

# RESTful web APIs & Multithreading

Bytedance Android Developer: Enlong Xia



# You will learn

- ❑ Web Basics
- ❑ Data formats
- ❑ 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 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



host/domain/authority

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

protocol/scheme

resource path

query

# RESTful web services

## ❑ Style Guide

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

- ❑ [REST - wikipedia](#)
- ❑ [Restapitutorial](#)
- ❑ [HTTP methods for RESTful Services](#)

# Data Formats

- ❑ Suppose we have an API that returns a list of employees:
  
- ❑ There are two employees, Alex and Bob. Alex's job is developer and Bob's is tester.
- ❑ Alex is familiar with Python, C/C++ and pascal; Bob is familiar with lisp, FORTRAN, and REST APIs.
  
- ❑ How do we represent such data?

# Data Formats

## ❑ YAML

```
# Employee Records
```

```
- Employee one:
```

```
  name: Alex
```

```
  job: Developer
```

```
  skills:
```

```
    - python
```

```
    - C/C++
```

```
    - pascal
```

```
- Employee two:
```

```
  name: Bob
```

```
  job: Tester
```

```
  skills:
```

```
    - lisp
```

```
    - FORTRAN
```

```
    - REST APIs
```

## ❑ XML

```
<?xml version="1.0"?>
```

```
<EmployeeRecord>
```

```
  <Employee id="emp01">
```

```
    <name>Alex</name>
```

```
    <job>Developer</job>
```

```
    <skills>python, C/C++, pascal</skills>
```

```
  </Employee>
```

```
  <Employee id="emp02">
```

```
    <name>Bob</name>
```

```
    <job>Tester</job>
```

```
    <skills>lisp, FORTRAN, REST APIs</skills>
```

```
  </Employee>
```

```
</EmployeeRecord>
```

## ❑ JSON

```
{
```

```
  "Employees": [
```

```
{
```

```
  "id": "emp01",
```

```
  "name": "Alex",
```

```
  "job": "Developer",
```

```
  "skills": "python, C/C++, pascal"
```

```
,
```

```
{
```

```
  "id": "emp02",
```

```
  "name": "Bob",
```

```
  "job": "Tester",
```

```
  "skills": "lisp, FORTRAN, REST APIs"
```

```
}
```

```
]
```

```
}
```

# JSON format

- ❑ JavaScript Object Notation
- ❑ 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)

# RESTful web services

## ❑ Examples

- ❑ [Github API](#)
- ❑ [The Guardian](#)
- ❑ [Chuck Norris](#)
- ❑ [The Cat Api](#)
- ❑ URLs in your browser

# JSON Data Types

- ❑ In JSON, values must be one of the following data types:

# JSON Data Types

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

# 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



```
{  
    "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("\n");  
    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("\n");
    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();
}
```

# Glide

## Doc

```
// For a simple view:  
@Override public void onCreate(Bundle savedInstanceState) {  
    ...  
    ImageView imageView = (ImageView) findViewById(R.id.my_image_view);  
  
    Glide.with(this).load("http://goo.gl/gEgYUd").into(imageView);  
}
```

# Break

- ❑ Checkout [Introducing\\_multithreading\\_and\\_retrofit](#)
- ❑ 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

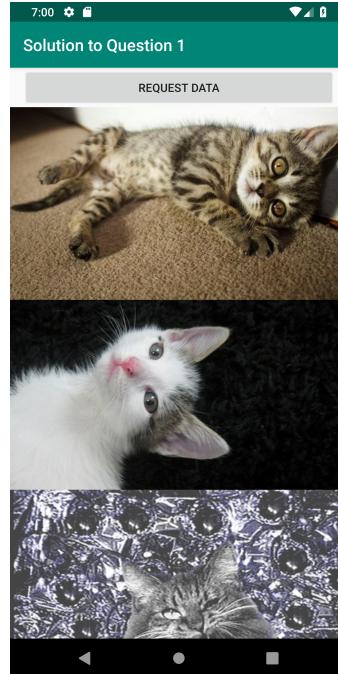
- ❑ 1. Do not block the UI thread
- ❑ 2. Do not access the Android UI toolkit from outside the UI thread

## ❑ Worker threads

- ❑ `Activity.runOnUiThread(Runnable)`
- ❑ `View.post(Runnable)`
- ❑ `View.postDelayed(Runnable, long)`

# 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://test.androidcamp.bytedance.com/mini\\_douyin/invite/video](http://test.androidcamp.bytedance.com/mini_douyin/invite/video)
- Parameters

Name	Type	Description
student_id	string	<b>Required</b> , e.g. 2220186666
user_name	string	<b>Required</b> , e.g. doudou
cover_image	file	<b>Required</b> .
video	file	<b>Required</b> .

# Classwork #2 RESTful APIs

- ❑ POST [http://test.androidcamp.bytedance.com/mini\\_douyin/invoke/video](http://test.androidcamp.bytedance.com/mini_douyin/invoke/video)
- ❑ Response

```
{  
    "success": true,  
    "item": {  
        "student_id": "2220186666",  
        "user_name": "doudou",  
        "image_url": "https://sf6-hscdn-tos.pstatp.com/obj/developer-baas/baas/  
tt7217xbo2wz3cem41/9c6bbc2aa5355504_1560563154279.jpg",  
        "_id": "5d044dd222e26f0024157401",  
        "video_url": "https://lf1-hscdn-tos.pstatp.com/obj/developer-baas/baas/tt7217xbo2wz3cem41/  
a8efa55c5c22de69_1560563154288.mp4"  
    }  
}
```

# Classwork #2 RESTful APIs

- ❑ GET [http://test.androidcamp.bytedance.com/mini\\_douyin/invoke/video](http://test.androidcamp.bytedance.com/mini_douyin/invoke/video)
- ❑ Response

```
{"feeds": [  
    {  
        "student_id": "2220186666",  
        "user_name": "doudou",  
        "image_url": "https://sf6-hscdn-tos.pstatp.com/obj/developer-baas/baas/tt7217xbo2wz3cem41/9c6bbc2aa5355504_1560563154279.jpg",  
        "_id": "5d044dd222e26f0024157401",  
        "video_url": "https://lf1-hscdn-tos.pstatp.com/obj/developer-baas/baas/tt7217xbo2wz3cem41/a8efa55c5c22de69_1560563154288.mp4",  
        "createdAt": "2019-06-15T01:45:54.368Z",  
        "updatedAt": "2019-06-15T01:45:54.368Z"  
    },  
    {"success": true  
}
```



# THANKS



Email: [xiaenlong@bytedance.com](mailto:xiaenlong@bytedance.com)

WeChat: 18514698697