



PYTHON PROGRAMMING AND MACHINE LEARNING

MODEL ENGINEERING

- **Server**
- Model Deployment

- WSGI Server
 - A Web Server Gateway Interface (WSGI) server implements the web server side of the WSGI interface for running Python web applications.
 - Flask
 - Lightweight Web application framework
 - *Not scalable and not recommended for production environment, we can develop a Waitress Server to load a Flask application*
 - Waitress
 - is a WSGI *application*
- # under your conda environment
- Type '**pip install flask waitress**' in the command line / terminal



Writing a simple service -1

```
from flask import Flask,  
request, jsonify  
from waitress import serve  
  
app = Flask(__name__)  
  
@app.route("/")  
def hello():  
    return "Hello world!"  
  
if __name__ == '__main__':  
    serve(app, host="0.0.0.0",  
port=8080)
```

First parameter represent the name of the module if we use multi packages in Flask. In a single module, it's not so important

This code is only executed once.

Very common python code in many frameworks



Writing a simple service -2

```
from flask import Flask,  
request, jsonify  
from waitress import serve  
  
app = Flask(__name__)  
  
@app.route("/")  
def hello():  
    return "Hello world!"  
  
if __name__ == '__main__':  
    serve(app, host="0.0.0.0",  
port=8080)
```

We bind the root URL "/" to
hello() function.

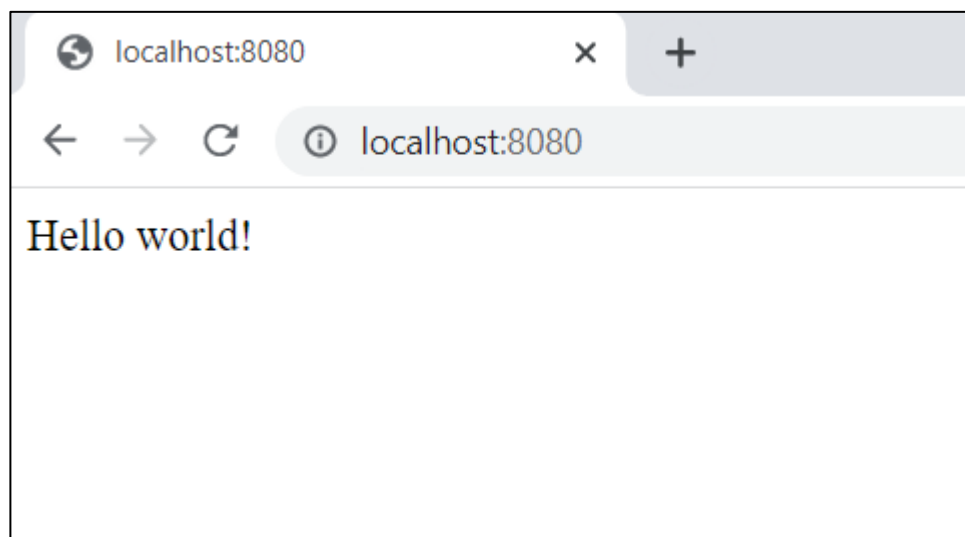
The return type must be
a string,
a tuple of (response, status
code),
a Response object or
a WSGI callable (need to
understand WSGI, etc)

Can be used to return
HTML for web application,
or JSON/XML/text for web
services



