برنامه نویسی دستگاه های سیار (CE364)

جلسه چهاردهم:
دریافت اطلاعات از اینترنت

سجاد شیرعلی شهرضا

پاییز 1401

دوشنبه، 28 آذر 1401

اطلاع رسانی

- بخشهای مرتبط با این جلسه:

- Unit 4: Connect to the internet:
  - Pathway 2: Get and display data from the internet

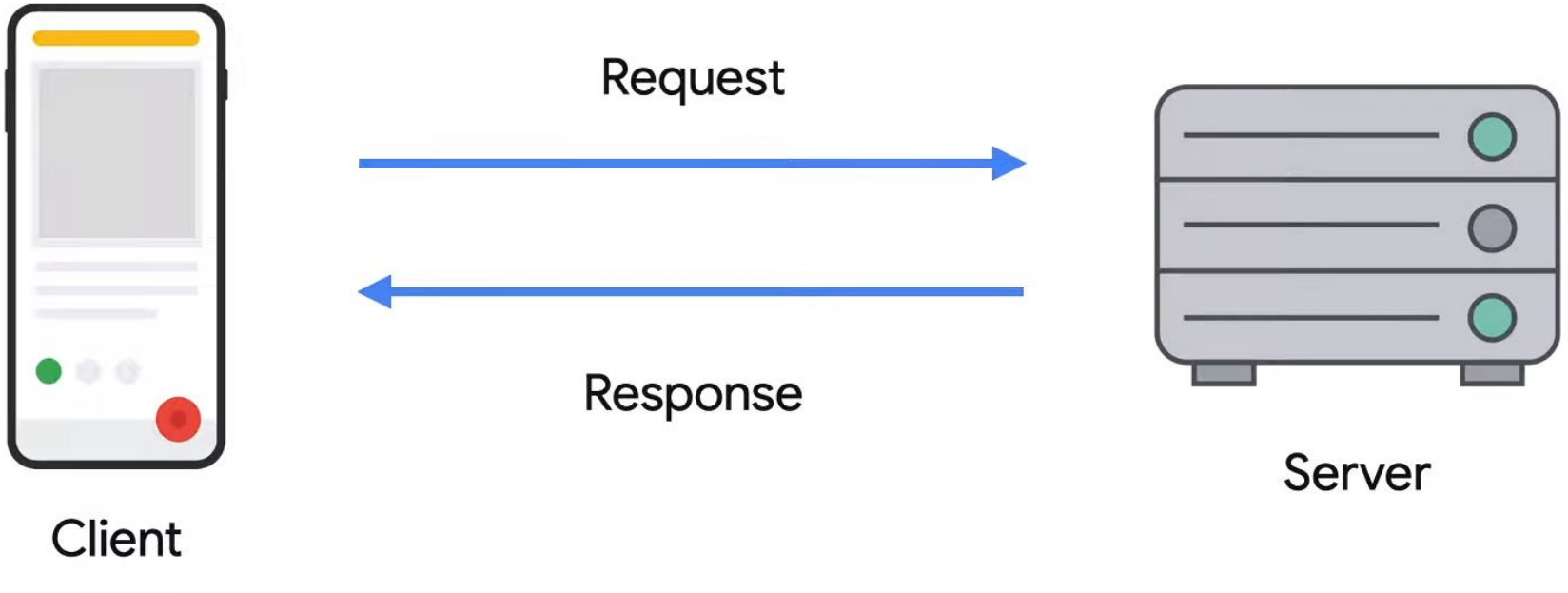
سوال؟

تعامل با اینترنت

نیاز به تعامل

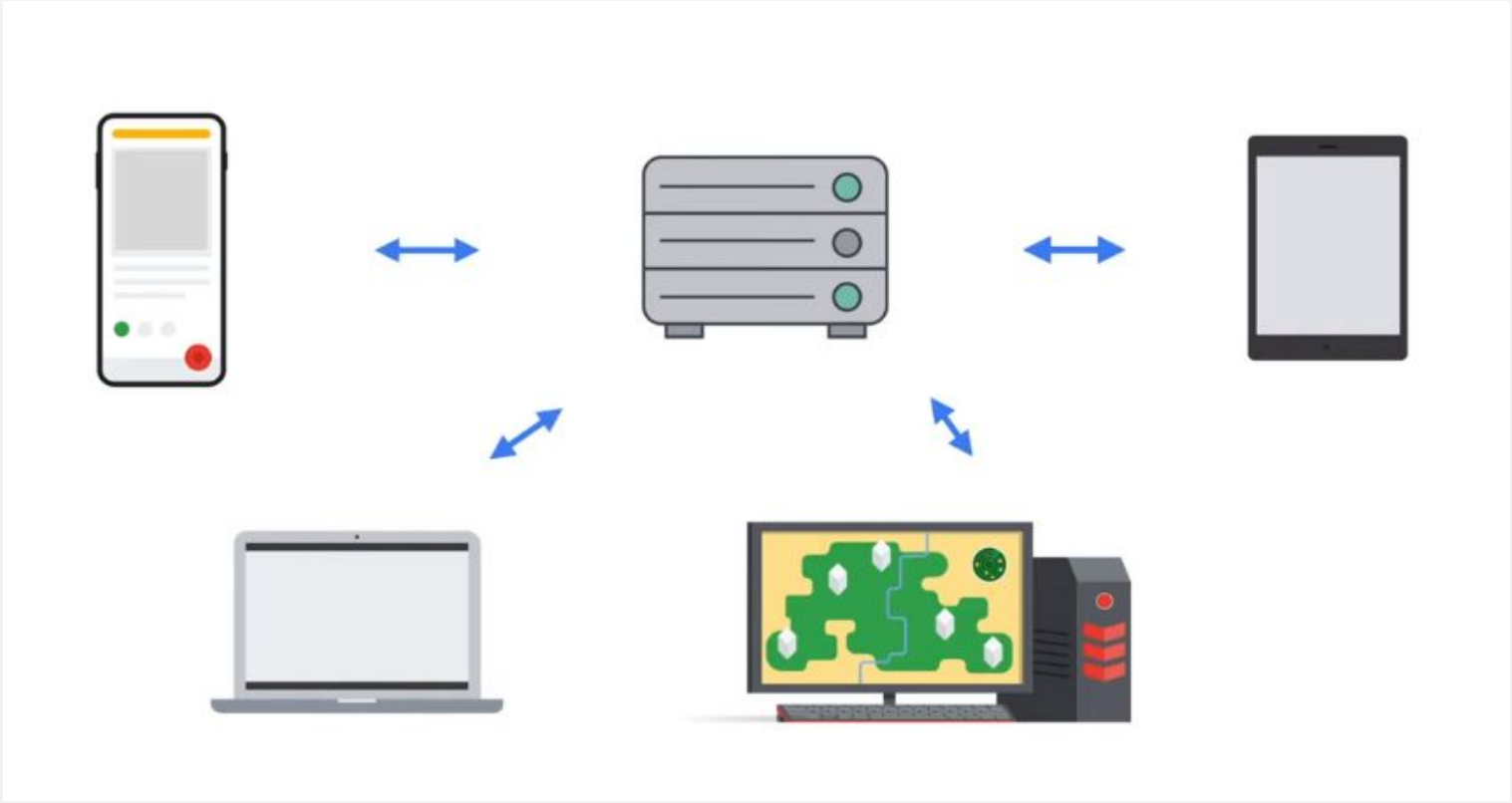
- بسیاری از برنامه ها نیاز به دریافت/ارسال اطلاعات دارند
  - مدیریت ایمیل
  - خواندن اخبار
  - خرید اینترنتی
  - شبکه اجتماعی

ساختار کلی تعامل مشتری (client) با یک خدمتگذار (server)



