EBook Gratuito

APPRENDIMENTO cucumber

Free unaffiliated eBook created from
Stack Overflow contributors.

#cucumber

Sommario

Di

Capitolo 1: Iniziare con il cetriolo

Osservazioni

A proposito di Cetriolo

Cucumber è uno strumento che esegue le specifiche eseguibili del software. Le specifiche, chiamate "caratteristiche", sono scritte in un linguaggio naturale strutturato. Cucumber esegue una funzionalità mappando ciascuno dei suoi passaggi su una "definizione di passo" scritta nel linguaggio di programmazione supportato da tale implementazione di Cucumber. Cucumber è implementato in molti linguaggi di programmazione tra cui Ruby (l'originale), Java e Javascript . È anche tradotto in molte lingue umane.

Cucumber è stato scritto per supportare la metodologia agile denominata Behaviour Driven Development (BDD). In BDD, uno inizia lo sviluppo all'esterno scrivendo test di accettazione che descrivono le funzionalità del software dal punto di vista dell'utente (piuttosto che dal punto di vista di un programmatore come nei test di unità). Le funzioni di cetriolo fungono da questi test di accettazione.

In generale, le funzionalità di Cucumber sono documenti leggibili dall'utente, che è anche una suite di test eseguibile, il che significa che la documentazione e i test sono sempre d'accordo. Cucumber è utile per comunicare con gli stakeholder non programmatori sulla documentazione e sui test. Consente inoltre ai programmatori di scrivere test a livello concettuale senza problemi di linguaggio di programmazione irrilevanti.

Cucumber viene spesso utilizzato per specificare e testare le applicazioni Web, utilizzando un driver del browser come Selenium o PhantomJS. Tuttavia, può essere utilizzato con qualsiasi software che può essere eseguito e il cui stato o i cui risultati possono essere determinati dal linguaggio di programmazione supportato da un'implementazione di Cucumber.

Altra documentazione

La documentazione ufficiale è su https://cucumber.io/docs . Documentazione generata dalle funzionalità di Cucumber che descrivono le implementazioni di Cucumber

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

https://relishapp.com/explore include alcuni altri strumenti ed esempi relativi a Cucumber, anche se, sfortunatamente, Cucumber-JVM.

Questo argomento

Questo argomento dovrebbe fornire solo alcuni esempi che introducono il lettore ai concetti di Cucumber. Altre sezioni forniranno esempi completi di installazione, utilizzo da riga di comando e IDE, funzionalità, definizioni dei passaggi, ecc.

Examples

Una funzionalità di cetriolo

Cucumber utilizza la sintassi Gherkin per descrivere i comportamenti del tuo software in un linguaggio naturale strutturato.

In quanto tale, Cucumber **non** è un framework di test (un comune malinteso), ma un *framework di documentazione di sistema* , non molto diverso da altri come Use Case Scenario. Il comune fraintendimento è dovuto al fatto che la documentazione di Cucumber *può essere automatizzata per garantire che rifletta il comportamento reale del sistema* .

Una suite di documentazione di Cucumber è composta da `Features` , ognuna delle quali descrive una funzionalità del software, scritta in Gherkin e ospitata nel proprio file. Organizzando questi file in una struttura di directory è possibile *raggruppare* e *organizzare le* funzionalità:

- bancario/
  - withdrawal.feature
  - atm.feature
  - personal-loan.feature
- negoziazione /
  - portfolio.feature
  - intraday.feature
- mutuo/
  - evaluation.feature
  - accounting.feature

Ogni `Feature` è un file di testo semplice composto da una sezione introduttiva facoltativa, non strutturata, puramente informativa e uno o più `Scenarios` , ognuno dei quali rappresenta una condizione di utilizzo o un caso d'uso.

Esempio:

```
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"
```

Ogni riga che inizia con *Given* , *When* , *And* , *But* or *Then* è detta `Step` . Qualsiasi passaggio può iniziare con una qualsiasi di quelle parole indipendentemente dall'ordine, ma è normale utilizzarle nel modo più naturale possibile.

Le caratteristiche possono anche essere organizzate tramite `Tags` , annotazioni che l'editor può

mettere su una `Feature` o uno `Scenario` per categorizzarlo ulteriormente.

L'eseguibilità di una feature è ottenuta tramite il *glue* code che può essere scritto in molti linguaggi diversi (Java, Ruby, Scala, C / C ++): ogni `Step` è abbinato al codice della colla per identificare le `Step Definitions` (comunemente abbreviato in *StepDef*) tramite espressioni regolari.

Ogni `Step` può avere solo una `Step Definition` associata.

