Martin Tschammer
CTO, GoBox
@mtschammer
mjt on IRC

irc.freenode.org
#djangocph

Django School

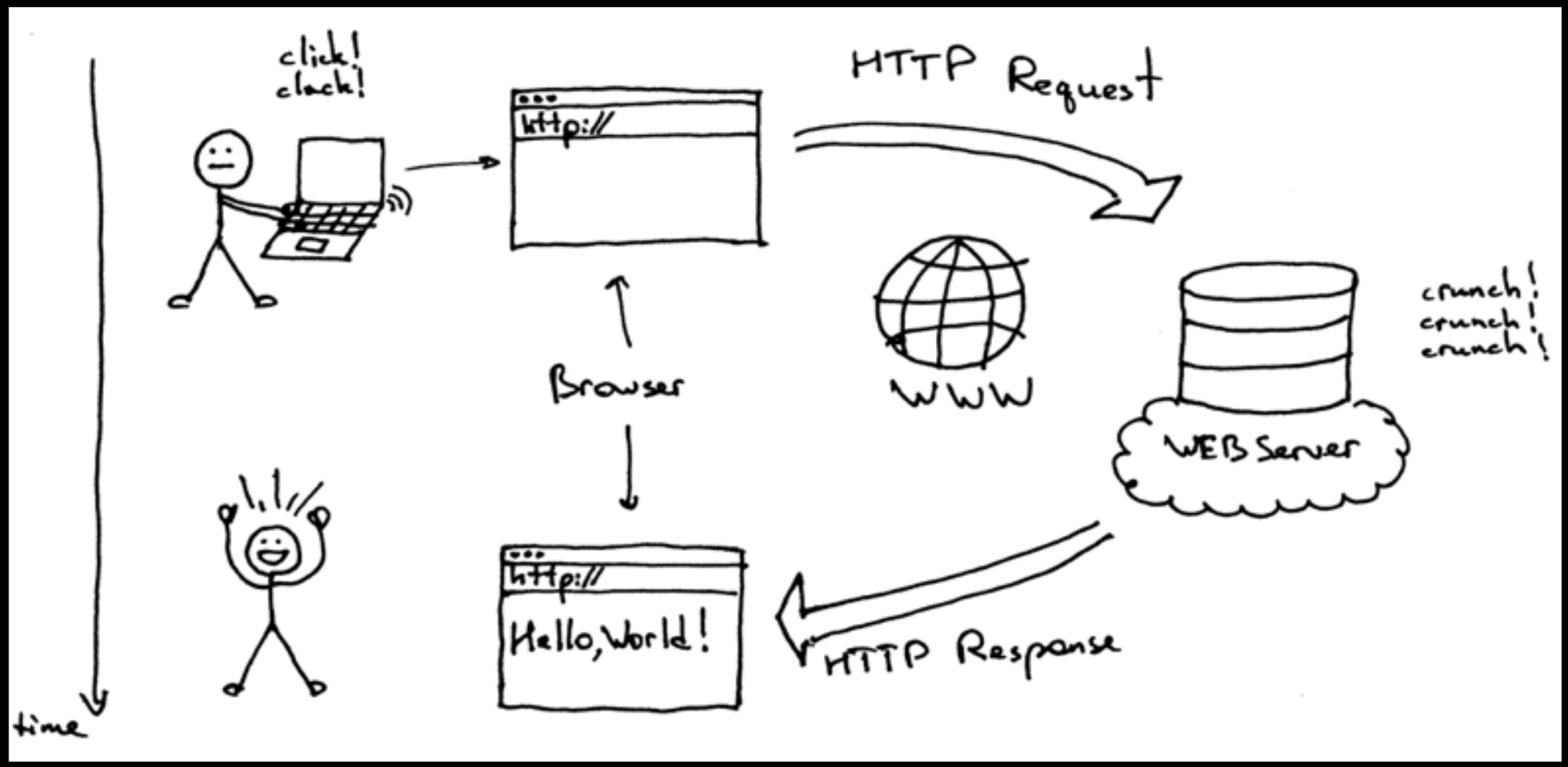
Lesson 03

How to speak with Django

Today

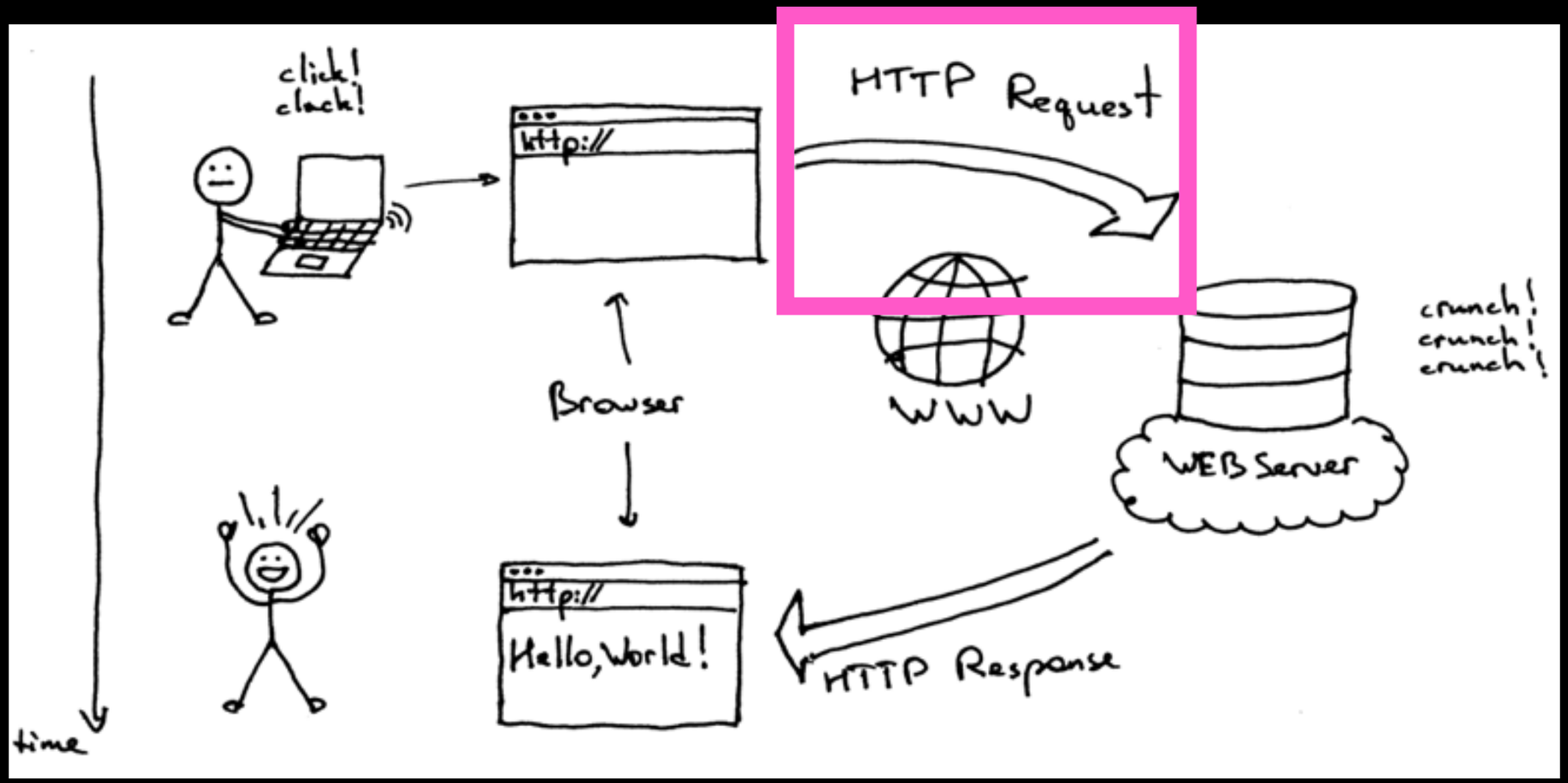
1. Talking with webservers
2. How does that translate to Django?
3. WORK IT BABY

