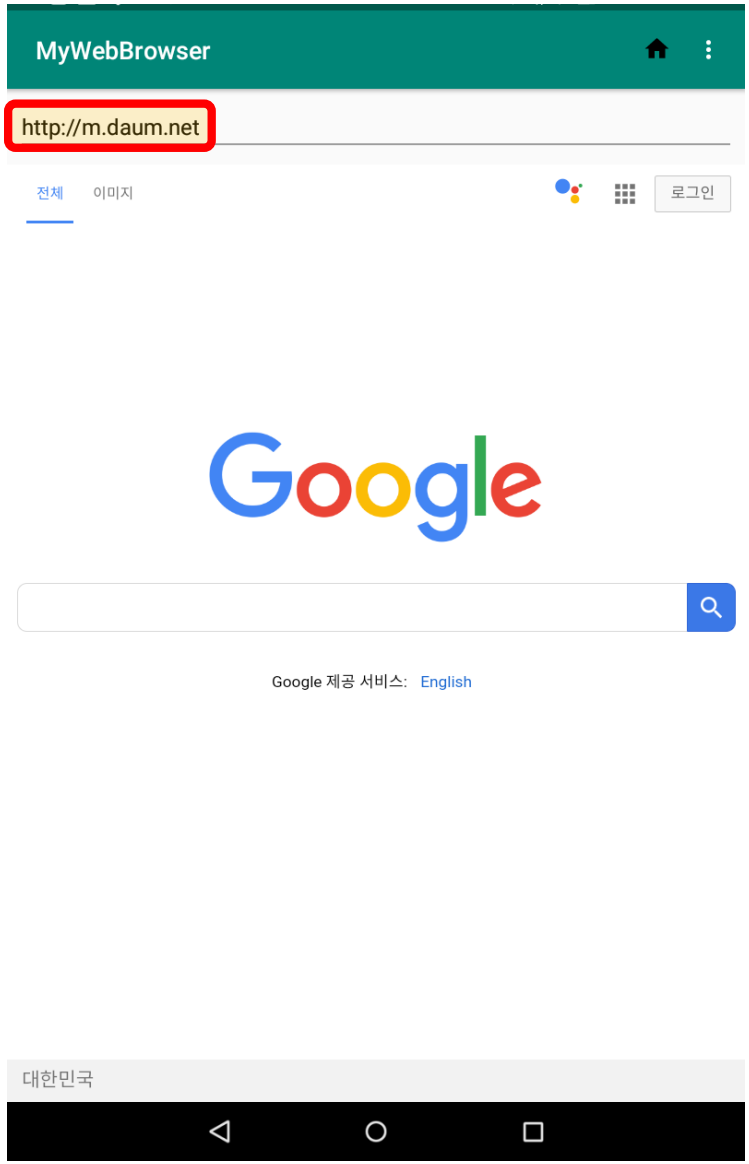
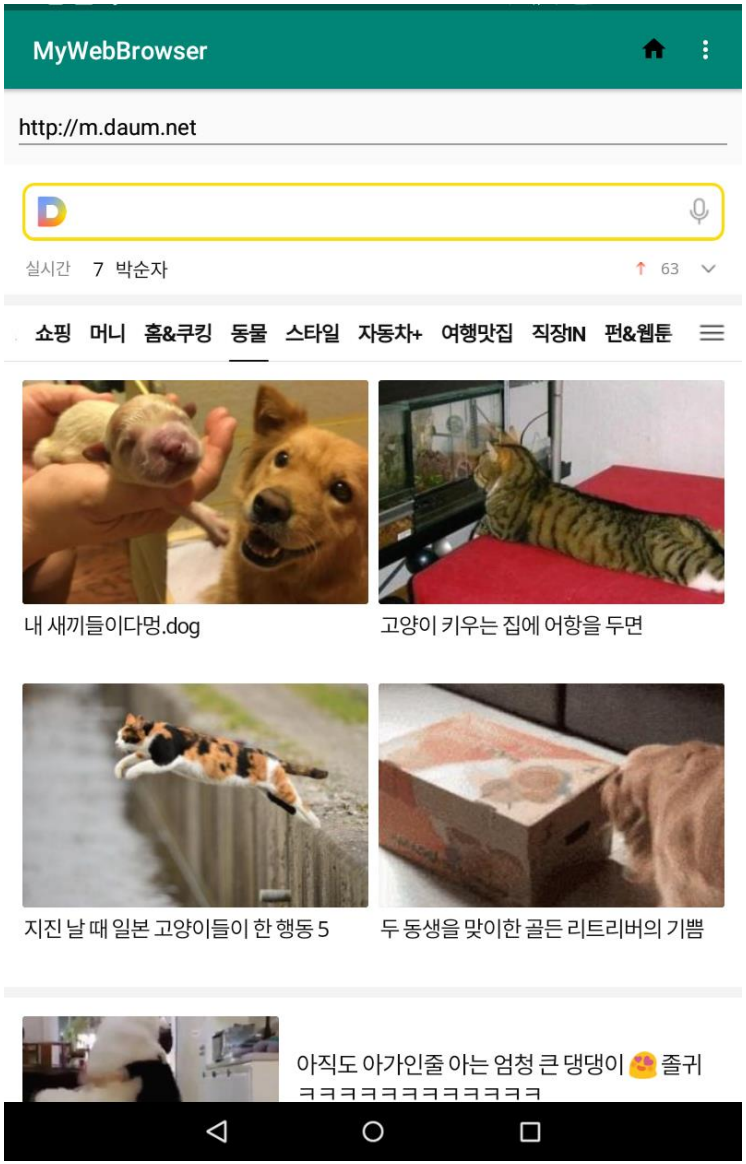
Kotlin을 이용한 Android 프로그래밍

웹 브라우저 만들기

Contents

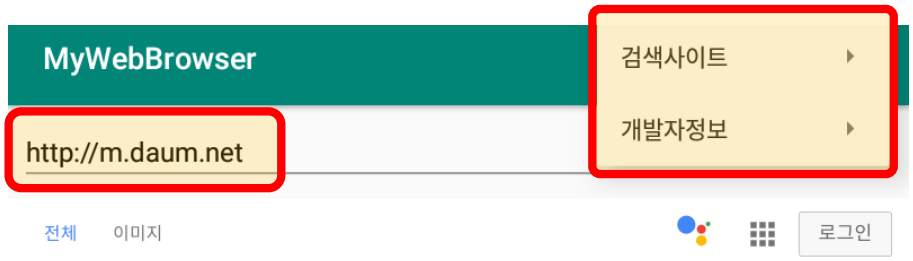
나만의 웹 브라우저

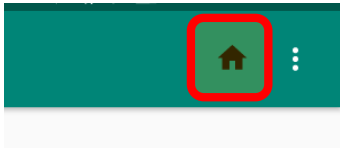

MyWebBrowser
http://m.daum.net
전체
이미지
로그인
Google
Google 제공 서비스: English
대한민국

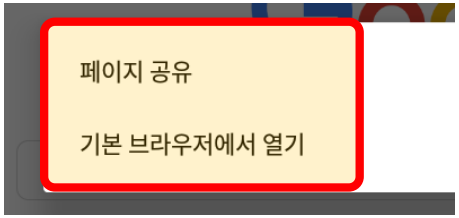

MyWebBrowser
http://m.daum.net
실시간 7 박순자
63
쇼핑 머니 홈&쿠킹 동물 스타일 자동차+ 여행맛집 직장IN 펀&웹툰
내 새끼들이다멍.dog
고양이 키우는 집에 어항을 두면
지진 날 때 일본 고양이들이 한 행동 5
두 동생을 맞이한 골든 리트리버의 기쁨
아직도 아가인줄 아는 엄청 큰 댕댕이 졸귀

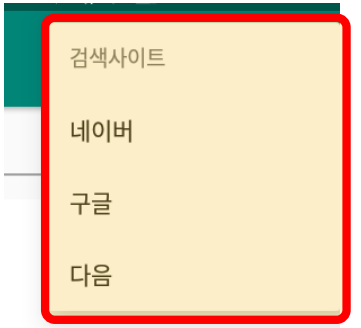
나만의 웹 브라우저

- 프로젝트 명 : MyWebBrowser
- 기능
  - 웹 페이지를 탐색
  - 홈 메뉴를 클릭하여 첫 페이지로 이동
  - 옵션 메뉴에는 검색 사이트와 개발자 정보가 표시
  - 페이지를 문자나 메일로 공유
- 구성요소
  - WebView : 웹페이지를 표시하는 뷰
  - 옵션 메뉴 : 상단 툴바에 표시하는 메뉴
  - 컨텍스트 메뉴 : 뷰를 롱클릭하면 표시되는 메뉴
  - 암시적 인텐트 : 문자 보내기, 이메일 보내기와 같이 미리 정의된 인텐트
- 라이브러리 설정
  - Anko 라이브러리 : 인텐트, 다이얼로그, 로그 등을 효율적으로 구현하도록 제공









