Sexy Using Cucumber - BDD in your project

Presented by :

Mr. Vu Nguyen

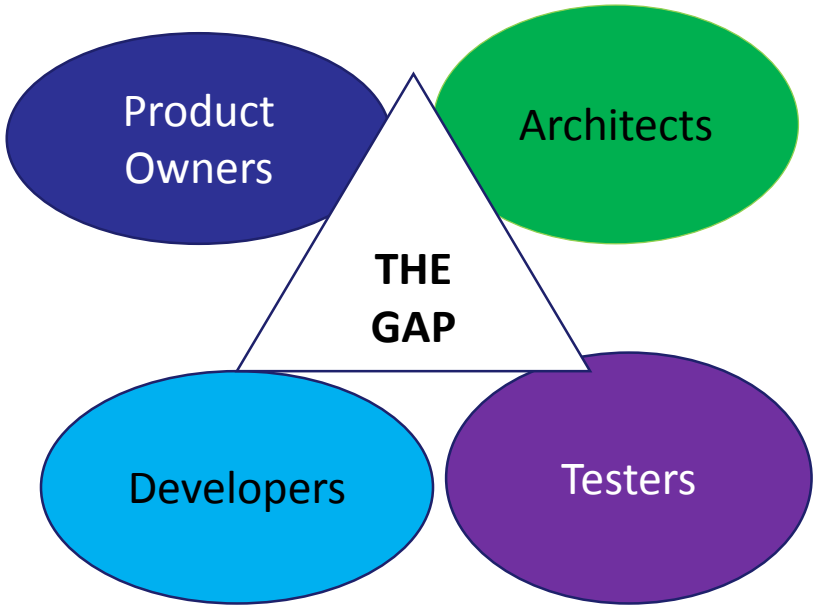
Senior Automation Engineer

- Working as an QA Automation
- Over 8 years experience in software engineering experience
- Skype ID: vunguyen2588
- Email: hoaivunguyenpham@gmail.com

Agenda

- ❑ Behavior Driven Development
- ❑ Cucumber
- ❑ Advantages of Cucumber
- ❑ Environment Setup
- ❑ Gherkins Language
- ❑ Cucumber Execution Flow
- ❑ Runner Class
- ❑ Features
- ❑ Steps Definitions
- ❑ Cucumber Data
- ❑ Cucumber Report
- ❑ Creating project cucumber with maven

Behavior Driven Development



A set of test scenarios with the simple english language

Implementing an application by describing its behaviour by the perspective of its stakeholders

Cucumber

Cucumber

- THE OPEN SOURCE TOOL
- SUPPORTS BDD
- SUPPORTS MANY DIFFERENT LANGUAGES
- FEATURE FILE
- STEP DEFINITION
- RUNNER CLASS

