08

# 08-2. Volley API

네트워크 프로그래밍

# 18-2 HTTP 통신하기

• 인터넷 퍼미션 선언

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

- 안드로이드 앱은 네트워크 통신을 할 때 기본으로 HTTPS 보안 프로토콜을 사용
- 만약 일반 HTTPS 프로토콜로 통신하려면 특정 도메인만 허용하도록 선언

• 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

# 18-2 HTTP 통신하기

- XML 파일을 매니페스트의 <application> 태그에 networkSecurityConfig 속성으로 지정

• 매니페스트에 HTTP 허용 XML 등록

```
<application
    (... 생략 ...)
    android:networkSecurityConfig="@xml/network_security_config">
```

- 또는 매니페스트에서 usesCleartextTraffic 속성을 true로 설정하면 앱 전체에서 모든 도메인의 서버와 HTTP 통신을 할 수 있습니다.

• 모든 HTTP 통신 허용

```
<application
    (... 생략 ...)
    android:usesCleartextTraffic="true">
```

# 18-2 HTTP 통신하기

## Volley 라이브러리

- 구글에서 제공하는 Volley와 스퀘어에서 제공하는 Retrofit
- Volley는 2013년 구글 IO 행사에서 공개된 라이브러리

• Volley 라이브러리 등록

```
implementation("com.android.volley:volley:1.2.1")
```

- Volley에서 핵심 클래스
  - RequestQueue: 서버 요청자
  - XXXRequest: XXX 타입의 결과를 받는 요청 정보

# 18-2 HTTP 통신하기

- 문자열 데이터 요청하기 — StringRequest
  - StringRequest(int method, String url, Response.Listener<String> listener, Response.ErrorListener errorListener)

• 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")
    })
```

• 서버에 요청하기

```
val queue = Volley.newRequestQueue(this)
queue.add(stringRequest)
```

# 18-2 HTTP 통신하기

- POST 방식에서는 StringRequest를 상속받은 클래스를 이용

• POST 방식으로 데이터 전송

```
val stringRequest = object : StringRequest(
    Request.Method.POST,
    url,
    Response.Listener<String> {
        Log.d("kkang", "server data : $it")
    },
    Response.ErrorListener { error ->
        Log.d("kkang", "error.....$error")
    }){
    override fun getParams(): MutableMap<String, String> {
        return mutableMapOf<String, String>("one" to "hello", "two" to "world")
    }
}
```

# 18-2 HTTP 통신하기

- 이미지 데이터 요청하기 — ImageRequest
  - public ImageRequest(String url, Response.Listener<Bitmap> listener, int maxWidth, int maxHeight, ScaleType scaleType, Config decodeConfig, @Nullable Response.ErrorListener errorListener)

  - 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

# 18-2 HTTP 통신하기

• 이미지 요청 정의

```
val imageRequest = ImageRequest(
    url,
    Response.Listener { response -> binding.imageView.setImageBitmap(response) },
    0,
    0,
    ImageView.ScaleType.CENTER_CROP,
    null,
    Response.ErrorListener { error ->
        Log.d("kkang", "error.............$error")
    })

val queue = Volley.newRequestQueue(this)
queue.add(imageRequest)
```

# 18-2 HTTP 통신하기

- 화면 출력용 이미지 데이터 요청하기 — NetworkImageView
  - NetworkImageView는 Volley에서 제공하는 이미지 출력용 뷰
  - setImageUrl(String url, ImageLoader imageLoader)

• 화면 출력용 이미지 데이터 요청하기

```
<com.android.volley.toolbox.NetworkImageView
    android:id="@+id/networkImageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

• setImageUrl() 함수로 요청하기

```
val queue = Volley.newRequestQueue(this)
val imgMap = HashMap<String, Bitmap>()
imageLoader = ImageLoader(queue, object : ImageLoader.ImageCache {
    override fun getBitmap(url: String): Bitmap? {
        return imgMap[url]
    }
    override fun putBitmap(url: String, bitmap: Bitmap) {
        imgMap[url] = bitmap
    }
})
binding.networkImageView.setImageUrl(url, imageLoader)
```

# 18-2 HTTP 통신하기

- JSON 데이터 요청하기 — JsonObjectRequest
  - JsonObjectRequest(int method, String url, JSONObject jsonRequest, Response.Listener<JSONObject> listener, Response.ErrorListener errorListener)

• JSON 데이터 요청하기

```
val jsonRequest =
    JsonObjectRequest(
        Request.Method.GET,
        url,
        null,
        Response.Listener<JSONObject> {  response ->
            val title = response.getString("title")
            val date = response.getString("date")
            Log.d("kkang","$title, $date")
        },
        Response.ErrorListener { error -> Log.d("kkang","error....$error")
    })
val queue = Volley.newRequestQueue(this)
queue.add(jsonRequest)
```

# 18-2 HTTP 통신하기

- JSON 배열 요청하기 — JsonArrayRequest
  - JsonArrayRequest(String url, JSONArray jsonRequest, Response.Listener<JSONArray> listener, Response.ErrorListener errorListener)

• JSON 배열 요청하기

```
val jsonArrayRequest = JsonArrayRequest(
    Request.Method.GET,
    url,
    null,
    Response.Listener<JSONArray> { response ->
        for (i in 0 until response.length()) {
            val jsonObject = response[i] as JSONObject
            val title = jsonObject.getString("title")
            val date = jsonObject.getString("date")
            Log.d("kkang","$title, $date")
        }
    },
    Response.ErrorListener { error -> Log.d("kkang", "error....$error") }
)
val queue = Volley.newRequestQueue(this)
queue.add(jsonArrayRequest)
```

# 감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare