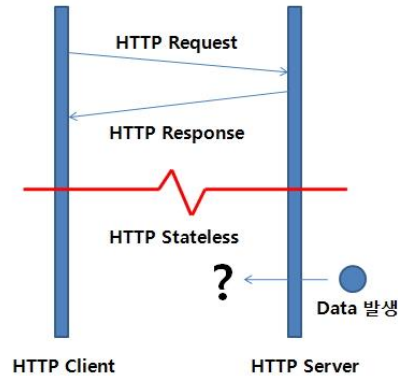




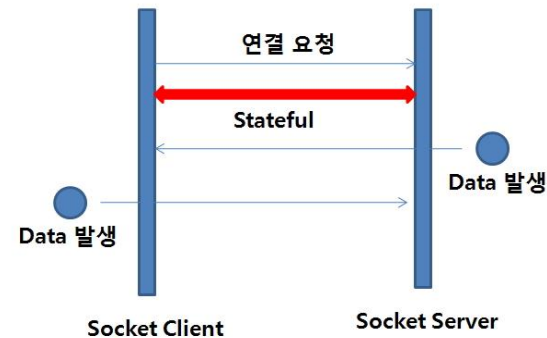
# 26.1 소켓 프로그램

## 26.1.1. 소켓 프로그램 작성 방법

- HTTP 통신



- 소켓 프로그램



# 26.1 소켓 프로그램

- 안드로이드에서 소켓 프로그램은 HTTP와 마찬가지로 Java API를 그대로 이용하여 구현

```
<uses-permission android:name="android.permission.INTERNET"/>
```

- 서버와 소켓 연결

```
Socket socket = new Socket();  
SocketAddress remoteAddr = new InetSocketAddress(serverIp, serverPort);  
socket.connect(remoteAddr, 10 * 1000);
```

- 데이터를 송수신 IO 객체를 생성

```
BufferedOutputStream bout = new BufferedOutputStream(socket.getOutputStream());  
BufferedInputStream bin = new BufferedInputStream(socket.getInputStream());
```

- 데이터를 송신

```
bout.write(((String) msg.obj).getBytes());  
bout.flush();
```



# 26.1 소켓 프로그램

- 데이터를 수신

```
String message = null;
int size = bin.read(buffer);
if (size > 0) {
    message = new String(buffer, 0, size, "utf-8");
}
```

## 26.1.2. 소켓 프로그램 작성 시 주의점

- Connection, Read, Write 모두 스레드로 처리
- Read 행위를 스레드로 처리

```
class ReadThread extends Thread {
    @Override
    public void run() {
        byte[] buffer = null;
        while (flagRead) {
            buffer = new byte[1024];
            try {
                String message = null;
                int size = bin.read(buffer);
                if (size > 0) {
                    //read 후 업무처리
                }
            } catch (IOException e) {
            }
        }
    }
}
```

# 26.1 소켓 프로그램

- 서버 연결 부분을 스레드로 처리

```
class SocketThread extends Thread {  
    public void run() {  
        while (flagConnection) {  
            try {  
                if (!isConnected) {  
                    socket = new Socket();  
                    SocketAddress remoteAddr = new InetSocketAddress(serverIp,  
serverPort);  
                    socket.connect(remoteAddr, 10 * 1000);  
                    //...  
                } else {  
                    SystemClock.sleep(10000);  
                }  
            } catch (Exception e) {  
            }  
        }  
    }  
}
```



# 26.1 소켓 프로그램

- Write 스레드

```
class WriteThread extends Thread {  
    @Override  
    public void run() {  
        Looper.prepare();  
        writeHandler = new Handler(){  
            @Override  
            public void handleMessage(Message msg) {  
                try {  
                    bout.write(((String) msg.obj).getBytes());  
                    bout.flush();  
                    //...                }catch (Exception e){                }  
                }  
            };  
        Looper.loop();  
    }  
}
```

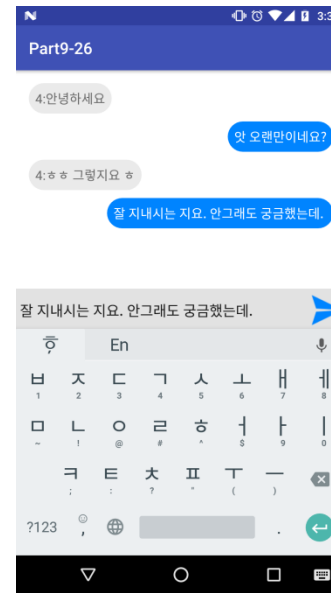


# Step by Step 26-1 – 소켓 통신

## Socket 프로그램을 테스트

- 간단한 채팅이며 동일 서버에 접속한 모든 유저가 하나의 채팅방에서 채팅을 한다는 개념
- Socket 연결 및 데이터 송수신은 Activity에서 처리할 것이며 본 교제가 안드로이드 앱 개발이 목적으로 Socket 서버는 공유된 파일을 이용

1. Module 생성
2. 파일 복사
3. AndroidManifest.xml 작업
4. MainActivity.java 작성
5. 서버 준비
6. 실행





## 26.2 Firebase Cloud Message

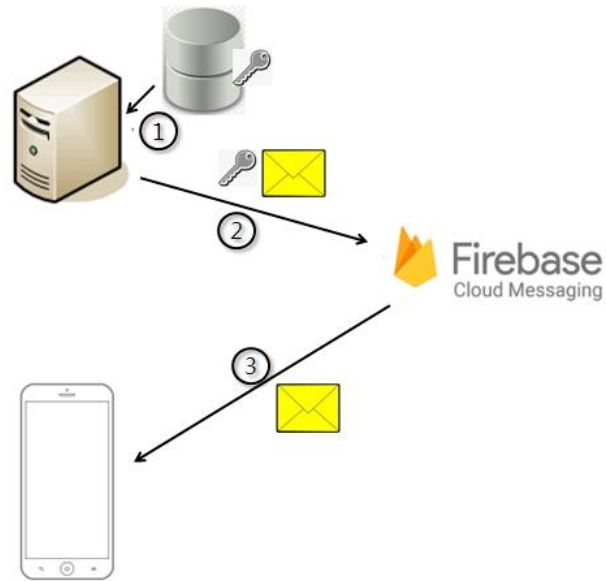
- Firebase는 구글의 모바일 앱 개발 통합 플랫폼이며, Firebase에서 제공하는 여러 가지 서비스 중 하나가 FCM
- FCM은 2016년 구글 I/O 행사에서 Firebase를 개선하여 GCM(Google Cloud Message)을 대체하기 위해 선보인 서비스
- FCM의 동작 원리
- 첫 번째 단계가 앱을 위한 키를 FCM 서버를 통해 얻는 단계





## 26.2 Firebase Cloud Message

- 두 번째 단계로 서버에서 데이터를 스마트폰에 전달



# Step by Step 26-2 – Firebase 클라우드 메시지

FCM 방식으로 서버의 데이터를 받은후 앱에서 Notification을 띄워 유저에게 데이터 송신을 알려주는 방식으로 테스트

1. SHA1 지문 획득
2. Firebase Console 작업
3. gradle 설정
4. FirebaseMessagingService 작성
5. FirebaseInstanceIdService 작성
6. 앱 실행
7. NodeJS로 FCM 데이터 전달

