EBook Gratis

APRENDIZAJE cucumber

Free unaffiliated eBook created from
Stack Overflow contributors.

#cucumber

Tabla de contenido

Acerca de

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

Capítulo 1: Empezando con el pepino

Observaciones

Sobre el pepino

El pepino es una herramienta que ejecuta especificaciones ejecutables de software. Las especificaciones, llamadas "características", están escritas en lenguaje natural estructurado. Cucumber ejecuta una función asignando cada uno de sus pasos a una "definición de paso" escrita en el lenguaje de programación compatible con esa implementación de Cucumber. Pepino se implementa en muchos lenguajes de programación, incluidos Ruby (el original), Java y Javascript . También se traduce a muchos idiomas humanos.

Pepino fue escrito para respaldar la metodología ágil llamada Behavior-Driven Development (BDD). En BDD, uno comienza el desarrollo desde afuera entrando las pruebas de aceptación que describen la funcionalidad del software desde el punto de vista del usuario (en lugar de hacerlo desde el punto de vista de un programador, como en las pruebas unitarias). Las características del pepino sirven como estas pruebas de aceptación.

En general, las características de Cucumber son documentación legible por humanos, que también es un conjunto de pruebas ejecutables, lo que significa que la documentación y las pruebas siempre están de acuerdo. El pepino es útil para comunicarse con partes interesadas que no son programadores sobre documentación y pruebas. También permite a los programadores escribir pruebas a un nivel conceptual sin preocupaciones irrelevantes del lenguaje de programación.

El pepino se utiliza con mayor frecuencia para especificar y probar aplicaciones web, utilizando un controlador de navegador como Selenium o PhantomJS. Sin embargo, se puede usar con cualquier software que se pueda ejecutar y cuyo estado o resultados se puedan determinar a partir del lenguaje de programación que admite una implementación de Cucumber.

Otra documentación

La documentación oficial está en https://cucumber.io/docs . La documentación generada a partir de las características de Cucumber que describen las implementaciones de Cucumber se encuentra en

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

https://relishapp.com/explore incluye algunas otras herramientas y ejemplos relacionados con Cucumber, aunque no, desafortunadamente, Cucumber-JVM.

Este tema

Este tema solo debe dar algunos ejemplos que introducen al lector a los conceptos de pepino. Otras secciones brindarán ejemplos completos de instalación, uso de línea de comandos e IDE,

características, definiciones de pasos, etc.

Examples

Una característica de pepino

Cucumber utiliza la sintaxis de Gherkin para describir los comportamientos de su software en lenguaje natural estructurado.

Como tal, Pepino **no** es un marco de prueba (un malentendido común), sino un *marco de documentación del sistema* , no muy diferente de otros como el caso de uso. El malentendido común se debe al hecho de que la documentación de Pepino *se puede automatizar para garantizar que refleje el comportamiento real del sistema* .

Un paquete de documentación de Cucumber está compuesto por `Features` , cada una de las cuales describe una característica de su software, escrita en Gherkin y alojada en su propio archivo. Al organizar esos archivos en una estructura de directorios, puede *agrupar* y *organizar* funciones:

- bancario/
  - retiro.una característica
  - atm.feature
  - personal-loan.feature
- comercio/
  - portfolio.feature
  - intradía.
- hipoteca/
  - evaluación.
  - característica de contabilidad

Cada `Feature` es un archivo de texto sin formato compuesto por una sección de introducción opcional, no estructurada, puramente informativa y uno o más `Scenarios` , cada uno de los cuales representa una condición de uso o un caso de uso.

Ejemplo:

```
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"
```

Cada línea que comienza con *Dado* , *Cuándo* , *Y* , *Pero* o *Entonces* se llama `Step` . Cualquier paso puede comenzar con cualquiera de esas palabras, independientemente del orden, pero es

convencional usarlas de la manera más natural posible.

Las características también se pueden organizar a través de `Tags` , anotaciones que el editor puede colocar en una `Feature` o un `Scenario` para categorizarlo aún más.

La ejecutabilidad de una característica se logra a través del código de *cola* que puede escribirse en muchos idiomas diferentes (Java, Ruby, Scala, C / C ++): cada `Step` se compara con el código de cola para identificar `Step Definitions` (comúnmente abreviadas a *StepDef*) a través de expresiones regulares.

Cada `Step` puede tener solo una `Step Definition` asociada.

Cuando se ejecuta una `Feature` se ejecuta cada `Scenario` composición, lo que significa que cada StepDef que coincida con el `Step` s en cada `Scenario` se ejecuta.

