

Android class 05.

액티비티 간의 정보 주고받기

```
Intent intent = new Intent(IntentMainActivity.this, IntentSubActivity.class);
intent.putExtra("name", name);
intent.putExtra("age", age);
intent.putExtra("tel", tel);
startActivity(intent);
```

보내기

```
Intent intent = getIntent();
txt_name.append(intent.getStringExtra("name"));
txt_age.append(intent.getStringExtra("age"));
txt_tel.append(intent.getStringExtra("tel"));
```

받기

달력입력받기

```
View.OnClickListener dateClick = new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        //달력에 최초로 표기 될 날짜를 구한다.
        Calendar now = Calendar.getInstance();
        int year = now.get(Calendar.YEAR);
        int month = now.get(Calendar.MONTH); //달은 0부터 시작
        int day = now.get(Calendar.DAY_OF_MONTH);
        dialog = new DatePickerDialog(IntentMainActivity.this,
            dateSetListener, year, month, day);
```

```
        dialog.show();
    }
};

//달력의 변경사항을 감지하는 이벤트 감지자
DatePickerDialog.OnDateSetListener dateSetListener = new
DatePickerDialog.OnDateSetListener() {
    @Override
    public void onDateSet(DatePicker datePicker, int year, int month,
int day) {
        String result = String.format("%d-%02d-%02d",year, month+1,
day);
        edit_b_day.setText(result);
    }
};
```

뒤로가기 두번 눌러 종료

```
int clickCount = 0;
@Override
public void onBackPressed() {
    handler.sendEmptyMessageDelayed(0,2000);
    clickCount++;
    if(clickCount==2){
        finish();
    }
    Toast.makeText(this, "한번더 눌러야 종료됩니다.",
    Toast.LENGTH_SHORT).show();
}

Handler handler = new Handler(){
    @Override
    public void handleMessage(@NonNull Message msg) {
        clickCount=0;
    }
};
```

Shared Preferences [임시 저장소]

간단한 정보를 저장해야 하는 경우 DB를 쓰기에 오히려 손해인 경우가 있다.
내부적으로 파일 형태로 기록되는 저장방식을 통해 휴대폰이 종료되거나 재시작되어도
앱을 삭제하기 전까지 남아있는 데이터를 만들 수 있는 것

```
sharedPreferences = getSharedPreferences("SHARE", MODE_PRIVATE);
```

저장모드

```
sharedPreferences = getSharedPreferences("SHARE", MODE_PRIVATE);
value = sharedPreferences.getInt("save", 0);
text_value.setText(String.valueOf(value));
```

getInt 두번째 인자는 default 값이다.

```
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putInt("save", value);
editor.commit(); //물리적으로 세이브 데이터를 저장하는 작업
```

저장하기.

Inflater 인플레이터

새로운 참조파일 xml 파일을 만든후에 임의의 레이아웃에 삽입해보는 실습.

```
LinearLayout layout;
LayoutInflater inflater;

@Override
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_inflater);

    layout = findViewById(R.id.layout);
    inflater = (LayoutInflater)
getSystemService(LAYOUT_INFLATER_SERVICE);
    View inner = inflater.inflate(R.layout.my_inflater, inner);
    //LinearLayout 안쪽에 inner View를 추가 해준다.
    //layout.addView(inner);
}
```

인플레이터로 선언한 레이아웃 속 내부 버튼 선언

```
btn1 = findViewById(R.id.btn_1);
btn1 = inner.findViewById(R.id.btn_1);
```

두개 모두 사용가능하지만 inner로 내부를 표현해서 보여주는걸 추천

NAVER API 사용 -검색

AndroidManifest.xml -인터넷 사용 권한 주기

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.lece.ex_naverapi">

    <uses-permission android:name="android.permission.INTERNET" />
    - 인터넷을 사용하기 위해 꼭 필요한 권한 -
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Ex_NaverAPI"
        tools:targetApi="31"
        android:networkSecurityConfig="@xml/network_security_config">
        <activity
            android:name=".NaverActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

network_security_config.xml (경로 - res / xml /)

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <!--Set application-wide security config using base-config tag.-->
    <base-config cleartextTrafficPermitted="true"/>
</network-security-config>
```

JSON 으로 받기

```
public class ParserJSON {  
    //웹에서 받은 요소들을 저장 및 List로 저장  
    BookVO vo;  
    String myQuery = ""; //검색어  
    public String connectNaver(){  
        ArrayList<BookVO> bookList = new ArrayList<>();  
        String result="";  
        try {  
            //검색어 ( myQuery ) 를 UTF-8 형태로 인코딩  
            myQuery =  
                URLEncoder.encode(NaverActivity.search.getText().toString(), "UTF-8");  
  
            String apiUrl =  
                "https://openapi.naver.com/v1/search/book.json?query="+myQuery+"&display=100";  
  
            URL url = new URL(apiUrl);  
            HttpURLConnection httpURLConnection = (HttpURLConnection)  
url.openConnection();  
  
            //발급 받은 ID 와 Secret을 서버로 전달  
            httpURLConnection.setRequestMethod("GET");  
            httpURLConnection.setRequestProperty("X-Naver-Client-Id",  
NaverProperty.id);  
  
            httpURLConnection.setRequestProperty("X-Naver-Client-Secret", NaverProperty.key)  
;  
            //httpURLConnection.setRequestProperty("Content-type",  
"application/json");  
            //URL을 수행하여 받은 자원들을 parsing 해야한다.  
  
