ТЕХНОТРЕК

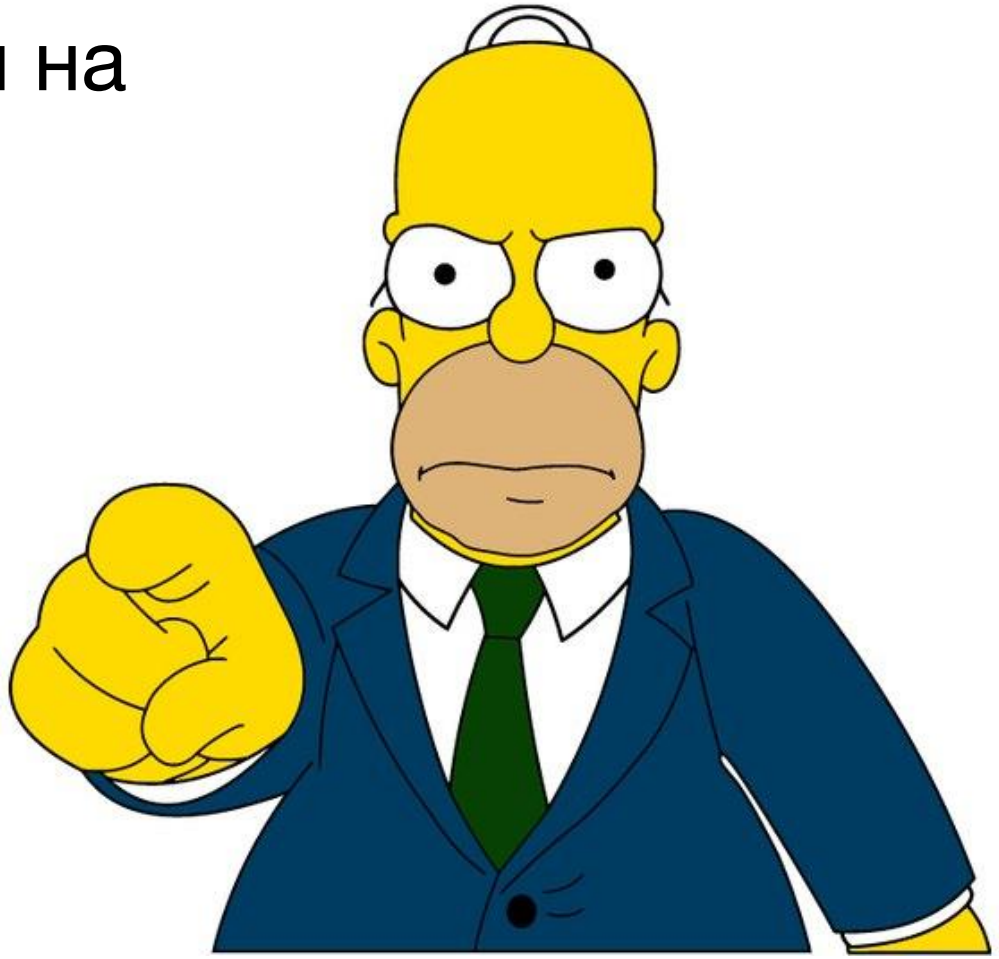
Занятие №6

Разработка приложений на Android

Кирилл Филимонов
Юрий Береза

Напоминание

А ты отметился о присутствии на занятии?



Agenda

- Сеть
- WebView
- JSON и GSON
- OkHttp
- Retrofit

О чем нужно помнить

- Вся работа с сетью должна быть вне UI потока
- Надо экономить трафик
- Скорость и качество связи может быть плохим
- Батарейка не вечная
- Персональные данные пользователя

