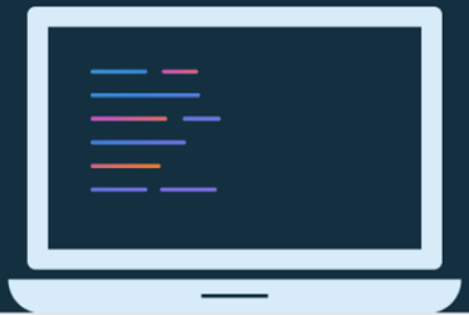
Bài học 11: Kết nối Internet



Giới thiệu về bài học này

Bài học 11: Kết nối Internet

- Các quyền của Android
- Kết nối và sử dụng các tài nguyên mạng
- Kết nối với một dịch vụ web
- Hiển thị hình ảnh
- Tóm tắt

Các quyền của Android

Quyền

- Bảo vệ quyền riêng tư của người dùng Android
- Được khai báo bằng thẻ `<uses-permission>` trong tệp `AndroidManifest.xml`

Các quyền được cấp cho ứng dụng của bạn

- Quyền có thể được cấp trong thời gian cài đặt hoặc thời gian chạy, tùy vào mức độ bảo vệ.
- Mỗi quyền đều có một mức độ bảo vệ: thông thường, chữ ký hoặc nguy hiểm.
- Đối với các quyền được cấp trong thời gian chạy, hãy nhắc người dùng cấp hoặc từ chối quyền truy cập vào ứng dụng của bạn một cách rõ ràng.

Mức độ bảo vệ của quyền

Mức độ bảo vệ	Được cấp khi nào?	Phải nhắc trước khi sử dụng?	Ví dụ
Thông thường	Thời gian cài đặt	Không	`ACCESS_WIFI_STATE`, `BLUETOOTH`, `VIBRATE`, `INTERNET`
Chữ ký	Thời gian cài đặt	Không	N/A
Nguy hiểm	Thời gian chạy	Có	`GET_ACCOUNTS`, `CAMERA`, `CALL_PHONE`

Thêm quyền vào tệp kê khai

Trong tệp `AndroidManifest.xml`:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.sampleapp">

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

    <application>
        <activity
            android:name=".MainActivity" ... >
            ...
        </activity>
    </application>
</manifest>
```

Quyền truy cập Internet

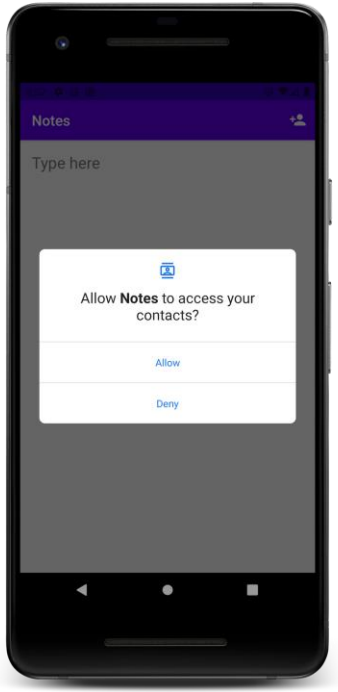
Trong tệp `AndroidManifest.xml`:

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

Yêu cầu các quyền có mức độ bảo vệ nguy hiểm

- Nhắc người dùng cấp quyền khi họ cố truy cập vào chức năng yêu cầu một quyền có mức độ bảo vệ nguy hiểm.
- Giải thích cho người dùng về lý do cần có quyền này.
- Có giải pháp dự phòng khéo léo nếu người dùng từ chối quyền này (ứng dụng vẫn phải hoạt động).

Lời nhắc đối với quyền có mức độ bảo vệ nguy hiểm



Các phương pháp hay nhất về quyền cho ứng dụng

- Chỉ sử dụng các quyền cần thiết cho hoạt động của ứng dụng.
- Chú ý đến các quyền mà thư viện yêu cầu.
- Minh bạch.
- Nêu rõ ràng các quyền truy cập hệ thống.

Kết nối và sử dụng các tài nguyên mạng

Retrofit

- Thư viện nối mạng biến API HTTP của bạn thành giao diện Kotlin và Java
- Cho phép xử lý các yêu cầu và phản hồi thành các đối tượng để ứng dụng của bạn sử dụng
  - Cung cấp tính năng hỗ trợ cơ sở để phân tích cú pháp các loại phản hồi phổ biến, chẳng hạn như XML và JSON
  - Có thể được mở rộng để hỗ trợ các loại phản hồi khác

Tại sao nên sử dụng Retrofit?

- Xây dựng dựa trên các thư viện theo tiêu chuẩn ngành, như OkHttp, với những ưu điểm sau:
  - Hỗ trợ HTTP/2
  - Gộp kết nối
  - Hoạt động lưu phản hồi vào bộ nhớ đệm và tính năng bảo mật nâng cao
- Giảm bớt bước thiết lập ban đầu cho nhà phát triển khi chạy một yêu cầu

Thêm phần phụ thuộc vào Gradle

