Uygulamalı Laboratuvar: Kapsam ile Test Durumlarını Çalıştırma

Gerekli tahmini süre: 30 dakika

Kapsam ile Test Durumlarını Çalıştırma laboratuvarına hoş geldiniz. Test kapsamı, tüm testler sırasında yürütülen kod satırlarının yüzdesidir. Yüksek test kapsamı, test sırasında büyük miktarda kodun yürütüldüğünden emin olmanızı sağlar. Dolayısıyla, testler aracılığıyla yürütülen kod satırlarının sayısı arttıkça, kodunuzun beklenildiği gibi çalıştığından daha fazla emin olabilirsiniz.

Bu laboratuvarda, bir test kapsamı raporu oluşturarak test kapsamınızı nasıl geliştireceğinizi, raporu yorumlayarak hangi kod satırlarının test durumlarına sahip olmadığını belirlemeyi ve bu satırları kapsayacak test durumları yazmayı öğreneceksiniz.

Öğrenme Hedefleri

Bu laboratuvarı tamamladıktan sonra şunları yapabileceksiniz:

- Bir test kapsamı raporu oluşturmak
- Hangi kod satırlarının test durumları ile kapsanmadığını belirlemek için bir test kapsamı raporunu yorumlamak
- %100 test kapsamına ulaşmak için yeterli yeni test durumları yazmak

Theia Hakkında

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

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 için `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/04_test_coverage
```

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\[\033[00m\]]\$ "
```

Kod Editörünü Aç

IDE'de, `duwjx-tdd_bdd_PracticeCode/labs/04_test_coverage` klasörüne gidin. Bu klasör, bu laboratuvar için kullanacağınız tüm kaynak kodunu içerir.

```
duwjx-tdd_bdd_PracticeCode/labs/04_test_coverage
```

Dosya `tests/test_account.py` üzerinde tüm düzenleme işlerinizi yapacaksınız. Başlamak için, aşağıdaki butona tıklayarak bu dosyayı kod editöründe açın.

Open **test_account.py** in IDE

Testleri Çalıştırarak Başlayın

Herhangi bir kod yazmadan önce, test durumlarının geçtiğinden emin olmalısınız. Aksi takdirde, başarısız olduklarında, kodu bozup bozmadığınızı veya kodun başlamadan önce bozuk olup olmadığını bilemezsiniz.

`nosetests` komutunu çalıştıralım ve eksik kod kapsamını belirlemek için bir `coverage` raporu oluşturalım:

```
nosetests
```

Başlangıç raporunuz şöyle görünmelidir:

```
Name                 Stmts   Miss  Cover   Missing
--------------------------------------------------
models/__init__.py       6      0   100%
models/account.py       40     13    68%   26, 30, 34-35, 45-48, 52-54, 74-75
--------------------------------------------------
TOTAL                   46     13    72%
----------------------------------------------------------------------
Ran 2 tests in 0.349s
```

Test kapsamınız **%72** ile başlıyor. Hedefiniz **%100** ulaşmak! Hadi `account.py` dosyasındaki ilk atlanan satıra, satır **26**'ya bakalım ve bunun için bir test durumu yazıp yazamayacağımıza bakalım.

Adım 1: Eksik Satır 26

Kapsam raporu, `account.py` dosyasındaki `26` numaralı satırın çalıştıran bir test durumuna sahip olmadığını belirtti.

Göreviniz

Test kapsamınızı artırmak için, öncelikle `models/account.py` dosyasındaki `26` numaralı satırı inceleyin. Bu dosya, repo kökünden `model` paketinde yer alıyor.

`models/account.py` dosyasını açmak için aşağıdaki butona tıklayabilirsiniz:

Open **account.py** in IDE

`25` ve `26` numaralı satırlarda aşağıdaki kodu görmelisiniz.

```
def __repr__(self):
    return '<Account %r>' % self.name
```

Dikkat edin ki bu yöntem, sınıfı yazdırırken temsil etmek için çağrılan sihirli yöntemlerden biridir.

