数据类

```
class Link(var id: String,
           var length: Double,
           var name: String,
           var attributes: String)
```

数据类

```
data class Link(var id: String,
                var length: Double,
                var name: String,
                var attributes: String)
```

数据类

```
data class Link(var id: String,
                var length: Double,
                var name: String,
                var attributes: String)
```

```
val link = Link("1", 326.0, "苏州街", "01|13")

val anotherInstanceOfThisLink = link.copy()
```

数据类

```
data class Link(var id: String,
                var length: Double,
                var name: String,
                var attributes: String)
```

```
val link = Link("1", 326.0, "苏州街", "01|13")
val anotherInstanceOfThisLink = link.copy()
```

```
public final Link copy(@NotNull String id,
        double length, @NotNull String name, @NotNull String attributes) {
  return new Link(id, length, name, attributes);
}
```

数据类

```
data class Link(var id: String,
                var length: Double,
                var name: String,
                var attributes: String)
```

```
val link = Link("1", 326.0, "苏州街", "01|13")
val anotherInstanceOfThisLink = link.copy()
```

id: String = ..., length: Double = ..., name: String = ..., attributes: String = ...

数据类

```
data class Link(var id: String,
                var length: Double,
                var name: String,
                var attributes: String)
```

```
val link = Link("1", 326.0, "苏州街", "01|13")
val anotherInstanceOfThisLink = link.copy(name = "彩和坊路")
```

数据类

```
data class Link(var id: String,
                var length: Double,
                var name: String,
                var attributes: String)
```

```
val link = Link("1", 326.0, "苏州街", "01|13")
val anotherInstanceOfThisLink = link.copy(name = "彩和坊路")
val (id, length, name, attributes) = link
```

数据类

```
data class Link(var id: String,
                var length: Double,
                var name: String,
                var attributes: String)
```

```
val link = Link("1", 326.0, "苏州街", "01|13")
val anotherInstanceOfThisLink = link.copy(name = "彩和坊路")
val (id, length, name, attributes) = link
```

数据类

```
data class Pair<out A, out B>(
    public val first: A,
    public val second: B) : Serializable
```

数据类

```
data class Triple<out A, out B, out C>(
    public val first: A,
    public val second: B,
    public val third: C) : Serializable
```

数据类

```
fun returnPair(): Pair<Double, Double>{
    return 2.0 to 3.0
}
```

数据类

```
fun returnPair(): Pair<Double, Double>{
    return 2.0.to(3.0)
}
```

数据类

```
fun returnPair(): Pair<Double, Double>{
    return 2.0.to(3.0)
}
```

```
val (first, second) = returnPair()
```

类型别名

```
public class LatLng implements Parcelable {
    public final double latitude;
    public final double longitude;
     ...
}
```

类型别名

```
data class Point(val latitude: Double, val longitude: Double)
```

类型别名

```
//data class Point(val latitude: Double, val longitude: Double)

typealias Point = LatLng
```

定义“静态成员”

```
class Statics{

}
```

定义“静态成员”

```
class Statics{
    companion object {


    }
}
```

定义“静态成员”

```
class Statics{
    companion object {
        fun notStaticFun(){
            println("I'm not a static method! I mean it!")
        }
    }
}
```

定义“静态成员”

```
class Statics{
    companion object {
        fun notStaticFun(){
            println("I'm not a static method! I mean it!")
        }
        val `me2!!`: String = "Me either!"
    }
}
```

定义“静态成员”

```
Statics.notStaticFun()
println(Statics.`me2!!`)
```

定义“静态成员”

```
fun main(args: Array<String>) {
    Statics.notStaticFun()
    println(Statics.`me2!!`)
}
```

函数是“一等公民”

```
@Override
public void onSucess() {
    listView.post(new Runnable() {
        @Override
        public void run() {
            adapter.notifyDataSetChanged();
        }
    });
}
```

函数是“一等公民”

```
override fun onSucess() {
    listView.post(object: Runnable{
        override fun run() {
            adapter.notifyDataSetChanged()
        }
    } )
}
```

函数是“一等公民”

```
override fun onSucess() {
    listView.post( Runnable { adapter.notifyDataSetChanged() } )
}
```

函数是“一等公民”

```
override fun onSucess() {
    listView.post( { adapter.notifyDataSetChanged() } )
}
```

函数是“一等公民”

```
override fun onSucess() {
    listView.post{ adapter . notifyDataSetChanged() }
}
```

函数是“一等公民”

```
override fun onSucess() {
    listView.post( adapter :: notifyDataSetChanged )
}
```

函数是“一等公民”

```
override fun onSucess() {
    listView.post( adapter :: notifyDataSetChanged )
}
```

action: Runnable!
action: (() -> Unit)!

函数是“一等公民”

```
public void notifyDataSetChanged()
```

函数是“一等公民”

```
public void notifyDataSetChanged()
```

函数是“一等公民”

```
public void notifyDataSetChanged()
```

Lambda

```
val aLambda = {
    println("I am in a lambda!")
}
```

Lambda

```
val aLambda = {
    println("I am in a lambda!")
}
```

```
aLambda()
```

```
I am in a lambda!

Process finished with exit code 0
```

Lambda

```
val aLambda = {
    println("I am in a lambda!")
}
```

```
aLambda.invoke()
```

```
I am in a lambda!

Process finished with exit code 0
```

Lambda

```
val aLambda = {
    println("I am in a lambda!")
}
```

Lambda

() → Unit

```
val aLambda = {
    println("I am in a lambda!")
}
```

Lambda

(Int, Int) → Int

```
val aLambda = { left: Int, right: Int ->
    left * right
}
```

Lambda

```
val aLambda = { left: Int, right: Int ->
    left * right
}
```

```
aLambda(2, 3)
```

Lambda

```
val aLambda = { left: Int, right: Int ->
    left * right
}
```

```
aLambda(2, 3).let(::println)
```

```
6

Process finished with exit code 0
```

高阶函数

```
inline fun <T, R> T.let(block: (T) -> R): R = block(this)
```

高阶函数

```
inline fun <T, R> T.let(block: (T) -> R): R = block(this)
```

```
"Hello World".let(::println)
```

高阶函数

```
inline fun <T, R> T.let(block: (T) -> R): R = block(this)
```

```
"Hello World".let(::println)
```

高阶函数

```
inline fun <T, R> Array<out T>.map(transform: (T) -> R): List<R>
inline fun <T, R> Iterable<T>.map(transform: (T) -> R): List<R>
```

高阶函数

```
inline fun <T, R> Array<out T>.map(transform: (T) -> R): List<R>
inline fun <T, R> Iterable<T>.map(transform: (T) -> R): List<R>
```

```
Array(10){ it }.map { it * 2 }.joinToString().let(::println)
List(11){ it }
    .map { Math.abs(it - 5) }
    .filter { it != 0 }
    .map { "* " * it }.joinToString("\n").let(::println)
```

高阶函数

```
inline fun <T, R> Array<out T>.map(transform: (T) -> R): List<R>
inline fun <T, R> Iterable<T>.map(transform: (T) -> R): List<R>
```

```
Array(10){ it }.map { it * 2 }.joinToString().let(::println)
List(11){ it }
    .map { Math.abs(it - 5) }
    .filter { it != 0 }
    .map { "* " * it }.joinToString("\n").let(::println)
```

```
0, 2, 4, 6, 8, 10, 12, 14, 16, 18
* * * * *
* * * *
* * *
* *
*
*
* *
* * *
* * * *
* * * * *

Process finished with exit code 0
```

高阶函数

```
operator fun String.times(other: Int): String{
    return (1..other).fold(StringBuilder()){ acc, i ->
        acc.append(this)
        acc
    }.toString()
}
```

空类型安全

```
File someDir = ...;
for (File file : someDir.listFiles()) {
    System.out.println(file.getName());
}
```

空类型安全

```
File someDir = new File("IAmNotADir");
for (File file : someDir.listFiles()) {
    System.out.println(file.getName());
}
```

空类型安全

```
File someDir = new File("IAmNotADir");
for (File file : someDir.listFiles()) {
```

Dereference of 'someDir.listFiles()' may produce 'java.lang.NullPointerException' more... (⌘F1)

```
}
```

