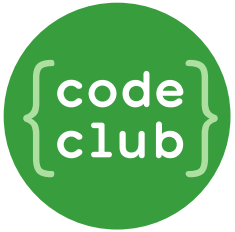
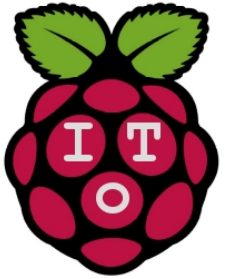


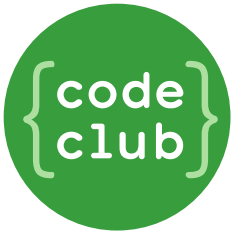
Internet of Things



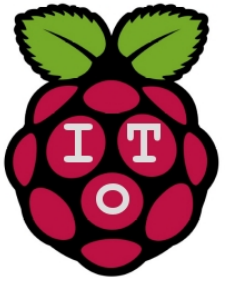
REST в IoT

Шадринск
2018-2019

М. В. Шохирев



REST



Распределённые системы эффективно работают на основе архитектурного стиля **REST** (Representational State Transfer), который основан на 3-х общеизвестных стандартах (RFC):

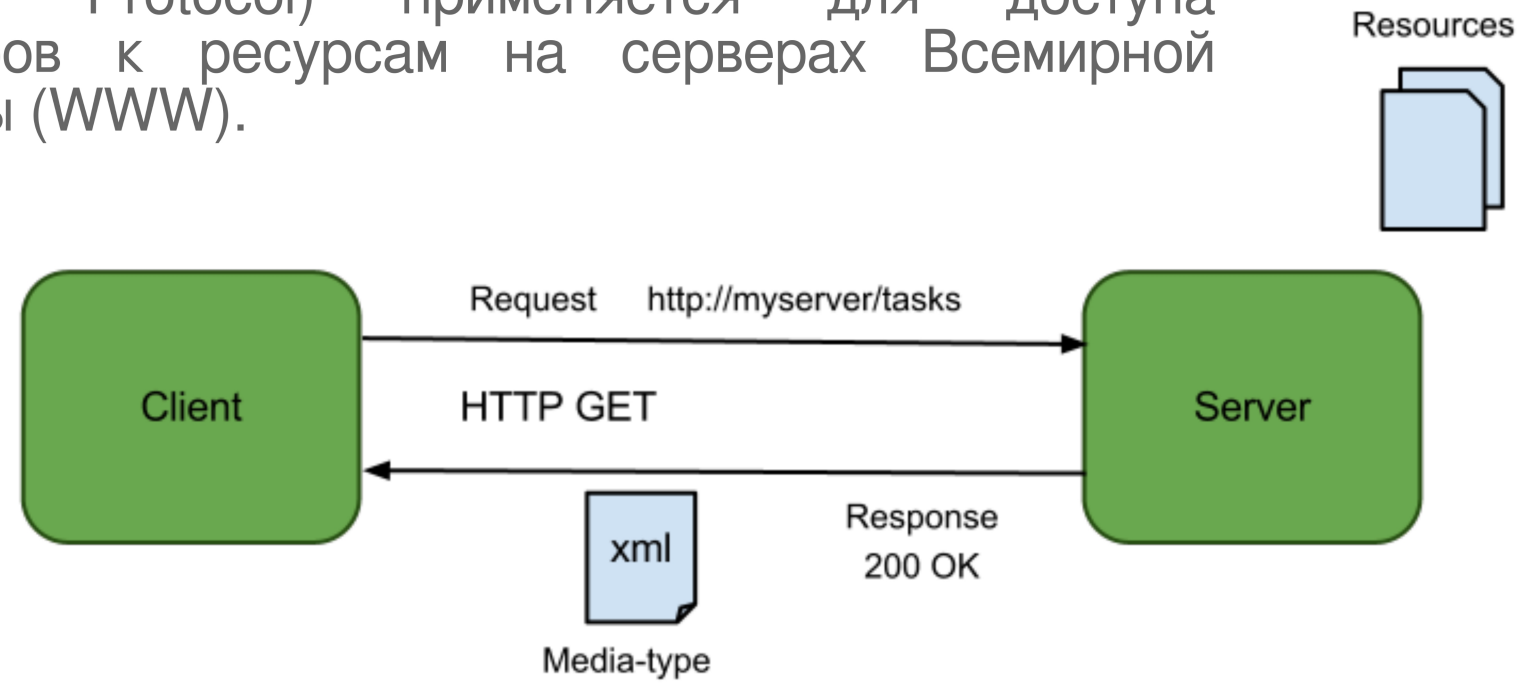
I. Протокол **HTTP** — *для доступа к данным (ресурсам).*
II. Идентификаторы **URI** — *для именования ресурсов*.
III. Форматы **MIME** — *для представления данных разных типов.*

