Uygulamalı Laboratuvar: İlk Adımlarınızı Uygulama

Gerekli tahmini süre: 45 dakika

İlk Adımlarınızı Uygulama laboratuvarına hoş geldiniz. Önceki laboratuvarda adım şablonlarını oluşturduğunuza göre, şimdi bunları uygulama zamanı.

Bu laboratuvar, `NotImplementedError` istisnalarını gerçek kodla değiştireceksiniz.

Öğrenme Hedefleri

Bu laboratuvarı tamamladıktan sonra şunları yapabileceksiniz:

- Bir testi uygulamak için adımları çağırmak
- Testlerin geçmesi için kod yazmak

Theia Hakkında

Theia, masaüstünde veya bulutta çalıştırılabilen açık kaynaklı bir IDE'dir (Entegre Geliştirme Ortamı). Bu laboratuvarı yapmak için Theia IDE'sini kullanacaksınız. Theia ortamına giriş yaptığınızda, yalnızca sizin için ayrılmış bir 'bulut üzerindeki bilgisayar' ile karşılaşırsınız. Bu, laboratuvarlar üzerinde çalıştığınız sürece sizin için mevcuttur. Çıkış yaptığınızda, bu 'bulut üzerindeki bilgisayar' ve oluşturmuş olabileceğiniz dosyalar silinir. Bu nedenle, laboratuvarlarınızı tek bir oturumda tamamlamak iyi bir fikirdir. Laboratuvarın bir kısmını tamamlayıp daha sonra Theia laboratuvarına döndüğünüzde, baştan başlamanız gerekebilir. Tüm Theia laboratuvarlarınızı tamamlamak için zamanınız olduğunda çalışmayı planlayın.

Laboratuvar Ortamını Kurma

Laboratuvara başlamadan önce biraz hazırlık yapmanız gerekiyor. Bir terminal açmanız ve bazı sistem bağımlılıklarını ve bazı Python bağımlılıklarını yüklemeniz gerekiyor.

Terminali Açın

Editördeki menüyü kullanarak bir terminal penceresi açın: Terminal > Yeni Terminal.

Terminalde, eğer `/home/projects` klasöründe değilseniz, şimdi proje klasörünüze geçin.

```
cd /home/project
```

Kodu Klonlayın

Şimdi test etmeniz gereken kodu alın. Bunu yapmak için, git deposunu klonlamak için `git clone` komutunu kullanın:

```
git clone https://github.com/ibm-developer-skills-network/duwjx-tdd_bdd_PracticeCode.git
```

Repo Klasörüne Geçin

Son olarak, laboratuvar dosyalarını içeren dizine geçmelisiniz:

```
cd /home/project/duwjx-tdd_bdd_PracticeCode
```

Laboratuvar Bağımlılıklarını Kurun

Depoyu klonladıktan sonra, geliştirme ortamına bazı ön koşul yazılımlarını kurmanız gerekiyor:

```
bash ./bin/setup.sh
```

Laboratuvar Klasörüne Geçin

Sonraki adımda, laboratuvar dosyalarının bulunduğu dizine geçmelisiniz:

```
cd /home/project/duwjx-tdd_bdd_PracticeCode/labs/12_implementing_steps
```

Python Bağımlılıklarını Yükleyin

Son hazırlık adımı, laboratuvar için gerekli Python paketlerini yüklemek üzere `pip` kullanmaktır:

```
pip install -r requirements.txt
```

Artık laboratuvara başlamak için hazırsınız.

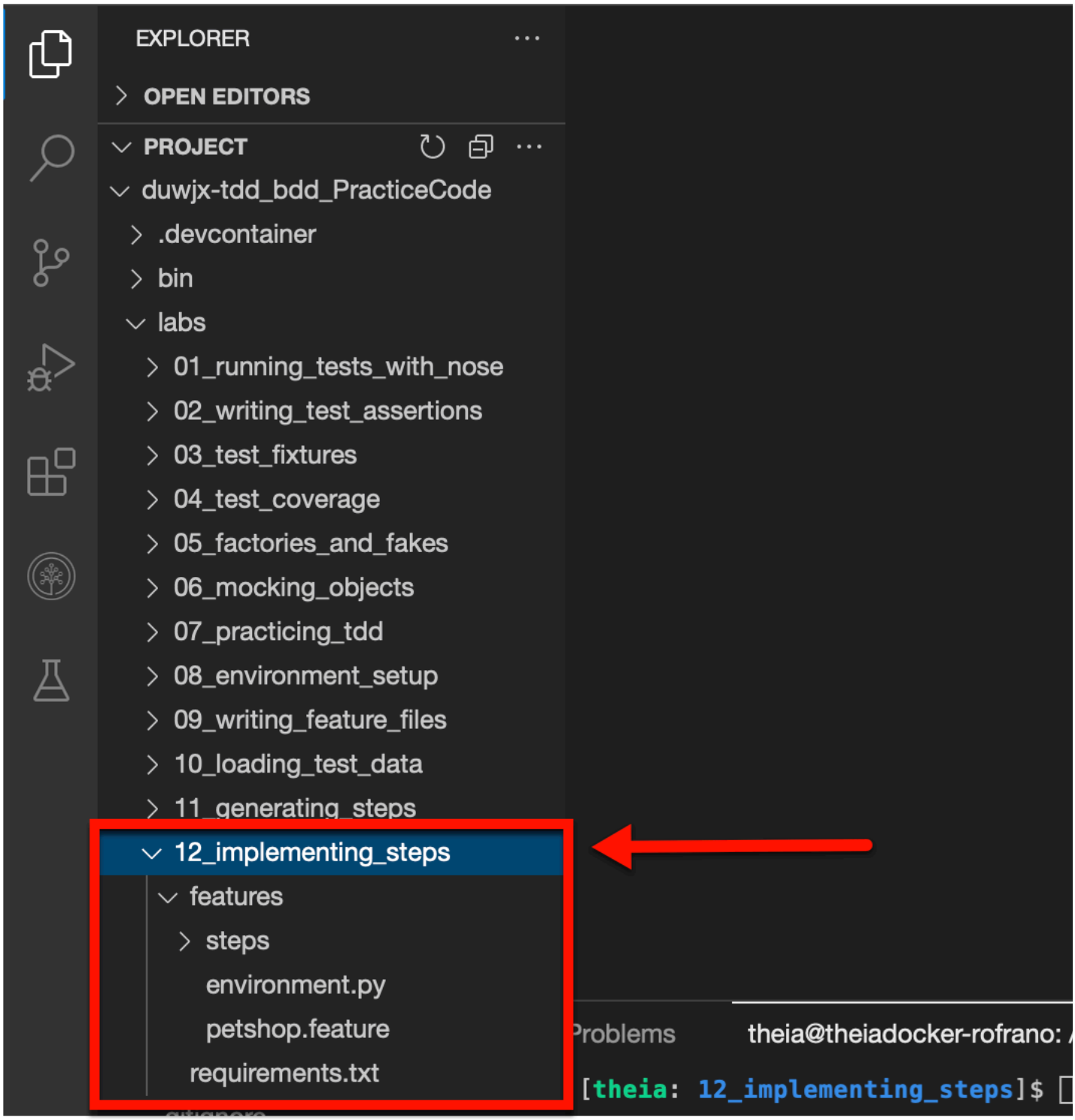
İsteğe Bağlı

Eğer terminalde çalışmak, komut isteminin uzunluğu nedeniyle zorlaşırsa, aşağıdaki komutu kullanarak istemi kısaltabilirsiniz:

```
export PS1="[\[\033[01;32m\]\u\[\033[00m\]: \[\033[01;34m\]\W\[\033[00m\]]\$ "
```

Koda Git

IDE'de `duwjx-tdd_bdd_PracticeCode/labs/12_implementing_steps` klasörüne gidin. Bu klasör, bu laboratuvar için kullanacağınız tüm kaynak kodunu içerir.



