# Without pretty plugin see the output-

|  |
| --- |
| package testRunners;  import io.cucumber.junit.Cucumber;  import io.cucumber.junit.CucumberOptions;  import org.junit.runner.RunWith;  @RunWith(Cucumber.class)  @CucumberOptions( features = {  "src/test/resources/AppFeatures/Uber.feature" }, tags = "@All", glue = { "StepDefinitions" })  //tags cannot be applied to given, when, then etc etc.  //tags can be applied at feature level. by default it will be applied on all scenarios.  //tags can be applied at scenario level.  //tags can be applied at scenario outline level.  //we can run with multiple tags.  public class UberBookingTest {  } |

Console:

A screenshot of a computer program

Description automatically generated with medium confidence

Pretty will prettily print on output.

# Naveen deleted target folder and tried to run program and got the below error-

A screenshot of a computer

Description automatically generated with medium confidence

To solve this, right click on project -> maven -> update project.

Run again and it should be solved. Some of the file may have been deleted from target folder which is needed for project, so rebuilding the project works in such case.

# Add json and junit report-

|  |
| --- |
| package testRunners;  import io.cucumber.junit.Cucumber;  import io.cucumber.junit.CucumberOptions;  import org.junit.runner.RunWith;  @RunWith(Cucumber.class)  //we want to create pretty, json and junit report in the mentioned location  @CucumberOptions( plugin= {"pretty", "json:target/MyReports/report.json",  "junit:target/MyReports/report.xml"},  features = {"src/test/resources/AppFeatures/Uber.feature" },  tags = "@All",  glue = { "StepDefinitions", "MyHooks"})  //tags cannot be applied to given, when, then etc etc.  //tags can be applied at feature level. by default it will be applied on all scenarios.  //tags can be applied at scenario level.  //tags can be applied at scenario outline level.  //we can run with multiple tags.  public class UberBookingTest {  } |

Refresh project by right clicking on it.

Target folder.

My reports.

Open the reports.

Note- even if we delete the target folder it will be regenerated during the run.

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

# Double click on junit report and it is the same we use daily-

A screenshot of a computer

Description automatically generated

# Now cucumber has given option to publish report-

This has come from cucumber 6.9.

A screenshot of a computer

Description automatically generated with low confidence

The above message comes in console when we run the test.

We can use any of the four options cucumber is suggesting.

# Publish cucumber reports-

Runner:

|  |
| --- |
| package testRunners;  import io.cucumber.junit.Cucumber;  import io.cucumber.junit.CucumberOptions;  import org.junit.runner.RunWith;  @RunWith(Cucumber.class)  //we want to create pretty, json and junit report in the mentioned location  @CucumberOptions( plugin= {"pretty", "json:target/MyReports/report.json",  "junit:target/MyReports/report.xml"},  features = {"src/test/resources/AppFeatures/Uber.feature" },  tags = "@All",  glue = { "StepDefinitions", "MyHooks"},  publish=true)  //tags cannot be applied to given, when, then etc etc.  //tags can be applied at feature level. by default it will be applied on all scenarios.  //tags can be applied at scenario level.  //tags can be applied at scenario outline level.  //we can run with multiple tags.  public class UberBookingTest {  } |

Console:

Now it says this message in console.

A screenshot of a computer

Description automatically generated with medium confidence

Open url and see the beautiful report:

A screenshot of a computer

Description automatically generated

To see the feature file, expand the small arrow:

A screenshot of a computer

Description automatically generated

# See the cucumber versions needed when you are working with different languages to publish the report-

A screenshot of a computer

Description automatically generated with medium confidence

# FAQs-

A screenshot of a computer

Description automatically generated with medium confidence

# Purposely failed a before hook method-

A screenshot of a computer program

Description automatically generated with low confidence

Run the runner:

Open report.

A screenshot of a computer

Description automatically generated

Wherever the “before hook” was used those scenarios failed, rest of the scenarios where “before” was not used they are passed. These reports get auto deleted after a day.

A screenshot of a computer program

Description automatically generated with low confidence

# Another way to publish the reports-

We used the fourth option earlier. Now lets go with the first option.

A screenshot of a computer

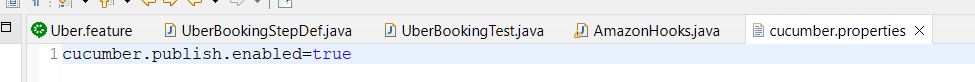
Description automatically generated with low confidence

Create properties file.

A screenshot of a computer

Description automatically generated with medium confidence

Mention this property and save it.



Run feature file.

It ran -> go to the file path.

A screenshot of a computer

Description automatically generated

Report generated.

A screenshot of a computer

Description automatically generated

Good coding practice is to maintain the property in separate file like above.

# Monochrome-

Alignment not proper in console or console messed up.

This is for console formatting.

Runner file:

|  |
| --- |
| package testRunners;  import io.cucumber.junit.Cucumber;  import io.cucumber.junit.CucumberOptions;  import org.junit.runner.RunWith;  @RunWith(Cucumber.class)  //we want to create pretty, json and junit report in the mentioned location  @CucumberOptions( plugin= {"pretty", "json:target/MyReports/report.json",  "junit:target/MyReports/report.xml"},  features = {"src/test/resources/AppFeatures/Uber.feature" },  tags = "@All",  glue = { "StepDefinitions", "MyHooks"},  monochrome=true)  //tags cannot be applied to given, when, then etc etc.  //tags can be applied at feature level. by default it will be applied on all scenarios.  //tags can be applied at scenario level.  //tags can be applied at scenario outline level.  //we can run with multiple tags.  public class UberBookingTest {  } |

Console :

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence

Now monochrome is not used much.

