

Протокол HTTP

Вызовы удаленных серверов



Григорий
Вахмистров



Григорий Вахмистров


Backend Developer в Tennisi.bet

План занятия

1. [Вспоминаем прошлые занятия](#)
2. [Протоколы](#)
3. [Протокол HTTP](#)
4. [Запрос по HTTP из Java](#)
5. [Что такое HTTPS?](#)
6. [Итоги](#)
7. [Домашнее задание](#)



Вспоминаем прошлые занятия



Могут ли быть созданы объекты из классов, помеченных модификатором `abstract`?

Поставьте в чате номер **одного** правильного ответа.


1. Да
2. Нет

Могут ли быть созданы объекты из классов, помеченных модификатором `abstract`?

Поставьте в чате номер **одного** правильного ответа.

1. Да

2. Нет, так у них может не быть реализации каких-либо методов



Обязательно ли в классе-наследнике реализовывать методы, помеченные как `abstract`?


Поставьте в чате номер **одного** правильного ответа:

1. Да, это обязательно во всех случаях
2. Нет, реализация методов не является обязательной
3. Да, если класс-наследник сам является не `abstract`

Обязательно ли в классе-наследнике реализовывать методы, помеченные как `abstract`?

Поставьте в чате номер **одного** правильного ответа:

1. Да, это обязательно во всех случаях
2. Нет, реализация методов не является обязательной
3. Да, если класс-наследник сам является не `abstract`



Какие функции выполняет понятие `interface` в Java?

Поставьте в чате номер **одного** правильного ответа:

1. Описывает поведение класса (описывает методы)
2. Описывает интерфейс использования полей класса
3. Описывает поведение класса (описывает методы) и конструкторы класса

Какие функции выполняет понятие interface в Java?

Поставьте в чате номер **одного** правильного ответа:

1. **Описывает поведение класса (описывает методы)**
2. Описывает интерфейс использования полей класса
3. Описывает поведение класса (описывает методы) и конструкторы класса

Возможна ли реализация класса от нескольких интерфейсов в одновременно?

```
class Cat implements Animal, Big, GetUpAt5Oclock {  
    ...  
}
```

Поставьте в чате номер **одного** правильного ответа:

1. Да
2. Нет

Возможна ли реализация класса от нескольких интерфейсов в одновременно?

```
class Cat implements Animal, Big, GetUpAt5Oclock {  
    ...  
}
```

Поставьте в чате номер **одного** правильного ответа:

1. Да, максимум можно реализовать 65535 в одном классе (specification)

2. Нет

Допускается/возможна ли реализация методов в interface?

Например, так:

```
interface Animal {  
    void run() {  
        System.out.println("moving fast");  
    }  
}
```

Поставьте в чате номер правильного ответа:

1. Да
2. Нет

Допускается/возможна ли реализация методов в interface?

Например, так:

```
interface Animal {  
    void run() {  
        System.out.println("moving fast");  
    }  
}
```

Поставьте в чате номер правильного ответа:

И да, и нет. Начиная с java 8, для interface появился модификатор default, позволяющий добавлять реализацию метода по умолчанию.



Протоколы

Какие бывают протоколы?

Несколько популярных определений понятия “протокол” в сети Интернет:

- Документ, содержащий запись всего происходившего на заседании, собрании, судебном процессе.
- Документ, удостоверяющий какой-либо факт (например, протокол осмотра).
- Акт о нарушении общественного порядка (применяется в Полиции).
- Дипломатическая совокупность правил, соблюдаемых правительствами и их дипломатическими представителями в международном общении.

Как вы думаете, какое определение близко термину “протокол” в IT?

Варианты ответа:

- Документ, содержащий запись всего происходившего на заседании, собрании, судебном процессе.
- Документ, удостоверяющий какой-либо факт (например, протокол осмотра).
- Акт о нарушении общественного порядка (применяется в Полиции).
- Дипломатическая совокупность правил, соблюдаемых правительствами и их дипломатическими представителями в международном общении.

Как вы думаете, какое определение близко термину “протокол” в IT?

