
CHANNELS
Andrew Godwin
@andrewgodwin

Hi, I'm

Andrew Godwin

- Django core developer
- Senior Software Engineer at **Eventbrite**
- Used to complain about migrations a lot

It's magic.

~~It's magic.~~

1/ The Problem

The Web is changing.

WebSockets

Long-polling

WebRTC

WebSockets

MQTT

Server-Sent Events

Python is synchronous.

Django is synchronous.

Synchronous code is easier to write.

Single-process async is not enough.

Proven design pattern

Not too hard to reason about

What could fit these constraints?

2/ Loose Coupling

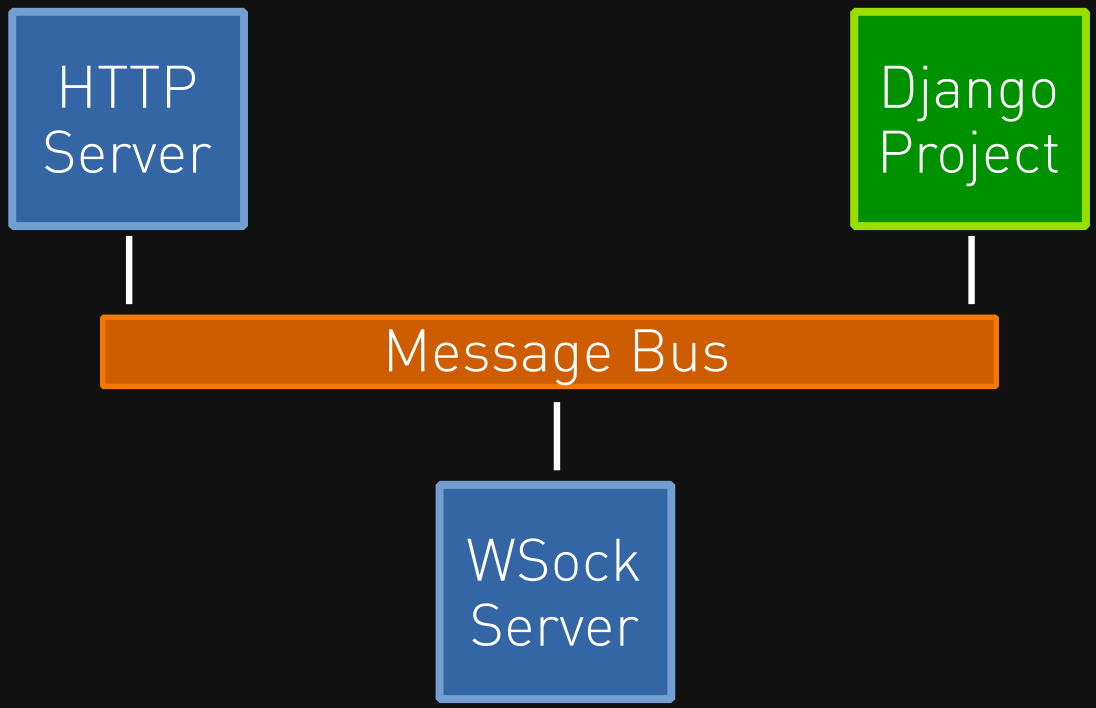
Not too tied to WebSockets

Not too tied to Django

Well-defined, minimal interfaces

Easy to swap out or rewrite

The Message Bus



What do you send?

How do you send it?