Kullanıcı Arayüzü

Aşağıda bu laboratuvar ortamında Selenium ve Behave ile test edeceğiniz Kullanıcı Arayüzü bulunmaktadır:

Pet Demo REST API Service

Status: Success

Create, Retrieve, Update, and Delete a Pet:

Pet ID: 1

Name: Fido

Category: dog id=pet_category

Available: True

Gender: Male

Birthday: 11/18/2019

Search Clear Create Update

id=search-btn

ID	Name	Category	Ava
1	Fido	dog	true
2	Kitty	cat	true

id=search_

Resim, aşağıdakileri göstermek için not edilmiştir:

- Test ettiğiniz web sitesi, manipüle edilmesi gereken her HTML elemanının `id` niteliği için tutarlı bir adlandırma kuralı kullanmaktadır. Hepsi `pet_` öneki ile başlayıp elemanın küçük harfle yazılmış adıyla devam eder. Örneğin, Kategori elemanının `id`si `pet_category`dir.
- Test edilen web sitesi, butonların `id=` niteliğini oluşturmak için aşağıdaki adlandırma kuralını kullanmaktadır. Butondaki küçük harfli metni alır ve sonuna `-btn` ekler. Örneğin, **Arama** butonunun `id`si `search-btn` ve **Oluştur** butonunun `id`si `create-btn`dir vb.
- İki özel `id` bulunmaktadır: `flash_message` ve `search_results`. `flash_message` alanı, gerçekleştirdiğiniz eylemlerden dönen mesajları tutar. `search_results` alanı, **Arama** butonuna tıkladığınızda dönen veri tablosunu içerir.

Selenium Değişiklikleri

* * * Önemli Duyuru * * *

Video derslerinin oluşturulmasından bu yana, Python Selenium paketi birçok yeni sürüm ve `3.141.0`'dan `4.19.0`'a büyük bir güncelleme geçirdi. Bu süre zarfında, video dersinde öğretilen API `find_element_by_id(element_id)` kullanım dışı bırakıldı ve artık kaldırıldı. Eşdeğer API şimdi: `find_element(By.ID, element_id)`. Bu laboratuvar talimatları ve örnek kod yeni sözdizimine göre değiştirildi.

Bu laboratuvar için kullanmanız gereken kod değişikliği, videodan farklıdır:

Eski Kod

```
context.driver.find_element_by_id(element_id)
```

Yeni Kod

Selenium `By` sınıfını şu şekilde içe aktarmalısınız:

```
from selenium.webdriver.common.by import By
```

O zaman bir öğeyi id ile bulmanız gerektiğinde, `find_element()` çağırmalı ve ilk parametre olarak `By.ID` geçmelisiniz, şöyle:

```
context.driver.find_element(By.ID, element_id)
```

Bu laboratuvardaki ipuçları ve öneriler, yeni kod sözdizimini hatırlatacaktır. Laboratuvardaki kodun artık videoda sunulan kodla eşleşmediğini bilmenizi istiyoruz, ancak kavramlar %100 aynı, sadece bir sözdizimi değişikliği oldu.

Başla: Behave'i Çalıştır

Son laboratuvarında, uygulanması gereken adımlar için işlev şablonları oluşturdun. Bu laboratuvarında, uygulamaları ekleyeceksin.

`behave` komutunu çalıştırarak hangi adımın başarısız olduğunu görelim, böylece onu uygulayabiliriz.

Görevin

1. `Terminal -> Yeni Terminal` seçeneğini kullanarak bir bash kabuğu aç ve `behave` komutunu çalıştır:

```
behave
```

```
Feature: Search for pets by category # features/petshop.featu
  As a pet shop customer
  I need to be able to search for a pet by category
  So that I only see the category of the pet I am interested
  Background:   # features/petshop.feature:7

  Scenario: Search for dogs                          # features/pe
    Given the following pets                         # features/st
      | name  | category | available | gender | birthday   |
      | Fido  | dog      | True      | MALE   | 2019-11-18 |
      | Kitty | cat      | True      | FEMALE | 2020-08-13 |
      | Leo   | lion     | False     | MALE   | 2021-04-01 |
    Given I am on the "Home Page"                    # features/st
      Traceback (most recent call last):
        File "/home/theia/venv/lib/python3.9/site-packages/be
          match.run(runner.context)
        File "/home/theia/venv/lib/python3.9/site-packages/be
          self.func(context, *args, **kwargs)
        File "features/steps/web_steps.py", line 14, in step_
          raise NotImplementedError(u'STEP: Given I am on the
      NotImplementedError: STEP: Given I am on the "Home Page
    When I set the "Category" to "dog"               # None
    And I click the "Search" button                  # None
    Then I should see the message "Success"          # None
    And I should see "Fido" in the results           # None
    But I should not see "Kitty" in the results      # None
    And I should not see "Leo" in the results        # None


Failing scenarios:
  features/petshop.feature:14  Search for dogs

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
1 step passed, 1 failed, 6 skipped, 0 undefined
Took 0m0.049s
```

Gördüğünüz gibi, başarısız olan adım **Given I am on the "Home Page"** olup, bu `features/steps/web_steps.py` dosyasının **12**. satırında bulunmaktadır.

Bunu şimdi uygulayalım.

Adım 1: Ana Sayfada Olduğumuzu Belirtme

`features/steps/web_steps.py` dosyasını IDE editöründe açın. Laboratuvarın geri kalanında bu dosyada çalışacaksınız.

İlk görev, 'Ana Sayfada Olduğumu Belirt' adımını uygulamaktır. Adım şu anda aşağıda gördüğünüz gibi sadece bir fonksiyon taslağıdır:

```
@given('I am on the "Home Page"')
def step_impl(context):
    raise NotImplementedError(u'STEP: Given I am on the "Home Page"')
```

Bu adım, senaryoya başlangıç noktası olarak sizi ana sayfada yerleştirmek için servisin kök URL'sini (`/`) çağırmalıdır.

Arka Plan

Çözümü uygulamanıza yardımcı olacak bazı önemli arka plan bilgileri:

- `environment.py` dosyasında, test edilen servisin temel URL'sini içeren `context.base_url` adında bir değişken ayarladığımızı unutmayın.
- Ayrıca, `context.driver.get()` çağrısıyla web sürücüsünde bir HTTP `GET` isteği yapabileceğinizi bilmek önemlidir.

  Örnek:

  ```
  context.driver.get("http://www.website.com")
  ```

Bu bilgileri, adımı uygulamanıza yardımcı olması için kullanın.

Göreviniz

Aşağıdakileri uygulayan kodu yazın:

1. `context.driver.get()` kullanarak `context.base_url`'yi argüman olarak geçin ve yanıtı `context.response`'de saklayarak mevcut kod satırını değiştirin.

▼ İpucu için buraya tıklayın.

```
{the response} = context.driver.get(...place the base url here...)
```

2. Çalışmanızı kaydedin

Çözüm

Devam etmeden önce kodunuzun aşağıdaki çözümle eşleştiğinden emin olun.

▼ Çözüm için buraya tıklayın.

