



REST Assured Dependency Setup

To use **REST Assured** in your project, you can add it either:

-  **Using Maven (recommended)**
-  **Manually (JAR download and setup)**

1. Add REST Assured Dependency using Maven

In your `pom.xml`, add the following dependency:

```
xml

<dependencies>
  <!-- REST Assured for API Testing -->
  <dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <version>5.4.0</version>
    <scope>test</scope>
  </dependency>

  <!-- JSON Path (for parsing JSON responses) -->
  <dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>json-path</artifactId>
    <version>5.4.0</version>
  </dependency>

  <!-- To support Hamcrest Matchers -->
  <dependency>
    <groupId>org.hamcrest</groupId>
    <artifactId>hamcrest</artifactId>
    <version>2.2</version>
  </dependency>
</dependencies>
```

 Replace **5.4.0** with the latest version if needed (check: <https://mvnrepository.com/artifact/io.rest-assured/rest-assured>)

Then, run:

```
bash

mvn clean install
```

2. Add REST Assured Manually (Without Maven)


If you're not using Maven, follow these steps:

Step 1: Download Required JARs

From [Maven Central](https://mvnrepository.com/artifact/io.rest-assured/rest-assured) or <https://rest-assured.io>:

You'll need the following JAR files:

- `rest-assured-5.4.0.jar`
- `json-path-5.4.0.jar`
- `xml-path-5.4.0.jar` (if working with XML)
- `hamcrest-2.2.jar`
- `commons-codec.jar` (transitive dependency)
- `groovy.jar` (as REST Assured internally uses Groovy)

 Place them in a `lib/` folder inside your project

Step 2: Add JARs to Classpath

- In Eclipse/IntelliJ:
 - Right-click project → **Build Path** → **Configure Build Path**
 - Add external JARs from the `lib/` folder
- Or compile via command line:

```
bash

javac -cp ".;lib/*" YourTestClass.java
```

Let me know if you want me to generate a sample test class to verify your REST Assured setup!

Here are the **required import statements** for a basic REST Assured test in Java:

✓ Basic REST Assured Imports

```
java

import io.restassured.RestAssured;
import io.restassured.response.Response;
import io.restassured.specification.RequestSpecification;

import static io.restassured.RestAssured.*;
import static org.hamcrest.Matchers.*;
```

🔍 Additional Optional Imports (commonly used)

```
java

// For JSON parsing
import io.restassured.path.json.JsonPath;

// For assertion
import org.testng.Assert; // or use JUnit's org.junit.Assert
```

📌 Example Usage

```
java

public class RestAssuredExample {

    public static void main(String[] args) {

        // Set base URI
        RestAssured.baseURI = "https://jsonplaceholder.typicode.com";

        // Create request
        RequestSpecification request = given();

        // Get response
        Response response = request.get("/posts/1");

        // Print response
        System.out.println("Response Body:\n" + response.getBody().asString());

        // Validate status code
        response.then().statusCode(200);

        // Validate JSON field
        response.then().body("userId", equalTo(1));

    }
}
```

Let me know if you're using **TestNG**, **JUnit**, or just want to run standalone Java `main()` methods — I can customize imports accordingly.

Here's a **basic reusable structure** for **REST Assured API testing** using both **GET** and **POST** methods in Java.

✓ Project Setup

- Language: **Java**
- Framework: **REST Assured**
- Build Tool: **Maven** (*or manual setup as discussed*)
- Testing Framework: **TestNG** (*you can switch to JUnit if needed*)

📁 Folder Structure (Typical Maven/Java Project)

```
bash

project-root/
├── src/
│   ├── test/
│   │   └── java/
│   │       └── api/
│   │           ├── GetRequestTest.java
│   │           └── PostRequestTest.java
└── pom.xml
```

1 GetRequestTest.java – Basic GET Example

```
java

package api;

import io.restassured.RestAssured;
import io.restassured.response.Response;
import org.testng.annotations.Test;

import static io.restassured.RestAssured.*;
import static org.hamcrest.Matchers.*;

public class GetRequestTest {

    @Test
    public void testGetPostById() {
        RestAssured.baseURI = "https://jsonplaceholder.typicode.com";

        given().
            log().all().
        when().
            get("/posts/1").
        then().
            log().all().
            statusCode(200).
            body("id", equalTo(1)).
            body("userId", equalTo(1));
    }
}
```

2 PostRequestTest.java – Basic POST Example

```
java

package api;

import io.restassured.RestAssured;
import io.restassured.http.ContentType;
import org.testng.annotations.Test;

import static io.restassured.RestAssured.*;
import static org.hamcrest.Matchers.*;

public class PostRequestTest {

    @Test
    public void testCreatePost() {
        RestAssured.baseURI = "https://jsonplaceholder.typicode.com";

        String requestBody = ""
        {
            "title": "foo",
            "body": "bar",
            "userId": 1
        }
        """;

        given().
            contentType(ContentType.JSON).
            body(requestBody).
            log().all().
        when().
            post("/posts").
        then().
            log().all().
            statusCode(201).
            body("title", equalTo("foo")).
            body("body", equalTo("bar")).
            body("userId", equalTo(1));
    }
}
```

✓ TestNG Dependency (Add to pom.xml)

```
xml

<dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>7.9.0</version>
  <scope>test</scope>
</dependency>
```

✓ How to Run

- In IDE: Right-click → Run as **TestNG Test**
- Or via Maven:

```
bash
```

```
mvn test
```

Let me know if you want the same structure using **JUnit** or without TestNG (just plain `main()`), or if you want to **parameterize** your tests (data-driven).