Протокол IP4

255.255.255.255 : 9999

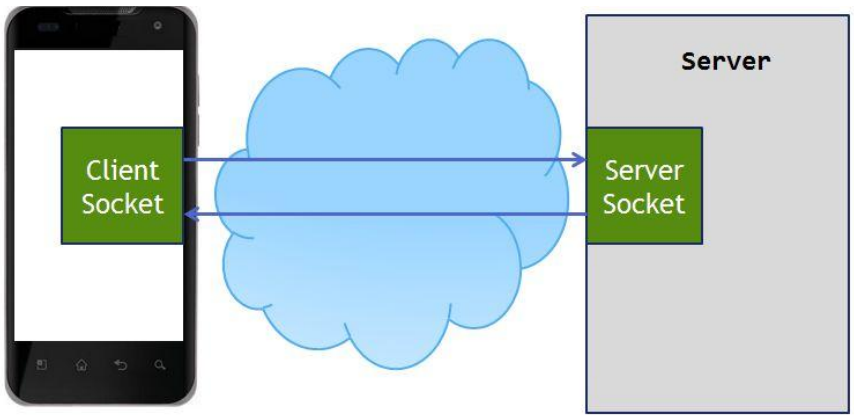
4 300 000 000

Хватит на всех!

Типы сетевых взаимодействий

На основе сокетов (Socket API)

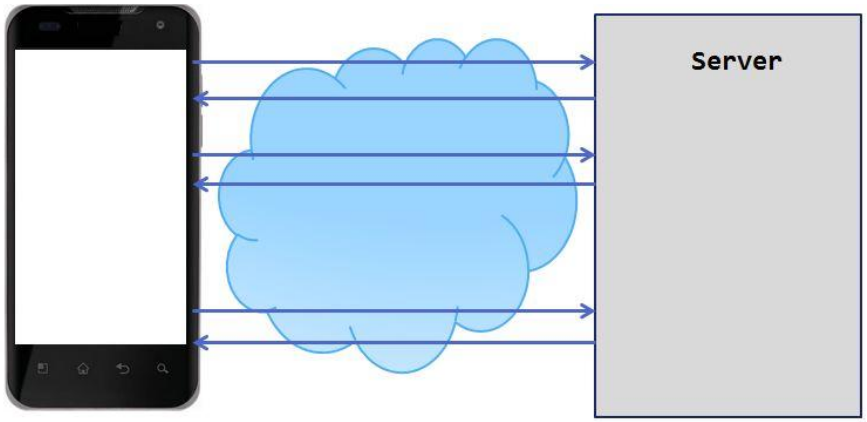
- Постоянное соединение
- Скорость
- Порядок доставки (TCP)
- Контроль доставки (TCP)



Типы сетевых взаимодействий

Частые опросы (Polling):

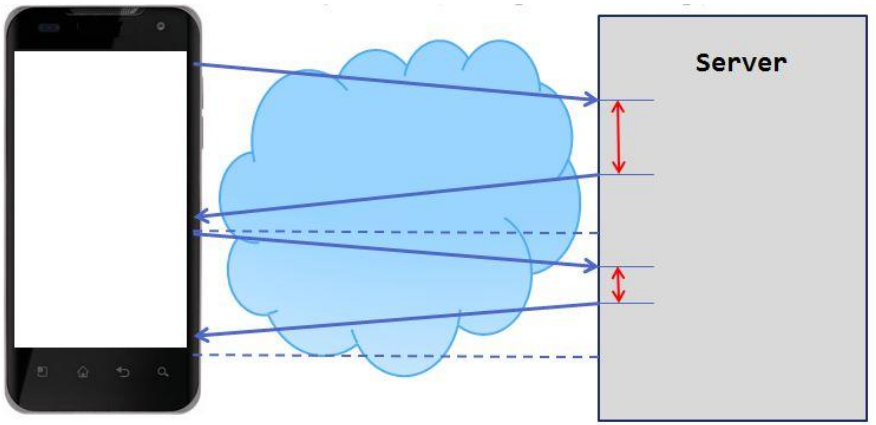
- Соединяемся, спрашиваем, получаем ответ отсоединяемся
- Частые установки соединения
- Стучимся всегда
- Много лишнего трафика
- Большая нагрузка на сервер



