

```
public final static String ONLINE_LIST_PATH =  
"https://gitee.com/xiaolou023/resources/raw/master/musics/online/alist.json";  
public final static String LOCAL_FOLDER_PATH = "/storage/emulated/0/Music";  
public final static String ONLINE_FOLDER_PATH =  
"https://gitee.com/xiaolou023/resources/raw/master/musics/online";
```

本地音乐下载地址: <https://gitee.com/xiaolou023/resources/tree/master/musics/offline>

知识点

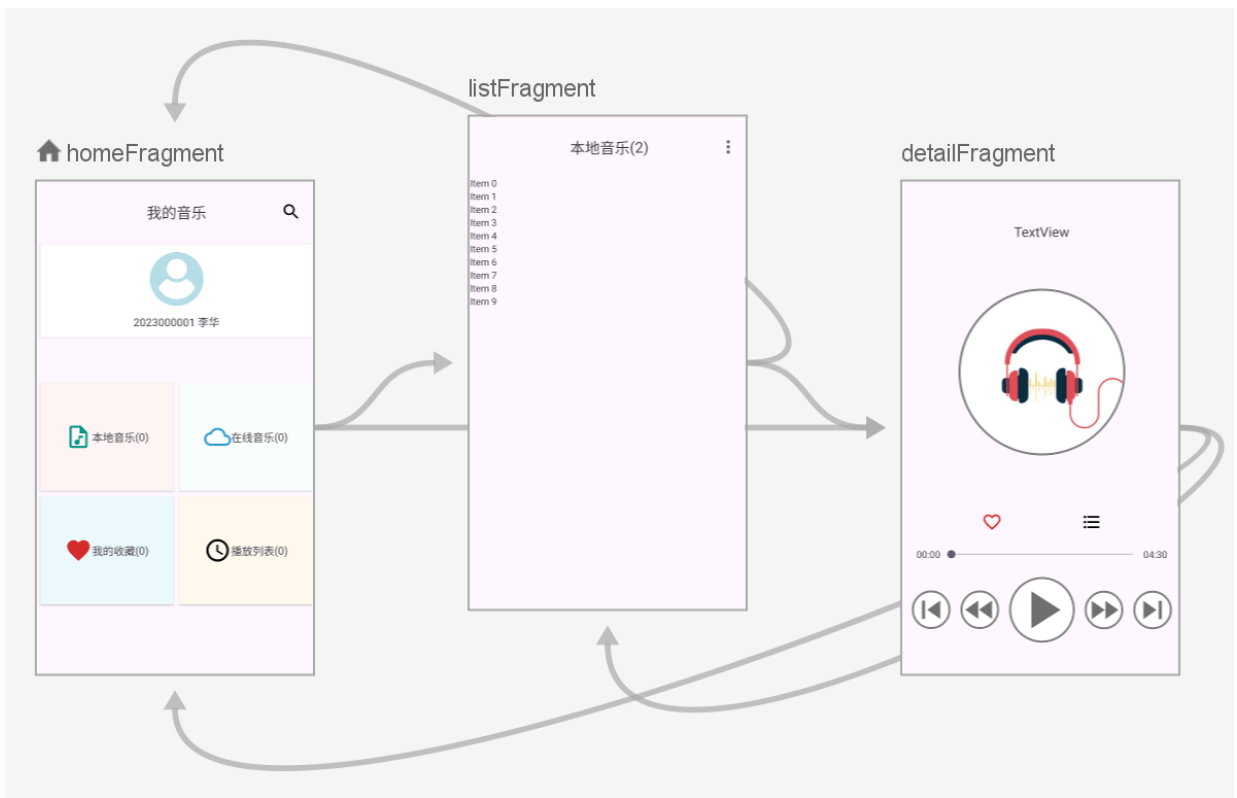
资源的管理与引用、控件与事件监听、界面布局、消息框、对话框、PopupMenu、RecyclerView、属性动画、ShapeableImageView、Bitmap、Fragment与Navigation、ViewModel、LiveData与数据观察、SharedPreferences、外部存储、Room、Intent、ContentProvider与ContentResolver、Broadcast与BroadcastReceiver、Handler消息传递机制、多线程、Service、OkHttp、MediaPlayer

