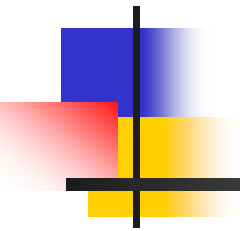


URLConnection 실습

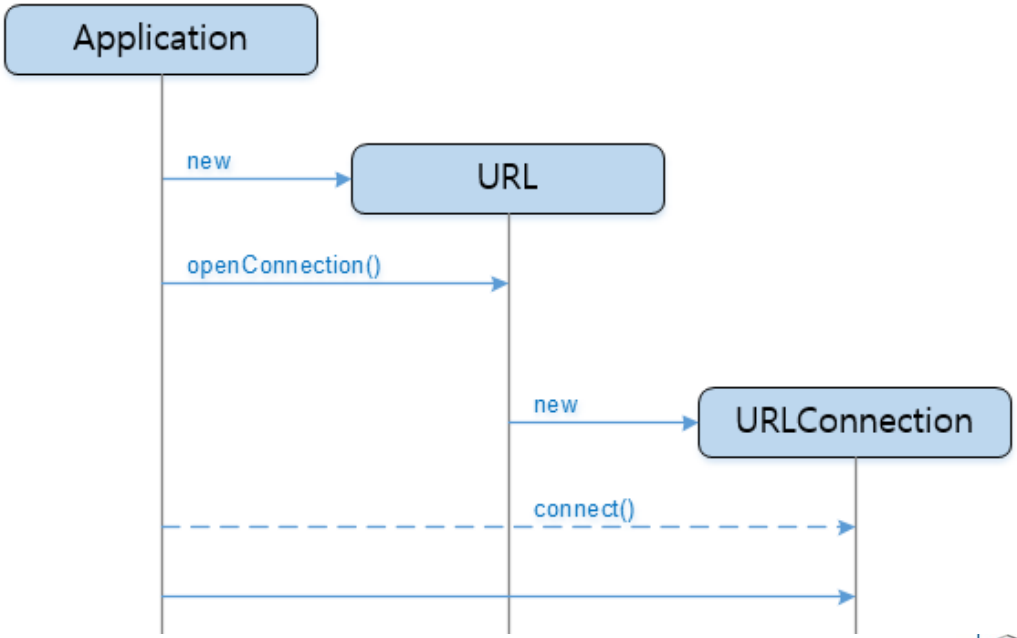
배 희호 교수
경복대학교
소프트웨어융합과

URLConnection 클래스

- java.net.URLConnection 클래스
- 추상 클래스
  - URLConnection 클래스는 자체적으로 생성자를 통해서 객체(Instance)를 생성할 수 없음
- Application과 URL 간의 통신 Link를 위한 모든 클래스들에 대한 상위 클래스
- 객체는 URL을 이용하여 참조된 Resource에 대해 읽고 쓰는 작업을 할 수 있도록 해줌

URLConncection 클래스

- Network으로 연결된 Remote Computer의 Resource를 알아내는 기능을 제공하는 클래스
- URL 주소의 내용을 읽어 오거나, 반대로 URL 주소가 가리키는 Web Application에게 GET 방식이나 POST 방식으로 추가적인 정보를 전달할 수 있고, 원격지 File을 읽어오는 다양한 메소드를 제공



URLConnection 클래스

- URL이 가리키는 Resource에 대한 활성화된 연결(Connection)을 나타내는 추상 클래스
- 2가지 목적
  - Server(특히 HTTP Server)와 통신하는데 있어 URL 클래스보다 더 좋은 제어 방법을 제공
    - URLConnection은 Server가 보낸 Header를 확인하고 그에 따른 적절한 Response를 보낼 수 있을 뿐만 아니라, Client Request에 사용된 Header Field를 설정할 수 있음
    - URLConnection은 다양한 HTTP Request 메소드를 사용해서 Web Server로 Data를 보낼 수도 있음
  - URLConnection 클래스는 URIStreamHandler 클래스를 비롯한 JAVA의 Protocol Handler Mechanism의 한 부분으로서 동작

URLConnection 클래스

- URL 클래스는 Remote Server Resource의 결과만을 가져오지만, URLConnection 클래스는 Remote Server Resource의 결과와 Remote Server의 Header 정보를 가져올 수 있음
- URLConnection 클래스는 Remote Resource에 접근하는 데 필요한 정보를 가지고 있음
  - 필요한 정보란 Remote Server의 Header 정보, 해당 Resource의 길이와 Type 정보, 언어 등의 정보
- URLConnection은 POST, PUT, 그리고 그 밖의 다른 HTTP Request 메소드를 사용하여 Web Server에 Data를 보낼 수 있음

URLConnection 클래스

- 사용자 인증이나 보안이 설정되어 있지 않은 Web Server에 접속하여 File 등을 Download 하는데 많이 사용됨
- URLConnection은 Resource에 연결하기 전에 구성 되어야 함
- URLConnection Instance는 재사용 될 수 없음
  - 각 Resource에 대한 Connection마다 다른 Instance를 사용해야 함
- 객체는 URL을 이용하여 참조된 Resource에 대해 읽고 쓰는 작업을 할 수 있도록 해줌
  - URL 주소의 내용을 읽어 오거나, 반대로 URL 주소가 가리키는 Web Application에게 GET 방식이나 POST 방식으로 추가적인 정보를 전달할 수 있고, Remote File을 읽어오는 다양한 메소드를 제공

URLConnection 클래스

- 생성자
  - 추상 클래스이기 때문에 단독적으로 객체를 생성할 수 없음
  - URL 클래스의 객체를 생성해서 URL 클래스의 openConnection() 메소드를 이용해서 객체를 생성함
  - URLConnection 객체가 생성 되었다면 connect() 메소드를 호출해야 객체가 완성됨

```
URL url = new URL("http://java.sun.com");
URLConnection connection = url.openConnection();
connection.connect();
```

URLConnection 클래스

- 메소드

메소드	설명
String getContentType()	mime 타입 반환
getContentEncoding()	인코딩 반환
getContentEncoding()	헤더의 content-encoding에 대한 정보 반환
getContentLength()	헤더 필드의 content-length에 대한 값 반환
getHeaderFields()	헤더 정보 반환
InputStream getInputStream()	문서를 읽기 위한 InputStream 객체 반환
OutputStream getOutputStream()	URLConnetion 객체로부터 OutputStream 반환
URL getURL()	URL 필드 반환

URLConnection 클래스

- 메소드

메소드	설명
void addRequestProperty (String key, String value)	<키-값> 쌍으로 지정된 일반 요청 속성을 추가
void connect()	URL에서 참조하는 리소스에 대한 통신 링크를 염(이러한 연결이 아직 설정되지 않은 경우)
boolean getAllowUserInteraction()	개체에 대한 allowUserInteraction 필드의 값을 반환
int getConnectionTimeout()	연결 시간 초과에 대한 설정을 반환
Object getContent()	It retrieves the contents of the URL connection.
Object getContent(Class[] classes)	It retrieves the contents of the URL connection.

