Channels

Saturday, 16 November 2024 4:07 PM

Introduction

* Django supports async view
* Channel is a wrapper around Django's async view, allowing it to handle long standing protocols like WebSockets, MQTT, Chatbots, amateur radio.
* It provides consumer & router which are equivalent to views & urls in WSGI django.
* Consumer makes the most basic unit of Channels
* Channels works with django to provide middleware like components for protocols like websockets
* Key feature of Channel is the communication implementation. "Messaging layer" used to send messages between the different parts of application eg. Two users sending & receiving messages. There will be two different instances of consumer for each user and the channel layes makes it possible to communicate between these two instances

* Channels & ASGI split up incoming connections into two components:
  1. Scope
  2. Series of events

Scope:

* Set of details about incoming connection

  Eg. IP address of incoming request
  The path from the web request was made from.
  User who made the request

* Scope persists throughout the connection

Events:

* Represents user interaction during the period of the scope

  Eg. Making the HTTP request
  Sending a websocket frame

Example

HTTP request:

```
GET /api/v1/data?filter=active HTTP/1.1
Host: example.com
User-Agent: curl/7.68.0
```

Scope:

```
{
    'type': 'http',   # The protocol type (HTTP in this case)
    'asgi': {
        'version': '3.0',
        'spec_version': '2.1'
    },
    'http_version': '1.1',
    'method': 'GET',   # HTTP method
    'path': '/api/v1/data',   # Path of the request
    'query_string': b'filter=active',   # Query string (bytes)
    'headers': [
        (b'host', b'example.com'),
        (b'user-agent', b'curl/7.68.0'),
    ],
    'client': ('127.0.0.1', 52134),   # Client IP and port
    'server': ('127.0.0.1', 8000),   # Server IP and port
    'scheme': 'http',   # URL scheme
}
```

Event:

ASGI Events for HTTP

For HTTP, there are typically two events:

1. `http.request`: The body of the HTTP request is sent in one or more chunks.
   * Example:

```python
{
    'type': 'http.request',
    'body': b'',   # Request body (empty for GET requests)
    'more_body': False,   # True if more chunks are coming
}
```

2. `http.response.start` and `http.response.body`: Sent by the application to send the HTTP response.

☆ Channels & ASGI applications (consumer) are instantiated once per SCOPE

Example

HTTP

1. User makes HTTP request
2. An instance of Consumer is created based on the url pattern request is made
3. An HTTP type scope is opened containg request's path, methods, headers etc.
4. ASGI & Channel instance generates http.response event to send back to the browser & close the application
5. The HTTP request is completed, so the Scope is deleted.

Websocket

1. User sends first message to chatbot
2. This initiates consumer corresponding to the url pattern
3. A scope is opened, containing user's username, chosen name, user ID, etc.
4. The application is given a chat.recieve_message event with the event text. It does not have to respond, but could send one, or more chat message back as chat.send-message event.

How does ASGI & Channel work together?

1. ASGI Servers
   * Handles incoming connections (HTTP, Websockets)
   * Break them into Scope & events
   * Routes the connections to Django channels based on ASGI_APPLICATION in settings.py

2. Channels Framework
   * Process the connection using the configured URL router
   * Instantiates the appropriate Consumer class
   * Manages the connection's lifecycle & handles events using Consumer methods:
     * connect
     * receive
     * disconnect

3. Channel Layer
   * Allows consumers to communicate between each other

ASGI Configuration

```python
# mysite/asgi.py
import os

from channels.auth import AuthMiddlewareStack
from channels.routing import ProtocolTypeRouter, URLRouter
from channels.security.websocket import AllowedHostsOriginValidator
from django.core.asgi import get_asgi_application

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "mysite.settings")
# Initialize Django ASGI application early to ensure the AppRegistry
# is populated before importing code that may import ORM models.
django_asgi_app = get_asgi_application()

from chat.routing import websocket_urlpatterns

application = ProtocolTypeRouter(
    {
        "http": django_asgi_app,
        "websocket": AllowedHostsOriginValidator(
            AuthMiddlewareStack(URLRouter(websocket_urlpatterns))
        ),
    }
)
```

* ProtocolTypeRouter inspects the connection, if its HTTP then its routed to django-asgi-app.
* If the connetion is websocket its given to AuthMiddlewareStack.
* AuthMiddlewareStack will populate the connection's scope with a reference to authenticated user.
* This is then passed to URLRouter.
* The URLRouter then inspects the path and forwards it to the particular consumer.