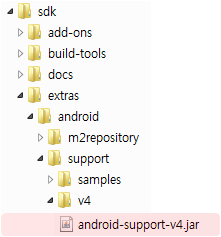
**22.3.9 프래그먼트 라이브러리**

프래그먼트는 안드로이드 API 11 버전 이상부터 추가되었다. 따라서 API 10 이하 버전에서는 사용할 수 없다. 하지만 프래그먼트는 화면을 분할하여 코드의 재사용성과 유연성을 높이기 때문에 하위 버전에서 사용할 수 없다면 너무 아쉽다.이를 위해 안드로이드는 프래그먼트가 API 4 버전까지 지원하도록 추가적으로 라이브러리를 배포했다. 이를 android-support-v4 라이브러리라고 한다. 참고로 라이브러리명 끝에 v4는 API 4 버전까지 호환되는 라이브리러를 의미한다. 해당 라이브러리는 안드로이드가 상위버전으로 올라가면서 추가된 다양한 클래스들이 하위버전에서 사용할 수 있도록 포함되어 있다. 즉 프래그먼트 기능만을 위한 라이브러리는 아니라는 의미다.

그렇다면 이 라이브러리는 어디에 존재할까? 다음의 그림을 살펴보자.

그림 22-52 android-support-v4 라이브러리 위치



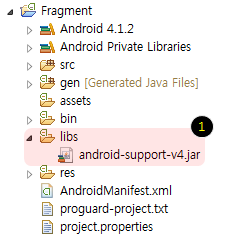
[그림 22-52]와 같이 라이브러리는 다음과 같이 내려받은 안드로이드 SDK 폴더에 존재한다.

*/안드로이드 SDK 폴더/extras/android/support/v4/android-support-v4.jar*

이 파일이 존재하지 않는다면 안드로이드 SDK매니저를 통해 다시 내려받을 수 있다. 상세한 내용은 2장 2.4절에서 설명되었다.

이제 android-support-v4 라이브러리를 사용해보자. 물론 이 라이브러리를 사용하려면 패키지에 해당 라이브러리를 추가해야 한다. 그런데 재미있게도 이 라이브러리는 새로운 패키지를 생성할 때 자동으로 패키지 내에 포함된다. 안드로이드는 이 라이브러리의 접근성을 높이기 위해 자동으로 패키지에 포함시키는 것이다. [그림 22-53]을 살펴보자.

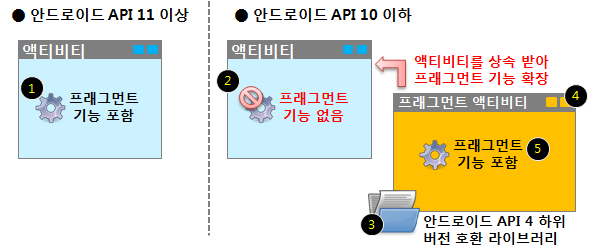
그림 22-53 기본으로 패키지에 포함된 android-support-v4 라이브러리



①     패키지 탐색기에서 예제 패키지 하위 libs 폴더를 보면 android-support-v4 라이브러리가 존재한다.

android-support-v4를 이용해서 프래그먼트를 사용할 모든 준비는 이미 끝나 있었다. 그렇다면 라이브러리의 어떤 클래스를 이용해서 프래그먼트를 사용해야할까? [그림 22-54]를 살펴보자.

그림 22-54 프래그먼트 기능이 존재하는 클래스



①     프래그먼트는 안드로이드 API 11 이상부터 추가되었고, 이 기능이 포함된 곳은 바로 액티비티 클래스다.

②     하지만 API 10 이하 버전은 액티비티에 프래그먼트 기능 자체가 없다.

③     이를 위해 안드로이드는 android-support-v4 라이브러리에 프래그먼트 기능을 추가했다.

④     라이브러리에 포함된 프래그먼트는 프래그먼트 액티비티FragmentActivity클래스를 통해 제공된다.

⑤     즉 프래그먼트 액티비티에 프래그먼트 기능이 포함되어 있다.