Типы сетевых взаимодействий

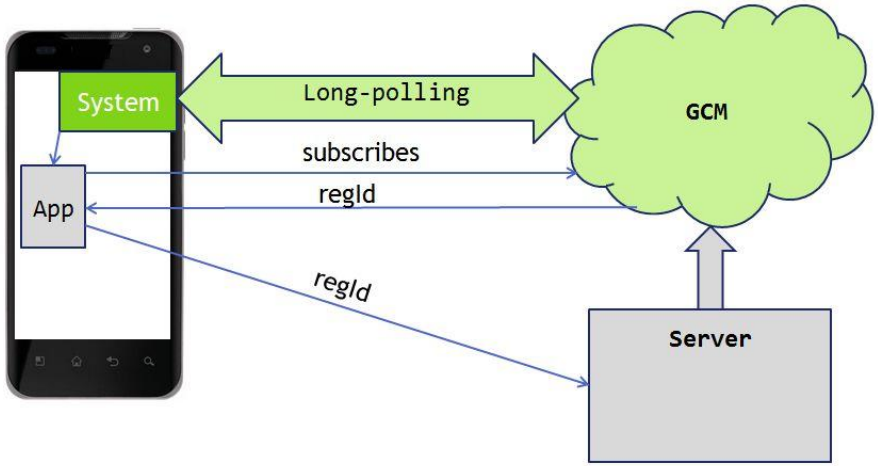
Длинные опросы (Long polling):

- Шлем запрос и ждем до тех пор пока не появятся данные
- Меньше расходуется трафика
- Не так сильно расходуется батарея
- Быстрее чем polling



Long polling руками Google (FCM)

- Надо подписаться у сервиса FCM
- Вся логика уведомлений о новых данных на стороне Google
- Нагружены сервера Google
- Есть ограничения на количество сообщений
- Невозможно этим пользоваться если вы публикуете приложение не в Play Store



Socket

```
fun performSocketConnection(address: String, port: Int): String {
    val socket = Socket(address, port)
    val os = socket.getOutputStream()
    os.write(getRequestString(address))
    os.flush()
    val output = socket.inputStream.asString()
    os.close()
    socket.close()
    return output
}

fun getRequestString(address: String): ByteArray {
    val request = "GET / HTTP/1.1\r\nHost: $address\r\nConnection: Close\r\n\r\n"
    return request.toByteArray(Charsets.UTF_8)
}

fun InputStream.asString() : String {
    return bufferedReader(Charsets.UTF_8).use { it.readText() }
}
```

