主題 3：spring-boot-starter-mail

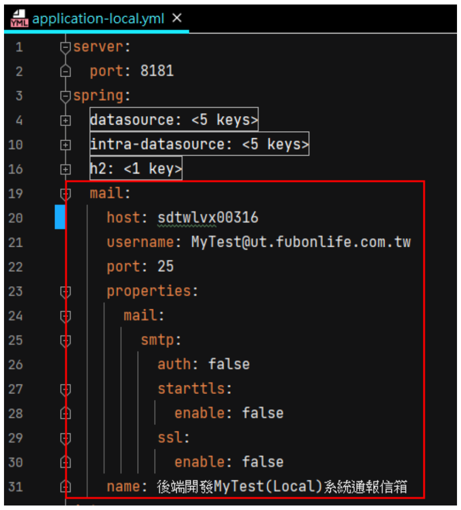
開始實作

> *步驟一：配置 maven 依賴（D:\workspace-IntelliJ\mytest-backend\pom.xml）*

pom.xml（mytest-backend）

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

        ....

        <properties>
                ....
            <mail.version>2.5.7</mail.version>
        </properties>

        ....

   <dependencyManagement>
        <dependencies>
                        ....
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-mail</artifactId>
                <version>${mail.version}</version>
            </dependency>
        </dependencies>
    </dependencyManagement>


    <dependencies>
        ....
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-mail</artifactId>
        </dependency>
    </dependencies>

    ....

</project>
```

> *步驟二：在「application-xxx.yml」設定郵件伺服器、寄件者資訊*



```yaml
server:
  port: 8181
spring:
  datasource: <5 keys>
  intra-datasource: <5 keys>
  h2: <1 key>
  mail:
    host: sdtwlvx00316
    username: MyTest@ut.fubonlife.com.tw
    port: 25
    properties:
      mail:
        smtp:
          auth: false
          starttls:
            enable: false
          ssl:
            enable: false
    name: 後端開發MyTest(Local)系統通報信箱
```

> *步驟三：在資料庫建立資料表，並產生相對應的Entity、Repository、Service*

資料表名稱	用途	備註
ATTACHMENT	上傳附件清單	
EMAIL_HISTORY	歷史信件紀錄	因欄位 SEND_TO、CC 包含電子郵件信箱，所以需要有遮罩
SYS_CODE_MAIN	代碼檔，紀錄用途	系統管理員EMAIL
SYS_CODE_DETAIL	代碼檔，紀錄大分類	MainId=sys_admin_email_list
SYS_CODE_GROUP	代碼檔，記錄小分類	信件主旨 MainId=mail、DetailId=subject 信件內容模板 MainId=mail、DetailId=content

> *步驟四：實作 EMailService（介面）提供非同步的寄信功能，以及留存信件的歷史記錄*
>
> *實作 EMailHistoryService（介面）來讓資料表 EMAIL_HISTORY 能留存信件的歷史記錄*

EMailServiceImpl.java

```java
package com.fubonlife.mytest.common.service.impl;

import com.fubonlife.mytest.common.entity.primary.SysCodeDetail;
import com.fubonlife.mytest.common.entity.primary.SysCodeGroup;
import com.fubonlife.mytest.common.model.email.EMailVo;
import com.fubonlife.mytest.common.repository.primary.SysCodeDetailRepository;
import com.fubonlife.mytest.common.service.EMailHistoryService;
import com.fubonlife.mytest.common.service.EMailService;
import com.fubonlife.mytest.common.service.SysCodeGroupService;
import com.fubonlife.mytest.common.service.SysCodeMainService;
import com.fubonlife.mytest.common.util.CommonUtil;
import com.google.common.collect.Lists;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang3.ObjectUtils;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.scheduling.annotation.Async;
import org.springframework.stereotype.Service;

import javax.mail.MessagingException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import java.io.UnsupportedEncodingException;
import java.nio.charset.StandardCharsets;
import java.util.Collections;
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

@Service
@Slf4j
public class EMailServiceImpl implements EMailService {

    @Autowired
    JavaMailSender javaMailSender;