⑥     그리고 프래그먼트 액티비티는 액티비티를 상속받아 프래그먼트 기능을 확장한 클래스다. 따라서 우리는 액티비티 대신 프래그먼트 액티비티를 상속받아 액티비티를 구현하면 된다.

지금까지 작성된 예제를 안드로이드 API 10 버전에서도 동작하도록 수정해보자.

먼저 AndroidManifest.xml에 예제 앱이 안드로이드 API 10 버전에도 설치 및 실행되도록 수정한다.

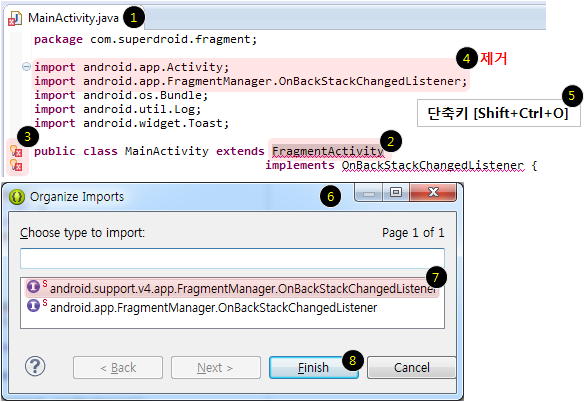
예제 22-52 예제 앱을 안드로이드 API 10까지 지원하도록 AndroidManifest.xml 수정

|  |
| --- |
| **AndroidManifest.xml**    ...      <uses-sdk  **~~android:minSdkVersion=~~*~~"16"~~***  **android:minSdkVersion=*"10"***         android:targetSdkVersion=*"16"* /> |

기존 minSdkVersion이 16으로 설정된 것을 10으로 변경했다. 이제 안드로이드 API 10 버전에서도 예제앱은 설치 및 실행될 수 있다.

다음은 예제 액티비트에 프래그먼트 라이브러리를 적용한다. 먼저 다음의 그림을 통해 그 방법을 살펴본다.

그림 22-55 android-support-v4 라이브러리의 프래그먼트 액티비티 사용하기



①     예제의 MainActivity.java에 프래그먼트 라이브러리를 적용한다.

②     프래그먼트를 사용하는 액티비티는 일반 액티비티를 상속받아 구현할 수 없다. 그 이유는 앞서 말했듯이 안드로이드 API 10 버전 이하 액티비티에 프래그먼트 기능이 없기 때문이다. 따라서 라이브러리로 제공되는FragmentActivity를 상속받도록 변경한다.

③     이렇게 변경하고 저장하면 MainActivity 소스에 컴파일 에러가 발생될 것이다. 그 이유는 FragmentActivity가 어느 클래스에 포함되어 있는지 적어주지 않았기 때문이다. 따라서 이클립스는 FragmentActivity 클래스와 관련된 부분을 에러로 표시한다. 그렇다면 해당 클래스 임포트import하는 부분을 살펴보자.

④     Activity와 OnBackStackChangedListerner가 임포트되어 있다. 하지만 이 두 가지는 경로가 잘못되었다. Activity의 경우 FragmentActivity를 OnBackStackChangedListerner는 프래그먼트 라이브러리의OnBackStackChangedListerner를 참조해야하기 때문이다. 따라서 이 두 가지 내용을 삭제하자.

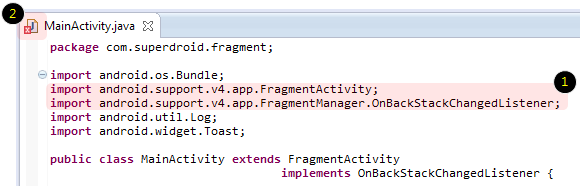
⑤     이제 제대로 된 경로를 임포트해야 한다. 이를 위해 라이브러리에서 프래그먼트 클래스 위치를 찾아야 할 것이다. 하지만 매우 간단한 방법이 있다. 바로 소스 내에서 사용되는 클래스들의 경로를 자동으로 임포트해주는 단축키다. [Shift+Ctrl+O]를 누르자.