Протокол HTTP

Метод GET

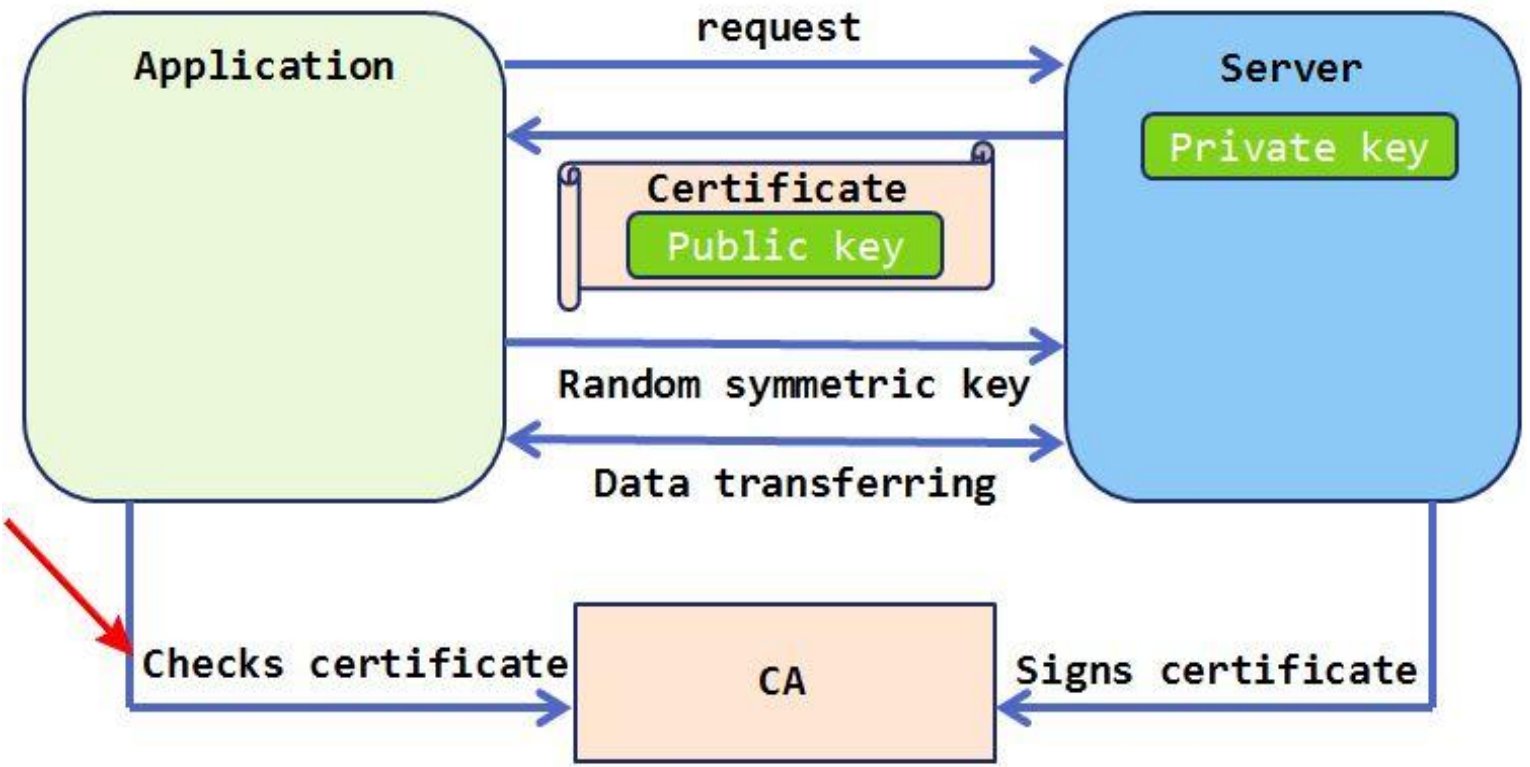
```
GET / HTTP/1.1
Host: mail.ru
User-Agent: WebKit 1.0
Accept: text/html
Connection: close


HTTP/1.1 200 OK
Date: Tu, 27 Oct 2015 08:00:00 GMT
Server: Apache
Content-Language: ru
Content-Type: text/html; charset=utf-8
Content-Length: 1234
Connection: close


<html>
<head>Mail.Ru</head>
<body>
Hello, World!
</body>
</html>
```

Протокол HTTP

Метод POST

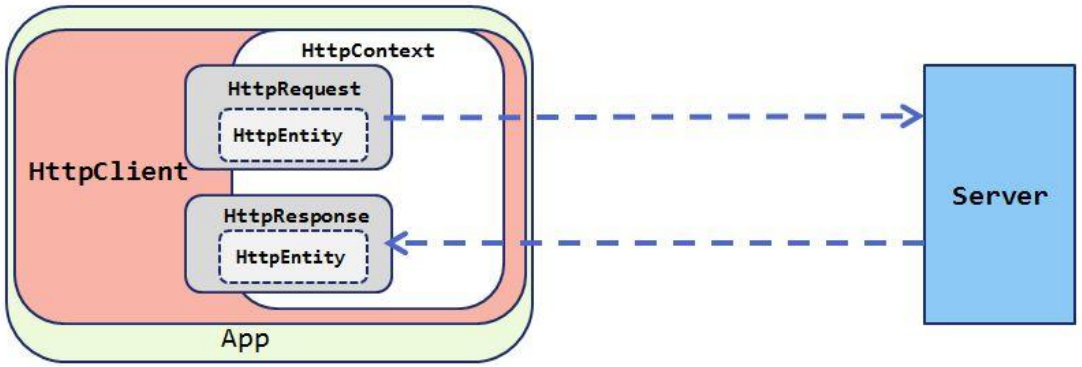
```
POST / HTTP/1.1
Host: mail.ru
User-Agent: WebKit 1.0
Accept: text/html
Connection: close


Name=Yury&Email=ybereza@gmail.com


HTTP/1.1 200 OK
Date: Tu, 27 Oct 2015 08:00:00 GMT
Server: Apache
Content-Language: ru
Content-Type: text/html; charset=utf-8
Content-Length: 1234
Connection: close
```