Advantages of Cucumber

Environment Setup

Gherkins Language

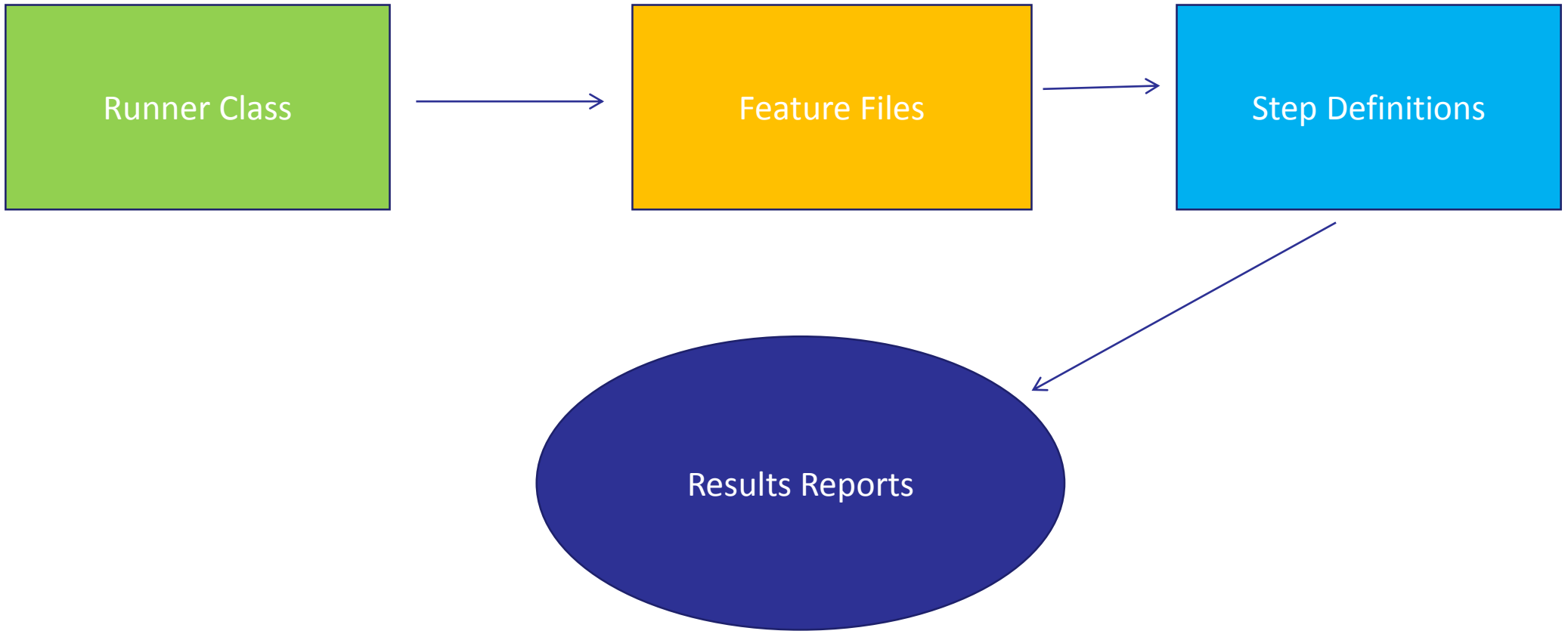
Is a language, which is used to write features, scenarios and steps.

Have the extension .feature

Define set of keywords

Each keyword has it own significance: Feature, Scenarios, Given, When, Then, Add, But, Background ...

Cucumber Execution Flow



Runner Class

```
package sample;
import org.junit.runner.RunWith;
import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(
features = src/test/java/sample/Demo.feature",
format = { "pretty", "html:target/cucumber-reports" }
)

public class RunnerClass {

}
```

Tells JUnit that tests should run using Cucumber class present in cucumber.api.junit package

Tells Cucumber a lot of things like where to look for feature files, what reporting system to use and some other things also

- features: provides the location of the feature file
- glue: provides the path of the step definition class
- format: all report formaters to use
- tags: what tags in the features file should be executed