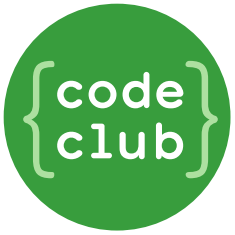
Именно на этих 3-х стандартах основана Всемирная Паутина (WWW).

REST используется как программный интерфейс (API) для манипуляциями с любыми ресурсами (данными, расположенными на сервере).

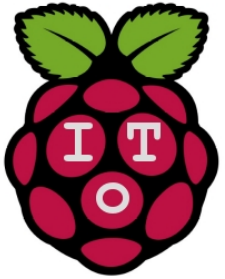
HTTP

I. Протокол передачи гипертекста **HTTP** (HyperText Transfer Protocol) применяется для доступа браузеров к ресурсам на серверах Всемирной Паутины (WWW).



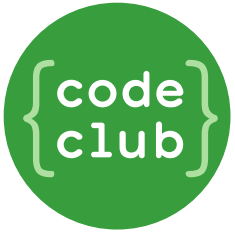


Изменение данных

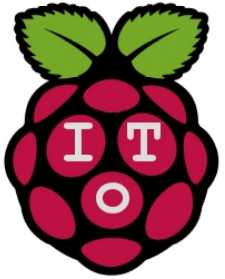


В протоколе **HTTP** предусмотрен стандартный набор операций (POST, GET, PUT, DELETE) для манипулирования данными: добавить, прочитать, изменить, удалить (Create, Read, Update, Delete = CRUD).

GET /resource/collection	запросить коллекцию ресурсов
POST /resource/collection	добавить единичный ресурс в коллекцию
GET /resource/collection/*id*	запросить единичный ресурс
PUT /resource/collection/*id*	изменить ресурс (или коллекцию)
DELETE /resource/collection/*id*	удалить ресурс (или коллекцию)

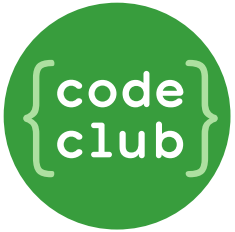


WWW как БД

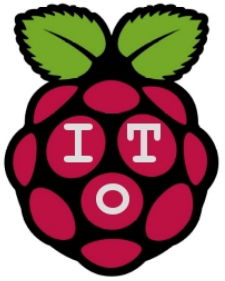


Операции (POST, GET, PUT, DELETE) протокола **HTTP** соответствуют основным командам **SQL** для изменения информации в базах данных: INSERT, SELECT, UPDATE, DELETE).

HTTP		**SQL**
GET /resource/collection	запросить коллекцию ресурсов	SELECT * FROM collection
POST /resource/collection	добавить единичный ресурс в коллекцию	INSERT INTO collection (names) VALUES (list)
GET /resource/collection/*id*	запросить единичный ресурс	SELECT * FROM collection WHERE pk=*id*
PUT /resource/collection/*id*	изменить ресурс (или коллекцию)	UPDATE collection WHERE pk=*id* SET *name=value*
DELETE /resource/collection/*id*	удалить ресурс (или коллекцию)	DELETE FROM collection WHERE pk=*id*



URL



II. Универсальный указатель ресурса **URL** (Uniform Resource Locator) применяется для именования ресурсов Всемирной Паутины (WWW):