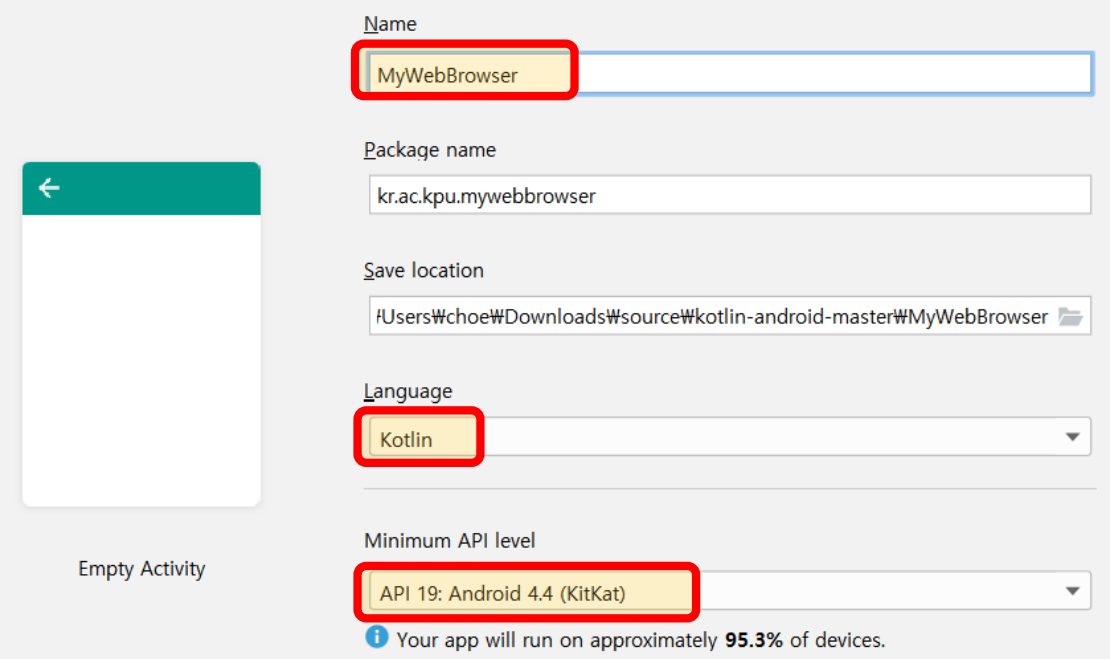
나만의 웹 브라우저

- 프로젝트 설계
  - 웹을 표시하는 화면 구성
  - 옵션 메뉴와 컨텍스트 메뉴를 추가
  - 웹 페이지 주소를 공유하는 방법으로 암시적 인텐트 사용
- 구현 순서

1) 프로젝트 생성 및 안드로이드 설정
2) 테마 수정
3) 기본 웹 브라우저 기능 구현
4) 옵션 메뉴 추가
5) 컨텍스트 메뉴 추가
6) 암시적 인텐트 사용

나만의 웹 브라우저

- 프로젝트 생성
  - 프로젝트 명 : MyWebBrowser
  - minSdkVersion : 19
  - 기본 액티비티 : Empty Activity



- 이전 강의자료를 참고하여 Anko 라이브러리 주입

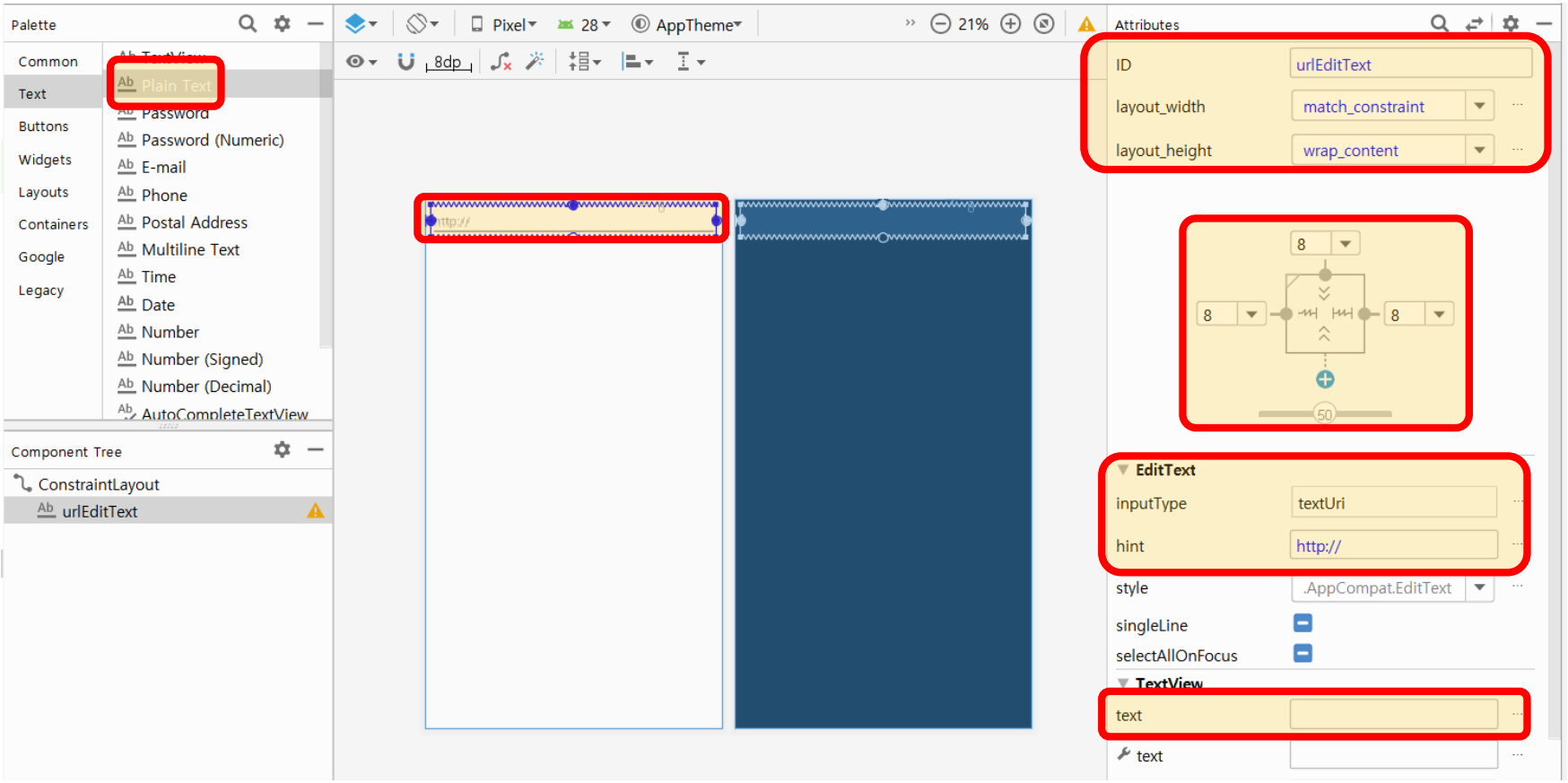
나만의 웹 브라우저

- 테마 수정
  - 웹 브라우저 기능
    - url 주소 입력
    - 검색 버튼을 클릭하면 웹 페이지 표시
    - 홈 아이콘을 클릭하면 미리 지정한 홈페이지로 이동
    - 옵션 메뉴의 하위 메뉴로 검색 사이트와 개발자 정보로 구성
    - 개발자 정보에서는 전화 걸기, 문자 보내기 등이 가능
    - 웹 화면을 길게 클릭하면 컨텍스트 메뉴가 표시
  - 구현 순서
    - 검색 창으로 사용할 EditText 배치
    - WebView 배치

