Zephyr Integration (Zapi)

Ostan Dsouza
CodeCraft Technologies
10/06/2020

Table of Contents

Chapter 1: **Introduction**

Zephyr provides a suite of tools to optimize speed and quality of software testing, empowering you with the flexibility, visibility, and insights you need to achieve Continuous Testing Agility. Zephyr for JIRA is a native application that exists in JIRA and brings quality test management capabilities to any JIRA project. When Zephyr is used with JIRA, the test can be created, viewed in any JIRA project, and executed immediately or as part of a testing cycle that may be linked to other issues

Section 1.1: **Features of Zephyr for JIRA**

- Native to JIRA allowing users to test right inside JIRA
    - Testing is integrated into the project cycles and it enables you to track software quality and make empowered go/no-go decisions.
- Create, Plan, and Execute Tests
    - Create, view, and modify test steps and attachments for individual tests. Build test execution cycles, execute the tests, and link defects to specific tests.
- Track Quality Metrics
    - Zephyr provides easy-to-use dashboard which provides testing metrics on the testing activities throughout every project.

Chapter 2: **Zephyr Api(ZAPI)**

ZAPI exposes Zephyr for JIRA data via REST APIs which allows you to build your own integration with Zephyr for JIRA. ZAPI (or Zephyr API) is an Add-on to Zephyr for JIRA that allows access to its testing data including the ability to view and upload information programmatically.

Section 2.1: **Advantages of Zapi**

Using ZAPI's capabilities, Zephyr for JIRA users can:

- Integrate with test automation tools
- Integrate with continuous integration tools
- Create extensive custom reports for testing
- Integrate with business intelligence tools
- Use the testing data for other purposes

Section 2.2: **Prerequisites**

There are a few quick things to be aware of about ZAPI:

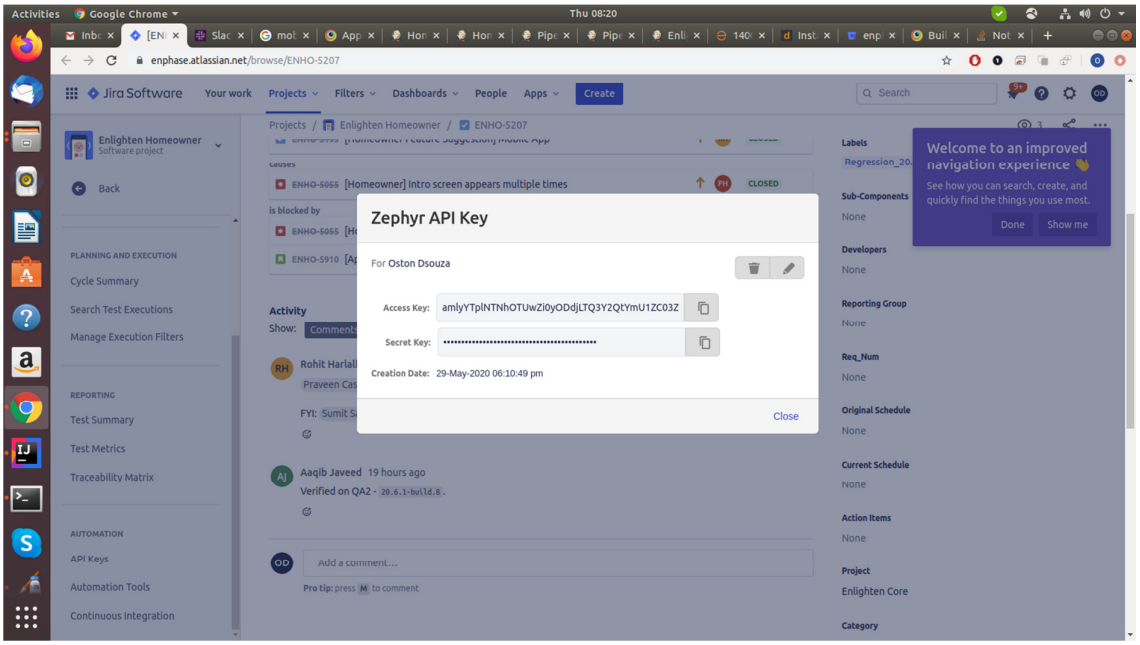
- ZAPI works only with Zephyr for JIRA Server versions 2 and higher
- A valid license of Zephyr for JIRA Server is needed for ZAPI to work
- ZAPI is only available for Zephyr for JIRA Server
- The API is entirely HTTP-based
- Parameters have certain expectations
- There are pagination limits
- Encoding affects status character counts
- No anonymous access is allowed; the security model is the same as that of Zephyr for JIRA (which in turn uses JIRA's security model)

