RESTful web APIs & Multithreading

Bytedance Android Developer: Yuzhong Jiang

You will learn

- ❏ Web Basics
- ❏ Data formats: JSON, XML
- ❏ Android Multithreading
- ❏ Fetch data for you Android App

Web services to access data

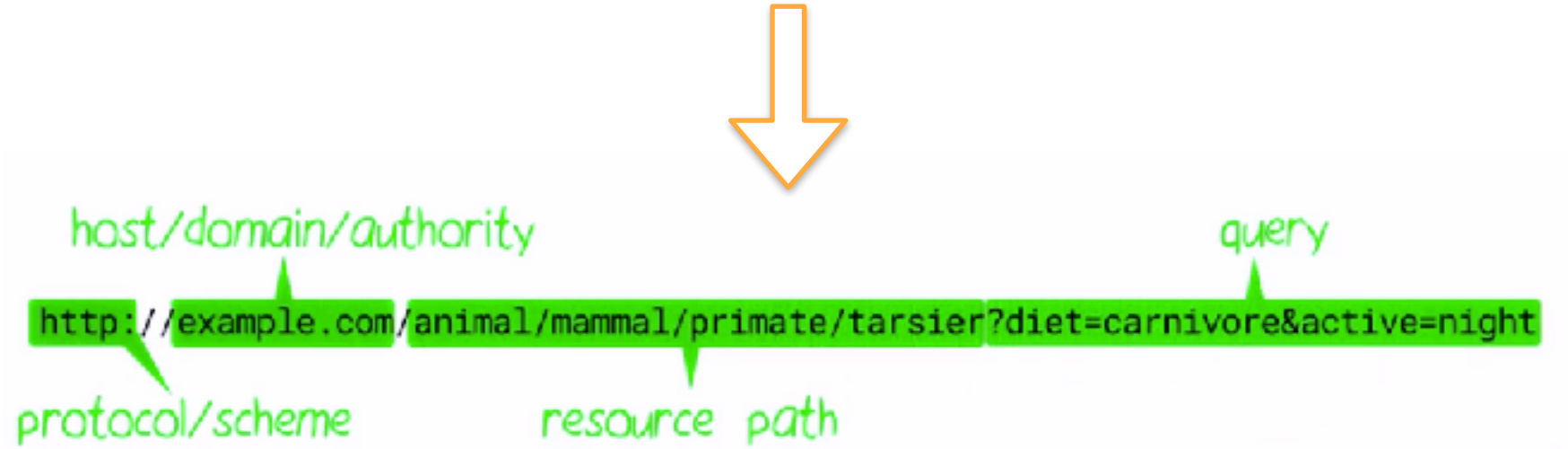
- Many apps access data through a web layer
- **Client** (app) makes queries by contacting certain specific URLs
- **Server** (web URL) sends the appropriate database data back
- Client parses the data, displays it, etc.

Web services

- Web service: a set of functionality offered over a server using the web, but not web pages / HTML
  - Use the web's **HTTP** protocol to connect and transfer data
  - Client connects to specific URLs to request specific data, which is then sent back in some documented format as XML or **JSON**
  - **REST**: Representational State Transfer. Common style of web services
    - "RESTful web services" or "RESTful APIs"

HTTP URL

http://example.com/animal/mammal/primate/tarsier?diet=carnivore&active=night



RESTful web services

- Style Guide

Uniform Resource Identifier (URI)	GET	PUT	PATCH	POST	DELETE
Collection, such as `https://api.example.com/resources/`	*List* the URIs and perhaps other details of the collection's members.	*Replace* the entire collection with another collection.	Not generally used	*Create* a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.	*Delete* the entire collection.
Element, such as `https://api.example.com/resources/item5`	*Retrieve* a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	*Replace* the addressed member of the collection, or if it does not exist, create it.	*Update* the addressed member of the collection.	Not generally used. Treat the addressed member as a collection in its own right and create a new entry within it.[17]	*Delete* the addressed member of the collection.

- REST - wikipedia
- Restapitutorial
- HTTP methods for RESTful Services

RESTful web services

- Examples
  - Github API
  - The Guardian
  - Chuck Norris
  - The Cat Api
  - URLs in your browser

JSON format

- **J**ava**S**cript **O**bject **N**otation
- The most common response format
- Easy for humans to read and write
- Easy for machines to parse and generate
- Language-independent
- vs XML, vs YAML （Tutorial）

JSON Data Types

- In JSON, values must be one of the following data types:
  - a string
  - a number
  - an object (JSON object)
  - an array
  - a boolean
  - null

JSON Data Types

- JSON strings `{ "name":"John" }`
- JSON Numbers, integer or float `{ "age":30 }`
- JSON Objects `{"employee":{ "name":"John", "age":30, "city":"New York" }}`
- JSON Arrays `{"employees":[ "John", "Anna", "Peter" ]}`
- JSON Booleans `{ "sale":true }`
- JSON null `{ "middlename":null }`

