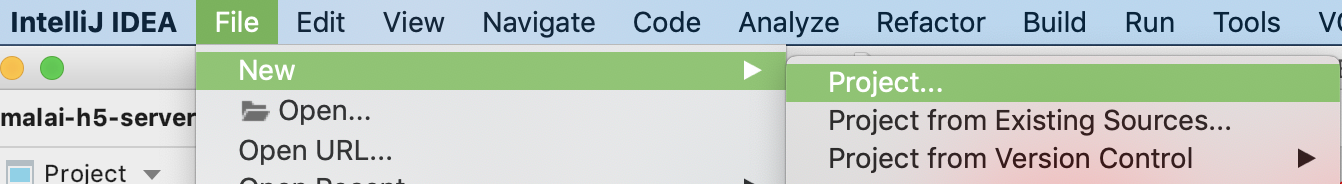
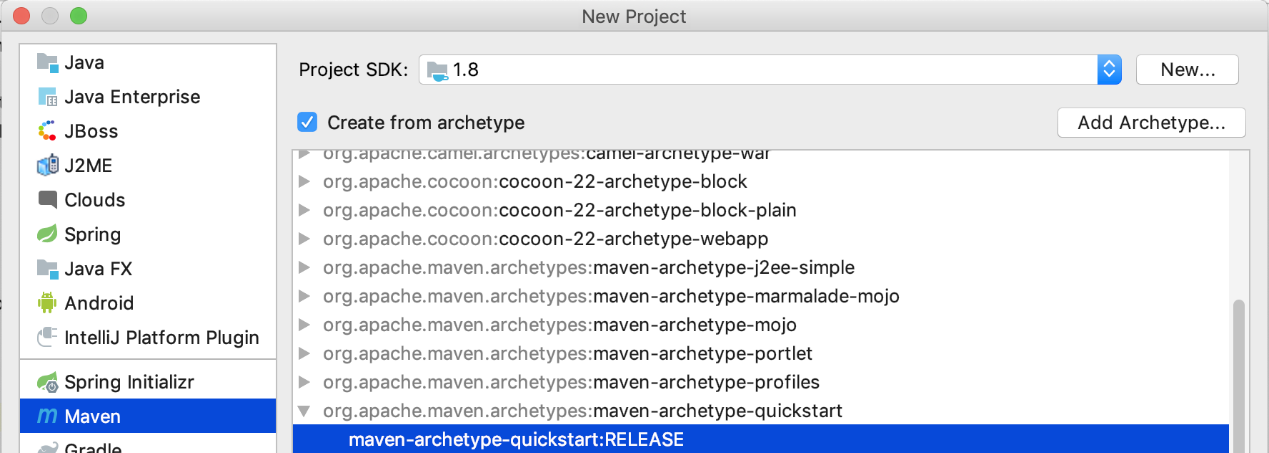


* 第 1 個：DAO（Data Access Object）數據訪問對象，它是一個面向對象的數據庫接口，負責持久層的操作，為業務層提供接口，主要用來封裝對數據庫的訪問，常見操作無外乎 CURD。我們也可以認為一個 DAO 對應一個 POJO 的對象，它位於業務邏輯與數據庫資源中間，可以結合 PO 對數據庫進行相關的操作。
* 第 2 個：PO（Persistent Object）持久層對象，它是由一組屬性和屬性的get和set方法組成，最簡單的PO 就是對應數據庫中某個表中的一條記錄（也就是說，我們可以將數據庫表中的一條記錄理解為一個持久層對象），多個記錄可以用PO 的集合，PO 中應該不包含任何對數據庫的操作。 PO 的屬性是跟數據庫表的字段一一對應的，此外 PO 對象需要實現序列化接口。
* 第 3 個：BO（Business Object）業務層對象，是簡單的真實世界的軟件抽象，通常位於中間層。 BO 的主要作用是把業務邏輯封裝為一個對象，這個對象可以包括一個或多個其它的對象。舉一個求職簡歷的例子，每份簡歷都包括教育經歷、項目經歷等，我們可以讓教育經歷和項目經歷分別對應一個PO，這樣在我們建立對應求職簡歷的BO 對象處理簡歷的時候，讓每個BO 都包含這些PO 即可。
* 第 4 個：VO（Value Object）值對象，通常用於業務層之間的數據傳遞，和PO 一樣也是僅僅包含數據而已，但VO 應該是抽像出的業務對象，可以和表對應，也可以不對應，這根據業務的需要。如果鍋碗瓢盆分別為對應的業務對象的話，那麼整個碗櫃就是一個值對象。此外，VO 也可以稱為頁面對象，如果稱為頁面對象的話，那麼它所代表的將是整個頁面展示層的對象，也可以由需要的業務對象進行組裝而來。
* 第 5 個：DTO（Data Transfer Object）數據傳輸對象，主要用於遠程調用等需要大量傳輸對象的地方，比如我們有一個交易訂單表，含有25 個字段，那麼其對應的PO 就有25 個屬性，但我們的頁面上只需要顯示5 個字段，因此沒有必要把整個PO 對像傳遞給客戶端，這時我們只需把僅有5 個屬性的DTO 把結果傳遞給客戶端即可，而且如果用這個對象來對應界面的顯示對象，那此時它的身份就轉為VO。使用 DTO 的好處有兩個，一是能避免傳遞過多的無用數據，提高數據的傳輸速度；二是能隱藏後端的表結構。常見的用法是：將請求的數據或屬性組裝成一個 RequestDTO，再將響應的數據或屬性組裝成一個 ResponseDTO.
* 第 6 個：POJO
* POJO（Plain Ordinary Java Object）簡單的 Java 對象，實際就是普通的 JavaBeans，是為了避免和 EJB（Enterprise JavaBean）混淆所創造的簡稱。 POJO 實質上可以理解為簡單的實體類，其中有一些屬性及其getter和setter方法的類，沒有業務邏輯，也不允許有業務方法，也不能攜帶有connection之類的方法。 POJO 是JavaEE 世界裡面最靈活的對象，在簡單系統中，如果從數據庫到頁面展示都是POJO 的話，它可以是DTO；如果從數據庫中到業務處理中都是POJO 的話，它可以是BO；如果從數據庫到整個頁面的展示的話，它也可以是VO.