Bu, ilk adımı eklemek için çözümdür.

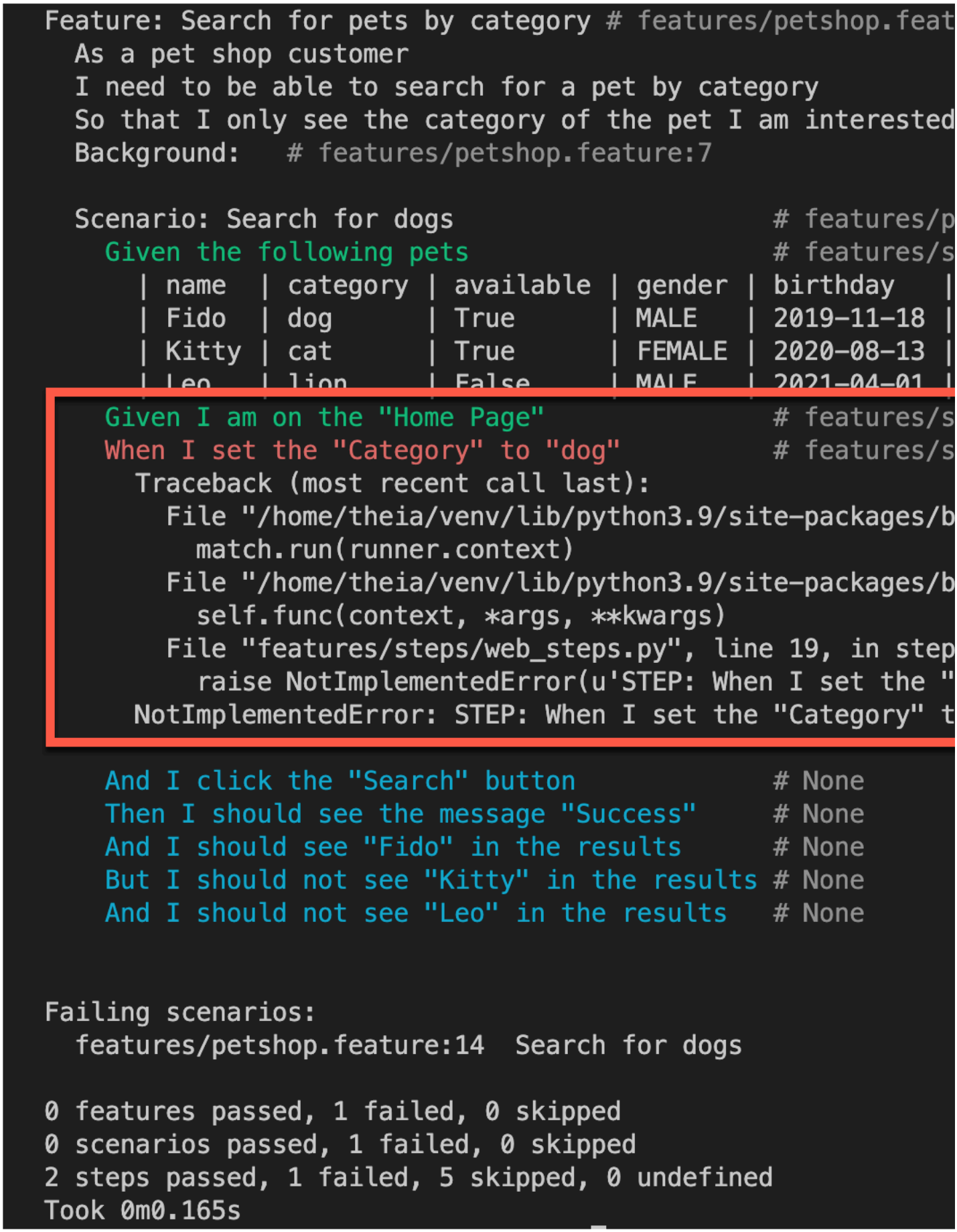
```
@given('I am on the "Home Page"')
def step_impl(context):
    context.response = context.driver.get(context.base_url)
```

Sonuçlar

`behave` komutunu çalıştırın.

```
behave
```

“Ana Sayfa”da olduğumda artık yeşil ve **“Kategori”yi “köpek” olarak ayarladığımda** artık kırmızı olmalı.



Şimdi gidip bir sonraki kırmızı adımı uygulayalım.

Adım 2: Kategoriyi köpek olarak ayarladığımda

Bir sonraki kırmızı adım **Kategoriyi "köpek" olarak ayarladığımda**. `web_steps.py` dosyasına baktığınızda şu anda şu şekilde göründüğünü göreceksiniz:

```
@when('I set the "Category" to "dog"')
def step_impl(context):
```

```
    raise NotImplementedError(u'STEP: When I set the "Category" to "dog"')
```

Bir sonraki adım, `pet_category` adlı elementi almak, temizlemek ve alana “dog” kelimesini yazmaktır.

Arka Plan

Çözümü uygulamanıza yardımcı olacak bazı önemli arka plan bilgileri:

- Test ettiğiniz web sitesi, manipüle edilmesi gereken her HTML elementinin `id` niteliği için tutarlı bir adlandırma kuralı kullanmaktadır. Hepsi `pet_` öneki ile başlayıp, ardından elementin küçük harfle yazılmış adı gelir. Örneğin, Kategori elementi `pet_category` `id`‘sine sahiptir.
- Selenium ile bir web sayfasındaki elementleri bulmanın bir yolu, onları `id=` niteliği ile adreslemektir. Bunu yapmak için Selenium çağrısı: `find_element(By.ID)`

```
element = context.driver.find_element(By.ID, '...element adı buraya...')
```

- Bir web elementinden herhangi bir veriyi temizlemek için, o element üzerinde `clear()` fonksiyonunu kullanabilirsiniz.

```
element.clear()
```

- Bir giriş alanı elementine veri yazmak için, o element üzerinde `send_keys()` fonksiyonunu kullanabilirsiniz.

```
element.send_keys('...metin dizisi buraya...')
```

Bu bilgileri kullanarak adımı uygulamanıza yardımcı olun.

Göreviniz

Aşağıdakileri uygulayan kodu yazın:

1. `context.driver.find_element(By.ID)` çağrısını yapın, `'pet_category'` değerini geçirin ve sonuçları `element` adlı bir değişkende saklayın.

   ▼ İpucu için buraya tıklayın.

   ```
   element = context.driver.find_element(By.ID, ... element id buraya ...)
   ```

2. Önceki satırdan `element` kullanarak, içindeki verileri kaldırmak için `clear()` fonksiyonunu çağırın.

   ▼ İpucu için buraya tıklayın.

   ```
   element.{function_call_name_here}()
   ```

3. Yine önceki satırdan aynı `element` kullanarak, `send_keys()` fonksiyonunu çağırın ve “dog” dizisini geçin.

   ▼ İpucu için buraya tıklayın.

   ```
   element.send_keys(...gönderilecek diziyi buraya...)
   ```

4. Çalışmanızı kaydedin.

Çözüm

Kodunuzun aşağıdaki çözümle eşleştiğinden emin olun.

▼ Çözüm için buraya tıklayın.

Bu, Kategori alanına “dog” yazmak için çözümdür.

