Бесплатная электронная книга

УЧУСЬ

cucumber

Free unaffiliated eBook created from
Stack Overflow contributors.

#cucumber

1
1: 2
2
Examples 3
3
Ruby 4
Ruby 4
2: pom.xml Maven_ cucumber. 6
6
Examples 6
pom.xml 6
3: 8
8
Examples 8
Ruby 8
4: 10
10
10
Examples 10
10
12
12
13
14
16
5: Intellij 17
17
17
Examples 17
17
IntelliJ Cucumber Java Plugin (Mac) 18

6: 26
..... 26
..... 26
Examples 26
..... 26
..... 26
..... 27
..... 28

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: cucumber

It is an unofficial and free cucumber ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official cucumber.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с огурцом

замечания

О огурце

Огурец - это инструмент, который выполняет исполняемые спецификации программного обеспечения. Спецификации, называемые «функциями», написаны на структурированном естественном языке. Cucumber выполняет функцию, сопоставляя каждый ее шаг с «определением шага», написанным на языке программирования, поддерживаемом этой реализацией Cucumber. Огурцы реализованы на многих языках программирования, включая Ruby (оригинал), Java и Javascript . Он также переведен на многие человеческие языки.

Огурец был написан для поддержки гибкой методологии под названием «Поведенческое развитие» (BDD). В BDD начинается разработка за пределами, написав приемочные тесты, которые описывают функциональность программного обеспечения с точки зрения пользователя (а не с точки зрения программиста, например, в модульных тестах). Этим приемочным испытаниям служат функции огурца.

В общем, функции Cucumber - это документация, читаемая человеком, которая также является исполняемым набором тестов, что означает, что документация и тесты всегда согласуются. Огурцы полезны для общения с заинтересованными лицами, не являющимися программистами, о документации и тестах. Это также позволяет программистам писать тесты на концептуальном уровне без проблем, связанных с программированием.

Огурец чаще всего используется для определения и тестирования веб-приложений с использованием драйвера браузера, такого как Selenium или PhantomJS. Однако его можно использовать с любым программным обеспечением, которое может быть выполнено, и состояние или результаты которого могут быть определены на языке программирования, поддерживаемом реализацией Cucumber.

Другая документация

Официальная документация находится по адресу https://cucumber.io/docs . Документация, созданная из функций Cucumber, которые описывают реализации Cucumber, находится в

- JavaScript: https://relishapp.com/cucumber/cucumber-js/docs
- Рубин: https://relishapp.com/cucumber/cucumber/docs

https://relishapp.com/explore включает в себя некоторые другие инструменты и примеры, связанные с огурцем, хотя, к сожалению, это не Cucumber-JVM.

Эта тема

В этой теме следует привести лишь несколько примеров, которые приводят читателя к концепциям огурца. В других разделах приводятся полные примеры установки, использования командной строки и IDE, функций, определений шагов и т. Д.

Examples

Функция огурца

Cucumber использует синтаксис Gherkin для описания поведения вашего программного обеспечения на структурированном естественном языке.

Поскольку такой огурец **не** является тестовой основой (распространенным недоразумением), а *основой системы документации* , не очень отличающейся от других, таких как сценарий использования использования. Общее недоразумение связано с тем, что документация огурца *может быть автоматизирована, чтобы обеспечить ее отражение реального поведения системы* .

Комплект документации Cucumber состоит из `Features` , каждый из которых описывает функцию вашего программного обеспечения, написанную в Gherkin и размещенную в собственном файле. Организуя эти файлы в структуру каталогов, вы можете *группировать* и *организовывать* функции:

- банковское дело/
  - withdrawal.feature
  - atm.feature
  - личностно-loan.feature
- торговля /
  - portfolio.feature
  - intraday.feature
- ипотечный кредит/
  - evaluation.feature
  - accounting.feature

Каждая `Feature` представляет собой текстовый файл, состоящий из необязательного, неструктурированного, чисто информационного вводного раздела и одного или нескольких `Scenarios` , каждый из которых представляет собой условие использования или прецедент.

Пример:

```
Feature: Documentation
As a StackOverflow user or visitor
I want to access the documentation section

    Scenario: search documentation on Stack Overflow
        Given I am on StackOverflow
        And I go to the Documentation section
        When I search for "cucumber"
```

```
        And I follow the link to "cucumber"
        Then I should see documentation for "cucumber"
```

Каждая строка, начинающаяся с *Given* , *When* , *And* , *But* или *Then* , называется `Step` . Любой шаг может начинаться с любого из этих слов независимо от порядка, но традиционно использовать их наиболее естественным образом.

Функции также могут быть организованы с помощью `Tags` , аннотаций, которые редактор может нанести на `Feature` или `Scenario` для дальнейшей категоризации.

Исполняемость функции достигается с помощью кода *клея,* который может быть написан на разных языках (Java, Ruby, Scala, C / C ++): каждый `Step` сопоставляется с кодом клея, чтобы идентифицировать определения `Step Definitions` (обычно сокращенно до *StepDef*) через регулярные выражения.