나만의 웹 브라우저

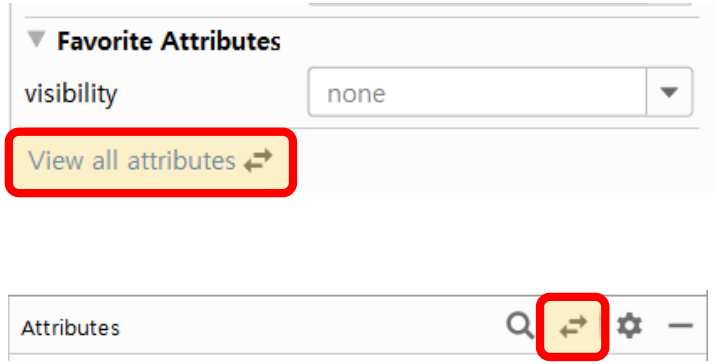
검색 창 EditText 배치

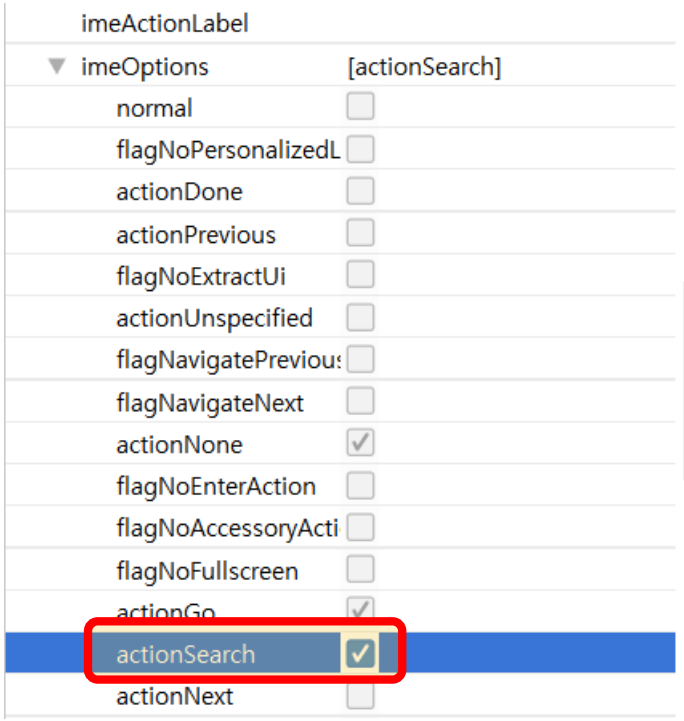
웹 페이지 주소만 입력 받도록 inputType을 textUri로 설정



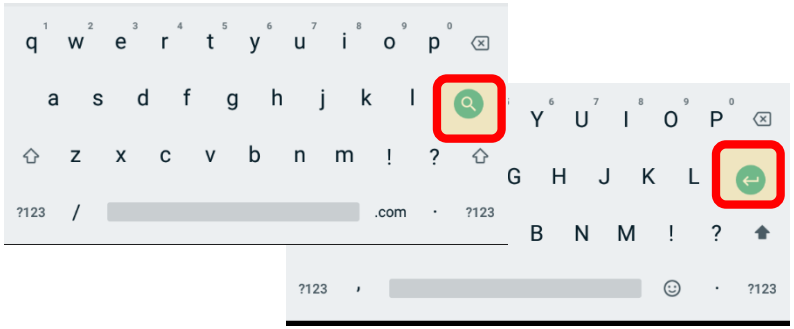
나만의 웹 브라우저

- 소프트 키보드에 돋보기 아이콘을 추가
  - View all attributes - imeOptions 속성 펼치기 - actionSearch 체크





  - Enter 키가 돋보기로 변경
    - 아이콘만 변경되기 때문에 별도로 동작을 구현



나만의 웹 브라우저

- WebView 배치
  - 에디트 텍스트의 아래 영역을 웹 뷰로 채우고 페이지를 표시
  - AutoConnect 모드를 off로 변경하고 웹 뷰를 추가
    - 이때는 가이드 선이 없는 곳으로 적당히 배치하여도 자동으로 아래 영역에 **가득 차도록 배치**됨
    - On상태에서 배치하면 레이아웃과 제약이 발생
  - 자동 제약 설정 아이콘을 클릭하여 제약을 자동추가 시킴

나만의 웹 브라우저

- 기본 브라우저 기능 구현
  - 웹 페이지를 표시하기 위하여 인터넷 권한 설정
    - 안드로이드는 특정 권한이 필요한 동작을 할 경우 해당 권한을 추가해야 함
    - 인터넷 사용은 과금을 유발할 수 있으므로 권한이 필요
  - 구현 순서
    - 인터넷 권한 설정
    - 웹 뷰에 웹 페이지 표시
    - 키보드의 검색 버튼 동작 정의
    - 뒤로 가기 동작 재정의

나만의 웹 브라우저

- 인터넷 권한 설정
  - 해당 권한이 추가된 앱은 플레이스토어에서 다운로드하여 설치되면 '인터넷 권한을 사용하는 앱' 이라고 표기되어 사용자가 해당내용을 인지하여 설치하도록 함
  - 메니페스트에서 인터넷 권한을 추가
    - <uses-permission android:name="android.permission.INTERNET" />
    - <manifest>태그 내부에 추가 - 코드 자동 완성으로도 가능

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3          package="kr.ac.kpu.mywebbrowser">
4
5      <uses-permission android:name="android.permission.INTERNET" />
6
7      <application
8          android:allowBackup="true"
```

나만의 웹 브라우저

▶웹 뷰에 웹 페이지 표시

▶웹 뷰 기본 설정과 구글 페이지를 코딩하는 코드를 작성

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

    webView.apply { this: WebView!
        settings.javaScriptEnabled = true
        webViewClient = WebViewClient()
    }
    webView.loadUrl( url: "http://www.google.com")
}
```

//웹 뷰 기본 설정 : 아래 두가지는 반드시 설정

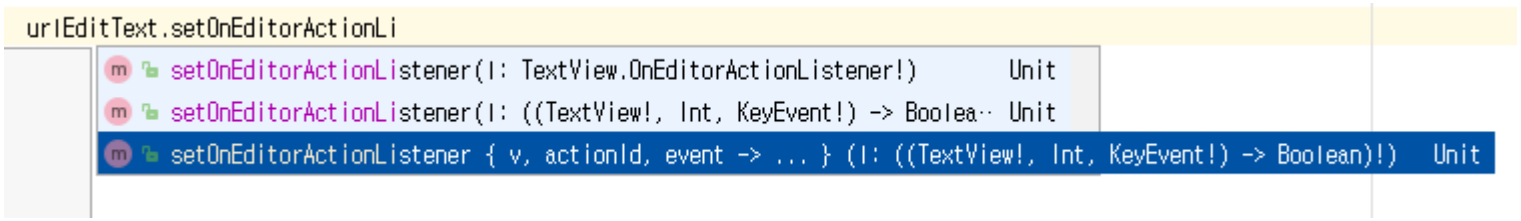
// 1)자바 스크립트를 동작시키도록 javaScriptEnabled 기능을 활성화해야 함

// 2)webViewClient를 클래스로 지정하지 않으면 웹뷰가 아니고 자체 웹 브라우저가 동작하게 됨

// loadUrl()메서드를 사용하여 “http:”가 포함된 Url을 전달하면 웹뷰에 해당 페이지가 로딩

나만의 웹 브라우저

- 키보드의 검색 버튼 동작 구현
  - 웹 뷰의 검색 창에 URL을 입력하고 소프트 키보드의 검색 아이콘을 클릭하여 웹 페이지가 웹 뷰에 표시되도록 정의
  - 자동 완성




```
webView.loadUrl( url: "http://www.google.com")
                        //반응하는 인자로는 뷰, 액션ID, 이벤트이고 사용하지 않을 경우 _ 로 표기
urlEditText.setOnEditorActionListener { _, actionId, _ -> //에디트텍스트에 글자가 입력될때마다 호출
    if (actionId == EditorInfo.IME_ACTION_SEARCH) { //androidId값이 EditorInfo클래스의 검색 버튼에 해당하는
        webView.loadUrl(urlEditText.text.toString())  상수와 비교하여 검색 버튼이 눌렸는지 확인
         ^setOnEditorActionListener true              //검색 창에 입력한 주소를 웹부에 전달하여 로딩하고
    } else {                                          true를 반환하여 종료
         ^setOnEditorActionListener false
    }
}
```

  - 검색 창에 http:// 를 포함하여 입력한 URL로 이동하면 성공적으로 동작 구현

나만의 웹 브라우저

- 뒤로 가기 동작 재정의
  - 뒤로 가기 기능을 구현하지 않고 뒤로 가기 키를 누르면 웹 뷰가 종료 됨
  - onBackPressed() 메소드를 오버라이드(override)

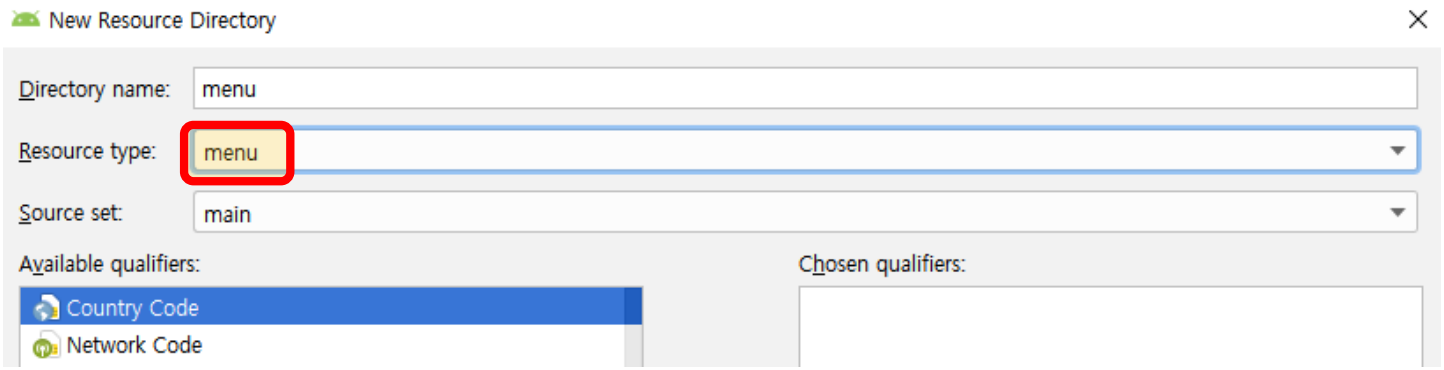
```
override fun onBackPressed() {
    if (webView.canGoBack()) {  //웹 뷰가 이전페이지로 갈수 있다면
        webView.goBack()  //이전 페이지로 이동하고
    } else {
        super.onBackPressed() //그럴 수 없다면 원래 동작을 수행. 즉, 종료
    }
}
```

