

06 고급 위젯 다루기



학습목표

- ❖ 고급 위젯을 다루는 방법을 익힌다.
- ❖ 뷰 컨테이너와 그 응용법을 이해한다.
- ❖ 매니페스트 파일 설정법을 익힌다.

차례

1. 고급 위젯
2. 뷰 컨테이너

01 날짜와 시간 관련 위젯

■ 아날로그시계, 디지털시계

```
java.lang.Object
└─ android.view.View
    └─ android.widget.AnalogClock
    └─ android.widget.TextView
        └─ android.widget.DigitalClock
```

아날로그시계, 디지털시계 계층도

- 화면에 시간을 표시하는 위젯으로 시계를 표현하는 용도
- View 클래스에서 상속받기 때문에 background 속성 등을 설정 가능함
 - DigitalClock은 textColor 같은 속성도 설정 가능함

예제 6-1 시계 관련 XML 코드

```
1 <LinearLayout>
2     <AnalogClock
3         android:layout_width="match_parent"
4         android:layout_height="wrap_content" />
5     <DigitalClock
6         android:layout_width="match_parent"
7         android:layout_height="wrap_content"
8         android:gravity="center" />
9 </LinearLayout>
```



01 날짜와 시간 관련 위젯

■ 크로노미터

```
java.lang.Object
└─ android.view.View
    └─ android.widget.TextView
        └─ android.widget.Chronometer
```

크로노미터 계층도

- 타이머 형식의 위젯
- 일반적으로 시간을 측정할 때 많이 사용함
- start(), stop(), reset() 메소드: 각각 크로노미터를 시작, 정지, 초기화함

예제 6-2 크로노미터의 XML 코드

```
1 <LinearLayout>
2     <Chronometer
3         android:id="@+id/chronometer1"
4         android:layout_width="match_parent"
5         android:layout_height="wrap_content"
6         android:format="시간 측정 : %s"
7         android:gravity="center"
8         android:textSize="30dp" />
9 </LinearLayout>
```

시간 측정 : 00:00

01 날짜와 시간 관련 위젯

■ 타임피커, 데이트피커, 캘린더뷰

```
java.lang.Object
└─ android.view.View
    └─ android.view.ViewGroup
        └─ android.widget.FrameLayout
            └─ android.widget.TimePicker
                └─ android.widget.DatePicker
                    └─ android.widget.CalendarView
```

타임피커, 데이트피커, 캘린더뷰 계층도

- 타임피커(TimePicker) : 시간 표시와 조절
- 데이트피커(DatePicker)와 캘린더뷰(CalendarView) : 날짜 표시와 조절
- 캘린더뷰의 XML 속성
 - showWeekNumber : 현재 몇 주 차인지를 각 주의 맨 앞에 출력함
 - 디폴트는 true이지만 false로 하는 것이 더 깔끔하고 보기 좋음

01 날짜와 시간 관련 위젯

예제 6-3 타임피커와 데이트피커의 XML 코드

```

1 <LinearLayout>
2     <TimePicker
3         android:timePickerMode="spinner"
4         android:layout_width="match_parent"
5         android:layout_height="wrap_content" />
6     <DatePicker
7         android:datePickerMode="spinner"
8         android:layout_width="match_parent"
9         android:layout_height="wrap_content" />
10 </LinearLayout>

```



- 시간과 날짜는 대화상자(다이얼로그 박스)로 표현하는 것을 권장함
 - 대화상자로 나타내려면 Kotlin 코드에서 TimePickerDialog와 DatePickerDialog 클래스를 사용해야 함

01 날짜와 시간 관련 위젯

■ <실습 6-1> 날짜/시간 예약 앱 만들기

- 타이머 기능: <예약 시작>과 <예약완료>를 클릭하면 크로노미터가 타 이머로 동작함
- 날짜/시간 설정: <날짜 설정>과 <시간 설정>을 클릭하면 예약 날짜와 시간을 변경할 수 있음
- <예약완료>를 클릭하면 설정한 날짜와 시간이 결정됨

■ 안드로이드 프로젝트 생성

- (1) 새 프로젝트 만들기
 - 프로젝트 이름 : 'Project6_1'
 - 패키지 이름 : 'com.cookandroid.project6_1'
 - 그 외 규칙은 [실습 2-4]의 (1)~(4)를 따름

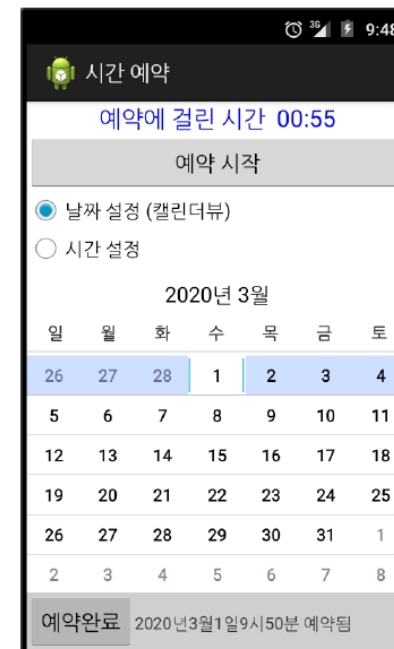


그림 6-1 날짜/시간 예약 앱 결과 화면

01 날짜와 시간 관련 위젯

- 화면 디자인 및 편집
 - (2) [app]-[res][layout]-[activity_main.xml]을 열고, 아래쪽의 [Text] 탭을 클릭하여 화면을 코딩함
 - 바깥 리니어레이아웃 안에 다음과 같이 화면을 구성함
 - 리니어레이아웃 : 크로노미터 1개, Button 1개 생성
 - » 위젯의 id는 chronometer1, btnStart
 - 라디오그룹 : 라디오버튼 2개 생성
 - » 위젯의 id는 rdoCal, rdoTime
 - 리니어레이아웃 : layout_weight를 1로 설정
 - » 프레임레이아웃을 두고 안에 캘린더뷰 1개, 타임피커 1개 생성
 - » 위젯의 id는 calendarView1, timePicker1
 - 리니어레이아웃 : 버튼 1개, 텍스트뷰 10개 생성
 - » 버튼의 id는 btnEnd
 - » 텍스트뷰는 홀수 차례에만 id는 tvYear, tvMonth, tvDay, tvHour, tvMinute

01 날짜와 시간 관련 위젯

예제 6-4 activity_main.xml

```

1 <LinearLayout>
2   <LinearLayout
3     android:orientation="vertical" >
4     <Chronometer
5       android:id="@+id/chronometer1"
6       android:format="예약에 걸린 시간 %s"
7       android:gravity="center"
8       android:textSize="20dp" />
9     <Button
10      android:id="@+id/btnStart"
11      android:text="예약 시작" />
12   </LinearLayout>
13   <RadioGroup>
14     <RadioButton
15       android:id="@+id/rdoCal"
16       android:text="날짜 설정 (캘린더뷰)" />
17     <RadioButton
18       android:id="@+id/rdoTime"
19       android:text="시간 설정" />
20   </RadioGroup>
21   <LinearLayout
22     android:layout_weight="1" >
23     <FrameLayout>
24       <CalendarView
25         android:id="@+id/calendarView1"
26         android:showWeekNumber="false" />
27       <TimePicker
28         android:id="@+id/timePicker1" />
29     </FrameLayout>

```



01 날짜와 시간 관련 위젯

```
30 </LinearLayout>
31 <LinearLayout
32     android:background="#CCCCCC" >
33     <Button
34         android:id="@+id/btnEnd"
35         android:text="예약완료" />
36     <TextView
37         android:id="@+id/tvYear"
38         android:text="0000" />
39     <TextView
40         android:text="년" />
41     ~~~ 생략(텍스트뷰 8개) ~~~
42 </LinearLayout>
43 </LinearLayout>
```

01 날짜와 시간 관련 위젯

- Kotlin 코드 작성 및 수정
 - (3) [app]-[java]-[패키지 이름]-[MainActivity]를 열기
 - (4) 다음과 같은 변수를 전역변수로 선언
 - activity_main.xml에서 id를 부여한 12개 위젯에 대응할 위젯 변수 12개

예제 6-5 Kotlin 코드 1

```

1  ~~~ 생략(import 문) ~~~
2  class MainActivity : AppCompatActivity() {
3      lateinit var chrono : Chronometer
4      lateinit var btnStart : Button
5      lateinit var btnEnd : Button
6      lateinit var rdoCal : RadioButton
7      lateinit var rdoTime : RadioButton
8      lateinit var calView : CalendarView
9      lateinit var tPicker : TimePicker
10     lateinit var tvYear : TextView
11     lateinit var tvMonth : TextView
12     lateinit var tvDay : TextView
13     lateinit var tvHour : TextView
14     lateinit var tvMinute : TextView
15     var selectYear : Int = 0
16     var selectMonth : Int = 0
17     var selectDay : Int = 0
18
19     override fun onCreate(savedInstanceState: Bundle?) {
20         ~~~ 생략 ~~~

```

01 날짜와 시간 관련 위젯

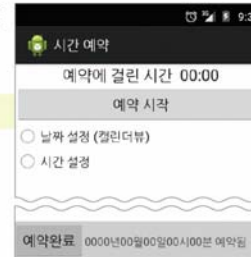
- (5) 메인 함수 onCreate() 내부 코딩
 - 위젯 변수 12개에 위젯을 대입
 - 처음에는 캘린더뷰와 타임피커가 보이지 않게 설정하고 프로젝트 실행

예제 6-6 Kotlin 코드 2

```

1  override fun onCreate(savedInstanceState: Bundle?) {
2      super.onCreate(savedInstanceState)
3      setContentView(R.layout.activity_main)
4      title = "시간 예약"
5
6      btnStart = findViewById<Button>(R.id.btnStart)
7      btnEnd = findViewById<Button>(R.id.btnEnd)
8
9      chrono = findViewById<Chronometer>(R.id.chronometer1)
10
11     rdoCal = findViewById<RadioButton>(R.id.rdoCal)
12     rdoTime = findViewById<RadioButton>(R.id.rdoTime)
13
14     tPicker = findViewById<TimePicker>(R.id.timePicker1)
15     calView = findViewById<CalendarView>(R.id.calendarView1)
16
17     tvYear = findViewById<TextView>(R.id.tvYear)
18     tvMonth = findViewById<TextView>(R.id.tvMonth)
19     tvDay = findViewById<TextView>(R.id.tvDay)
20     tvHour = findViewById<TextView>(R.id.tvHour)
21     tvMinute = findViewById<TextView>(R.id.tvMinute)
22
23     tPicker.visibility = View.INVISIBLE
24     calView.visibility = View.INVISIBLE
25 }

```



01 날짜와 시간 관련 위젯

- (6) 메인 함수 onCreate() 내부에 계속 코딩
 - [예제 6-6]의 24행과 25행 사이
 - 라디오버튼을 클릭하면 캘린더뷰와 타임피커 중 하나씩만 보이도록 클릭 이벤트 람다식 작성

예제 6-7 Kotlin 코드 3

```

1 rdoCal.setOnClickListener {
2     tPicker.visibility = View.INVISIBLE
3     calView.visibility = View.VISIBLE
4 }
5
6 rdoTime.setOnClickListener {
7     tPicker.visibility = View.VISIBLE
8     calView.visibility = View.INVISIBLE
9 }

```



01 날짜와 시간 관련 위젯

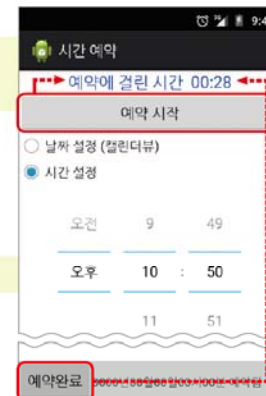
- (7) 메인 함수 onCreate() 내부에 계속 코딩(클릭 이벤트 람다식 작성)
 - <예약 시작>을 클릭하면 크로노미터 시작
 - <예약완료>를 클릭하면 정지

예제 6-8 Kotlin 코드 4

```

1  btnStart.setOnClickListener {
2      chrono.base = SystemClock.elapsedRealtime()
3      chrono.start()
4      chrono.setTextColor(Color.RED)
5  }
6
7  btnEnd.setOnClickListener {
8      chrono.stop()
9      chrono.setTextColor(Color.BLUE)
10 }
11
12 calView.setOnDateChangeListener { view, year, month, dayOfMonth ->
13     selectYear = year
14     selectMonth = month + 1
15     selectDay = dayOfMonth
16 }

```



01 날짜와 시간 관련 위젯

- (8) [예제 6-8]의 9행과 10행 사이(btnEnd의 클릭 이벤트 람다식 안)에 다음 예제를 코딩
 - <예약완료>를 클릭하면 캘린더뷰에서 설정한 연, 월, 일과 타임피커에서 설정한 시, 분이 맨 아래 텍스트뷰에 채워짐

예제 6-9 Kotlin 코드 5

```
1 tvYear.text = Integer.toString(selectYear)
2 tvMonth.text = Integer.toString(selectMonth)
3 tvDay.text = Integer.toString(selectDay)
4
5 tvHour.text = Integer.toString(tPicker.currentHour)
6 tvMinute.text = Integer.toString(tPicker.currentMinute)
```

■ 프로젝트 실행 및 결과 확인

- (9) <예약 시작> → <날짜 설정> → 캘린더뷰에서 날짜 선택 → <시간 설정> → 타임피커에서 시간 선택
→ <예약완료>의 순서로 클릭
 - [그림 6-1]과 같은 결과 화면이 나옴

01 날짜와 시간 관련 위젯

▶ 직접 풀어보기 6-1

[실습 6-1]을 다음과 같이 수정하라.

- 캘린더뷰 대신에 데이트피커를 사용하여 날짜를 설정한다.
- <예약 시작>과 <예약완료>를 없앤다. 대신 <예약 시작> 기능은 크로노미터를 클릭하면 동작하게 하고, <예약완료> 기능은 화면 하단의 연도(0000년)를 롱클릭하면 동작하게 한다.
- 크로노미터를 클릭하기 전에는 라디오버튼, 데이트피커, 타임피커가 안 보이도록 설정하고, 크로노미터를 클릭하면 라디오버튼이 나타나게 한다. 그리고 화면 하단의 연도(0000년)를 롱클릭하면 라디오버튼, 데이트피커, 타임피커가 다시 사라지게 한다.

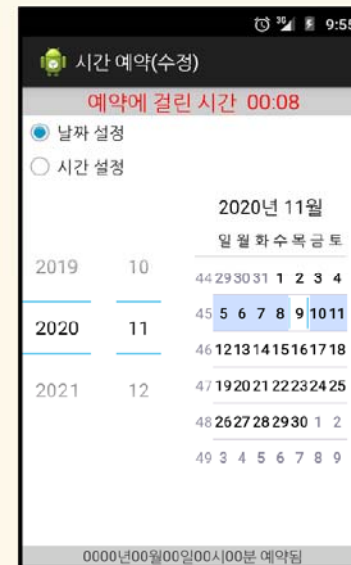


그림 6-2 수정된 날짜/시간 예약 앱

02 기타 위젯

■ 프로그레스바, 시크바, 레이팅바

```
java.lang.Object
└─ android.view.View
    └─ android.widget.ProgressBar
    └─ android.widget.AbsSeekBar
        └─ android.widget.RatingBar
        └─ android.widget.SeekBar
```

