// Testes

Cucumber


cucumber

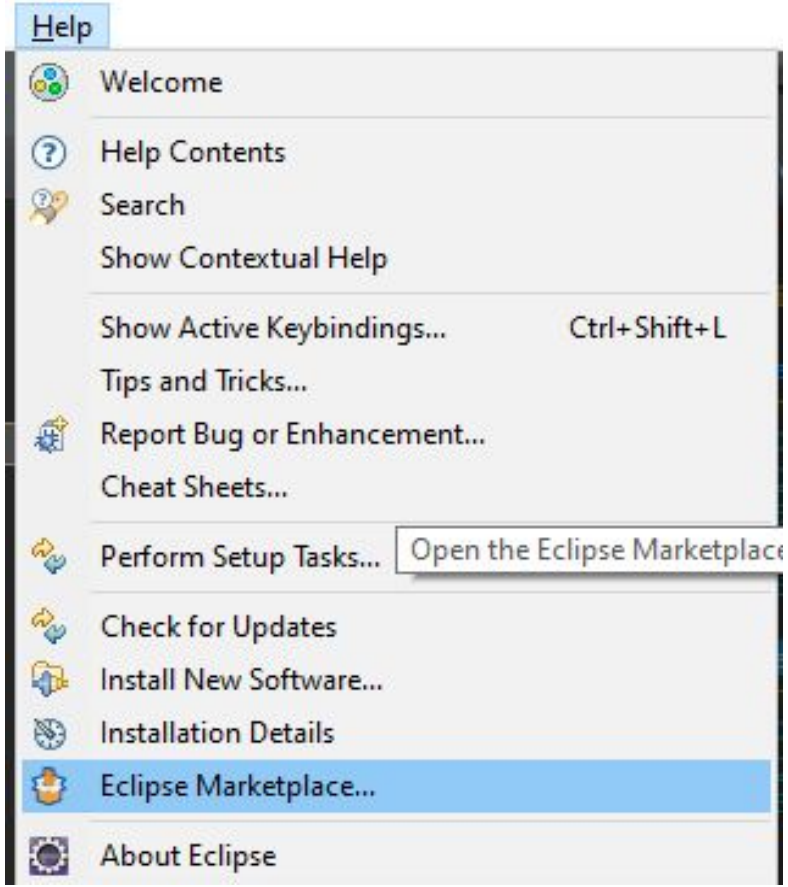
O que é Cucumber?

O Cucumber foi originalmente escrito na linguagem de programação Ruby e foi usado exclusivamente para testes em Ruby como um complemento à estrutura RSpec BDD para apoiar o desenvolvimento de testes de aceitação automatizado utilizando o formato BDD. Desde então o Cucumber cresceu e foi traduzido em várias linguagens(Java, java script, php, .net e etc), permitindo assim que vários desenvolvedores desfrutem de suas vantagens.

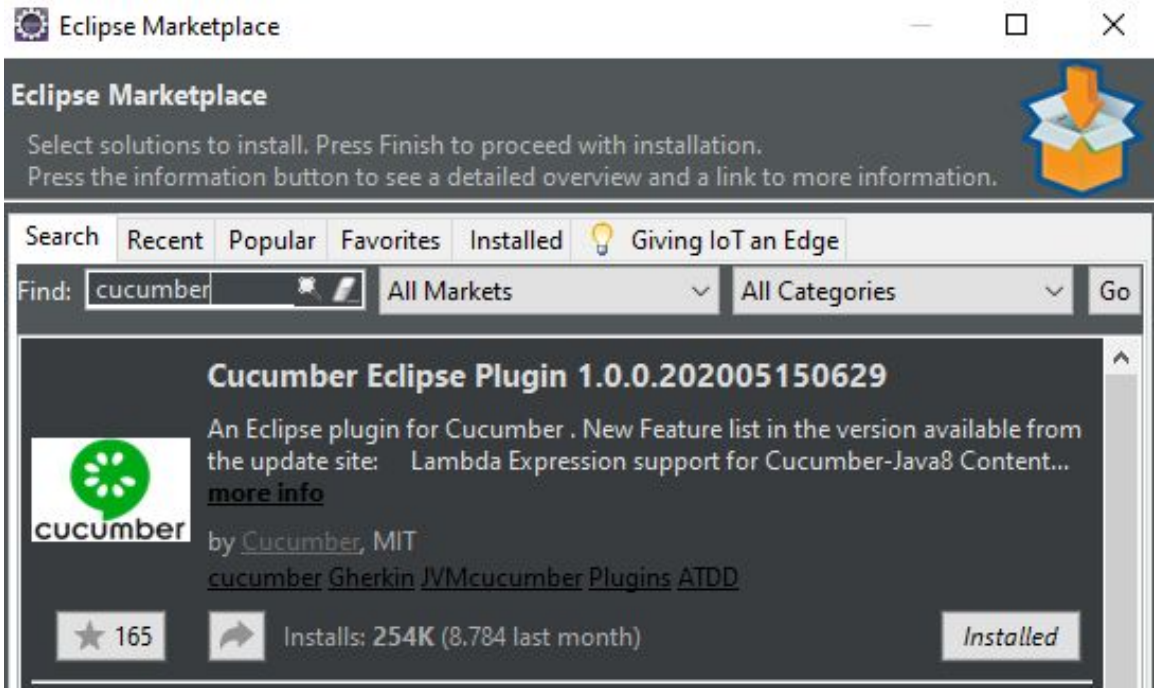
Vamos para a prática!