URLConnection 클래스

- 메소드

메소드	설명
String getContentEncoding()	헤더 필드의 content-encoding에 대한 value를 반환. 인코딩 타입을 String으로 리턴
int getContentLength()	헤더 필드의 content-length에 대한 value를 반환 content 길이가 -1이면 정상적으로 값이 넘어오지 않았음을 의미한다
String getHeaderField (String name)	헤더 필드의 이름(name)에 대한 value를 반환 필드의 이름은 content-encoding, content-type등이 올 수 있다
getHeaderField(int field)	Head Field 값 반환 (http Head 값)
Map<String, List<String>> getHeaderFields()	헤더 필드의 구조를 Map으로 반환
InputStream getInputStream()	URLConnection 객체로부터 읽기 위한 InputStream 객체를 반환

URLConnection 클래스

- 메소드

메소드	설명
OutputStream getOutputStream()	URLConnection 객체로부터 출력(쓰기)하기 위한 OutputStream객체를 반환.
URL getURL()	URLConnection의 멤버 변수로 설정된 url 필드의 값을 반환
addRequestProperty (String a, String b)	키(a) 와 값(b)을 가지고 요청할 수 있는 Properity 값을 미리 설정해 놓음 특정 키값을 가지고 읽을 수 있도록 함
connect()	연결 된 곳에 접속 할 때 (connect() 호출해야 실제 통신 가능함)
getAllowUserInteraction()	연결 된 곳에 사용자가 서버와 통신 할 수 있는 환경 확인(boolean)in/output이 해당 서버 , 연결 포트로 가능한지 확인
getContent()	content 값을 리턴 (inputStream 값을 리턴)

URLConnection 클래스

- 메소드

메소드	설명
getContent(Class[])	위 내용을 class[] 배열 값을 입력
getContentType()	content 가 http로 되어 있는지 타입 (ex: http-type)
getDate()	content의 날짜 (new Date(~~) 으로 변환해 줘야 함 / Long 리턴)
getDefaultAllowUserInteraction()	기본적으로 User와 통신 가능한 상태인지 (boolean)
getDefaultUserCaches()	cache를 사용할 것 인지 (boolean)
getDoInput()	Server에서 온 데이터를 입력 받을 수 있는 상태인지 (본인 상태-default : true)
getDoOutput()	Server에서 온 데이터를 출력 할 수 있는 상태인지 (Client 상태 -default : false)

URLConnection 클래스

- 메소드

메소드	설명
getExpiration()	유효 기간
getFileNameMap()	File Name Map
getHeaderFiled(String)	
getLastModified()	마지막 수정 시간
setDoInput(boolean)	Server 통신에서 입력 가능한 상태로 만들기
setDoOutput(boolean)	Server 통신에서 출력 가능한 상태로 만들기
	Server와 통신을 하고자 할 때는 반드시 setDoInput(), setDoOutput() 두 method를 true로 해 놓아야 함

doInput 필드가 **true**로 설정되면 **URLConnection** 객체로 표현되는 **URL** 연결이 입력을 위해 사용됨을 의미
doOutput 필드가 **true**로 설정되면 출력을 위해 사용됨을 의미

URLConnection 클래스

- 메소드

메소드	설명
URLConnection openConnetion()	URLConnection 객체 생성
void setRequestMethod (String method)	요청 방식을 지정하는 함수 GET나 POST 문자열을 인자로 전달
void setRequestProperty (Stiring field, String newValue)	요청시 헤더에 들어가는 필드 값 지정
String getContentType()	해당 문서의 mime 타입 반환
String getContentEncoding()	인코딩 리턴
Strring getHeaderFields()	헤더 정보 리턴
InputStream getInputStream()	문서를 읽기 위한 InputStream 객체 리턴

URLConnection 클래스

- 메소드

메소드	설명
abstract void connect()	URL에 의해 참조되는 외부 리소스와 통신 연결 설정
Object getContent()	URL 연결에서 컨텐트를 가져옴
boolean getDoInput()	URLConnection 객체의 doInput 필드 값 반환
boolean getDoOutput()	URLConnection 객체의 doOutput 필드 값 반환
OutputStream getOutputStream()	설정된 연결로 데이터를 출력할 출력 스트림 반환
URL getURL()	URLConnection 객체의 URL 필드 값 반환
void setDoInput (boolean doInput)	URLConnection 객체의 doInput 필드 값 설정
void setDoOutput (boolean doOutput)	URLConnection 객체의 doOutput 필드 값 설정

URLConnection 클래스

- 메소드

메소드	설명
int getContentLength()	해당 문서의 길이를 바이트 수로 반환
long getDate()	해당 문서의 생성 날짜를 반환
long getExpiration()	해당 문서의 파기 날짜를 반환
long getLastModified()	해당 문서의 마지막 수정 날짜를 반환

URLConnection 클래스

- 사용 방법
  - URLConnection 클래스를 사용하는 Program은 아래와 같은 단계를 따라 작성해야 함

단계	수행 내용
1	URL 객체 생성
2	생성된 URL에 대한 URLConnection 객체를 얻기 위해 URL객체의 openConnection() 메소드 호출
3(생략)	반환된 URLConnection 객체 설정
4	Header Field 읽기
5	입력 Stream을 구하고, Data 읽기
6	출력 Stream을 구하고, Data 쓰기
7	연결 종료

URLConnection 클래스

- 사용 방법
  - 항상 이 모든 절차를 수행해야 하는 것은 아님
  - URLConnection의 기본 설정으로 해당 URL을 처리할 수 있다면 3번째 단계인 반환된 "URLConnection 객체 설정" 단계는 생략해도 됨

URLConnection 클래스

- Remote Resource를 가져오는 순서
  - 연결을 원하는 Computer의 정보를 가진 URL 객체 생성
  - openConnection() 메소드를 이용하여 URLConnection 객체 생성
  - URLConnection 객체를 이용하여 속성을 알아냄
  - URLConnection 클래스의 getInputStream() 메소드나 URL 클래스의 openStream() 메소드를 이용하여 Remote로부터 정보를 읽기 위한 InputStream 객체를 생성
    - URL 클래스의 openStream() 메소드를 사용하는 경우 이 메소드는 자동으로 URL 클래스의 openConnection() 메소드를 호출한 다음 openStream() 메소드를 수행
  - InputStream 객체를 이용하여 Remote의 정보를 읽음

URLConnection 클래스

- 주어진 Remote의 주소 URL에 Network 접속 후 Data를 보내거나 받을 수 있도록 하는 기능
- URL 객체 생성 방법
  - URL.openConnection() 이용

```
URL url = new URL("http://www.naver.com");
URLConnection conn = url.openConnection();
                                        // 원격지와 연결
```

  - URLConnection 생성자 이용

```
URL url = new URL("http://www.naver.com");
URLConnection conn = new URLConnection(url);
conn.connect();    // 원격지와 연결
```

    - 연결하기 전에 여러 가지 인자들과 Request과 관련된 속성들을 설정 가능

URLConnection 클래스

- Response Code
  - 200번 대 : 정상 응답
  - 300번 대 : Redirection 중
  - 400번 대 : Client 오류
    (URL Error나 권한이 없거나 하는 등의 Error)
  - 500번 대 : Server Error
    (Server의 Logic이 잘못 수행되는 경우)

