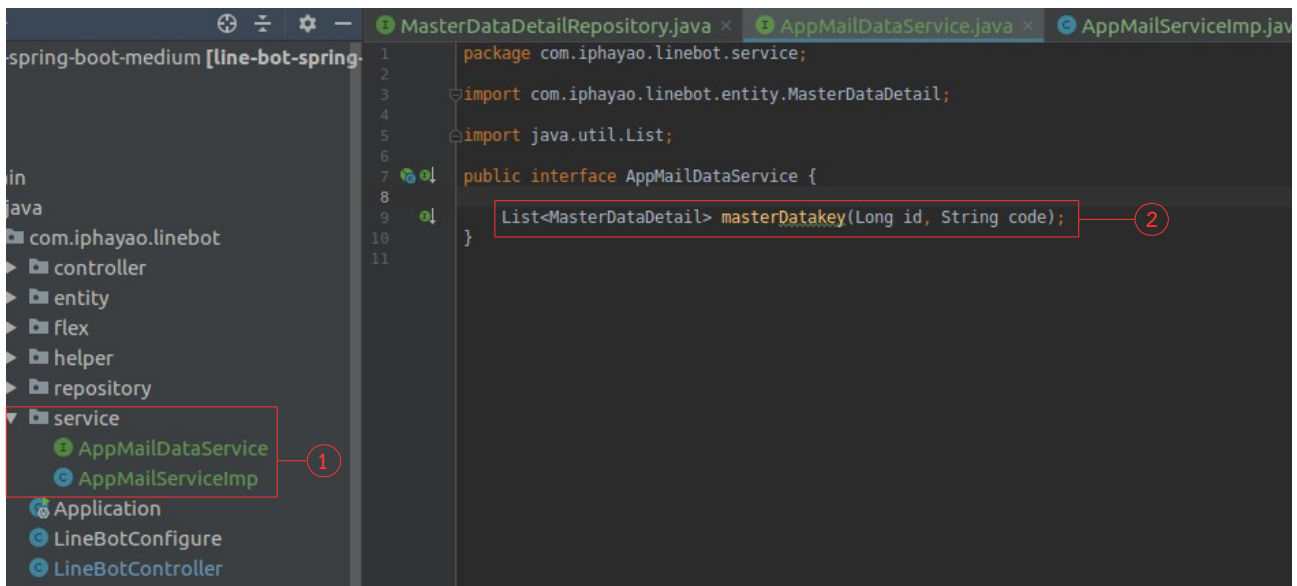
2.6.1 สร้าง interface ของ MasterData และ MasterDataDetail

2.6.2 Query หาค่า Keyword Methods ชื่อ findMasterDataDetailsByIdEquals

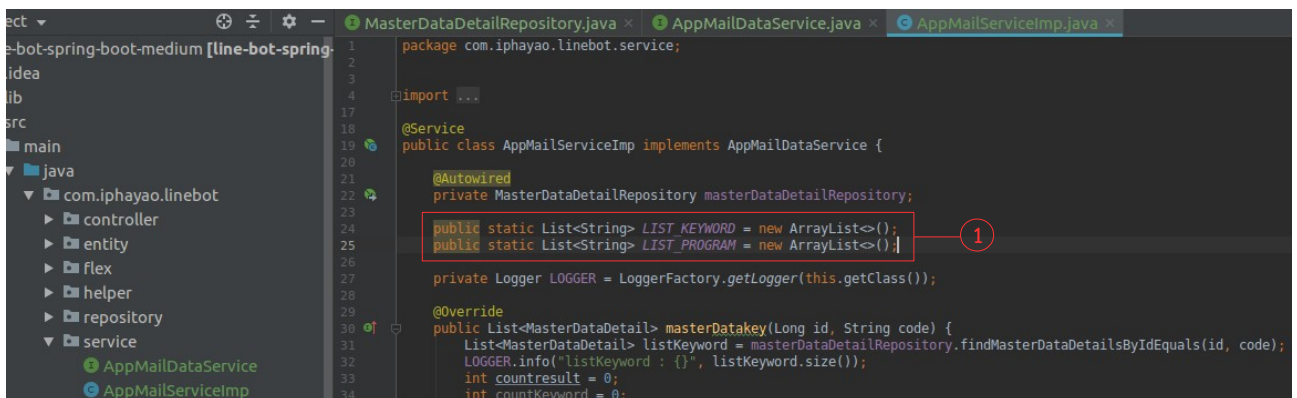
2.6.3 เป็นแนวทางในการ Query

2.7 สร้าง Service

สร้าง Methods ชื่อ masterDatakey โดยรับค่า id และ code



รูปที่ 2.7 แสดงแผนภาพ AppMailDataService



รูปที่ 2.8 แสดงแผนภาพ AppMailServiceImpl

2.7.1 สร้าง ArrayList ชื่อ LIST_KEYWORD และ LIST_PROGRAM เพื่อจะนำค่า Keyword ในฐานข้อมูล มาเก็บ

