主題 1：spring-boot-starter-web

跟後端 Rest API 服務有關的概念

1. 【 RestController 】是指前後端會透過 REST風格 + JSON格式進行溝通OR交換資料
    - 後端程式的「 Controller」就像是前後端溝通的「窗口」
2. 【 API 】可視為服務合約，而這份合約會定義兩者如何使用要求與回應彼此進行溝通OR交換資料
3. 【 Rest API 】的設計要點
    - 看 URI 就知道要前後端要處理什麼資源
    - 看 Http method 就知道要對資源做什麼
    - 看 Http status code 就知道處理結果

撰寫 Rest API

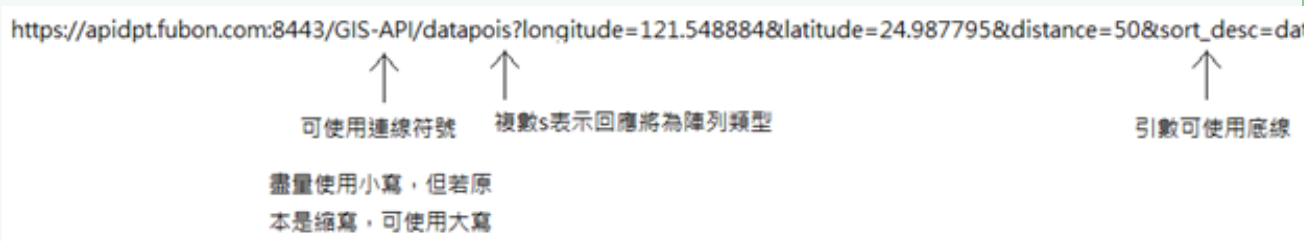
> *方法*
>
> 1. *創一個 Class*
> 2. *Class 上面加上 `@RestController`*
> 3. *Class 上面加上 `@RequestMapping`*
>   - *`Rest API URL`預設是【 `/api/<>`*
> 4. *Class 裡面創一個 public method*
> 5. *Method 上面加上適合的 http method ，預設是【 `@GetMapping` 或 @PostMapping 】*
>   - *替 `@XxxxMapping` 設定 Rest API 的 URL，例如【 `/<>`*
> 6. *這個 method 會回傳【 ResponseEntity 】*
>   - *將自動被轉換成對應的 JSON*

設計 URI 規則

【設計要點】看 URI 就知道要前後端要處理什麼資源

【樣式】http(s)://host:port/業務別(系統別)/服務別/資源或方法名稱.../...

- URI命名代表資源名稱，以名詞命名。
- URI末尾不出現斜槓/。
    - URI中使用斜槓/是代表層級關係。
- URI中名詞可使用連線符號-提升可讀性。
    - URI中不可使用下劃線_，但最後的引數(最後帶入的參數)允許使用下劃線_。
- URI中儘量使用小寫字元。
- URI中不可以使用副檔名(如.asmx)。
- URI中請儘量使用複數形式來代表集合。



資料來源：API命名規則（S037 資訊架構管理科）

選擇適合的 http method 來對資源進行操作

【設計要點】看 Http method 就知道要對資源做什麼

雖然 Restful 風格一般是以 http method 的不同來區別不一樣的功能，但實務上有可能還是仍常以 GET 或 POST 來操作資源

> *請留意...「GET」 方法不適用在能改變後端資料的 API*

設計前端傳送資料的形式

1. 若為 GET 請求，可使用 @RequestParam，同時也可搭配 @PathVariable

2. 若為 POST 請求，僅可使用 @RequestParam 或 @RequestBody，同時也可搭配 @PathVariable

	@RequestParam	@RequestBody	@PathVariable
用途	用於搜尋指定物件	用於傳送表單	用於搜尋指定物件
形式	http://localhost:8080/api/v1/books?page=1&size=10 「?」表示參數開頭，參數之間透過「&」連結	傳送的是 JSON 的字串	http://localhost:8080/aip/v1/books/{id}