URLConnection 클래스

- 특정 Protocol을 작성하려면 개발자가 Sub Classing해야 하는데 이 서브 클래스들은 Application의 Run time 시에 Loading될 수 있음
- Web Browser가 Compress와 같은 낯선 Protocol을 사용하는 Site에 접속했을 때, 처리할 수 없다는 Error Message를 보여주는 대신에 해당 Protocol에 대한 Handler를 다운로드 받아 Server와 통신할 수도 있음
- 이처럼 java.net 패키지에는 추상 클래스인 URLConnection만 있으며, 나머지 구상 서브 클래스들은 sun.net 패키지의 계층적인 구조에 숨겨져 있음
- URLConnection 클래스는 단일 생성자 뿐만 아니라 많은 메소드와 Field가 protected로 선언되어 있음

URLConnection 클래스

- URLConnection 클래스의 Instance나 서브 클래스를 통해서만 접근이 가능하다는 의미
  - 이 클래스를 사용할 때, URLConnection 객체를 직접 Instance화 하는 일은 거의 없음
- 대신에 Program 실행 중에 사용 중인 Protocol에 맞게 필요에 따라 만들어 사용하며, Compile시에 종류를 알 수 없는 클래스의 경우에는 java.lang.Class 클래스의 forName()과 newInstance() 메소드를 사용해서 인스턴스화
- JAVA의 클래스 Library에는 잘 설계된 URLConnection API가 없음
- URLConnection 클래스의 문제점 중 하나는 HTTP와 너무 밀접 하다는 점인데, 예를 들어 이 클래스는 각각의 File이 전송되기 전에 MIME Header나 이와 비슷한 것이 먼저 전송될 것이라고 가정. 하지만 FTP와 SMTP 같은 대부분의 오래된 Protocol은 MIME Header를 사용하지 않음

URLConnection 클래스

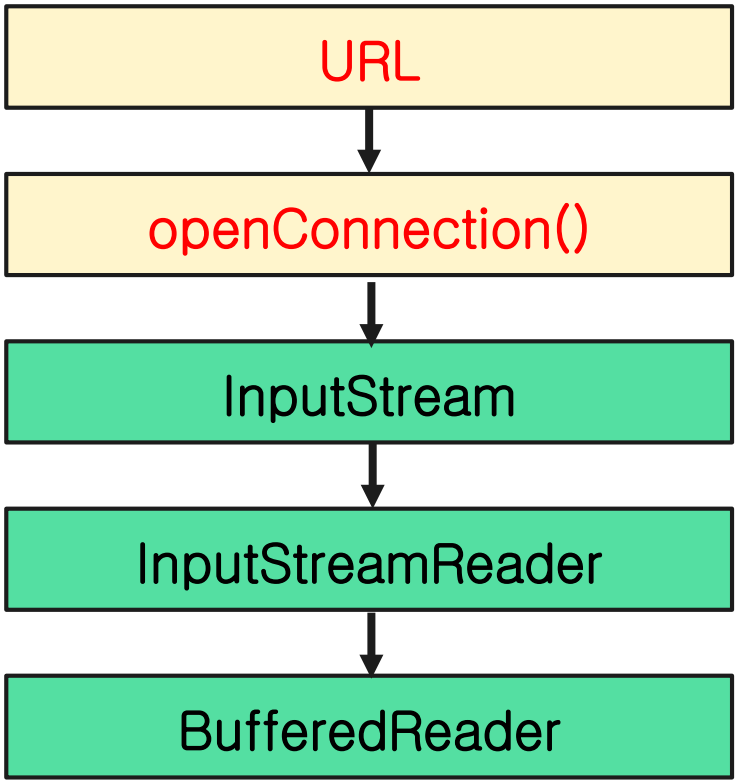
- Protocol Handler
  - Protocol Handler는 Protocol을 처리하는 부분을 다음과 같이 나누는 개념의 Mechanism
    - Data를 처리하는 부분
    - 사용자 Interface를 제공하는 부분
    - 모놀리식(monolithic) Web Browser가 수행하는 그 밖의 나머지 부분

URLConnection 클래스

- URL과 URLConnection의 차이점
  - URL 객체와 달리 HTTP POST 방식으로 Server에 Data를 전송할 수 있음
  - URLConnection은 URL 내용을 읽어오거나, URL 주소에 GET / POST로 Data를 전달 할 때 사용함
  - Web Page나 Servlet에 Data를 전달 수 있음
    - URL -> openConnection() -> URLConnection -> getInputStream -> InputStream (내용 읽음)
    - URL의 OpenStream()
      - URL의 입력 Stream만 개설 (차이점)
    - URLConnection
      - URL의 입력, 출력 Stream 개설

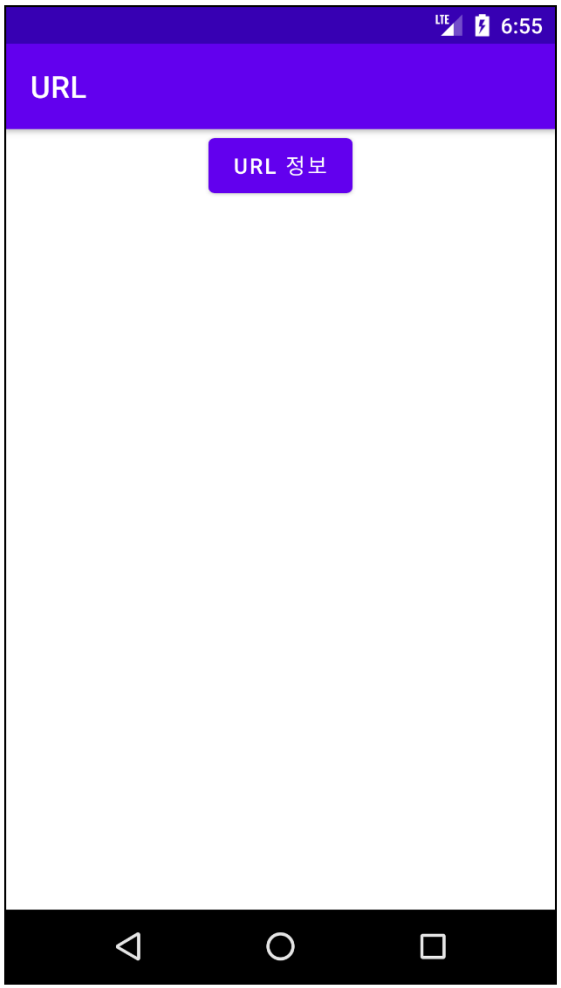
URLConnection 클래스

- Web Server에서 Text 문서 받아오는 순서



URLConnection 예제 1

- https://m.naver.com의 속성값을 출력해보자



URLConnection 예제 1

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/load"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="URL 정보" />
```

URLConnection 예제 1

- 사용자 인터페이스

```
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:id="@+id/result"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="" />
    </ScrollView>
</LinearLayout>
```

URLConnection 예제 1

- MainActivity.JAVA

```
public class MainActivity extends AppCompatActivity {
    String page = "https://m.naver.com";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        StrictMode.ThreadPolicy policy =
                new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        TextView result = findViewById(R.id.result);
        Button button = findViewById(R.id.load);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                try {
                    URL url = new URL(page);
                    URLConnection conn = url.openConnection();
```

URLConnection 예제 1

- MainActivity.JAVA

