Lightweight Web App Framework for C++

Mengdi Lin
Yasutaka Tanaka
Yi Qi

2017 @ Columbia University CS4995 Project

Lightweight Web App Framework for C++

Cjango Unchained

Mengdi Lin
Yasutaka Tanaka
Yi Qi

2017 @ Columbia University CS4995 Project

4/27/17

Acknowledgments

Professor Bjarne Stroustrup

Jonathan Barrios

David (his explanation on HexRacer gave us ideas about using Json)

Creators of web frameworks including Django, Crow, Silicon, treeFrog

Library Developers of nlohmann/json, simplefilewatcher

What is Web Application Framework (Framework)?

Web App Framework

- provides an abstraction (framework) for faster&easier web developement
- runs an application by `app.run(80)` instead of writing low-level socket handlings
- URL mappings to **callback functions** by predefined precedences
  - e.g. if URL is like "/diary/[0-9]{4}/[0-9]{1,2}", then call `render_diary_page()`
- accesses user's HTTP Request contents through Cjango::`HttpRequest` class
- automate database schema changes
- etc, etc.

Common Web App Frameworks

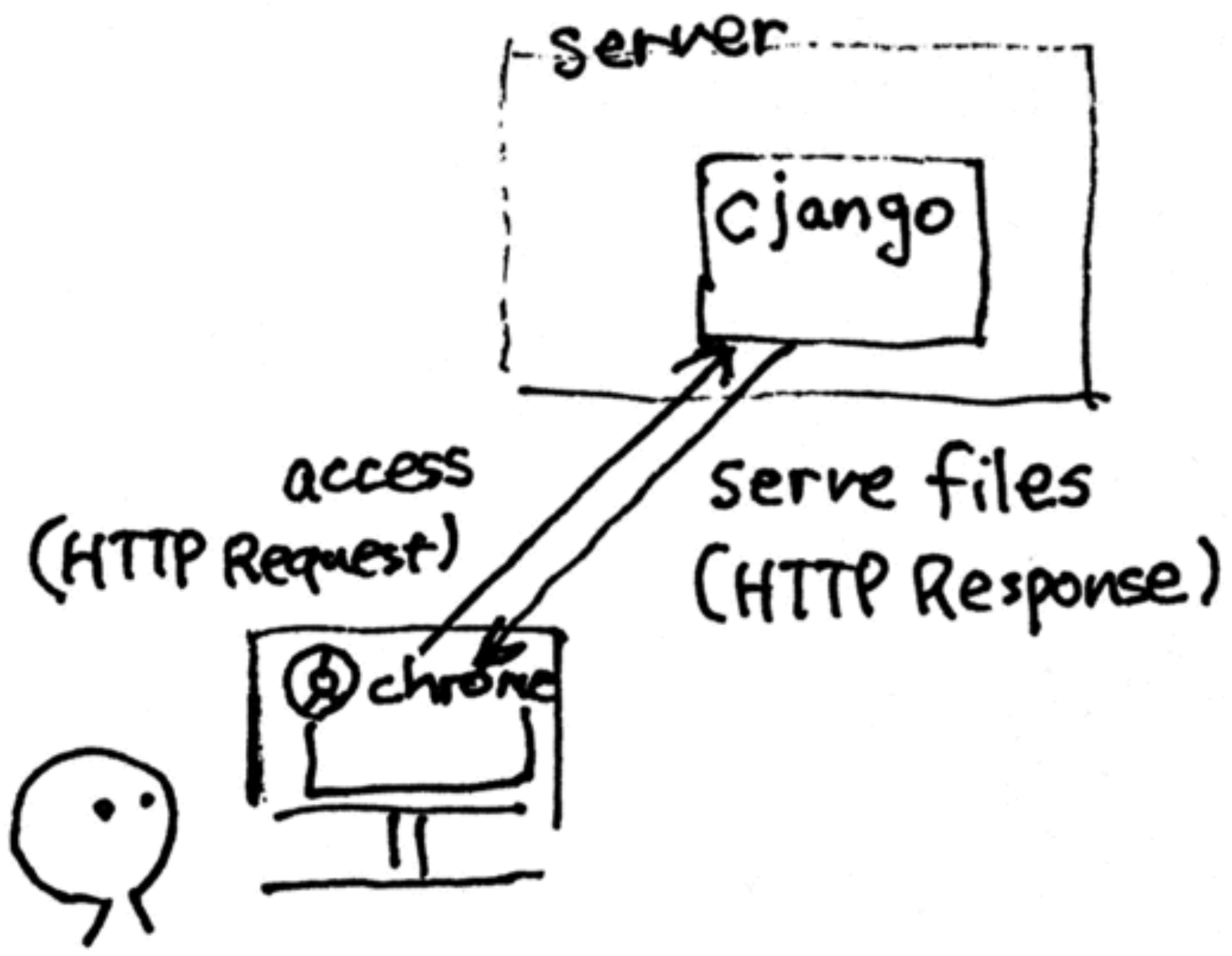
- exist in various languages
- In Python: Flask web development, one drop at a time (20K+ Github stars), django (Python, 20K+), …
- In C++: Crow (a **Flask** clone, 3K+ stars), Silicon (1K+ stars)

What is Cjango-Unchained? Why do we need yet another framework?

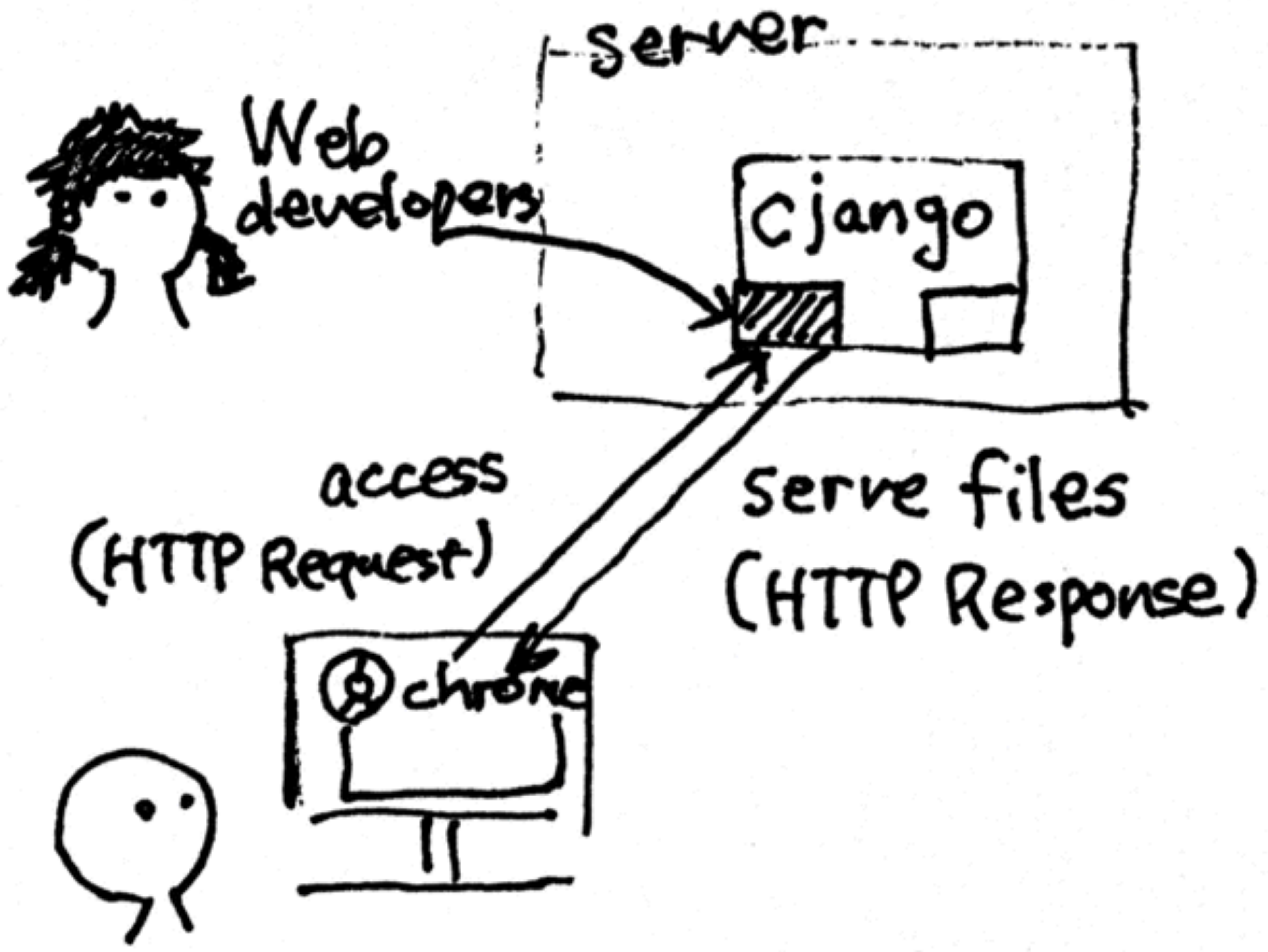
Cjango-Unchained (Cjango) is a lightweight C++ web app framework

- provides a Django-like abstraction (framework) on top of C++
- features
    - full compatibility for HTTP 1.0 (GET/POST)
    - HTTP Session support (thread-safe)
    - sqlite3 / template-engine support
- **high-speed**
    - response speed: **~30% faster than Django (Python)** by async req. handling
        - comparable when compared with other C++ frameworks
    - development speed: boosted by **dynamic callback loading**
        - new callback functions can be loaded at runtime
        - callbacks can be compiled separately
- **user-friendly**
    - Crow (Github Star #1) doesn't have a tutorial :(
    - easy-to-use loggers (log-level and category)
    - auto-set Make compile commands

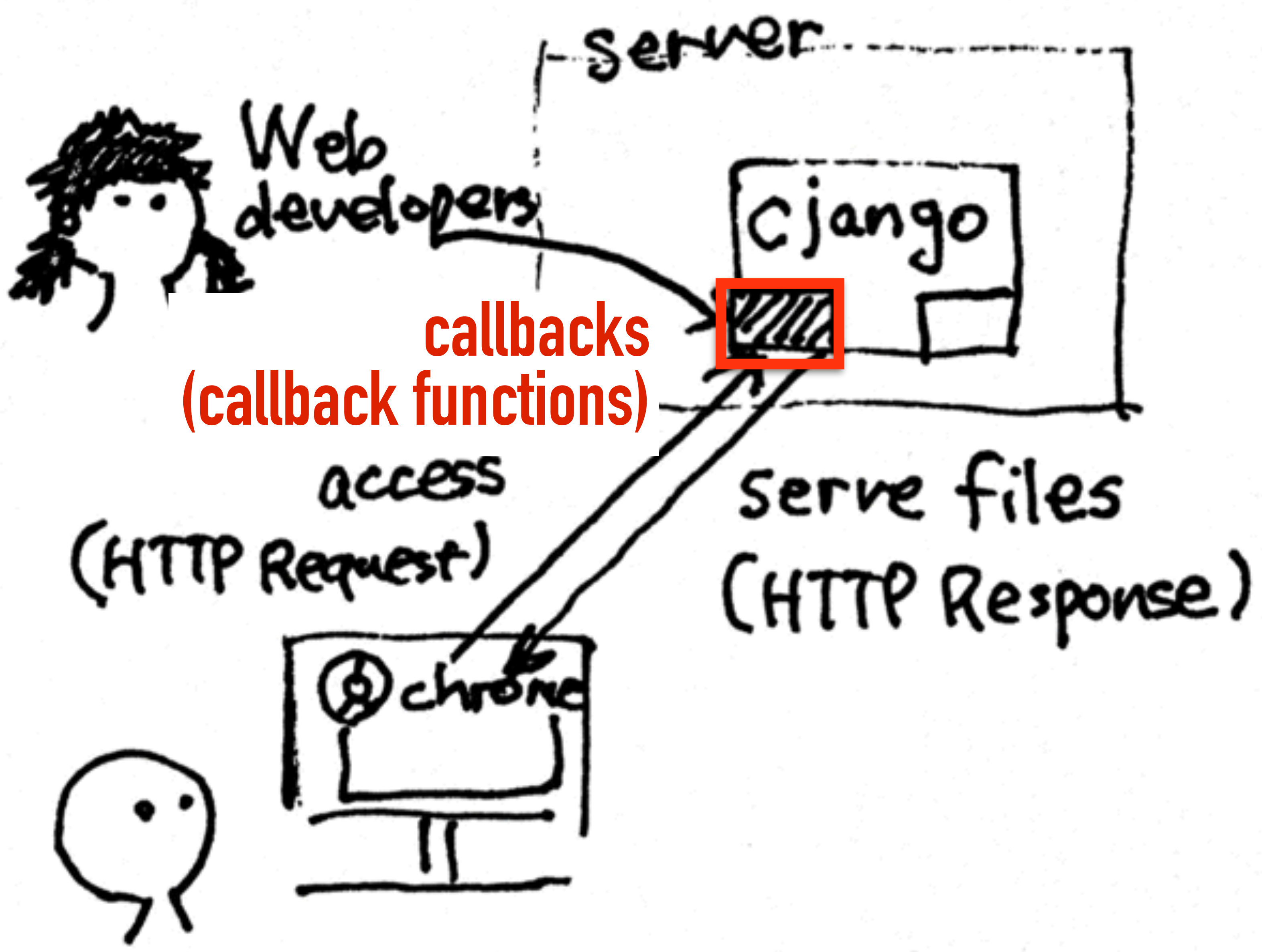
High-Level Sketch (Abstraction) of Cjango's Benefits