舉例說明

```
@RestController
@RequestMapping(value = "/api/announcement")
@Api(tags = "Announcement", description = "")
public class AnnouncementController {

    @Autowired
    AnnouncementService announcementService;

    @Autowired
    ResponseEntityService responseEntityService;


    @GetMapping(value = "/category/list")  // <--- URI = /api/announcement/category/list
    @ApiOperation("")
    public ResponseEntity<ResponseBodyDto<List<SelectOptionDto>>> listCategoryInAnnouncement() {

        List<SelectOptionDto> result = announcementService.getAnnouncementCategoryOption();
        return (result != null && !result.isEmpty()) ? responseEntityService.ok(result) : responseEntityService.
notFound();
    }

    @PostMapping(value = "/add")  // <--- URI = /api/announcement/add
    @ApiOperation("")
    @PreAuthorize("hasAnyRole('ROOT')")
    public ResponseEntity<ResponseBodyDto<String>> addInAnnouncement(@RequestBody AddAnnouncement data) {

        if (data.getUnitCodeList().isEmpty()) {
            String result = announcementService.create(data);
            return StringUtils.isNotBlank(result) ? responseEntityService.ok(result) : responseEntityService.
badRequest("");
        } else {
            String result = announcementService.createByUnit(data);
            return StringUtils.isNotBlank(result) ? responseEntityService.ok(result) : responseEntityService.
badRequest("()");
        }
    }

    @GetMapping(value = "/unwatched")  // <--- URI = /api/announcement/unwatch
    @ApiOperation("")
    public ResponseEntity<ResponseBodyDto<String>> unwatchInAnnouncement() {

        boolean result = announcementService.hasUnwatched();
        return result ? responseEntityService.ok() : responseEntityService.notFound();
    }
```

```
    @PostMapping(value = "/list")  // <--- URI = /api/announcement/list
    @ApiOperation("")
    public ResponseEntity<ResponseBodyDto<Page<AnnouncementDto>>> listInAnnouncement(@RequestBody
ListAnnouncement data) {

        Page<AnnouncementDto> result = announcementService.getRecentList(data);
        return (result != null && result.getTotalElements() > 0) ? responseEntityService.ok(result) :
responseEntityService.notFound();
    }

    @GetMapping(value = "/{announcementId}/search")  // <--- URI = /api/announcement/search
    @ApiOperation("")
    public ResponseEntity<ResponseBodyDto<AnnouncementDto>> searchInAnnouncement(
        @PathVariable @ApiParam(value = "ID", required = true) String announcementId) {

        AnnouncementDto result = announcementService.getAnnouncement(announcementId);
        return (result != null) ? responseEntityService.ok(result) : responseEntityService.notFound();
    }

    @PostMapping(value = "/remove")  // <--- URI = /api/announcement/remove
    @ApiOperation("")
    @PreAuthorize("hasAnyRole('ROOT')")
    public ResponseEntity<ResponseBodyDto<String>> removeInAnnouncement(
        @RequestParam @ApiParam(value = "ID", required = true) String announcementId) {

        boolean result = announcementService.removeAnnouncement(announcementId);
        return result ? responseEntityService.ok() : responseEntityService.notFound();
    }

    @PostMapping(value = "/modify")  // <--- URI = /api/announcement/modify
    @ApiOperation("")
    @PreAuthorize("isAuthenticated()")
    public ResponseEntity<ResponseBodyDto<String>> modifyInAnnouncement(@RequestBody @Valid ModifyAnnouncement
data) {

        String result = announcementService.modify(data);
        return StringUtils.isNotBlank(result) ? responseEntityService.ok(result) : responseEntityService.
badRequest("");
    }
}
```

設計 ResponseEntity

【設計要點】看 Http status code 就知道處理結果

當 Http status code 等於 200 時

可能的情境	Status Code	ResponseEntity.status	ResponseEntity.body
程式正常執行，未拋出異常（有資料處理）	200	HttpStatus.OK	Service層的回傳值，或客製化訊息
程式正常執行，未拋出異常（無資料處理）	200	HttpStatus.OK	Service層的回傳值，或客製化訊息

