中介層與過濾器

原文：ENG-05-Middleware-and-Filter.md

在 HttpController 的範例中，getInfo 方法應在回傳使用者資訊前先檢查是否已登入。雖然可直接在 getInfo 方法中撰寫此邏輯，但檢查登入屬於多數介面都會用到的通用邏輯，應獨立抽出並於 handler 執行前配置，這正是 filter 的用途。

drogon 的中介層（middleware）採用洋蔥模型。框架完成 URL 路徑比對後，會依序呼叫該路徑註冊的中介層。每個中介層可選擇攔截或放行請求，並加入前置與後置處理邏輯。

若中介層攔截請求，則不會進入洋蔥內層，對應的 handler 也不會被呼叫，但仍會執行外層中介層的後置邏輯。

filter 實際上是省略後置操作的 middleware。註冊路徑時可同時使用 middleware 與 filter。

內建中介層／過濾器

drogon 內建常用 filter 如下：

- `drogon::IntranetIpFilter`：僅允許內網 IP 的 HTTP 請求，否則回傳 404。
- `drogon::LocalHostFilter`：僅允許 127.0.0.1 或 ::1 的 HTTP 請求，否則回傳 404。

自訂中介層／過濾器

中介層定義

使用者可自訂 middleware，需繼承 `HttpMiddleware` 類別模板，模板型別為子類型。例如，若要讓部分路由支援跨域，可如下定義：

```
class MyMiddleware : public HttpMiddleware<MyMiddleware>
{
public:
    MyMiddleware(){};  // 不可省略建構子

    void invoke(const HttpRequestPtr &req,
                MiddlewareNextCallback &&nextCb,
                MiddlewareCallback &&mcb) override
    {
        const std::string &origin = req->getHeader("origin");
        if (origin.find("www.some-evil-place.com") != std::string::npos)
        {
            // 直接攔截
            mcb(HttpResponse::newNotFoundResponse(req));
            return;
        }
        // 執行進入下一層前的處理
        nextCb([mcb = std::move(mcb)](const HttpResponsePtr &resp) {
            // 執行回傳後的處理
            resp->addHeader("Access-Control-Allow-Origin", origin);
```

```
            resp->addHeader("Access-Control-Allow-Credentials","true");
            mcb(resp);
        });
    }
};
```

需覆寫父類別的 `invoke` 虛擬函式以實作 filter 邏輯。

此虛擬函式有三個參數：

- **req**：HTTP 請求物件；
- **nextCb**：進入洋蔥內層的 callback，呼叫即執行下一個 middleware 或最終 handler。呼叫時可傳入另一函式，該函式於回傳時執行，並接收內層回傳的 HttpResponsePtr。
- **mcb**：回到洋蔥外層的 callback，呼叫即回到外層。若略過 nextCb 僅呼叫 mcb，表示攔截請求直接回外層。

過濾器定義

使用者亦可自訂 filter，需繼承 HttpFilter 類別模板，模板型別為子類型。例如，若要建立 LoginFilter，可如下定義：

```
class LoginFilter:public drogon::HttpFilter<LoginFilter>
{
public:
    void doFilter(const HttpRequestPtr &req,
                  FilterCallback &&fcb,
                  FilterChainCallback &&fccb) override ;
};
```

可用 `drogon_ctl` 指令建立 filter，詳見 drogon_ctl。

需覆寫父類別的 doFilter 虛擬函式以實作 filter 邏輯。

此虛擬函式有三個參數：

- **req**：HTTP 請求物件；
- **fcb**：filter callback，型別為 void (HttpResponsePtr)，當判斷請求不合法時，透過此 callback 回傳特定回應給瀏覽器；
- **fccb**：filter chain callback，型別為 void ()，當判斷請求合法時，通知 drogon 執行下一個 filter 或最終 handler。

具體實作可參考 drogon 內建 filter。

中介層／過濾器註冊

註冊 middleware／filter 時，通常會搭配 controller 註冊。前述巨集（PATH_ADD、METHOD_ADD 等）可於最後加上 middleware／filter 名稱。例如將前述 getInfo 方法註冊行改為：

```
```

```
METHOD_ADD(User::getInfo,"/{1}/info?token=
{2}",Get,"LoginFilter","MyMiddleware");
```

路徑比對成功後，僅當下列條件皆成立才會呼叫 getInfo 方法：

1. 請求必須為 HTTP Get；
2. 請求方必須已登入；

可見 middleware／filter 註冊與配置非常簡單。controller 原始檔註冊 middleware 時無需 include 其標頭檔，middleware 與 controller 完全解耦。

> **注意：若 middleware／filter 定義於命名空間，註冊時必須完整寫出命名空間。**

下一步: 視圖