На каждом `Step` может быть только одно связанное `Step Definition` .

Когда выполняется `Feature` , выполняется каждый `Scenario` составления, что означает, что каждый StepDef, соответствующий `Step` s, выполняется в каждом `Scenario` .

Чистая установка Ruby

Чтобы установить Cucumber для использования с Ruby, просто используйте команду

```
gem install cucumber
```

Кроме того, если вы используете bundler, вы можете добавить следующую строку в свой Gemfile

```
gem 'cucumber'
```

А затем запустите комплектщик

```
bundle install
```

[Я думаю, это относится к своей теме «Установка». Я создал эту тему и скопировал этот пример. Когда эта тема будет одобрена, я переведу туда и удалю копию.]

Определение шага огурца в Ruby

В функциях / step_definitions / documentation.rb:

```
When /^I go to the "([^"]+)" documentation$/ do |section|
  path_part =
    case section
      when "Documentation"
        "documentation"
```

```
      else
        raise "Unknown documentation section: #{section}"
    end
  visit "/documentation/#{path_part}/topics"
end

Then /^I should see the "([^"]+) documentation"$/ do |section|
  expect(page).to have_css('h2.doctag_title a', text: section)
end
```

На этих этапах выполняется веб-приложение. Они примерно так же просты, как и могут быть, пока они практичны.

Каждый шаг начинается с ключевого слова Gherkin, которое в файле определения шага является методом, который регистрирует шаг с огурцом. Метод определения шага принимает регулярное выражение, которое соответствует строке в сценарии, и блок, который выполняется, когда сценарий попадает в соответствующую строку. Группы захвата в регулярном выражении передаются блоку в качестве параметров блока.

В шаге « `When` есть простой пример в строке «Переход от удобочитаемой ссылки на страницу (« Документация ») к URL-адресу. Реальные комплекты Cucumber обычно используют эту логику в отдельном методе. Метод `visit` предоставляется Capybara. Capybara не требуется использовать огурец, хотя он очень часто используется с ним. `visit` браузер, контролируемый Capybara, чтобы посетить данный URL.

Шаг `Then` показывает, как можно проверить содержимое страницы. `expect` / `to` предоставляется RSpec (опять же, не требуется огурцом, но очень часто используется с ним). `have_css` предоставляется `have_css` . Ожидается, что данный CSS-селектор соответствует элементу на странице, который содержит данный текст. Обратите внимание, что это ожидание не сработает, если запрос браузера не удался.

Дополнительные примеры см. В разделе «Определение шага» .

Прочитайте Начало работы с огурцом онлайн: https://riptutorial.com/ru/cucumber/topic/4875/начало-работы-с-огурцом

глава 2: pom.xml для проекта Maven_ cucumber.

Вступление

Модель объекта проекта ниже - шаблон pom.xml. Если вы хотите создать проект maven с огурцом, вы можете использовать приведенный ниже пример в качестве шаблона

Examples

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

4.0.0

```
<groupId>Project name</groupId>
<artifactId>MulitClients</artifactId>
<version>1.0-SNAPSHOT</version>


<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.11</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>info.cukes</groupId>
        <artifactId>cucumber-core</artifactId>
        <version>1.2.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>info.cukes</groupId>
        <artifactId>cucumber-testng</artifactId>
        <version>1.2.0</version>
    </dependency>
    <dependency>
        <groupId>info.cukes</groupId>
        <artifactId>cucumber-junit</artifactId>
        <version>1.2.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>info.cukes</groupId>
        <artifactId>cucumber-java</artifactId>
        <version>1.2.0</version>
        <scope>test</scope>
    </dependency>
```

```
    <dependency>
        <groupId>info.cukes</groupId>
        <artifactId>gherkin</artifactId>
        <version>2.12.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>2.53.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-firefox-driver</artifactId>
        <version>2.53.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-htmlunit-driver</artifactId>
        <version>2.53.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.yaml</groupId>
        <artifactId>snakeyaml</artifactId>
        <version>1.13</version>
    </dependency>
    <dependency>
        <groupId>com.esotericsoftware.yamlbeans</groupId>
        <artifactId>yamlbeans</artifactId>
        <version>1.06</version>
    </dependency>


</dependencies>
```

Прочитайте pom.xml для проекта Maven_ cucumber. онлайн:
https://riptutorial.com/ru/cucumber/topic/8331/pom-xml-для-проекта-maven--cucumber-

глава 3: Определения шагов

замечания

Определения шагов находятся на языке программирования, поддерживаемом данной реализацией Cucumber. В этом разделе приведены примеры определений шагов в каждом поддерживаемом языке программирования и примеры использования вызовов API Cucumber в определениях шагов.

Examples

Некоторые простые определения шага Ruby

В функциях / step_definitions / documentation.rb:

```
When /^I go to the "([^"]+)" documentation$/ do |section|
  path_part =
    case section
      when "Documentation"
        "documentation"
      else
        raise "Unknown documentation section: #{section}"
    end
  visit "/documentation/#{path_part}/topics"
end

Then /^I should see the "([^"]+) documentation"$/ do |section|
  expect(page).to have_css('h2.doctag_title a', text: section)
end
```

На этих этапах выполняется веб-приложение. Они примерно так же просты, как и могут быть, пока они практичны.

