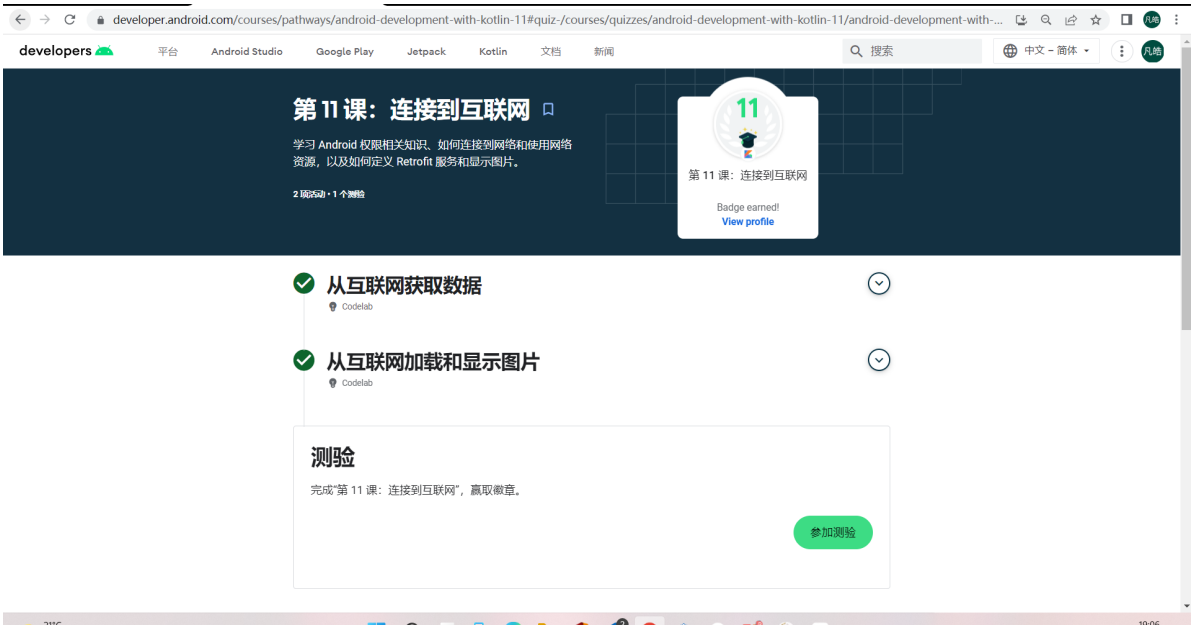
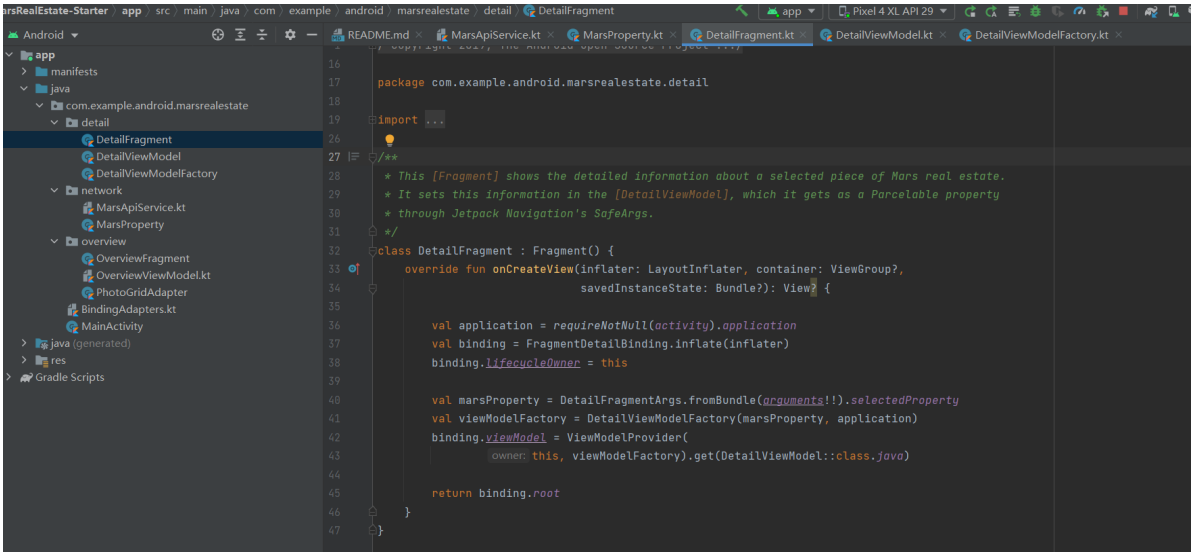
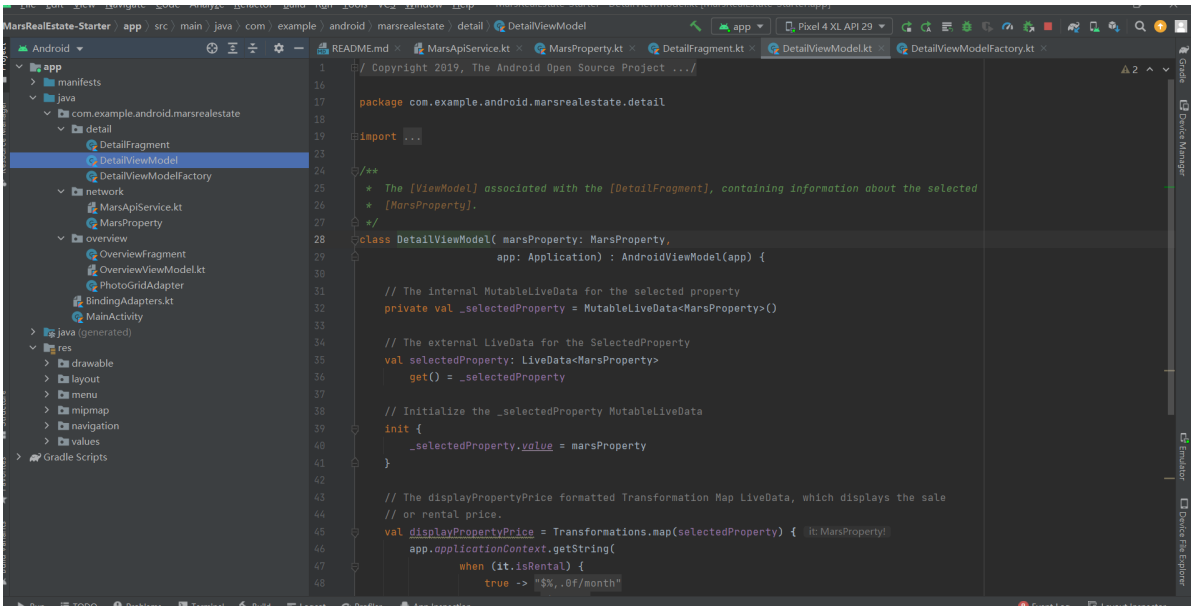
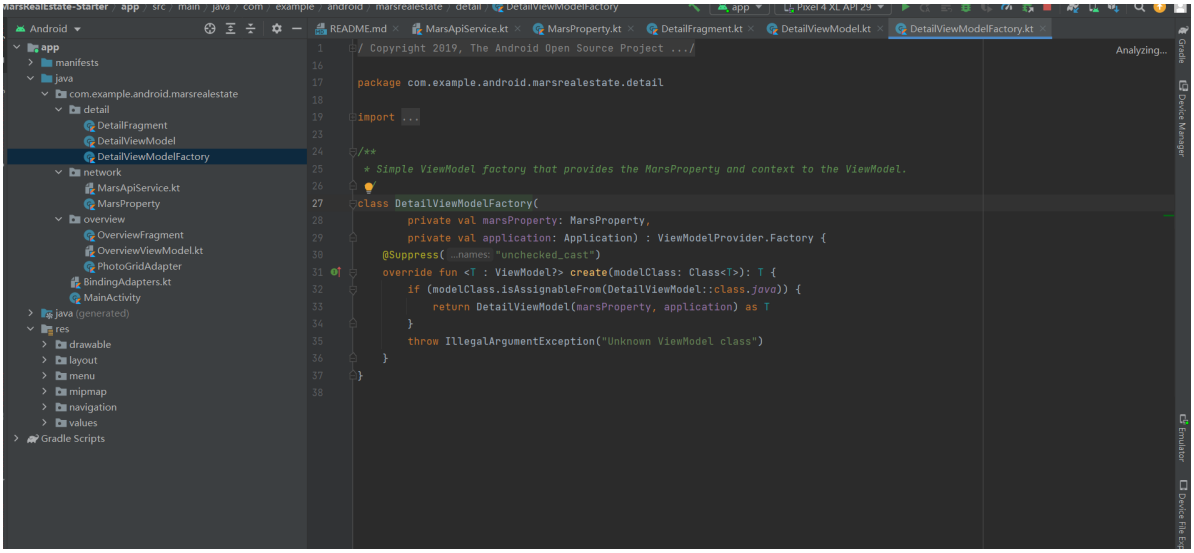
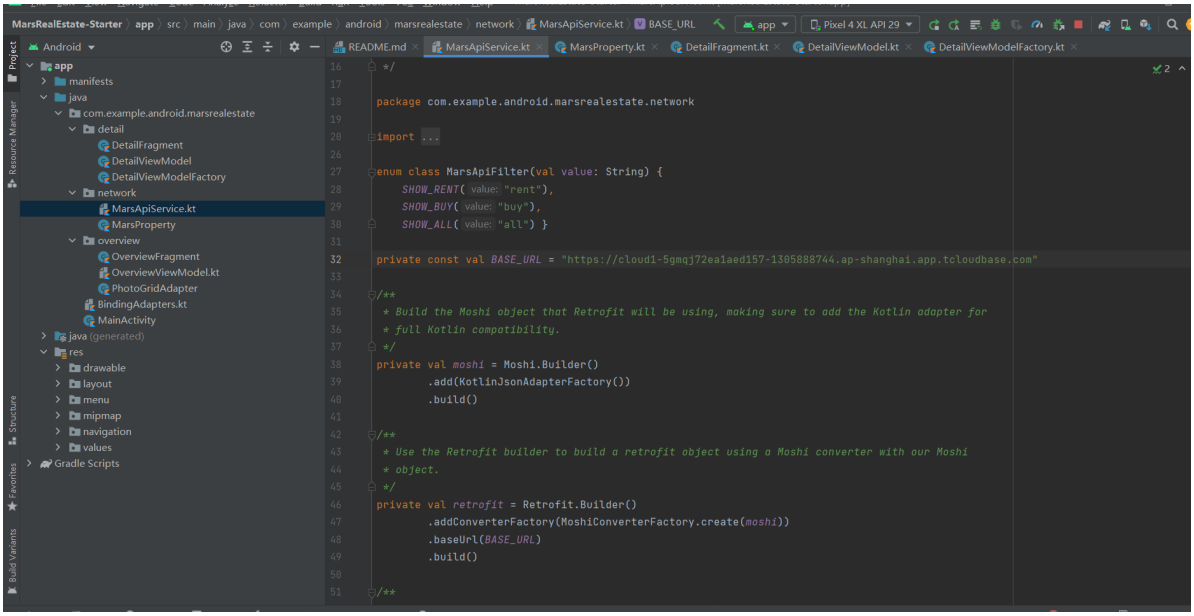