Controller 層

```
@GetMapping(value = "/file/download")
@ApiOperation(" GET  download API")
public ResponseEntity<byte[]> downloadUsingGet(@RequestParam("param") String param) {

  byte[] fileBytes = new byte[];
  String fileName = "File Name";

  //  attachment header
  HttpHeaders headers = new HttpHeaders();
  headers.setContentDispositionFormData("attachment", encodedFileName);

  return ResponseEntity.status(HttpStatus.OK).headers(headers).body(fileBytes);
}
```

Controller 層

```
@PostMapping(value = "/file/download")
@ApiOperation(" POST download API")
public ResponseEntity<byte[]> downloadUsingPost(@RequestBody QueryTO queryTO) {

  byte[] fileBytes = new byte[];
  String fileName = "File Name";

  return ResponseEntity.ok().headers(headers).body(fileBytes);
}
```

*** 檔名如果有非 ASCII 字元會發生什麼事？特殊字元呢？試試看如何讓 Chrome 可以正確的上傳和下載中文檔名的檔案吧。

當 Http status code 大於等於 400 時

可能的情境	Controller層/Service層拋出異常	Status Code	ResponseEntity.status	ResponseEntity.body

自己 try-catch 捕抓到的任何異常	BadRequestException	400	HttpStatus.BAD_REQUEST	**統一錯誤回應格式**
				{
				"status": "BAD_REQUEST",
				"apiErrorCode":"
				INVALID_PRODUCT_COST",
				"message": "XXXXXXXX",
				"errors": [
				"XXXXXXXX"
				]
				}

登入失敗	AccessDeniedException	401	HttpStatus.UNAUTHORIZED	
沒有權限	AccessDeniedException	403	HttpStatus.FORBIDDEN	
缺少重要資源導致中止執行	ResourceNotFoundException	404	HttpStatus.NOT_FOUND	
罄竹難書的所有異常	Exception	500	HttpStatus. INTERNAL_SERVER_ERROR	

> *自己 try-catch 捕抓到的異常，若有要往外拋，需轉換成自定義異常（BadRequestException、ResourceNotFoundException）才向外層拋出*
>
> *也不建議用一個 Exception 來捕捉異常，這會無法明確區分出 apiErrorCode*

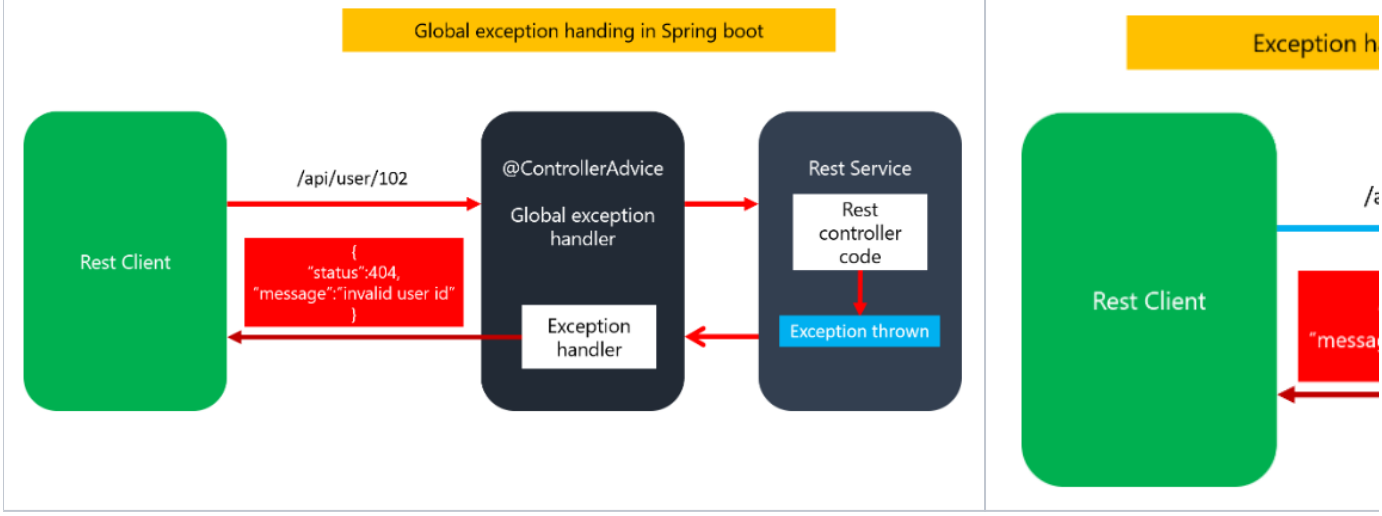
Controller層/Service層拋出異常

```
//  try-catch  400 (Bad Request)
throw new BadRequestException(ApiErrorCode.DUPLICATED_ACCOUNT);
//  ApiErrorCode  Body  apiErrorCode

//  try-catch  404 (Not Found)
throw new NotFoundException(ApiErrorCode.ACCOUNT_NOT_FOUND);

//  RuntimeException  500
throw new RuntimeException("Oops");
```