나만의 웹 브라우저

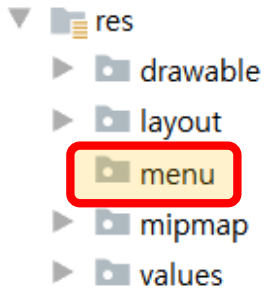
- 옵션 메뉴 사용하기
  - 상단 툴바에 표시되는 메뉴를 옵션 메뉴라고 함
  - 옵션 메뉴의 구현 순서
    - 메뉴 리소스 준비
    - onCreateOptionsMenu()메서드를 재정의하여 원하는 메뉴를 추가하고 true를 반환
    - onOptiosItemSelected()메서드를 재정의하여 메뉴 아이템이 선택되었을 경우 처리를 분기

나만의 웹 브라우저

- 메뉴 리소스 파일 생성 및 벡터 이미지 준비
  - 안드로이드의 모든 메뉴는 메뉴 리소스 작성부터 시작
  - File - New - Android Resource Directory 클릭
  - menu 선택



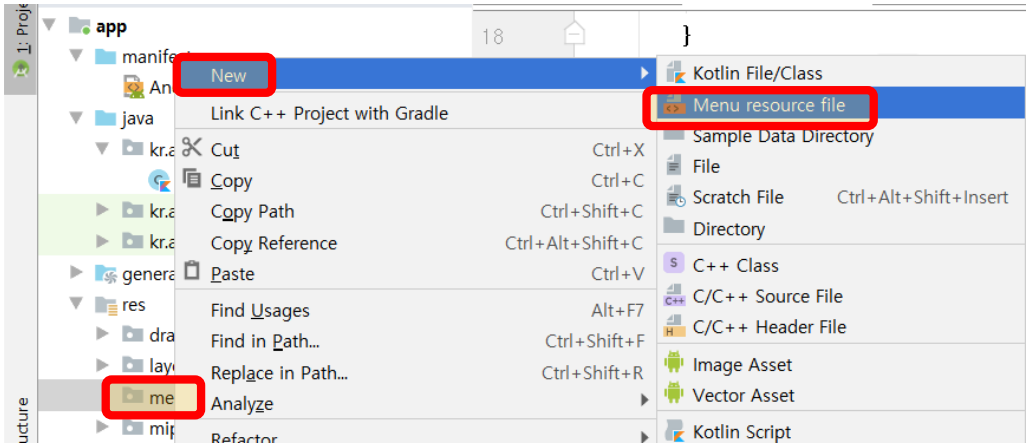
  - 프로젝트 창에 menu 폴더 추가 확인



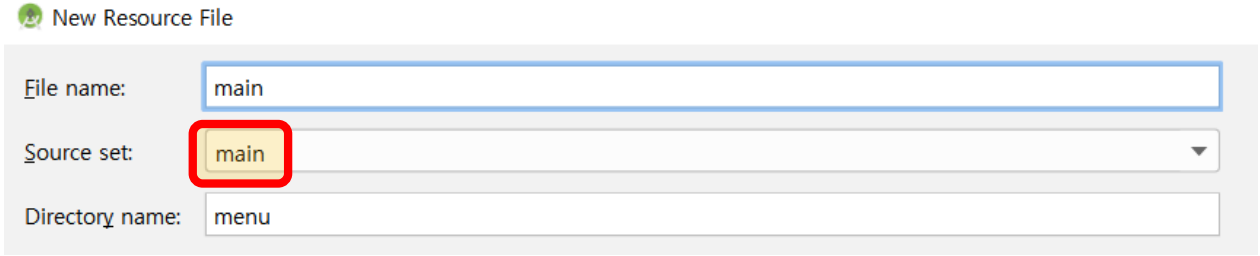
나만의 웹 브라우저

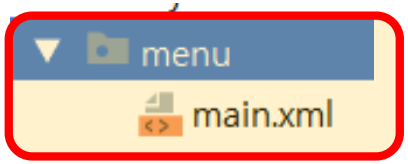
메뉴 리소스 파일 생성 및 벡터 이미지 준비

메뉴 폴더에서 마우스 우 클릭 후 Menu resource file 선택



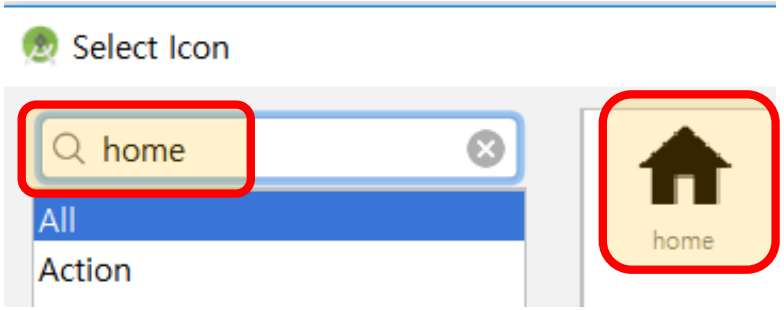
main.xml을 만들기 위하여 새로운 리소스 파일을 생성하는 화면에서 File name에 main을 입력





나만의 웹 브라우저

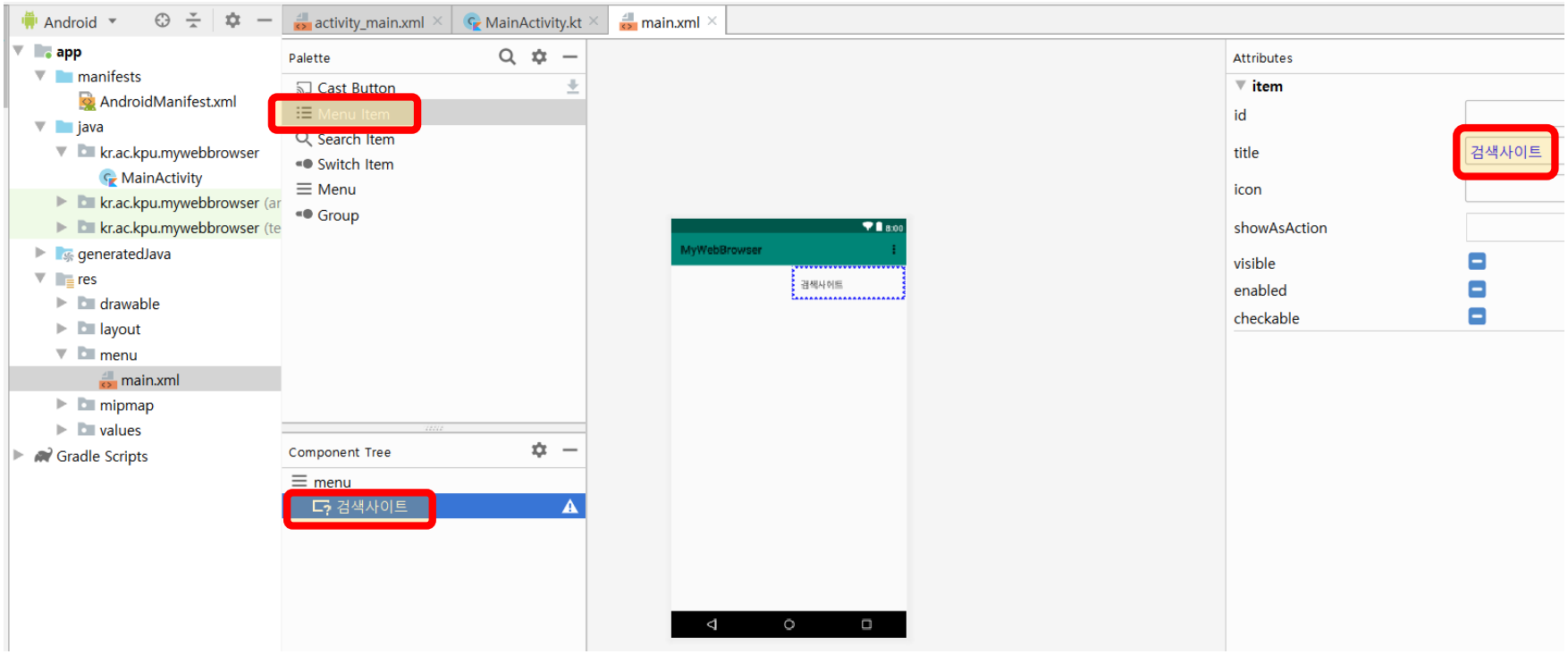
- 메뉴 리소스 파일 생성 및 벡터 이미지 준비
  - 메뉴에 사용할 벡터 이미지 준비