developer.android.com/courses/pathways/android-development-with-kotlin-11#quiz-/courses/quizzes/android-development-with-kotlin-11/android-development-with-...
developers
平台
Android Studio
Google Play
Jetpack
Kotlin
文档
新闻
搜索
中文 – 简体
第 11 课：连接到互联网
学习 Android 权限相关知识、如何连接到网络和使用网络资源，以及如何定义 Retrofit 服务和显示图片。
2 项活动 · 1 个测验
11
第 11 课：连接到互联网
Badge earned!
View profile
从互联网获取数据
Codelab
从互联网加载和显示图片
Codelab
测验
完成"第 11 课：连接到互联网"，赢取徽章。
参加测验


MarsRealEstate-Starter › app › src › main › java › com › example › android › marsrealestate › detail › DetailFragment
Android
app
manifests
java
com.example.android.marsrealestate
detail
DetailFragment
DetailViewModel
DetailViewModelFactory
network
MarsApiService.kt
MarsProperty
overview
OverviewFragment
OverviewViewModel.kt
PhotoGridAdapter
BindingAdapters.kt
MainActivity
java (generated)
res
Gradle Scripts
README.md
MarsApiService.kt
MarsProperty.kt
DetailFragment.kt
DetailViewModel.kt
DetailViewModelFactory.kt
Pixel 4 XL API 29
package com.example.android.marsrealestate.detail
import ...
/**
* This [Fragment] shows the detailed information about a selected piece of Mars real estate.
* It sets this information in the [DetailViewModel], which it gets as a Parcelable property
* through Jetpack Navigation's SafeArgs.
*/
class DetailFragment : Fragment() {
override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?): View? {
val application = requireNotNull(activity).application
val binding = FragmentDetailBinding.inflate(inflater)
binding.lifecycleOwner = this
val marsProperty = DetailFragmentArgs.fromBundle(arguments!!).selectedProperty
val viewModelFactory = DetailViewModelFactory(marsProperty, application)
binding.viewModel = ViewModelProvider(
owner: this, viewModelFactory).get(DetailViewModel::class.java)
return binding.root
}
}