2.8 เรียกใช้ masterDatakey เพื่อรับค่าจาก Controller

```
29  
30 1  
31 public List<MasterDataDetail> masterDatakey(Long id, String code) {  
32     List<MasterDataDetail> listKeyword = masterDataDetailRepository.findMasterDataDetailsByIdEquals(id, code);  
33     LOGGER.info("listKeyword : {}", listKeyword.size());  
34     String resultKeyword = "";  
35     List<String> keywordSplitList = null;  
36     resultKeyword = listKeyword.get(0).getVariable1();  
37     LOGGER.info("num : {}", resultKeyword);  
38     switch (code) {  
39         case "program.list": {  
40             LIST_PROGRAM = null;  
41             LIST_PROGRAM = new ArrayList<>();  
42             keywordSplitList = Arrays.asList(resultKeyword.split( regex: "\\s*,\\s*"));  
43             LOGGER.info("program.list : {}", keywordSplitList);  
44             for (String program : keywordSplitList) {  
45                 LIST_PROGRAM.add(program);  
46             }  
47             LOGGER.info("program : {}", LIST_PROGRAM);  
48         }  
49         break;  
50         case "keyword.list": {  
51             LIST_KEYWORD = null;  
52             LIST_KEYWORD = new ArrayList<>();  
53             keywordSplitList = Arrays.asList(resultKeyword.split( regex: "\\s*,\\s*"));  
54             LOGGER.info("keyword.list : {}", keywordSplitList);  
55             for (String keyword : keywordSplitList) {  
56                 LIST_KEYWORD.add(keyword);  
57             }  
58             LOGGER.info("keyword : {}", LIST_KEYWORD);  
59         }  
60         break;  
61         default:  
62             break;  
63     }  
64     LOGGER.info("keyword : {}", LIST_KEYWORD);  
65     LOGGER.info("program : {}", LIST_PROGRAM);  
}
```

รูปที่ 2.9 แสดงแผนภาพ AppMailServiceImp

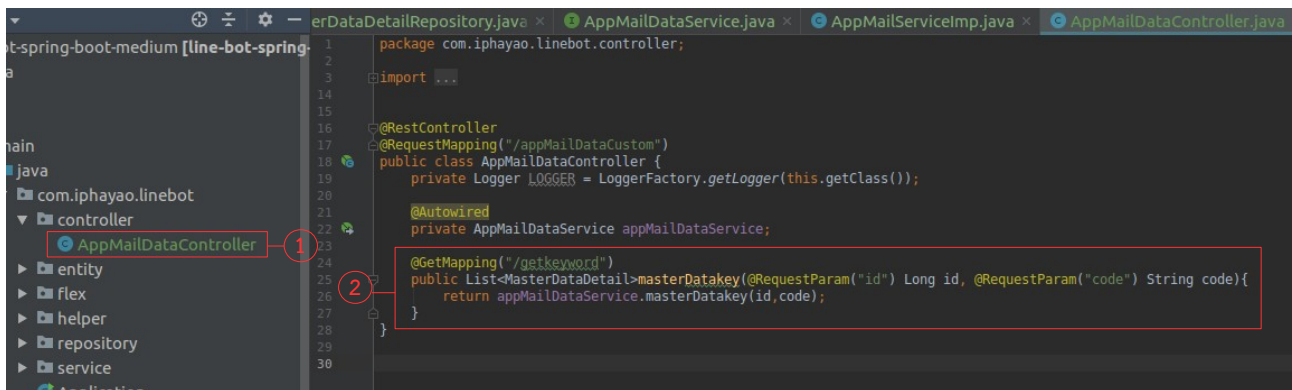
2.8.1 เมื่อ Controller ส่งค่า id และ code มาจะทำการ Query Keyword ออกมาเก็บไว้ที่ listKeyword

2.8.2 get ค่าใน listKeyword ตำแหน่งที่ 0 Column ชื่อ Variable1 เก็บไว้ที่ resultKeyword โดยกำหนดให้เป็น null ทุกครั้งก่อนเรียกใช้ เพื่อไม่ให้ Keyword ค้าง

