**知識網格元件編寫**

Ver. 0.4 2019.03.28

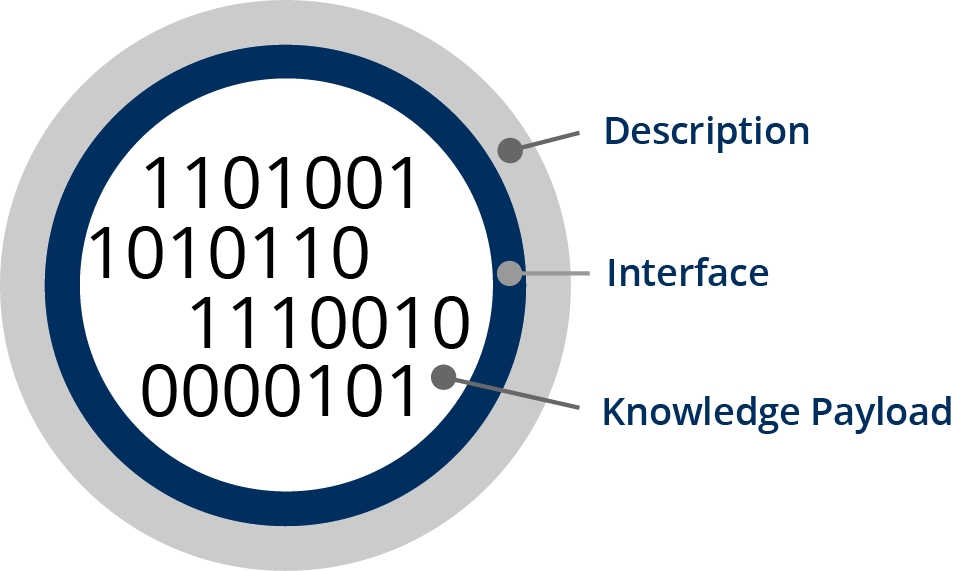
wucy1 @ github

說明：本文基於KGrid Activator 1.0.5版撰寫，新版(1.0.x版)的知識網格在知識元件的結構上與舊版(0.x版)有極大的差異，舊版的知識元件為單一檔案，新版的知識元件為一具有特定結構的資料夾，而其結構相較於1.0.3版以前適用的版本又做了一次簡化，元件內程式碼(payload)的編譯工作依然是由adapter執行，但activator本體會內含javascript的adapter以便編譯元件內由javascript寫成的主程式碼(payload)。

作業環境：以下說明是基於在windows 10上執行知識網格的activator，但是在Ｍac或Linux上應該也可以直接運作。

如何編寫知識網格：

一、知識元件的結構概念



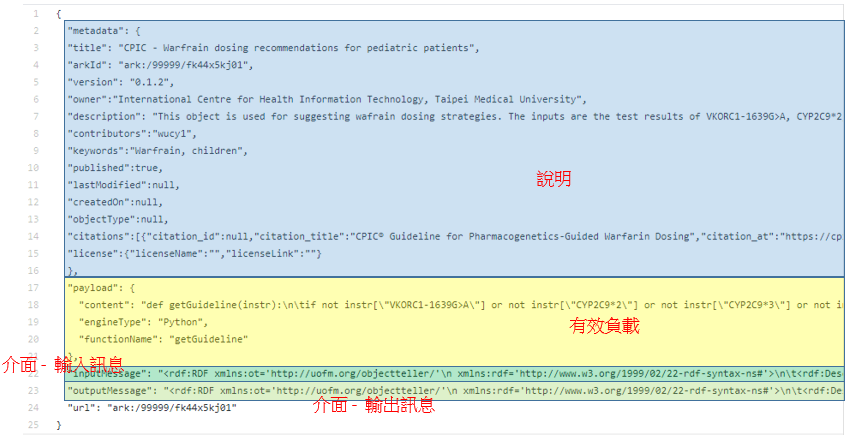
圖一、知識元件的組成(註一)

知識元件以模組化、計算機可處理的形式來儲存知識，由知識有效負載(payload/編碼成任何計算機語言或可計算格式)、詳細的描述(description)及介面(interface/含輸出入訊息)三部分組成(見圖一)；將知識可計算化涉及將健康知識表示成生產規則、網路圖、結構化判斷、方程式以及機率的技術，一旦這些技術應用於將知識可執行化與可計算化，便產生知識有效負載，被包裹於知識元件之中；知識元件有效負載的例子包括預測模型、可計算化的指引、計量問卷、可計算化的表型以及臨床決策支援規則等；知識元件以模組化、可計算機處理的形式表示知識資源，將其外部化並提供服務；關於知識元件結構的概念如上所述，但是在具體的編寫方式上，在新舊版本的知識網格中有很大的不同。

二、舊版知識元件格式

1. 0.x版：

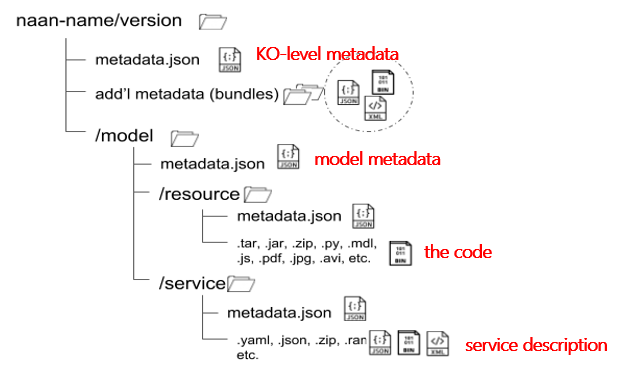
0.x版的元件撰寫方式，是將上述三個部分整個以JSON格式打包到一個檔案中，以"CPIC - Warfrain dosing recommendations for pediatric patients"元件為例，各部分呈現如圖二：