Göreviniz, bir Hesap üzerinde `__repr__()` yöntemini çağıran `test_account.py` dosyasına yeni bir test durumu eklemektir.

▼ İpucu için buraya tıklayın.

`account` üzerinde `str()` fonksiyonunu kullanarak `__repr__()` yöntemini çağırın.

Çözüm

▶ Çözüm için buraya tıklayın.

```
    def test_repr(self):
        """Test the representation of an account"""
        account = Account()
        account.name = "Foo"
        self.assertEqual(str(account), "<Account 'Foo'>")
```

▼ Details

::page{title="Adım 2: Eksik Satır 30"}

`nosetests` komutunu tekrar çalıştırın, böylece satır 26'nın artık testlerle kapsandığından emin olun ve yeni bir test durumu yazmanız gereken bir sonraki kod satırını belirleyin:

```
nosetests
```

Eğer önceki adımı doğru bir şekilde tamamladıysanız, test kapsamı raporunuz şöyle görünmelidir:

```
Name                 Stmts   Miss  Cover   Missing
----------------------------------------------------
models/__init__.py       6      0   100%
models/account.py       40     12    70%   30, 34-35, 45-48, 52-54, 74-75
----------------------------------------------------
TOTAL                   46     12    74%
----------------------------------------------------------------------
Ran 3 tests in 0.387s
```

Not edin ki genel test kapsamınız %72'den %74'e yükseldi ve yeni rapor, Eksik sütununda `26` numaralı satırı listelemiyor. O sütunda listelenen bir sonraki kod satırı `30`. Bu satırı `account.py` dosyasında inceleyerek o kodun ne yaptığını bulun.

Göreviniz

`28` ile `30` arasındaki satırlarda aşağıdaki kodu görmelisiniz.

```
    def to_dict(self) -> dict:
        """Serializes the class as a dictionary"""
        return {c.name: getattr(self, c.name) for c in self.__table__.columns}
```

Dikkat edin, bu kod `to_dict()` metodudur. Göreviniz, `test_account.py` dosyasına bir yeni test durumu eklemektir; bu test durumu bir Account üzerinde `to_dict()` metodunu çalıştırmalıdır.

Çözüm

▼ Çözüm için buraya tıklayın.

Bu kodu `test_account.py` dosyasına yapıştırın. Doğru bir şekilde girinti yaptığınızdan emin olun.

```
    def test_to_dict(self):
        """ Test account to dict """
        data = ACCOUNT_DATA[self.rand] # get a random account
        account = Account(**data)
        result = account.to_dict()
        self.assertEqual(account.name, result["name"])
        self.assertEqual(account.email, result["email"])
        self.assertEqual(account.phone_number, result["phone_number"])
        self.assertEqual(account.disabled, result["disabled"])
        self.assertEqual(account.date_joined, result["date_joined"])
```

▼ Details

::page{title="Adım 3: Eksik Satırlar 34-35"}

`nosetests` komutunu tekrar çalıştırarak 30. satırın artık testlerle kapsandığından emin olun ve yeni bir test durumu yazmanız gereken bir sonraki kod satırını belirleyin:

```
nosetests
```

Eğer önceki adımları doğru bir şekilde tamamladıysanız, test kapsamı raporunuz şu şekilde görünmelidir:

```
Name                 Stmts   Miss  Cover   Missing
--------------------------------------------------
models/__init__.py       6      0   100%
models/account.py       40     11    72%   34-35, 45-48, 52-54, 74-75
--------------------------------------------------
TOTAL                   46     11    76%
----------------------------------------------------------------------
Ran 4 tests in 0.368s
```

Not edin ki genel test kapsamınız %74'ten %76'ya yükseldi. Eksik sütununda listelenen sonraki kod satırları `34-35`'tir. Bu satırları `account.py` dosyasında inceleyerek o kodun ne yaptığını öğrenin.