Talking to web servers

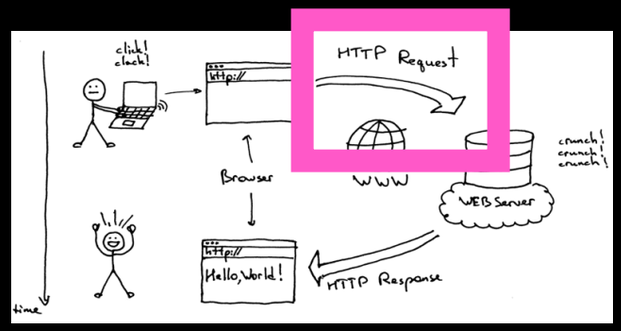


http://ruslanspivak.com/lsbaws-part1/

Requests


click!
clack!
http://
HTTP Request
WWW
crunch!
crunch!
crunch!
WEB Server
Browser
http://
Hello, World!
HTTP Response
time

Requests



GET

POST

PUT

PATCH

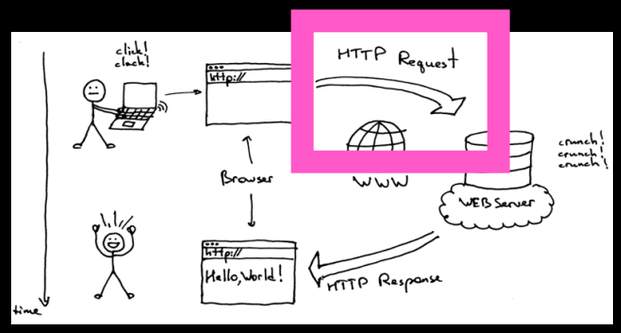
OPTIONS

DELETE

HEAD

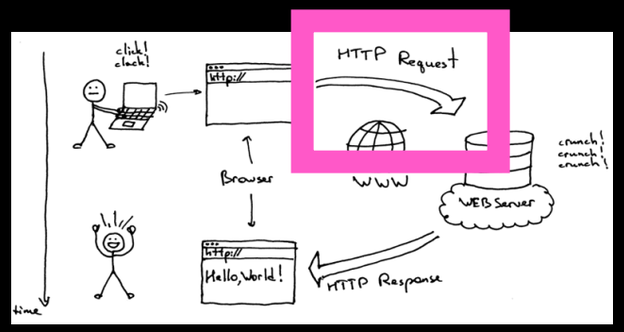
CONNECT

Requests



GET	POST
- GET information from the webserver - “GET the HTML for this part of the website”	- POST information to the webserver - “Create a new blog post with this title and content”

Requests

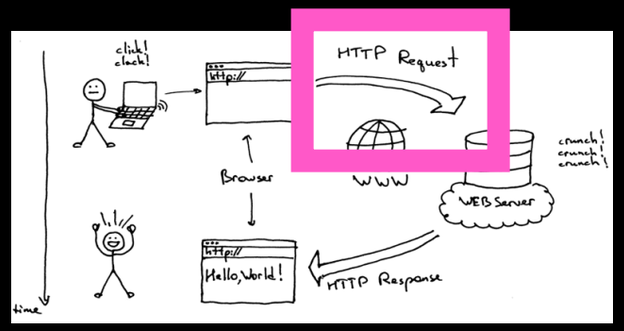


Sending information to the server:

GET - querystrings

POST - form data

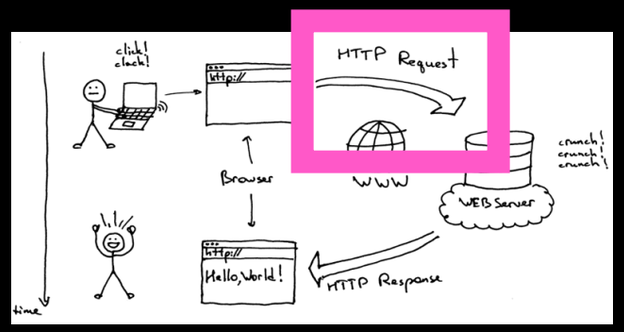
Requests



Querystrings - GET

- Filtering
- Sorting
- Extra information
- Just for fun YOLO

Requests

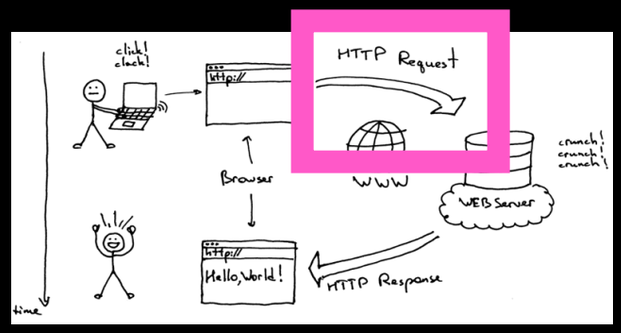


Querystrings - GET

http://example.com/over/there?name=Spike

http://example.com/over/there?name=Spike&race=ferret

Requests



Form data - POST

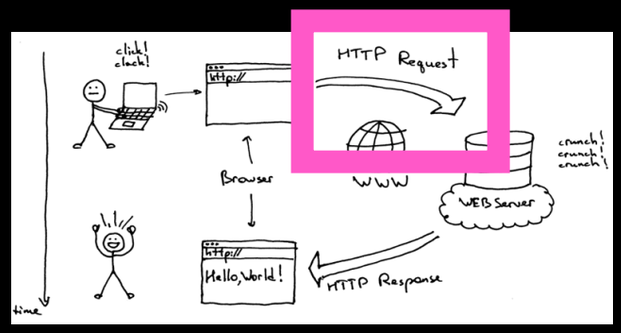
```
1 <form action="http://foo.com" method="get">
2   <input name="say" value="Hi">
3   <input name="to" value="Mom">
4   <button>Send my greetings</button>
5 </form>
```

→

```
1 POST / HTTP/1.1
2 Host: foo.com
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 13
5
6 say=Hi&to=Mom
```

