**RESTAssured Framework Documentation**

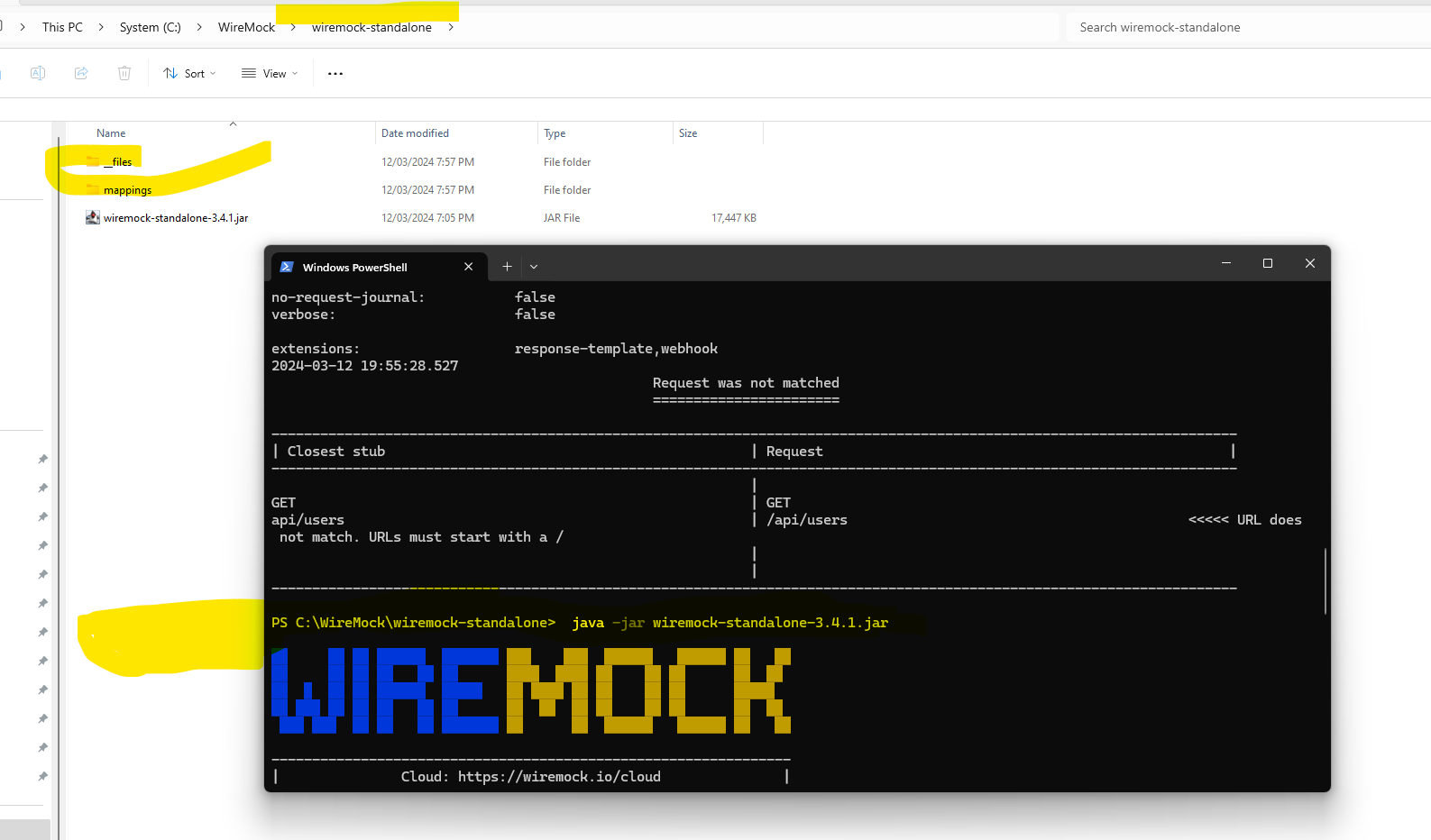
|  |  |
| --- | --- |
| List of items added to framework | * Added PropertiesUtility, ReportsUtility, DBUtility * Added test details to report, logging payload, logging BaseURL, response etc., * added GET, POST, PUT Method in Tests * Added Code for authorization – but not used anywhere * Base Test – has only @BeforeClass, @BeforeMethod …. Annotations. * Common Methods class – has rest of reusable methods - eg., GET, POST, PUT, addDetailsToReport, VerifyAssertions etc, * Added Assertions after getting response. * Gather the response key/value , so we can use it in another request. * Added wiremock test – only for GET list of users. |
| PropertiesUtility | * We provide file path where we can find the application.properties file. * We load this file and extract each property to a “static” variable, so we can use this anywhere else in the project w/o instantiating the class. Eg., PropertiesUtility.BASE\_URI * We have provided environment property, based on this property we read that specific environment properties. * These properties can be used anywhere., i.e., in any classes.   For example, these properties are used in Tests classes, and also in ReportUtility to show the details in report. |
| Report Utility | * We provide file path to say where to create the report * We create/setup the report (Spark report/html report/….) in this class with some information * We created One static variable ExtentReports *report*. * *report* is to setup report, and we hook all the information to this report eg., report.setdSystemInfo(); * There is another static variable, *report* in Base Test. We assign ReportsUtility *report* to BaseTest *report*   eg*.,* report = ReportsUtility.setReport();   * This *report* is used throughout the tests and we hook info to it. |
| DBUtility | * DBUtility is just added to the framework, but not used., * Created a test, but ignored, as we do not have any DB to connect to * So, ignored the code. |
| Authorization code | * Added Authorization code to Base Test, but commented it out * It is kept there for reference, when we need to use |
| Tests | * Base Test is the Parent Class * Common Methods is a child class for Base Test * Reqres is the Child class where all tests are there. * So, BaseTest 🡪CommonMethods🡪Reqres * There are only GET, POST and PUT requests in Tests |
| Things to remember when pushing the file to Github /BitBucket Repository | * Add .gitignore file ( make sure target folder, .iml files, Results folder/file are not pushed to Git repository) * Add README.md file |
