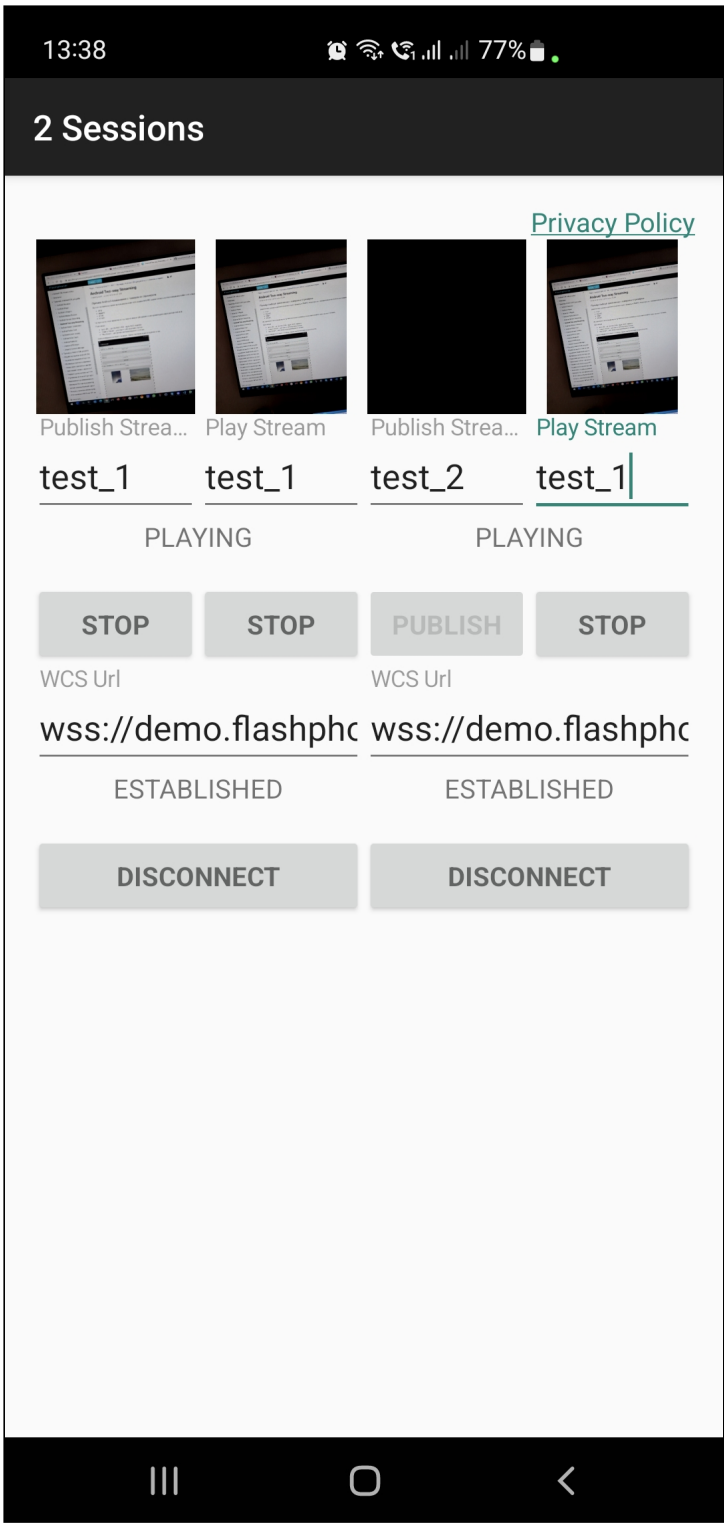
Android 2 Sessions

Пример Android приложения с двумя Websocket сессиями

Данное приложение демонстрирует использование одновременно двух Websocket сессий для подключения к одному или двум разным WCS серверам. Публиковать при этом можно только один поток, т.к. Android не позволяет захватить одновременно две камеры, либо дважды одну и ту же камеру. Воспроизводить можно одновременно два потока.

На скриншоте представлен пример публикации на один сервер и проигрывания этого потока в двух сессиях



Работа с кодом примера

Для разбора кода возьмем класс TwoSessionsActivity.java примера `2sessions`, который доступен для скачивания в соответствующей сборке 1.1.0.57.

Реализация представляет собой два примера Two Way Streaming в одном приложении, поэтому далее рассмотрим подробно код одной из двух сессий

1. Инициализация API

`Flashphoner.init()` code

Независимо от количества сессий, API инициализируется один раз на приложение

```
Flashphoner.init(this);
```

2. Создание сессии

`Flashphoner.createSession()` code

Методу передается объект `SessionOptions` со следующими параметрами

- URL WCS-сервера
- `SurfaceViewRenderer local1Renderer`, который будет использоваться для отображения видео с камеры
- `SurfaceViewRenderer remote1Renderer1, который будет использоваться для отображения воспроизводимого потока

```
SessionOptions sessionOptions = new
SessionOptions(mWcsUrl1View.getText().toString());
sessionOptions.setRemoteRenderer(remote1Render);
sessionOptions.setLocalRenderer(local1Render);

/**
 * Session for connection to WCS server is created with method
createSession().
 */
session1 = Flashphoner.createSession(sessionOptions);
```