Utilizaremos o nosso projeto criado anteriormente utilizando restAssured e faremos com que os nossos testes escritos sejam executados em conjunto com o cucumber.

Primeiramente em Help > Eclipse Marketplace



Vamos instalar o “Cucumber Eclipse Plugin” que facilita a leitura, escrita e manipulação de features no eclipse.



No arquivo **pom.xml** adicionar as seguintes dependências:

```
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>6.9.1</version>
    <scope>test</scope>
</dependency>
```

Fonte: https://mvnrepository.com/

No arquivo **pom.xml** adicionar as seguintes dependências:

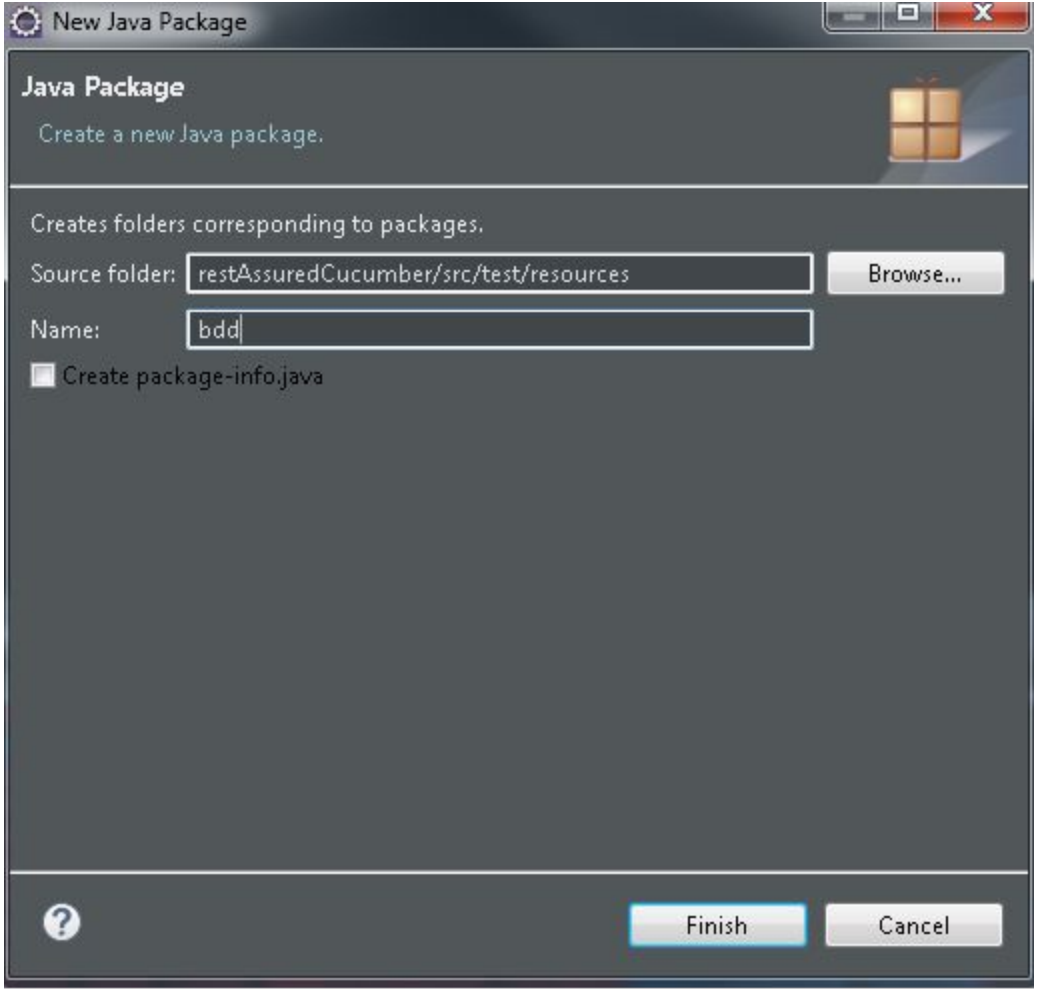
```
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>6.9.1</version>
    <scope>test</scope>
</dependency>
```

