手把手邁向HIS微服務之路（下）

口述／孫培然　彙整／CIO編輯室

上期我們談到在我們要開發一個系統時，會先跟使用者訪談需求後，再由敏捷軟體開發小組撰寫一份系統設計書，系統設計書裡面的大綱有七大重點如下：

1. UI & Event Binding
2. ViewModel
3. View Service
4. WebAPI Controller
5. Mode Service
6. Stored Procedure
7. Interactive

首先要畫出 UI（User Interface），每個 UI 都要有 Event Binding，也就是某個按鈕按下去要做什麼動作。所以每一個按鈕都要定義一個事件名稱<圖1>，事件名稱前面一定要加一個 on，如 onAddClick 就是定義一個新增按鈕的選點事件。

序	事件名稱
1	onAddClick($event)
2	onSaveClick($event)
3	onColumnChange($event)
4	onCellEditComplete($event)
5	onMaintainFocusout($event)
6	onRowEditClick($event)
7	onRowRemoveClick($event)

<圖1> UI&Event Binding

我們跟使用者討論完把 UI 畫出來以後，就要開始寫每一個事件裡面所要產生的東西。產生出來的畫面可能會呈現很多內容，所以接下來就要把相關的 ViewModel 設計出來，如<圖2>所示，這就是一個前、後端共享的 ViewModel 類別。

第三個就是當 Property/Event 對應的按鈕按下去以後，會呼叫前端的哪一個 View Service，Input 及 Output 是什麼，都要列很清楚如<圖3>。

然後這些功能會相對到後端的 WebAPI Controller，如<圖4>，也就是要呼叫哪一個 API。如 API 的 Method 是要 GET 還是 POST，是 PUT 還是 DELETE？URL 裡面是要呼叫的 API ，如「getAppStoreInfo」，以及 Input 及 Output 的內容。

WebAPI Controller 的下一個階段，則是要呼叫哪一個 View Model，也就是要呼叫Model Service，如<圖5>。要特別注意的是，從 View Service、WebAPI 到 Model，甚至到 Stored Procedure，如<圖6>，相關名稱除非有特殊性或差異性，不然功能名稱都要力求統一。

串聯整個系統的資料溝通機制，是透過底層共用的 ViewModel 來溝通，如<圖7>，所以從前端到後端，再到資料庫的 Stored Procedure，中間資料的傳遞格式都是用 JSON 的方式來做傳遞，完整對應到前、後端的 MVC 架構。這是每一個系統設計師所必須要規劃出來的架構，有了這個架構以後，就已經把單體拆成微服務架構了，而且整個程式碼、API 也都已經分類分好了，它們之間透過 ViewModel 來互相溝通，這系統設計書的七項大綱，是根據近幾年來，在建立微服務架構所累積的實

AppPageOption

Name	DataType
+ pageId	: number
+ pageTitle	: string
+ icon?	: string
+ templateUrl	: string
+ versionId	: string
+ fontSize?	: number
+ developer	: Developer
+ systemUser	: number
+ isUsed	: boolean
+ entityState	: string

AppPageInfo

Name	DataType
+ appPageOption	: AppPageOption
+ remoteName	: string
+ moduleName	: string

Developer

Name	DataType
+ developerNo	: number
+ developerCode?	: string
+ developerName?	: string

<圖2> ViewModel

Page	Description	Property / Event	View Service	Input	Output
應用頁面建檔	畫面初始設定	ngOnInit()	getDevelopers()	N/A	Developer[]
			getAppPageInfos()	N/A	AppPageInfo[]
	按下新增按鈕	onAddClick($event)	getNewPageId()	N/A	pageId
	按下儲存按鈕	onSaveClick($event)	setAppPageInfos()	AppPageInfo[]	true, false
	輸入頁面代碼	onPageIdFocusout($event)	checkPageIdUsed()	pageId	true, false
	輸入網頁路徑	onTemplateUrlFocusout($event)	checkTemplateUrlUsed()	pageUrl	true, false
	按下刪除按鈕	onRowRemoveClick($event)	checkAppPageUsed()	pageId	string

<圖3> View Service

Description	Method	URL(/webApi/appPage/)	Input	Output
取得所有開發人員	GET	getDevelopers()	N/A	Developer[]
取得所有應用頁面	GET	getAppPageInfos()	N/A	AppPageInfo[]
儲存所有應用頁面的更動	POST	setAppPageInfos()	AppPageInfo[]	true, false
檢查特定應用頁面是否被使用	GET	checkAppPageUsed()	pageId	string