Защищенное соединение (HTTPS)



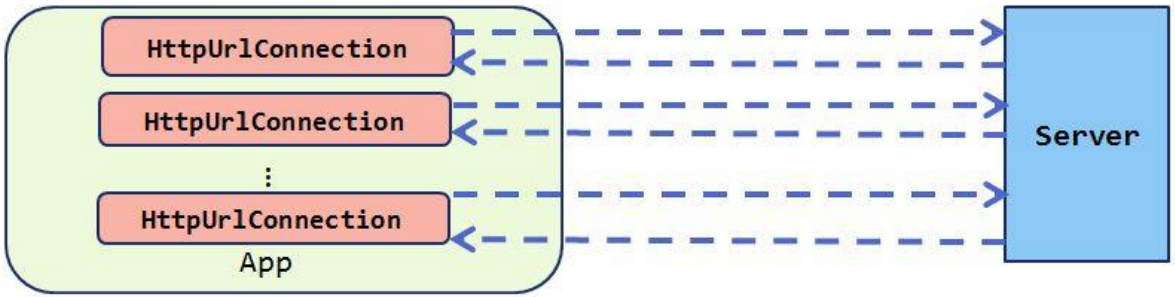
HttpClient

- Спроектирован с учетом ООП
- В самом простом случае мы будем работать с пятью разными интерфейсами: HttpRequest, HttpResponse, HttpEntity и HttpContext
- Достаточно тяжеловесный
- Как правило, на все приложение существует всего один экземпляр класса HttpClient
- С версии 22 является Deprecated



HttpUrlConnection

- Один класс и все!
- Родительский класс URLConnection был спроектирован для работы не только по HTTP-протоколу, а еще по таким, как file, mailto, ftpa
- Следует создавать на каждый запрос новый экземпляр клиента
- Легковесный
- Почти не настраиваются параметры keep alive



Как использовать

Запрос страницы

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

```
val url = URL("http://mail.ru")
val content : String = try {
    val connection = url.openConnection() as HttpURLConnection
    connection.requestMethod = "GET"
    connection.connectTimeout = 5000
    connection.readTimeout = 5000
    val responseCode = connection.responseCode
    when(responseCode) {
        HttpRequest.REQUEST_OK -> connection.inputStream.bufferedReader.readText()
        else -> ""
    }
}
finally {
    connection.disconnect()
    ""
}
```

Как использовать

Запрос страницы

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

```
val content : String = URL("http://mail.ru").readText()
```

Как использовать

Отправка данных на сервер

```
val url = URL("http://mail.ru")
val content : String = try {
    val connection = url.openConnection() as HttpURLConnection
    connection.requestMethod = "POST"
    connection.doOutput = true
    connection.addRequestProperty("Content-Type", "application/json; charset=utf-8")
    connection.connectTimeout = 5000
    connection.readTimeout = 5000
    connection.outputStream.bufferedWriter().write("""{"param" : "value"}""")

    val responseCode = connection.responseCode
    when(responseCode) {
        HttpRequest.REQUEST_OK -> connection.inputStream.bufferedReader.readText()
        else -> ""
    }
}
finally {
    connection.disconnect()
    ""
}
```

