

Flask ile Pratik Bölüm 1



Gerekli tahmini süre: 20 dakika

Capstone kursunun ilk laboratuvarına hoş geldiniz. Bu laboratuvar da Flask ile çalışma pratiği yapacaksınız. Bu laboratuvar için ihtiyaç duyduğunuz tüm kavramları önceki video setinden bilmeniz gerekiyor. Bir görevi nasıl gerçekleştireceğinizden emin değilseniz ya da daha fazla bilgiye ihtiyacınız varsa laboratuvarı duraklatmaktan çekinmeyin ve modülü gözden geçirin.

Öğrenme Hedefleri

Bu laboratuvarı tamamladıktan sonra şunları yapabileceksiniz:

- Geliştirme modunda bir Flask sunucusu oluşturmak ve çalıştırmak
- Bir uç noktadan JSON döndürmek
- İstek nesnesini anlamak

Skills Network Cloud IDE Hakkında

Skills Network Cloud IDE (Theia ve Docker tabanlı) kurs ve proje ile ilgili laboratuvarlar için uygulamalı bir ortam sağlar. Theia, masaüstünde veya bulutta çalışabilen açık kaynaklı bir IDE'dir (Entegre Geliştirme Ortamı). Bu laboratuvarı tamamlamak için, Theia ve Docker konteynerinde çalışan MongoDB tabanlı Cloud IDE'yi kullanacaksınız.

Bu laboratuvar ortamı hakkında önemli bir not

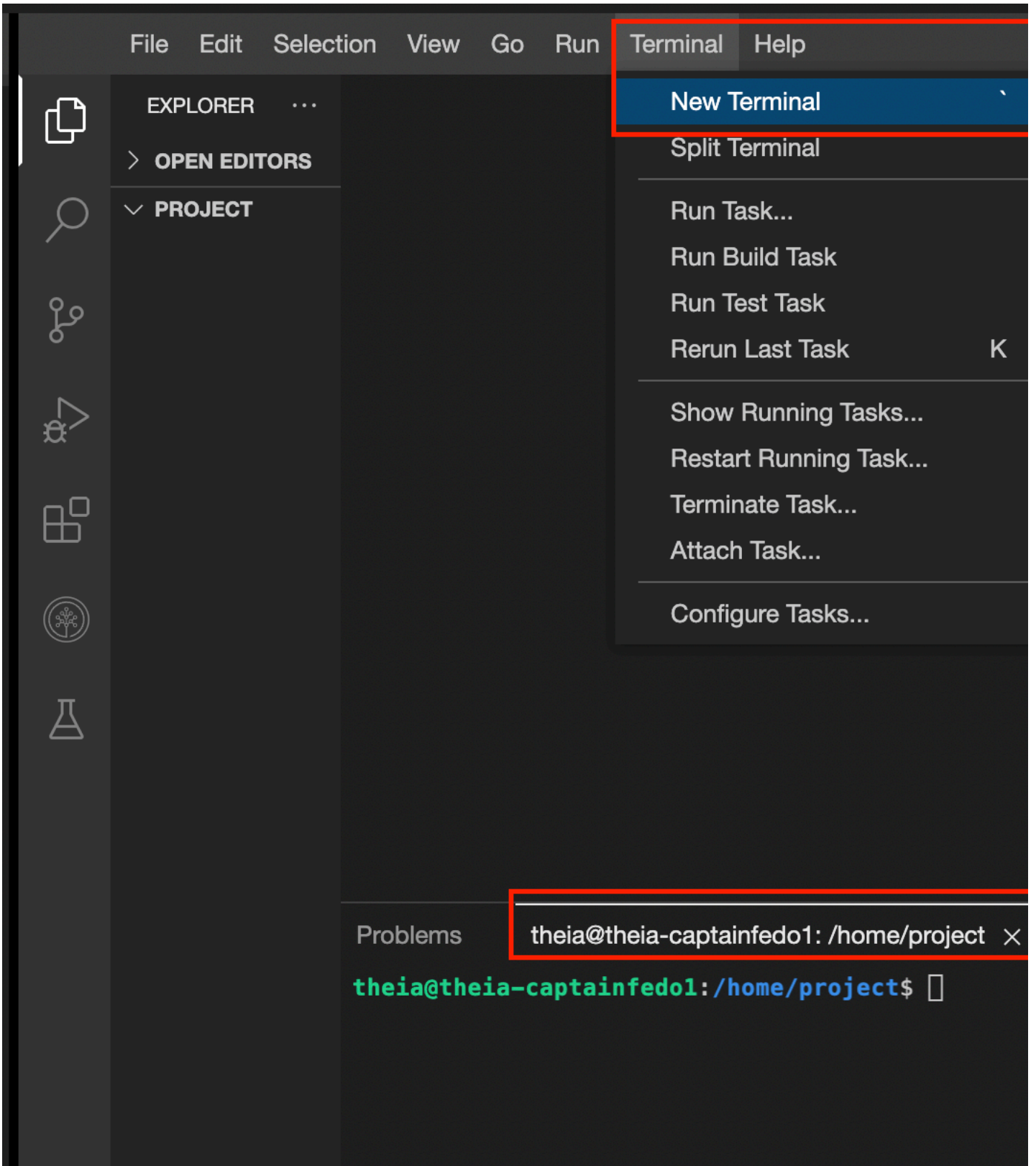
Bu laboratuvar ortamında oturumların kalıcı olmadığını lütfen unutmayın. Bu laboratuvara her bağlandığınızda, sizin için yeni bir ortam oluşturulur. Önceki oturumlarda kaydettiğiniz veriler kaybolacaktır. Verilerinizi kaybetmemek için bu laboratuvarları tek bir oturumda tamamlamayı planlayın.