나만의 웹 브라우저

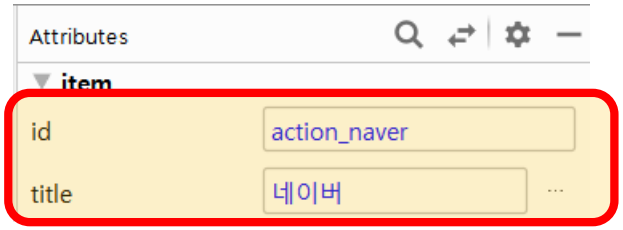
메뉴 작성

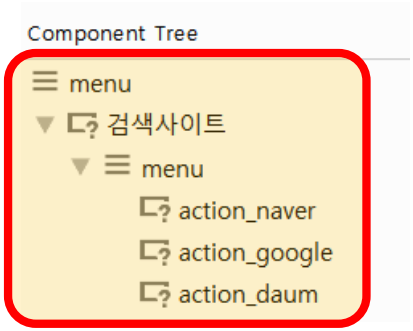
- main.xml 파일을 클릭하면 메뉴 에디터가 표시되며 팔레트 창에서 Menu Item을 선택하여 컴포넌트 트리 창에 있는 menu아래에 드래그 하여 배치

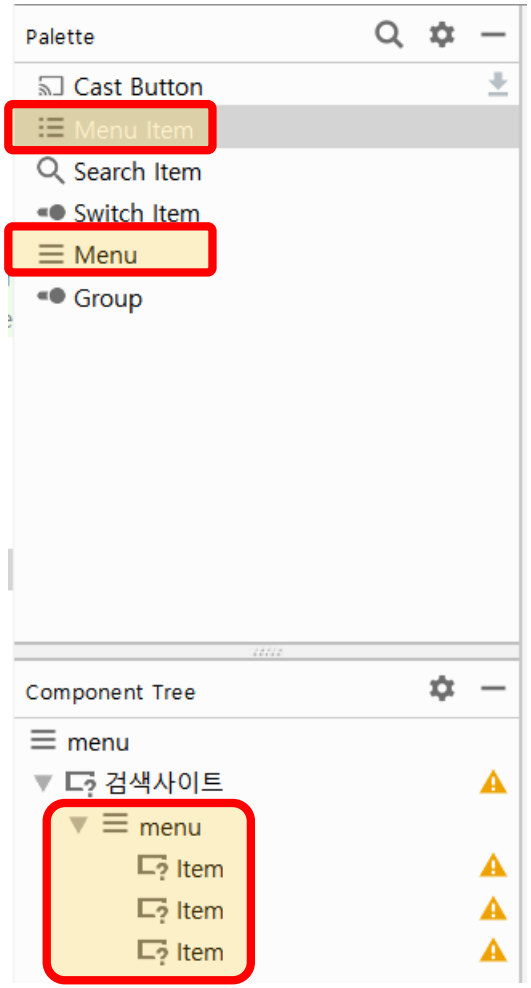


나만의 웹 브라우저

- 메뉴 작성
  - 동일한 방식으로 서브 메뉴 추가 구성
  - 서브 메뉴의 id와 title을 아래와 같이 정의
    - id : action_naver / title : 네이버
    - id : action_google / title : 구글
    - Id : action_daum / title : 다음





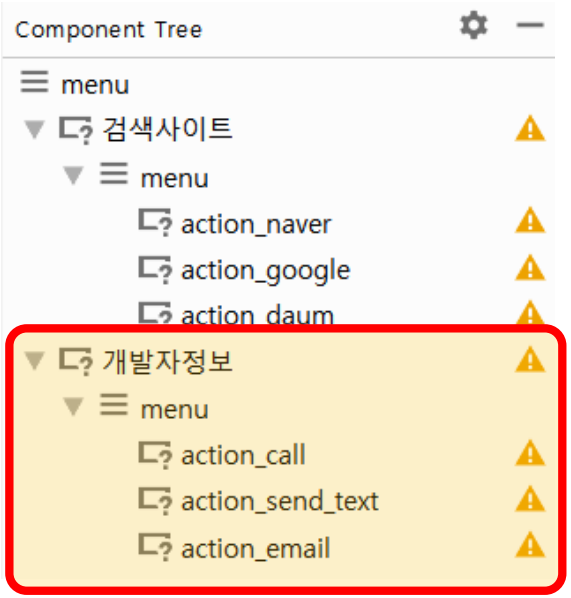


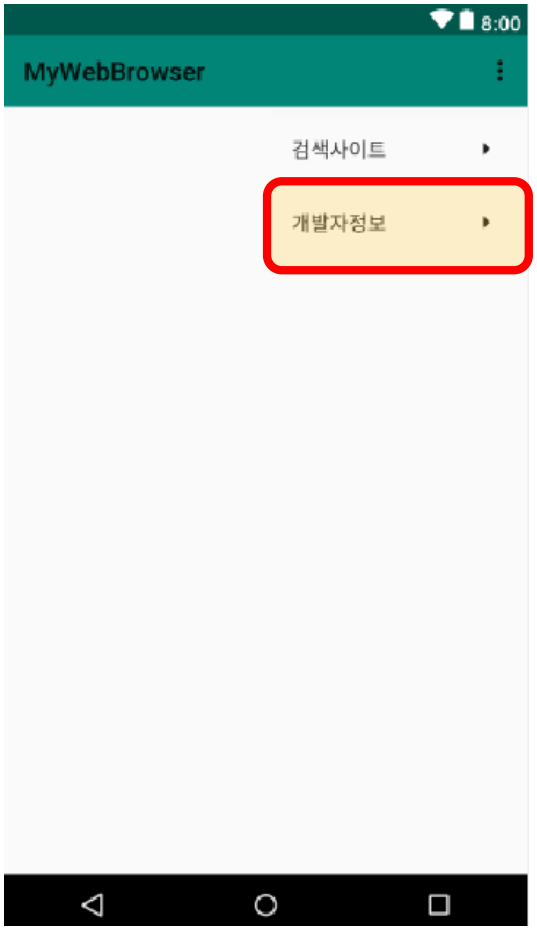
나만의 웹 브라우저

메뉴 작성

동일한 방법으로 개발자 정보 메뉴 및 서브 아이템 추가

- id : action_call / title : 전화하기
- id : action_send_text / title : 문자 보내기
- id : action_email / title : 이메일 보내기