空类型安全

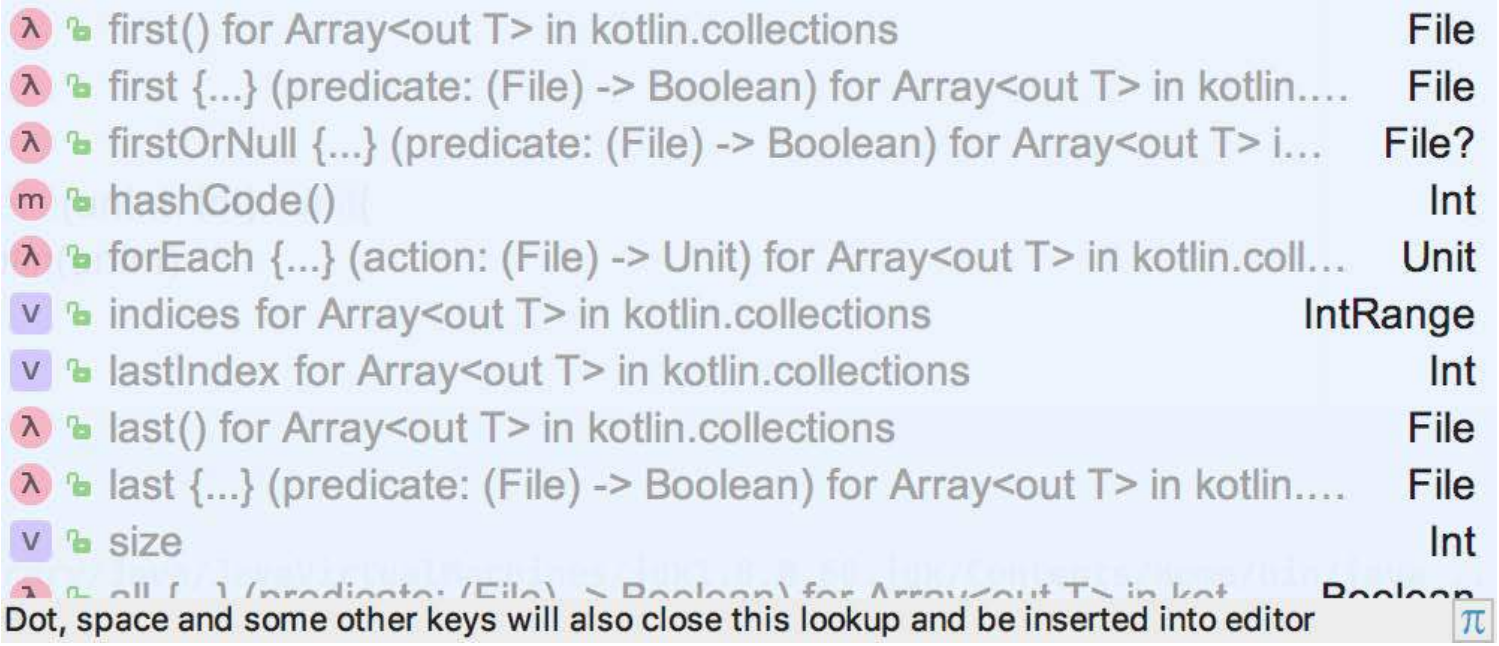
```
val someDir = File("IAmNotADir")
```

空类型安全

```
val someDir = File("IAmNotADir")

val subFiles: Array<File>? = someDir.listFiles()
```

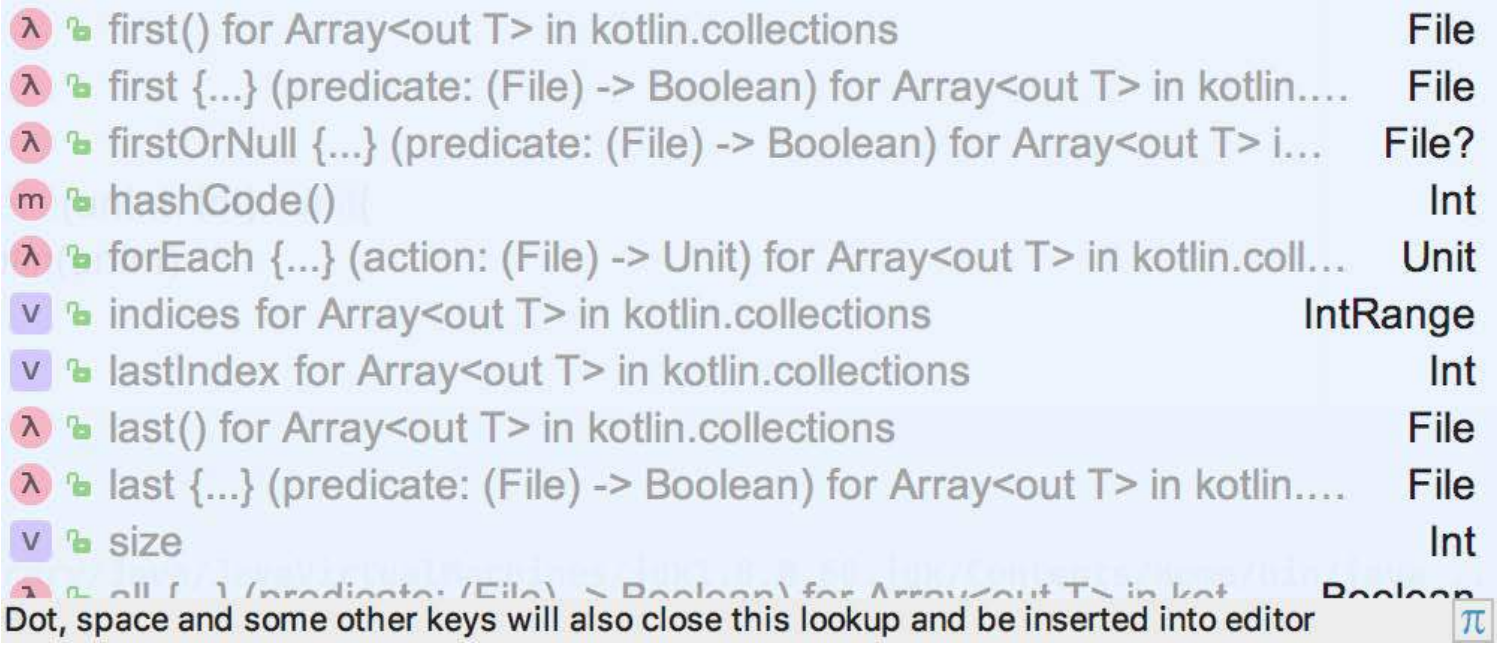
空类型安全

```
val someDir = File("IAmNotADir")

val subFiles: Array<File>? = someDir.listFiles()

subFiles.
```



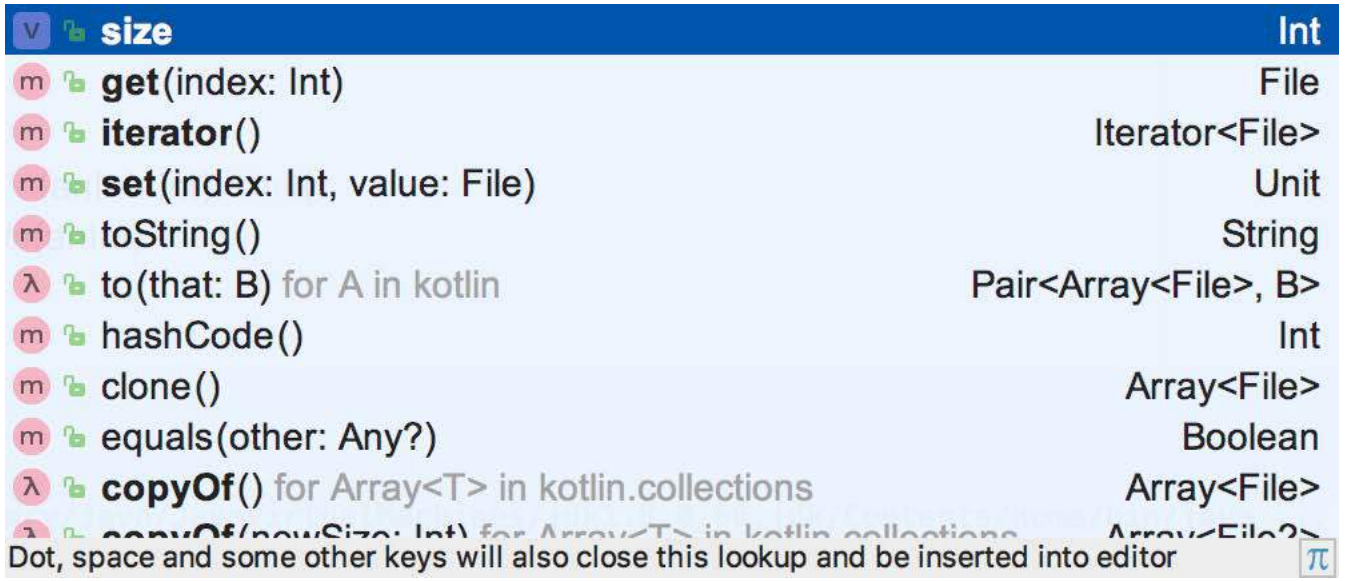
空类型安全

```
val someDir = File("IAmNotADir")

val subFiles: Array<File>? = someDir.listFiles()

subFiles.
```



空类型安全

```
val someDir = File("IAmNotADir")

val subFiles: Array<File>? = someDir.listFiles()

subFiles?.
```



类型智能转换

```
val someDir = File("IAmNotADir")

val subFiles: Array<File>? = someDir.listFiles()

if(subFiles != null){
    subFiles.
}
```

size	Int
get(index: Int)	File
iterator()	Iterator<File>
set(index: Int, value: File)	Unit
toString()	String
to(that: B) for A in kotlin	Pair<Array<File>, B>
hashCode()	Int
clone()	Array<File>
equals(other: Any?)	Boolean
copyOf() for Array<T> in kotlin.collections	Array<File>
copyOf(newSize: Int) for Array<T> in kotlin.collections	Array<File?>

Dot, space and some other keys will also close this lookup and be inserted into editor

类型智能转换

```
val someDir = File("IAmNotADir")

val subFiles: Array<File>? = someDir.listFiles()

if(subFiles != null){
    subFiles.
}
```

智能转换为
Array<File>类型

size	Int
get(index: Int)	File
iterator()	Iterator<File>
set(index: Int, value: File)	Unit
toString()	String
to(that: B) for A in kotlin	Pair<Array<File>, B>
hashCode()	Int
clone()	Array<File>
equals(other: Any?)	Boolean
copyOf() for Array<T> in kotlin.collections	Array<File>

Dot, space and some other keys will also close this lookup and be inserted into editor

类型智能转换

```
View view = ...;
TextView textView = ...;
if(view instanceof ViewGroup){
    ((ViewGroup) view).addView(textView);
}
```

类型智能转换

```
val view: View = ...
val textView = ...
if(view is ViewGroup) {
    view.addView(textView)
}
```

类型安全转换

```
val view: View = ...
val textView = ...
(view as? ViewGroup)?.addView(textView)
```

尝试转换，失败就返回null

类型安全转换

```
val view: View = ...
ViewGroup? tView = ...
(view as? ViewGroup)?.addView(textView)
```