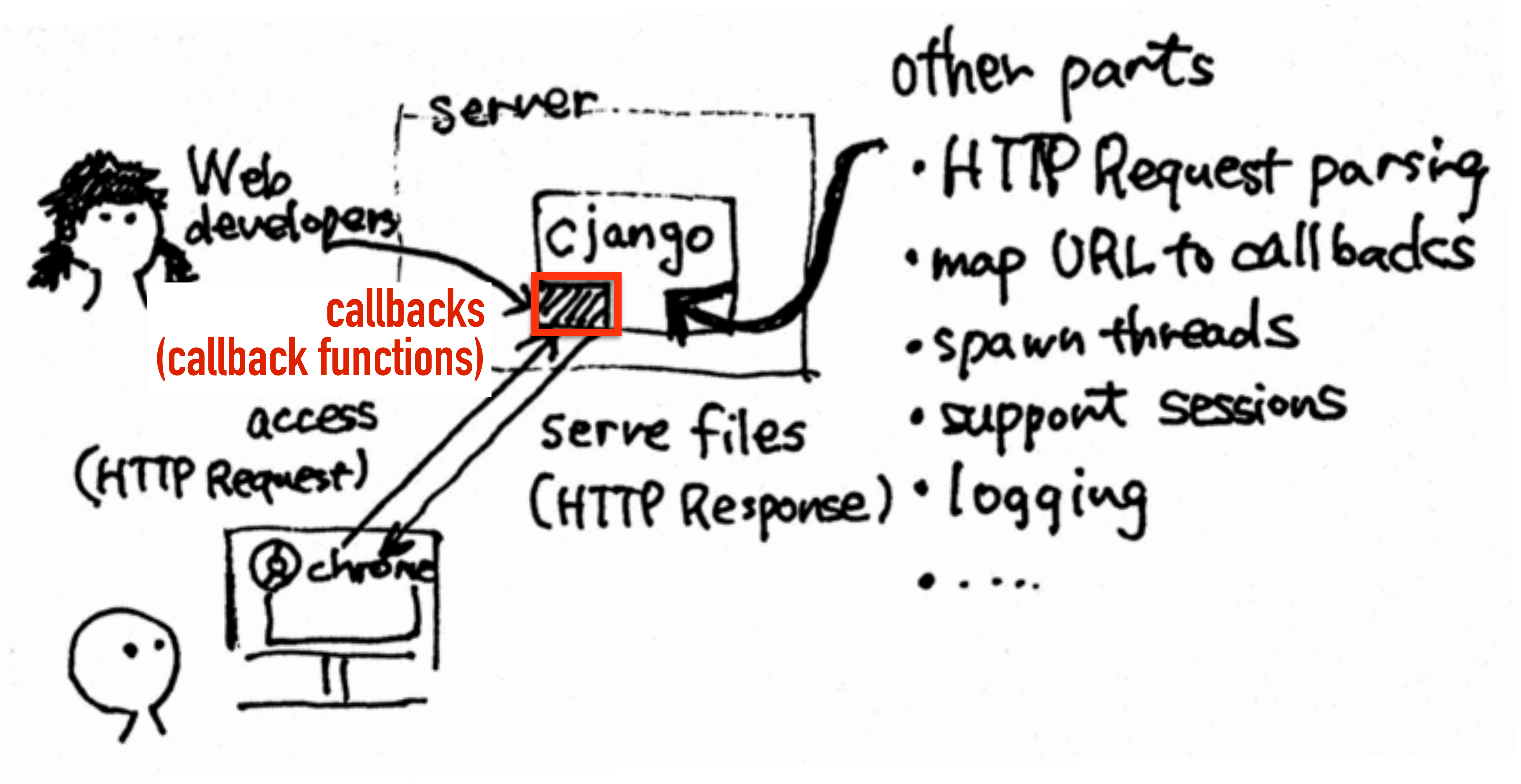
High-Level Sketch (Abstraction) of Cjango's Benefits



High-Level Sketch (Abstraction) of Cjango's Benefits



High-Level Sketch (Abstraction) of Cjango's Benefits



Cjango's Design Philosophy

"Unchained"

:= Everything should be achieved without frustrating restrictions

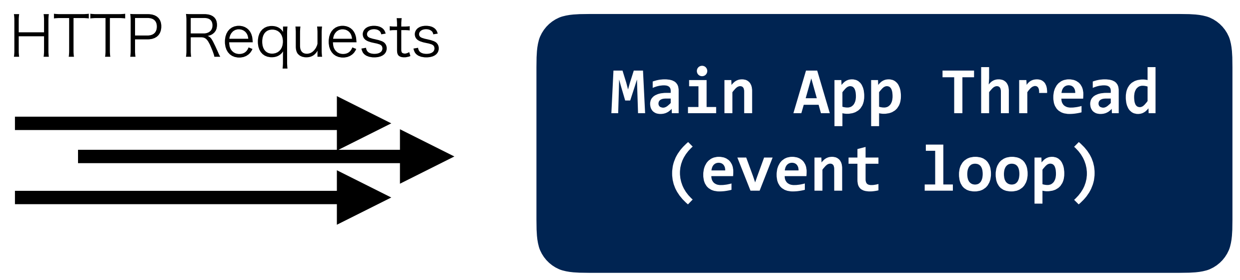
- X should be fast
  - parsing requests
  - HTTP responses
  - function unit tests
  - compile time (c.f. #1 Crow)

- X can be customized by user needs
  - URL patterns
  - directory hierarchy

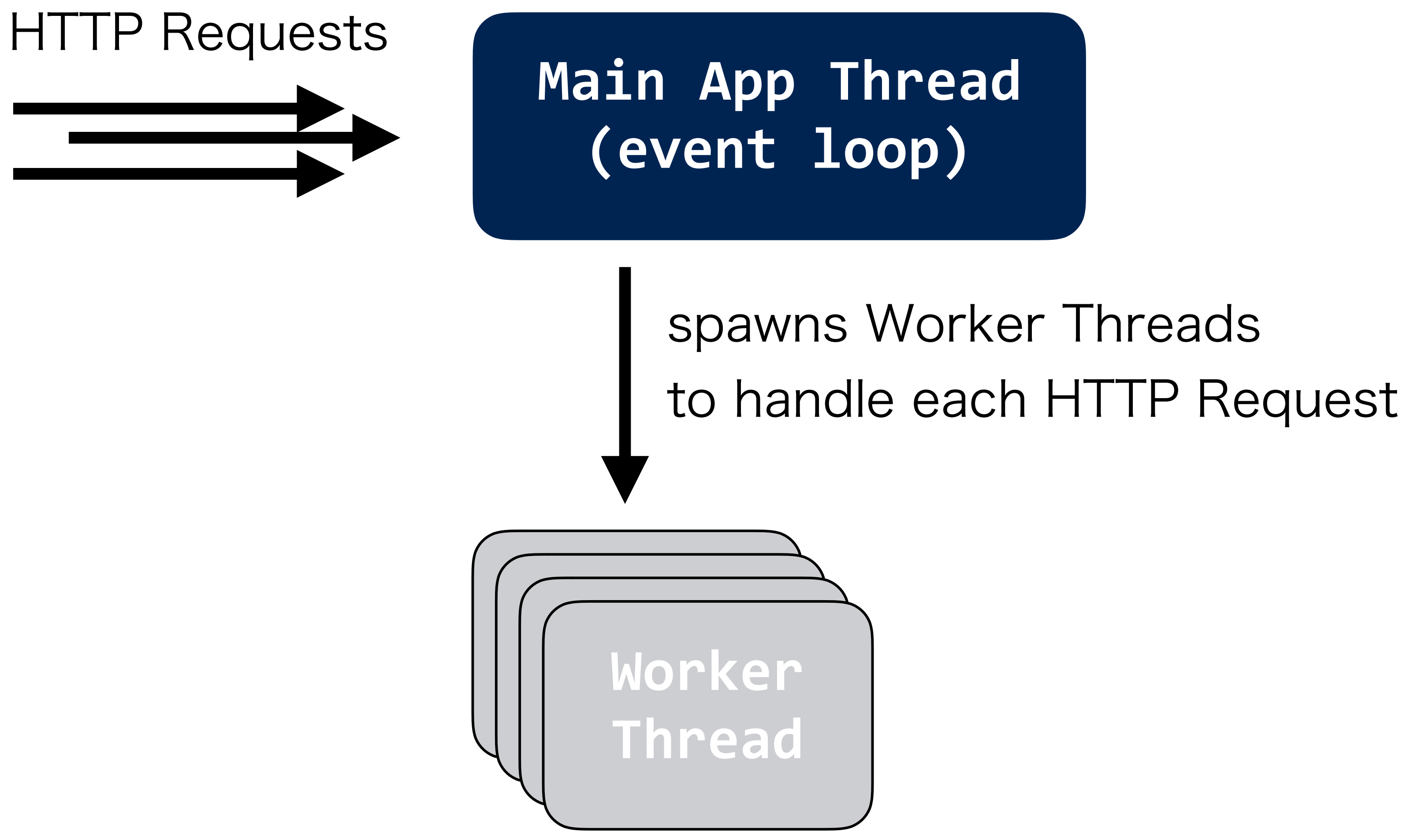
Cjango's main execution flow

Main App Thread
(event loop)

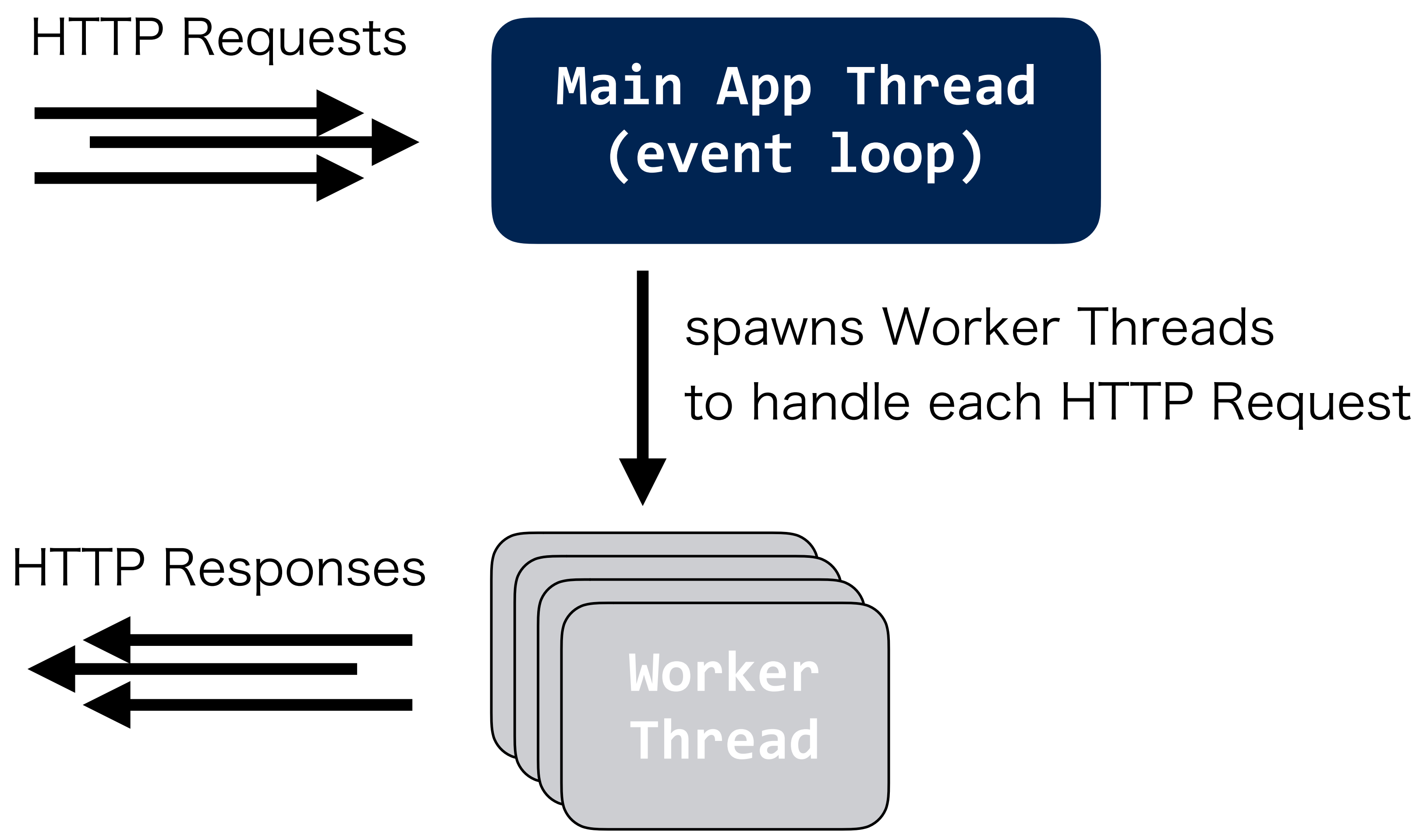
Cjango's main execution flow



Cjango's main execution flow



Cjango's main execution flow



API Comparisons: Running App

```
python manage.py runserver 8000
```

or

```
./manage runserver 8000
```

- manage.py is a wrapper script just to call ./manage inside
  - follows django's interface

```
python manage.py runserver 8000
```