编程步骤

1. 用管理者权限打开Android Studio
2. 创建新项目，以Music+学号命名，例如：Music20230000001
3. 添加依赖，并同步

```
// fix duplicated class  
implementation(platform("org.jetbrains.kotlin:kotlin-bom:1.8.0"))  
// navigation  
implementation("androidx.navigation:navigation-fragment:2.7.6")  
implementation("androidx.navigation:navigation-ui:2.7.6")  
// room  
val room_version = "2.6.1"  
implementation("androidx.room:room-runtime:$room_version")  
annotationProcessor("androidx.room:room-compiler:$room_version")  
// shapeableImageView  
implementation("com.google.android.material:material:1.12.0")  
// Gson  
implementation("com.google.code.gson:gson:2.8.6")  
// OkHttp  
implementation("com.squareup.okhttp3:okhttp:4.4.0")
```

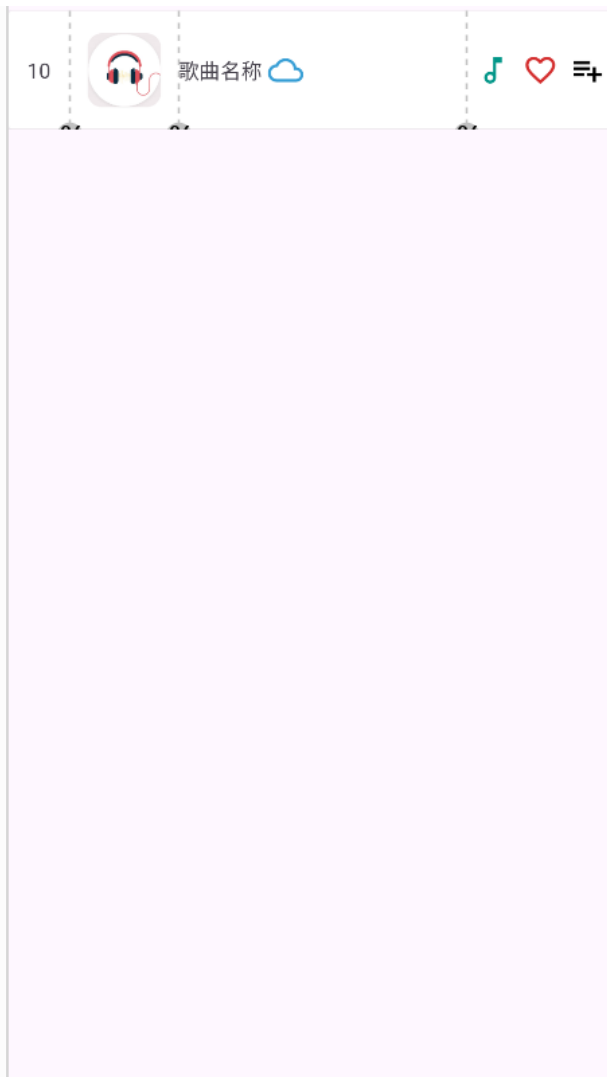
4. 运行项目并通过Device Explorer将本地音乐导入到/storage/emulated/0/Music
5. 创建或导入资源，例如：drawable、colors、dimens、strings、styles
6. 创建三个Fragments：HomeFragment、ListFragment、DetailFragment
 - 。更新Fragments对应的布局文件
7. 创建导航图 Navigation Graph



8. 更新MainActivity对应的布局文件：分为两部分，上面是NavHostFragment，下面是cardView



9. 创建并设计RecyclerView中item对应的布局文件



10. 创建并设计PopupMenu对应的布局文件



11. 更新清单文件

- 静态授权

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

- 自定义App的图标和名字

```
android:icon="@drawable/music_logo"
android:label="@string/app_name"
android:roundIcon="@drawable/music_logo"
```