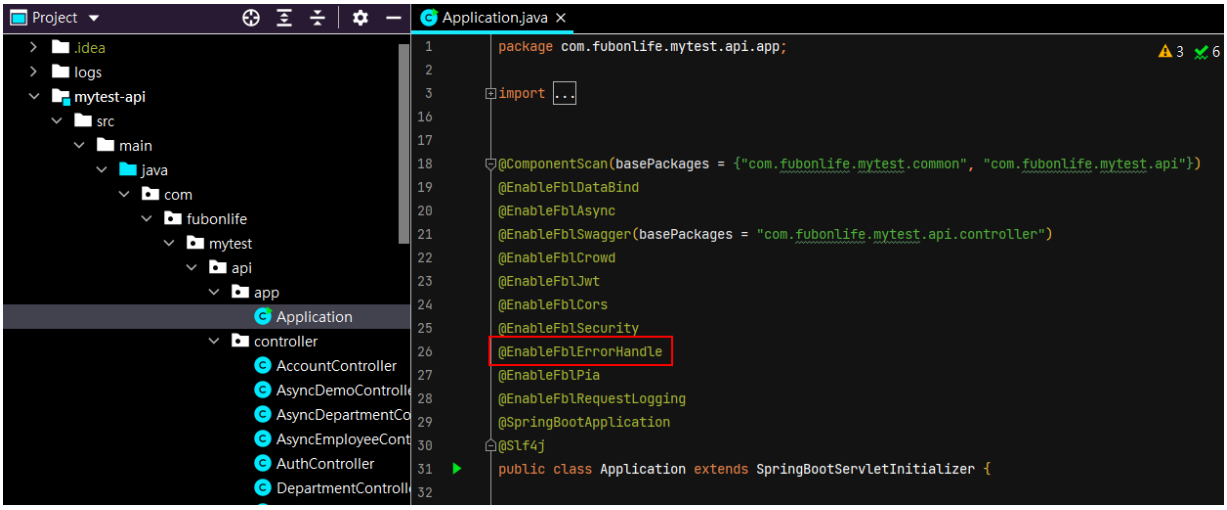
通用錯誤處理機制

> *在 Controller 層添加統一的異常處理*

較佳的作法



啟用方式



相關的程式

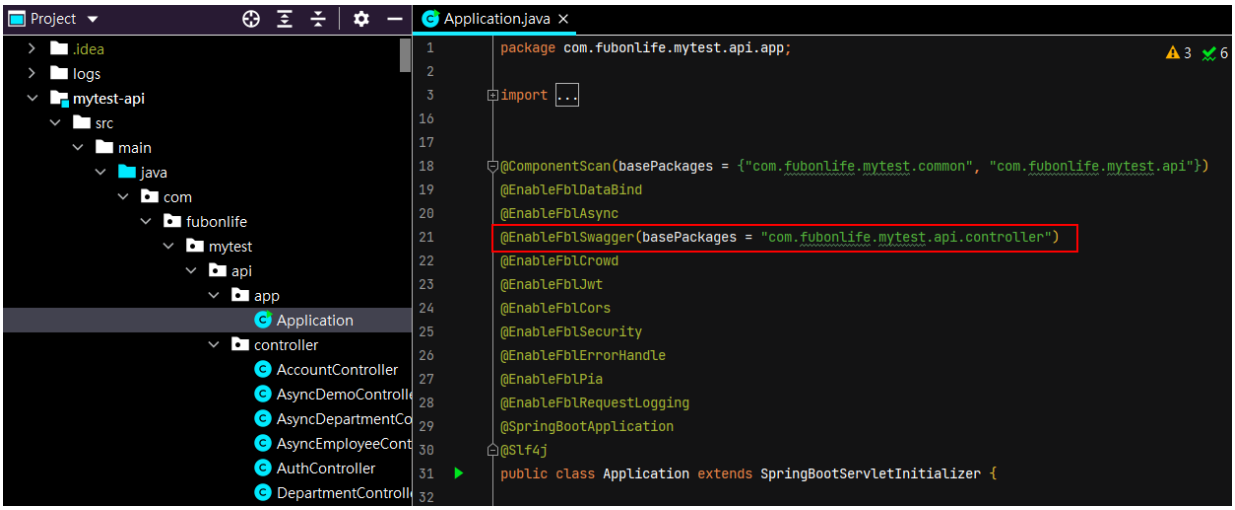
程式碼位置：com.fubonlife.boot.error.handler.CustomRestExceptionHandler.java

啟用 Swagger2