| Pushing the code in local to github for the first time | **Add a project to GitHub the easy way**   1. Create a new GitHub repo that contains a README file. 2. Use Git to clone the GitHub repo locally. 3. Copy your project files into the folder created by the clone. 4. Perform a git add . and a git commit. 5. Push your changes up to GitHub. |
| Helpful links | Assertion Site - <https://assertible.com/docs/guide/assertions#add-a-new-assertion>  AssertJ - <https://joel-costigliola.github.io/assertj/assertj-core-quick-start.html> |
| Repository Links | Github – Create a Repository - <https://docs.github.com/en/repositories/creating-and-managing-repositories/quickstart-for-repositories> |
| Wiremock | Website - Wiremock.org  Videos –   * <https://www.youtube.com/watch?v=zb_UHJZNDOo> * <https://www.youtube.com/watch?v=_hAns6gzk8A> - This is very good to understand how to stub data * <https://www.youtube.com/watch?v=9VJkrrt04-w> - This is good video too, especially for automation |
| Wiremock Notes | * <https://wiremock.org/docs/standalone/java-jar/> * You can get the jar from above link. * Start the server ( refer screen shots) * Once the server is up and running, you can hit the following end points * <http://localhost:8080/__admin/mappings> * When server starts running, two folders are created in the folder where wiremock-standalone jar exists. * Folders are \_\_files and mappings folders. * Mappings folder is where we save the json files. Json files will have request and response details. * Multiple stubbings can be done on one endpoint., eg., * /api/users/<1>, * /api/users/<2> etc., * You can also add mappings during runtime ( Use Postman – POST <http://localhost:8080/__admin/mappings> with some json body) – Check the video link above from 15:00 mins * GET <http://localhost:8080/__admin/mappings> to see all mappings including runtime * These runtime mappings will be deleted when server is stopped. * [http://localhost:8080/\_\_admin/docs](http://localhost:8080/__admin/docs%20) , then go to Swagger UI * <http://localhost:8080/__admin/swagger-ui/> * <http://localhost:8080/__admin/recorder/> - This is the recorder to record the data - <https://www.youtube.com/watch?v=_hAns6gzk8A> - Watch the link ( from 20 mins) * When recording, do not use <https://localhost:8080>, but use <http://localhost:8080/reqres/in> * Follow the documentation too. <https://wiremock.org/docs/record-playback/> |

