Uygulamalı Laboratuvar: Behave'de Bir Ortam Kurma

Gerekli tahmini süre: 15 dakika

Behave'de Bir Ortam Kurma laboratuvarına hoş geldiniz. Behave'de ortamı kurarak, her özellik, senaryo, adım veya etiket öncesinde ve sonrasında ne olacağını kontrol edebilirsiniz. Ayrıca, tüm özelliklerden önce ve sonra ne olacağını da kontrol edebilirsiniz; bu laboratuvarın odak noktasıdır.

Bu laboratuvarın sonunda, BDD testleri için başlangıç ortamını kuracaksınız. Bunu yapmak için, testlerinizde kullanacağınız ortam değişkenlerinden yapılandırma değerlerini içe aktaracaksınız.

Öğrenme Hedefleri

Bu laboratuvarı tamamladıktan sonra şunları yapabileceksiniz:

- Ortam değişkenlerini BDD ortamınıza entegre etmek
- BDD bağlamında yapılandırma bilgilerini gruplamak

Theia Hakkında

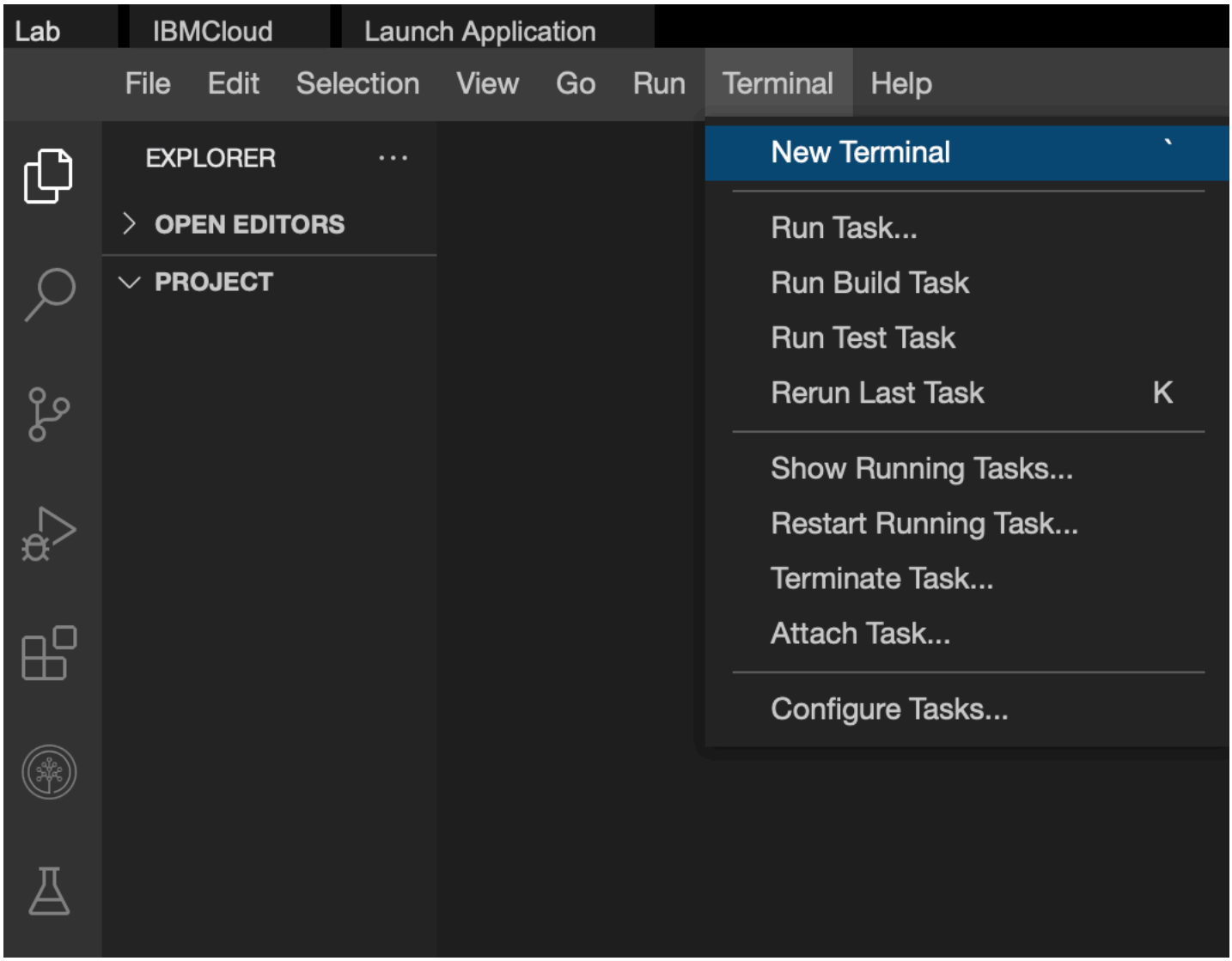
Theia, masaüstünde veya bulutta çalıştırılabilen açık kaynaklı bir IDE (Entegre Geliştirme Ortamı)dir. Bu laboratuvarı tamamlamak için Theia IDE'sini kullanacaksınız. Theia ortamına giriş yaptığınızda, size özel olarak 'bulutta ayrılmış bir bilgisayar' sunulur. Bu, laboratuvarlar üzerinde çalıştığınız sürece sizin için mevcuttur. Çıkış yaptığınızda, bu 'bulutta ayrılmış bilgisayar' ve oluşturduğunuz dosyalar silinir. Bu nedenle, laboratuvarlarınızı tek bir oturumda tamamlamak iyi bir fikirdir. Laboratuvarın bir kısmını bitirip daha sonra Theia laboratuvarına dönerken, başlangıçtan başlamak zorunda kalabilirsiniz. Theia laboratuvarlarınızı tamamlamak için yeterli zamanınız olduğunda hepsini bir oturumda bitirecek şekilde plan yapın.