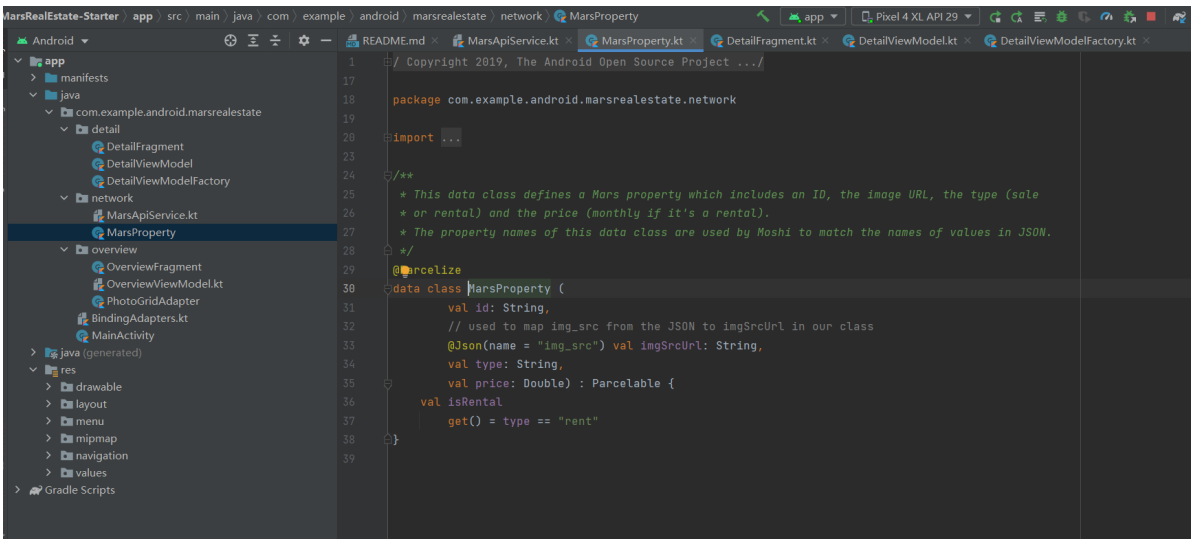

MarsRealEstate-Starter › app › src › main › java › com › example › android › marsrealestate › detail › DetailViewModel
Android
app
manifests
java
com.example.android.marsrealestate
detail
DetailFragment
DetailViewModel
DetailViewModelFactory
network
MarsApiService.kt
MarsProperty
overview
OverviewFragment
OverviewViewModel.kt
PhotoGridAdapter
BindingAdapters.kt
MainActivity
java (generated)
res
drawable
layout
menu
mipmap
navigation
values
Gradle Scripts
README.md
MarsApiService.kt
MarsProperty.kt
DetailFragment.kt
DetailViewModel.kt
DetailViewModelFactory.kt
Pixel 4 XL API 29
/ Copyright 2019, The Android Open Source Project .../
package com.example.android.marsrealestate.detail
import ...
/**
* The [ViewModel] associated with the [DetailFragment], containing information about the selected
* [MarsProperty].
*/
class DetailViewModel(marsProperty: MarsProperty,
app: Application) : AndroidViewModel(app) {
// The internal MutableLiveData for the selected property
private val _selectedProperty = MutableLiveData<MarsProperty>()
// The external LiveData for the SelectedProperty
val selectedProperty: LiveData<MarsProperty>
get() = _selectedProperty
// Initialize the _selectedProperty MutableLiveData
init {
_selectedProperty.value = marsProperty
}
// The displayPropertyPrice formatted Transformation Map LiveData, which displays the sale
// or rental price.
val displayPropertyPrice = Transformations.map(selectedProperty) { it: MarsProperty!
app.applicationContext.getString(
when (it.isRental) {
true -> "$%,.0f/month"



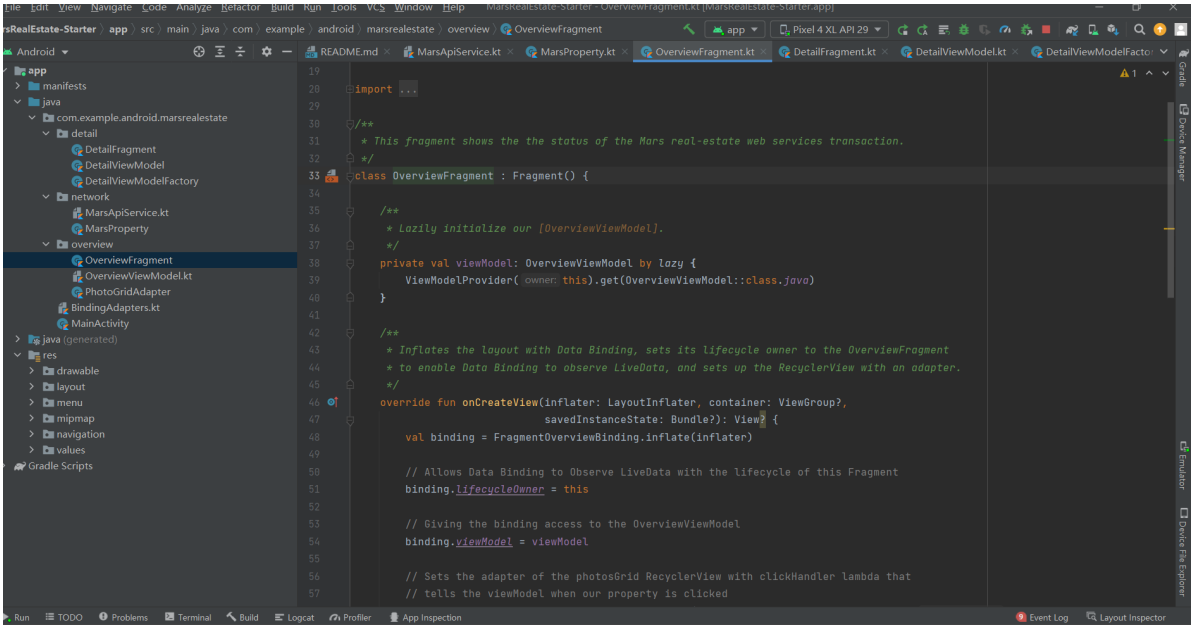
```
/ Copyright 2019, The Android Open Source Project .../

package com.example.android.marsrealestate.detail

import ...

/**
 * Simple ViewModel factory that provides the MarsProperty and context to the ViewModel.
 */
class DetailViewModelFactory(
        private val marsProperty: MarsProperty,
        private val application: Application) : ViewModelProvider.Factory {
    @Suppress( ...names: "unchecked_cast")
    override fun <T : ViewModel?> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(DetailViewModel::class.java)) {
            return DetailViewModel(marsProperty, application) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```



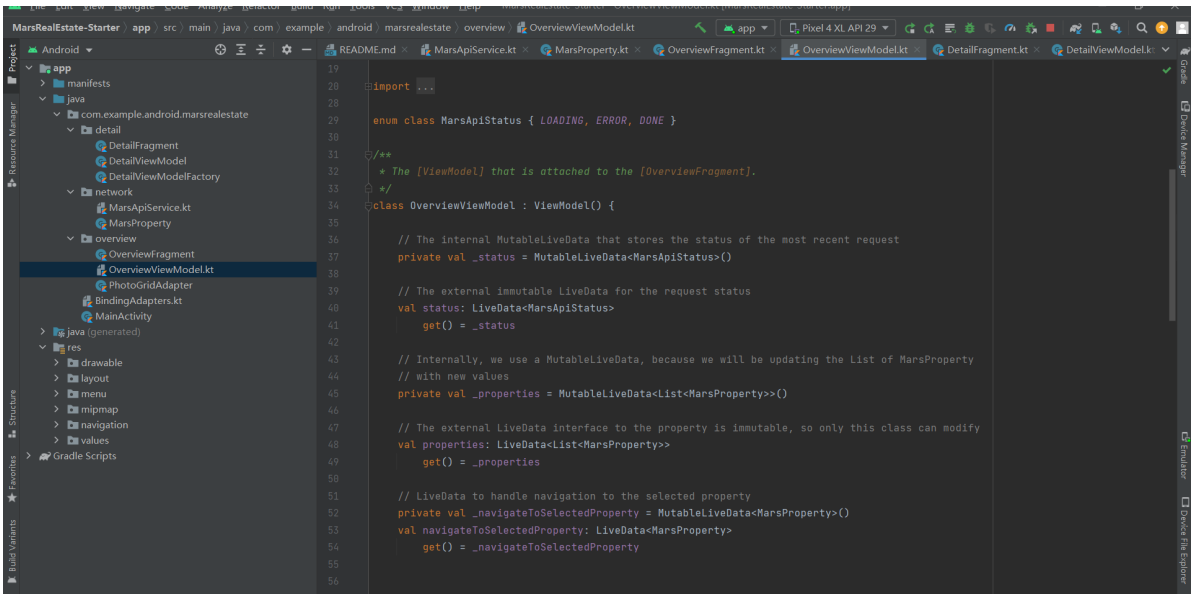
```
*/

