한국ICT인재개발원


[한국ICT인재개발원] 안드로이드

# 13. 리사이클러 뷰와 비동기 통신

# 강의자 소개 : 채종훈 강사

**前) 광고데이터 분석 1년**

**前) IT강의 경력 2년 6개월**

**前) 머신러닝을 활용한 데이터 분석 프로젝트반 운영 1년**

**前) 리그오브 레전드 데이터 분석 등...**

**現) 국비반 강의 진행중**

# 리사이클러 뷰와 비동기 통신



안드로이드의 비동기 통신은 **Retrofit2**를 사용합니다.

마치 **javascript**의 **getJSON, ajax**요청처럼 **Retrofit2**를 활용해

특정 이벤트마다 **json** 데이터를 요청해 받아온 다음

그 데이터를 화면에 표출하면 됩니다.

이 과정에서 **ListView**를 대신할 **RecyclerView**를 사용해서

좀 더 자유롭게 꾸며진 뷰를 활용해보도록 하겠씁니다.

# 리사이클러 뷰와 비동기 통신

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

먼저 외부 접속을 수행하기 위해

**manifests**의 내부에 **uses-permision**을 추가해주고,

**application** 태그 내부에도 **userCleartextTraffic**을 설정해줍니다.

# 리사이클러 뷰와 비동기 통신



```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
    implementation 'com.squareup.retrofit2:retrofit:2.6.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.6.0'
}
```

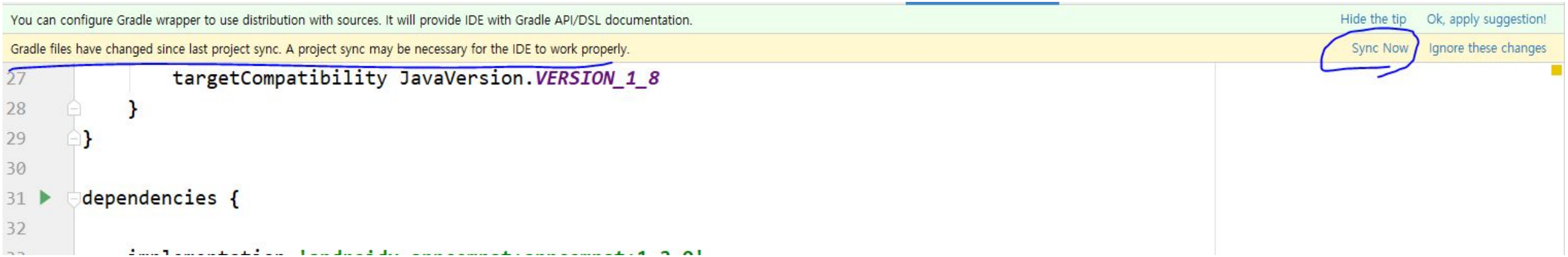
다음 **Retrofit2**를 사용하기 위한 세팅부터 하겠습니다.

**Gradle Scripts** 하단의 **build.gradle**의 **module : ….app**

을 열고 **dependencies** 내부에 위와 같이 집어넣어줍니다.

# 리사이클러 뷰와 비동기 통신






변동이 생긴 부분에 대해서 **Sync now**를 클릭해서 적용해주면 됩니다.

마치 **pom.xml**에 스프링 프로젝트의 메이븐 의존성 설정을 했듯

안드로이드 스튜디오에서 외부 라이브러리를 끌어다 쓰는것입니다.

# 리사이클러 뷰와 비동기 통신



```
implementation 'com.squareup.retrofit2:retrofit:2.6.0'
implementation 'com.squareup.retrofit2:converter-gson:2.6.0'
```

위의 **retrofit2**는 **ajax** 통신을 안드로이드에서 구현할 때 사용합니다.

**gson**은 **json**파일을 다시 안드로이드 내부적으로 쓸 수 있게 만드는 역할을 합니다.

비동기 통신은 이 두 개를 활용해 구현합니다.

# 리사이클러 뷰와 비동기 통신




비동기 통신 백엔드 로직은 구현 이전에 먼저 **API** 호출 예제를 살펴봅니다.

https://www.kobis.or.kr/kobisopenapi/homepg/apiservice/searchServiceInfo.do

영진위의 일별 박스오피스 순위를 받아와보겠습니다.