Göreviniz

`34` ve `35` numaralı satırlarda aşağıdaki kodu görmelisiniz.

```
    def from_dict(self, data: dict) -> None:
        """Sets attributes from a dictionary"""
        for key, value in data.items():
            setattr(self, key, value)
```

Dikkat edin, bu kod `from_dict()` yöntemidir. Göreviniz, `test_account.py` dosyasına bir Hesap üzerinde `from_dict()` yöntemini çalıştıran yeni bir test durumu eklemektir.

Çözüm

▼ Çözüm için buraya tıklayın.

Bu kodu `test_account.py` dosyasına yapıştırın. Doğru şekilde girintilediğinizden emin olun.

```
    def test_from_dict(self):
        """ Test account from dict """
        data = ACCOUNT_DATA[self.rand] # get a random account
        account = Account()
        account.from_dict(data)
        self.assertEqual(account.name, data["name"])
        self.assertEqual(account.email, data["email"])
        self.assertEqual(account.phone_number, data["phone_number"])
        self.assertEqual(account.disabled, data["disabled"])
```

▼ Details

::page{title="Adım 4: Eksik Satırlar 45-48"}

`nosetests` komutunu tekrar çalıştırın, böylece 34 ve 35. satırların artık testlerle kapsandığından emin olun ve yeni bir test durumu yazmanız gereken bir sonraki kod satırını belirleyin:

```
nosetests
```

Eğer önceki adımları doğru bir şekilde tamamladıysanız, test kapsama raporunuz şöyle görünmelidir:

```
Name                 Stmts   Miss  Cover   Missing
----------------------------------------------------
models/__init__.py       6      0   100%
models/account.py       40      9    78%   45-48, 52-54, 74-75
----------------------------------------------------
TOTAL                   46      9    80%
----------------------------------------------------------------------
Ran 5 tests in 0.377s
```

Test kapsamınız **%80**'e çıktı! İyi iş!

Göreviniz

Şimdi `account.py` dosyasındaki `45-48` satırlarını inceleyin ve o kodun ne yaptığını bulun.

```
    def update(self):
        """Updates an account to the database"""
        logger.info("Saving %s", self.name)
        if not self.id:
            raise DataValidationError("Update called with empty ID field")
        db.session.commit()
```

Bu kodun `update()` metodu olduğunu unutmayın. Göreviniz, bir Hesap üzerinde `update()` metodunu çalıştıran `test_account.py` dosyasına yeni bir test durumu eklemektir.

Çözüm

▼ Çözüm için buraya tıklayın.

Bu kodu `test_account.py` dosyasına yapıştırın. Doğru bir şekilde girintilediğinizden emin olun.

```
    def test_update_an_account(self):
        """ Test Account update using known data """
        data = ACCOUNT_DATA[self.rand] # get a random account
        account = Account(**data)
        account.create()
        self.assertIsNotNone(account.id)
        account.name = "Foo"
        account.update()
        found = Account.find(account.id)
        self.assertEqual(found.name, "Foo")
```

::page{title="Adım 5: Eksik Satır 47"}

`nosetests` komutunu tekrar çalıştırarak 45 ile 48 arasındaki satırların artık testlerle kapsandığından emin olun ve yeni bir test durumu yazmanız gereken bir sonraki kod satırını belirleyin:

```
nosetests
```

Bu sefer sonuçlarınız şöyle görünmelidir:

```
Name                 Stmts   Miss  Cover   Missing
----------------------------------------------------
models/__init__.py       6      0   100%
models/account.py       40      4    90%   47, 52-54
----------------------------------------------------
TOTAL                   46      4    91%
----------------------------------------------------------------------
Ran 6 tests in 0.366s
```

You have **%91** test kapsamı! Ama `47` numaralı satırda ne oldu? Yazdığınız önceki test, `45` ile `48` arasındaki satırları kapsıyordu, ancak kapsam raporu hala `47` numaralı satırı Eksik sütununda listeliyor. Açıkça, önceki testte bazı koşullu mantık çalıştırılmamış.