- 调整键盘的影响（写在标签内）

```
android:windowSoftInputMode="adjustNothing"
```

12. 创建Song、SongDao、SongDatabase及SongRepository

- Song中包括了私有属性、构造方法、getter和setter方法，记得@Entity、@PrimaryKey等注解。
- SongDao是接口，包括了对数据库的增删改查各操作的声明，记得@Dao、@Query等注解
- SongDatabase是抽象类，包括了一个构造方法和一个获取WordDao的抽象方法，采用单例模式，记得@Database注解
- SongRepository封装了SongDao的抽象方法，多线程编程写在SongRepository内。记得需要手动在子线程中执行增删改，而无需手动在子线程中执行以LiveData为返回值的查询。

```
c 📁 SongRepository
  m 📁 SongRepository(Context)
  m ◦ insert(Song...): void
  m ◦ delete(Song...): void
  m ◦ update(Song...): void
  m ◦ renewLocalSongs(Song...): void
  m ◦ renewOnlineSongs(Song...): void
  m ◦ getAllSongs(): LiveData<List<Song>>
  m ◦ getLocalSongs(): LiveData<List<Song>>
  m ◦ getOnlineSongs(): LiveData<List<Song>>
  m ◦ getFavoriteSongs(): LiveData<List<Song>>
  m ◦ getPlayList(): LiveData<List<Song>>
  m ◦ queryByKeyword(String): LiveData<List<Song>>
```

13. 创建一个Param类和一个DataHelper类

- Param类保存了所有需要的常量（常量用public static final等关键字声明）
- DataHelper类采用单例模式，提供了外部存储和网络编程的方法，主要用于获取歌曲的image或audio资源。注意：获取网络资源需要在子线程中执行。

```

C DataHelper
m getInstance(): DataHelper
m getLocalData(): Song[]
m getOnlineData(): Song[]
m loadSongImage(ImageView, Song): void
m changeFormat(int): String

```

14. 创建ViewModel：管理UI数据

- 主要属性：

```

f repository: SongRepository
f allSongs: LiveData<List<Song>>
f localSongs: LiveData<List<Song>>
f onlineSongs: LiveData<List<Song>>
f favoriteSongs: LiveData<List<Song>>
f playList: LiveData<List<Song>>
f listType: int
f curSongIndex: MutableLiveData<Integer>
f playStatus: MutableLiveData<Integer>
f shp: SharedPreferences
f renewData: boolean
f photoPath: String

```

15. 创建Service

- 代码要点：Handler发送消息、Timer发布定时任务、MediaPlayer控制音乐播放、发送广播、创建自定义Binder类
- 主要属性：

```

f mediaPlayer: MediaPlayer
f timer: Timer
f songs: List<Song>
f curSong: Song
f curSongIndex: int
f playStatus: int
f handler: Handler

```

- 主要方法：

```
C MusicService
> C MusicBinder
  m onCreate(): void ↑Service
  m onBind(Intent): IBinder ↑Service
  m onDestroy(): void ↑Service
  m addTimer(): void
  m cancelTimer(): void
  m sendCurSongIndex(): void
  m sendPlayStatus(): void
```

。 Binder的主要方法：

```
C MusicBinder
  m addToPlayList(Song): void
  m getSongs(): List<Song>
  m set(List<Song>, int): void
  m addToPlayList(List<Song>): void
  m removeFromPlayList(int): void
  m clearPlayList(): void
  m updateFavorite(Song, int): void
  m getCurSong(): Song
  m getSong(int): Song
  m startPlay(int): void
  m startPlay(): void
  m playOrPause(): void
  m resumePlay(): void
  m pausePlay(): void
  m stopPlay(): void
  m playPreviousSong(): void
  m playNextSong(): void
  m rewind(): void
  m forward(): void
  m seekTo(int): void
```