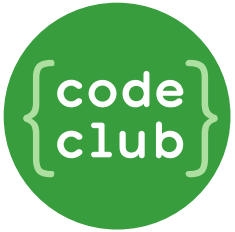
<схема>://[<логин>[:<пароль>]@]<хост>[:<порт>]][/<URL-путь>][?<параметры>][#<якорь>]

Общепринятые схемы (протоколы) URL:

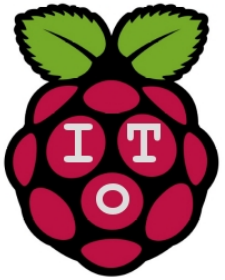
- ftp — Передача файлов по FTP
- http — Передача гипертекста по HTTP
- https — Передача гипертекста по HTTP с использованием шифрования (как правило, SSL или TLS)
- mailto — Отправка электронной почты по адресу
- smb — Передача файлов по SMB/CIFS (разделяемые папки в MS Windows)
- telnet — Сеанс интерактивного доступа по Telnet
- xmpp — Мгновенный обмен сообщениями по XMPP (Jabber)
- file — Передача локальных файлов на хосте
- data — Непосредственные данные (Data: URL)
- tel — Звонок по указанному телефону

В **REST** применяется URL с доступом по HTTP:

http://subdomain.domain.tld/path/to/resource.format



URL ← URI



URL применяется в **REST** для именования ресурсов, которые подразделяются на 2 вида:

- единичные ресурсы (объекты) и
- коллекции (списки, наборы, массивы) ресурсов.

Но доступ к любому из видов производится однотипно:

- server.net/resource/collection — коллекция;
- server.net/resource/collection/filter — часть коллекции;
- server.net/resource/collection/id — ресурс;
- application.info/class/ — список объектов класса;
- application.info/class/id — объект из списка;
- application.info/class/subclass — объекты подкласса;
- localhost/path/to/dir/ — список файлов в каталоге;
- localhost/path/to/dir/file.txt — файл;

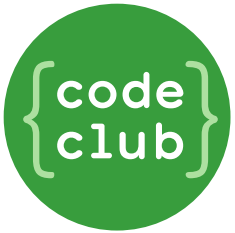
URL — один из видов уникального идентификатора ресурса **URI** (Uniform Resource Identifier).

MIME: типы данных

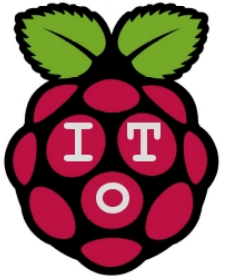
III. Типы данных **MIME** (Multipurpose Internet Mail Extensions) — стандартизированные представления данных, в т. ч. мультимедийных, были разработаны ещё для вложений электронной почты и дополнены новыми типами после появления WWW:

- application/json - JavaScript Object Notation, JSON (RFC 4627);
- application/octet-stream - двоичный файл (RFC 2046);
- application/pdf - Portable Document Format, PDF (RFC 3778);
- application/xml - eXtensible Markup Language, XML;
- audio/mpeg - MPEG-аудио, включая MP3 (RFC 3003);
- image/png - Portable Network Graphics, PNG (RFC 2083);
- message/rfc822 - сообщения E-mail (RFC 2045, 2046);
- text/csv - Comma-Separated Values, CSV (RFC 4180);
- text/html - HyperText Markup Language, HTML (RFC 2854);
- text/plain - текстовые данные (RFC 2046, 3676);
- video/mpeg - MPEG-1 (RFC 2045, 2046);

Нестандартные типы (x): application/x-font-ttf, text/x-jquery-tmpl, ...
Типы вендоров (vnd): application/vnd.ms-excel, audio/vnd.wave, ...