Starting the Python Server

Start the server and send a request from the browser





Service – Binding url Argument

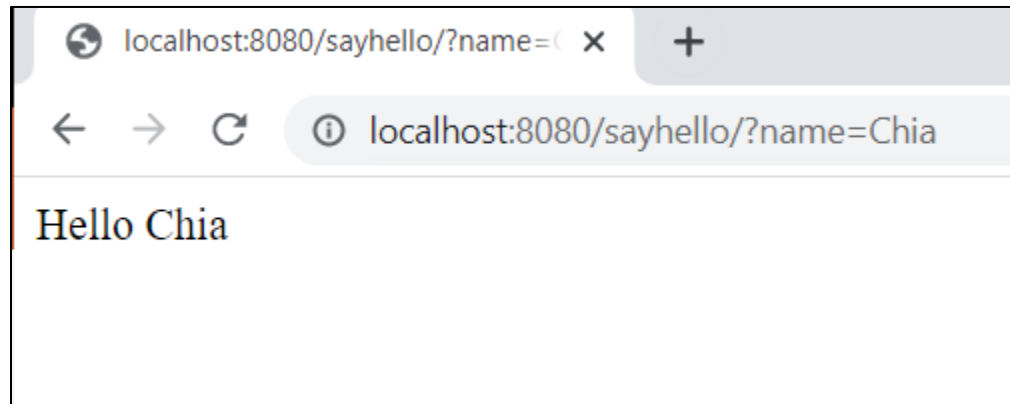
app.py

```
@app.route('/sayhello/', methods=['GET', 'POST'])
def say_hello():
    name = request.args.get('name') or request.form.get('name')
    return "Hello " + str(name or '')
```

- We bind the URL **"/sayhello"** to **say_hello()** function.
- We get the name either from the URL's **query string** or from the **form** submitted using POST method – therefore we use the 'or' trick here
- We return a simple text

Testing the Service -1

- Get Request:





Testing the Post Request

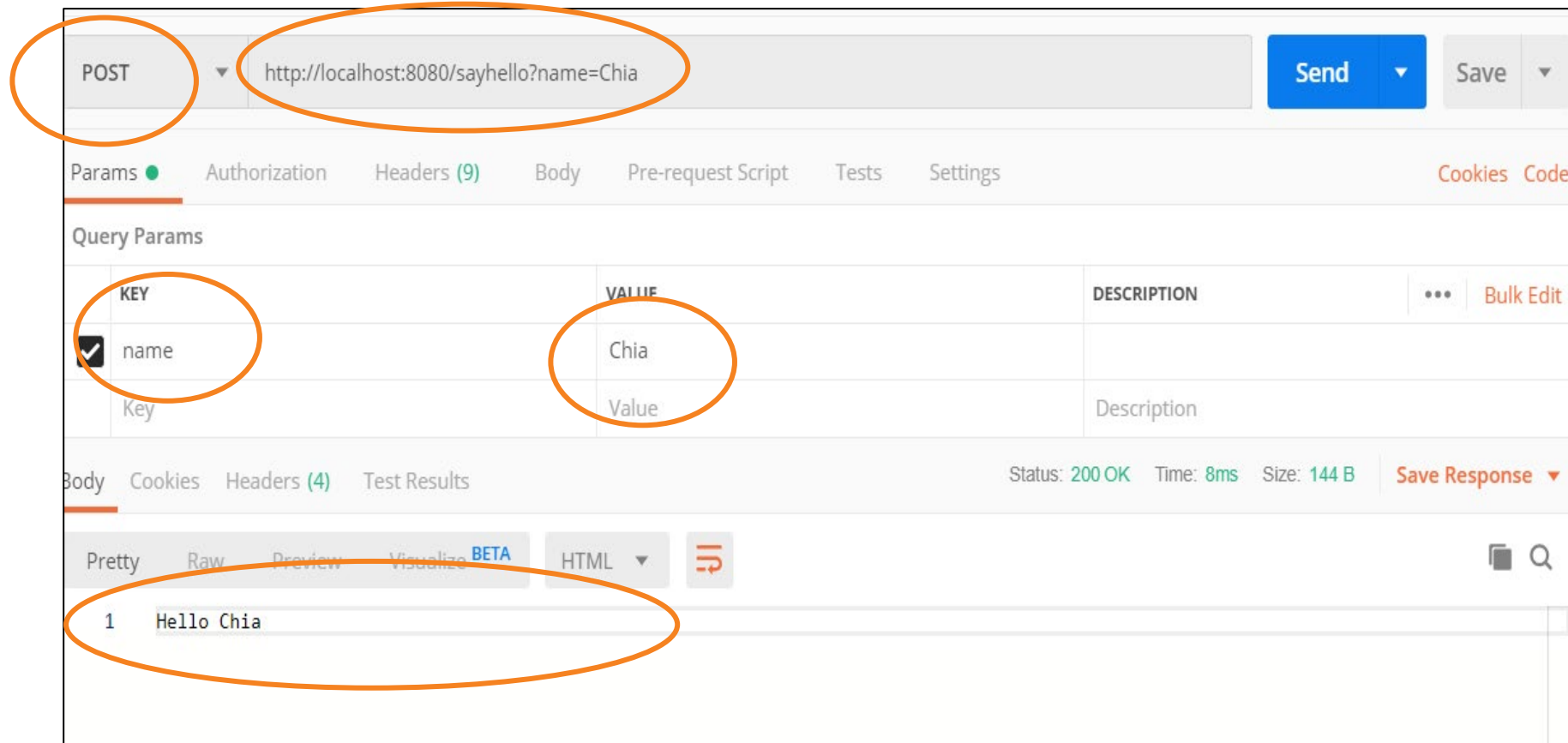
Alternatives

- Create a web application (e.g. .Net MVC web client)
- Using Postman (which could be downloaded from the internet)



Testing the Service -2

- Form Post Request:




POST Send Save

Params ☒ Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> name	Chia			
Key	Value	Description		

Body Cookies Headers (4) Test Results Status: 200 OK Time: 8ms Size: 144 B Save Response