3. Подключение к серверу

`Session.connect()` code

```
session1.connect(new Connection());
```

4. Получение от сервера события, подтверждающего успешное соединение

`session.onConnected()` code

По данному событию запрашиваются права на доступ к камере и микрофону

```
@Override
public void onConnected(final Connection connection) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            mConnect1Button.setText(R.string.action_disconnect);
            mConnect1Button.setTag(R.string.action_disconnect);
            mConnect1Button.setEnabled(true);
            mConnect1Status.setText(connection.getStatus());
            mPlay1Button.setEnabled(true);
            if (mPublish2Button.getTag() == null ||
Integer.valueOf(R.string.action_publish).equals(mPublish2Button.getTag())) {
                mPublish1Button.setEnabled(true);
            }

            ActivityCompat.requestPermissions(TwoSessionsActivity.this,
                    new String[]{Manifest.permission.RECORD_AUDIO,
Manifest.permission.CAMERA},
                    PUBLISH_REQUEST_CODE);
        }
    });
}
```

5. Публикация потока по нажатию кнопки `Publish1Button`

`Session.createStream()`, `Stream.publish()` code

```
mPublish1Button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View view) {
        mPublish1Button.setEnabled(false);
        mPublish2Button.setEnabled(false);
        mConnect1Button.setEnabled(false);
        mConnect2Button.setEnabled(false);
        if (mPublish1Button.getTag() == null ||
Integer.valueOf(R.string.action_publish).equals(mPublish1Button.getTag())) {
            /**
             * The options for the stream to play are set.
             * The stream name is passed when StreamOptions object is
created.
             * SurfaceViewRenderer to be used to display the video stream is
set using method StreamOptions.setRenderer().
             */
            StreamOptions streamOptions = new
StreamOptions(mPublish1StreamView.getText().toString());
            streamOptions.setRenderer(local1Render);

            /**
             * Stream is created with method Session.createStream().
             */
            publish1Stream = session1.createStream(streamOptions);
            ...
            publish1Stream.publish();
```

```
        } else {
            ...
        }
    }
});
```

6. Воспроизведение потока при нажатии `Play1Button`

`Session.createStream()`, `Stream.play()` code

```
mPlay1Button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View view) {
        mPlay1Button.setEnabled(false);
        if (mPlay1Button.getTag() == null ||
Integer.valueOf(R.string.action_play).equals(mPlay1Button.getTag())) {
            /**
             * The options for the stream to play are set.
             * The stream name is passed when StreamOptions object is
created.
             * SurfaceViewRenderer to be used to display the video stream is
set using method StreamOptions.setRenderer().
             */
            StreamOptions streamOptions = new
StreamOptions(mPlay1StreamView.getText().toString());
            streamOptions.setRenderer(remote1Render);

            /**
             * Stream is created with method Session.createStream().
             */
            play1Stream = session1.createStream(streamOptions);
            ...
            play1Stream.play();
            ...
        } else {
            ...
        }
    }
});
```

7. Остановка воспроизведение потока

`Stream.stop()` code

```
mPlay1Button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View view) {
        mPlay1Button.setEnabled(false);
        if (mPlay1Button.getTag() == null ||
Integer.valueOf(R.string.action_play).equals(mPlay1Button.getTag())) {
            ...
        } else {
            /**
```

```
             * Method Stream.stop() is called to stop playback of the stream.
             */
            play1Stream.stop();
            play1Stream = null;
        }
        ...
    }
});
```

8. Остановка публикации потока

`Stream.stop()` code

```
mPublish1Button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View view) {
        mPublish1Button.setEnabled(false);
        mPublish2Button.setEnabled(false);
        mConnect1Button.setEnabled(false);
        mConnect2Button.setEnabled(false);
        if (mPublish1Button.getTag() == null ||
Integer.valueOf(R.string.action_publish).equals(mPublish1Button.getTag())) {
            ...
        } else {
            /**
             * Method Stream.stop() is called to stop playback of the stream.
             */
            publish1Stream.stop();
            publish1Stream = null;
        }
        ...
    }
});
```

9. Закрытие соединения

`Session.disconnect()` code

```
mConnect1Button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View view) {
        if (mConnect1Button.getTag() == null ||
Integer.valueOf(R.string.action_connect).equals(mConnect1Button.getTag())) {
            ...
        } else {
            mConnect1Button.setEnabled(false);

            /**
             * Connection to WCS server is closed with method
Session.disconnect().
             */
            session1.disconnect();
        }
```

```
        ...
    }
});
```

10. Получение события, подтверждающего разъединение

`session.onDisconnection()` code

```
public void onDisconnection(final Connection connection) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            mConnect1Button.setText(R.string.action_connect);
            mConnect1Button.setTag(R.string.action_connect);
            mConnect1Button.setEnabled(true);
            mPlay1Button.setText(R.string.action_play);
            mPlay1Button.setTag(R.string.action_play);
            mPlay1Button.setEnabled(false);
            mPublish1Button.setEnabled(false);
            mPublish1Button.setText(R.string.action_publish);
            mPublish1Button.setTag(R.string.action_publish);
            mConnect1Status.setText(connection.getStatus());
            mPlay1Status.setText("");
            if (mConnect2Button.getTag() == null ||
Integer.valueOf(R.string.action_disconnect).equals(mConnect2Button.getTag()))
{
                mPublish2Button.setEnabled(true);
            }
        }
    });
}
```