MIME: представления



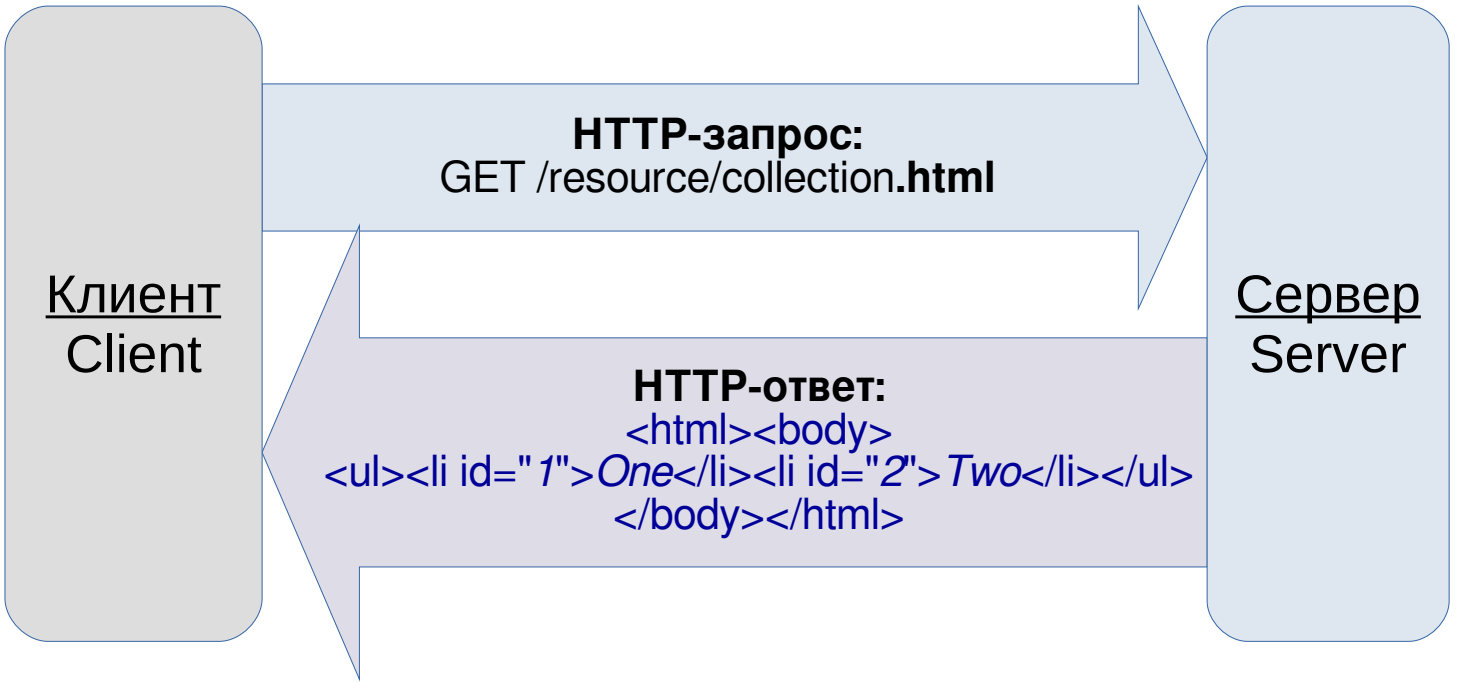
1. Клиент может запросить у HTTP-сервера требуемое представление ресурса (media type) одним из способов:

- как часть URI (выглядит, как суффикс файла):
  - resource.HTML
  - resource.XML
  - resource.JSON
  - resource.PDF
- в заголовке Accept запроса HTTP:
  - Accept: text/html
  - Accept: application/xml
  - Accept: application/json
  - Accept: application/pdf

2. Сервер сообщает о представлении передаваемого ресурса в заголовке Content-Type ответа HTTP.

