آموزش برنامه نویسی اندروید
در محیط اندروید استودیو

آشنایی با اکتیویتی (Activity)

مدرس : سید مهدی مطهری

www.android-studio.ir

به نام خدا

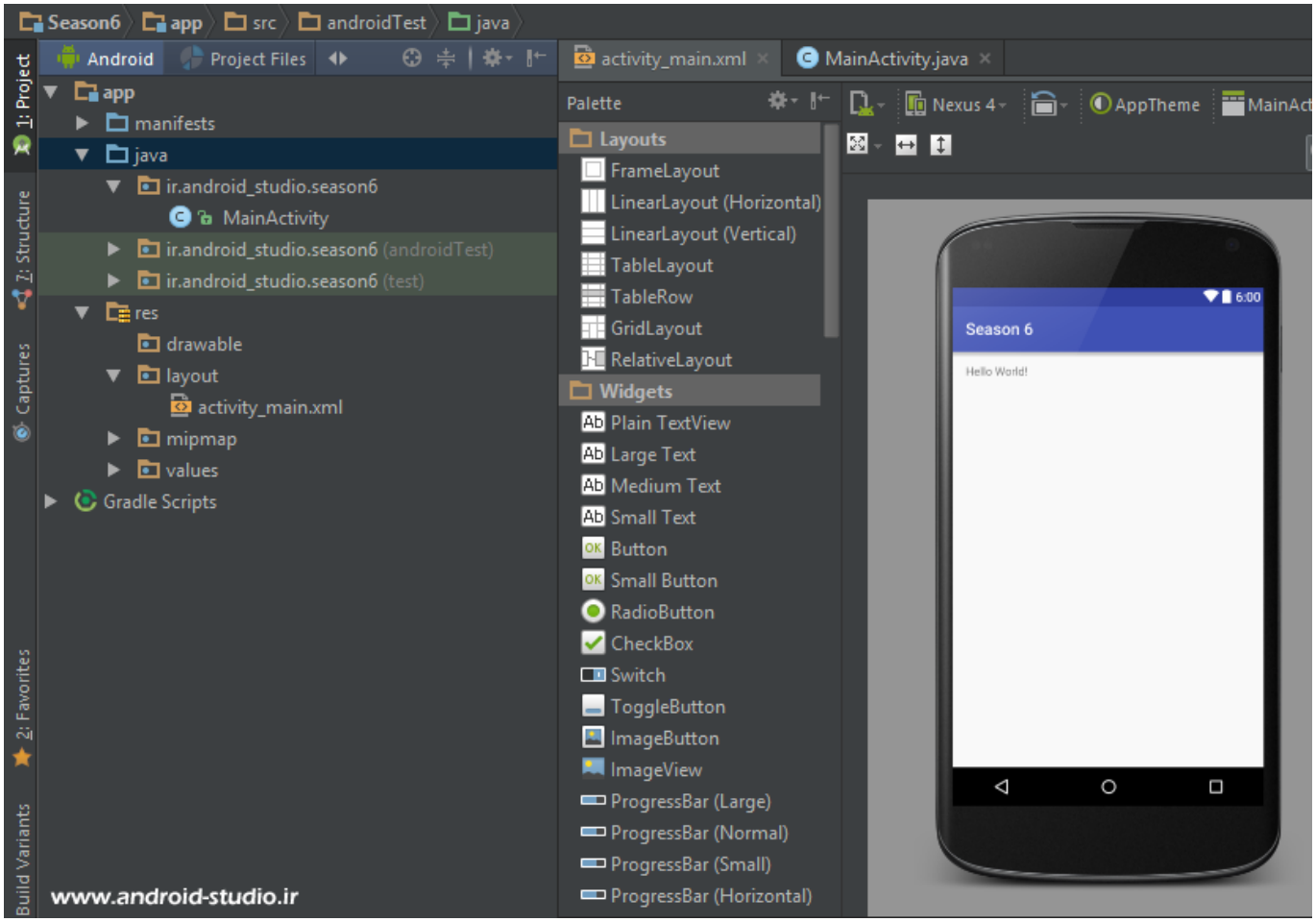
اکتیویتی

ساده بخواهیم بگوییم، هر صفحه از اپلیکیشن را یک Activity می نامیم. یک اپلیکیشن را در نظر بگیرید که با باز کردن آن، صفحه مربوط به ورود توسط وارد کردن نام کاربری و رمز عبور نمایش داده می شود و پس از انجام این موارد، کاربر به صفحه بعدی که محتوای خاصی را نمایش می دهد هدایت می شود. این اَپ دو اکتیویتی را شامل شده است. یک اکتیویتی برای ساخت صفحه فرم ورود و اکتیویتی دوم برای صفحه نمایش محتوا. تعداد اکتیویتی هر اپ بسته به نیاز و سلیقه توسعه دهنده متفاوت بوده و از حداقل یک اکتیویتی تا ده ها عدد متغییر می باشد.

هر اکتیویتی به دو بخش Front-end و Back-end تقسیم می شود.قسمت Front-end به بخشی گفته می شود که کاربر با آن تعامل دارد، یعنی همان User interface یا به اختصار (UI) یا همان رابط کاربری که در اندروید همانطور که در فصل قبل معرفی کردیم، قسمت UI توسط کدهای XML و در فایلی با همین پسوند قرار می گیرند. قسمت Back-end مربوط به کدهای پشت صحنه بوده که از دید کاربر پنهان است که فایلی با پسوند java. و شامل کدهای زبان جاوا می باشد.

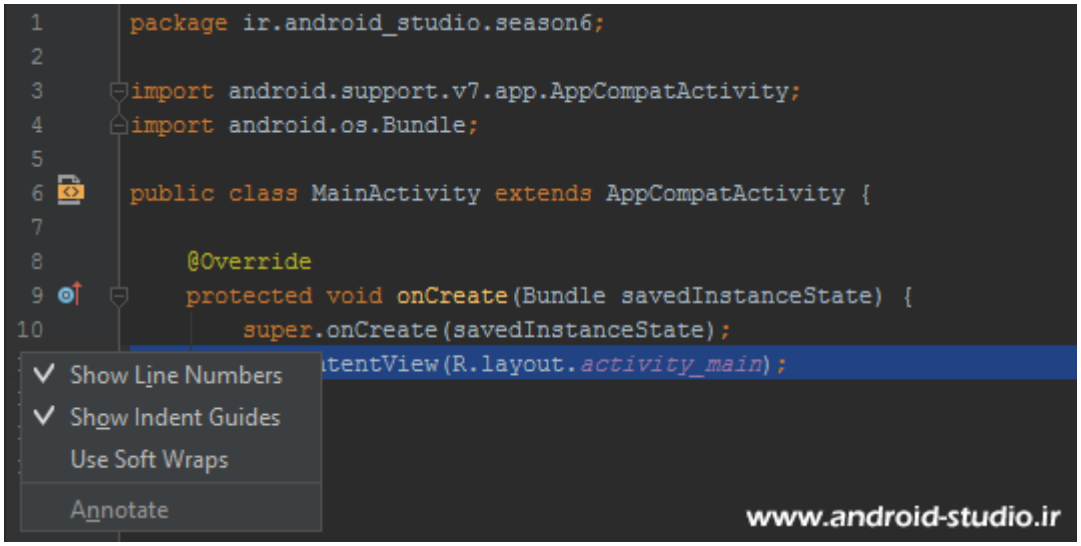
این دو قسمت هر اکتیویتی باید به نحوی به یکدیگر پیوند داده شوند. به عبارتی باید فایل xml را درون فایل java تعریف کنیم.

یک پروژه جدید ایجاد می کنیم تا به صورت عملی بررسی کنیم.



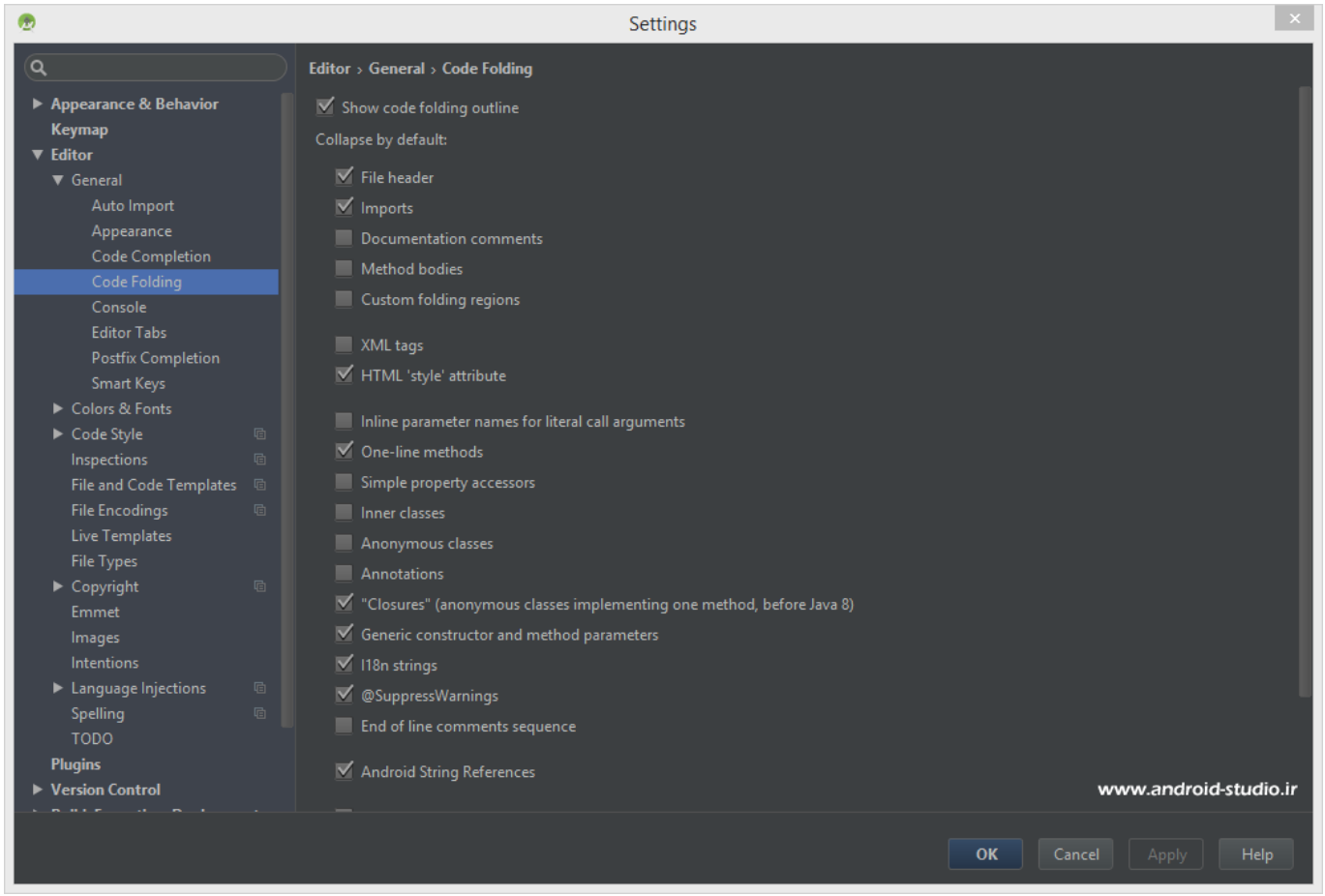
مشاهده می کنید که اندروید استودیو برای پروژه جدید ما به صورت پیش فرض یک اکتیویتی ایجاد نموده که شامل activity_main.xml و MainActivity.java می باشد. البته نام این فایلها را هنگام ساخت پروژه می توانستیم تغییر بدهیم که نیازی نبود. با ساختار layout ها و بخش دیزاین در فصل گذشته آشنا شدیم. حال به بررسی بخش دستورات مربوط به اکتیویتی ها که شامل کدهای جاوا هست می پردازیم.