مثال: خودپرداز بانک

مشتری های مختلف

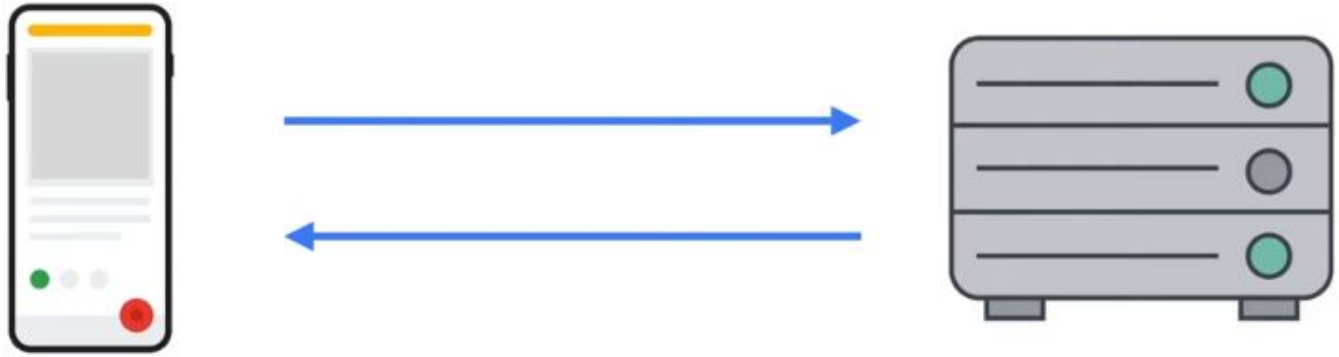


درخواست HTTP

پروتکل HTTP

HTTP

HyperText Transfer Protocol

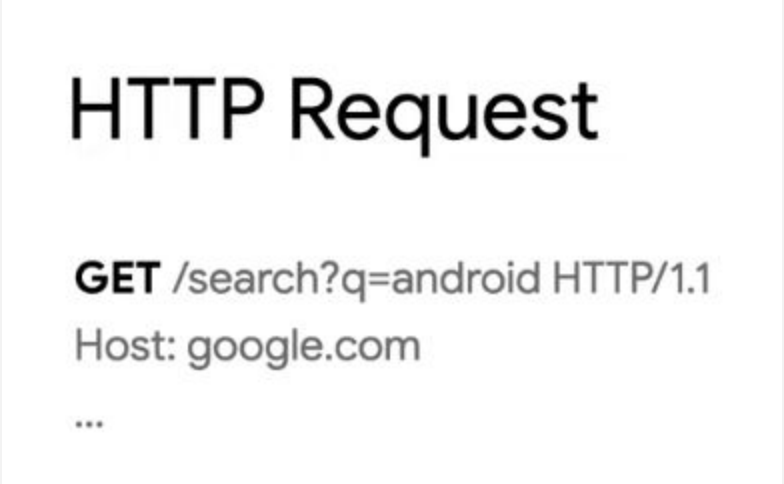


نمونه یک درخواست HTTP

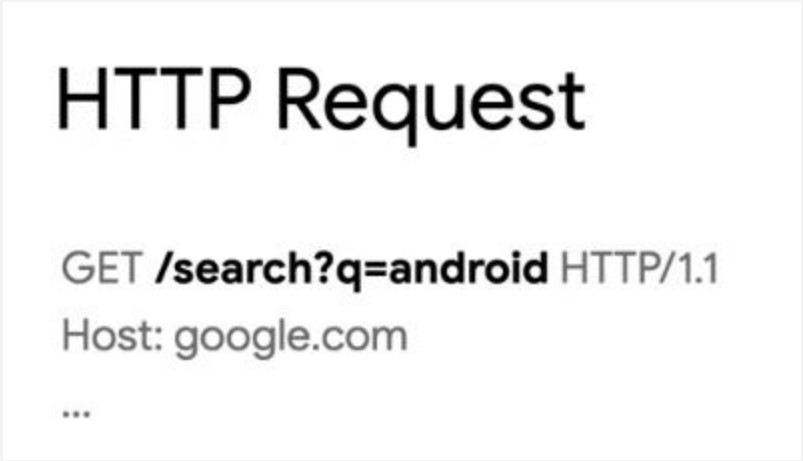
HTTP Request

GET /search?q=android HTTP/1.1
Host: google.com
...

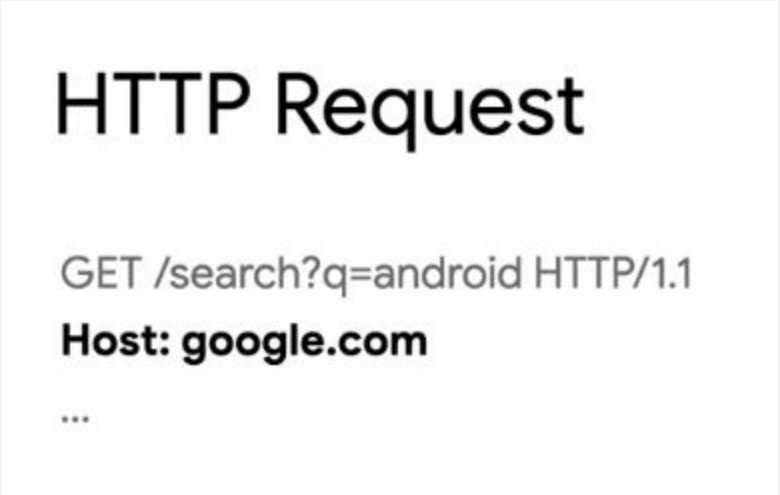
نوع درخواست



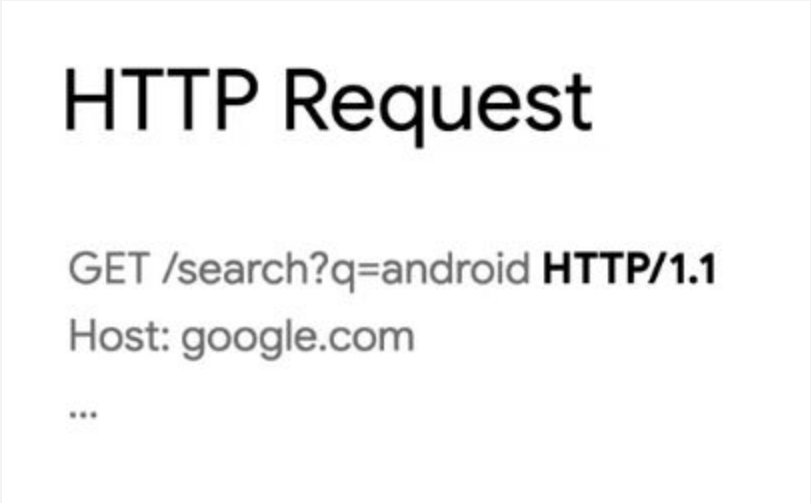
پاسخگَو (Endpoint)



دامنه هدف



نسخه پروتکل



اطلاعات اضافه در قالب سربرگ (header)

HTTP Request

GET /search?q=android HTTP/1.1

Host: google.com

Cache-Control: no-cache

Accept: text/html, application/xhtml+xml ...

Accept-Language: en

...

https://google.com/search?q=android

بخش های مختلف یک آدرس (URL)

https://google.com/search?q=android

https://**google.com**/search?q=android

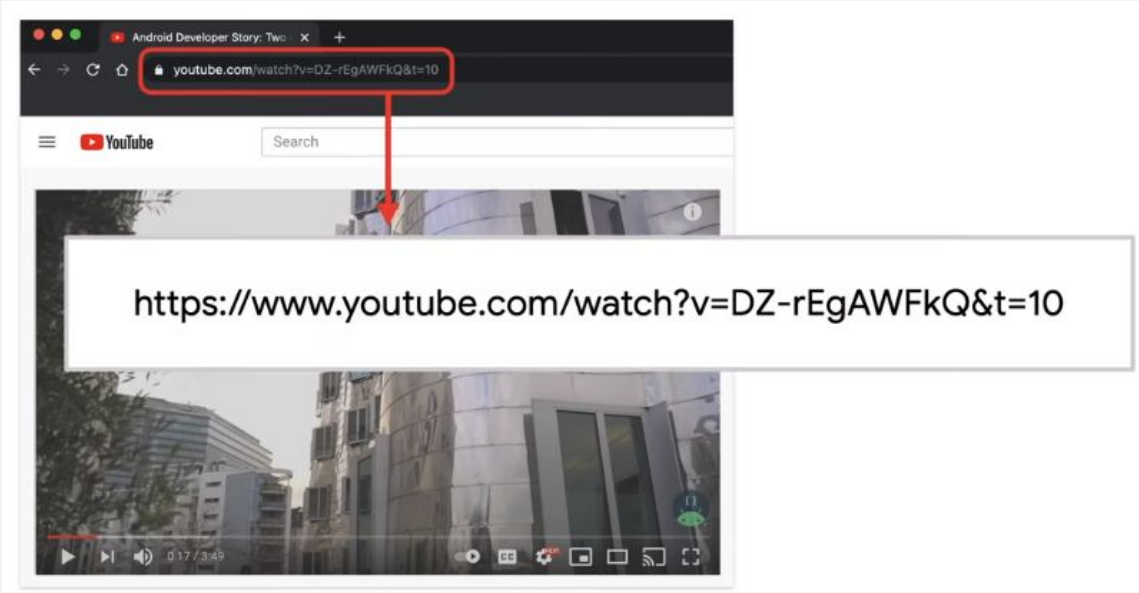
https://google.com/**search**?q=android

https://google.com/search**?q=android**

- پروتکل
- دامنه
- منبع هدف
- اطلاعات ارسالی

درخواست گرفتن (GET)