WebView

AndroidManifest.xml

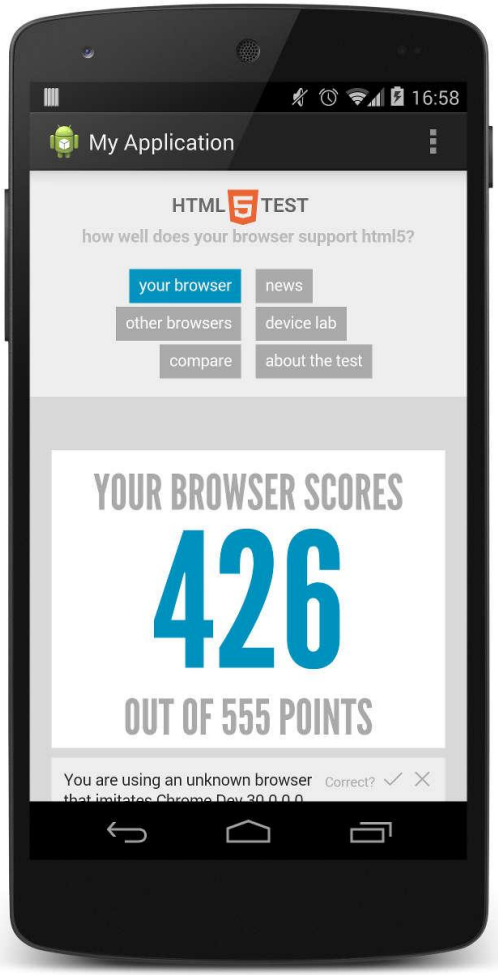
```
<uses-permission android:name="android.permission.INTERNET"/>
```

activity_layout.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <WebView
        android:id="@+id/web_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</LinearLayout>
```

Code

```
val webView = findViewById<WebView>(R.id.web_view)
webView.loadUrl("https://mail.ru")
```



JSON - JavaScript Object Notation

- Компактный
- Легкочитаемый (human readable)
- Использует простые типы данных
- Гибкий
- Очень популярный

```
{
    hey: "guy",
    anumber: 243,
  - anobject: {
        whoa: "nuts",
      - anarray: [
            1,
            2,
            "thr<h1>ee"
        ],
        more: "stuff"
    },
    awesome: true,
    bogus: false,
    meaning: null,
    japanese: "明日がある。",
    link: http://jsonview.com,
    notLink: "http://jsonview.com is great"
}
```

JSON. Пакет org.json

- JSONArray
- JSONObject
- JSONString
- JSONTokener
- JSONException

```
val jtk = JSONTokener(request.content)
try {
    val jsonObject = jtk.nextValue() as JSONObject
    val builder = StringBuilder()
    for (key in jsonObject.keys()) {
        builder.append("$key : ${jsonObject.get(key)}").append("\n")
    }
    return builder.toString()
} catch (ex: JSONException) {
    ex.printStackTrace()
    return ""
}
```

JSON. GSON

https://github.com/google/gson
http://www.jsonschema2pojo.org/

Файл build.gradle

```
dependencies {
    implementation 'com.google.code.gson:gson:2.8.2'
}
```

```
val gson = GsonBuilder().create()
val geoIP = gson.fromJson(request.content, GeoIP::class.java)
```

OkHttp

Современный HTTP клиент для Java и Android

- поддержка пулов подключений
- прозрачной сжатие GZIP
- кэширование ответов
- поддержка TLS и дополнительных возможностей
- восстановление после сбоев
- синхронный и асинхронный интерфейсы

OkHttp

OkHttpClient

```
fun createSimpleClient() = OkHttpClient()
```

Использование билдера

```
fun createClient(): OkHttpClient {
    return OkHttpClient.Builder()
            .connectTimeout(10, TimeUnit.SECONDS)
            .writeTimeout(10, TimeUnit.SECONDS)
            .readTimeout(30, TimeUnit.SECONDS)
            .addInterceptor(LoggingInterceptor(simpleLogger))
            .build()
}
```

OkHttp

Request

используется шаблон Builder