    @Autowired
    EMailHistoryService eMailHistoryService;

    @Autowired
    SysCodeMainService sysCodeMainService;

    @Autowired
    SysCodeGroupService sysCodeGroupService;

    @Autowired
    SysCodeDetailRepository sysCodeDetailRepository;


    @Value("${spring.profiles.active}")
    private String springProfile;

    @Value("${spring.mail.name}")
    private String mailName;

    @Value("${spring.mail.username}")
    private String fromAddress;

    @Value("${spring.mail.allow-single-recipient:false}")
    private boolean allowSingleRecipient;


    //
    @Override
    public String getSubject(String sysCodeGroupId) {
        Optional<SysCodeGroup> result = sysCodeGroupService.get("mail", "subject", sysCodeGroupId);

        //
        return String.format("%s%s",
            StringUtils.equalsIgnoreCase("prod", springProfile) ? "" : "" + springProfile.toUpperCase() + "",
            result.isPresent() ? result.get().getDefaultText() : sysCodeGroupId + "");
    }

    //
    @Override
    public String getContent(String sysCodeGroupId) {
        Optional<SysCodeGroup> result = sysCodeGroupService.get("mail", "content", sysCodeGroupId);
        return result.isPresent() ? result.get().getDefaultText() : sysCodeGroupId + "";
    }

    //
    @Async
    @Override
    public void sendEMail(EMailVo eMailVo) {
        if (StringUtils.equalsIgnoreCase("local", springProfile)) {
            log.info("I Am a Fake Postman\n{}", eMailVo);
        } else {
            try {
                Thread.sleep(500); // 16
            } catch (InterruptedException ignored) {
            }

            // sendEmail(@Async) sysCodeMainService.listSysCodeDetailByMainId("sys_admin_email_list")
            List<String> adminList = sysCodeDetailRepository.findBySysCodeMainId("sys_admin_email_list").
stream()
                .filter(SysCodeDetail::isEnabled)
                .map(SysCodeDetail::getDefaultText)
                .collect(Collectors.toList());

            try {
                //
                List<String> originalRecipients = eMailVo.getToAddress();

                List<String> toAddresses = filterAddress(originalRecipients);
                if (toAddresses == null) {

                    //
                    toAddresses = Lists.newArrayList(adminList);
                    eMailVo.setHasBcc(false);

                    //
                    if (toAddresses.stream().anyMatch(x -> x.toLowerCase().contains("ut.fubonlife.com.tw"))) {
                        if (ObjectUtils.isNotEmpty(originalRecipients)) {
                            List<List<String>> data = originalRecipients.stream()
                                .filter(StringUtils::isNotBlank)
                                .map(Collections::singletonList)
                                .collect(Collectors.toList());
                            data.add(0, Collections.singletonList(""));
                            eMailVo.setContent(String.format("%s<br><br><br>%s", eMailVo.getContent(),
CommonUtil.getHtmlCodeForTable(data, true)));
                        }
                    }
                }
                eMailVo.setToAddress(toAddresses);

                //
                List<String> ccAddresses = filterAddress(eMailVo.getCcAddress());
                if (ccAddresses == null) {
                    //
                    if (allowSingleRecipient && toAddresses.size() == 1 && ObjectUtils.isNotEmpty(adminList) &&
!adminList.contains(toAddresses.get(0))) {
                        ccAddresses = adminList;
                        eMailVo.setHasBcc(false);
                    }
                }
                eMailVo.setCcAddress(ccAddresses);

                //
                callPostman(eMailVo);
            } catch (Exception e) {
                log.error("\n{}", eMailVo, e);
            }
        }
    }

    // EMAIL
    private List<String> filterAddress(List<String> address) {
        if (ObjectUtils.isEmpty(address) || !StringUtils.equalsIgnoreCase("prod", springProfile)) {
            return null;
        } else {
            address = address.stream()
                .filter(x -> StringUtils.isNotBlank(x) && !x.toLowerCase().contains("ut.fubonlife.com.tw"))
                .collect(Collectors.toList());
            return address.isEmpty() ? null : address;
        }
    }