> *Swagger 可以快速產生 REST API 的 documentation，並且提供 UI 介面可以直接呼叫來做測試。*

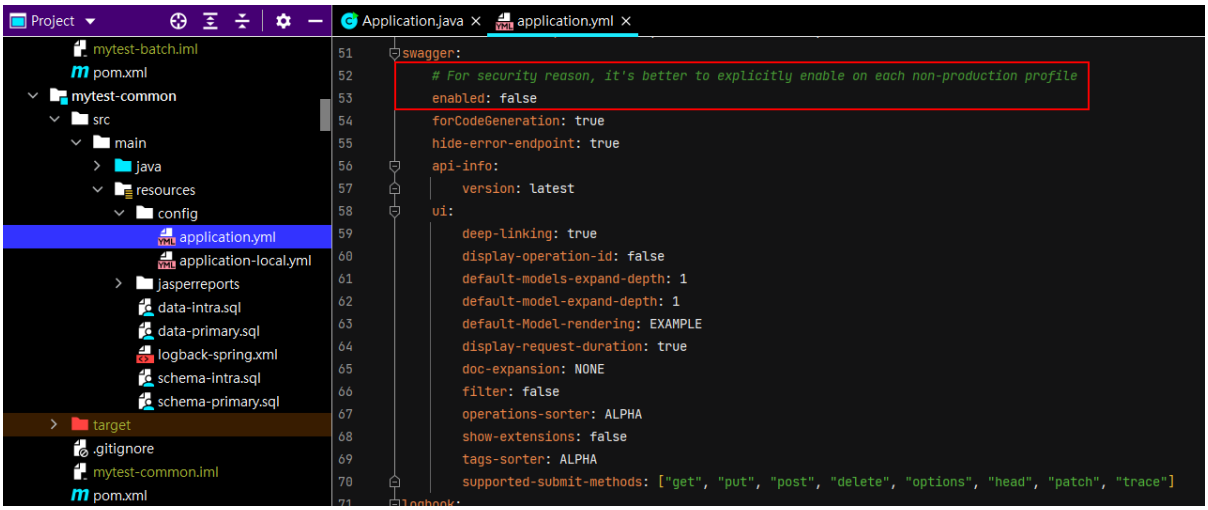
啟用方式

> *如果某個 API 不需要加入 Swagger 中，只需要在 API 上面加上 @ApiIgnore 註解就可以了*
>
> *啟動 Spring Boot Server 後，前往 http://<your ip>:8181/swagger-ui.html 就可以看到 Swagger UI 的介面。*