Github: mengdilin/Cjango-Unchained

API Comparisons: Setting URL to a callback

hello.cpp → hello.so (by Make rules)

```
extern "C" auto hello_world(HttpRequest req) {
  return HttpResponse("helloWorld");
}
```

urls.json

```
{
  "/hello": {
    "file": "hello.so",
    "funcname": "hello_world"
  }
}
```

hello.py

```
def hello_world(request):
  return HttpResponse("HelloWorld")
```

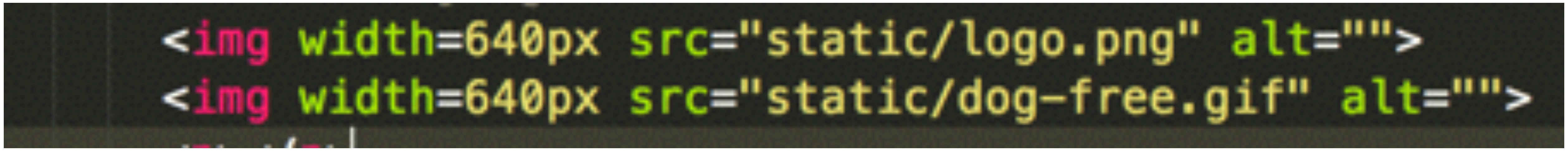
urls.py

```
urlpatterns = [
    url(r'^hello/', hello_world)
]
```

- extern "C" directive is required for combining C's library
- When saved, **Mappings in urls.json are automatically reloaded as urls.py**
  - If hello.so changed, "`touch urls.json`" reloads the new `hello_world()`
- both of urls.json and urls.py can use Regex for pattern matchings
- None of Crow/Silicon take this runtime-loading approach

Static file routings are auto-generated

official_page.html



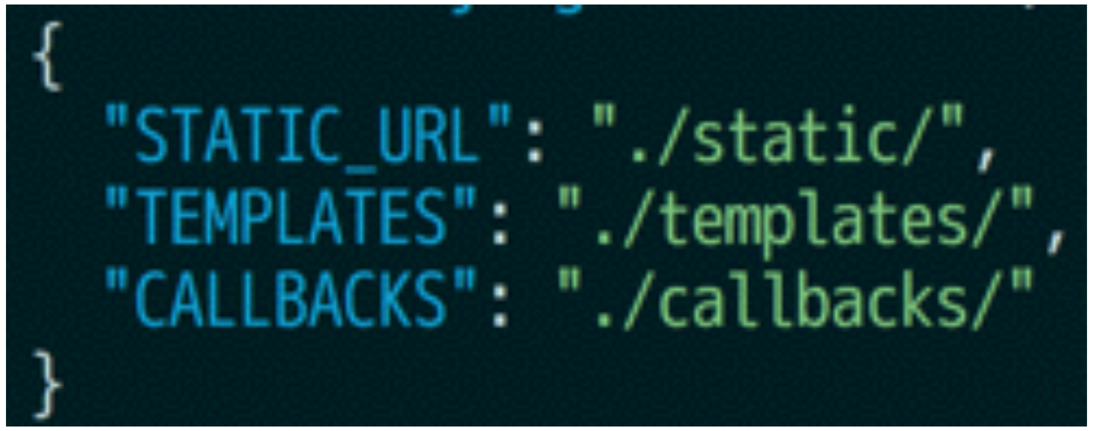
```
        <img width=640px src="static/logo.png" alt="">
        <img width=640px src="static/dog-free.gif" alt="">
```

Files under static/ can be accessed without URL mappings

- Cjango generates rules automatically
- users can customize the static file directory path
  (an example setting in the next slide)

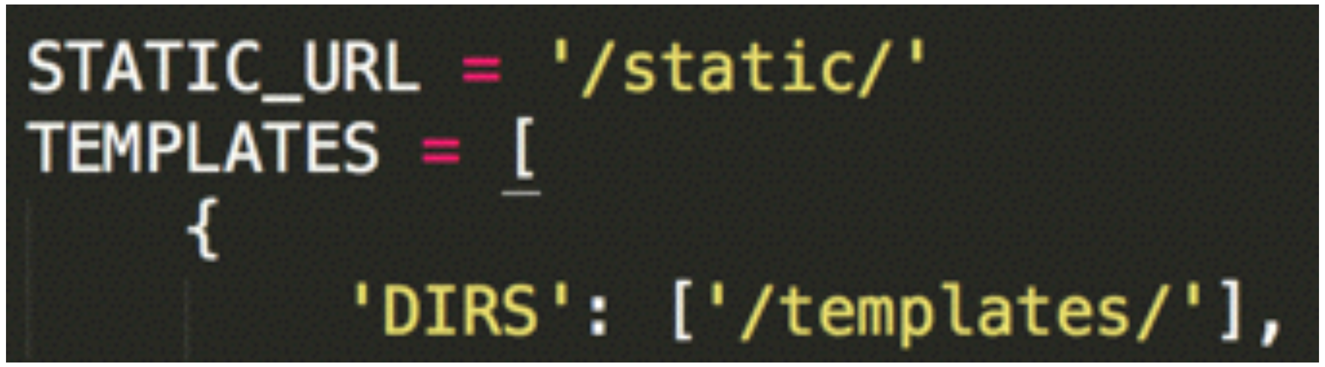
API Comparisons: Setting Custom Static/Template Root Foloders

settings.json



```
{
   "STATIC_URL": "./static/",
   "TEMPLATES": "./templates/",
   "CALLBACKS": "./callbacks/"
}
```

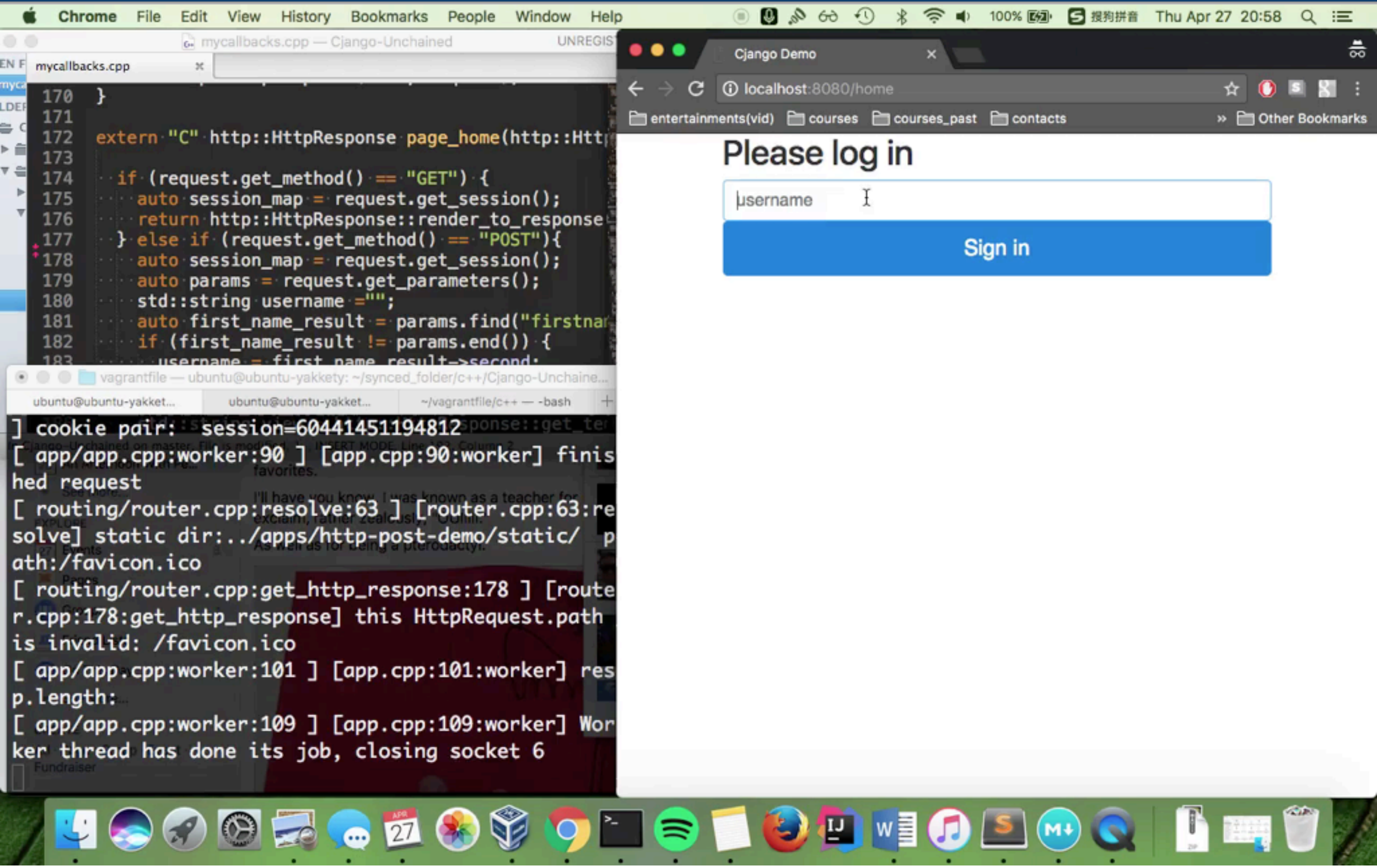
settings.py



```
STATIC_URL = '/static/'
TEMPLATES = [
    {
        'DIRS': ['/templates/'],
```

- Users can set their own root paths
  - templates (fragments of HTML files) are typically reused between apps
- Cjango can also load directory paths for templates and static files
  - **Key observation:** C++ can handle runtime configurations by text files

30 secs demo (1): changing to a different callback function

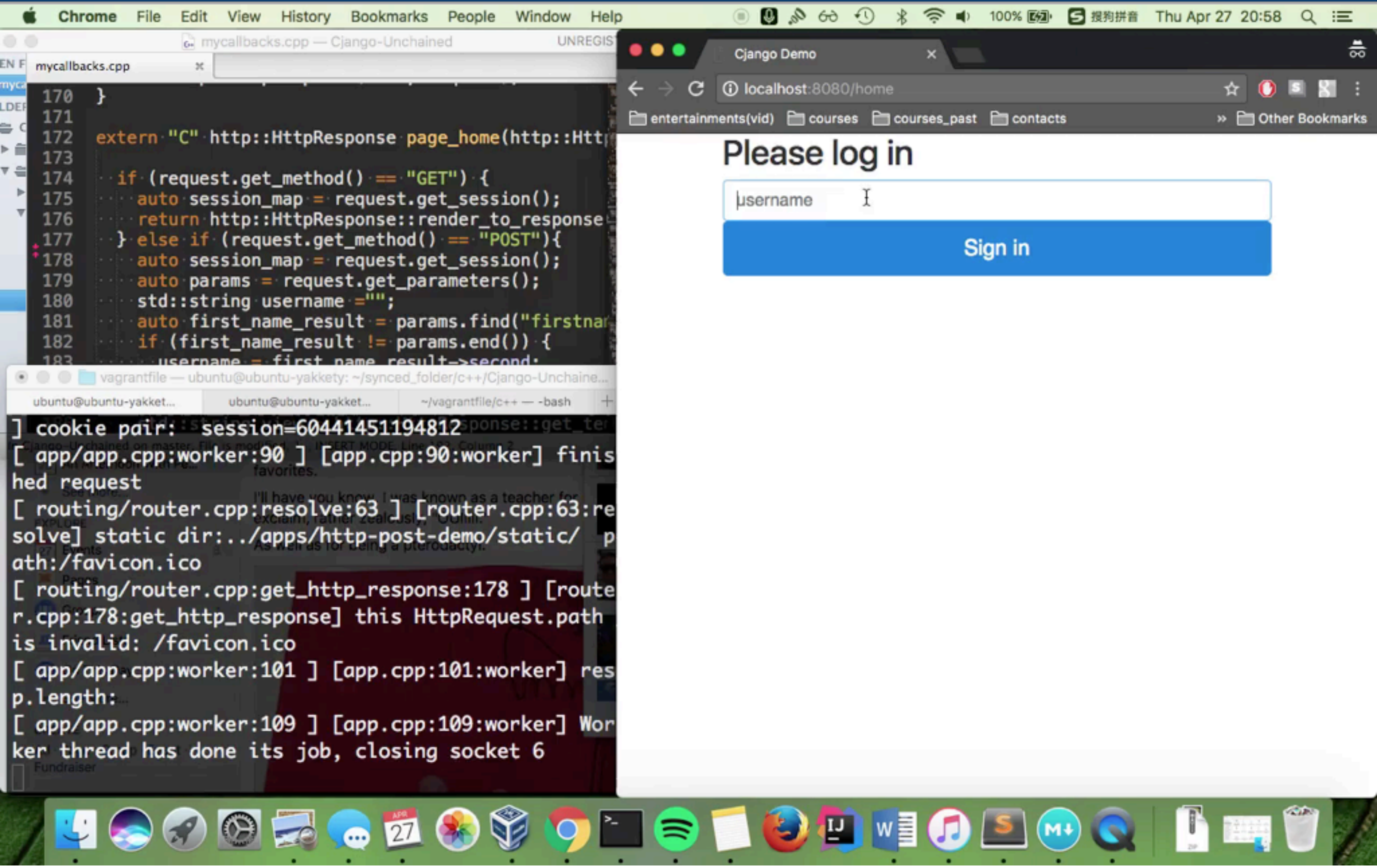


```
}

extern "C" http::HttpResponse page_home(http::Htt

  if (request.get_method() == "GET") {
    auto session_map = request.get_session();
    return http::HttpResponse::render_to_response
  } else if (request.get_method() == "POST"){
    auto session_map = request.get_session();
    auto params = request.get_parameters();
    std::string username ="";
    auto first_name_result = params.find("firstna
    if (first_name_result != params.end()) {
      username = first_name_result->second;
```

```
] cookie pair:  session=60441451194812
[ app/app.cpp:worker:90 ] [app.cpp:90:worker] finis
hed request
[ routing/router.cpp:resolve:63 ] [router.cpp:63:re
solve] static dir:../apps/http-post-demo/static/  p
ath:/favicon.ico
[ routing/router.cpp:get_http_response:178 ] [route
r.cpp:178:get_http_response] this HttpRequest.path
is invalid: /favicon.ico
[ app/app.cpp:worker:101 ] [app.cpp:101:worker] res
p.length:
[ app/app.cpp:worker:109 ] [app.cpp:109:worker] Wor
ker thread has done its job, closing socket 6
```