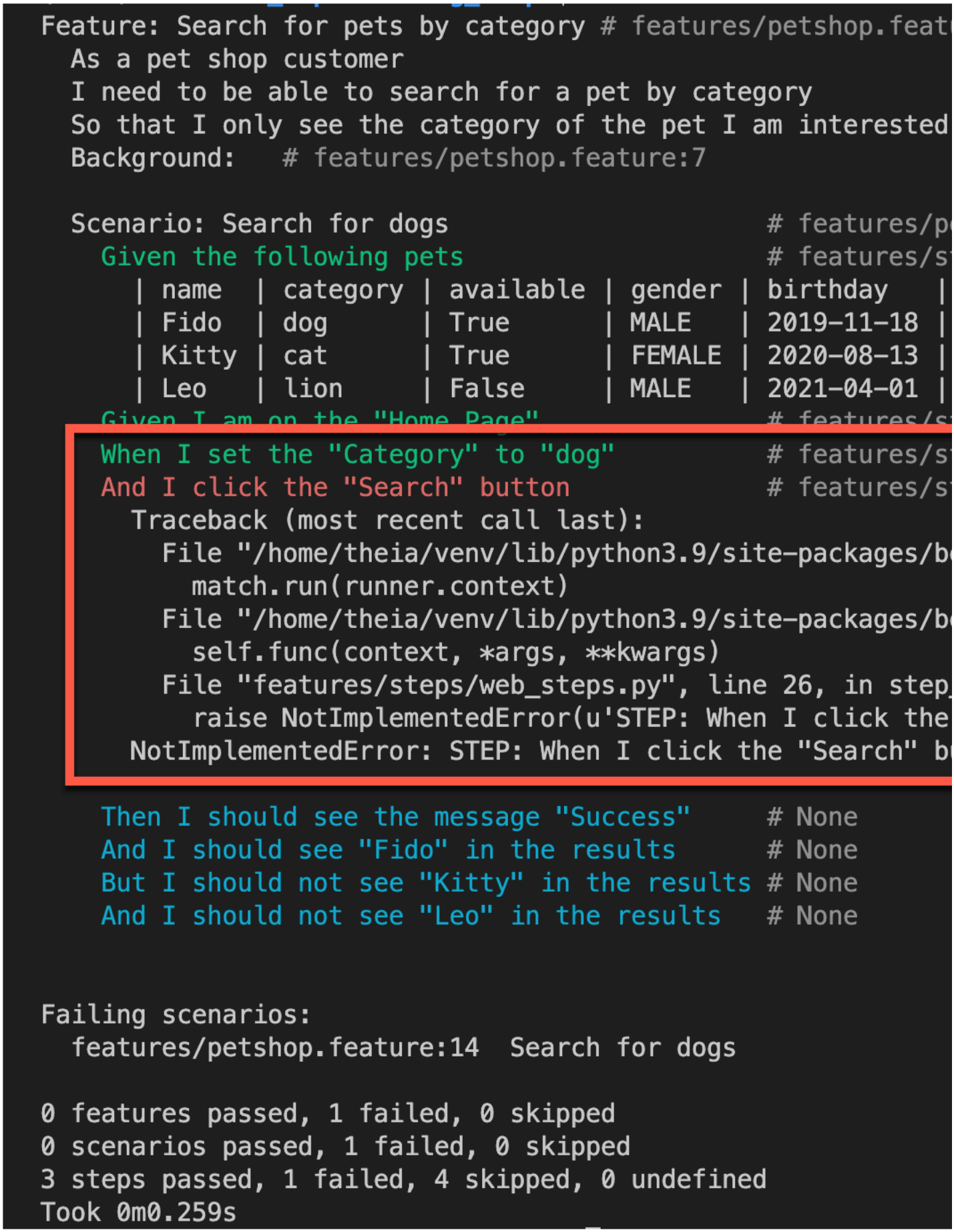
```
@when('I set the "Category" to "dog"')
def step_impl(context):
    element = context.driver.find_element(By.ID, 'pet_category')
    element.clear()
    element.send_keys('dog')
```

Sonuçlar

`behave` komutunu çalıştırın.

```
behave
```

Görmelisiniz ki **“Kategori”yi “köpek” olarak ayarladığımda** artık yeşil ve **“Ara” butonuna tıkladığımda** artık kırmızı.



Şimdi bir sonraki kırmızı adımı uygulamaya geçelim.

Adım 3: Ve Arama butonuna tıklıyorum

Bir sonraki kırmızı adım **Ve Arama butonuna tıklıyorum**. `web_steps.py` dosyasına bakarsanız, şu anda şöyle göründüğünü göreceksiniz:

```
@when('I click the "Search" button')
def step_impl(context):
```

```
    raise NotImplementedError(u'STEP: When I click the "Search" button')
```

Dikkat edin ki dekoratör `@when()` ve değil `@and()`. Bunun nedeni, **And** ve **But** anahtar kelimelerinin önceki **Given**, **When** veya **Then** ifadesinin anlamını almasıdır. Önceki ifade bir **When** olduğu için `@when()` dekoratörünü kullanıyoruz.

Arka Plan

Çözümü uygulamanıza yardımcı olacak bazı önemli arka plan bilgileri:

- Test edilen web sitesi, butonlar için `id=` niteliğini oluştururken aşağıdaki adlandırma kuralını kullanır. Butonun küçük harflerle yazılmış metnini alır ve sonuna `-btn` ekler. Örneğin, Arama butonunun `id`'si `search-btn`'dir.
- Daha önce Selenium ile gördüğünüz gibi, bir web sayfasındaki öğeleri bulmanın bir yolu, onları `id=` niteliği ile adreslemektir. Bunu yapmak için Selenium çağrısı: `find_element(By.ID, )`

```
    element = context.driver.find_element(By.ID, '...element adı buraya...')
```

- Bir buton veya hiper metin bağlantısı olan bir web öğesini tıklamak için, o öğe üzerinde `click()` fonksiyonunu kullanabilirsiniz.

```
    element.click()
```

Bu bilgileri kullanarak adımı uygulamanıza yardımcı olun.

Göreviniz

Aşağıdakileri uygulayan bir kod yazın:

1. `context.driver.find_element(By.ID)` çağrısını `'search-btn'` ile yapın ve sonucu `element` adlı bir değişkende saklayın.

   ▼ İpucu için buraya tıklayın.

   ```
       element = context.driver.find_element(By.ID, ... öğe id'si buraya ...)
   ```

2. Önceki satırdaki `element`i kullanarak, butona tıklamayı simüle etmek için `click()` fonksiyonunu çağırın.

   ▼ İpucu için buraya tıklayın.

   ```
       element.{fonksiyon_çağrısı_adı_buraya}()
   ```

3. Çalışmanızı kaydedin.

Çözüm

Kodunuzun çözümle eşleştiğinden emin olun.

▼ Çözüm için buraya tıklayın.

