ТЕХНОТРЕК

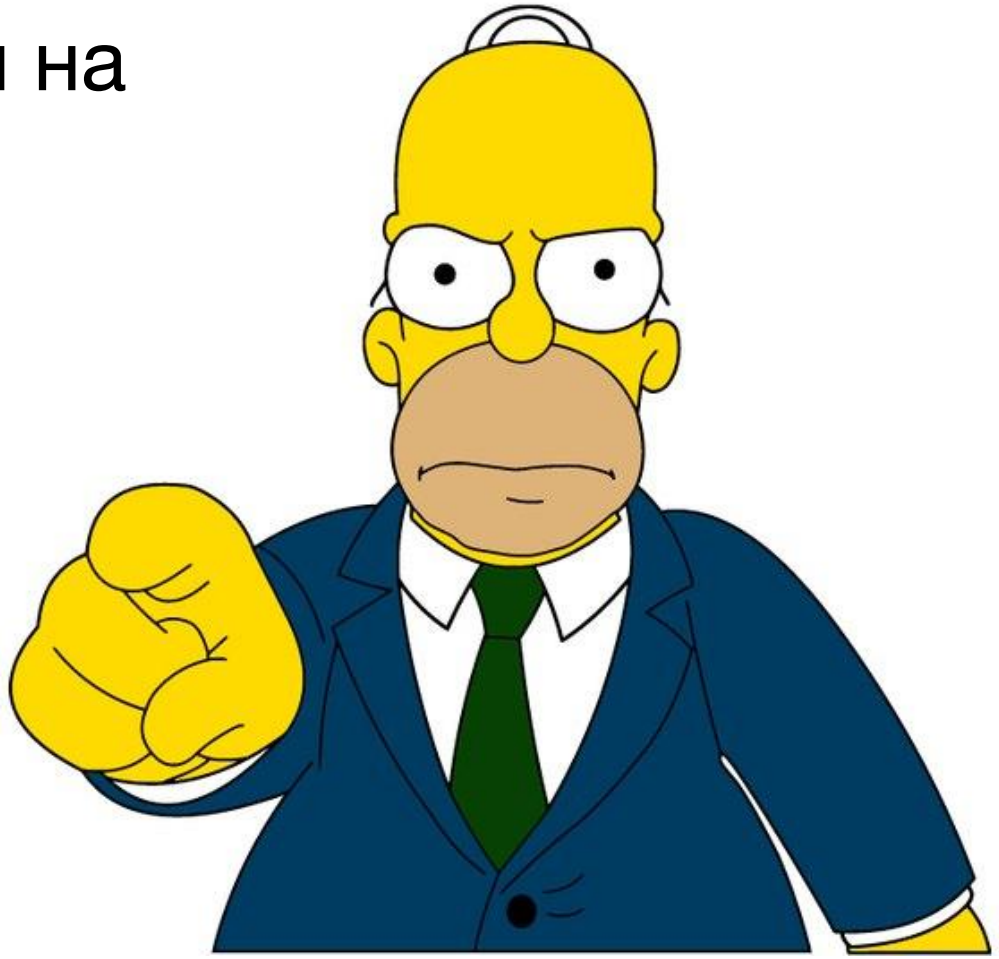
Занятие №4

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

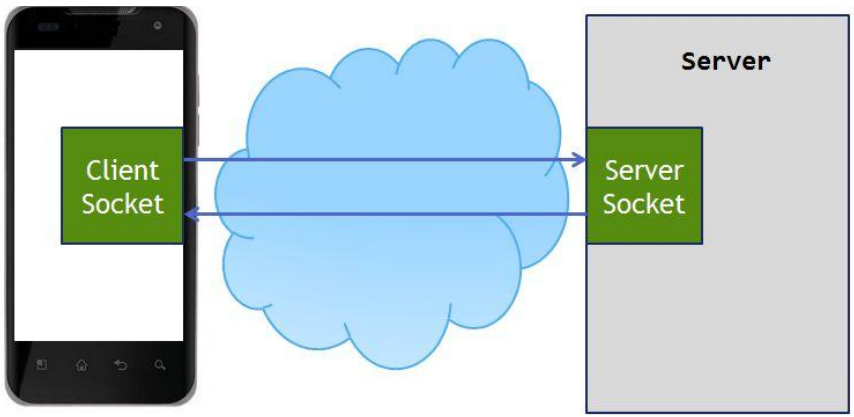
4 300 000 000

Хватит на всех!