Laboratuvar Ortamını Kurma

Laboratuvara başlamadan önce bazı ön hazırlıklar gereklidir.

Bir Terminal Açın

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



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

```
cd /home/project
```

Laboratuvar dizinini oluşturun

Artık sunucu dosyanız için bir dizin oluşturabilirsiniz.

```
mkdir lab
```

lab dizinine geçin:

```
cd lab
```

Python sürümünü kontrol et ve Flask'ı kur

python3 --version komutunu kullanarak laboratuvar ortamındaki python3 sürümünü kontrol edin. Aşağıdaki gibi bir çıktı görmelisiniz:

```
theia@theiadocker-captainfedo1:/home/project/lab$ python3 --version
Python 3.10.12
```

Sonraki adımda, Flask'ı aşağıdaki komutla kurun:

```
pip3 install flask
```

Eğer sistemde Flask mevcutsa, aşağıdaki mesajı göreceksiniz:

```
Requirement already satisfied: flask in /home/theia/.local/lib/python3.10/site-packages (3.1.2)
Requirement already satisfied:
...
```

Artık laboratuvara başlamaya hazırsınız.

İsteğe Bağlı

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

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

Adım 1: Hello World sunucusunu oluřtur

Görevleriniz

1. server.py dosyasını oluřturun.

Öncelikle, terminalde server.py adında boş bir dosya oluřturun veya dosya düzenleyici menüsünü kullanın.

- ▼ İpucu almak için buraya tıklayın.

Ařağıdaki komut, boş dosyayı doğru dizinde oluřturacaktır.

```
touch /home/project/lab/server.py
```

server.py dosyasını editörde açın

Open **server.py** in IDE

Bu dosyayı açtıktan sonra Python - Get Started adında yeni bir sekme açılırsa, python dosyasına geri dönmek için kapatabilirsiniz.

2. Flask modülünü içe aktarın.

Bir sonraki adımda, bu dosyada Flask modülünü içe aktararak sunucu kodlamaya başlayabilirsiniz.

- ▼ İpucu için buraya tıklayın.

Modül adını değıřtirerek bu dosyada Flask sınıfını içe aktarın.

```
from flask import {insert module name here}
```

3. Flask uygulamasını oluřturun

Flask modülünü içe aktardıktan sonra, Flask sınıfını başlatarak Flask uygulamanızı oluřturun.

- ▼ İpucu için buraya tıklayın.