类型安全转换

```
val view: View = ...
ViewGroup? tView = ...
(view as? ViewGroup)?.addView(textView)
```

类型转换

```
val view: View = RelativeLayout(this)
val textView = ...
 (view as? ViewGroup)?.addView(textView)
```

类型转换

```
val view: View = RelativeLayout(this)
val textView = ...
 (view as ViewGroup).addView(textView)
```

Elvis 运算符

```
val view: View = ...
ViewGroup? tView = ...
(view as? ViewGroup)?.addView(textView)
```

Elvis 运算符

```
val view: View = ...
Unit? textView = ...
(view as? ViewGroup)?.addView(textView)
```

Elvis 运算符

```
val view: View = ...
val textView = ...
if(view is ViewGroup) {
    view.addView(textView)
} else {
    ...
}
```

Elvis 运算符

```
val view: View = ...
val textView = ...
 (view as? ViewGroup)?.addView(textView)
```

Elvis 运算符

```
val view: View = ...
val textView = ...
(view as? ViewGroup)?.addView(textView) ?: toast("对不起，我们尽力了")
```

let … "else" …

```
(view as? ViewGroup)?.let {
    it.addView(view)
}?: run {
    toast("对不起，我们尽力了")
}
```

let … "else" …

```
(view as? ViewGroup)?.let {
    it.addView(view)
}?: run {
    toast("对不起，我们尽力了")
}
```

let ... ~~"else" ...~~

```
(view as? ViewGroup)?.let {
    it.addView(view)
}?. run {
    toast("对不起，我们尽力了")
}
```

let … ~~"else" …~~

```
(view as? ViewGroup)?.let {
    it.addView(view)
}?. run {
    toast("很好，我们添加了一个 View")
}
```

泛型

```
public interface List<out E> : Collection<E> {
    ...
}
```

泛型

```
public interface List<out E> : Collection<E> {
    ...
}
```

泛型

```
public interface Comparable<in T> {
    ...
}
```

Reified

```
String json = ...;
Gson gson = ...;
Link link = gson.fromJson(json, Link.class);
```

Reified

```
String json = ...;
Gson gson = ...;
Link link = gson.fromJson(json, Link.class);
```

Reified

```
String json = ...;
Gson gson = ...;
Link link = gson.fromJson(json, Link.class);
```

Reified

```
public <T> T fromJson(String json){
    return gson.fromJson(json, T.class);
}
```

Reified

```
public <T> T fromJson(String json){
    return gson.fromJson(json, T.class);
}
```

Cannot select from a type variable

Reified

```
fun <T> Gson.fromJson(json: String)
    = fromJson(json, T::class.java)
```

Cannot use 'T' as reified type parameter. Use a class instead.

Reified

```
inline fun <reified T> Gson.fromJson(json: String)
    = fromJson(json, T::class.java)
```

Reified

```
inline fun <reified T> Gson.fromJson(json: String)
    = fromJson(json, T::class.java)
```

Reified

```
inline fun <reified T> Gson.fromJson(json: String)
    = fromJson(json, T::class.java)
```

Reified

```
inline fun <reified T> Gson.fromJson(json: String)
    = fromJson(json, T::class.java)
```

```
val link: Link = gson.fromJson(json)
```

Reified

```
inline fun <reified T> Gson.fromJson(json: String)
    = fromJson(json, T::class.java)
```

```
useLink( gson.fromJson(json) )

fun useLink(link: Link){
    ...
}
```

受检异常

```
FileWriter fw = new FileWriter(file);
fw.write(string, 0, string.length());
fw.close();
```

受检异常

```
try {
    FileWriter fw = new FileWriter(file);
    fw.write(string, 0, string.length());
    fw.close();
} catch (IOException e) {
    e.printStackTrace();
}
```

受检异常

```
FileWriter fw = null;
try {
    fw = new FileWriter(file);
    fw.write(string, 0, string.length());
} catch (IOException e) {
    e.printStackTrace();
} finally {
    fw.close();
}
```

受检异常

```
FileWriter fw = null;
try {
    fw = new FileWriter(file);
    fw.write(string, 0, string.length());
} catch (IOException e) {
    e.printStackTrace();
} finally {
    try {
        fw.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

受检异常

```
FileWriter fw = null;
try {
    fw = new FileWriter(file);
    fw.write(string, 0, string.length());
} catch (IOException e) {
    e.printStackTrace();
} finally {
    try {
        fw.close();
```

Method invocation 'close' may produce 'java.lang.NullPointerException' more... (⌘F1)

```
        e.printStackTrace();
    }
}
```

受检异常

```
FileWriter fw = null;
try {
    fw = new FileWriter(file);
    fw.write(string, 0, string.length());
} catch (IOException e) {
    e.printStackTrace();
} finally {
    try {
        fw.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

受检异常

```
FileWriter fw = null;
try {
    fw = new FileWriter(file);
    fw.write(string, 0, string.length());
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        fw.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

受检异常

```java
FileWriter fw = new FileWriter(file);
fw.write(string, 0, string.length());
fw.close();
```


```java
FileWriter fw = null;
try {
    fw = new FileWriter(file);
    fw.write(string, 0, string.length());
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        fw.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

受检异常

```
file.writer().use {
    it.write(string)
}
```

DSL

```
buildscript {
    ext.kotlin_version = '1.1.51'
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    }
}
```

DSL

```
buildscript {
    ext.kotlin_version = '1.1.51'
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    }
}
```

DSL

```
buildscript {
    ext.kotlin_version = '1.1.51'
    repositories {
      jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    }
}
```

DSL

```
buildscript {
    ext.kotlin_version = '1.1.51'
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    }
}
```

DSL

```
buildscript {
    extra["kotlinVersion"] = "1.1.51"
    repositories {
        jcenter()
    }

    val kotlinVersion: String by extra
    dependencies {
        classpath("com.android.tools.build:gradle:2.3.0")
        classpath("org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlinVersion")
    }
}
```

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
    }
}
```

See: https://try.kotl.in/#/Examples/Longer%20examples/HTML%20Builder/HTML%20Builder.kt

DSL

```
html {
    head {
        title { +"XML encoding w
    }
    body {
        h1 { +"XML encoding wi
        p { +"this format can be
        // an element with attribut
        a(href = "http://jetbrains.
    }
}
```

```
<html>
  <head>
    <title>
      XML encoding with Kotlin
    </title>
  </head>
  <body class="fullScreen">
    <h1>
      XML encoding with Kotlin
    </h1>
    <p>
      this format can be used as an alternative markup to XML
    </p>
    <a href="http://jetbrains.com/kotlin">
      Kotlin
    </a>
  </body>
</html>

Process finished with exit code 0
```

See: https://try.kotl.in/#/Examples/Longer%20examples/HTML%20Builder/HTML%20Builder.kt

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
    }
}
```

See: https://try.kotl.in/#/Examples/Longer%20examples/HTML%20Builder/HTML%20Builder.kt

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        attributes["class"] = "fullScreen"
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
    }
}
```

DSL

```
html {
    head {
        title { +"XML encoding wit
    }
    body {
        attributes["class"] = "fullS
        h1 { +"XML encoding with
        p { +"this format can be u
        // an element with attributes
        a(href = "http://jetbrains.c
    }
}
```

```
<html>
  <head>
    <title>
      XML encoding with Kotlin
    </title>
  </head>
  <body class="fullScreen">
    <h1>
      XML encoding with Kotlin
    </h1>
    <p>
      this format can be used as an alternative markup to XML
    </p>
    <a href="http://jetbrains.com/kotlin">
      Kotlin
    </a>
  </body>
</html>

Process finished with exit code 0
```

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        attributes["class"] = "fullScreen"
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
    }
}
```

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        "class"("fullScreen")
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
    }
}
```

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        "class"("fullScreen")
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
    }
}
```

```
operator fun String.invoke(value: String){
    attributes[this] = value
}
```

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        "class"("fullScreen")
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
    }
}
```

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        "class"("fullScreen")
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
         "android"{
              "Kotlin"{    +"We have started using Kot lin in Android!!" }
        }
    }
}
```

DSL

```
html {
    head {
        title { +"XML encoding with K
    }
    body {
        "class"("fullScreen")
        h1 { +"XML encoding with K
        p { +"this format can be use
        // an element with attributes a
        a(href = "http://jetbrains.com
         "android"{
              "Kotlin"{    +"We have s
        }
    }
}
```

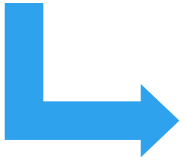
```
<html>
  <head>
    <title>
      XML encoding with Kotlin
    </title>
  </head>
  <body class="fullScreen">
    <h1>
      XML encoding with Kotlin
    </h1>
    <p>
      this format can be used as an alternative markup to XML
    </p>
    <a href="http://jetbrains.com/kotlin">
      Kotlin
    </a>
    <android>
      <Kotlin>
        We have started using Kotlin in Android!!
      </Kotlin>
    </android>
  </body>
</html>

Process finished with exit code 0
```

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        "class"("fullScreen")
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
         "android"{
              "Kotlin"{    +"We have started using Kot lin in Android!!" }
        }
    }
}
```

DSL

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        "class"("fullScreen")
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://jetbrains.com/kotlin") { +"Kotlin" }
        "android"{
            "Kotlin"{    +"We have started using Kot lin in Android!!" }
        }
    }
}
```



```
operator fun String.invoke(init: BodyTag.() -> Unit)
    = object: BodyTag(this@invoke){}
    .apply { this@Tag.initTag(this, init) }
```

反射

- 一个 2.5M 大小的 jar 包 compile "org.jetbrains.kotlin:kotlin-reflect:$kotlin_version"
- 不支持的 built-in Kotlin types
- 还没来得及优化的性能