Типы сетевых взаимодействий

На основе сокетов (Soket API)

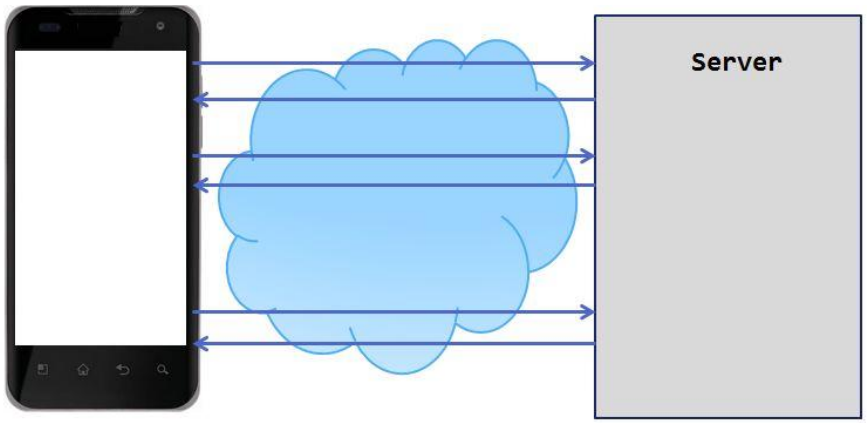
- Постоянное соединение
- Скорость
- Порядок доставки (TCP)
- Контроль доставки (TCP)



Типы сетевых взаимодействий

Частые опросы (Polling):

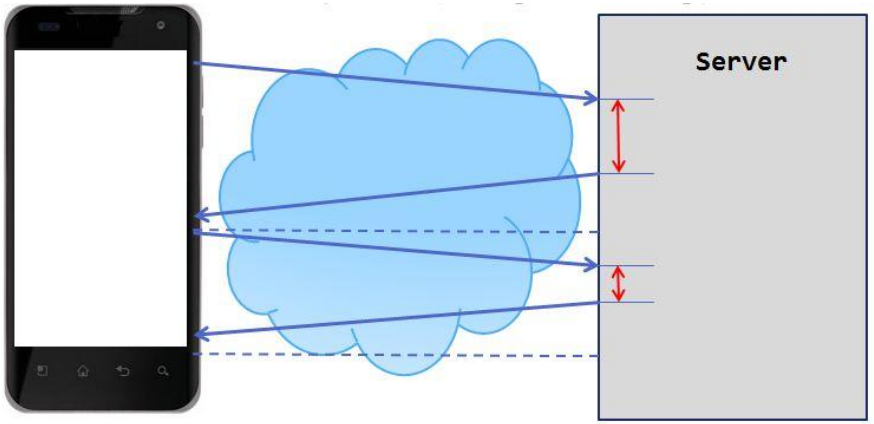
- Соединяемся, спрашиваем, получаем ответ отсоединяемся
- Частые установки соединения
- Стучимся всегда
- Много лишнего трафика
- Большая нагрузка на сервер



Типы сетевых взаимодействий

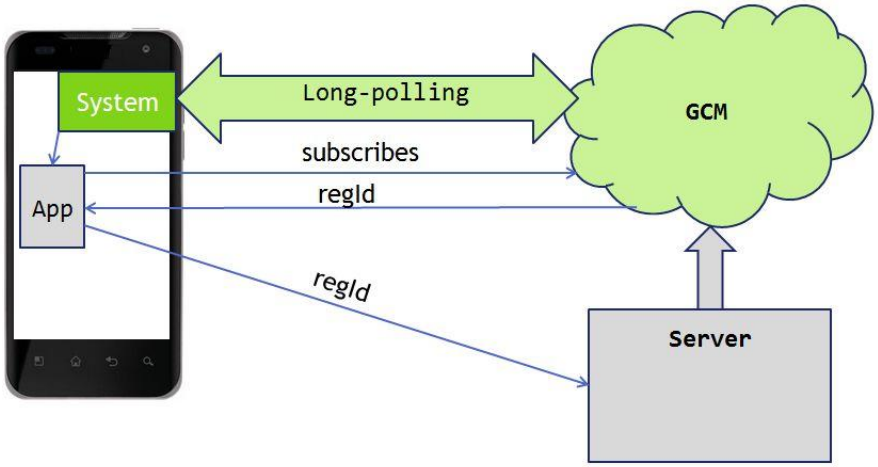
Длинные опросы (Long polling):

