25장. HTTP 통신

25.1 Java API를 이용한 HTTP 통신

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

- 웹 서버와 연결하려면 URL 클래스와 HttpURLConnection 클래스를 이용

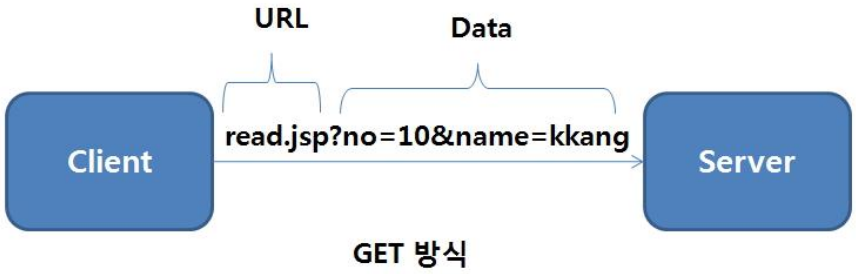
```
URL text = new URL(url);
HttpURLConnection http = (HttpURLConnection) text.openConnection();
```

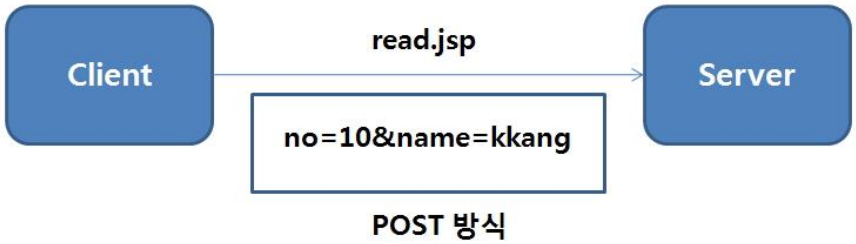
HttpURLConnection 클래스
- setConnectTimeput(): 연결 타임아웃 시간 지정
- setReadTimeout(): 읽기 타임아웃 시간 지정
- setUseCaches(): 캐시 사용 여부 지정
- setDoInput(): 데이터를 읽을 것인지 지정
- setDoOutput(): 데이터를 요청 몸체에 쓸 것인지 지정
- setRequestProperty(): 요청 헤더 지정
- setRequestMethod(): HTTP 요청 Method 지정

```
http.setRequestProperty("Content-type", "application/x-www-form-                    urlencoded;charset=UTF-8");
http.setConnectTimeout(10 * 1000);
http.setReadTimeout(10 * 1000);
http.setRequestMethod("POST");
http.setDoInput(true);
http.setDoOutput(true);
```

25.1 Java API를 이용한 HTTP 통신

- 서버에 데이터 전송





- GET 방식으로 전송하면 HTTP 요청 헤더의 URL 뒤에 붙어서 서버에 전송
- POST 방식으로 데이터를 전송하려면 setRequestMethod( ) 함수를 이용하여 POST 방식을 사용하겠다고 선언해야 하며, 정식의 IO 객체를 이용해 전송

```
PrintWriter writer = new PrintWriter(new OutputStreamWriter( http.getOutputStream(), "UTF-8"));
writer.write(postData);
writer.flush();
```

25.1 Java API를 이용한 HTTP 통신

- 데이터를 수신

```
BufferedReader in = new BufferedReader(new                    InputStreamReader( http.getInputStream(), "UTF-8"));
StringBuffer sb = new StringBuffer();
String inputLine;
while ((inputLine = in.readLine()) != null) {
   sb.append(inputLine);
}
response = sb.toString();
```

- 이미지 수신

```
Bitmap bitmap = BitmapFactory.decodeStream(http.getInputStream());
```

Step by Step 실습 25-1 – HTTP 통신

HttpURLConnection을 이용한 서버 연동을 테스트

서버에서 문자열과 이미지 데이터를 획득하여 화면에 출력하는 테스트이며 문자열은 JSON문자열로 테스트
NodeJS 기반의 간단한 서버를 구축하여 테스트
서버 사이드 프로그램은 실습을 위해 제공되는 파일을 복사하여 테스트 하겠습니다.