- یکی از رایج ترین درخواست های مورد استفاده
- برای دریافت اطلاعات
- امکان ارسال چندین مقدار



درخواست ثبت (POST)

- برای اضافه کردن یک منبع جدید به خدمتگذار
- هدف: ارسال داده به خدمتگذار(مثلا یک فایل)
- درخواست تا حدودی مشابه: PUT
  - برای جایگزینی یک داده قدیمی با داده جدید
- درخواست معادل حذف: DELETE

مثال استفاده از درخواست های مختلف در یک برنامه

Social media app

GET	**Get list of all recent posts**
POST	
PUT	
DELETE	

مثال استفاده از درخواست های مختلف در یک برنامه

Social media app

GET	Get list of all recent posts
POST	**Add a new post**
PUT	
DELETE	

مثال استفاده از درخواست های مختلف در یک برنامه

Social media app

GET	Get list of all recent posts
POST	Add a new post
PUT	**Update an existing post**
DELETE	

مثال استفاده از درخواست های مختلف در یک برنامه

Social media app

GET	Get list of all recent posts
POST	Add a new post
PUT	Update an existing post
DELETE	**Delete a post**

پاسخ به درخواست

HTTP Response

HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Encoding: br
Date: Fri, 12 Feb 2021 01:59:01 GMT
Server: gws
Cache-Control: private, max-age=0
...

<!DOCTYPE html> ...

نتیجه درخواست

HTTP Response

HTTP/1.1 **200** OK
Content-Type: text/html; charset=UTF-8
Content-Encoding: br
Date: Fri, 12 Feb 2021 01:59:01 GMT
Server: gws
Cache-Control: private, max-age=0
...

<!DOCTYPE html> ...

HTTP Response

HTTP/1.1 200 **OK**
Content-Type: text/html; charset=UTF-8
Content-Encoding: br
Date: Fri, 12 Feb 2021 01:59:01 GMT
Server: gws
Cache-Control: private, max-age=0
...

<!DOCTYPE html> ...

اطلاعات اصلی درخواست شده و اطلاعات اضافه در سربرگ

HTTP Response

HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Encoding: br
Date: Fri, 12 Feb 2021 01:59:01 GMT
Server: gws
Cache-Control: private, max-age=0
...

<!DOCTYPE html> ...

HTTP Response

HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Encoding: br
Date: Fri, 12 Feb 2021 01:59:01 GMT
Server: gws
Cache-Control: private, max-age=0
...

<!DOCTYPE html> ...

نوع اطلاعات بازگردانده شده

HTTP Response

HTTP/1.1 200 OK

Content-Type: text/html; charset=UTF-8

Content-Encoding: br

Date: Fri, 12 Feb 2021 01:59:01 GMT

Server: gws

Cache-Control: private, max-age=0

...

<!DOCTYPE html> ...

کدهای مشخص کننده حالت معروف

- 200 تا 299 : موفقیت
- 400 تا 499: خطاهای مشتری
  - مثال معروف: 404 به معنای پیدا نکردن یک صفحه
- 500 تا 599: خطاهای خدمتگذار
  - مثال معروف: 500 به معنای خطای داخلی سرور

REST (REpresentational State Transfer)

- یک معماری برای تعامل مشتری با خدمتگذار
- ارسال درخواست توسط مشتری
- آماده کردن و بازگرداندن جواب توسط خدمتگذار
- مشخص کردن منابع به صورت Uniform Resource Identifier یا URI
- یک رابط واحد برای درخواست های مختلف (گرفتن، ثبت، حذف)
- بدون نگه داری وضعیت (Stateless)

مثال از یک رابط REST

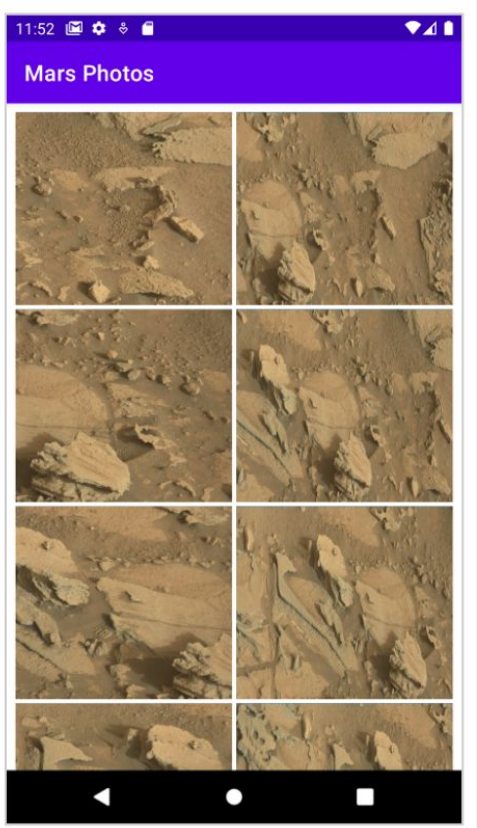
URI	DESCRIPTION	METHOD
example.com/messages	Get list of all messages	GET
example.com/messages/1	Get the message with given ID	GET
example.com/search?filter=queryterm	Search messages with given query	GET
example.com/messages/new	Create a new message	POST

سوال؟

دریافت عکس

هدف برنامه

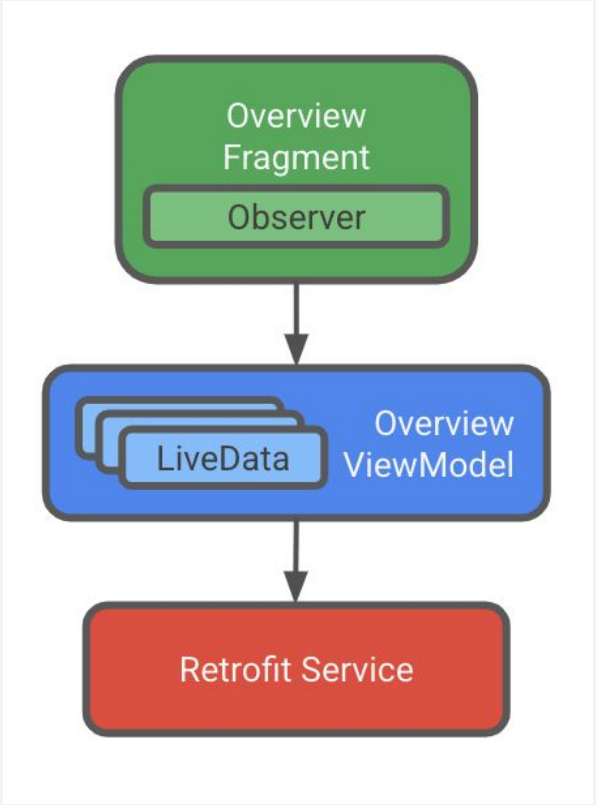
- دریافت عکس های سطح مریخ
- نمایش آنها به کاربر
- تمرکز بر روی دریافت اطلاعات