Instalación de puro rubí

Para instalar Cucumber para usar con Ruby simplemente usa el comando

```
gem install cucumber
```

Alternativamente, si está usando un agrupador, puede agregar la siguiente línea a su Gemfile

```
gem 'cucumber'
```

Y luego ejecute bundler

```
bundle install
```

[Creo que esto pertenece a su propio tema, Instalación. Creé ese tema y copié este ejemplo allí. Cuando se apruebe ese tema, lo moveré allí y eliminaré la copia.]

Una definición de paso de pepino en rubí

En features / 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
```

Estos pasos ejercitan una aplicación web. Son lo más simples que pueden ser al mismo tiempo que siguen siendo prácticos.

Cada paso comienza con una palabra clave Gherkin, que en un archivo de definición de paso es un método que registra un paso con Cucumber. El método de definición de pasos toma una expresión regular, que coincide con una línea en un escenario, y un bloque, que se ejecuta cuando el escenario llega a una línea coincidente. Los grupos de captura en la expresión regular se pasan al bloque como parámetros de bloque.

El paso `When` tiene un ejemplo simple, en línea, de pasar de una referencia legible a una página ("Documentación") a una URL. Las suites de Real Cucumber usualmente ponen esta lógica en un método separado. El método de `visit` es proporcionado por Capybara. Capybara no está obligada a usar Pepino, aunque es muy común que se use con él. `visit` le dice al navegador controlado por Capybara que visite la URL dada.

El paso `Then` muestra cómo se puede probar el contenido de una página. `expect` / `to` es proporcionado por RSpec (de nuevo, no es requerido por Cucumber pero se usa muy comúnmente con él). `have_css` es proporcionado por Capybara. La expectativa es que el selector CSS dado coincida con un elemento en la página que contiene el texto dado. Tenga en cuenta que esta expectativa fallaría si la solicitud del navegador hubiera fallado.

Para más ejemplos, vea el tema "Definición de pasos" .

Lea Empezando con el pepino en línea:
https://riptutorial.com/es/cucumber/topic/4875/empezando-con-el-pepino

Capítulo 2: Caracteristicas

Introducción

Puedes usar pepino como complemento en QTP y Selenium. Los pasos en el escenario del pepino son variables globales. Se puede implementar una vez y llamar muchas veces. Por lo tanto, reduce el mantenimiento del código y puede reutilizar el mismo código cuando sea necesario.

Observaciones

Las características del pepino se escriben en el lenguaje Gherkin y se almacenan en archivos con el sufijo `.feature` . Este tema da ejemplos de cada característica de Gherkin.

Examples

Una característica mínima de pepino

En características / documentación.

```
Feature: Documentation

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

Una característica mínima tiene una línea de `Feature` y un `Scenario` con uno o más pasos que comienzan con `When` , `Then` u otra palabra clave de Gherkin.

Un escenario sensato probablemente tendría más de un paso.

Esquema del escenario

Plantilla como abajo

```
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         |
```

Uso de la sintaxis

```
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    |
```

Las siguientes palabras clave son intercambiables, pero según el contexto, puede ser mejor usar:

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

Lea Caracteristicas en línea: https://riptutorial.com/es/cucumber/topic/6023/caracteristicas

Capítulo 3: Definiciones de pasos

Observaciones

Las definiciones de los pasos están en el lenguaje de programación soportado por una implementación dada de Cucumber. Este tema proporciona ejemplos de definiciones de pasos en cada lenguaje de programación compatible y ejemplos de uso de llamadas a la API de Cucumber en definiciones de pasos.

Examples

Algunas definiciones simples de Ruby Step.