2.8.3 เงื่อนไข Switch case โดยที่ code ต้องเท่ากับค่าใน case นั้นถึงจะทำงาน เช่น id = 100 และ code = program.list จะทำการตัดคำเป็นแบบ ArrayList โดยใช้ “,” เป็นการตัดคำจากนั้น วง loop เพิ่มคำลงไปใน LIST_KEYWORD เพื่อไปใช้ในการหาคำของ LINE Bot

2.9 สร้าง Controller

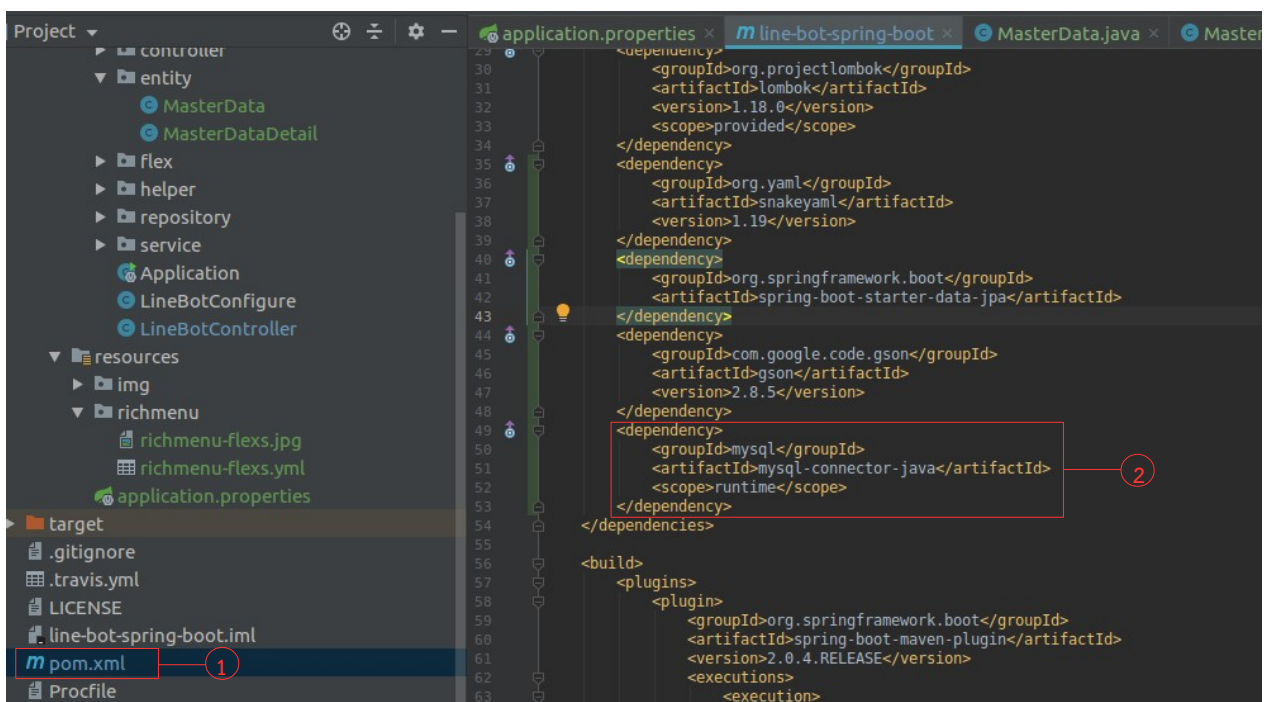
เพื่อส่งค่าไปหา AppMailServiceImp



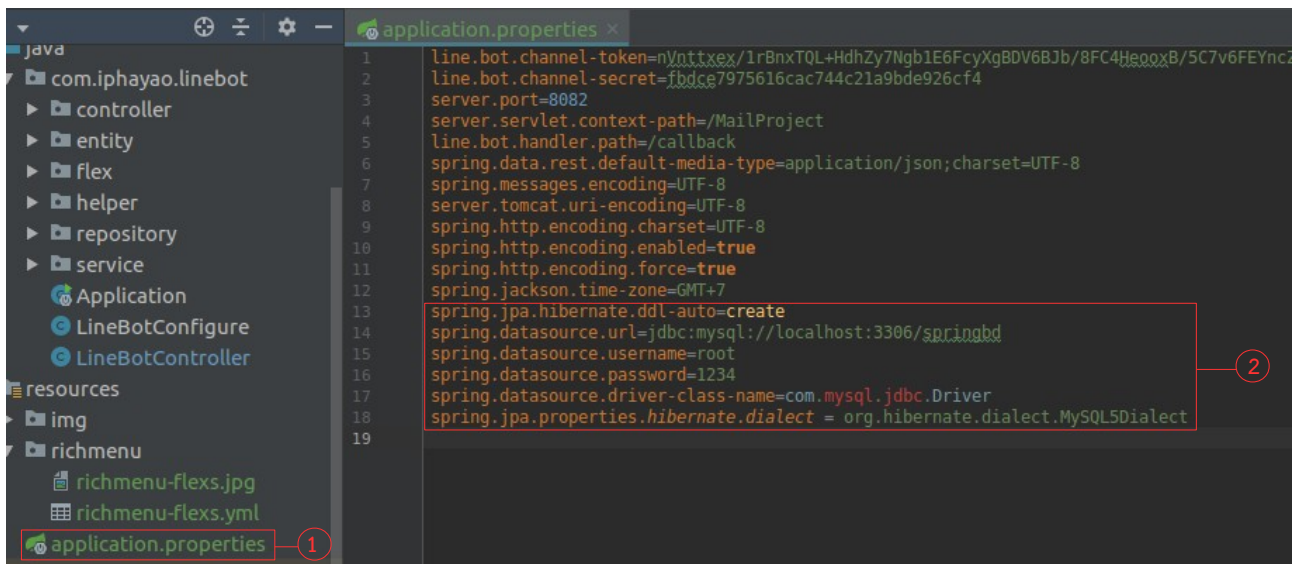
รูปที่ 2.10 แสดงแผนภาพ AppMailServiceImp

2.10 ใช้ฐานข้อมูล ในที่นี้คือ MySQL

Import MySQL เข้ากับ Project โดยไปที่ pom.xml



รูปที่ 2.11 แสดงแผนภาพ Import MySQL



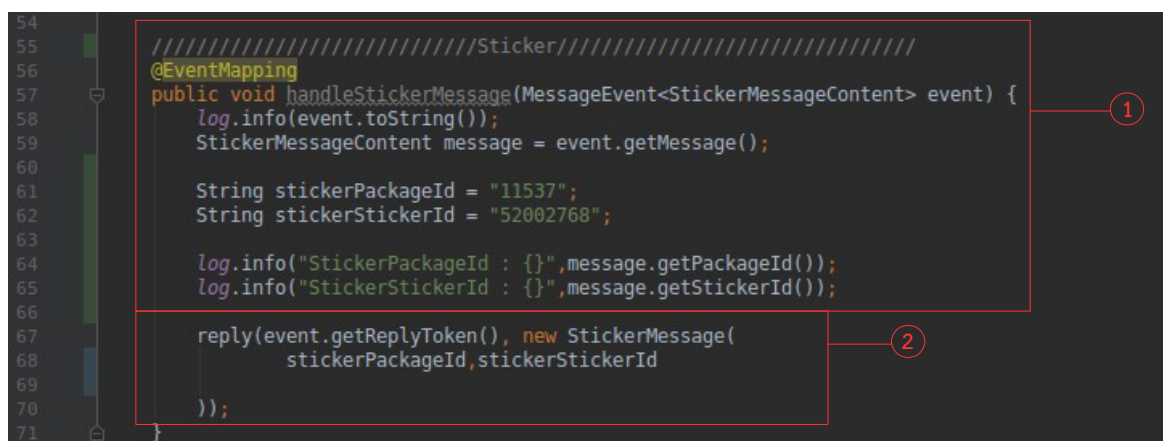
รูปที่ 2.12 แสดงแผนภาพ ตั้งค่า MySQL ชื่อคือ springbd