```
result.append("conn.toString() : " + conn.toString());
result.append("₩ngetAllowUserInteraction() : " +
                                        conn.getAllowUserInteraction());
result.append("₩ngetConnectTimeout() : " +
                                          conn.getConnectTimeout());
result.append("₩ngetContent() : " + conn.getContent());
result.append("₩ngetContentEncoding() : " +
                                       conn.getContentEncoding());
result.append("₩ngetContentLength() : " +
                                         conn.getContentLength());
result.append("₩ngetContentType() : " + conn.getContentType());
result.append("₩ngetDate() : " + conn.getDate());
result.append("₩ngetDefaultAllowUserInteraction() : " +
                                    conn.getDefaultAllowUserInteraction());
result.append("₩ngetUseChaches() : " + conn.getUseCaches());
result.append("₩ngetDefaultUserCaches() : " +
                                      conn.getDefaultUseCaches());
result.append("₩ngetDoInput() : " + conn.getDoInput());
```

String, StringBulder, StringBuffer

- String
  - String과 다른 클래스(StringBuffer, StringBuilder)의 기본적인 차이는 String은 immutable(불변), StringBuffer는 mutable(변함)에 있음
    - String 객체는 한번 생성되면 할당된 Memory 공간이 변하지 않음
  - String은 문자열을 대표하는 것으로 문자열을 조작하는 경우 유용하게 사용할 수 있음
  - 문자열, 숫자, char 등은 concat()할 때는 StringBuffer, StringBuilder를 사용할 수 있음
  - 단, 복잡한 경우 의미가 있고, 단순한 경우에는 굳이 StringBuffer, StringBuilder를 쓰지 않고 + 연산자를 활용해 직접 합치면 됨

String, StringBulder, StringBuffer

- String
  - + 연산자 또는 concat() 메소드를 통해 기존에 생성된 String 클래스 객체 문자열에 다른 문자열을 붙여도 기존 문자열에 새로운 문자열을 붙이는 것이 아니라, 새로운 String 객체를 만든 후, 새 String 객체에 연결된 문자열을 저장하고, 그 객체를 참조하도록 함
  - 즉, String 클래스 객체는 Heap 메모리 영역(Garbage Collection이 동작하는 영역)에 생성. 한번 생성된 객체의 내부 내용을 변화시킬 수 없음. 기존 객체가 제거되면 Java의 Garbage Collection이 회수
  - String 객체는 이러한 이유로 문자열 연산이 많은 경우, 그 성능이 좋지 않음
  - Immutable한 객체는 간단하게 사용하고, 동기화에 대해 신경 쓰지 않아도 되기 때문에(Thread-safe), 내부 Data를 자유롭게 공유 가능

String, StringBulder, StringBuffer

- StringBuffer와 StringBuilder
  - StringBuffer/StringBuilder는 String과 다르게 문자열 연산 등으로 기존 객체의 공간이 부족하게 되는 경우, 기존의 버퍼 크기를 늘리며 유연하게 동작함
  - StringBuffer와 StringBuilder 클래스가 제공하는 메소드는 서로 동일함
  - 두 클래스의 차이점은 동기화 여부임
  - StringBuffer는 각 메소드 별로 Synchronized Keyword가 존재하여, Multi Thread 환경에서도 동기화를 지원
    - 반면, StringBuilder는 동기화를 보장하지 않음
  - Multi Thread 환경이라면 값 동기화 보장을 위해 StringBuffer를 사용하고, Single Thread 환경이라면 StringBuilder를 사용하는 것이 좋음

String, StringBulder, StringBuffer

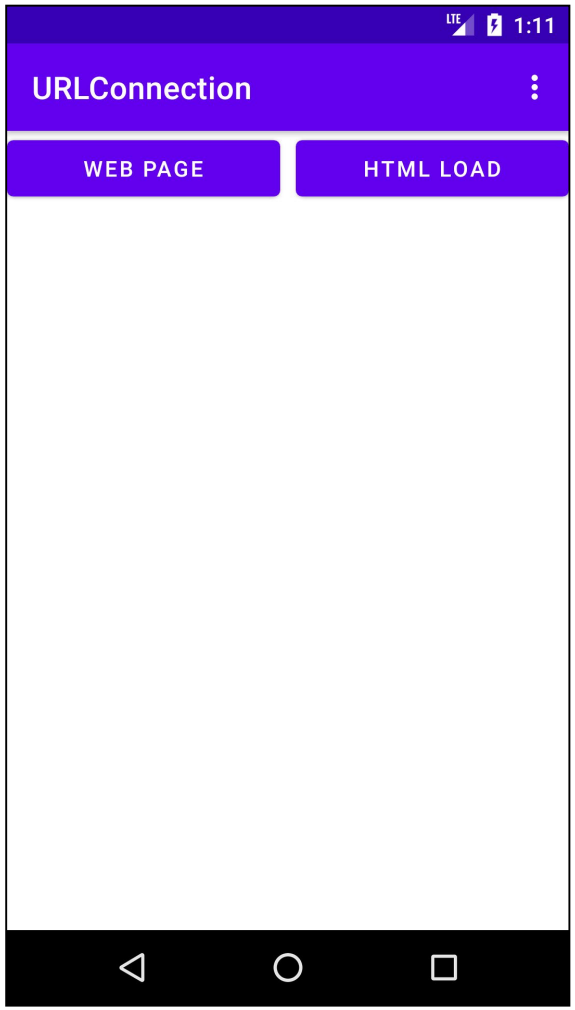
- StringBuffer와 StringBuilder
  - Single Thread 환경에서 StringBuffer를 사용한다고 문제가 되는 것은 아니지만, 동기화 관련 처리로 인해 StringBuilder에 비해 성능이 좋지 않음
  - String은 짧은 문자열을 더할 경우 사용
  - StringBuffer는 Thread에서 안전한 Program이 필요할 때나, 개발 중인 System의 부분이 Thread에 안전한지 모를 경우 사용하면 좋음
  - StringBuilder는 Thread에서 안전한지 여부가 전혀 관계없는 Program을 개발할 때 사용하면 좋음

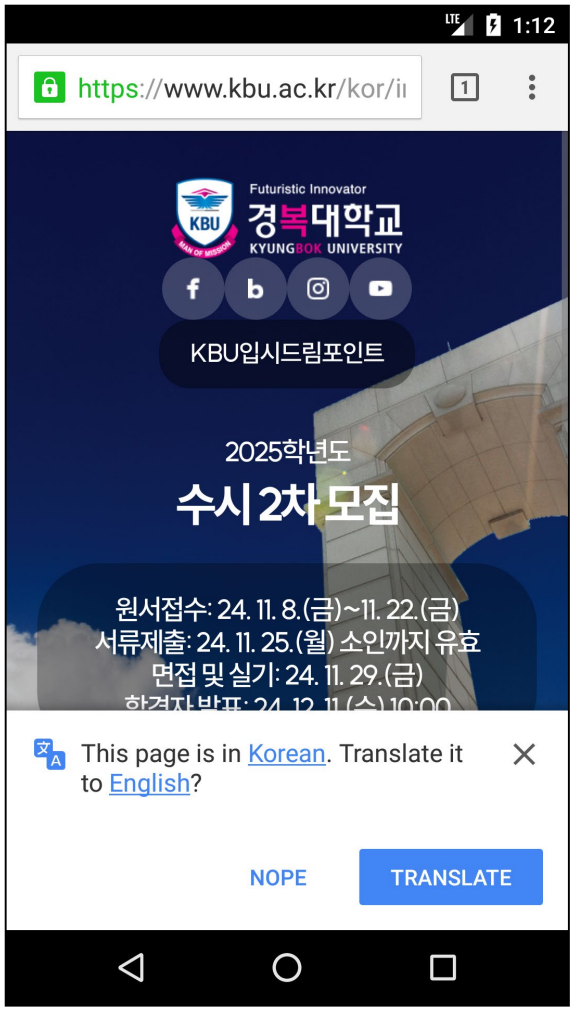
String, StringBulder, StringBuffer

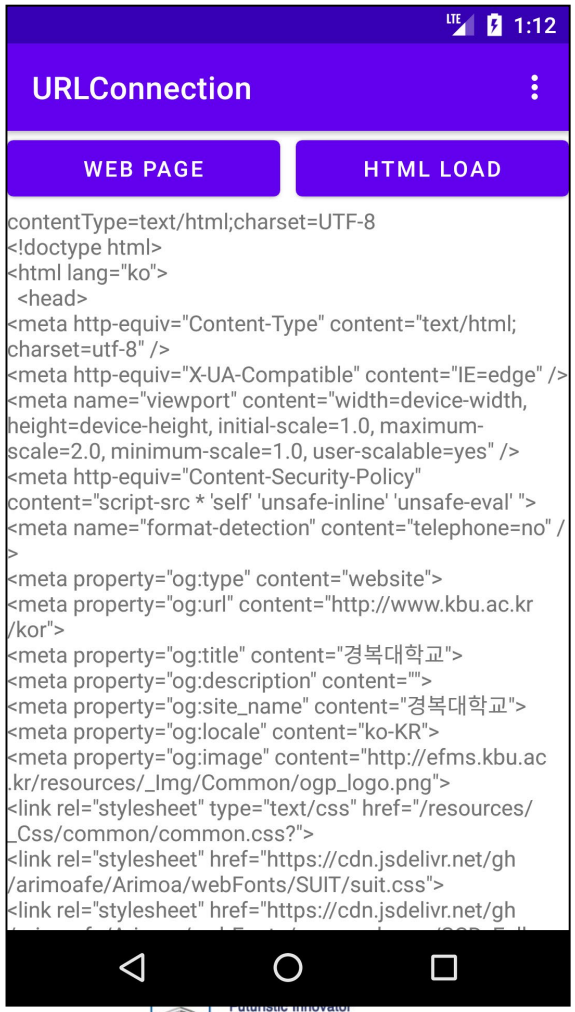
- StringBuffer와 StringBuilder
  - 사실 JDK 1.5버전 이전에서는 문자열 연산(+, concat)을 할 때에는 조합된 문자열을 새로운 메모리에 할당하여 참조함으로 인해서 성능상의 이슈가 있음
  - JDK1.5 버전 이후에는 컴파일 단계에서 String 객체를 사용하더라도 StringBuilder로 컴파일 되도록 변경
  - JDK 1.5 이후 버전에서는 String 클래스를 활용해도 StringBuilder와 성능상으로 차이가 없음
  - 하지만 반복 루프를 사용해서 문자열을 더할 때에는 객체를 계속 추가한다는 사실에는 변함이 없음
  - String 클래스를 쓰는 대신, Thread와 관련이 있으면 StringBuffer를, Thread 안전 여부와 상관이 없으면 StringBuilder를 사용하는 것을 권장
  - 단순히 성능만 놓고 본다면 연산이 많은 경우, StringBuilder > StringBuffer >>> String

URLConnection 예제 2

- http://www.kbu.ac.kr의 html 문서를 다운받아보자







URLConnection 예제 2