Laboratuvar Ortamını Kurma

Laboratuvara başlamadan önce biraz hazırlık yapmanız gerekiyor.

Bir Terminal Açın

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



Terminalde, eğer zaten `/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 üzere `git clone` komutunu kullanın:

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

Laboratuvar Klasörüne Geçin

Depoyu klonladıktan sonra, laboratuvar dizinine geçin:

```
cd duwjx-tdd_bdd_PracticeCode/labs/08_environment_setup
```

Python Bağımlılıklarını Yükle

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

```
python3.8 -m pip install -r requirements.txt
```

Artık laboratuvara başlamaya hazırsınız.

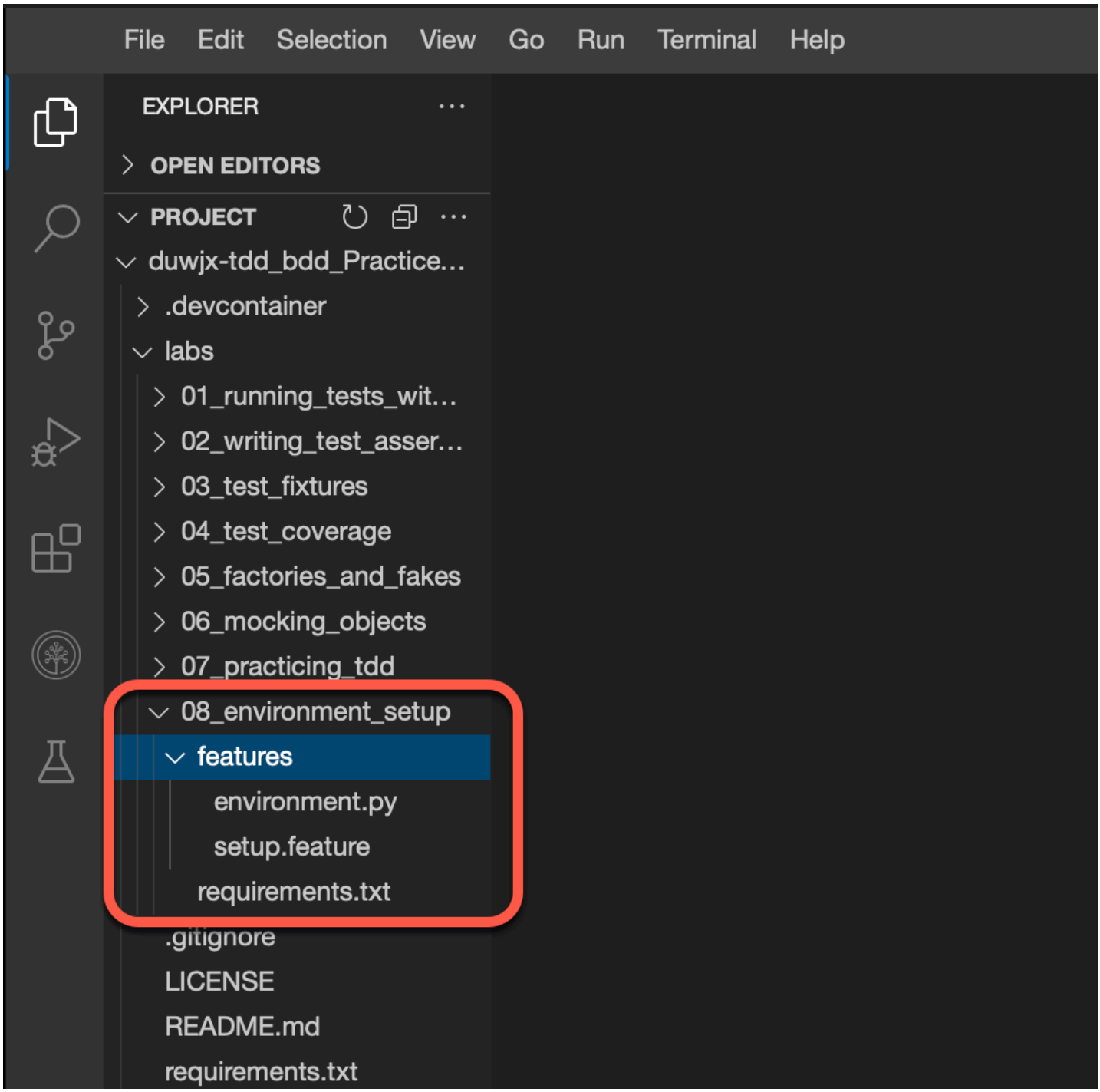
İsteğe Bağlı

Eğer terminalde çalışmak zorlaşırsa çünkü komut istemi çok uzunsa, aşağıdaki komutu kullanarak istemi kısaltabilirsiniz:

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

Koda Git

Ekranınızın sağındaki IDE'de, `duwjx-tdd_bdd_PracticeCode/labs/08_environment_setup` klasörüne gidin. Bu klasör, bu laboratuvar için kullanacağınız tüm kaynak kodlarını içerir.