Quando viene eseguita una `Feature` viene eseguito ogni `Scenario` composizione, vale a dire che ogni StepDef che corrisponde allo `Step` s in ogni `Scenario` viene eseguito.

Installazione Pure Ruby

Per installare Cucumber per l'uso con Ruby è sufficiente utilizzare il comando

```
gem install cucumber
```

In alternativa, se stai usando bundler, puoi aggiungere la seguente riga al tuo Gemfile

```
gem 'cucumber'
```

E quindi eseguire bundler

```
bundle install
```

[Penso che questo appartenga al suo stesso argomento, Installazione. Ho creato quell'argomento e ho copiato questo esempio lì. Quando l'argomento è approvato, lo sposterò lì ed eliminerò la copia.]

Una definizione di passo Cucumber in Ruby

In 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
```

Questi passaggi esercitano un'applicazione web. Sono semplici quanto possono essere mentre sono ancora pratici.

Ogni fase inizia con una parola chiave Gherkin, che in un file di definizione passo è un metodo che registra un passo con Cucumber. Il metodo di definizione passo prende un'espressione regolare, che corrisponde a una riga in uno scenario e un blocco, che viene eseguito quando lo scenario raggiunge una linea corrispondente. Cattura i gruppi nell'espressione regolare vengono passati al blocco come parametri di blocco.

Il passaggio `When` ha un semplice esempio in linea di passare da un riferimento leggibile da un utente a una pagina ("Documentazione") a un URL. Le suite di Real Cucumber di solito mettono questa logica in un metodo separato. Il metodo di `visit` è fornito da Capybara. Capibara non è tenuto a usare Cetriolo, sebbene sia molto usato con esso. `visit` dice al browser controllato da Capybara di visitare l'URL indicato.

Il passaggio `Then` mostra come è possibile testare il contenuto di una pagina. `expect` / `to` è fornito da RSpec (di nuovo, non richiesto da Cucumber ma molto comunemente usato con esso). `have_css` è fornito da Capybara. L'aspettativa è che il selettore CSS specificato corrisponda a un elemento della pagina che contiene il testo specificato. Si noti che questa aspettativa fallirebbe se la richiesta del browser fosse fallita.

Per ulteriori esempi, vedere l'argomento "Definizione step" .

Leggi Iniziare con il cetriolo online: https://riptutorial.com/it/cucumber/topic/4875/iniziare-con-il-cetriolo

Capitolo 2: Caratteristiche

introduzione

Puoi usare cetriolo come plugin in QTP e selenio. I passaggi nello scenario cetriolo sono variabili globali. Puoi implementare una volta e chiamare più volte. Quindi riduce la manutenzione del codice e può riutilizzare lo stesso codice quando richiesto.

Osservazioni

Caratteristiche cetriolo sono scritti nel linguaggio Gherkin e memorizzati in file con il suffisso `.feature` . Questo argomento fornisce esempi di ciascuna caratteristica di Gherkin.

Examples

Una caratteristica minima di Cetriolo

In caratteristiche / documentazione.feature:

```
Feature: Documentation

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

Una funzionalità minima ha una linea `Feature` e uno `Scenario` con uno o più passaggi che iniziano con `When` , `Then` o un'altra parola chiave Gherkin.

Uno scenario ragionevole avrebbe probabilmente più di un passo.

Scenario

Modello come sotto

```
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 della sintassi

```
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    |
```

Le seguenti parole chiave sono intercambiabili, ma a seconda del contesto, potrebbe essere meglio usare:

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

Leggi Caratteristiche online: https://riptutorial.com/it/cucumber/topic/6023/caratteristiche

Capitolo 3: Definizioni passo

Osservazioni

Le definizioni di passaggio sono nel linguaggio di programmazione supportato da una determinata implementazione di Cucumber. Questo argomento fornisce esempi di definizioni di passaggi in ciascun linguaggio di programmazione supportato ed esempi di utilizzo di chiamate API Cucumber nelle definizioni di passaggi.

Examples

Alcune semplici definizioni di passi di Ruby

In 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
```

Questi passaggi esercitano un'applicazione web. Sono semplici quanto possono essere mentre sono ancora pratici.

Ogni fase inizia con una parola chiave Gherkin, che in un file di definizione passo è un metodo che registra un passo con Cucumber. Il metodo di definizione passo prende un'espressione regolare, che corrisponde a una riga in uno scenario e un blocco, che viene eseguito quando lo scenario raggiunge una linea corrispondente. Cattura i gruppi nell'espressione regolare vengono passati al blocco come parametri di blocco.