Варианты ответа:

- Документ, содержащий запись всего происходившего на заседании, собрании, судебном процессе.
- Документ, удостоверяющий какой-либо факт (например, протокол осмотра).
- Акт о нарушении общественного порядка (применяется в Полиции).
- **Дипломатическая совокупность правил, соблюдаемых правительствами и их дипломатическими представителями в международном общении.**

Протоколы в IT

Мы уже познакомились с основополагающими протоколами - **UDP** и **TCP**.

UDP и TCP* являются протоколами транспортного уровня, то есть непосредственно отвечают за:

- транспортировку данных (байт);
- описание размера передаваемых данных;
- описание отправителя и мест назначения.

* В прикладном использовании протоколы UDP и TCP позволяют передавать любые данные и не регламентируют формат полезных данных.

Протоколы в IT

Если изменить порядок байт в протоколах **UDP** и **TCP**, то отправленные данные с одного компьютера в сети никогда *не найдут свой адресат*.

При использовании только протокола TCP или UDP, разработчику потребуется *каждый раз придумывать свой формат*, а всем кто захочет подключиться к нашему серверу, придется изучать этот формат передачи.

Такой подход был неудобен в разработке корпоративных и веб-приложений. Поэтому в модель OSI были включены протоколы прикладного уровня для стандартизации обмена данными.

Популярные протокола прикладного уровня

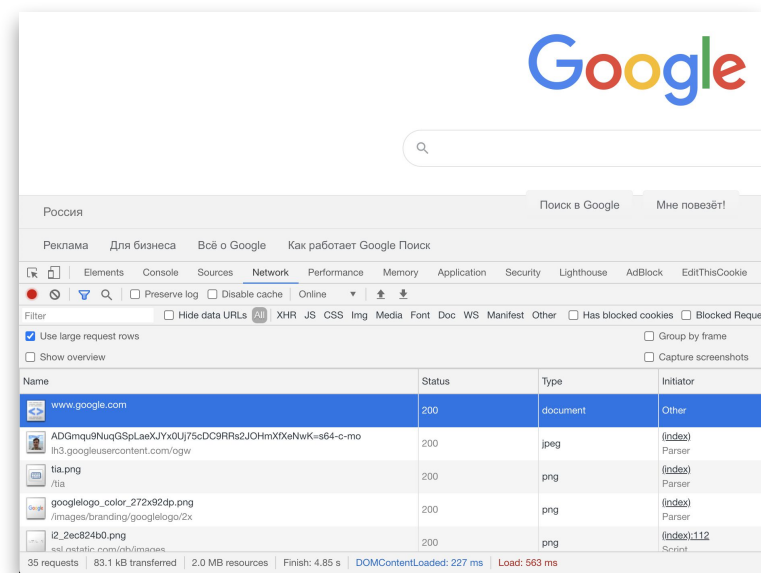
1. **HTTP** (HyperText Transfer Protocol) - самый популярный протокол благодаря сети Интернет, протокол обмена текстовой информацией, где приставка Hyper означать возможно передавать что-то большее чем просто текст - картинки, видео.
2. **FTP** (File Transfer Protocol) - протокол передачи файлов между компьютерами
3. **DNS** (Domain Name Service) - протокол для получения информации о доменных именах компьютеров зарегистрированных в сети. (доменное имя - текстовый псевдоним IP-адреса компьютера, например google.com, yandex.ru)
4. **POP3** и **SMTP** - протокола обмена почтовыми сообщениями
5. **SSH** (Secure Shell), **TELNET** - протокола позволяющие подключиться к удаленному компьютеру или приложению.

Протокол HTTP

HTTP - один из популярных прикладных протоколов. Он является базовым протоколом для обмена данных в сети Интернет.









Каждый раз, когда вы открываете сайт в вашем любимом браузере, браузер использует HTTP для получения текста страницы, картинок и видео.

Каждая строка в консоли вашего браузера - это HTTP-запрос. Сегодня мы рассмотрим HTTP подробнее.