프로그레스바, 시크바, 레이팅바 계층도

- 진행 상태를 표시하는 기능

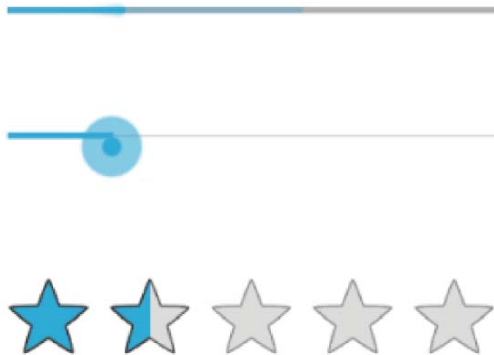


그림 6-3 프로그레스바, 시크바, 레이팅바

02 기타 위젯

- 프로그레스바(ProgressBar)
 - 작업의 진행 상태를 바(bar) 또는 원 형태로 제공
 - 바 형태
 - 어느 정도 진행되었는지를 확인할 수 있음
 - 원 형태
 - 현재 진행 중이라는 상태만 확인할 수 있음
 - 주로 사용되는 XML 속성
 - `max` : 범위를 지정
 - `progress` : 시작 지점을 지정
 - `secondaryProgress` : 두 번째 프로그레스바 지정

02 기타 위젯

■ 시크바(SeekBar)

- 프로그레스바의 하위 클래스로 사용법이 비슷함
- 사용자가 터치로 임의 조절이 가능함
- 음량을 조절하거나 동영상 재생 시 사용자가 재생 위치를 지정하는 용도로 사용할 수 있음

■ 레이팅바(RatingBar)

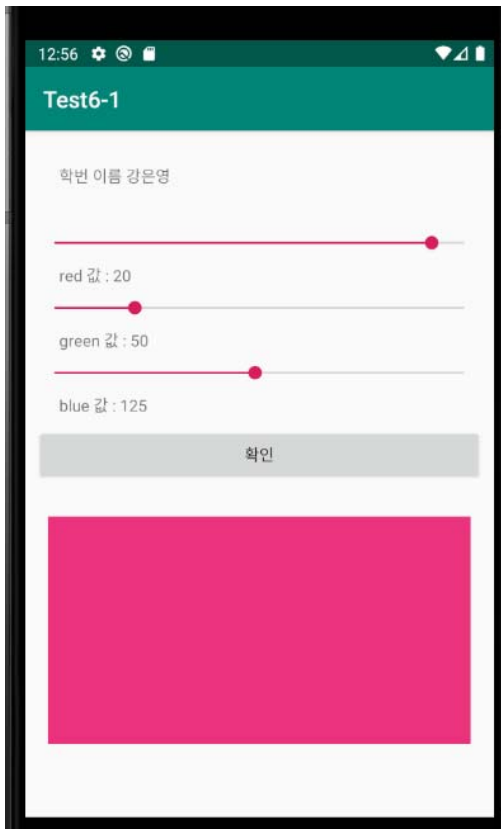
- 진행 상태를 별 모양으로 표시
- 프로그레스바의 하위 클래스로 사용법이 비슷함
- 서적, 음악, 영화 등에 대한 선호도를 나타낼 때 주로 사용됨
- 주로 사용되는 XML 속성
 - `numStars` : 별의 개수를 정함
 - `rating` : 초깃값 지정
 - `stepSize` : 한 번에 채워지는 개수를 정함

02 기타 위젯

예제 6-12 프로그레스바, 시크바, 레이팅바의 XML 코드

```
1 <LinearLayout>
2     <ProgressBar
3         style="?android:attr/progressBarStyleHorizontal"
4         android:max="100"
5         android:progress="20"
6         android:secondaryProgress="50" />
7     <SeekBar
8         android:progress="20" />
9     <RatingBar
10        android:numStars="5"
11        android:rating="1.5"
12        android:stepSize="0.5" />
13 </LinearLayout>
```

02 기타 위젯 활용



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
```

```
<TextView
    android:id="@+id/text"
    android:layout_margin="30dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="학번 이름 강은영"
/>
```

```
<SeekBar
    android:id="@+id/redBar"
    android:layout_margin="10dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:max="255"
    android:progress="20" />
```

```
<TextView
    android:id="@+id/redText"
    android:layout_marginLeft="30dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="red 값 : "
/>
```

```
<SeekBar
    android:id="@+id/blueBar"
    android:layout_margin="10dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:max="255"
    android:progress="125" />
```

```
<TextView
    android:id="@+id/blueText"
    android:layout_marginLeft="30dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="blue 값 : "
/>
```

```
<Button
    android:id="@+id/btnOk"
    android:layout_margin="10dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="확인" />
```

```
<View
    android:id="@+id/dpView"
    android:background="#000000"
    android:layout_margin="20dp"
    android:layout_width="match_parent"
    android:layout_height="200dp" />
```

```
</LinearLayout>
```

중간 SeekBar,
TextView 생략

02 기타 위젯 활용

```
class MainActivity : AppCompatActivity() {
    var red = 0
    var green = 0
    var blue = 0

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        red = redBar.progress
        green = greenBar.progress
        blue = blueBar.progress
        redText.text = "red 값 : " + red
        greenText.text = "green 값 : " + green
        blueText.text = "blue 값 : " + blue
        var seekBarInstance = NoNameSeekBar()
        redBar.setOnSeekBarChangeListener(seekBarInstance)
        greenBar.setOnSeekBarChangeListener(object : SeekBar.OnSeekBarChangeListener {
            override fun onProgressChanged(p0: SeekBar?, p1: Int, p2: Boolean) {
                green = p1
                greenText.text = "그린값 : " + green
            }
            override fun onStartTrackingTouch(p0: SeekBar?) {
            }
            override fun onStopTrackingTouch(p0: SeekBar?) {
            }
        })
    }
}
```

39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

```
blueBar.setOnSeekBarChangeListener(object : OnSeekBarChangeListener {
    override fun onProgressChanged(p0: SeekBar?, p1: Int, p2: Boolean) {
        blue = p1
        blueText.text = "블루값 : " + blue
    }
    override fun onStartTrackingTouch(p0: SeekBar?) {
    }
    override fun onStopTrackingTouch(p0: SeekBar?) {
    }
})

btnOk.setOnClickListener { it: View!
    dpView.setBackgroundColor(Color.rgb(red, green, blue))
    Toast.makeText(applicationContext, text = "버튼을 눌렀어요~~~~~",
        Toast.LENGTH_LONG).show()
}

btnOk.setOnClickListener(object : View.OnClickListener {
    override fun onClick(p0: View?) {
        dpView.setBackgroundColor(Color.rgb(red, green, blue))
        Toast.makeText(applicationContext, text = "버튼을 눌렀어요~~~~~",
            Toast.LENGTH_LONG).show()
    }
})