Il passaggio `When` ha un semplice esempio in linea di passare da un riferimento leggibile da un utente a una pagina ("Documentazione") a un URL. Le suite di Real Cucumber di solito mettono questa logica in un metodo separato. Il metodo di `visit` è fornito da Capybara. Capibara non è tenuto a usare Cetriolo, sebbene sia molto usato con esso. `visit` dice al browser controllato da Capybara di visitare l'URL indicato.

Il passaggio `Then` mostra come è possibile testare il contenuto di una pagina. `expect` / `to` è fornito da RSpec (di nuovo, non richiesto da Cucumber ma molto comunemente usato con esso). `have_css` è fornito da Capybara. L'aspettativa è che il selettore CSS specificato corrisponda a un

elemento della pagina che contiene il testo specificato. Si noti che questa aspettativa fallirebbe se la richiesta del browser fosse fallita.

Leggi Definizioni passo online: https://riptutorial.com/it/cucumber/topic/5681/definizioni-passo

Capitolo 4: Installa il plugin cetriolo in Intellij

introduzione

I plugin Cucumber per IntelliJ IDEA offrono comode funzionalità IDE per lavorare con i file di feature Gherkin in un progetto IntelliJ utilizzando il framework Cucumber. I plugin sono disponibili per le lingue Java, Scala o Groovy.

Osservazioni

Il plugin IntelliJ di Cucumber for Java offre funzionalità IDE per lo sviluppo pratico con Cucumber, incluso

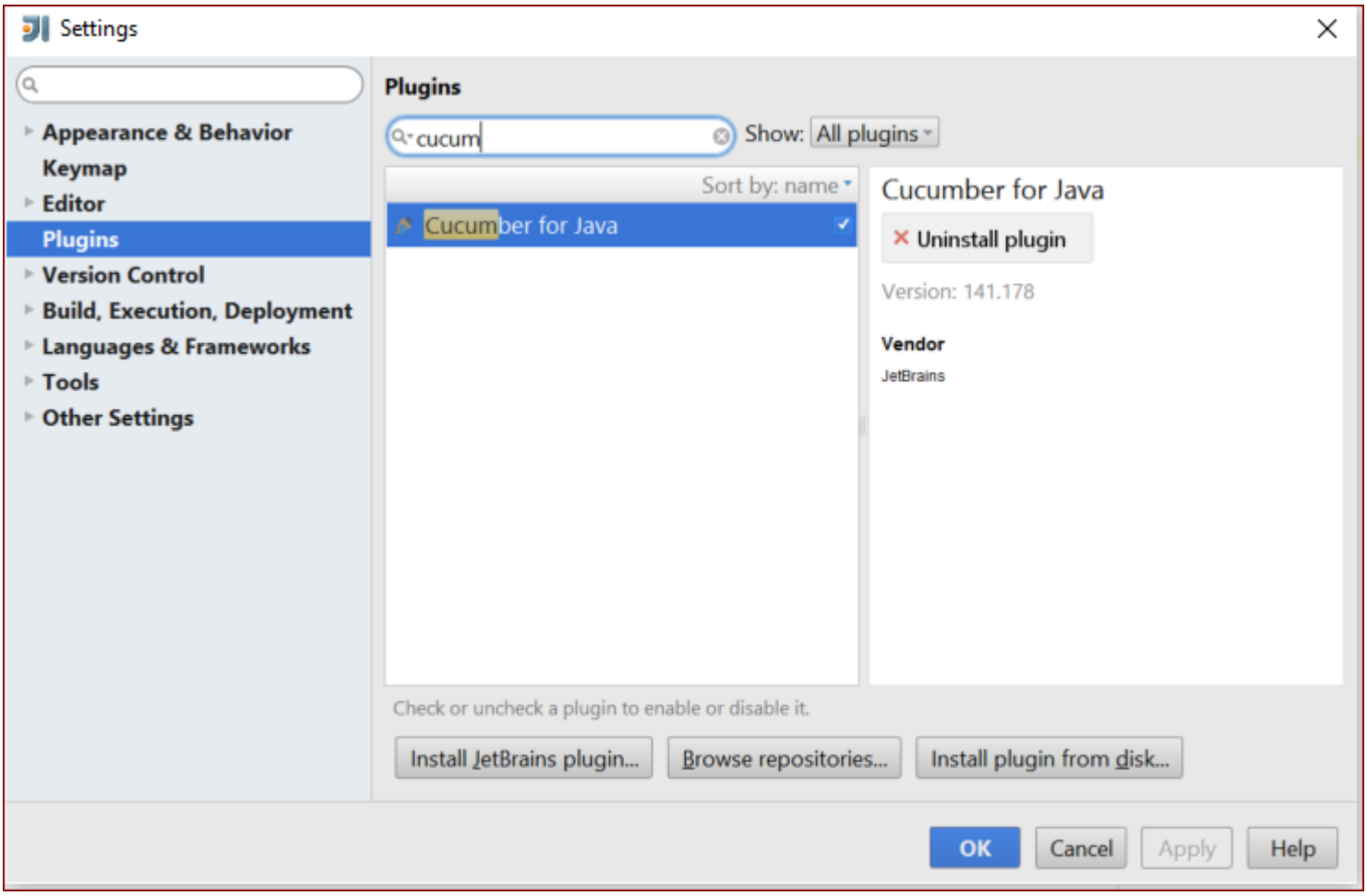
- Generazione di step step Gherkin per passaggi non implementati
- Completamento del codice passo Gherkin
- Salto del codice metodo step-to-glue
- Evidenziazione della sintassi Gherkin nei file ".feature" che corrispondono alla regex del passo

