


Class: System Integration

Exercise: SI_12a_expose_and_integrate_with_a_webhook_system_integrator

Student: Hero Englund

Python script to set up a uvicorn webserver

 integrator_receiver.py x

```
1  from fastapi import FastAPI, Request
2  import uvicorn
3  from collections import deque
4  import datetime
5
6  app = FastAPI()
7  received = deque(maxlen=100)
8
9  @app.post("/webhook-receiver") new *
10 async def receive(req: Request):
11     payload = await req.json()
12     entry = {
13         "timestamp": datetime.datetime.now(datetime.UTC).isoformat() + "Z",
14         "headers": dict(req.headers),
15         "payload": payload,
16     }
17     received.appendleft(entry)
18     print("Webhook received:", entry)
19     return {"status": "ok", "received": entry}
20
21 @app.get("/events") new *
22 def list_events():
23     return {"last_events": list(received)}
24
25 if __name__ == "__main__":
26     uvicorn.run(app="integrator_receiver:app", host="0.0.0.0", port=8000)
```

Run script that sets up webserver

`python integrator_receiver.py`

```
SirKittyH@ROTTIS MINGW64 /f/IntelliJWorkspace/kea-system-integration/SI_12a/
$ python integrator_receiver.py
INFO: Started server process [11612]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

Expose webserver via static Ngrok domain

`ngrok http 8000 --hostname=sunfish-fitting-goat.ngrok-free.app`

```

Session Status      online
Account             XXXXXXXXXX com (Plan: Free)
Update              update available (version 3.25.0, Ctrl-U to update)
Version             3.22.1
Region             Europe (eu)
Latency             25ms
Web Interface       http://127.0.0.1:4040
Forwarding           https://sunfish-fitting-goat.ngrok-free.app -> http://localhost:8000

Connections
ttl    opn    rt1    rt5    p50    p90
0       0       0.00   0.00   0.00   0.00

```

Run CLI commands to integrate with exposed system

usage: webhook_manager.py [-h] [--exposee EXPOSEE] [--callback CALLBACK] [--eventType EVENTTYPE] [--id ID] {register,unregister,ping,list}
webhook_manager.py: error: the following arguments are required: action

CLI actions

```
"action", choices=["register", "unregister", "ping", "list"]
```

CLI defaults

```

"--exposee", default=os.environ.get("EXPOSEE_URL")
"--callback", default=os.environ.get("WEBHOOK_URL")
"--eventType", default="Register"

```

Register webhooks

```
python webhook_manager.py register
```

```

SirKittyH@ROTTIS MINGW64 /f/IntelliJWorkspace/kea
$ python webhook_manager.py register --id 1234
REGISTER: 200 {"response":"Webhook registered"}

```

```

SirKittyH@ROTTIS MINGW64 /f/IntelliJWorkspace/kea
$ python webhook_manager.py register --id 6789
REGISTER: 200 {"response":"Webhook registered"}

```

See list of registered webhooks

```
python webhook_manager.py list
```

```

SirKittyH@ROTTIS MINGW64 /f/IntelliJWorkspace/kea-system-integration/SI_12a_expose_and_integrate_with_a_webhook_system/integrator (main)
$ python webhook_manager.py list
LIST: 200 {"webhooks":[{"id":1754054788,"url":"https://sunfish-fitting-goat.ngrok-free.app/webhook-receiver","eventType":"Register"},{"id":1234,"url":"https://sunfish-fitting-goat.ngrok-free.app/webhook-receiver","eventType":"Register"},{"id":6789,"url":"https://sunfish-fitting-goat.ngrok-free.app/webhook-receiver","eventType":"Register"}]}

```

Ping to trigger calling registered webhooks

```
python webhook_manager.py ping
```

```

SirKittyH@ROTTIS MINGW64 /f/IntelliJWorkspace/kea-:
$ python webhook_manager.py ping
PING: 200 {"message":"Ping sent","resultCount":3}

```

Unregister webhooks