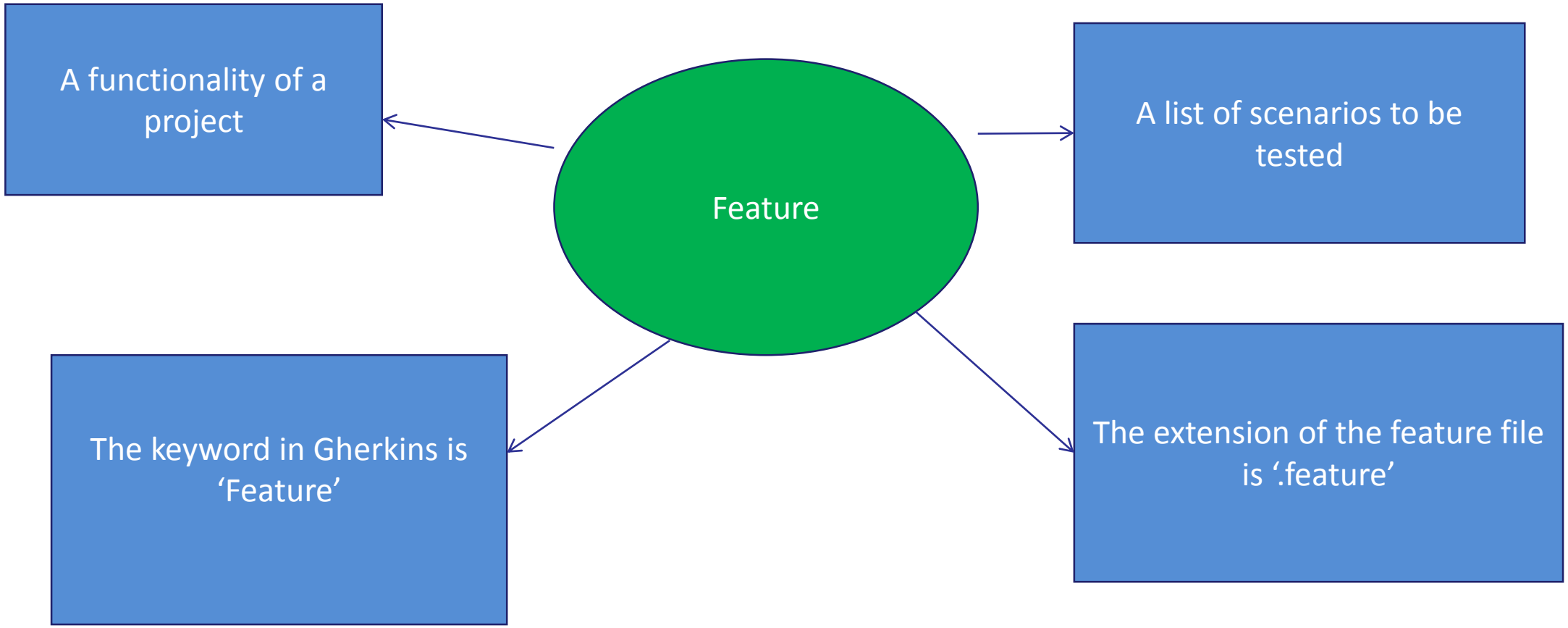
Runner Class

As main class in typical java program from where the execution starts

An interlink between feature files and step definition classes

Multiple types of test runners such as JUnit runner, CLI runner, Android runner ...

Features


A functionality of a project
Feature
A list of scenarios to be tested
The keyword in Gherkins is ‘Feature’
The extension of the feature file is ‘.feature’

Features

Feature:
As a user
I want to be able to login to gmail
So that i can view my email

Name of the feature under test

Describe about feature under test - optional

Scenario: *The user login to gmail*

What is the test scenario

Given *The user navigate to gmail page*

Prerequisite before the test steps get executed

When *The user click on Sign in button*
And *The user input the right username*
And *The user input the right password*

