Flask’ta Dekoratörler

Gerekli tahmini süre: **10** dakika

Hedefler

Bu sayfayı okuduktan sonra şunları yapabileceksiniz:

1. Dekoratörlerin ne olduğunu anlamak
2. Bir Python uygulamasında karşılaştığınız iki tür dekoratörü anlamak
3. Dekoratörleri ne zaman ve nasıl kullanacağınızı öğrenmek.

Dekoratörler Nedir

Dekoratörler, yöntemleri anot etmekte ve belirli bir yöntemin ne amaçla kullanıldığını belirtmekte yardımcı olur. Bunlar yorumlardan farklıdır. Yorumlayıcı, kodu çalıştırırken bunları kullanır.

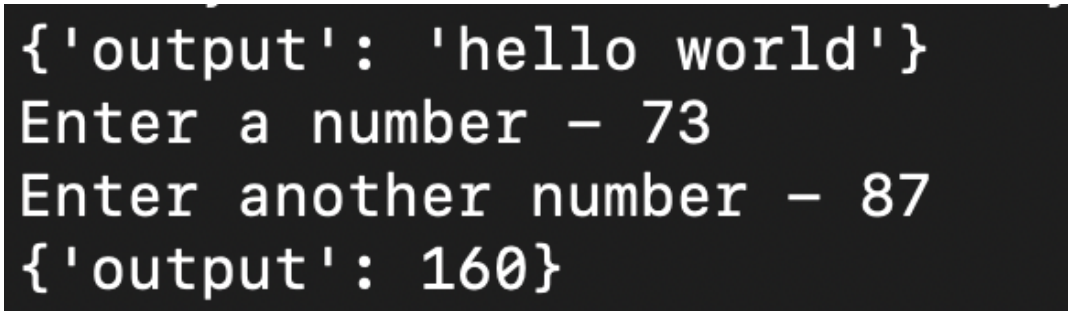
Yöntem dekoratörleri

Diyelim ki, tüm çıktının JSON formatında olmasını istediğimiz bir Python kodumuz var. Bu kodu her bir yöntemde dahil etmek mantıklı değil çünkü bu, kod satırlarını gereksiz hale getirir. Böyle durumlarda, bunu bir dekoratör ile halledebiliriz.

```
def jsonify_decorator(function):
    def modifyOutput():
        return {"output":function()}
    return modifyOutput
@jsonify_decorator
def hello():
    return 'hello world'
@jsonify_decorator
def add():
    num1 = input("Enter a number - ")
    num2 = input("Enter another number - ")
    return int(num1)+int(num2)
print(hello())
print(add())
```

Yöntem dekoratörü, dekorasyon yaptığı fonksiyonun çıktısını saran sarıcı olarak da adlandırılır. Yukarıdaki kod örneğinde, `jsonify-decorator` dekoratör yöntemidir. Bu dekoratörü `hello()` ve `add()` yöntemlerine ekledik. Bu yöntem çağrılarının çıktısı artık jsonify_decorator ile sarılacak ve süslenecektir.

Yukarıdaki Python kodunu çalıştırdığınızda elde edeceğiniz çıktı:



Gördüğünüz gibi, yöntem çağrısı dekoratör ile sarılmıştır. Kodlayıcı, dekoratörün adını seçme özgürlüğüne sahiptir. Burada seçilen isim `jsonify_decorator`‘dır.

Route Decorators

Herhangi bir alan adını ziyaret ettiğinizde kök ve erişebileceğiniz diğer uç noktaları görebilirsiniz. Aşağıdaki örneklere bakın.

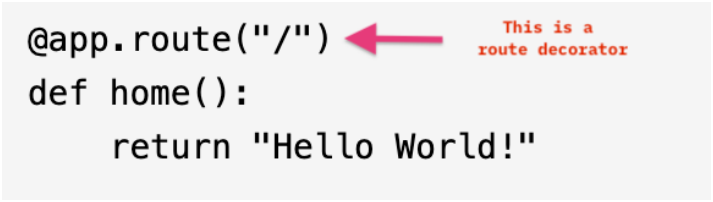
`https://mydomain.com/`

`https://mydomain.com/about`

`https://mydomain.com/register`

Bu uç noktaları, takip eden laboratuvarlarda flask modülünü kullanarak bir web uygulaması oluşturduğumuzda oluşturmayı inceleyeceğiz.

Ancak bu uç noktaları Python’da tanımlamak için **Route Decorators** olarak adlandırdığımız şeyi kullanıyoruz.



@app.route("/") Flask'ın uygulamamızdaki URL'leri fonksiyonlara kolayca atamak için sağladığı bir Python dekoratörüdür. Dekoratörün, kullanıcı uygulamamızın alanını ziyaret ettiğinde, bizim durumumuzda `home()` fonksiyonunu çalıştırmasını söylediğini kolayca anlayabilirsiniz.

Birden fazla rotayı, çağrıldığında hangi yöntemin çalıştırılması gerektiği üzerine ek dekoratörler ekleyerek tek bir fonksiyonla yönetebiliriz. Aşağıda, 3 ayrı rota için aynı "Hello World!" mesajını sunmanın geçerli bir örneği bulunmaktadır:

```
@app.route("/")
@app.route("/home")
@app.route("/index")
def home():
    return "Hello World!"
```

Route dekoratörleri ayrıca `method`'u ikinci parametre olarak alır, bu da rotanın destekleyeceği HTTP yöntemlerinin türünü ayarlamak için kullanılabilir. Bunu daha sonraki bölümlerde öğreneceğiz.

Route dekoratörü ayrıca daha spesifik olabilir. Örneğin, kullanıcıId'si U0001 olan bir kullanıcının detaylarını almak için `http://mydomain.com/userdetails/U0001` adresine gidebilirsiniz. İlgilendiğiniz her kullanıcı için farklı bir rota tanımlamak mantıklı değildir. Böyle durumlarda, rotayı şu şekilde tanımlıyoruz.

```
@app.route("/userdetails/<userid>")
def getUserDetails(userid):
    return "User Details for  "+userid
```

Tebrikler! Python'da decorator'lar hakkında bilgi edindiniz. Hemen kodunuzda kullanın.

Yazarlar

Lavanya