اگر شماره خطوط سمت چپ کدها را مشاهده نمی کنید با راست کلیک روی این ناحیه و گزینه Show line numbers این قابلیت فعال می شود :



خط اول نام پکیج هست که هنگام ساخت پروژه تعیین کرده ایم. خطوط ۳ و ۴ کلاس (Class) هایی هستند که به صورت پیش فرض روی این اکتیویتی ایمپورت شده اند.اگر کلاس ها در حالت پیش فرض به صورت import... نمایش داده شده، برای نمایش کامل لازم است روی علامت + سمت چپ آن کلیک کنید و یا از طریق تنظیمات اندروید استودیو تعیین کنید تا هیچگاه لیست کلاس های ایمپورت شده جمع نشود :

File > Settings > Editor > General > Code Folding



در اینجا علاوه بر قسمت import ، امکان فعال یا غیر فعال نمودن قابلیت جمع شدن سایر موارد را هم دسترسی دارید. توسط گزینه Show code folding outline می توان به طور کلی این قابلیت را فعال و غیر فعال نمود که به دلخواه بوده و ما فعلا به گزینه File header و Imports اکتفا می کنیم.

کلاس android.support.v7.app.AppCompatActivity مربوط به کتابخانه AppCompat می باشد. این کتابخانه توسط گوگل توسعه داده شده و پشتیبانی و بروزرسانی می شود. ورژن فعلی این کتابخانه ۷ می باشد که با نام appcompat-v7 شناخته می شود. هر چند مدت یک بروزرسانی جدید تحت عنوان Revesion نیز منتشر می شود که در زمان تهیه این فایل آموزشی 24.2.1 جدیدترین نسخه می باشد.کتابخانه appcompat یکی از چندین کتابخانه موجود در پکیج Android Support Library می باشد که قبلا آن را نصب کرده اید (آموزش فصل دوم).

اطلاعات بیشتر و مشاهده آخرین نسخه منتشر شده این کتابخانه و سایر کتابخانه های موجود در Android Support Library :

https://developer.android.com/topic/libraries/support-library/features.html

برای افزودن کتابخانه ها به پروژه چند روش وجود دارد :

۱ - روش آنلاین :

در روش آنلاین به دو نحو می توان کتابخانه را به پروژه افزود :

۱ – ۱ : روش دستی :

در این روش، خط کدی که توسط توسعه دهنده کتابخانه موردنظر معرفی می شود را داخل فایل build.gradle (Module: app) درون dependencies اضافه (اگر dependencies را ندارید خودتان ایجاد کنید) و سپس Sync می کنیم (گزینه Sync Now نمایش داده شده در سمت راست و یا گزینه Sync Project with gradle files) تا کتابخانه از مخزن (سرور) دریافت و به پروژه اضافه شود. این خط کد با کلمه compile شروع می شود و در ادامه نام کتابخانه ذکر شده.به عنوان مثال اگر به لاین ۲۵ دقت کنید کد مربوط به کتابخانه appcompat را مشاهده می کنید که به صورت پیش فرض روی پروژه قرار داده شده. یا اگر این کتابخانه درون پروژه شما موجود نیست و قصد اضافه کردن آن را دارید، کافیست

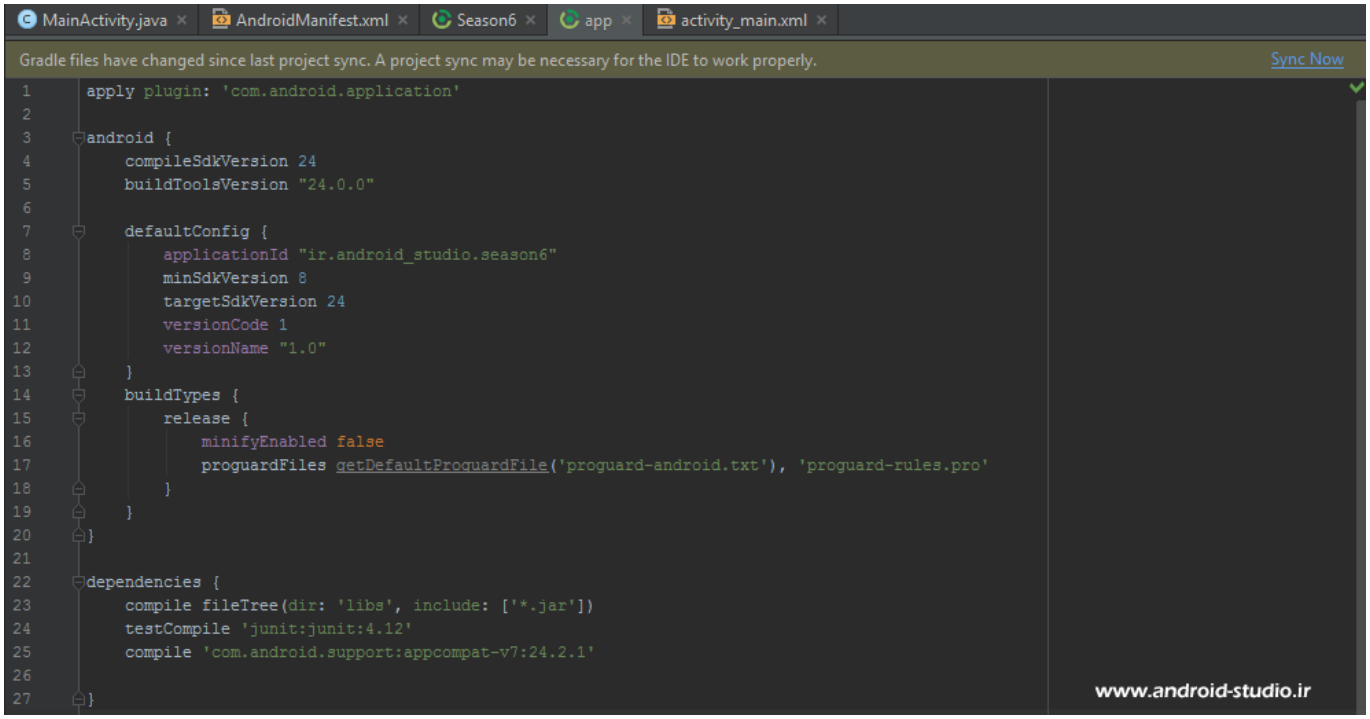
```
compile 'com.android.support:appcompat-v7:24.2.1'
```

را درون dependencies اضافه و سینک کنید.

MainActivity.java | AndroidManifest.xml | Season6 | app | activity_main.xml

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
    buildToolsVersion "24.0.0"