⑥     이후 Organize Imports 창이 열리는데 그 이유는 OnBackStackChangedListerner 클래스가 존재하는 곳이 두 곳이기 때문이다. 내용을 살펴보면 android-support-v4 라이브러리와 안드로이드 기본 SDK에OnBackStackChangedListerner가 각각 존재한다. 따라서 어느 경로에 있는 OnBackStackChangedListerner를 사용할지 결정해야 한다.

⑦     우리는 당연히 android-support-v4 라이브러리의 클래스를 선택한다.

⑧     Finish 버튼을 누르면 임포트가 완료된다. 계속 [그림 22-56]을 살펴보자.

그림 22-56 임포트된 프래그먼트 액티비티



①     FragmentActivity, OnBackStackChangedListerner 클래스 경로가 android-support-v4 라이브러리를 참조하도록 수정되었다.

②     그런데 아직 MainActivity.java는 에러가 존재한다. 더 수정해야할 것들이 남아 있다는 의미다.

MainActivity.java에서 에러가 계속 발생되는 이유는 바로 프래그먼트매니저를 참조하는 getFragmentManager 함수 때문이다. 이상하게도 android-support-v4 라이브러리의 FragmentActivity는 프래그먼트매니저를 참조하는 함수명을 기존의getFragmentManager를 사용하지 않고 getSupportFragmentManager로 변경했다. 저자도 정확한 이유에 대해서는 이해가 되지 않는다. 어쨌든 소스 내에 모든 getFragmentManager 함수를 getSupportFragmentManager로 바꿔 주어야 한다. 다음과 같이 소스를 수정하자.

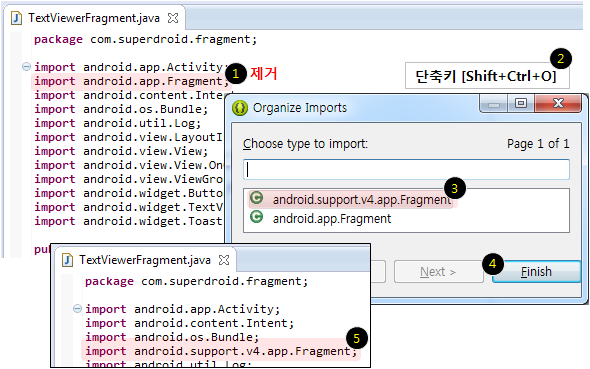
예제 22-53 액티비티를 라이브러리 프래그먼트를 사용하도록 수정

|  |
| --- |
| **/src/MainActivity.java**    **public** **class** MainActivity **extends** FragmentActivity  **implements** OnBackStackChangedListener {        ...        @Override  **protected** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);            Log.*d*("superdroid", "Activity onCreate()");            setContentView(R.layout.*activity\_main*);            mTextViewerFragment = TextViewerFragment.*newInstance*();          mImageViewerFragment = ImageViewerFragment.*newInstance*();    **if**( savedInstanceState == **null** )          {  **getSupportFragmentManager()**              .beginTransaction()                  .add( R.id.*viewer\_fragment\_container*,                        mTextViewerFragment, "TEXT\_VIEWER")              .commit();          }          ... |

[예제 22-53]는 getFragmentManager 함수를 getSupportFragmentManager로 변경한 일부를 보여주었다. 이 부분 외에도 여러 곳에서 getFragmentManager 함수가 사용되고 있으니 모두 변경하도록 하자. 그리고 예제에 포함된 모든 액티비티들도 같은 방법으로 수정한다.

이제 모든 액티비티들을 수정했다. 하지만 아직 액티비티에는 에러 코드가 남아 있을 것이다. 이는 프래그먼트 액티비티에서 참조하는 프래그먼트들이 라이브러리를 사용하도록 수정되지 않았기 때문이다. 그렇다면 모든 프래그먼트를android-support-v4 라이브러리를 사용하도록 수정해보자. 대표로 텍스트뷰어 프래그먼트를 수정한다. 먼저 다음의 그림을 통해 그 방법을 살펴본다.