Bu, “Arama” butonuna tıklamak için çözümdür:

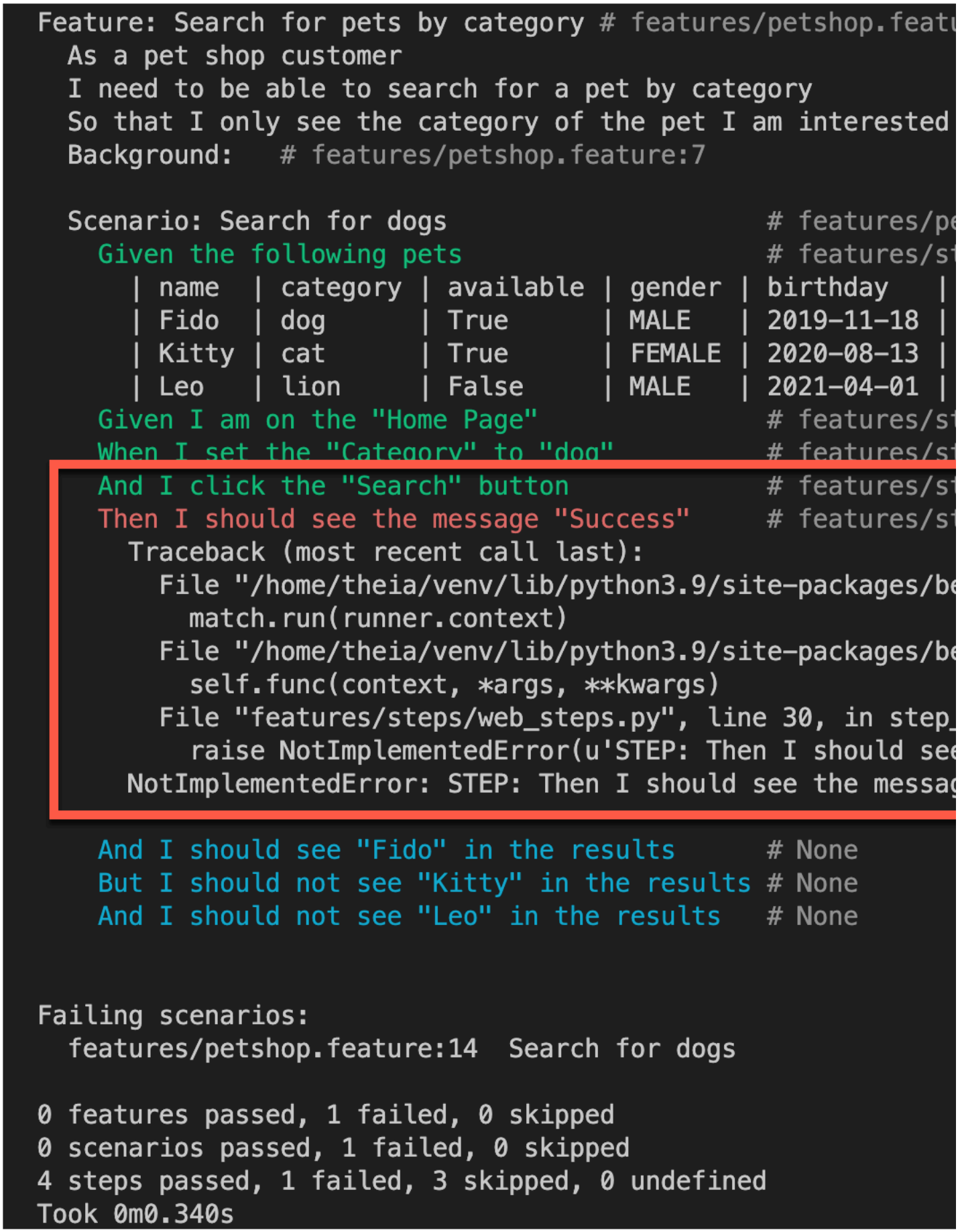
```
@when('I click the "Search" button')
def step_impl(context):
    element = context.driver.find_element(By.ID, 'search-btn')
    element.click()
```

Sonuçlar

`behave` komutunu çalıştırın

```
behave
```

Görmelisiniz ki **“Arama” butonuna tıkladım** şimdi yeşil ve **“Başarı” mesajını görmeliyim** şimdi kırmızı.



Şimdi bir sonraki kırmızı adımı uygulamaya gidelim.

Adım 4: Başarı mesajını görmeliyim

Kırmızı olan bir sonraki adım **O zaman "Başarı" mesajını görmeliyim**. `web_steps.py` dosyasına bakarsanız, şu anda şöyle göründüğünü göreceksiniz:

```
@then('I should see the message "Success"')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should see the message "Success"')
```

Arka Plan

İşte çözümü uygulamanıza yardımcı olacak önemli arka plan bilgileri:

- Test edilen web sitesi, tüm mesajlarını `flash_message` `id`'sine sahip bir elemana yazar.
- Daha önce Selenium ile gördüğünüz gibi, bir web sayfasındaki öğeleri bulmanın bir yolu, onları `id=` niteliğiyle adreslemektir. Bunu yapmak için Selenium çağrısı: `find_element(By.ID)`

```
element = context.driver.find_element(By.ID, '...öğe adı buraya...')
```

- Mesaj öğesinin metnini `text` niteliğini kontrol ederek okuyabilirsiniz.

```
element.text
```

Bu bilgileri, adımı uygulamanıza yardımcı olmak için kullanın.

Göreviniz

Aşağıdakileri uygulayan bir kod yazın:

1. `context.driver.find_element(By.ID)` çağrısını yapın, `'flash_message'` değerini geçirin ve sonuçları web öğesini temsil etmek için `element` adlı bir değişkende saklayın.

   ▼ İpucu için buraya tıklayın.

   ```
   element = context.driver.find_element(By.ID, ... öğe id'si buraya ...)
   ```

2. Önceki satırdaki `element` kullanarak, `text` niteliğinin `"Success"` kelimesini içerdiğini `assert` edin.

   ▼ İpucu için buraya tıklayın.

   ```
   assert {bir değer} in {öğenin metni}
   ```

3. Çalışmanızı kaydedin

Çözüm

▼ Çözüm için buraya tıklayın.

Bu, "Success" mesajını görmek için çözümdür:

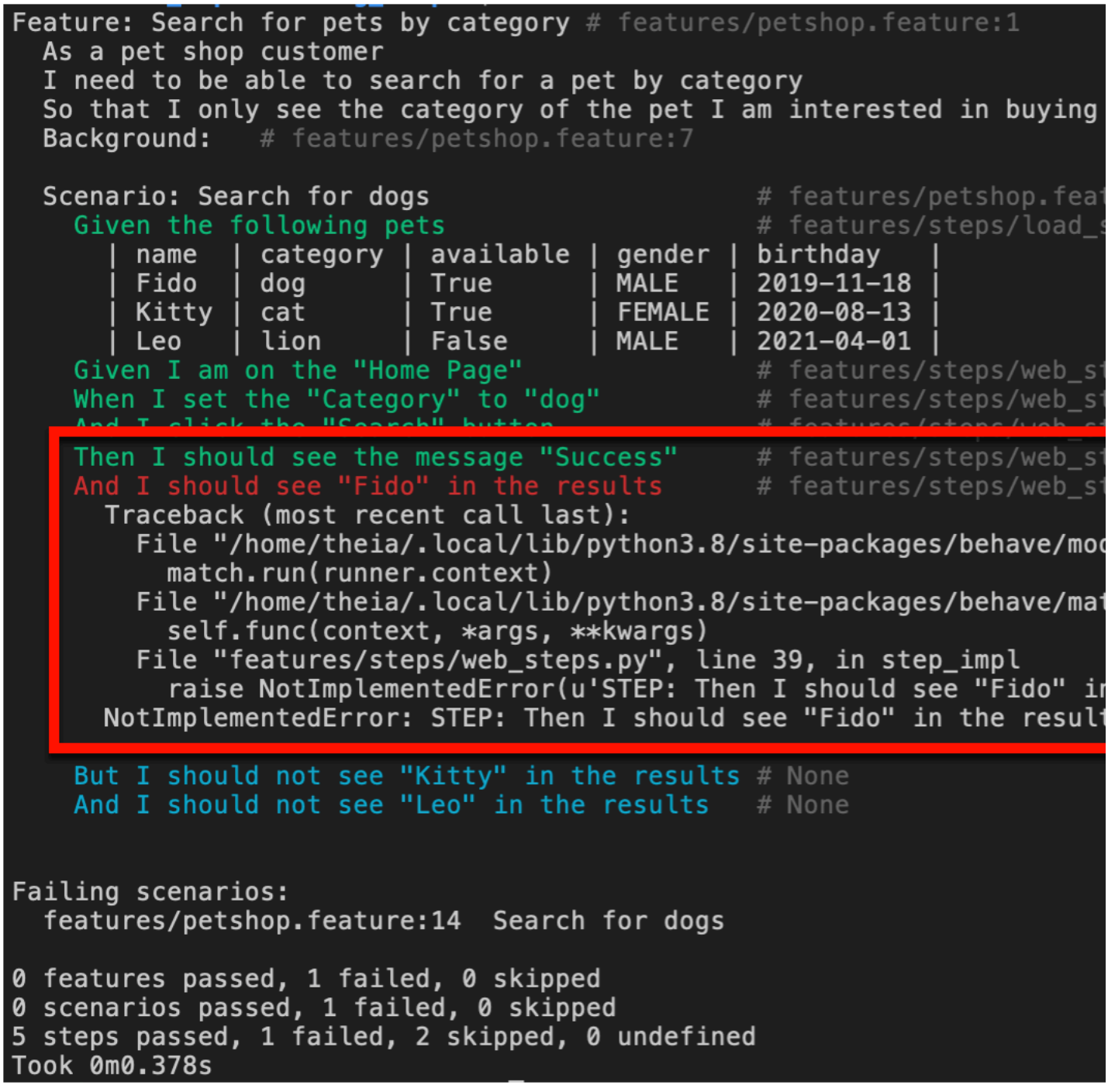
```
@then('I should see the message "Success"')
def step_impl(context):
    element = context.driver.find_element(By.ID, 'flash_message')
    assert "Success" in element.text
```