Specific condition which should match in order to execute the next step

And *The user click on Login button*

Keyword is used to show conjunction between two conditions

Then *The user login to gmail successfully*

What should happen

Steps Definitions

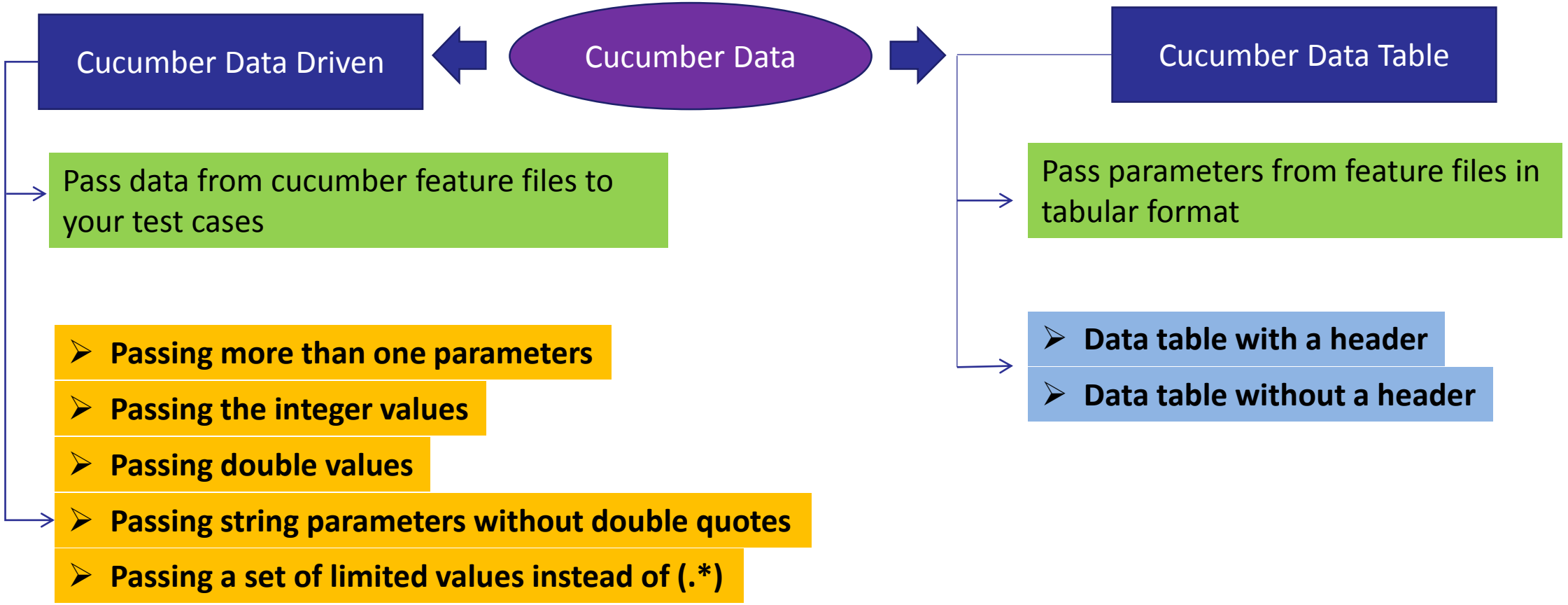
The mapping between each step of the scenario defined in the feature file with a code of function to be executed

```
@And ("^I create a new account$")
public void iCreateNewAccount() {
    NewAccount objNewAccount = new NewAccount(obj);
    objNewAccount.clickNewAccount();
    objNewAccount.createNewAccount(customerID, "30000");
    accountID = objNewAccount.getAccountID();
    Assert.assertTrue(objNewAccount.getSuccessMessage().contains("Account Generated Successfully!!!"));
}
```

This function can be written both Java and Selenium commands

```
@And ("^I deposit to new account$")
public void iDepositToNewAccount() {
    Deposit objDeposit = new Deposit(obj);
    objDeposit.clickDeposit();
    objDeposit.createDeposit(accountID, "2000", "abc");
    Assert.assertTrue(objDeposit.getSuccessMessage().contains("Transaction details of Deposit for Account " + accountID));
}

@Then("^I close the browser$")
public void close_browser() {
    obj.quit();
}
```

Cucumber Data



Passing more than one parameters

```
And I login with user "abc" and password "123456"
```

On feature file, we use
“(string)”

```
@And("^I login with user \"([^\"]*)\" and password \"([^\"]*)\"$")
public void loginToSystem(String username, String password) {
    System.out.println(username);
    System.out.println(password);
}
```

On step definition file, we use
\”([^\”]*)\”

Passing the integer values

```
And I want to verify that 2 plus 3 equals 5
```

On feature file, we use (int)

```
@And("^I want to verify that (\\d+) plus (\\d+) equals (\\d+)$")
public void i_want_to_verify_sum_of_numbers(int firstNum, int secondNum, int sum)  {
    System.out.println(firstNum);
    System.out.println(secondNum);
    System.out.println(sum);
}
```

On step definition file, we use (\\d+)

Passing double values

```
And I want to verify that 2.2 plus 3.3 equals 5.5
```

On feature file, we use (double)

```
@And("^I want to verify that (\\d+.\\d+) plus (\\d+.\\d+) equals (\\d+.\\d+)$")
public void i_want_to_verify_sum_of_numbers(double firstNum, double secondNum, double sum)  {
    System.out.println(firstNum);
    System.out.println(secondNum);
    System.out.println(sum);
}
```

On step definition file, we use (\\d+.\\d+)

Passing string parameters without double quotes

```
And I login with user abc and password 123456
```

On feature file, we use (string)

```
@And("^I login with user (.*) and password (.*)$")
public void loginToSystem(String username, String password)  {
    System.out.println(username);
    System.out.println(password);
}
```

On step definition file, we use (.*)