```
<?xml version="1.0" encoding="utf-8" ?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group
        android:checkableBehavior="single"
        android:enabled="true">
        <item
            android:id="@+id/item1"
            android:checked="true"
            android:title="StrictMode" />
        <item
            android:id="@+id/item2"
            android:title="AsyncTask" />
        <item
            android:id="@+id/item3"
            android:title="Thread" />
        <item
            android:id="@+id/item4"
            android:title="Runnable" />
    </group>
</menu>
```

URLConnection 예제 2

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <Button
            android:id="@+id/button1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_weight="1"
            android:text="Web Page" />
```

URLConnection 예제 2

- 사용자 인터페이스

```
    <Button
        android:id="@+id/button2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginStart="10dp"
        android:layout_weight="1"
        android:text="Html Load" />
</LinearLayout>

<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <WebView
        android:id="@+id/webWiew"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
```

URLConnection 예제 2

- 사용자 인터페이스

```
        <ScrollView
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/textViewt"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="" />
        </ScrollView>
    </FrameLayout>
</LinearLayout>
```

URLConnection 예제 2

- MainActivity.JAVA

```
public class MainActivity2 extends AppCompatActivity {
    final String page = "http://www.kbu.ac.kr";
    private TextView textView;
    private WebView webView;
    private int type = 1;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        textView = findViewById(R.id.textView);
        webView = findViewById(R.id.webView);
        webView.getSettings().setJavaScriptEnabled(true);
```

URLConnection 예제 2

- MainActivity.JAVA

```
Button button1 = findViewById(R.id.button1);
button1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        textView.setText("");
        webView.loadUrl(page);
    }
});
```

URLConnection 예제 2

```
Button button2 = findViewById(R.id.button2);
button2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        webView.clearView();
        textView.setText("");
        if (type == 1) {
            StrictMode.ThreadPolicy policy =
                    new StrictMode.ThreadPolicy.Builder().permitAll().build();
            StrictMode.setThreadPolicy(policy);
            DownHtml downLoad = new DownHtml(MainActivity2.this);
            textView.setText(downLoad.download(page));
        } else if (type == 2) {
            DownHtmlTask task = new DownHtmlTask(MainActivity2.this);
            try {
                textView.setText(task.execute(page).get());
            } catch (ExecutionException | InterruptedException e) {
                Toast.makeText(getBaseContext(), e.getMessage(),
                                                    Toast.LENGTH_SHORT).show();
            }
```

URLConnection 예제 2

- MainActivity.JAVA

```
}else if (type == 3) {
    DownHtmlThread thread =
                        new DownHtmlThread(getBaseContext(), page);
    thread.start();
    try {
        thread.join();
    } catch (InterruptedException e) {
        Toast.makeText(getBaseContext(), e.getMessage(),
                                    Toast.LENGTH_SHORT).show();
    }
    textView.setText(thread.getResult());
} else {
```

URLConnection 예제 2

- MainActivity.JAVA

```
                DownHtmlRunnable runnable =
                        new DownHtmlRunnable(getBaseContext(), page);
                Thread thread = new Thread(runnable);
                thread.start();
                try {
                    thread.join();
                } catch (InterruptedException e) {
                    Toast.makeText(getBaseContext(), e.getMessage(),
                                                        Toast.LENGTH_SHORT).show();
                }
                textView.setText(runnable.getResult());
            }
        }
    });
}
```