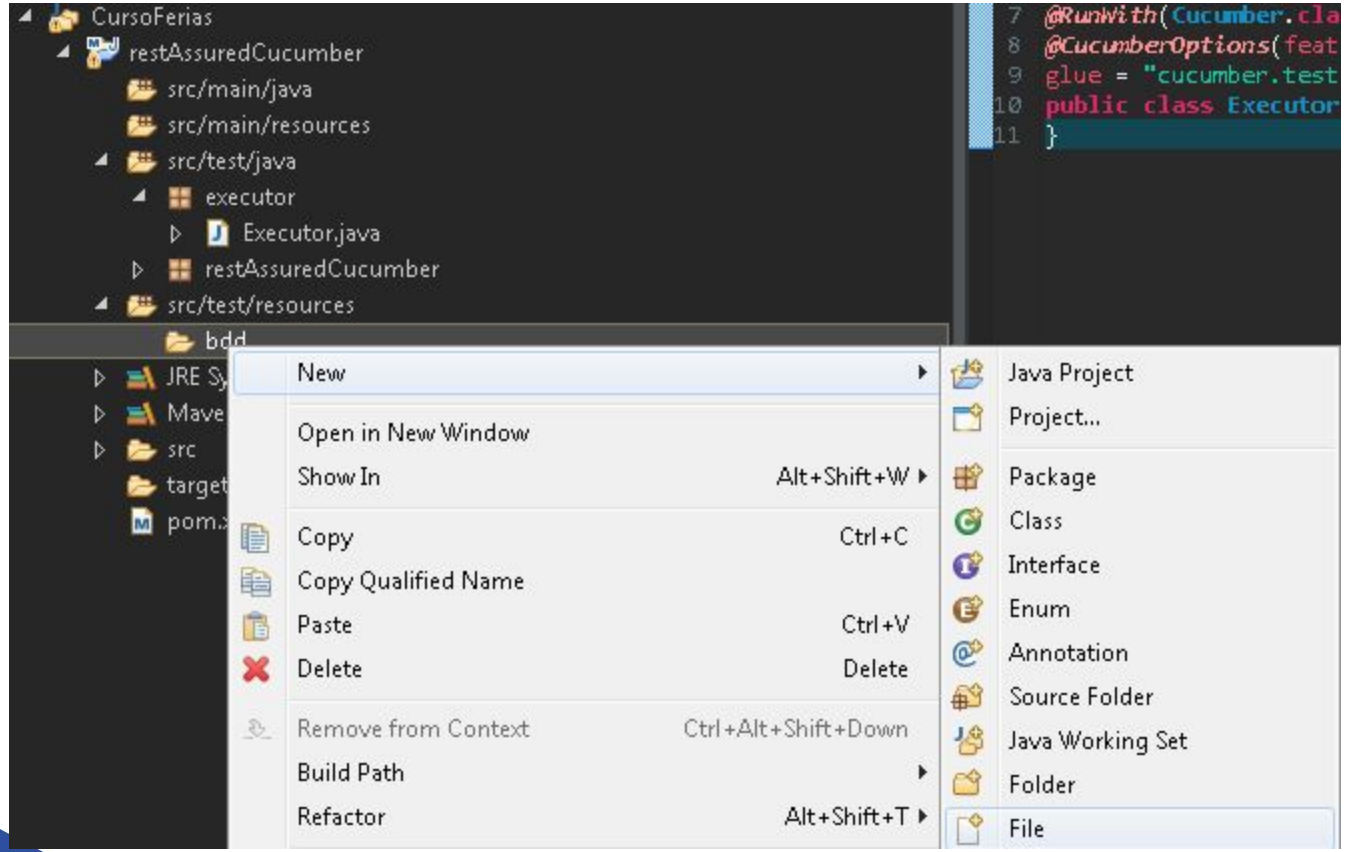
Fonte: https://mvnrepository.com/

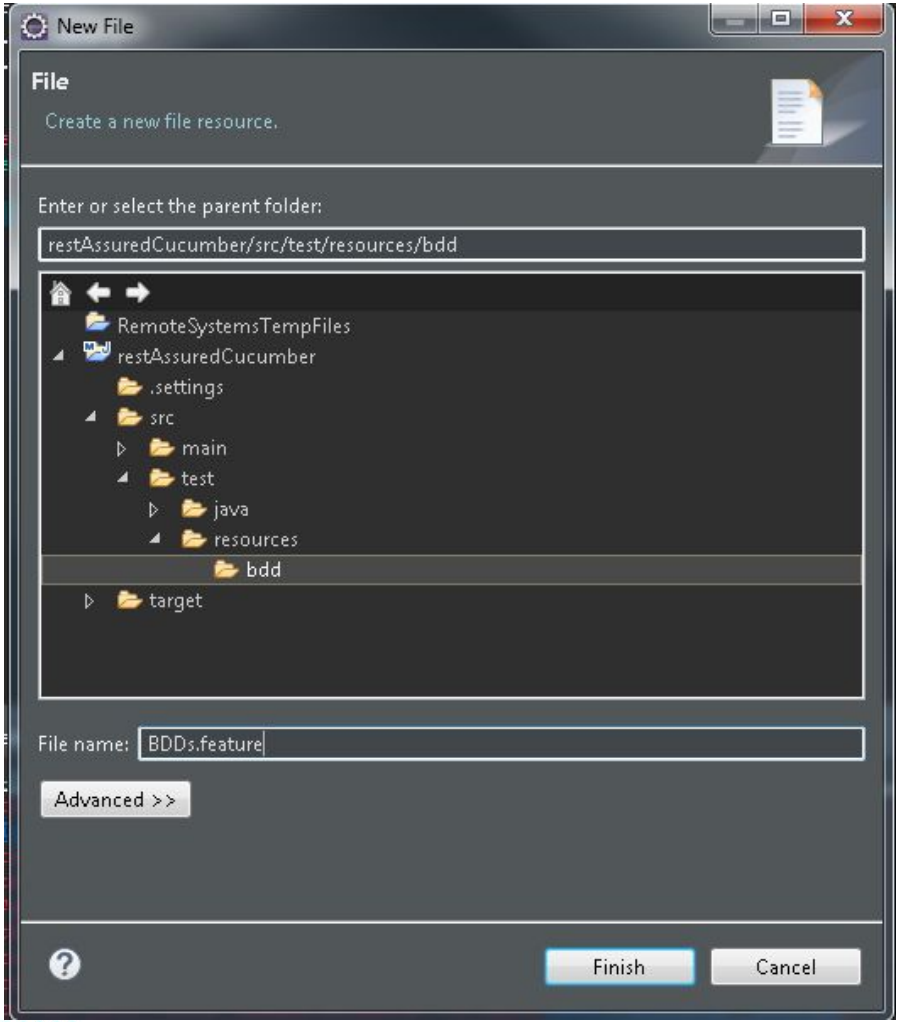
No arquivo **pom.xml** adicionar as seguintes dependências:

```
<dependency>
   <groupId>io.cucumber</groupId>
   <artifactId>gherkin</artifactId>
   <version>16.0.0</version>
</dependency>
```

Fonte: https://mvnrepository.com/


New Java Package
Java Package
Create a new Java package.
Creates folders corresponding to packages.
Source folder:
restAssuredCucumber/src/test/resources
Browse...
Name:
bdd
Create package-info.java
Finish
Cancel


CursoFerias
restAssuredCucumber
src/main/java
src/main/resources
src/test/java
executor
Executor.java
restAssuredCucumber
src/test/resources
bdd
JRE Sy
Mave
src
target
pom.x
@RunWith(Cucumber.cla
@CucumberOptions(feat
glue = "cucumber.test
public class Executor
}
New
Open in New Window
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Refactor Alt+Shift+T
Java Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File


New File
File
Create a new file resource.
Enter or select the parent folder:
restAssuredCucumber/src/test/resources/bdd
RemoteSystemsTempFiles
restAssuredCucumber
.settings
src
main
test
java
resources
bdd
target
File name:
BDDs.feature
Advanced >>
Finish
Cancel

Palavras-chave

#: Utilizado para escrever comentário;

@:Simbologia para marcar uma tag;

Feature(Funcionalidade): É uma palavra-chave onde é feita uma descrição de alto nível.

Cenario: É um cenário a ser executado

Palavras-chave

Dado: É utilizado para descrever um contexto inicial do cenário. Quando o Cucumber executa a palavra-chave “Dado”, espera-se que o cenário esteja em um estado definido, por meio e exemplo de uma criação ou configuração de objetos;

Quando: Utilizado para descrever um evento ou ação. Por exemplo, uma pessoa interagindo com o sistema ou um evento desencadeado por outro sistema;

Entao: É utilizado para descrever um resultado esperado.