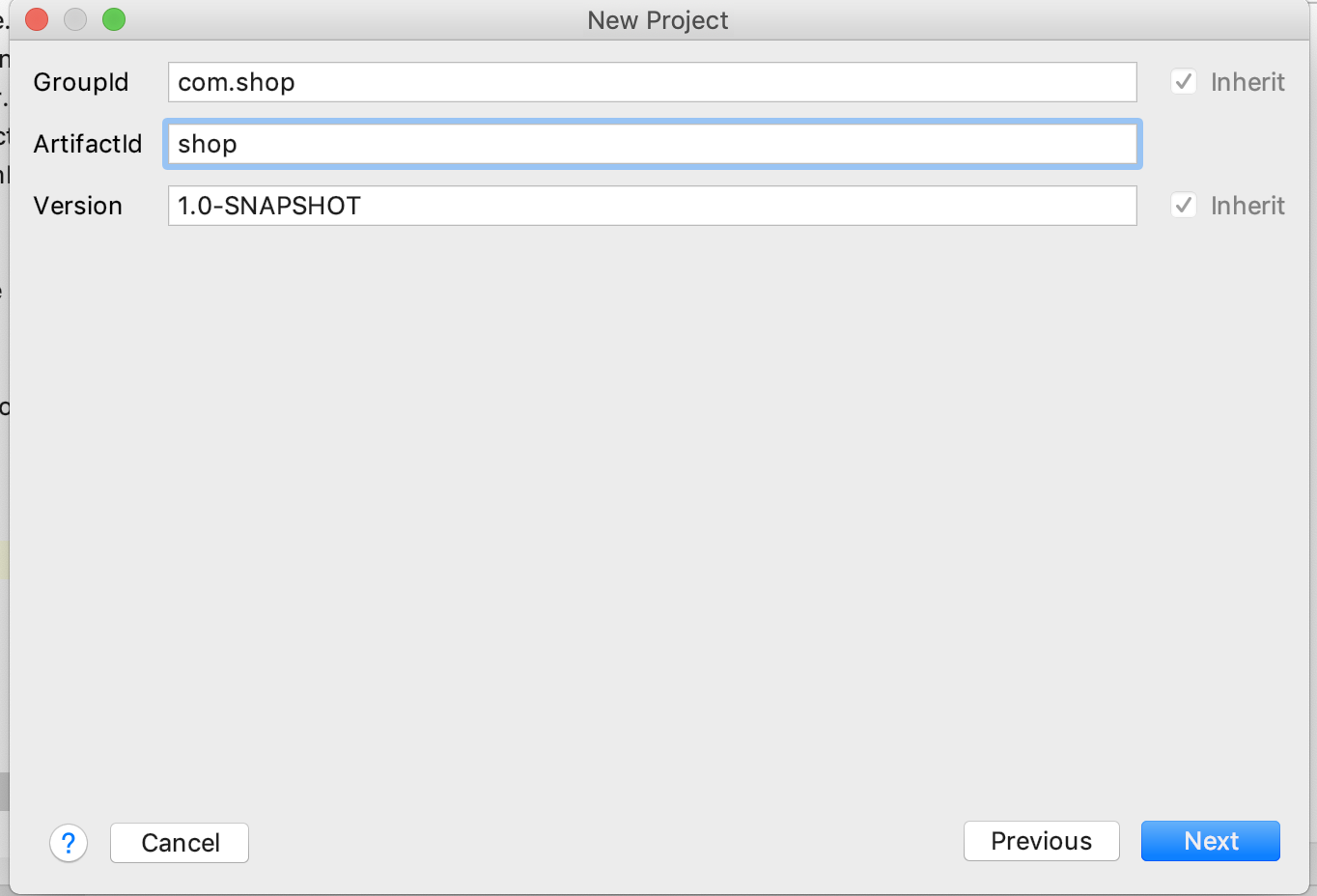
使用IDEA+Maven搭建SpringBoot開發環境



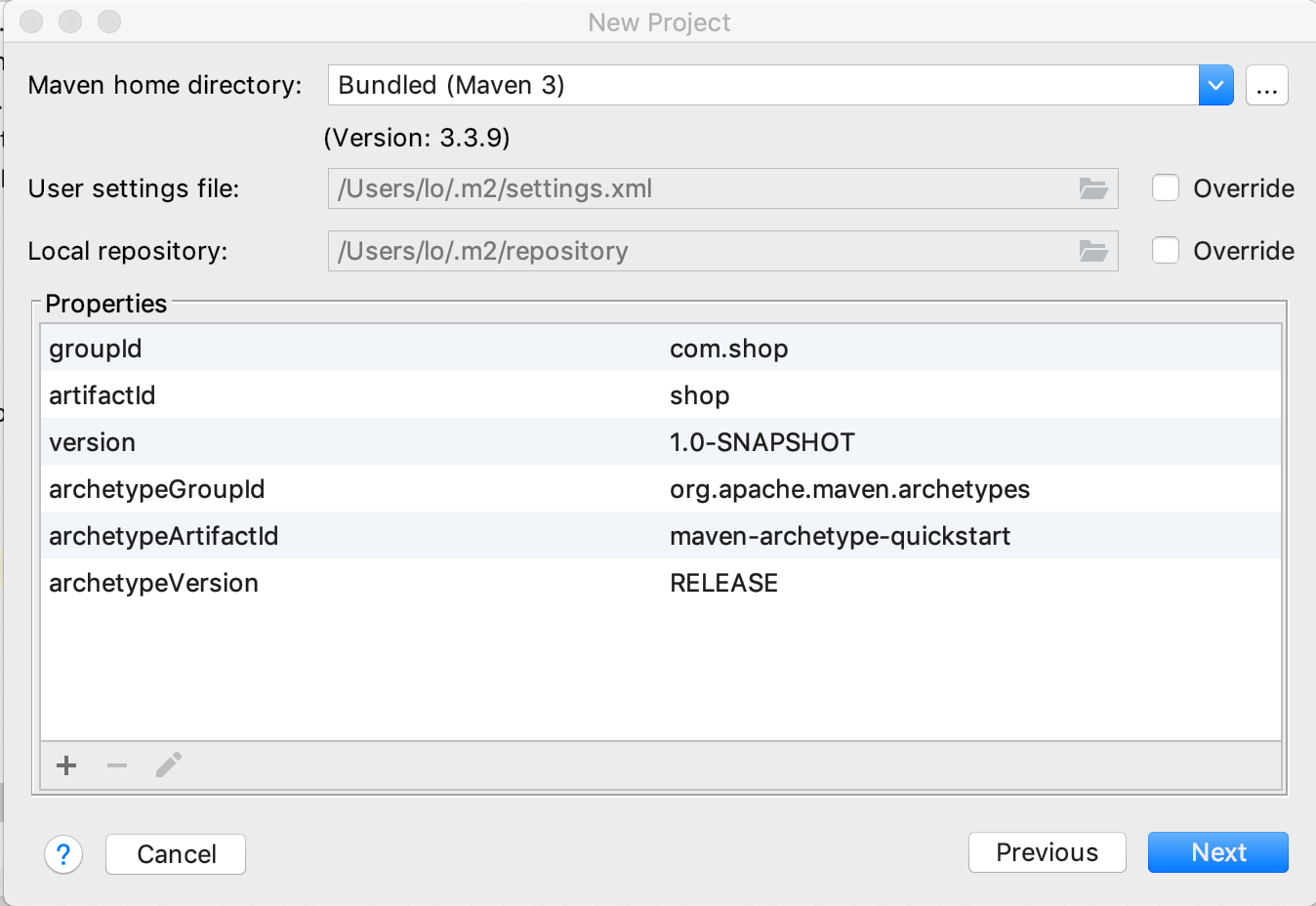
開啟IntelliJ IDEA(以下簡稱IDEA)點選 File > New > Project