16. 修改MainActivity的代码

。 代码要点：获取权限、创建ViewModel、加载音乐数据、绑定服务、注册广播接收器、重写back键的功能、数据观察、事件监听

```

C MainActivity
> C ◦ MusicServiceConnection
> C ◦ MusicReceiver
  m ? onCreate(Bundle): void ↑FragmentActivity
  m ? onPause(): void ↑FragmentActivity
  m ? onDestroy(): void ↑AppCompatActivity
  m 🔒 renewData(): void
  m 🔒 getPermission(): void
  m ? onRequestPermissionsResult(int, String[], int[]): void
  m ? onBackPressed(): void ↑ComponentActivity
  m 🔒 exit(): void
  m 🔒 setClickListenerOnPlayBar(): void
  m 🔒 observeLiveData(): void

```

17. 修改HomeFragment的代码：删除多余代码、保留onCreate()和onCreateView(), 重写onViewCreated()

- 代码要点：更换头像、数据观察、事件监听

```

? HomeFragment
  m ? onCreate(Bundle): void ↑Fragment
  m ? onCreateView(LayoutInflater, ViewGroup, Bundle): View
  m ? onViewCreated(View, Bundle): void ↑Fragment
  m 🔒 createSelectImageLauncher(): void
  m 🔒 loadPhoto(): void
  m 🔒 changePhoto(): void
  m 🔒 navigateToListFragment(View): void

```

16. 创建MusicRecyclerViewAdapter

- 代码要点：创建自定义的ViewHolder、重写MusicRecyclerViewAdapter的三个方法、收藏与取消收藏、添加到播放列表或从播放列表中移除、播放指定音乐

```











C MusicRecyclerViewAdapter
> C ◦ MusicViewHolder
  m ? MusicRecyclerViewAdapter(MusicViewModel, MusicBinder)
  m ? setSongs(List<Song>): void
  m ? getSongs(): List<Song>
  m ? onCreateViewHolder(ViewGroup, int): MusicViewHolder ↑Adapter
  m ? onBindViewHolder(MusicViewHolder, int): void ↑Adapter
  m ? getItemCount(): int ↑Adapter
  m 🔒 setFavorite(Song): void
  m 🔒 cancelFavorite(Song): void
  m 🔒 sendFavoriteMessage(boolean): void
  m 🔒 addToPlayList(Song): void
  m 🔒 removeFromPlayList(Song): void

```

17. 修改ListFragment的代码：删除多余代码、保留onCreate()和onCreateView(), 重写onViewCreated()

- 。代码要点：根据listType自定义界面UI、RecyclerView、搜索音乐、收起键盘、popupMenu、播放全部音乐、将全部添加到播放列表、清空播放列表、数据观察、事件监听








ListFragment

```
m  onCreate(Bundle): void ↑Fragment
m  onCreateView(LayoutInflater, ViewGroup, Bundle): View
m  onViewCreated(View, Bundle): void ↑Fragment
m  initView(): void
m  observeLiveData(): void
m  clearPlayList(): void
m  playAll(): void
m  addAll(): void
m  hideKeyBoard(): void
m  onDestroyView(): void ↑Fragment
```

18. 修改DetailFragment的代码：删除多余代码、保留onCreate()和onCreateView(), 重写onViewCreated()

- 。代码要点：隐藏playBar、handler机制、属性动画、数据观察、事件监听

DetailFragment

```
m  onCreate(Bundle): void ↑Fragment
m  onCreateView(LayoutInflater, ViewGroup, Bundle): View
m  onViewCreated(View, Bundle): void ↑Fragment
m  setFavorite(Song): void
m  cancelFavorite(Song): void
m  onDestroyView(): void ↑Fragment
m  observeLiveData(): void
```