e altre funzioni utili.

Examples

Installa il plugin Cucumber

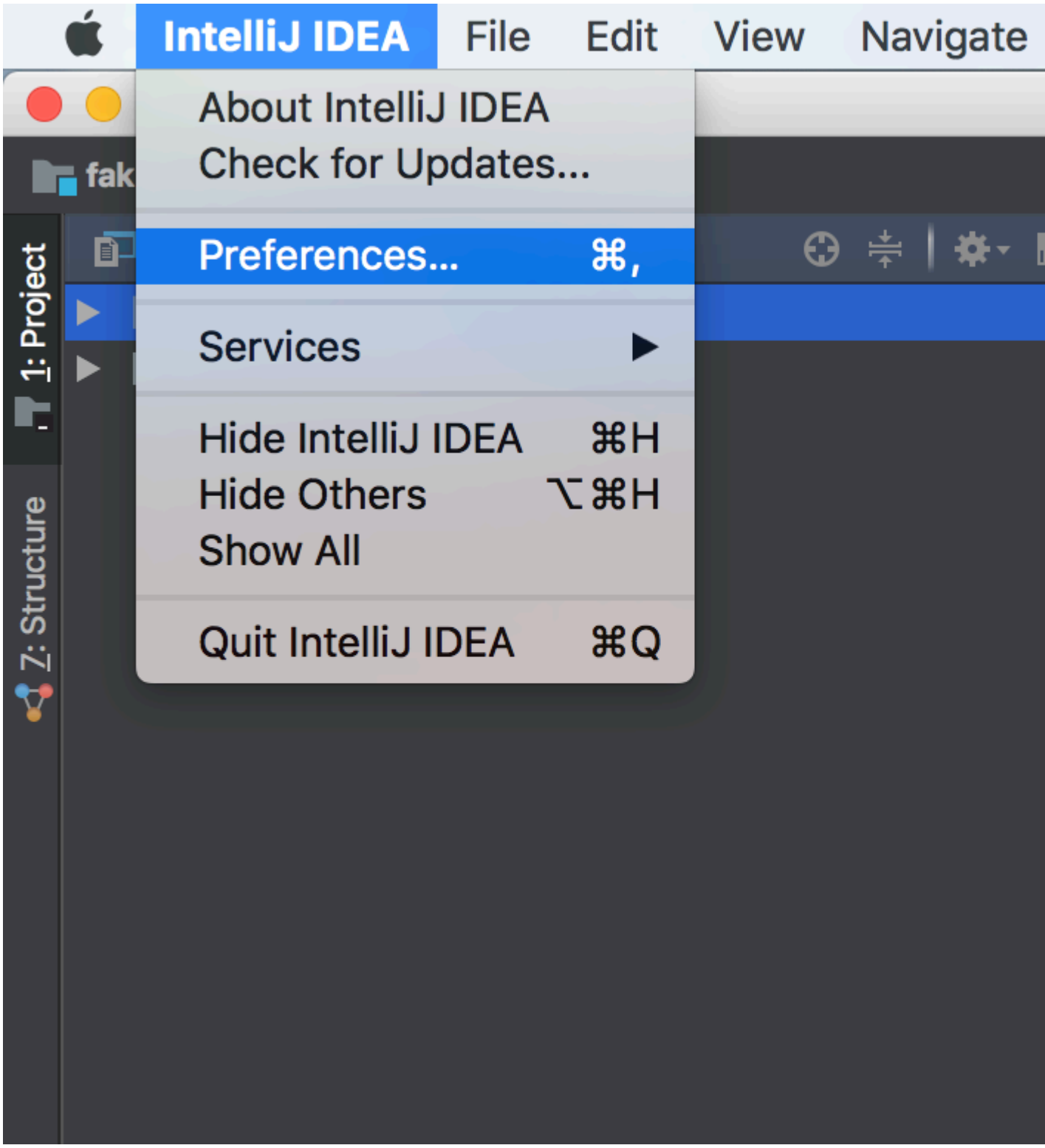
Vai su File -> Impostazioni -> fai clic su plug-in nel riquadro a sinistra -> Cerca cetriolo -> Installa plug-in



