Python: usare WebRTC

WebRTC, acronimo di Web Real-Time Communication, è una tecnologia open-source che consente la comunicazione in tempo reale tra browser e applicazioni web senza la necessità di plugin aggiuntivi. Questa tecnologia è diventata sempre più popolare per lo sviluppo di applicazioni di videochiamata, chat e trasmissioni audio/video in tempo reale su Internet.

Python è uno dei linguaggi di programmazione più diffusi al mondo, ed è possibile utilizzarlo insieme a WebRTC per creare applicazioni web avanzate e interattive. In questa guida, esploreremo come utilizzare WebRTC in Python per creare una semplice applicazione di videochiamata utilizzando la libreria `aiortc`.

Cos'è `aiortc`?

`aiortc` è una libreria Python che fornisce un'interfaccia per utilizzare WebRTC in applicazioni Python. Essa offre funzionalità per creare sia client che server WebRTC, consentendo agli sviluppatori di implementare facilmente la comunicazione in tempo reale nelle proprie applicazioni.

Installazione di `aiortc`

Per utilizzare `aiortc`, è necessario installare la libreria utilizzando pip, il gestore di pacchetti Python. Eseguire il seguente comando nel terminale:

```
pip install aiortc
```

Con `aiortc` installato nel nostro ambiente virtuale, siamo pronti per creare la nostra applicazione di videochiamata.

Creazione di un semplice server di videochiamata

Di seguito, viene fornito un esempio di come creare un semplice server di videochiamata utilizzando `aiortc`. Questo server consentirà a due utenti di connettersi e comunicare tra loro tramite video e audio.

```
import asyncio
from aiohttp import web
from aiortc import RTCPeerConnection,
RTCSessionDescription
from aiortc.contrib.aiohttp import websockets

async def index(request):
    content = open("index.html", "r").read()
    return web.Response(content_type="text/html",
text=content)

async def websocket_handler(request):
    ws = web.WebSocketResponse()
    await ws.prepare(request)

    pc = RTCPeerConnection()

    @pc.on("track")
    def on_track(track):
        print("Track received")
        pc.addTrack(track)

    async for msg in ws:
        if msg.type == web.WSMsgType.TEXT:
            if msg.data == 'offer':
```

```
                offer = await ws.receive_json()
                await
pc.setRemoteDescription(RTCSessionDescription(offer))
                answer = await pc.createAnswer()
                await pc.setLocalDescription(answer)
                await ws.send_json(answer)
            elif msg.data == 'answer':
                answer = await ws.receive_json()
                await
pc.setRemoteDescription(RTCSessionDescription(answer)
)
            elif msg.data == 'candidate':
                candidate = await ws.receive_json()
                candidate = candidate['candidate']
                await pc.addIceCandidate(candidate)
        elif msg.type == web.WSMsgType.ERROR:
            print('ws connection closed with
exception %s' %
                  ws.exception())

    return ws

app = web.Application()
app.router.add_get("/", index)
app.router.add_get("/ws", websocket_handler)

if __name__ == "__main__":
    web.run_app(app)
```

Creazione del file HTML per il client

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
width, initial-scale=1.0">
    <title>WebRTC Video Chat</title>
</head>
<body>
    <video id="localVideo" autoplay></video>
    <video id="remoteVideo" autoplay></video>
    <script>
        const localVideo =
document.getElementById('localVideo');
        const remoteVideo =
document.getElementById('remoteVideo');

        const ws = new WebSocket('ws://' +
window.location.host + '/ws');

        const pc = new RTCPeerConnection();

        pc.ontrack = function(event) {
            console.log("Track received");
            remoteVideo.srcObject = event.streams[0];
        };

        pc.onicecandidate = function(event) {
            if (event.candidate) {
                ws.send(JSON.stringify({
                    type: 'candidate',
                    candidate: event.candidate
                }));
```

```
            }
        };

        async function startCall() {
            const offer = await pc.createOffer();
            await pc.setLocalDescription(offer);
            ws.send('offer');
            ws.send(JSON.stringify(offer));
        }

        ws.onmessage = async function(event) {
            const data = JSON.parse(event.data);
            if (data.type === 'answer') {
                await pc.setRemoteDescription(new
RTCSessionDescription(data));
            }
        };

        startCall();
    </script>
</body>
</html>
```

Esecuzione dell'applicazione

Per eseguire l'applicazione, salvare il codice Python in un file chiamato `server.py` e il codice HTML in un file chiamato `index.html`. Assicurarsi che entrambi i file siano nella stessa directory.

Successivamente, eseguire il seguente comando nel terminale:

```
python server.py
```

Aprire quindi il browser e accedere all'indirizzo `http://localhost:8080`. Dovresti vedere due video, uno per la tua webcam locale e uno per la webcam remota (del tuo partner di videochiamata, se presente).

Conclusioni

In questa guida, abbiamo esplorato come utilizzare WebRTC in Python per creare un semplice server di videochiamata utilizzando la libreria `aiortc`. Questo è solo un esempio di come è possibile utilizzare WebRTC per implementare la comunicazione in tempo reale nelle proprie applicazioni Python. Conoscendo questi fondamenti, puoi esplorare ulteriormente le funzionalità di WebRTC e sviluppare applicazioni web sempre più avanzate e interattive.