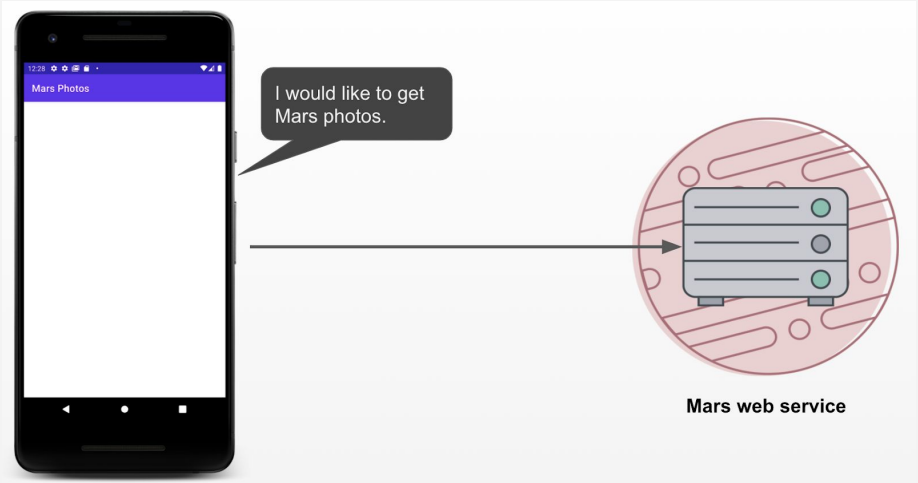
ساختار برنامه اولیه

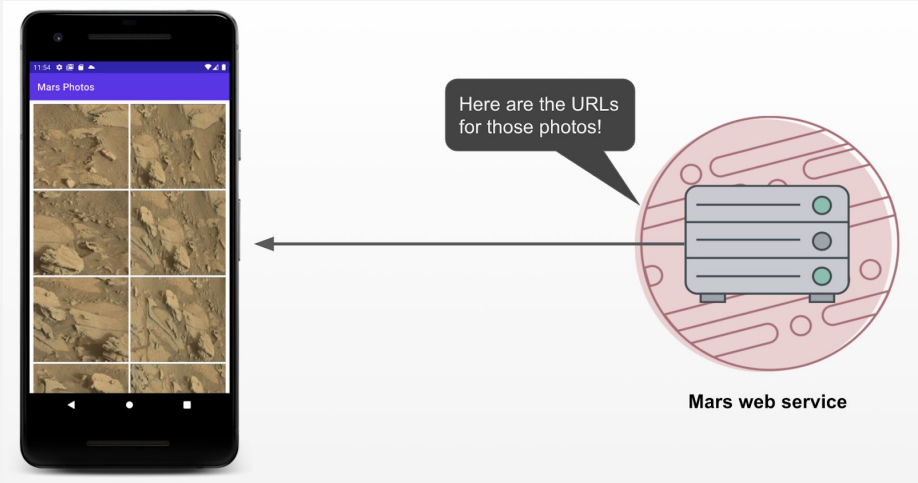
- قطعه نمایش دهنده کل اطلاعات: OverviewFragment
  - استفاده از داده زنده (LiveData)
- مدل نگه دارنده اطلاعات: OverviewViewModel
  - مدل نما برای قطعه
- چیدمان نمایش اطلاعات res/layout/fragment_overview.xml
- کار (task) اصلی برنامه : MainActivity

ساختار کلی برنامه نهایی



چگونگی کارکرد برنامه





خدمتگَذار مورد استفاده

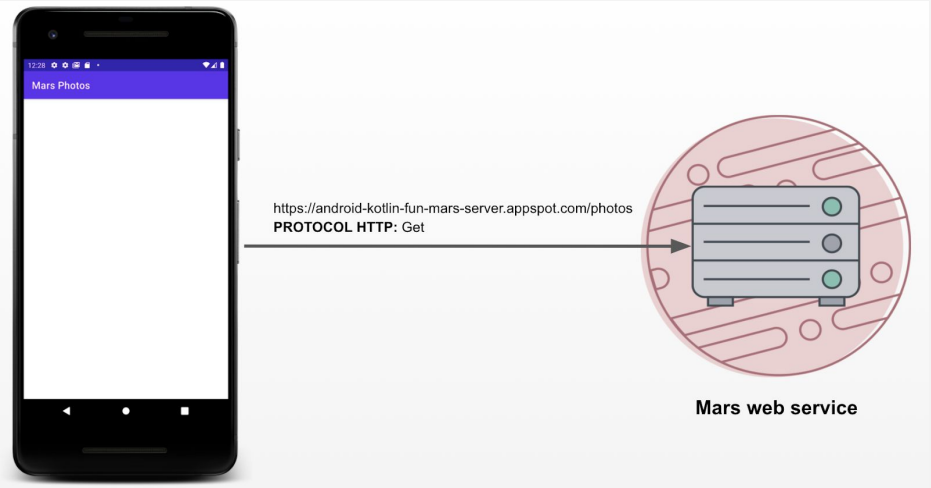
- آدرس خدمتگذار: https://android-kotlin-fun-mars-server.appspot.com
- درگاه های موجود: /realestate و /photos
- برگرداندن اطلاعات در قالب JSON
- گرفتن لیست املاک موجود:

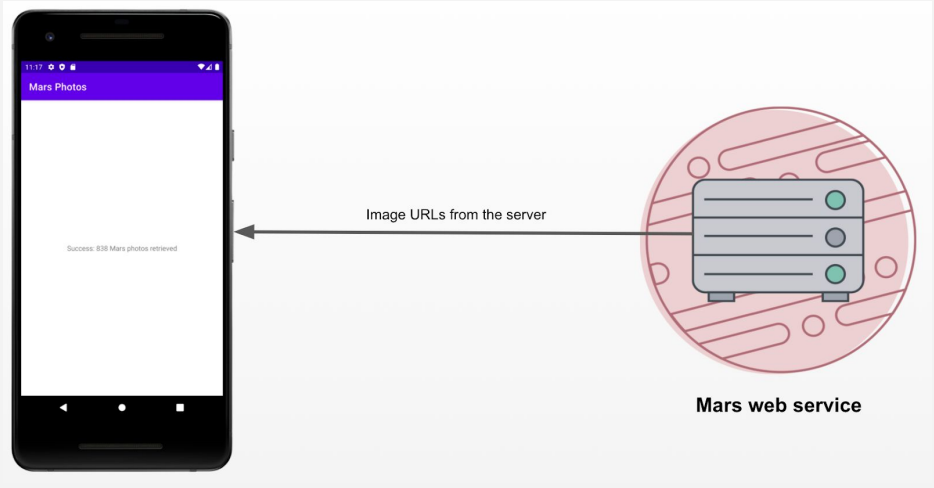
https://android-kotlin-fun-mars-server.appspot.com/realestate

- گرفتن لیست تصاویر موجود:

https://android-kotlin-fun-mars-server.appspot.com/photos

گرفتن لیست تصاویر

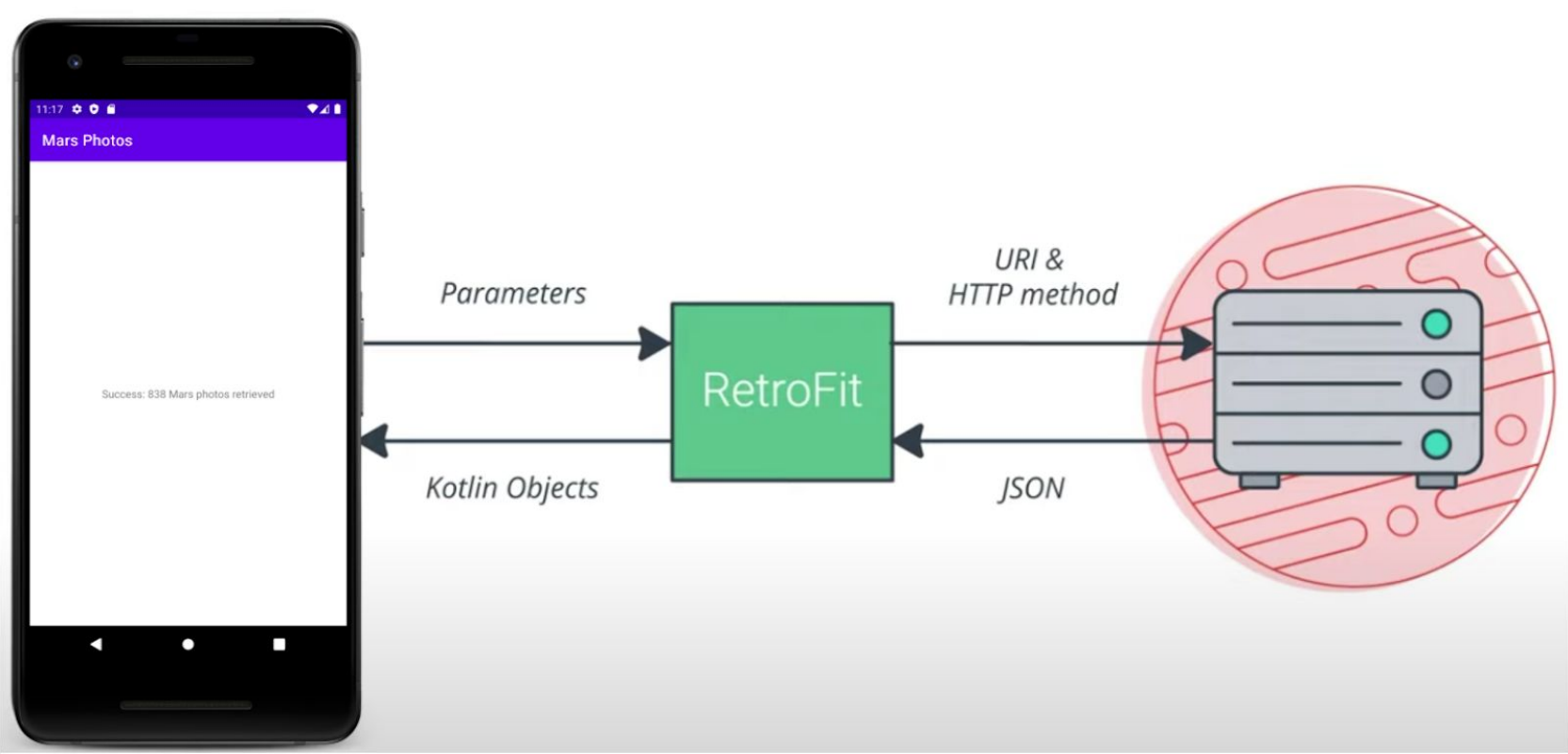




استفاده از کتابخانه های آماده

- صرفه جویی در زمان و هزینه
- امکان استفاده از کد با کیفیت
- نیاز به توجه به مخاطرات آن
  - امکان اضافه کردن کد ناخواسته به برنامه
- بررسی کتابخانه
  - افرادی که با آن مرتبط هستند
  - میزان فعالیت و نگه داری
  - تعداد و وضعیت مشکلات باز (issues)