### 3. 인터페이스

• 요청 인터페이스

| 요청 변수 | 값 | 설명 |
|---|---|---|
| key | 문자열(필수) | 발급받은키 값을 입력합니다. |
| targetDt | 문자열(필수) | 조회하고자 하는 날짜를 yyyymmdd 형식으로 입력합니다. |
| itemPerPage | 문자열 | 결과 ROW 의 개수를 지정합니다.(default : "10", 최대 : "10") |
| multiMovieYn | 문자열 | 다양성 영화/상업영화를 구분지어 조회할 수 있습니다.<br>"Y" : 다양성 영화 "N" : 상업영화 (default : 전체) |
| repNationCd | 문자열 | 한국/외국 영화별로 조회할 수 있습니다.<br>"K: : 한국영화 "F" : 외국영화 (default : 전체) |
| wideAreaCd | 문자열 | 상영지역별로 조회할 수 있으며, 지역코드는 공통코드 조회 서비스에서 "0105000000" 로서 조회된 지역코드입니다. (default : 전체) |

**key, targetDt를 기본 URL인**

**http://kobis.or.kr/kobisopenapi/webservice/rest/boxoffice/searchDailyBoxOfficeList.json**
뒤에 붙여서 가져오겠습니다.

# 리사이클러 뷰와 비동기 통신

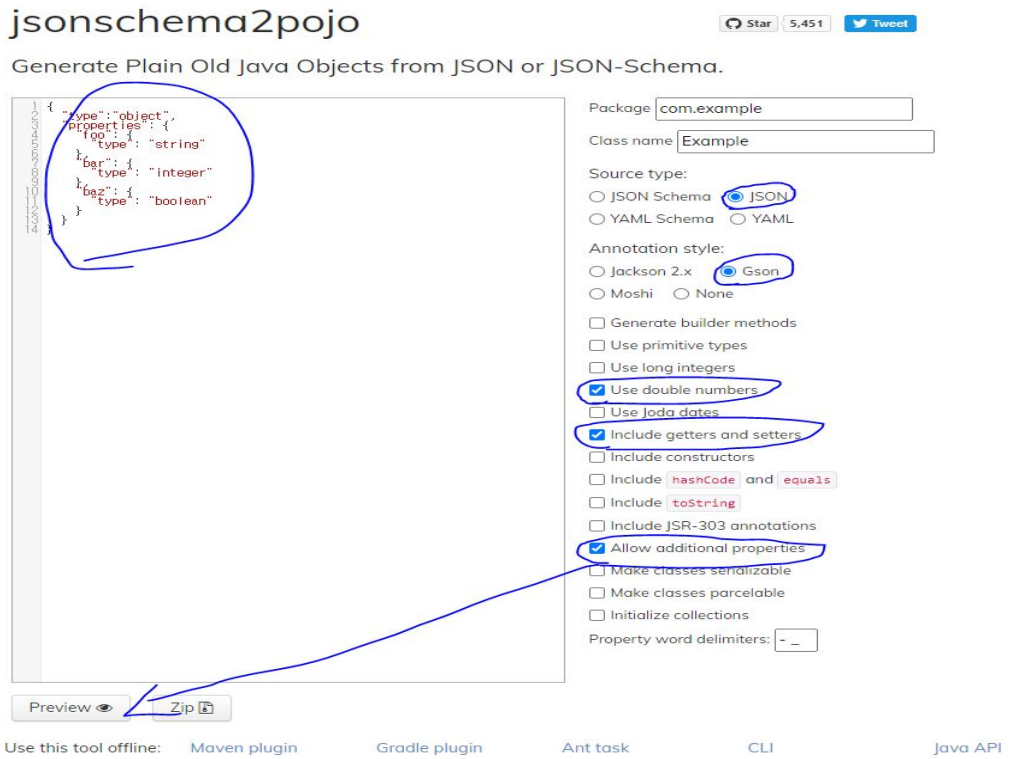


{"boxOfficeResult":{"boxofficeType":"일별 박스오피스","showRange":"20210101~20210101","dailyBoxOfficeList":[{"rnum":"1","rank":"1","rankInten":"0","rankOldAndNew":"OLD","movieCd":"20192567","movieNm":"원더 우먼 1984","openDt":"2020-12-
23","salesAmt":"274407080","salesShare":"51.4","salesInten":"41920830","salesChange":"18","salesAcc":"3764838540","audiCnt":"29884","audiInten":"2462","audiChange":"9","audiAcc":"420870","scrnCnt":"1877","showCnt":"4329"},
{"rnum":"2","rank":"2","rankInten":"1","rankOldAndNew":"OLD","movieCd":"20202703","movieNm":"뱅가드","openDt":"2020-12-
30","salesAmt":"44924000","salesShare":"8.4","salesInten":"2922600","salesChange":"7","salesAcc":"124210900","audiCnt":"5190","audiInten":"232","audiChange":"4.7","audiAcc":"15570","scrnCnt":"259","showCnt":"516"},
{"rnum":"3","rank":"3","rankInten":"-1","rankOldAndNew":"OLD","movieCd":"20040725","movieNm":"화양연화","openDt":"2000-10-
20","salesAmt":"38790050","salesShare":"7.3","salesInten":"-5062310","salesChange":"-11.5","salesAcc":"438770750","audiCnt":"4314","audiInten":"-878","audiChange":"-16.9","audiAcc":"52328","scrnCnt":"434","showCnt":"607"},
{"rnum":"4","rank":"4","rankInten":"0","rankOldAndNew":"OLD","movieCd":"20201002","movieNm":"조제","openDt":"2020-12-
10","salesAmt":"29130490","salesShare":"5.5","salesInten":"-6914350","salesChange":"-19.2","salesAcc":"1702558060","audiCnt":"3338","audiInten":"-1103","audiChange":"-24.8","audiAcc":"189985","scrnCnt":"472","showCnt":"625"},
