Building Your First Android App

Nate Ebel
ANDROID DEVELOPER
@n8ebel www.goobar.io

Overview

Build a simple Android app using Kotlin

- How to load remote data?
- How to display and interact with list data?
- How to navigate to a new screen?

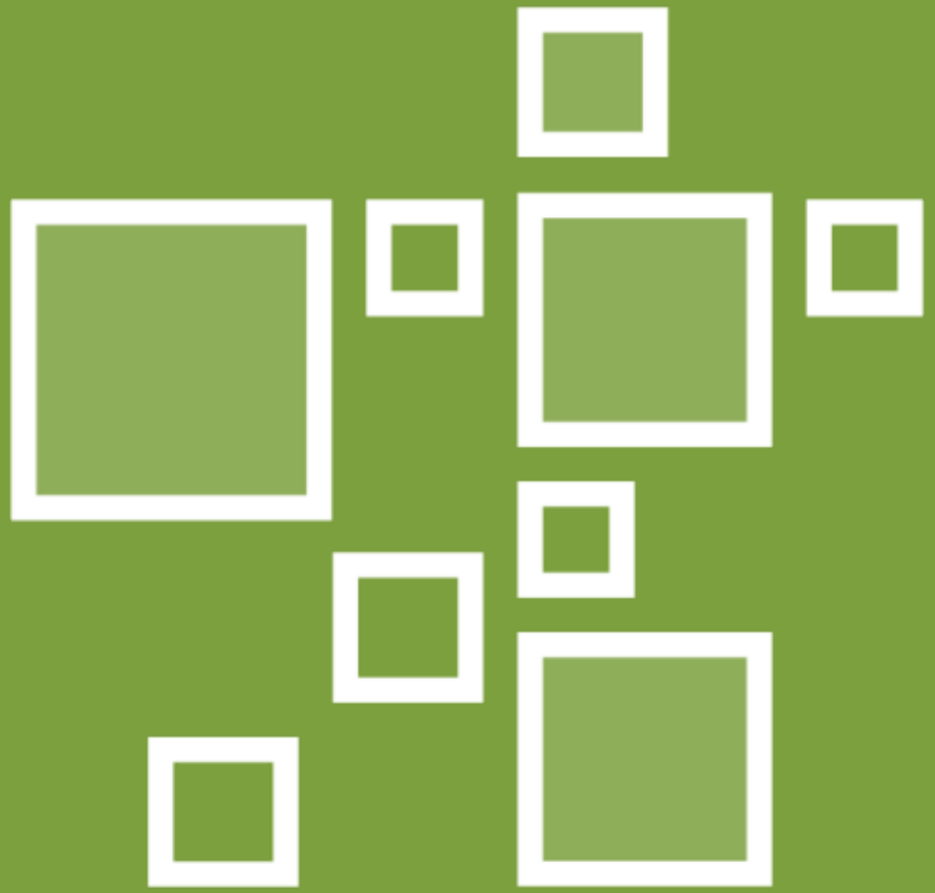
Building Your First Android App

Simple GitHub browser

- Load popular Android-related repositories
- Display repo data in a list
- Select a list item to view repo details

Understanding the Android Layout System

Understanding the Android Layout System



Views

A View is an individual element drawn to the screen. A View manages its measure and draw phases and responds to events.

Common Views

TextView

Button

EditText

ImageView

ViewGroups

A ViewGroup is a View that can contain other Views. The ViewGroup controls how child views are laid out and drawn within the bounds of the ViewGroup.

Common ViewGroups

LinearLayout

ConstraintLayout

RecyclerView

RecyclerView

A RecyclerView is a ViewGroup designed to manage the display of large collections of data. A RecyclerView responds to scroll gestures and is ideal for displaying lists and grids.