Chapter 3: **ZAPI Integration**

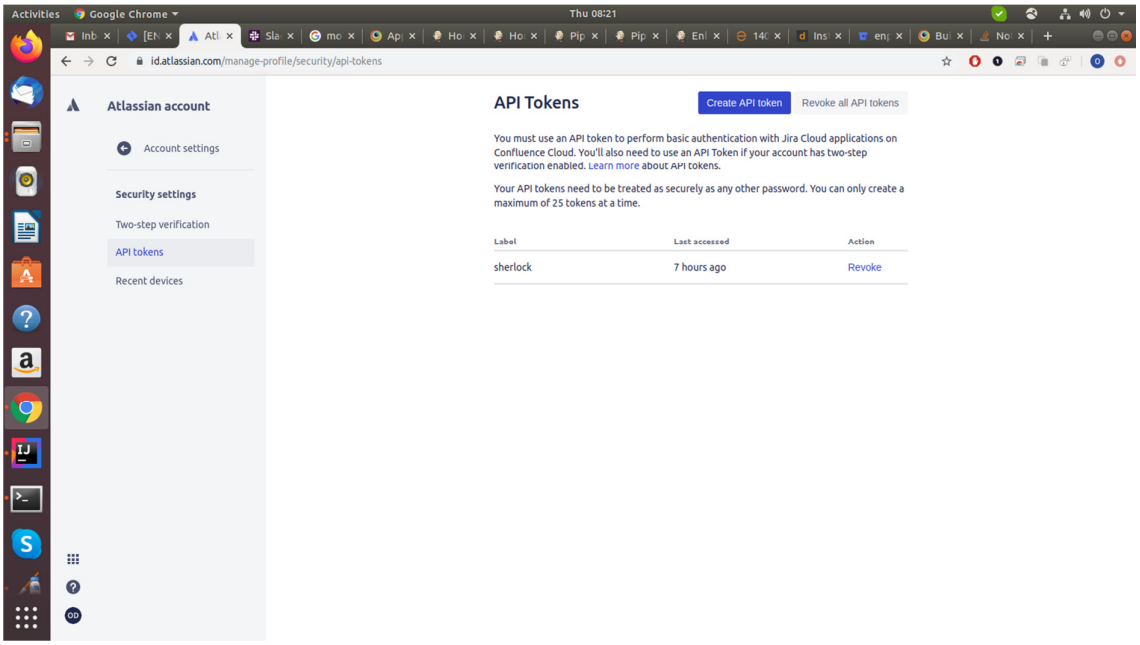
Follow Below Steps for integration zephyr with automation framework.

Section 3.1: **API Keys**

Once After Zapi is installed, you may get api keys for zapi simply by navigating to Zephyr->Automation->API Keys

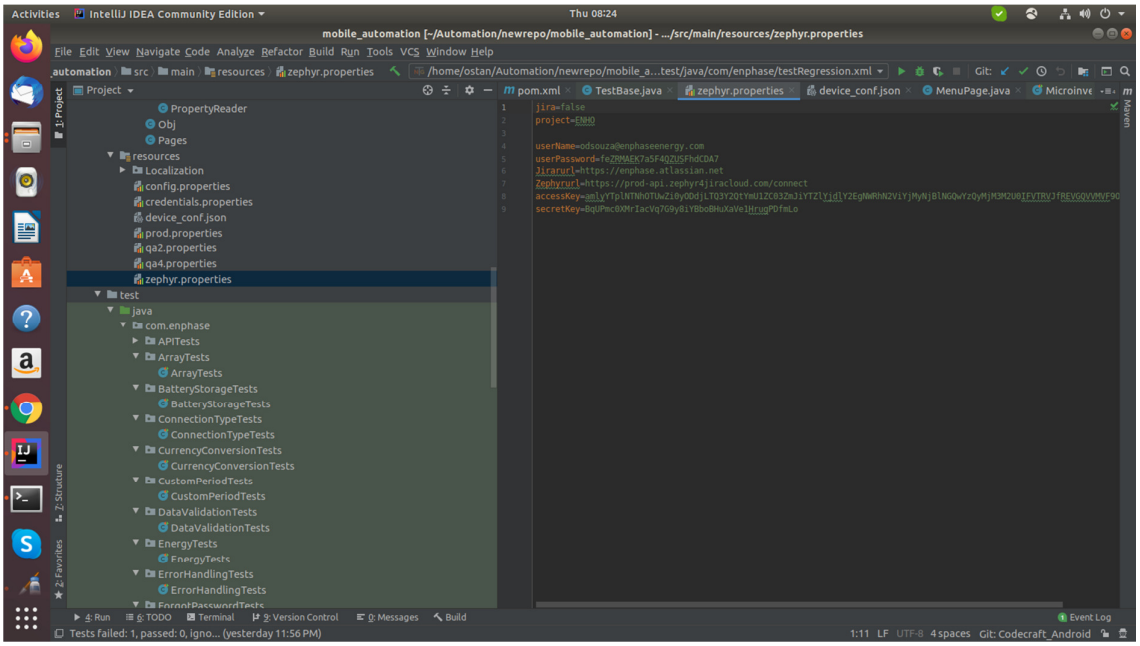


For Jira Authentication, you may generate API token simply by navigating to Accounts settings->Security->Create and manage API Token-> generate API Token



Zephyr Property:

Create Zephyr.property file inside root directory containing user and api tokens required for authentications along with jira flag that can enable or disable this feature.



