

Swift 디버깅 시작하기

System Engineering Laboratory

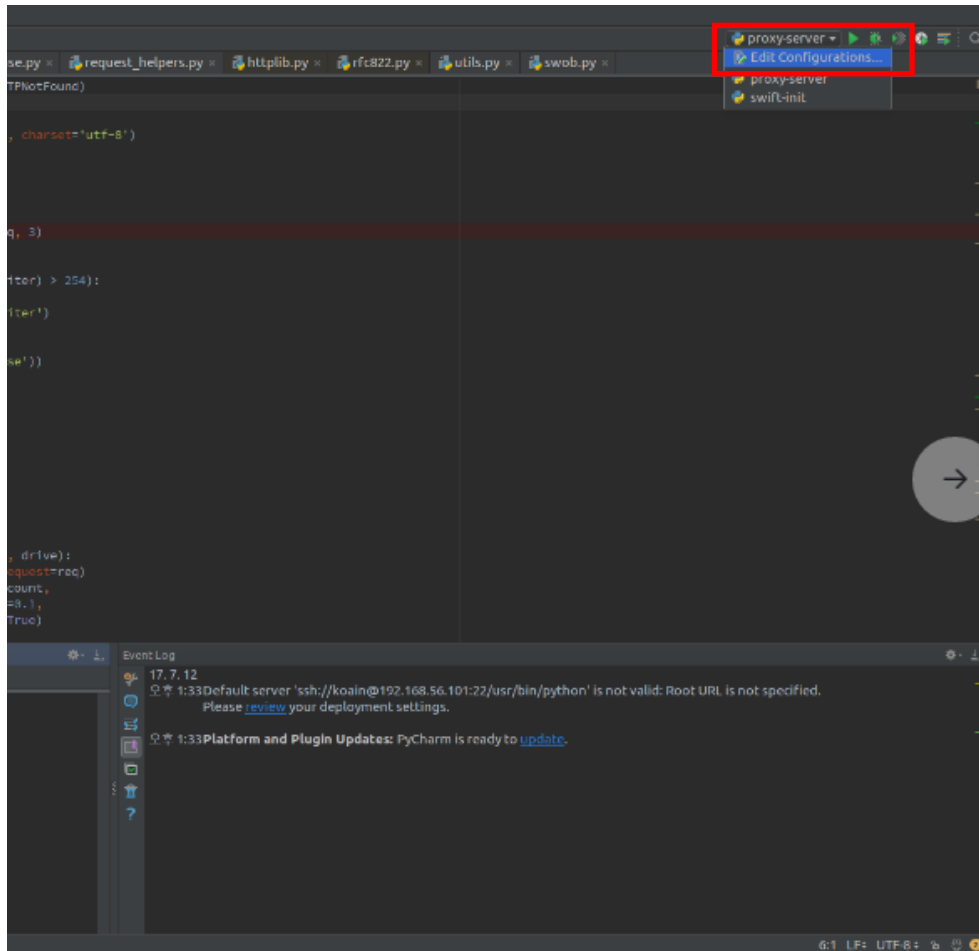
koain@naver.com

Kim Young Woo

1. 사전 준비 하기

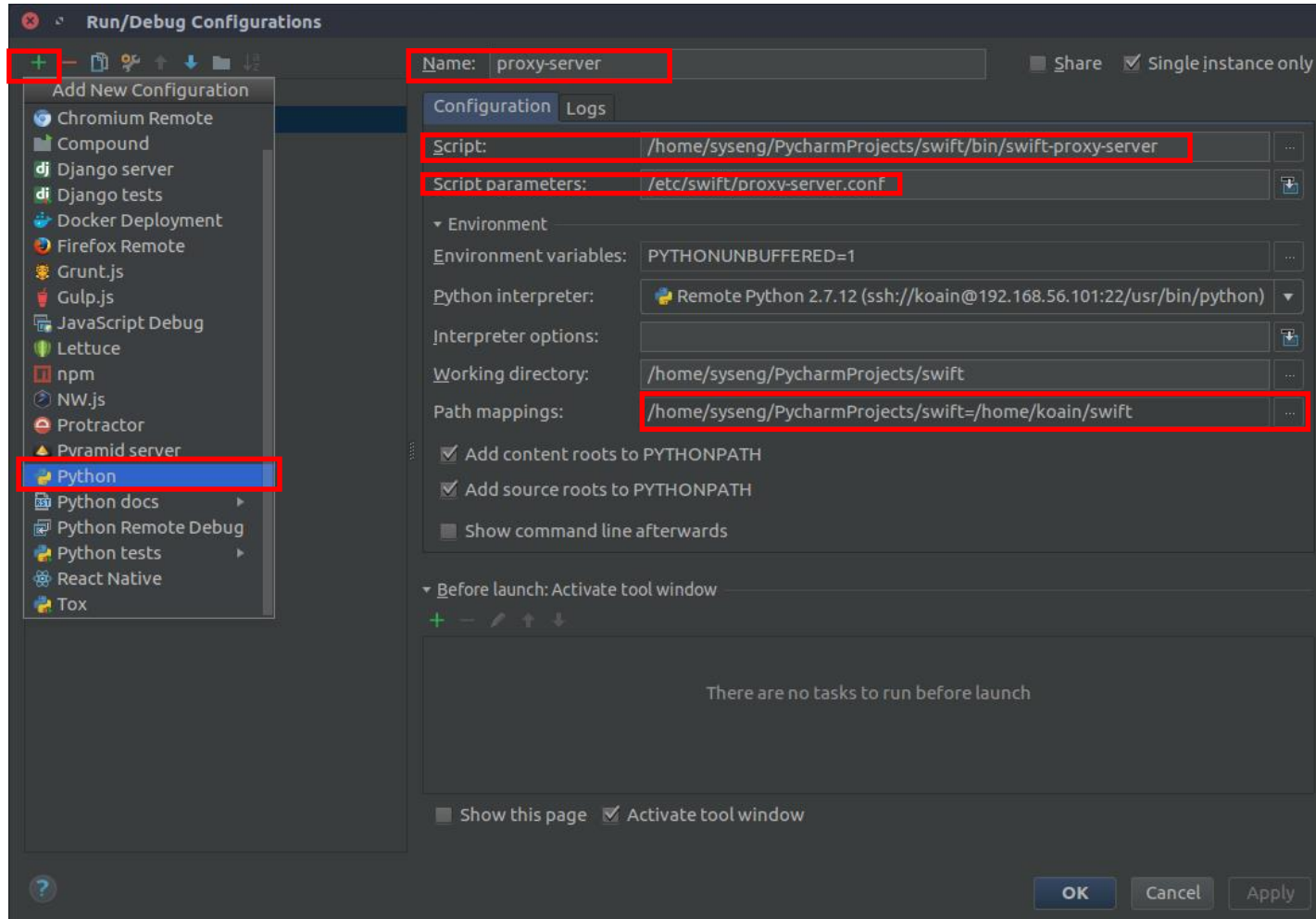
- SAIO 설치
 - 수동 설치 - https://docs.openstack.org/swift/latest/development_saio.html
 - DevStack - <https://github.com/openstack-dev/devstack/#swift>
- 파이참 설치
 - 프로페셔널 에디션(원격 디버깅) - <https://www.slideshare.net/koain/ss-75453424>
 - 커뮤니티 에디션(로컬 디버깅) - SAIO 설치 머신에 파이참 설치

2. 실행 파일 설정



- 파이참 우 상단의 Edit Configuration 클릭

2. 실행 파일 설정



- 좌 상단 + 버튼 클릭
- Python 선택
- 이름 지정
- Script 지정 (로컬 머신에서 실행할 파일)
- Script Parameters 지정 (설정파일 지정)
- Path mappings 지정 (로컬과 원격 머신의 경로 매핑)