```
implementation "com.squareup.retrofit2:retrofit:2.9.0"
implementation "com.squareup.retrofit2:converter-moshi:2.9.0"

implementation "com.squareup.moshi:moshi:$moshi_version"
implementation "com.squareup.moshi:moshi-kotlin:$moshi_version"
kapt "com.squareup.moshi:moshi-kotlin-codegen:$moshi_version"
```

Kết nối với một dịch vụ web

Các phương thức HTTP

- GET
- POST
- PUT
- DELETE

Ví dụ về API dịch vụ web

URL	NỘI DUNG MÔ TẢ	PHƯƠNG THỨC
`example.com/posts`	Lấy danh sách tất cả các bài đăng	
`example.com/posts/username`	Lấy danh sách các bài đăng của người dùng	
`example.com/posts/search?filter=queryterm`	Tìm kiếm các bài đăng bằng bộ lọc	
`example.com/posts/new`	Tạo bài đăng mới	

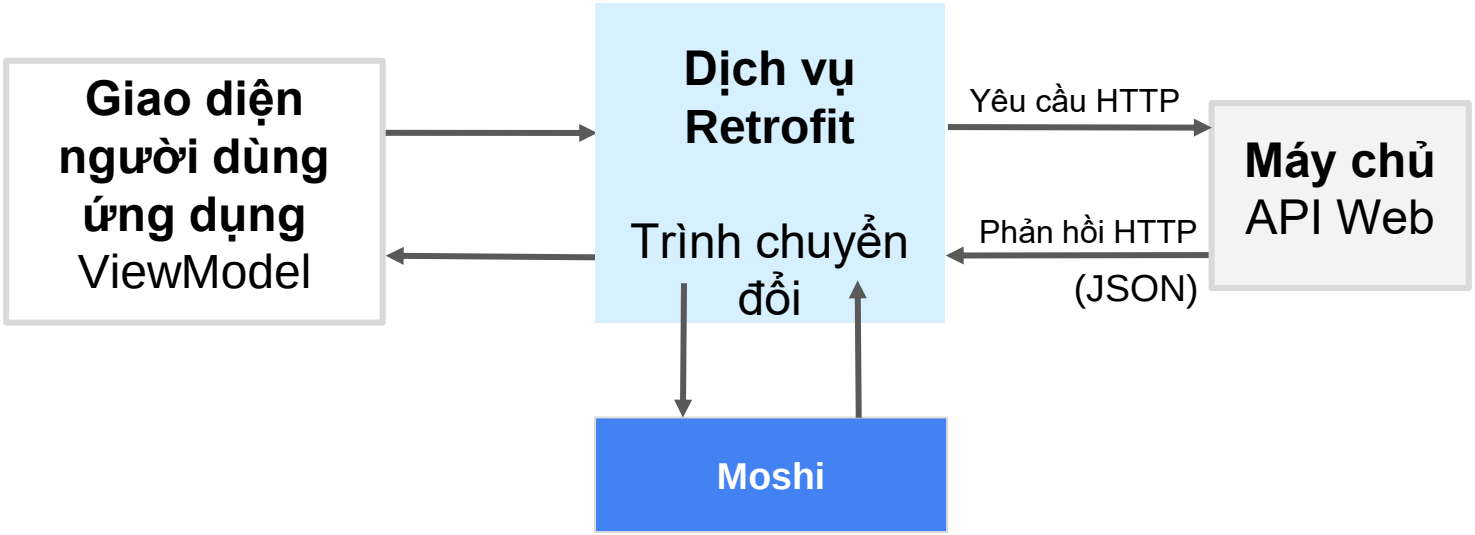
Xác định dịch vụ Retrofit

```
interface SimpleService {

    @GET("posts")
    suspend fun listPosts(): List<Post>

    @GET("posts/{userId}")
    suspend fun listByUser(@Path("userId") userId:String): List<Post>

    @GET("posts/search")  // becomes post/search?filter=query
    suspend fun search(@Query("filter") search: String): List<Post>

    @POST("posts/new")
    suspend fun create(@Body post : Post): Post
}
```

Tạo đối tượng Retrofit để truy cập mạng

```
val retrofit = Retrofit.Builder()
    .baseUrl("https://example.com")
    .addConverterFactory(...)
    .build()

val service = retrofit.create(SimpleService::class.java)
```

Sơ đồ hai đầu



Converter.Factory

Giúp chuyển đổi từ một loại phản hồi sang các đối tượng lớp

- JSON (Gson hoặc Moshi)
- XML (Jackson, SimpleXML, JAXB)
- Vùng đệm giao thức
- Vô hướng (gốc, đóng hộp và Chuỗi)

Moshi

- Thư viện JSON để phân tích cú pháp JSON thành các đối tượng và ngược lại
- Thêm các phần phụ thuộc của thư viện Moshi vào tệp Gradle của ứng dụng.
- Định cấu hình trình tạo Moshi của bạn để sử dụng với Retrofit.

Mã hóa JSON của Moshi

```
@JsonClass(generateAdapter = true)
data class Post (
    val title: String,
    val description: String,
    val url: String,
    val updated: String,
    val thumbnail: String,
    val closedCaptions: String?)
```

Mã JSON

```
{
    "title":"Android Jetpack: EmojiCompat",
    "description":"Android Jetpack: EmojiCompat",
    "url":"https://www.youtube.com/watch?v=sYGKUtM2ga8",
    "updated":"2018-06-07T17:09:43+00:00",
    "thumbnail":"https://i4.ytimg.com/vi/sYGKUtM2ga8/hqdefault.jpg"
}
```

Thiết lập Retrofit và Moshi

```
private val moshi = Moshi.Builder()
    .add(KotlinJsonAdapterFactory())
    .build()

val retrofit = Retrofit.Builder()
    .addConverterFactory(MoshiConverterFactory.create(moshi))
    .baseUrl(BASE_URL)
        .build()

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

Dùng Retrofit với coroutine

Chạy một coroutine mới trong mô hình chế độ xem:

```
viewModelScope.launch {
    Log.d("posts", API.retrofitService.searchPosts("query"))
}
```

Hiển thị hình ảnh

Glide

- Thư viện tải hình ảnh của bên thứ ba trong Android
- Tập trung vào hiệu suất để cuộn mượt mà hơn
- Hỗ trợ hình ảnh, video tĩnh và ảnh GIF động

Thêm phần phụ thuộc vào Gradle

```
implementation "com.github.bumptech.glide:glide:$glide_version"
```

Tải hình ảnh

```
Glide.with(fragment)
     .load(url)
     .into(imageView);
```

Tùy chỉnh một yêu cầu bằng RequestOptions

- Áp dụng một ảnh cắt cho hình ảnh
- Áp dụng chuyển đổi
- Đặt các tùy chọn cho hình ảnh phần giữ chỗ hoặc hình ảnh lỗi
- Đặt chính sách lưu vào bộ nhớ đệm

Ví dụ về RequestOptions

```
@BindingAdapter("imageUrl")
fun bindImage(imgView: ImageView, imgUrl: String?) {

    imgUrl?.let {
        val imgUri = imgUrl.toUri().buildUpon().scheme("https").build()

        Glide.with(imgView)
            .load(imgUri)
            .apply(RequestOptions()
                .placeholder(R.drawable.loading_animation)
                .error(R.drawable.ic_broken_image))
            .into(imgView)
    }
}
```

Tóm tắt

Tóm tắt

Trong Bài học 11, bạn đã tìm hiểu cách:

- Khai báo các quyền mà ứng dụng của bạn cần dùng trong tệp `AndroidManifest.xml`
- Dùng 3 mức độ bảo vệ cho quyền: thông thường, chữ ký và nguy hiểm (nhắc người dùng vào thời gian chạy đối với các quyền có mức độ bảo vệ nguy hiểm)
- Dùng thư viện Retrofit để thực hiện các lệnh gọi API dịch vụ web từ ứng dụng của bạn
- Dùng thư viện Moshi để phân tích cú pháp phản hồi JSON thành các đối tượng lớp
- Tải và hiển thị hình ảnh từ Internet bằng thư viện `Glide`

Tìm hiểu thêm

- Các phương pháp hay nhất về quyền cho ứng dụng
- Retrofit
- Moshi
- Glide

Lộ trình

Thực hành những gì bạn đã học được bằng cách hoàn thành lộ trình này:

Bài học 11: Kết nối Internet