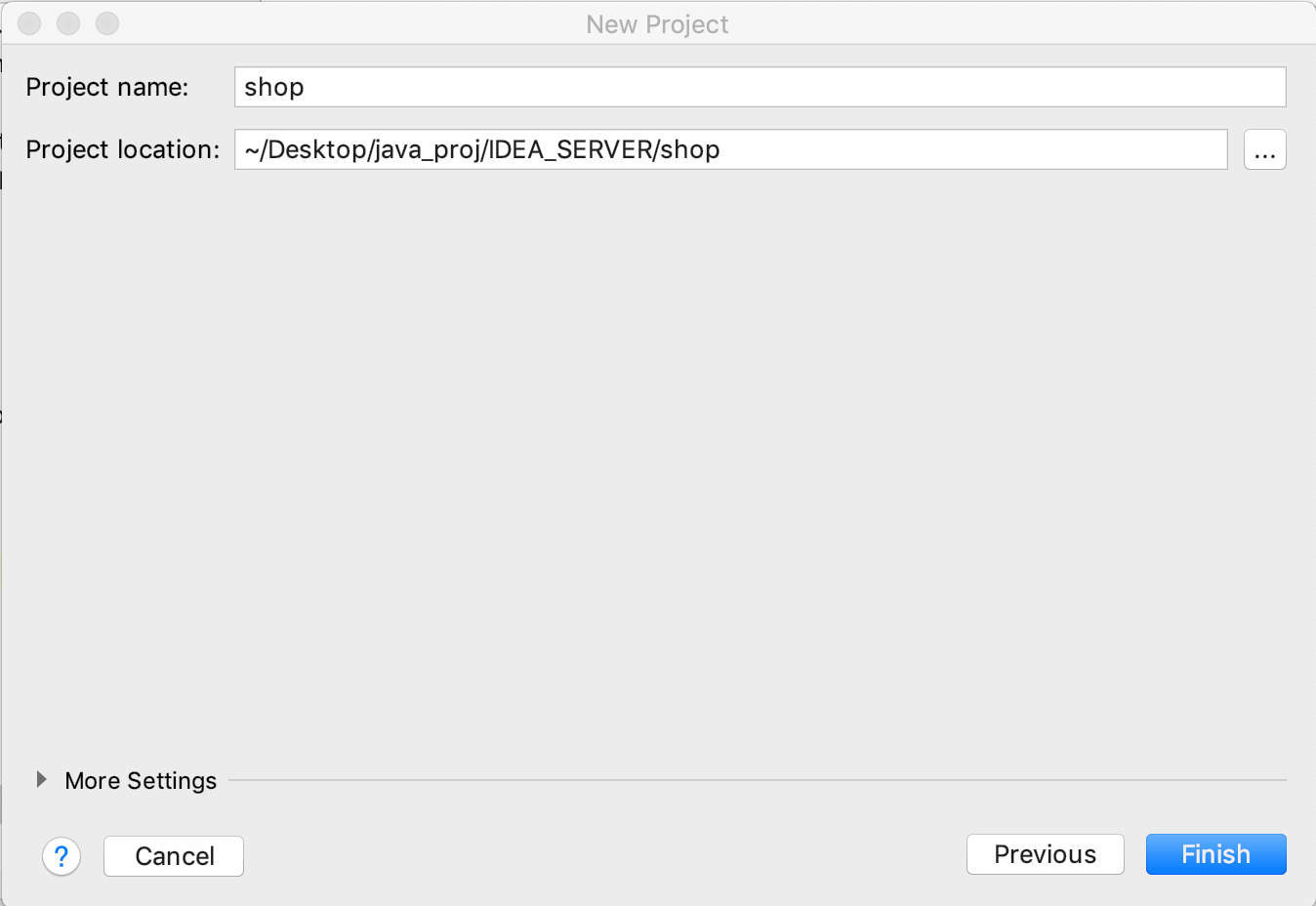
New Project -> 左邊選擇 Maven > 右邊選擇結尾是 -quickstart 的選項(SDK請選擇1.8)



