會話（Session）

原文：ENG-07-Session.md

`Session` 是 Web 應用的重要概念，用於在伺服器端保存用戶端狀態，通常搭配瀏覽器的 `cookie` 使用，drogon 也有內建 session 支援。drogon 預設**關閉** session，可透過下列介面開啟或關閉：

```
void disableSession();
void enableSession(const size_t timeout=0, Cookie::SameSite
sameSite=Cookie::SameSite::kNull);
```

上述方法皆透過 `HttpAppFramework` 單例呼叫。timeout 參數為 session 失效時間（秒），預設 1200 秒（20 分鐘未存取即失效），設為 0 則 session 保留至應用結束；sameSite 參數可設定 Set-Cookie HTTP 回應標頭的 SameSite 屬性。

開啟 session 前請確認用戶端支援 cookie，否則 drogon 會為每個無 `SessionID` cookie 的請求建立新 session，造成記憶體與運算資源浪費。

Session 物件

drogon 的 session 物件型別為 `drogon::Session`，用法類似 `HttpViewData`，可用關鍵字存取任意型別物件，支援並發讀寫，詳見 Session 類別說明。

drogon 框架會將 session 物件傳給 `HttpRequest` 物件，使用者可透過下列 `HttpRequest` 介面取得 Session 物件：

```
SessionPtr session() const;
```

此介面回傳 Session 物件智慧指標，可用來存取各種物件。

Session 範例

以下新增一個需 session 支援的功能：限制用戶存取頻率，若 10 秒內重複存取則回傳錯誤，否則回傳 ok。需在 session 記錄上次存取時間，並與本次存取時間比對。

建立一個 Filter 實作此功能，假設類別名為 TimeFilter，實作如下：

```
#include "TimeFilter.h"
#include <trantor/utils/Date.h>
#include <trantor/utils/Logger.h>
#define VDate "visitDate"
void TimeFilter::doFilter(const HttpRequestPtr &req,
                          FilterCallback &&cb,
                          FilterChainCallback &&ccb)
{
```

```
    trantor::Date now=trantor::Date::date();
    LOG_TRACE<<"";
    if(req->session()->find(VDate))
    {
        auto lastDate=req->session()->get<trantor::Date>(VDate);
        LOG_TRACE<<"last:"<<lastDate.toFormattedString(false);
        req->session()->modify<trantor::Date>(VDate,
                                    [now](trantor::Date &vdate) {
                                        vdate = now;
                                    });
        LOG_TRACE<<"update visitDate";
        if(now>lastDate.after(10))
        {
            //10 秒後可再次存取
            ccb();
            return;
        }
        else
        {
            Json::Value json;
            json["result"]="error";
            json["message"]="存取間隔需至少 10 秒";
            auto res=HttpResponse::newHttpJsonResponse(json);
            cb(res);
            return;
        }
    }
    LOG_TRACE<<"首次存取，插入 visitDate";
    req->session()->insert(VDate,now);
    ccb();
}
```

接著於 `/slow` 路徑註冊 lambda 並掛上 TimeFilter，程式如下：

```
drogon::HttpAppFramework::instance()
            .registerHandler("/slow",
                             [=](const HttpRequestPtr &req,
                                 std::function<void (const HttpResponsePtr
&)> &&callback)
                             {
                                 Json::Value json;
                                 json["result"]="ok";
                                 auto
resp=HttpResponse::newHttpJsonResponse(json);
                                 callback(resp);
                             },
                             {Get,"TimeFilter"});
```

呼叫框架介面開啟 session：

```
drogon::HttpAppFramework::instance().enableSession(1200);
```

以 CMake 重新編譯專案，執行目標程式 webapp，即可於瀏覽器測試效果。

下一步: 資料庫