配置文件 application.yml 預設不顯示Swagger UI 介面

> *若有需要啟用，建議在 application-xxx.yml 內覆寫設定*



撰寫API規格

> *舉例說明*

Controller 層

```
@RestController
@RequestMapping(value = "/api/sys-code")
@Api(tags = "SysCode", description = "")
@Slf4j
public class SysCodeController {

    @Autowired
    SysCodeService sysCodeService;


    @PostMapping("/add-or-modify")
    @ApiOperation("OR")
    @PreAuthorize("hasAnyRole('ROOT')")
    public ResponseEntity<String> addOrModifyInParameter(@RequestBody AddOrModifySysCode data) {
        String id = sysCodeService.executeAddOrModify(data);

        if (id != null) {
            return ResponseEntity.ok().body("");
        } else {
            return ResponseEntity.badRequest().body("");
        }
    }

    @PostMapping("/list")
    @ApiOperation("")
    @PreAuthorize("hasAnyRole('ROOT')")
    public ResponseEntity<Page<SysCodeDto>> listInParameter(@RequestBody ListSysCode data) {
        Page<SysCodeDto> sysCodeDto = sysCodeService.list(data);

        return ResponseEntity.ok().body(sysCodeDto);
    }

}
```

Dto 檔

```
@ApiModel(description = "")
@Data
public class SysCodeDto {

    // =====()==========================================================================//

    @ApiModelProperty(value = "()", position = 1, required = true, example = "brand")
    private String sysCodeMainId;

    @ApiModelProperty(value = "()", position = 2, required = true, example = "")
    private String defaultTextInSysCodeMain;

    @ApiModelProperty(value = "", position = 3, example = "false")
    private boolean doubleCheck;

    @ApiModelProperty(value = "", position = 4, example = "false")
    private boolean isEditableInSysCodeMain;

    @ApiModelProperty(value = "", position = 5, example = "!!")
    private String remarkInSysCodeMain;

    // =====()==========================================================================//

    @ApiModelProperty(value = "()", position = 6, example = "sport")
    private String sysCodeDetailId;

    @ApiModelProperty(value = "()", position = 7, example = "")
    private String defaultTextInSysCodeDetail;

    @ApiModelProperty(value = "", position = 8, example = "1")
    private int sequenceInSysCodeDetail;

    @ApiModelProperty(value = "", position = 9)
    private boolean enabledInDetail;

    @ApiModelProperty(value = "", position = 10, example = "false")
    private boolean isEditableInSysCodeDetail;

    @ApiModelProperty(value = "", position = 11, example = "sport")
    private String remarkInSysCodeDetail;

    // =====()==========================================================================//

    @ApiModelProperty(value = "()", position = 12, example = "nike")
    private String sysCodeGroupId;

    @ApiModelProperty(value = "()", position = 13, example = "")
    private String defaultTextInSysCodeGroup;

    @ApiModelProperty(value = "(1)", position = 14, example = "1")
    private int sequenceInSysCodeGroup;

    @ApiModelProperty(value = "", position = 15)
    private boolean enabledInGroup;

    @ApiModelProperty(value = "", position = 16, example = "false")
    private boolean isEditableInSysCodeGroup;

    @ApiModelProperty(value = "", position = 17, example = "nike")
    private String remarkInSysCodeGroup;
}
```

Vo 檔

```
@ApiModel(description = "OR")
@Data
@EqualsAndHashCode(callSuper = true) //
public class AddOrModifySysCode extends SysCodeDto {

}
```

Vo 檔(含分頁參數)

```
@ApiModel(description = "()")
@Data
public class SearchSysCode {

    @ApiModelProperty(value = "()", position = 1, required = true, example = "crawler")
    private String sysCodeMainId;

    @ApiModelProperty(value = "()", position = 2, example = "examination_focus")
    private String sysCodeDetailId;

    @ApiModelProperty(value = "()", position = 3, example = "main_page_url")
    private String sysCodeGroupId;

    // =====  ==========//

    @ApiModelProperty(value = " N (=0)", required = true, position = 97, example = "0")
    private int pageIndex;

    @ApiModelProperty(value = " N ", required = true, position = 98, example = "10", allowableValues = "10, 20,
50, 100")
    private int pageSize;

}
```