	构造对象	访问属性	修改属性	调用方法
Java 反射	12.7	25.2	12.2	18.8
Kotlin 反射	14938.0	85247.5	1316.7	326.3

Kotlin 公众号文章：Kotlin 反射你敢用吗？

Kotlin-Android-Extensions

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

Kotlin-Android-Extensions

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'
```

Kotlin-Android-Extensions

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Hello World!"/>

</RelativeLayout>
```

Kotlin-Android-Extensions

```xml
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <TextView
      android:id="@+id/textView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerInParent="true"
      android:text="Hello World!"/>

</RelativeLayout>
```

```kotlin
class MainActivity : AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      textView.text = "Hello"
   }
}
```

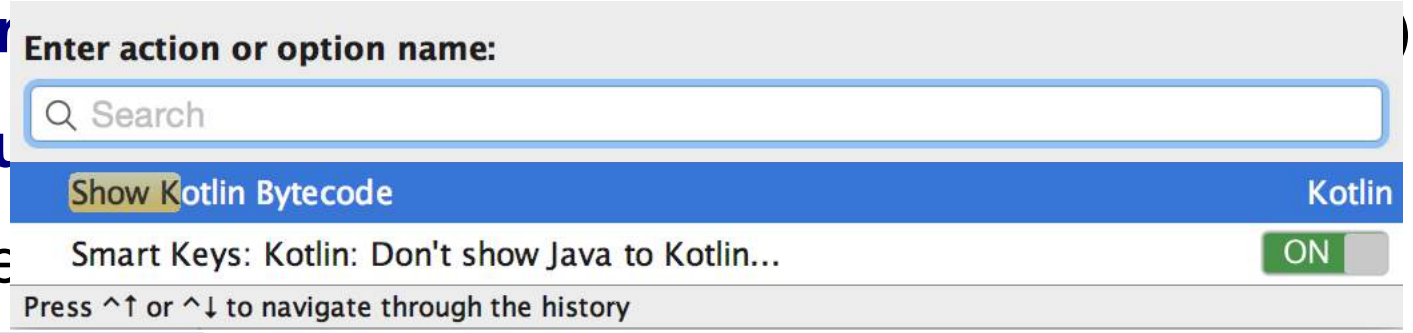
Kotlin-Android-Extensions

```xml
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <TextView
      android:id="@+id/textView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerInParent="true"
      android:text="Hello World!"/>

</RelativeLayout>
```

```kotlin
class MainActivity : AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      textView.text = "Hello"
   }
}
```

Kotlin-Android-Extensions

```
class MainActivity : AppCompatActivity() {

    over                                                    ) {

        su

        se

        textView.text = "Hello"

    }

}
```



Kotlin-Android-Extensions

```
LINENUMBER 11 L2
ALOAD 0
GETSTATIC com/bennyhuo/testrelection/R$id.textView : I
INVOKEVIRTUAL com/bennyhuo/testrelection/MainActivity._$_findCachedViewById (I)Landroid/view/View;
CHECKCAST android/widget/TextView
LDC "Hello"
CHECKCAST java/lang/CharSequence
INVOKEVIRTUAL android/widget/TextView.setText (Ljava/lang/CharSequence;)V
```

Kotlin-Android-Extensions

```
LINENUMBER 11 L2
ALOAD 0
GETSTATIC com/bennyhuo/testrelection/R$id.textView : I
INVOKEVIRTUAL com/bennyhuo/testrelection/MainActivity._$_findCachedViewById (I)Landroid/view/View;
CHECKCAST android/widget/TextView
LDC "Hello"
CHECKCAST java/lang/CharSequence
INVOKEVIRTUAL android/widget/TextView.setText (Ljava/lang/CharSequence;)V
```

Kotlin-Android-Extensions

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp">

  <TextView
    android:id="@+id/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_vertical"
    android:layout_marginLeft="10dp"
    android:textColor="#000"
    android:textSize="16sp"/>

</FrameLayout>
```

Kotlin-Android-Extensions

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp">

  <TextView
    android:id="@+id/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_vertical"
    android:layout_marginLeft="10dp"
    android:textColor="#000"
    android:textSize="16sp"/>

</FrameLayout>
```

```kotlin
override fun getView(position: Int,
        convertView: View?,
        parent: ViewGroup
): View {
    val view = convertView
            ?: inflater.inflate(R.layout.item, parent, false)
    view.name.text = "benny"
    return view
}
```

Kotlin-Android-Extensions

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp">

  <TextView
    android:id="@+id/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_vertical"
    android:layout_marginLeft="10dp"
    android:textColor="#000"
    android:textSize="16sp"/>

</FrameLayout>
```

```kotlin
override fun getView(position: Int,
        convertView: View?,
        parent: ViewGroup
): View {
    val view = convertView
            ?: inflater.inflate(R.layout.item, parent, false)
    view.name.text = "benny"
    return view
}
```

Anko 扩展

```
compile "org.jetbrains.anko:anko:$anko_version"
```

Anko 扩展

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(TestActivity.this, "I'm clicked!", Toast.LENGTH_SHORT);
    }
});
```

Anko 扩展

```
button.setOnClickListener(View.OnClickListener {
    Toast.makeText(this@MainActivity, "I'm clicked!", Toast.LENGTH_SHORT)
})
```

Anko 扩展

```
button.setOnClickListener({
    Toast.makeText(this@MainActivity, "I'm clicked!", Toast.LENGTH_SHORT)
})
```

Anko 扩展

```
button.setOnClickListener{
    Toast.makeText(this@MainActivity, "I'm clicked!", Toast.LENGTH_SHORT)
}
```

Anko 扩展

```
button.onClick {
    Toast.makeText(this@MainActivity, "I'm clicked!", Toast.LENGTH_SHORT)
}
```

Anko 扩展

```
button.onClick {
    Toast.makeText(this@MainActivity, "I'm clicked!", Toast.LENGTH_SHORT).show()
}
```

Anko 扩展

```
button.onClick {
    toast("I'm clicked!")
}
```

Anko 扩展

```
AlertDialog alertDialog = new AlertDialog.Builder(this)
        .setTitle("警告！")
        .setMessage("95后都在玩，再不学 Kotlin 就说明你老啦！")
        .setPositiveButton("朕知道了", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        })
        .setNegativeButton("朕就不学", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(TestActivity.this, "反了你们了", Toast.LENGTH_SHORT).show();
                dialog.dismiss();
            }
        }).create();
alertDialog.show();
```

Anko 扩展

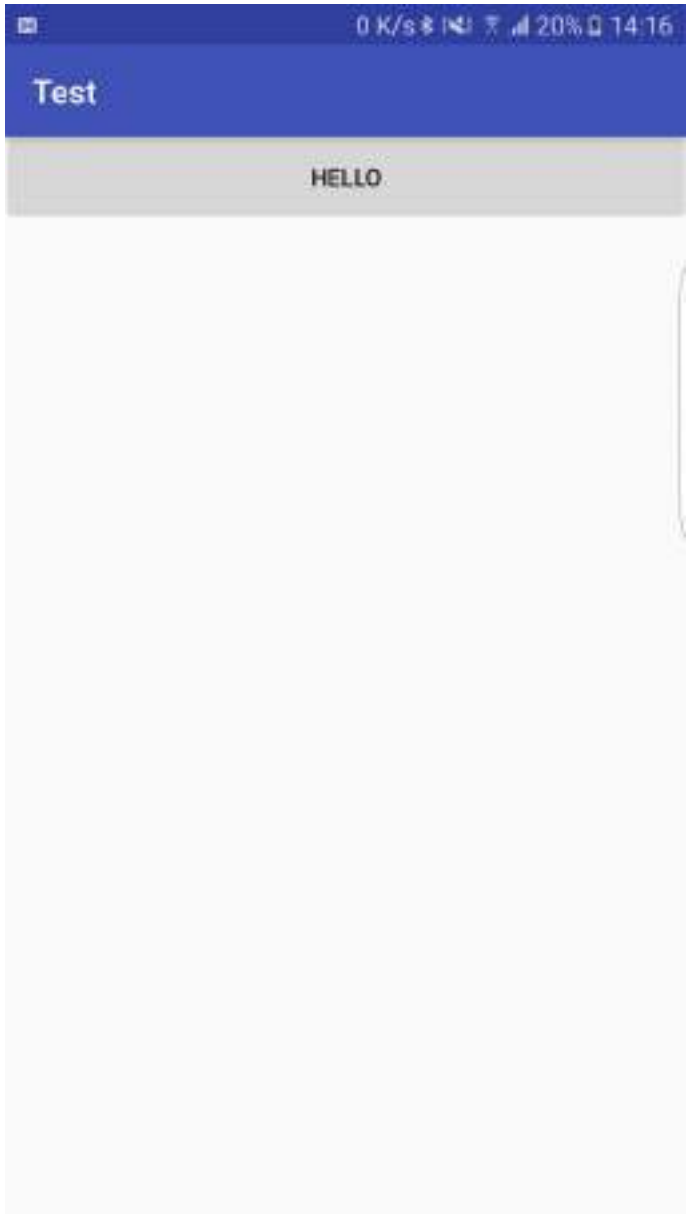
```
alert {
    title = "祝贺！"
    message = "恭喜成功入坑 Kotlin！"
    positiveButton("朕知道了"){
        toast("多大点儿事儿")
    }
    negativeButton("行啦行啦"){
        toast("退下吧")
    }
}.show()
```

Anko 扩展

```
alert {
    title = "祝贺！"
    message = "恭喜成功入坑 Kotlin！"
    positiveButton("朕知道了"){
        toast("多大点儿事儿")
    }
    negativeButton("行啦行啦"){
        toast("退下吧")
    }
}.show()
```