**There are 3 ways to record the request and response and use it in automation\.**

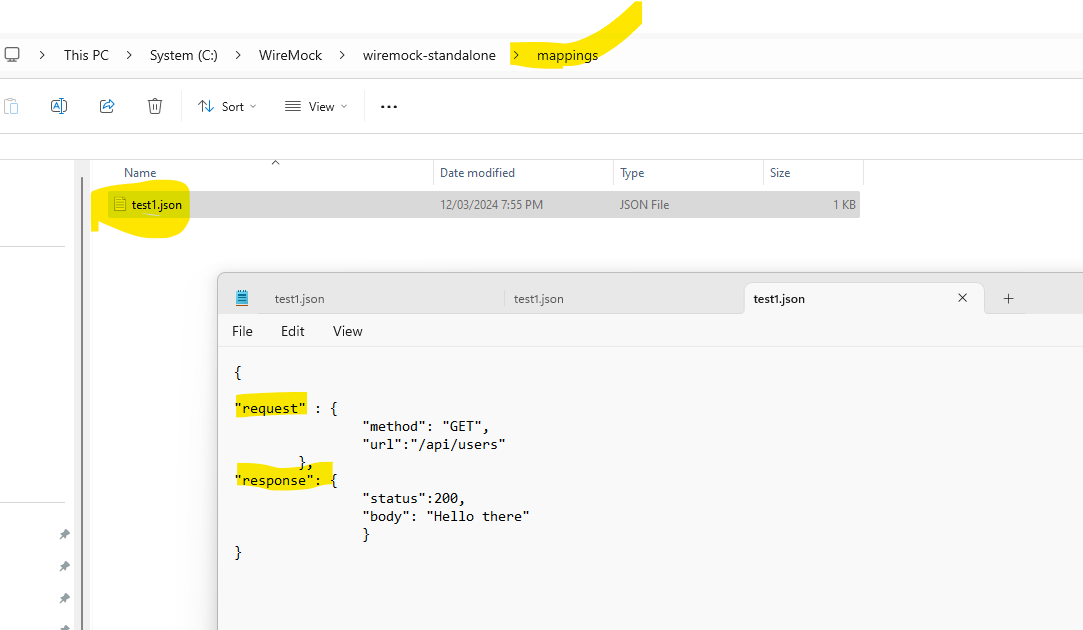
1. Using wiremock-standalone jar file ( see the instructions below)
2. Wiremock cloud - <https://www.youtube.com/watch?v=Iwj2vSQzITU>
3. Wiremock recording in automation.

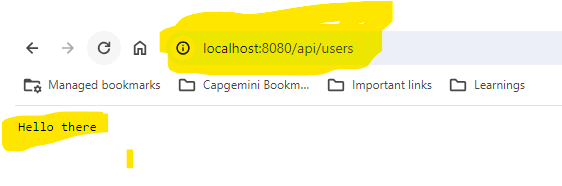
**1. Using Wiremock-standalone JAR file:**

1. You run the server manually as shown in the screenshot below
2. You add request and response in json file under “mappings” folder
3. You hit the localhost either in browser or in postman (or in some other form)
4. You can also record the stubbings from recorder, when wiremock server is up and running.
5. The recordings will store in “mappings” folder, you may then edit the json file to your needs.
6. You may then use these json files as stub data in automation.