3. swift 데몬 띄우기

```
koaln@ywkim-salo: ~$ swift-init all shutdown
No container-updater running
No account-auditor running
No object-replicator running
No container-sync running
No container-replicator running
No object-auditor running
No object-expirer running
No container-auditor running
Signal container-server pid: 9536 signal: 1
Signal container-server pid: 9537 signal: 1
Signal container-server pid: 9538 signal: 1
Signal container-server pid: 9539 signal: 1
No object-reconstructor running
Signal object-server pid: 9544 signal: 1
Signal object-server pid: 9545 signal: 1
Signal object-server pid: 9546 signal: 1
Signal object-server pid: 9547 signal: 1
No account-reaper running
No proxy-server running
No account-replicator running
No object-updater running
No container-reconciler running
Signal account-server pid: 9540 signal: 1
```

- 떠있는 swift 데몬들 다 죽이기
- \$ swift-init all shutdown

3. swift 데몬 띄우기

```
koain@ywkim-saio: ~$ swift-init main start
Starting proxy-server...(etc/swift/proxy-server.conf)
Starting container-server...(etc/swift/container-server/1.conf)
Starting container-server...(etc/swift/container-server/2.conf)
Starting container-server...(etc/swift/container-server/3.conf)
Starting container-server...(etc/swift/container-server/4.conf)
Starting account-server...(etc/swift/account-server/1.conf)
Starting account-server...(etc/swift/account-server/2.conf)
Starting account-server...(etc/swift/account-server/3.conf)
Starting account-server...(etc/swift/account-server/4.conf)
Starting object-server...(etc/swift/object-server/1.conf)
Starting object-server...(etc/swift/object-server/2.conf)
Starting object-server...(etc/swift/object-server/3.conf)
Starting object-server...(etc/swift/object-server/4.conf)
```

- Swift 데몬들 모두 시작하기
- \$ swift-init main start

3. swift 데몬 띄우기

```
koain@ywkim-saio: ~
koain@ywkim-saio:~$ ps -ef | grep swift-proxy
koain  15791  1  1 14:38 ? 00:00:03 /usr/bin/python /usr/local/bin/swift-proxy-server /etc/swift/
proxy-server.conf
koain  15854 15791  0 14:38 ? 00:00:00 /usr/bin/python /usr/local/bin/swift-proxy-server /etc/swift/
proxy-server.conf
koain  15964 14575  0 14:42 pts/21 00:00:00 grep --color=auto swift-proxy
koain@ywkim-saio:~$ kill -9 15791
koain@ywkim-saio:~$ ps -ef | grep swift-proxy
koain  15854  1  0 14:38 ? 00:00:00 /usr/bin/python /usr/local/bin/swift-proxy-server /etc/swift/
proxy-server.conf
koain  15966 14575  0 14:43 pts/21 00:00:00 grep --color=auto swift-proxy
koain@ywkim-saio:~$ kill -9 15854
koain@ywkim-saio:~$ ps -ef | grep swift-proxy
koain  15968 14575  0 14:43 pts/21 00:00:00 grep --color=auto swift-proxy
koain@ywkim-saio:~$
```

- 파이참으로 proxy-server를 실행시킬 것이므로 proxy-server만 죽임
- \$ ps -ef | grep swift-proxy # pid 확인
- \$ kill -9 PID # 프로세스 죽이기