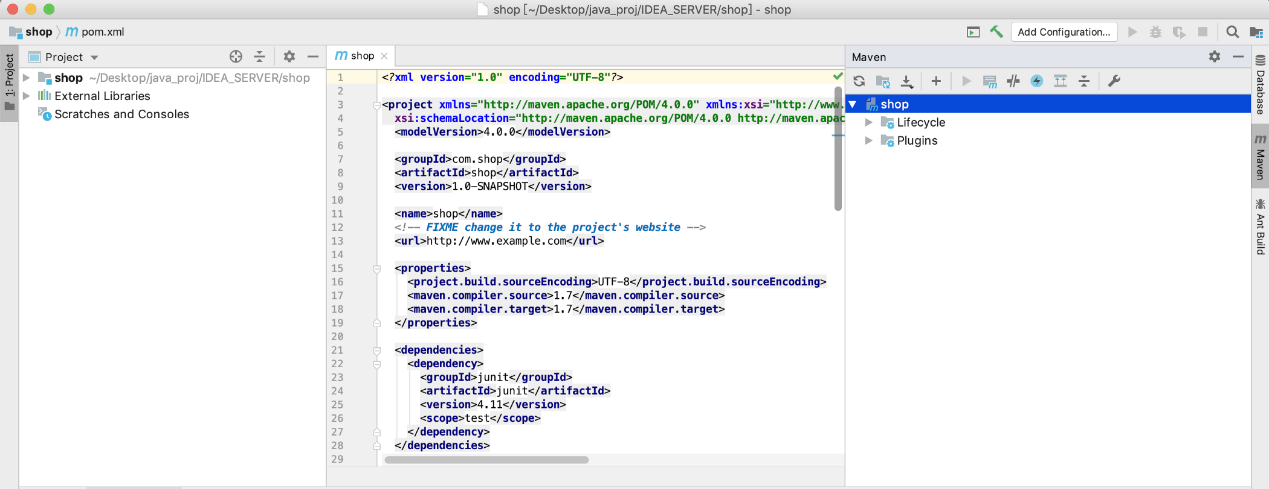
以上資訊自已可打上不一樣的



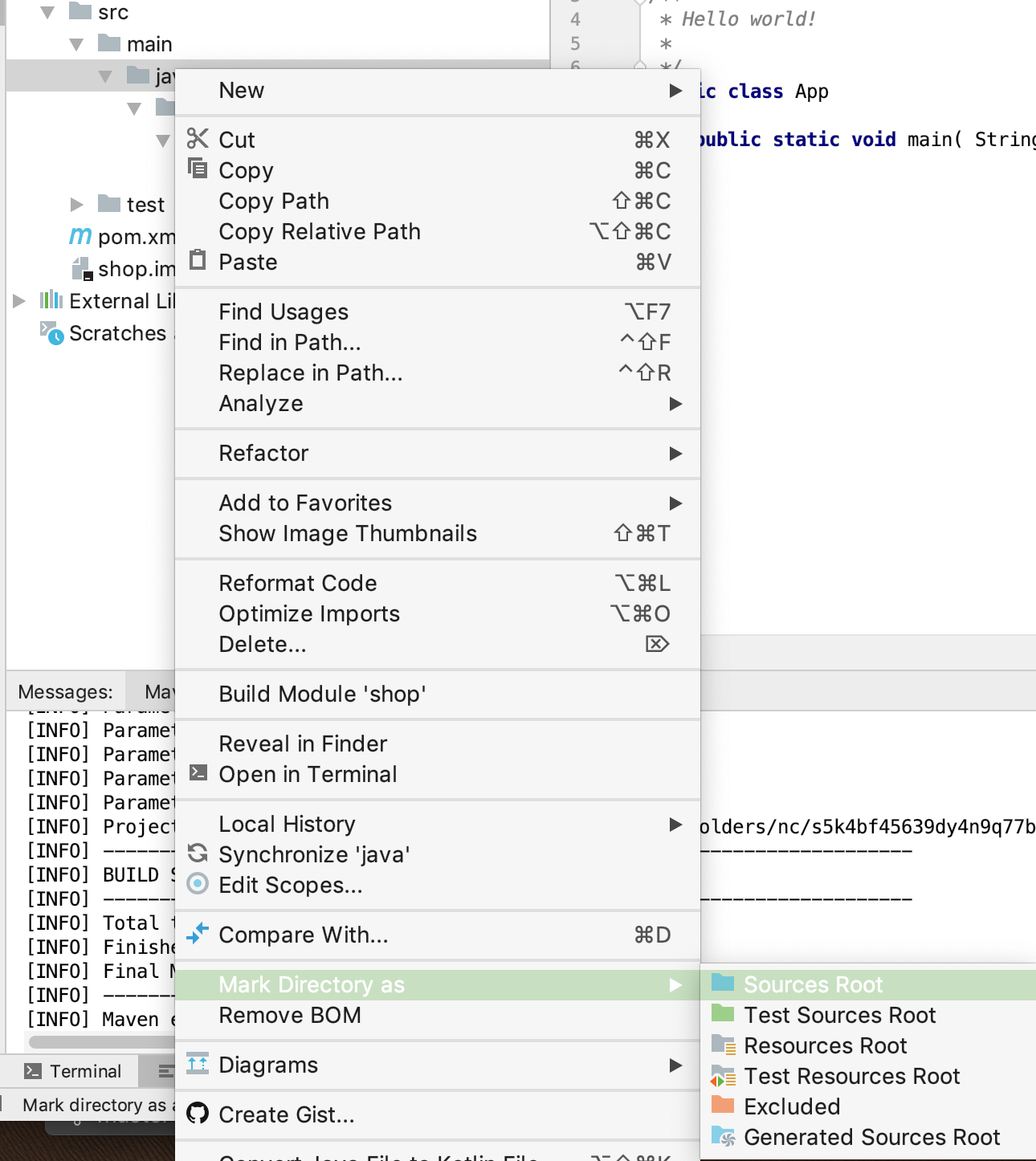
這邊不用變



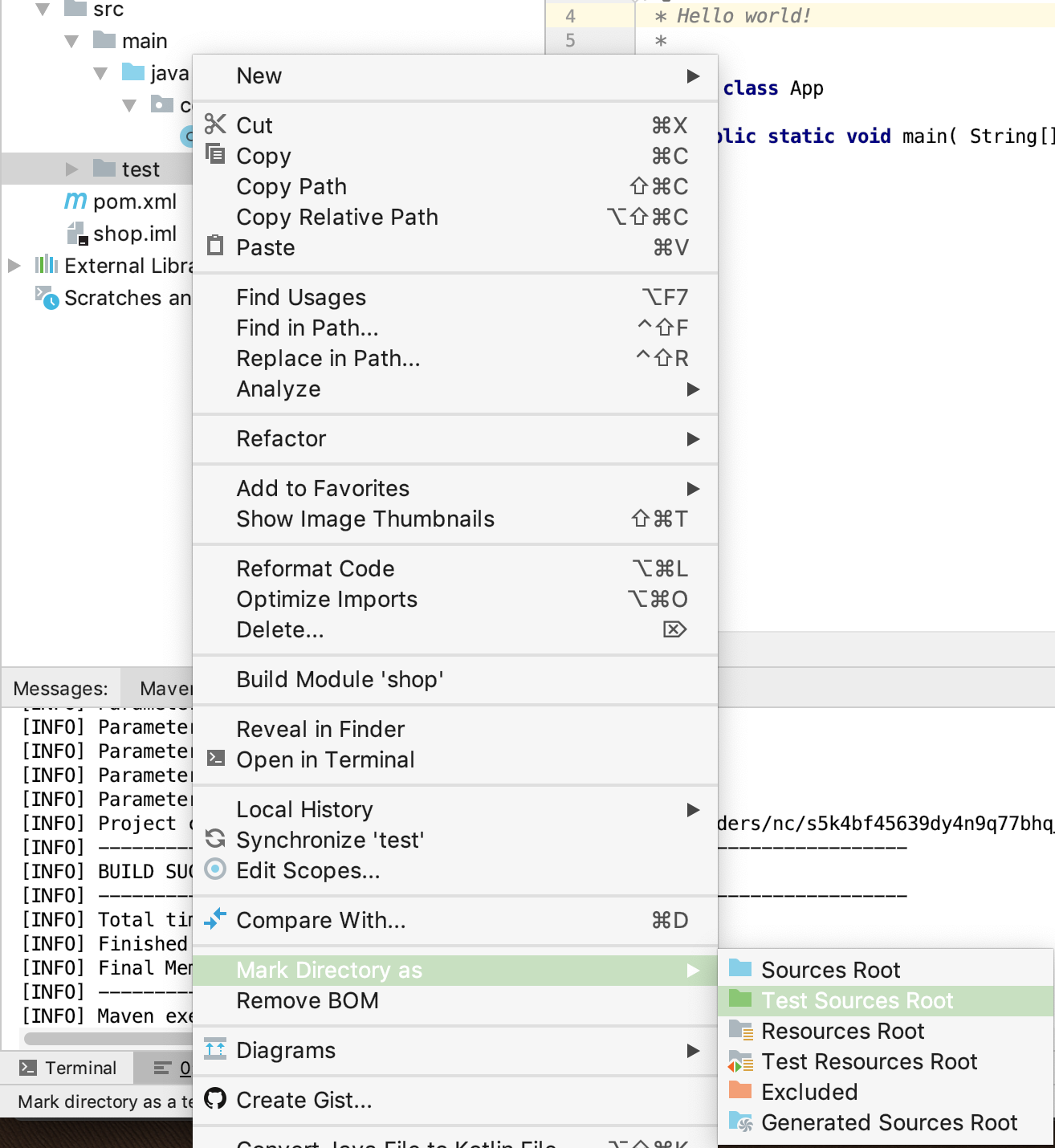