URLConnection 예제 2

- MainActivity.JAVA

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater menuInflater = getMenuInflater();
    menuInflater.inflate(R.menu.menu, menu);
    return true;
}
```

URLConnection 예제 2

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.item1:
                type = 1;
                break;
            case R.id.item2:
                type = 2;
                break;
            case R.id.item3:
                type = 3;
                break;
            case R.id.item4:
                type = 4;
        }
        webView.clearView();
        result.setText("");
        item.setChecked(true);
        return true;
    }
}
```

URLConnection 예제 2

- DownLoad.JAVA

```
public class DownLoad {
    private Context context;

    public DownLoad(Context context) {
        this.context = context;
    }

    public String download(String texturl) {
        StringBuilder buffer = new StringBuilder();
        try {
            URL url = new URL(texturl);
            URLConnection conn = url.openConnection();
            InputStream inputStream = conn.getInputStream();
            InputStreamReader streamReader = new InputStreamReader(inputStream);
            BufferedReader reader = new BufferedReader(streamReader);
            buffer.append("contentType=").append(conn.getContentType()).
                                                    append("₩n");
```

URLConnection 예제 2

- DownLoad.JAVA

```
            String line;
            while ((line = reader.readLine()) != null) {
                buffer.append(line).append("₩n");
            }
            inputStream.close();
            streamReader.close();
            reader.close();
        } catch (Exception e) {
            Toast.makeText(context, e.getMessage(), Toast.LENGTH_SHORT).show();
        }

        return buffer.toString();
    }
}
```

URLConnection 예제 2

- DownLoadTask.JAVA

```
public class DownLoadTask extends AsyncTask<String, String, String> {
    private Context context;

    public DownLoadTask(Context context) {
        this.context = context;
    }

    @Override
    protected String doInBackground(String... strings) {
        StringBuffer buffer = new StringBuffer();
        try {
            URL url = new URL(strings[0]);
            URLConnection conn = url.openConnection();
            InputStream inputStream = conn.getInputStream();
            InputStreamReader streamReader = new InputStreamReader(inputStream);
            BufferedReader reader = new BufferedReader(streamReader);
            buffer.append("contentType=" + conn.getContentType() + "₩n");
```

URLConnection 예제 2

- DownLoadTask.JAVA

```
            String line;
            while ((line = reader.readLine()) != null) {
                buffer.append(line + "₩n");
            }
            inputStream.close();
            streamReader.close();
            reader.close();
        } catch (Exception e) {
            publishProgress(e.getMessage());
        }
        return buffer.toString();
    }

    @Override
    protected void onProgressUpdate(String... values) {
        Toast.makeText(context, values[0], Toast.LENGTH_SHORT).show();
    }
}
```

URLConnection 예제 2

- DownLoadThread.JAVA

```
public class DownLoadThread extends Thread{
    private Context context;
    private String page;
    private Handler handler = new Handler();
    private StringBuffer buffer = new StringBuffer();

    public DownLoadThread(Context context, String page) {
        this.context = context;
        this.page = page;
    }

    @Override
    public void run() {
        try {
            URL url = new URL(page);
            URLConnection conn = url.openConnection();
            InputStream inputStream = conn.getInputStream();
            InputStreamReader streamReader = new InputStreamReader(inputStream);
            BufferedReader reader = new BufferedReader(streamReader);
```

URLConnection 예제 2

```
            buffer.append("contentType=" + conn.getContentType() + "₩n");
            String line;
            while ((line = reader.readLine()) != null) {
                buffer.append(line + "₩n");
            }
        } catch (Exception e) {
            handler.post(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(context, e.getMessage(),
                                                        Toast.LENGTH_SHORT).show();
                }
            });
        }
    }

    public String getResult() {
        return buffer.toString();
    }
}
```

URLConnection 예제 2

- DownLoadRunnable.JAVA

```
public class DownLoadRunnable implements Runnable{
    private Context context;
    private String page;
    private Handler handler = new Handler();
    private StringBuffer buffer = new StringBuffer();

    public DownLoadRunnable(Context context, String page) {
        this.context = context;
        this.page = page;
    }

    @Override
    public void run() {
        try {
            URL url = new URL(page);
            URLConnection conn = url.openConnection();
            InputStream inputStream = conn.getInputStream();
            InputStreamReader streamReader = new InputStreamReader(inputStream);
            BufferedReader reader = new BufferedReader(streamReader);
```

URLConnection 예제 2

- DownLoadRunnable.JAVA

```
            buffer.append("contentType=" + conn.getContentType() + "₩n");
            String line;
            while ((line = reader.readLine()) != null) {
                buffer.append(line + "₩n");
            }
        } catch (Exception e) {
            handler.post(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(context, e.getMessage(),
                                                    Toast.LENGTH_SHORT).show();
                }
            });
        }
    }
    public String getResult() {
        return buffer.toString();
    }
}
```

URLConnection 예제 2

- AsyncTask의 동작 순서
  - execute( ) 명령어를 통해 AsyncTask을 실행
  - AsyncTask로 Background 작업을 실행하기 전에 onPreExcuted( )가 실행. 이 부분에는 Image Loading 작업이라면 Loading 중 Image를 띄워 놓기 등, Thread 작업 이전에 수행할 동작을 구현
  - 새로운 Thread에서 Background 작업을 수행 execute( ) 메소드를 호출할 때 사용한 매개변수를 전달 받음
  - doInBackground( )에서 중간 중간 진행 상태를 UI에 Update 하도록 하려면 publishProgress( ) 메소드를 호출
  - onProgressUpdate( ) 메소드는 publishProgress( )가 호출 될 때 마다 자동으로 호출
  - doInBackground( ) 메소드에서 작업이 끝나면 onPostExcuted( )로 결과 매개변수를 반환하면서 그 반환 값을 통해 Thread 작업이 끝났을 때의 동작을 구현

URLConnection 예제 3

- URLConnection을 이용하여 image를 Down 받아보자

URLConnection 예제 3

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity3">
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="top|center_horizontal"
        android:layout_margin="25dp"
        android:text="Display Image From URL" />
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

URLConnection 예제 3

- MainActivity.JAVA

```
public class MainActivity3 extends AppCompatActivity {
    String url = "http://192.168.219.100:8080/image/image.jpg";
    ImageView image;
    int type = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);

        image = findViewById(R.id.imageView);
        Button button = findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (type == 1) {
```

URLConnection 예제 3

- MainActivity.JAVA

```
        StrictMode.ThreadPolicy policy =
                new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        ImageDown downLoad = new ImageDown(MainActivity3.this);
        image.setImageBitmap(downLoad.download(url));
    } else if (type == 2) {
        ImageDownTask task = new ImageDownTask(MainActivity3.this);
        try {
            image.setImageBitmap(task.execute(url).get());
        } catch (ExecutionException | InterruptedException e) {
            Toast.makeText(getBaseContext(), e.getMessage(),
                                                Toast.LENGTH_SHORT).show();
        }
    } else if (type == 3) {
```

