**HƯỚNG DẪN CÀI ĐẶT REST ASSURED VÀ MỘT SỐ LÝ THUYẾT CƠ BẢN**

**I. CÀI ĐẶT REST ASSURED CHO DỰ ÁN.**

**1.1 Tạo Maven project.**  
 <Tham khảo : REF\_1\_CREATE\_MAVEN\_PROJECT.docx>

**1.2 Cài đặt testNG.**

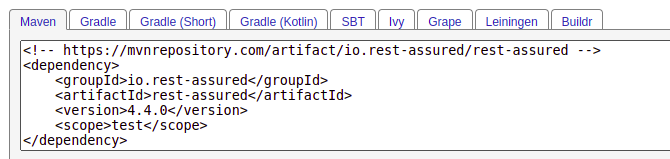
<Tham khảo: REF\_2\_Setup\_TestNG.docx

**1.3 Cài đặt thư viện Rest Assured .**

Với Rest Assured, chúng ta chỉ cần thêm thư viện từ maven để có thể sử dụng các   
 class và method của Rest Assured

Ví dụ: chúng ta dùng version 4.4.0 - version ổn định và có số lượng người tải nhiều

https://mvnrepository.com/artifact/io.rest-assured/rest-assured/4.4.0



**1.4. Cài đặt các thư viện bổ sung.**

Trong project chúng ta sẽ sử dụng thêm một số thư viện bổ sung dưới đây:

• **Hamcrest librar**y: thư viện cung cấp các class và method tiện ích cho assertion  
 VD: version 2.2

https://mvnrepository.com/artifact/org.hamcrest/hamcrest/2.2

• **jackson-databind**  : xử lý Json

VD: version 2.9.8

https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind/2.9.8

# Add tất cả các thư viện bên trên vào pom file → **Ctrl + S**

| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.test.automation</groupId>  <artifactId>api-test</artifactId>  <version>0.0.1-SNAPSHOT</version>    <dependencies>  <dependency>  <groupId>org.hamcrest</groupId>  <artifactId>hamcrest</artifactId>  <version>2.2</version>  </dependency>    <dependency>  <groupId>org.testng</groupId>  <artifactId>testng</artifactId>  <version>7.4.0</version>  </dependency>    <dependency>  <groupId>com.fasterxml.jackson.core</groupId>  <artifactId>jackson-databind</artifactId>  <version>2.9.8</version>  </dependency>    <dependency>  <groupId>io.rest-assured</groupId>  <artifactId>rest-assured</artifactId>  <version>4.4.0</version>  <scope>test</scope>  </dependency>    </dependencies>  </project> |
| --- |

Sau khi Crtl + S để lưu file thì các thư viện sẽ được tự động tải về máy.  
  
  
  
  
  
  
  
  
  
  
I**I. LÝ THUYẾT CƠ BẢN  
2.1 JSON**

JavaScript Object Notation (Json) là một kiểu dữ liệu mở trong JavaScript.

* Định dạng dữ liệu nhẹ, phù hợp để lưu trữ hoặc trao đổi dữ liệu.
* Cấu trúc của Json là bọc dữ liệu trong dấu {}
* Dữ liệu được thể hiện thông qua cặp Key - value
* Value có thể là 1 trường dữ liệu, 1 mảng hoặc 1 đối tượng Json khác.

VD:

| {  "name": "Product", // value là một trường  "id": { // value là một đối tượng Json  "type": "number",  "description": "Product identifier",  "required": true  },  "sub": [ // value là một mảng đối tượng Json  {  "id": 1,  "name": "J1",  "required": true  },  {  "id": 2,  "name": "J1",  "required": true  }  ]  } |
| --- |

**2.2 JSON Path**  Cấu trúc của một Json sẽ bao gồm các nút và các phần tử.

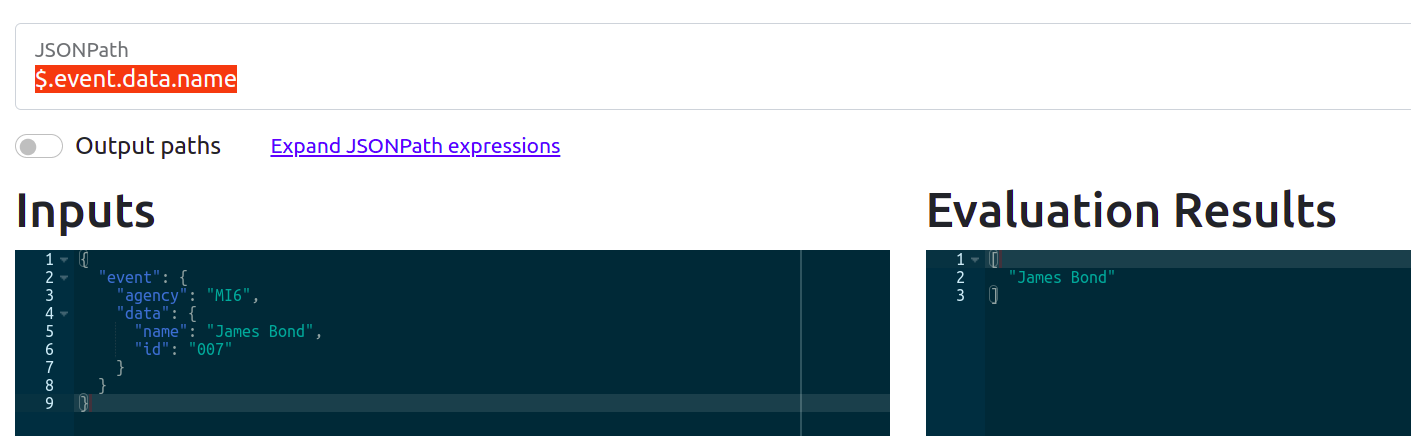
| {  "event": {  "agency": "MI6",  "data": {  "name": "James Bond",  "id": "007"  }  }  } |
| --- |

JSONPath là ngôn ngữ truy vấn cho JSON, nó tương tự như Xpath. Nó cho phép bạn lựa chọn và trích xuất dữ liệu từ một JSON. Chúng ta có thể sử dụng biểu thức JSONPath để duyệt qua đường dẫn đến một phần tử trong cấu trúc JSON và trích xuất dữ liệu của phần tử. Quá trình duyệt sẽ xuất phát từ nút gốc (root : $)

**VD:** để lấy ra thông tin **name** trong JSON trên thì ta sẽ sử dụng JSONPath sau

| $.event.data.name |
| --- |

Các bạn có thể thực hành JSONPath tại đây: <https://jsonpath.com/>



**2.3 REST API Method.**

* **GET ():** Trả về một Resource hoặc một danh sách Resource.
* **POST ():** Tạo mới một Resource.
* **PUT ():** Cập nhật thông tin cho Resource.
* **DELETE ():** Xoá một Resource.