Pretty Raw Preview Visualize BETA HTML 

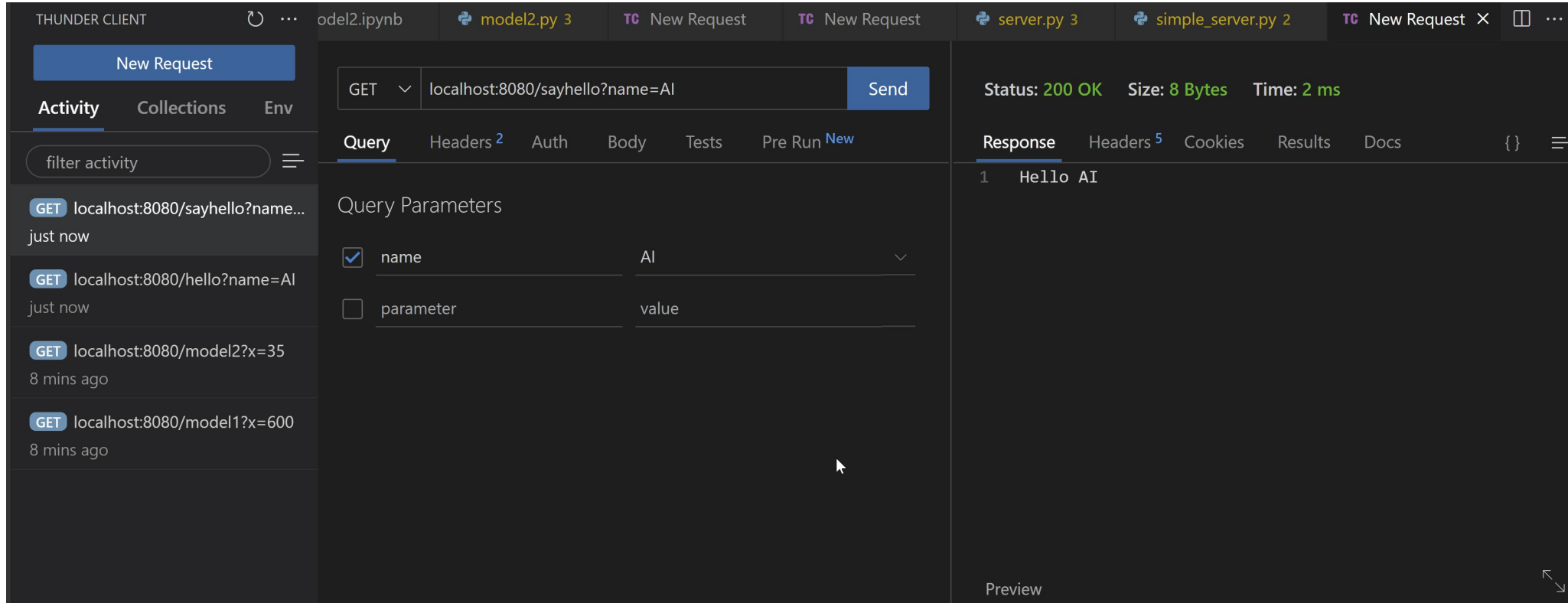
1 Hello Chia

Testing the service using Postman



Testing the Service -2 (cont.)

- You can add extension called 'Thunder Client' in Visual Studio Code to test Get/Post requests.



THUNDER CLIENT

odel2.ipynb model2.py 3 TC New Request TC New Request server.py 3 simple_server.py 2 TC New Request X

New Request

Activity Collections Env

filter activity

GET localhost:8080/sayhello?name=AI just now

GET localhost:8080/hello?name=AI just now

GET localhost:8080/model2?x=35 8 mins ago

GET localhost:8080/model1?x=600 8 mins ago

GET localhost:8080/sayhello?name=AI

Send

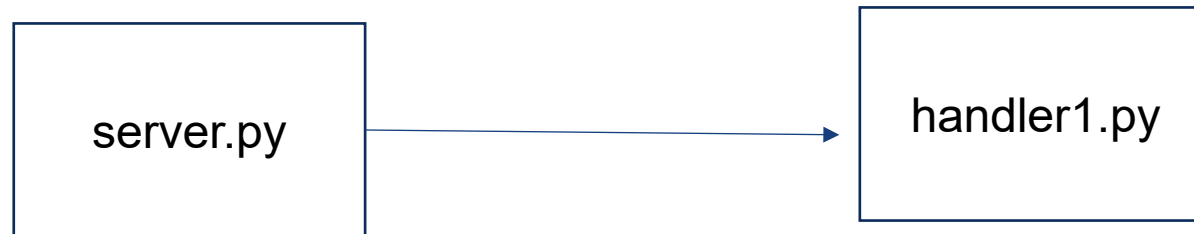
Status: 200 OK Size: 8 Bytes Time: 2 ms

Response Headers 5 Cookies Results Docs

1 Hello AI

Preview

- The codes in the server could refer to another python program file



Putting the code in Separate files -1

- The operations could be coded in separate python files

handler1.py

```
def hello():  
    return "Hello world!"  
  
def say_hello():  
    name = request.args.get('name') or  
           request.form.get('name')  
    return "Hello " + str(name or '')
```

Putting the code in Separate files -2

server.py

```
#. . .  
  
app = Flask(__name__)  
  
app.add_url_rule('/',view_func=handler1.hello)  
  
app.add_url_rule('/sayhello/',  
                view_func=handler1.say_hello, methods=['GET','POST'])  
  
#. . .
```

- Server
- **Model Deployment**

- Ensure you have installed the following packages in your environment
 - flask
 - waitress
 - pickle