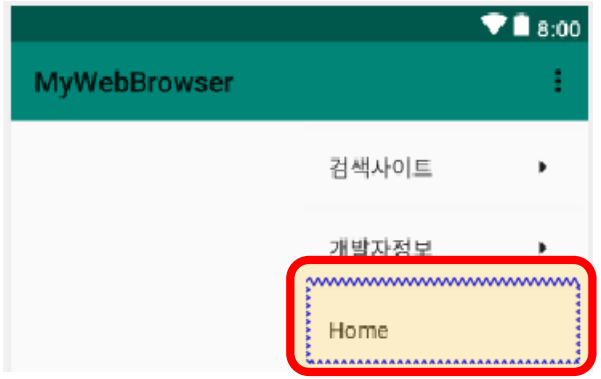


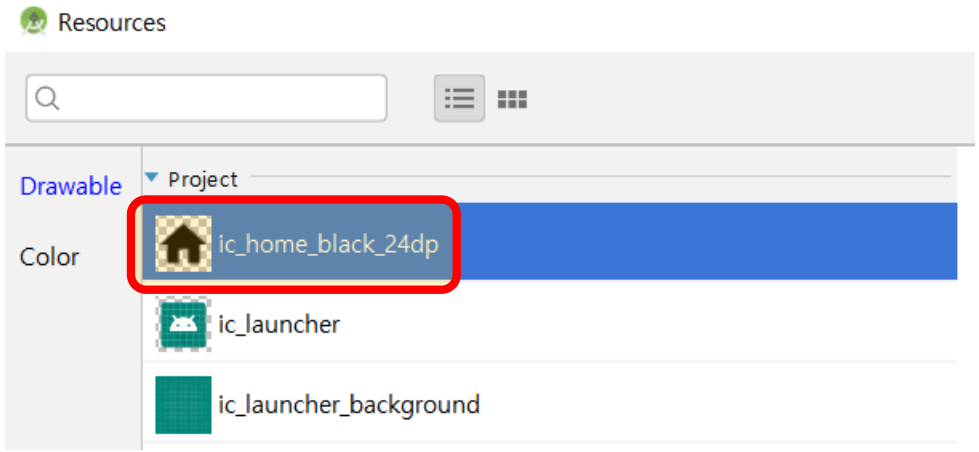


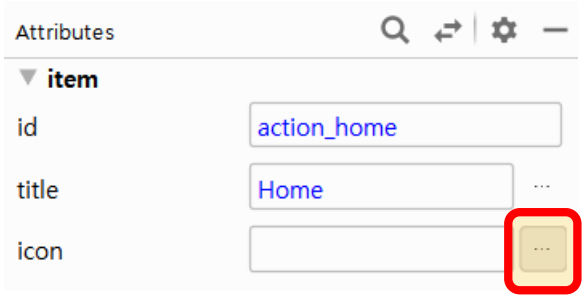
나만의 웹 브라우저

메뉴 작성

- 툴바에 집 모양 아이콘을 표시하는 메뉴 추가
- id : action_home







나만의 웹 브라우저

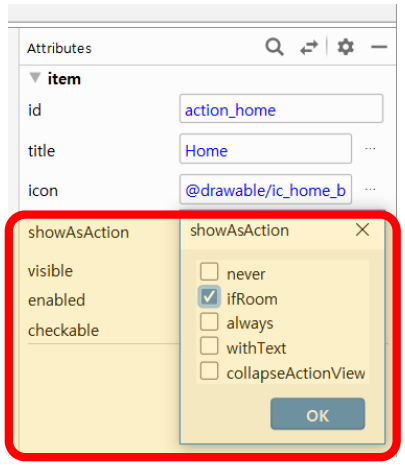
메뉴 작성

툴바에 집 모양 아이콘을 표시하는 메뉴 추가

메뉴 아이템을 툴바 밖으로 노출시키려면 showAsAction 속성을 수정

- never : 밖으로 노출 시키지 않음
- ifRoom : 툴바에 공간이 있으면 노출
- Always : 항상 노출
- withText : 글자와 아이콘을 함께 표시
- collapseActionView : 액션 뷰와 결합하면 축소되는 메뉴를 만들 수 있음

ifRoom으로 선택



나만의 웹 브라우저

옵션 메뉴를 액티비티에 표시

액티비티에서 onCreateOptionsMenu()메서드를 오버라이드하여 메뉴 리소스 파일을 지정

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    menuInflater.inflate(R.menu.main, menu)  //menuinflater객체의 inflate()메서드를 사용하여 리소스 지정
    return true  //true를 반환하면 액티비티에 메뉴가 있음을 알림
}
```

나만의 웹 브라우저

옵션 메뉴 클릭 이벤트 처리

- 옵션 메뉴를 선택했을 경우의 이벤트를 처리하기 위하여 onOptionsItemSeleted()메서드를 오버라이드하여 메뉴 아이템의 id로 분기하여 처리

```
override fun onOptionsItemSelected(item: MenuItem?): Boolean {
    when (item?.itemId) {  //메뉴 아이템으로 분기
        R.id.action_google, R.id.action_home -> { //구글이나 집 아이콘을 선택하면 구글 사이트로 이동
            webView.loadUrl( url: "http://www.google.com")
            return true
        }
        R.id.action_naver -> {
            webView.loadUrl( url: "http://www.naver.com")
            return true
        }
        R.id.action_daum -> {
            webView.loadUrl( url: "http://www.daum.net")
            return true
        }
```

나만의 웹 브라우저

옵션 메뉴 클릭 이벤트 처리(계속)

- 옵션 메뉴를 선택했을 경우의 이벤트를 처리하기 위하여 onOptionsItemSeleted()메서드를 오버라이드하여 메뉴 아이템의 id로 분기하여 처리

```
        R.id.action_call -> {  //암시적 인텐트를 이용하여 연락처를 클릭하면 전화를 연결
            val intent = Intent(Intent.ACTION_DIAL)
            intent.data = Uri.parse( uriString: "tel:031-123-4567")
            if (intent.resolveActivity(packageManager) != null) {
                startActivity(intent)
            }
            return true  //각 메뉴 처리를 끝내면 true를 반환
        }
        R.id.action_send_text -> {
            //문자 보내기
            return true
        }
        R.id.action_email -> {
            //이메일 보내기
            return true
        }
    }
    return super.onOptionsItemSelected(item) //when문에서 분기하지 않은 예외의 경우에는 super 메서드를 호출하는 것이 안드로이드 시스템의 규칙
}
```

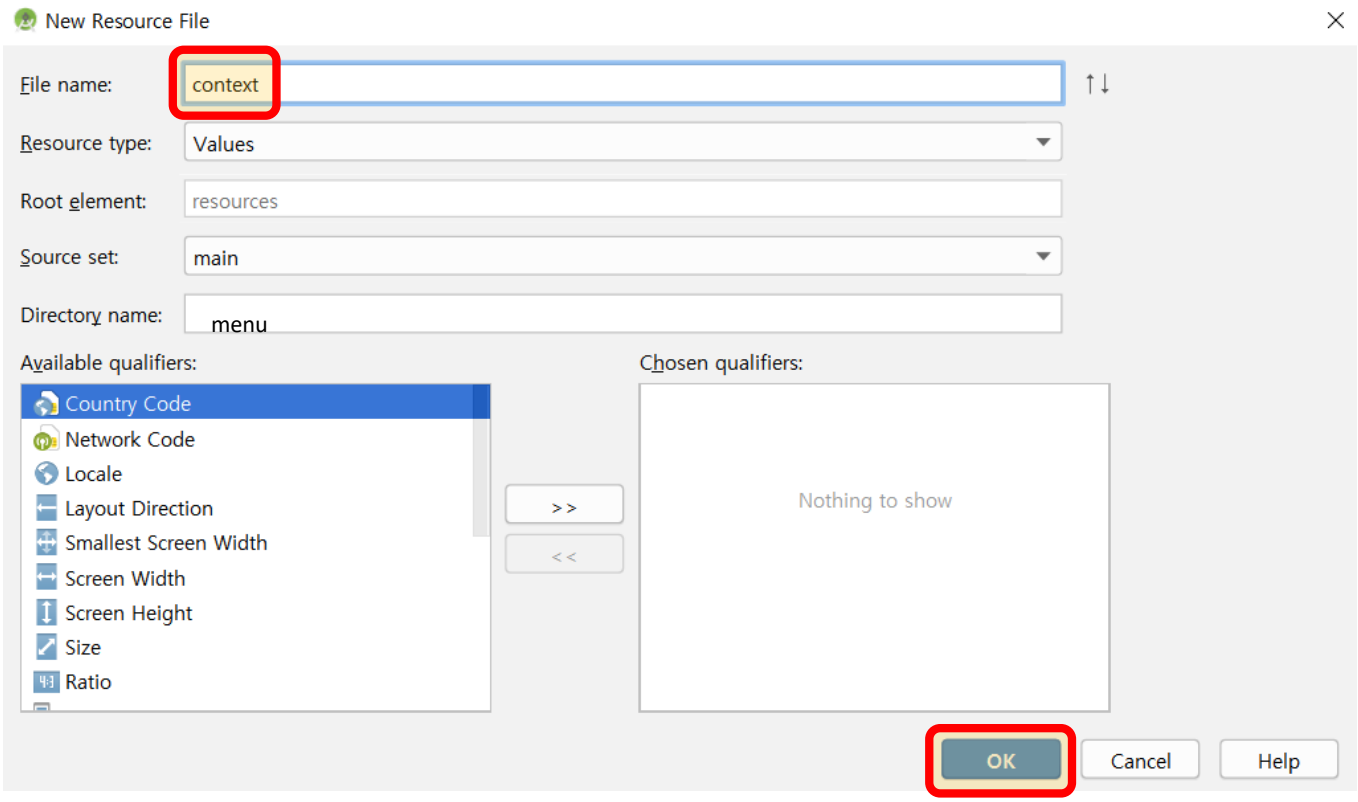
- 앱을 실행하여 정상동작을 확인

나만의 웹 브라우저

- 컨텍스트 메뉴 사용
  - 특정 뷰를 롱 클릭하면 특정한 메뉴를 표시
  - 구현 과정
    - 메뉴 리소스 생성
    - 메뉴 작성
    - 컨텍스트 메뉴 등록
    - 컨텍스트 메뉴 클릭 이벤트

나만의 웹 브라우저

- 메뉴 리소스 파일 생성
  - File - New - Menu Resource File
  - 컨텍스트 메뉴에 표시할 리소스의 메뉴 리소스 파일의 이름을 context로 입력하고 OK

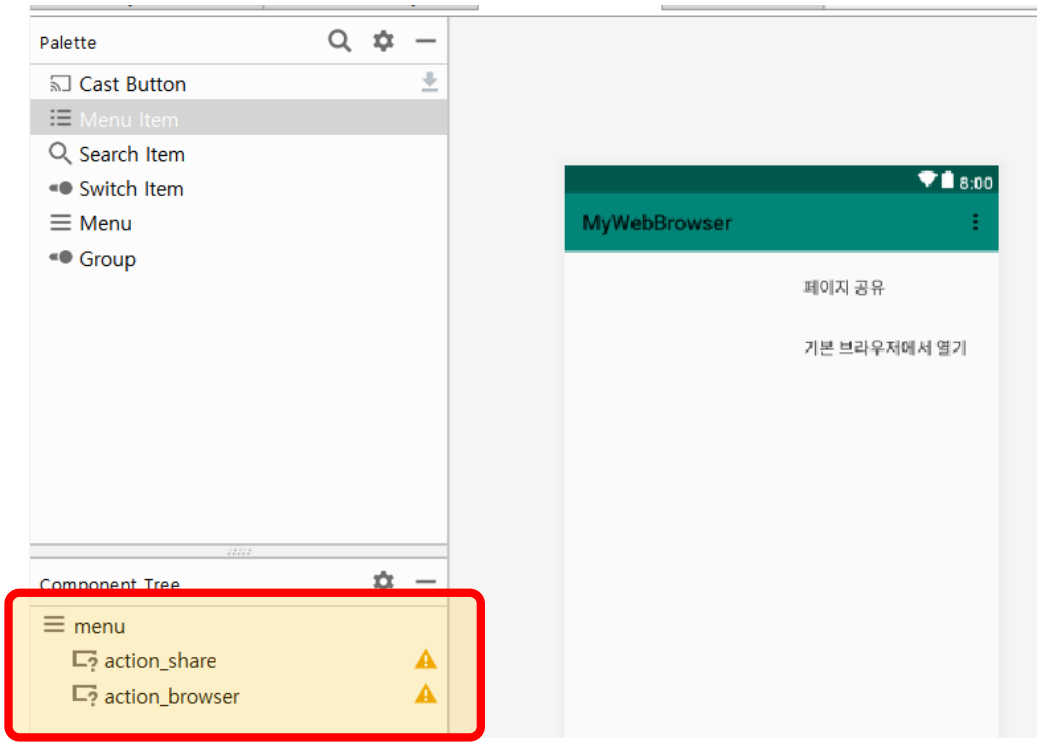


나만의 웹 브라우저

메뉴 작성

context.xml 파일을 열고 메뉴 아이템 두 개 추가

- id : action_share / title : 페이지 공유
- id : action_browser / title : 기본 브라우저에서 열기



나만의 웹 브라우저

컨텍스트 메뉴 등록

onCreateContextMenu()메소드를 오버라이드하여 아래와 같이 코드 작성