Description	Method	URL(/webApi/appStore/)	Input	Output
取得特定應用程式	GET	getAppStoreInfo()	appId	AppStoreInfo
取得所有應用程式	GET	getAppStoreInfos()	N/A	AppStoreInfo[]
取得所有 Icon 圖片名稱	GET	getAppStoreIcons()	N/A	string[]
檢查特定應用程式是否被使用	GET	checkAppStoreUsed()	appId	string
儲存特定應用程式的更動	POST	setAppStoreInfo()	AppStoreInfo	true, false
刪除特定應用程式	DELETE	removeAppStore()	appId	true, false

<圖4> WebAPI Controller

Description	Service Function	Input	Output
取得所有開發人員	getDevelopers()	N/A	Developer[]
取得所有應用頁面	getAppPageInfos()	N/A	AppPageInfo[]
儲存所有應用頁面的更動	setAppPageInfos()	AppPageInfo[]	true, false
檢查特定應用頁面是否被使用	checkAppPageUsed()	pageId	string

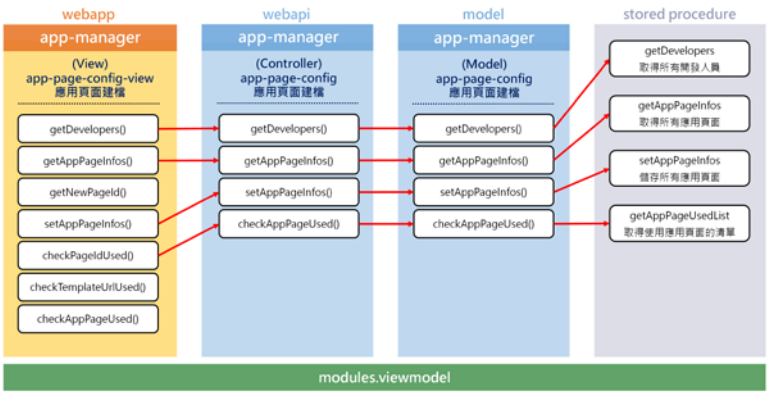
Description	Service Function	Input	Output
取得特定應用程式	getAppStoreInfo()	appId	AppStoreInfo
取得所有應用程式	getAppStoreInfos()	N/A	AppStoreInfo[]
取得所有 Icon 圖片名稱	getAppStoreIcons()	N/A	string[]
檢查特定應用程式是否被使用	checkAppStoreUsed()	appId	string
儲存特定應用程式的更動	setAppStoreInfo()	AppStoreInfo	true, false
刪除特定應用程式	removeAppStore()	appId	true, false

<圖5> Model Service

Description	Service Function	Input	Output
取得所有開發人員	getDevelopers()	N/A	Developer[]
取得所有應用頁面	getAppPageInfos()	N/A	AppPageInfo[]
儲存所有應用頁面的更動	setAppPageInfos()	AppPageInfo[]	true, false
取得使用應用頁面的清單	getAppPageUsedList()	pageId	AppPageUsedItem[]

Description	Service Function	Input	Output
取得所有應用程式	getAppStoreInfos()	N/A	AppStoreInfo[]
取得所有 Icon 圖片名稱	getAppStoreIcons()	N/A	string[]
取得使用應用程式的清單	getAppStoreUsedList()	appId	AppUsedMember[]
儲存特定應用程式的更動	setAppStoreInfo()	AppStoreInfo	true, false
刪除特定應用程式	removeAppStore()	appId	true, false

<圖6> Stored Procedure



<圖7> Interactive

務經驗，而整理出來的方法論架構，期望，對於想導入微服務架構的醫院或企業有所助益。

打造微服務要避免迷失

但俗話說：「水能載舟，亦能覆舟」，如果微服務架構用的不好，坦白講可能也會出問題。想要用的好，打造微服務架構時就應該要注意一些細節。

專欄專家孫培然博士，目前是擔任私立醫療院所協會醫院資訊暨智慧醫療發展促進會會長，也任職於中國醫藥大學附設醫院資訊室副主任，正進行體系HIS優化再造工程，是一位勇於接受挑戰及洞燭先機採用新技術的領導者，亦曾在中山醫大附設醫院任職資訊室主任時，回收委外HIS系統，重新改造的成功案例。

其實我現在一直在推廣「導入微服務重構現代化 HIS 架構」，期望台灣醫療數位轉型順利成功，但現在擔心的一點就是，大家雖然已經知道微服務很好，卻還是用舊有的傳統思維架構去看微服務，只是想要用新瓶來裝舊酒，最後無法執行時，卻說是新瓶子沒有用，就會更慘，因為單體跟微服務架構的觀念是完全背道而馳，所以還需要更深入的去談如何避免微服務的迷思，否則到時候問題會更大。