Palavras-chave

E: É utilizado em conjunto com o “Dado”, “Quando” e “Entao”, simbolizando uma adição dentro de um mesmo cenário;

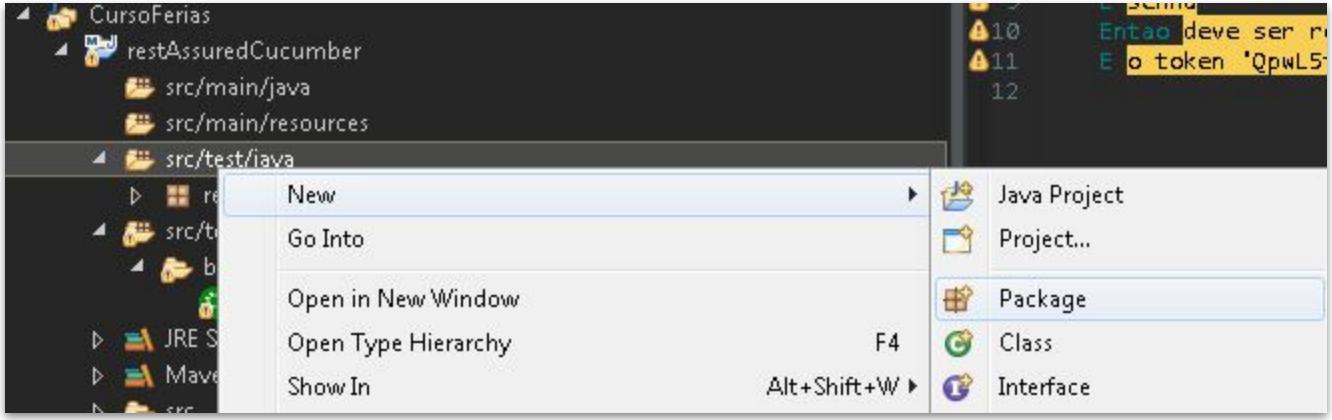
Temos também:
Rule (a partir de Gherkin 6), But, Background Scenario Outline, Examples

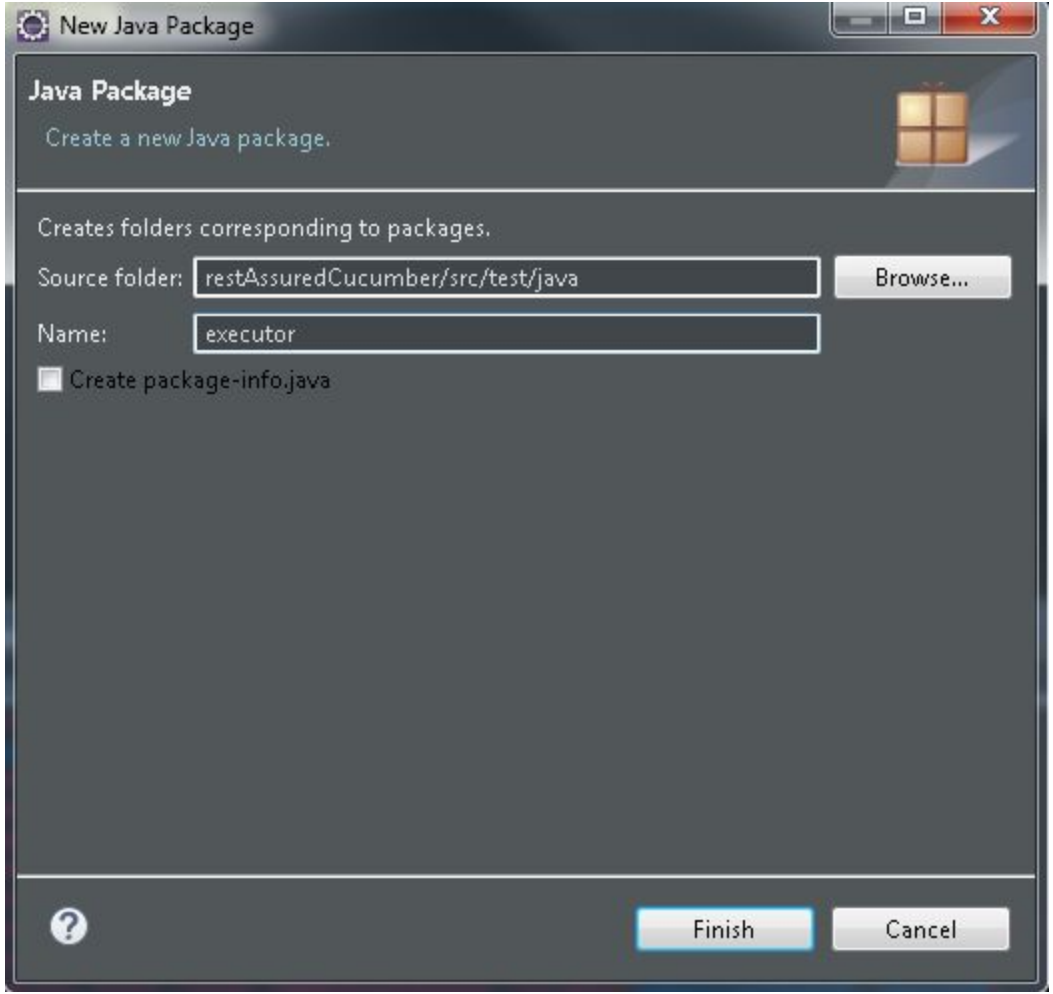
Vamos escrever agora o BDD utilizando as palavras-chave utilizadas pelo Gherkin.

```
# language: pt

Funcionalidade: Realizar requisições apis

  @TestePost
  Cenario: Retorna status 200 e token QpwL5tke4Pnpja7X4
    Dado que eu possua uma uri
    Quando informo email
    E senha
    Entao deve ser retornado o status '200'
    E o token "QpwL5tke4Pnpja7X4"
```

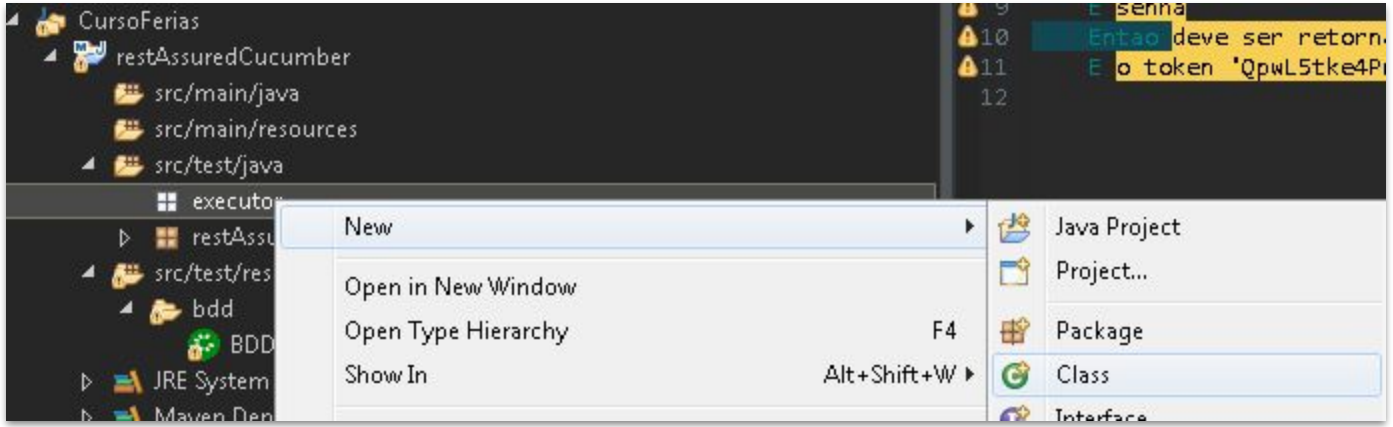
```
# language: pt

Funcionalidade: Realizar requisicoes apis

  @TestePost
  Cenario: Retorna status 200 e token QpwL5tke4Pnpja7X4
    Dado que eu possua uma uri
    Quando informo email
    E senha
    Entao deve ser retornado o status '200'
    E o token "QpwL5tke4Pnpja7X4"
```

