設定檔（Configuration File）

原文：ENG-11-Configuration-File.md

可透過多種介面設定 DrogonAppFramework 實例的 Http 伺服器行為，但建議使用設定檔，原因如下：

- 設定檔可在執行時決定應用行為，較原始碼設定更方便彈性；
- 設定檔可讓主程式更簡潔；

因此建議開發者以設定檔配置各項參數。

只需在呼叫 run() 前執行 loadConfigFile() 介面即可載入設定檔，例如：

```
int main()
{
    drogon::app().loadConfigFile("config.json");
    drogon::app().run();
}
```

上述程式載入 `config.json` 設定檔並執行應用。監聽埠、日誌、資料庫等皆可由設定檔配置。事實上，這段程式碼可作為 Web 應用主程式。

設定檔細節

範例設定檔可見於原始碼目錄頂層的 config.example.json。使用 drogon_ctl create project 指令建立專案時，專案目錄也會有 config.json。通常只需修改此檔即可完成 Web 應用設定。

檔案格式為 `JSON`，支援註解。可用 /**/ 或 // 註解不需的項目。

註解掉的選項會採用預設值，預設值可見於設定檔註解。

支援格式

- json
- yaml（需安裝 yaml-cpp 套件）

SSL

ssl 選項用於設定 https 服務的 SSL 檔案：

```
  "ssl": {
    "cert": "../../trantor/trantor/tests/server.pem",
    "key": "../../trantor/trantor/tests/server.pem",
    "conf":[
      ["Options", "Compression"],
      ["min_protocol", "TLSv1.2"]
    ]
  }
```

`cert` 為憑證檔路徑，`key` 為私鑰檔路徑。若同檔案含憑證與私鑰，路徑可相同，格式為 PEM。

`conf` 為可選 SSL 參數，會直接傳給 SSL_CONF_cmd 以低階設定加密，選項需為一或兩元素陣列。

---

listeners

listeners 用於設定 Web 應用監聽器，為 JSON 陣列，每個物件代表一個監聽器：

```
"listeners": [
    {
      "address": "0.0.0.0",
      "port": 80,
      "https": false
    },
    {
      "address": "0.0.0.0",
      "port": 443,
      "https": true,
      "cert": "",
      "key": ""
    }
  ]
```

- `address`：字串型，監聽 IP，預設 "0.0.0.0"
- `port`：整數型，監聽埠，必填
- `https`：布林型，是否啟用 https，預設 false
- `cert`、`key`：https 時有效，憑證與私鑰路徑，預設空字串，表示用全域 ssl 設定

---

db_clients

db_clients 用於設定資料庫 client，為 JSON 陣列，每個物件代表一個 client：

```
  "db_clients":[
    {
      "name":"",
      "rdbms": "postgresql",
      "host": "127.0.0.1",
      "port": 5432,
      "dbname": "test",
      "user": "",
      "passwd": "",
      "is_fast": false,
      "connection_number": 1,
      "filename": ""
    }
  ]
```

- `name`：client 名稱，預設 "default"，多 client 時需不同
- `rdbms`：資料庫型別，支援 "postgresql"、"mysql"、"sqlite3"
- `host`：資料庫主機，預設 "localhost"
- `port`：資料庫埠號
- `dbname`：資料庫名稱
- `user`：使用者名稱
- `passwd`：密碼
- `is_fast`：是否為 FastDbClient，預設 false
- `connection_number`：連線數，至少 1，預設 1，影響併發效能。is_fast 為 true 時為每個事件迴圈的連線數，否則為總連線數
- `filename`：sqlite3 資料庫檔名

---

threads_num

app 子選項，整數型，預設 1，表示 IO 執行緒數，影響網路併發。此值不宜過大，建議設為預期網路 IO 佔用的處理器數。設為 0 時，執行緒數等於硬體核心數。

```
"threads_num": 16,
```

Session

Session 相關選項亦為 app 子選項，控制是否啟用 session 及逾時：

```
"enable_session": true,
"session_timeout": 1200,
```

- `enable_session`：是否啟用 session，預設 false。若 client 不支援 cookie，請設為 false，否則每次請求都會建立新 session，浪費資源
- `session_timeout`：session 逾時秒數，預設 0（永久有效），僅在啟用 session 時有效

---

document_root

app 子選項，字串型，表示 Http 根目錄對應的文件路徑，也是靜態檔案下載根目錄，預設 "./"（程式執行目錄）。

```
"document_root": "./",
```

---

upload_path

app 子選項，字串型，表示檔案上傳預設路徑，預設 "uploads"。若非以 `/`、`./`、`../` 開頭，且不為 `.` 或 `..`，則為相對於 document_root 的路徑，否則為絕對路徑或相對於目前目錄。

```
"upload_path":"uploads",
```

---

client_max_body_size

app 子選項，字串型，表示請求 body 最大總大小。可用 k、m、g、t（大小寫皆可）指定單位。

```
"client_max_body_size": "10M",
```

---

client_max_memory_body_size

app 子選項，字串型，表示緩衝區最大大小，超過則快取至檔案。可用 k、m、g、t 指定單位。

```
"client_max_memory_body_size": "50K"
```