Passing a set of limited values

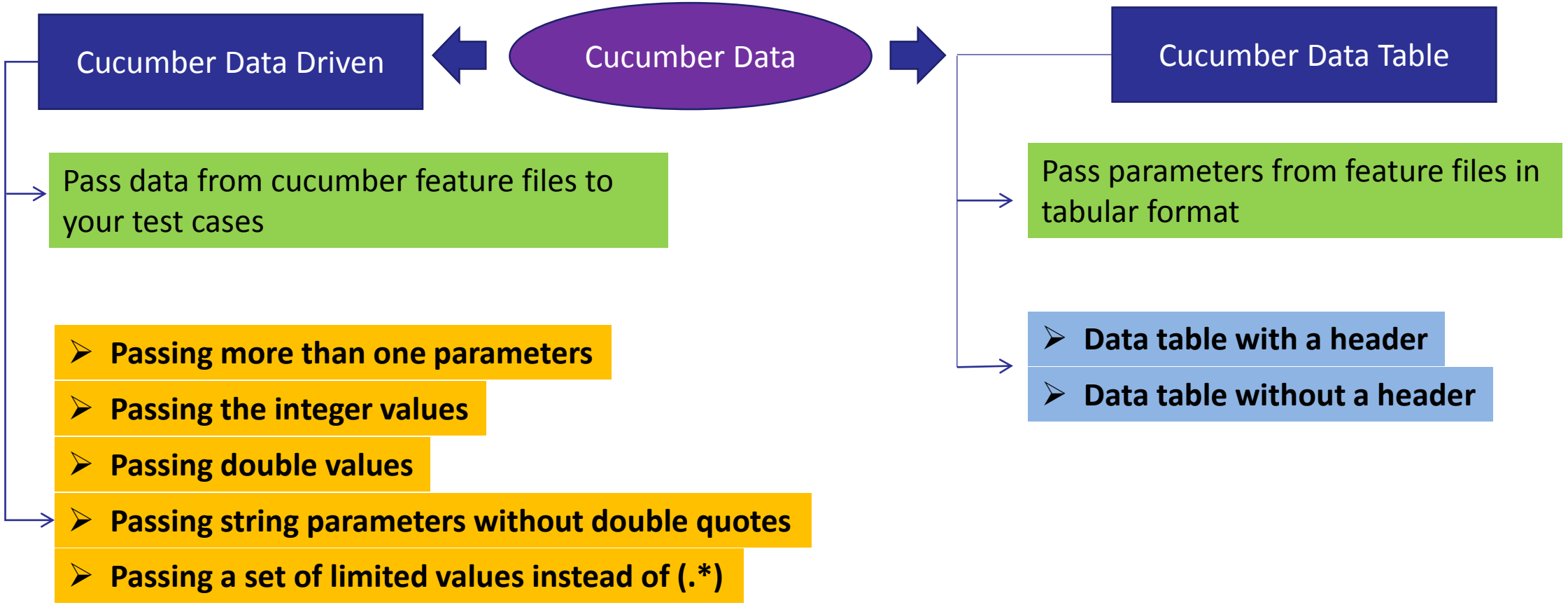
```
And I open application in Chrome browser
```

On feature file, we use
(String|String|....)

```
@And("^I open application in (Chrome|Firefox|Safari) browser$")
public void loginToSystem(String browser)  {
    if(browser.equalsIgnoreCase("Chrome")) {
        //  Code to launch Chrome
    } else if(browser.equalsIgnoreCase("Firefox")) {
        //  Code to launch Firefox
    } else if(browser.equalsIgnoreCase("Safari")) {
        //  Code to launch Safari
    }
}
```

On step definition file, we use
(String)

Cucumber Data



Cucumber data table without header

```
And I login with following credentials
    | admin | pass1234 |
```

On feature file, we use
(|values|values |...)

```
@And("^I login with following credentials$")
public void loginWithFollowingCredentials(DataTable dt)  {
    List<String> list = dt.asList(String.class);
    System.out.println("Username: " + list.get(0));
    System.out.println("Password: " + list.get(1));
}
```

On step definition file, we use (DataTable)
List<String> list = dt.asList(String.class);

Cucumber data table with header

```
And I create the new account form with the following data
    | First Name | Last Name | Phone No    | Password | DOB Year  | Gender |
    | Test FN    | Test LN   | 0123123123 | Pass1234 | 1990      | Male   |
```