Project name : 這裡是取名shop



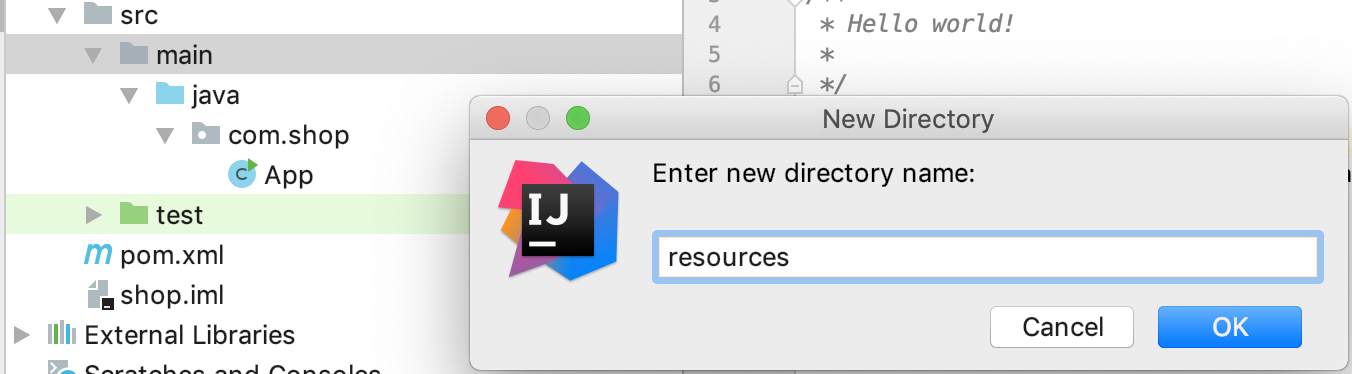
開啟新視窗後，畫面如上，Maven要點選右邊才會出現(eclipse不一樣)