2.11 ทำการเพิ่ม Keyword ลงไปในฐานข้อมูล และทำการยิง Path ของ Controller

1. INSERT INTO master_data (ID ,CODE)VALUES(100 ,'LINE');
2. INSERT INTO master_data_detail (id,code ,created_by ,variable1,masterdata) VALUES (1,'keyword.list' ,'system' ,'แจ้ง,ฟัง,เสีย,แก้ไข,ใช้ไม่ได้,ล่ม,มีปัญหา,แจ้ง,ไม่แสดง,ไม่พบ,ไม่ครบ,ห้าม,ไม่มี,หาย,ปฏิเสธ,ปฏิเสธ,ไม่เจอ,ไม่เห็น,ไม่คบ,ไม่ครบ,ไม่พร้อม,ผิด,เปลี่ยน,บัก,บัค,ไม่ขึ้น,พลาด,ไม่ออก,ไม่ดี,ไม่สมบูรณ์,ไม่สมบูรณ์,แย่,ค้าง,ยังไม่ได้,ไม่ได้,Bug,bug',100);
3. INSERT INTO master_data_detail (id,code ,created_by ,variable1,masterdata) VALUES (2,'program.list' ,'system' ,'แจ้ง,โปรแกรม,ระบบ,งาน,เว็บ,เว็บไซต์,หน้าจอ',100);

