# 3. Controller-簡介RESTful API

在網路上經常會看到 Spring Boot 是個容易設置 RESTful API 的架構，什麼是 RESTful API？先來看看 API，API 的全名是 Application Program Interface，也就是應用程式介面，簡單來說是一種溝通方式，**前端會發出請求（request），後端收到請求之後，會再給予回應（response）**。以生活化的例子來比喻，可以想成客人去餐廳吃飯，填好菜單後交給服務生，接著服務生便會通知廚師做菜，並在稍後送上料理。其中溝通的方式就是「服務生」與「菜單」。應用上前端能藉由 API 這個服務生進行點餐，後端收到菜單後會處理前端的點餐請求，最後請服務生送出料理。

再來要提到 HTTP Method，當使用者發出請求，意味對某種東西進行操作，比方取得產品、編輯產品。以「資源」來稱呼被操作的東西，而操作的方式則包含取得、編輯、新增、刪除等。HTTP 請求方法便是以這些操作方式為基礎，常見的有以下幾種，可以對應到資料庫的 CRUD 操作。

- GET：取得資源
- POST：新增資源
- PUT：覆蓋資源
- DELETE：刪除資源

是一種設計提供全球資訊網絡服務的軟體構建風格

回到 RESTful API，REST 的全稱為 Representational State Transfer，是一種設計風格，將網路上的東西都視為「資源」，並且有不同的操作方式。

一個完整的 RESTful API，包含請求方法與資源路徑。資源的表達方式有點像網址，會以「路徑」來稱呼。用網路書店舉例，一個商品頁面的網址為 `https://www.books.com.tw/products/E050035254` ，這就是資源的所在路徑。再加上各種不同的 HTTP Method，就可以組出 RESTful API。好比像是以下的例子：

- GET /products/E050035254
  取得編號為E050035254的商品
- GET /products
  取得所有商品
- POST /products
  新增商品
- PUT /products/E050035254
  覆蓋編號為E050035254的商品
- DELETE /products/E050035254
  刪除編號為E050035254的商品

如果不是 REST 風格，這段 API 的 url 會是這樣：

- POST/getProducts/E050035254
  取得編號為E050035254的商品
- POST/getAllProducts
  取得所有商品
- POST/createProducts
  新增商品
- POST/updateProducts/E050035254
  覆蓋編號為E050035254的商品
- POST/deleteProducts/E050035254
  刪除編號為E050035254的商品

差別在於 url 上的不同，如果統一都使用 POST 方法，url 就必須定義要完成的事情，然而 REST 風格則是用 HTTP method 來定義想做的操作。

回到原本的 `TestController` ，我們在 `TestController` 中寫下如下的程式碼，其中 `@RequestMapping` 定義了 url，雖然這邊沒有說要怎麼定義 HTTP Method，不過上一章我們是使用瀏覽器去直接進行測試，表示是使用 get 方法：

```
package com.example.demospringboot.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class TestController {
    @RequestMapping("/")
    @ResponseBody
    public String hello() {
       return "Hello Spring Boot";
    }
}
```

那如果不是使用 get 方法，該怎麼去打到做好的 RESTful API，這邊提供兩個方法來做，一個是上網直接安裝 postman：


另一個是使用 Spring Boot 的 swagger。postman 的安裝就直接下載官網上的執行檔執行即可，swagger 的話要先在 `pom.xml` 加入以下的 `<dependency>`

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.9.2</version>
</dependency>