Каждый шаг начинается с ключевого слова Gherkin, которое в файле определения шага является методом, который регистрирует шаг с огурцом. Метод определения шага принимает регулярное выражение, которое соответствует строке в сценарии, и блок, который выполняется, когда сценарий попадает в соответствующую строку. Группы захвата в регулярном выражении передаются блоку в качестве параметров блока.

В шаге « `When` есть простой пример в строке «Переход от удобочитаемой ссылки на страницу (« Документация ») к URL-адресу. Реальные комплекты Cucumber обычно используют эту логику в отдельном методе. Метод `visit` предоставляется Capybara. Capybara не требуется использовать огурец, хотя он очень часто используется с ним. `visit` браузер, контролируемый Capybara, чтобы посетить данный URL.

Шаг `Then` показывает, как можно проверить содержимое страницы. `expect` / `to`

предоставляется RSpec (опять же, не требуется огурцом, но очень часто используется с ним). `have_css` предоставляется `have_css` . Ожидается, что данный CSS-селектор соответствует элементу на странице, который содержит данный текст. Обратите внимание, что это ожидание не сработает, если запрос браузера не удался.

Прочитайте Определения шагов онлайн: https://riptutorial.com/ru/cucumber/topic/5681/определения-шагов

глава 4: Синтаксис Сердца

Вступление

Gherkin - деловой читаемый язык для автоматизации тестирования и тестовой документации. Это понимается Cucumber и вместе существует как инструмент, управляемый поведением.

Синтаксис

- Функция: это ключевое слово означает, что следующим является базовое описание или имя проверяемой или документированной функции.
- Предыстория: это ключевое слово означает шаги, которые будут выполняться перед каждым сценарием в этой функции.
- Сценарий: это ключевое слово представляет собой имя или базовое описание конкретного сценария, проверяющего эту функцию.
- Схема сценария: это ключевое слово означает, что сценарий будет выполняться N раз для каждого аргумента, указанного в примерах, явно переданных по имени столбца, заключенному в угловые скобки.
- Примеры: это ключевое слово отмечает список статических аргументов, которые будут переданы в схему сценария.
- Данное: данное ключевое слово представляет собой заданный шаг или предварительное условие, которое предполагается до продолжения. В параграфе «Упорядочить», «Акт», «Утверждение» задано «Упорядочить».
- Когда: это ключевое слово представляет собой шаг when или поведение, которое должно быть указано против. В параграфе «Упорядочить», «Действие», «Утверждение» данный представляет «Закон».
- Затем: это ключевое слово представляет собой шаг затем, или, другими словами, шаг, на котором проверяется результат поведения. В параграфе «Упорядочить», «Действие», «Утверждение» задано «Assert».
- И: Это ключевое слово используется в сочетании с любым из указанных выше ключевых слов. Если у вас есть два заданных выражения, вместо явного вызова «Предоставлено дважды» вы можете сказать «Дано A и B».

Examples

Основы

В этом примере рассмотрим базовую структуру файла функций Cucumber в Gherkin. В базовых синтаксисах функциональные файлы используют несколько ключевых слов.

Давайте рассмотрим основные ключевые слова:

- **Функция:** это ключевое слово означает, что следующим является базовое описание или имя проверяемой или документированной функции.
- **Сценарий:** это ключевое слово представляет собой имя или базовое описание конкретного сценария, проверяющего эту функцию.
- **Учитывая** это ключевое слово, представляет собой заданный шаг или предварительное условие, которое предполагается до продолжения. В параграфе «Упорядочить», «Акт», «Утверждение» задано «Упорядочить».
- **Когда** это ключевое слово представляет собой шаг when или поведение, которое должно быть указано против. В параграфе «Упорядочить», «Действие», «Утверждение» данный представляет «Закон».
- **Затем** это ключевое слово представляет собой шаг затем, или, другими словами, шаг, на котором проверяется результат поведения. В параграфе «Упорядочить», «Действие», «Утверждение» задано «Assert».
- **И** это ключевое слово используется в сочетании с любым из указанных выше ключевых слов. Если у вас есть два заданных выражения, вместо явного вызова «Предоставлено дважды» вы можете сказать «Дано A и B».
- **Но** это ключевое слово используется совместно с **данными** , **When** и **Then** для обозначения того, что что-то не должно произойти. Тогда A Но не B.

Все ключевые слова должны быть на новой строке и должны быть первым словом на новой строке, чтобы распознать парсер Gherkin. Ключевые слова Feature и Scenario должны иметь двоеточие сразу после, как показано в примере ниже. Дано, Когда, Затем, и И не требует двоеточия.