- 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 издаетесь не в Google Store



Socket

```
protected Void doInBackground(Void... arg0) {
  Socket socket = null;
  try {
    socket = new Socket(dstAddress, dstPort);

    ByteArrayOutputStream byteArrayOutputStream =
      new ByteArrayOutputStream(1024);
    byte[] buffer = new byte[1024];

    int bytesRead;
    InputStream inputStream = socket.getInputStream();

    while ((bytesRead = inputStream.read(buffer)) != -1){
      byteArrayOutputStream.write(buffer, 0, bytesRead);
      response += byteArrayOutputStream.toString("UTF-8");
    }

  } catch (UnknownHostException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }finally{
    if(socket != null){
      try {
        socket.close();
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
  }
  return null;
}
```

1. Создаете сокет
2. Получаете от него InputStream
3. Читаете блоками
4. Закрываете сокет
5. Все в блоке try

Протокол HTTP

Метод GET

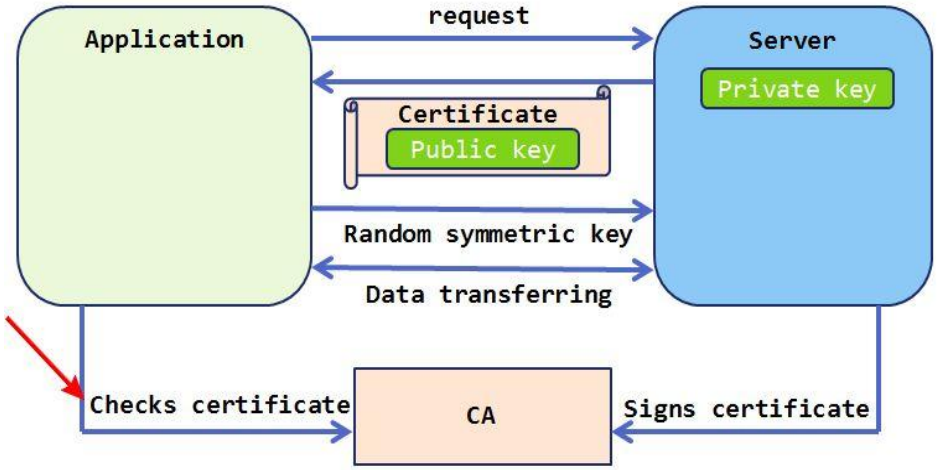
```
GET / HTTP/1.1
Host: mail.ru
User-Agent: WebKit 1.0
Accept: text/html
Connection: close


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)

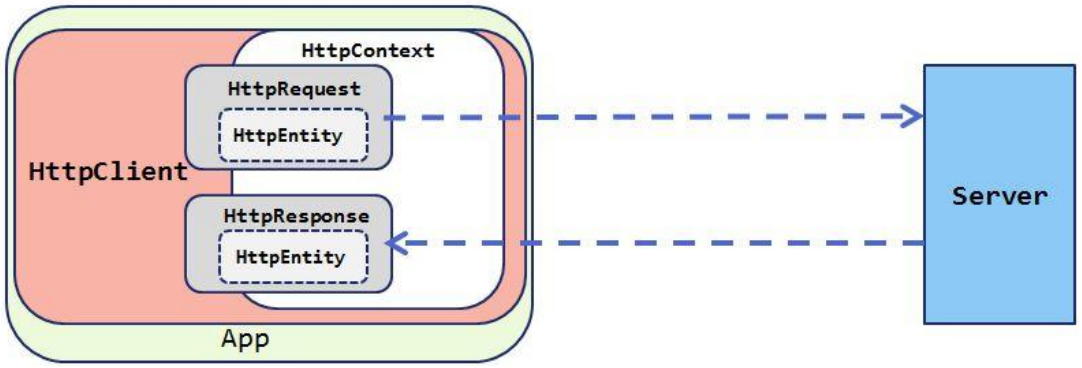


Как устанавливается защищенное соединение. Стрелка показывает на проверку подлинности сертификата.

- Есть проблемы
- На платформах < Android 4.0 при попытке выполнить сетевой запрос по HTTPS вылетит SSLHandshakeException
- На платформе ниже 4.0 можно:
  - Доверять любым сертификатам

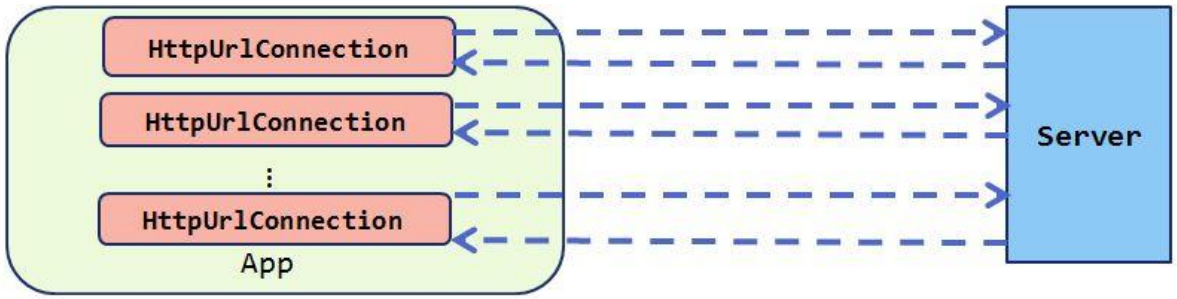
HttpClient

- 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

AndroidManifest.xml

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

Запрос страницы

