Note-

Always delete the contents of target folder and test output folder before every run so that reports will be nicely generated.

When I deleted target folder, the contents inside it got deleted but not the folder itself. I deleted the test output folders and all data within the test output as well as the test output folder itself got deleted.

Then I ran the project, extent spark pdf and html got generated. Thread report got generated. And re-run also worked properly.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen shot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

# Run the project with re-run plugin-

|  |
| --- |
| **package** parallel;  **import** org.testng.annotations.DataProvider;  **import** io.cucumber.testng.CucumberOptions;  **import** io.cucumber.testng.AbstractTestNGCucumberTests;  @CucumberOptions(  features = {"src/test/resources/parallel"},  glue = { "parallel" },  monochrome=**true**,  plugin = { "pretty",  "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:",  "timeline:test-output-thread/",  "rerun:target/failedScenarios.txt" /\*\*  \* rerun plugin should be used for getting failed cases and then run them individually.  \* it will give one text file, we can give it any name after :  \* it will generate entry of the failed scenario in the text file.  \* then create one more runner to run only the failed cases.  \* **@author** karan  \*  \*/  }  )  **public** **class** ParallelRunner **extends** AbstractTestNGCucumberTests {  /\*\*  \* we need to override a method from abstracttestngcucumbertests class  \* parallel=true means run in parallel, else it will run one by one.  \*  \* **@return** 2d array containing all scenarios.  \*/  @Override  @DataProvider(parallel = **true**)  **public** Object[][] scenarios() {  **return** **super**.scenarios();  }  } |

Refresh the project and failed scenario text file is generated-

A screenshot of a computer

Description automatically generated

This shows the file which got failed. “:8” is the line number where the scenario is present inside the file.

Feature file line number-

A screenshot of a computer screen

Description automatically generated

# Running only failed cases-

Runner file-

|  |
| --- |
| **package** parallel;  **import** org.testng.annotations.DataProvider;  **import** io.cucumber.testng.AbstractTestNGCucumberTests;  **import** io.cucumber.testng.CucumberOptions;  @CucumberOptions(  features = {"@target/failedScenarios.txt"}, // path of the failed text file.  glue = { "parallel" },  monochrome=**true**,  plugin = { "pretty",  "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:",  "timeline:test-output-thread/",  "rerun:target/failedScenarios.txt" // give the location of the text file which failed.  }  )  **public** **class** FailedRuns **extends** AbstractTestNGCucumberTests{  @Override  @DataProvider(parallel = **true**)  **public** Object[][] scenarios() {  **return** **super**.scenarios();  }  } |

Console output-

A screenshot of a computer

Description automatically generated

In case, multiple scenarios are failing and they are all present within one scenario file, then see the way the numbers are assigned-



Feature file with lines 8 and 12-

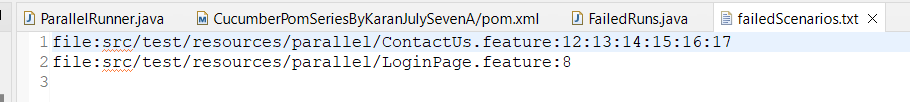
A computer screen shot of a computer screen

Description automatically generated

Note-

As mentioned by great Jatin, target folder, test output folders should go into gitignore.

# See how multiple failures in multiple files are displayed-



# Project structure-

A screenshot of a computer

Description automatically generated

# Codes from this chapter-

|  |
| --- |
| Parallel runner-  **package** parallel;  **import** org.testng.annotations.DataProvider;  **import** io.cucumber.testng.CucumberOptions;  **import** io.cucumber.testng.AbstractTestNGCucumberTests;  @CucumberOptions(  features = {"src/test/resources/parallel"},  glue = { "parallel" },  monochrome=**true**,  plugin = { "pretty",  "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:",  "timeline:test-output-thread/",  "rerun:target/failedScenarios.txt" /\*\*  \* rerun plugin should be used for getting failed cases and then run them individually.  \* it will give one text file, we can give it any name after :  \* it will generate entry of the failed scenario in the text file.  \* then create one more runner to run only the failed cases.  \* **@author** karan  \*  \*/  }  )  **public** **class** ParallelRunner **extends** AbstractTestNGCucumberTests {  /\*\*  \* we need to override a method from abstracttestngcucumbertests class  \* parallel=true means run in parallel, else it will run one by one.  \*  \* **@return** 2d array containing all scenarios.  \*/  @Override  @DataProvider(parallel = **true**)  **public** Object[][] scenarios() {  **return** **super**.scenarios();  }  } |

|  |
| --- |
| Failed runner-  **package** parallel;  **import** org.testng.annotations.DataProvider;  **import** io.cucumber.testng.AbstractTestNGCucumberTests;  **import** io.cucumber.testng.CucumberOptions;  @CucumberOptions(  features = {"@target/failedScenarios.txt"}, // path of the failed text file.  glue = { "parallel" },  monochrome=**true**,  plugin = { "pretty",  "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:",  "timeline:test-output-thread/",  "rerun:target/failedScenarios.txt" // give the location of the text file which failed.  }  )  **public** **class** FailedRuns **extends** AbstractTestNGCucumberTests{  @Override  @DataProvider(parallel = **true**)  **public** Object[][] scenarios() {  **return** **super**.scenarios();  }  } |

|  |
| --- |
| Failed scenario text file-  file:src/test/resources/parallel/ContactUs.feature:12:13:14:15:16:17  file:src/test/resources/parallel/LoginPage.feature:8 |