รูปที่ 2.13 แสดงแผนภาพ เพิ่ม Keyword ลงไปในฐานข้อมูล

2.12 เหตุการณ์ของสติ๊กเกอร์ (Sticker)



รูปที่ 2.14 แสดงแผนภาพ เหตุการณ์ของสติ๊กเกอร์ (Sticker)

2.12.1 เหตุการณ์ของสติ๊กเกอร์ (Sticker) กำหนดค่าของสติ๊กเกอร์ได้โดยใส่ค่าลงใน

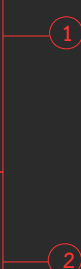
StickerPackageId และ StickerStickerId หาเลขของสติ๊กเกอร์โดย ลองส่งสติ๊กเกอร์หา LINE Bot โปรแกรมจะทำการ getStick ออกมาหน้า Terminal

2.12.2 การตอบกลับของ LINE Bot เป็นสติ๊กเกอร์ที่ได้กำหนดไว้ที่ StickerPackageId และ StickerStickerId

2.13 เหตุการณ์ของที่ตั้ง (Location)

```
/////////////////////////////////Location/////////////////////////////////
@EventMapping
public void handleLocationMessage(MessageEvent<LocationMessageContent> event) {
    log.info(event.toString());
    LocationMessageContent message = event.getMessage();
    log.info("Title : {}",message.getTitle());
    log.info("Address : {}",message.getAddress());
    log.info("Latitude : {}",message.getLatitude());
    log.info("Longitude : {}",message.getLongitude());

    reply(event.getReplyToken(), new LocationMessage(
        (message.getTitle() == null) ? "Location replied" : message.getTitle(),
        message.getAddress(),
        message.getLatitude(),
        message.getLongitude()
    ));
}
```

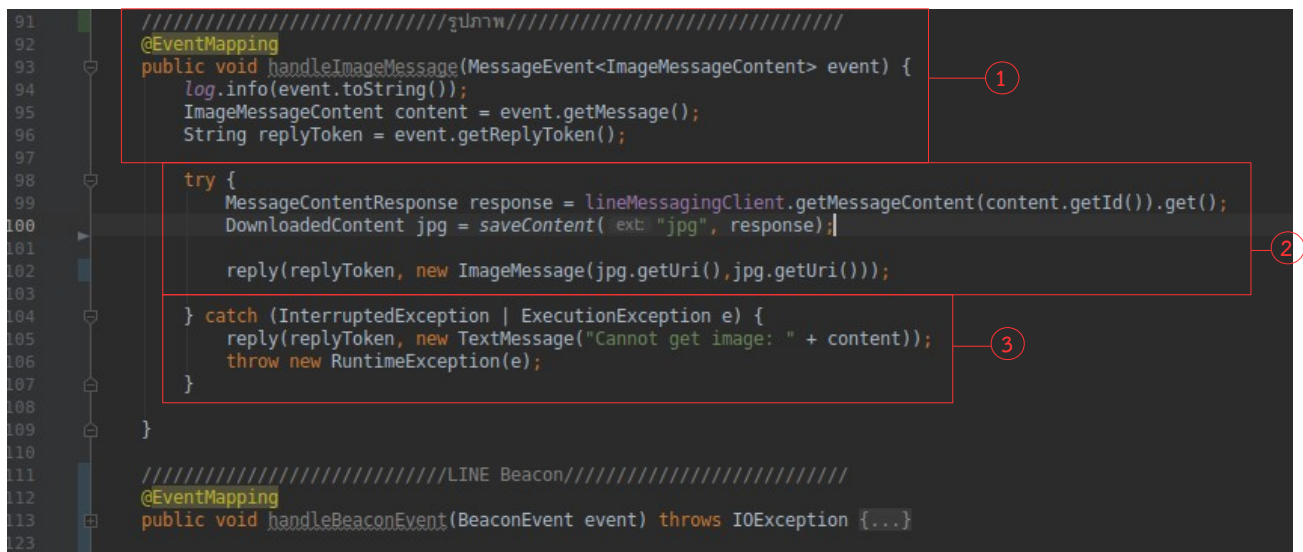


รูปที่ 2.15 แสดงแผนภาพ เหตุการณ์ของที่ตั้ง (Location)