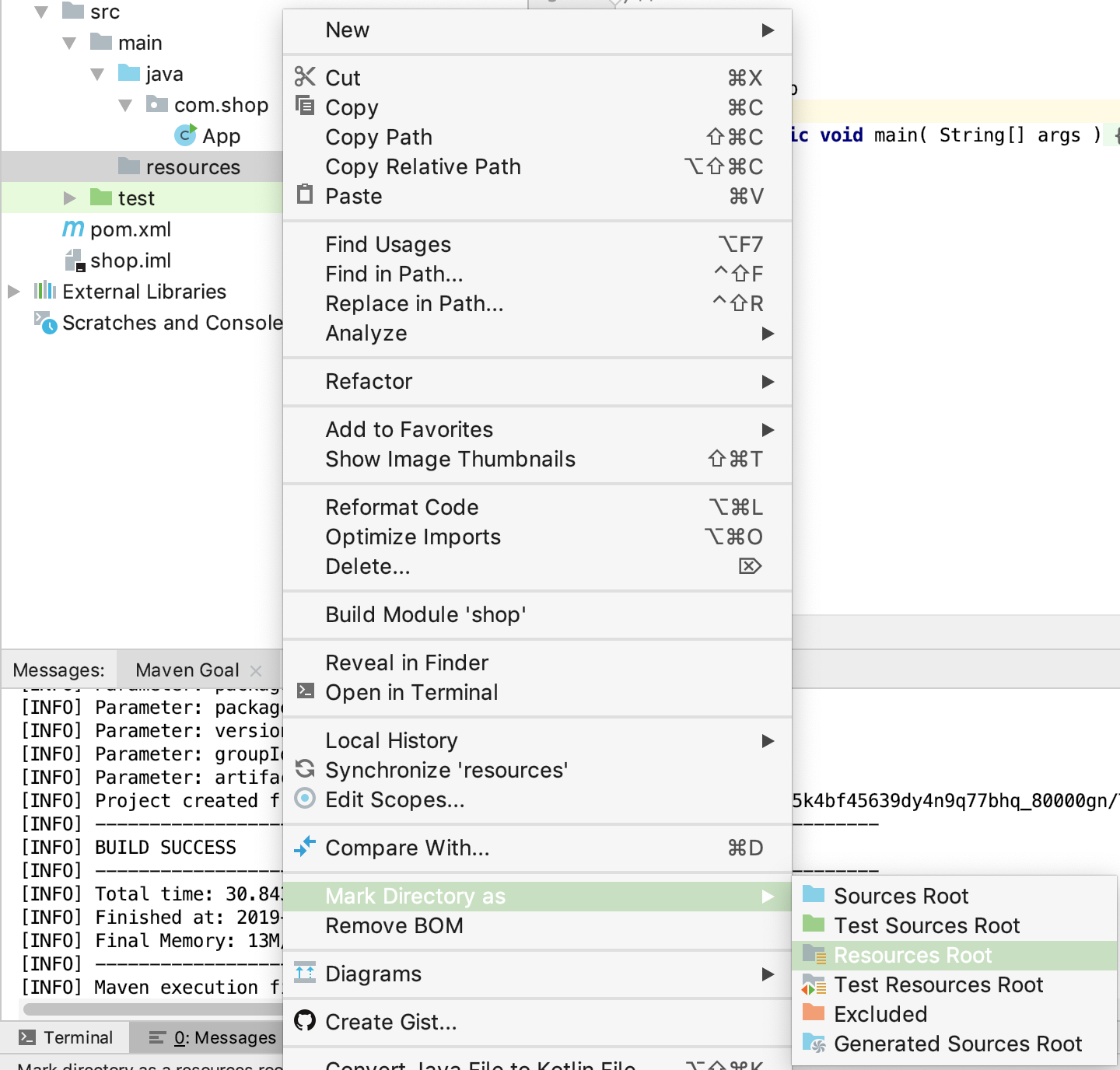
將src下的java目錄設為原始碼路徑



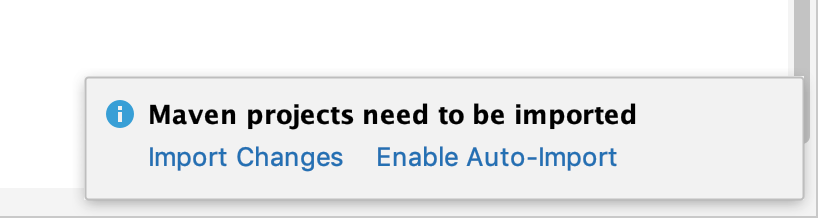
將src下的test目錄設為測試路徑



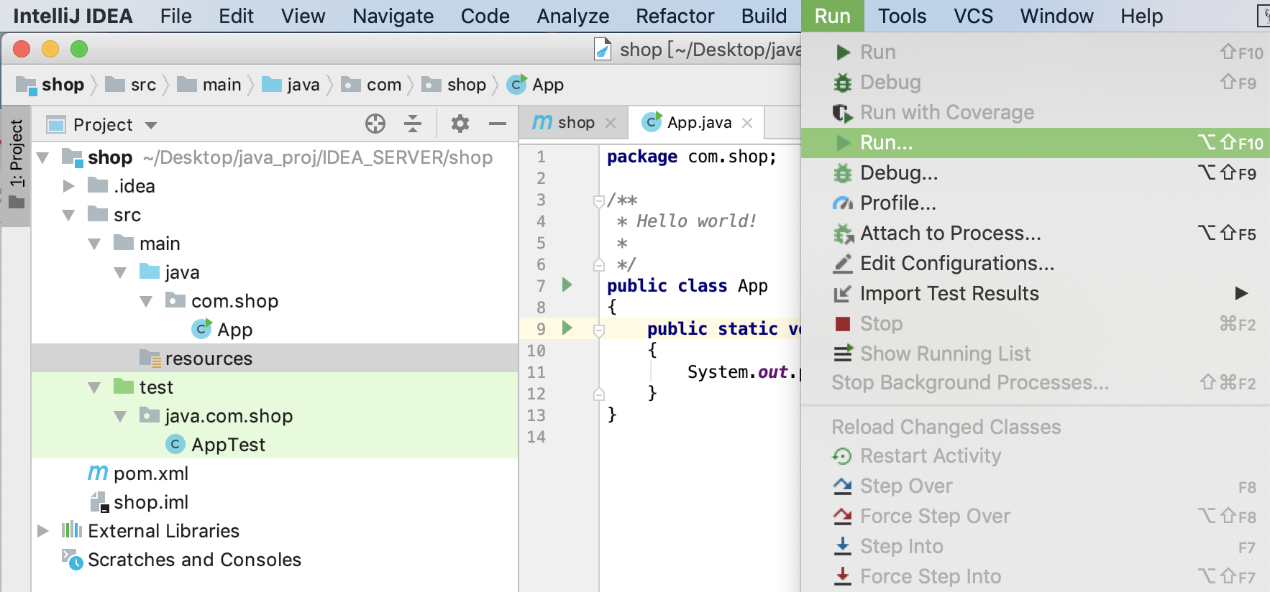
在main下面新建resources目錄



將resources設為Resources Root 路徑



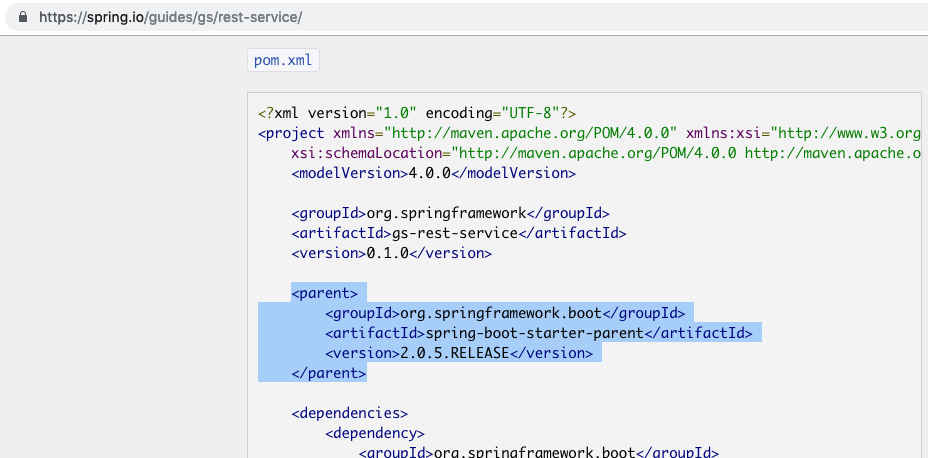
右下角如跳出此小視，選擇 Enable Auto-Import讓IDEA幫你自動化引入maven的設定包