Data is part of the message body - NOT the URL

Requests



GET

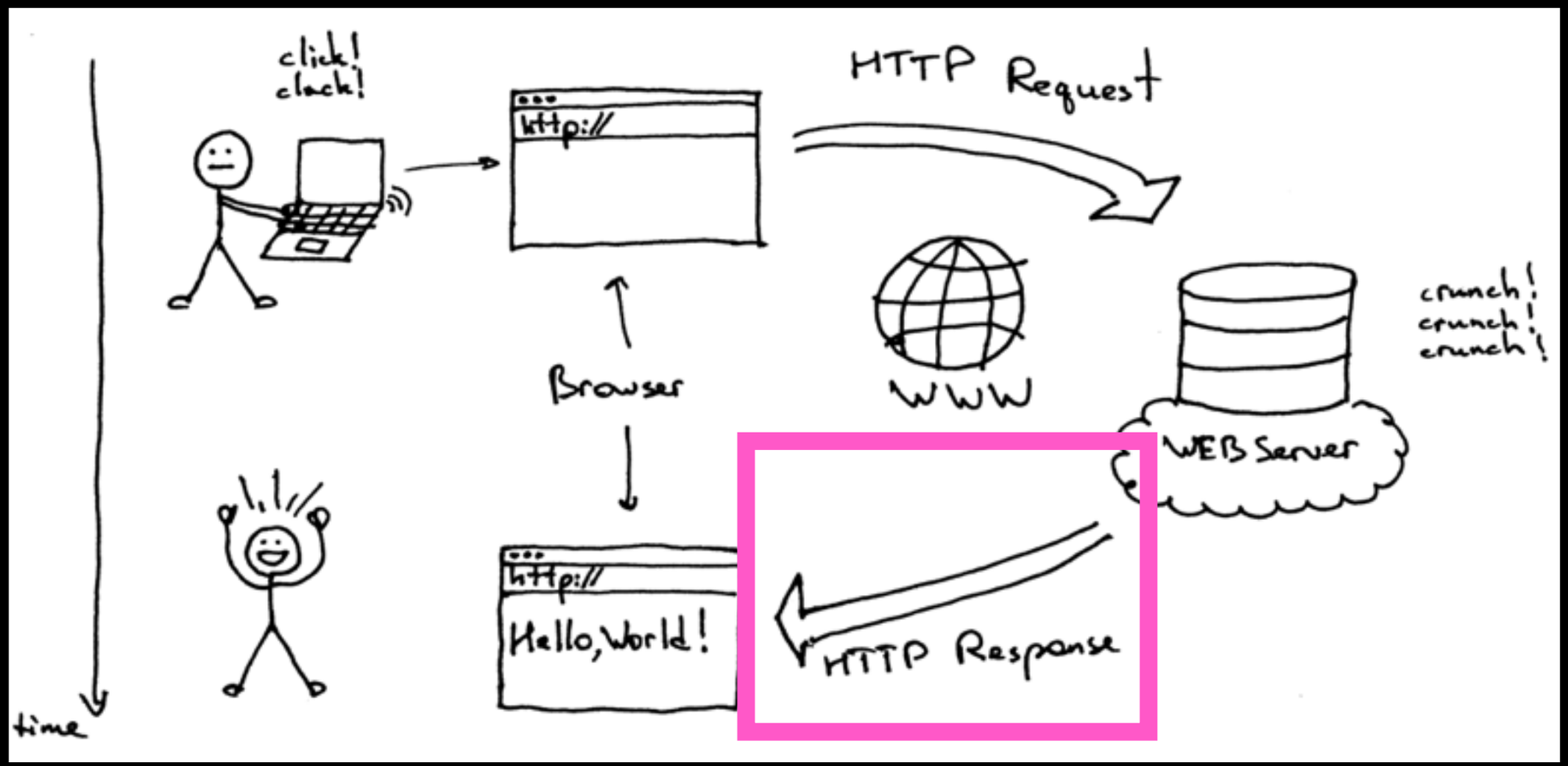
```
GET /?say=Hi&to=Mom HTTP/1.1
Host: foo.com
```

POST

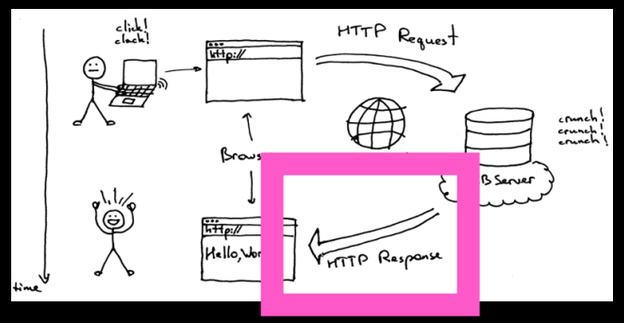
```
POST / HTTP/1.1
Host: foo.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

say=Hi&to=Mom
```

Responses


click!
clack!
http://
HTTP Request
Browser
WWW
crunch!
crunch!
crunch!
WEB Server
http://
Hello, World!
HTTP Response
time

Responses



1xx: Information "Just letting you know"

2xx: Successful "I did what you asked and everything's peachy"

3xx: Redirection "That's not where you wanna go. This is."

4xx: Client Error "You fucked up. I can't or won't do that."

5xx: Server Error "Ok, I fucked up. Something went horribly wrong."

Exercise!

Fill in the blanks

- "A GET request should be used only to retrieve data."

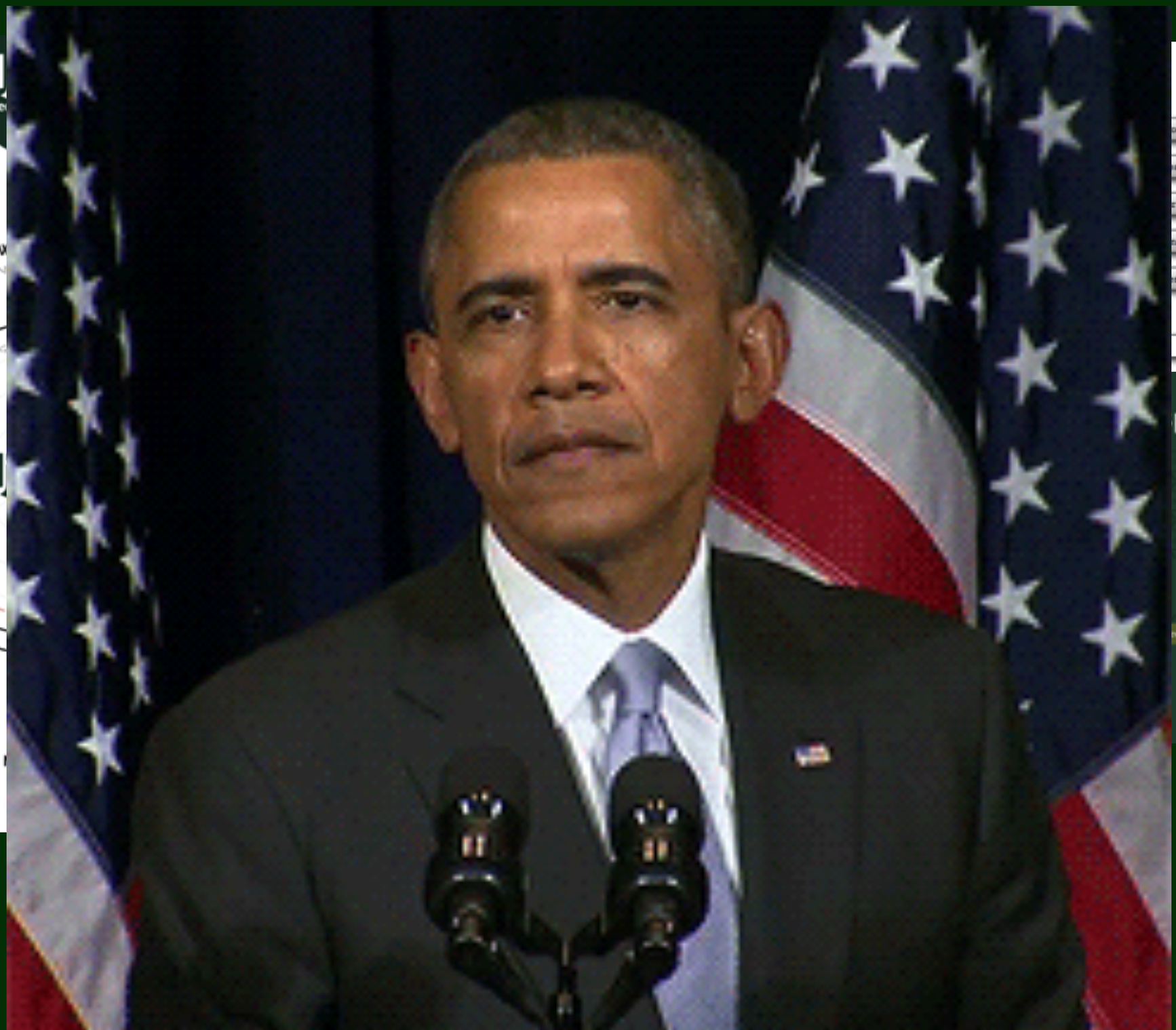
- "Form data is sent in the message body of a POST request."