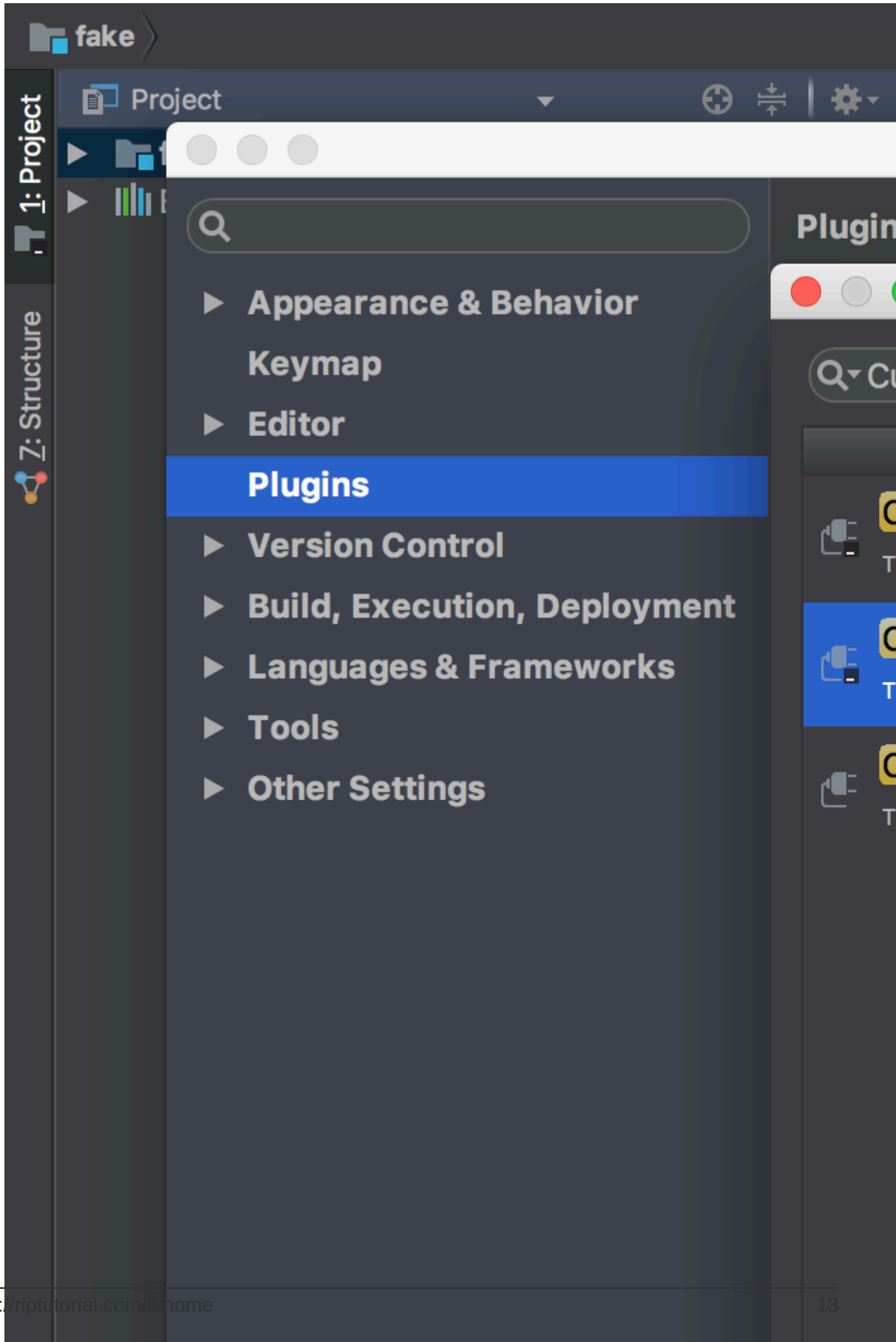
Installa IntelliJ Cucumber per Java Plugin (Mac)