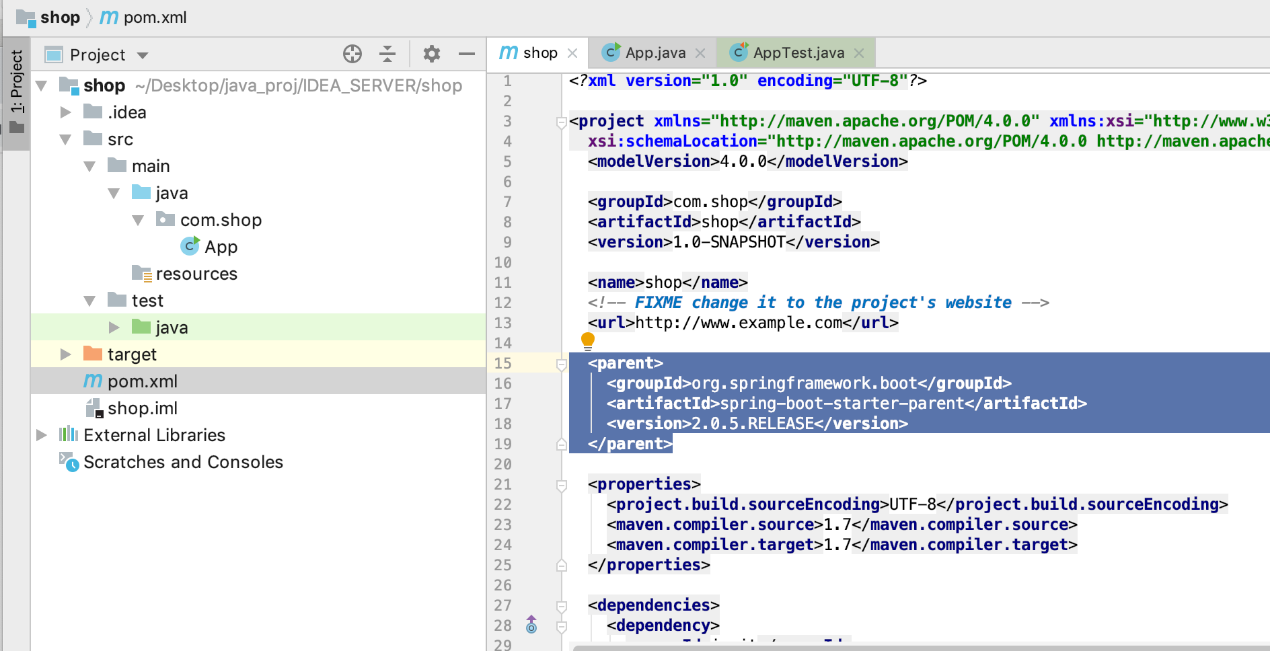
最終的資料結構如圖左，沒問題就按執行 Run，然後選擇App的類別進行啟動入口

引入 SpringBoot相關依賴的開發包

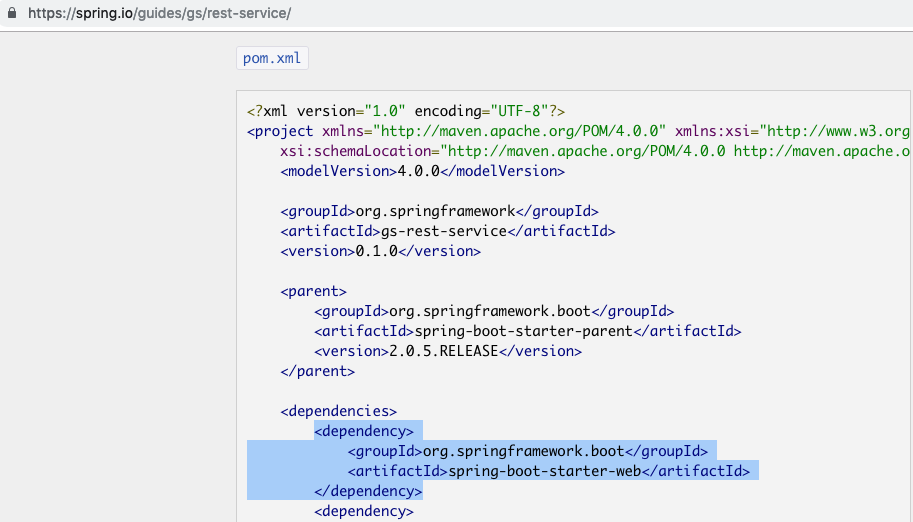
先進入 : <https://spring.io/guides/gs/rest-service/> 檢索我們要的目標  
我們的目標 : **Building a RESTful Web Service**