ASGI

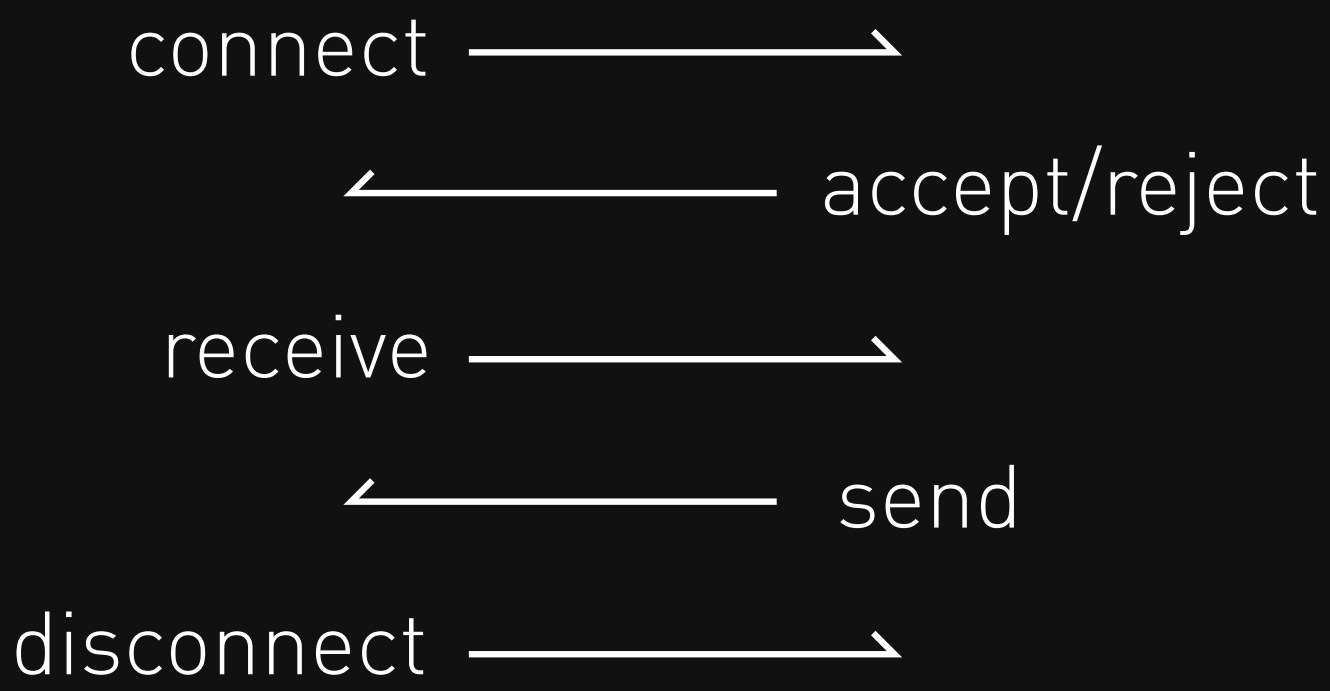
nonblocking send
blocking receive
add to group
discard from group
send to group