- "Querystrings are sent in the URL of a GET request."

- "In Django, you generally test for the kind of request in the views."

DJANGO, BABY

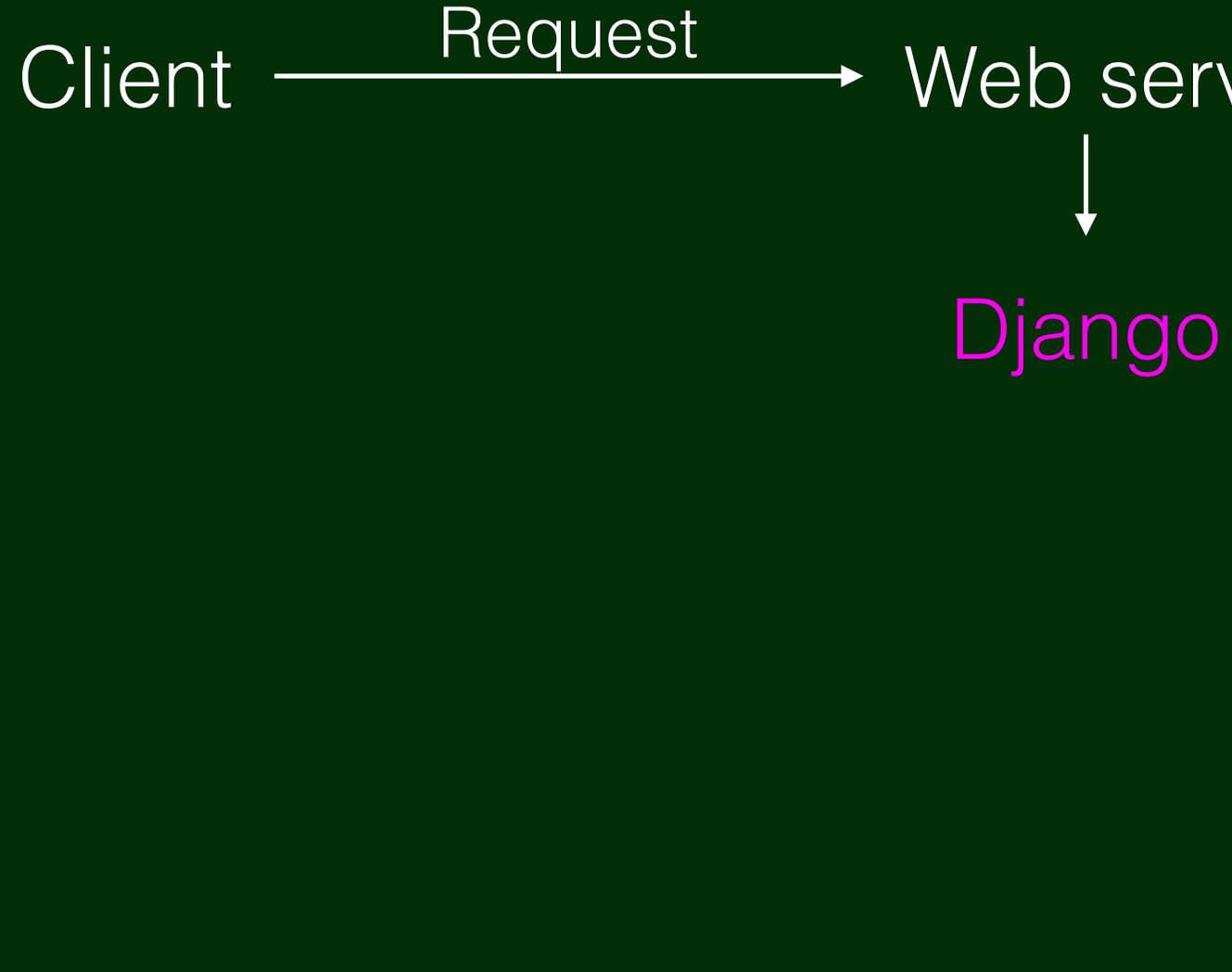
Request / Response cycle



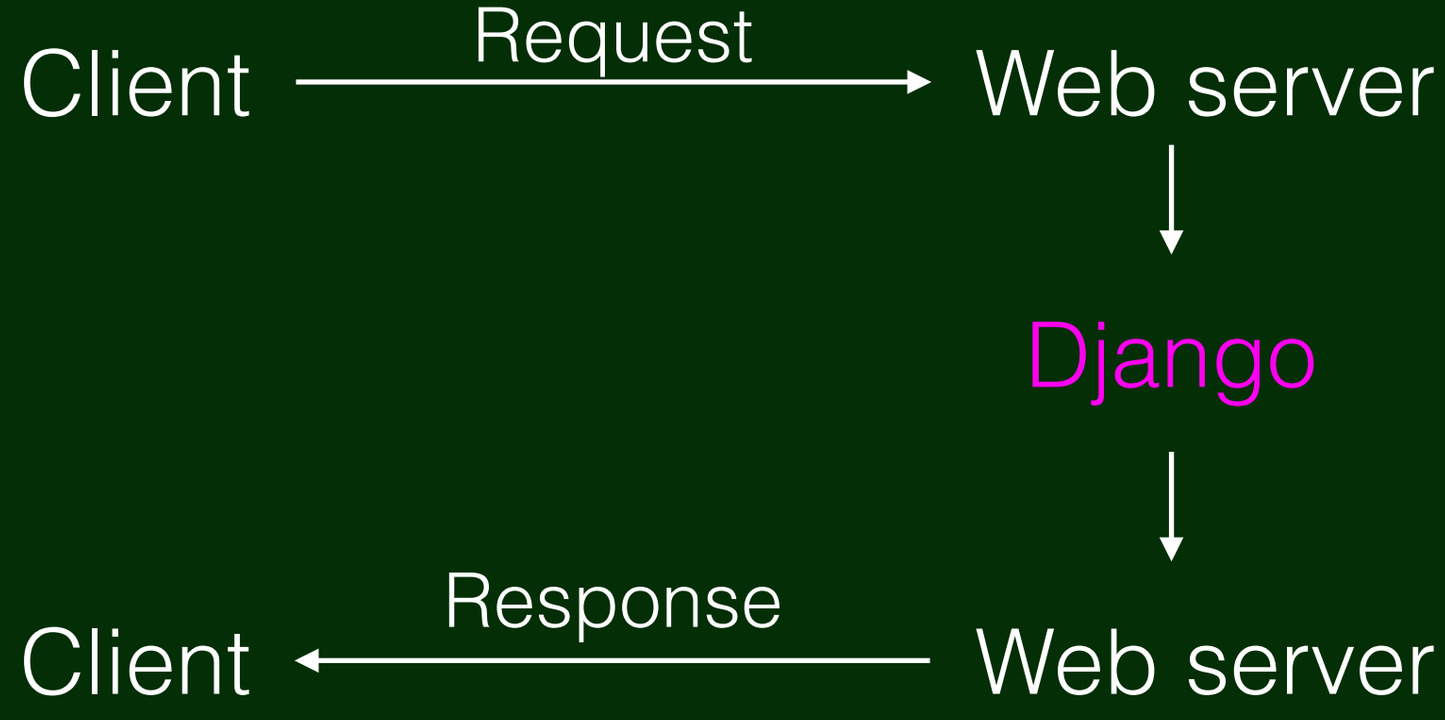
ponse Cycle