JSON Data Types

- Example data, try to create JSON:

```
temp
  min = 11.34
  max = 19.01
weather
  id = 801
  condition = Clouds
  description = null
success = true
notification_user = Adam, Bob, John
```

JSON Data Types

```
{
        "temp": {
                "min": 11.34,
                "max": 19.01
        },
        "weather": {
                "id": 801,
                "condition": "Clouds",
                "description": null
        },
        "success": true,
        "notification_user_id": [
                "Adam",
                "Bob",
                "John"
        ]
}
```

Break

- ❏ Download postman and register
- ❏ Fetch code (git repo)
- ❏ Find third_party_json_api under misc
- ❏ Q & A

Request Third Party APIs practice

- Simple GET requests: in third_party_json_api file
- Json prettify
- Postman
- Request picture in last response

RESTful APIs on the Android platform

- JSON lib
  - org.json.JSONObject
  - Gson
- HTTP lib
  - HttpUrlConnection
  - Retrofit

org.json.JSONObject

- Doc

```
{
        "os": [
                {
                        "name": "Pie",
                        "code": 28
                },
                {
                        "name": "Oreo",
                        "code": 27
                }
        ]
}
```

```
public static String RAW =
"{\"os\":[{\"name\":\"Pie\",\"code\":28}," +
        “{\"name\":\"Oreo\",\"code\":27}]}";

JSONObject root = new JSONObject(RAW);

JSONArray os = root.optJSONArray(“os”);

result = os.optJSONObject(0).
        optString("name");
```

Gson

- Github

```
{
        "os": [
                {
                        "name": "Pie",
                        "code": 28
                },
                {
                        "name": "Oreo",
                        "code": 27
                }
        ]
}
```

```
public class OSList {
    AndroidOS[] os;
}

public class AndroidOS {
    @SerializedName("name")
    String name;
    @SerializedName("code")
    int code;
}

OSList list = new Gson()
        .fromJson(RAW, OSList.class);
String result =
        list.getOs()[0].getName();
```

Before HTTP Connection

- Request Internet permissions in your AndroidManifest.xml

```
<manifest xmlns:tools="http://schemas.android.com/tools"
          package="com.bytedance.android.lesson.restapi"
          xmlns:android="http://schemas.android.com/apk/res/android">

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

    <application
      ...
    </application>

</manifest>
```

HttpURLConnection

- Doc

```
public static String getResponseWithHttpURLConnection(String url) {
    String result = null;
    InputStream in = null;
    HttpURLConnection urlConnection = null;
    try {
        URL netUrl = new URL(url);
        urlConnection = (HttpURLConnection) netUrl.openConnection();
        in = new BufferedInputStream(urlConnection.getInputStream());
        result = readStream(in);
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (urlConnection != null) {urlConnection.disconnect();}
        if (in != null) {
            try {in.close();}
            catch (IOException e) {e.printStackTrace();}
        }
    }
    return result;
}
private static String readStream(final InputStream inputStream) {
    final Scanner scanner = new Scanner(inputStream);
    scanner.useDelimiter("\\A");
    final String data = scanner.next();
    return data;
}
```

HttpURLConnection

- Doc

```
URL netUrl = new URL(url);
urlConnection = (HttpURLConnection) netUrl.openConnection();
in = new BufferedInputStream(urlConnection.getInputStream());
result = readStream(in);

private static String readStream(final InputStream inputStream) {
    final Scanner scanner = new Scanner(inputStream);
    scanner.useDelimiter("\\A");
    final String data = scanner.next();
    return data;
}
```

Retrofit

- Doc

```
// host, only once for one host
Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("http://api.icndb.com/")
        .addConverterFactory(GsonConverterFactory.create())
        .build();
// url
Response<Joke> response = retrofit.create(ICNDBService.class).randomJoke().execute();
return response == null ? null : response.body();

// definition
public interface ICNDBService {
    @GET("jokes/random") Call<Joke> randomJoke();
}
```

