18 네트워크 프로그래밍

대부분 앱은 네트워크 통신을 이용해 서버에서 데이터를 가져와 화면에 출력하거나 앱에서 발생한 데이터를 서버에 저장한다. 이처럼 서버와 네트워크 통신을 하려면 HTTP 통신 방법과 기기의 각종 네트워크 정보를 확인하는 방법을 알아야 한다.

1. 스마트폰 정보 구하기

- 전화 정보

앱에서 전화 기능을 이용하려면 PhoneStateListener를 상속받아 그 객체를 TelePhonyManager에 등록해야 한다. 그러면 스마트폰의 전화 관련 상태가 바뀔 때마다 PhoneStateListner의 함수가 자동으로 호출된다.

- **onCallForwardingIndicatorChanged**(boolean cfi): 통화 전달 상태 변경
- **onCallStateChanged**(int state, String incomingNumber): 통화 상태 변경
- **onCellLocationChanged**(CellLocation location): 폰의 기지국 위치 변경
- **onDataActivity**(int direction): 데이터 송수신 활동
- **onDataConnectionStateChanged**(int state, int networkType): 데이터 연결 상태 변경
- **onMessageWaitingIndicatorChanged**(boolean mwi): 메시지 대기 상태 변경
- **onServiceStateChanged**(ServiceState serviceState): 단말기의 서비스 상태 변경
- **onSignalStrengthsChanged**(SignalStrength signalStrength): 신호 세기 변경

listen() 함수의 두 번째 매개변수에는 감지할 상태를 지정해야 하는데 이때 PhoneState Listener 뒤에 다음과 같은 상수를 사용합니다.

- LISTEN_CALL_FORWARDING_INDICATOR: 통화 전달 지시자
- LISTEN_CALL_STATE: 통화 상태
- LISTEN_CELL_LOCATION: 기지국 위치
- LISTEN_DATA_ACTIVITY: 데이터 송수신 활동
- LISTEN_DATA_CONNECTION_STATE: 데이터 연결 상태
- LISTEN_MESSAGE_WAITING_INDICATOR: 메시지 대기 지시자
- LISTEN_SERVICE_STATE: 단말기의 서비스 상태
- LISTEN_SIGNAL_STRENGTHS: 신호 세기

– 서비스 상태 감지

onServiceStateChanged() 함수는 비행 모드나 긴급 통화 등 스마트폰의 서비스 상태가 바뀌는 순간 호출된다.

스마트폰의 서비스 상태가 바뀌면 onServiceStateChanged() 함수가 호출되면서 매개변수인 ServiceState 객체의 state값으로 전달됩니다. 이때 서비스 상태는 ServiceState 뒤에 다음과 같은 상수로 전달됩니다.

- STATE_IN_SERVICE: 서비스 가능 상태
- STATE_EMERGENCY_ONLY: 긴급 통화만 가능한 상태
- STATE_OUT_OF_SERVICE: 서비스 불가 상태
- STATE_POWER_OFF: 비행 모드 등 전화 기능을 꺼 놓은 상태

– 전화가 걸려오는 상태 감지

이번에는 전화가 걸려오는 상태를 감지하는 onCallStateChanged() 함수를 알아보겠습니다.

• 전화가 걸려오는 상태 감지

```
override fun onCallStateChanged(state: Int, phoneNumber: String?) {
    when (state) {
        TelephonyManager.CALL_STATE_IDLE -> Log.d("kkang", "IDLE..")
        TelephonyManager.CALL_STATE_RINGING ->
            Log.d("kkang", "RINGING..$phoneNumber")
        TelephonyManager.CALL_STATE_OFFHOOK -> Log.d("kkang",
        "OFFHOOK..$phoneNumber")
    }
}
```

onCallStateChanged() 함수의 첫 번째 매개변수로 전화가 걸려오는 상태를 구별할 수 있으며 TelephonyManager 뒤에 다음과 같은 상수가 전달됩니다.

- CALL_STATE_IDLE: 통화 대기 상태
- CALL_STATE_RINGING: 벨이 울리는 상태
- CALL_STATE_OFFHOOK: 통화 중인 상태

두 번째 매개변수는 걸려오는 전화번호인데 앱에서 전화번호를 받으려면 다음과 같은 퍼미션이 필요합니다.

• 전화번호를 받는 퍼미션

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

– 네트워크 제공 국가, 사업자, 전화번호 얻기

TelephonyManager는 네트워크 제공 국가, 사업자, 전화번호 등을 반환하는 함수 제공.

● 네트워크 접속 정보

ConnectivityManager로 네트워크 접속 정보 파악.

– getActiveNetwork() 함수 이용법

함수의 객체 Network를 얻어쓰면 되는데 23하위 버전은 Network인포 함수 이용.

activityNetwork 정보가 널이면 현재 스마트폰은 네트워크에 접속할 수 없다는 이야기입니다. activityNetwork로 얻은 Network 객체를 다시 getNetworkCapabilities() 함수의 매개변수로 지정하면 현재 접속된 네트워크 정보를 얻을 수 있습니다. 이 함수의 반환값은 NetworkCapabilities 객체이며 hasTransport() 함수를 이용해 와이파이에 접속된 상태인지 아니면 이동 통신망에 접속된 상태인지를 알 수 있습니다.

- requestNetwork() 함수 이용법

위와 비슷한 것임.

2. HTTP 통신하기

앱에서 네트워크 통신을 하려면 우선 매니페스트 파일에 퍼미션 선언해야 한다.