В дополнение к ключевым словам вы можете писать описания и комментарии. Описания происходят после ключевого слова, но в той же строке, где в комментариях появляются строки под ключевыми словами. При написании комментариев к функциям принято предоставлять явные правила, описывающие границы и условия, которые приводят к различным результатам поведения.

```
Feature: Product Login
    As a user, I would like to be able to use my credentials to successfully
    login.

    Rules:
    - The user must have a valid username
    - The user must have a valid password
    - The user must have an active subscription
    - User is locked out after 3 invalid attempts
    - User will get a generic error message following
      login attempt with invalid credentials

    Scenario: The user successfully logs in with valid credentials
        This scenario tests that a user is able to successfully login
        provided they enter a valid username, valid password, and
        currently have an active subscription on their account.
```

```
        Given the user is on the login page
        When the user signs in with valid credentials
        Then the user should be logged in
```

Параметрированные этапы

При написании Gherkin могут быть моменты, когда вы хотите параметризовать свои шаги для повторного использования инженером, который реализует планы тестирования. Шаги получают параметры через группы захвата регулярных выражений. (**Техническое примечание.** Если у вас нет соответствующих параметров для каждой группы захвата в вашем регулярном выражении, вы можете ожидать, что будет выбрано « CucumberException: несоответствие Arity»). В приведенном ниже примере мы решили обернуть аргументы в двойные кавычки, а также как принимать целые числа в качестве аргументов.

```
Feature: Product Login
   As a user, I would like to be able to use my credentials to successfully
   login.

   Rules:
   - The user must have a valid username
   - The user must have a valid password
   - The user must have an active subscription
   - User is locked out after 3 invalid attempts
   - User will get a generic error message following
     login attempt with invalid credentials

   Scenario: The user successfully logs in with valid credentials
       This scenario tests that a user is able to successfully login
       provided they enter a valid username, valid password, and
       currently have an active subscription on their account.

       Given the user is on the login page
       When the user signs in with "valid" credentials
       Then the user should be logged in

   Scenario: The user attempts to log in with invalid credentials
       This scenario tests that a user is not able to log in when
       they enter invalid credentials

       Given the user is on the login page
       When the user signs in with "invalid" credentials
       Then the user should be logged in

   Scenario: The user is locked out after too many failed attempts
       This scenario validates that the user is locked out
       of their account after failing three consecutive
       attempts to log in

       Given the user is on the login page
       When the user fails to log in 3 times
       Then the user should be locked out of their account
```

Особенность

Как вы могли заметить в приведенном выше примере, мы переписываем один и тот же шаг несколько раз:

```
Given the user is on the login page
```

Это может быть исключительно утомительным, особенно если у нас есть несколько заданных шагов, которые используются повторно. Gherkin предлагает решение для этого, дав нам другое ключевое слово для работы с: **Background:.**

Ключевое слово background служит для запуска шагов, объявленных под ним перед каждым сценарием в Feature. Не забудьте добавить фоновый шаг, если вы не уверены, что это необходимо для каждого сценария. Как и другие ключевые слова, за фоном следует описание или имя и могут содержать комментарии, перечисленные ниже. Подобно функции и сценарию, фон должен выполняться двоеточием.

```
Feature: Product Login
    As a user, I would like to be able to use my credentials to successfully
    login.

    Rules:
    - The user must have a valid username
    - The user must have a valid password
    - The user must have an active subscription
    - User is locked out after 3 invalid attempts
    - User will get a generic error message following
      login attempt with invalid credentials

    Background: The user starts out on the login page
        Given the user is on the login page

    Scenario: The user successfully logs in with valid credentials
        This scenario tests that a user is able to successfully login
        provided they enter a valid username, valid password, and
        currently have an active subscription on their account.

        When the user signs in with "valid" credentials
        Then the user should be logged in

    Scenario: The user attempts to log in with invalid credentials
        This scenario tests that a user is not able to log in when
        they enter invalid credentials

        When the user signs in with "invalid" credentials
        Then the user should be logged in

    Scenario: The user is locked out after too many failed attempts
        This scenario validates that the user is locked out
        of their account after failing three consecutive
        attempts to log in

        When the user fails to log in 3 times
        Then the user should be locked out of their account
```

Схема сценария

В некоторых случаях вы можете повторно запускать один и тот же сценарий снова и снова, заменяя аргументы. В этом случае Gherkin предоставляет несколько новых ключевых слов для размещения этой ситуации, **сценарий сценария:** и **пример:.** Ключевое слово «Сценарий» указывает Cucumber, что сценарий будет запускаться несколько раз, заменяя аргументы из списка. Ключевое слово Examples вызывается до того, как список будет явно отмечен. Аргументы для сценария Контуры должны быть обернуты в угловые скобки. В приведенном ниже примере обратите внимание, что имена аргументов, заключенные в угловые скобки, соответствуют именам столбцов, указанным в примерах. Каждый столбец разделяется вертикальными полосами с именами столбцов в первой строке.

```
Feature: Product Login
    As a user, I would like to be able to use my credentials to successfully
    login.

    Rules:
    - The user must have a valid username
    - The user must have a valid password
    - The user must have an active subscription
    - User is locked out after 3 invalid attempts
    - User will get a generic error message following
      login attempt with invalid credentials

    Background: The user starts out on the login page
        Given the user is on the login page

    Scenario Outline: The user successfully logs in with their account
         This scenario outlines tests in which various users attempt
         to sign in successfully

         When the user enters their <username>
         And the user enters their <password>
         Then the user should be successfully logged on

         Examples:
         | username | password |
         | frank    | 1234     |
         | jack     | 4321     |
```