<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.9.2</version>
</dependency>
```

有加新東西進pom記得加alt+F5更新套件

然後建了一個 `configuration package` ，此時專案架構會是以下的樣子：

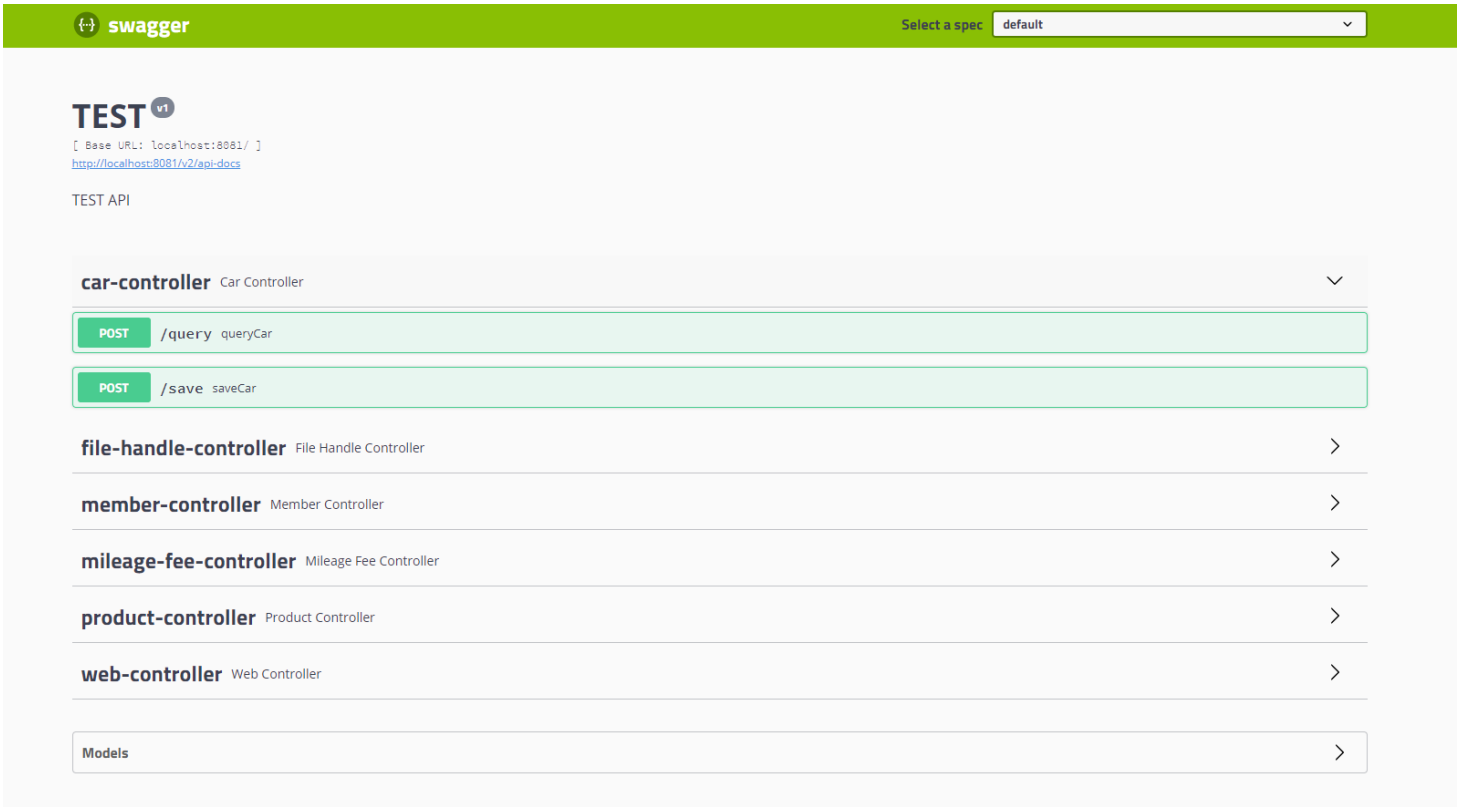
```
|--com.example.demospringboot
   |--DemospringbootApplication.java
|--com.example.demospringboot.configuration
   |--SwaggerConfig.java // 新增的檔案
|--com.example.demospringboot.controller
   |--TestController.java
```

在 `SwaggerConfig.java` 寫出以下的程式碼：

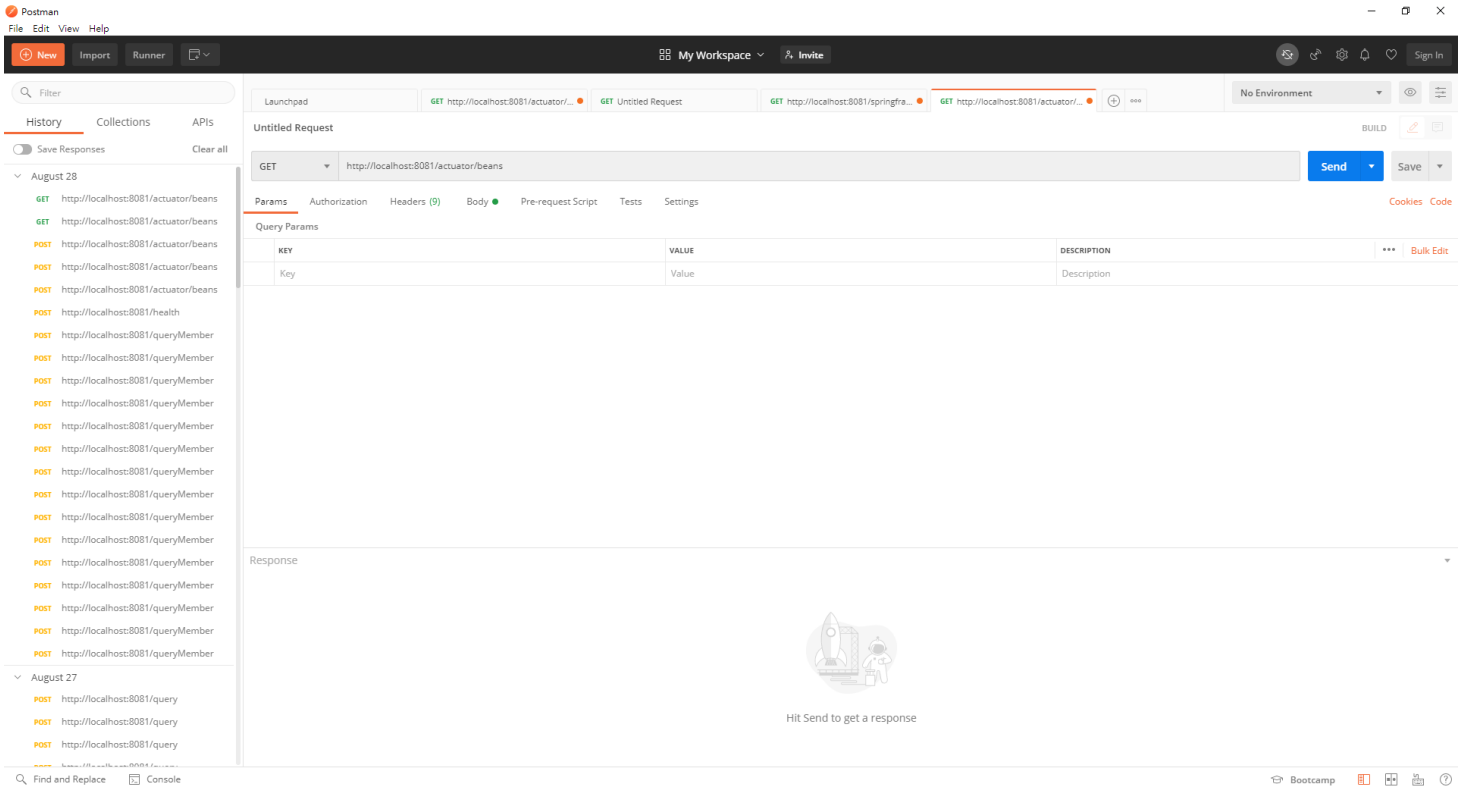
```java
@Configuration
@EnableSwagger2
public class SwaggerConfig {
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .apiInfo(apiInfo())
            .ignoredParameterTypes(Errors.class)
            .select()
            .apis(RequestHandlerSelectors.basePackage("com.example.demo"))
            .paths(PathSelectors.any())
            .build();
    }

    private ApiInfo apiInfo() {
        return new ApiInfoBuilder()
            .title("TEST")
            .description("TEST API")
            .version("v1")
            .build();
    }
}
```

路徑若跟講義不同記得替換

這樣在專案啟動時，就可使用 swagger 來測試自己做的 RESTful API，輸入網址 `http://localhost:8080/swagger-ui.html` 會有以下的介面：




而下面則是 POSTMAN 的介面：




# 參考

https://www.postman.com/

https://medium.com/chikuwa-tech-study/spring-boot-第2課-restful-api介紹-955f776da32d