REST: доступ к данным

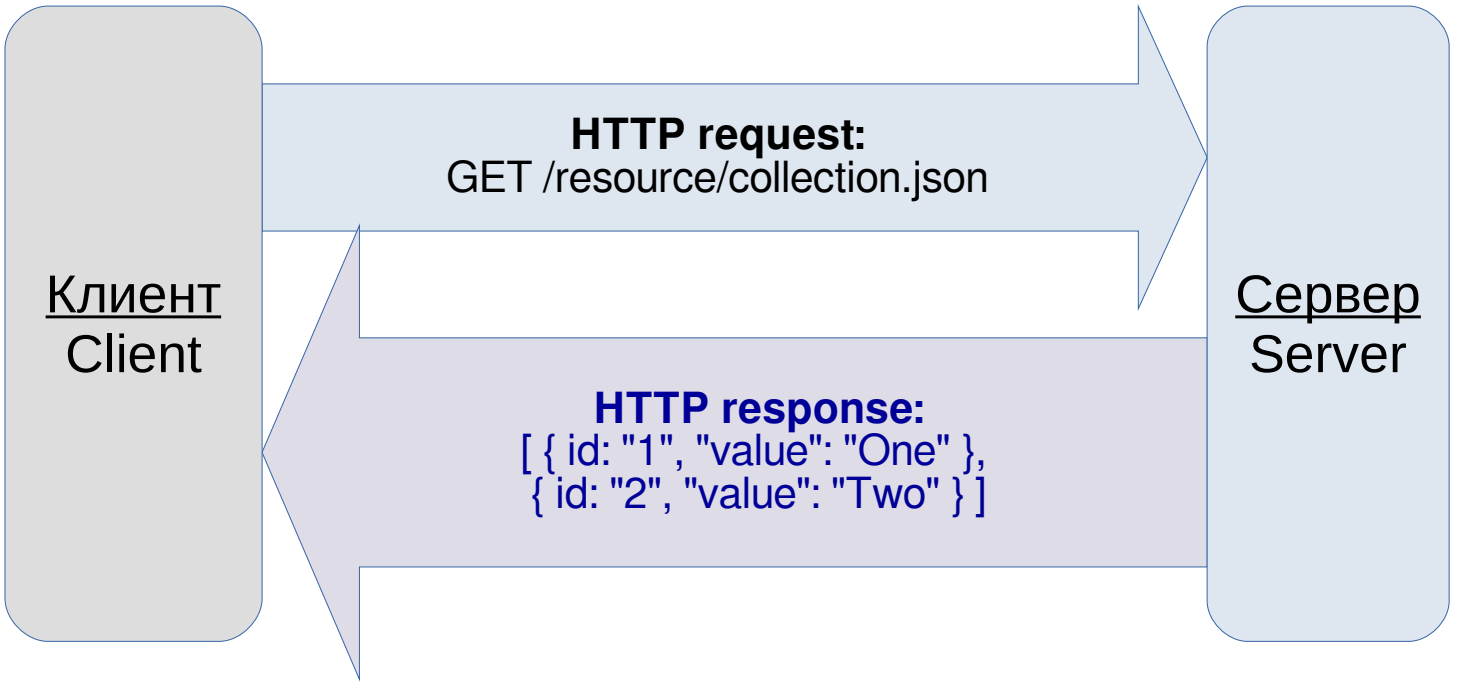
- Ресурс уникально **идентифицируется** по URI.
- **REST-запрос**: глагол HTTP (GET / PUT / POST / DELETE) для выполнения действия.
- **Данные** для обновления: в параметрах запроса.
- Желаемое **представление**: в параметрах запроса.
- **Ресурс** в нужном представление — в HTTP-ответе.