4. Swift API 맛보기 - URL과 TOKEN 얻기

```
koain@ywkim-saio: ~
koain@ywkim-saio:~$ curl -v -H 'X-Storage-User: test:tester' -H 'X-Storage-Pass: testing' http://127.0.0.1:8080/auth/v1.0
* Trying 127.0.0.1...
* Connected to 127.0.0.1 (127.0.0.1) port 8080 (#0)
> GET /auth/v1.0 HTTP/1.1
> Host: 127.0.0.1:8080
> User-Agent: curl/7.47.0
> Accept: */*
> X-Storage-User: test:tester
> X-Storage-Pass: testing
>
< HTTP/1.1 200 OK
< X-Storage-Url: http://127.0.0.1:8080/v1/AUTH_test
< X-Auth-Token-Expires: 86399
< X-Auth-Token: AUTH_tk551c77919bc94060975e41cacb95e36
< Content-Type: text/html; charset=UTF-8
< X-Storage-Token: AUTH_tk551c77919bc94060975e41cacb95e36
< Content-Length: 0
< X-Trans-Id: tx62d04ce479bf4cb7849a1-005965b7b5
< X-Openstack-Request-Id: tx62d04ce479bf4cb7849a1-005965b7b5
< Date: Wed, 12 Jul 2017 05:48:49 GMT
* Connection #0 to host 127.0.0.1 left intact
koain@ywkim-saio:~$
```

- Swift를 이용하기 위한 URL과 TOKEN 얻기
- account : test
- username : tester
- password : testing
- X-Storage-Url와 X-Auth-Token를 얻을 수 있다.
- \$ curl -v -H 'X-Storage-User: test:tester' -H 'X-Storage-Pass: testing' http://127.0.0.1:8080/auth/v1.0

4. Swift API 맛보기 - container 확인하기

```
koain@ywkim-saio:~$ AUTH_TOKEN=AUTH_tk551c77919bc94060975e41caccb95e36
koain@ywkim-saio:~$ BASE_URL=http://127.0.0.1:8080/v1/AUTH_test
koain@ywkim-saio:~$ curl -v -H 'X-Auth-Token: "$AUTH_TOKEN"' $BASE_URL
* Trying 127.0.0.1...
* Connected to 127.0.0.1 (127.0.0.1) port 8080 (#0)
> GET /v1/AUTH_test HTTP/1.1
> Host: 127.0.0.1:8080
> User-Agent: curl/7.47.0
> Accept: */*
> X-Auth-Token: AUTH_tk551c77919bc94060975e41caccb95e36
>
< HTTP/1.1 200 OK
< X-Account-Storage-Policy-Gold-Bytes-Used: 0
< Content-Length: 6
< X-Account-Storage-Policy-Gold-Object-Count: 0
< X-Account-Object-Count: 0
< X-Timestamp: 1496110908.21361
< X-Account-Storage-Policy-Gold-Container-Count: 1
< X-Account-Bytes-Used: 0
< X-Account-Container-Count: 1
< Content-Type: text/plain; charset=utf-8
< Accept-Ranges: bytes
< X-Trans-Id: tx178da7ff174a4ba989afd-005965bee8
< X-Openstack-Request-Id: tx178da7ff174a4ba989afd-005965bee8
< Date: Wed, 12 Jul 2017 06:17:16 GMT
<
cont1
* Connection #0 to host 127.0.0.1 left intact
koain@ywkim-saio:~$
```

- Account에 대해 GET
- 해당 계정에 존재하는 Container 목록이 출력됨
- `$ curl -X GET -v -H 'X-Auth-Token: "$AUTH_TOKEN"' $BASE_URL`

4. Swift API 맛보기 - Object 확인하기

```
koain@ywkim-saio: ~
koain@ywkim-saio:~$ curl -i $BASE_URL/cont1 -X GET -H 'X-Auth-Token: "$AUTH_TOKEN"'
HTTP/1.1 200 OK
Content-Length: 91
X-Container-Object-Count: 9
Accept-Ranges: bytes
X-Storage-Policy: gold
Last-Modified: Tue, 30 May 2017 02:22:36 GMT
X-Container-Bytes-Used: 269
X-Timestamp: 1496110908.24661
Content-Type: text/plain; charset=utf-8
X-Trans-Id: tx95a89726e21b4faa8c860-005965c047
X-Openstack-Request-Id: tx95a89726e21b4faa8c860-005965c047
Date: Wed, 12 Jul 2017 06:23:05 GMT

dloobj
dloobj/obj0
dloobj/obj1
dloobj/obj2
obj0
sloobj
sloobj/obj0
sloobj/objA
sloobj/objE
koain@ywkim-saio:~$
```