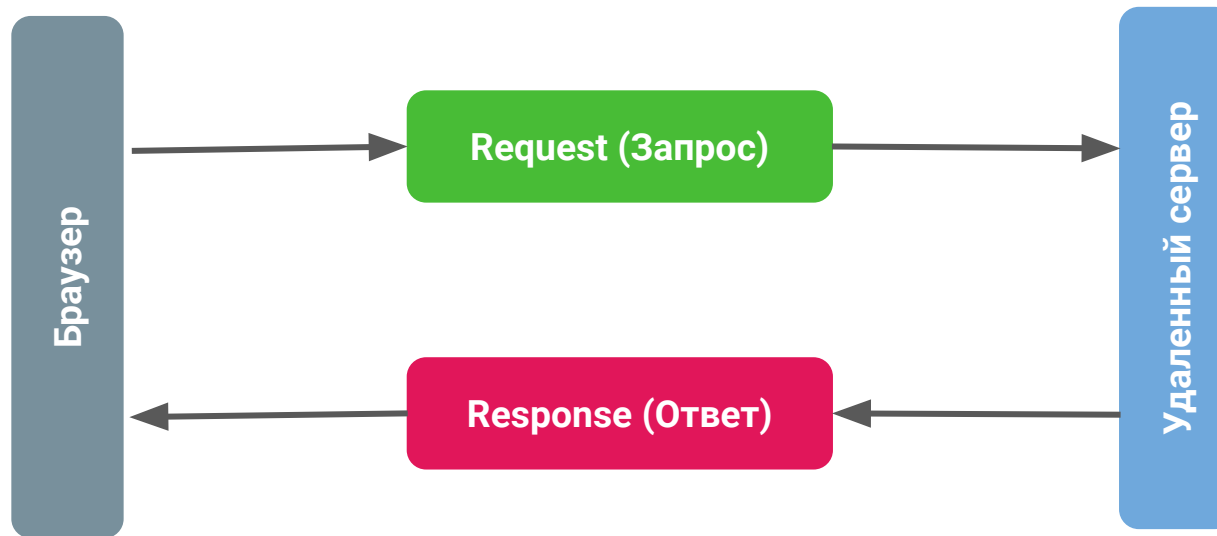
Протокол HTTP

Если кликнуть на любую строку, мы увидим детали запроса:

Name	× Headers Preview Response Initiator Timing Cookies
 www.google.com	▼ General Request URL: https://www.google.com/ Request Method: GET Status Code: 🟢 200 Remote Address: 108.177.14.104:443 Referrer Policy: no-referrer-when-downgrade
 ADGmqu9NuqGSpLaeXJYx0Uj75cDC9RRs2JOHmXfXeNwK=s64-c-mo lh3.googleusercontent.com/ogw	▼ Response Headers alt-svc: h3-27=":443"; ma=2592000,h3-25=":443"; ma=2592000, 92000,quic=":443"; ma=2592000; v="46,43" cache-control: private, max-age=0 content-encoding: br content-length: 62976 content-type: text/html; charset=UTF-8 date: Sat, 27 Jun 2020 13:08:33 GMT
 tia.png /tia	
 googlelogo_color_272x92dp.png /images/branding/googlelogo/2x	
 i2_2ec824b0.png ssl.gstatic.com/gb/images	
 tia.png www.gstatic.com/inputtools/images	
 desktop_searchbox_sprites302_hr.webp /images/searchbox	
 data:image/gif;base64...	

Описание протокола HTTP

HTTP - это клиент-серверный протокол. Каждый **запрос** (*request*) **отправляется серверу** - сервер его **обрабатывает и возвращает ответ** (*response*).



На месте браузера может быть любое приложение, умеющее работать с протоколом HTTP.