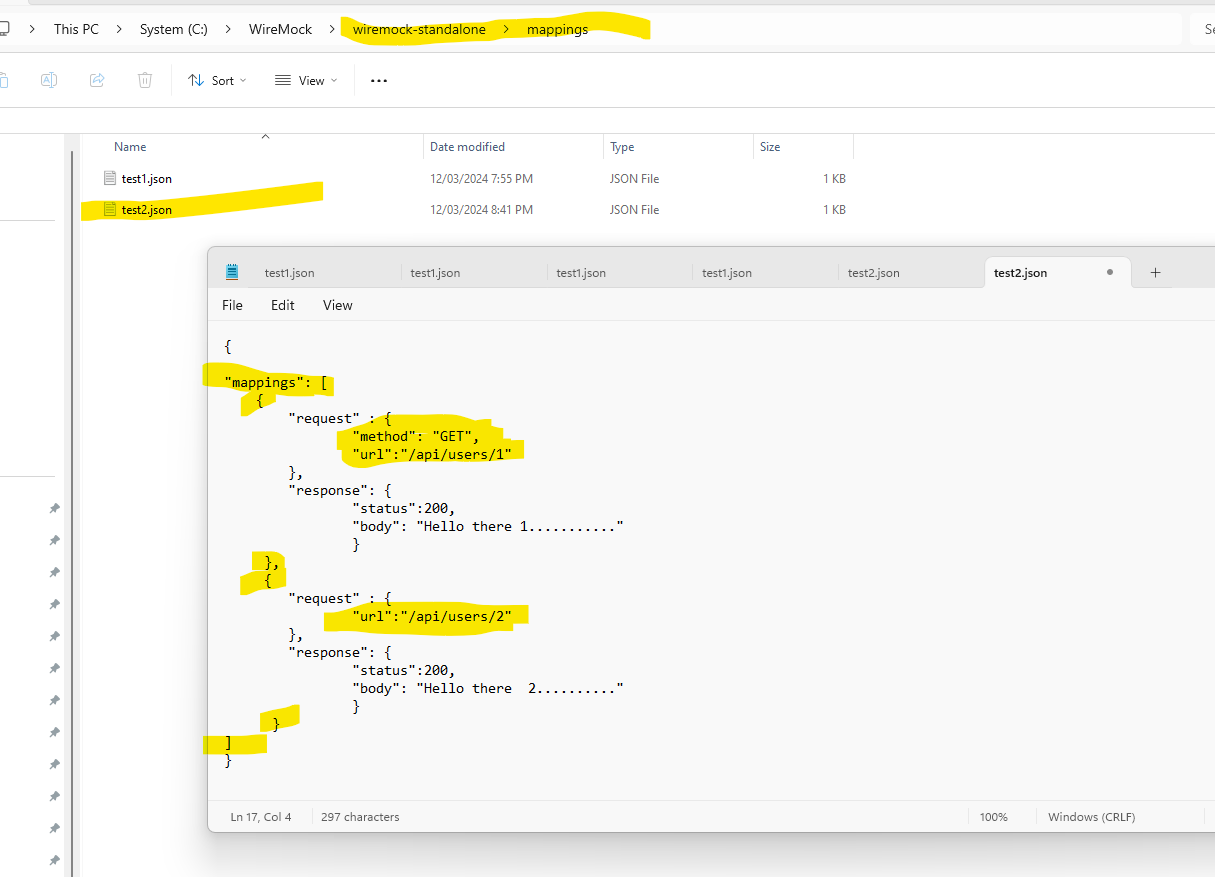


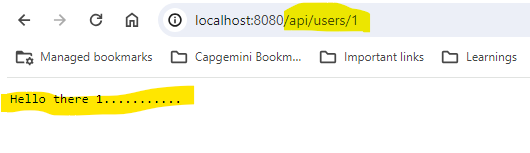
**Single Stubbing:**

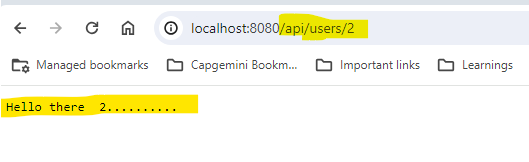




**Multiple Stubbings:**







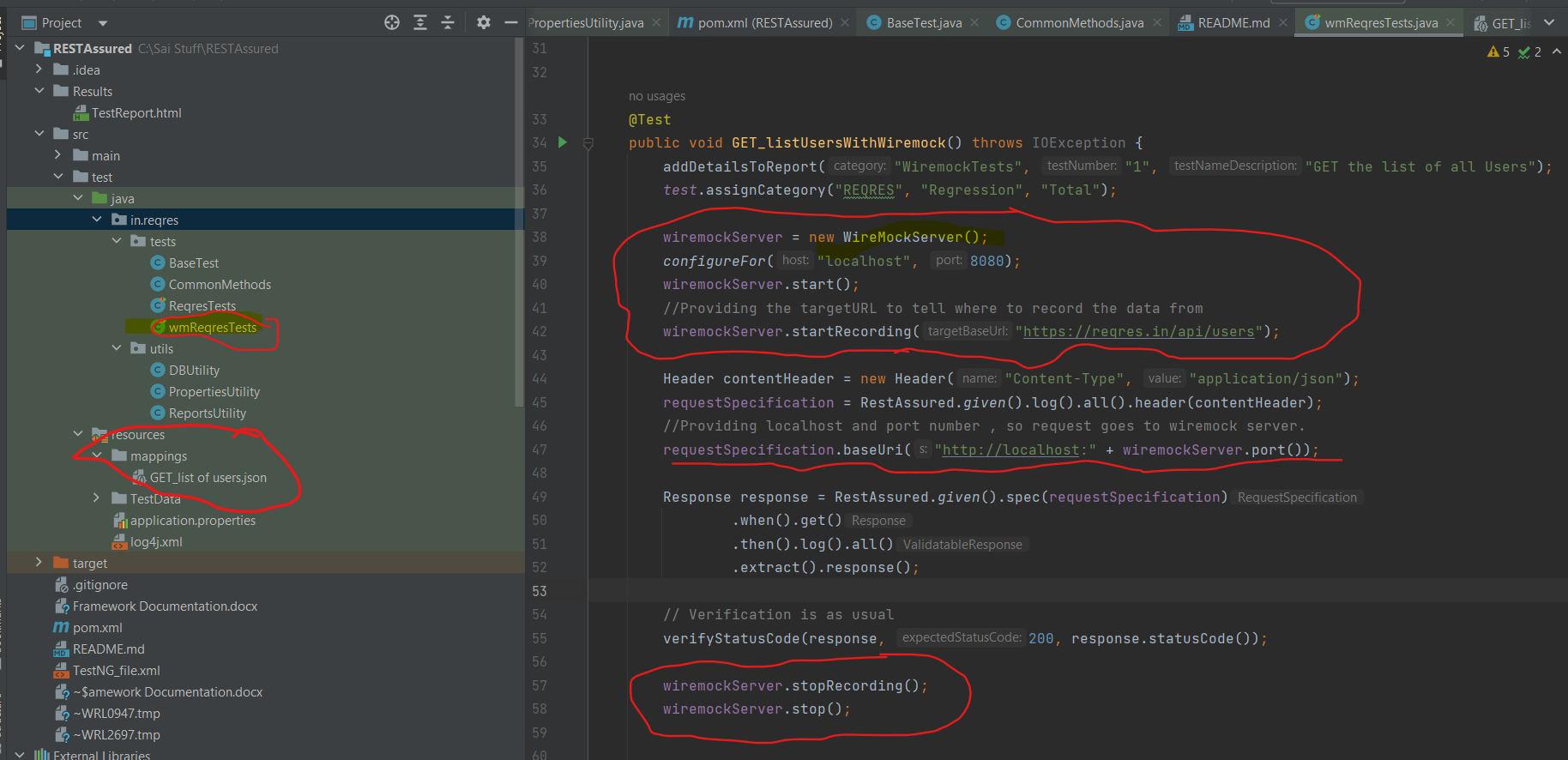
**2. Wiremock Cloud:** This is the easiest way to do. But consider organisation security when using this. If we hit third party APIs, this may not be an issue, but worth a check. Check on how to use this in the following youtube link.

<https://www.youtube.com/watch?v=Iwj2vSQzITU>

**3.Wiremock recording in Automation:**

1. Add the wiremock-standalone dependency in pom.xml file

2. See the screen shot below on code.



1. The recorded json files appear under mappings folder. You only record for first time, and you comment the wiremock.startRecording and wiremock.stopRecording lines to start using the stub data

**NOTE – If you do not want to record this in code, you can record it using the first 2 ways and store the json files under mappings and use that as stub data.**