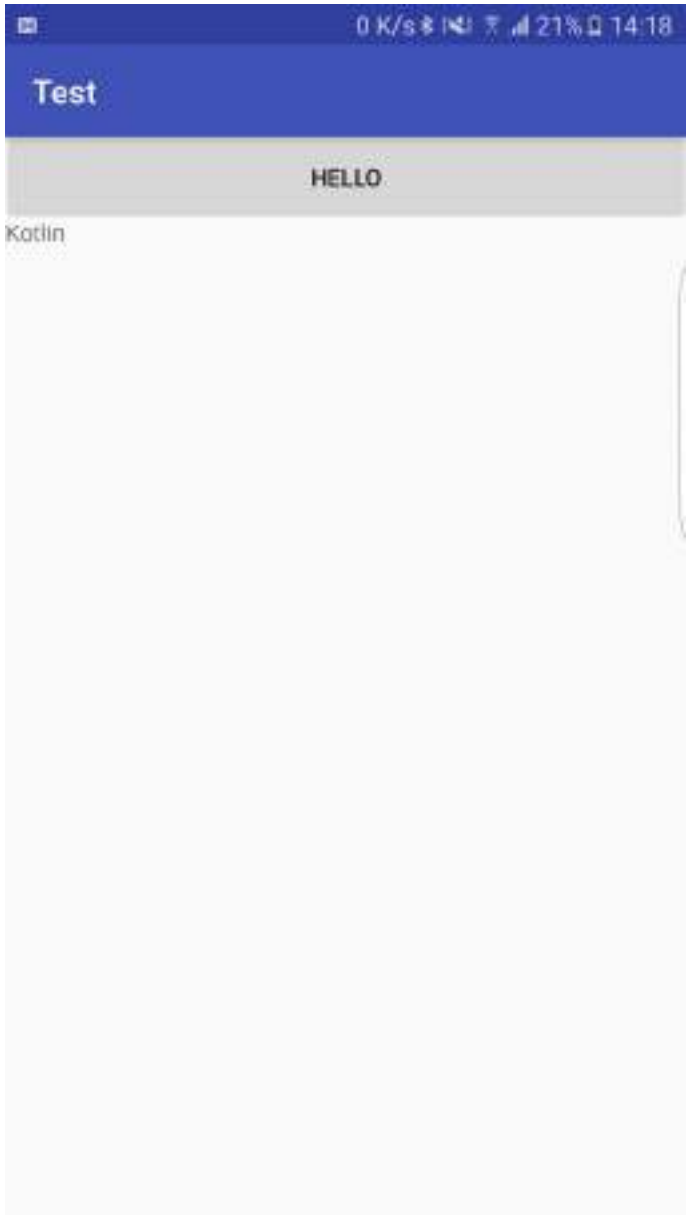
Anko 布局

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        verticalLayout {
            button("Hello")
        }
    }
}
```



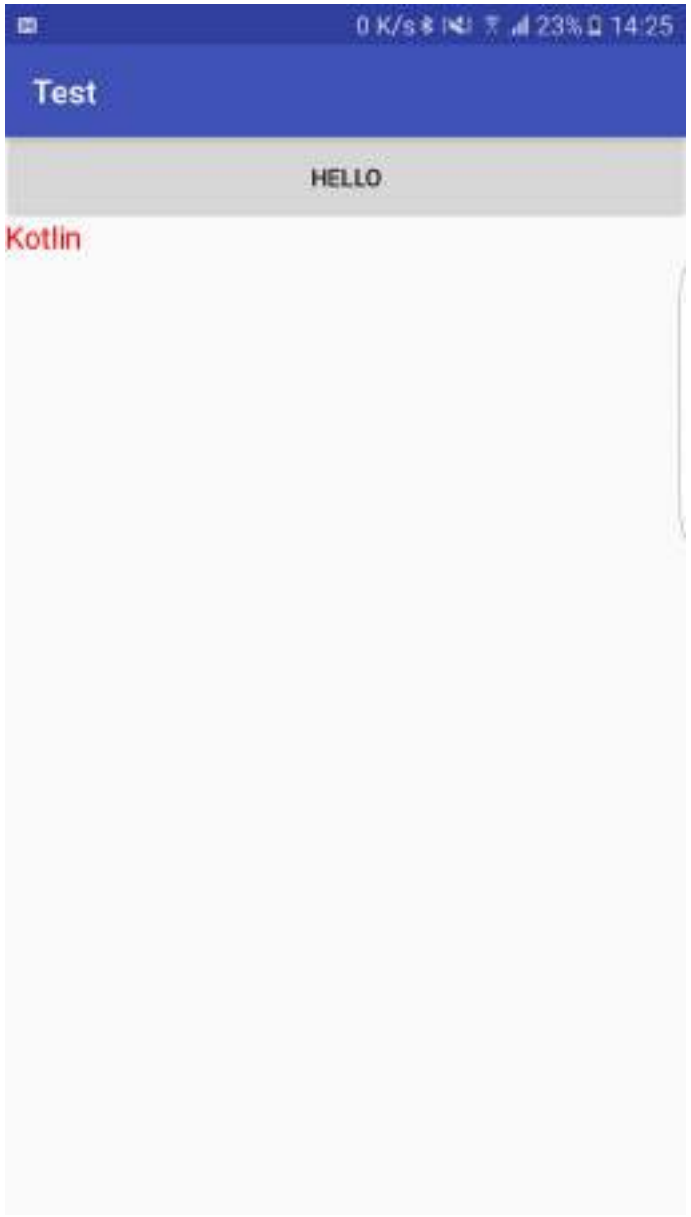
Anko 布局

```
verticalLayout {
    button("Hello")
    textView("Kotlin")
}
```



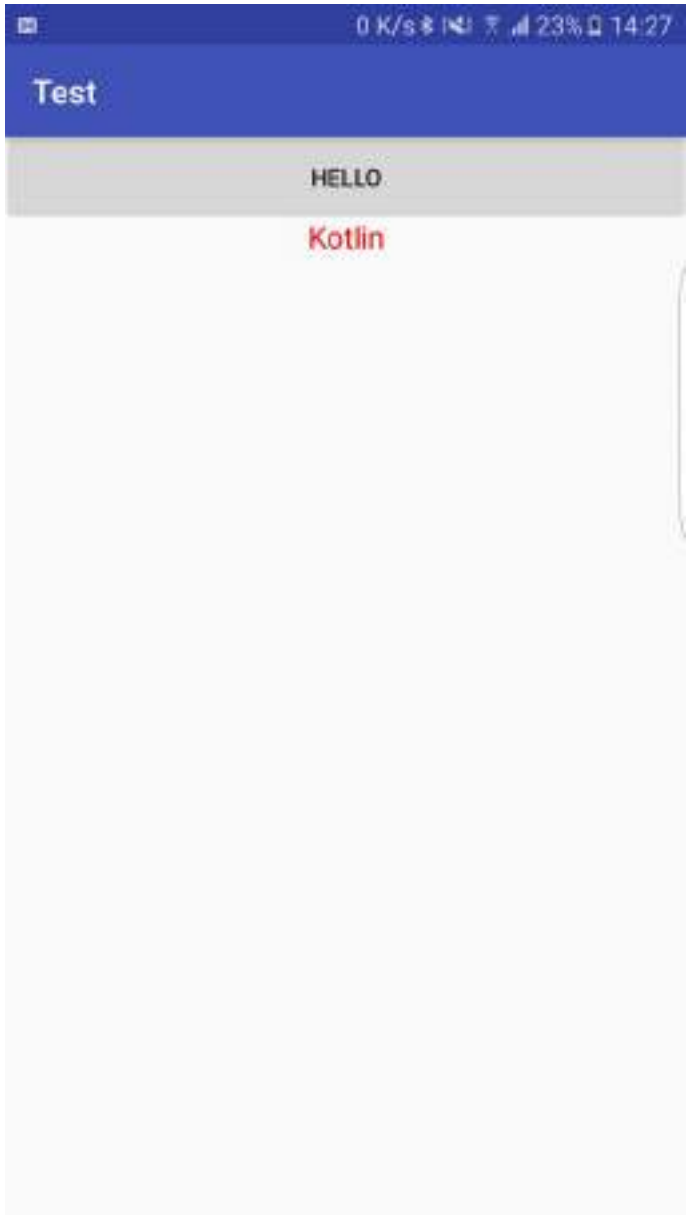
Anko 布局