**Из чего состоит
HTTP-запрос?**

HTTP запрос (request)

Запрос состоит из следующих аргументов:

1. **Стартовая строка** (Starting line) — определяет тип сообщения, она обязательна для любого сообщения.
2. **Заголовки** (Headers) — характеризуют тело сообщения, параметры передачи и прочие сведения.
3. **Тело сообщения** (Message Body) — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

Запрос. Стартовая строка

Стартовые строки для запроса и ответа отличаются. Строка запроса состоит из следующих параметров: **метода, пути к ресурсу и протокола.**

Пример:

GET	/index.html	HTTP/1.1
метод	путь к ресурсу (url)	версия протокола

Запрос. Стартовая строка

В спецификации **http** описаны следующие возможные **методы**:

- **GET** - самый популярный метод, используется для получения данных от сервера;
- **POST** - второй по популярности метод, используется для добавления новых данных;
- **PUT** - метод используется для замены данных (например по идентификатору);
- **PATCH** - метод используется для частичного обновления данных;
- **DELETE** - метод удаления данных;
- **HEAD** - используется так как и метод GET, но в случаях когда нам не нужно тело ответа;
- **CONNECT** - метод для запроса создания виртуального туннеля между клиентом и сервером;
- **OPTIONS** - метод используется для запроса какие методы и типы данных поддерживает сервер для отправки и получения;
- **TRACE** - метод для трассировки запроса и отладки.

Запрос. Стартовая строка

Путь к ресурсу (/index.html) - относительный путь ресурса на сервере, к которому мы хотим направить наш запрос.

Таким запросом может быть веб-страница, файл, картинки, видео и любой другой доступный на сервере контент.

В пути ресурса можно указать параметры запроса - после имени ресурса добавьте `?` и перечислите параметры в формате:

“имя1=значение1&имя2=значение2”.*

* Перечислять параметры нужно через `&`.

Запрос. Стартовая строка

Пример запроса ресурса с параметрами:

```
/index.html?name=ivan&surname=semenov
```

Версия протокола (HTTP/1.1) - как уже следует из названия, этот аргумент версия используемого в запросе протокола.

Самая популярная версия на текущий момент - 1.1, самая последняя версия - 2.0. Несмотря на то, что версия 2.0 вышла в 2015 (стандарт), примерно 50% до сих пор работают используя версию 1.1.

Запрос. Заголовки

Если параметры запроса, которые указаны **в пути к ресурсу**, считаются **основными**, то **заголовки** - **дополнительные параметры запроса**.

Но один из них является обязательным: начиная с версии HTTP/1.1 обязательными является заголовок **HOST** - именно в нем описывается адрес сервера куда мы хотим направить наш запрос.

Пример:

```
GET /index.html?name=ivan&surname=semenov HTTP/1.1
```

```
Host: google.com
```



Запрос. Заголовки

Какие еще бывают заголовки?

Их достаточно много, с большинством можно познакомиться на странице [wiki](#).

Мы познакомимся с наиболее часто используемыми.

Запрос. Часто используемые заголовки

1. **User-Agent** - имя приложения отправляющего запрос
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5)
2. **Authorization** - заголовок авторизации, через него обычно отправляются логин и пароль к ресурсу
Authorization: Basic AKJljns31jkl==
3. **Accept** - заголовок описывающий форматы данных, которые может обработать клиент
Accept: text/html,application/xhtml+xml
4. **Accept-Language** - в этом заголовке указывается локаль (язык) клиента
Accept-Language: en-us
5. **Cookie** - заголовок, который отправляет сохраненную в браузере информацию, благодаря ему вам не нужно каждый раз авторизовываться в gmail
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120

Запрос. Пример запроса с заголовками

```
GET /index.html HTTP/1.1
Host: google.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120
Pragma: no-cache
Cache-Control: no-cache
```

Запрос. Тело

Тело запроса - необязательный параметр, который используется для передачи данных (текста, данных в формате json и других форматах)*.

Добавление данных в запрос обозначается заголовками:

- **Content-Length** - длина тела запроса в байтах;
- **Content-Type** - описывает тип передаваемого контента, например: application/json, text/html; charset=utf-8 или multipart/form-data;
- **Content-Encoding** - описывает сжатие тела запроса gzip, compress, deflate, identity, br

Когда вы загружаете данные на google drive, dropbox или любое другое файловое хранилище в теле запроса передаются ваши файлы.