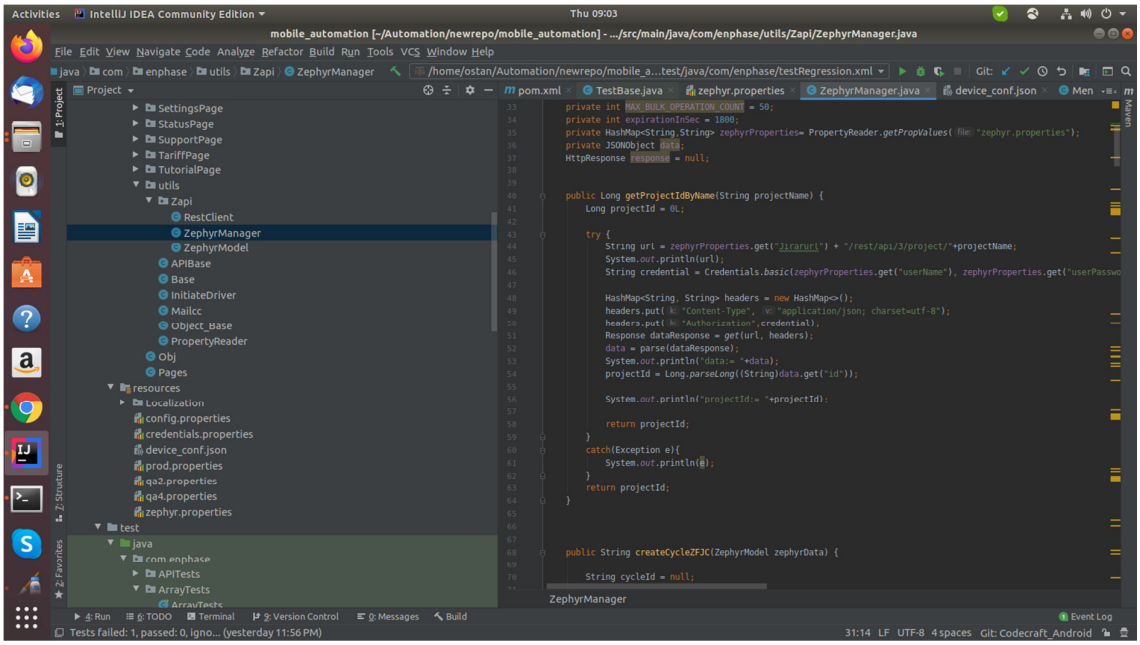
Section 3.2: Creating APIs

a) Get Project ID:

This is REST call to Jira API to get required project ID for specified project name. API document for this can be found here.

GET /rest/api/3/project/{projectIdOrKey}

projectIdOrKey	String	**REQUIRED**

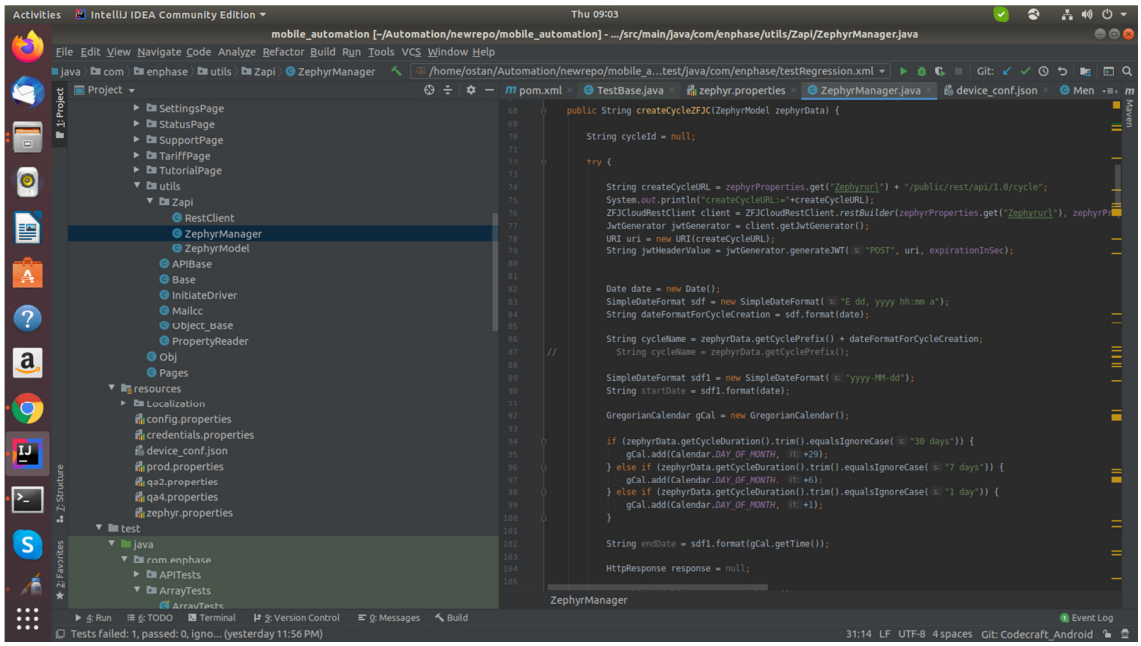


b) Create Cycle:

This is a REST call to zapi to create cycle on zephyr for specified project ID. API document for this can be found here.

https://prod-api.zephyr4jiracloud.com/connect/public/rest/api/1.0/cycle?expand=&clonedCycleId=

name	String	**REQUIRED**
build	String	**OPTIONAL**
environment	String	**OPTIONAL**
description	String	**OPTIONAL**
startDate	String	**OPTIONAL**
endDate	String	**OPTIONAL**
projectId	String	**REQUIRED**
versionId	String	**REQUIRED**