Chrome File Edit View History Bookmarks People Window Help
Thu Apr 27 20:58
mycallbacks.cpp — Cjango-Unchained
vagrantfile — ubuntu@ubuntu-yakkety: ~/synced_folder/c++/Cjango-Unchaine...
Cjango Demo
localhost:8080/home
entertainments(vid) courses courses_past contacts Other Bookmarks
Please log in
username
Sign in

30 secs demo (1): changing to a different callback function



```
}

extern "C" http::HttpResponse page_home(http::Htt
  
  if (request.get_method() == "GET") {
    auto session_map = request.get_session();
    return http::HttpResponse::render_to_response
  } else if (request.get_method() == "POST"){
    auto session_map = request.get_session();
    auto params = request.get_parameters();
    std::string username ="";
    auto first_name_result = params.find("firstnar
    if (first_name_result != params.end()) {
      username = first_name_result->second;
```

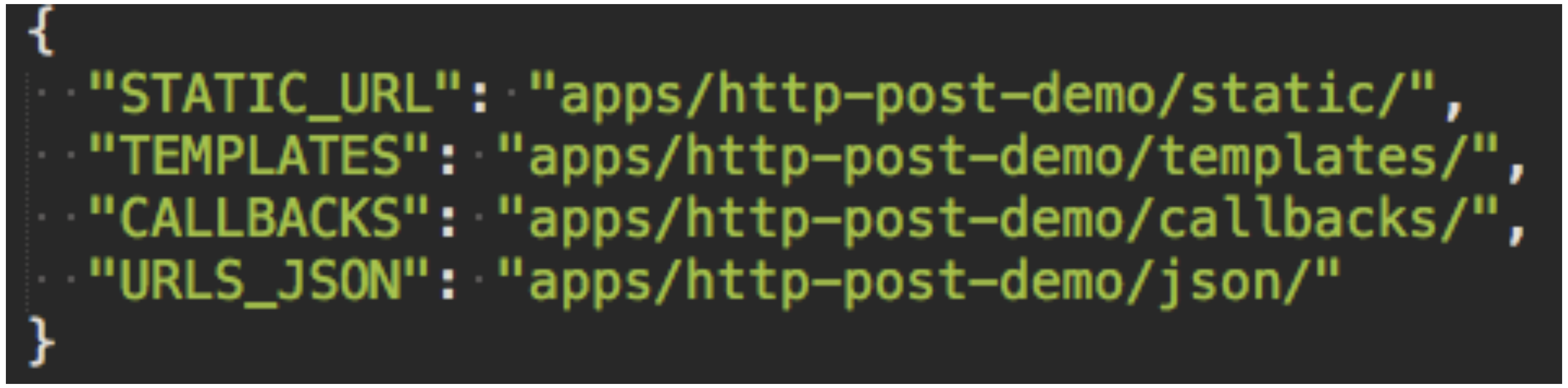
```
] cookie pair:  session=60441451194812
[ app/app.cpp:worker:90 ] [app.cpp:90:worker] finis
hed request
[ routing/router.cpp:resolve:63 ] [router.cpp:63:re
solve] static dir:../apps/http-post-demo/static/  p
ath:/favicon.ico
[ routing/router.cpp:get_http_response:178 ] [route
r.cpp:178:get_http_response] this HttpRequest.path
is invalid: /favicon.ico
[ app/app.cpp:worker:101 ] [app.cpp:101:worker] res
p.length:
[ app/app.cpp:worker:109 ] [app.cpp:109:worker] Wor
ker thread has done its job, closing socket 6
```

Tutorial: simple http server

- Step 1: directory setup:
    - callbacks/: a directory containing all callback definitions
    - json/: a directory containing settings.json and urls.json
    - static/: a directory containing all static files that will be referenced from html files (such as images, js files, and css files)
    - templates/: all html files

Tutorial: simple http server

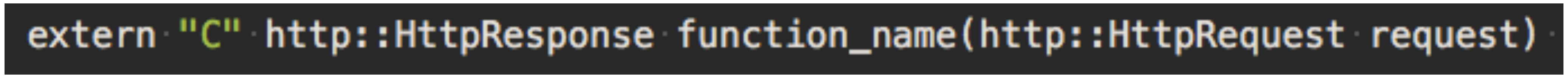
- Step 2: Tell Cjango about your directory setup:
    - Specify the paths to your four directories in json/settings.json file



```
{
  "STATIC_URL": "apps/http-post-demo/static/",
  "TEMPLATES": "apps/http-post-demo/templates/",
  "CALLBACKS": "apps/http-post-demo/callbacks/",
  "URLS_JSON": "apps/http-post-demo/json/"
}
```

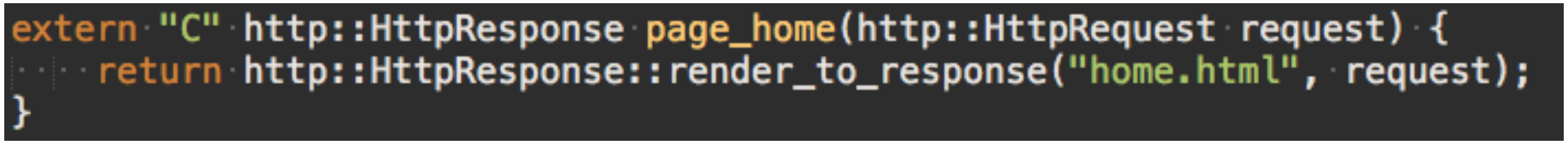
Tutorial: simple http server

- Step 3: Write a callback function
    - all callback functions must have the function signature



```
extern "C" http::HttpResponse function_name(http::HttpRequest request)
```

        - extern "C" is necessary for dynamic reloading
    - a simple callback function called "page_home" that returns "home.html" for a request like this:

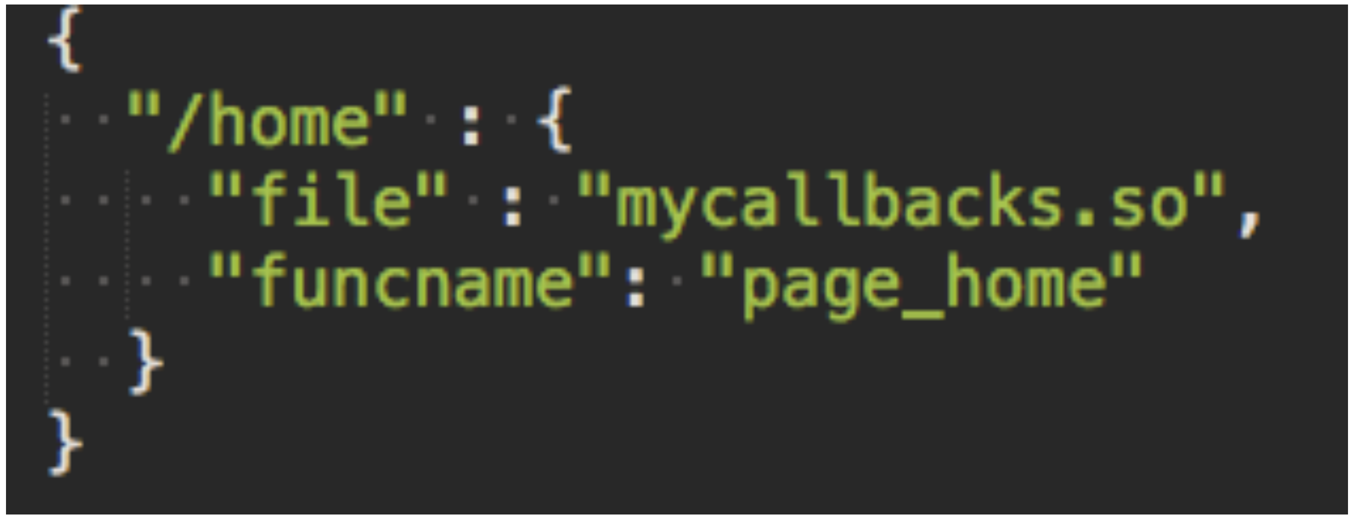


```
extern "C" http::HttpResponse page_home(http::HttpRequest request) {
    return http::HttpResponse::render_to_response("home.html", request);
}
```

        - notice that we only need to specify the html's file name without any path information. Cjango will find "home.html" in our templates/ directory.
    - compile it into a .so file (we have provided a generic Makefile for users' convenience)

Tutorial: simple simple http server

- Step 4: Define url mapping
  - Provide a url path “page_home” corresponds to inside json/urls.json



```
{
  "/home" : {
    "file" : "mycallbacks.so",
    "funcname": "page_home"
  }
}
```

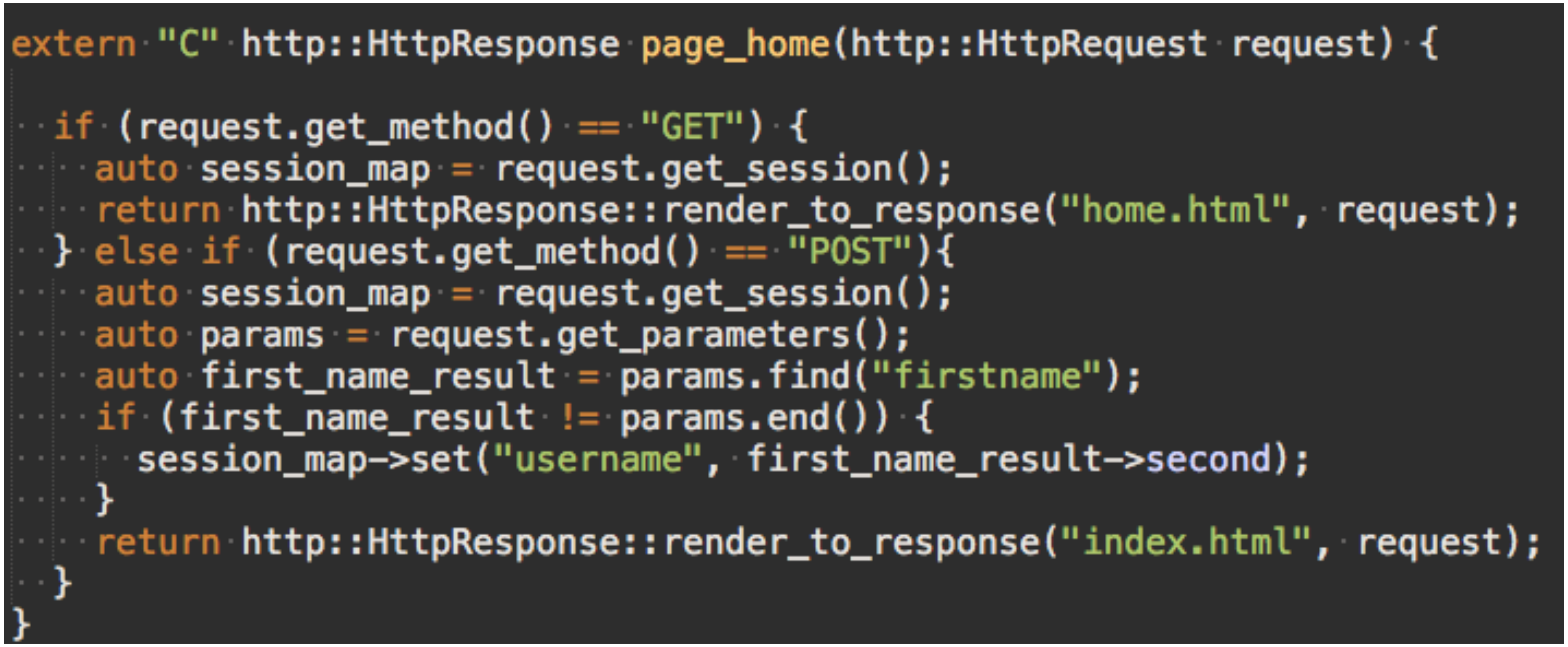
  - Now Cjango will find mycallbacks.so inside callbacks/ directory and run “page_home” whenever a client visits /home path

Tutorial: HttpRequest API

- HttpRequest class provides a helpful interface for retrieving a http request’s fields
  - request.get_method() -> returns http request’s method
  - request.get_meta() -> returns a map of http request’s headers
  - request.get_parameters() -> returns a map of http request’s parameters
  - request.get_session() -> returns a session object associated with the current request

Tutorial: Using HttpRequest In Callback Functions

- Using HttpRequest, our callback functions now can do something more complicated:

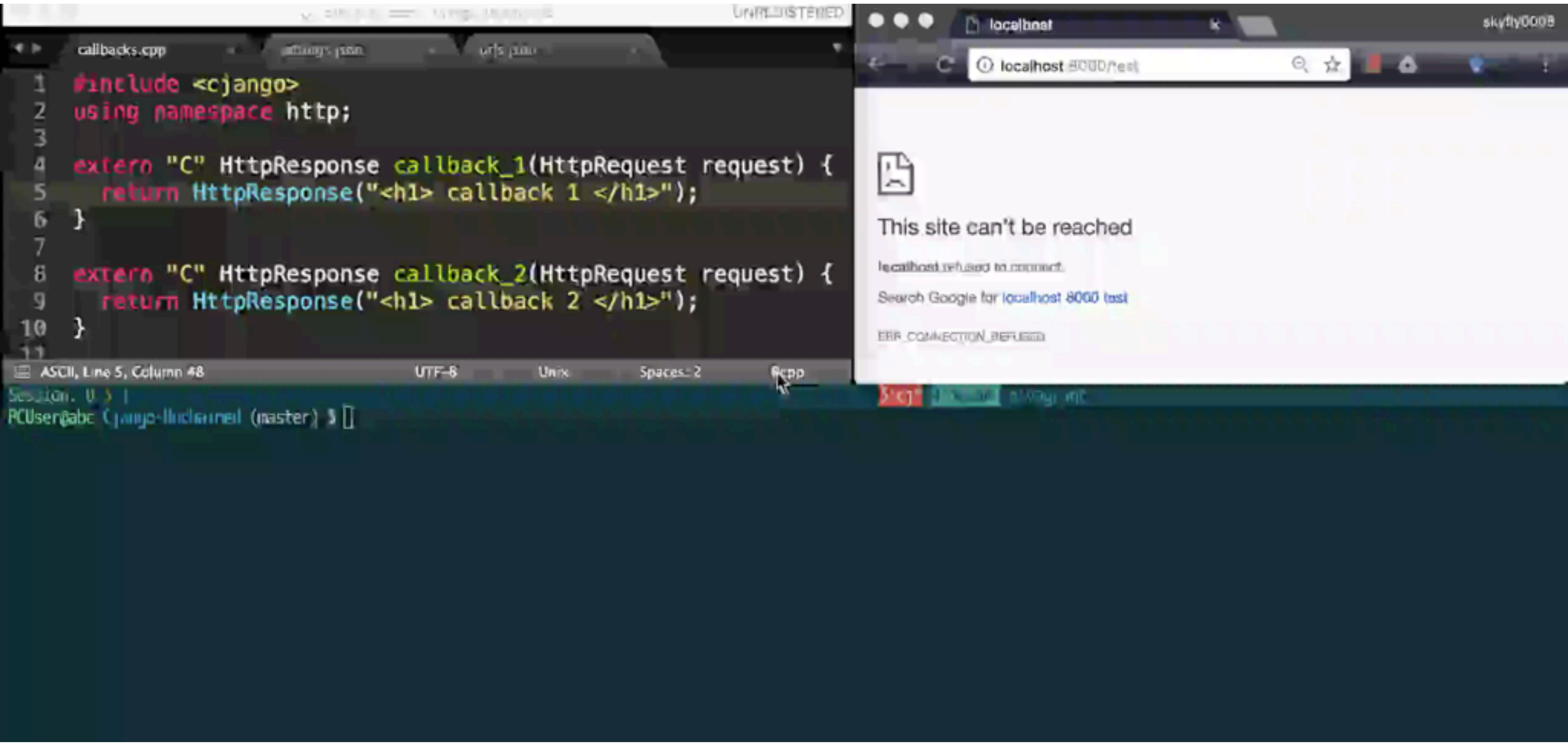


```
extern "C" http::HttpResponse page_home(http::HttpRequest request) {

  if (request.get_method() == "GET") {
    auto session_map = request.get_session();
    return http::HttpResponse::render_to_response("home.html", request);
  } else if (request.get_method() == "POST"){
    auto session_map = request.get_session();
    auto params = request.get_parameters();
    auto first_name_result = params.find("firstname");
    if (first_name_result != params.end()) {
      session_map->set("username", first_name_result->second);
    }
    return http::HttpResponse::render_to_response("index.html", request);
  }
}
```

Cjango's unique feature:
Dynamic (Runtime) Callback Loading

Github: mengdilin/Cjango-Unchained

30 secs demo (2): changing to a different callback function



```
#include <cjango>
using namespace http;

extern "C" HttpResponse callback_1(HttpRequest request) {
  return HttpResponse("<h1> callback 1 </h1>");
}

extern "C" HttpResponse callback_2(HttpRequest request) {
  return HttpResponse("<h1> callback 2 </h1>");
}
```

This site can't be reached

localhost refused to connect.

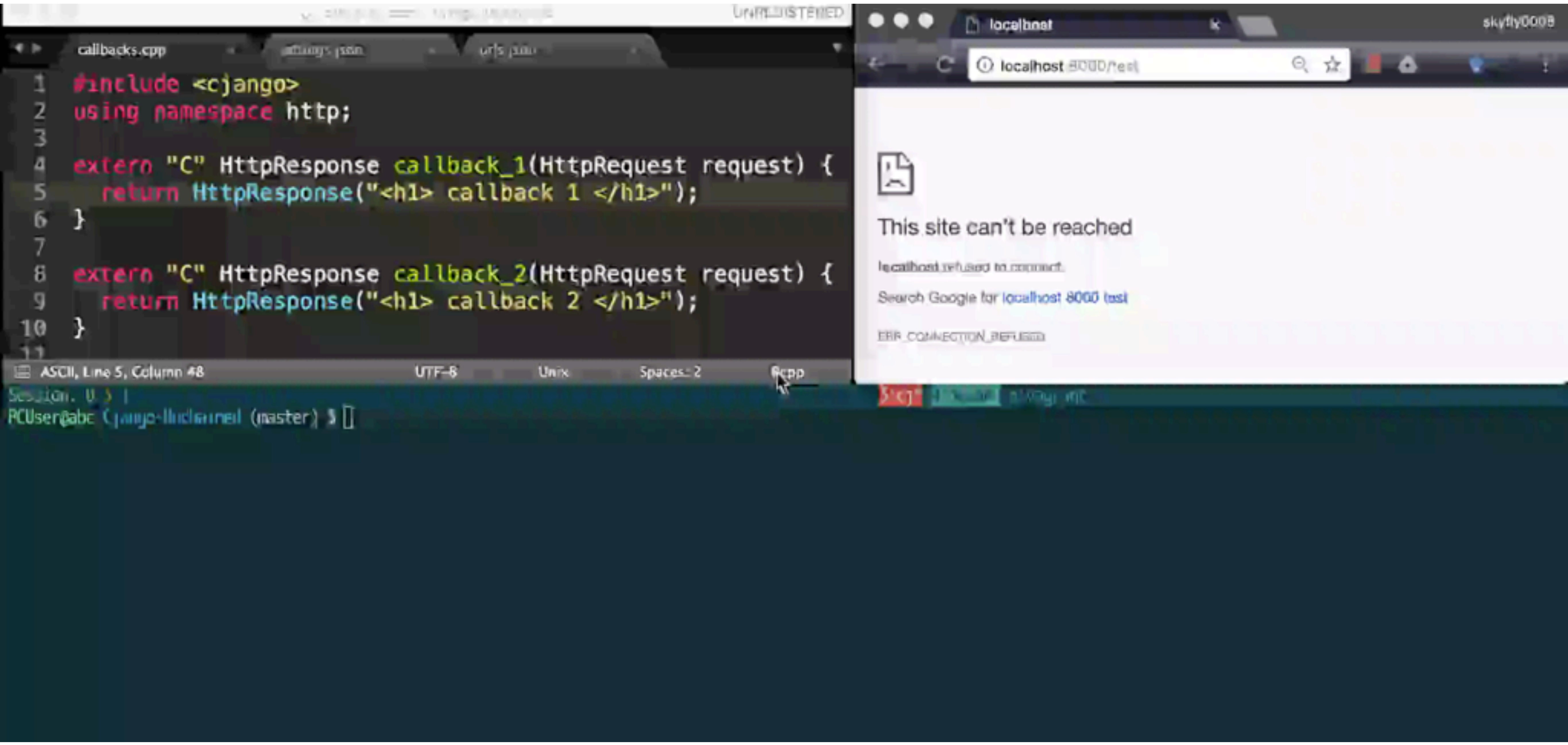
Search Google for localhost 8000 test

ERR_CONNECTION_REFUSED

```
RCUser@abc Cjango-Unchained (master) $
```

Github: mengdilin/Cjango-Unchained

30 secs demo (2): changing to a different callback function



```
#include <cjango>
using namespace http;

extern "C" HttpResponse callback_1(HttpRequest request) {
  return HttpResponse("<h1> callback 1 </h1>");
}

extern "C" HttpResponse callback_2(HttpRequest request) {
  return HttpResponse("<h1> callback 2 </h1>");
}
```

This site can't be reached

localhost refused to connect.

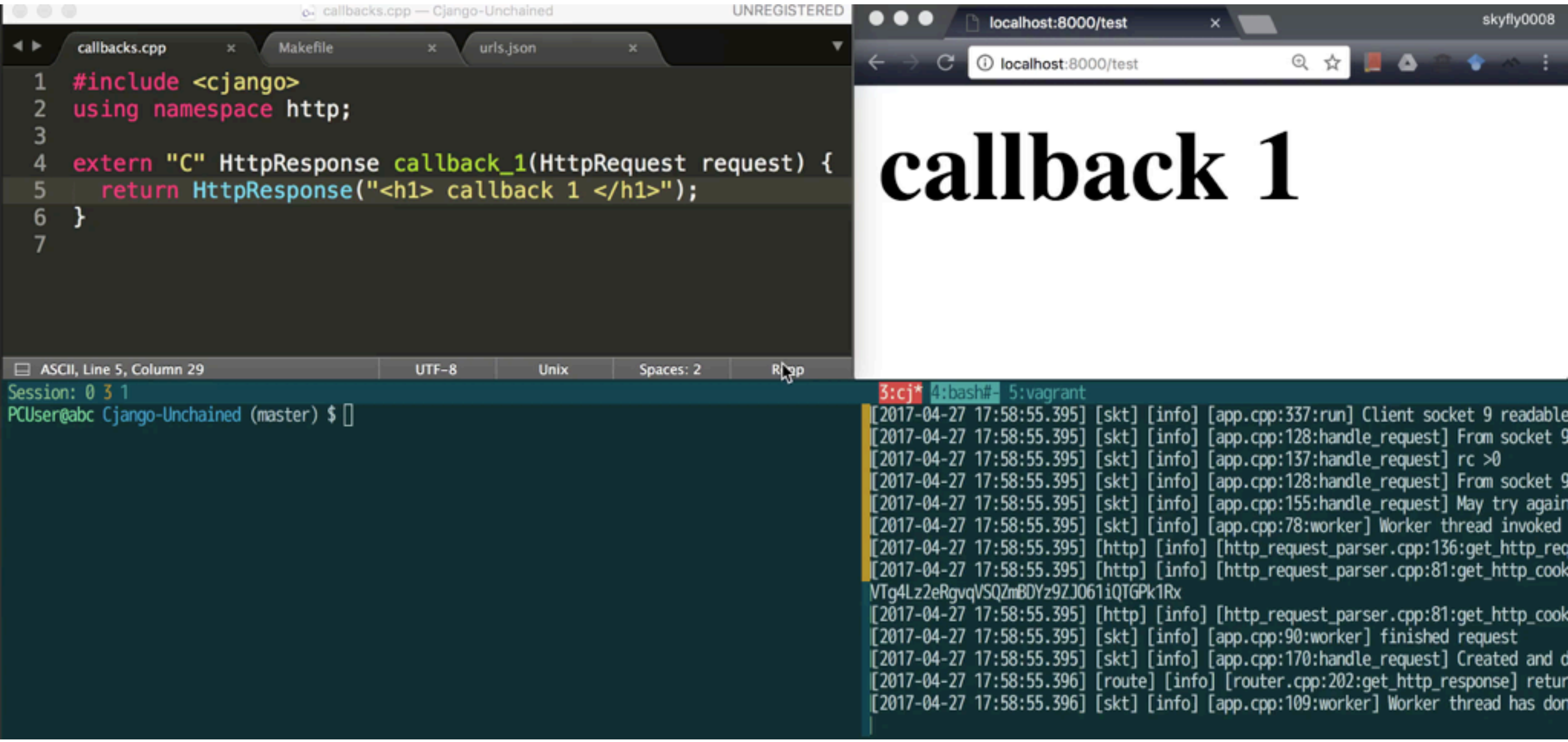
Search Google for localhost 8000 test

ERR_CONNECTION_REFUSED

```
RCUser@abc Cjango-Unchained (master) $
```

Github: mengdilin/Cjango-Unchained

30 secs demo (3): rewriting the same callback function



```
#include <cjango>
using namespace http;

extern "C" HttpResponse callback_1(HttpRequest request) {
  return HttpResponse("<h1> callback 1 </h1>");
}

