插件（Plugins）

原文：ENG-10-Plugins.md

插件可協助使用者建構複雜應用程式。在 Drogon 中，所有插件皆依設定檔建置並安裝至應用程式。Drogon 的插件為單一實例，使用者可自訂任意功能。

當 Drogon 執行 run() 介面時，會依設定檔逐一實體化每個插件並呼叫其 `initAndStart()` 介面。

設定方式

插件設定於設定檔，例如：

```
    "plugins": [
        {
        //name: 插件類別名稱
        "name": "DataDictionary",
        //dependencies: 依賴的其他插件，可省略
        "dependencies": [],
        //config: 插件設定，為初始化參數 json 物件，可省略
        "config": {
        }
        }],
```

每個插件有三個設定：

- name：插件類別名稱（含命名空間），框架會依此建立插件實例。若註解此項，插件即停用。
- dependencies：依賴其他插件的名稱列表，框架會依依賴順序建立與初始化所有插件，並於程式結束時反向關閉與銷毀。禁止循環依賴，若偵測到循環依賴，Drogon 會報錯並退出。若註解此項，依賴列表為空。
- config：初始化插件的 json 物件，會傳入插件的 `initAndStart()` 介面。若註解此項，則傳入空物件。

定義方式

自訂插件需繼承 drogon::Plugin 類別模板，模板參數為插件型別，定義如下：

```
class DataDictionary : public drogon::Plugin<DataDictionary>
{
public:
    virtual void initAndStart(const Json::Value &config) override;
    virtual void shutdown() override;
    ...
};
```

可用 drogon_ctl 指令建立插件原始碼檔案：

```
drogon_ctl create plugin <[namespace::]class_name>
```

取得實例

插件實例由 drogon 建立，使用者可透過以下介面取得插件實例：

```
template <typename T> T *getPlugin();
```

或

```
PluginBase *getPlugin(const std::string &name);
```

一般建議用第一種方式。例如上述 DataDictionary 插件可如下取得：

```
auto *pluginPtr=app().getPlugin<DataDictionary>();
```

建議於框架 run() 介面呼叫後再取得插件，否則僅會取得未初始化的插件實例（雖不一定出錯，但請確保初始化後再使用）。由於插件依賴順序初始化，在 `initAndStart()` 介面中取得其他插件實例亦無問題。

生命週期

所有插件於框架 run() 介面初始化，應用程式結束時銷毀。因此插件生命週期幾乎與應用程式一致，getPlugin() 介面無需回傳智慧指標。

下一步: 設定檔