    defaultConfig {
        applicationId "ir.android_studio.season6"
        minSdkVersion 8
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:24.2.1'
}
```

www.android-studio.ir



پس از زدن گزینه سینک، اندروید استودیو شروع به دریافت کتابخانه از مخزن می کند. در ابتدا MavenCentral مخزن کتابخانه ها بود که پس از مدتی توسط گوگل به jcenter تغییر یافت و در حال حاضر اکثر توسعه دهندگان کتابخانه ها، کتابخانه خود را بر روی این سرویس منتشر می کنند.

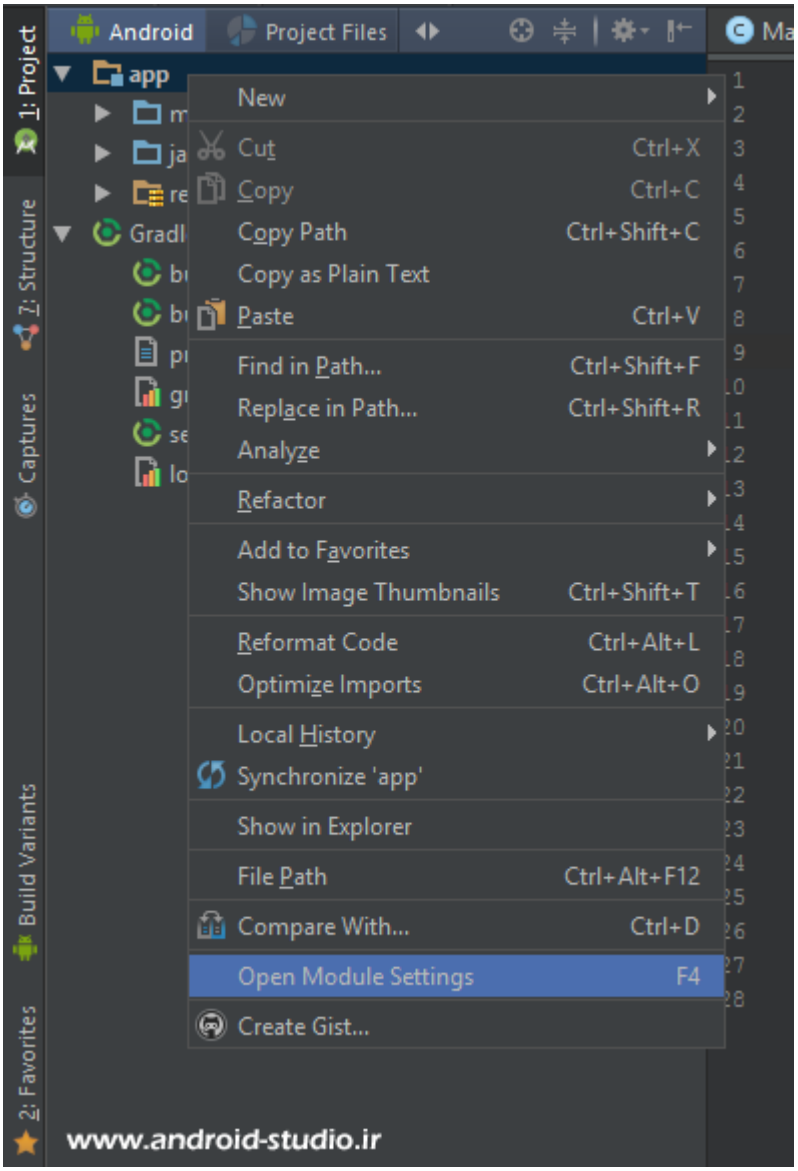
تذکر : در حال حاضر مخزن jcenter برای آی پی های ایران در دسترس نیست که نیاز به استفاده از ابزار تغییر آی پی دارید. اطلاعات بیشتر :

http://android-studio.ir/problems-and-questions

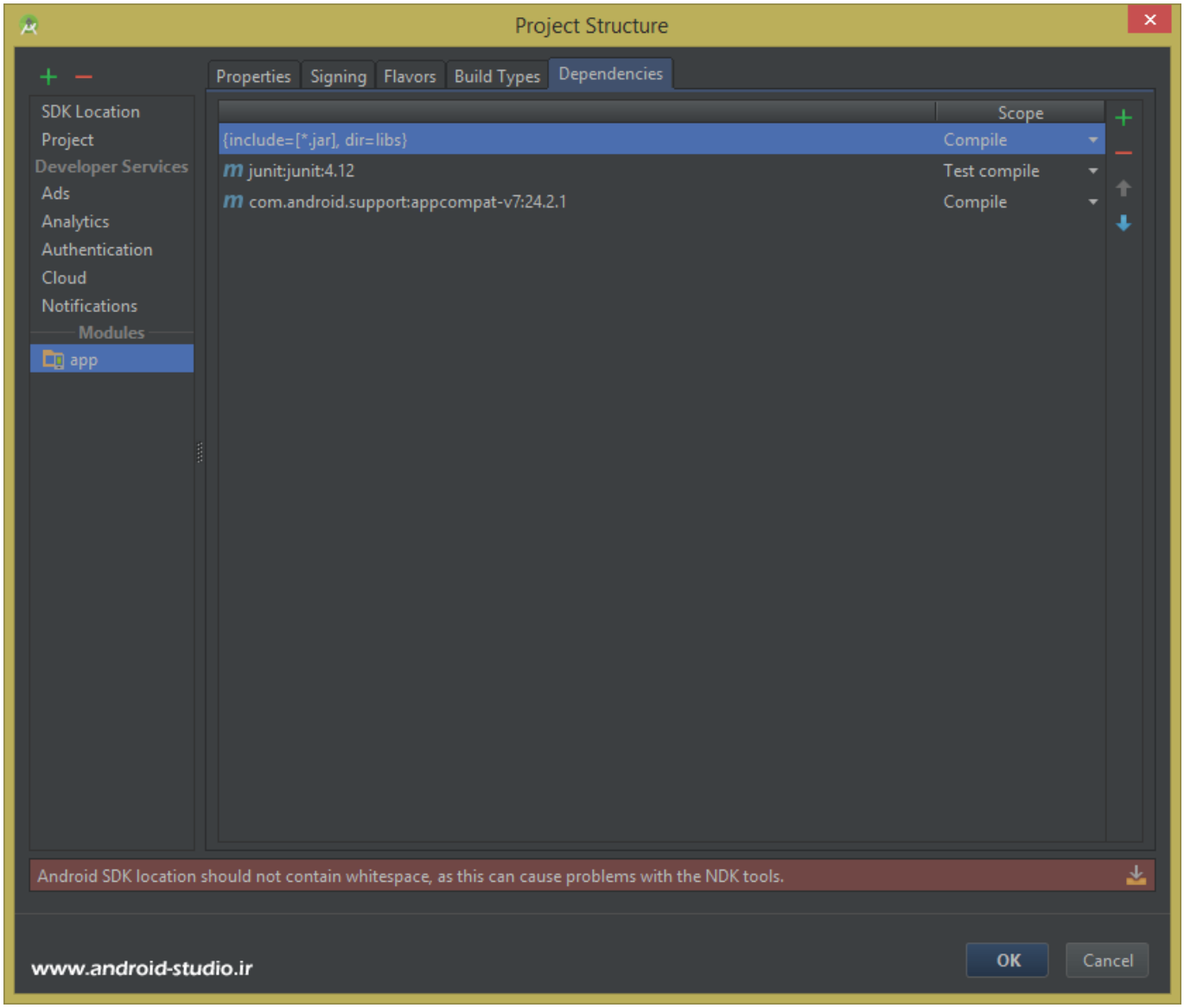
احتمالا این سوال برایتان بوجود آمده که کتابخانه چیست؟ کتابخانه را می توان مجموعه کدهایی دانست که در قالب یک بسته توسط شخص یا گروهی تهیه شده و در اختیار توسعه دهندگان قرار داده شده. از مزایای کتابخانه ها می توان به تسریع در توسعه اپلیکیشن اشاره کرد زیرا با وجود کتابخانه های آماده، در بسیاری از موارد توسعه دهنده نیازی به نوشتن کد موردنظر خود از پایه را ندارد و به سادگی می تواند از کتابخانه های مختلفی که در سطح اینترنت و بخصوص در http://github.com توسط توسعه دهندگان به صورت رایگان منتشر شده اند استفاده کند. علاوه بر اینکه باعث افزایش سرعت شما در توسعه اپلیکیشن می شود، از این جهت که این لایبرری به صورت عمومی منتشر شده و توسعه دهندگان زیادی از آن استفاده می کنند، مشکلات و به اصطلاح باگ های احتمالی موجود در مجموعه کدها به سرعت شناسایی و رفع می شود و در نهایت اپلیکیشن شما نیز باگ کمتری خواهد داشت. البته کتابخانه فقط در جاوا و اندروید کاربرد ندارد بلکه در همه زبانها و زمینه ها کتابخانه ها وجود دارند. به عنوان مثال در طراحی وب کتابخانه جاوا اسکریپت با نام jQuery هست که بسیار کاربردی بوده و در زمینه های مختلفی مانند اسلایدر و ... استفاده می شود (توجه داشته باشید جاوا اسکریپت و جاوا دو زبان کاملا متفاوت بوده و ارتباطی به یکدیگر ندارند. جاوا اسکریپت زبان سمت کاربر است). با مطالعه منابع مختلف معرفی کتابخانه های اندروید می توان به اکثر مواردی که برای توسعه اپ خود نیاز دارید دسترسی پیدا کنید.

۱ – ۲ : روش خودکار :

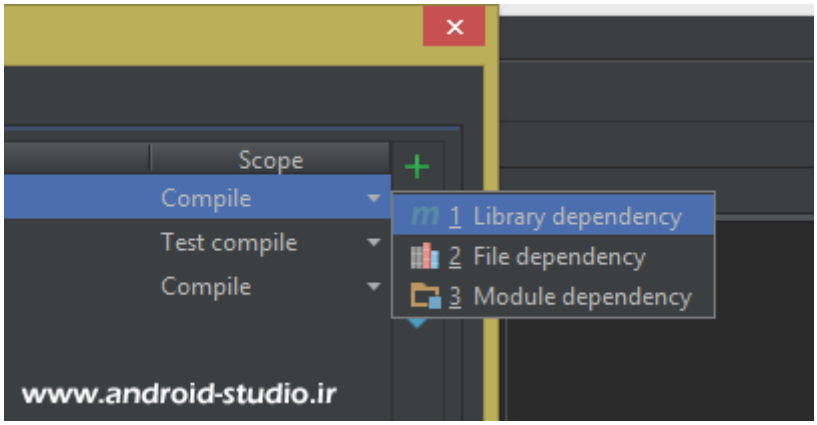
در این روش روی فولدر app پروژه راست کلیک و سپس Open Module Settings را انتخاب می کنیم (و یا از کلید میانبر F4 استفاده می کنیم) :



در پنجره جدید قسمت Module و گزینه app به صورت پیش فرض انتخاب شده (اگر نیست انتخاب کنید) و سپس به تب Dependencies بروید :

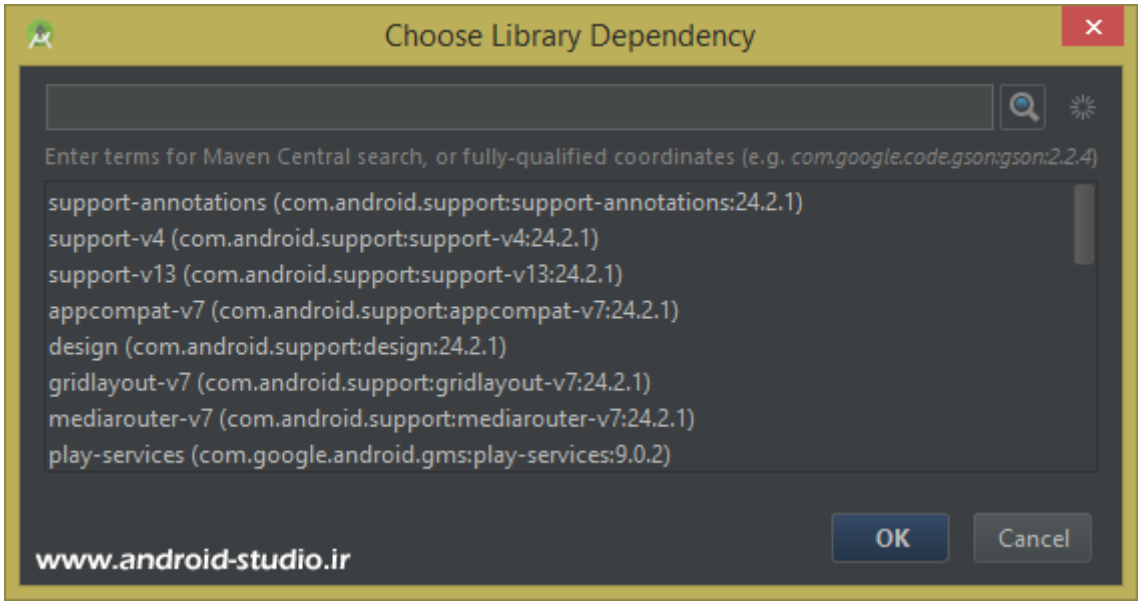


حال با کلیک روی گزینه + و انتخاب Library Dependency وارد صفحه بعدی می شویم :



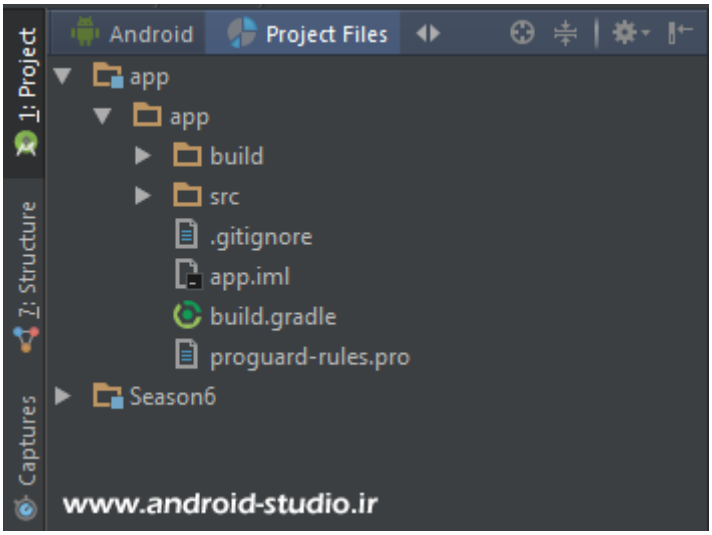
در کادر مربوطه نام کتابخانه را وارد می کنیم مانند :

com.android.support:appcompat-v7:24.2.1



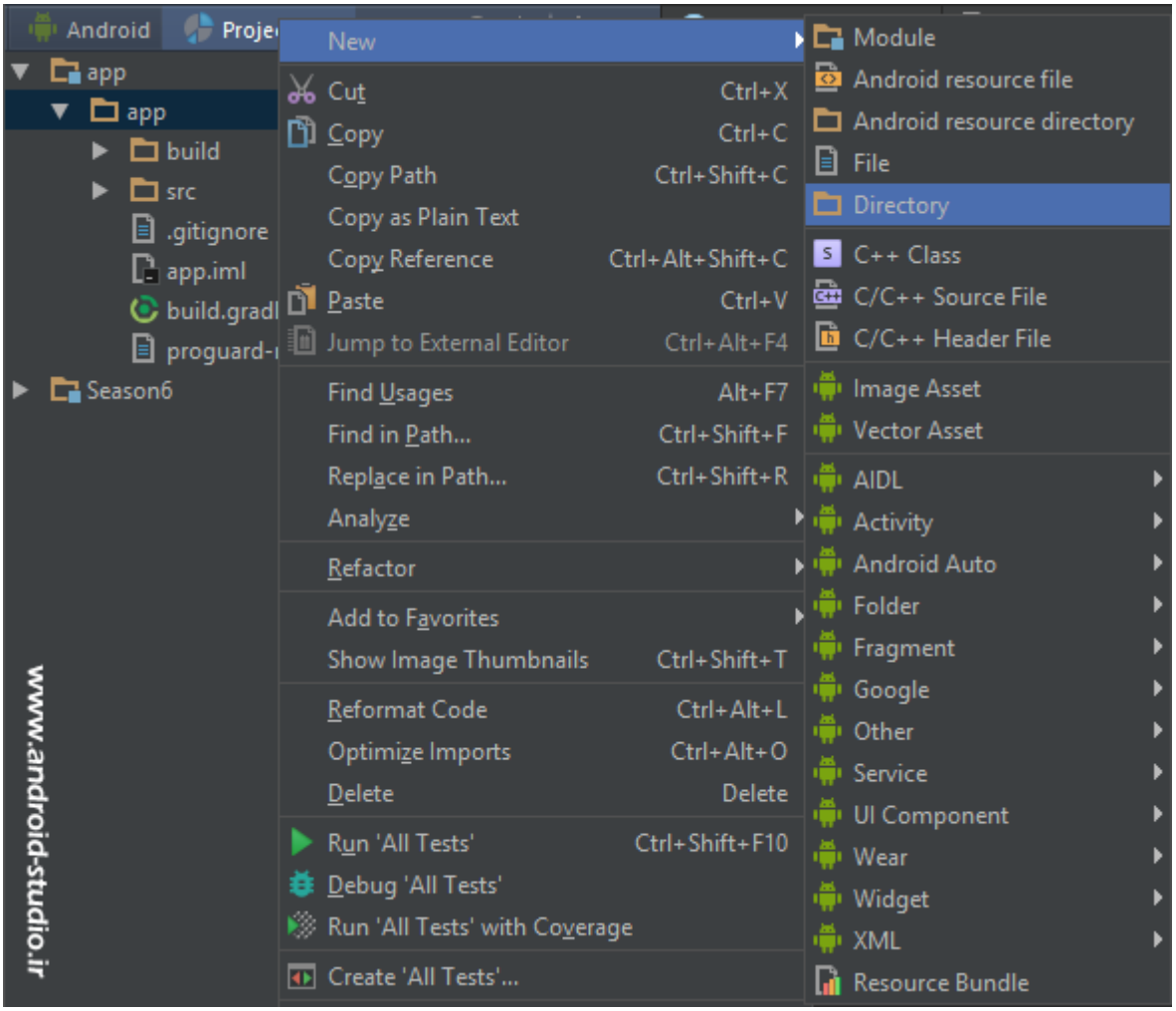
۲ – روش آفلاین :

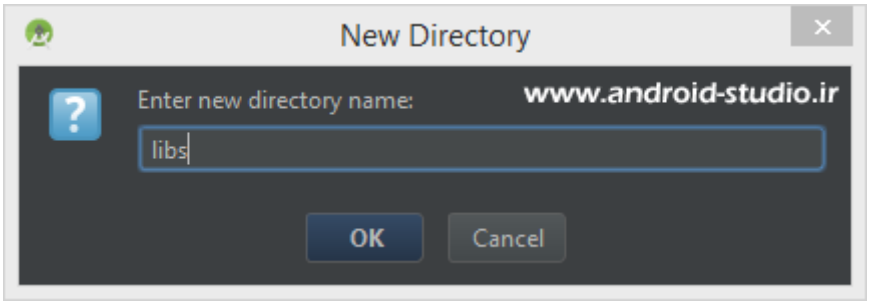
در این روش لازم است فایل کتابخانه را که قبلا با پسوند jar. دریافت کرده اید را به پروژه اضافه کنید.برای این کار ابتدا نحوه نمایش پروژه را از Android به Project Files تغییر می دهیم :

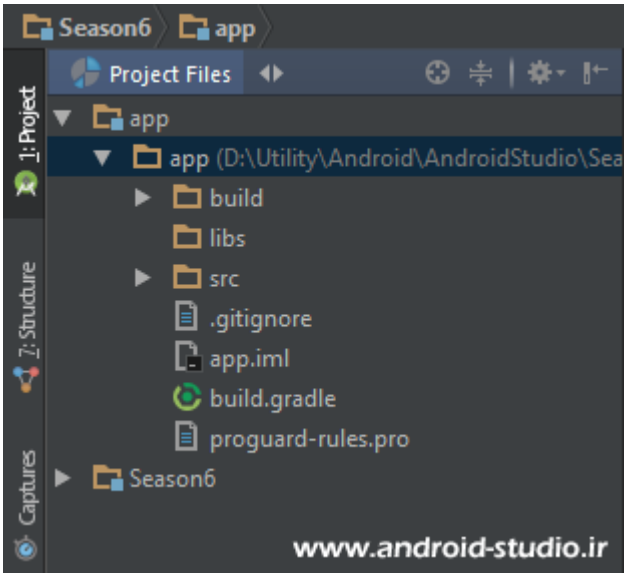


در کنار فولدرهای build و src باید فولدری با نام libs وجود داشته باشد که لازم است فایل کتابخانه با پسوند jar. را copy و سپس با راست کلیک روی این فولدر و گزینه Paste، آنرا به فولدر مخصوص

کتابخانه های پروژه خودمان منتقل کنیم. اگر این پوشه به صورت پیش فرض وجود نداشت روی فولدر app راست کلیک و از مسیر New > Directory یک مسیر جدید با عنوان libs ایجاد می کنیم :







مشاهده می کنیم فولدر libs مابین build و src اضافه شده است. بعد از انتقال فایل کتابخانه به این مسیر، به دو صورت می توان کتابخانه را به پروژه اضافه و از آن استفاده کرد.

در روش دستی کافیست buidl.gradle را باز نموده (بعد از انتقال فایل کتابخانه مجدد حالت نمایش ساختار پروژه را به Android سوئیچ نمایید) و مشابه حالت دستی آنلاین، توسط کدی که با compile شروع می شود، کتابخانه را به پروژه تعریف کنیم با این تفاوت که الان قصد داریم فایل را از مسیر libs فراخوانی کنیم نه مخزن آنلاین. پس به این صورت کد را به متد dependencies اضافه می کنیم :