2.13.1 เหตุการณ์ของที่ตั้ง (Location) เมื่อส่งที่ตั้ง โปรแกรมจะทำการ getLocation ออกมาหน้า Terminal

2.13.2 การตอบกลับของ LINE Bot เป็นที่ตั้ง

2.14 เหตุการณ์ของรูปภาพ



รูปที่ 2.16 แสดงแผนภาพ เหตุการณ์ของรูปภาพ

2.14.1 รับข้อความที่เป็นรูปภาพ มาเก็บไว้ที่ content และ get Token มาเก็บไว้ที่ replyToken

2.14.2 เรียกใช้ Service ของ lineMessagingClient จะได้ รูปภาพ และ Save รูปภาพเป็น jpg

และ LINE Bot ตอบกลับรูปเดิมที่ได้ส่งมา

2.14.3 ถ้าส่งรูปไม่สำเร็จ จะแสดงคำว่า Cannot get image: Id รูปภาพ

2.15 Save รูปภาพ



รูปที่ 2.17 แสดงแผนภาพ Save รูปภาพ

2.15.1 เป็นการ Save รูปภาพ โดยได้รูปภาพเป็น response และที่อยู่ File คือ tempFile

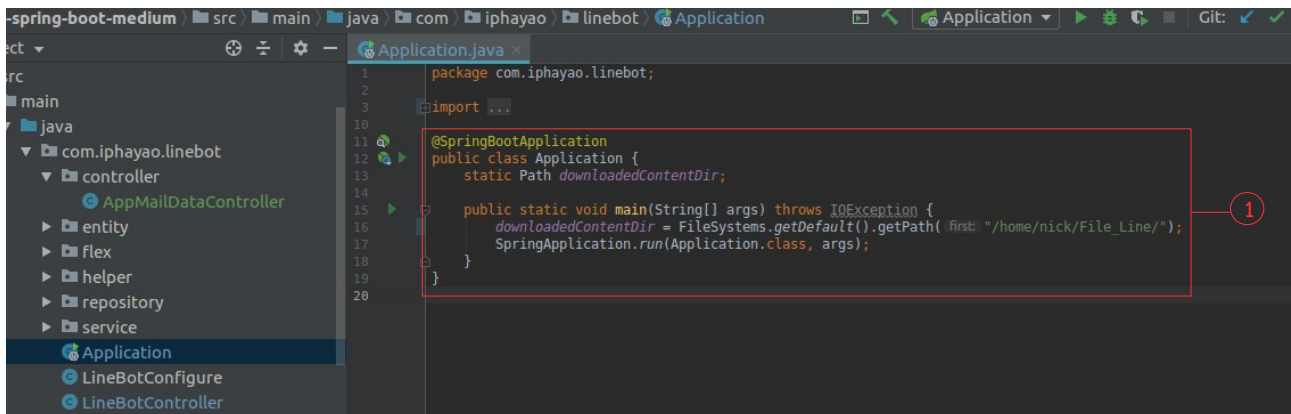
2.15.2 เป็นการ desof Path โดย From เป็นวันที่, -, randomUUID, ., jpg และ return กลับไป