---

file_types

app 子選項，字串陣列，預設如下，表示支援下載的靜態檔案類型。若請求副檔名不在此列表，框架回傳 404。

```
"file_types": [
      "gif",
      "png",
      "jpg",
      "js",
      "css",
      "html",
      "ico",
      "swf",
      "xap",
      "apk",
      "cur",
      "xml"
    ],
```

---

mime

app 子選項，字典型或字串陣列，宣告副檔名對應 MIME 類型（未被預設識別者）。此選項僅註冊 MIME，若副檔名不在 file_types，仍回傳 404。

```

```

```
"mime" : {
  "text/markdown": "md",
  "text/gemini": ["gmi", "gemini"]
}
```

連線數量控制

app 子選項有兩個：

```
"max_connections": 100000,
"max_connections_per_ip": 0,
```

- `max_connections`：最大同時連線數，預設 100000，達上限時新 TCP 連線會被拒絕
- `max_connections_per_ip`：單一 IP 最大連線數，預設 0（不限制）

日誌選項

app 子選項，JSON 物件，控制日誌輸出行為：

```
"log": {
      "log_path": "./",
      "logfile_base_name": "",
      "log_size_limit": 100000000,
      "log_level": "TRACE"
    },
```

- `log_path`：日誌檔儲存路徑，預設空字串（輸出至標準輸出）
- `logfile_base_name`：日誌檔名，預設空字串（即 drogon）
- `log_size_limit`：日誌檔大小上限（bytes），預設 100000000（100M），達上限時切換檔案
- `log_level`：最低日誌等級，預設 "DEBUG"，可選 "TRACE"↗"DEBUG"↗"INFO"↗"WARN"，TRACE 僅在 DEBUG 模式有效

> **注意：Drogon 的檔案日誌採非阻塞結構，可達百萬行/秒，效能可靠。**

應用程式控制

app 子選項有兩個：

```
    "run_as_daemon": false,
    "relaunch_on_error": false,
```

- `run_as_daemon`：是否以 daemon 方式在背景執行，預設 false
- `relaunch_on_error`：是否錯誤時自動重啟，預設 false

---

use_sendfile

app 子選項，布林型，表示傳送檔案時是否用 linux sendfile 系統呼叫，預設 true。sendfile 可提升效率並降低大檔案記憶體用量。

```
"use_sendfile": true,
```

> **注意：即使設為 true，實際是否用 sendfile 由框架最佳化策略決定。**

---

use_gzip

app 子選項，布林型，預設 true，表示 Http 回應 body 是否壓縮傳輸。啟用時：

- client 支援 gzip
- body 為文字型
- body 長度大於一定值

```
"use_gzip": true,
```

---

static_files_cache_time

app 子選項，整數型（秒），表示靜態檔案快取時間。重複請求於此時間內直接回傳快取內容，預設 5 秒，0 為永久快取（僅讀一次檔案，請慎用），負值為不快取。

```
"static_files_cache_time": 5,
```

---

simple_controllers_map

app 子選項，JSON 陣列，每個物件代表 Http 路徑對 HttpSimpleController 的映射。此設定為選用，詳見 HttpSimpleController。

```
"simple_controllers_map": [
      {
        "path": "/path/name",
        "controller": "controllerClassName",
        "http_methods": ["get","post"],
        "filters": ["FilterClassName"]
```

```
        }
    ],
```

- `path`：Http 路徑
- `controller`：HttpSimpleController 名稱
- `http_methods`：支援的 Http 方法陣列，未在列表者回傳 405
- `filters`：路徑上的過濾器列表，詳見 中介層與過濾器

閒置連線逾時控制

app 子選項，整數型（秒），預設 60。連線超過此時間未讀寫即強制斷線。

```
"idle_connection_timeout":60
```

動態視圖載入

app 子選項，控制動態視圖啟用與路徑，有兩個選項：

```
"load_dynamic_views":true,
"dynamic_views_path":["./views"],
```

- `load_dynamic_views`：布林型，預設 false。啟用時，框架會在視圖路徑動態編譯 .so 檔並載入，視圖檔變更時自動編譯與重載
- `dynamic_views_path`：字串陣列，動態視圖搜尋路徑。若非以 `/`、`./`、`../` 開頭，且不為 `.` 或 `..`，則為相對於 document_root 的路徑，否則為絕對路徑或相對於目前目錄

詳見 視圖

Server header field

app 子選項，設定所有回應的 server header 欄位，預設空字串。若空字串，框架自動產生 `Server: drogon/版本字串`。

```
"server_header_field": ""
```

Keepalive requests

keepalive_requests 設定 keep-alive 連線可處理的最大請求數，達上限即關閉連線，預設 0（不限制）。

```
"keepalive_requests": 0
```

Pipelining requests

pipelining_requests 設定 pipelining 緩衝區可快取的最大未處理請求數，達上限即關閉連線，預設 0（不限制）。詳見 rfc2616-8.1.1.2。

```
"pipelining_requests": 0
```

下一步: AOP 面向切面程式設計