```
compile files('libs/Library_file_name.jar')
```

واضح است که به جای Library_file_name.jar باید نام فایل کتابخانه موردنظر را جایگزین کنید مانند

```
compile files('libs/appcompat-v7.jar')
```

MainActivity.java | AndroidManifest.xml | Season6 | app | activity_main.xml

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

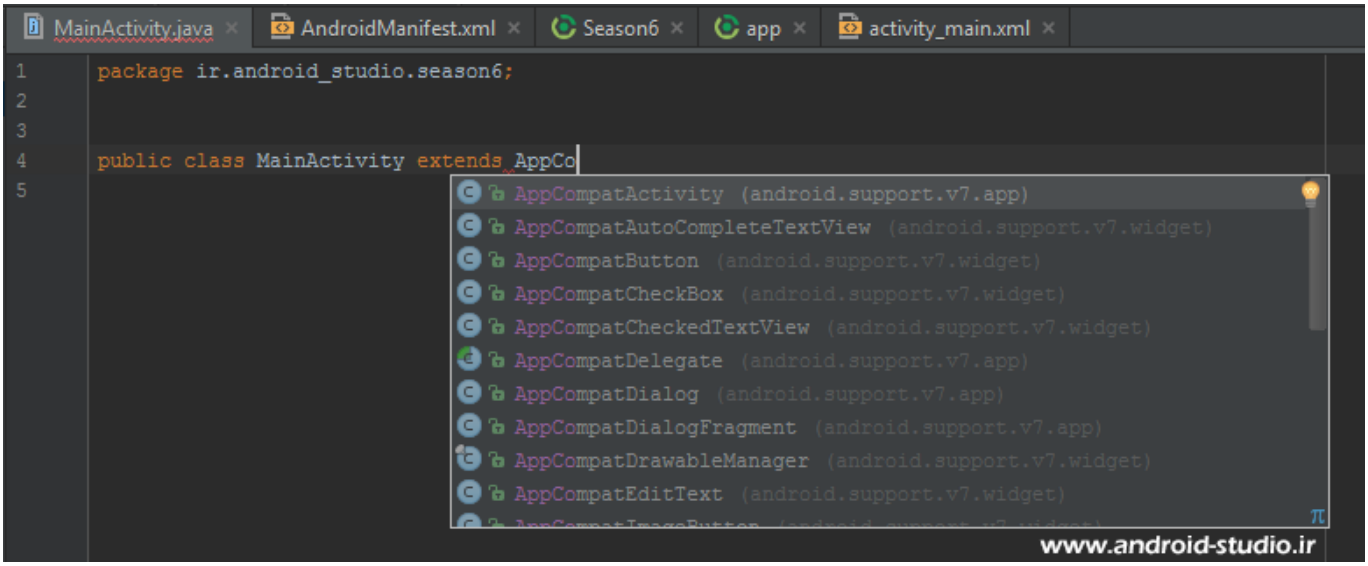
```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
    buildToolsVersion "24.0.0"