On feature file, we use

|Header 1|Header 2|...|
|values |values |...|

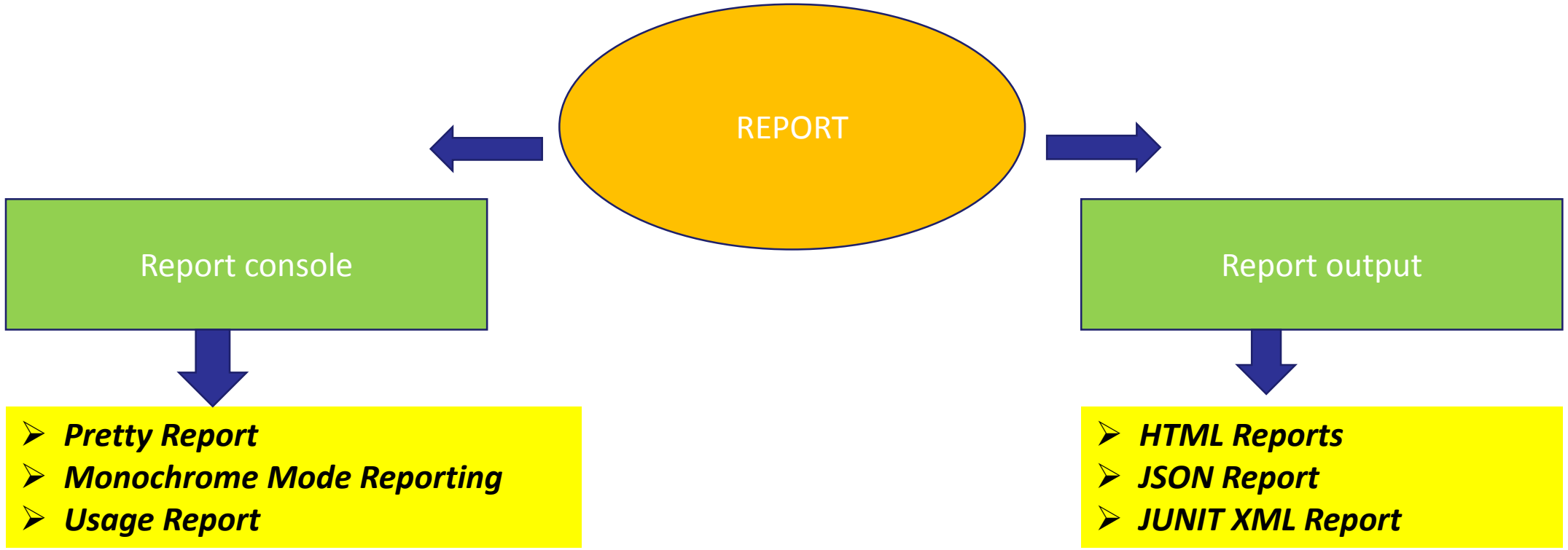
```
@And("^I create the new account form with the following data$")
public void createNewAccountWithFollowingData(DataTable dt)  {
    List<Map<String, String>> list = dt.asMaps(String.class, String.class);
    System.out.println(list.get(0).get("First Name"));
    System.out.println(list.get(0).get("Last Name"));
    System.out.println(list.get(0).get("Phone No"));
    System.out.println(list.get(0).get("Password"));
    System.out.println(list.get(0).get("DOB Year"));
    System.out.println(list.get(0).get("Gender"));
}

@And("^I create the new account form with the following data$")
public void createNewAccountWithFollowingData(DataTable dt)  {
    List<List<String>> list = dt.asLists(String.class);
    //  i starts from 1 because i=0 represents the header
    for(int i=1; i<list.size(); i++) {
        System.out.println(list.get(i).get(0));
        System.out.println(list.get(i).get(1));
        System.out.println(list.get(i).get(2));
        System.out.println(list.get(i).get(3));
        System.out.println(list.get(i).get(4));
        System.out.println(list.get(i).get(5));
    }
}
```

On step definition file, we use (DataTable)

List<Map<String, String>> list = dt.asMaps(String.**class,** String.**class);**

List<List<String>> list = dt.asLists(String.**class);**

Reports


REPORT
Report console
Report output
Pretty Report
Monochrome Mode Reporting
Usage Report
HTML Reports
JSON Report
JUNIT XML Report