{"rnum":"5","rank":"5","rankInten":"0","rankOldAndNew":"OLD","movieCd":"20192193","movieNm":"도굴","openDt":"2020-11-
04","salesAmt":"25599980","salesShare":"4.8","salesInten":"-4481330","salesChange":"-14.9","salesAcc":"13790280390","audiCnt":"2978","audiInten":"-866","audiChange":"-22.5","audiAcc":"1526492","scrnCnt":"294","showCnt":"353"},
{"rnum":"6","rank":"6","rankInten":"0","rankOldAndNew":"OLD","movieCd":"20202647","movieNm":"나이팅게일","openDt":"2020-12-
30","salesAmt":"19084320","salesShare":"3.6","salesInten":"-1865050","salesChange":"-8.9","salesAcc":"59429790","audiCnt":"2211","audiInten":"-337","audiChange":"-13.2","audiAcc":"7573","scrnCnt":"307","showCnt":"410"},
{"rnum":"7","rank":"7","rankInten":"0","rankOldAndNew":"NEW","movieCd":"20202130","movieNm":"빅풋 주니어2: 패밀리가 떴다","openDt":"2021-01-
06","salesAmt":"12584970","salesShare":"2.4","salesInten":"12584970","salesChange":"100","salesAcc":"12584970","audiCnt":"1565","audiInten":"1565","audiChange":"100","audiAcc":"1565","scrnCnt":"236","showCnt":"293"},
{"rnum":"8","rank":"8","rankInten":"0","rankOldAndNew":"OLD","movieCd":"20167904","movieNm":"라라랜드","openDt":"2016-12-
07","salesAmt":"8234000","salesShare":"1.5","salesInten":"-3207000","salesChange":"-28","salesAcc":"30881366808","audiCnt":"1175","audiInten":"-437","audiChange":"-27.1","audiAcc":"3743214","scrnCnt":"33","showCnt":"49"},
{"rnum":"9","rank":"9","rankInten":"0","rankOldAndNew":"OLD","movieCd":"20202128","movieNm":"100% 울프: 푸들이 될 순 없어","openDt":"2020-12-
24","salesAmt":"7889510","salesShare":"1.5","salesInten":"-1954090","salesChange":"-19.9","salesAcc":"106850730","audiCnt":"1009","audiInten":"-318","audiChange":"-24","audiAcc":"13366","scrnCnt":"147","showCnt":"164"},
{"rnum":"10","rank":"10","rankInten":"2","rankOldAndNew":"OLD","movieCd":"20181983","movieNm":"이웃사촌","openDt":"2020-11-
25","salesAmt":"8455600","salesShare":"1.6","salesInten":"-1000","salesChange":"0","salesAcc":"3656353560","audiCnt":"981","audiInten":"-82","audiChange":"-7.7","audiAcc":"425745","scrnCnt":"143","showCnt":"168"}]}}

먼저, 브라우저에서의 검색 결과가 위와 같이 나와야 합니다.

이제 저 데이터를 안드로이드에서 요청해 받아올 수 있도록 만드는것이 다음 작업입니다.