//onCreate() End
```

02 기타 위젯 활용

```

63     }
64     //onCreate() End
65     inner class NoNameSeekBar : OnSeekBarChangeListener {
66         override fun onProgressChanged(p0: SeekBar?, p1: Int, p2: Boolean) {
67             red = p1
68         }
69         override fun onStartTrackingTouch(p0: SeekBar?) {
70         }
71         override fun onStopTrackingTouch(p0: SeekBar?) {
72         }
73     }
74     inner class NoNameSeekBarListener : OnSeekBarChangeListener {
75         override fun onProgressChanged(p0: SeekBar?, p1: Int, p2: Boolean) {
76             red = p1
77             redText.text = "red 값 : " + red
78         }
79         override fun onStartTrackingTouch(p0: SeekBar?) {
80         }
81         override fun onStopTrackingTouch(p0: SeekBar?) {
82         }
83     }
84 }
85 }

```


02 기타 위젯 이벤트 처리 (람다식표현)

- 람다식표현 (OnClickListener (interface)는 abstract method 가 onClick하나임)
- ```
btnOk.setOnClickListener {
 dpView.setBackgroundColor(Color.rgb(red ,green,blue))
 Toast.makeText(applicationContext, "버튼을 눌렀어요", Toast.LENGTH_LONG).show()
}
```
- ```
btnOk.setOnClickListener (object: View.OnClickListener {
    override fun onClick(p0: View?) {
        dpView.setBackgroundColor(Color.rgb(red ,green,blue))
        Toast.makeText(applicationContext, "버튼을 눌렀어요", Toast.LENGTH_LONG).show()
    }
})
```
- ```
var ob = NoNameClick()
```
- ```
btnOk.setOnClickListener(ob)
```
- ```
Inner class NoNameClick : OnClickListener {
 override fun onClick(p0: View?) {
 dpView.setBackgroundColor(Color.rgb(red ,green,blue))
 Toast.makeText(applicationContext, "버튼을 눌렀어요", Toast.LENGTH_LONG).show()
 }
}
```
- ```
}
```

02 기타 위젯 이벤트 처리 (람다식표현 – 방법1)

- OnSeekBarChangeListener () (interface)는 **abstract method** 가 3개이므로 람다식을 사용못함
- redBar.setOnSeekBarChangeListener(SeekBar.OnSeekBarChangeListener의 객체)
- OnSeekBarChangeListener는 interface로 생성자를 이용해 객체를 만들수 없음.
- 내부 class 를 만드는데 OnSeekBarChangeListener를 구현하는 클래스 만들.
- 이 클래스의 객체를 만들어 인자값으로 사용
- `var seekList = NoNameSeekBarListener()`
`redBar.setOnSeekBarChangeListener(seekList)`

```
inner class NoNameSeekBarListener : OnSeekBarChangeListener {
    override fun onProgressChanged(p0: SeekBar?, p1: Int, p2: Boolean) {
        red = p1
        redText.text = "red 값 : " + red
    }
    override fun onStartTrackingTouch(p0: SeekBar?) {
    }
    override fun onStopTrackingTouch(p0: SeekBar?) {
    }
}
```

02 기타 위젯 이벤트 처리 (람다식표현 – 방법2)

- OnSeekBarChangeListener () (interface)는 **abstract method** 가 3개이므로 람다식을 사용못함
- greenBar.setOnSeekBarChangeListener(이름없는 클래스의 객체 직접 만들기)

```
greenBar.setOnSeekBarChangeListener(object : SeekBar.OnSeekBarChangeListener {
    override fun onProgressChanged(p0: SeekBar?, p1: Int, p2: Boolean) {
        green = p1
        greenText.text = "green 값 : " + green
    }
    override fun onStartTrackingTouch(p0: SeekBar?) {
    }
    override fun onStopTrackingTouch(p0: SeekBar?) {
    }
})
```

01 간단한 기능의 뷰 컨테이너

■ 스크롤뷰

```

java.lang.Object
└─ android.view.View
    └─ android.widget.ViewGroup
        └─ android.widget.FrameLayout
            └─ android.widget.ScrollView
  
```

스크롤뷰 계층도

- XML 코드 자체로 구성이 가능함
- 위젯이나 레이아웃이 화면에 넘칠 때 스크롤 효과를 낼 수 있음
- 스크롤뷰는 수직(위아래)으로 스크롤하는 기능
 - 수평(좌우)으로 스크롤하는 수평 스크롤 뷰(HorizontalScrollView)는 따로 존재함
- 스크롤뷰에는 단 하나의 위젯만 넣을 수 있음
 - 주로 스크롤뷰 안에 리니어레이아웃을 1개 넣고, 리니어레이아웃 안에 자신이 원하는 것을 여러 개 넣는 방법을 사용함

01 간단한 기능의 뷰 컨테이너

예제 6-13 스크롤뷰의 XML 코드

```

1 <ScrollView xmlns:android="http://www."
2   android:layout_width="fill_parent"
3   android:layout_height="fill_parent"
4   android:orientation="vertical" >
5
6   <LinearLayout
7     android:layout_width="fill_parent"
8     android:layout_height="fill_parent"
9     android:orientation="vertical" >
10
11     <Button
12       android:layout_width="match_parent"
13       android:layout_height="100dp"
14       android:text="버튼 1" />
15     ~~~ 생략(버튼 7개) ~~~
16   </LinearLayout>
17
18 </ScrollView>

```



01 간단한 기능의 뷰 컨테이너

■ 슬라이딩드로어

```
java.lang.Object
└ android.view.View
  └ android.widget.ViewGroup
    └ android.widget.SlidingDrawer
```

슬라이딩드로어 계층도

- 위젯을 서랍처럼 열어서 보여주거나 닫아서 감춤

– 슬라이딩드로어의 형태

```
<슬라이딩드로어 handle="핸들명" content="콘텐츠명" >
  <버튼 id="핸들명" /> // 서랍 손잡이 역할
  <리니어레이아웃 id="콘텐츠명">

    // 이곳이 필요한 위젯을 넣는 서랍 내부

  </리니어레이아웃>
</슬라이딩드로어>
```

01 간단한 기능의 뷰 컨테이너

- 슬라이딩드로어의 규칙
 - 슬라이딩드로어의 handle 속성에 지정된 이름과 슬라이딩드로어의 손잡이 역할을 하는 버튼의 id가 동일해야 함
 - 버튼 대신 이미지뷰나 이미지버튼으로 사용해도 상관없음
 - 슬라이딩드로어의 content 속성에 지정된 이름과 리니어레이아웃의 id도 동일해야 함
 - 리니어레이아웃이 아닌 다른 레이아웃도 가능함

01 간단한 기능의 뷰 컨테이너

예제 6-14 슬라이딩드로어의 XML 코드

```

1 <LinearLayout>
2     <TextView
3         android:text="여기는 서랍 밖입니다." />
4     <SlidingDrawer
5         android:content="@+id/content"
6         android:handle="@+id/handle" >
7
8         <Button
9             android:id="@+id/handle"
10            android:text="서랍 손잡이" />
11
12        <LinearLayout
13            android:id="@+id/content"
14            android:background="#00FF00"
15            android:gravity="center" >
16
17            <TextView
18                android:text="여기는 서랍 안입니다." />
19        </LinearLayout>
20    </SlidingDrawer>
21 </LinearLayout>

```



02 복잡한 기능의 뷰 컨테이너

■ 뷰플리퍼

```
java.lang.Object
└─ android.view.View
    └─ android.widget.ViewGroup
        └─ android.widget.FrameLayout
            └─ android.widget.ViewAnimator
                └─ android.widget.ViewFlipper
```

뷰플리퍼 계층도

- 안에 여러 개의 위젯을 배치하고 필요에 따라 화면을 왼쪽 또는 오른쪽으로 밀어서 위젯을 하나씩 보여주는 방식의 뷰 컨테이너
- 뷰플리퍼의 형태

```
<리니어레이아웃>
    <리니어레이아웃>
        // 왼쪽 또는 오른쪽으로 전환할 버튼 또는 이미지뷰
    </리니어레이아웃>
    <뷰플리퍼>

        // 한 번에 하나씩 보여줄 위젯 삽입

    </뷰플리퍼>
</리니어레이아웃>
```

02 복잡한 기능의 뷰 컨테이너

- 한 번에 보여줄 위젯이 여러 개라면 뷰플리퍼 안에 레이아웃을 여러 개 넣고 각 레이아웃에 필요 한 위젯을 배치함

예제 6-15 뷰플리퍼의 XML 코드

```

1 <LinearLayout>
2     <LinearLayout
3         android:orientation="horizontal" >
4         <Button
5             android:id="@+id/btnPrev"
6             android:text=" 이전화면 " />
7         <Button
8             android:id="@+id/btnNext"
9             android:text=" 다음화면 " />
10    </LinearLayout>
11    <ViewFlipper
12        android:id="@+id/viewFlipper1">
13        <LinearLayout
14            android:background="#ff0000" >
15            ~~~ 필요한 위젯 삽입 ~~~
16        </LinearLayout>
17        <LinearLayout
18            android:background="#00ff00" >
19            ~~~ 필요한 위젯 삽입 ~~~
20        </LinearLayout>
21        <LinearLayout
22            android:background="#0000ff" >
23            ~~~ 필요한 위젯 삽입 ~~~
24        </LinearLayout>
25    </ViewFlipper>
26 </LinearLayout>

```



02 복잡한 기능의 뷰 컨테이너

예제 6-16 뷰플리퍼의 Kotlin 코드