Запрос. Тело

Адрес

`http://google.ru/index.html`

Заголовки

`Content-Type: application/json, charset=utf-8`

Тело запроса

`{"id":"erfqwearf","product":"143"}`



**Можно ли добавить
свой HTTP-метод?**

Свой HTTP-метод



Техническая возможность есть, но на практике придется писать свой собственный клиент для работы с вашим новым методом. Такой подход **не рекомендуется**.



**Из чего состоит
HTTP-ответ?**

HTTP-ответ (response)

Формат ответа похож на формат запроса и **состоит из:**

1. **Стартовая строка** (Starting line) — описывает ответ;
2. **Заголовки** (Headers) — характеризуют дополнительное описание ответа;
3. **Тело сообщения** (Message Body) — данные ответа.

Ответ. Стартовая строка

Стартовая строка ответа **состоит из версии протокола и кода ответа** (к коду дополнительно может быть добавлено пояснение):

HTTP/1.1

версия протокола

200

Код ответа


Ok

Пояснение

Ответ. Стартовая строка

Коды ответа делятся на следующие группы:

- **100 - 199** - информационные;
- **200 - 299** - успешные
- **300 - 399** - перенаправления;
- **400 - 499** - клиентские ошибки;
- **500 - 599** - серверные ошибки.



**С какими популярными
кодами ответов вы
сталкивались?**

Ответ. Популярные код ответа

- **200 OK** - запрос прошел успешно.
- **301 Moved Permanently** - адрес запрашиваемого ресурса был изменен, на постоянной основе.
- **302 Found** - запрошенный ресурс временно изменен.
- **400 Bad Request** - "плохой запрос", сервер не понимает запрос из-за неверного синтаксиса.
- **401 Unauthorized** - "запрос неавторизован", для получения запрашиваемого ответа нужна аутентификация.
- **404 Not Found** - "не найден", сервер не может найти запрашиваемый ресурс.
- **405 Method Not Allowed** - сервер не знает о запрашиваемом методе.
- **500 Internal Server Error** - внутренняя ошибка сервера.
- **503 Service Unavailable** - сервис недоступен.

Ответ. Заголовки

Как и в случае с запросом, **заголовки несут дополнительную информацию** в ответе и не являются обязательными.

Используются для:

- Описание типа ответа **Content-Type: application/json, text/plain** и другие
- В случае кодов ответа 301 и 302 указывают на урл перенаправления **Location: <http://yandex.ru/index.html>**
- Возвращают значения для дальнейшего использования в запросах: **Set-Cookie: session_id=wfsfasfasfadsf; HttpOnly**

С остальными заголовками можно познакомиться на странице [wiki](#).

Ответ. Тело

Как и в случае с запросом, **тело ответа является необязательным параметром** ответа и используется для передачи данных - например, текста, данных формате json или любом другом формате.

Добавление данных в запрос обозначается заголовками:

- **Content-Length** - длина тела запроса в байтах;
- **Content-Type** - описывает тип передаваемого контента, например: application/json, text/html; charset=utf-8 или multipart/form-data;
- **Content-Encoding** - описывает сжатие тела запроса gzip, compress, deflate, identity, br

Ответ. Тело

Адрес

HTTP/1.1 200 OK

Заголовки

Content-Type: application/json, charset=utf-8

Тело ответа

```
{"product": 143, "cost": 300,}
```



Запрос по HTTP из Java

Способы отправки запроса по HTTP

Для отправки запроса средствами Java по HTTP нужно воспользоваться одним из следующих вариантов:

1. **Apache**
2. **OkHttp**
3. **JDK HttpClient since 11**
4. **JDK HttpURLConnection since 1.1**
(не рекомендуется, так как нет поддержки HTTP/2.0)

Эти способы похожи между собой, но используют разный набор настраиваемых параметров. Самый популярный способ - это использование **Apache Http Client** и **OkHttp Client**.

Мы с вами рассмотрим создание запросов с помощью Apache Http Client.