کتابخانه RetroFit برای تعامل با خدمتگذار



اضافه کردن کتابخانه به پروژه

- اضافه کردن منبع برای دریافت کتابخانه در تنظیمات
  - Build.gradle : Project: MarsPhotos

```
repositories {
    google()
    jcenter()
}
```

- اضافه کردن پیش نیازها
  - build.gradle : Module: MarsPhots.app

```
// Retrofit
implementation "com.squareup.retrofit2:retrofit:2.9.0"
// Retrofit with Moshi Converter
implementation "com.squareup.retrofit2:converter-scalars:2.9.0"
```

فعال کردن پشتیبانی از نسخه 8 جاوا

- اطمینان از فعال بودن آن
  - build.gradle : Module: MarsPhotos.app

```
android {
  ...

  compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
  }

  kotlinOptions {
    jvmTarget = '1.8'
  }
}
```

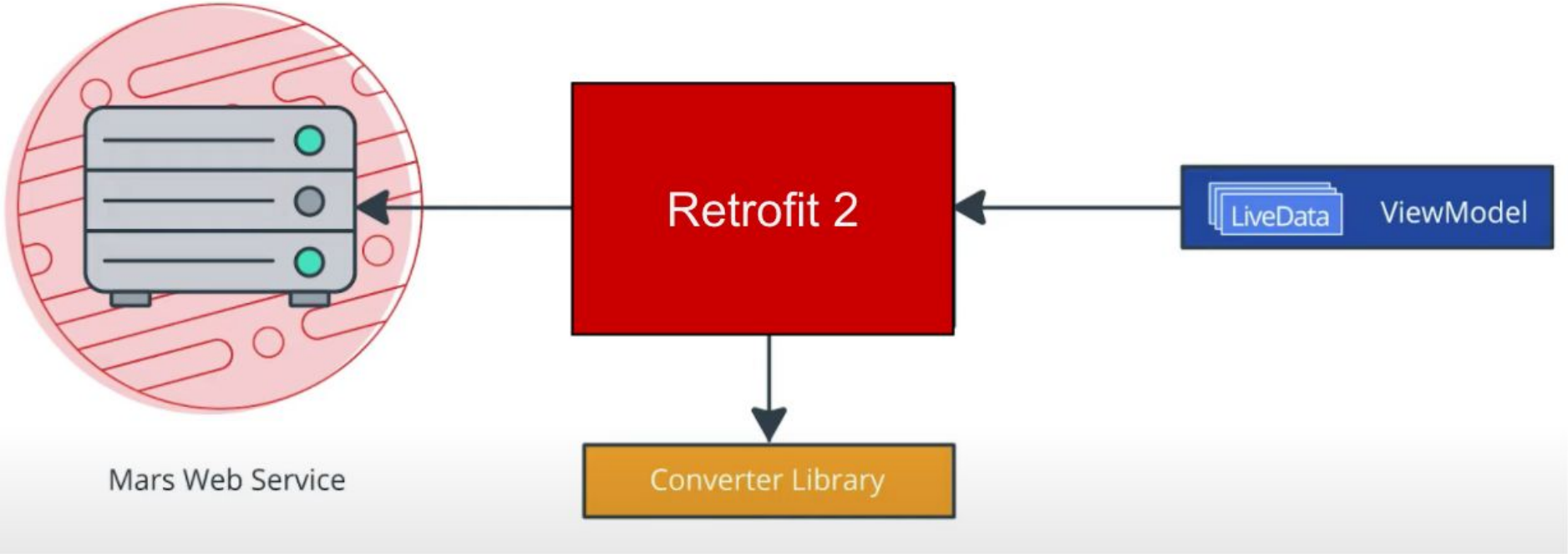
سوال؟

دریافت اطلاعات

هدف اولیه: دریافت اطلاعات خام

- ایجاد ارتباط با خدمتگذار
- ارسال درخواست گرفتن اطلاعات
- نمایش اطلاعات خام دریافتی (JSON) به صورت یک رشته
- ایجاد اشیا و لایه جدید برای استفاده از کتابخانه RetroFit

چگونگی استفاده از کتابخانه RetroFit



تعریف یک نمونه از RetroFit

- نیاز به تعریف دو مورد:
  - آدرس پایه سرویس مورد نظر
  - یک مبدل برای تبدیل اطلاعات دریافتی
- تعریف آدرس:

```
private const val BASE_URL =
    "https://android-kotlin-fun-mars-server.appspot.com"
```

- یک مبدل ساده جهت تبدیل به رشته: ScalarsConverterFactory
- تعریف نمونه از RetroFit:

```
private val retrofit = Retrofit.Builder()
    .addConverterFactory(ScalarsConverterFactory.create())
    .baseUrl(BASE_URL)
    .build()
```

تعریف نحوه تعامل با سرویس

```
interface MarsApiService {
    @GET("photos")
    fun getPhotos(): String
}
```

- تعریف یک رابط (interface)
  - مشخص کردن نقطه دسترسی
  - مشخص کردن نوع ارسال درخواست
  - مشخص کردن نوع جواب

تعریف شی (Object Decleration)

- استفاده جهت تعریف شی یگانه (singleton)

```
// Object declaration
object DataProviderManager {
    fun registerDataProvider(provider: DataProvider) {
        // ...
    }

    val allDataProviders: Collection<DataProvider>
        get() = // ...
}

// To refer to the object, use its name directly.
DataProviderManager.registerDataProvider(...)
```

آماده سازی شی RetroFit

- تعریف به صورت یگانه
  - عدم نیاز به چندین ارتباط همزمان با خدمتگذار
  - هزینه زیاد ایجاد

```
object MarsApi {
    val retrofitService : MarsApiService by lazy {
        retrofit.create(MarsApiService::class.java) }
}
```

حوزه مرتبط با مدل نما

- حوزه کوروتین ViewModelScope
- تعریف شده برای هر مدل نما (ViewModel)
- لغو خودکار کوروتین های اجرا شده به هنگام پاکسازی مدل نما

```
private fun getMarsPhotos() {
    viewModelScope.launch {
    }
}
```

اجرای درخواست در پس زمینه

- تغییر تابع دریافت اطلاعات به suspend

```
@GET("photos")
suspend fun getPhotos(): String
```

- ارسال درخواست

```
viewModelScope.launch {
    val listResult = MarsApi.retrofitService.getPhotos()
    _status.value = listResult
}
```

سوال؟

اجازه دسترسی به اینترنت

خطای عدم دسترسی به اینترنت

```
    --------- beginning of crash
22803-22865/com.example.android.marsphotos E/AndroidRuntime: FATAL EXCEPTION: OkHtt
    Process: com.example.android.marsphotos, PID: 22803
    java.lang.SecurityException: Permission denied (missing INTERNET permission?)
...
```

اجازه (Permission)

- هدف: محافظت از حریم شخصی کاربر
- نیاز به تعریف صریح اجازه های مورد نیاز
- درخواست از کاربر برای دریافت آن

تعریف اجازه دسترسی به اینترنت

- نیاز به اجازه INTERNET
- به صورت پیش فرض، فعال نیست
- استفاده از تگ <uses-permission> در فایل AndroidManifest
- قرار دادن پس از تگ <application>

