資料庫 - FastDbClient

原文：ENG-08-4-Database-FastDbClient.md

如其名，FastDbClient 相較於一般 DbClient 具備更高效能。不同於 DbClient 擁有獨立事件迴圈，FastDbClient 與網路 IO 執行緒及主執行緒共用事件迴圈，使其內部實作可採無鎖（lock-free）模式，效能更佳。

測試顯示，在極高負載下，FastDbClient 效能比 DbClient 高出 10% 至 20%。

建立與取得

FastDbClient 必須由框架自動依設定檔建立，或透過 app.createDbClient() 介面建立：

設定檔中 db_client 選項的子選項 `is_fast` 用來指定是否為 FastDbClient。或可呼叫 app.createDbClient() 方法，最後一個參數設為 true 即建立 FastDbClient。

框架會為每個 IO 事件迴圈及主事件迴圈分別建立 FastDbClient，每個 FastDbClient 內部管理多個資料庫連線。IO 事件迴圈數由框架的 "threads_num" 選項控制，通常設為主機 CPU 核心數。每個事件迴圈的 DB 連線數由 DB client 的 "connection_number" 選項決定。詳情請參考設定檔。因此 FastDbClient 持有的總連線數為 `(threads_num+1) * connection_number`。

取得 FastDbClient 的介面與一般 DbClient 類似，如下：

```
orm::DbClientPtr getFastDbClient(const std::string &name = "default");
/// 使用 drogon::app().getFastDbClient("clientName") 取得 FastDbClient 物件。
```

需特別注意，因 FastDbClient 的特殊性，必須在 IO 事件迴圈執行緒或主執行緒呼叫上述介面，才能取得正確的智慧指標。在其他執行緒僅會取得空指標，無法使用。

使用方式

FastDbClient 的使用方式幾乎與一般 DbClient 相同，但有以下限制：

- 取得與使用 FastDbClient 必須在框架的 IO 事件迴圈執行緒或主執行緒，否則會有不可預期錯誤（因破壞 lock-free 條件）。幸好大多數應用程式邏輯都在 IO 執行緒，如各控制器處理函式、過濾器 filter 函式等。FastDbClient 介面的各 callback 也都在目前 IO 執行緒，可安全巢狀使用。
- 絕不可使用 FastDbClient 的阻塞介面，因該介面會阻塞目前執行緒，而該執行緒同時負責此物件的資料庫 IO，將導致永久阻塞，使用者無法取得結果。
- FastDbClient 的同步交易建立介面可能會阻塞（當所有連線皆忙碌），因此同步交易建立介面會直接回傳空指標。若需在 FastDbClient 上使用交易，請改用非同步交易建立介面。
- 使用 FastDbClient 建立 Orm Mapper 物件後，也僅能使用 Mapper 的非同步非阻塞介面。

下一步: 自動批次模式