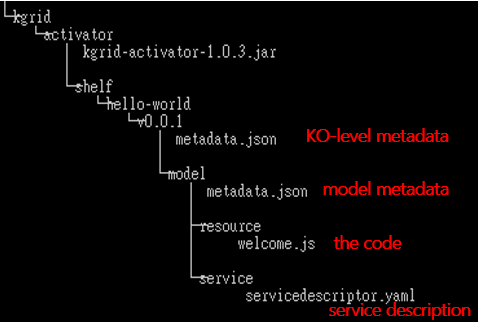
圖二、0.x版知識元件格式

2. 1.0.x版(1.0.3版以前)：

1.0.x版的元件撰寫方式，是將上述三個部分，分別置於不同的檔案及目錄，1.0.3版以前整個元件的架構，如圖三所示；一個知識元件的要素有四項，即元件層級的metadata(KO-level metadata)、模型層級metadata(model metadata)、服務描述(service description)以及主程式碼(the code)；圖四所示為一包含四要素的最小知識元件範例，關於這些版本的細節，若有需要深入了解，可參考知識網格元件編寫\_v0.2。



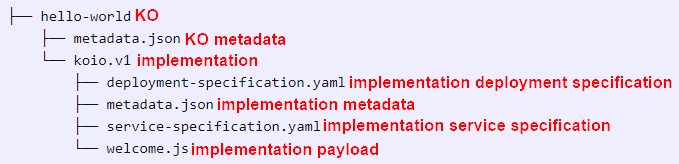
圖三、1.0.3版知識網格元件架構(註二)



圖四、1.0.3版知識網格元件範例

三、新版(1.0.4版以後)知識元件架構

新版(以1.0.5版為例)的元件新增了執行版本(implementation)層級，但是簡化了整體檔案結構，使得架構扁平化，提升易讀性。以官方範例元件hello-world為例，這個元件(KO)可以包含koio.v1以及其他執行版本(implementation)，在元件層級的metadata.json(KO metadata)內可以指定可執行的版本，每一個執行版本另有一個對應的metadata.json(implementation metadata)，可以指定服務規範(implementation service specification)、佈署規範(implementation deployment specification)以及有效附載(implementation payload)，服務規範描述基於這個元件執行版本的服務如何被提供，佈署規範指定服務端點的佈署細節，有效附載提供可計算化知識的主程式。



圖五、1.0.5版知識網格元件架構(註三)

四、元件層級的metadata(KO metadata)

以下各點以官方所附範例元件hello-world為例說明，打開hello-world目錄下的metadata.json，可以看到如下資訊：



圖六：元件的metadata.json(註三)

其中hasImplementation是最重要的參數，用以指定可以被執行的執行版本。

五、執行版本的metadata(implementation metadata)

在每一個下執行版本下，另有metadata.json檔案，以提供活化器所需要的資訊，如前所述，主要是指定服務規範(hasServiceSpecification)、佈署規範(hasDeploymentSpecification)以及有效附載(hasPayload)三個重要檔案。



圖七、執行版本的metadata.json(註三)

六、執行版本服務規範(implementation service specification)

元件各執行版本所提供的服務則由service-specification.yaml這個檔案來進一步定義，這個檔案的功能涵蓋圖一所提到的介面。檔案各部分內容概略功能標記如圖八，各項參數具體功能請參考註四。

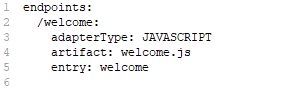




圖八、執行版本服務規範檔內容

七、執行版本佈署規範(implementation deployment specification)

元件各執行版本的佈署細節則由deployment-specification.yaml這個檔案來進一步定義，檔案內容如圖九所示，主要是定義服務端點的adapter類型(adapterType)、程式(artifact)以及入口(entry)。



圖九、執行版本佈署規範檔內容

八、執行版本有效附載(implementation payload)

範例元件執行版本的有效附載主程式碼welcome.js，包含一welcome函式，功能為接收一json格式的inputs，擷取其中的name參數，然後返回歡迎訊息。然後再由activator根據服務描述檔內的定義，以json格式回傳資訊。



圖十、有效附載主程式碼內容

九、知識元件的執行

範例元件可以在終端介面以curl指令測試：

curl -X POST -H "Content-Type:application/json" -d "{\"name\": \"Fred Flintstone\"}" <http://localhost:8080/hello/world/koio.v1/welcome>

以上指令意義為：

1. 呼叫服務端點<http://localhost:8080/hello/world/koio.v1/welcome>

以POST方式，透過Activator呼叫，每個元件的服務端點由前述服務規範檔定義

1. 向此端點以json格式送出查詢資料

此資料的格式跟必要欄位，也是由服務規範檔定義

1. 若查詢資料有誤，會依服務規範檔的定義傳回錯誤訊息，若正確，則會將資料傳送給有效附載主程式檔，等待主程式處理後，傳回訊息如下：



圖十一、範例元件呼叫結果

註一：<http://kgrid.org/tools.html#object>

註二：原參考網頁內容已變更

註三：<https://kgrid.org/getting-started/developing-kos/>

註四：<https://swagger.io/docs/specification/about/>