```
URL url = new URL("http://mail.ru");
HttpURLConnection connection = null;
try {
    connection = (HttpURLConnection) url.openConnection();
    connection.setRequestMethod("GET");

    InputStream is = new BufferedInputStream(connection.getInputStream());
    readStream(is);
}
finally {
    if (connection != null) {
        connection.disconnect();
    }
}
```

Как использовать

Отправка данных на сервер

```
URL url = new URL("http://mail.ru");
HttpURLConnection connection = null;
try {
    connection = (HttpURLConnection) url.openConnection();
    connection.addRequestProperty("User-Agent", "My Application 1.0");
    connection.addRequestProperty("Content-Type", "application/json; charset=utf-8");
    connection.setDoOutput(true);
    connection.setRequestMethod("POST");
    connection.setChunkedStreamingMode(0);

    OutputStream os = new BufferedOutputStream(connection.getOutputStream());
    writeStream(os);

    InputStream is = new BufferedInputStream(connection.getInputStream());
    readStream(is);
}
finally {
    if (connection != null) {
        connection.disconnect();
    }
}
```

WebView

AndroidManifest.xml

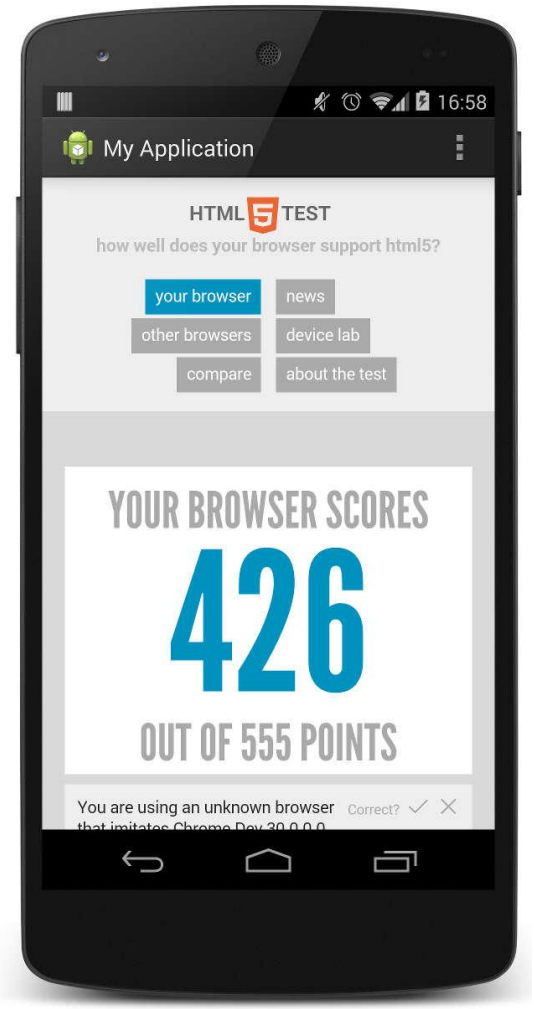
```
<uses-permission android:name="android.permission.INTERNET" />
```

activity_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<WebView  xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
/>
```

