Testing Databases, Storage, & Servers

Overview

- Testing Databases
- Remote Databases
- Servers

Testing Room Database

Gradle Dependencies

```
android {
    defaultConfig {
        //...
        testInstrumentationRunner "androidx.test.runner
         .AndroidJUnitRunner"
        testInstrumentationRunnerArguments clearPackageData: 'true'
    }
}

dependencies {
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
}
```

Testing Android Code

- `@RunWith(AndroidJUnit4::class)`
- `@Before`
- `@After`
- `@Test`

Creating a Test Class

```
@RunWith(AndroidJUnit4::class)
class DatabaseTest {

    private lateinit val colorDao: ColorDao
    private lateinit val db: ColorDatabase

    private val red = Color(hex = "#FF0000", name = "red")
    private val green = Color(hex = "#00FF00", name = "green")
    private val blue = Color(hex = "#0000FF", name = "blue")

    ...
```

Create and Close Database for Each Test

In `DatabaseTest.kt`:

```
@Before
fun createDb() {
    val context: Context = ApplicationProvider.getApplicationContext()
    db = Room.inMemoryDatabaseBuilder(context, ColorDatabase::class.java)
        .allowMainThreadQueries()
        .build()
    colorDao = db.colorDao()
}

@After
@Throws(IOException::class)
fun closeDb() = db.close()
```

Testing Insert and Retrieve - Database

In `DatabaseTest.kt`:

```
    @Test
    @Throws(Exception::class)
    fun insertAndRetrieve() {
        colorDao.insert(red, green, blue)
        val colors = colorDao.getAll()
        assert(colors.size == 3)
    }
```

Remote Databases

Firebase Introduction

- Store and sync data with the Firebase cloud database
- Data is synced across all clients, and remains available when your app goes offline
- Connected apps share data
- Hosted in the cloud
- Data is stored as JSON
- Data is synchronized in realtime to every connected client

How to Structure Data in Firebase

```
{
    "users": {
        "alovelace": {
            "name": "Ada Lovelace",
            "contacts": { "ghopper": true },
        },
        "ghopper": { ... },
        "eclarke": { ... }
    }
}
```

Other Popular Remote Databases

MongoDB

- A source-available cross-platform document-oriented database program
- Can support both transactional and warehouse-style workloads in the same system

Cloud Firestore

- A flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud
- A full back-end as a service (BEaaS) for the least possible effort

DynamoDB

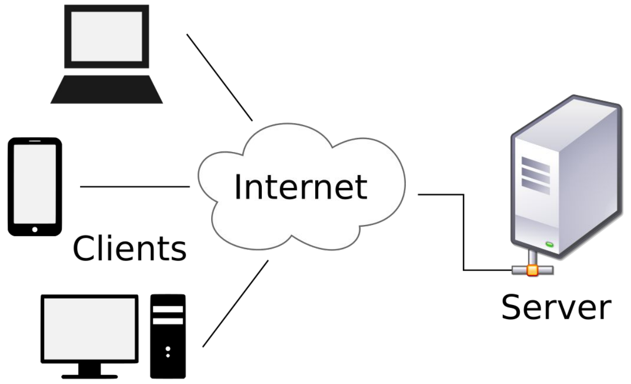
- A fully managed NoSQL database service that provides fast performance at any scale maintained by Amazon
- Supports relatively simple key-value workloads

Servers

What is a Server?

A server is a computer that serves information to other computers.