    //
```

```java
    private void callPostman(EMailVo eMailVo) throws MessagingException, UnsupportedEncodingException {
        eMailVo.setContent(eMailVo.getContent() + "<br><br><font color='red'></font><br><br>");
        log.info("Postman Coming\n{}", eMailVo);

        List<String> adminList = sysCodeDetailRepository.findBySysCodeMainId("sys_admin_email_list").stream()
            .filter(SysCodeDetail::isEnabled)
            .map(SysCodeDetail::getDefaultText)
            .collect(Collectors.toList());

        //
        MimeMessage mimeMessage = javaMailSender.createMimeMessage();

        MimeMessageHelper helper = new MimeMessageHelper(mimeMessage, true, StandardCharsets.UTF_8.name());
        helper.setSubject(eMailVo.getSubject());
        helper.setText(eMailVo.getContent(), true);
        helper.setFrom(fromAddress, mailName);
        helper.setTo(eMailVo.getToAddress().toArray(new String[0])); //

        if (ObjectUtils.isNotEmpty(eMailVo.getCcAddress())) { //
            helper.setCc(eMailVo.getCcAddress().toArray(new String[0]));
        }

        if (ObjectUtils.isNotEmpty(adminList) && eMailVo.isHasBcc()) { //
            InternetAddress[] bccAddresses = new InternetAddress[adminList.size()];
            for (int i = 0; i < adminList.size(); i++) {
                bccAddresses[i] = new InternetAddress(adminList.get(i));
            }
            helper.setBcc(bccAddresses);
        }

        if (ObjectUtils.isNotEmpty(eMailVo.getAttachment())) { //
            helper.setEncodeFilenames(true);
            eMailVo.getAttachment().forEach(x -> {
                try {
                    helper.addAttachment(x.getFileNameWithExtension(), x.getFile());
                } catch (MessagingException e) {
                    log.error("can not find any attachment file", e);
                }
            });
        }

        //
        javaMailSender.send(mimeMessage);
        log.info("Postman Left");

        //
        eMailHistoryService.add(eMailVo);
    }
}
```

EMailHistoryServiceImpl.java

```java
package com.fubonlife.mytest.common.service.impl;

import com.fubonlife.mytest.common.entity.primary.EMailHistory;
import com.fubonlife.mytest.common.model.email.EMailAttachmentVo;
import com.fubonlife.mytest.common.model.email.EMailHistoryDto;
import com.fubonlife.mytest.common.model.email.EMailVo;
import com.fubonlife.mytest.common.repository.primary.EMailHistoryRepository;
import com.fubonlife.mytest.common.service.EMailHistoryService;
import com.fubonlife.mytest.common.util.CommonUtil;
import lombok.extern.slf4j.Slf4j;
import org.modelmapper.ModelMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.time.LocalDateTime;
import java.util.List;
import java.util.stream.Collectors;

@Service
@Slf4j
public class EMailHistoryServiceImpl implements EMailHistoryService {

    @Autowired
    EMailHistoryRepository mailHistoryRepository;

    @Autowired
    ModelMapper modelMapper;

    @Value("${spring.mail.username}")
    private String fromAddress;


    @Override
    public List<EMailHistoryDto> getEMailHistory(LocalDateTime sDate, LocalDateTime eDate) {
        return mailHistoryRepository.findByCreateDateTimeBetween(sDate, eDate).stream()
            .map(x -> {
                EMailHistoryDto dto = modelMapper.map(x, EMailHistoryDto.class);
                return dto.toBuilder()
                    .sender(x.getCreateUser())
                    .sendTime(x.getCreateDateTime().toString())
                    .build();
            })
            .collect(Collectors.toList());
    }

    @Override
    @Transactional
    public void add(EMailHistory eMailHistory) {
        try {
            mailHistoryRepository.save(eMailHistory);
        } catch (Exception e) {
            log.error("({})", eMailHistory.getSendFrom(), e);
        }
    }