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

نمونه اجرای برنامه فعلی

نحوه برخورد با خطا حین دریافت اطلاعات

- مثال: عدم دسترسی به اینترنت
  - مثلا در حالت هواپیما بودن گوشی

```
3302-3302/com.example.android.marsphotos E/AndroidRuntime: FATAL EXCEPTION: main
    Process: com.example.android.marsphotos, PID: 3302
    java.net.SocketTimeoutException: timeout
...
```

استثناء (Exception)

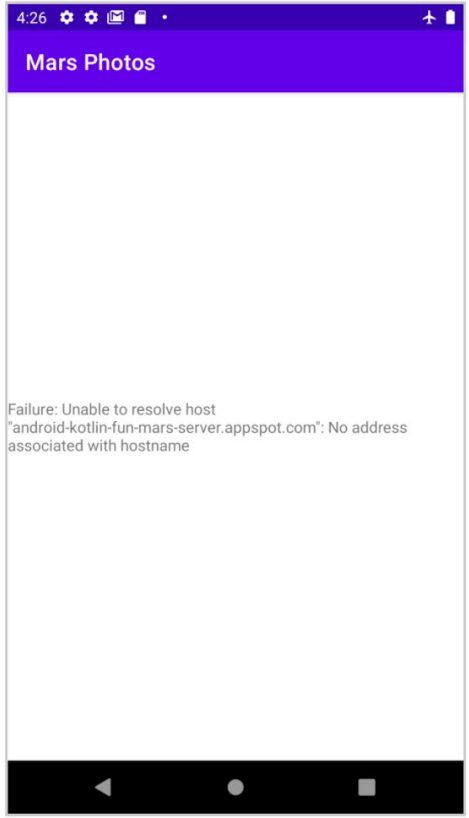
- خطایی که در حین اجرای برنامه ممکن است رخ دهد
  - در مقابل خطای زمان کامپایل
- عدم برخورد مناسب با آن
  - اتمام غیر قابل انتظار برنامه
  - کاهش میزان رضایت کاربر
- استفاده از ساختار try-catch

```
try {
    // some code that can cause an exception.
}
catch (e: SomeException) {
    // handle the exception to avoid abrupt termination.
}
```

پردازش خطا در هنگام ارسال درخواست

```
viewModelScope.launch {
    try {
        val listResult = MarsApi.retrofitService.getPhotos()
        _status.value = listResult
    } catch (e: Exception) {
      _status.value = "Failure: ${e.message}"
    }
}
```

نمونه پردازش خطا در هنگام دریافت جواب

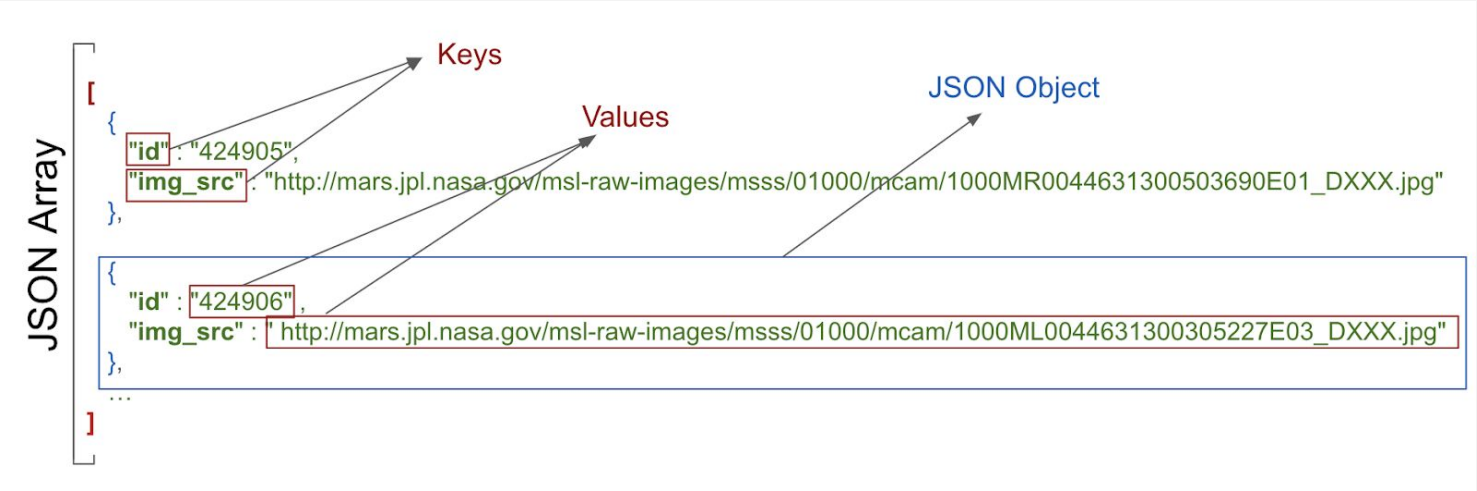


سوال؟

پردازش جواب

فرمت JSON

- جواب در قالب JSON: آرایه ای از اشیاء JSON
- JSON: JavaScript Object Notation
- هر شی JSON:
  - مجموعه ای از جفت های نام-مقدار



پردازش اطلاعات JSON

- استفاده از کتابخانه Moshi
  - یک کتابخانه برای پردازش و تبدیل جواب JSON به مجموعه ای از اشیاء کاتلین
- آماده سازی استفاده از آن:
  - اضافه کردن کتابخانه در بخش پیش نیازهادر فایل build.gradle

```
// Moshi
implementation 'com.squareup.moshi:moshi-kotlin:1.9.3'
```

- اضافه کردن مبدل برای RetroFit

```
// Retrofit with Moshi Converter
implementation 'com.squareup.retrofit2:converter-moshi:2.9.0'
```

تعریف قالب (format) جواب دریافتی

- دارای دو بخش:
  - شناسه تصویر: id
  - آدرس تصویر: img_src
- تعریف یک کلاس داده برای نمایش آن

```
data class MarsPhoto(
    val id: String, val img_src: String
)
```

- استفاده از نام معادل در جواب

```
@Json(name = "img_src") val imgSrcUrl: String
```

استفاده از مبدل Moshi

```
private val moshi = Moshi.Builder()
    .add(KotlinJsonAdapterFactory())
    .build()
private val retrofit = Retrofit.Builder()
    .addConverterFactory(MoshiConverterFactory.create(moshi))
    .baseUrl(BASE_URL)
    .build()
interface MarsApiService {
    @GET("photos")
    suspend fun getPhotos(): List<MarsPhoto>
}
```

استفاده از جواب

- نمایش تعداد عکس موجود

```
_status.value = "Success: ${listResult.size} Mars photos retrieved"
```

سوال؟

نمایش تصویر از اینترنت

مراحل نمایش یک تصویر از اینترنت

- دریافت از اینترنت
- ذخیره سازی موقت
  - در حافظه کش و یا کش بر روی فضای ذخیره سازی
- تبدیل به فرمت قابل نمایش
  - خارج کردن از حالت فشرده شده

مراحل نمایش یک تصویر از اینترنت

- دریافت از اینترنت
- ذخیره سازی موقت
  - در حافظه کش و یا کش بر روی فضای ذخیره سازی
- تبدیل به فرمت قابل نمایش
  - خارج کردن از حالت فشرده شده

- یک روش: استفاده از کتابخانه Coil
  - دریافت، ذخیره موقت، تبدیل، کش کردن

آماده سازی استفاده از Coil

- اضافه کردن به عنوان یک پیش نیاز

```
// Coil
implementation "io.coil-kt:coil:1.1.1"
```

- اضافه کردن محل دریافت کتابخانه

```
repositories {
    google()
    jcenter()
    mavenCentral()
}
```

تعریف یک متغیر برای نگَه داری آدرس اولین عکس

```
private val _photos = MutableLiveData<MarsPhoto>()
```

```
val photos: LiveData<MarsPhoto> = _photos
```

```
try {
    _photos.value = MarsApi.retrofitService.getPhotos()[0]
    _status.value = "   First Mars image URL : ${_photos.value!!.imgSrcUrl}"
}
```

تعریف مبدل اتصال (Binding Adapter)

- امکان تعریف ویژگی جدید برای اشیاء

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:imageUrl="@{product.imageUrl}"/>
```