```
verticalLayout {
    button("Hello")
    textView("Kotlin"){
        textSize = 18f
        textColor = Color.RED
    }
}
```



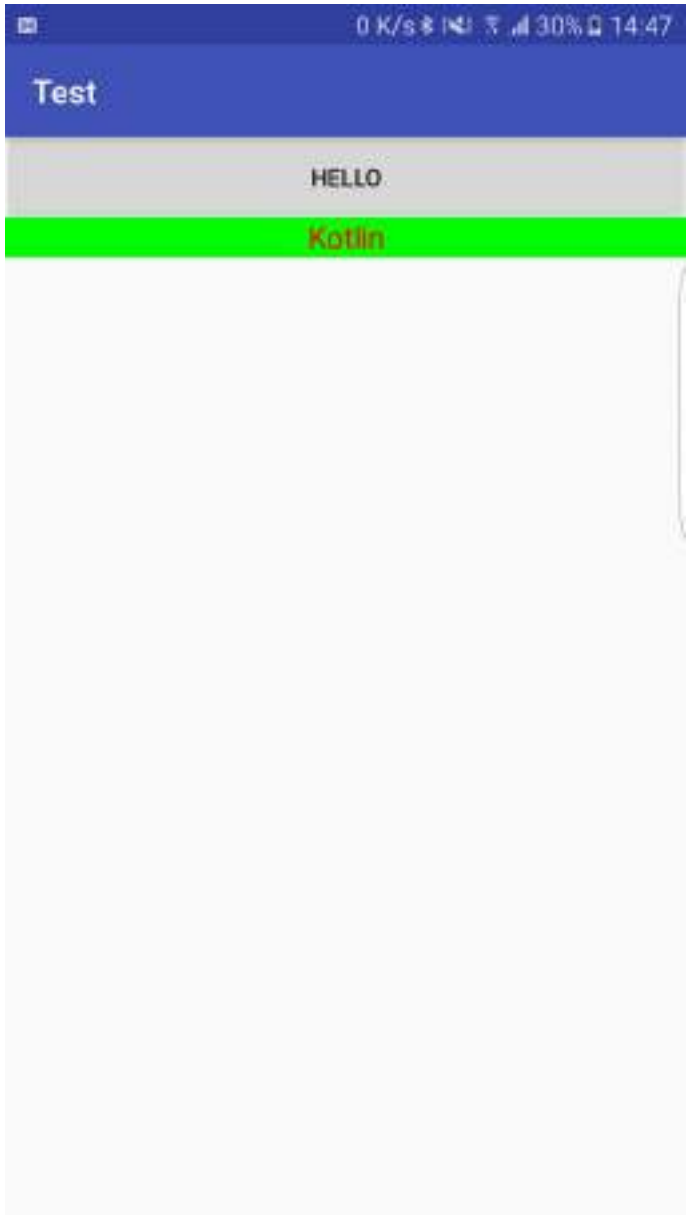
Anko 布局

```
verticalLayout {
    button("Hello")
    textView("Kotlin"){
        textSize = 18f
        textColor = Color.RED
        gravity = Gravity.CENTER
    }
}
```



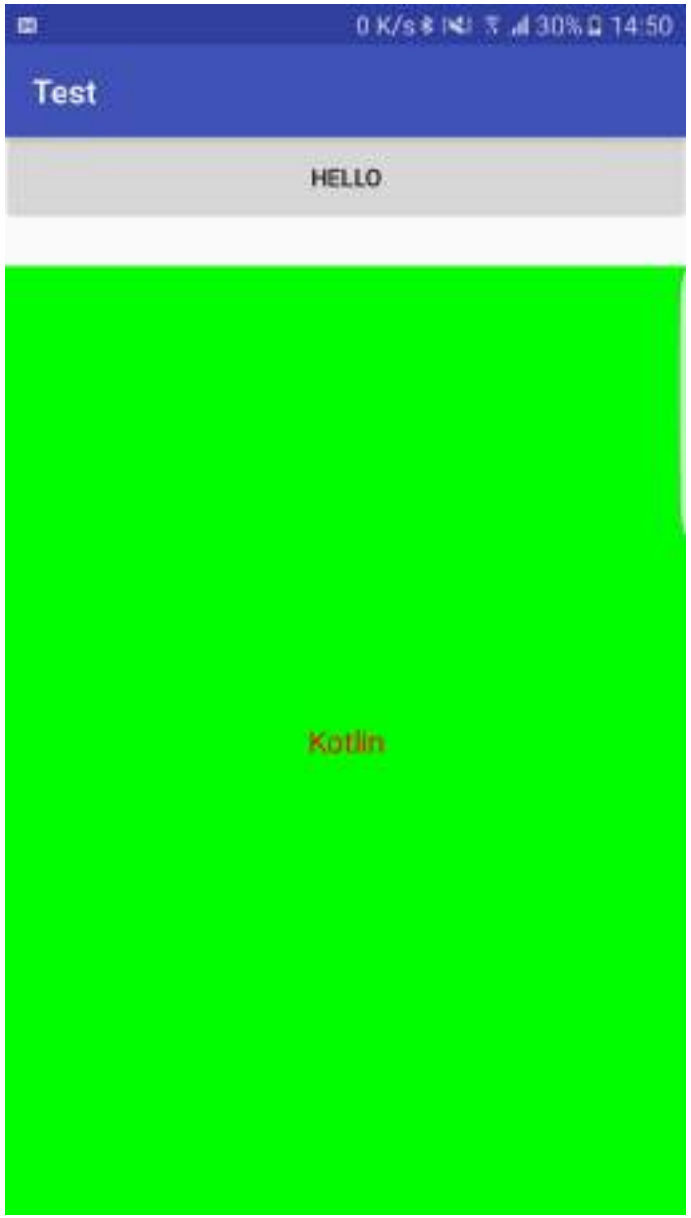
Anko 布局

```
verticalLayout {
    button("Hello")
    textView("Kotlin"){
        textSize = 18f
        textColor = Color.RED
        gravity = Gravity.CENTER
        backgroundColor = Color.GREEN
    }
}
```



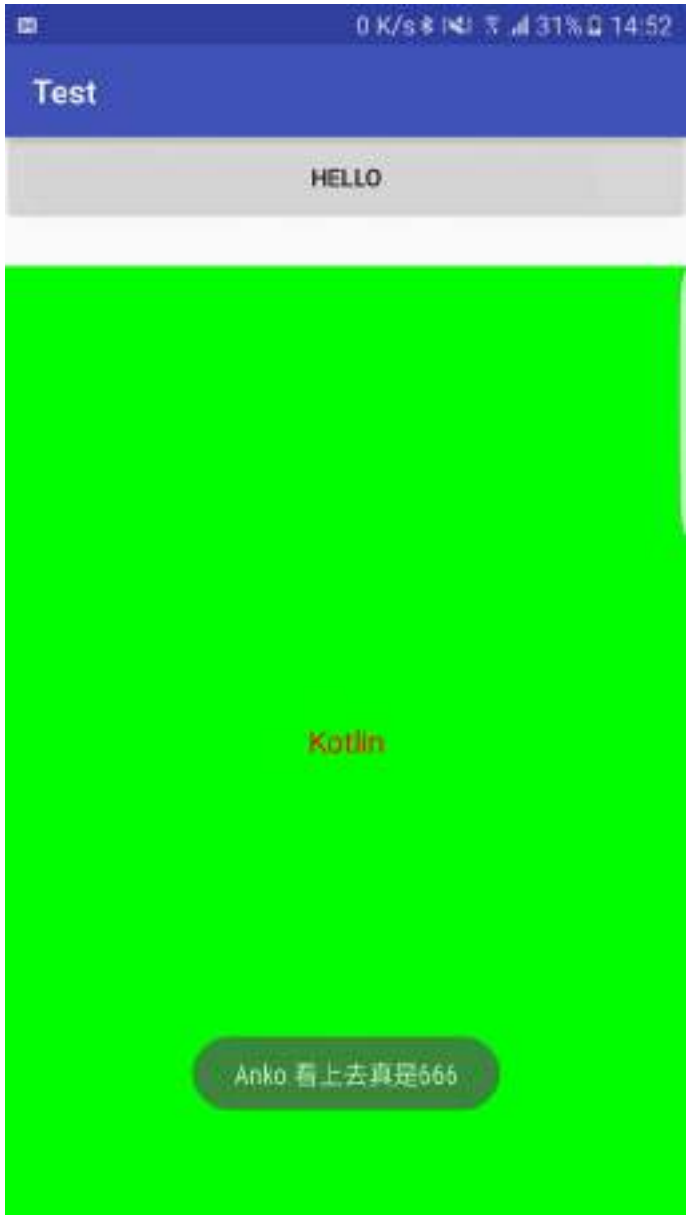
Anko 布局

```
verticalLayout {
    button("Hello")
    textView("Kotlin"){
        textSize = 18f
        textColor = Color.RED
        gravity = Gravity.CENTER
        backgroundColor = Color.GREEN
    }.lparams(matchParent, matchParent){
        topMargin = dip(30)
    }
}
```



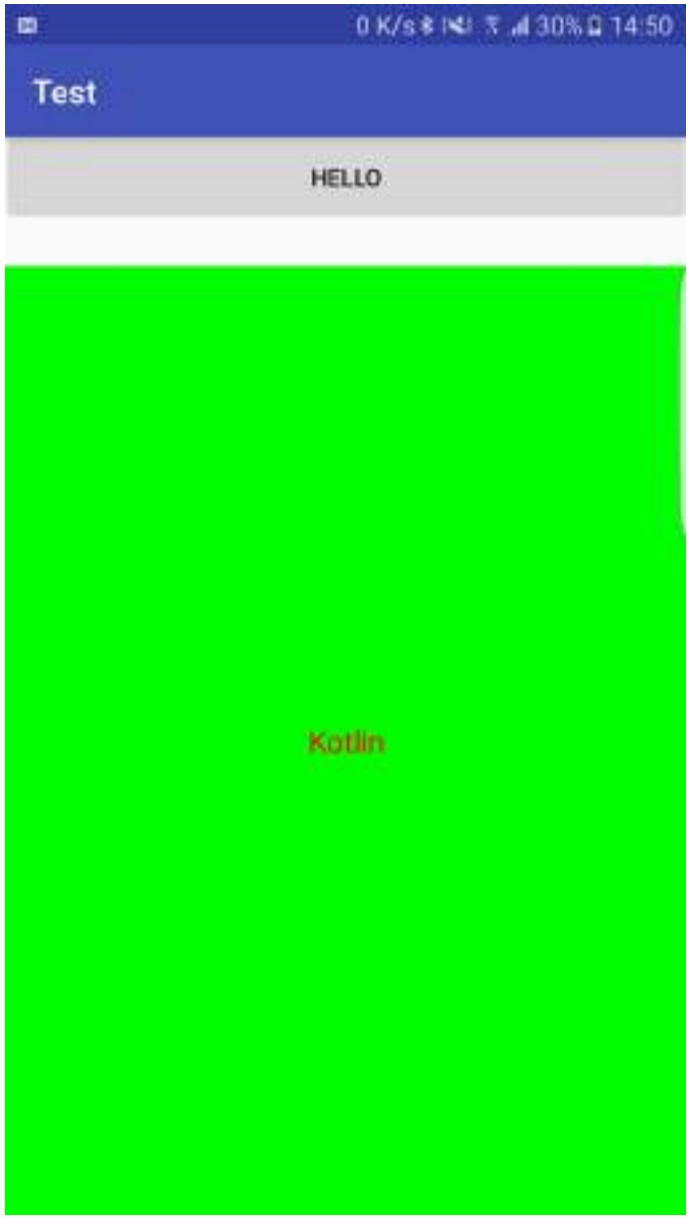
Anko 布局