    @Override
    public void add(EMailVo eMailVo) {
        EMailHistory entity = modelMapper.map(eMailVo, EMailHistory.class);

        entity.setSendFrom(CommonUtil.getMaskEMailAddress(fromAddress));
        entity.setSendTo(String.join(";", CommonUtil.getMaskEMailAddress(eMailVo.getToAddress())));

        if (eMailVo.getCcAddress() != null) {
            entity.setCc(String.join(";", CommonUtil.getMaskEMailAddress(eMailVo.getToAddress())));
        }

        if (eMailVo.getAttachment() != null) {
            String fileName = eMailVo.getAttachment().stream()
                .map(EMailAttachmentVo::getFileNameWithExtension)
                .collect(Collectors.joining(";"));
            entity.setFileName(fileName);
        }

        add(entity);
    }
}
```

CommonUtil.java

```
package com.fubonlife.mytest.common.util;

import org.apache.commons.lang3.StringUtils;
import org.apache.commons.lang3.math.NumberUtils;
import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

public class CommonUtil {

    /**
     * UUID
     */
    public static String getUUID() {
        return UUID.randomUUID().toString();
    }

    /**
     *
     */
    public static String trim2SpaceChar(String inputString) {
        if (StringUtils.isNotBlank(inputString)) {
            char[] chars = inputString.toCharArray();
            int inputStringLength = chars.length;

            int startIndex = (Character.isSpaceChar(chars[0])) ? 1 : 0;
            int endIndex = (Character.isSpaceChar(chars[inputStringLength - 1])) ? inputStringLength - 1 :
inputStringLength;

            return inputString.substring(startIndex, endIndex);
        } else {
            return "";
        }
    }

    /**
     *
     */
    public static String combineWords(String newWord, String oldWord) {
        if (StringUtils.isBlank(oldWord)) {
            return newWord;
        } else {
            return String.format("%s(%s)", newWord, oldWord);
        }
    }

    /**
     *
     */
    public static String convertToChineseNumber(Object number) {
        int result = 0;

        if (number instanceof String) {
            String s = String.valueOf(number);
            if (NumberUtils.isDigits(s)) {
                result = Integer.parseInt(s);
            }
        }

        switch (result) {
            case 1:
                return "";
            case 2:
                return "";
            case 3:
                return "";
            case 4:
                return "";
            case 5:
                return "";
            case 6:
                return "";
            case 7:
                return "";
            case 8:
                return "";
            case 9:
                return "";
            case 10:
                return "";
            case 11:
                return "";
            case 12:
                return "";
            default:
                return "";
        }
    }

    /**
     * HTML-
     */
    public static String getHtmlCodeForTable(List<List<String>> data, boolean hasHeader, String...
overrideTableStyle) {
        StringBuilder htmlCode = new StringBuilder(data.size() * 100);

        //
        if (overrideTableStyle != null && overrideTableStyle.length > 0) {
            htmlCode.append("<table border='1' cellspacing='0' cellpadding='5' style='");
            for (String tableStyle : overrideTableStyle) {
                htmlCode.append(tableStyle).append(";");
            }
            htmlCode.append("'>");
        } else {
            htmlCode.append("<table border='1' cellspacing='0' cellpadding='5' style='text-align:center;margin-
top:20;'>");
        }

        for (List<String> oneRow : data) {
            htmlCode.append("<tr>");
            if (hasHeader) {
                for (String contentForTH : oneRow) {
                    htmlCode.append("<th>").append(contentForTH).append("</th>");
                }
                hasHeader = false;
            } else {
                for (String contentForTD : oneRow) {
                    htmlCode.append("<td>").append(contentForTD).append("</td>");
                }
            }
            htmlCode.append("</tr>");
        }

        return htmlCode.append("</table>").toString();
    }

    /**
     *
     */
    public static String getMaskName(String username) {
        if (StringUtils.isNotBlank(username)) {

            int length = username.length();

            if ((username.getBytes().length != length) && (username.getBytes().length % length == 0)) {
                // 2*1
                if (length == 2) {
                    username = String.format("%s", username.charAt(0));
                }
                // 2*2
                else if (length > 2) {
                    username = String.format("%s%s", username.substring(0, length - 2), username.substring
(length - 1));
                }
            } else {
                if (length == 1) {
```