```
override fun onCreateContextMenu(menu: ContextMenu?, v: View?, menuInfo: ContextMenu.ContextMenuInfo?) {
    super.onCreateContextMenu(menu, v, menuInfo)
    menuInflater.inflate(R.menu.context, menu) // menuInflater.inflate()메소드로 메뉴 리소스를 컨텍스트 메뉴로 사용
}
```

컨텍스트 메뉴가 표시될 뷰를 지정하기 위하여 onCreate()메소드 내부에 아래의 코드를 추가

```
registerForContextMenu(webView) // registerForContextMenu()메서드에 컨텍스트 메뉴를 표시할 뷰로 웹 뷰를 지정
                                앱을 실행한 후 해당 웹 뷰를 롱클릭하면 컨텍스트 메뉴가 표시됨
```

나만의 웹 브라우저

- 컨텍스트 메뉴 클릭 이벤트 처리
  - onContextItemSelected()오버라이드하여 클릭에 따라 분기하여 처리

```
override fun onContextItemSelected(item: MenuItem?): Boolean {
    when (item?.itemId) {
        R.id.action_share -> {
            // 페이지 공유
        }
        R.id.action_browser -> {
            // 기본 웹 브라우저에서 열기
        }
    }
    return super.onContextItemSelected(item)
}
```

나만의 웹 브라우저

- 암시적 인텐트
  - 안드로이드에서 인텐트(intent)는 명시적 인텐트와 암시적(묵시적) 인텐트로 구분
  - 암시적 인텐트는 미리 정의된 인텐트
  - 옵션 메뉴에서 암시적 인텐트를 이용하여 전화 걸기

```
R.id.action_call -> {
    val intent = Intent(Intent.ACTION_DIAL) // 인텐트를 만들면서 Intent클래스에 정의된 액션 중 전화를 거는 액션을 지정
    intent.data = Uri.parse( uriString: "tel:031-123-4567")  //인텐트에 데이터를 지정, tel: 로 시작하는 Uri는 국제표준
    if (intent.resolveActivity(packageManager) != null) { // intent.resolveActivity는 동일한 인텐트를 수행하는 액티비티가
        startActivity(intent)                              있는지 검사하여, 없다면 null을 반환(전화기능이 없는 태블릿 등)
    }
    return true
}
```

  - 안드로이드에서는 다양한 암시적 인텐트가 존재

나만의 웹 브라우저

- Anko 라이브러리를 활용한 암시적 인텐트
  - Anko에서 자주 사용하는 암시적 인텐트
    - 전화걸기 : makecall(전화번호)
    - 문자보내기 : send(전화번호,[문자열])
    - 웹 브라우저에서 열기 : browse(url)
    - 문자열 공유 : share(문자열)
    - 이메일 보내기 : email(받는 메일주소, [제목], [내용])

나만의 웹 브라우저

Anko 라이브러리를 활용한 암시적 인텐트

암시적 인텐트를 사용하여 문자 보내기와 email 보내기 및 문자열 공유, 웹 브라우저에서 열기

컨텍스트 메뉴

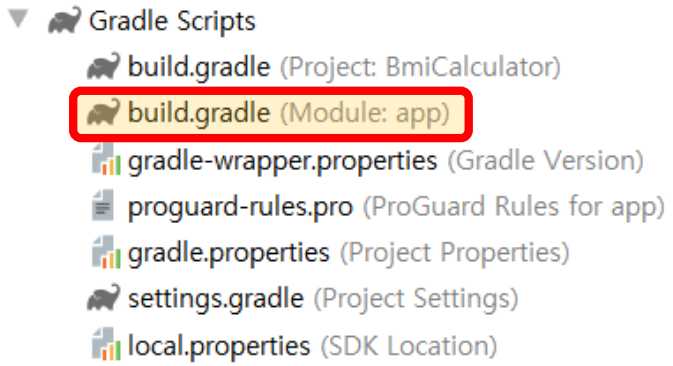
```
R.id.action_share -> {
    share(webView.url)
}
R.id.action_browser -> {
    browse(webView.url)
}
```

옵션 메뉴