En features / 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
```

Estos pasos ejercitan una aplicación web. Son lo más simples que pueden ser al mismo tiempo que siguen siendo prácticos.

Cada paso comienza con una palabra clave Gherkin, que en un archivo de definición de paso es un método que registra un paso con Cucumber. El método de definición de pasos toma una expresión regular, que coincide con una línea en un escenario, y un bloque, que se ejecuta cuando el escenario llega a una línea coincidente. Los grupos de captura en la expresión regular se pasan al bloque como parámetros de bloque.

El paso `When` tiene un ejemplo simple, en línea, de pasar de una referencia legible a una página ("Documentación") a una URL. Las suites de Real Cucumber usualmente ponen esta lógica en un método separado. El método de `visit` es proporcionado por Capybara. Capybara no está obligada a usar Pepino, aunque es muy común que se use con él. `visit` le dice al navegador controlado por Capybara que visite la URL dada.

El paso `Then` muestra cómo se puede probar el contenido de una página. `expect` / `to` es proporcionado por RSpec (de nuevo, no es requerido por Cucumber pero se usa muy comúnmente con él). `have_css` es proporcionado por Capybara. La expectativa es que el selector

CSS dado coincida con un elemento en la página que contiene el texto dado. Tenga en cuenta que esta expectativa fallaría si la solicitud del navegador hubiera fallado.

Lea Definiciones de pasos en línea: https://riptutorial.com/es/cucumber/topic/5681/definiciones-de-pasos

Capítulo 4: Instalar el complemento de pepino en Intellij

Introducción

Los complementos de Cucumber para IntelliJ IDEA ofrecen funciones IDE convenientes para trabajar con archivos de características de Gherkin en un proyecto de IntelliJ utilizando el marco de Cucumber. Los complementos están disponibles para los lenguajes Java, Scala o Groovy.

Observaciones

El complemento de IntelliJ de Cucumber for Java ofrece funciones IDE para desarrollarlo cómodamente con Cucumber, incluidas