- Container에 대해 GET
- 해당 Container 내부의 Object의 목록이 출력됨
- `$ curl -X GET -v $BASE_URL/cont1 -H 'X-Auth-Token: "$AUTH_TOKEN"'`

4. Swift API 맛보기 - Object 업로드

```
koain@ywkim-saio: ~/swift_data
koain@ywkim-saio:~/swift_data$ echo "12345678" > testData
koain@ywkim-saio:~/swift_data$ ls
testData
koain@ywkim-saio:~/swift_data$ curl -X PUT -i $BASE_URL/cont1/objTestData -T 'testData' -H 'X-Auth-Token: '$AUTH_TOKEN''
HTTP/1.1 201 Created
Last-Modified: Wed, 12 Jul 2017 07:54:50 GMT
Content-Length: 0
Etag: 23cdc18507b52418db7740cbb5543e54
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx6a4174f848fe49188499f-005965d5c9
X-Openstack-Request-Id: tx6a4174f848fe49188499f-005965d5c9
Date: Wed, 12 Jul 2017 07:54:49 GMT
X-Auth-Token: '$AUTH_TOKEN'

koain@ywkim-saio:~/swift_data$ curl -X GET -i $BASE_URL/cont1/ -H 'X-Auth-Token: '$AUTH_TOKEN''
HTTP/1.1 200 OK
Content-Length: 12
X-Container-Object-Count: 1
Accept-Ranges: bytes
X-Storage-Policy: gold
Last-Modified: Tue, 30 May 2017 02:22:36 GMT
X-Container-Bytes-Used: 9485
X-Timestamp: 1496110908.25511
Content-Type: text/plain; charset=utf-8
X-Trans-Id: txba456c09a3284a4fa864d-005965d5d0
X-Openstack-Request-Id: txba456c09a3284a4fa864d-005965d5d0
Date: Wed, 12 Jul 2017 07:54:56 GMT

objTestData
koain@ywkim-saio:~/swift_data$
```