Java code

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.loadUrl("http://www.example.com");
```



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

```
public static Lesson createFromJson(String jsonString) throws JSONException {
    JSONTokener json = new JSONTokener(jsonString);
    JSONObject data = (JSONObject)json.nextValue();
    Lesson lesson = Lesson.createWithName(data.getString("name"));
    lesson.timestamp = data.getLong("timestamp");
    lesson.filepath = data.getString("filepath");
    JSONArray cards = data.getJSONArray("cards");
    for (int i = 0; i < cards.length(); ++i) {
        final String cardString = cards.getString(i);
        lesson.add(Card.createFromString(cardString));
    }
    return lesson;
}
```

JSON. GSON

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

Файл build.gradle

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

```
import com.google.gson.Gson;

public static void toJsonAndBack() {
    Gson gson = new Gson();
    MyType target = new MyType();
    String json = gson.toJson(target);
    MyType target2 = gson.fromJson(json, MyType.class);
}
```

JSON. GSON - чуть более сложный пример

```
import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;

public class User {
    @SerializedName("login")
    @Expose
    private String login;

    @SerializedName("id")
    @Expose
    private Integer id;
}

public class Followers {
    private List<User> users;

    public static Followers createFromJSON(String json) {
        Gson gson = new GsonBuilder().create();
        Followers followers = new Followers();
        followers.users = Arrays.asList(gson.fromJson(json, User[].class));

        return followers;
    }

    public List<User> getUsers() {
        return users;
    }
}
```

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

```
private final OkHttpClient client = new OkHttpClient();
```

Использование билдера

```
private final OkHttpClient client;

public ConfigureTimeouts() throws Exception {
  client = new OkHttpClient.Builder()
      .connectTimeout(10, TimeUnit.SECONDS)
      .writeTimeout(10, TimeUnit.SECONDS)
      .readTimeout(30, TimeUnit.SECONDS)
      .build();
}
```

OkHttp

Request

используется шаблон Builder

```
Request request = new Request.Builder()
    .header("Authorization", "Client-ID " + IMGUR_CLIENT_ID)
    .url("https://api.imgur.com/3/image")
    .post(requestBody)
    .build();
```

```
Request request = new Request.Builder()
    .url("https://api.github.com/repos/square/okhttp/issues")
    .header("User-Agent", "OkHttp Headers.java")
    .addHeader("Accept", "application/json; q=0.5")
    .addHeader("Accept", "application/vnd.github.v3+json")
    .build();
```

OkHttp

Response

- синхронный вызов

```
Response response = client.newCall(request).execute()
```

- асинхронный вызов

```
client.newCall(request).enqueue(new Callback() {
  @Override public void onFailure(Call call, IOException e) {
    // handle failure
  }

  @Override public void onResponse(Call call, Response response) throws IOException {
    // handle reponse
  }
});
```

OkHttp

Authentication

Встроенный обработчик ошибок с кодом 401 Unauthorized

```
OkHttpClient client = new OkHttpClient.Builder()
    .authenticator(new Authenticator() {
      @Override public Request authenticate(Route route, Response response) throws IOException {

        String credential = Credentials.basic("login", "password1");