Sonuçlar

behave komutunu çalıştırın

```
behave
```

Görmelisiniz ki **O zaman "Başarı" mesajını yeşil olarak görmeliyim** şimdi yeşil ve **O zaman sonuçlarda "Fido"yu görmeliyim** şimdi kırmızı.



Şimdi bir sonraki kırmızı adımı uygulamaya geçelim.

Adım 5: Ve sonuçlarda Fido'yu görmeliyim

Kırmızı olan bir sonraki adım **Ve sonuçlarda "Fido"yu görmeliyim**. `web_steps.py` dosyasına bakarsanız, şu anda şöyle göründüğünü göreceksiniz:

```
@then('I should see "Fido" in the results')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should see "Fido" in the results')
```

Arka Plan

İşte çözümü uygulamanıza yardımcı olacak bazı önemli arka plan bilgileri:

- Test edilen web sitesi, tüm arama sonuçlarını `search_results` `id`'sine sahip bir elemana yazar.
- Daha önce Selenium ile gördüğünüz gibi, bir web sayfasındaki elemanları bulmanın bir yolu, onları `id=` niteliği ile adreslemektir. Bunu yapmak için Selenium çağrısı: `find_element(By.ID)`

  ```
  element = context.driver.find_element(By.ID, '...eleman adı buraya...')
  ```

- Arama sonuçları elemanının metnini okumak için `text` niteliğini kontrol edebilirsiniz.

  ```
  element.text
  ```

Bu bilgileri kullanarak adımı uygulamanıza yardımcı olun.

Göreviniz

Aşağıdakileri uygulayan bir kod yazın:

1. `context.driver.find_element(By.ID)` çağrısını yapın, `'search_results'` değerini geçin ve web elemanını temsil etmek için sonuçları `element` adında bir değişkende saklayın.

   ▼ İpucu için buraya tıklayın.

   ```
   element = context.driver.find_element(By.ID, ... eleman id'si buraya ...)
   ```

2. Önceki satırdaki `element` kullanarak, `text` niteliğinin `"Fido"` kelimesini içerdiğini `assert` edin.

   ▼ İpucu için buraya tıklayın.

   ```
   assert {bir değer} in {elemanın metni}
   ```

3. Çalışmanızı kaydedin

Çözüm

▼ Çözüm için buraya tıklayın.

Bu, arama sonuçlarında “Fido” görmenin çözümüdür:

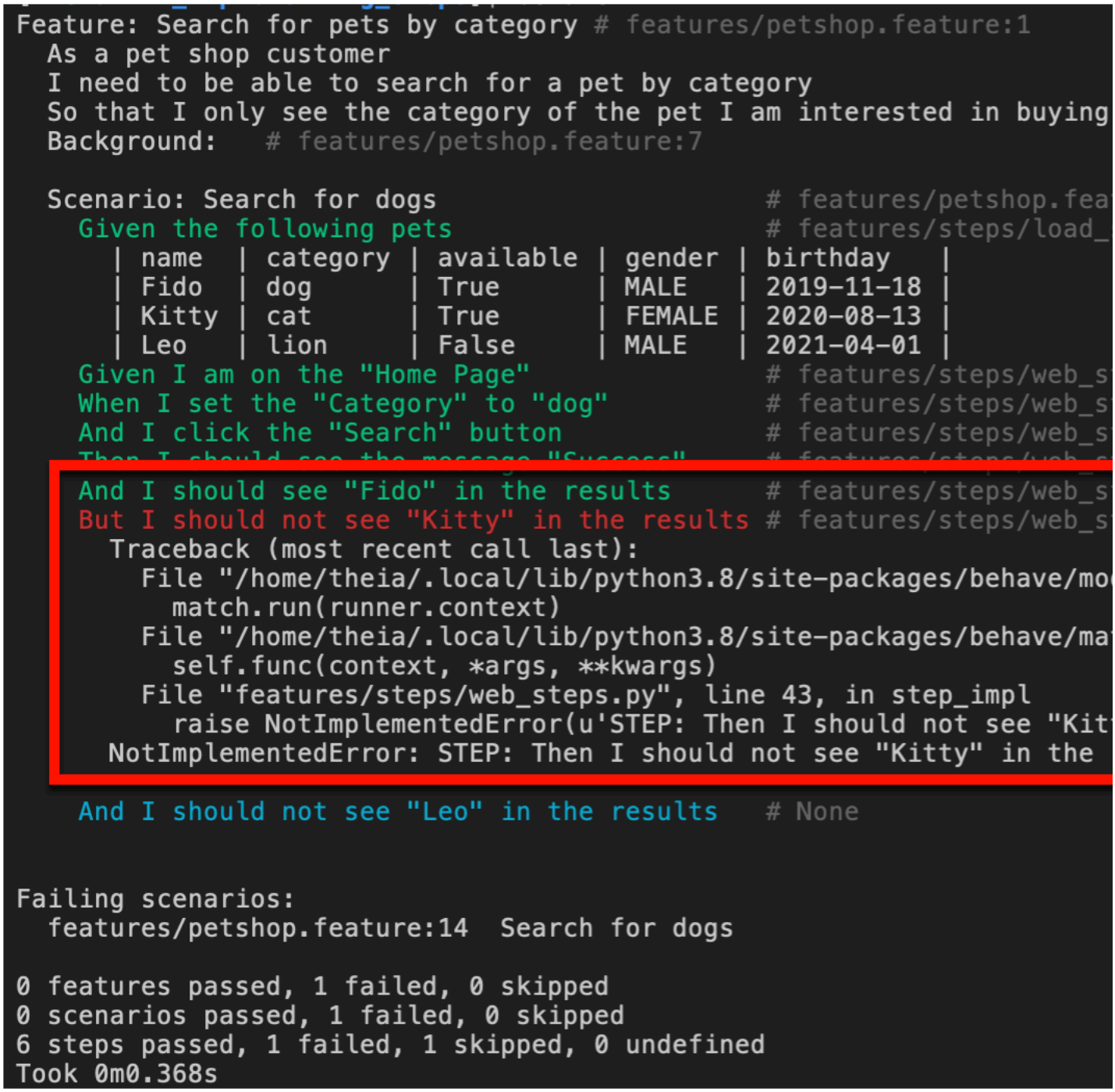
```
@then('I should see "Fido" in the results')
def step_impl(context):
    element = context.driver.find_element(By.ID, 'search_results')
    assert "Fido" in element.text
```