考试/录屏要求

步骤	预期结果
（卸载后重新）安装并运行	弹出对话框获取权限； 点击允许后显示界面
点击头像（更换头像）	打开相册； 选择一张图片后自动返回并更新头像
点击playBar	弹出Toast
点击搜索框	跳转到ListFragment 显示搜索框和所有音乐（滑动到最后一首来证明）
输入歌名搜索	根据输入内容实时更新显示的音乐
输入结束后滚动recyclerView	收起键盘
（搜索框有文字的情况下）返回并再次点击搜索框	显示所有音乐（滑动到最后一首来证明）
返回并点击本地音乐	正确显示本地音乐（滑动到最后一首来证明）
点击右上角的图标	弹出popup menu
点击全部播放	正确播放（从第一首开始播放） 正确更新playBar的UI 正确更新RecyclerView
返回HomeFragment	正确显示播放列表的数量
点击播放列表	正确显示播放列表中的所有音乐（滑动到最后一首来证明）
点击右上角的图标	弹出popup menu
点击清空	弹出AlertDialog
依次点击取消和确认	点击取消时：对话框消失 点击确认时： 正确停止播放 播放列表已清空 正确更新playBar的UI
返回并再次点击本地音乐	正确显示本地音乐（注意playingStatus）
点击右上角的图标	弹出popup menu
点击全部添加到播放列表	正确更新RecyclerView
返回HomeFragment	正确显示播放列表的数量
点击播放列表	正确显示播放列表中的所有音乐
点击右上角的图标	弹出popup menu
点击清空	弹出AlertDialog
点击确认	正确停止播放 播放列表已清空 正确更新playBar的UI
返回并再次点击本地音乐	跳转到ListFragment 显示标题和所有本地音乐

步骤	预期结果
依次点击4首（不顺序的）音乐	正确播放 正确更新playBar的UI 正确更新RecyclerView
点击playBar的暂停键	正确暂停播放 正确更新playBar的UI
再次点击playBar的暂停键	正确继续播放 正确更新playBar的UI
点击playBar的下一首键	正确播放（注意是播放列表中的顺序） 正确更新playBar的UI 正确更新RecyclerView
点击playBar的上一首键	正确播放 正确更新playBar的UI 正确更新RecyclerView
依次点击两首歌的收藏键（重复进行三次）	弹出Toast 正确更新recyclerView的UI（收藏与取消收藏）
返回HomeFragment	正确显示收藏列表和播放列表的数量
点击我的收藏	正确显示我的收藏中的所有音乐
返回并点击播放列表	正确显示播放列表中的所有音乐
返回并点击在线音乐	正确显示在线的所有音乐
陆续点击2首（不顺序的）音乐 （注：播放线上歌曲需要获取网络资源，需要等待，不要急着连续操作，容易报错）	正确播放 正确更新playBar的UI 正确更新RecyclerView
返回并点击播放列表	正确显示播放列表中的所有音乐
点击当前首、前一首、后一首的“从播放列表中移除”按钮 （如果当前首为最后一首，则后一首为第一首）	前一首和后一首被移除 当前歌曲弹出Toast
点击一次playBar的上一首、两次下一首	正确播放更新后的上一首和下一首 正确更新playBar的UI
点击playBar	正确显示DetailFragment
拖动进度条	正确跳转播放
点击两次的rewind和两次的forward	正确播放 正确更新进度条
依次点击两次的暂停键	正确暂停和播放 正确更新动画
依次点击下一首、上一首	正确切换音乐并播放
把进度条拖到快结束的时候，等待结束	播放结束后自动播放下一首
点击DetailFragment中收藏按钮	弹出Toast 正确更新detailFragment的UI
点击“列表”按钮	正确显示播放列表 正确显示当前音乐的收藏状态
点击播放列表中当前歌曲的收藏按钮	弹出Toast 正确更新recyclerView的UI

步骤	预期结果
点击playBar	跳转到detailFragment 正确显示当前音乐的收藏状态
点击返回键	返回到播放列表
点击返回键	返回到HomeFragment
点击返回键	弹出AlertDialog
点击取消	AlertDialog消失
点击返回键	弹出AlertDialog
点击确认	退出App（音乐停止播放）
重新进入App	正确显示界面UI（playBar）
点击plarBar的暂停键	播放当前音乐
陆续点击plarbar的上一首、下一首	播放上一首、上一首 正确更新playBar的UI
退出App	显示自定义的logo和App名字