```
R.id.action_send_text -> {
    sendSMS("031-123-4567", webView.url)
    return true
}
R.id.action_email -> {
    email("test@example.com", "좋은 사이트", webView.url)
    return true
}
```

나만의 웹 브라우저

- Anko 라이브러리 추가
  - 본 앱 개발에서는 편리하게 코드를 작성할 수 있도록 Anko 라이브러리를 사용
  - 안드로이드는 이클립스와 다르게 코드 편집과 빌드를 분리해서 수행
  - 안드로이드 스튜디오는 소스코드 편집을, 그레이들(gradle)은 빌드를 담당
    - SDK와 앱 버전 및 라이브러리를 관리
  - 그레이들은 기본적으로 프로젝트 수준의 그레이들과 모듈 수준의 그레이들로 구분
    - Build.gradle의 괄호안에 project가 있는 것이 프로젝트 수준, module이라 적혀 있으면 모듈 수준의 그레이들
  - 프로젝트 창에서 모듈 수준의 build.gradle 파일을 더블 클릭하여 열기



나만의 웹 브라우저

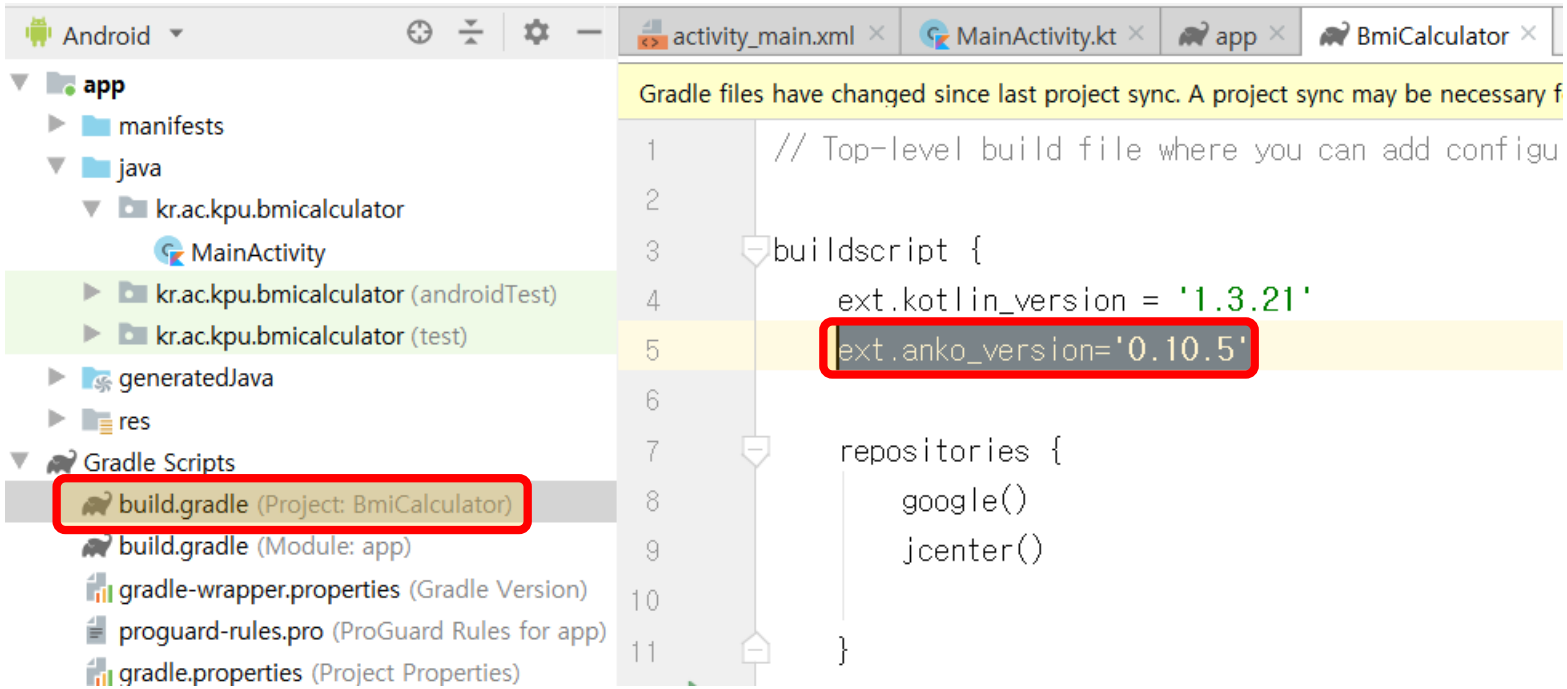
- Anko 라이브러리 추가
  - dependencies 항목에 Anko 라이브러리를 추가
    - implementation "org.jetbrains.anko:anko-commons:$anko_version"
    - 이러한 과정을 라이브러리 의존성 추가(주입)이라고 함

```
25  dependencies {
26      implementation "org.jetbrains.anko:anko-commons:$anko_version"
27
28      implementation fileTree(dir: 'libs', include: ['*.jar'])
29      implementation"org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
30      implementation 'com.android.support:appcompat-v7:28.0.0'
31      implementation 'com.android.support.constraint:constraint-layout:1.1.3'
32      testImplementation 'junit:junit:4.12'
33      androidTestImplementation 'com.android.support.test:runner:1.0.2'
34      androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
```

  - 이와 같이 dependencies 항목에 각 라이브러리를 추가하면 그레이들 빌드 시스템에서 자동으로 라이브러리를 다운로드 받아서 프로젝트에서 사용 가능

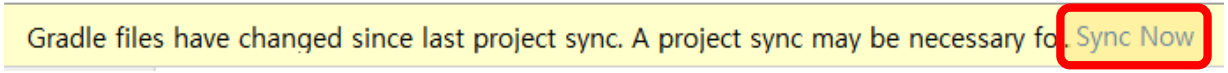
나만의 웹 브라우저

- Anko 라이브러리 추가
  - 프로젝트 수준의 build.gradle 파일을 더블 클릭하여 열기
  - Anko 라이브러리 버전을 변수에 지정
    - ext.anko_version='0.10.5'

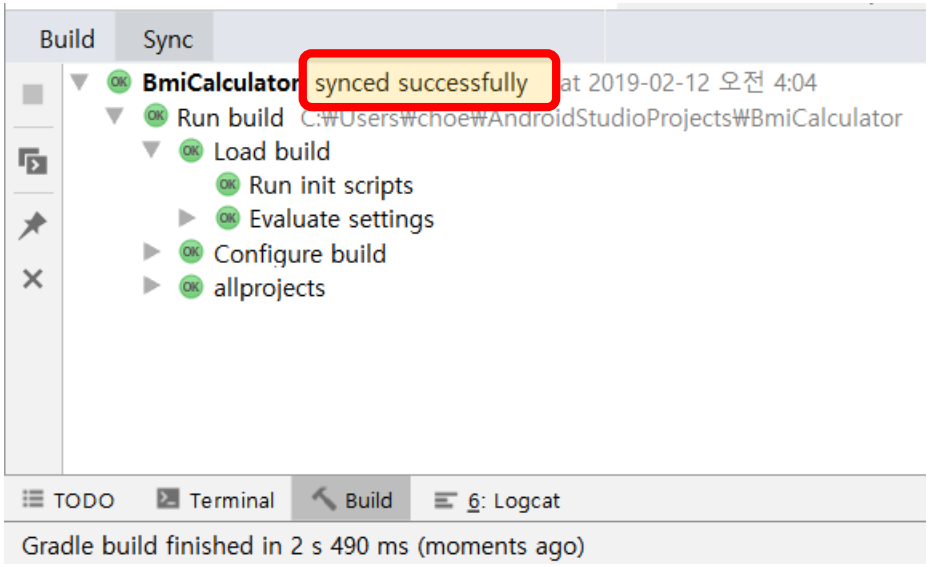


나만의 웹 브라우저

- Anko 라이브러리 추가
  - 그레이들 파일을 수정하고 이를 적용하려면 싱크(동기화)가 필요
  - Sync now 링크를 클릭하여 진행

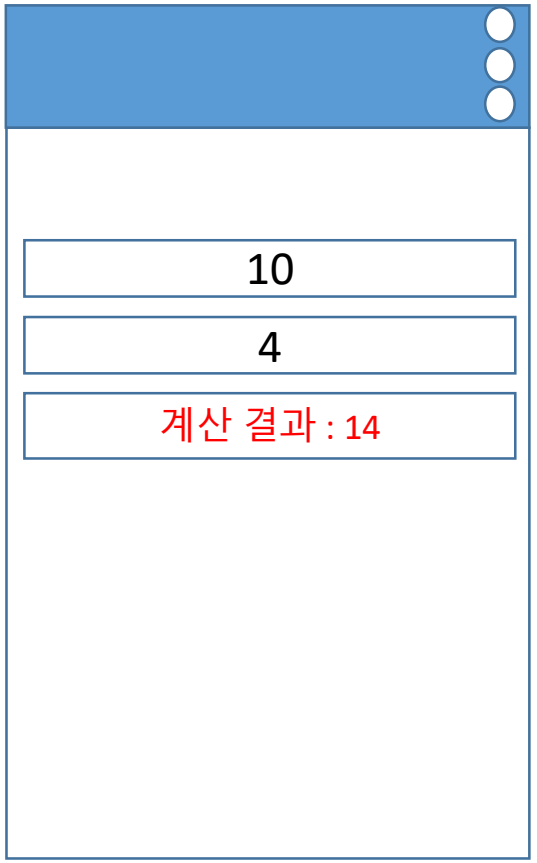


  - 싱크를 하면 프로젝트가 다시 빌드되며 과정 중에 오류가 발생하지 않는다면 성공적으로 Anko 라이브러리 추가
    - 앱을 개발할 경우에 외부 라이브러리를 자주 사용하므로 해당 과정을 숙지



실습문제 1

- 계산기 만들기 - 1
  - 지난번 과제인 계산기를 참고하여 구현
  - 사칙연산은 옵션 메뉴에서 선택
  - 숫자1, 숫자2, 전체 숫자 삭제는 컨텍스트 메뉴로 작성하고 텍스트 뷰에 등록
    - 텍스트 뷰를 롱 클릭하면 컨텍스트 메뉴가 활성화 되도록
    - 전체 숫자 삭제를 선택하면 텍스트 뷰의 계산 결과도 삭제



실습문제 2

- 계산기 만들기 - 2
  - 실습문제 1을 업그레이드 하시오.
  - 계산결과는 실수(소수점)으로 출력
  - 숫자를 입력할 때 소프트 키보드가 아닌, 직접 작성한 숫자버튼으로 입력하시오.
  - SharedPreference를 사용하여 가장 최근에 처리된 피연산자와 계산결과를 표시
  - 전체 숫자 삭제를 선택하면 텍스트 뷰의 계산 결과도 삭제

Q & A