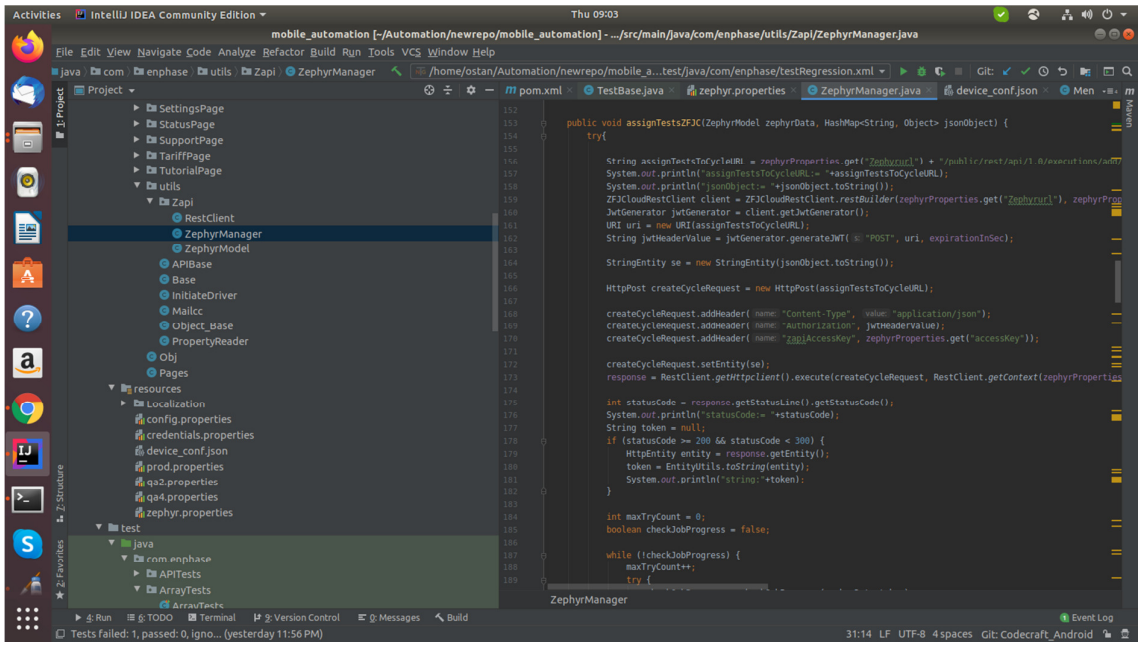


c) Add TestCases:

This is a REST call to zapi to add testcases on zephyr for specified cycle ID. API document for this can be found here

https://prod-api.zephyr4jiracloud.com/connect/public/rest/api/1.0/executions/add/cycle/cycleId

cycleId	String	**REQUIRED**

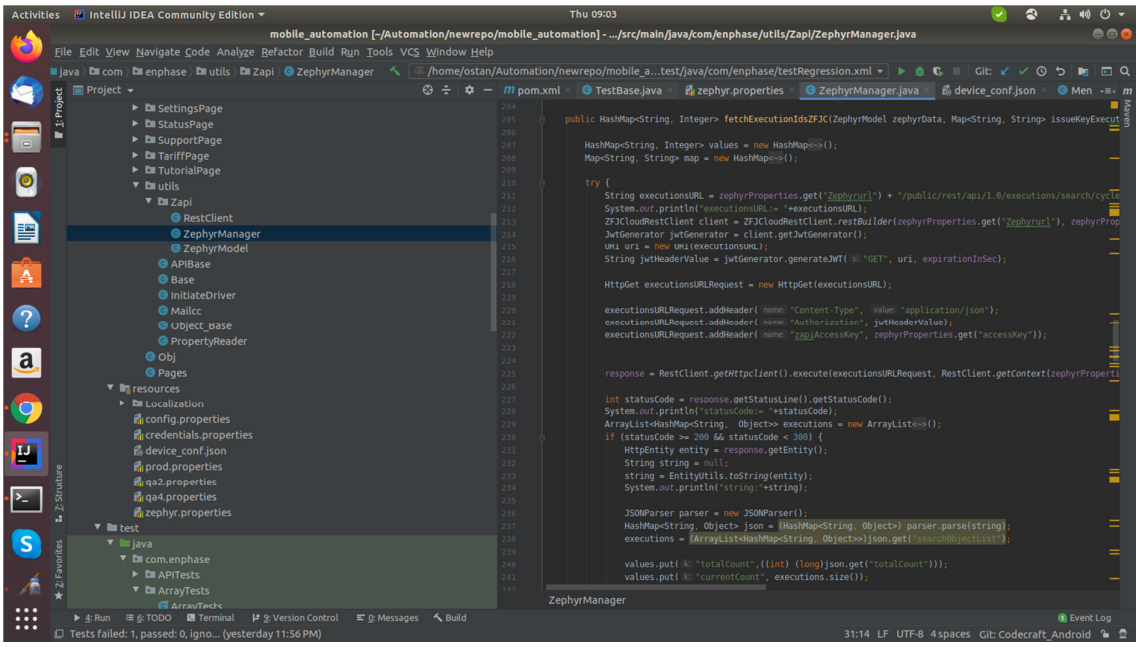

Activities
IntelliJ IDEA Community Edition
Thu 09:03
mobile_automation [~/Automation/newrepo/mobile_automation] - .../src/main/java/com/enphase/utils/Zapi/ZephyrManager.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
java com enphase utils Zapi ZephyrManager
/home/ostan/Automation/newrepo/mobile_a...test/java/com/enphase/testRegression.xml
Git:
Project
SettingsPage
StatusPage
SupportPage
TariffPage
TutorialPage
utils
Zapi
RestClient
ZephyrManager
ZephyrModel
APIBase
Base
InitiateDriver
Mailcc
Object_Base
PropertyReader
Obj
Pages
resources
Localization
config.properties
credentials.properties
device_conf.json
prod.properties
qa2.properties
qa4.properties
zephyr.properties
test
java
com.enphase
APITests
ArrayTests
ArrayTests
pom.xml
TestBase.java
zephyr.properties
ZephyrManager.java
device_conf.json
Men
public void assignTestsZFJC(ZephyrModel zephyrData, HashMap<String, Object> jsonObject) {
try{
String assignTestsToCycleURI = zephyrProperties.get("Zephyrurl") + "/public/rest/api/1.0/executions/add?
System.out.println("assignTestsToCycleURL:= "+assignTestsToCycleURL);
System.out.println("jsonObject:= "+jsonObject.toString());
ZFJCloudRestClient client = ZFJCloudRestClient.restBuilder(zephyrProperties.get("Zephyrurl"), zephyrProp
JwtGenerator jwtGenerator = client.getJwtGenerator();
URI uri = new URI(assignTestsToCycleURL);
String jwtHeaderValue = jwtGenerator.generateJWT(s: "POST", uri, expirationInSec);
StringEntity se = new StringEntity(jsonObject.toString());
HttpPost createCycleRequest = new HttpPost(assignTestsToCycleURL);
createCycleRequest.addHeader(name: "Content-Type", value: "application/json");
createCycleRequest.addHeader(name: "Authorization", jwtHeaderValue);
createCycleRequest.addHeader(name: "zapiAccessKey", zephyrProperties.get("accessKey"));
createCycleRequest.setEntity(se);
response = RestClient.getHttpclient().execute(createCycleRequest, RestClient.getContext(zephyrProperties
int statusCode = response.getStatusLine().getStatusCode();
System.out.println("statusCode:= "+statusCode);
String token = null;
if (statusCode >= 200 && statusCode < 300) {
HttpEntity entity = response.getEntity();
token = EntityUtils.toString(entity);
System.out.println("string:"+token);
}
int maxTryCount = 0;
boolean checkJobProgress = false;
while (!checkJobProgress) {
maxTryCount++;
try {
ZephyrManager
4: Run
6: TODO
Terminal
9: Version Control
0: Messages
Build
Event Log
Tests failed: 1, passed: 0, igno... (yesterday 11:56 PM)
31:14 LF UTF-8 4 spaces Git: Codecraft_Android