```
verticalLayout {
    button("Hello") {
        onClick { toast("Anko 看上去真是666") }
    }
    textView("Kotlin"){
        textSize = 18f
        textColor = Color.RED
        gravity = Gravity.CENTER
        backgroundColor = Color.GREEN
    }.lparams(matchParent, matchParent){
        topMargin = dip(30)
    }
}
```



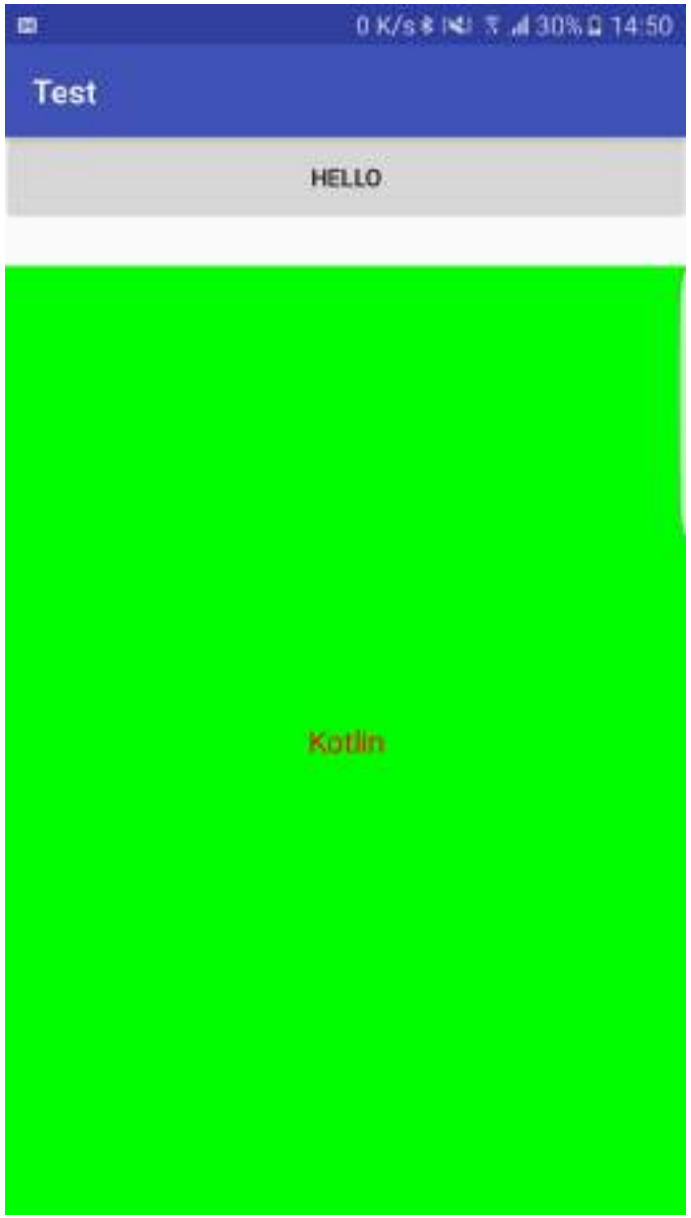
Anko 布局

```
verticalLayout {
    button("Hello") {
        onClick { toast("Anko 看上去真是666") }
    }
    themedTextView(
            "Kotlin",
            R.style.commonText
    ).lparams(matchParent, matchParent) {
            topMargin = dip(30)
            }
}
```



Anko 布局

```
<color name="red">#FF0000</color>
<color name="green">#00FF00</color>
<style name="commonText">
    <item name="android:textSize">18sp</item>
    <item name="android:textColor">@color/red</item>
    <item name="android:gravity">center</item>
    <item name="android:background">@color/green</item>
</style>
```



Anko 布局

- 无运行时开销，类型安全
- 代码更容易复用
- 预览功能受限制，必须编译才可预览
- 使用体验一般，Kotlin-Android-extensions 无效

- 除非硬编码布局可考虑使用，XML 布局仍是最佳的布局方案。

更多关于 Anko

https://github.com/Kotlin/anko

Coroutine

- 轻量级调度执行
- 异步代码写起来看上去如同同步代码一般
- 异常处理更轻松

Coroutine 认知三步走

- 应用
- 标准库
- 字节码

Coroutine 应用

```
interface ImageRequestCallback{
    fun onSuccess(bitmap: Bitmap)
    fun onError(e: Throwable)
}
fun fetchImageWithCallback(url: String, callback: ImageRequestCallback){
    try {
        callback.onSuccess(fetchImage(url))
    }catch (e: Exception){
        callback.onError(e)
    }
}
```

Coroutine 应用

```
interface ImageRequestCallback{
    fun onSuccess(bitmap: Bitmap)
    fun onError(e: Throwable)
}
fun fetchImageWithCallback(url: String, callback: ImageRequestCallback){
    try {
        callback.onSuccess(fetchImage(url))
    }catch (e: Exception){
        callback.onError(e)
    }
}
```

```
fetchImageWithCallback(urlA, object : ImageRequestCallback {
    override fun onSuccess(bitmap: Bitmap) {
        handler.post { imageViewA.setImageBitmap(bitmap) }
    }

    override fun onError(e: Throwable) {
        e.printStackTrace()
        handler.post { showErrorOnUi(e.message) }
    }
})
```

Coroutine 应用

使用 Kotlinx.coroutines Android 库

compile **"org.jetbrains.kotlinx:kotlinx-coroutines-android:$version"**

Coroutine 应用

```
launch(UI) {
    val imageA = async { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewA.setImageBitmap(imageA.await())
    imageViewB.setImageBitmap(imageB.await())
}
```

Coroutine 应用

```
launch(UI) {
    val imageA = async { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewA.setImageBitmap(imageA.await())
    imageViewB.setImageBitmap(imageB.await())
}
```

Coroutine 应用 – Anko 扩展

```
compile "org.jetbrains.anko:anko-coroutines:$anko_version"
```

Coroutine 应用 – Anko 扩展

```
launch(UI) {
    val imageA = bg { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewA.setImageBitmap(imageA.await())
    imageViewB.setImageBitmap(imageB.await())
}
```

Coroutine 应用 – Anko 扩展

```
launch(UI) {
    val imageA = bg { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewA.setImageBitmap(imageA.await())
    imageViewB.setImageBitmap(imageB.await())
}
```

Coroutine 应用 – Anko 扩展

```
internal var POOL = newFixedThreadPoolContext(2 * Runtime.getRuntime().availableProcessors(), "bg")

inline fun <T> bg(crossinline block: () -> T): Deferred<T> = async(POOL) {
    block()
}
```

Coroutine 应用 – Anko 扩展

```
internal var POOL = newFixedThreadPoolContext(2 * Runtime.getRuntime().availableProcessors(), "bg")

inline fun <T> bg(crossinline block: () -> T): Deferred<T> = async(POOL) {
    block()
}
```

Coroutine 应用 – Anko 扩展

```
launch(UI) {
    val imageA = async { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewA.setImageBitmap(imageA.await())
    imageViewB.setImageBitmap(imageB.await())
}
```

很耗时

Activity 内存泄露

Coroutine 应用 – Anko 扩展

```
val imageViewARef = imageViewA.asReference()
val imageViewBRef = imageViewB.asReference()
launch(UI) {
    val imageA = bg { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewARef().setImageBitmap(imageA.await())
    imageViewBRef().setImageBitmap(imageB.await())
}
```

Coroutine 应用 – Anko 扩展

```
val imageViewARef = imageViewA.asReference()
val imageViewBRef = imageViewB.asReference()
launch(UI) {
    val imageA = bg { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    im ImageView! ().setImageBitmap(imageA.await())
    imageViewBRef().setImageBitmap(imageB.await())
}
```

Coroutine 应用 – Anko 扩展

```
class Ref<out T : Any> internal constructor(obj: T) {
    private val weakRef = WeakReference(obj)
    suspend operator fun invoke(): T {
        return suspendCoroutineOrReturn {
            val ref = weakRef.get() ?: throw CancellationException()
            ref
        }
    }
}
fun <T : Any> T.asReference() = Ref(this)
```

Coroutine 应用 – Anko 扩展

```
class Ref<out T : Any> internal constructor(obj: T) {
    private val weakRef = WeakReference(obj)
    suspend operator fun invoke(): T {
        return suspendCoroutineOrReturn {
            val ref = weakRef.get() ?: throw CancellationException()
            ref
        }
    }
}
fun <T : Any> T.asReference() = Ref(this)
```

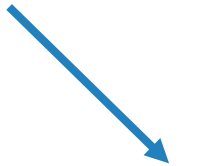
No expression found

```
throw CancellationException()
```

```
weakRef.get()
```

throw CancellationException()

Nothing



T (ImageView)

weakRef.get()

T (ImageView)

Coroutine 应用 – Anko 扩展

```
val imageViewARef = imageViewA.asReference()
val imageViewBRef = imageViewB.asReference()
launch(UI) {
    val imageA = bg { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewARef().setImageBitmap(imageA.await())
    imageViewBRef().setImageBitmap(imageB.await())
}
```

Coroutine 应用 – 处理异常