```
1  override fun onCreate(savedInstanceState: Bundle?) {
2      super.onCreate(savedInstanceState)
3      setContentView(R.layout.activity_main)
4
5      var btnPrev : Button
6      var btnNext : Button
7      var vFlipper : ViewFlipper
8
9      btnPrev = findViewById<Button>(R.id.btnPrev)
10     btnNext = findViewById<Button>(R.id.btnNext)
11     vFlipper = findViewById<ViewFlipper>(R.id.viewFlipper1)
12
13     btnPrev.setOnClickListener {
14         vFlipper.showPrevious()
15     }
16     ~~~ 생략(다음 화면 버튼 1개) ~~~
17 }
```

02 복잡한 기능의 뷰 컨테이너

▶ 직접 풀어보기 6-2

뷰플리퍼를 이용하여 자동 사진 보기 앱을 작성하라.

- 적절한 이미지 여러 장이 자동으로 넘어가는 앱을 만든다.
- <사진보기 시작>과 <사진보기 정지>를 만들고, <사진보기 시작>을 클릭하면 1초 단위로 화면이 자동으로 넘어가게 한다.
- 뷰플리퍼 안에 리니어레이아웃을 배치할 필요는 없고 직접 이미지 뷰가 나오면 된다.

HINT 화면 넘김 시작 메소드로 `startFlipping()`, 정지 메소드로 `stopFlipping()`, 화면 넘김 간격 메소드로 `setFlipInterval(밀리초)`을 사용한다.



그림 6-4 자동 사진 보기 앱

02 복잡한 기능의 뷰 컨테이너

■ 탭호스트

```
java.lang.Object
└─ android.view.View
    └─ android.widget.ViewGroup
        └─ android.widget.FrameLayout
            └─ android.widget.TabHost
```

탭호스트 계층도

- 여러 탭을 두고 각 탭을 클릭할 때마다 해당 화면이 나오도록 설정하는 뷰 컨테이너

– 탭호스트의 형태

```
<탭호스트 id="@android:id/tabhost">
  <리니어레이아웃>
    <탭위젯 id="@android:id/tabs" />
    <프레임레이아웃 id="@android:id/tabcontent">

        // 이곳에 각 탭스펙에 대응할 탭 화면(레이아웃)을 3개 삽입

    </프레임레이아웃>
  </리니어레이아웃>
</탭호스트>
```

02 복잡한 기능의 뷰 컨테이너

- 주의할 점
 - 탭호스트, 탭위젯, 프레임레이아웃은 id가 지정되어 있는데 이 지정된 id를 변경하지 않고 그대로 사용해야 안드로이드가 탭호스트의 구성을 인식함
 - 탭호스트, 리니어레이아웃, 탭위젯, 프레임레이아웃은 Kotlin 코드에서 특별히 접근할 일이 없음

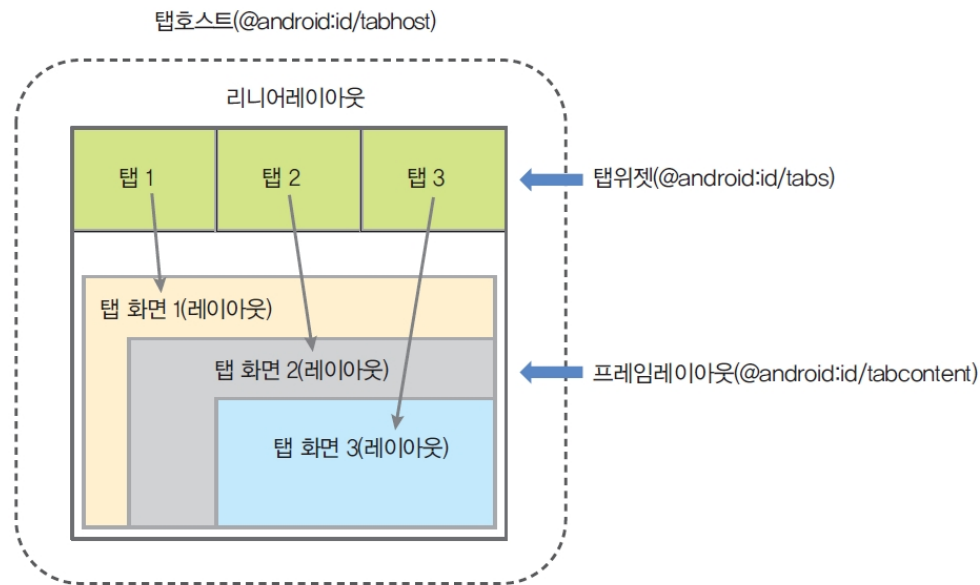


그림 6-5 탭호스트의 구성 방식

02 복잡한 기능의 뷰 컨테이너

- Kotlin 코드에서 할 작업
 - 탭위젯 안에 들어가는 3개의 탭을 생성하기
 - 각 탭과 대응되는 탭 화면(레이아웃) 연결하기
 - 탭을 생성 하고 탭 화면을 연결하기 위한 Kotlin 코드의 형식

```
var tabHost = this.tabHost // 탭호스트 변수 생성
// 탭스펙 생성
var tabSpec1 = tabHost.newTabSpec("TAB1").setIndicator("탭에 출력될 글자")
tabSpecSong.setContent(R.id.tab1) // 탭스펙을 탭과 연결
tabHost.addTab(tabSpec1) // 탭을 탭호스트에 부착
```

- 탭스펙(TabSpec)
 - 탭을 구성하는 요소들의 집합
 - 탭스펙을 준비하여 탭호스트에 붙여넣으면 탭이 됨

02 복잡한 기능의 뷰 컨테이너

예제 6-17 탭호스트의 XML 코드

```

1 <TabHost xmlns:android="http://schemas.android.com/apk/res/android"
2     android:id="@android:id/tabhost">
3     <LinearLayout>
4         <TabWidget
5             android:id="@android:id/tabs">
6         </TabWidget>
7         <FrameLayout
8             android:id="@android:id/tabcontent">
9             <LinearLayout
10                 android:id="@+id/tabSong"
11                 android:background="#f00000">
12             </LinearLayout>
13             <LinearLayout
14                 android:id="@+id/tabArtist"
15                 android:background="#f0f000">
16             </LinearLayout>
17             <LinearLayout
18                 android:id="@+id/tabAlbum"
19                 android:background="#f000ff">
20             </LinearLayout>
21         </FrameLayout>
22     </LinearLayout>
23 </TabHost>

```



02 복잡한 기능의 뷰 컨테이너

예제 6-18 탭호스트의 Kotlin 코드