        return response.request().newBuilder()
            .header("Authorization", credential)
            .build();
      }
    })
    .build();
```

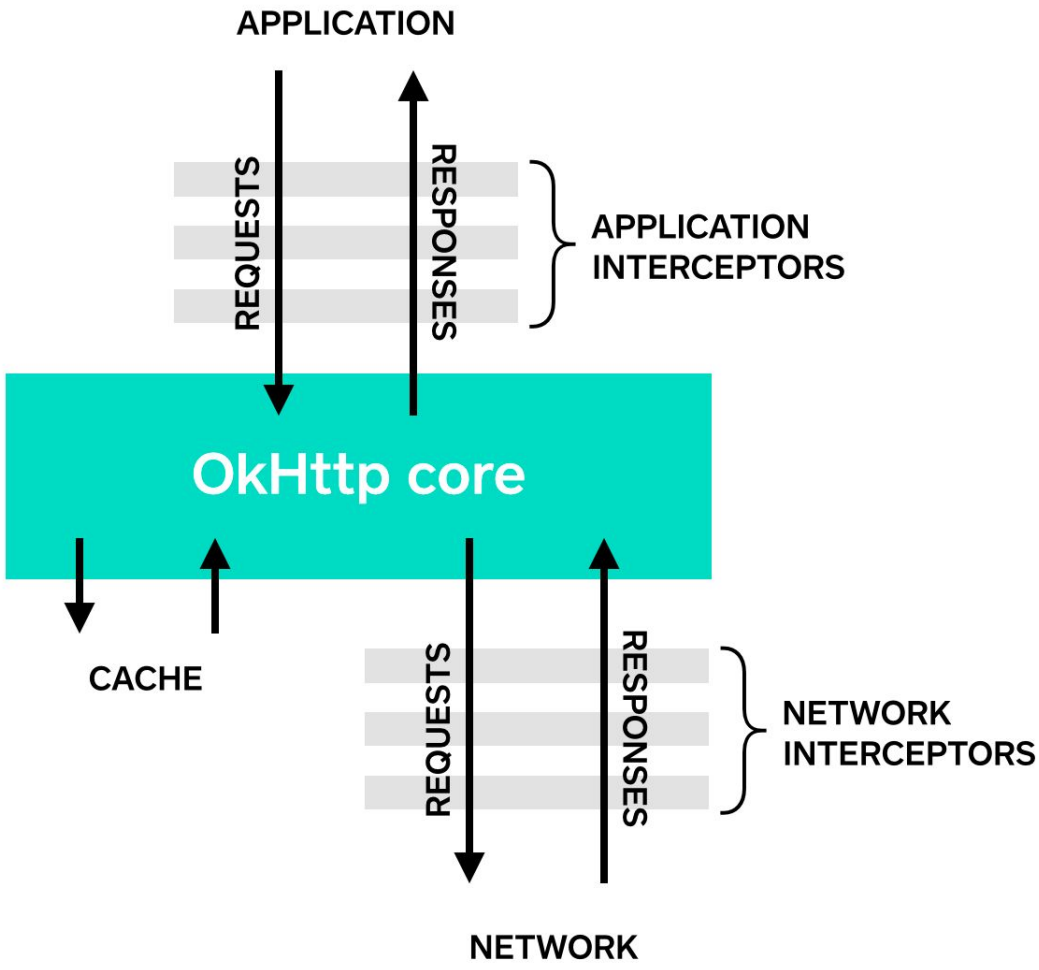
OkHttp

Interceptors

Возможность перехватывать и обрабатывать запросы и ответы в процессе выполнения

```
class LoggingInterceptor implements Interceptor {
  @Override public Response intercept(Interceptor.Chain chain) throws IOException {
    Request request = chain.request();

    long t1 = System.nanoTime();
    logger.info(String.format("Sending request %s on %s%n%s",
        request.url(), chain.connection(), request.headers()));

    Response response = chain.proceed(request);

    long t2 = System.nanoTime();
    logger.info(String.format("Received response for %s in %.1fms%n%s",
        response.request().url(), (t2 - t1) / 1e6d, response.headers()));

    return response;
  }
}
```

OkHttp


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

OkHttp

Interceptors

Добавление перехватчика

```
OkHttpClient client = new OkHttpClient.Builder()
    .addInterceptor(new LoggingInterceptor())
    .build();
```

```
OkHttpClient client = new OkHttpClient.Builder()
    .addNetworkInterceptor(new LoggingInterceptor())
    .build();
```

Retrofit

HTTP клиент для Java и Android

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

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

Retrofit

Описание API

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

```
@GET("group/{id}/users")
Call<List<User>> groupList(@Path("id") int groupId, @Query("sort") String sort);
```

```
@POST("users/new")
Call<User> createUser(@Body User user);
```

```
@Multipart
@PUT("user/photo")
Call<User> updateUser(@Part("photo") RequestBody photo, @Part("description") RequestBody description);
```

Retrofit

Создание сервиса

- используется Builder

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

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

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