            //url에 정보를 불러와서 저장  
            StringBuilder sb = new StringBuilder();  
            BufferedReader br = new BufferedReader(new  
InputStreamReader(httpURLConnection.getInputStream(), "UTF-8"));  
            String inputLine;  
            while((inputLine=br.readLine()) != null){  
                sb.append(inputLine);  
            }  
            result = sb.toString();  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

```

        parsingJSON(result);
        return result;
    }

private void parsingJSON(String result) {
    try {
        JSONObject jsonObject = new JSONObject(result);
        JSONArray jsonArray = jsonObject.getJSONArray("items");

        for (int i = 0; i < jsonArray.length(); i++) {
            JSONObject bookObject = jsonArray.getJSONObject(i);
            System.out.println("bookObject.get(\"title\") = " +
bookObject.get("title"));
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
}

```

파싱해서 받아오는것은 가능 -> VO 저장만 진행하면됨

```

ParserJSON parserJSON = new ParserJSON();
search_btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Thread thread = new Thread(){
            @Override
            public void run() {
                result = parserJSON.connectNaver();
            }
        };
        thread.start();
        try {
            thread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        text_result.setText(result);
    }
});

```

비동기 통신 (AsyncTask)

- 백그라운드에서의 서버통신을 위해 반드시 필요한 클래스 세개의 제네릭 타입을 가지고 있습니다. 파라미터 타입, onProgressUpdate가 오버라이딩 되어 있다면 이 메서드에서 사용할 타입, 반환형

```
class NaverAsync extends AsyncTask<String, Void, ArrayList<BookVO>>{  
  
    @Override  
    protected ArrayList<BookVO> doInBackground(String... strings) {  
        return parser.connectNaver();  
    }  
  
    @Override  
    protected void onPostExecute(ArrayList<BookVO> bookVOS) {  
        //최종 작업결과를 받는 메서드 최종 return 값을 bookVOS가 받음  
        super.onPostExecute(bookVOS);  
  
    }  
}
```

String... strings : variable arguments 배열의 개수를 따지지 않고 파라미터로 들어오는 모든 것들을 배열로 만들어 주는 기능을 가진다.

ListView 간선하기

순서

1. Adapter 클래스 생성
2. ListView xml 파일 인플레이터
3. 리스트뷰에 하나씩 간선할수 있게 Set

```
public class ViewModelAdapter extends ArrayAdapter<BookVO> {
    Context context;
    int resource;
    BookVO bookVO;
    ArrayList<BookVO> list;
    TextView book_title, book_author, book_price;
    ImageView book_image;
    ImageAsync imageAsync;

    public ViewModelAdapter(Context context, int resource, ArrayList<BookVO> list) {
        super(context, resource, list);
        this.context = context;
        this.resource = resource;
        this.list = list;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        //myListView.setAdapter(adapter) 했을 때 호출되는 메서드(getView())
        //생성자의 파라미터를 받은 사이즈만큼 getView() 메서드가 반복 호출
        imageAsync = new ImageAsync();

        //인플레이터로 Layout을 view로 바꿔주는 작업도 여기서 수행해 준다.
        LayoutInflator layoutInflater = (LayoutInflator)
        context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        //book_item.xml 파일을 view 형태로 변경해준다.
        convertView = layoutInflater.inflate(resource, null);

        BookVO bookVO = list.get(position);
        book_title = convertView.findViewById(R.id.book_title);
        book_author = convertView.findViewById(R.id.book_author);
        book_price = convertView.findViewById(R.id.book_price);
        book_image = convertView.findViewById(R.id.book_image);
        book_title.setText(bookVO.getB_title());
        book_author.setText(bookVO.getB_author());
        book_price.setText(bookVO.getB_price() + "원");
        imageAsync.execute(bookVO.getB_image());

        return convertView;
    }
}
```

```
//이미지를 위한 AsyncTask 내부 클래스 생성
class ImageAsync extends AsyncTask<String, Void, Bitmap>{

    @Override
    protected Bitmap doInBackground(String... strings) {
        Bitmap bitmap = null;
        try {
            String img_url = strings[0];
            URL url = new URL(img_url);
            bitmap =
BitmapFactory.decodeStream(url.openConnection().getInputStream());
        } catch (IOException e) {
            e.printStackTrace();
        }
        return bitmap;
    }
    @Override
    protected void onPostExecute(Bitmap bitmap) {
        book_image.setImageBitmap(bitmap);
    }
}
}
```

Naver AsyncTask 수정 -

```
class NaverAsync extends AsyncTask<String, Void, ArrayList<BookVO>>{

    @Override
    protected ArrayList<BookVO> doInBackground(String... strings) {
        return parser.connectNaver();
    }

    @Override
    protected void onPostExecute(ArrayList<BookVO> bookVOS) {
        //최종 작업결과를 받는 메서드 최종 return 값을 bookVOS가 받음

        //ListView를 그리기 위해 Adapter 클래스에게 넘겨줘야 한다.
        viewModelAdapter = new ViewModelAdapter(NaverActivity.this,
R.layout.book_item, bookVOS);
        myListview.setAdapter(viewModelAdapter);
    }
}
```