```
1  ~~~ 생략(import 문) ~~~
2  @Suppress("deprecation")
3  class MainActivity : TabActivity() {
4
5      override fun onCreate(savedInstanceState: Bundle?) {
6          super.onCreate(savedInstanceState)
7          setContentView(R.layout.activity_main)
8
9          var tabHost = this.tabHost
10
11         var tabSpecSong = tabHost.newTabSpec("SONG").setIndicator("음악별")
12         tabSpecSong.setContent(R.id.tabSong)
13         tabHost.addTab(tabSpecSong)
14
15         var tabSpecArtist = tabHost.newTabSpec("ARTIST").setIndicator("가수별")
16         tabSpecArtist.setContent(R.id.tabArtist)
17         tabHost.addTab(tabSpecArtist)
18
19         var tabSpecAlbum = tabHost.newTabSpec("ALBUM").setIndicator("앨범별")
20         tabSpecAlbum.setContent(R.id.tabAlbum)
21         tabHost.addTab(tabSpecAlbum)
22
23         tabHost.currentTab = 0
24     }
25 }
```

02 복잡한 기능의 뷰 컨테이너

▶ 직접 풀어보기 6-3

탭호스트를 이용하여 동물 선택 앱을 작성하라.

- 탭위젯을 아래쪽에 배치하고 탭 4개가 나오게 한다.
- 프레임레이아웃 안의 리니어레이아웃 3개를 제거하고 4개의 이미지 뷰로 배치한다.

HINT 프레임레이아웃의 layout_weight 속성을 1로 한다.

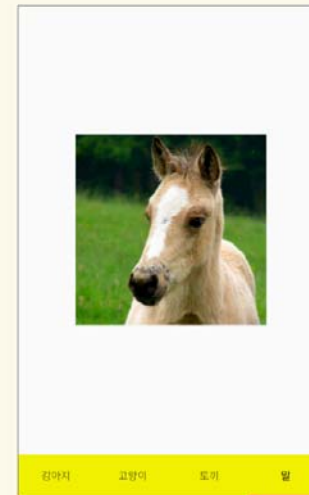


그림 6-6 동물 선택 앱

02 복잡한 기능의 뷰 컨테이너

■ 액션바와 프래그먼트

- 액션바
 - 허니콤(Android 3.0, API 11)부터 지원됨
 - 태블릿과 같은 대형 화면에서 여러 화면을 사용하기 위해 고안됨
 - 태블릿, 스마트폰 등 다양한 크기의 화면을 디자인하는데 활용할 수 있고 메뉴를 대체하여 사용할 수도 있음
 - 액션바를 구현할 때는 프래그먼트(fragment)를 사용하는 것이 좋음
- 프래그먼트
 - 허니콤에서 소개된 기능으로 대형 태블릿의 화면을 효율적으로 사용하기 위함
 - 액티비티보다 작은 단위의 화면
 - 대형 화면에서 액티비티 화면을 분할하여 표현할 수 있음
 - 소형 화면에서는 화면의 분할 보다는 실행 중에 화면을 동적으로 추가하거나 제거하는 데 더 많이 활용됨

02 복잡한 기능의 뷰 컨테이너

- 액션바와 프래그먼트를 활용하여 구현할 화면
 - 액션바를 생성한 후 탭(Tab) 위젯을 액션바에 등록하는 방법을 사용함
 - MainActivity에 TabListener 인터페이스를 구현해야 함

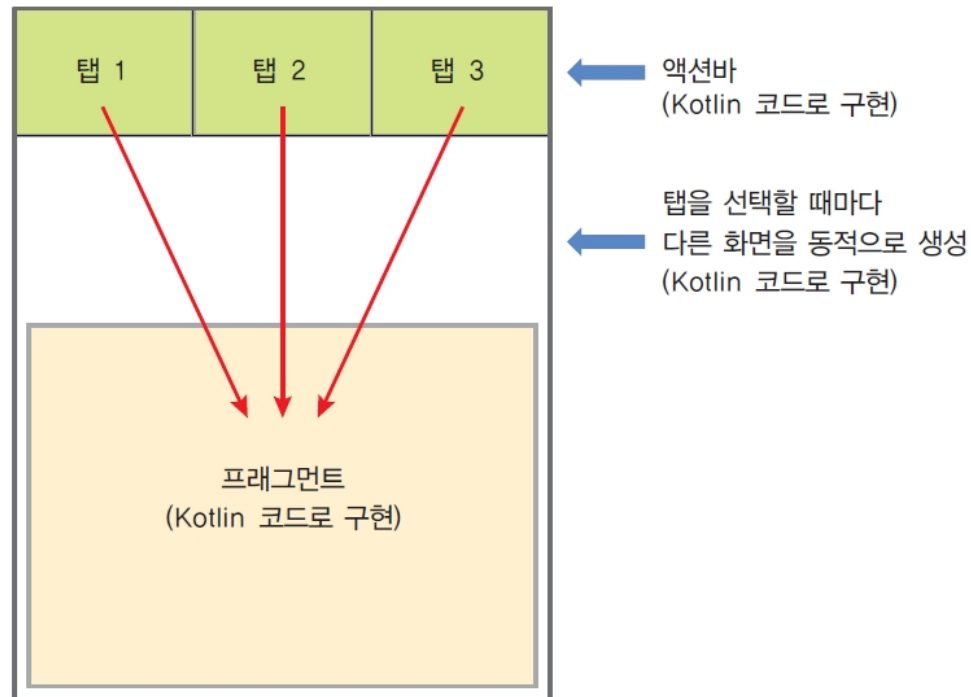


그림 6-7 액션바와 프래그먼트의 구성 방식

02 복잡한 기능의 뷰 컨테이너

- 액션바에 탭을 추가하는 Kotlin 코드의 형식

```
var bar = this.supportActionBar // 상단에 표시할 액션바 준비
// 탭호스트와 같이 탭의 모양이 되도록 설정
bar!!.navigationMode = ActionBar.NAVIGATION_MODE_TABS
tabSong = bar.newTab() // 액션바에 탭 생성
tabSong.text = "음악별" // 탭의 글자 설정
tabSong.setTabListener(this) // 탭을 터치하면 작동하는 리스너 지정
bar.addTab(tabSong) // 액션바에 탭 추가
```

- [예제 6-19]의 액션바는 XML 코드 없이 작성할 것이므로 activity_main.xml 파일을 삭제해도 무관함

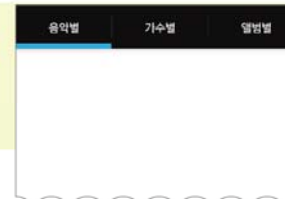
02 복잡한 기능의 뷰 컨테이너

예제 6-19 액션바의 Kotlin 코드 1

```

1  ~~~ 생략(import 문) ~~~
2  @Suppress("deprecation")
3  class MainActivity : AppCompatActivity(), ActionBar.TabListener {
4
5      lateinit var tabSong : ActionBar.Tab
6      lateinit var tabArtist : ActionBar.Tab
7      lateinit var tabAlbum : ActionBar.Tab
8
9      override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         var bar = this.supportActionBar
12         bar!!.navigationMode = ActionBar.NAVIGATION_MODE_TABS
13
14         tabSong = bar.newTab()
15         tabSong.text = "음악별"
16         tabSong.setTabListener(this)
17         bar.addTab(tabSong)
18         ~~~ 생략(Artist 탭, Album 탭) ~~~
19     }
20     override fun onTabSelected(tab:ActionBar.Tab, ft:FragmentTransaction) {}
21     override fun onTabUnselected(tab:ActionBar.Tab, ft:FragmentTransaction) {}
22     override fun onTabReselected(tab:ActionBar.Tab, ft:FragmentTransaction) {}
23
24 }

```



02 복잡한 기능의 뷰 컨테이너

- 탭을 클릭하면 아래쪽에 해당 프래그먼트가 나오도록 코딩

예제 6-20 액션바의 Kotlin 코드 2

```
1 class MyTabFragment : androidx.fragment.app.Fragment() {
2     var tabName : String? = null
3     override fun onCreate(savedInstanceState: Bundle?) {
4         super.onCreate(savedInstanceState)
5         var data = arguments
6         tabName = data!!.getString("tabName")
7     }
8
9     override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
10         savedInstanceState: Bundle?): View? {
11         var params = LinearLayout.LayoutParams(LinearLayout.LayoutParams.
12             MATCH_PARENT, LinearLayout.LayoutParams.MATCH_PARENT)
13
14         var baseLayout = LinearLayout(super.getActivity())
15         baseLayout.orientation = LinearLayout.VERTICAL
16         baseLayout.layoutParams = params
17
18         if (tabName === "음악별") baseLayout.setBackgroundColor(Color.RED)
19         if (tabName === "가수별") baseLayout.setBackgroundColor(Color.GREEN)
20         if (tabName === "앨범별") baseLayout.setBackgroundColor(Color.BLUE)
21
22         return baseLayout
23     }
24 }
```

02 복잡한 기능의 뷰 컨테이너

- 멤버변수(전역변수)로 프래그먼트 배열 변수를 추가하고, onTabSelected() 메소드를 코딩

예제 6-21 액션바의 Kotlin 코드 3