# 리사이클러 뷰와 비동기 통신




**VO객체를 내부적으로 설정하기에는 안드로이드 스튜디오가 너무 불친절합니다.**

따라서 외부 도구를 활용합니다.
**https://www.jsonschema2pojo.org/**

# 리사이클러 뷰와 비동기 통신

```java
package com.example;

import javax.annotation.Generated;
import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;

@Generated("jsonschema2pojo")
public class Bar {

@SerializedName("type")
@Expose
private String type;

public String getType() {
return type;
}

public void setType(String type) {
this.type = type;
}

}
```

결과로 위와같이 클래스가 나올 텐데, 복사 붙여넣기로 **VO**를 패키지 내에 만들어주시되, **javax.annotation.Generated**는 현재 사용하는 환경에서 지원하지 않기 때문에 지워주시면 됩니다.
만들어지는 클래스는 총 **3**개입니다.

# 리사이클러 뷰와 비동기 통신



이제 **retrofit**에 대한 내용은 직접 구현하면서 살펴보겠습니다.

**RetrofitInterface**를 이용해 먼저 메서드들을 구현하고,

만들어진 **VO**와 연동시킬 **RetrofitClient** 객체를 이용해 비동기 통신을 진행하도록 할 수 있습니다.

이렇게 구현된 **RetrofitClient**의 동작은 **MainActivity**가 추후에 호출하게 됩니다.

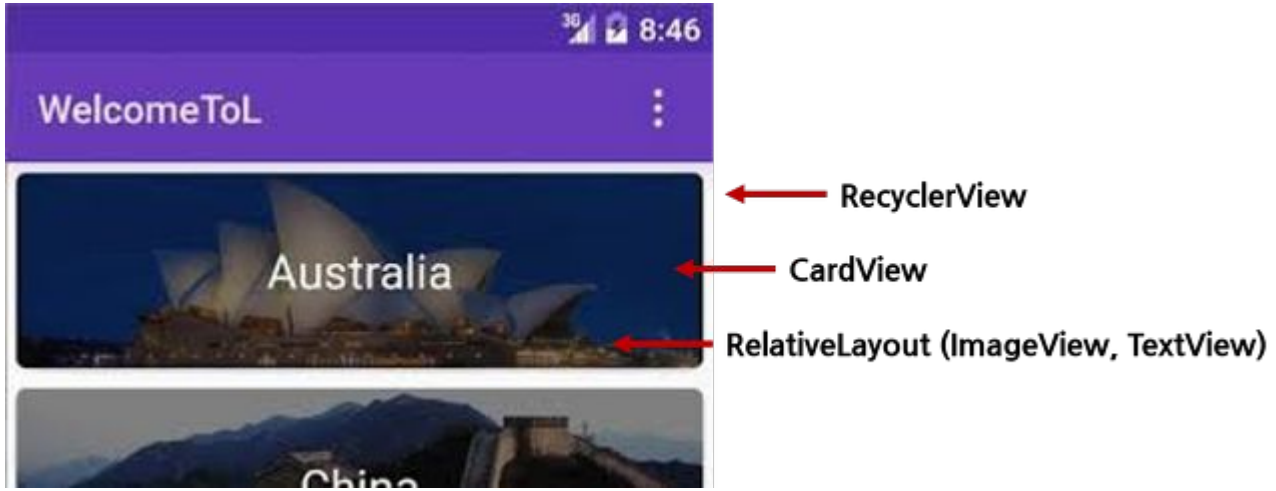
우선 **Retrofit** 자체만 백엔드로직 구현에 활용한 다음

리사이클러 뷰로 프론트단을 작성해보겠습니다.

# 리사이클러 뷰와 비동기 통신



```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```




리사이클러 뷰 역시 **xml파일**에 먼저 구현해 놓고

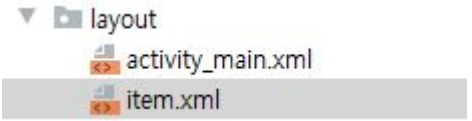
어댑터를 이용해 반복적으로 생성해 사용합니다.

내부에는 보통 **CardView**라는 요소를 이용해 많이 구성합니다.

# 리사이클러 뷰와 비동기 통신

```
<androidx.constraintlayout.widget.ConstraintLa
    xmlns:app="http://schemas.android.com/apk/
    xmlns:tools="http://schemas.android.com/to
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```




이를 위해 리사이클러뷰만 **activity_main.xml** 에 정의해놓고

그 리사이클러 뷰 내에 들어갈 다른 레이아웃을 별도 **xml**파일에 작성한 뒤 합쳐주는 방식을 취합니다.