Теги

Для целей документации вы можете фильтровать планы тестирования или сценарии по категориям. Разработчики могут захотеть запускать тесты на основе тех же самых категорий. Gherkin позволяет классифицировать функции, а также отдельные сценарии через пользователя тегов. В приведенном ниже примере обратите внимание на то, что ключевое слово Feature является тегом «@Automation». Gherkin признает это как тег пользователем символа «@». В этом примере инженер хочет дать понять, что эти тесты используются для автоматизации, где не каждый тест автоматизирован, некоторые тесты должны проводиться с помощью ручного QA.

Также обратите внимание на то, что тег @Production был добавлен в блокировку

пользователя для тестирования сценариев. В этом примере это происходит потому, что этот сценарий активен только в рабочей среде приложения. Разработчики не хотят, чтобы их учетные записи в песочнице были заблокированы во время разработки. Эти теги позволяют им обеспечить, чтобы этот тест выполнялся только в рабочей среде.

Наконец, в разделе «Сценарий сценария» есть тег @Staging. Для целей этого примера это происходит потому, что используемые учетные записи являются промежуточными учетными записями и не будут работать в других средах. Подобно тегу @Production, это гарантирует, что эти тесты будут выполняться только в среде Staging.

Это всего лишь несколько примеров того, где, как и почему вы можете использовать теги. В конечном итоге эти теги будут иметь смысл для вас и разработчиков и могут быть любыми и использоваться для категоризации, но вы считаете нужным.

```
@Automation
Feature: Product Login
    As a user, I would like to be able to use my credentials to successfully
    login.

    Rules:
    - The user must have a valid username
    - The user must have a valid password
    - The user must have an active subscription
    - User is locked out after 3 invalid attempts
    - User will get a generic error message following
      login attempt with invalid credentials

    Background: The user starts out on the login page
        Given the user is on the login page

    Scenario: The user successfully logs in with valid credentials
        This scenario tests that a user is able to successfully login
        provided they enter a valid username, valid password, and
        currently have an active subscription on their account.

        When the user signs in with "valid" credentials
        Then the user should be logged in

    Scenario: The user attempts to log in with invalid credentials
        This scenario tests that a user is not able to log in when
        they enter invalid credentials

        When the user signs in with "invalid" credentials
        Then the user should be logged in

    @Production
    Scenario: The user is locked out after too many failed attempts
        This scenario validates that the user is locked out
        of their account after failing three consecutive
        attempts to log in

        When the fails to log in 3 times
        Then the user should be locked out of their account

    @Staging
    Scenario Outline: The user successfully logs in with their account
```

```
    This scenario outlines tests in which various users attempt
    to sign in successfully

    When the user enters their <username>
    And the user enters their <password>
    Then the user should be successfully logged on

    Examples:
    | username | password |
    | frank    | 1234     |
    | jack     | 4321     |
```

Советы по ожогу

- Каждый сценарий проверяет одно поведение
- Сценарии написаны декларативным образом
- Избегайте случайных деталей внутри сценария
- Опустить очевидное
- Избегайте конъюнктивных шагов
- Сократите ваши сценарии
- Не нужно много сценариев в одной и той же функции
- Использовать описательные имена сценариев
- Имейте только один Когда шаг
- Используйте «should» в шагах Then

Прочитайте Синтаксис Сердца онлайн: https://riptutorial.com/ru/cucumber/topic/9296/синтаксис-сердца

глава 5: Установить плагин огурца в Intellij

Вступление

Плагины Cucumber для IntelliJ IDEA предлагают удобные функции IDE для работы с файлами функций Gherkin в проекте IntelliJ с использованием инфраструктуры Cucumber. Плагины доступны для языков Java, Scala или Groovy.

замечания

Плагин Cucumber для Java IntelliJ предлагает функции IDE для удобного использования с Cucumber, включая