```

1  var myFrgs = arrayOfNulls<MyTabFragment>(3)
2  ~~~ 생략(onCreate() 메소드) ~~~
3  override fun onTabSelected(tab: ActionBar.Tab, ft: FragmentTransaction) {
4      var myTabFrag : MyTabFragment? = null
5
6      if (myFrgs[tab.position] == null) {
7          myTabFrag = MyTabFragment()
8          val data = Bundle()
9          data.putString("tabName", tab.text.toString())
10         myTabFrag.arguments = data
11         myFrgs[tab.position] = myTabFrag
12     } else
13         myTabFrag = myFrgs[tab.position]
14
15     ft.replace(android.R.id.content, myTabFrag!!)
16 }

```



02 복잡한 기능의 뷰 컨테이너

■ 웹뷰

```
java.lang.Object
└─ android.view.View
    └─ android.widget.ViewGroup
        └─ android.widget.AbsoluteLayout
            └─ android.webkit.WebView
```

웹뷰 계층도

- 사용자가 웹브라우저 기능을 앱 안에 직접 포함할 수 있는 위젯
- 웹뷰를 실제 웹브라우저처럼 만들려면 아주 많은 작업이 필요함

02 복잡한 기능의 뷰 컨테이너

■ <실습 6-2> 간단 웹브라우저 만들기

- 에디트텍스트에 URL을 입력하면 해당 링크로 이동하는 앱 만들기
- 화면의 로고와 프로그램의 아이콘도 변경하기

■ 안드로이드 프로젝트 생성

- (1) 새 프로젝트를 만들기
 - 프로젝트 이름 : 'Project6_2'
 - 패키지 이름 : 'com.cookandroid.project6_2'
 - 그 외 규칙은 [실습 2-4]의 (1)~(4)를 따름



그림 6-8 간단 웹브라우저 앱 결과 화면

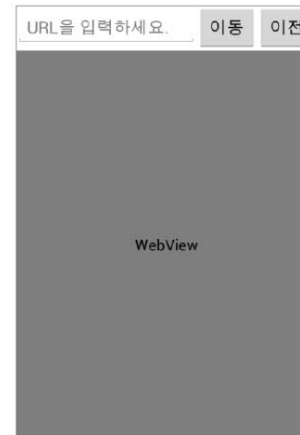
02 복잡한 기능의 뷰 컨테이너

- 화면 디자인 및 편집
 - (2) [app]-[res]-[layout]-[activity_main.xml]을 열고, 아래쪽의 [Text] 탭을 클릭하여 화면을 코딩함. 다음과 같이 화면을 구성함
 - 리니어레이아웃을 하나 더 만들기
 - » 그 안에 에디트텍스트 1개와 버튼 2개 구성
 - 하단에 웹뷰를 만들기
 - 각 위젯의 id는 edtUrl, btnGo, btnBack, webView1

02 복잡한 기능의 뷰 컨테이너

예제 6-22 activity_main.xml

```
1 <LinearLayout>
2     <LinearLayout>
3         <EditText
4             android:id="@+id/edtUrl"
5             android:layout_weight="1"
6             android:singleLine="true" >
7     </EditText>
8     <Button
9         android:id="@+id/btnGo"
10        android:text="이동" />
11    <Button
12        android:id="@+id/btnBack"
13        android:text="이전" />
14 </LinearLayout>
15 <WebView
16     android:id="@+id/webView1" />
17 </LinearLayout>
```



02 복잡한 기능의 뷰 컨테이너

- (3) 로고의 타이틀과 아이콘을 변경하기 위해 메니페스트 파일 수정
 - AndroidManifest.xml(간략히 매니페스트라고도 함) : 프로젝트의 전반적인 환경을 설정하는 파일
 - [app]-[manifests]-[AndroidManifest.xml]을 열기

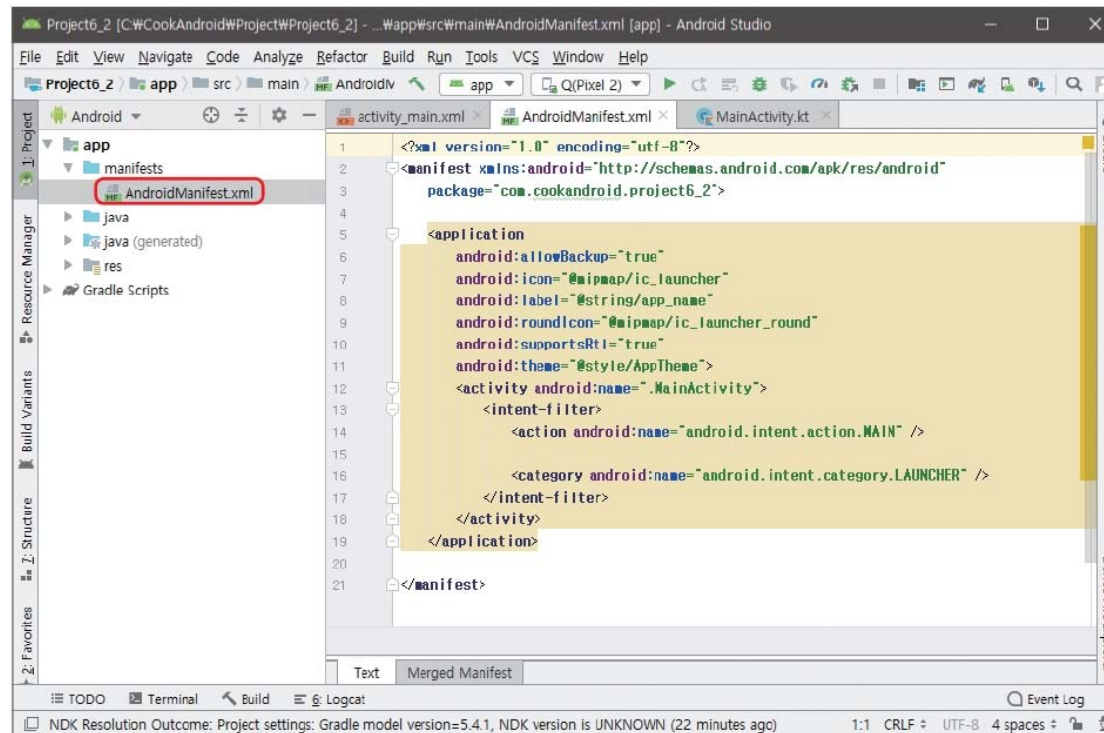


그림 6-9 AndroidManifest.xml 파일 열기

02 복잡한 기능의 뷰 컨테이너

- (4) 메니페스트 파일을 수정함

예제 6-23 AndroidManifest.xml