Reports Console

```java
package sample;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "src/test/java/sample/ReportSample.feature",
        plugin = { "pretty" }
        )

public class RunnerClass {

}
```

```
Scenario:                    [90m# src/test/java/sample/ReportSample.feature:2[0m
  User login to Guru99

  [32mGiven [0m[32mI navigate to the guru99[0m [90m# DefinitionGuru.iNavigateGuru99()[0m
  [32mAnd [0m[32mI login to the Guru99[0m      [90m# DefinitionGuru.iLoginGuru99()[0m
  [32mThen [0m[32mI close the browser[0m       [90m# DefinitionGuru.close_browser()[0m

1 Scenarios ([32m1 passed[0m)
3 Steps ([32m3 passed[0m)
1m2.273s
```

```java
package sample;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "src/test/java/sample/ReportSample.feature",
        plugin = { "pretty" },
        monochrome = true
        )

public class RunnerClass {

}
```

```
Scenario:                       # src/test/java/sample/ReportSample.feature:2
  User login to Guru99

  Given I navigate to the guru99 # DefinitionGuru.iNavigateGuru99()
  And I login to the Guru99      # DefinitionGuru.iLoginGuru99()
  Then I close the browser       # DefinitionGuru.close_browser()

1 Scenarios (1 passed)
3 Steps (3 passed)
0m31.573s
```

Reports Console

```
1  package sample;
2  import org.junit.runner.RunWith;
5
6  @RunWith(Cucumber.class)
7  @CucumberOptions(
8          features = "src/test/java/sample/ReportSample.feature",
9          plugin = { "usage" },
10         monochrome = true
11         )
12
13 public class RunnerClass {
14
15 }
16
```

```
{
  "source": "^I navigate to the guru99$",
  "steps": [
    {
      "name": "I navigate to the guru99",
      "aggregatedDurations": {
        "average": 16.802061948,
        "median": 16.802061948
      },
      "durations": [
        {
          "duration": 16.802061948,
          "location": "src/test/java/sample/ReportSample.feature:4"
        }
      ]
    }
  ]
},
```

Report Output

```java
package sample;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "src/test/java/sample/ReportSample.feature",
        plugin = { "pretty", "html:target/cucumber-reports" },
        monochrome = true
        )

public class RunnerClass {

}
```

Feature: Guru Testing

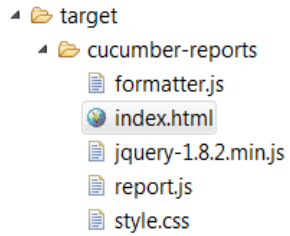
Scenario:

User login to Guru99

Given I navigate to the guru99

And I login to the Guru99

Then I close the browser




```java
package sample;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "src/test/java/sample/ReportSample.feature",
        plugin = { "pretty", "json:target/cucumber-reports/Cucumber.json" },
        monochrome = true
        )

public class RunnerClass {

}
```

```json
[
  {
    "line": 1,
    "elements": [
      {
        "line": 2,
        "name": "",
        "description": "User login to Guru99",
        "id": "guru-testing;",
        "type": "scenario",
        "keyword": "Scenario",
        "steps": [
          {
            "result": {
              "duration": 15895881236,
              "status": "passed"
            },
            "line": 4,
            "name": "I navigate to the guru99",
            "match": {
              "location": "DefinitionGuru.iNavigateGuru99()"
            },
            "keyword": "Given "
          },
          {
            "result": {
              "duration": 3664984731,
              "status": "passed"
            },
```