```
launch(UI) {
    try {
        ...
    } catch (e: Exception) {
        showErrorOnUi(e.message)
    }
}
```

Coroutine 应用 – 处理异常

```
launch(UI) {
    ...
}.invokeOnCompletion {
    it?.printStackTrace()
}
```

无异常时为null

Coroutine 应用 – AsyncAwait 扩展

compile **'co.metalab.asyncawait:asyncawait:1.0.0'**

Coroutine 应用 – AsyncAwait 扩展

```
async {
    val imageA = await { fetchImage(urlA) }
    val imageB = await { fetchImage(urlB) }
    imageViewARef().setImageBitmap(imageA)
    imageViewBRef().setImageBitmap(imageB)
}.onError {
    showErrorOnUi(it.message)
} .finally{
    releaseRefs()
}
```

Coroutine 应用对比

```
async {
    val imageA = await { fetchImage(urlA) }
    val imageB = await { fetchImage(urlB) }
    imageViewARef().setImageBitmap(imageA)
    imageViewBRef().setImageBitmap(imageB)
}.onError {
    showErrorOnUi(it.message)
} .finally{
    releaseRefs()
}
```

```
launch(UI) {
    val imageA = async { fetchImage(urlA) }
    val imageB = async { fetchImage(urlB) }
    imageViewARef().setImageBitmap(imageA.await())
    imageViewBRef().setImageBitmap(imageB.await())
}.invokeOnCompletion {
    it?.printStackTrace()
}
```

有何不同？

Coroutine 认知三步走

- 应用
- 标准库：较为底层，主要提供给应用层框架开发者使用
- 字节码：主要提供给编译器用

可参考Kotlin 公众号文章：

- 深入理解 Kotlin Coroutine
- 深入理解 Kotlin Coroutine (2)
- 深入理解 Kotlin Coroutine（3）

案例参考

案例参考 – 腾讯云图床上传工具

- 地址：**QCloudImageUploaderForMarkDown**
- 简介：一个完整的 JVM 程序，涉及文件操作、属性读写、正则表达式等内容。可以一键上传文件夹下所有图片到腾讯云并替换对应的 Markdown 文件中的图片地址。

案例参考 – 腾讯云图床上传工具

- 腾讯云图床上传工具使用方法
  - 开通腾讯云对象存储服务
  - 在腾讯云上创建对象存储的 Bucket
  - 获取 AppId、SecretId、SecretKey、BucketName 以及区域（例如华北就是tj）
  - 配置好 conf 目录下面的 settings.properties 文件

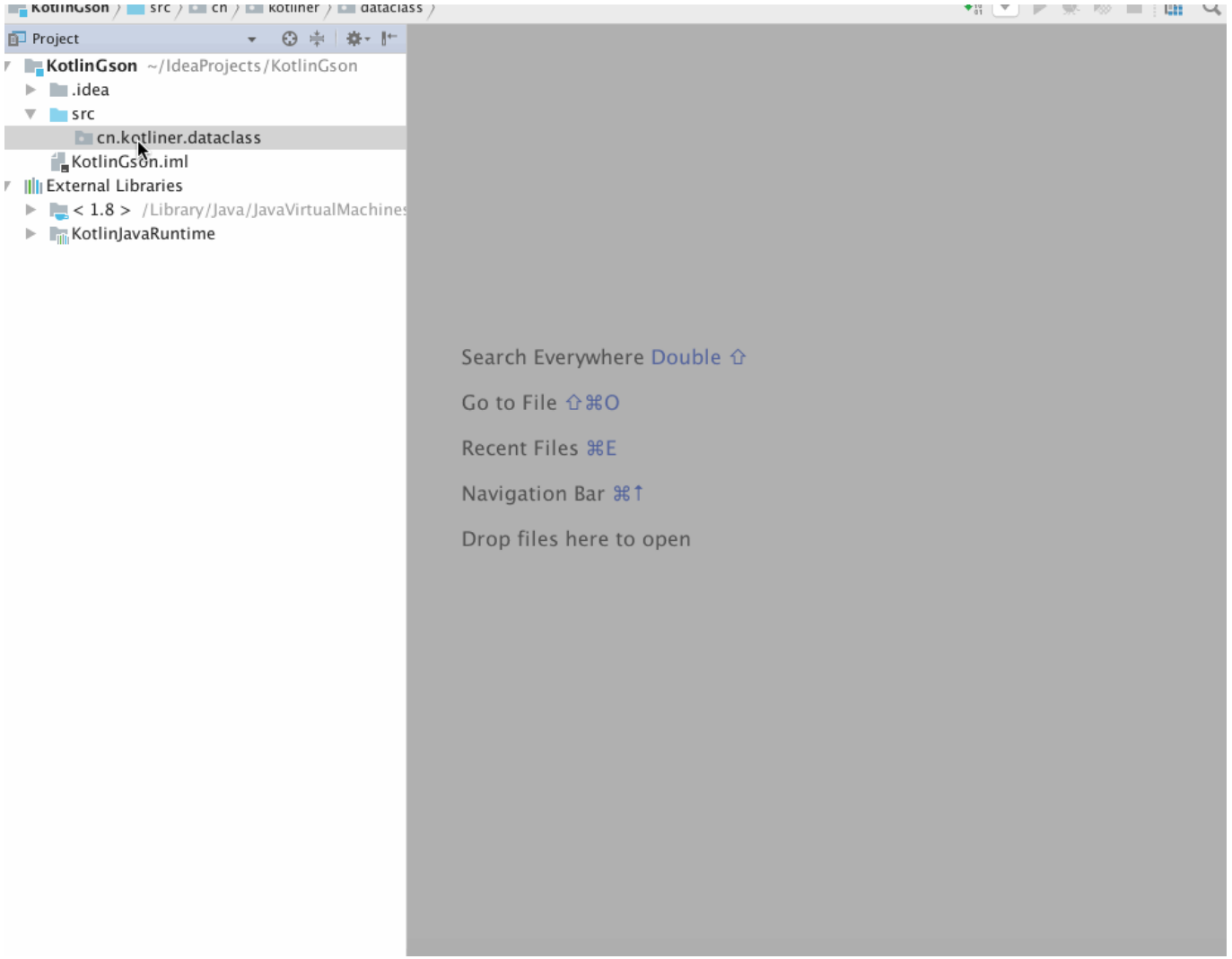
番外篇：博客/公众号文章编辑器

- MWEB：Markdown 本地编辑器
  - 支持复制图片直接粘贴到 Markdown
  - 支持博客目录文件管理（例如 Hexo 搭建的博客等）
  - 支持部分图床的上传（七牛、WordPress 等）
- 微信公众号排版工具：http://md.barretlee.com/
- 腾讯云图床上传工具
  - 支持以目录为单位上传和单文件上传，同一目录图片不重复上传
  - 支持直接替换原文档图片地址，可选择图片传后即删

案例参考 – Kotlin 版 GsonFormat

- 地址：**NewDataClassAction**
- 简介：将 Json 数据转换为 Kotlin data class 的 IntelliJ 插件。

案例参考 – Kotlin 版 GsonFormat

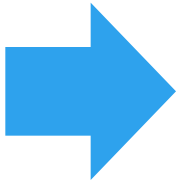


案例参考 – Kotlin 版 GsonFormat

```
{
  "name": "Bennyhuo",
  "company": "@Tencent",
  "site": "www.kotliner.cn",
  "location": "Beijing",
  "email": "enbandari@qq.com",
  "bio": "微信公众号 Kotlin"
}
```

案例参考 – Kotlin 版 GsonFormat

```
{
  "name": "Bennyhuo",
  "company": "@Tencent",
  "site": "www.kotliner.cn",
  "location": "Beijing",
  "email": "enbandari@qq.com",
  "bio": "微信公众号 Kotlin"
}
```



```
data class User(var name: String,
                var company: String,
                var site: String,
                var location: String,
                var email: String,
                var bio: String)
```

案例参考 – 谷歌官方 Android 案例

- 地址：Android Samples
- 简介：官方给出了较多案例，可供大家参考如何将 Kotlin 应用到 Android 开发当中。

案例参考 – Kotlin 版设计模式

- 地址：[Design-Patterns-In-Kotlin]
- 简介：使用 Kotlin 编写的设计模式案例

Table of Contents

案例参考 – Kotlin 版设计模式

```
class Printer(val stringFormatterStrategy: (String) -> String) {
    fun printString(string: String)
        = println(stringFormatterStrategy.invoke(string))
}

val lowerCaseFormatter: (String) -> String = { it.toLowerCase() }

val upperCaseFormatter = { it: String -> it.toUpperCase() }
```

案例参考 – Kotlin 版 ButterKnife

- 地址：**kotterknife**
- 简介：出自 Jake Wharton 之手，是属性代理的很好的学习案例。

案例参考 – Kotlin 版 ButterKnife

```
public class PersonView(context: Context, attrs: AttributeSet?)
    : LinearLayout(context, attrs) {
  val firstName: TextView by bindView(R.id.first_name)
  val lastName: TextView by bindView(R.id.last_name)
}
```

案例参考 – 协程框架 AsyncAwait

- 地址：AsyncAwait
- 简介：使用标准库 API 实现的协程封装，麻雀虽小五脏俱全，如果你想了解标准库中的协程 API，又无意于深入研究和扩展协程框架，推荐阅读它的源码。

案例参考 – 协程框架 Kotlinx.coroutines

- 地址：**kotlinx.coroutines**
- 简介：官方提供的协程框架，提供了较为完善和丰富的协程框架体系。如果你想更进一步深入学习研究 Kotlin 协程 API 的原理和使用，建议你仔细阅读这个框架。

“

纸上得来终觉浅，绝知此事要躬行

”

谢谢大家！