```

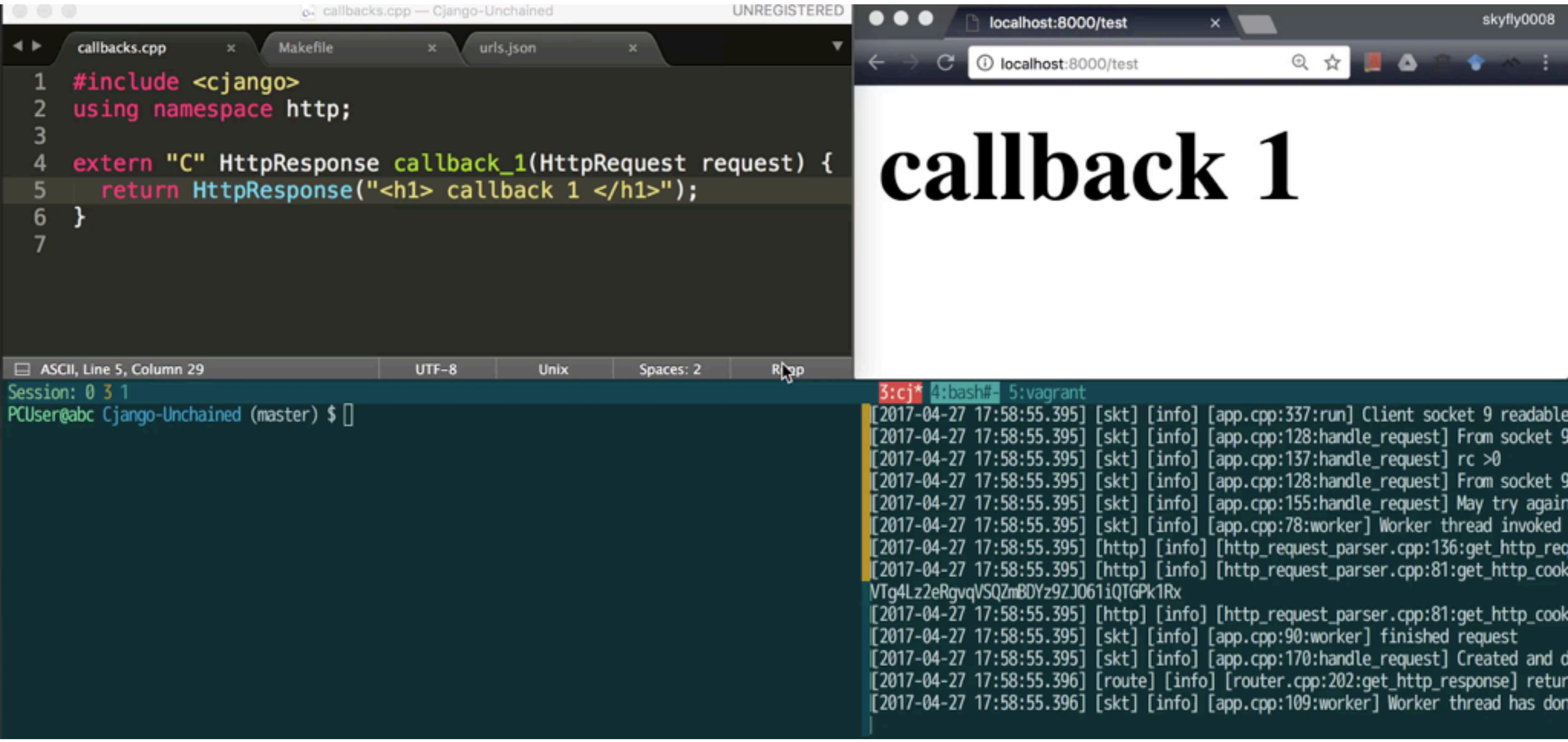
callback 1

```
Session: 0 3 1
PCUser@abc Cjango-Unchained (master) $
```

```
3:cj* 4:bash#- 5:vagrant
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:337:run] Client socket 9 readable
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:128:handle_request] From socket 9
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:137:handle_request] rc >0
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:128:handle_request] From socket 9
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:155:handle_request] May try agair
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:78:worker] Worker thread invoked
[2017-04-27 17:58:55.395] [http] [info] [http_request_parser.cpp:136:get_http_rec
[2017-04-27 17:58:55.395] [http] [info] [http_request_parser.cpp:81:get_http_cook
VTg4Lz2eRgvqVSQZmBDYz9ZJ061iQTGPk1Rx
[2017-04-27 17:58:55.395] [http] [info] [http_request_parser.cpp:81:get_http_cook
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:90:worker] finished request
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:170:handle_request] Created and c
[2017-04-27 17:58:55.396] [route] [info] [router.cpp:202:get_http_response] retur
[2017-04-27 17:58:55.396] [skt] [info] [app.cpp:109:worker] Worker thread has dor
```

Github: mengdilin/Cjango-Unchained

30 secs demo (3): rewriting the same callback function



```cpp
#include <cjango>
using namespace http;

extern "C" HttpResponse callback_1(HttpRequest request) {
  return HttpResponse("<h1> callback 1 </h1>");
}
```

callback 1

```
PCUser@abc Cjango-Unchained (master) $
```

```
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:337:run] Client socket 9 readable
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:128:handle_request] From socket 9
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:137:handle_request] rc >0
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:128:handle_request] From socket 9
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:155:handle_request] May try again
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:78:worker] Worker thread invoked
[2017-04-27 17:58:55.395] [http] [info] [http_request_parser.cpp:136:get_http_req
[2017-04-27 17:58:55.395] [http] [info] [http_request_parser.cpp:81:get_http_cook
VTg4Lz2eRgvqVSQZmBDYz9ZJ061iQTGPk1Rx
[2017-04-27 17:58:55.395] [http] [info] [http_request_parser.cpp:81:get_http_cook
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:90:worker] finished request
[2017-04-27 17:58:55.395] [skt] [info] [app.cpp:170:handle_request] Created and d
[2017-04-27 17:58:55.396] [route] [info] [router.cpp:202:get_http_response] retur
[2017-04-27 17:58:55.396] [skt] [info] [app.cpp:109:worker] Worker thread has don
```

Use cases of Dynamic Callback Reloading

Use case 1: changing a URL-to-callback mapping

- the first demo video
- just changing the function name in urls.json

Use case 2: updating a callback

- the second demo video

(A) “`make`“

+ the compiled callback is reloaded automatically

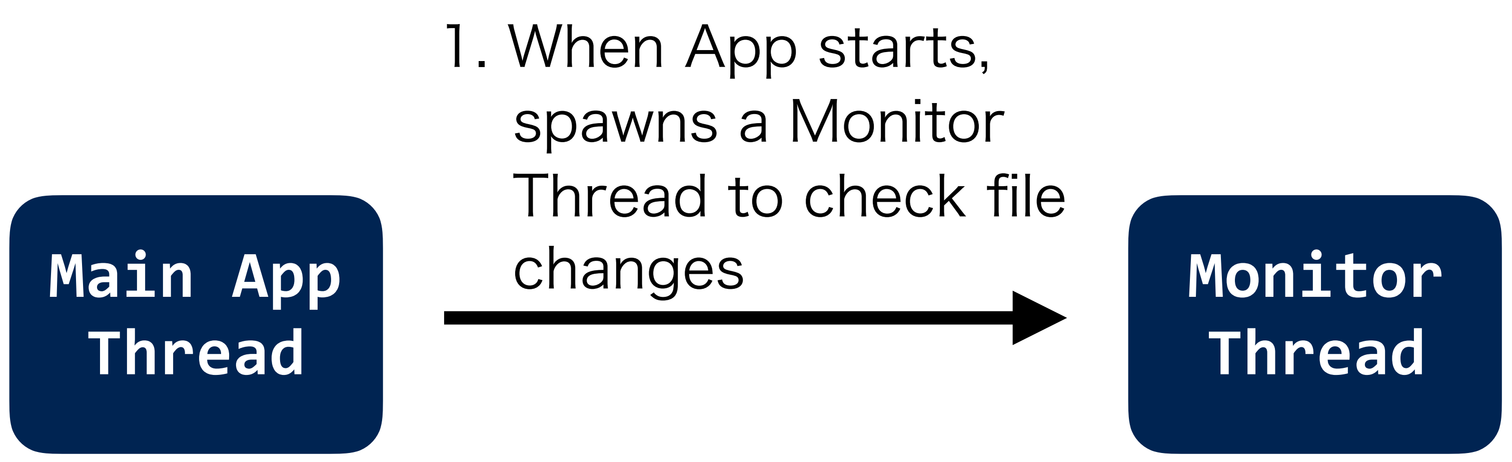
(B) “`make TOUCH=0`”

+ the compiled callback is **not** reloaded
  - useful for unit testing
+ if he finishes his work, run “`touch urls.json`” in the end

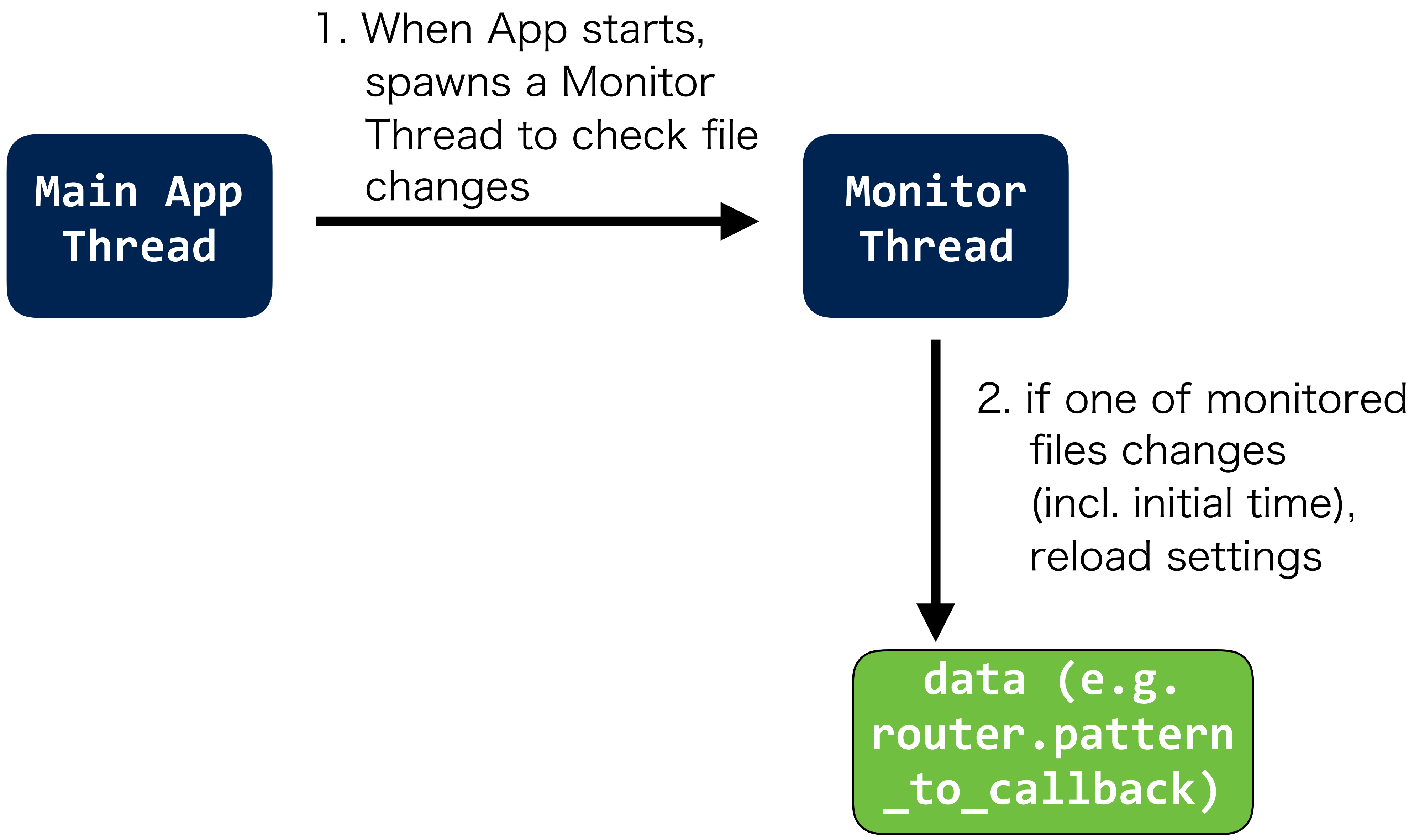
Github: mengdilin/Cjango-Unchained

Cjango's callback updating flow

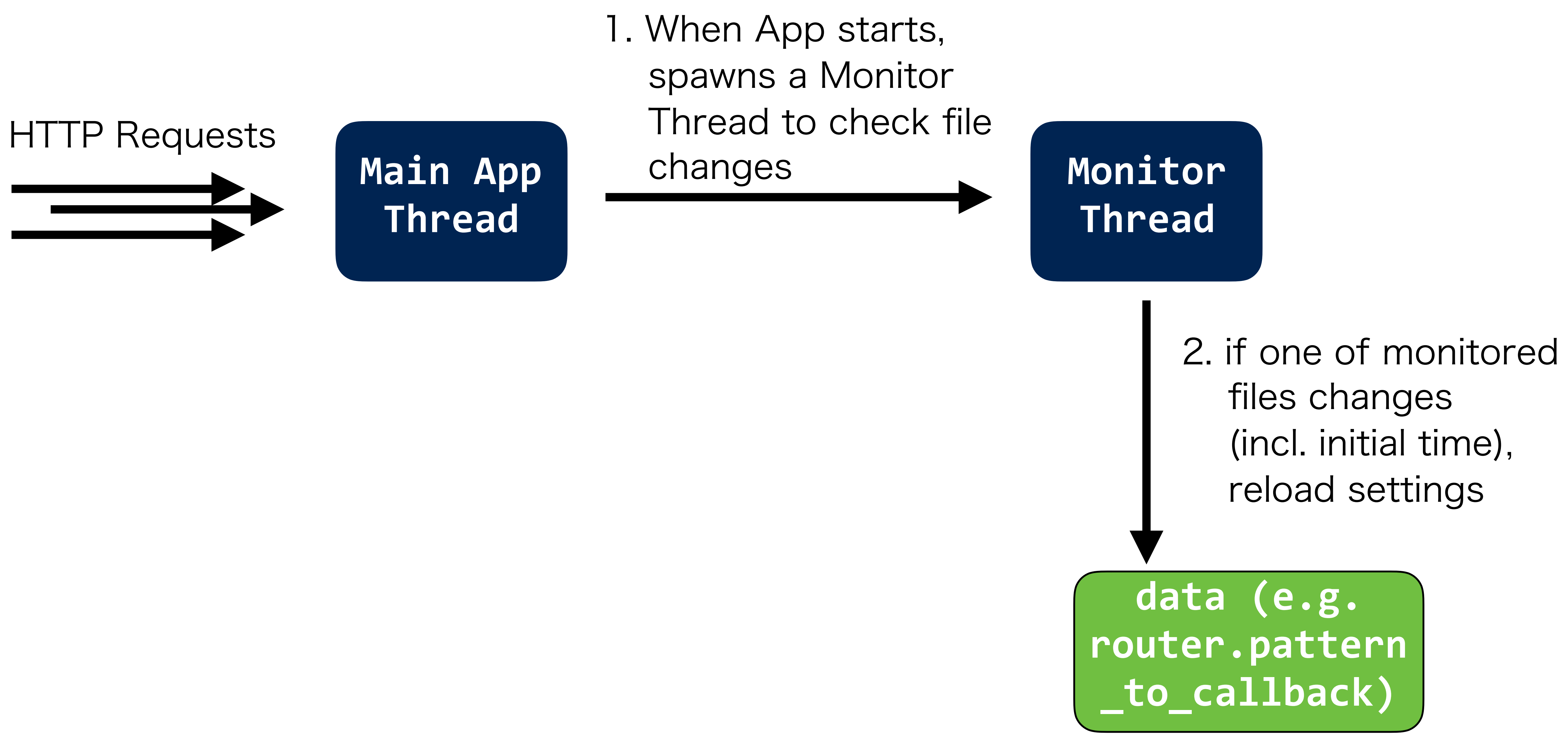
Cjango's callback updating flow


Main App Thread
1. When App starts, spawns a Monitor Thread to check file changes
Monitor Thread