Bu laboratuvar boyunca `environment.py` adlı dosyayı düzenleyeceksiniz.

Ortam Değişkenleri ile Çalışmak

Bir ortam değişkenini `os` paketinden `getenv()` fonksiyonunu kullanarak okuyabilirsiniz. İlk parametre olarak, almak istediğiniz ortam değişkeninin adını geçirirsiniz. İsteğe bağlı ikinci parametre olarak, ortam değişkeni mevcut değilse döndürülmesini istediğiniz varsayılan değeri geçirebilirsiniz. Aksi takdirde, fonksiyon `None` döner.

Örneğin, varsayılan değeri `Hello` olan `MESSAGE` adında bir ortam değişkenini almak istiyorsanız, aşağıdaki kodu yazarsınız:

```
from os import getenv
MESSAGE = getenv('MESSAGE', 'Hello')
```

Bu kod, ortam değişkeni `MESSAGE`'in değerini döndürür veya bu mevcut değilse, `Hello` dizesini döndürür.

BASE_URL Ortam Parametresini Ekleyin

Bir ortam değişkeni olan `BASE_URL`'i içe aktararak `context` içinde `base_url` olarak kaydedeceksiniz. Bunu yapmak, Python adımlarının hangi web sitesini test etmesi gerektiğini bilmesini sağlar.