```
private fun getUsingOkHttp(url: String): Followers {
    try {
        val request = Request.Builder()
                .url(url)
                .header("User-Agent", "Technotrack")
                .build()

        val response = createClient().newCall(request).execute()
        if (response.isSuccessful) {
            return Followers.createFromJSON(response.body()!!.string())
        }
    } catch (e: IOException) {
        e.printStackTrace()
    }
    return Followers()
}
```

OkHttp

Response

- синхронный вызов

```
val response = createClient().newCall(request).execute()
```

- асинхронный вызов

```
client.newCall(request).enqueue(object : Callback {
    override fun onFailure(call: Call, e: IOException) {
        //handle failure
    }

    override fun onResponse(call: Call, response: Response) {
        //handle success
    }
})
```

OkHttp

Authentication

Встроенный обработчик ошибок с кодом 401 Unauthorized

```
fun crateClientWithBasicAuth() : OkHttpClient {
    return OkHttpClient.Builder().authenticator(object : Authenticator {
        override fun authenticate(route: Route, response: Response): Request? {
            val credentials = Credentials.basic("login", "password")
            return response.request().newBuilder().addHeader("Authorization", credentials).build()
        }

    }).build()
}
```

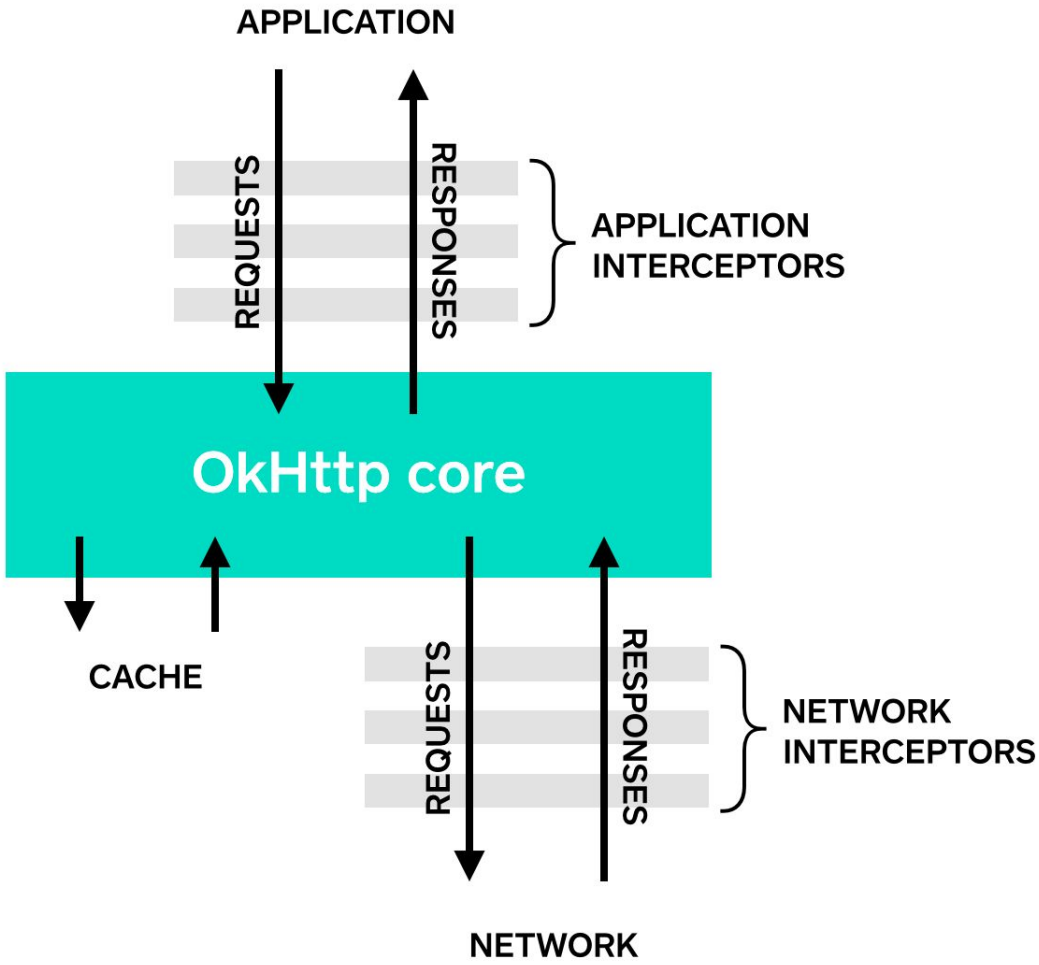
OkHttp

Interceptors

Возможность перехватывать и обрабатывать запросы и ответы в процессе выполнения