Flask sınıfından yeni bir uygulama başlatın.

```
from flask import {insert module name here}
app = {insert module name here}(__name__)
```

4. Ana rotayı oluřturun.

Önceki görevde oluřturduğunuz uygulamayı kullanarak ilk rotanızı oluřturabilirsiniz.

- ▼ İpucu için buraya tıklayın.

Kök URL'sini oluřturmak için uygulama dekoratörünü kullanın “/”.

```
# Import the Flask class from the flask module
from flask import Flask
# Create an instance of the Flask class, passing in the name of the current module
app = Flask(__name__)
# Define a route for the root URL ("/")
@app.route("/")
```

```
def home():  
    return "Hello, World!"
```

5. Ana kök URL'si için yöntemi tanımlayın.

Öncelikle bu dosyada Flask'ı içe aktarın.

▼ İpucu için buraya tıklayın.

Yöntem tanımına başlayın.

```
# Import the Flask class from the flask module  
from flask import Flask  
# Create an instance of the Flask class, passing in the name of the current module  
app = Flask(__name__)  
# Define a route for the root URL ("/")  
@app.route("/")  
def home():  
    # Function that handles requests to the root URL  
    return "Hello, World!"
```

6. İstemciye “Hello World” mesajını döndürün.

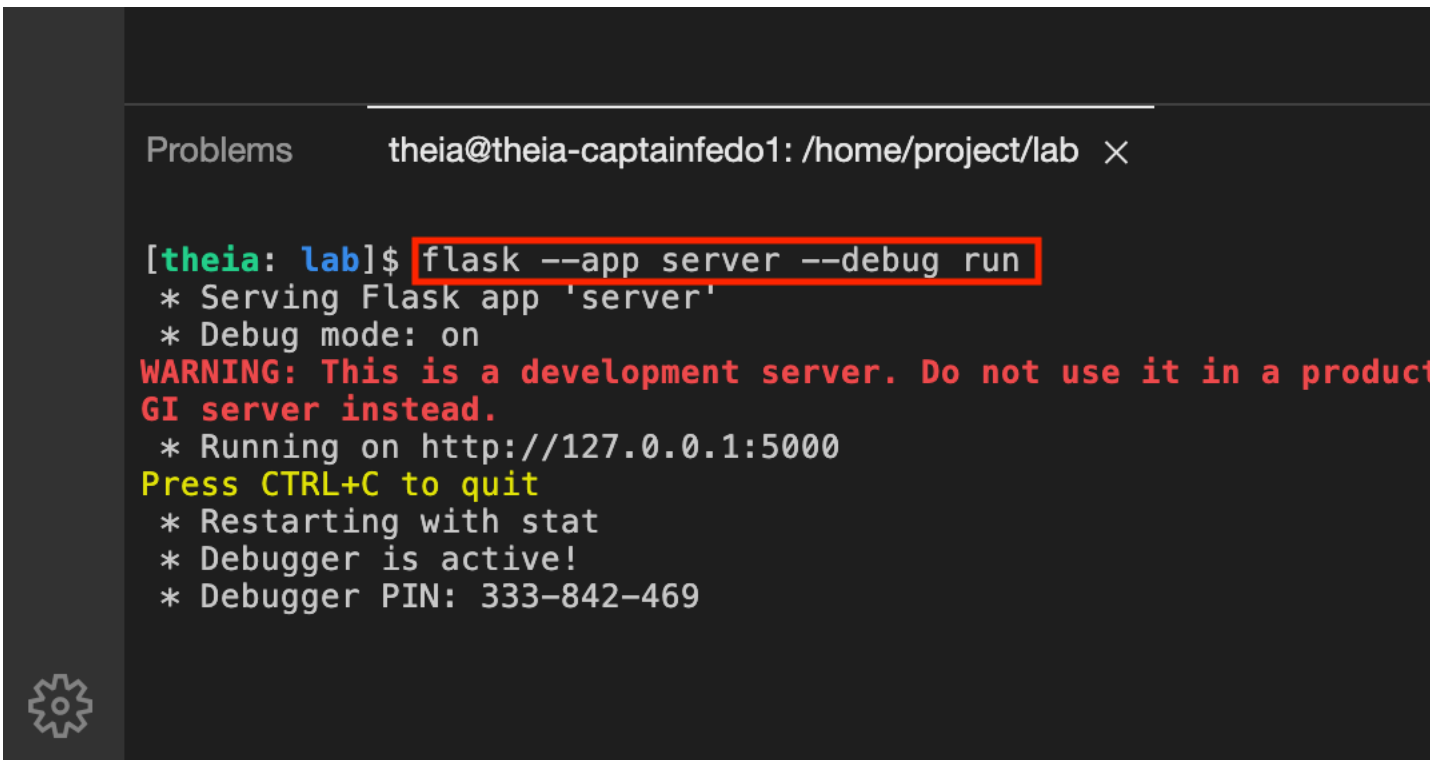
İstemciye “Hello World” dizesini döndürün.