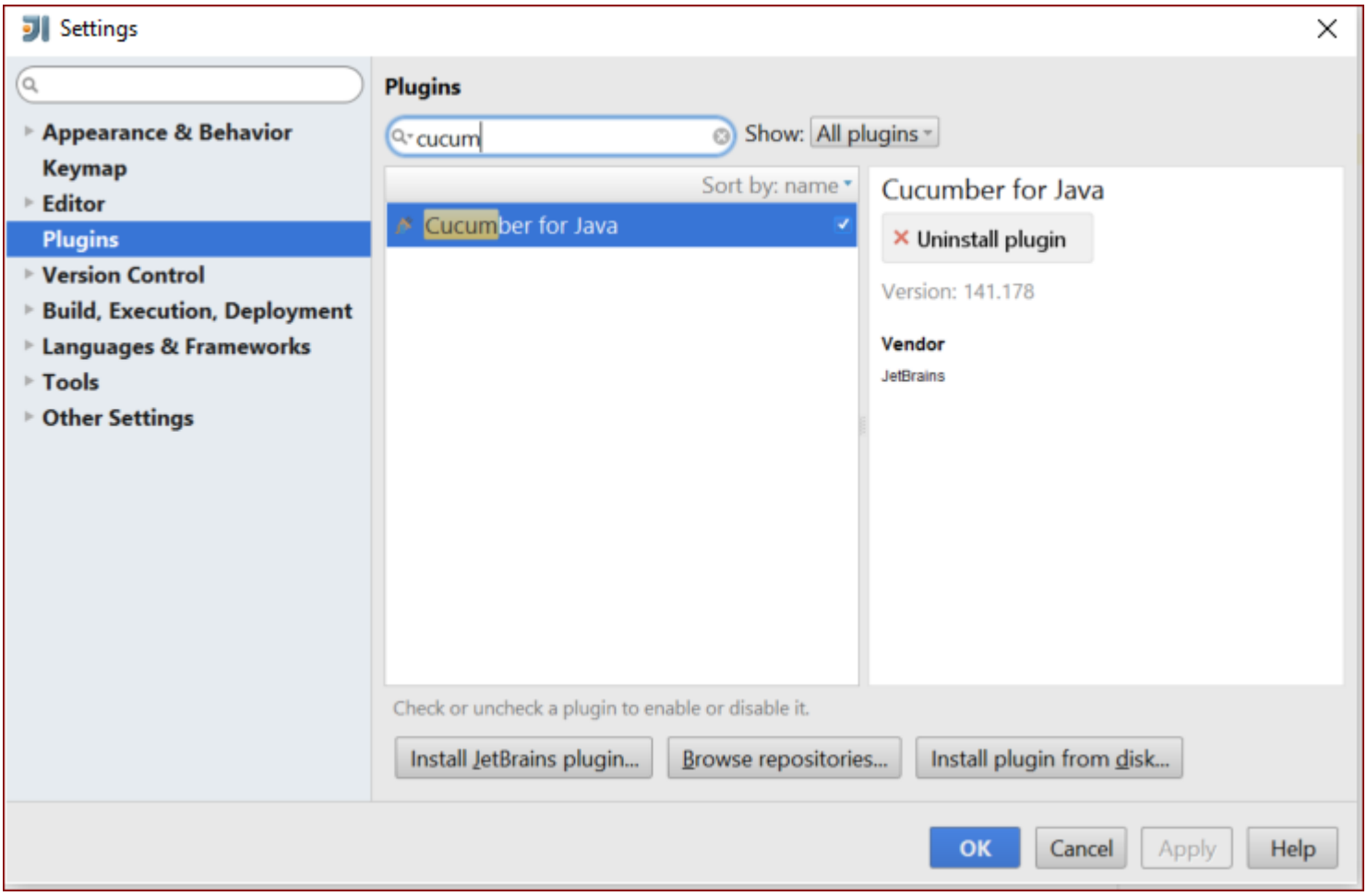
- Пошаговое создание клейкого ядра для невыполненных шагов
- Пошаговое завершение кода
- Перемещение по методу «код-клей»
- Выделение синтаксиса Gherkin в файлах «.feature», соответствующих шагу regex

и другие удобные функции.