    defaultConfig {
        applicationId "ir.android_studio.season6"
        minSdkVersion 9
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:24.2.1'
    compile files('libs/Library_file_name.jar')
}
```

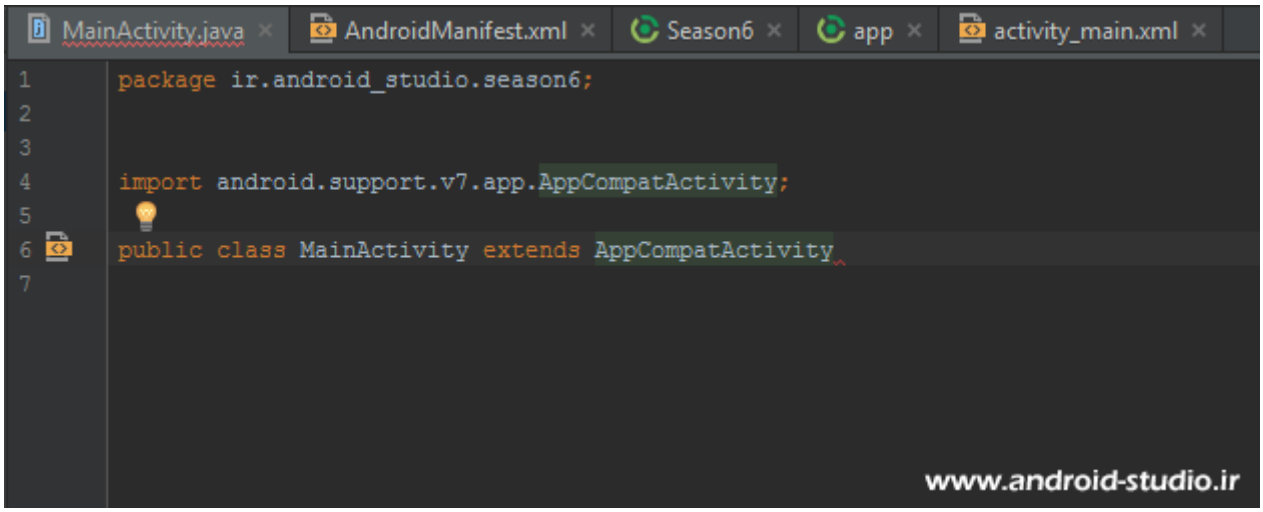
www.android-studio.ir

اما در روش خودکار پس از انتقال فایل به مسیر libs به جای وارد کردن کد در build.gradle ، مانند روش خودکار در حالت آنلاین، از مسیر Open Module Settings > app به تب dependencies رفته و پس از کلیک روی علامت + ، توسط گزینه File Dependencies به مسیرها از جمله libs دسترسی داریم که با باز کردن فولدر libs، کتابخانه (ها) یی که قبلا به این مسیر منتقل کرده ایم نمایش داده شده که قابل انتخاب کردن هستند.

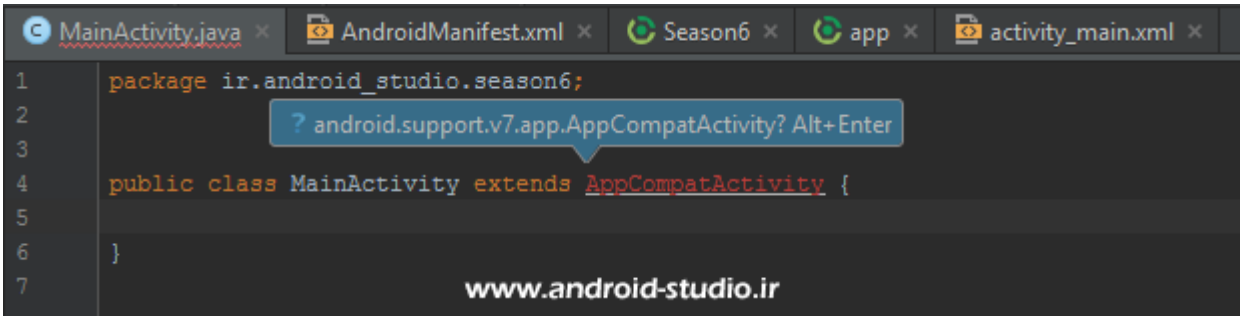
خب! برگردیم به بررسی ساختار اکتیویتی. خط چهارم کلاس android.os.Bundle ایمپورت شده که وظیفه آن، انتقال اطلاعات مابین اکتیویتی ها و ذخیره اطلاعات می باشد.

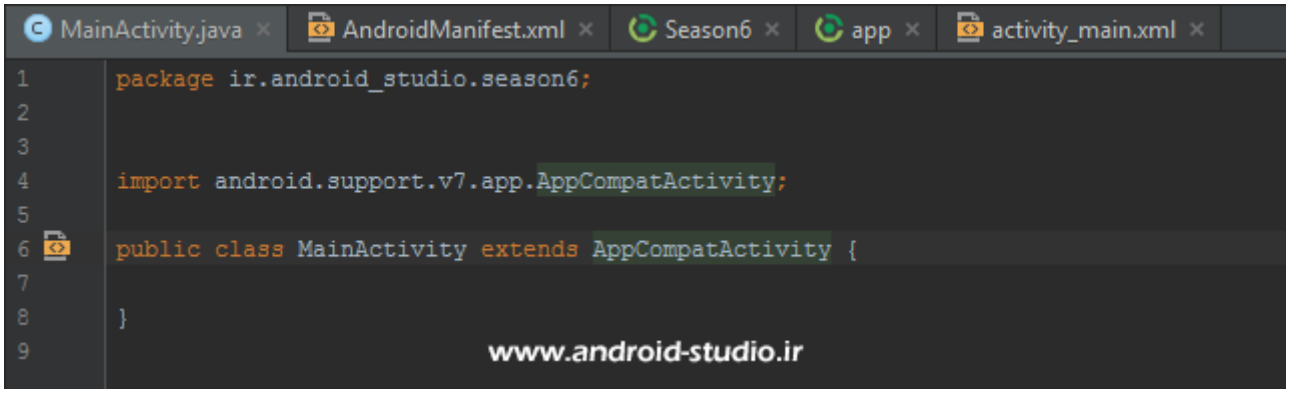
در خط ۶ کلاس اصلی اکتیویتی را داریم که به صورت پیش فرض توسط اندروید استودیو از کلاس AppCompatActivity ارث بری شده است. در واقع کلاسی که قبلا در خط ۳ ایمپورت شده بود در اینجا استفاده می شود. ایمپورت ها توسط اندروید استودیو به صورت خودکار انجام شده و نیاز به تایپ دستی آنها نیست. برای مثال اگر بخواهید کلاس MainActivity را حذف و مجدد خودتان بسازید و از AppCompatActivity ارث بری کنید، کلاس مورد نیاز به صورت خودکار ایمپورت می شود (به جز خط ۱ مابقی خطوط را پاک کردیم) :





مشاهده می شود پس از انتخاب AppCompatActivity از لیست انتخابها، خط ۴ بلافاصله اضافه شد. اگر لیست انتخاب باز نشد و کلاس را به صورت دستی تا انتها نوشتید و یا بعد از انتخاب از لیست باز هم عمل ایمپورت انجام نشد، پیغامی نمایش داده می شود که از شما می خواهد توسط دو کلید Alt+Enter انجام ایمپورت را تایید کنید:





این کلاس می تواند به جای ارث بری از AppCompatActivity ، از کلاس Activity نیز ارث بری شود اما استفاده از AppCompatActivity توصیه شده زیرا این کتابخانه برای دیزاین مدرن با نام متریال دیزاین (Material Design) بسیار کار توسعه دهنده را ساده کرده و خروجی کار یک اپلیکیشن با طراحی مدرن و ساده و استاندارد خواهد بود.اکثر اپلیکیشن های جدید به خصوص اپ های حرفه ای و معتبر، از این سبک طراحی پیروی می کنند (به عنوان نمونه اپلیکیشن Gmail. در آموزشهای بعدی با متریال دیزاین بیشتر آشنا می شویم.)

به طور کلی هرگاه یک شیئی جدید تعریف کنیم، کتابخانه مورد نیاز آن به صورت خودکار اضافه می شود که در مباحث آینده بیشتر آشنا خواهیم شد.

در خط ۶ متد onCreate را می بینیم. در هر اکتیویتی، همه چیز از این متد شروع می شود و تقریبا بیشتر کد مورد نیاز را باید داخل این متد بنویسیم و هر کاری که مدنظرمان باشد را می توانیم داخل این متد اجرا کنیم. در مقابل این متد، پارامتری با عنوان (Bundle savedInstanceState) اضافه شده که در واقع شیئی است با نام savedInstanceState ساخته شده از کلاس Bundle که قبلا به وظیفه این کلاس اشاره ای داشتیم. در خط بعد به وسیله دستور super اعلام می کنیم قصد داریم علاوه بر خصوصیاتی که توسط ما درون متد Override شده تعریف می شود، به کلاس اصلی که از آن ارث بری شده یعنی AppCompatActivity نیز دسترسی داشته باشیم.

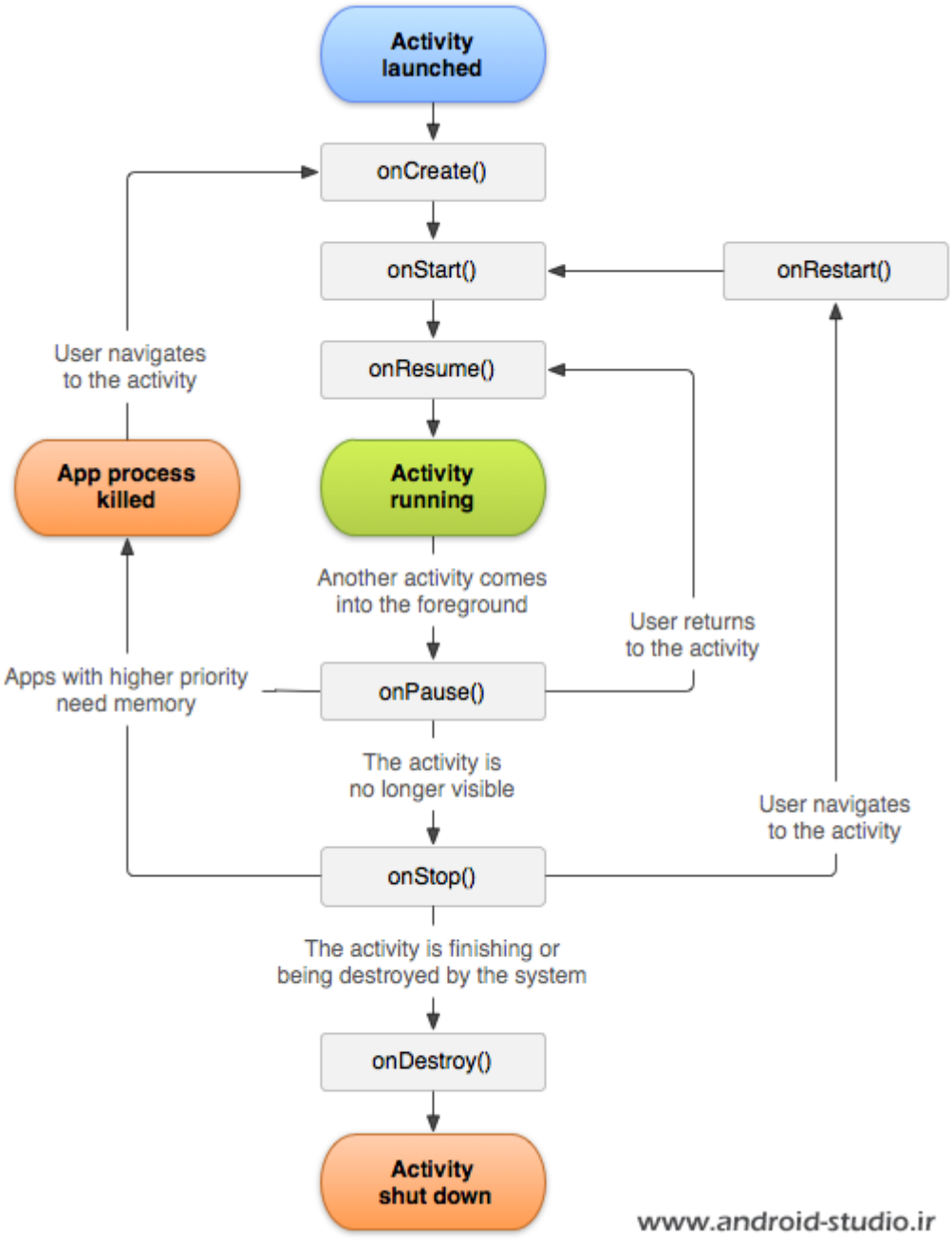
در خط ۱۱ متد ()setContentView وظیفه نمایش محتوای اکتیویتی برای کاربر را دارد که به صورت

R.layout.activity_main

تعریف شده. کاراکتر R حرف اول لغت Resource به معنی منبع بوده که برای استفاده از عناصر و فایلها درون کد جاوا در اندروید استفاده می شود. با اضافه کردن layout. به R اعلام می کنیم که قصد نمایش یک Layout را داریم و نهایتا با افزودن نام فایل xml صفحه مدنظر که در اینجا صفحه اصلی می باشد (بدون ذکر پسوند) ، activity_main.xml را به اکتیویتی MainActivity.java متصل می کنیم :

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

چرخه حیات اکتیویتی



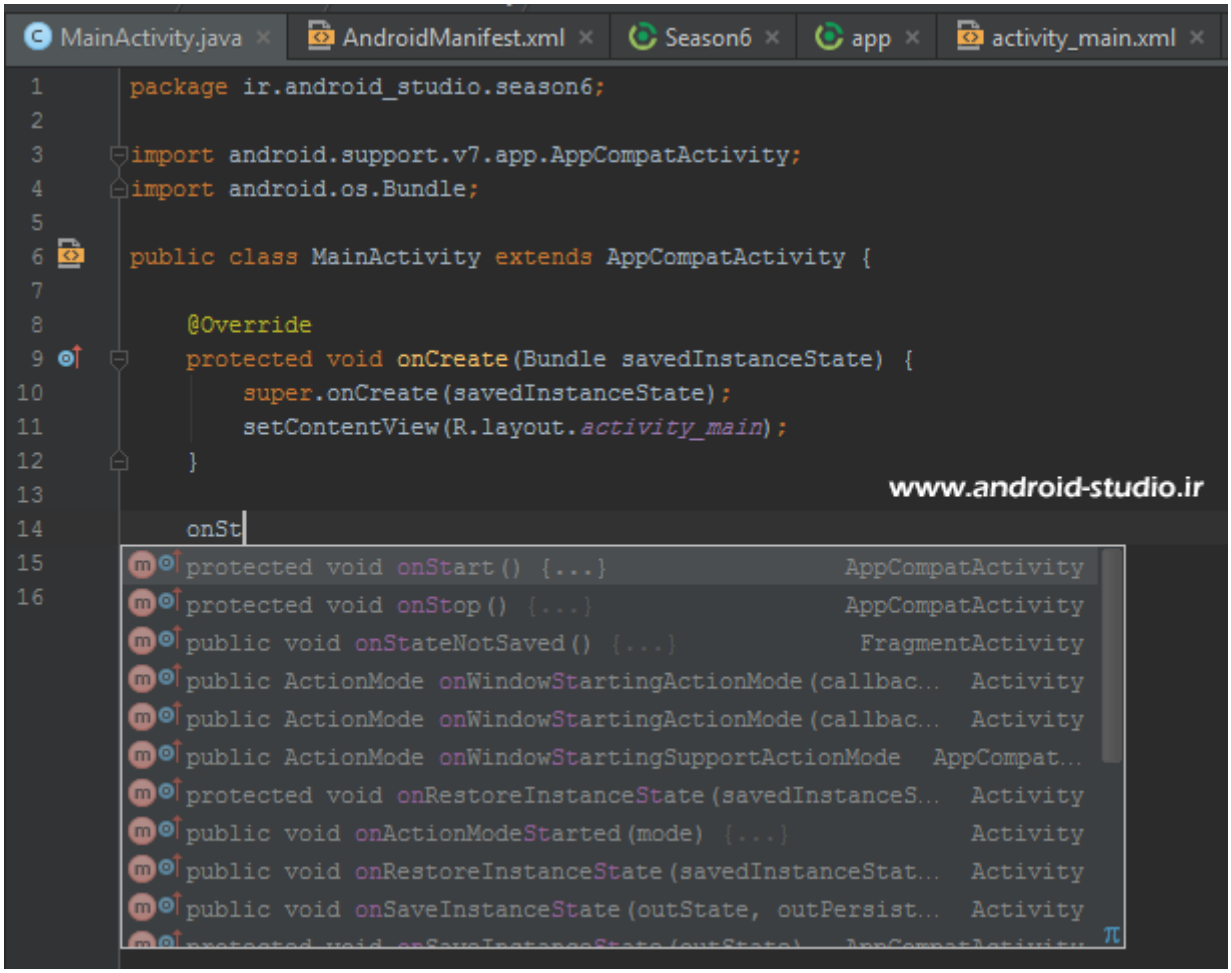
هر اکتیویتی شامل یک چرخه حیات بوده که وضعیت های مختلفی را شامل می شود. مانند زمانی که اکتیویتی برای اولین بار اجرا می شود، یا زمانی که کاربر از یک اکتیویتی به اکتیویتی دیگر می رود یا

هنگامی که به اکتیویتی باز می گردد. این چرخه حیات را کامل بررسی می کنیم. مجدد به MainActivity برمی گردیم.

MainActivity.java activity_main.xml