▼ İpucu için buraya tıklayın.

```
# Import the Flask class from the flask module  
from flask import Flask  
# Create an instance of the Flask class, passing in the name of the current module  
app = Flask(__name__)  
# Define a route for the root URL ("/")  
@app.route("/")  
def hello_world():  
    # Function that handles requests to the root URL  
    return "Hello, World!"
```

Sunucuyu çalıştırmak için hazırsınız. Sunucuyu terminalden çalıştırmak için aşağıdaki komutu kullanın:

```
flask --app server --debug run
```

A terminal window titled 'theia@theia-captainfedo1: /home/project/lab' with a 'Problems' tab. The command 'flask --app server --debug run' is entered and highlighted with a red box. The output shows the server starting in debug mode on http://127.0.0.1:5000, with a warning to use a production server instead. The terminal also shows the server restarting with stat and the debugger being active with a PIN of 333-842-469.

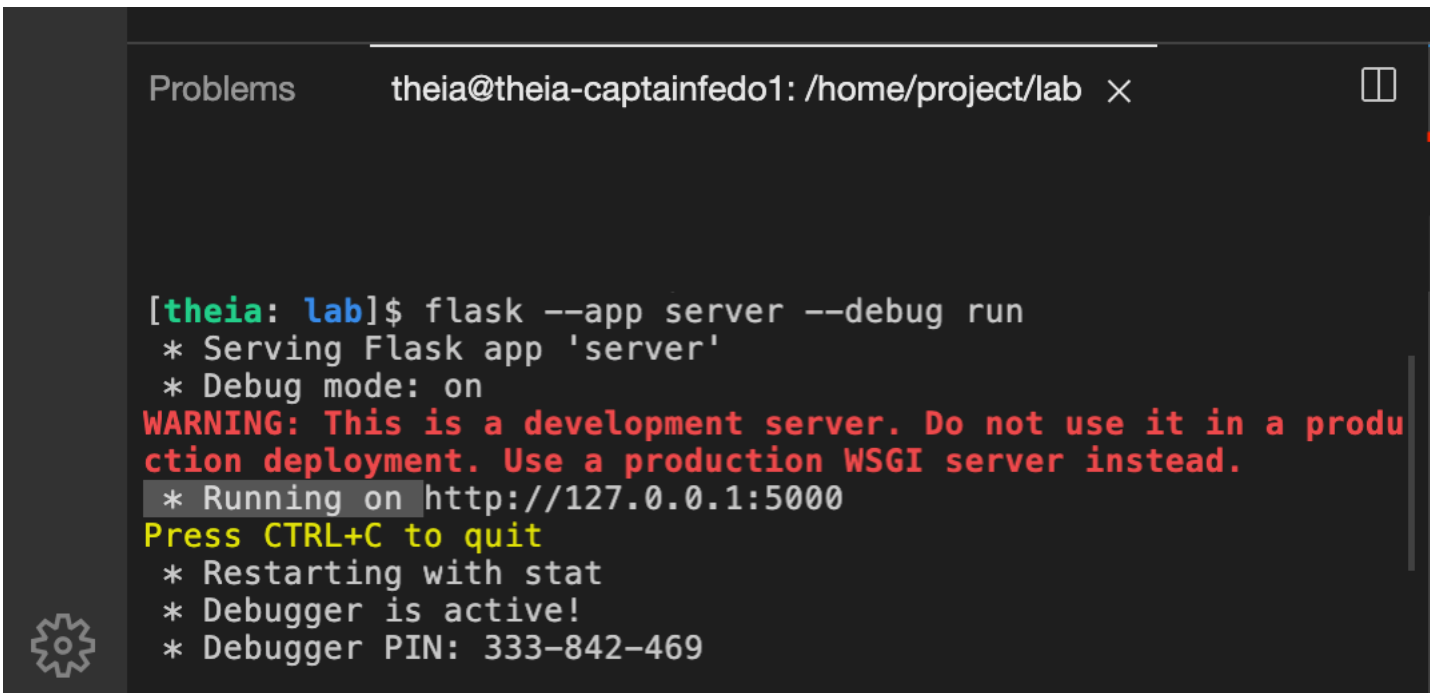
```
theia@theia-captainfedo1: /home/project/lab ×  
[theia: lab]$ flask --app server --debug run  
* Serving Flask app 'server'  
* Debug mode: on  
WARNING: This is a development server. Do not use it in a production  
GI server instead.  
* Running on http://127.0.0.1:5000  
Press CTRL+C to quit  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 333-842-469
```