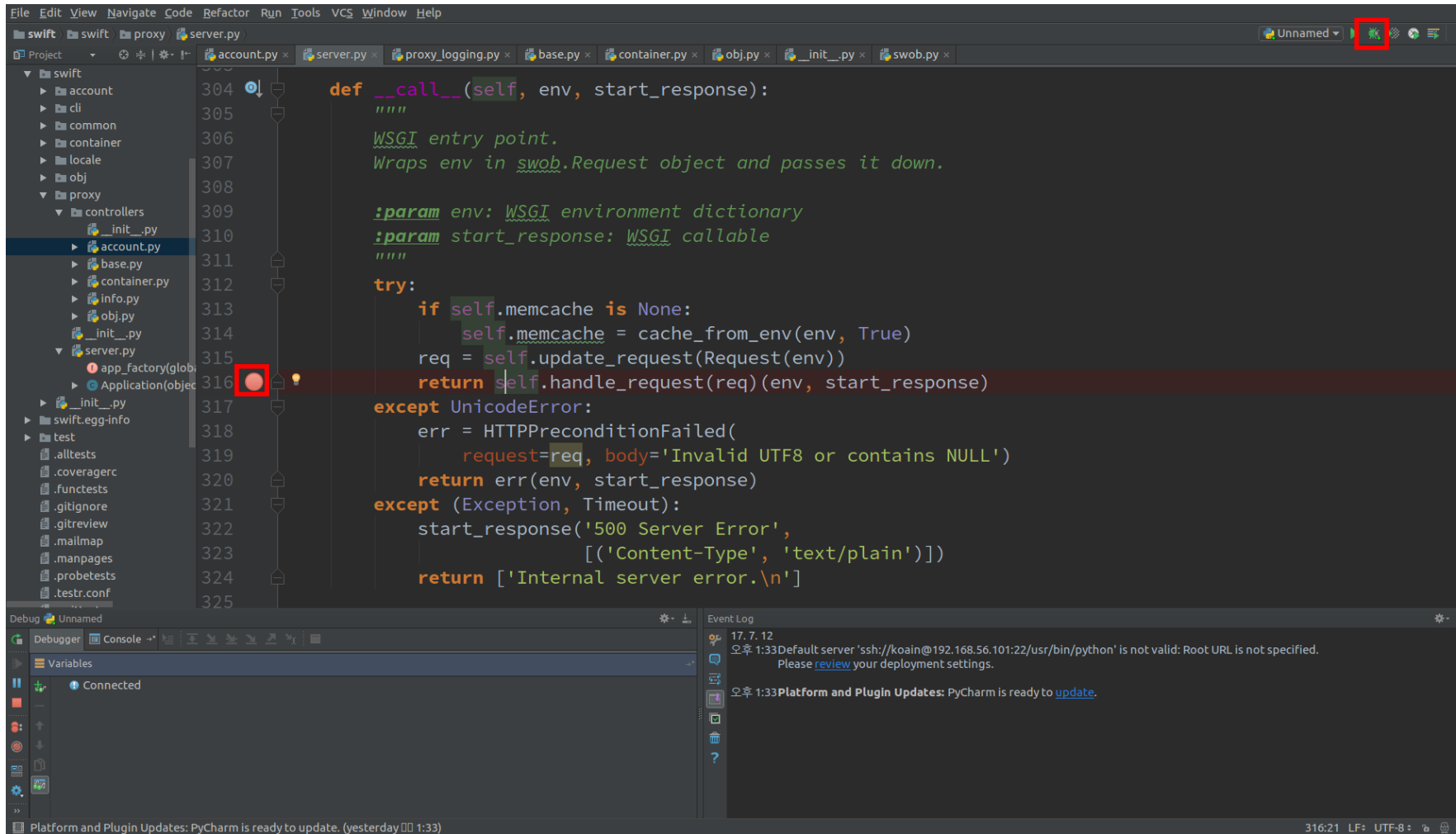
- Object를 PUT
- 파일을 생성하고 이를 Swift에 업로드
- `curl -X PUT -i $BASE_URL/cont1/testObj1 -T 'TestFile' -H 'X-Auth-Token: "$AUTH_TOKEN'"`

5. Proxy-server 디버깅

- WSGI(Web Server Gateway Interface) : HTTP를 통해 요청을 받아 응답하는 어플리케이션에 대한 명세로 이러한 명세를 만족시키는 클래스나 함수,(__call__을 통해 부를 수 있는)객체를 WSGI 어플리케이션 이라고 한다.
- Middleware : WSGI 자체는 서버가 어플리케이션과 통신하는 명세를 다룬다. 따라서 추가적인 기능은 미들웨어로 작성한다. (Swift, DLO, SLO, ACL ... swift/swift/common/middleware에 존재)
- swift/swift/proxy/server.py의 Application 클래스 __call__ 함수부터 디버깅 시작