```
1  package ir.android_studio.season6;
2
3  import android.support.v7.app.AppCompatActivity;
4  import android.os.Bundle;
5
6  public class MainActivity extends AppCompatActivity {
7
8      @Override
9      protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_main);
12     }
13 }
14
```


متد onCreate را از قبل داریم. مابقی متد های اکتیویتی را به ترتیب زیر این متد اضافه و سپس با اجرای پروژه روی شبیه ساز، رفتار اکتیویتی را در logcat بررسی می کنیم تا به صورت عملی درک کنیم در هر زمان دقیقا در کدام مرحله از چرخه حیات قرار داریم. یادآوری می کنیم نیاز به نوشتن کامل کدها نیست و به عنوان مثال برای متد onStart تنها با نوشتن onSt لیست باز شده و با انتخاب اولین گزینه، متد به صورت کامل اضافه می شود:


MainActivity.java
AndroidManifest.xml
Season6
app
activity_main.xml
package ir.android_studio.season6;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
www.android-studio.ir
onSt
protected void onStart() {...} AppCompatActivity
protected void onStop() {...} AppCompatActivity
public void onStateNotSaved() {...} FragmentActivity
public ActionMode onWindowStartingActionMode(callbac... Activity
public ActionMode onWindowStartingActionMode(callbac... Activity
public ActionMode onWindowStartingSupportActionMode AppCompat...
protected void onRestoreInstanceState(savedInstanceS... Activity
public void onActionModeStarted(mode) {...} Activity
public void onRestoreInstanceState(savedInstanceStat... Activity
public void onSaveInstanceState(outState, outPersist... Activity

MainActivity.java | AndroidManifest.xml | Season6 | app | activity_main.xml

```java
package ir.android_studio.season6;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

    @Override
    protected void onStart() {
        super.onStart();
    }
}
```

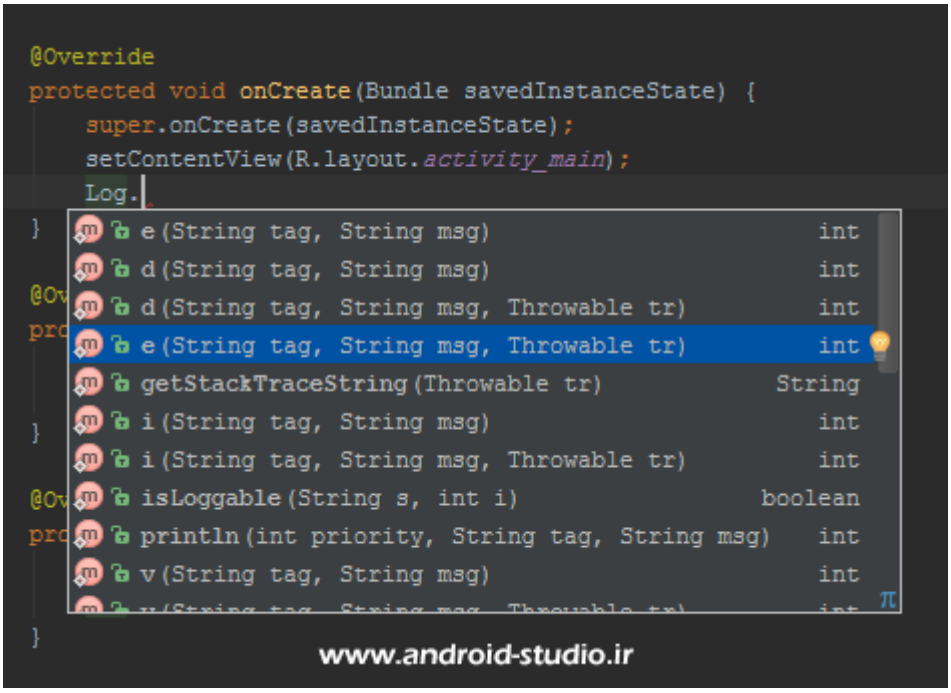

مابقی را نیز به همین صورت اضافه می کنیم :

```
package ir.android_studio.season6;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

    @Override
    protected void onStart() {
        super.onStart();
    }

    @Override
    protected void onResume() {
        super.onResume();
    }

    @Override
    protected void onPause() {
        super.onPause();
    }

    @Override
    protected void onStop() {
        super.onStop();
    }

    @Override
    protected void onRestart() {
        super.onRestart();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }


}
```

البته با کلید ترکیبی Alt + insert هم می توانید به متدها، Constructor ها و ... دسترسی داشته باشید:

اگر کیبورد شما کلید insert اختصاصی نداشته باشد (به عنوان مثال کلید insert لپ تاپ بنده با کلید عدد صفر ترکیب شده) احتمالا نیاز باشد alt + shift + insert را بگیرید. همانطور که در تصویر مشاهده می کنید برای دسترسی به Override Methods به صورت مستقیم از کلید Crtl + O هم می شود استفاده کرد.

برای اضافه کردن و چاپ لاگ دلخواه لازم است از دستور Log استفاده کنیم که می بایست درون هر متد این دستور را اضافه نماییم.



ما Log.e را انتخاب می کنیم (e مخفف error) تا لاگ های موردنظر ما با رنگ قرمز چاپ شده و متمایز باشد.در تصویر بالا مشاهده می کنید که برای لاگ یک Tag (برچسب) و یک متن پیام لازم داریم. به عنوان مثال می شود به صورت زیر کامل کرد :

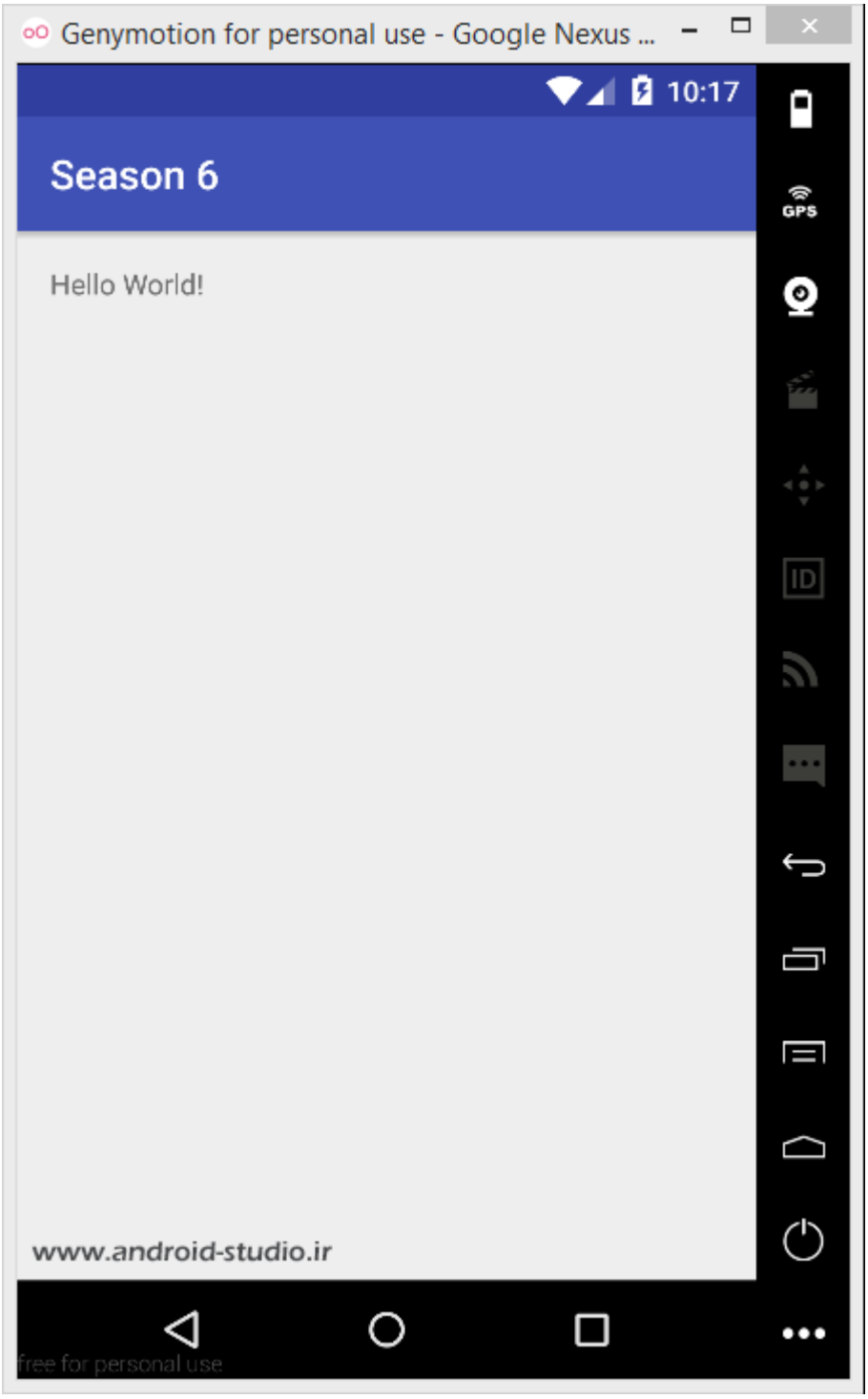
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.e("Activity method is " , "onCreate");
}
```

و این دستور را برای بقیه متد ها اضافه کنیم با این تفاوت که بخش دوم یعنی message را هم نام با آن متد قرار دهیم. اما بیایم حرفه ای تر عمل کنیم و بجای تکرار String تگ درون هر متد، یک بار String تگ را تعریف کرده و حجم کدمان را کاهش دهیم :

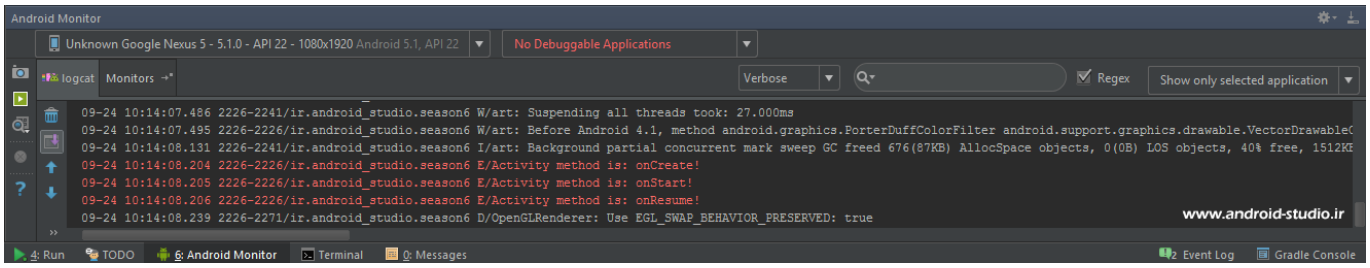
```
package ir.android_studio.season6;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;

public class MainActivity extends AppCompatActivity {

    private final String MyLog = "Activity method is ";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.e(MyLog , "onCreate!");
    }

    @Override
    protected void onStart() {
        super.onStart();
        Log.e(MyLog , "onStart!");
    }

    @Override
    protected void onResume() {
        super.onResume();
        Log.e(MyLog , "onResume!");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.e(MyLog , "onPause!");
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.e(MyLog , "onStop!");
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        Log.e(MyLog , "onRestart!");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.e(MyLog , "onDestroy!");
    }

}
```

استفاده از دستور لاگ به کتابخانه android.util.log نیاز داشت که با اضافه کردن اولین دستور Log، به صورت خودکار به بالای کد ایمپورت شد.

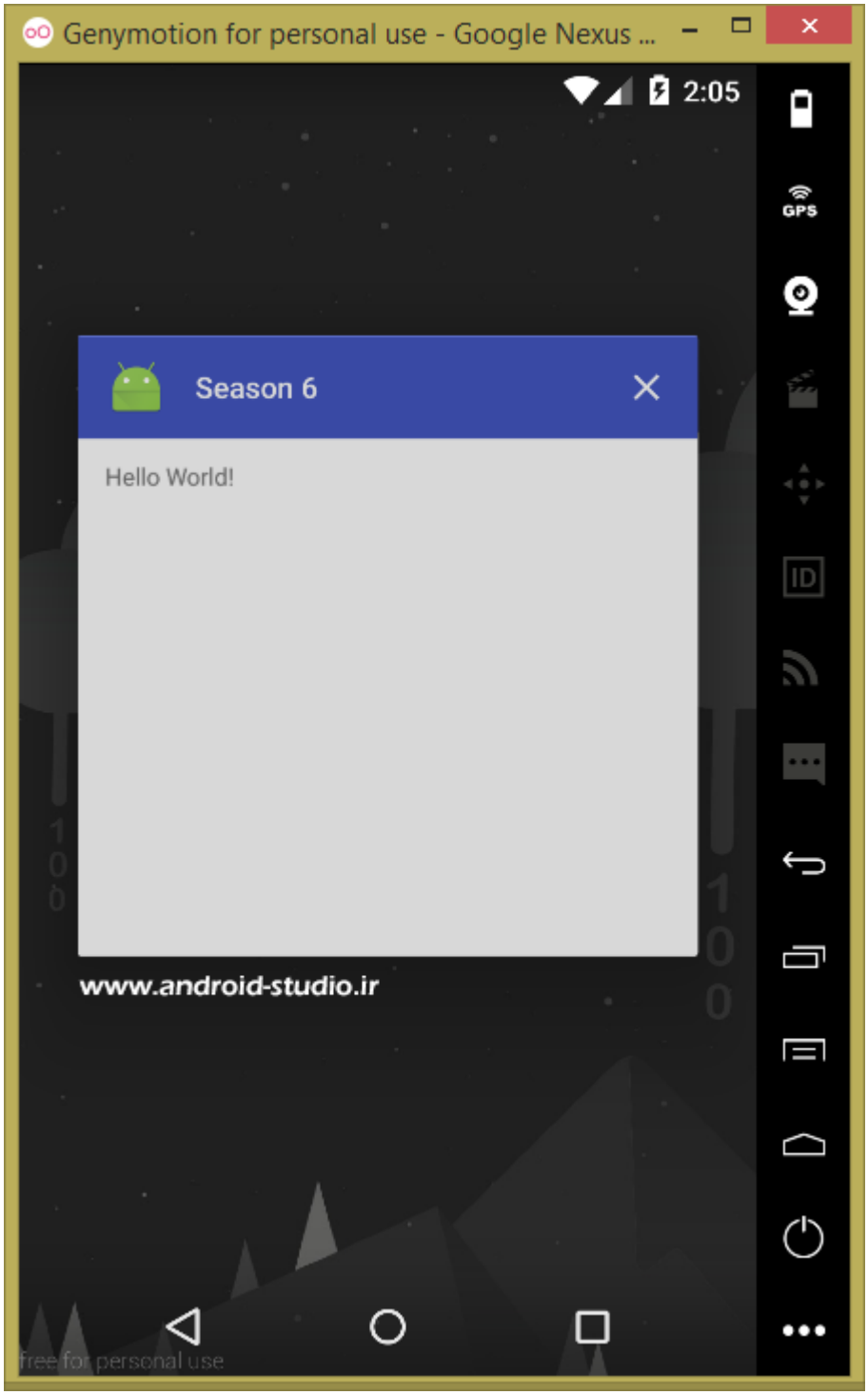
حالا پروژه را روی شبیه ساز جنی موشن Run می کنیم :