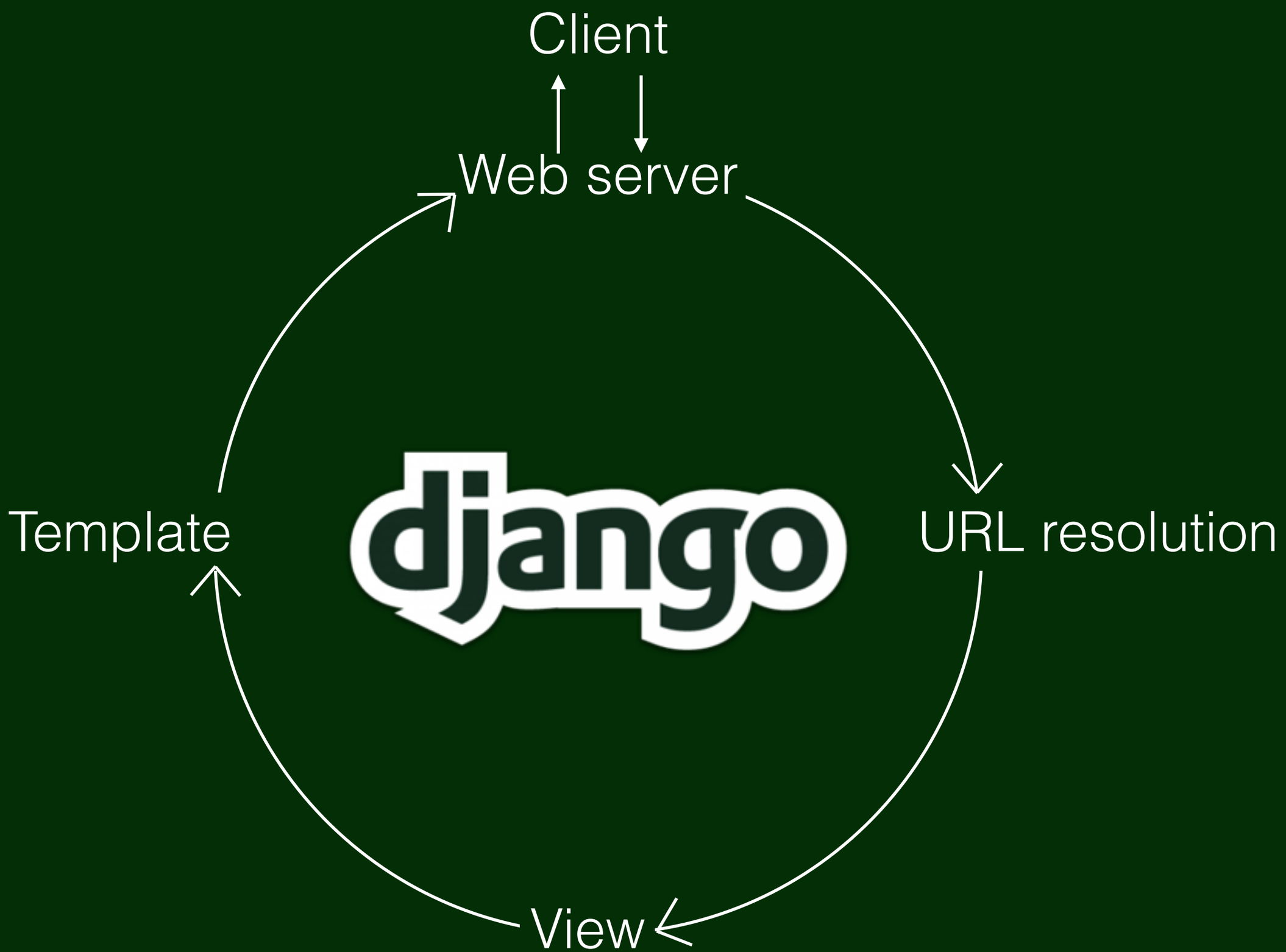
Request / Response cycle

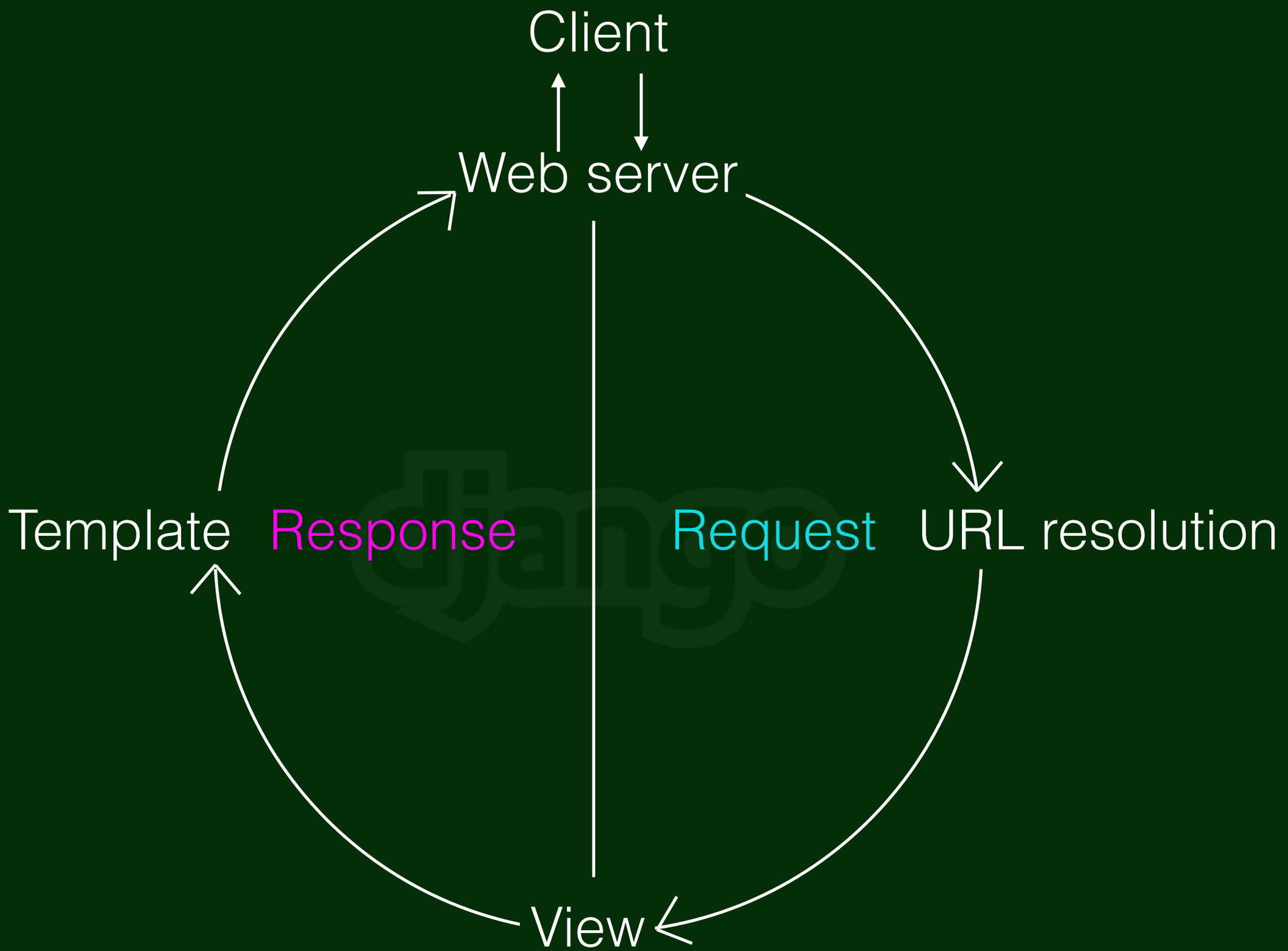
Request / Response cycle

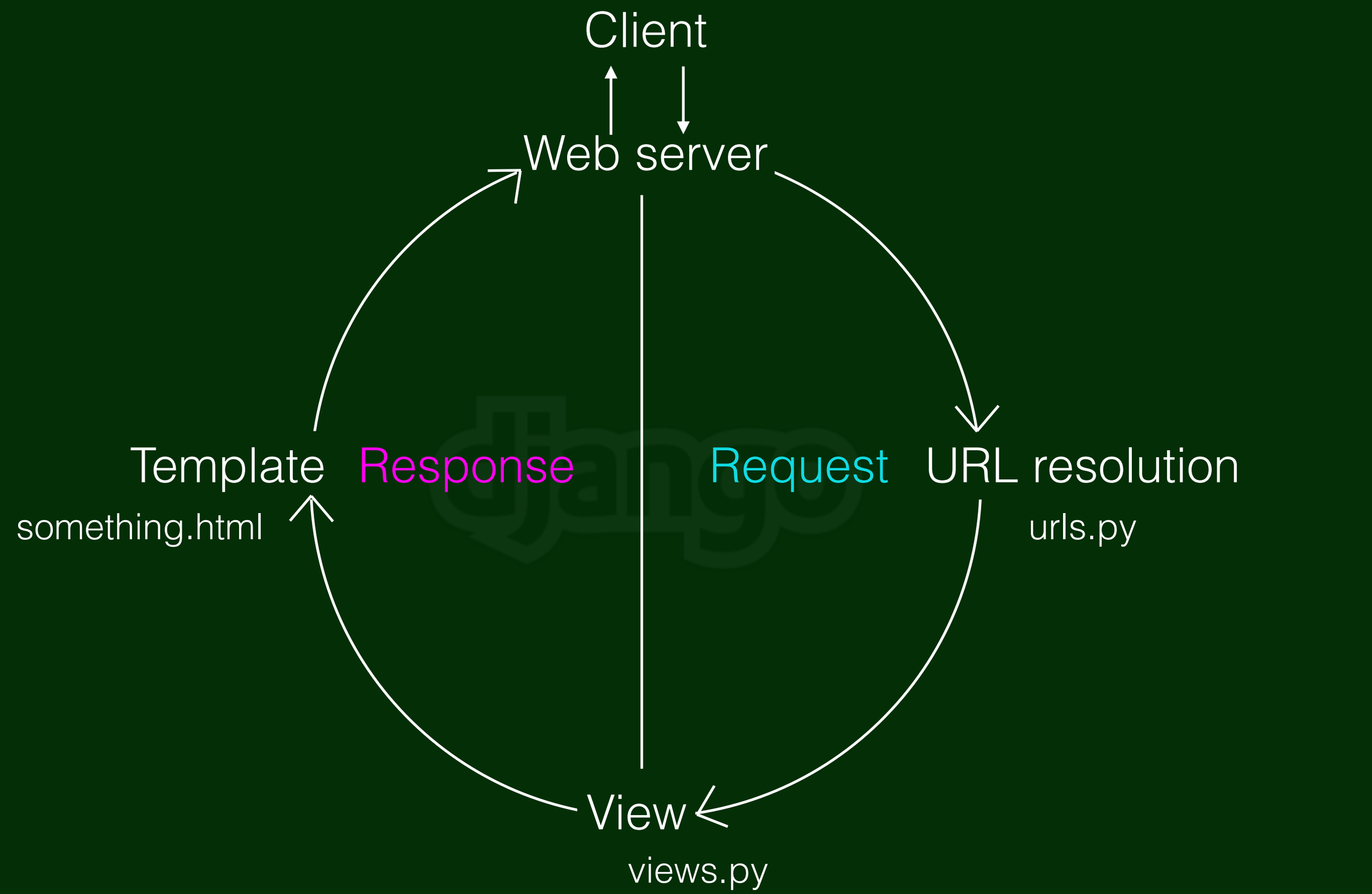


Request / Response cycle



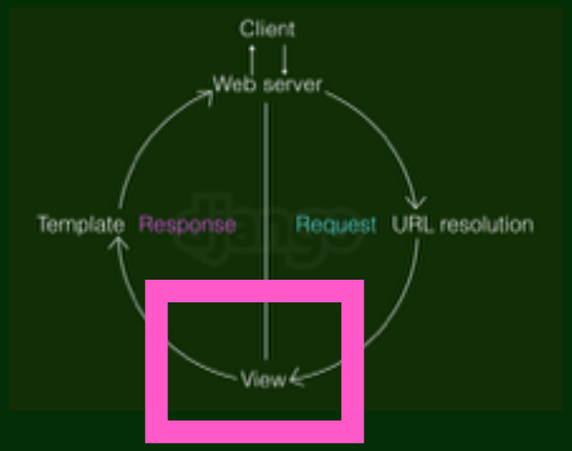

Client
Web server
URL resolution
View
Template
django


Client
Web server
Template
Response
Request
URL resolution
View


Client
Web server
Template
Response
Request
URL resolution
something.html
urls.py
View
views.py

Wait a second…

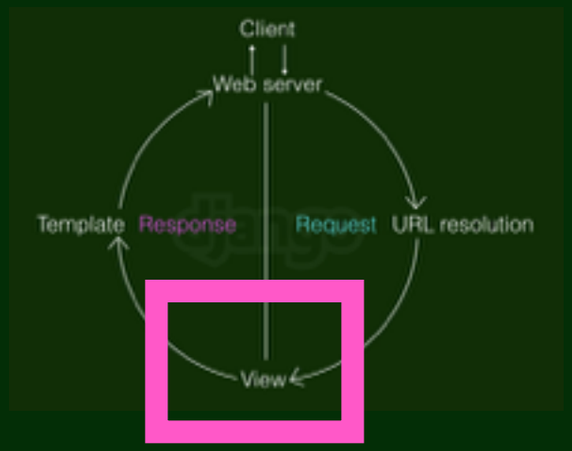
What about forms? And models?