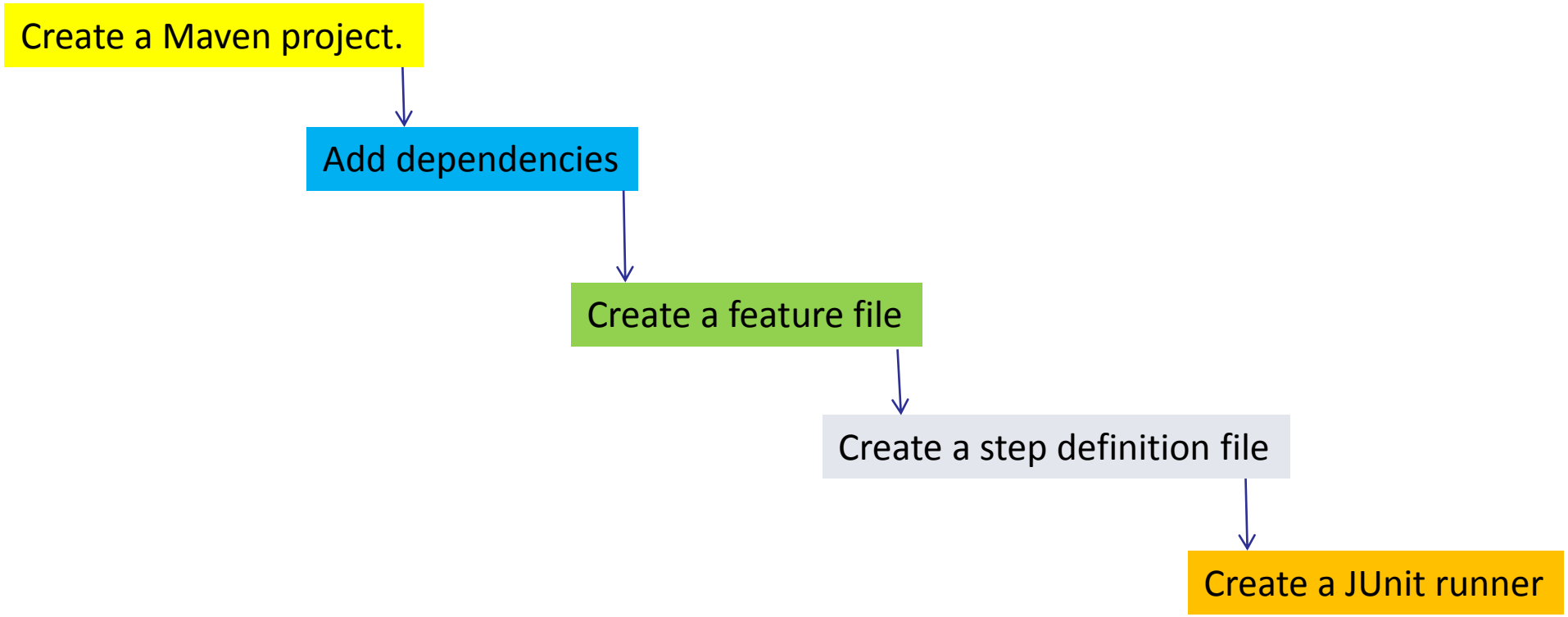
Report Output

```
1 package sample;
2 import org.junit.runner.RunWith;
5
6 @RunWith(Cucumber.class)
7 @CucumberOptions(
8         features = "src/test/java/sample/ReportSample.feature",
9         plugin = { "pretty", "junit:target/cucumber-reports/Cucumber.xml"},
10        monochrome = true
11        )
12
13 public class RunnerClass {
14
15 }
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<testsuite failures="0" name="cucumber.runtime.formatter.JUnitFormatter" skipped="0" tests="1" time="18.18976">
<testcase classname="Guru Testing" name="" time="18.18976">
<system-out><![CDATA[Given I navigate to the guru99.....passed
And I login to the Guru99.....passed
Then I close the browser.....passed
]]></system-out>
</testcase>
</testsuite>
```

```
package sample;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "src/test/java/sample/ReportSample.feature",
        plugin = { "pretty", "json:target/cucumber-reports/Cucumber.json",
                   "junit:target/cucumber-reports/Cucumber.xml",
                   "html:target/cucumber-reports"},
        monochrome = true
        )

public class RunnerClass {

}
```

Creating project cucumber with maven



Creating project cucumber with maven

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.11</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.14.0</version>
    </dependency>
    <dependency>
        <groupId>info.cukes</groupId>
        <artifactId>cucumber-java</artifactId>
        <version>1.2.5</version>
    </dependency>
    <dependency>
        <groupId>info.cukes</groupId>
        <artifactId>cucumber-junit</artifactId>
        <version>1.2.5</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Q&A

Reference

- http://toolsqa.com
- http://b4usolution.com/tutorial/
- http://www.automationtestinghub.com
- https://stackoverflow.com
- https://www.tutorialspoint.com
- https://www.slideshare.net/