URLConnection 예제 3

- MainActivity.JAVA

```
            ImageDownThread thread =
                                    new ImageDownThread(MainActivity3.this, url);
            thread.start();
            try {
                thread.join();
            } catch (InterruptedException e) {
                Toast.makeText(getBaseContext(), e.getMessage(),
                                                Toast.LENGTH_SHORT).show();
            }
            image.setImageBitmap(thread.getResult());
        } else {
```

URLConnection 예제 3

- MainActivity.JAVA

```
                ImageDownRunnable runnable =
                        new ImageDownRunnable(MainActivity3.this, url);
                Thread thread = new Thread(runnable);
                thread.start();
                try {
                    thread.join();
                } catch (InterruptedException e) {
                    Toast.makeText(getBaseContext(), e.getMessage(),
                                                        Toast.LENGTH_SHORT).show();
                }
                image.setImageBitmap(runnable.getResult());
            }
        }
    });
}
```

URLConnection 예제 3

- MainActivity.JAVA

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater menuInflater = getMenuInflater();
    menuInflater.inflate(R.menu.menu, menu);
    return true;
}
```

URLConnection 예제 3

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.item1:
                type = 1;
                break;
            case R.id.item2:
                type = 2;
                break;
            case R.id.item3:
                type = 3;
                break;
            case R.id.item4:
                type = 4;
        }
        image.setImageBitmap(null);
        item.setChecked(true);
        return true;
    }
}
```

URLConnection 예제 3

- ImageDown.JAVA

```
public class ImageDown {
    private Context context;

    public ImageDown(Context context) {
        this.context = context;
    }

    public Bitmap download(String page) {
        Bitmap bitmap = null;
        try {
            URL url = new URL(page);
            URLConnection conn = url.openConnection();
            if (conn != null) {
                InputStream inputStream = conn.getInputStream();
                bitmap = BitmapFactory.decodeStream(inputStream);
                inputStream.close();
```

URLConnection 예제 3

- ImageDown.JAVA

```
            } else {
                Toast.makeText(context, "Network이 연결되지 않았음",
                                        Toast.LENGTH_SHORT).show();
            }
        } catch (IOException e) {
            Toast.makeText(context, e.getMessage(), Toast.LENGTH_SHORT).show();
        }

        return bitmap;
    }
}
```

URLConnection 예제 3

- ImageDownTask.JAVA

```
public class ImageDownTask extends AsyncTask<String, String, Bitmap> {
    private Context context;

    public ImageDownTask(Context context) {
        this.context = context;
    }

    @Override
    protected Bitmap doInBackground(String... strings) {
        Bitmap bitmap = null;
        try {
            URL url = new URL(strings[0]);
            URLConnection conn = url.openConnection();
            if (conn != null) {
                conn.setConnectTimeout(10000);
                conn.setDoInput(true);
                conn.setDoOutput(false);
```

URLConnection 예제 3

- ImageDownTask.JAVA

```
                InputStream inputStream = conn.getInputStream();
                bitmap = BitmapFactory.decodeStream(inputStream);
                inputStream.close();
            } else {
                publishProgress("Network이 연결되지 않았음");
            }
        } catch (IOException e) {
            publishProgress(e.getMessage());
        }

        return bitmap;
    }

    @Override
    protected void onProgressUpdate(String... values) {
        Toast.makeText(context, values[0], Toast.LENGTH_SHORT).show();
    }
}
```

URLConnection 예제 3

- ImageDownThread.JAVA

```
public class ImageDownThread extends Thread {
    private Context context;
    private String page;
    private Handler handler = new Handler();
    private Bitmap bitmap = null;

    public ImageDownThread(Context context, String page) {
        this.context = context;
        this.page = page;
    }

    @Override
    public void run() {
        try {
            URL url = new URL(page);
            URLConnection conn = url.openConnection();
```

URLConnection 예제 3

- ImageDownThread.JAVA

```
        if (conn != null) {
            conn.setConnectTimeout(10000);
            conn.setDoInput(true);
            conn.setDoOutput(false);
            InputStream inputStream = conn.getInputStream();
            bitmap = BitmapFactory.decodeStream(inputStream);
            inputStream.close();
        } else {
            handler.post(new Runnable() {
                public void run() {
                    Toast.makeText(context, "Network이 연결되지 않았음",
                                                    Toast.LENGTH_SHORT).show();
                }
            });
        }
    } catch (IOException e) {
```

URLConnection 예제 3

- ImageDownThread.JAVA

```
            handler.post(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(context, e.getMessage(),
                                                        Toast.LENGTH_SHORT).show();
                }
            });
        }
    }

    public Bitmap getResult() {
        return bitmap;
    }
}
```

URLConnection 예제 3

- ImageDownRunnable.JAVA

```
public class ImageDownRunnable implements Runnable {
    private Context context;
    private String page;
    private Handler handler = new Handler();
    private Bitmap bitmap = null;

    public ImageDownRunnable(Context context, String page) {
        this.context = context;
        this.page = page;
    }

    @Override
    public void run() {
        try {
            URL url = new URL(page);
            URLConnection conn = url.openConnection();
```

URLConnection 예제 3

- ImageDownRunnable.JAVA

```
        if (conn != null) {
            conn.setConnectTimeout(10000);
            conn.setDoInput(true);
            conn.setDoOutput(false);
            InputStream inputStream = conn.getInputStream();
            bitmap = BitmapFactory.decodeStream(inputStream);
            inputStream.close();
        } else {
            handler.post(new Runnable() {
                public void run() {
                    Toast.makeText(context, "Network이 연결되지 않았음",
                                                    Toast.LENGTH_SHORT).show();
                }
            });
        }
    } catch (IOException e) {
```

URLConnection 예제 3

- ImageDownRunnable.JAVA

```
            handler.post(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(context, e.getMessage(),
                                                        Toast.LENGTH_SHORT).show();
                }
            });
        }
    }

    public Bitmap getResult() {
        return bitmap;
    }
}
```

URLConnection 예제 4

- Data Upload

URLConnection 예제 4

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity4">

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Data UpLoad(Post)"/>
    <TextView
        android:id="@+id/result"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20dp"/>
</LinearLayout>
```

URLConnection 예제 4

- MainActivity.JAVA

```
public class MainActivity4 extends AppCompatActivity {
    String postPage = "https://reqres.in/api/users";
    TextView textView;
    int type = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main4);

        textView = findViewById(R.id.result);
        Button button = findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (type == 1) {
```

URLConnection 예제 4

- MainActivity.JAVA

```
        StrictMode.ThreadPolicy policy =
                new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);

        UpLoad upLoad = new UpLoad(MainActivity4.this);
        textView.setText(upLoad.upload(postPage));
    } else if (type == 2) {
        UpLoadTask task = new UpLoadTask(MainActivity4.this);
        try {
            textView.setText(task.execute(postPage).get());
        } catch (ExecutionException | InterruptedException e) {
            Toast.makeText(getBaseContext(), e.getMessage(),
                                                Toast.LENGTH_SHORT).show();
        }
    } else if (type == 3) {
```

URLConnection 예제 4

- MainActivity.JAVA