View views.py

Model models.py

Database

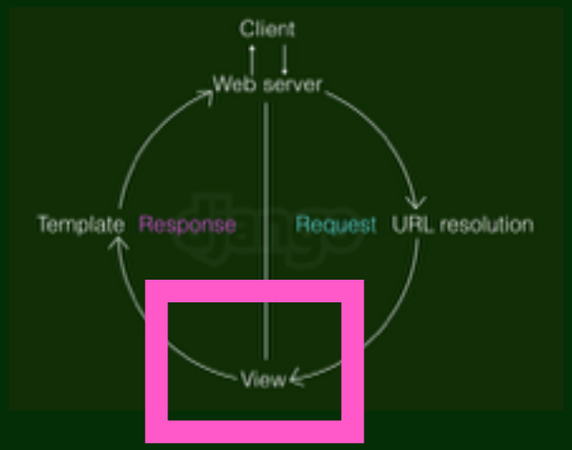


View views.py

Form forms.py

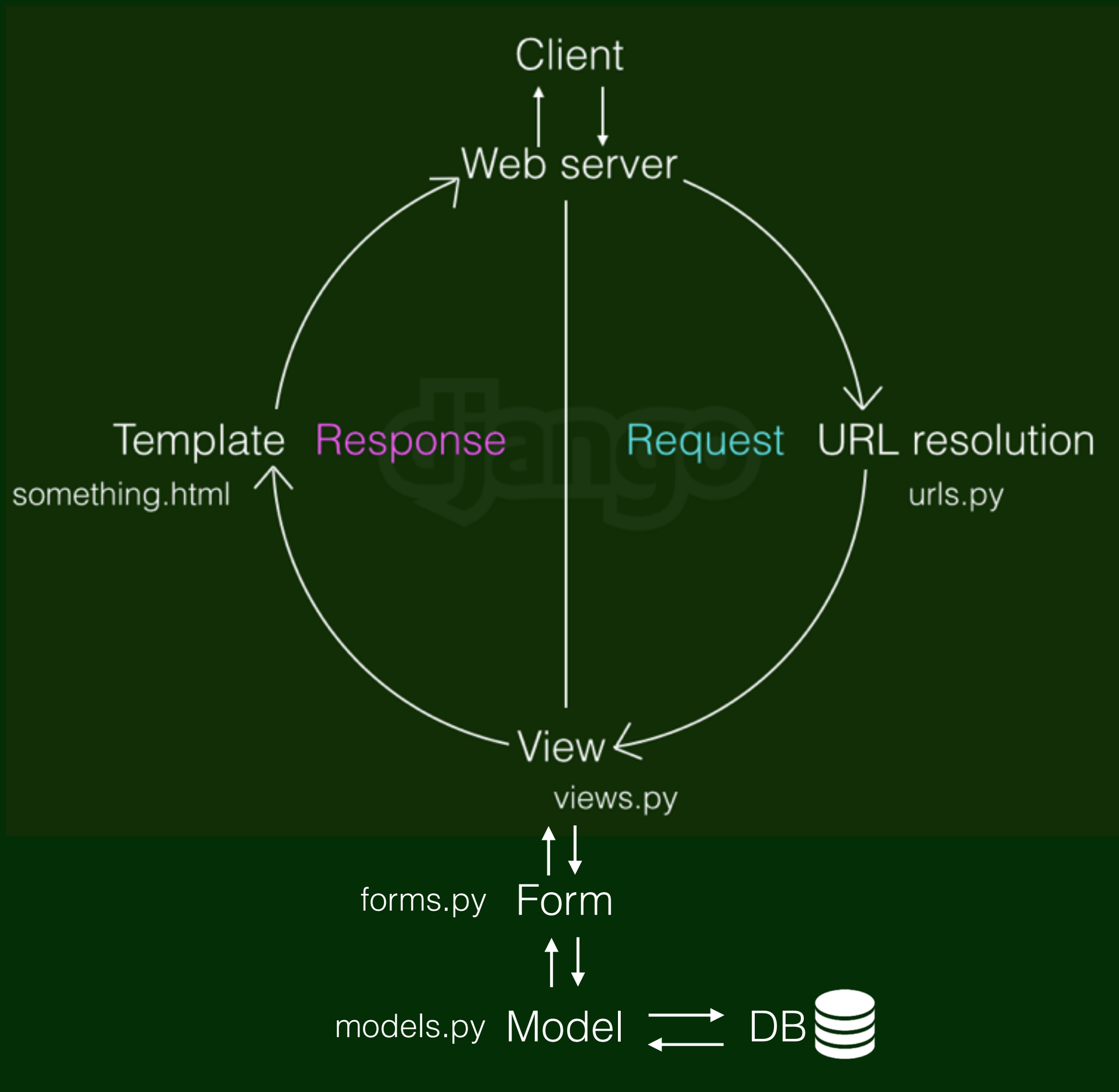
Model models.py

Database



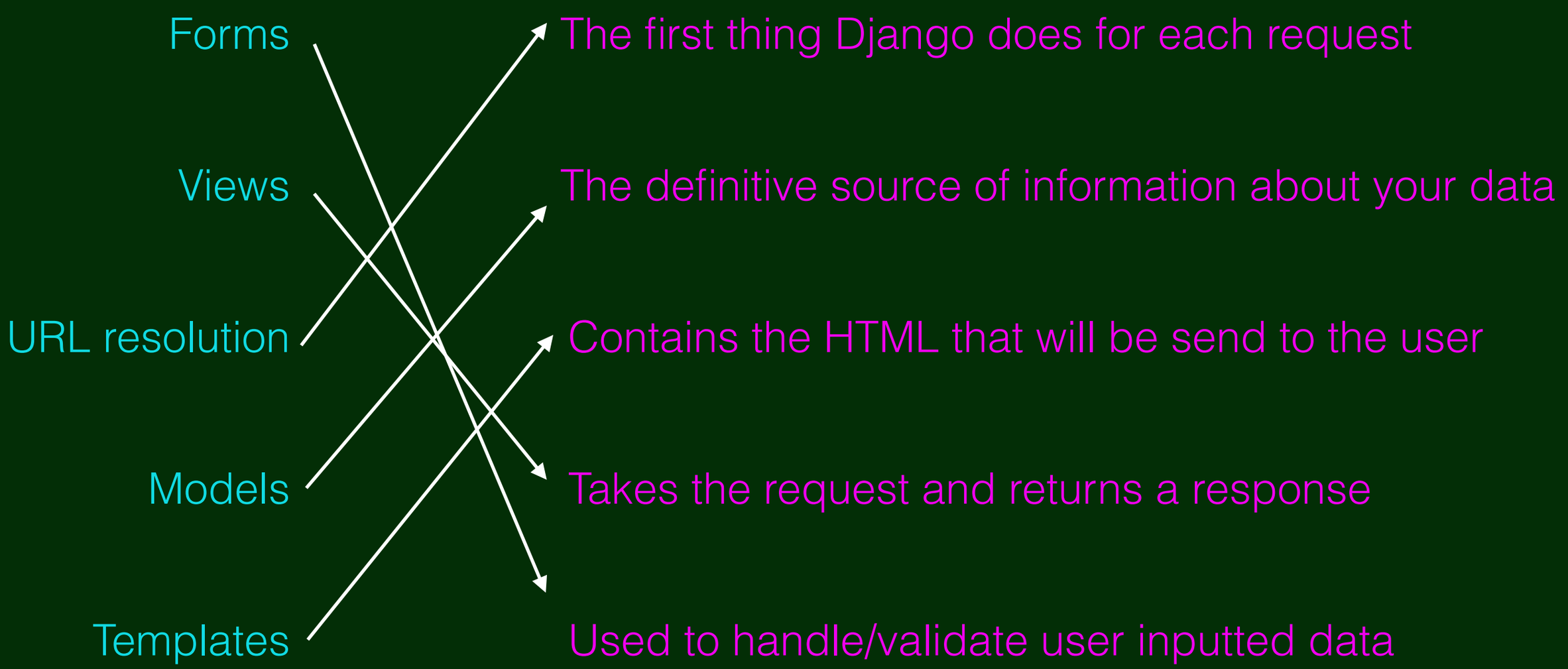
View views.py

Form forms.py


Client
Web server
Template
Response
Request
URL resolution
something.html
urls.py
View
views.py
forms.py
Form
models.py
Model
DB

Exercise!