DownloadedContent จะได้ค่า path และ uri

2.15.3 เรียกใช้ Service ขอ uri เพื่อให้ Online

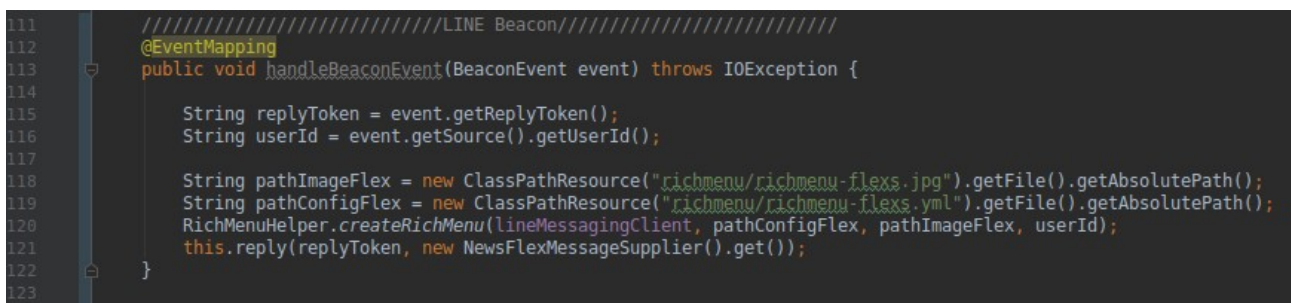
2.15.4 เก็บค่าของ path และ uri

2.16 ตั้งค่า path



รูปที่ 2.18 แสดงแผนภาพ ตั้งค่า path

2.17 เหตุการณ์ของ Beacon

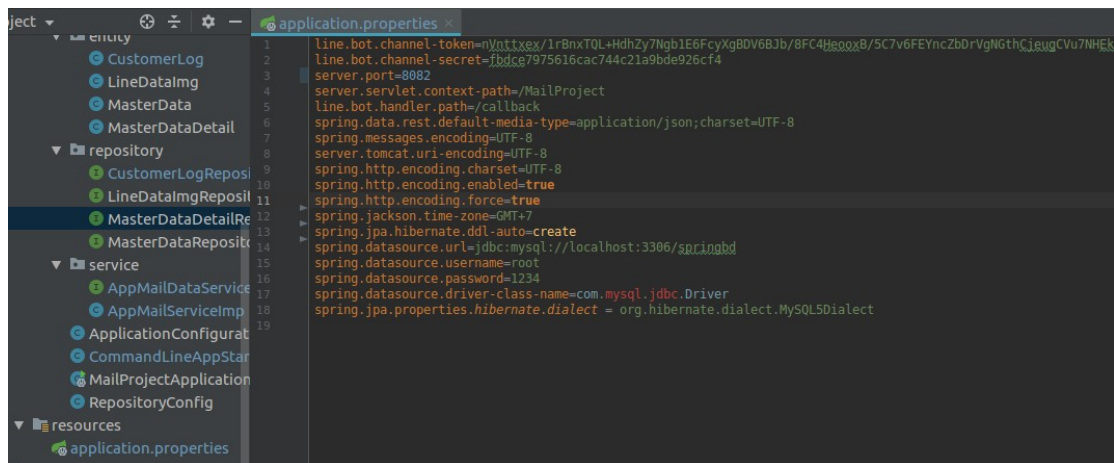


รูปที่ 2.19 แสดงแผนภาพ เหตุการณ์ของ Beacon

ดูได้ที่ <https://drive.google.com/file/d/1Q5C2y4pLSc5NmFRkrifWDHRR69cZ7dGQ/view?usp=sharing>