package com.example.android.marsrealestate.network

import ...

enum class MarsApiFilter(val value: String) {
    SHOW_RENT( value: "rent"),
    SHOW_BUY( value: "buy"),
    SHOW_ALL( value: "all") }

private const val BASE_URL = "https://cloud1-5gmqj72ea1aed157-1305888744.ap-shanghai.app.tcloudbase.com"

/**
 * Build the Moshi object that Retrofit will be using, making sure to add the Kotlin adapter for
 * full Kotlin compatibility.
 */
private val moshi = Moshi.Builder()
        .add(KotlinJsonAdapterFactory())
        .build()

/**
 * Use the Retrofit builder to build a retrofit object using a Moshi converter with our Moshi
 * object.
 */
private val retrofit = Retrofit.Builder()
        .addConverterFactory(MoshiConverterFactory.create(moshi))
        .baseUrl(BASE_URL)
        .build()

/**
```



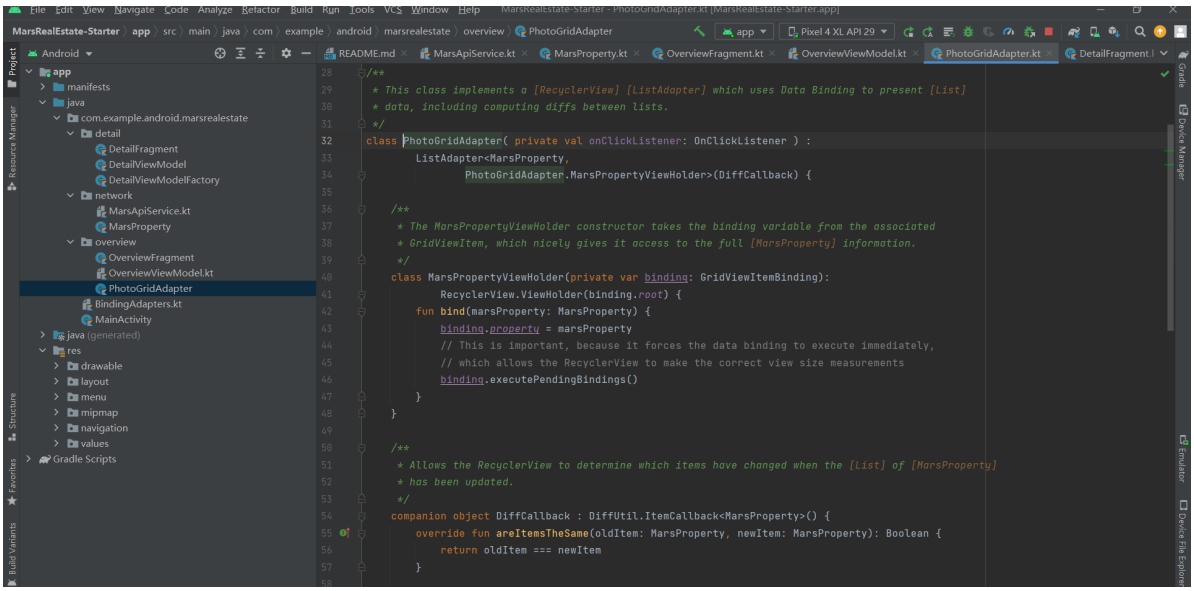
```
/ Copyright 2019, The Android Open Source Project .../

package com.example.android.marsrealestate.network

import ...