```
class LoggingInterceptor(private val logger: (String) -> Unit) : Interceptor {

    override fun intercept(chain: Interceptor.Chain): Response {
        val request = chain.request()

        val t1 = System.nanoTime()
        logger("Sending request ${request.url()} ${request.headers()}")

        val response = chain.proceed(request)

        val t2 = System.nanoTime()
        logger("Received response for ${response.request().url()} in ${(t2 - t1) / 1e6} ${response.headers()}")

        return response
    }
}
```

OkHttp


APPLICATION
REQUESTS
RESPONSES
APPLICATION
INTERCEPTORS
OkHttp core
CACHE
REQUESTS
RESPONSES
NETWORK
INTERCEPTORS
NETWORK

OkHttp

Interceptors

Добавление перехватчика

```
val client = OkHttpClient.Builder()
        .addInterceptor(LoggingInterceptor(::simpleLogger))
        .build()

val client = OkHttpClient.Builder()
        .addNetworkInterceptor(LoggingInterceptor(::simpleLogger))
        .build()
```

Retrofit

HTTP клиент для Java и Android

- позволяет HTTP API в виде Java интерфейса
- имеет настраиваемые HTTP клиенты
- использует аннотации

```
interface GeoIPRetrofitService {

    @GET("{ip}")
    fun getInfo(@Path("ip") ip: String,
                @Query("access_key") key: String): Call<GeoIP>

    companion object Factory {
        fun create(): GeoIPRetrofitService {
            val retrofit = Retrofit.Builder()
                    .addConverterFactory(GsonConverterFactory.create())
                    .baseUrl("http://api.ipapi.com/")
                    .build()

            return retrofit.create(GeoIPRetrofitService::class.java);
        }
    }
}
```

Retrofit

Описание API

- аннотации GET, POST, PUT, DELETE, HEAD
- возможность передачи параметров в URL
- настройка заголовков

```
@GET("group/{id}/users")
fun getGroupUsers(@Path("id") groupId : Int, @Query("sort") sort : String) : Call<List<User>>

@POST("users/new")
fun createUser(@Body User user) : Call<User>

@Multipart
@PUT("user/photo")
fun updateUserPhoto(@Part("photo") photo : RequestBody, @Part("description") desc : RequestBody) : Call<User>
```

Retrofit

Создание сервиса

- используется Builder

```
val retrofit = Retrofit.Builder()
    .baseUrl("https://github.com")
    .addConvertorFactory(GsonConvertorFactory.create())
    .build()

val githubService = retrofit.create(GitHubService::class.java)
```

Retrofit

Вызов метода

```
fun requestGeoIPWithRetrofit(result : (String) -> Unit) {
    val service = GeoIPRetrofitService.create()

    service.getInfo("address", "key").
            enqueue(object : Callback<GeoIP> {
                override fun onFailure(call: Call<GeoIP>, t: Throwable) {
                    result("Can not get GeoIP info")
                }

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

“Нет лучше насилия, чем насилие над самим собой.

Гомер Симпсон

ТЕХНОТРЕК

Спасибо за внимание!

Кирилл Филимонов - Kirill.Filimonov@gmail.com

Юрий Береза - ybereza@gmail.com