Break

- Checkout integration_1st (Tag)
- Run and check the result

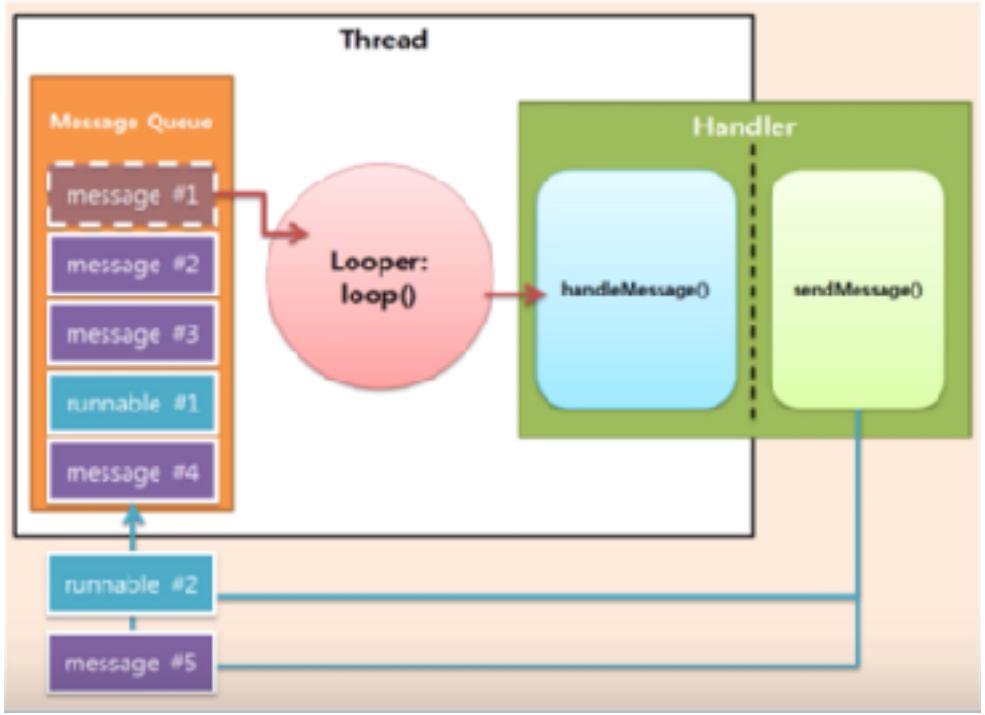
Problems

- What is **NetworkOnMainThreadException**? - google it!
- **NetworkOnMainThreadException doc**
- Do it on a **worker thread**
- **SecurityException**? Connect to network doc
- **CalledFromWrongThreadException**? try Retrofit async!
- Why does it work?

Android Thread Model

- Main Threads
  - 1. Do not block the UI thread
  - 2. Do not access the Android UI toolkit from outside the UI thread
- Worker threads
  - Activity.runOnUiThread(Runnable)
  - View.post(Runnable)
  - View.postDelayed(Runnable, long)

Looper, Handler, Message



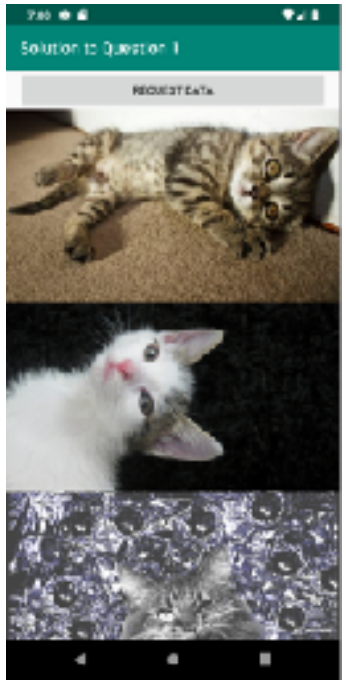
- Producer-Consumer pattern
- Looper.loop() - infinite loop
- post(Runnable) - also seed message
- Looper.getMainLooper()
- HandlerThread

Advanced Multithreading

- AsyncTask
- HandlerThread
- IntentService
- ThreadPoolExecutor
- Official tutorials

Classwork #1

- Create App **RandomCatPics**, an Android App that requests TheCatApi for 5 random cat pictures and displays them, like:



Classwork #2

- Create App **MiniDouyinApi**, an Android App that can upload pictures from your album, request the “feed” interface for picture urls and display them as App RandomCatPics does.

Classwork #2 RESTful APIs

- POST http://10.108.10.39:8080/minidouyin/video
- Parameters

Name	Type	Description
`student_id`	`string`	**Required**, e.g. 3120186666
`user_name`	`string`	**Required**, e.g. 小青
`cover_image`	`file`	**Required.**
`video`	`file`	**Required.**

Classwork #2 RESTful APIs

- POST http://10.108.10.39:8080/minidouyin/video
- Response


```
{
        "success": true,
        "item": {
                "student_id": "3120186666",
                "user_name": "小青",
                "image_url": "http://10.108.10.39:8080/minidouyin/storage/image?path=32336667/ahe/1548059515950/IMG_20180820_201006.png",
                "video_url": "http://10.108.10.39:8080/minidouyin/storage/video?path=32336667/ahe/1548059515950/b063fc96c6fd7a570180b6acccd7569d.mp4"
        }
}
```

Classwork #2 RESTful APIs

- GET http://10.108.10.39:8080/minidouyin/feed
- Response


```
{
        "feeds": [
                {
                        "student_id": "3120186666",
                        "user_name": "小青",
                        "image_url": "http://10.108.10.39:8080/minidouyin/storage/image?path=32336667/aha/
1548055081311/IMG_20180820_201006.png",
                        "video_url": "http://10.108.10.39:8080/minidouyin/storage/video?path=32336667/aha/
1548055081312/b063fc96c6fd7a570180b6acccd7569d.mp4"
                }
        ],
        "success": true
}
```

THANKS

.

Email: jiangyuzhong@bytedance.com

WeChat: yuzhong-kelv