```
            UpLoadThread thread =
                                new UpLoadThread(MainActivity4.this, postPage);
            thread.start();
            try {
                thread.join();
            } catch (InterruptedException e) {
                Toast.makeText(getBaseContext(), e.getMessage(),
                                                Toast.LENGTH_SHORT).show();
            }
            textView.setText(thread.getResult());
        } else {
            UpLoadRunnable runnable =
                                new UpLoadRunnable(MainActivity4.this, postPage);
            Thread thread = new Thread(runnable);
            thread.start();
```

URLConnection 예제 4

- MainActivity.JAVA

```
                try {
                    thread.join();
                } catch (InterruptedException e) {
                    Toast.makeText(getBaseContext(), e.getMessage(),
                                                        Toast.LENGTH_SHORT).show();
                }
                textView.setText(runnable.getResult());
            }
        }
    });
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater menuInflater = getMenuInflater();
    menuInflater.inflate(R.menu.menu, menu);
    return true;
}
```

URLConnection 예제 4

- MainActivity.JAVA

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.item1:
            type = 1;
            break;
        case R.id.item2:
            type = 2;
            break;
        case R.id.item3:
            type = 3;
            break;
        case R.id.item4:
            type = 4;
    }
    textView.setText("");
    item.setChecked(true);
    return true;
}
```

URLConnection 예제 4

- UpLoad.JAVA

```
public class UpLoad {
    private Context context;

    public UpLoad(Context context) {
        this.context = context;
    }

    public String upload(String page) {
        StringBuilder builder = new StringBuilder();
        try {
            URL url = new URL(page);
            URLConnection conn = url.openConnection();
            conn.setRequestProperty("Content-Type", "application/json");
            conn.setDoOutput(true);
            conn.setDoInput(true);
            JsonObject postData = new JsonObject();
            postData.addProperty("name", "경복대");
            postData.addProperty("job", "leader");
```

URLConnection 예제 4

- UpLoad.JAVA

```
            OutputStream out = new BufferedOutputStream(conn.getOutputStream());
            BufferedWriter writer = new BufferedWriter(
                                    new OutputStreamWriter(out, "UTF-8"));
            writer.write(postData.toString());
            writer.flush();
            InputStream stream = conn.getInputStream();
            InputStreamReader reader = new InputStreamReader(stream);
            BufferedReader buffer = new BufferedReader(reader);
            String line;
            while ((line = buffer.readLine()) != null) {
                builder.append(line);
            }
            buffer.close();
        } catch (IOException e) {
            Toast.makeText(context, e.getMessage(), Toast.LENGTH_SHORT).show();
        }
        return builder.toString();
    }
}
```

URLConnection 예제 4

- UpLoadTask.JAVA

```
public class UpLoadTask extends AsyncTask<String, String, String> {
    private Context context;

    public UpLoadTask(Context context) {
        this.context = context;
    }

    @Override
    protected String doInBackground(String... strings) {
        StringBuilder builder = new StringBuilder();
        try {
            URL url = new URL(strings[0]);
            URLConnection conn = url.openConnection();
            conn.setRequestProperty("Content-Type", "application/json");
            conn.setDoOutput(true);
            conn.setDoInput(true);
            JsonObject postData = new JsonObject();
            postData.addProperty("name", "경복대");
            postData.addProperty("job", "leader");
```

URLConnection 예제 4

- UpLoadTask.JAVA

```
        OutputStream out = new BufferedOutputStream(conn.getOutputStream());
        BufferedWriter writer = new BufferedWriter(
                                                new OutputStreamWriter(out, "UTF-8"));
        writer.write(postData.toString());
        writer.flush();
        InputStream stream = conn.getInputStream();
        InputStreamReader reader = new InputStreamReader(stream);
        BufferedReader buffer = new BufferedReader(reader);
        String line;
        while ((line = buffer.readLine()) != null) {
            builder.append(line);
        }
        buffer.close();
    } catch (IOException e) {
        publishProgress(e.getMessage());
    }
    return builder.toString();
}
```

URLConnection 예제 4

- UpLoadTask.JAVA

```
    @Override
    protected void onProgressUpdate(String... values) {
        Toast.makeText(context, values[0], Toast.LENGTH_SHORT).show();
    }
}
```

URLConnection 예제 4

- UpLoadThread.JAVA

```
public class UpLoadThread extends Thread{
    private Context context;
    private String page;
    private Handler handler = new Handler();
    private StringBuilder builder = new StringBuilder();

    public UpLoadThread(Context context, String page) {
        this.context = context;
        this.page = page;
    }

    @Override
    public void run() {
        try {
            URL url = new URL(page);
            URLConnection conn = url.openConnection();
            conn.setRequestProperty("Content-Type", "application/json");
            conn.setDoOutput(true);
            conn.setDoInput(true);
```

URLConnection 예제 4

- UpLoadThread.JAVA

```
JsonObject postData = new JsonObject();
postData.addProperty("name", "경복대");
postData.addProperty("job", "leader");

OutputStream out = new BufferedOutputStream(conn.getOutputStream());
BufferedWriter writer = new BufferedWriter(
                        new OutputStreamWriter(out, "UTF-8"));
writer.write(postData.toString());
writer.flush();
InputStream stream = conn.getInputStream();
InputStreamReader reader = new InputStreamReader(stream);
BufferedReader buffer = new BufferedReader(reader);
String line;
while ((line = buffer.readLine()) != null) {
    builder.append(line);
}
buffer.close();
```

URLConnection 예제 4

- UpLoadThread.JAVA

```
        } catch (IOException e) {
            handler.post(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(context, e.getMessage(),
                                                        Toast.LENGTH_SHORT).show();
                }
            });
        }
    }

    public String getResult() {
        return builder.toString();
    }
}
```

URLConnection 예제 4

- UpLoadRunnable.JAVA

```
public class UpLoadRunnable implements Runnable{
    private Context context;
    private String page;
    private Handler handler = new Handler();
    private StringBuilder builder = new StringBuilder();

    public UpLoadRunnable(Context context, String page) {
        this.context = context;
        this.page = page;
    }

    @Override
    public void run() {
        try {
            URL url = new URL(page);
            URLConnection conn = url.openConnection();
            conn.setRequestProperty("Content-Type", "application/json");
            conn.setDoOutput(true);
            conn.setDoInput(true);
```

URLConnection 예제 4

- UpLoadRunnable.JAVA

```
JsonObject postData = new JsonObject();
postData.addProperty("name", "경복대");
postData.addProperty("job", "leader");

OutputStream out = new BufferedOutputStream(conn.getOutputStream());
BufferedWriter writer = new BufferedWriter(
                        new OutputStreamWriter(out, "UTF-8"));
writer.write(postData.toString());
writer.flush();
InputStream stream = conn.getInputStream();
InputStreamReader reader = new InputStreamReader(stream);
BufferedReader buffer = new BufferedReader(reader);
String line;
while ((line = buffer.readLine()) != null) {
    builder.append(line);
}
buffer.close();
```

URLConnection 예제 4

- UpLoadRunnable.JAVA

```
        } catch (IOException e) {
            handler.post(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(context, e.getMessage(),
                                                        Toast.LENGTH_SHORT).show();
                }
            });
        }
    }

    public String getResult() {
        return builder.toString();
    }
}
```