```
python webhook_manager.py unregister --id 1754054586
```

```
SirKittyH@ROTTIS MINGW64 /f/IntelliJWorkspace/kea-system
$ python webhook_manager.py unregister --id 1754054788
JNREGISTER: 200 {"response":"Webhook unregistered"}
```

List command confirms that webhook registration was removed:

```
SirKittyH@ROTTIS MINGW64 /f/IntelliJWorkspace/kea-system-integration/SI_12a_expose_and_integrate_with_a_
webhook_system/integrator (main)
$ python webhook_manager.py list
LIST: 200 {"webhooks":[{"id":1234,"url":"https://sunfish-fitting-goat.ngrok-free.app/webhook-receiver",
eventType":"Register"}, {"id":6789,"url":"https://sunfish-fitting-goat.ngrok-free.app/webhook-receiver",
eventType":"Register"}]}
```

CLI code

```

1  import argparse
2  import httpx
3  import os
4  import sys
5  from dotenv import load_dotenv
6
7  # webhook_manager.py is a thin CLI client -
8  # a convenience wrapper to register, list, ping, and unregister against the exposee API
9  # from the shell instead of hand-crafting curl/httpx calls each time.
10 # It parses the console arguments, builds the appropriate HTTP request to the Heroku exposee,
11 # and prints the response.
12
13 def parse_args(): 1 usage new *
14     p = argparse.ArgumentParser()
15     p.add_argument(*name_or_flags: "action", choices=["register", "unregister", "ping", "list"])
16     p.add_argument(*name_or_flags: "--exposee", help="URL of the exposee service", default=os.environ.get("EXPOSEE_URL"))
17     p.add_argument(*name_or_flags: "--callback", help="public URL the exposee will call", default=os.environ.get("WEBHOOK_URL"))
18     p.add_argument(*name_or_flags: "--eventType", default="Register")
19     p.add_argument(*name_or_flags: "--id", type=int, help="webhook id to remove")
20     return p.parse_args()
21
22 def exit_err(msg): 4 usages new *
23     print(msg, file=sys.stderr)
24     sys.exit(1)
25
26 def post_parse_sanity_check(args): 1 usage new *
27     if not args.exposee:
28         exit_err("exposee URL required (provide --exposee or set EXPOSEE_URL)")
29     if args.action == "register" and not args.id:
30         exit_err("id required for register (provide --id)")
31     if args.action == "register" and not args.callback:
32         exit_err("callback URL required for register (provide --callback or set WEBHOOK_URL)")
33     if args.action == "unregister" and not args.id:
34         exit_err("id required for unregister")
35
36 def register(base, callback_url, event_type, webhook_id): 1 usage new *
37     payload = {"id": webhook_id, "url": callback_url, "eventType": event_type}
38     r = httpx.post(url=f"{base}/webhooks", json=payload, timeout=10)
39     print("REGISTER:", r.status_code, r.text)
40
41 def unregister(base, webhook_id): 1 usage new *
42     r = httpx.delete(url=f"{base}/webhooks/{webhook_id}", timeout=10)
43     print("UNREGISTER:", r.status_code, r.text)
44
45 def ping(base): 1 usage new *
46     r = httpx.post(url=f"{base}/ping", timeout=10)
47     print("PING:", r.status_code, r.text)
48
49 def list_hooks(base): 1 usage new *
50     r = httpx.get(url=f"{base}/webhooks", timeout=10)
51     print("LIST:", r.status_code, r.text)

```

```
53 > if __name__ == "__main__":
54     load_dotenv()
55
56     args = parse_args()
57
58     post_parse_sanity_check(args)
59
60     exp = args.exposee.rstrip("/")
61
62     if args.action == "register":
63         register(exp, args.callback, args.eventType, args.id)
64     elif args.action == "unregister":
65         unregister(exp, args.id)
66     elif args.action == "ping":
67         ping(exp)
68     elif args.action == "list":
69         list_hooks(exp)
```