d) Fetch ExectionID's:

This is a REST call to zapi to fetch execution ID's for all testcases added to Cycle. API document for this can be found here

https://prod-api.zephyr4jiracloud.com/connect/public/rest/api/1.0/executions/search/cycle/cycleId?versionId=&expand=&offset=&size=&sortOrder=&action=&sortBy=&projectId=

cycleId	String	**REQUIRED**
projectId	String	**REQUIRED**
versionId	String	**REQUIRED**

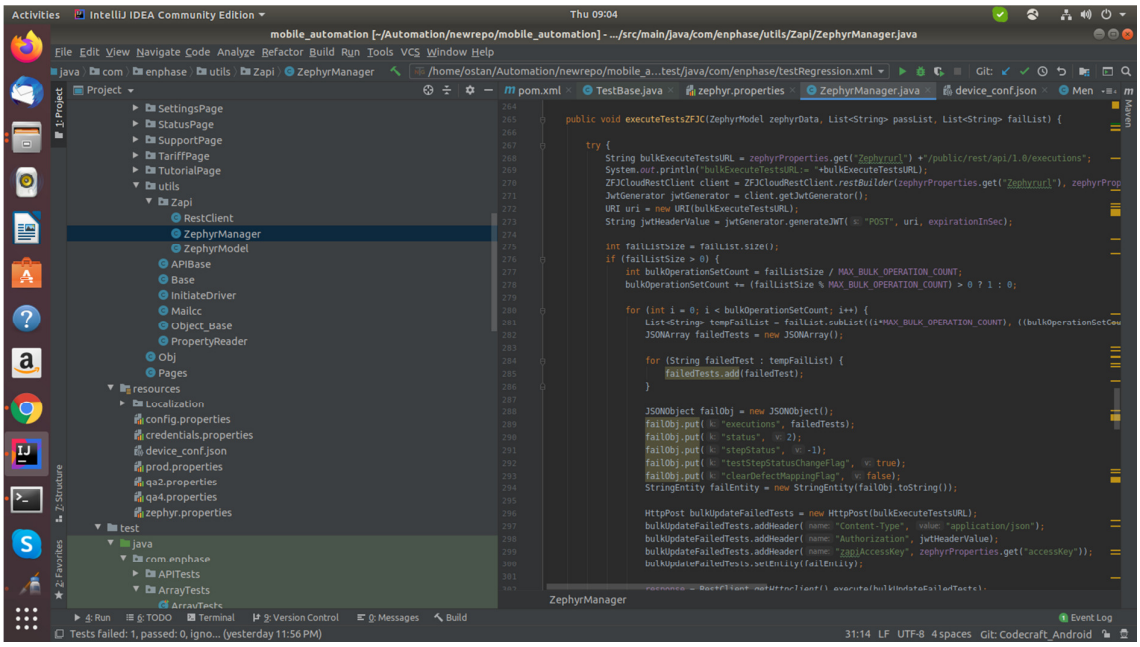


e) Execute Tests:

This is a REST call to zapi to execute test based on execution Id's. API document for this can be found here