開放 CORS 跨域請求

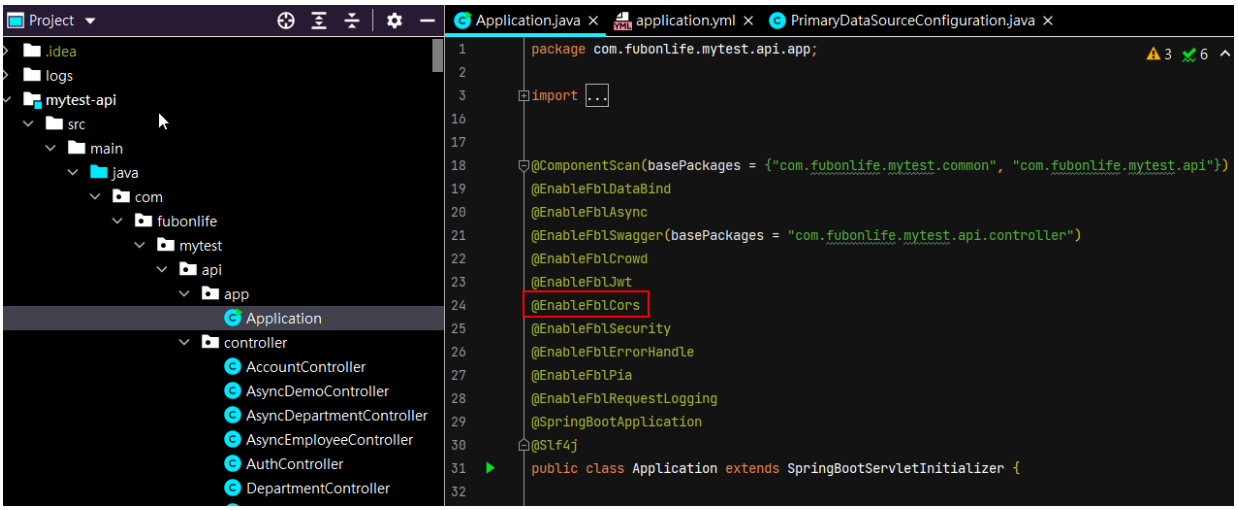
瀏覽器預設會檢查所有跨網域的 AJAX calls

> *例如：*
>
> *你在 https://example.com 要呼叫一個 https://example2.com/api/myRest*
>
> *瀏覽器會先發送一個 preflight request，用 OPTIONS method 送往該 REST API，並看他回傳的資訊是不是符合要求。*

由於前後端分離開發時，可能產生前後端域名不同，而被瀏覽器跨域阻擋導致 API 呼叫失敗。 本功能可用來暫時讓後端開放跨域請求。

> *僅供開發階段使用，除非真的在正式區有跨域的必要，建議部署時要移除此功能。*

啟用方式



最主要檢查的為下面兩項。如果檢查成功，瀏覽器才會真的去 call 目的地的 Rest API，如果檢查失敗則停止呼叫。

- Access-Control-Allow-Origin: 原本的網域有沒有落在這個值中，例如上面的範例，值可以是 "https://example.com" 或是 "*" 代表所有網域都可以
- Access-Control-Allow-Methods: 原本要求的 HTTP Method 有沒有落在這個值中，回傳值可以是複數，例如 "POST, GET, PUT"