Examples

Установить плагин огурца

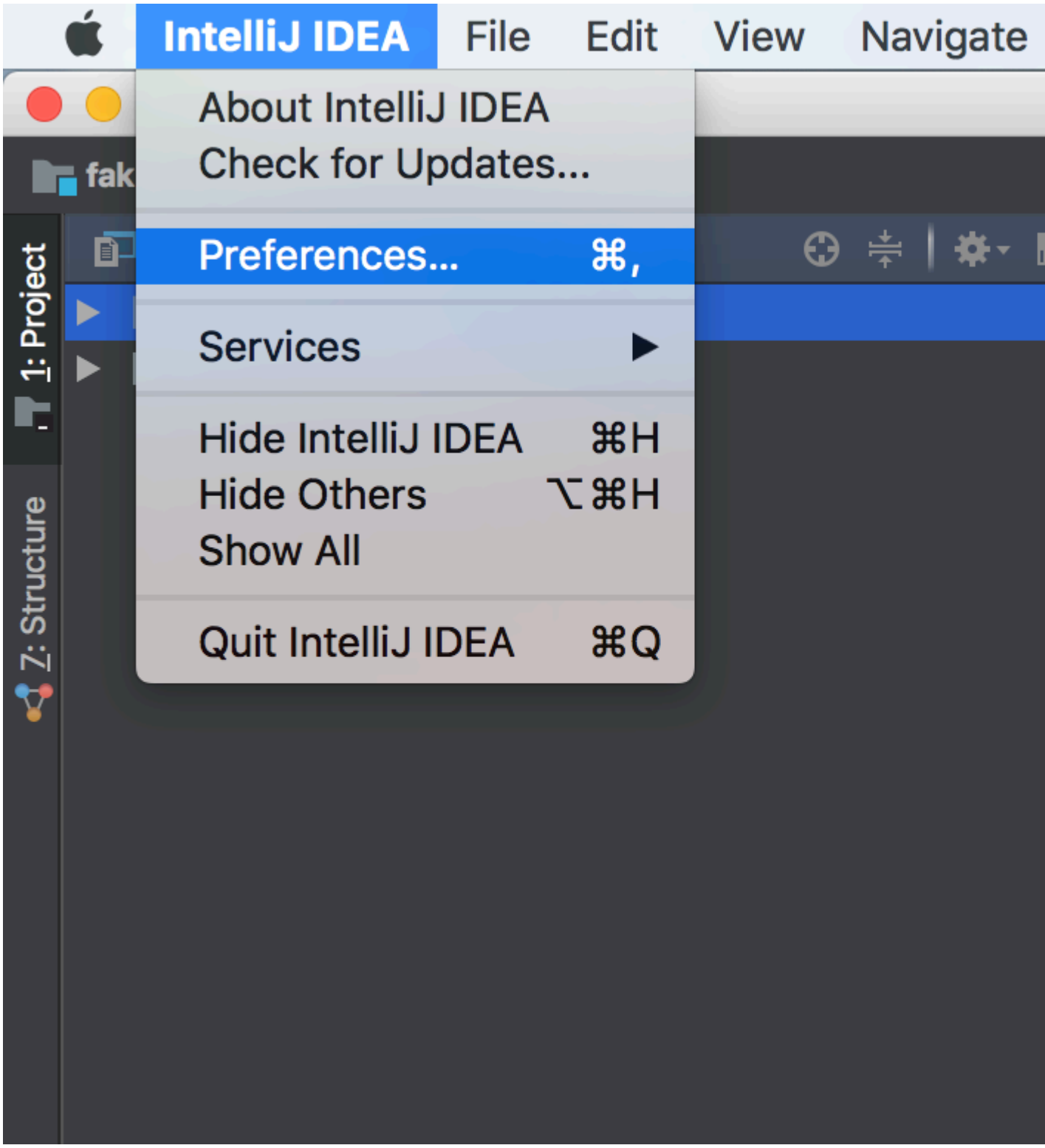
Перейдите в меню Файл -> Настройки -> щелкните плагины в левой панели -> Поиск огурца -> Установить плагин



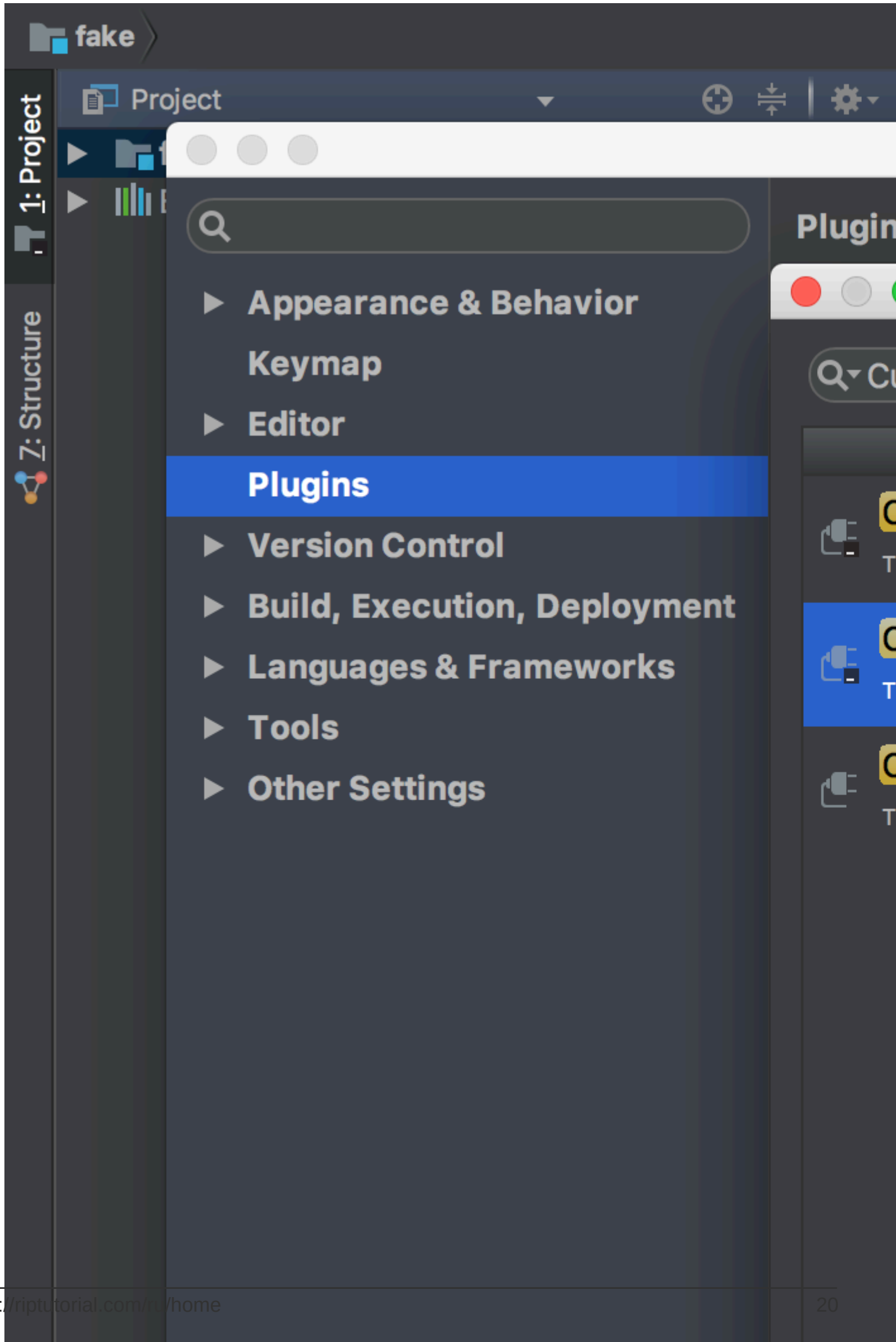
Установите IntelliJ Cucumber для Java Plugin (Mac)

Чтобы установить плагин Cucumber для Java для IntelliJ на Mac,

1. Запустите IntelliJ IDEA.
2. Нажмите на вкладку «IntelliJ IDEA» в верхней панели.



3. Нажмите «Настройки».
4. В разделе «Настройки / Настройки» нажмите «Плагины» в левой панели.
5. Нажмите кнопку «Browse Repositories», которая откроет новое окно.
6. Найдите «Огурцы» в строке поиска.


fake
Project
1: Project
Z: Structure
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
Tools
Other Settings

7. Установите плагин «Огурец для Java».
8. Перезапустите среду IDE для того, чтобы плагин вступил в силу. Теперь установлен Cucumber для Java.

Cucumber для Java.

https://riptutorial.com/ru/cucumber/topic/8356/установить-плагин-огурца-в-intellij

глава 6: Характеристики

Вступление

Вы можете использовать огурец в качестве плагина в QTP и Selenium. Шагами в сценарии огурца являются глобальные переменные. Вы можете реализовать один раз и позвонить много раз. Следовательно, уменьшает обслуживание кода и может повторно использовать тот же код, когда это необходимо.

замечания

Функции огурца записываются на языке Геркина и хранятся в файлах с суффиксом `.feature` . В этом разделе приводятся примеры каждой особенности Геркина.

Examples

Минимальная функция огурца

В функциях / documentation.feature:

```
Feature: Documentation

  Scenario: User views documentation
    When I go to the "Cucumber" documentation
    Then I should see the "Cucumber" documentation
```

Минимальная функция имеет линию `Feature` и `Scenario` с одним или несколькими шагами, начинающимися с « `When` , « `Then` или «другое ключевое слово Gherkin».

Разумный сценарий, вероятно, имел бы более одного шага.

Схема сценария

Шаблон, как показано ниже

```
Scenario Outline: As a homemaker i want to buy and pay for the below product
  Given I purchase <a product>
    And I require a carry bag to take things to home
  When I pay bill using <payment method> to successfully checkout
  Then I should have a receipt

Examples:
| a product     | payment method |
|  Cake         | Visa           |
|  Coke         | Paypal         |
```

Использование синтаксиса

```
Feature: Some terse yet descriptive text of what is desired
     Textual description of the business value of this feature
     Business rules that govern the scope of the feature
     Any additional information that will make the feature easier to understand

Background:
    Given some precondition needed for all scenarios in this file
        And another precondition

Scenario: Some determinable business situation
  Textual description of the business value of this scenario
  Business rules that govern the scope of the scenario
  Any additional information that will make the scenario easier to understand
    Given some precondition
      And some other precondition
    When some action by the actor
      And some other action
      And yet another action
    Then some testable outcome is achieved
      And something else we can check happens too
      But something else we can check does not happen

Scenario Outline: Some determinable business situation
    Given I am <precondition>
        And some other precondition
    When some action by the actor
    Then I have <outcome> rights

Examples:
    | precondition | outcome  |
    | username1    | customer |
    | username2    | admin    |
```

Следующие ключевые слова являются взаимозаменяемыми, но в зависимости от контекста могут быть лучше использованы:

- `Feature:` | `Ability:` | `Business Need:`
- `Scenario Outline:` | `Scenario Template:`
- `Examples:` | `Scenarios:`
- `Given` | `When` | `Then` | `And` | `But` | `*` |

Прочитайте Характеристики онлайн: https://riptutorial.com/ru/cucumber/topic/6023/характеристики

кредиты

S. No	Главы	Contributors
1	Начало работы с огурцом	Community, Dave Schweisguth, Mo H., Roberto Lo Giacco, SirLenz0rlot, user3554664
2	pom.xml для проекта Maven_ cucumber.	user
3	Определения шагов	Dave Schweisguth
4	Синтаксис Сердца	jordiPons, tramstheman, user3554664
5	Установить плагин огурца в Intellij	George Pantazes, Priya
6	Характеристики	Dave Schweisguth, Kyle Fairns, Priya