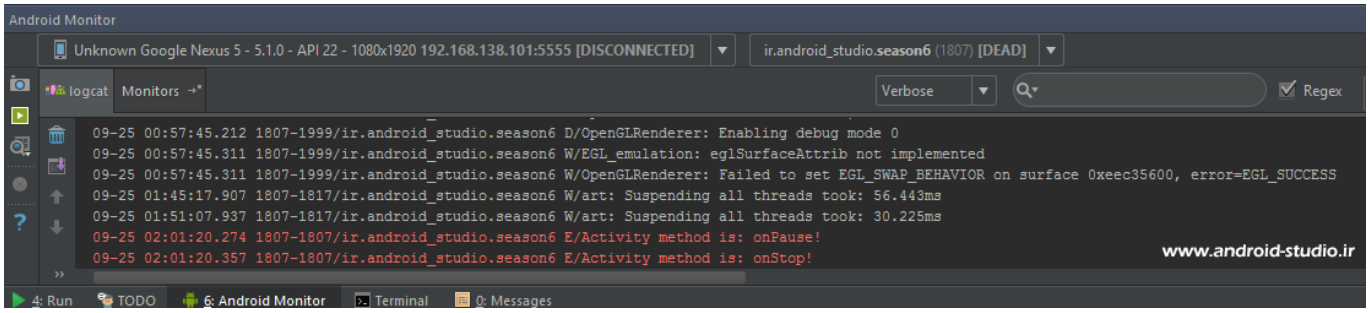
در پایین اندروید استودیو تب Android Monitor و قسمت logcat باز شده که در این قسمت تمامی اتفاقاتی که در جریان اجرا و کار با اپلیکیشن روی شبیه ساز Genymotion رخ می دهد را نشان می دهد. ما از دستور Log.e استفاده کرده بودیم پس انتظار می رود لاگ های مدنظرمان را با رنگ قرمز ببینیم :



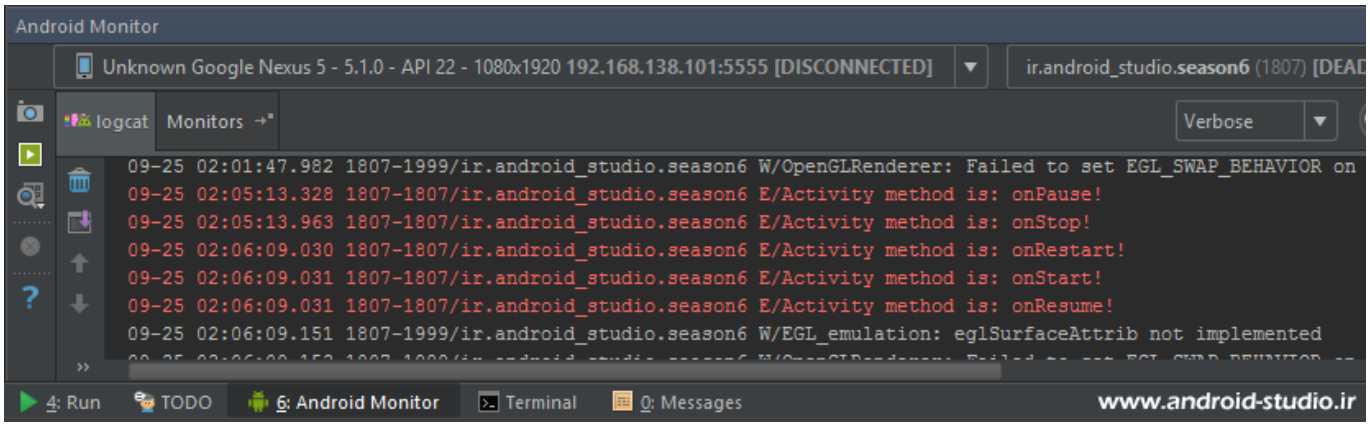
سه خط قرمز رنگ که با تگ تعریف شده ما یعنی Activity method is شروع شده بود مشاهده می شود. به ترتیب متد های onCreate ، onStart و onResume لیست شده. این یعنی برای نمایش اکتیویتی این سه متد اجرا می شود و با متد onResume اکتیویتی بر روی صفحه نمایش قرار می گیرد. حالا گزینه دسترسی به اپ های در حال اجرا (مربع) را کلیک می کنیم :


Genymotion for personal use - Google Nexus ...
2:05
GPS
Season 6
Hello World!
www.android-studio.ir
free for personal use

خروجی logcat :

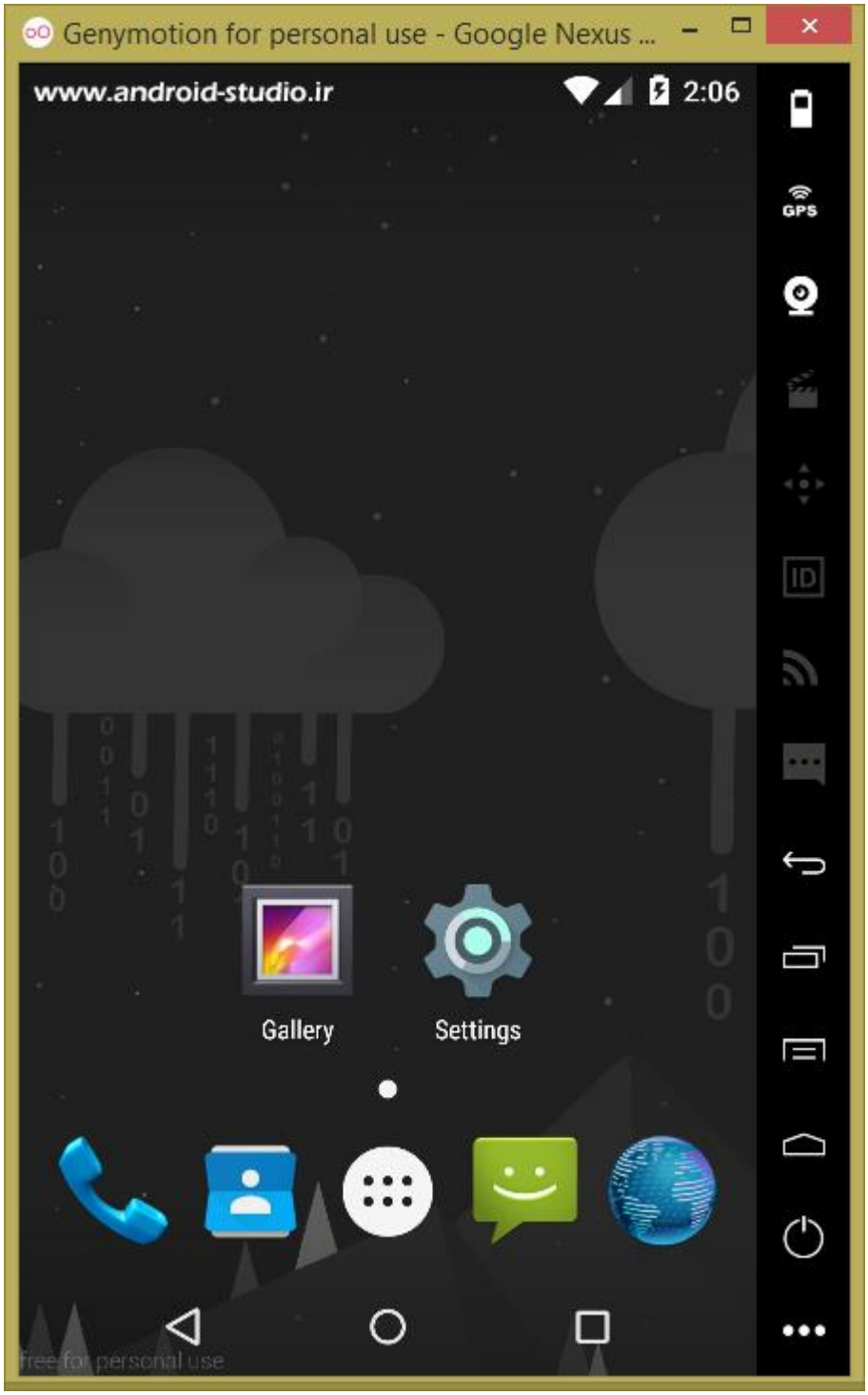


مشاهده می کنیم که اکتیویتی ابتدا onPause و سپس onStop شده. مجدد روی اکتیویتی کلیک میکنیم تا نمایش داده شود. خروجی :

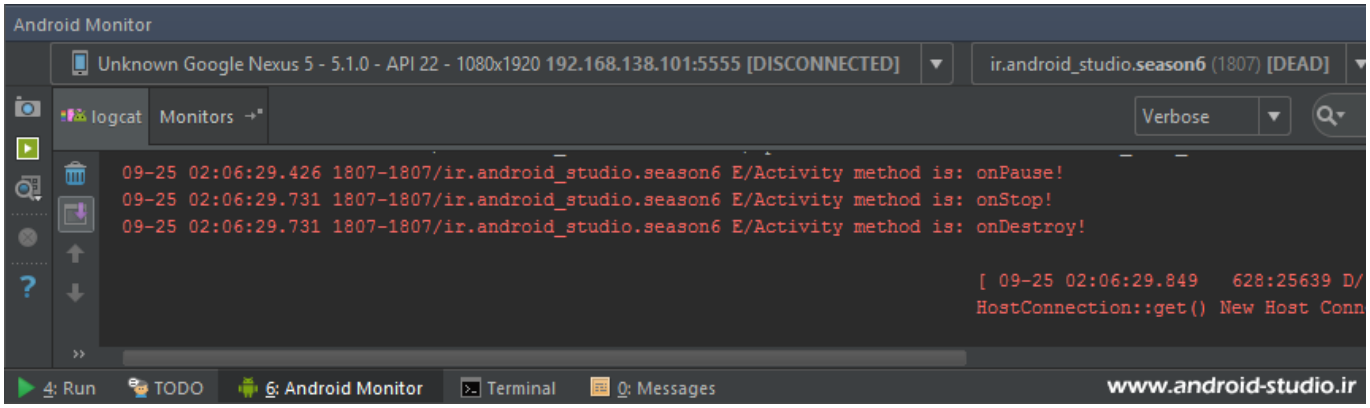


اکتیویتی ابتدا Restart شده (متد onRestart) سپس مانند مرحله اول پس از onStart و onResume روی صفحه نمایش قرار می گیرد.

در مرحله آخر از اکتیویتی خارج می شویم :


Genymotion for personal use - Google Nexus ...
www.android-studio.ir
2:06
GPS
Gallery
Settings
free for personal use

خروجی logcat :



اکتیویتی ابتدا Pause شده ، سپس Stop و در نهایت با متد onDestroy اکتیویتی اصطلاحا Shut Down می شود.

امیدواریم از این آموزش بهره کافی برده باشید. خوشحال می شویم انتقادات و پیشنهادات خود را در پست مربوط به همین آموزش در وب سایت ما به نشانی www.android-studio.ir عنوان فرمایید تا در تهیه آموزش های بعدی لحاظ شود.