https://prod-api.zephyr4jiracloud.com/connect/public/rest/api/1.0/executions

cycleId	String	**REQUIRED**

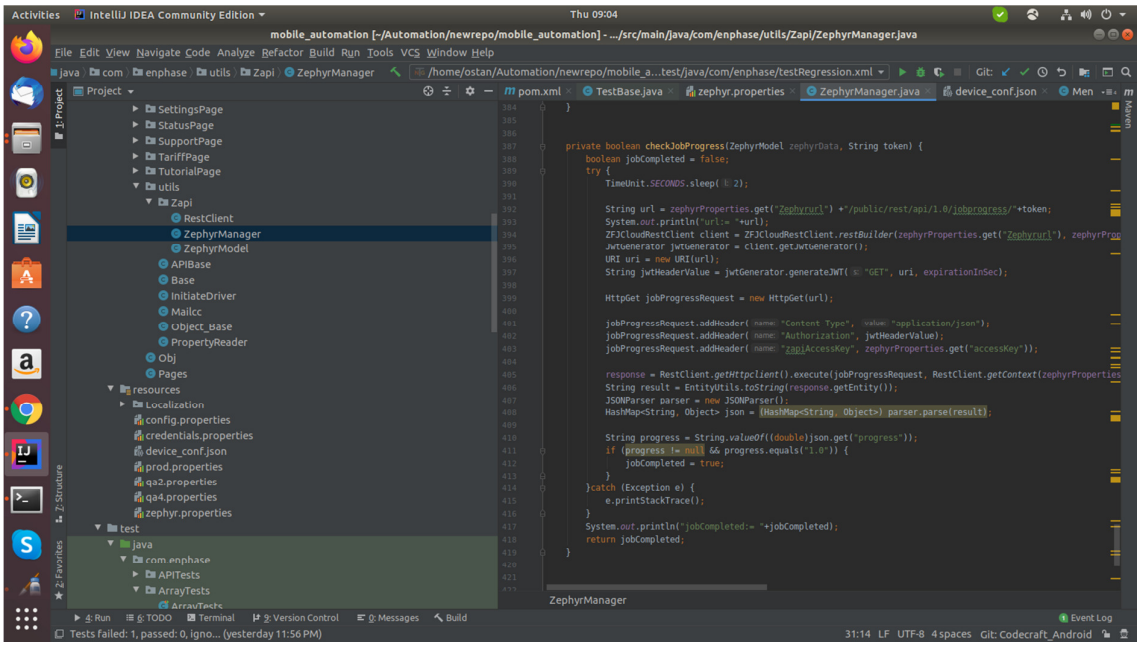


f) Check Progress:

This is a REST call to zapi to get the update on job progress based on job progress token when bulk update is done. API document for this can be found here

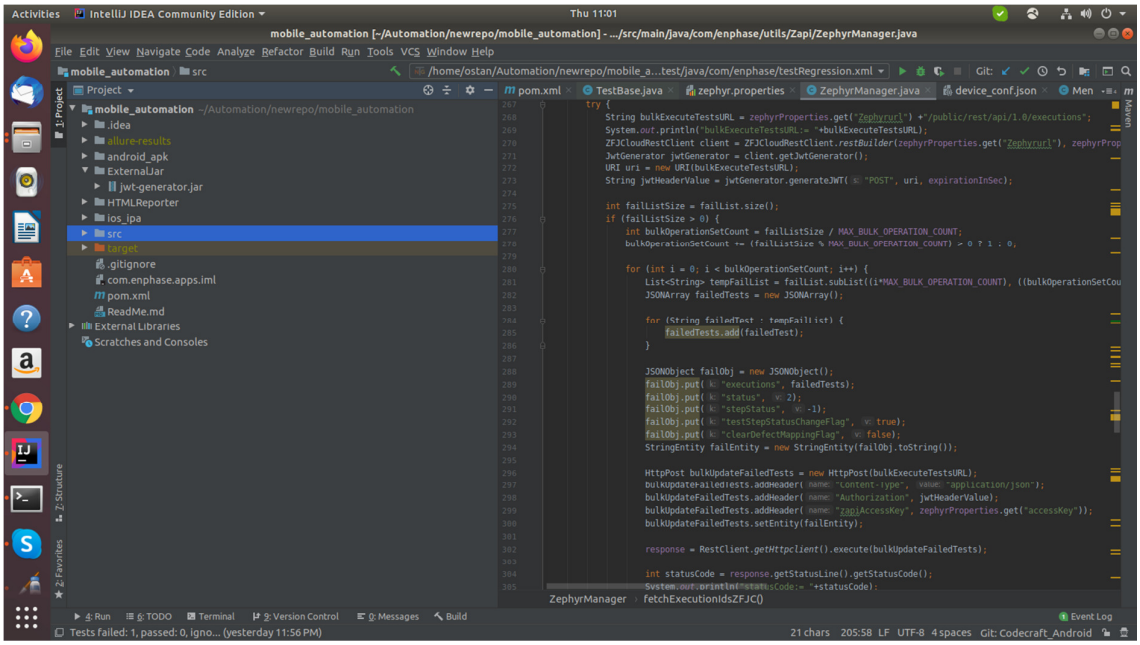
https://prod-api.zephyr4jiracloud.com/connect/public/rest/api/1.0/jobprogress/jobProgressTicket

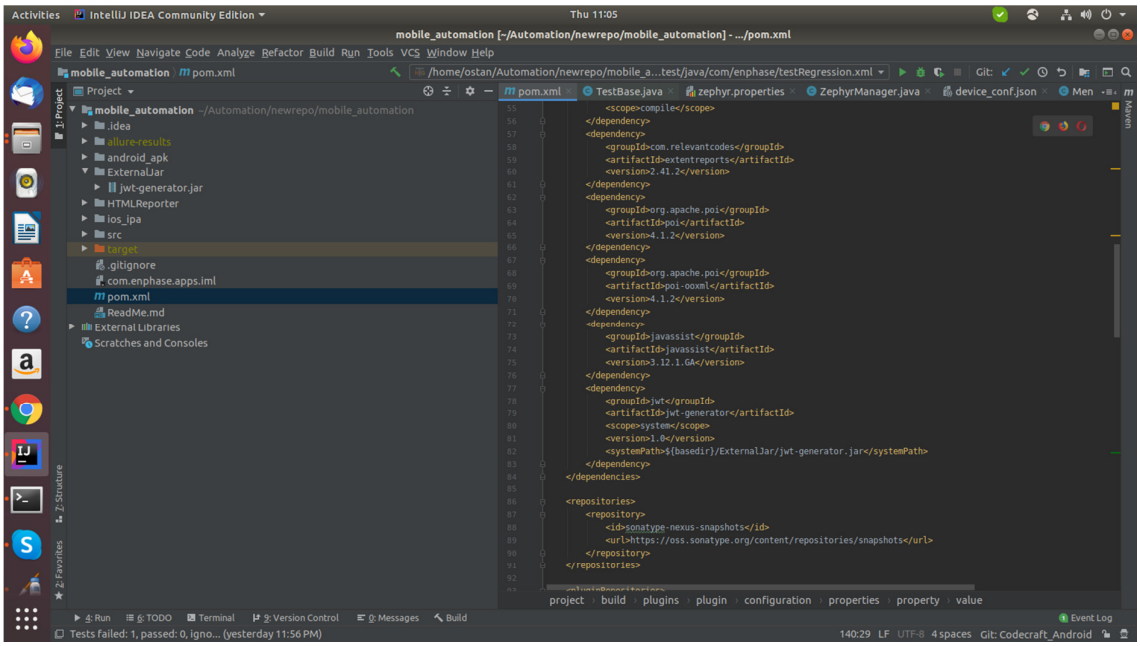
jobProgressTicket	String	**REQUIRED**



Section 3.3: **JWT Token**

Above API calls make use of JWT token for authentication. So here we have made use of external jar provided by zephyr team for this purpose. You can download jar from [here]