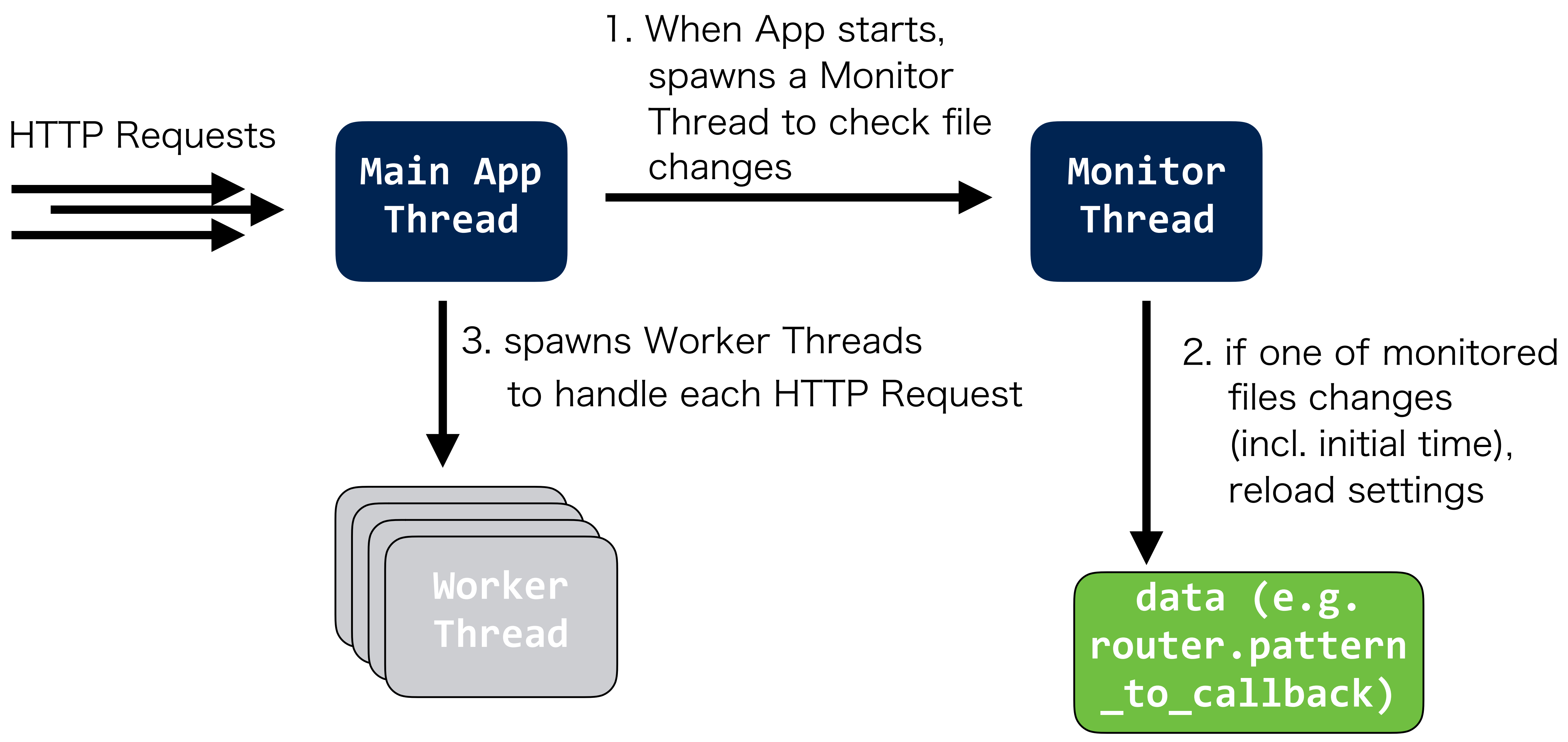
Cjango's callback updating flow


Main App Thread
1. When App starts, spawns a Monitor Thread to check file changes
Monitor Thread
2. if one of monitored files changes (incl. initial time), reload settings
data (e.g. router.pattern_to_callback)

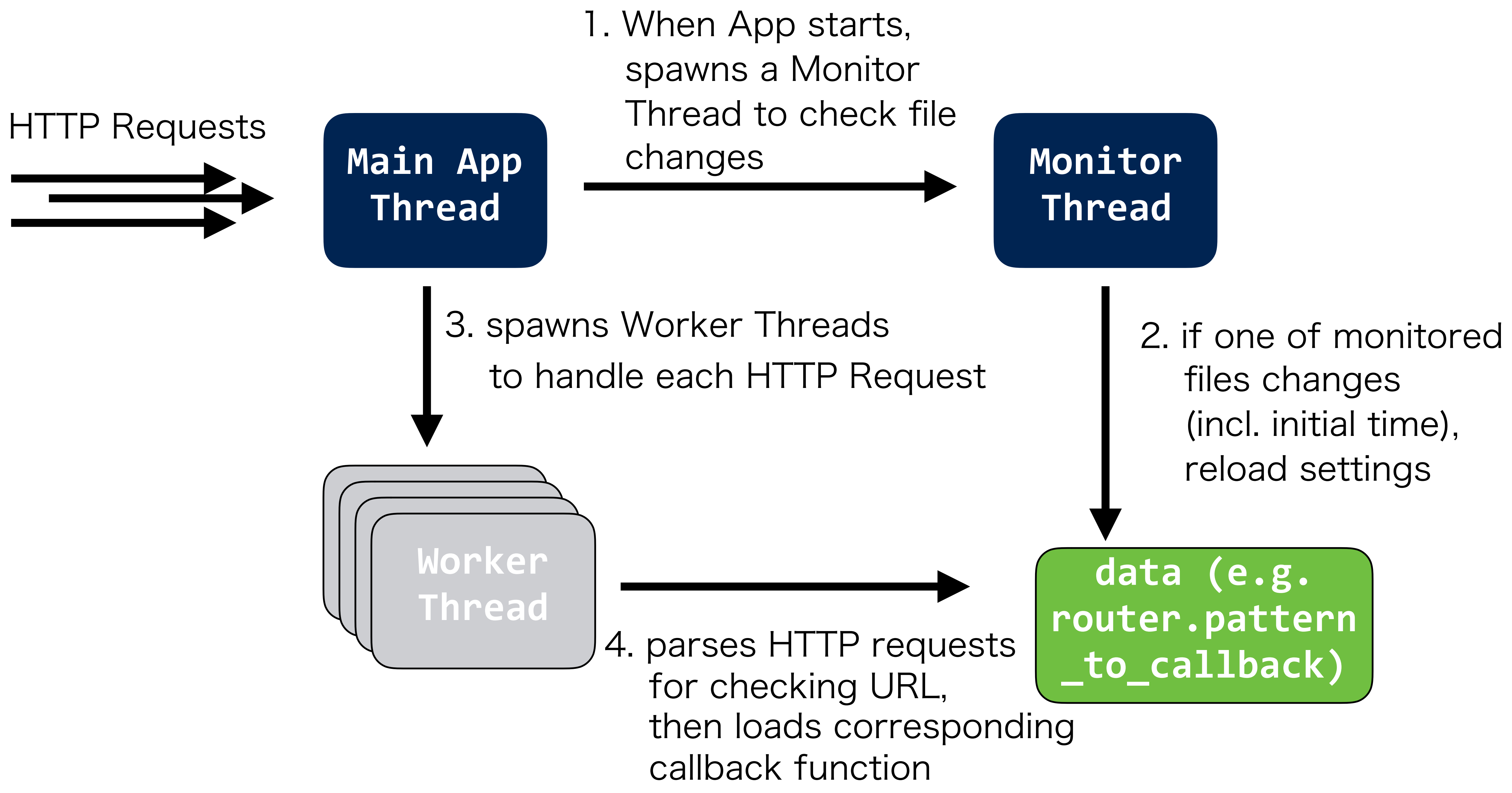
Cjango's callback updating flow


HTTP Requests
Main App Thread
1. When App starts, spawns a Monitor Thread to check file changes
Monitor Thread
2. if one of monitored files changes (incl. initial time), reload settings
data (e.g. router.pattern_to_callback)

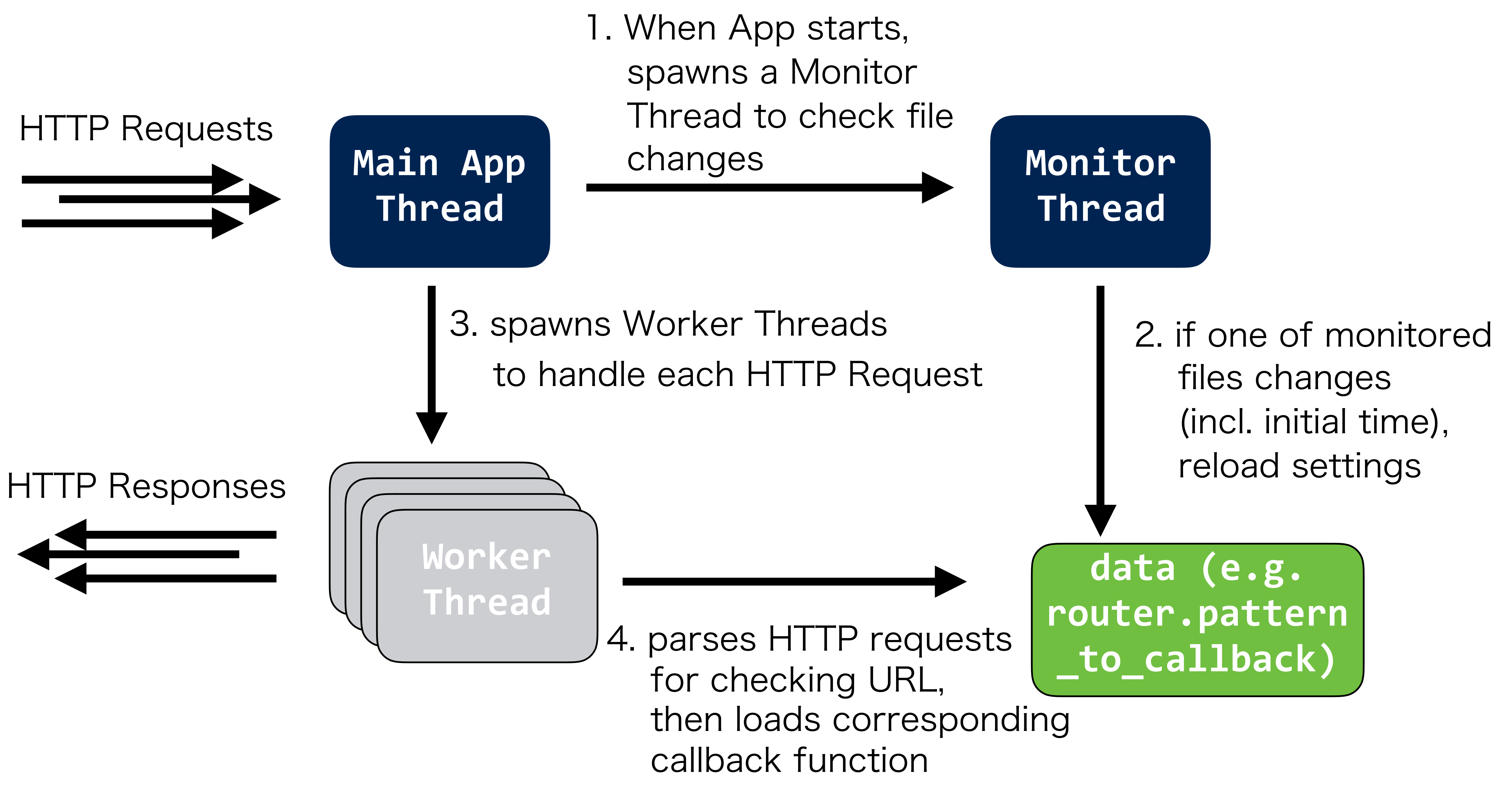
Cjango's callback updating flow


HTTP Requests
Main App Thread
1. When App starts, spawns a Monitor Thread to check file changes
Monitor Thread
3. spawns Worker Threads to handle each HTTP Request
2. if one of monitored files changes (incl. initial time), reload settings
Worker Thread
data (e.g. router.pattern_to_callback)