相關的程式

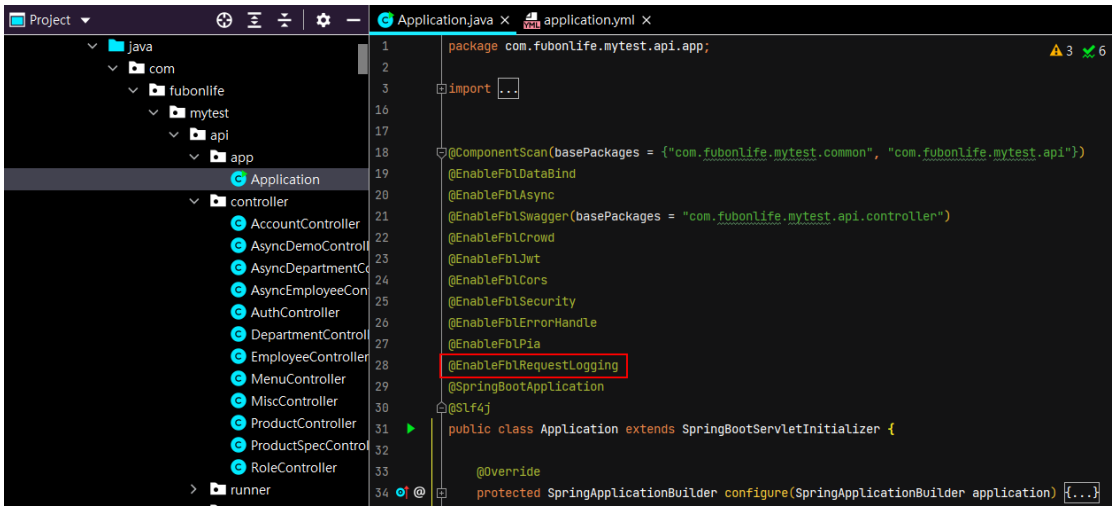
程式碼位置

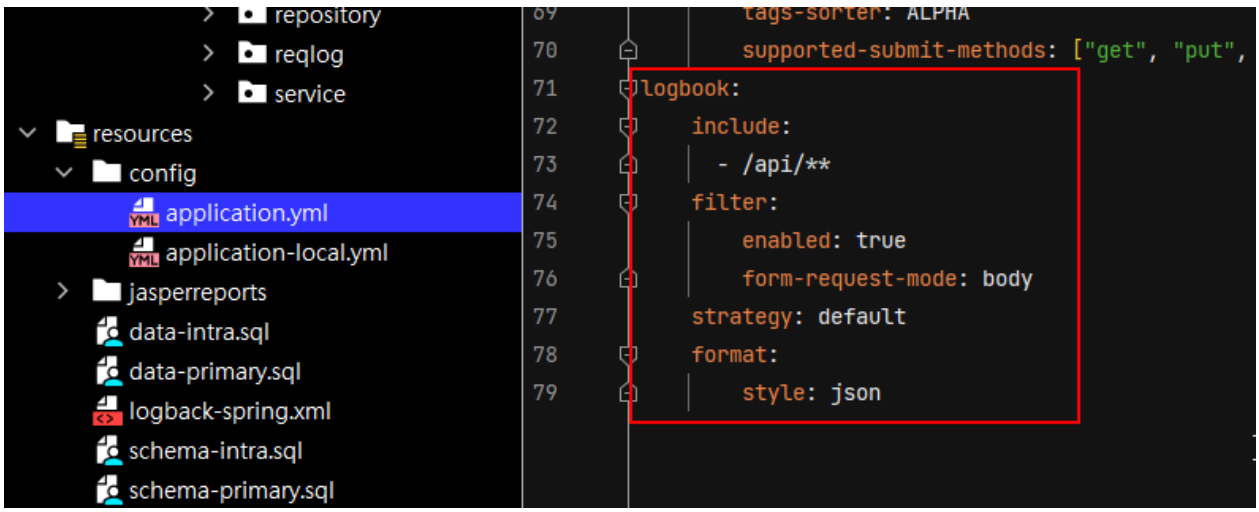
- com.fubonlife.boot.cors.CorsConfiguration.java
- com.fubonlife.boot.cors.CORSFilter.java

啟用 Logbook

> *是一種專門用來記錄 HTTP 請求和響應的日誌*

啟用方式





相關的程式

程式碼位置

- com.fubonlife.mytest.common.repository.primary.LogTraceRepository.java
- com.fubonlife.mytest.common.service.LogTraceService.java