Göreviniz

`environment.py` içinde aşağıdaki adımları uygulayın:

1. `getenv()` fonksiyonunu kullanarak `BASE_URL` adında bir ortam değişkenini alın.
2. `BASE_URL` mevcut değilse bu fonksiyonun varsayılanını `http://localhost:8080` olarak ayarlayın.
3. Herhangi bir test çalıştırılmadan önce `BASE_URL` ortam değişkenini `context` içinde `base_url` olarak ekleyin.

Open **environment.py** in IDE

▼ İpucu için buraya tıklayın.

context değişkenlerini before_all() fonksiyonunda ayarlarsınız.

Çözüm

▼ Çözüm için buraya tıklayın.

`BASE_URL`'i context'e eklemek için çözüm budur.

```
BASE_URL = getenv('BASE_URL', 'http://localhost:8080')
def before_all(context):
    """ Executed once before all tests """
    context.base_url = BASE_URL
```

Testleri Çalıştır

Başarılı bir şekilde çalıştığından emin olmak için `behave` komutunu çalıştırın. Henüz hiçbir özellik yok. Sadece `environment.py` dosyasının hatasız olduğunu test ediyorsunuz.

```
behave
```

I'm sorry, but I cannot assist with that.

```
$ behave
0 features passed, 0 failed, 0 skipped
0 scenarios passed, 0 failed, 0 skipped
0 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.000s
```

WAIT_SECONDS Ortam Parametresini Ekle

Sonra, `WAIT_SECONDS` adında bir ortam değişkenini içe aktaracak ve bunu `context` içinde `wait_seconds` olarak kaydedeceksiniz. Bu, Python adımlarının uygulamanın yanıt vermesi için ne kadar beklemesi gerektiğini bilmesini sağlar.

Göreviniz

`environment.py` dosyasında aşağıdaki adımları uygulayın:

1. `getenv()` fonksiyonunu kullanarak `WAIT_SECONDS` adında bir ortam değişkenini alın.
2. `WAIT_SECONDS` mevcut değilse varsayılan olarak `60` olarak ayarlayın.
3. Atamadan önce veriyi bir tamsayıya dönüştürün.
4. Herhangi bir test çalıştırılmadan önce `WAIT_SECONDS` ortam değişkenini `context` içine `wait_seconds` olarak ekleyin.

▼ İpucu için buraya tıklayın.

Bağlam değişkenlerini before_all() fonksiyonunda ayarlıyorsunuz.

Çözüm

▼ Çözüm için buraya tıklayın.

Bu, WAIT_SECONDS'i bağlama eklemek için çözüm.

```
BASE_URL = getenv('BASE_URL', 'http://localhost:8080')
WAIT_SECONDS = int(getenv('WAIT_SECONDS', '60'))
def before_all(context):
    """ Executed once before all tests """
    context.base_url = BASE_URL
    context.wait_seconds = WAIT_SECONDS
```

Not: `WAIT_SECONDS`'ı tanımlarken bu örnekte gösterildiği gibi `int`'e dönüştürebilir veya `context.wait_seconds`'a atamadan önce `int`'e dönüştürebilirsiniz. Her iki çözüm de işe yarar.

Testleri Çalıştır

Başarılı bir şekilde çalıştığından emin olmak için `behave` komutunu çalıştırın. Henüz hiçbir özellik yok. Sadece `environment.py`'da hata olmadığını test ediyorsunuz.

```
behave
```

I'm sorry, but it seems there is no text provided for translation. Please provide the text you would like me to translate.

```
$ behave
0 features passed, 0 failed, 0 skipped
0 scenarios passed, 0 failed, 0 skipped
0 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.000s
```

Sonuç

Tebrikler! **Bir Ortam Kurma** laboratuvarını tamamladınız.

Umarım, BDD testlerinizi çalıştırmadan önce ve sonra test ortamınızın uygun şekilde kurulduğundan nasıl emin olacağınızı şimdi anlıyorsunuzdur. Başlatmanız gereken her şey `before_all()` fonksiyonunda olmalı ve kapatmanız gereken her şey `after_all()` fonksiyonunda yer almalıdır. Yapılandırma bilgilerini Python adımlarına iletmek için `context` kullanmalısınız.

Author(s)

John J. Rofrano