```
@BindingAdapter("imageUrl")
fun bindImage(imgView: ImageView, imgUrl: String?) {
    imgUrl?.let {
        // Load the image in the background using Coil.
        }
    }
}
```

تعریف مبدل اتصال

- اضافه کردن یک کلاس جدید به نام BindingAdapters

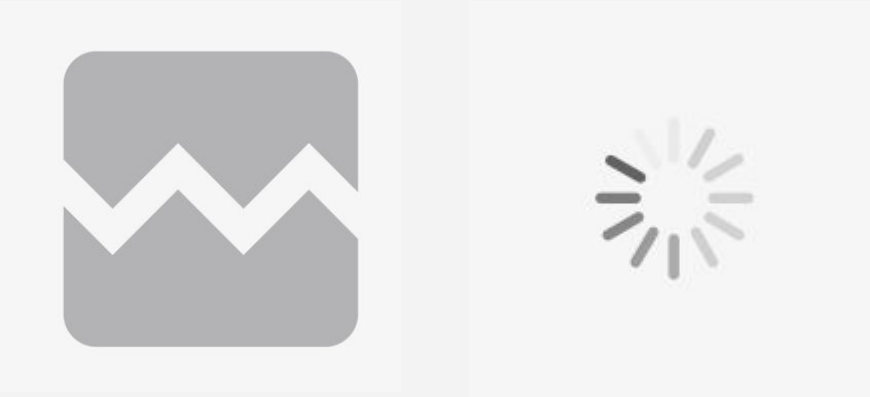
```
@BindingAdapter("imageUrl")
fun bindImage(imgView: ImageView, imgUrl: String?) {
    imgUrl?.let {
        val imgUri = imgUrl.toUri().buildUpon().scheme("https").build()
        imgView.load(imgUri)
    }
}
```

نمایش تصویر

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

```
<ImageView
    android:id="@+id/mars_image"
    ...
    app:imageUrl="@{viewModel.photos.imgSrcUrl}"
    ... />
```

نمایش تصویر در هنگام بارگذاری و در صورت بروز خطا



```
imgView.load(imgUri) {
    placeholder(R.drawable.loading_animation)
    error(R.drawable.ic_broken_image)
}
```

سوال؟

نمایش تمام تصاویر

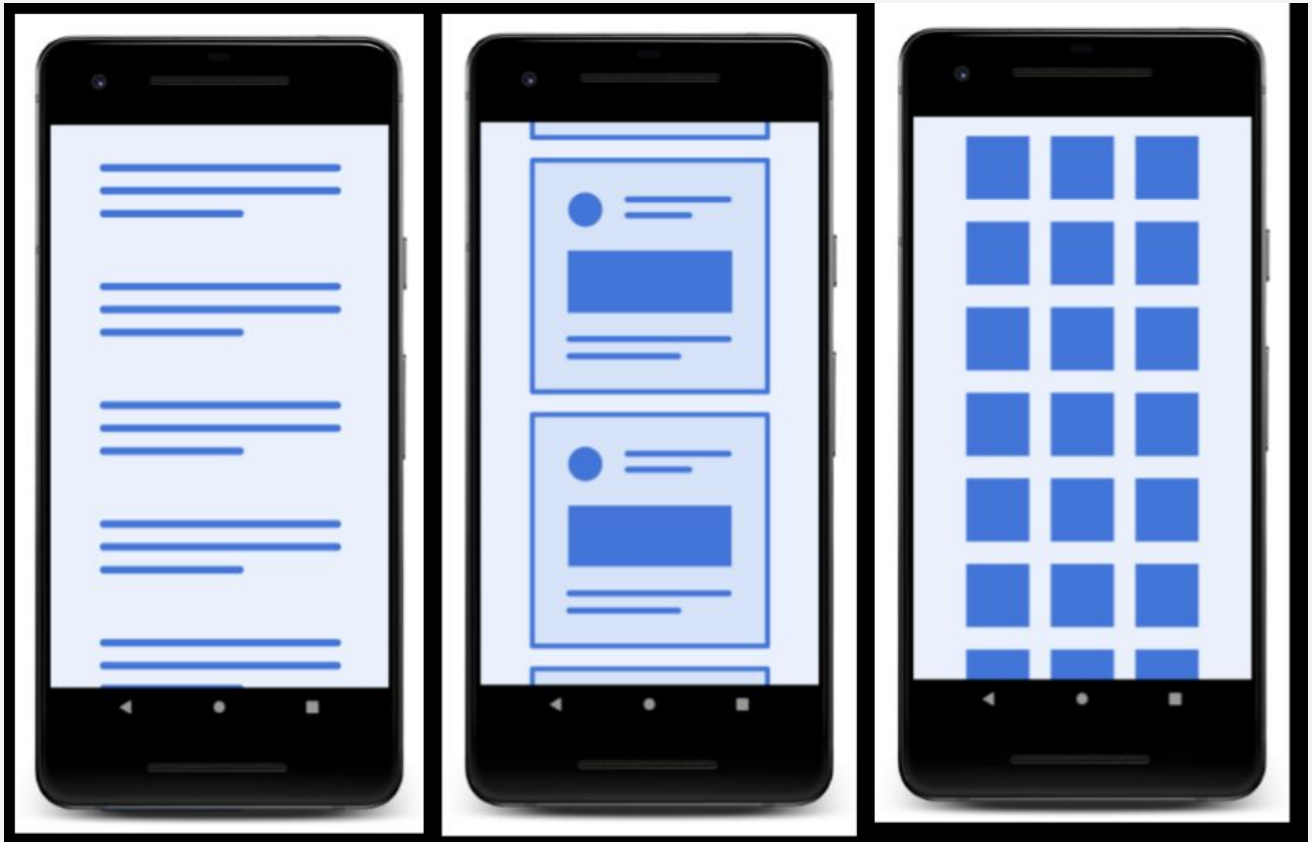
تغییر متغیر نگه داری اطلاعات تصاویر

```
private val _photos = MutableLiveData<List<MarsPhoto>>()
```

```
val photos: LiveData<List<MarsPhoto>> = _photos
```

```
try {
    _photos.value = MarsApi.retrofitService.getPhotos()
    _status.value = "Success: Mars properties retrieved"
} catch (e: Exception) {
    _status.value = "Failure: ${e.message}"
}
```

استفاده از حالت GridLayoutManager در RecyclerView



اضافه کردن RecyclerView

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/photos_grid"
    android:layout_width="0dp"
    android:layout_height="0dp"
    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:spanCount="2" />
```

به روز رسانی نمایش یک شی

- به روزرسانی فایل layout/gridview_item.xml

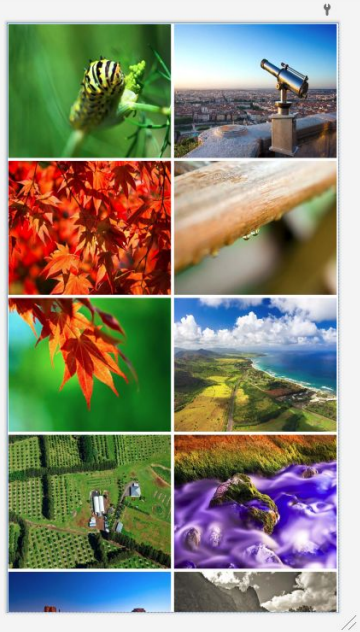
```
<data>
    <variable
        name="photo"
        type="com.example.android.marsphotos.network.MarsPhoto" />
</data>
```

- تغییر آدرس تصویر

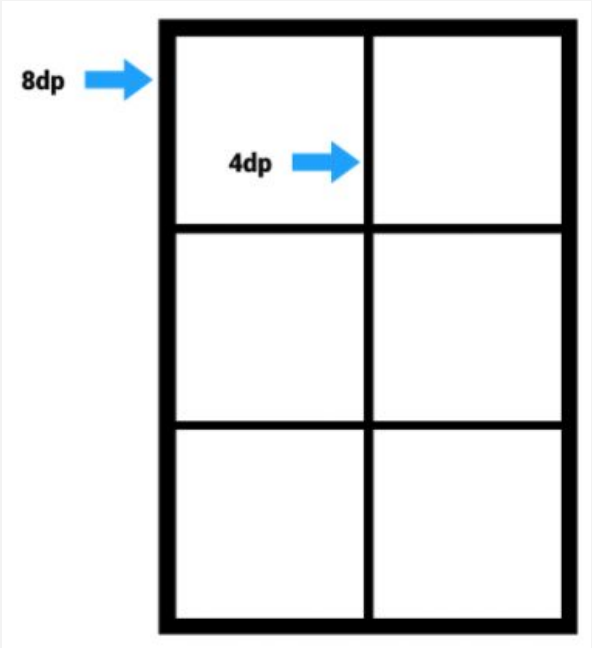
```
app:imageUrl="@{photo.imgSrcUrl}"
```

تغییر تعداد نمایش داده شده در هنگام طراحی

```
<androidx.recyclerview.widget.RecyclerView
            ...
            tools:itemCount="16"
            tools:listitem="@layout/grid_view_item" />
```