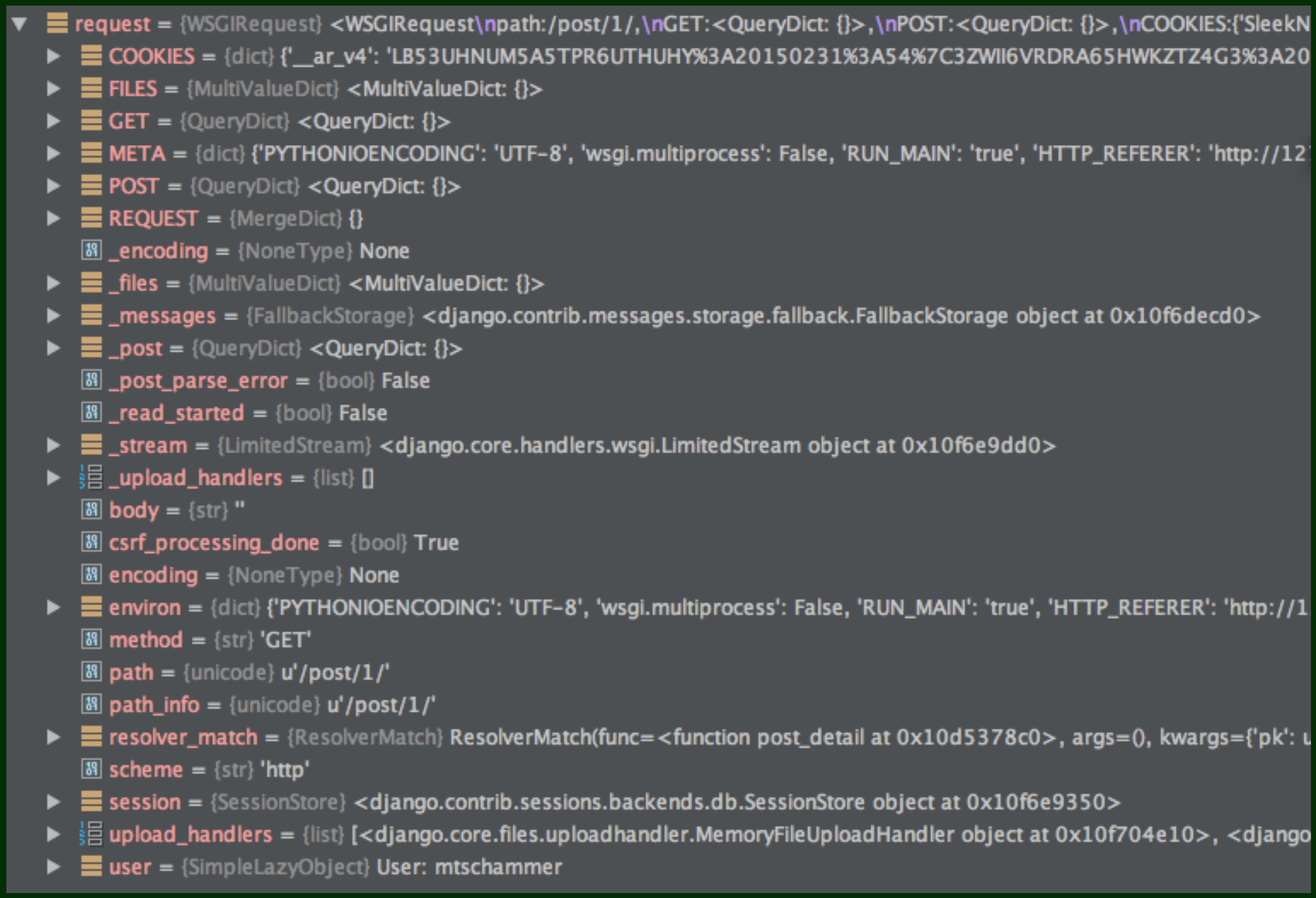
Connect the right terms with the right descriptions



WTF is the request object

```
22  def post_detail(request, pk):
23      post = Post.objects.get(pk=pk)
24      return render(
25          request,
26          'wall/post_details.html', {
27              'post': post,
28          }
29      )
```

request is AWESOME


request = {WSGIRequest} <WSGIRequest\npath:/post/1/,\nGET:<QueryDict: {}>,\nPOST:<QueryDict: {}>,\nCOOKIES:{'SleekN
COOKIES = {dict} {'__ar_v4': 'LB53UHNUM5A5TPR6UTHUHY%3A20150231%3A54%7C3ZWII6VRDRA65HWKZTZ4G3%3A20
FILES = {MultiValueDict} <MultiValueDict: {}>
GET = {QueryDict} <QueryDict: {}>
META = {dict} {'PYTHONIOENCODING': 'UTF-8', 'wsgi.multiprocess': False, 'RUN_MAIN': 'true', 'HTTP_REFERER': 'http://12
POST = {QueryDict} <QueryDict: {}>
REQUEST = {MergeDict} {}
_encoding = {NoneType} None
_files = {MultiValueDict} <MultiValueDict: {}>
_messages = {FallbackStorage} <django.contrib.messages.storage.fallback.FallbackStorage object at 0x10f6decd0>
_post = {QueryDict} <QueryDict: {}>
_post_parse_error = {bool} False
_read_started = {bool} False
_stream = {LimitedStream} <django.core.handlers.wsgi.LimitedStream object at 0x10f6e9dd0>
_upload_handlers = {list} []
body = {str} ''
csrf_processing_done = {bool} True
encoding = {NoneType} None
environ = {dict} {'PYTHONIOENCODING': 'UTF-8', 'wsgi.multiprocess': False, 'RUN_MAIN': 'true', 'HTTP_REFERER': 'http://1
method = {str} 'GET'
path = {unicode} u'/post/1/'
path_info = {unicode} u'/post/1/'
resolver_match = {ResolverMatch} ResolverMatch(func=<function post_detail at 0x10d5378c0>, args=(), kwargs={'pk': u
scheme = {str} 'http'
session = {SessionStore} <django.contrib.sessions.backends.db.SessionStore object at 0x10f6e9350>
upload_handlers = {list} [<django.core.files.uploadhandler.MemoryFileUploadHandler object at 0x10f704e10>, <django
user = {SimpleLazyObject} User: mtschammer

HOLY SHIT THAT'S A LOT OF USEFUL INFORMATION

request.user

request.GET

request.POST

request.user

- The user making the request
- Might be AnonymousUser
- Use user.is_authenticated() to test if user is authenticated.

request.GET

- Remember querystrings? This is where you find them.
- A “QueryDict”. Just use request.GET.get(“some_name”)

request.POST

- Form data!
- Also a QueryDict. Usually passed to a form.

A quick demonstration

DjDT

Django Debug Toolbar

Excuse me while I alt+tab

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Exercise

Some users are douchebags

1. Add a new user via admin (username: "**douchebag_user**")

2. Add a new model to the Wall project: **UserBlacklist**
- User (ForeignKey to auth.User)
- Reason for ban (TextField)
- Datetime (DatetimeField with default to now())

3. Add **douchebag_user** to user blacklist via admin

4. Modify the project, so that if **douchebag_user** tries to create a post or comment, he's redirected to a page with:
- Message telling him he is banned
- Why he is banned and as of what date

Bonus:
Modify the project so that a banned user cannot even view the list of posts.