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

25.2 Volley API를 이용한 HTTP 통신

- Volley는 2013년 구글 IO 행사에서 공개된 API

```
implementation 'com.android.volley:volley:1.0.0'
```

- RequestQueue: 서버 요청자. 다른 Request 클래스들의 정보대로 서버에 요청을 보내는 역할
- StringRequest: 문자열을 결과로 받는 요청 정보
- ImageRequest: 이미지를 결과로 받는 요청 정보
- JsonObjectRequest: JSONObject를 결과로 받는 요청 정보
- JsonArrayRequest: JSONArray를 결과로 받는 요청 정보

```
StringRequest stringRequest = new StringRequest(...);

RequestQueue queue = Volley.newRequestQueue(this);
queue.add(stringRequest);
```

25.2 Volley API를 이용한 HTTP 통신

- StringRequest는 서버로부터 문자열 데이터를 얻을 목적

```
StringRequest stringRequest = new StringRequest(Request.Method.GET, url, new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
        //결과 처리
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {

    }
});
```

- 클라이언트 데이터를 서버에 전송

```
StringRequest stringRequest = new StringRequest(...){
    protected Map<String, String> getParams() throws                    com.android.volley.AuthFailureError {
        Map<String, String> params = new HashMap<String, String>();
        params.put("a1", "kkang");
        return params;
    }
};
```

25.2 Volley API를 이용한 HTTP 통신

- JSON 데이터를 목적으로 JsonObjectRequest와 JsonArrayRequest을 제공

```
JsonObjectRequest jsonRequest = new JsonObjectRequest(Request.Method.GET, url,                    null, new
Response.Listener<JSONObject>() {
    @Override
    public void onResponse(JSONObject response) {
        //서버 응답 후 사후처리
        //JSONObject에서 데이터 획득
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
    }
});
```

- 이미지 연동

```
ImageRequest imageRequest=new ImageRequest (url, new Response.Listener<Bitmap>() {
    @Override
    public void onResponse(Bitmap response) {
        //결과 처리
    }
    //maxWidth Maximum width 등의 사이즈 정보 설정 가능. 0은 설정 안하겠다는의미},0,0,
ImageView.ScaleType.CENTER_CROP,null, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {

    }
});
```

25.2 Volley API를 이용한 HTTP 통신

- 서버의 이미지 화면에 출력

```
<com.android.volley.toolbox.NetworkImageView    //.../>
```

```
ImageLoader imageLoader = new ImageLoader(queue, new ImageLoader.ImageCache() {
    public void putBitmap(String url, Bitmap bitmap) {
    }

    public Bitmap getBitmap(String url) {
        return null;
    }
});
//이미지 요청
imageView = (NetworkImageView) findViewById(R.id.lab2_image);
imageView.setImageUrl(url, imageLoader);
```

Step by Step 25-2 – Volley API

Volley API을 이용한 서버 연동을 테스트

실습 25-1에서 HttpURLConnection으로 서버 요청했던 업무를 그대로 Volley API로 구현하는 방식으로 진행

1. Activity 생성
2. 파일 복사
3. gradle 설정
4. activity_lab25_2.xml 작성
5. Lab25_2Activity 작성
6. Lab25_2Activity.java 실행

25.3 Glide를 활용한 HTTP 통신

- Glide 라이브러리는 이미지 핸들링을 제공하기 위한 라이브러리

```
implementation 'com.github.bumptech.glide:glide:3.7.0'
```

- 리소스 이미지

```
Glide.with(this)
        .load(R.drawable.photos)
        .into(resourceView);
```

- 파일 이미지

```
File file = new File(path);
Uri imageUri = Uri.fromFile(file);
Glide.with(this)
        .load(imageUri)
        .into(imageView);
```

- 서버 이미지

```
Glide.with(this)
                .load(url)
                .into(imageView);
```

25.3 Glide를 활용한 HTTP 통신

- GIF 이미지

```
Glide.with(this)
        .load(R.raw.loading)
        .asGif()
        .diskCacheStrategy(DiskCacheStrategy.SOURCE)
        .into(imageView);
```

- override( )

```
Glide.with(this)
        .load(url)
        .override(400, 400)
        .placeholder(R.raw.loading)
        .error(R.drawable.error)
        .into(imageView);
```

- Target 클래스

```
Glide.with(this)
        .load(imageUri)
        .asBitmap()
        .into(new BitmapImageViewTarget(imageView){
            @Override
            protected void setResource(Bitmap resource) {
                super.setResource(resource);
            }
        });
```

25.3 Glide를 활용한 HTTP 통신

- SimpleTarget

```
Glide.with(this)
        .load(imageUri)
        .asBitmap()
        .into(new SimpleTarget<Bitmap>(250, 250) {
            @Override
            public void onResourceReady(Bitmap resource, GlideAnimation<? super                    Bitmap>
glideAnimation) {
                //...
            }
        });
```

Step by Step 25-3 – Glide API

Glide API을 이용한 이미지 획득을 테스트

간단하게 GIF 이미지와 서버 이미지를 획득하여 화면에 출력하는 테스트

1. Activity 생성
2. 파일 복사
3. gradle 설정
4. Lab25_3Activity 작성
5. Lab25_2Activity.java 실행