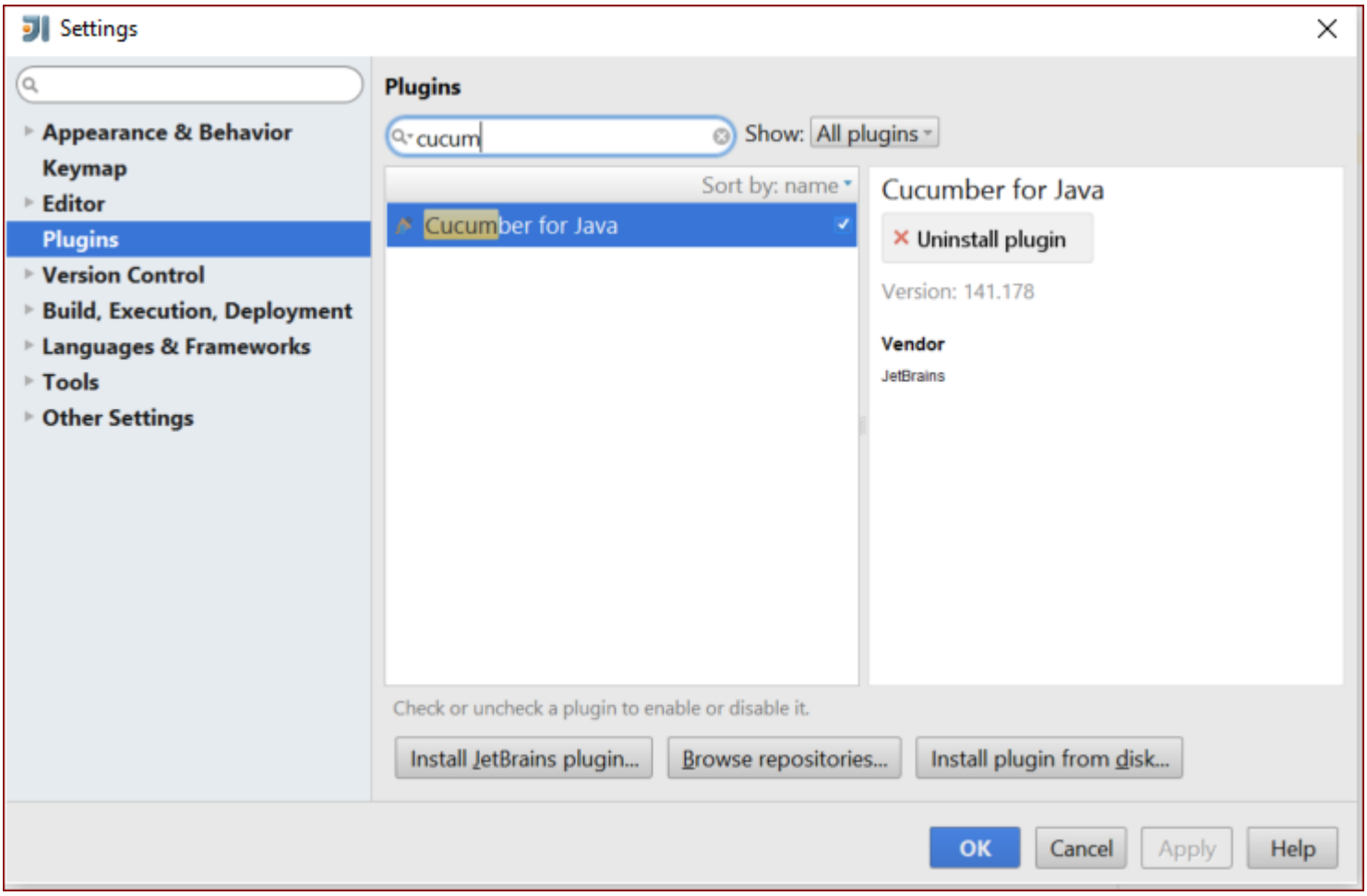
- Gherkin step glue generation para pasos no implementados.
- Finalización del código de paso de pepinillo
- Salto de código de método de paso a pegar
- Resaltado de sintaxis de Gherkin en archivos ".feature" que coinciden con la expresión regular de pasos

y otras características convenientes.

Examples

Instalar el complemento de pepino

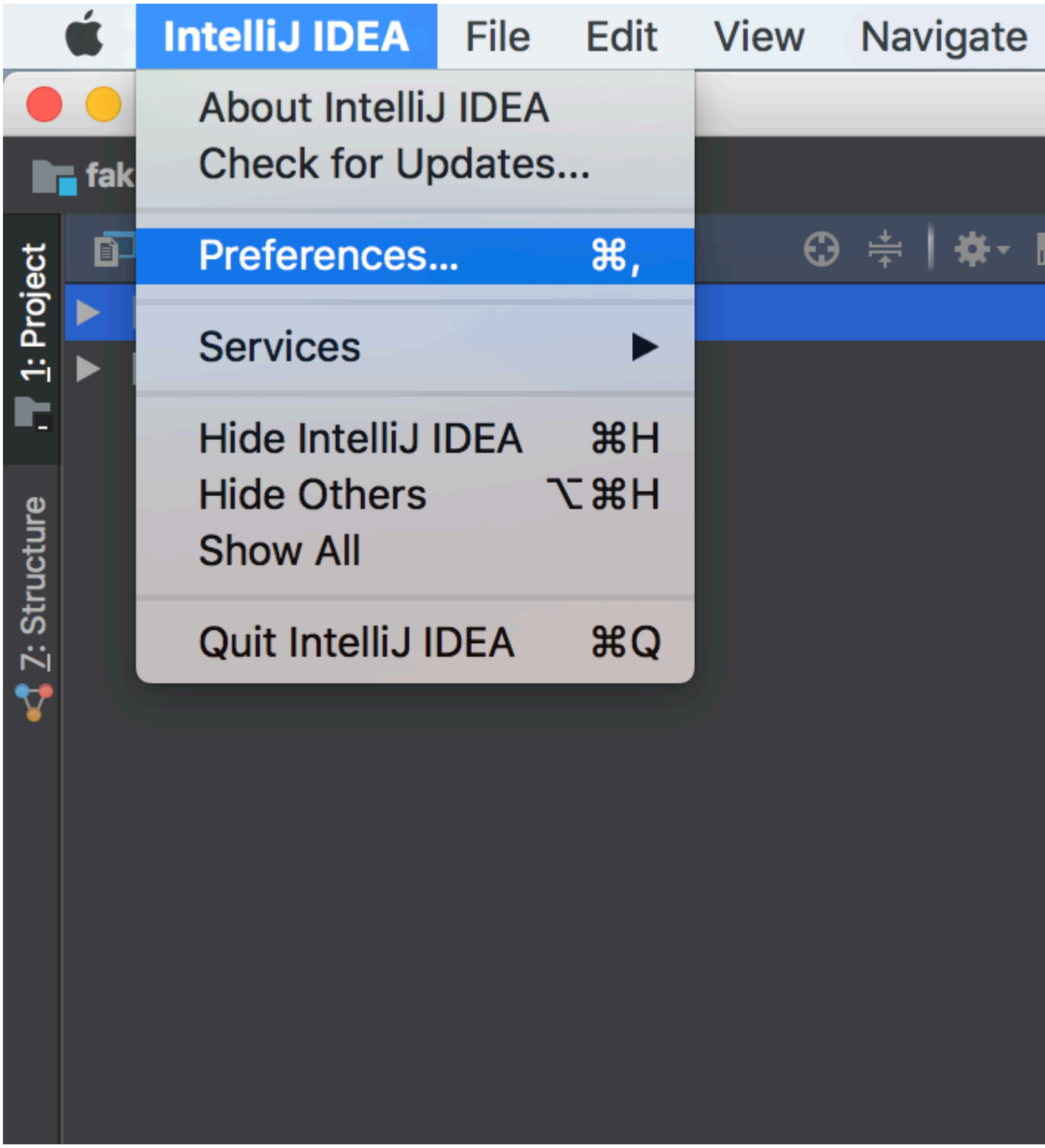
Vaya a Archivo -> Configuración -> haga clic en complementos en el panel de la izquierda -> Buscar pepino -> Instalar complemento



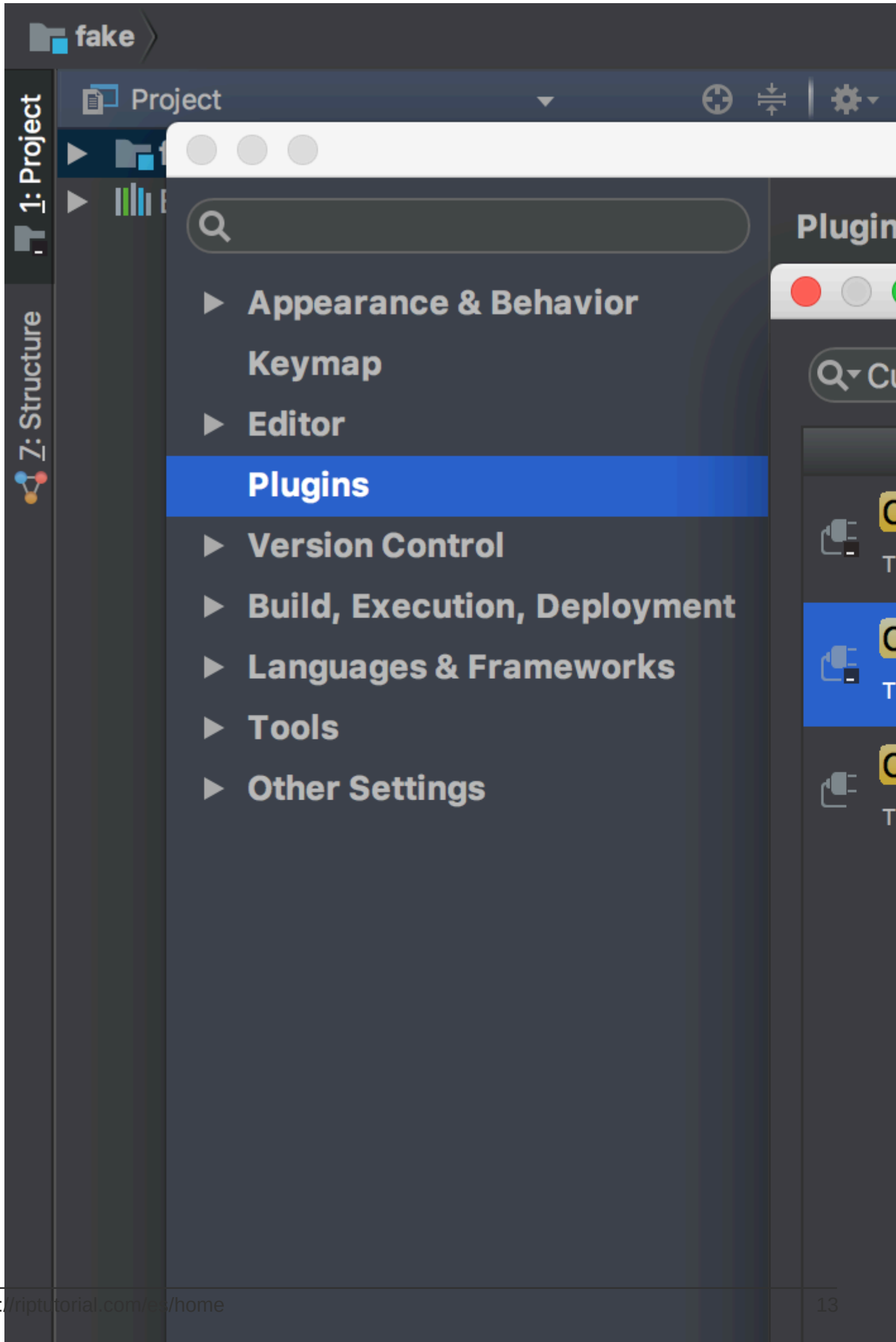
Instalar IntelliJ Cucumber para Java Plugin (Mac)

Para instalar el complemento Cucumber for Java para IntelliJ en una Mac,

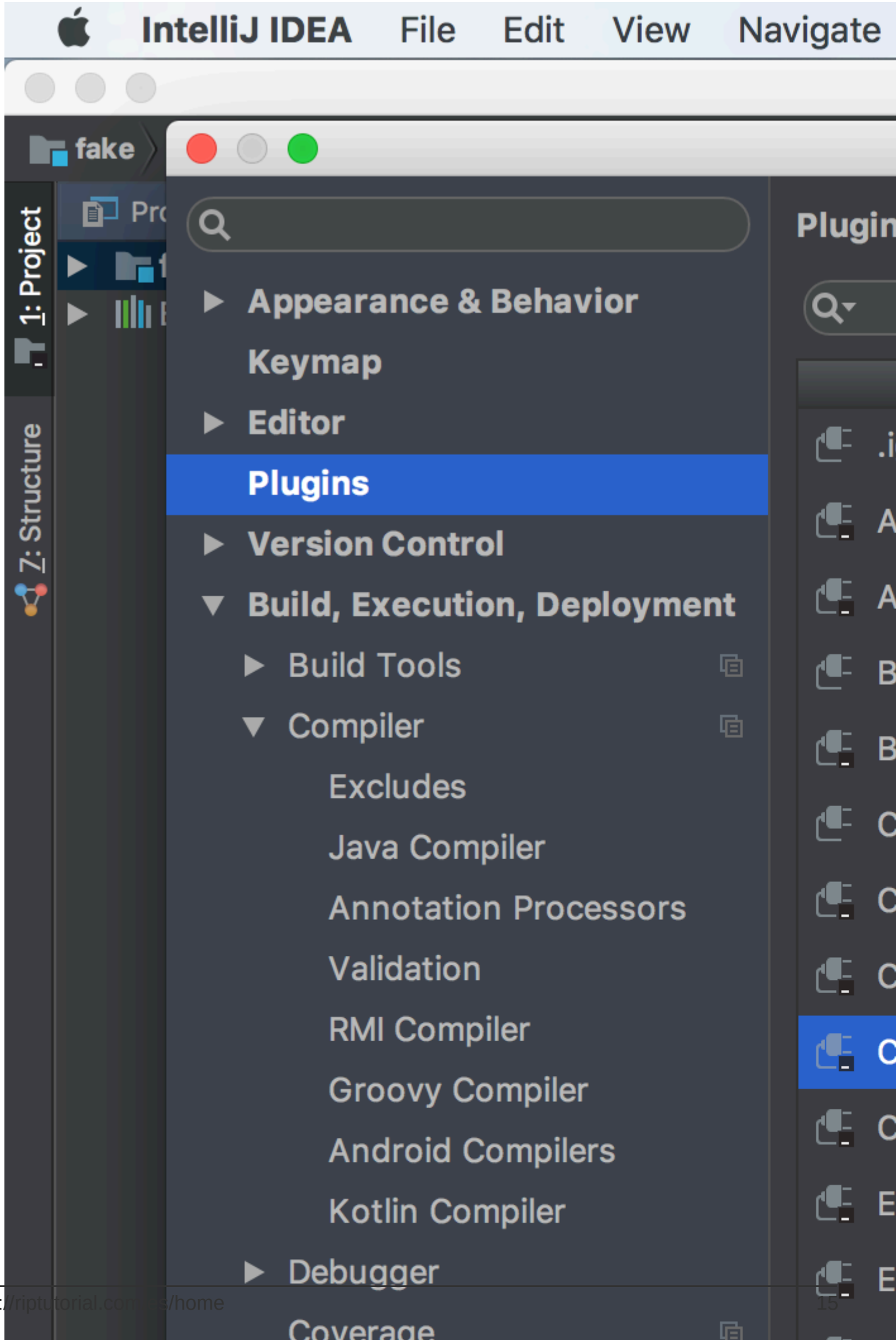
1. Iniciar IntelliJ IDEA.
2. Haga clic en la pestaña "IntelliJ IDEA" en la barra superior.



3. Haga clic en "Preferencias".
4. En Preferencias / Configuración, haga clic en "Complementos" en el panel de la izquierda.
5. Haga clic en el botón "Examinar repositorios", que abre una nueva ventana.
6. Busca "Pepino" en la barra de búsqueda.


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

7. Instala el plugin "Cucumber for Java".
8. Reinicie el IDE para que el complemento surta efecto. El pepino para Java ya está instalado.


IntelliJ IDEA
File
Edit
View
Navigate
fake
1: Project
7: Structure
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Build Tools
Compiler
Excludes
Java Compiler
Annotation Processors
Validation
RMI Compiler
Groovy Compiler
Android Compilers
Kotlin Compiler
Debugger
Coverage

https://riptutorial.com/es/cucumber/topic/8356/instalar-el-complemento-de-pepino-en-intellij

Capítulo 5: pom.xml para el proyecto pepino Maven_

Introducción

El siguiente modelo de objeto de proyecto es la plantilla pom.xml. Si desea crear un proyecto de maven con pepino, puede usar el siguiente ejemplo como plantilla

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>
```