/**
 * This data class defines a Mars property which includes an ID, the image URL, the type (sale
 * or rental) and the price (monthly if it's a rental).
 * The property names of this data class are used by Moshi to match the names of values in JSON.
 */
@Parcelize
data class MarsProperty (
        val id: String,
        // used to map img_src from the JSON to imgSrcUrl in our class
        @Json(name = "img_src") val imgSrcUrl: String,
        val type: String,
        val price: Double) : Parcelable {
    val isRental
        get() = type == "rent"
}
```



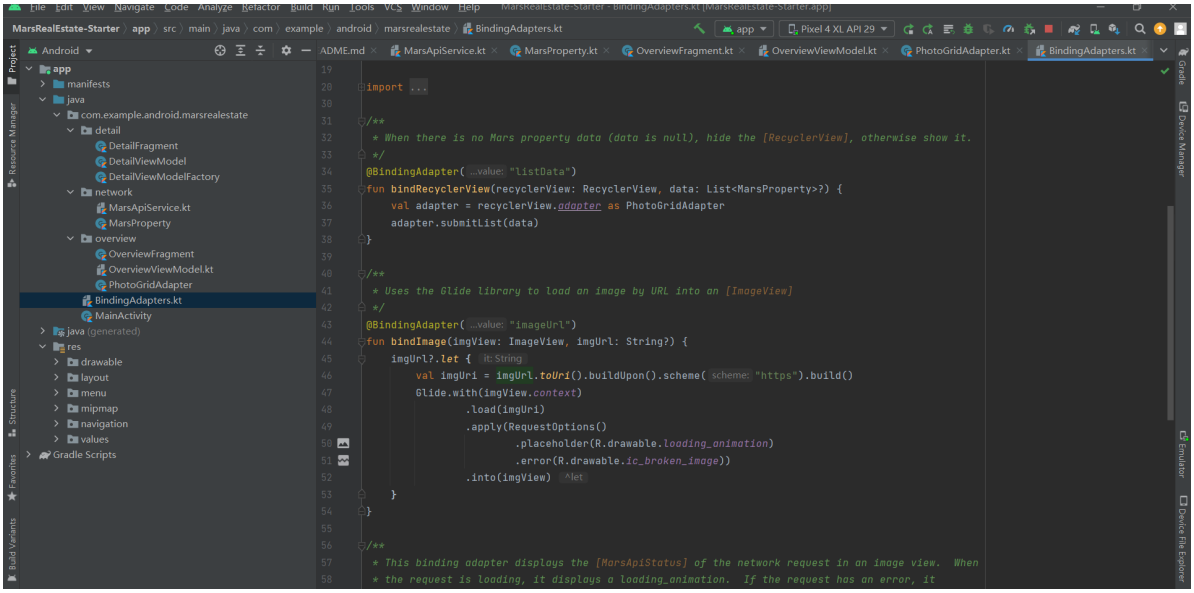
```
import ...

/**
 * This fragment shows the the status of the Mars real-estate web services transaction.
 */
class OverviewFragment : Fragment() {

    /**
     * Lazily initialize our [OverviewViewModel].
     */
    private val viewModel: OverviewViewModel by lazy {
        ViewModelProvider( owner: this).get(OverviewViewModel::class.java)
    }

    /**
     * Inflates the layout with Data Binding, sets its lifecycle owner to the OverviewFragment
     * to enable Data Binding to observe LiveData, and sets up the RecyclerView with an adapter.
     */
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {
        val binding = FragmentOverviewBinding.inflate(inflater)

        // Allows Data Binding to Observe LiveData with the lifecycle of this Fragment
        binding.lifecycleOwner = this

        // Giving the binding access to the OverviewViewModel
        binding.viewModel = viewModel

        // Sets the adapter of the photosGrid RecyclerView with clickHandler lambda that
        // tells the viewModel when our property is clicked
```



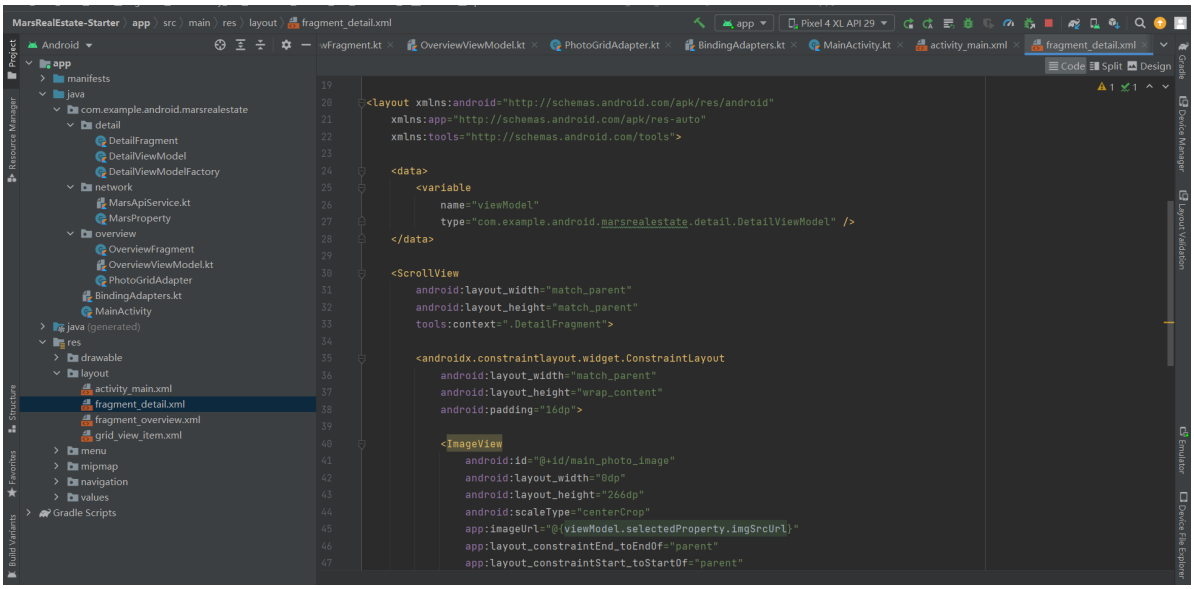
```
import ...