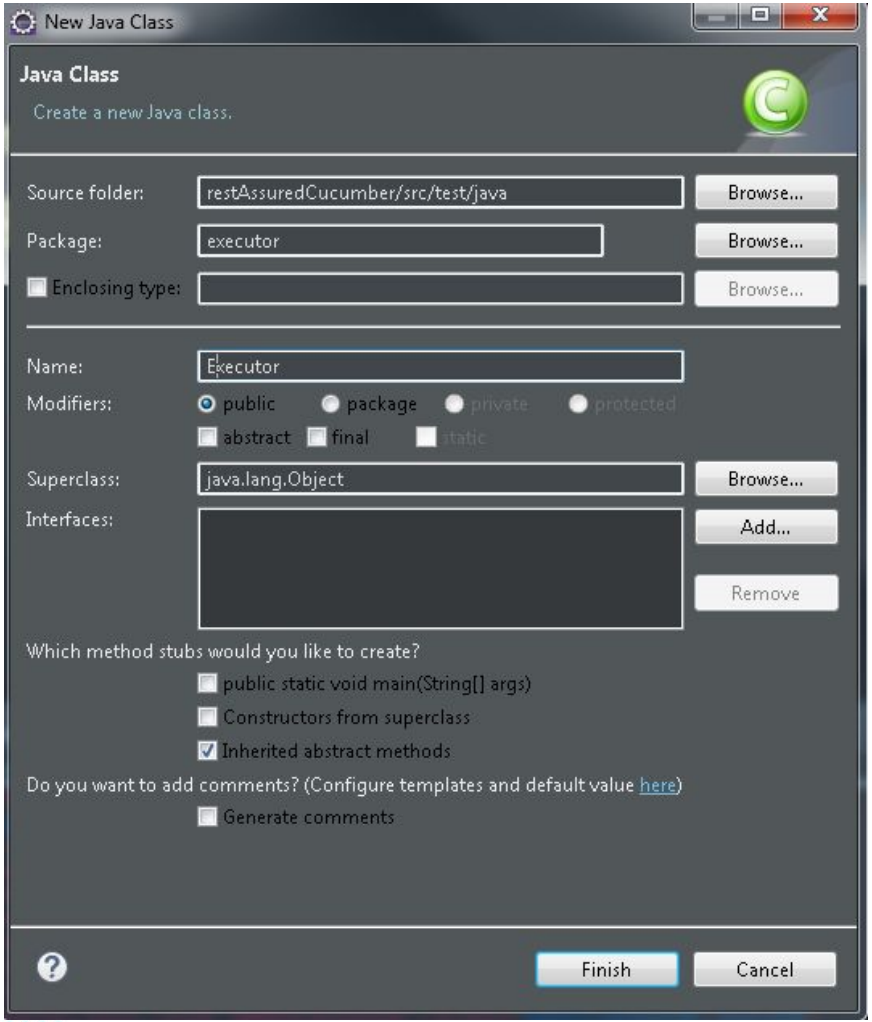
Vamos criar uma package que será incluída a classe que executa nosso teste.




New Java Package
Java Package
Create a new Java package.
Creates folders corresponding to packages.
Source folder:
restAssuredCucumber/src/test/java
Browse...
Name:
executor
Create package-info.java
Finish
Cancel

Agora a classe que executa o cucumber..




New Java Class
Java Class
Create a new Java class.
Source folder:
restAssuredCucumber/src/test/java
Browse...
Package:
executor
Browse...
Enclosing type:
Browse...
Name:
Executor
Modifiers:
public
package
private
protected
abstract
final
static
Superclass:
java.lang.Object
Browse...
Interfaces:
Add...
Remove
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel

```java
package executor;

import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(features = "classpath:bdd",
glue = "restAssuredCucumber", monochrome = true, dryRun = false)
public class Executor {
}
```

Observe que na classe Executor.java existe uma anotação @RunWith(Cucumber.class), isso diz ao JUnit que o Cucumber irá assumir o controle da execução dos testes nesta classe;

Outra anotação é @CucumberOptions, onde podemos definir parâmetros customizáveis utilizados pelo Cucumber na execução dos testes;

Features: É utilizada para ajudar o Cucumber na localização das features;

Glue: É utilizada para ajudar o Cucumber na localização das classes que contém os passos para os testes de aceitação;

Monochrome: É utilizado para definir a formatação do resultado dos testes na saída da console. Se marcado como "true", o resultado dos testes sai na forma legível, mas se "false", a legibilidade é um pouco mais complexa;

DryRun: esta opção pode ser definida como “true” ou “false“. Se estiver marcado como “true”, significa que o Cucumber só verifica se cada etapa definida no arquivo .feature tem código correspondente. Considerando ainda “true”, se na execução de um arquivo .feature o Cucumber não achar nenhum código (Java) correspondente a esse arquivo, então o Cucumber gera o código correspondente. Se marcado como “false”, o Cucumber não faz essa verificação.

Tags: É utilizada para definir as tags neste parâmetro, uma vez uma mesma tag definida neste atributo e no (s) arquivo (s) .feature. Quando o Cucumber executar, esta classe só executará em conjunto apenas os arquivos .feature marcados com a mesma tag;
EX: tags = {"@TestePost, @TesteGet"}