Sonuçlar

`behave` komutunu çalıştırın.

```
behave
```

Görmelisiniz ki **O zaman sonuçlarda “Fido”yu görmeliyim** şimdi yeşil ve **Ama sonuçlarda “Kitty”yi görmemeliyim** şimdi kırmızı.



Şimdi bir sonraki kırmızı adımı uygulayalım.

Adım 6: Ama sonuçlarda Kitty'yi görmemeliyim

Bir sonraki kırmızı adım **Ama sonuçlarda 'Kitty'yi görmemeliyim**. `web_steps.py` dosyasına bakarsanız, şu anda şöyle göründüğünü göreceksiniz:

```
@then('I should not see "Kitty" in the results')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should not see "Kitty" in the results')
```

Arka Plan

İşte çözümü uygulamanıza yardımcı olacak bazı önemli arka plan bilgileri:

- Test edilen web sitesi, tüm arama sonuçlarını `search_results id`'sine sahip bir elemana yazar.
- Daha önce Selenium ile gördüğünüz gibi, bir web sayfasındaki öğeleri bulmanın bir yolu, onları `id=` niteliği ile adreslemektir. Bunu yapmak için Selenium çağrısı: `find_element(By.ID)`

```
element = context.driver.find_element(By.ID, '...eleman adı buraya...')
```

- Arama sonuçları öğesinin metnini okumak için `text` niteliğini kontrol edebilirsiniz.

```
element.text
```

Bu bilgileri, adımı uygulamanıza yardımcı olması için kullanın.

Göreviniz

Aşağıdakileri uygulayan bir kod yazın:

1. `context.driver.find_element(By.ID)` çağrısını yapın, `'search_results'` değerini geçin ve sonuçları web öğesini temsil etmek için `element` adında bir değişkende saklayın.

   ▼ İpucu için buraya tıklayın.

   ```
   element = context.driver.find_element(By.ID, ... eleman id'si buraya ...)
   ```

2. Önceki satırdan `element` kullanarak, `text` niteliğinin "Kitty" kelimesini içermediğini `assert` edin.

   ▼ İpucu için buraya tıklayın.

   ```
   assert {bir değer} not in {elemanın metni}
   ```

3. Çalışmanızı kaydedin

Çözüm

▼ Çözüm için buraya tıklayın.

Bu, arama sonuçlarında "Kitty" görmemek için çözümdür:

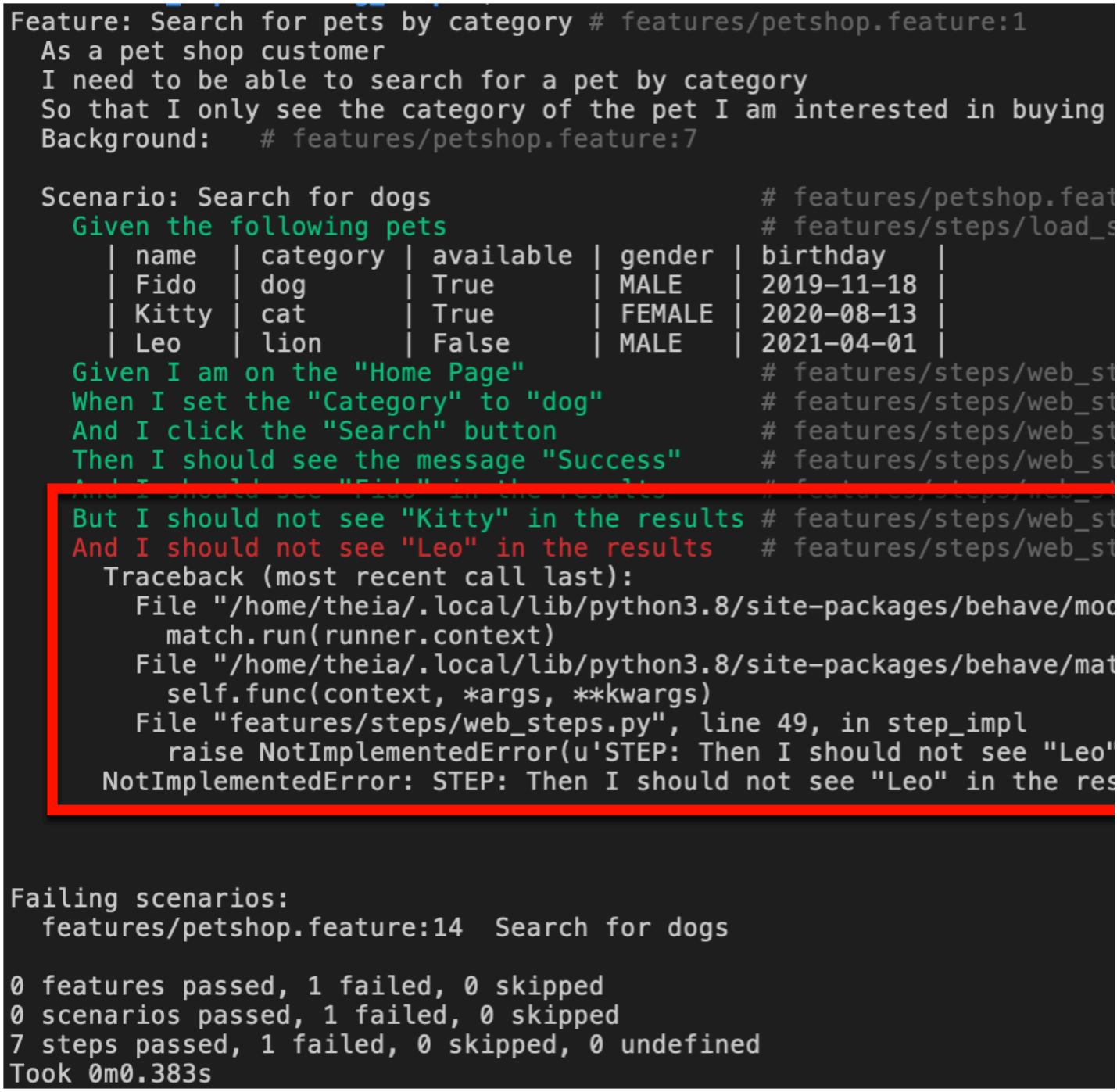
```
@then('I should not see "Kitty" in the results')
def step_impl(context):
    element = context.driver.find_element(By.ID, 'search_results')
    assert "Kitty" not in element.text
```

Sonuçlar

`behave` komutunu çalıştırın.

```
behave
```

Görmelisiniz ki **Ama “Kitty”yi sonuçlarda görmemeliyim** artık yeşil ve **Ve “Leo”yu sonuçlarda görmemeliyim** artık kırmızı.