Создание и настройка JAVA клиента

1. Создадим maven проект и добавим зависимость:

```
<dependency>
  <groupId>org.apache.httpcomponents</groupId>
  <artifactId>httpclient</artifactId>
  <version>4.5.12</version>
</dependency>
```

2. Создадим Main класс и добавим в методе main класс HttpClientBuilder:

```
import org.apache.http.client.config.RequestConfig;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClientBuilder;

import java.io.IOException;

public class Main {

    public static void main(String[] args) throws IOException {
        CloseableHttpClient httpClient = HttpClientBuilder.create()
            .setUserAgent("My Test Service")
            .setDefaultRequestConfig(RequestConfig.custom()
                .setConnectTimeout(5000) // максимальное время ожидание подключения к серверу
                .setSocketTimeout(30000) // максимальное время ожидания получения данных
                .setRedirectsEnabled(false) // возможность следовать редиректу в ответе
            ).build())
            .build();
    }
}
```

Вызов удаленного сервера

Вызовем удаленный сервер по url: <https://jsonplaceholder.typicode.com/posts> и посмотрим какие данные вернет нам сервер. Чтобы сделать вызов, нам нужно создать объект запроса, который необходимо передать в созданный http client

```
public static final String REMOTE_SERVICE_URI = "https://jsonplaceholder.typicode.com/posts";

public static void main(String[] args) throws IOException {
    CloseableHttpClient httpClient = HttpClientBuilder.create()
        .setUserAgent("My Test Service")
        .setDefaultRequestConfig(RequestConfig.custom()
            .setConnectTimeout(5000)
            .setSocketTimeout(30000)
            .setRedirectsEnabled(false)
            .build())
        .build();

    // создание объекта запроса с произвольными заголовками
    HttpGet request = new HttpGet(REMOTE_SERVICE_URI);
    request.setHeader(HttpHeaders.ACCEPT, ContentType.APPLICATION_JSON.getMimeType());

    // отправка запроса
    CloseableHttpResponse response = httpClient.execute(request);

    // вывод полученных заголовков
    Arrays.stream(response.getAllHeaders()).forEach(System.out::println);

    // чтение тела ответа
    String body = new String(response.getEntity().getContent().readAllBytes(), StandardCharsets.UTF_8);
    System.out.println(body);
}
```

Вывод заголовков ответа от сервера

Date: Sat, 04 Jul 2020 14:07:25 GMT

Content-Type: application/json; charset=utf-8

Transfer-Encoding: chunked

Connection: keep-alive

Set-Cookie: __cfduid=dd37e41856980b837e4145e4c910fa0441593871645; expires=Mon, 03-Aug-20 14:07:25

X-Powered-By: Express

X-Ratelimit-Limit: 10

X-Ratelimit-Remaining: 9

X-Ratelimit-Reset: 1593853735

Vary: Origin, Accept-Encoding

Access-Control-Allow-Credentials: true

Cache-Control: max-age=43200

Pragma: no-cache

Expires: -1

X-Content-Type-Options: nosniff

Etag: W/"6b80-Ybsq/K6GwwqrYkAsFxqDXGC7DoM"

Via: 1.1 vegur

CF-Cache-Status: HIT

Вывод тела ответа от сервера

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae o"
  },
  {
    "userId": 1,
    "id": 2,
    "title": "qui est esse",
    "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditi"
  }
  ...
]
```

Сервер нам возвращает список постов пользователей **текстом в формате json**, для полноценной работы их нужно **преобразовать в список java объекты**.

Преобразование json в java объекты

Для преобразования из json в java, нужно добавить библиотеку Jackson или Gson, это две самые популярные библиотеки для работы с этим форматом в java. **Jackson** - популярна на backend, **Gson** - популярна в Android. Мы будем использовать первую.

Добавим зависимость в maven:

```
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.11.1</version>
</dependency>
```

И создадим класс, в который будем преобразовывать json post в java post - Post.java с полями:

```
int userId;
int id;
String title;
String body;
```

Класс Post.java

```
import com.fasterxml.jackson.annotation.JsonProperty;

public class Post {

    private final int userId;
    private final int id;
    private final String title;
    private final String body;

    public Post(
        @JsonProperty("userId") int userId,
        @JsonProperty("id") int id,
        @JsonProperty("title") String title,
        @JsonProperty("body") String body
    ) {
        this.userId = userId;
        this.id = id;
        this.title = title;
        this.body = body;
    }

    public int getUserId() {
        return userId;
    }

    // ... все getters

    @Override
    public String toString() {
        return "Post" +
            "\n  userId=" + userId +
            "\n  id=" + id +
            "\n  title=" + title +
            "\n  body=" + body;
    }
}
```

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere",
    "body": "quia et suscipit..."
  },
  {
    "userId": 1,
    "id": 2,
    "title": "qui est esse",
    "body": "est rerum tempore vitae\nsequi sint ni..."
  },
  {
    "userId": 1,
    "id": 3,
    "title": "ea molestias quasi exercitationem repellat",
    "body": "et iusto sed quo iure\nvoluptatem..."
  }
  ...
  {
    "userId": 10,
    "id": 100,
    "title": "at nam consequatur ea labore ea harum",
    "body": "cupiditate quo est a modi nesciunt..."
  }
]
```

Преобразование json в java объекты

Аннотации `@JsonProperty` в классе `Post.java` нужны чтобы конструктор понимаю какие поля в json соотносятся с полями в java классе.

Создадим в классе `Main.java`, json mapper:

```
public static ObjectMapper mapper = new ObjectMapper();
```

А в методе `main` добавим код для преобразования json в java:

```
List<Post> posts = mapper.readValue(  
    response.getEntity().getContent(), new  
    TypeReference<List<Post>>() {});
```

Выведем список постов на экран:

```
posts.forEach(System.out::println);
```


Преобразование json в java объекты

```
public class Main {

    public static final String REMOTE_SERVICE_URI = "https://jsonplaceholder.typicode.com/posts";
    public static final ObjectMapper mapper = new ObjectMapper();

    public static void main(String[] args) throws IOException {
        CloseableHttpClient httpClient = HttpClientBuilder.create()
            .setUserAgent("My Test Service")
            .setDefaultRequestConfig(RequestConfig.custom()
                .setConnectTimeout(5000)
                .setSocketTimeout(30000)
                .setRedirectsEnabled(false)
                .build())
            .build();

        HttpGet request = new HttpGet(REMOTE_SERVICE_URI);
        request.setHeader(HttpHeaders.ACCEPT, ContentType.APPLICATION_JSON.getMimeType());

        CloseableHttpResponse response = httpClient.execute(request);
        Arrays.stream(response.getAllHeaders()).forEach(System.out::println);

        List<Post> posts = mapper.readValue(response.getEntity().getContent(), new TypeReference<List<Post>>() {});
        posts.forEach(System.out::println);
    }
}
```

Результат запуска программы

Headers...

Post

userId=1

id=1

title=sunt aut facere repellat provident occaecati excepturi optio reprehenderit

body=quia et suscipit

suscipit recusandae consequuntur expedita et cum

reprehenderit molestiae ut ut quas totam

nostrum rerum est autem sunt rem eveniet architecto

Post

userId=1

id=2

title=qui est esse

body=est rerum tempore vitae

sequi sint nihil reprehenderit dolor beatae ea dolores neque

fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis

qui aperiam non debitis possimus qui neque nisi nulla

...



Что такое HTTPS?

HTTPS

HTTPS (Hypertext Transport Protocol Secure) – это протокол в основе которого лежит протокол HTTP, он обеспечивает конфиденциальность обмена данными между сайтом и пользовательским устройством.

Безопасность информации обеспечивается за счет использования криптографических протоколов SSL/TLS, имеющих **3 уровня защиты**:

- 1. Шифрование данных** - позволяет избежать их перехвата.
- 2. Сохранность данных** - любое изменение данных фиксируется.
- 3. Аутентификация** - защищает от перенаправления пользователя.

Все запросы в этом случае подписываются сертификатами шифрования (ключами) и дешифруются на стороне получателя - на сервере или клиенте).

Итоги

- Детально рассмотрели протокол http.
- Создали запрос к удаленному серверу по http.
- Разобрали json ответ от сервера с помощью библиотеки Jackson.



Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и
пишите отзыв о лекции!**

Григорий Вахмистров