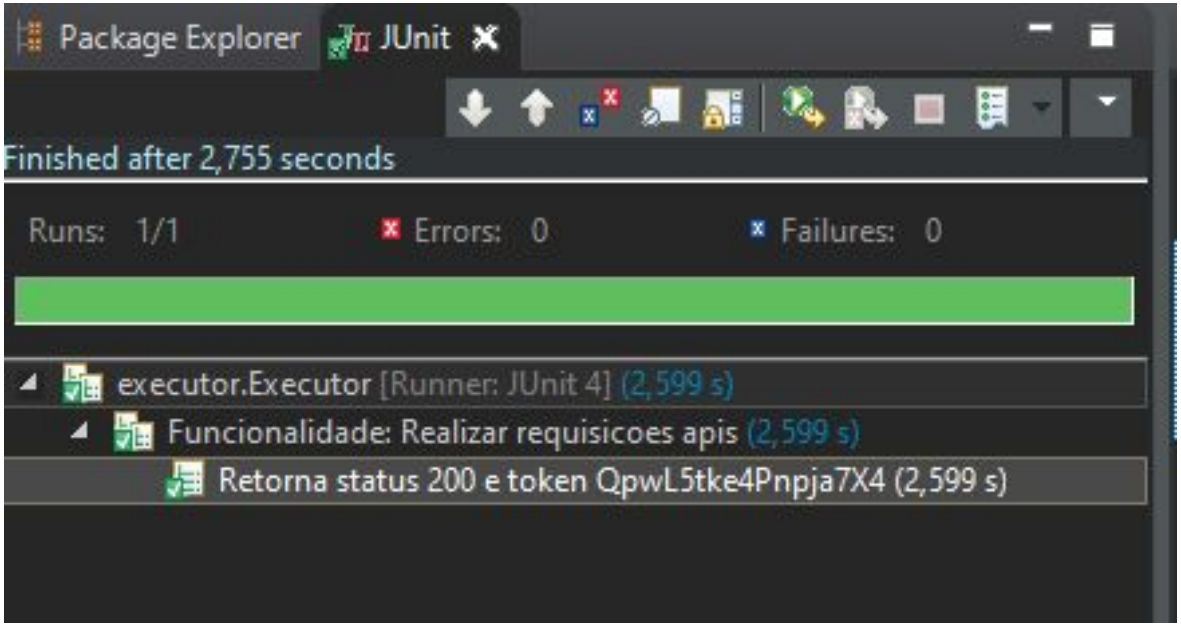
Vamos refatorar o código da classe TesteApiPost separando por métodos onde cada linha(Dado, Quando, Entao e E) definida no nosso bdd(BDDs.feature) será um método na nossa classe TesteApiPost

```
@Dado("^que eu possua uma uri$")
	public void que_eu_possua_uma_uri() throws Throwable {
		baseURI = "https://reqres.in/api/login";
	}
```

```
@Quando("^informo email$")
	public void informo_email() throws Throwable {
		myJson = "{\"email\":\"eve.holt@reqres.in\",\"password\":
\"cityslicka\"}";
	}

	@E("^senha$")
	public void senha() throws Throwable {

	}
```

```
@Entao("^deve ser retornado o status '200'$")
	public void deve_ser_retornado_o_status_200() throws
Throwable {

	}

	@E("^o token \"(.*?)\"$")
	public void o_token_(String token) throws Throwable {
		given().contentType(ContentType.JSON).body(myJson).
when().post(baseURI).then().statusCode(200).body("token",
				containsString(token));
	}
```

Feito Isso nosso código poderá ser executado e poderemos verificar os resultados dos testes de aceitação utilizando Rest-Assured em conjunto com o Cucumber.


Package Explorer
JUnit
Finished after 2,755 seconds
Runs: 1/1
Errors: 0
Failures: 0
executor.Executor [Runner: JUnit 4] (2,599 s)
Funcionalidade: Realizar requisicoes apis (2,599 s)
Retorna status 200 e token QpwL5tke4Pnpja7X4 (2,599 s)

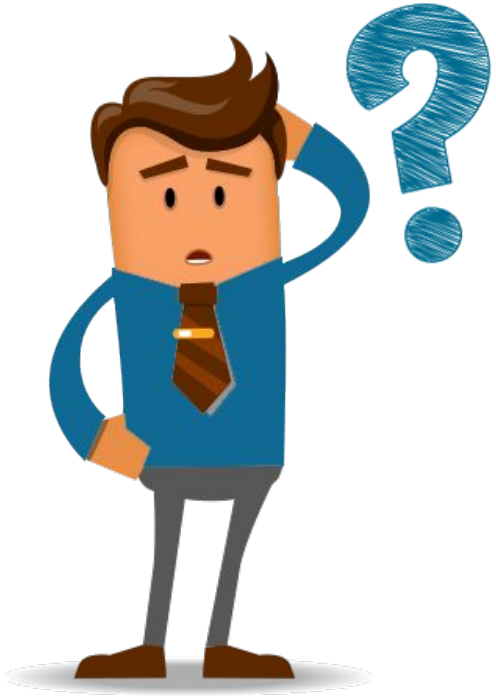
EXERCÍCIO:

Refatore a classe TesteApiGet.java separando por métodos onde cada linha(Dado, Quando, Entao e E) definidos no arquivo BDDs.feature:

Cenario: Verifica se possui first_name Michael e Rachel

Dado que eu possua uma uri get
Quando informo a pagina 2
Entao deve ser retornado o first_name "Michael" e "Rachel"

Dúvidas?



matera

www.matera.com