enum class MarsApiStatus { LOADING, ERROR, DONE }

/**
 * The [ViewModel] that is attached to the [OverviewFragment].
 */
class OverviewViewModel : ViewModel() {

    // The internal MutableLiveData that stores the status of the most recent request
    private val _status = MutableLiveData<MarsApiStatus>()

    // The external immutable LiveData for the request status
    val status: LiveData<MarsApiStatus>
        get() = _status

    // Internally, we use a MutableLiveData, because we will be updating the List of MarsProperty
    // with new values
    private val _properties = MutableLiveData<List<MarsProperty>>()

    // The external LiveData interface to the property is immutable, so only this class can modify
    val properties: LiveData<List<MarsProperty>>
        get() = _properties

    // LiveData to handle navigation to the selected property
    private val _navigateToSelectedProperty = MutableLiveData<MarsProperty>()
    val navigateToSelectedProperty: LiveData<MarsProperty>
        get() = _navigateToSelectedProperty
```



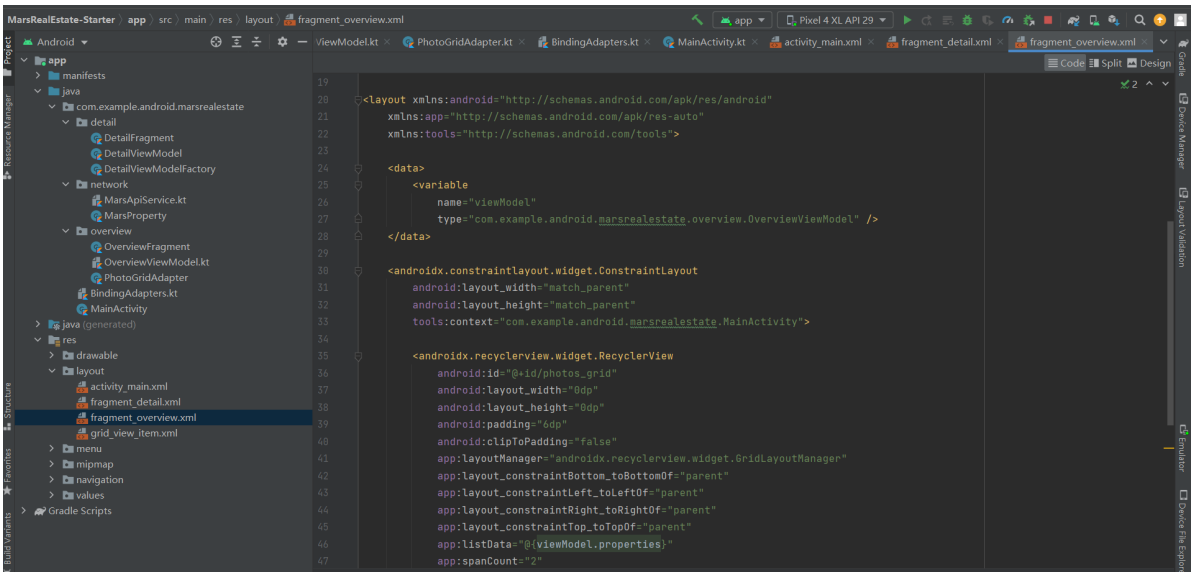
```
/**
 * This class implements a [RecyclerView] [ListAdapter] which uses Data Binding to present [List]
 * data, including computing diffs between lists.
 */
class PhotoGridAdapter( private val onClickListener: OnClickListener ) :
        ListAdapter<MarsProperty,
                PhotoGridAdapter.MarsPropertyViewHolder>(DiffCallback) {

    /**
     * The MarsPropertyViewHolder constructor takes the binding variable from the associated
     * GridViewItem, which nicely gives it access to the full [MarsProperty] information.
     */
    class MarsPropertyViewHolder(private var binding: GridViewItemBinding):
            RecyclerView.ViewHolder(binding.root) {
        fun bind(marsProperty: MarsProperty) {
            binding.property = marsProperty
            // This is important, because it forces the data binding to execute immediately,
            // which allows the RecyclerView to make the correct view size measurements
            binding.executePendingBindings()
        }
    }