Lea pom.xml para el proyecto pepino Maven_ en línea:
https://riptutorial.com/es/cucumber/topic/8331/pom-xml-para-el-proyecto-pepino-maven-

Capítulo 6: Sintaxis de pepinillo

Introducción

Gherkin es un lenguaje legible para negocios para la automatización de pruebas y la documentación de pruebas. Se entiende por pepino y en conjunto existe como una herramienta de desarrollo impulsada por el comportamiento.

Sintaxis

- Característica: esta palabra clave significa que lo que sigue es una descripción básica o el nombre de la característica que se está probando o documentando.
- Fondo: esta palabra clave indica los pasos que se ejecutarán antes de cada escenario en la función.
- Escenario: esta palabra clave representa el nombre o la descripción básica de un escenario particular que prueba la característica.
- Esquema del escenario: esta palabra clave significa que el escenario se ejecutará N veces para cada argumento enumerado en los ejemplos que se pasaron explícitamente por el nombre de la columna entre paréntesis angulares.
- Ejemplos: esta palabra clave anota la lista de argumentos estáticos que se pasarán a un esquema de escenario.
- Dado: esta palabra clave representa un paso dado, o condición previa que se supone antes de continuar. En el paradigma Organizar, Actuar, Asertar, dado representa "Organizar".
- Cuándo: esta palabra clave representa un paso cuándo, o el comportamiento contra el que se va a afirmar. En el paradigma Organizar, Actuar, Afirmar, dado representa "Actuar".
- Luego: esta palabra clave representa un paso en ese momento, o en otras palabras, el paso en el que se valida el resultado de un comportamiento. En el paradigma Organizar, Actuar, Asertar, dado representa "Afirmar".
- Y: esta palabra clave se utiliza junto con cualquiera de las palabras clave anteriores. Si tiene dos declaraciones dadas, en lugar de llamar explícitamente a Dadas dos veces, puede decir "Dado A y B".

Examples

Los basicos

Este ejemplo repasará la estructura básica de un archivo de características de Cucumber en Gherkin. Los archivos de características utilizan varias palabras clave en la sintaxis básica.

Veamos las palabras clave básicas:

- **Característica:** esta palabra clave significa que lo que sigue es una descripción básica o el nombre de la característica que se está probando o documentando.
- **Escenario:** esta palabra clave representa el nombre o la descripción básica de un escenario

particular que prueba la característica.
- **Dada** esta palabra clave representa un paso dado, o condición previa que se asume antes de continuar. En el paradigma Organizar, Actuar, Asertar, dado representa "Organizar".
- **Cuando** esta palabra clave representa un paso cuándo, o el comportamiento contra el que se va a afirmar. En el paradigma Organizar, Actuar, Afirmar, dado representa "Actuar".
- **Luego,** esta palabra clave representa un paso en ese momento, o en otras palabras, el paso en el que se valida el resultado de un comportamiento. En el paradigma Organizar, Actuar, Asertar, dado representa "Afirmar".
- **Y** esta palabra clave se utiliza junto con cualquiera de las palabras clave anteriores. Si tiene dos declaraciones dadas, en lugar de llamar explícitamente a Dadas dos veces, puede decir "Dado A y B".
- **Pero** esta palabra clave se usa en conjunto **Dadas** , **cuándo** y **luego** para indicar que algo no debería suceder. Entonces A, pero no B.

Todas las palabras clave deben estar en una nueva línea y deben ser la primera palabra en una nueva línea para que el analizador Gherkin las reconozca. Las palabras clave Característica y Escenario deben tener dos puntos inmediatamente después, como se expresa en el siguiente ejemplo. Dado, cuándo, entonces y no requieren dos puntos.

Además de las palabras clave, puede escribir descripciones y comentarios. Las descripciones aparecen después de la palabra clave pero en la misma línea, mientras que los comentarios se producen en las líneas debajo de las palabras clave. Al escribir comentarios de características, es habitual proporcionar reglas explícitas que describan los bordes y las condiciones que conducen a diferentes resultados de los comportamientos.

```
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
```

Pasos parametrizados

Al escribir Gherkin, puede haber ocasiones en las que desee parametrizar sus pasos para que el ingeniero que está implementando los planes de prueba reutilice el proceso. Los pasos reciben parámetros a través de grupos de captura de expresiones regulares. (**Nota de ingeniería:** si no

tiene parámetros coincidentes para cada grupo de captura en su expresión regular, puede esperar que se genere una "Descepción de pepino: Desigualdad de aridad") En el siguiente ejemplo, también hemos decidido envolver los argumentos entre comillas dobles. como aceptar enteros como argumentos.

```
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
```