```
                username = "*";
            }
            // 7*2
            else if (length <= 7) {
                username = String.format("%s%s", username.substring(0, length - 2), username.substring
(length - 1));
            }
            // 7*1~6
            else {
                username = String.format("%s", username.substring(6));
            }
        }
    } else {
        username = "";
    }

    return username;
}

/**
 * EMail address
 */
public static List<String> getMaskEMailAddress(List<String> mailAddresses) {
    return mailAddresses.stream().map(CommonUtil::getMaskEMailAddress).collect(Collectors.toList());
}

/**
 * EMail address
 */
public static String getMaskEMailAddress(String eMailAddress) {
    if(StringUtils.isNotBlank(eMailAddress)) {
        String [] mailArr = eMailAddress.split("@");
        if(mailArr.length > 0) {
            return String.format("%s@%s", getMaskString(mailArr[0], 4, 0), getMaskString(mailArr[1], 4, 8));
        }
    }
    return "";
}

/**
 *
 *
 * @param  originalString
 * @param  startIndex index
 * @param  endIndex index
 * @return String
 */
private static String getMaskString(String originalString, int startIndex, int endIndex) {
    if(StringUtils.isNotBlank(originalString)) {
        // indexindex-1
        if(startIndex > originalString.length()) {
            return getMaskString(originalString, startIndex - 1, endIndex);
        } else {
            startIndex = startIndex <= 1 ? 1 : startIndex - 1;
            if(endIndex == 0) { // index
                return String.format("%s", originalString.substring(0, startIndex));
            } else {
                endIndex = endIndex > originalString.length() ? originalString.length() - 1 : endIndex;
                return String.format("%s%s", originalString.substring(0, startIndex), originalString.
substring(endIndex));
            }
        }
    }
    return "";
}
}
```

SysCodeMainServiceImpl.java

```java
package com.fubonlife.mytest.common.service.impl;

import com.fubonlife.mytest.common.entity.primary.SysCodeDetail;
import com.fubonlife.mytest.common.entity.primary.SysCodeMain;
import com.fubonlife.mytest.common.repository.primary.SysCodeMainRepository;
import com.fubonlife.mytest.common.service.SysCodeMainService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Collections;
import java.util.List;
import java.util.Optional;

@Service
@Slf4j
public class SysCodeMainServiceImpl implements SysCodeMainService {

    @Autowired
    SysCodeMainRepository sysCodeMainRepository;


    @Override
    public List<SysCodeDetail> listSysCodeDetailByMainId(String sysCodeMainId) {
        Optional<SysCodeMain> main = sysCodeMainRepository.findById(sysCodeMainId);
        return main.isPresent() ? main.get().getSysCodeDetailList() : Collections.emptyList();
    }
}
```

SysCodeGroupServiceImpl.java

```java
package com.fubonlife.mytest.common.service.impl;

import com.fubonlife.mytest.common.entity.primary.SysCodeGroup;
import com.fubonlife.mytest.common.entity.primary.pk.SysCodeGroupPK;
import com.fubonlife.mytest.common.repository.primary.SysCodeGroupRepository;
import com.fubonlife.mytest.common.service.SysCodeGroupService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
@Slf4j
public class SysCodeGroupServiceImpl implements SysCodeGroupService {

    @Autowired
    SysCodeGroupRepository sysCodeGroupRepository;


    //
    @Override
    public Optional<SysCodeGroup> get(String sysCodeMainId, String sysCodeDetailId, String sysCodeGroupId) {
        return sysCodeGroupRepository.findById(SysCodeGroupPK.builder()
            .sysCodeMainId(sysCodeMainId)
            .sysCodeDetailId(sysCodeDetailId)
            .sysCodeGroupId(sysCodeGroupId)
            .build());
    }
}
```

> *步驟五：郵件伺服器僅供機器對機器連線，所以寄信功能只能在測試機上測試*

灑花~~~