    /**
     * Allows the RecyclerView to determine which items have changed when the [List] of [MarsProperty]
     * has been updated.
     */
    companion object DiffCallback : DiffUtil.ItemCallback<MarsProperty>() {
        override fun areItemsTheSame(oldItem: MarsProperty, newItem: MarsProperty): Boolean {
            return oldItem === newItem
        }
```



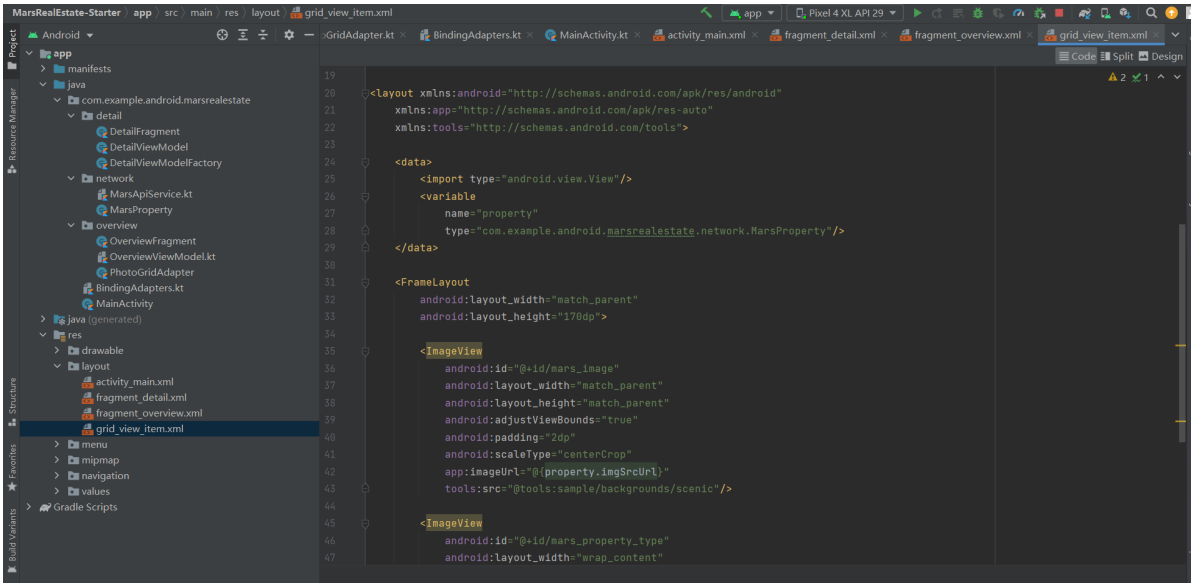
```kotlin
import ...

/**
 * When there is no Mars property data (data is null), hide the [RecyclerView], otherwise show it.
 */
@BindingAdapter("listData")
fun bindRecyclerView(recyclerView: RecyclerView, data: List<MarsProperty>?) {
    val adapter = recyclerView.adapter as PhotoGridAdapter
    adapter.submitList(data)
}

/**
 * Uses the Glide library to load an image by URL into an [ImageView]
 */
@BindingAdapter("imageUrl")
fun bindImage(imgView: ImageView, imgUrl: String?) {
    imgUrl?.let {
        val imgUri = imgUrl.toUri().buildUpon().scheme("https").build()
        Glide.with(imgView.context)
                .load(imgUri)
                .apply(RequestOptions()
                        .placeholder(R.drawable.loading_animation)
                        .error(R.drawable.ic_broken_image))
                .into(imgView)
    }
}

/**
 * This binding adapter displays the [MarsApiStatus] of the network request in an image view.  When
 * the request is loading, it displays a loading_animation.  If the request has an error, it
```



```xml
<layout 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">

    <data>
        <variable
            name="viewModel"
            type="com.example.android.marsrealestate.detail.DetailViewModel" />
    </data>

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".DetailFragment">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="16dp">

            <ImageView
                android:id="@+id/main_photo_image"
                android:layout_width="0dp"
                android:layout_height="266dp"
                android:scaleType="centerCrop"
                app:imageUrl="@{viewModel.selectedProperty.imgSrcUrl}"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
```



```xml
<layout 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">

    <data>
        <variable
            name="viewModel"
            type="com.example.android.marsrealestate.overview.OverviewViewModel" />
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context="com.example.android.marsrealestate.MainActivity">

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/photos_grid"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:padding="6dp"
            android:clipToPadding="false"
            app:layoutManager="androidx.recyclerview.widget.GridLayoutManager"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:listData="@{viewModel.properties}"
            app:spanCount="2"
```


MarsRealEstate-Starter
app
src
main
res
layout
grid_view_item.xml
Pixel 4 XL API 29
Android
GridAdapter.kt
BindingAdapters.kt
MainActivity.kt
activity_main.xml
fragment_detail.xml
fragment_overview.xml
grid_view_item.xml
Code
Split
Design
app
manifests
java
com.example.android.marsrealestate
detail
DetailFragment
DetailViewModel
DetailViewModelFactory
network
MarsApiService.kt
MarsProperty
overview
OverviewFragment
OverviewViewModel.kt
PhotoGridAdapter
BindingAdapters.kt
MainActivity
java (generated)
res
drawable
layout
activity_main.xml
fragment_detail.xml
fragment_overview.xml
grid_view_item.xml
menu
mipmap
navigation
values
Gradle Scripts
<layout 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">
<data>
<import type="android.view.View"/>
<variable
name="property"
type="com.example.android.marsrealestate.network.MarsProperty"/>
</data>
<FrameLayout
android:layout_width="match_parent"
android:layout_height="170dp">
<ImageView
android:id="@+id/mars_image"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:adjustViewBounds="true"
android:padding="2dp"
android:scaleType="centerCrop"
app:imageUrl="@{property.imgSrcUrl}"
tools:src="@tools:sample/backgrounds/scenic"/>
<ImageView
android:id="@+id/mars_property_type"
android:layout_width="wrap_content"