JSON-compatible,
dictionary-based
messages onto
named channels

3/ Concrete Ideas

Develop using concrete examples

WebSocket



WebSocket

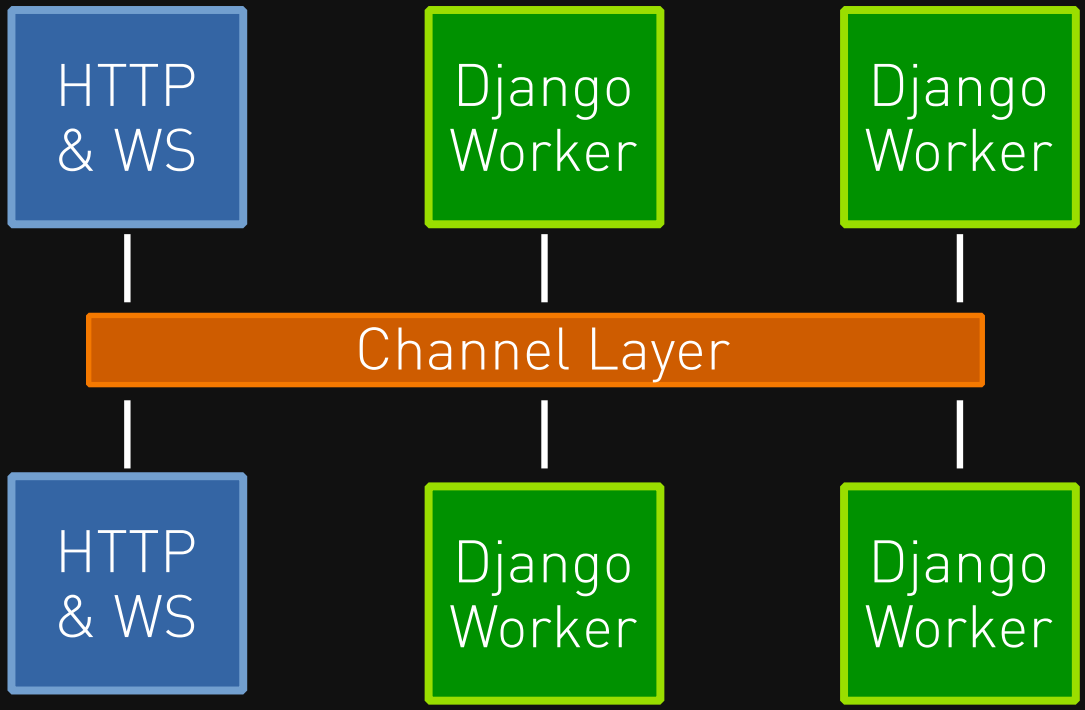
websocket.connect

websocket.receive

websocket.disconnect

websocket.send!abc1234

At-most-once
First In First Out
Backpressure via capacity
Not sticky
No guaranteed ordering
No serial processing


HTTP & WS
Django Worker
Django Worker
Channel Layer
HTTP & WS
Django Worker
Django Worker

```
{
    "text": "Hello, world!",
    "path": "/chat/socket/",
    "reply_channel": "websocket.send!9m12in2p",
}
```

Developed and spec'd

HTTP WebSocket

Rough drafts

IRC Email Slack

Please, no.

Minecraft Mainframe Terminal

```
{
    "reply_channel": "http.response!g23vD2x5",
    "method": "GET",
    "http_version": "2",
    "path": "/chat/socket/",
    "query_string": "foo=bar",
    "headers": [["cookie", "abcdef..."]],
}
```

At-most-once
First In First Out
Backpressure via capacity
Not sticky
No guaranteed ordering
No serial processing

At-most-once
First In First Out
Backpressure via capacity
Not sticky
No guaranteed ordering
No serial processing

"order" key on receive messages

Connection acceptance

Daphne

HTTP/WebSocket Server

Channels

Django integration

asgi-redis

Redis backend

asgi-ipc

Local memory backend

asgiref

Shared code and libs

4/ Django-ish

It can take several tries
to get a nice API.

Consumers based on Views

Callable that takes an object

Decorators for functionality

Class-based generics

```python
@channel_session
def chat_receive(message):
    name = message.channel_session["name"]
    message.reply_channel.send({"text": "OK"})
    Group("chat").send({
        "text": "%s: %s" % (name, message["text"]),
    })
    Message.objects.create(
        name=name,
        content=message["text"],
    )
```

Routing based on URLs

List of regex-based matches

Includes with prefix stripping on paths

More standardised interface

```
routing = [
    route(
        "websocket.receive",
        consumers.chat_receive,
        path=r"^/chat/socket/$",
    ),
    include("stats.routing", path="^/stats/"),
    route_class(ConsumerClass, path="^/v1/"),
]
```

Sessions are the only state

Sessions hang off reply channels not cookies

Uses same sessions backends

Available on the consumer's argument

Can also access long-term cookie sessions

```python
@enforce_ordering
def receive_consumer(message):
    Log.objects.create(...)
```

```
session = session_for_reply_channel(
    message.reply_channel.name
)
if not session.exists(session.session_key):
    try:
        session.save(must_create=True)
    except CreateError:
        # Session wasn't unique
        raise ConsumeLater()
message.channel_session = session
```

No Middleware

New-style middleware half works

No ability to capture sends

Decorators replace most cases

View/HTTP Django still there

Can intermingle or just use one type

View system is just a consumer now

```
def view_consumer(message):
  replies = AsgiHandler()(message)
  for reply in replies:
    while True:
      try:
        message.reply_channel.send(reply)
      except ChannelFull:
        time.sleep(0.05)
      else:
        break
```

Signals and commands

runserver works as expected

Signals for handling lifecycle

staticfiles configured for development

5/ Beyond

Generalised async communication

Service messaging

Security/CPU separation

Sync & Async / Py2 & Py3

Diversity of implementations

More web servers

More channel layers

More optimisation

More efficient bulk sends

Less network traffic on receive

More maintainers

More viewpoints, more time

1.0 coming soon

Stable APIs for everything except binding

Thanks.

Andrew Godwin
@andrewgodwin

channels.readthedocs.io
github.com/andrewgodwin/channels-examples