Son kırmızı adımı uygulayalım.

Adım 7: Ve sonuçlarda Leo'yu görmemeliyim

Kırmızı olan bir sonraki adım **Ve sonuçlarda “Leo”yu görmemeliyim**. `web_steps.py` dosyasına bakarsanız, şu anda şöyle göründüğünü göreceksiniz:

```
@then('I should not see "Leo" in the results')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should not see "Leo" in the results')
```

Dikkat edin ki bu adım, önceki adıma çok benzer. Gelecek bir laboratuvar çalışmasında, değişken yerleştirmesi hakkında bilgi edinecek ve bu iki testi de ele alacak tek bir adım yazmayı öğreneceksiniz; ancak şu anda, yalnızca test edilen metinle farklılık gösteren bazı tekrar eden kodlarınız olacak.

Arka Plan

Çözümünüzü uygulamanıza yardımcı olacak bazı önemli arka plan bilgileri:

- Test edilen web sitesi, tüm arama sonuçlarını `search_results` kimliğine sahip bir öğeye yazar.
- Daha önce Selenium ile gördüğünüz gibi, bir web sayfasındaki öğeleri bulmanın bir yolu, onları `id=` niteliği ile adreslemektir. Bunu yapmak için Selenium çağrısı: `find_element(By.ID)`

```
element = context.driver.find_element(By.ID, '...öğe adı buraya...')
```

- Arama sonuçları öğesinin metnini `text` niteliğini kontrol ederek okumak için

```
element.text
```

Bu bilgileri kullanarak adımı uygulayın.

Göreviniz

Aşağıdakileri uygulayan bir kod yazın:

1. `context.driver.find_element(By.ID)` çağrısını yapın, `'search_results'` değerini geçirin ve sonuçları web öğesini temsil eden `element` adında bir değişkende saklayın.

   ▼ İpucu için buraya tıklayın.

   ```
   element = context.driver.find_element(By.ID, ... öğe kimliği buraya ...)
   ```

2. Önceki satırdaki `element`i kullanarak, `text` niteliğinin `"Leo"` kelimesini içermediğini `assert` edin.

   ▼ İpucu için buraya tıklayın.

   ```
   assert {bir değer} not in {öğenin metni}
   ```

3. Çalışmanızı kaydedin.

Çözüm

▼ Çözüm için buraya tıklayın.

Bu, arama sonuçlarında “Leo” görmemeniz için çözümdür:

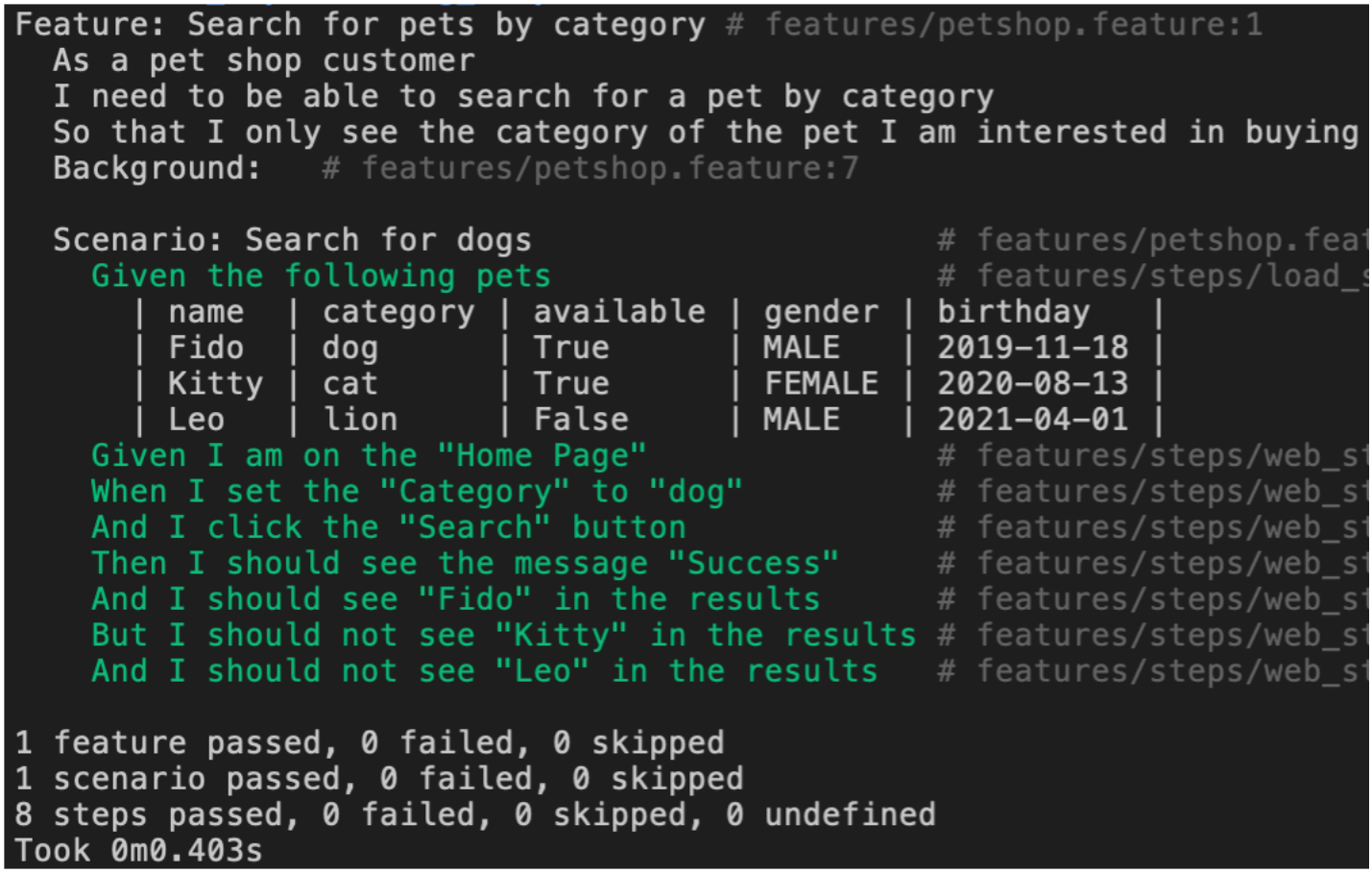
```
@then('I should not see "Leo" in the results')
def step_impl(context):
    element = context.driver.find_element(By.ID, 'search_results')
    assert "Leo" not in element.text
```

Sonuçlar

`behave` komutunu çalıştırın

```
behave
```

Görmelisiniz ki **Ve sonuçlarda "Leo"yu görmemeliyim** artık yeşil.



Tebrikler, tüm adımlar artık geçti.

Sonuç

Tebrikler! **İlk Adımlarınızı Uygulama** laboratuvarını tamamladınız. Artık `behave` aracının ürettiği adımları uygulayacak becerilere sahipsiniz, böylece test senaryoları geçebilir.

Bir sonraki zorluğunuz, bu teknikleri projelerinize uygulamak, uygulamanızın davranışını tanımlayan bir `.feature` dosyası oluşturmak ve ardından `behave` ile başlangıç adımlarını oluşturmak ve bu adımları geçecek şekilde uygulamaktır. Bu süreç, gerçek Davranışa Dayalı Geliştirme (Behavior Driven Development) uygulamalarını takip etmektedir.

Author(s)

John J. Rofrano

Contributors