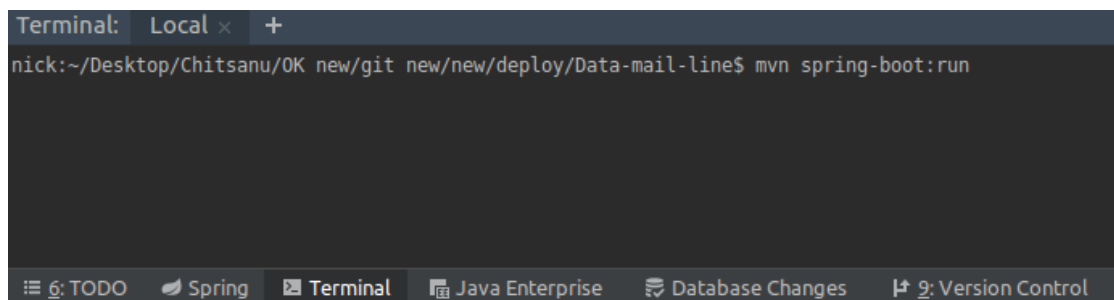
3. Run LINE Bot

3.1 ตั้งค่าต่างๆได้ที่ Application.properties

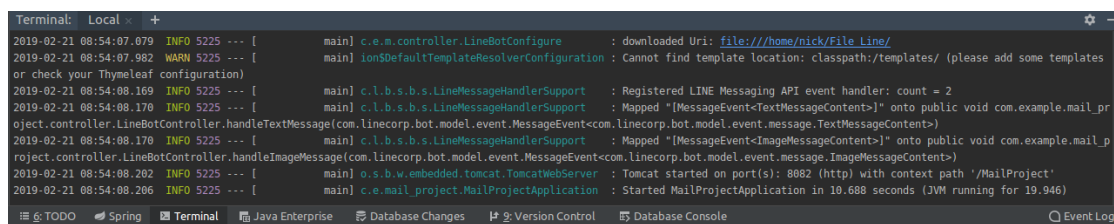


รูปที่ 3.1 แสดงแผนภาพ ตั้งค่าใน Application.properties

3.2 ไปที่หน้า Terminal เพื่อ Run



รูปที่ 3.2 แสดงแผนภาพ mvn spring-boot:run



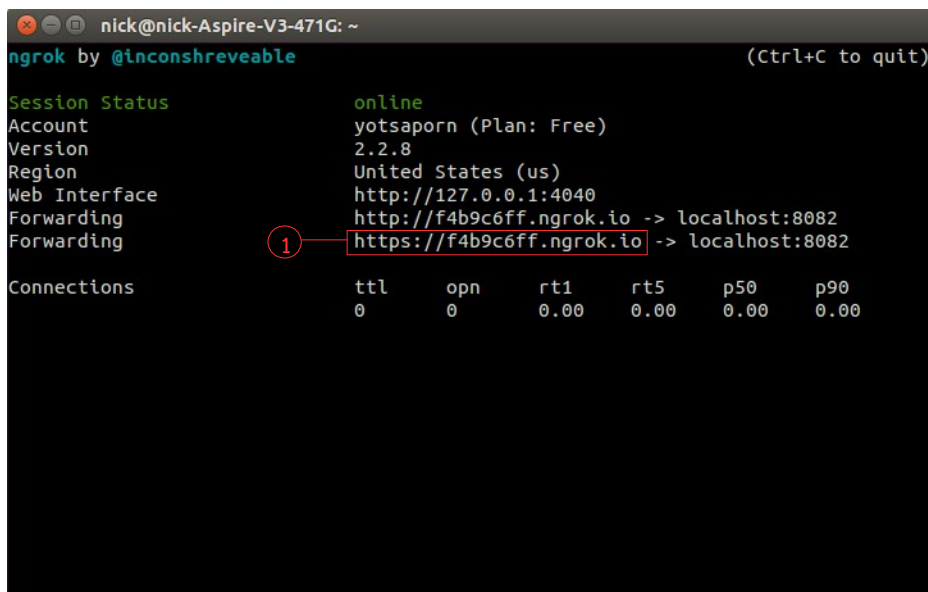
รูปที่ 3.3 แสดงแผนภาพ port ที่ได้ 8082

4. ทำให้ LINE Bot Online

ทำการติดตั้ง ngrok เพื่อให้ Online

```
$ mkdir ngrok
$ cd ngrok/
$ wget -c https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
$ unzip ngrok-stable-linux-amd64.zip
$ ./ngrok http 8082
```

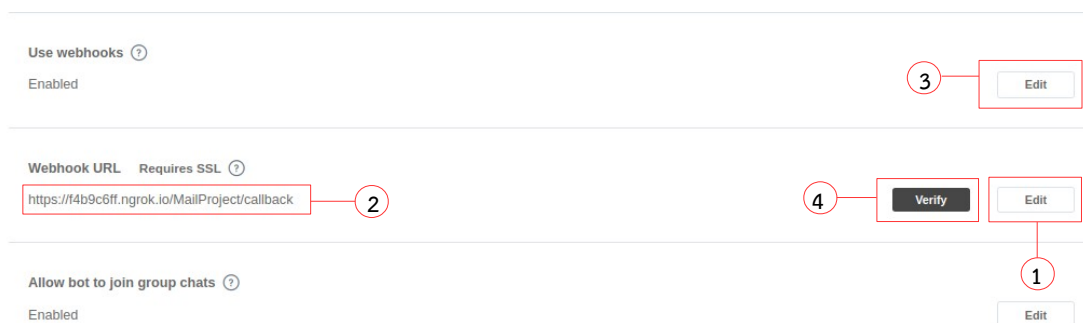
รูปที่ 4.1 แสดงแผนภาพ ติดตั้ง ngrok



รูปที่ 4.2 แสดงแผนภาพ Run ./ngrok http 8082

4.1 ตั้งค่าให้กับ Messaging API

เพื่อให้ LINE Bot ทำงานกด Use webhooks ให้เป็น Enabled และ Webhook URL ให้นำ https ที่ได้จาก ngrok มาใส่ตามด้วย callback



รูปที่ 4.3 แสดงแผนภาพ ตั้งค่าให้กับ Messaging API