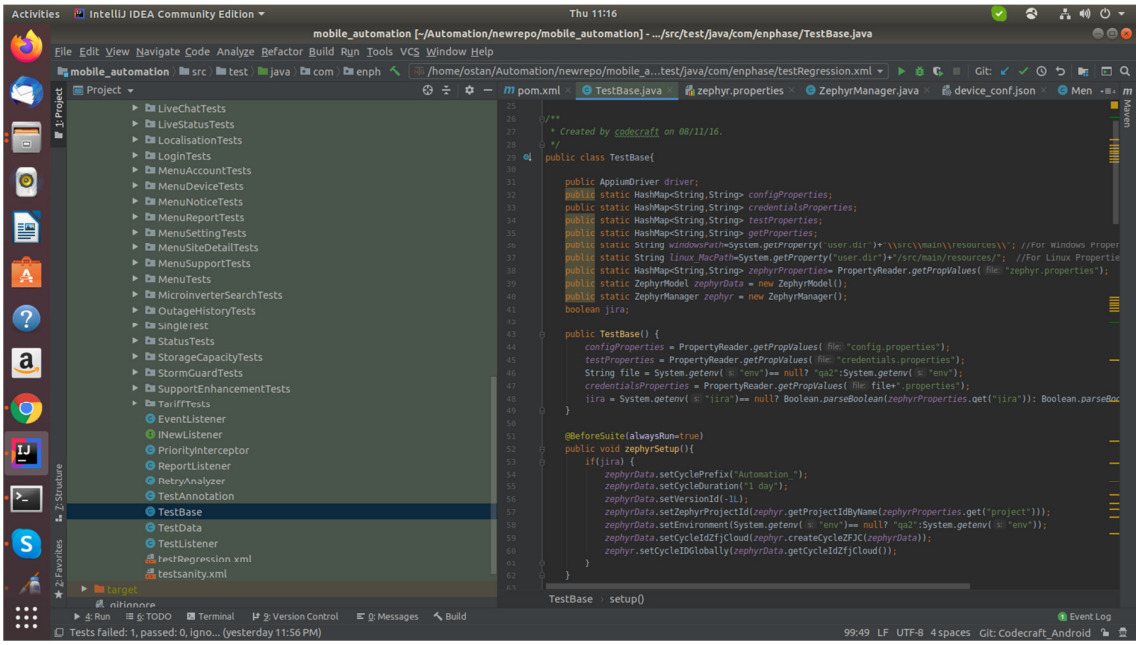
Chapter 4: **Framework Changes**

Below are the steps to be followed to integrate zephyr with automation framework.

Section 4.1: Zephyr Logic

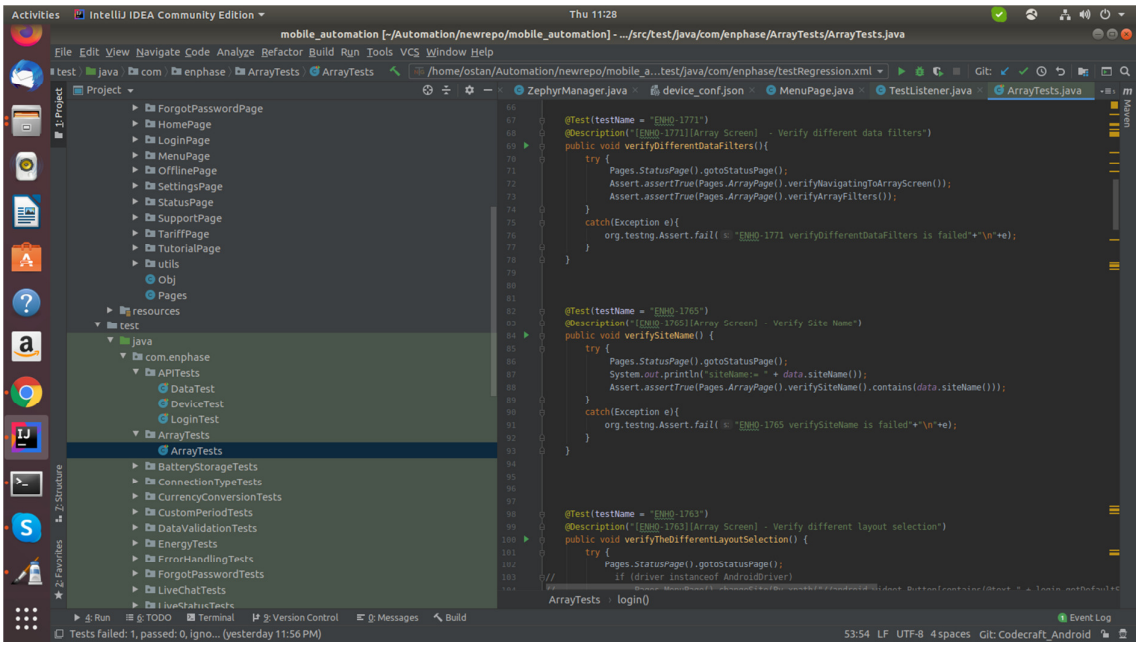
a) Create Zephyr Cycle:

Zephyr cycle create should happen before test case execution starts. Hence we are making use of "**BeforeSuite**" annotation from TestNG for this purpose. And in response Cycle ID is stored in global variable for future use.



b) TestName Paramter:

All testcases must have an testname parameter under @test annotation, which will have testcase ID for identification.