```
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2     package="com.cookandroid.project6_2">
3
4     <application
5         android:allowBackup="true"
6         android:icon="@drawable/emo_im_cool"
7         android:label="쿡북 웹브라우저 "
8         android:logo="@drawable/web"
9         android:theme="@style/AppTheme" >
10
11         <activity
12             android:name=".MainActivity"
13             android:label="간단 웹브라우저" >
14
15             <intent-filter>
16                 <action android:name="android.intent.action.MAIN" />
17
18                 <category android:name="android.intent.category.LAUNCHER" />
19             </intent-filter>
20         </activity>
21     </application>
22 </manifest>
```

02 복잡한 기능의 뷰 컨테이너

- (5) 프로그램 정보를 확인함
 - 아이콘을 꾹누르고 [앱 정보]를 선택하면 설치된 앱 정보를 확인할 수 있음

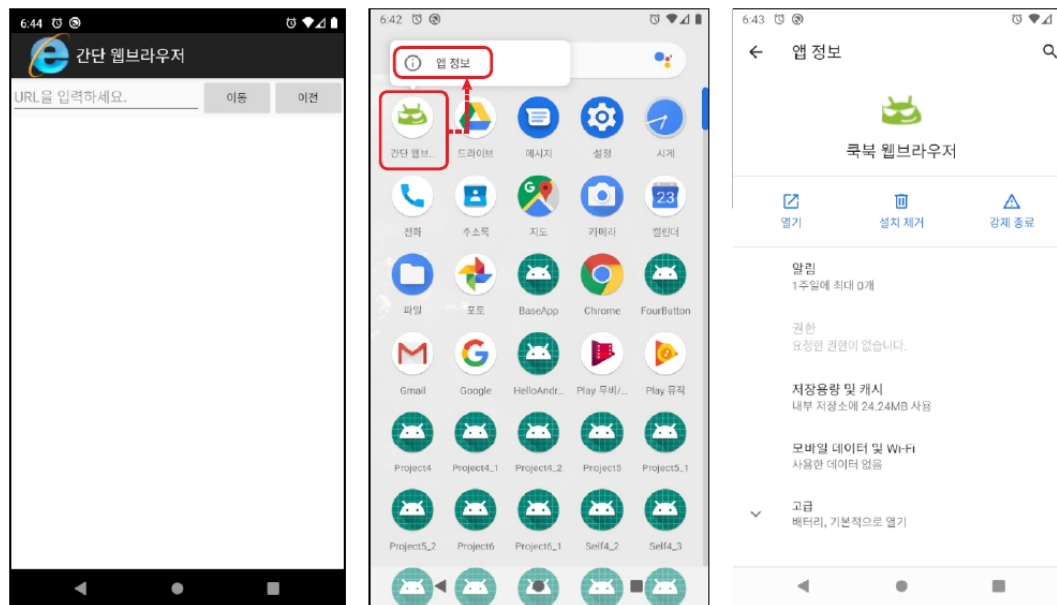


그림 6-10 매니페스트에서 설정한 내용 확인

02 복잡한 기능의 뷰 컨테이너

- Kotlin 코드 작성 및 수정
 - (6) [app]-[java]-[패키지 이름]-[MainActivity]를 열기
 - (7) activity_main.xml의 4개 위젯에 대응할 위젯 변수 4개를 전역변수로 선언하고, onCreate() 메소드 안의 각 변수에 위젯을 대입함

예제 6-24 Kotlin 코드 1

```
1  ~~~ 생략(import 문) ~~~
2  class MainActivity : AppCompatActivity() {
3      lateinit var edtUrl : EditText
4      lateinit var btnGo : Button
5      lateinit var btnBack : Button
6      lateinit var web : WebView
7
8      override fun onCreate(savedInstanceState: Bundle?) {
9          super.onCreate(savedInstanceState)
10         setContentView(R.layout.activity_main)
11
12         edtUrl = findViewById<EditText>(R.id.edtUrl)
13         btnGo = findViewById<Button>(R.id.btnGo)
14         btnBack = findViewById<Button>(R.id.btnBack)
15         web = findViewById<WebView>(R.id.webView1)
16     }
17
18 }
```


02 복잡한 기능의 뷰 컨테이너

- (8) WebViewClient의 상속을 받는 CookWebViewClient 클래스를 정의함

예제 6-25 Kotlin 코드 2

```
1 class CookWebViewClient : WebViewClient() {
2
3 }
```

- (9) CookWebViewClient 클래스 안에 커서를 두고 [Code]-[Override Methods]를 선택하면 부모 클래스에서 오버라이딩 또는 임플리먼트가 가능한 메소드 목록을 확인할 수 있음

- 메소드 중에서
should OverrideUrlLoading(view: WebView!, url: String!): Boolean
메소드를 선택하고 <OK>를 클릭

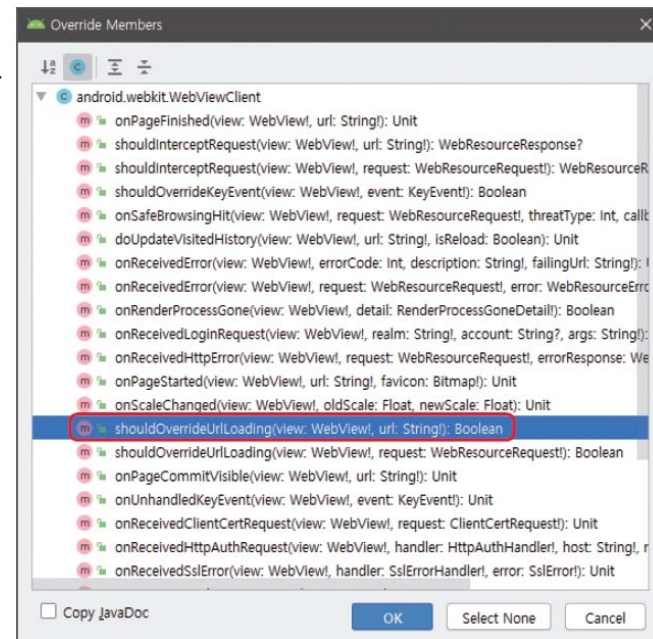


그림 6-11 메소드의 자동 오버라이딩

02 복잡한 기능의 뷰 컨테이너

예제 6-26 Kotlin 코드 3

```
1      ~~~ 생략([예제 6-24]와 동일) ~~~
2      web = findViewById<WebView>(R.id.webView1)
3  }
4
5  class CookWebViewClient : WebViewClient() {
6      override fun shouldOverrideUrlLoading (view: WebView?,
7                                              url: String?) : Boolean {
8          return super.shouldOverrideUrlLoading(view, url)
9      }
10 }
11 }
```

02 복잡한 기능의 뷰 컨테이너

- (10) onCreate() 메소드 안에 다음 내용을 계속 코딩
 - 생성한 CookWebViewClient 클래스를 웹뷰에 설정함
 - 화면 확대/축소 아이콘이 보이도록 설정함
 - 두 버튼을 클릭하면 입력한 URL로 이동하거나 이전 화면으로 돌아가게 함

예제 6-27 Kotlin 코드 4

```
1 web.webViewClient = CookWebViewClient()
2
3 var webSet = web.settings
4 webSet.builtInZoomControls = true
5
6 btnGo.setOnClickListener {
7     web.loadUrl edtUrl.text.toString()
8 }
9
10 btnBack.setOnClickListener {
11     web.goBack()
12 }
```

02 복잡한 기능의 뷰 컨테이너

- 4 프로젝트 실행 및 결과 확인
 - (11) 완성된 코드를 실행해보기
 - 에디트텍스트에 'https://m.daum.net'을 입력하고 <이동>을 클릭하면 웹페이지가 열리지 않음
 - 프로젝트에 인터넷을 사용할 수 있는 퍼미션을 주지 않았기 때문
 - 퍼미션은 AndroidManifest.xml에서 지정해야 함



그림 6-12 웹페이지가 열리지 않음

02 복잡한 기능의 뷰 컨테이너

- [반복] 화면 디자인 및 편집
 - (12) AndroidManifest.xml을 열고 <application> 행 위에 다음 예제의 4행을 추가
 - 4행 : 프로젝트의 인터넷 사용을 허가한다는 의미
 - AndroidManifest.xml은 프로젝트마다 별도로 생성되는 파일로 인터넷을 사용하는 프로젝트마다 AndroidManifest.xml에 이 행을 추가해야 함

예제 6-28 AndroidManifest.xml

```

1  ~~~ 생략 ~~~
2
3
4  <uses-permission android:name="android.permission.INTERNET" />
5
6  <application
7      android:icon="@drawable/emo_im_cool"
8  ~~~ 생략 ~~~

```

- [반복] 프로젝트 실행 및 결과 확인
 - (13) 모두 저장한 다음 프로젝트를 다시 실행해보면 [그림 6-8]과 같이 정상적으로 웹서핑이 가능함