Per installare il plugin Cucumber per Java per IntelliJ su un Mac,

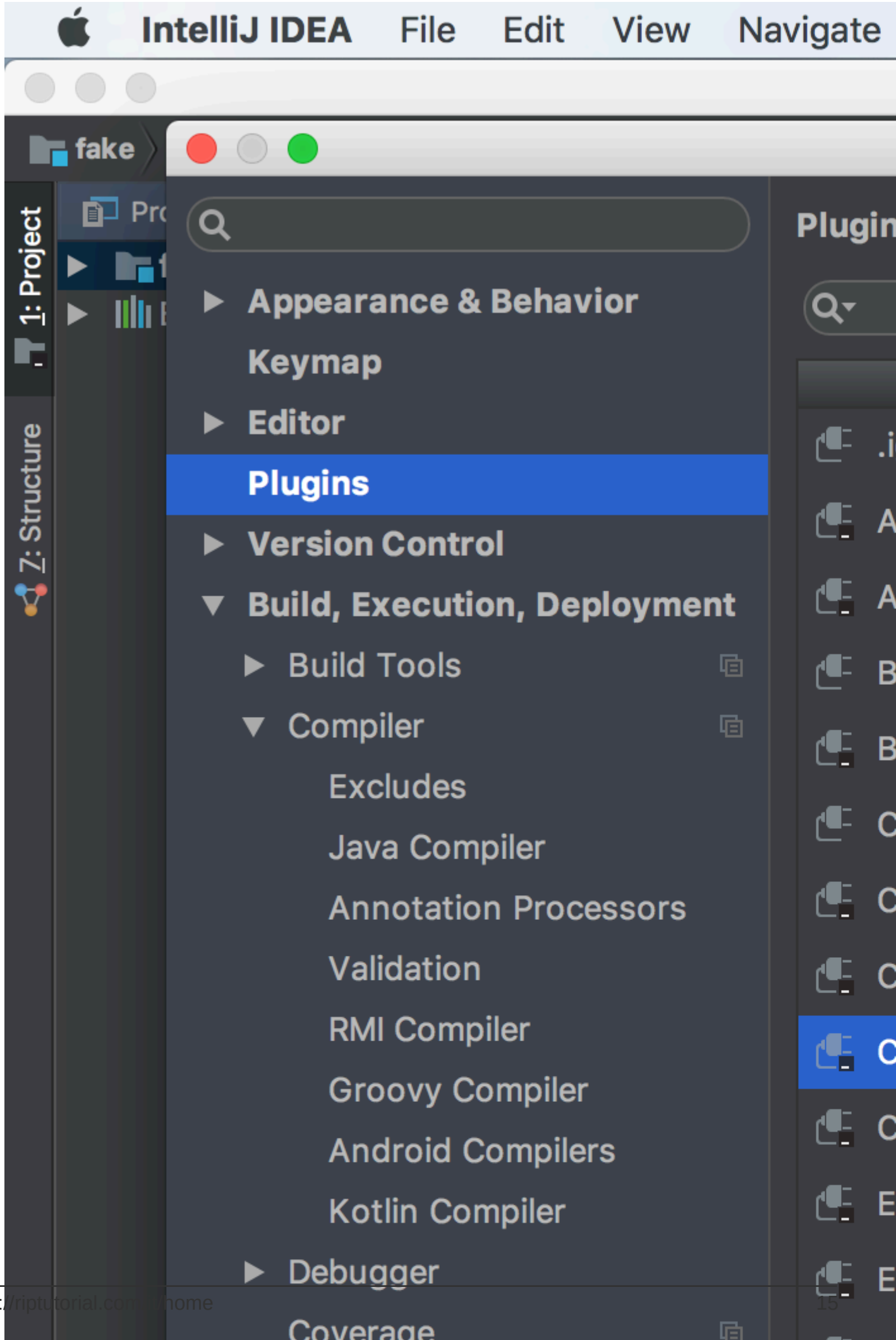
1. Avvia IntelliJ IDEA.
2. Fare clic sulla scheda "IntelliJ IDEA" nella barra superiore.



3. Clicca su "Preferenze".
4. In Preferenze / Impostazioni, fai clic su "Plug-in" nel riquadro a sinistra.
5. Fai clic sul pulsante "Sfoglia i repository", che apre una nuova finestra.
6. Cerca "Cetriolo" nella barra di ricerca.


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

7. Installa il plugin "Cucumber per Java".
8. Riavviare l'IDE affinché il plug-in abbia effetto. Il Cucumber per Java è ora installato.