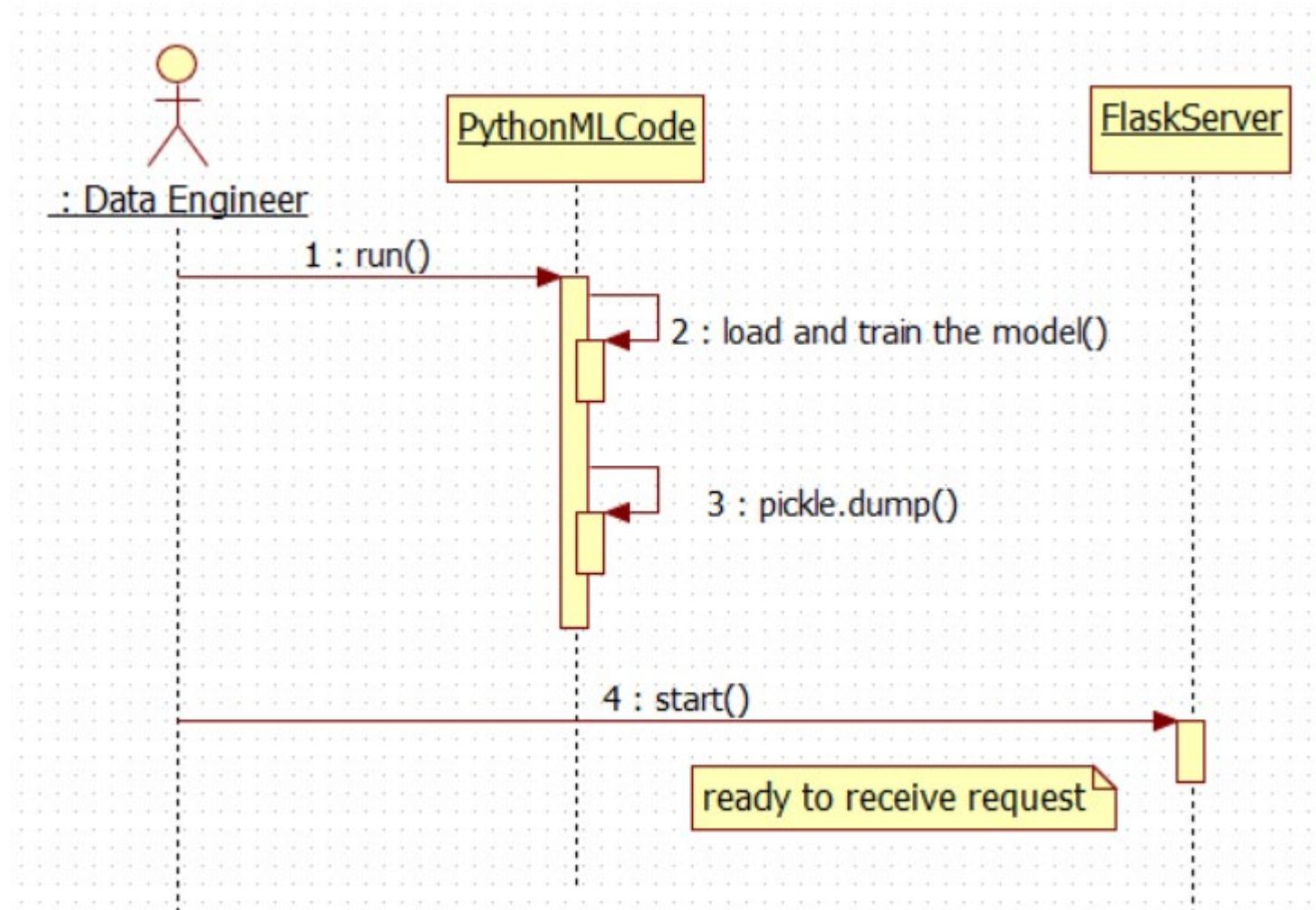
Pickle Module

- The **pickle** module implements binary protocols for serializing and de-serializing a Python object structure.
- “*Pickling*” is the process whereby a Python object hierarchy is converted into a byte stream

<https://docs.python.org/3/library/pickle.html#:~:text=%E2%80%9CPickling%E2%80%9D%20is%20the%20process%20whereby,back%20into%20an%20object%20hierarchy.>