.

c) Making use of iTestListerner Stages:

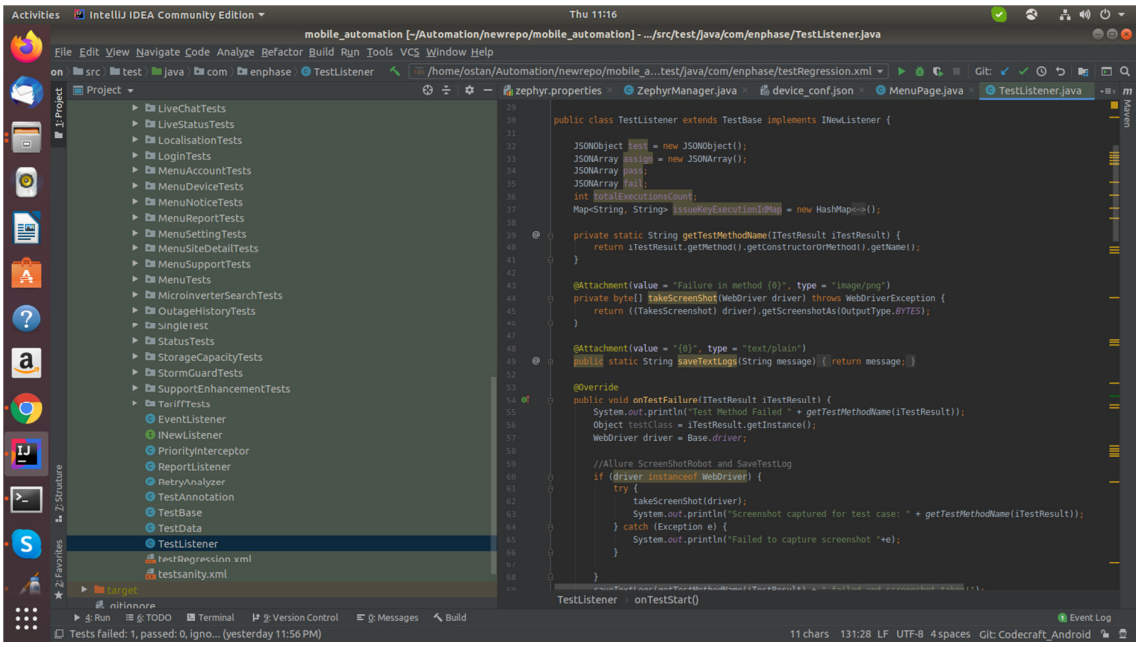
- **OnStart:** In this stage all declaration and initialization of variable that are going to be used further will happen in here.

- **OnTestSuccess:** In this stage test under execution is added to the testcycle by using testname parameter and execution ID's for this test is also fetched add stored in array by name **Pass**.

- **OnTestFailure:** In this stage test under execution is added to the testcycle by using testname parameter and execution ID's for this test is also fetched add stored in array by name **Fail**.

- **OnTestSkipped:** In this stage test under execution is added to the testcycle by using testname parameter and execution ID's for this test is also fetched add stored in array by name **Skip**.

- **OnFinish:** In this stage all test are executed based on array where we had stored test case ID.


Activities
IntelliJ IDEA Community Edition
Thu 11:16
mobile_automation [~/Automation/newrepo/mobile_automation] - .../src/test/java/com/enphase/TestListener.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Project
LiveChatTests
LiveStatusTests
LocalisationTests
LoginTests
MenuAccountTests
MenuDeviceTests
MenuNoticeTests
MenuReportTests
MenuSettingTests
MenuSiteDetailTests
MenuSupportTests
MenuTests
MicroinverterSearchTests
OutageHistoryTests
StatusTests
StorageCapacityTests
StormGuardTests
SupportEnhancementTests
EventListener
INewListener
PriorityInterceptor
ReportListener
RetryAnalyzer
TestAnnotation
TestBase
TestData
TestListener
testsanity.xml
target
zephyr.properties
ZephyrManager.java
device_conf.json
MenuPage.java
TestListener.java
public class TestListener extends TestBase implements INewListener {
JSONObject test = new JSONObject();
JSONArray assign = new JSONArray();
JSONArray pass;
JSONArray fail;
int totalExecutionsCount;
Map<String, String> issueKeyExecutionIdMap = new HashMap<>();
private static String getTestMethodName(ITestResult iTestResult) {
return iTestResult.getMethod().getConstructorOrMethod().getName();
}
@Attachment(value = "Failure in method {0}", type = "image/png")
private byte[] takeScreenShot(WebDriver driver) throws WebDriverException {
return ((TakesScreenshot) driver).getScreenshotAs(OutputType.BYTES);
}
@Attachment(value = "{0}", type = "text/plain")
public static String saveTextLogs(String message) { return message; }
@Override
public void onTestFailure(ITestResult iTestResult) {
System.out.println("Test Method Failed " + getTestMethodName(iTestResult));
Object testClass = iTestResult.getInstance();
WebDriver driver = Base.driver;
//Allure ScreenShotRobot and SaveTestLog
if (driver instanceof WebDriver) {
try {
takeScreenShot(driver);
System.out.println("Screenshot captured for test case: " + getTestMethodName(iTestResult));
} catch (Exception e) {
System.out.println("Failed to capture screenshot "+e);
}
}
TestListener › onTestStart()
4: Run
6: TODO
Terminal
9: Version Control
0: Messages
Build
Event Log
Tests failed: 1, passed: 0, igno... (yesterday 11:56 PM)
11 chars 131:28 LF UTF-8 4 spaces Git: Codecraft_Android