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/it/cucumber/topic/8356/installa-il-plugin-cetriolo-in-intellij

Capitolo 5: pom.xml per il progetto Maven_ cetriolo.

introduzione

Il modello di oggetto del progetto sottostante è il modello pom.xml. Se si desidera creare un esperimento con progetto cetriolo, è possibile utilizzare l'esempio seguente come modello

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>
```

Leggi pom.xml per il progetto Maven_ cetriolo. online: https://riptutorial.com/it/cucumber/topic/8331/pom-xml-per-il-progetto-maven--cetriolo-

Capitolo 6: Sintassi di Gherkin

introduzione

Gherkin è un linguaggio leggibile dagli utenti per l'automazione dei test e la documentazione di test. È compreso da Cucumber e insieme esiste come uno strumento di sviluppo guidato dal comportamento.

Sintassi

- Funzionalità: questa parola chiave indica che quanto segue è una descrizione o un nome di base della funzionalità testata o documentata.
- Sfondo: questa parola chiave indica i passaggi che verranno eseguiti prima di ogni scenario nella funzione.
- Scenario: questa parola chiave rappresenta il nome o la descrizione di base di uno scenario particolare che verifica la funzione.
- Scenario Struttura: questa parola chiave significa che lo scenario verrà eseguito N volte per ogni argomento elencato negli esempi esplicitamente passati per nome di colonna racchiuso tra parentesi angolate.
- Esempi: questa parola chiave prende nota dell'elenco di argomenti statici che verranno passati in uno schema di scenario.
- Dato: questa parola chiave rappresenta un determinato passo, o presupposto che si presume prima di continuare. Nel paradigma Disponi, Agita, Assert, indicato rappresenta "Disponi".
- Quando: questa parola chiave rappresenta un passo quando, o il comportamento che deve essere dichiarato contro. Nel paradigma Disponi, Agisci, Assert, indicato rappresenta "Act".
- Quindi: questa parola chiave rappresenta un passaggio, o in altre parole, il passo in cui viene convalidato il risultato di un comportamento. Nel paradigma Disponi, Agisci, Assert, indicato rappresenta "Assert".
- E: questa parola chiave viene utilizzata insieme a una qualsiasi delle parole chiave sopra. Se hai due dichiarazioni date, invece di chiamare esplicitamente Dato due volte, puoi dire "Dato A e B".

Examples

Le basi

Questo esempio analizzerà la struttura di base di un file di feature Cucumber in Gherkin. I file di caratteristiche utilizzano più parole chiave nella sintassi di base.

Vediamo le parole chiave di base:

- **Funzionalità:** questa parola chiave indica che quanto segue è una descrizione o un nome di base della funzionalità testata o documentata.

- **Scenario:** questa parola chiave rappresenta il nome o la descrizione di base di uno scenario particolare che verifica la funzione.
- **Data** questa parola chiave rappresenta un determinato passo, o presupposto che si presume prima di continuare. Nel paradigma Disponi, Agita, Assert, indicato rappresenta "Disponi".
- **Quando** questa parola chiave rappresenta un passo quando, o il comportamento che deve essere affermato contro. Nel paradigma Disponi, Agisci, Assert, indicato rappresenta "Act".
- **Quindi** questa parola chiave rappresenta un passaggio, o in altre parole, il passo in cui viene convalidato il risultato di un comportamento. Nel paradigma Disponi, Agisci, Assert, indicato rappresenta "Assert".
- **E** questa parola chiave viene utilizzata insieme a una qualsiasi delle parole chiave sopra. Se hai due dichiarazioni date, invece di chiamare esplicitamente Dato due volte, puoi dire "Dato A e B".
- **Ma** questa parola chiave viene utilizzata insieme a **Given** , **When** e **Then** per indicare che qualcosa non dovrebbe accadere. Quindi A But not B.

Tutte le parole chiave devono essere su una nuova riga e devono essere la prima parola su una nuova riga per essere riconosciute dal parser Gherkin. Le parole chiave Feature e Scenario devono avere subito due punti, come nell'esempio seguente. Dato, quando, poi, e E non richiedono due punti.

Oltre alle parole chiave, puoi scrivere descrizioni e commenti. Le descrizioni si verificano dopo la parola chiave ma sulla stessa riga, dove i commenti si presentano sulle linee sotto le parole chiave. Quando si scrivono commenti di feature, è consuetudine fornire regole esplicite che definiscano i bordi e le condizioni che conducono a diversi esiti di comportamenti.

```
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
```

Passi parametrizzati

Durante la scrittura di Gherkin, ci possono essere momenti in cui si desidera parametrizzare i passaggi per la riusabilità da parte dell'ingegnere che sta implementando i piani di test. I passaggi

ricevono i parametri attraverso i gruppi di acquisizione di espressioni regolari. (**Nota di progettazione:** se non si dispone di parametri di corrispondenza per ciascun gruppo di acquisizione nell'espressione regolare, è possibile che venga generata una "CucumberException: mismatch di Arity") Nell'esempio seguente, abbiamo deciso di includere gli argomenti tra virgolette doppie come accettare gli interi come argomenti.