# 리사이클러 뷰와 비동기 통신

```
: class MovieAdapter extends RecyclerView.Adapter<MovieAdapter.ViewHolder>{

ivate List<DailyBoxOfficeList> items;

blic MovieAdapter(List<DailyBoxOfficeList> items) { this.items = items; }

lonNull
verride
blic MovieAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viev
  View itemView = LayoutInflater.from(parent.getContext()).inflate(R.layout.item ,
  return new ViewHolder(itemView);
```

와중에, **ListView**와는 달리 여러 요소를 한 뷰에 표출할 수 있는데

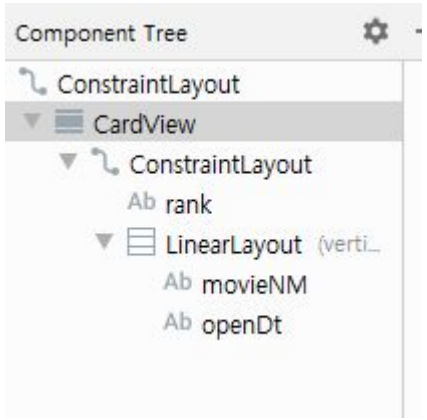
위와같이 **VO**를 **List**형태로 가져온 다음, **ViewHolder**라는것으로 묶어줍니다.

# 리사이클러 뷰와 비동기 통신

```
blic static class ViewHolder extends RecyclerView.Vi
  private TextView rank, movieNm, openDt;

  public ViewHolder(View itemView) {
      super(itemView);
      rank = itemView.findViewById(R.id.rank);
      movieNm = itemView.findViewById(R.id.movieNM);
      openDt = itemView.findViewById(R.id.openDt);
  }

  public void setItem(DailyBoxOfficeList item){
      rank.setText(item.getRank());
      movieNm.setText(item.getMovieNm());
      openDt.setText(item.getOpenDt());
  }
```




**ViewHolder**는 **Adapter**에서 구현하는 내부 클래스인데, 현재 영화 순위같은 경우는
순위, 영화이름, 개봉일이라는 세 개의 정보를 한 리사이클러뷰에
표출할것이므로 위와 같이 세 개의 변수를 하나의 **ViewHolder**에 묶어줍니다.

# 리사이클러 뷰와 비동기 통신



```
setContentView(R.layout.activity_main);

recyclerView = findViewById(R.id.recyclerView);
LinearLayoutManager layoutManager = new LinearLayoutManager(getApplicationContext(),
                                                            LinearLayoutManager.VERTICAL,
                                                                    reverseLayout: false);
recyclerView.setLayoutManager(layoutManager);
```

```
mAdapter = new MovieAdapter(boxOfficeResult.getDailyBoxOfficeList());

recyclerView.setAdapter(mAdapter);
```

어댑터 설정이 끝났다면 리사이클러뷰를 **MainActivity**에서 선언한 후에
**LinearLayoutManager**를 이용해 배열 방향(수직, 수평, 그리드, 불균형 그리드 4가지 중 선택가능)을 설정하고,
얻어온 데이터를 어댑터객체에 적용한 다음 **setAdapter**로 처리하면 됩니다.

# 리사이클러 뷰와 비동기 통신



역시 남은 내용도 모두 실습코드를 작성하면서 설명을 해 드리겠습니다.

**ListView**에 비해 추가해야할 요소가 많아서 어렵게 느껴질 수도 있지만

좀 더 커스터마이징된 리스트뷰를 만들 수 있는 만큼

**DB**와 통신해 가져온 데이터를 좀 더 보기좋게 정렬하는데 큰 도움이 됩니다.

만약 **API**서버나 외부**DB**가 아닌

로컬**PC**의 **DB**를 이용하거나, 미리 만들어둔 스프링 **REST** 서버를 활용하고 싶다면

접속 **DB**주소는 **http://localhost:**포트번호/ 가 **base url**이 아니고

http://10.0.2.2:포트번호 가 대신합니다.

이렇게 하는 이유는 **AVD**의 가상 휴대폰이 본체**PC**와는 별개의 기기로 간주되기 때문에

**AVD**에서 접속할 때는 **localhost**를 외부주소가 아닌 휴대폰 자기자신으로 간주하기 때문입니다.

# 리사이클러 뷰와 비동기 통신

**RetrofitInterface**에서

**@GET()** 대신

**@POST(), @DELETE(), @PUT(), @FETCH()**

를 활용하면 웹 개발 시절의 **ajax** 통신을 그대로 쓸 수 있습니다.

이를 이용하면 백엔드 서버는 하나만 둔 채로

웹 서비스와 앱 서비스를 동시에 개발할 수 있습니다.