RecyclerView

RecyclerView

- Contains individual item views

LayoutManager

- Controls how views are arranged

Adapter

- Binds data into item views

RecyclerView Adapter

RecyclerView.Adapter

- Base adapter class for RecyclerView

ListAdapter

- An Adapter for efficient list display

RecyclerView.ViewHolder

- Binds data into views

Implementing a RecyclerView

Adapter

ViewHolder

List Item Layout XML

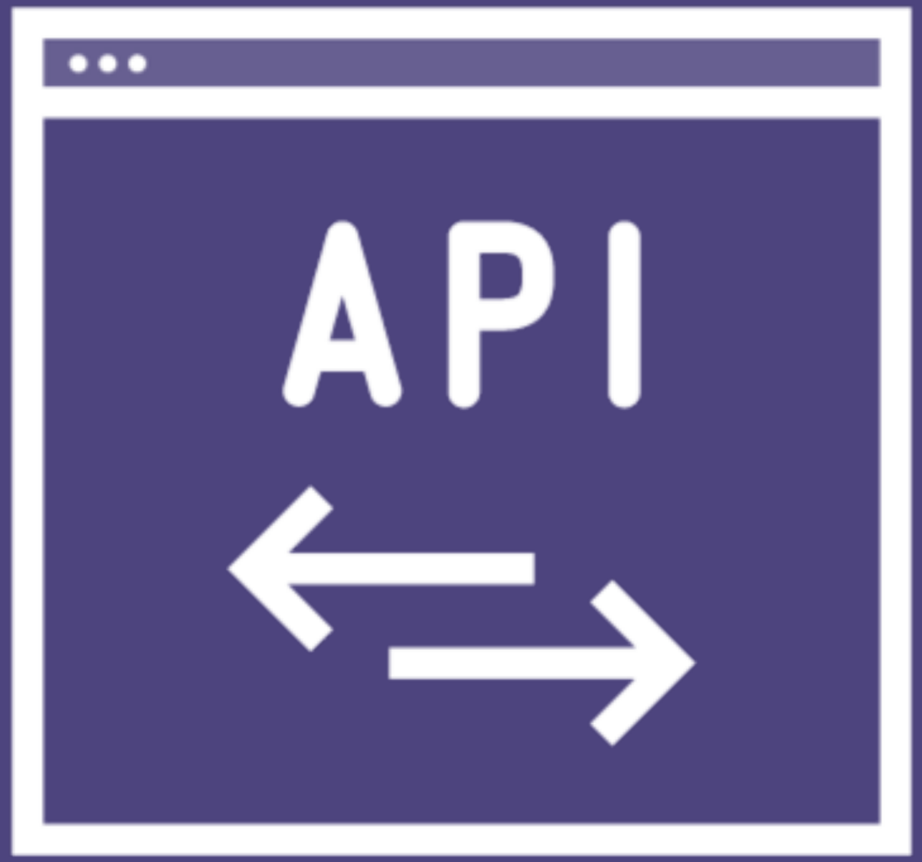
DiffUtil.ItemCallback

Demo

Displaying List Data

- Add a RecyclerView to MainActivity
- Set a LayoutManager
- Create an Adapter for the RecyclerView
- Pass list data to the Adapter

Loading Remote Data with Retrofit



Retrofit

Retrofit is a popular HTTP client for Android and Java projects. It uses annotation to convert simple interfaces into functional HTTP clients for working with remote data.