Artık localhost:5000/ üzerinde CURL komutunu kullanabilmelisiniz. Terminalin zaten sunucuyu çalıştırdığını unutmayın, terminali bölmek için Split Terminal butonunu kullanarak ikinci sekmede aşağıdaki komutu çalıştırabilirsiniz.

Not: Bağlantı reddi hatasıyla karşılaşmamak için /home/project/lab dizininde Server.py dosyasının bulunduğundan emin olun.

```
curl -X GET -i -w '\n' localhost:5000
```

-X argümanı GET komutunu belirtir ve -i argümanı yanıtın başlığını gösterir.

A terminal window titled 'theia@theia-captainfedo1: /home/project/lab' with a 'Problems' tab. The command 'flask --app server --debug run' is entered. The output shows the server starting in debug mode on http://127.0.0.1:5000, with a warning to use a production server instead. The terminal also shows the server restarting with stat and the debugger being active with a PIN of 333-842-469.

```
theia@theia-captainfedo1: /home/project/lab ×  
[theia: lab]$ flask --app server --debug run  
* Serving Flask app 'server'  
* Debug mode: on  
WARNING: This is a development server. Do not use it in a produ  
ction deployment. Use a production WSGI server instead.  
* Running on http://127.0.0.1:5000  
Press CTRL+C to quit  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 333-842-469
```

CURL komutunun çıktısı olarak Hello World döndüğünü görmelisiniz. HTTP 200 OK dönüş durumunu ve Content-type'in text/html olduğunu not edin. Bu laboratuvarın bir sonraki bölümünde düz metin yerine JSON ile özel bir durum döndürmeniz isteniyor.

Çözüm

Çalışmanızın aşağıdaki çözümle eşleştiğinden emin olun.

▼ Cevap için buraya tıklayın.

```
# Import the Flask class from the flask module
from flask import Flask
# Create an instance of the Flask class, passing in the name of the current module
app = Flask(__name__)
# Define a route for the root URL ("/")
@app.route("/")
def hello_world():
    # Function that handles requests to the root URL
    return "Hello, World!"
```

Adım 2: JSON Döndür

Göreviniz

Flask sunucusunda ilk rota işleyicinizi oluşturduğunuz için tebrikler. `@app.route()` yöntemlerinden birçok farklı içerik türü döndürebilirsiniz. Bu projenin amacı için, **Merhaba Dünya** dizesi yerine aşağıdaki JSON'u döndürelim.

```
"message": "Hello World"
```

Hatırlayın ki, videolardan bir JSON nesnesini yöntemde döndürmenin iki yolu vardır:

1. Bir Python sözlüğü döndürmek
2. Bir dize üzerinde **jsonify()** yöntemini kullanmak

Bu laboratuvar çalışmasında birinci yöntemi kullanmanız isteniyor.

İpucu

Mevcut **index** yöntemini istenen JSON mesajını döndürmek için düzenleyebilirsiniz.