Fondo de la característica

Como habrá notado en el ejemplo anterior, estamos reescribiendo el mismo paso varias veces:

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

Esto puede ser excepcionalmente tedioso, especialmente si tenemos más de un paso dado que se reutiliza. Gherkin proporciona una solución para esto al darnos otra palabra clave con la que trabajar: **Fondo:.**

La palabra clave de fondo sirve para ejecutar los pasos declarados debajo de ella antes de cada escenario en la Característica. Asegúrese de no agregar un paso de fondo a menos que esté

seguro de que es necesario para cada escenario. Al igual que las otras palabras clave, el Fondo va seguido de una descripción o nombre y puede tener comentarios enumerados debajo. Al igual que Característica y Escenario, el fondo debe ser procesado por dos puntos.

```
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
```

Esquema del escenario

En algunos casos, es posible que desee volver a ejecutar el mismo escenario una y otra vez, sustituyendo los argumentos. En este caso, Gherkin proporciona varias palabras clave nuevas para adaptarse a esta situación, **Esquema del escenario:** y **Ejemplo:.** La palabra clave del esquema del escenario le dice a Cucumber que el escenario se ejecutará varias veces sustituyendo los argumentos de una lista. La palabra clave de ejemplos se llama antes de que la lista sea anotada explícitamente. Los argumentos para los contornos del escenario se deben envolver en corchetes angulares. En el ejemplo a continuación, tenga en cuenta que los nombres de argumento envueltos en los corchetes en ángulo corresponden a los nombres de columna enumerados en Ejemplos. Cada columna está separada por barras verticales, con nombres de columna en la primera fila.

```
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     |
```

Etiquetas

Para los fines de la documentación, es posible que desee filtrar planes de prueba o escenarios por categorías. Los desarrolladores pueden querer ejecutar pruebas basadas en esas mismas categorías. Gherkin le permite clasificar las características, así como los escenarios individuales a través del usuario de etiquetas. En el ejemplo a continuación, observe que la palabra clave Característica anterior es la etiqueta "@Automation". Gherkin reconoce esto como una etiqueta del usuario del símbolo "@". En este ejemplo, el ingeniero desea dejar en claro que estas pruebas se utilizan para la automatización, donde no todas las pruebas se pueden automatizar, algunas pruebas deben realizarse mediante el control de calidad manual.

Observe también que la etiqueta @Production se ha agregado al bloqueo de usuario de prueba de escenario. En este ejemplo, esto se debe a que este escenario solo está activo en el entorno de producción de la aplicación. Los desarrolladores no quieren que sus cuentas de sandbox se bloqueen durante el desarrollo. Estas etiquetas les permiten imponer que esta prueba solo se ejecutará en el entorno de producción.

Por último, el esquema del escenario tiene la etiqueta @Staging. A los efectos de este ejemplo, esto se debe a que las cuentas que se utilizan son cuentas provisionales y no funcionarán en otros entornos. Al igual que la etiqueta @Production, esto garantiza que estas pruebas solo se ejecutarán en el entorno de ensayo.

Estos son solo algunos ejemplos de dónde, cómo y por qué puede usar etiquetas. En última instancia, estas etiquetas tendrán un significado para usted y para los desarrolladores, y pueden ser de cualquier tipo y se pueden utilizar para clasificarlas como mejor le parezca.

```
@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     |
```

Consejos de pepinillo

- Cada escenario prueba un comportamiento.
- Los escenarios se escriben de forma declarativa.
- Evitar detalles incidentales dentro del escenario.

- Omitir lo obvio
- Evita los pasos conjuntivos.
- Mantenga sus escenarios cortos
- No tienes que tener muchos escenarios en la misma característica.
- Utilizar nombres de escenarios descriptivos.
- Tener solo uno cuando paso
- Usa el "debería" en los siguientes pasos.

Lea Sintaxis de pepinillo en línea: https://riptutorial.com/es/cucumber/topic/9296/sintaxis-de-pepinillo

Creditos

S. No	Capítulos	Contributors
1	Empezando con el pepino	Community, Dave Schweisguth, Mo H., Roberto Lo Giacco, SirLenz0rlot, user3554664
2	Caracteristicas	Dave Schweisguth, Kyle Fairns, Priya
3	Definiciones de pasos	Dave Schweisguth
4	Instalar el complemento de pepino en Intellij	George Pantazes, Priya
5	pom.xml para el proyecto pepino Maven_	user
6	Sintaxis de pepinillo	jordiPons, tramstheman, user3554664