앱은 통신을 할 때 기본적으로 HTTPS 프로토콜을 사용한다. 만약 일반 HTTPS 프로토콜로 통신하려면 특정 도메인만 허용하도록 선언해야 한다. Res/xml 폴더에 임의의 이름으로 XML 파일을 만들고

• HTTP 통신 허용

```
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">xxx.xxx.xxx.xxx</domain>
    </domain-config>
</network-security-config>
```

HTTP 프로토콜로 접속을 허용할 IP나 도메인

이렇게 작성한다.

- Volley 라이브러리

구글에서 제공하는 Volley 와 스퀘어에서 제공하는 Retrofit은 네트워크 프로그래밍을 돕는 라이브러리다.

RequestQueue 객체는 서버에 요청을 보내는 역할을 하며 이때 서버 URL과 결과를 가져 오는 콜백 등 다양한 정보는 XXXRequest 객체에 담아서 전송한다.

– StringRequest - 문자열 데이터 요청하기

StringRequest의 생성자에는 HTTP 메서드, 서버 URL 그리고 서버로부터 결과를 받을 때 호출할 콜백과 서버 연동에 실패할 때 호출할 콜백을 지정합니다.

• 문자열 요청 정의

```
val stringRequest = StringRequest(
    Request.Method.GET,
    url,
    Response.Listener<String> {
        Log.d("kkang", "server data : $it")
    },
    Response.ErrorListener { error ->
        Log.d("kkang", "error.....$error")
    })
```

– ImageRequest - 이미지 데이터 요청하기

ImageRequest의 생성자에 지정해야 하는 매개변수는 다음과 같습니다.

- url: 서버 URL
- listener: 결과를 가져오는 콜백
- maxWidth, maxHeight: 지정한 값으로 이미지 크기 조절해서 전달. 만약 0으로 설정하면 크기 조절 없이 서버가 전달하는 이미지를 그대로 받음
- scaleType: 영역에 맞게 이미지의 크기를 확대 또는 축소하는 스케일 타입
- decodeConfig: 이미지 형식 지정
- errorListener: 오류 콜백

– 화면 출력용 이미지 데이터 요청하기 - NetworkImageView

– JSON 데이터 요청하기 - JsonObjectRequest
– JSON 배열 요청하기 - JsonArrayRequest

- Retrofit 라이브러리

1. 통신용 함수를 선언한 인터페이스를 작성합니다.
2. Retrofit에 인터페이스를 전달합니다.
3. Retrofit이 통신용 서비스 객체를 반환합니다.
4. 서비스의 통신용 함수를 호출한 후 Call 객체를 반환합니다.
5. Call 객체의 enqueue() 함수를 호출하여 네티워크 통신을 수행합니다.

– 라이브러리 선언
빌드 그래들 파일에 임플리멘태이션.

– 모델 클래스 선언
모델 클래스란 서버와 주고받는 데이터를 표현하는 클래스이다.
VO(value-object)클래스라고도 하며 JSON, XML 데이터를 파싱해 모델 클래스 객체에 담는 것을 자동화해 준다.

– 서비스 인터페이스 정의
Retrofit을 이용할 때 가장 중요한 것은 네트워크 통신이 필요한 순간에 호출할 함수를 포함하는 서비스 인터페이스를 작성하는 것이다.

– Retrofit 객체 생성
가장 먼저 Retrofit 객체를 생성하는 코드를 실행해야 한다.

– 인터페이스 타입의 서비스 객체 얻기

Retrofit 객체를 생성한 다음에는 이 객체로 서비스 인터페이스를 구현한 클래스의 객체를 얻습니다.

• 서비스 객체 얻기

```
var networkService: INetworkService = retrofit.create(INetworkService::class.java)
```

Retrofit의 create() 함수에 앞에서 만든 서비스 인터페이스 타입을 전달합니다. 그러면 이 인터페이스를 구현한 클래스의 객체를 반환해 줍니다. 실제 네트워크가 필요할 때 이 객체의 함수를 호출하면 됩니다.

- 네트워크 통신 시도

● Retrofit 애너테이션
ㅈㄴ 무슨 소린지 모름 ㅎ

3. 이미지 처리하기 - Glide 라이브러리
서버에서 이미지를 내려받을 때 Glide를 이용하면 네트워크 부분을 Volley나 Retrofit 보다 더 쉽고 빠르게 개발할 수 있다. 모든 종류의 이미지를 가능한 한 빠르게 가져와서 이용할 수 있게 한다.

먼저 빌드 그래들에 등록한다.

● 이미지를 가져와 출력하기
앱에서 이용하는 이미지는 리소소, 파일, 서버 이미지이다. 이 이미지를 Glide로 쉽게 가져와 이미지 뷰에 출력할 수 있다.

● 크기 조절
Oom 문제 때문에 크기를 줄여야 했지만 글라이드는 알아서 불러온다.

● 로딩, 오류 이미지 출력
이러한 기능을 구현하려면 placeholer(), error() 함수를 이용해

해당 상황에 표시할 이미지를 지정한다.

- 이미지 데이터 이용하기

맨위처럼 하셈..

4. 뉴스 앱 만들기

- 정리

1. TelephonyManager를 이용하면 앱에서 스마트폰의 다양한 정보를 알 수 있다.
2. ConnectivityManager를 이용하면 스마트폰의 네트워크 접속 정보를 알 수 있다.
3. Volley와 Retrofit은 모두 HTTP 통신을 구현할 때 사용하는 라이브러리이다.
4. 이미지를 다룰 때 Glide 라이브러리를 이용하면 좀 더 편리하게 작성할 수 있다.