<https://spoqa.github.io/2012/01/16/wsgi-and-flask.html>

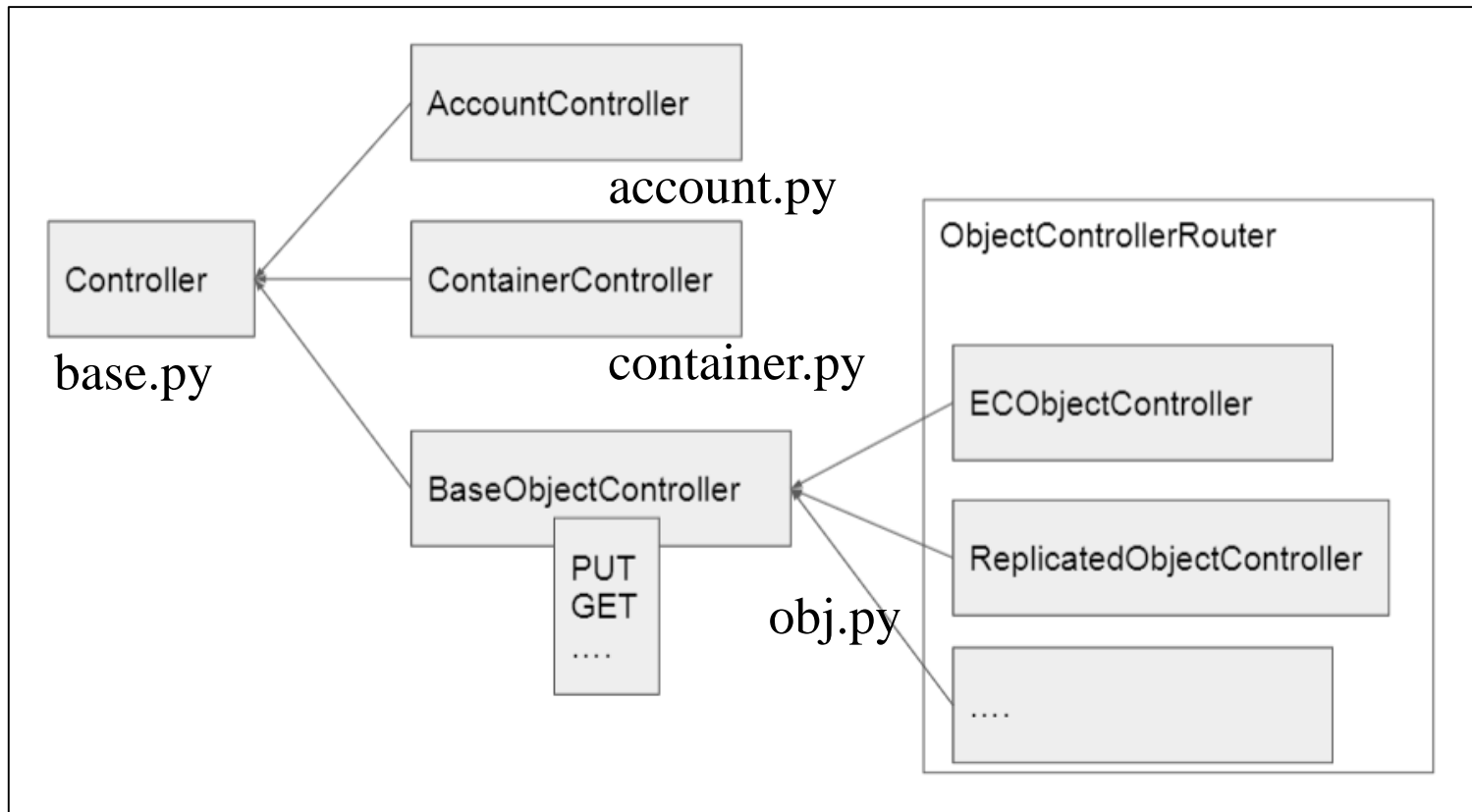
5. Proxy-server 디버깅 - Object에 대한 GET 분석



- swift/swift/proxy/server.py
- 중단점 걸고 딱정벌레 클릭
- 서버가 뜨고 API 보내면 중단점에 걸림
- 디버깅 시작

5. Proxy-server 디버깅 - Object에 대한 GET 분석

swift/swift/proxy/controllers



- 각 컴포넌트 별 처리 Controller