그림 22-57 android-support-v4 라이브러리 프래그먼트 사용하기



①     기존에 임포트된 프래그먼트 클래스 경로는 기본 SDK 라이브러리의 경로를 참조하고 있다. 따라서 android-support-v4 라이브러리에 프래그먼트를 사용하도록 수정해야 한다. 해당 줄을 삭제하자.

②     이제 프래그먼트 클래스 경로를 임포트해야 한다. 이번에도 자동으로 임포트하는 단축키 [Shift+Ctrl+O]를 누른다.

③     android-support-v4 라이브러리의 프래그먼트 클래스를 선택한다.

④     Finish 버튼을 누르면 임포트가 완료된다.

⑤     텍스트뷰어 프래그먼트 소스에 android-support-v4 라이브러리의 프래그먼트 클래스가 임포트된 것을 볼 수 있다.

그런데 혹시 프래그먼트에서도 프래그먼트매니저를 참조할 때 getSupportFragmentManager로 변경해야 할까? 참 헷갈리게도 그렇지 않다. 좀 일관성이 없지만 이해하고 넘어가자.

[참고 예제 소스]

[[첨부파일](javascript:checkVirus('grpid%3D1MWA2%26fldid%3DpgqT%26dataid%3D11%26fileid%3D1%26regdt%3D20141018000207&url=http%3A%2F%2Fcfile250.uf.daum.net%2Fattach%2F23395142544A3B071C43A1')) 22-19. 하위 호환을 위한 프래그먼트 라이브러리 사용.zip](javascript:checkVirus('grpid%3D1MWA2%26fldid%3DpgqT%26dataid%3D11%26fileid%3D1%26regdt%3D20141018000207&url=http%3A%2F%2Fcfile250.uf.daum.net%2Fattach%2F23395142544A3B071C43A1'))

모든 변경은 끝났다. 간단하지 않은가! 이제 안드로이드 API 10 버전의 AVD를 만들고 에뮬레이터를 구동시킨뒤 예제를 실행해보자.

그림 22-58 라이브러리 프래그먼트 적용 테스트



①     안드로이드 API 10 버전의 진저브래드에서 프래그먼트 예제가 성공적으로 실행되었다.

②     이미지 보기 버튼을 누른다.

③     이미지뷰어 프래그먼트도 정상적으로 바뀐다.

android-support-v4 라이브러리의 프래그먼트로 적용하는 과정이 결코 어렵지 않음을 느꼈을 것이다. 마지막으로 적용 과정을 정리해보자.

①     프래그먼트를 사용하는 모든 액티비티들을 프래그먼트 액티비티를 상속받도록 수정한다.

②     액티비티와 프래그먼트 소스에 프래그먼트 임포트 정보를 모두 android-support-v4 라이브러리의 경로로 수정한다. 이때 단축키 [Shift+Ctrl+O]를 사용하면 편리하다.

③     프래그먼트 액티비티 소스 내에 모든 getFragmentManager 함수를 getSupportFragmentManager로 바꾼다.

**22.4 마치며**

22.3.9절 프래그먼트 라이브러리를 끝으로 22장을 마무리 합니다. ^^

아마도 몇몇 분들은 왜 DialogFragment, ListFragment...나

혹은 ViewPager 그리고 FragmentPagerAdapter / FragmentStatePagerAdapter

등은 설명하지 않지라고 생각하실 수 있습니다.

하지만 프래그먼트는 22장에서 설명한 것이 기본이고 전부입니다.

위에 나열한 설명되지 않은 것들은

프래그먼트의 파생클래스 이거나

프래그먼트를 다루는 뷰그룹에 불과하죠.

그런데 많은 분들이 프래그먼트의 기본 내용을 이해하지 않은 상태에서

위에 나열한 것들을 먼저 접합니다. 따라서 프래그먼트가 어렵게 느껴지죠.

아마도 22장을 정확히 이해하셨다면 이제 프래그먼트와 관련된 클래스들은

쉽게 다가갈 수 있을 것입니다.

이제 다음 장에 설명될 내용은 프래그먼트 응용입니다.

즉 DialogFragment, ListFragment...등을 설명하게 될 것입니다.

감사합니다.