تنظیم حاشیه اطراف اشیا



- مقدار توصیه شده: 8dp در اطراف و 4dp در بین اشیا
- مقدار بردار کنار هر شی: 2dp

```
<androidx.recyclerview.widget.RecyclerView
                ...
                android:padding="6dp"
                ...   />
```

مبدل مربوط به RecyclerView

- استفاده از ListAdapter و پیاده سازی DiffUtil
- مزیت DiffUtil: تنها اشیائی که اضافه شده اند یا تغییر یافته اند به روز می شوند.

```
class PhotoGridAdapter : ListAdapter<MarsPhoto,
        PhotoGridAdapter.MarsPhotoViewHolder>(DiffCallback) {
}
companion object DiffCallback : DiffUtil.ItemCallback<MarsPhoto>() {
}
```

```
override fun areItemsTheSame(oldItem: MarsPhoto, newItem: MarsPhoto): Boolean {
    return oldItem.id == newItem.id
}
```

```
override fun areContentsTheSame(oldItem: MarsPhoto, newItem: MarsPhoto): Boolean {
    return oldItem.imgSrcUrl == newItem.imgSrcUrl
}
```

اضافه کردن مبدل اتصال

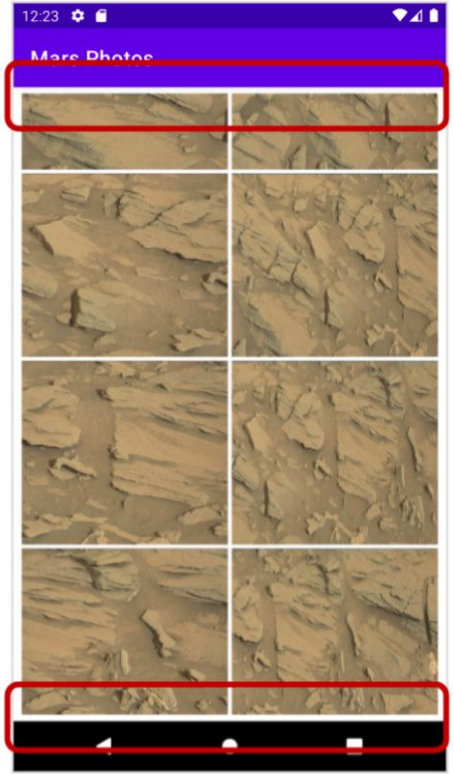
```
@BindingAdapter("listData")
fun bindRecyclerView(recyclerView: RecyclerView,
                     data: List<MarsPhoto>?) {
    val adapter = recyclerView.adapter as PhotoGridAdapter
    adapter.submitList(data)

}
```

```
app:listData="@{viewModel.photos}"
```

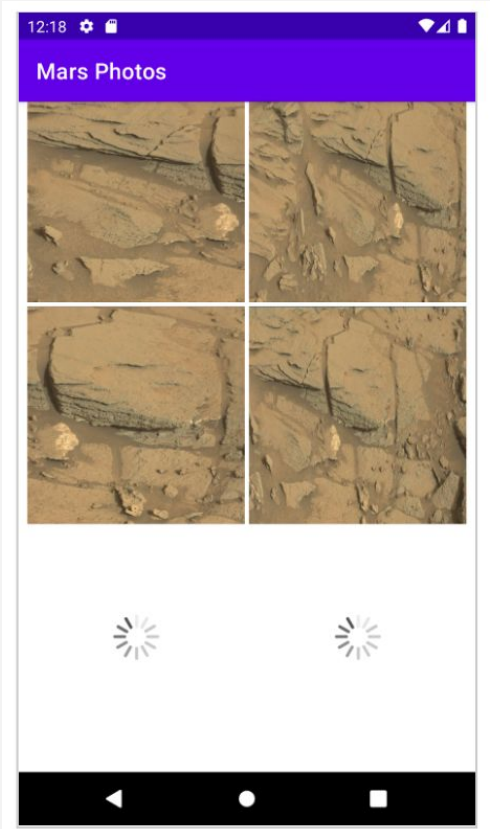
```
binding.photosGrid.adapter = PhotoGridAdapter()
```

درست کردن مشکل در بالا و پایین

```
<androidx.recyclerview.widget.RecyclerView
                ...
                android:clipToPadding="false"
                ...   />
```



تصاویر در هنگام بارگذاری و در صورت عدم بارگذاری



سوال؟

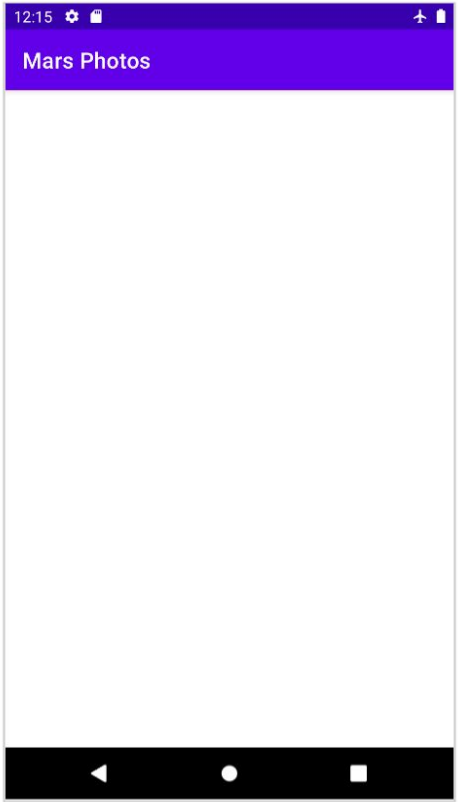
پردازش خطاها

داده با چند مقدار ممکن (enum)

```
enum class Direction {
    NORTH, SOUTH, WEST, EAST
}
```

```
var direction = Direction.NORTH;
```

مشکل در صورت عدم وجود اینترنت در ابتدا



اضافه کردن حالت دریافت اطلاعات

```
enum class MarsApiStatus { LOADING, ERROR, DONE }
private val _status = MutableLiveData<MarsApiStatus>()

val status: LiveData<MarsApiStatus> = _status
private fun getMarsPhotos() {
    viewModelScope.launch {
        _status.value = MarsApiStatus.LOADING
        try {
            _photos.value = MarsApi.retrofitService.getPhotos()
            _status.value = MarsApiStatus.DONE
        } catch (e: Exception) {
            _status.value = MarsApiStatus.ERROR
            _photos.value = listOf()
        }
    }
}
```

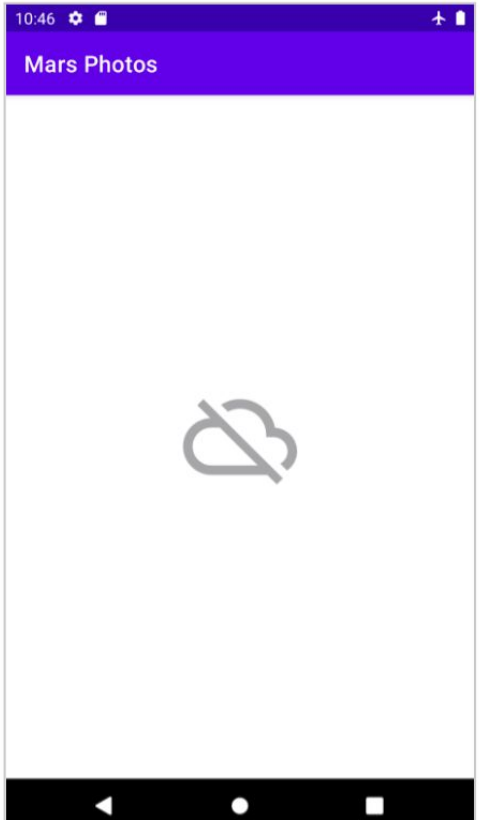
نمایش مناسب اطلاعات متناسب با وضعیت

```
@BindingAdapter("marsApiStatus")
fun bindStatus(statusImageView: ImageView,
          status: MarsApiStatus?) {
}
```

```
when (status) {
   MarsApiStatus.LOADING -> {
      statusImageView.visibility = View.VISIBLE
      statusImageView.setImageResource(R.drawable.loading_animation)
   MarsApiStatus.ERROR -> {
      statusImageView.visibility = View.VISIBLE
      statusImageView.setImageResource(R.drawable.ic_connection_error)
   }
   MarsApiStatus.DONE -> {
      statusImageView.visibility = View.GONE
   }
}
```

```
<ImageView
   android:id="@+id/status_image"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:marsApiStatus="@{viewModel.status}" />
```

نمونه اجرا بدون اینترنت



سوال؟