Loading Data From GitHub Using Retrofit

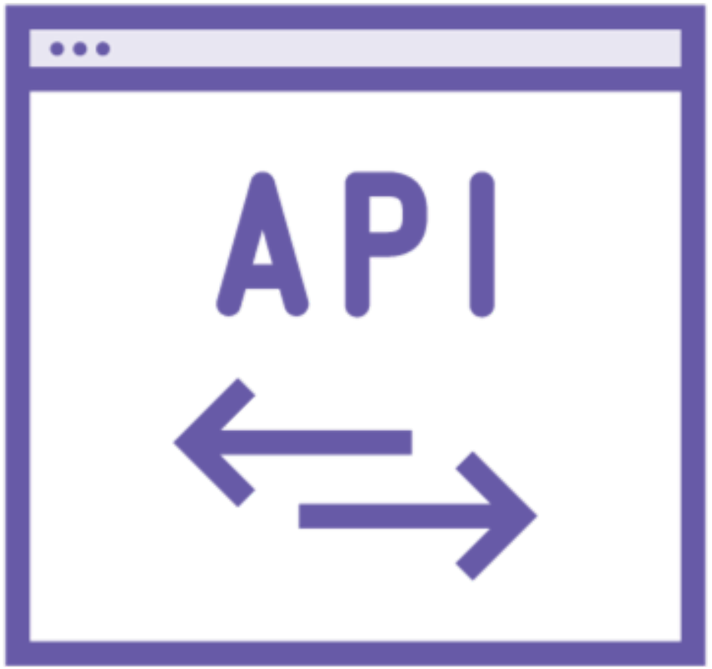
https://api.github.com

Loading Data From GitHub Using Retrofit

https://api.github.com/search/repositories

Loading Data From GitHub Using Retrofit

https://api.github.com/search/repositories?q=android&sort=stars



Define api requests with a Kotlin interface

Retrofit generates an interface implementation

Use implementation to interact with api

```
interface GitHubApiService {
  @Get(“/search/repositories”)
  fun searchRepositories(@Query(“q”) query: String) : Call<SearchResult>
}
```

Define API Interface

Define which endpoints you will interact with and how requests are made

```
val retrofit = Retrofit.Builder()
    .baseUrl(“https://api.github.com")
    .build()

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

Generate Service Interface Implementation

Use Retrofit to generate an interface implementation to interact with the GitHub api

```
val retrofit = Retrofit.Builder()
    .baseUrl(“https://api.github.com”)
    .addConverterFactory(MoshiConverterFactory.create())
    .build()

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

Deserialize Response Using Moshi

Add a converter factory to Retrofit.Builder to deserialize http response into custom data types

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

Make API Requests With Generated Service

Once the interface implementation has been generated, it can be used to interact with the http endpoints

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

service.searchRepositories("android").enqueue(object:Callback<SearchResult> {
  override fun onFailure(call: Call<SearchResult>, error: Throwable) {
    // handle error
  }

  override fun onResponse(call: Call<SearchResult>, response: Response<SearchResult>) {
    val result = response.body()
  }
}
```

Make API Requests With Generated Service

Once the interface implementation has been generated, it can be used to interact with the http endpoints

Demo

Loading Data from the GitHub API

- Model the network response
- Define a Retrofit interface
- Create a Retrofit service interface
- Load and display the network data

Demo

Responding to List Item Selection

- Pass a click listener to the Adapter
- Connect click listener to list item selection
- Display selection feedback to the user

Working With Intents

Intent

An Intent represents some action to be performed by apps and the Android operating system. Intents act as messages between different Android components such as Activities.

Working With Intents

Start an Activity

Send a message to a Service

Send an email

Open a deep link

Intent Types

Implicit Intents

Explicit Intents

Implicit Intents

Send an email

Send a text message

Select a file

Explicit Intents

Open RepoDetailsActivity

Open power settings

Send playback message to AudioPlaybackService

```
val detailsActivityIntent = Intent(context, RepoDetailActivity::class.java)
```

Launch an Activity Using an Explicit Intent

Start a new Activity directly by creating an explicit Intent for that Activity

```
val detailsActivityIntent = Intent(context, RepoDetailActivity::class.java)
context.startActivity(detailsActivityIntent)
```

Launch an Activity Using an Explicit Intent

Start a new Activity directly by creating an explicit Intent for that Activity

Demo

Displaying Selected Item Details

- Create a new Activity for item details
- Start Activity using an explicit Intent
- Parse Intent extras to get selected details
- Display the selected item details

Summary

Building Your First Android App

- Displayed, and interacted, with list data
- Loaded and displayed remote data
- Created and navigated to a second screen in the app

Building Your First Android App

Consider your own app

- What types of UI elements will your app need?
- From where will you load data?
- How man screens will your app require?

Layouts / ViewGroups / Views

Network / Database

User Experience