```
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
```

Feature Background

Come avrai notato nell'esempio sopra, stiamo riscrivendo lo stesso passo più volte:

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

Questo può essere eccezionalmente noioso, specialmente se abbiamo più di un dato passaggio che viene riutilizzato. Gherkin fornisce una soluzione per questo dandoci un'altra parola chiave con cui lavorare: **Background:.**

La parola chiave background serve per eseguire i passaggi dichiarati al di sotto di esso prima di

ogni scenario nella Feature. Assicurati di non aggiungere passaggi in background a meno che tu non sia positivo, è necessario per ogni scenario. Come le altre parole chiave, Sfondo è seguito da una descrizione o nome e può contenere commenti sotto di esso. Molto simile a Feature and Scenario, Background deve essere preceduto da due punti.

```
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
```

Scenario

In alcuni casi è possibile ripetere lo stesso scenario più e più volte, sostituendo gli argomenti. In questo caso, Gherkin fornisce diverse nuove parole chiave per soddisfare questa situazione, **Profilo Scenario:** ed **Esempio:.** La parola chiave Scenario Outline indica a Cucumber che lo scenario verrà eseguito più volte sostituendo gli argomenti da un elenco. La parola chiave Examples viene chiamata prima che l'elenco sia esplicitamente notificato. Gli argomenti per i contorni degli scenari devono essere racchiusi tra parentesi angolari. Nell'esempio seguente, si noti che i nomi degli argomenti racchiusi tra parentesi angolari corrispondono ai nomi delle colonne elencati in Esempi. Ogni colonna è separata da barre verticali, con i nomi delle colonne sulla prima riga.

```
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     |
```

tag

Ai fini della documentazione, è possibile filtrare i piani di test o gli scenari per categorie. Gli sviluppatori potrebbero voler eseguire test basati su quelle stesse categorie. Gherkin consente di classificare le funzionalità e i singoli scenari tramite l'utente dei tag. Nell'esempio seguente, si noti quanto sopra la parola chiave Feature è il tag "@Automation". Gherkin riconosce questo come un tag dall'utente del simbolo "@". In questo esempio, l'ingegnere vuole chiarire che questi test sono utilizzati per l'automazione, dove non tutti i test sono automatizzati, alcuni test devono essere eseguiti dal QA manuale.

Si noti inoltre che il tag @Production è stato aggiunto allo scenario testing user lock out. In questo esempio, questo è perché questo scenario è attivo solo nell'ambiente di produzione dell'applicazione. Gli sviluppatori non vogliono che i loro account sandbox vengano bloccati durante lo sviluppo. Questo tag consente loro di far rispettare questo test solo con l'ambiente di produzione.

Infine, lo Scenario Outline ha il tag @Staging. Ai fini di questo esempio, questo è dovuto al fatto che gli account utilizzati sono account di gestione temporanea e non funzioneranno negli altri ambienti. Come il tag @Production, ciò garantisce che questi test vengano eseguiti solo nell'ambiente di gestione temporanea.

Questi sono solo alcuni esempi di dove, come e perché potresti usare i tag. In definitiva questi tag avranno significato per te e per gli sviluppatori e possono essere qualsiasi cosa e utilizzati per categorizzare come ritieni opportuno.

```
@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     |
```

Punte di cetriolino

- Ogni scenario verifica un comportamento
- Gli scenari sono scritti in modo dichiarativo
- Evita dettagli accidentali all'interno dello scenario

- Ometti l'ovvio
- Evita i passaggi congiuntivi
- Mantieni i tuoi scenari brevi
- Non avere molti scenari nella stessa funzione
- Usa nomi di scenari descrittivi
- Ne hai solo uno Quando passo
- Utilizzare il "dovrebbe" in Quindi passaggi

Leggi Sintassi di Gherkin online: https://riptutorial.com/it/cucumber/topic/9296/sintassi-di-gherkin

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con il cetriolo	Community, Dave Schweisguth, Mo H., Roberto Lo Giacco, SirLenz0rlot, user3554664
2	Caratteristiche	Dave Schweisguth, Kyle Fairns, Priya
3	Definizioni passo	Dave Schweisguth
4	Installa il plugin cetriolo in Intellij	George Pantazes, Priya
5	pom.xml per il progetto Maven_ cetriolo.	user
6	Sintassi di Gherkin	jordiPons, tramstheman, user3554664