Göreviniz

`account.py` dosyasının `45-48` numaralı satırlarını inceleyin ve o kodun ne yaptığını görün.

```
if not self.id:
    raise DataValidationError("Update called with empty ID field")
```

Dikkat edin ki `47` numaralı satır yalnızca `update()` metodu `id` olarak `None` ile çağrıldığında çalışır. Göreviniz, `update()` metodunu çalıştıran ve bu kod satırının çalışmasına neden olan yeni bir test durumunu `test_account.py` dosyasına eklemektir.

Çözüm

▼ Çözüm için buraya tıklayın.

Bu kodu `test_account.py` dosyasına yapıştırın. Doğru şekilde girintilediğinizden emin olun.

```
    def test_invalid_id_on_update(self):
        """ Test invalid ID update """
        data = ACCOUNT_DATA[self.rand] # get a random account
        account = Account(**data)
        account.id = None
        self.assertRaises(DataValidationError, account.update)
```

▼ Details

::page{title="Adım 6: Eksik Satırlar 53-54"}

`nosetests` komutunu tekrar çalıştırın, böylece 47. satırın artık testler aracılığıyla kapsandığından emin olun ve yeni bir test durumu yazmanız gereken bir sonraki kod satırını belirleyin:

```
nosetests
```

Bu sefer sonuçlarınız şöyle görünmelidir:

```
Name                 Stmts   Miss  Cover   Missing
----------------------------------------------------
models/__init__.py       6      0   100%
models/account.py       40      3    92%   52-54
----------------------------------------------------
TOTAL                   46      3    93%
----------------------------------------------------------------------
Ran 7 tests in 0.385s
```

Göreviniz

`account.py` dosyasının `52-54` satırlarını inceleyin ve o kodun ne yaptığını görün.

```
    def delete(self):
        """Removes an account from the data store"""
        logger.info("Deleting %s", self.name)
        db.session.delete(self)
        db.session.commit()
```

Dikkat edin ki `52-54` satırları `delete()` metodudur. Göreviniz, bir Hesap üzerinde `delete()` metodunu çalıştıran `test_account.py` dosyasına yeni bir test durumu eklemektir.

Çözüm

▼ Çözüm için buraya tıklayın.

Bu kodu `test_account.py` dosyasına yapıştırın. Doğru şekilde girintilediğinizden emin olun.

```
    def test_delete_an_account(self):
        """ Test Account update using known data """
        data = ACCOUNT_DATA[self.rand] # get a random account
        account = Account(**data)
        account.create()
        self.assertEqual(len(Account.all()), 1)
        account.delete()
        self.assertEqual(len(Account.all()), 0)
```

▼ Details

::page{title="Adım 7: %100 Test Kapsamı"}

Son bir kez `nosetests` çalıştırarak test kapsamınızı görün:

```
nosetests
```

Başarı!

Hiçbir eksik satır olmadan %100 kod kapsama oranına ulaştınız!

```
Name                 Stmts   Miss  Cover   Missing
--------------------------------------------------
models/__init__.py       6      0   100%
models/account.py       40      0   100%
--------------------------------------------------
TOTAL                   46      0   100%
----------------------------------------------------------------------
Ran 8 tests in 0.415s
OK
```

::page{title="Sonuç"}

Test Kapsama Laboratuvarını Tamamladığınız İçin Tebrikler

Başardınız! `account` modülündeki her bir kod satırını çalıştıracak kadar test durumu yazdınız. Artık bilinen verilerle test ettiğinizde her bir kod satırının çalıştığını biliyorsunuz. Ancak, kodunuzda hala hatalar olabilir ve bu hatalar yalnızca kodunuza kötü veya beklenmedik veriler gönderildiğinde ortaya çıkacaktır. Daha fazla olasılığı kapsayan yeni test durumları yazmaktan asla vazgeçmeyin.

Author(s)

John J. Rofrano