- Computers/devices (clients), can connect to servers through a network, such as the internet
- The clients establish connections through API calls

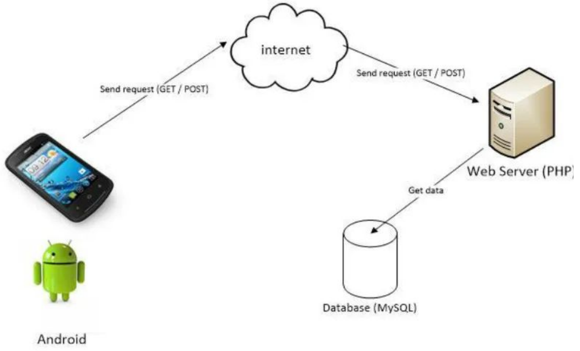


Offloading Databases onto a Server

- Devices have limited storage
- Allows for sharing data among users

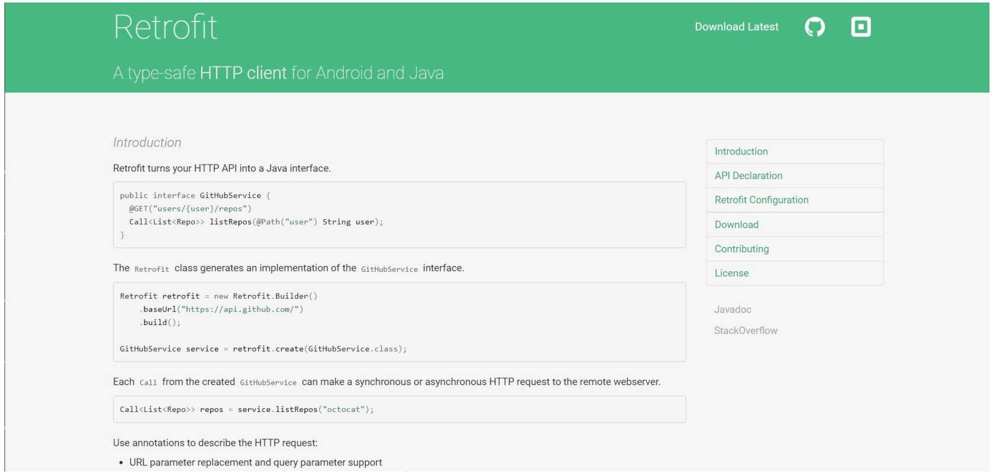
What candies do you have, can you send them to me

Here you go: { kitkat, hershey, snickers, … }



Communicating to a Server

- Make Network calls to send/receive data
- There are many libraries that help with this process
- The industry standard is to use Retrofit



Retrofit

Download Latest

A type-safe HTTP client for Android and Java

Introduction

Retrofit turns your HTTP API into a Java interface.

```
public interface GitHubService {
  @GET("users/{user}/repos")
  Call<List<Repo>> listRepos(@Path("user") String user);
}
```

The `Retrofit` class generates an implementation of the `GitHubService` interface.

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

GitHubService service = retrofit.create(GitHubService.class);
```

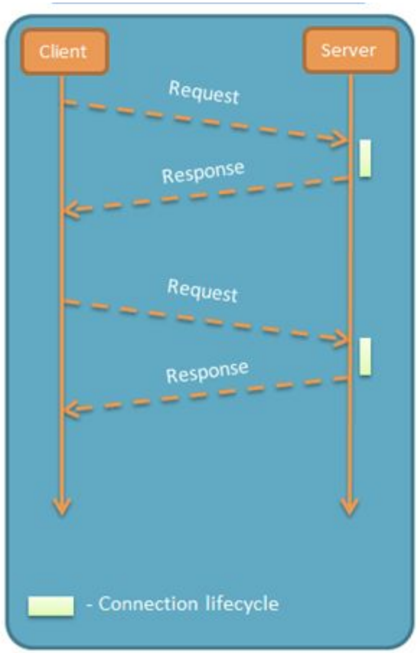
Each `Call` from the created `GitHubService` can make a synchronous or asynchronous HTTP request to the remote webserver.

```
Call<List<Repo>> repos = service.listRepos("octocat");
```

Use annotations to describe the HTTP request:

- URL parameter replacement and query parameter support

Introduction
API Declaration
Retrofit Configuration
Download
Contributing
License
Javadoc
StackOverflow



Client
Server
Request
Response
Request
Response
- Connection lifecycle