▼ İpucu için buraya tıklayın.

Index yönteminde Hello World mesajını içeren bir sözlük döndürün.

```
# Import the Flask class from the flask module
from flask import Flask, jsonify
# Create an instance of the Flask class, passing in the name of the current module
app = Flask(__name__)
# Define a route for the root URL ("/")
@app.route("/")
def index():
    return {insert dictionary here}
```

Çözüm

Yaptığınız işin aşağıdaki çözümle eşleştiğinden emin olun.

▼ Cevap için buraya tıklayın.

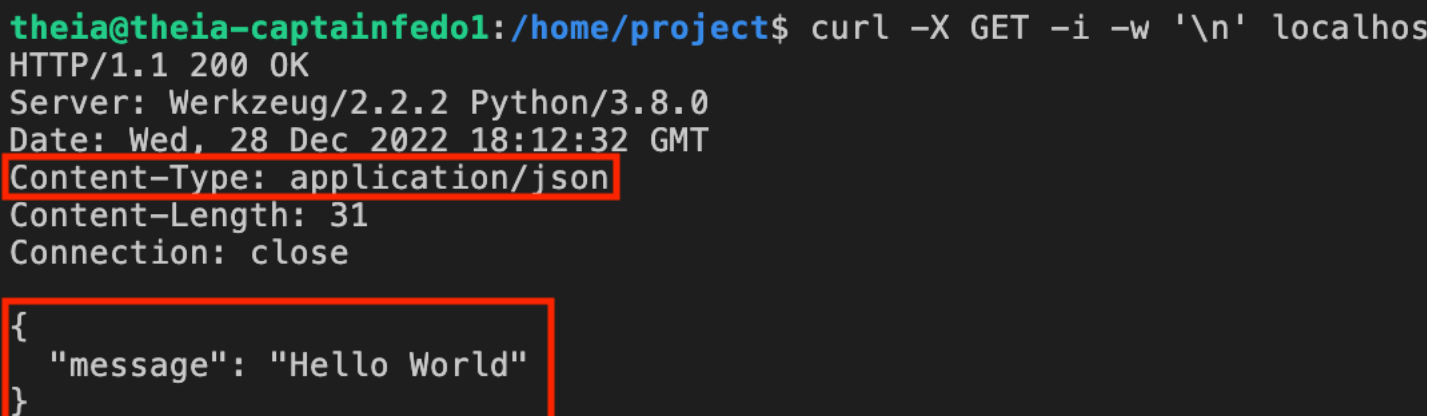
```
# Import the Flask class from the flask module
from flask import Flask
# Create an instance of the Flask class, passing in the name of the current module
app = Flask(__name__)
# Define a route for the root URL ("/")
@app.route("/")
def index():
    # Function that handles requests to the root URL
    # Create a dictionary to return as a response
    return {"message": "Hello World!"}
```

Sunucu çalışıyorsa, devam edebilirsiniz. Aksi takdirde, sunucuyu aşağıdaki komutla tekrar çalıştırabilirsiniz:

```
flask --app server --debug run
```

Artık localhost:5000/ ile CURL komutunu kullanabilmeniz gerekiyor. Terminalin sunucuyu çalıştırdığını unutmayın, terminali bölmek için `Split Terminal` butonunu kullanarak ikinci sekmede aşağıdaki komutu çalıştırabilirsiniz.

```
curl -X GET -i -w '\n' localhost:5000
```



```
theia@theia-captainfedo1:/home/project$ curl -X GET -i -w '\n' localhost:5000
HTTP/1.1 200 OK
Server: Werkzeug/2.2.2 Python/3.8.0
Date: Wed, 28 Dec 2022 18:12:32 GMT
Content-Type: application/json
Content-Length: 31
Connection: close

{"message": "Hello World"}
```

`{"message": "Hello World"}` JSON'unun CURL komutunun çıktısı olarak döndüğünü görmelisiniz. Bu sefer HTTP 200 OK dönüş durumunu ve Content-type'in application/json olduğunu not edin.

Author(s)

CF

© IBM Corporation 2023. Tüm hakları saklıdır.