Cjango's callback updating flow


HTTP Requests
Main App Thread
1. When App starts, spawns a Monitor Thread to check file changes
Monitor Thread
3. spawns Worker Threads to handle each HTTP Request
2. if one of monitored files changes (incl. initial time), reload settings
Worker Thread
data (e.g. router.pattern_to_callback)
4. parses HTTP requests for checking URL, then loads corresponding callback function

Cjango's callback updating flow


HTTP Requests
Main App Thread
1. When App starts, spawns a Monitor Thread to check file changes
Monitor Thread
3. spawns Worker Threads to handle each HTTP Request
2. if one of monitored files changes (incl. initial time), reload settings
HTTP Responses
Worker Thread
data (e.g. router.pattern_to_callback)
4. parses HTTP requests for checking URL, then loads corresponding callback function

Dynamic loading internals

We used C library called "dl" (Dynamic Loading) declared in <dlfcn.h>

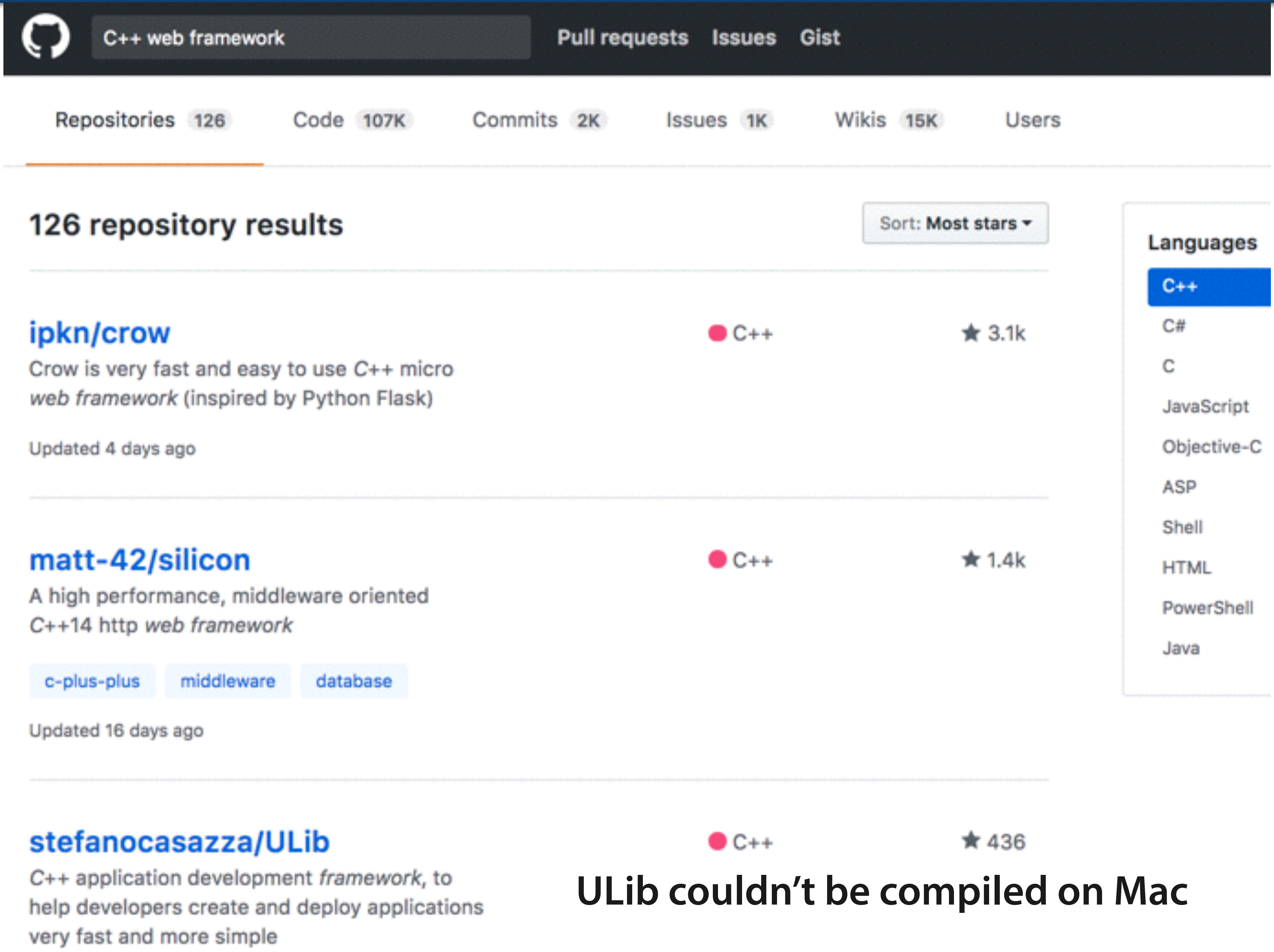
- opens an object file by dlopen()
- accesses a symbol in the file by dlsym()
- dlclose() closes a shared object file like file's close()
- dlerror() for getting error types
    - when no file/symbol is found, Cjango sets:
        - a callback returning "500 Internal Server Error" in deploy mode
        - a callback returning "Invalid Callback Specified" in debug mode

- when no URL pattern is matched
    - a callback returning "404 Page Not Found"

Performance

Github: mengdilin/Cjango-Unchained

Compare with Github's Top 3 C++ Web App Frameworks (and Django)



C++ web framework   Pull requests   Issues   Gist

Repositories 126   Code 107K   Commits 2K   Issues 1K   Wikis 15K   Users

126 repository results

Sort: **Most stars** ▾

Languages
- C++
- C#
- C
- JavaScript
- Objective-C
- ASP
- Shell
- HTML
- PowerShell
- Java

ipkn/crow   ● C++   ★ 3.1k

Crow is very fast and easy to use C++ micro *web framework* (inspired by Python Flask)

Updated 4 days ago

matt-42/silicon   ● C++   ★ 1.4k

A high performance, middleware oriented C++14 http *web framework*

c-plus-plus   middleware   database

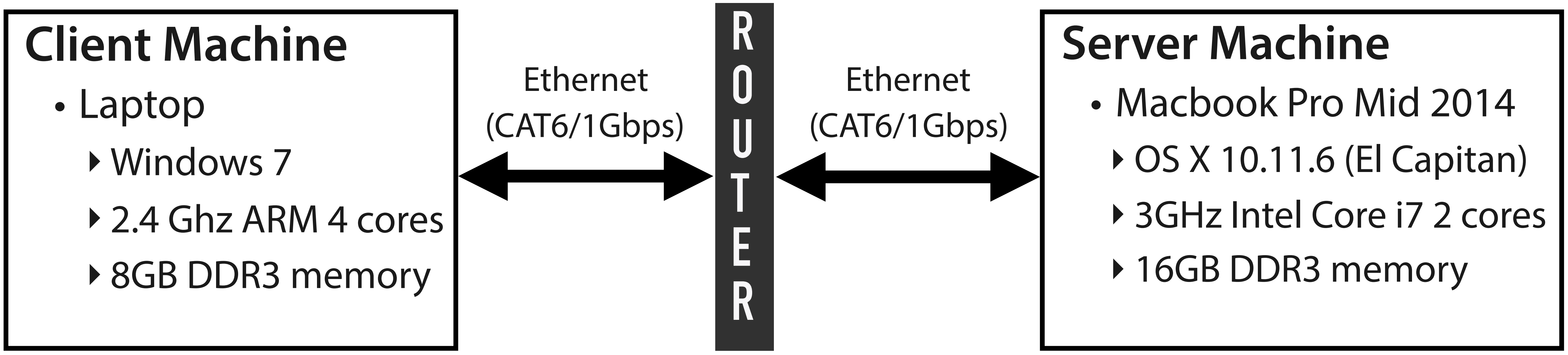
Updated 16 days ago

stefanocasazza/ULib   ● C++   ★ 436

C++ application development *framework*, to help developers create and deploy applications very fast and more simple

ULib couldn't be compiled on Mac

Experiment Conditions (Results in Next Slide)

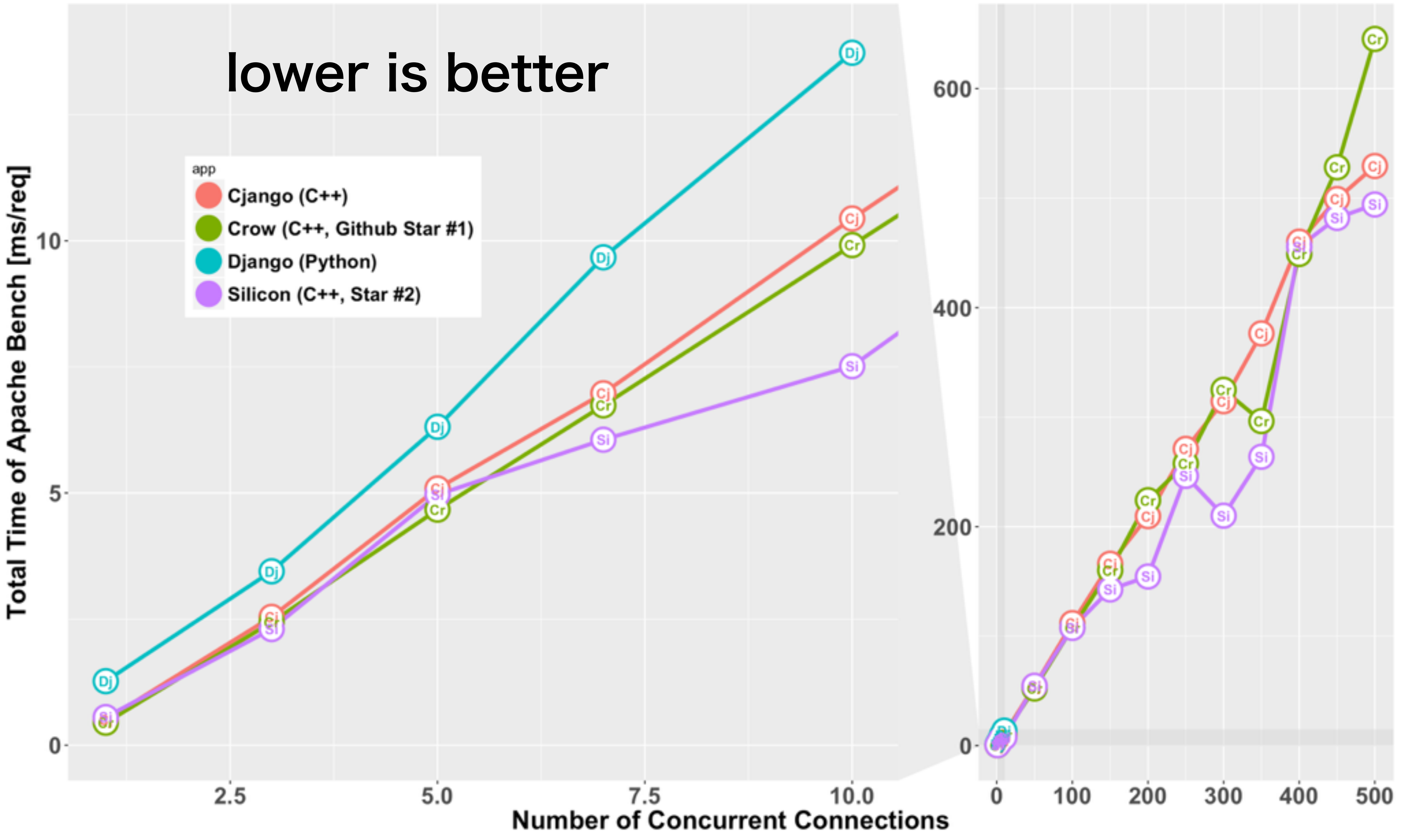


Settings

- One of Django, Cjango, Crow or Silicon apps is running on Server Machine
  - each app serves a simple "<html>HelloWorld</html>" string
  - **goal:** measure effects on overheads of Cjango's dynamic loading, original HTTP parser, original HTTP request handlings
- Client Machine accesses by Apatch Bench (common Http benchmark software)
  - `ab -n 10K -c {1,3,5,7,10,50,100,…,500} http://[IP]:8000/`
    - -n : # of total HTTP requests
    - -c : # of concurrent HTTP requests
- Raw data and more details are uploaded at github Cjango-Unchained/src/bench/

Github: mengdilin/Cjango-Unchained

Cjango is faster than Django by ~20% and has comparable speeds with other C++ frameworks for concurrent HTTP requests



Performance Conclusion: Cjango is as fast as other common C++ web app frameworks

Future

Add features targeted for personal use

- http 1.1
- https
- URL queries/parameters

Add a Validator for callback specification

- Currently no check for user-defined callback
- for more type safety

Learn more:

Cjango is hosted on Github

- mengdilin/Cjango-Unchained
- tutorials
- API documents (by doxygen) are under /src/doc

References

- C++ language core issue reports
  http://www.open-std.org/jtc1/sc22/wg21/docs/cwg_defects.html#195
    - spec about conversion between object pointer and function pointer (dlsym)
- http://pubs.opengroup.org/onlinepubs/9699919799/functions/dlsym.html
- Crow's reputation after publicity
    - https://news.ycombinator.com/item?id=8002604
    - Discussions on header-only effectivity