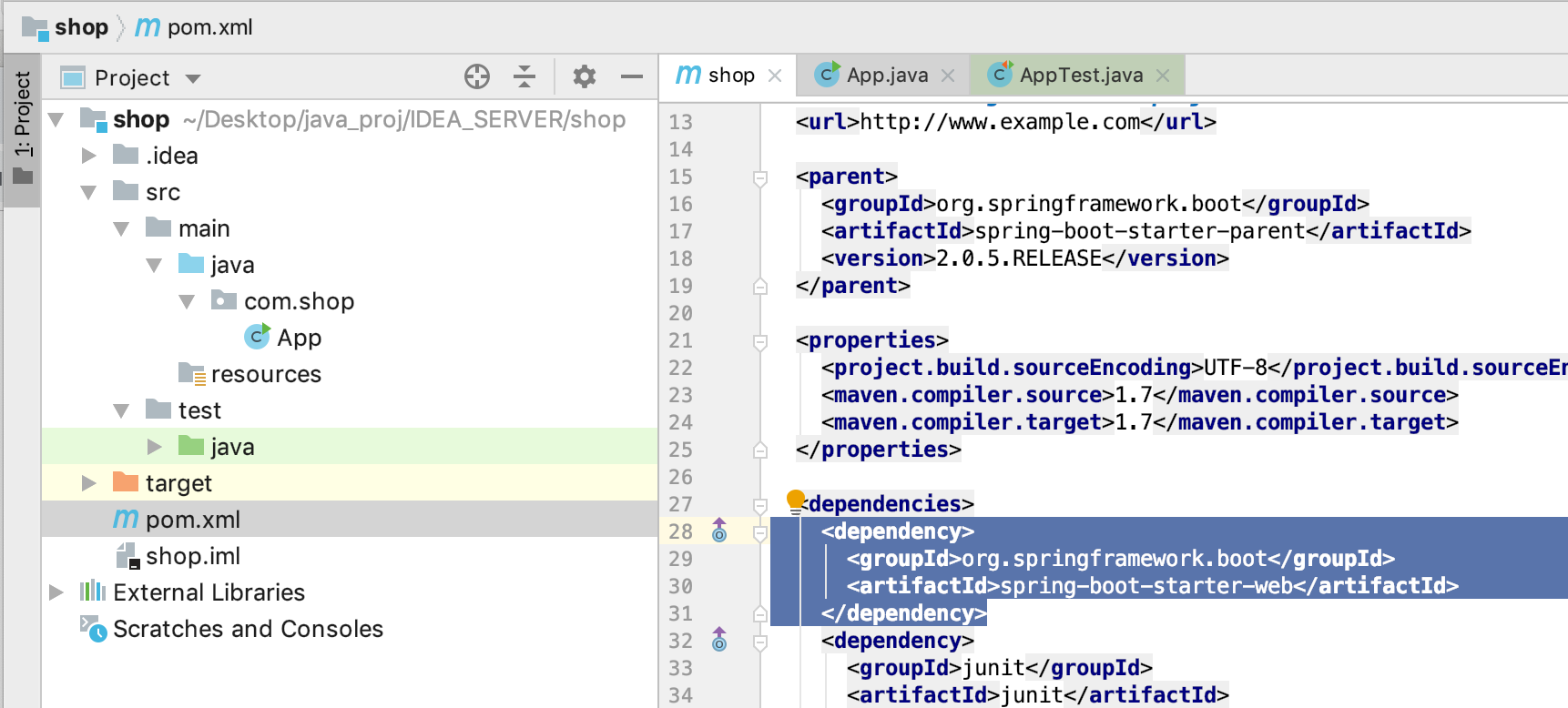
在官網上找到 maven的pom.xml結構檔，將選取的這一段複制下來



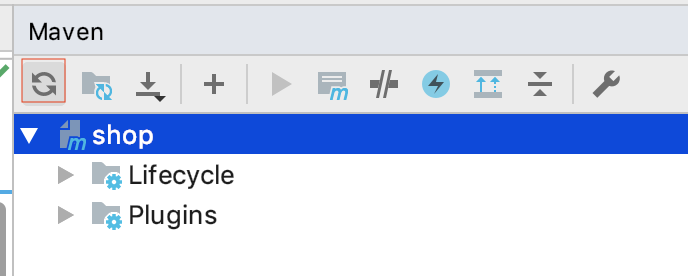
貼到我們shop專案的pom.xml裡面



在官網複制下一段

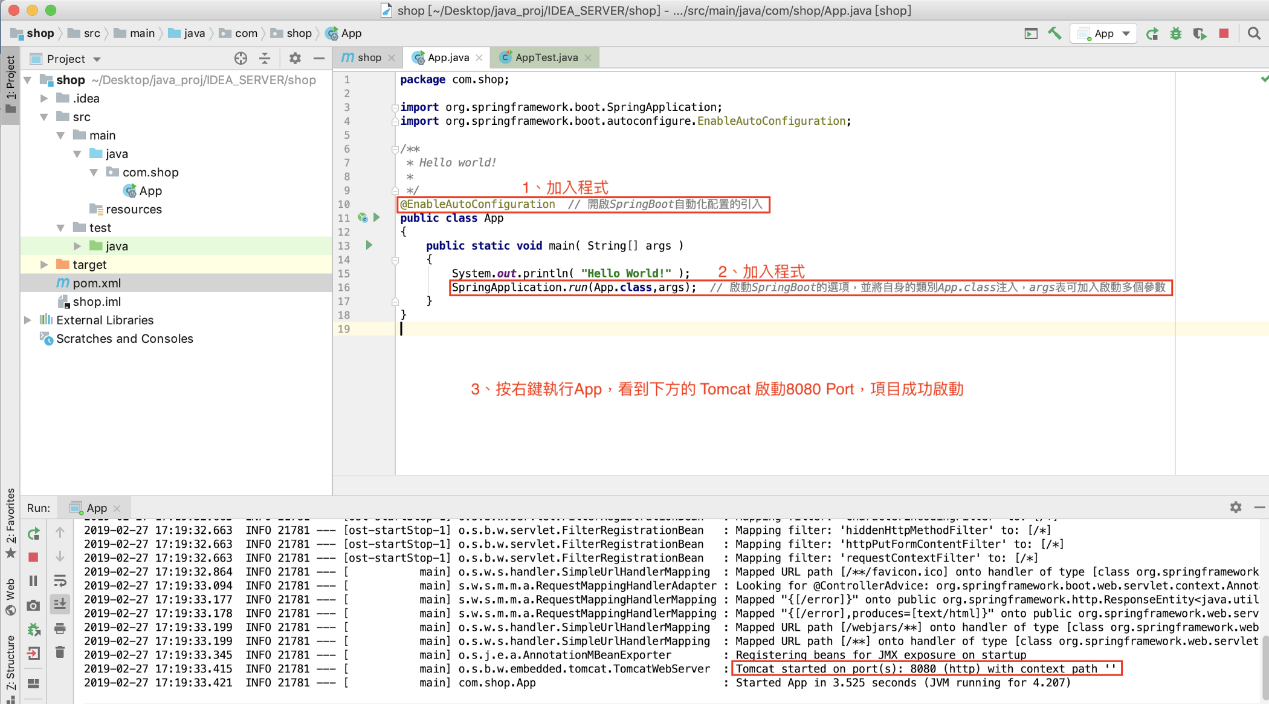


再貼到我們的pom.xml檔案裡

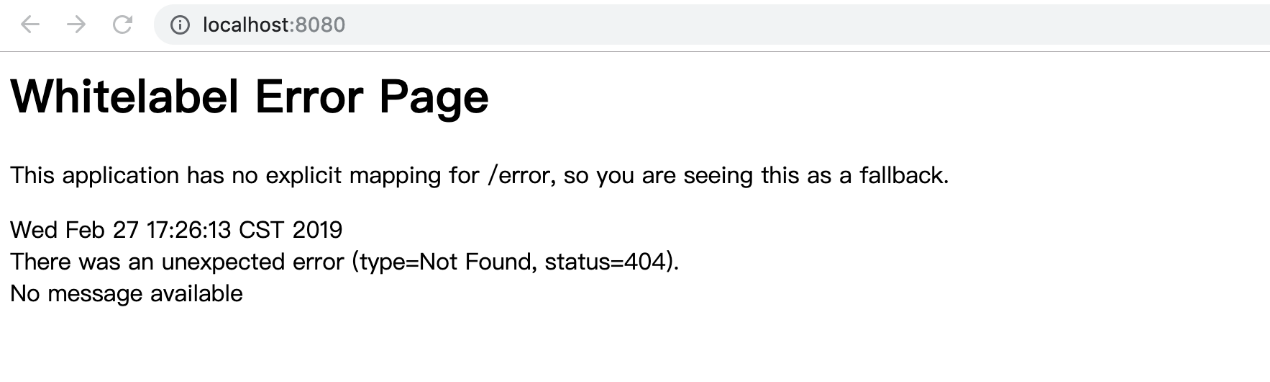


如果有紅字的話，按一下紅色框框，讓maven下載剛才的依賴的jar包，如果上面有選擇自動引jar包的選項的話，IDEA會自動即時將maven的依賴jar包下載下來

實現一個基礎的WEB項目



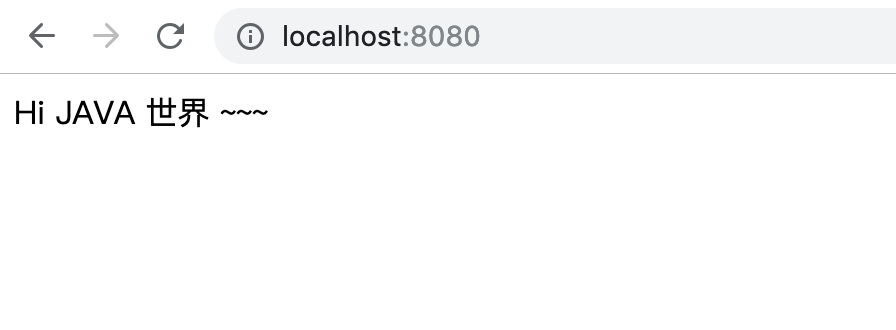
@EnableAutoConfiguration *// 開啟SpringBoot自動化配置的引入***public class** App   
{  
 **public static void** main( String[] args )  
 {  
 System.***out***.println( **"Hello World!"** );  
 SpringApplication.*run*(App.**class**,args); *// 啟動SpringBoot的選項，並將自身的類別App.class注入，args表可加入啟動多個參數* }  
}



打開瀏覽器輸入 http:// localhost:8080 啟動，看到上面即表示成功



加入紅色框框的程式碼，並執行程式



打開瀏覽器輸入 http:// localhost:8080/ 啟動，看到上面即表示成功