Cucumber 6 and above is already giving color and better option.

# Dry run-

Runner file:

|  |
| --- |
| package testRunners;  import io.cucumber.junit.Cucumber;  import io.cucumber.junit.CucumberOptions;  import org.junit.runner.RunWith;  @RunWith(Cucumber.class)  //we want to create pretty, json and junit report in the mentioned location  @CucumberOptions( plugin= {"pretty", "json:target/MyReports/report.json",  "junit:target/MyReports/report.xml"},  features = {"src/test/resources/AppFeatures/Uber.feature" },  tags = "@All",  glue = { "StepDefinitions", "MyHooks"},  monochrome=false,  dryRun =true  )  //tags cannot be applied to given, when, then etc etc.  //tags can be applied at feature level. by default it will be applied on all scenarios.  //tags can be applied at scenario level.  //tags can be applied at scenario outline level.  //we can run with multiple tags.  public class UberBookingTest {  } |

We added this one step in feature file:

A picture containing text, screenshot, font, number

Description automatically generated

Now when we run the runner it will not give error, but dry run will point this step to null indicating there are no step definitions to the line.

A screenshot of a computer

Description automatically generated with medium confidence

# When you make dry run as false, nothing changes except in left hand side we get the step to implement-

A screenshot of a computer program

Description automatically generated with medium confidence

Console:

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with low confidence

# With true also the same thing is applicable-

Dry run is depreciated seems like.

A screenshot of a computer

Description automatically generated

Useful for sharing this runner file amongst teams as this mapping helps to track feature files with step definition files.

# Strict-

If you want to ignore the error, like for example, you wrote feature but no step definition then use strict=false.

But strict is removed now. We need to add in runner in cucumber options.



Strict=true means strictly follow the rules. Strict =false means no need to follow the rules.

A screenshot of a computer

Description automatically generated with medium confidence

It gives failure trace that strict is out.

# Use strict=true and it works regularly-

A screenshot of a computer

Description automatically generated

# Project structure-

A screenshot of a computer

Description automatically generated with medium confidence

# Codes used-

Feature file.

|  |
| --- |
| @All  Feature: Uber Booking Feature  @Smoke  Scenario: Booking Cab Sedan  Given User selects car type "Sedan" from app  When user selects car as "Sedan" and pick up as "bangalore" and drop as "pine"  Then driver starts the journey  And driver ends the journey  Then user pays 100 us dollars  #Then user checks the trip details  @Regression @Smoke  Scenario: Booking Cab SUV  Given User selects car type "suv" from app  When user selects car as "suv" and pick up as "bangalore" and drop as "hyderabad"  Then driver starts the journey  And driver ends the journey  Then user pays 100 us dollars  @Prod  Scenario: Booking Cab MINI  Given User selects car type "mini" from app  When user selects car as "mini" and pick up as "mumbai" and drop as "pine"  Then driver starts the journey  And driver ends the journey  Then user pays 100 us dollars |

Runner.

|  |
| --- |
| package testRunners;  import io.cucumber.junit.Cucumber;  import io.cucumber.junit.CucumberOptions;  import org.junit.runner.RunWith;  @RunWith(Cucumber.class)  //we want to create pretty, json and junit report in the mentioned location  @CucumberOptions( plugin= {"pretty", "json:target/MyReports/report.json",  "junit:target/MyReports/report.xml"},  features = {"src/test/resources/AppFeatures/Uber.feature" },  tags = "@All",  glue = { "StepDefinitions", "MyHooks"},  monochrome=false,  dryRun =true,  strict=true  )  //tags cannot be applied to given, when, then etc etc.  //tags can be applied at feature level. by default it will be applied on all scenarios.  //tags can be applied at scenario level.  //tags can be applied at scenario outline level.  //we can run with multiple tags.  public class UberBookingTest {  } |

Hooks.

|  |
| --- |
| package MyHooks;  import io.cucumber.java.After;  import io.cucumber.java.AfterStep;  import io.cucumber.java.Before;  import io.cucumber.java.BeforeStep;  import io.cucumber.java.Scenario;  public class AmazonHooks {  @Before("@Smoke")  public void setUpBrowser(Scenario sc) {  System.out.println("launching chrome brower");  System.out.println(sc.getName());  //int i=9/0;  }  @Before(order = 2)  public void setUpURL() {  System.out.println("launching url");  }  @After("@Regression")  public void tearDownClose(Scenario sc) {  System.out.println("close the browser");  System.out.println(sc.getName());  }  @After(order = 2)  public void tearDownLogout() {  System.out.println("logout from the application");  }  // @BeforeStep  // public void takeScreenshot() {  // System.out.println("takes screenshot after every step");  // }  //  // @AfterStep  // public void refreshBrowser() {  // System.out.println("refresh the browser after every step");  // }  } |

Cucumber.properties.

|  |
| --- |
| cucumber.publish.enabled=true |