Deployment Workflow

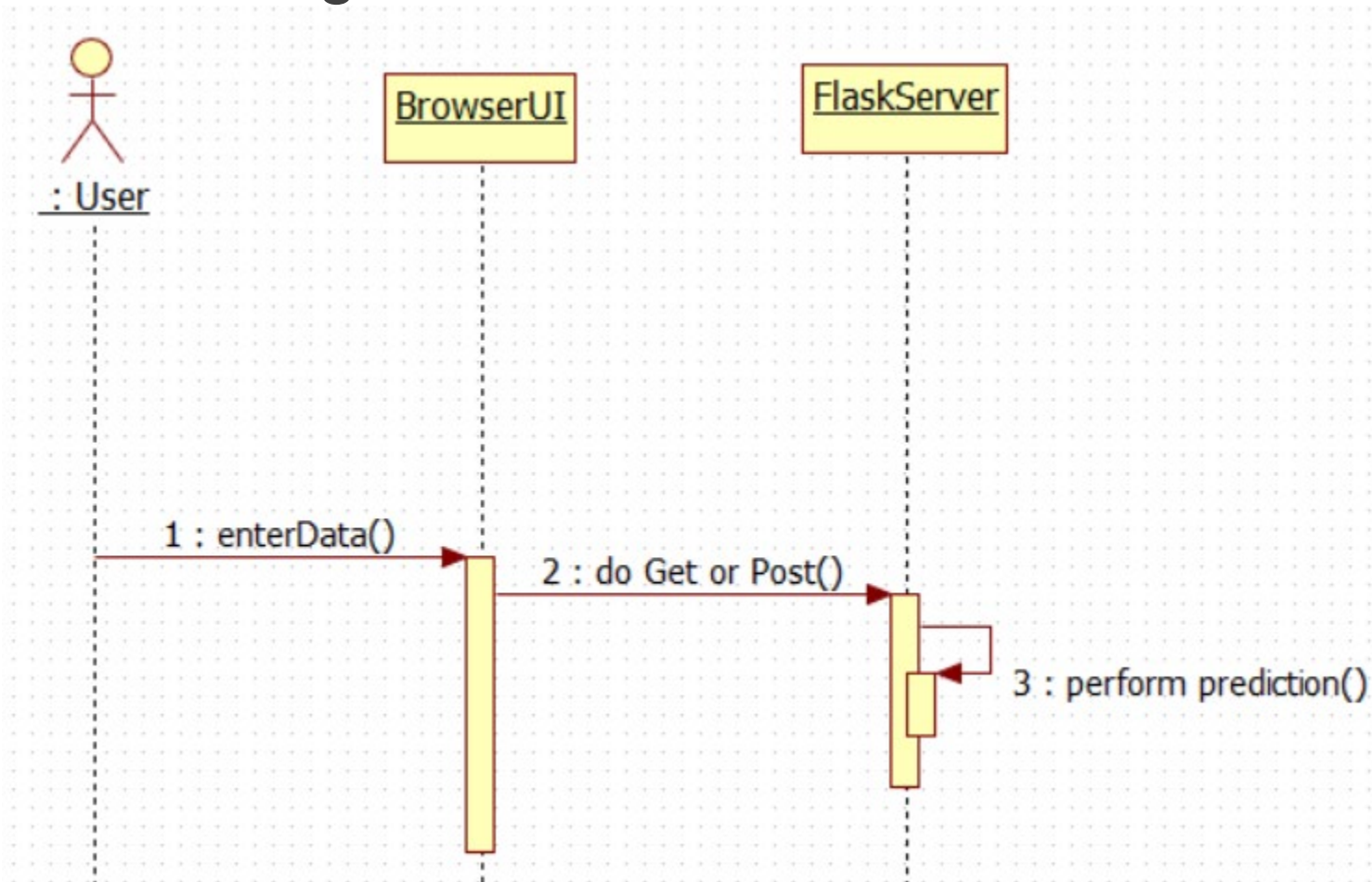
- Serialize the objects through pickle





Executing the Deployed Object

- The user sends a request to the server, the performs the required action and returns a message





Example of Model 1

THUNDER CLIENT

New Request

Activity Collections Env

filter activity

GET localhost:8080/sayhello?name... just now

GET localhost:8080/model2?x=35 8 mins ago

GET localhost:8080/model1?x=600 8 mins ago

GET localhost:8080/model1?x=600

Query Parameters

<input checked="" type="checkbox"/>	x	600
<input type="checkbox"/>	parameter	value

Status: 200 OK Size: 17 Bytes Time: 5 ms

Response Headers Cookies Results Docs

1 6261.733121525021

Preview

- Server
- Model Deployment
 - Flask
 - Waitress
 - Pickle