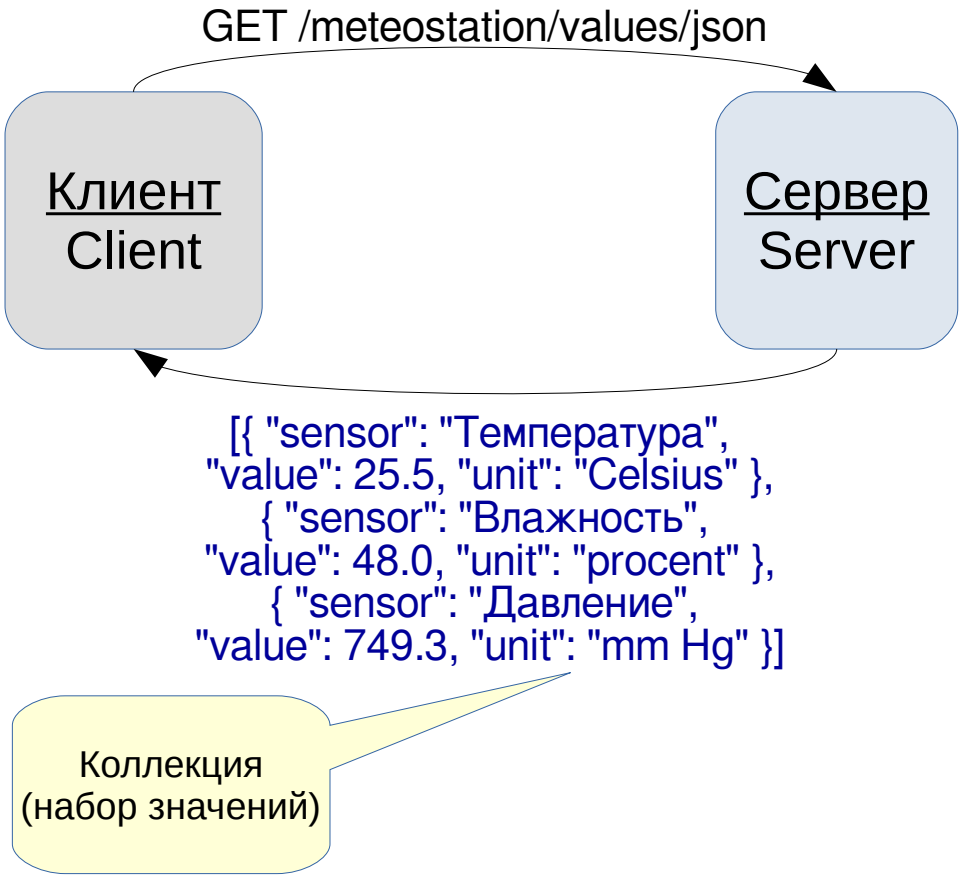
REST: представление данных

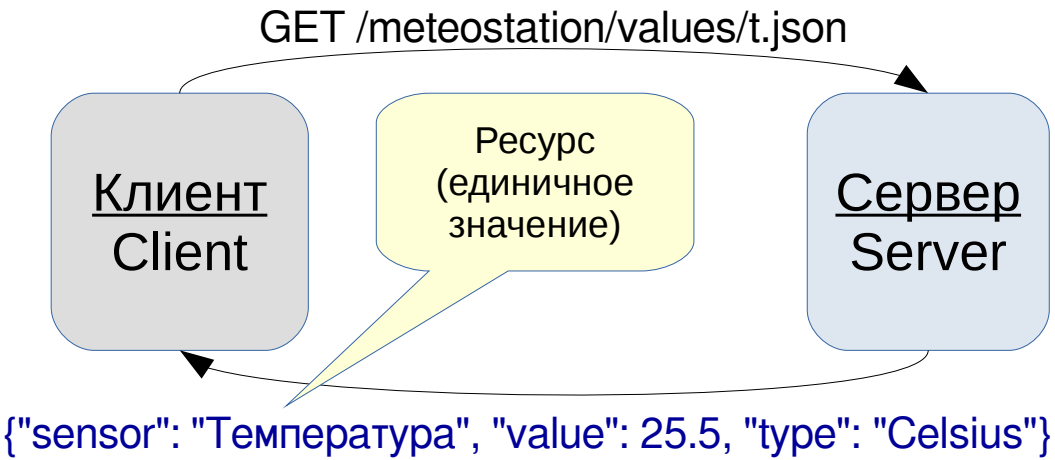
Клиент может запросить тот же самый ресурс в другом представлении (JSON).

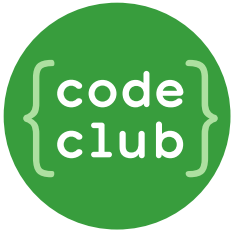
Сервер может преобразовать ресурс из неизвестной для клиента формы хранения (реляционная БД) в запрошенное представление.



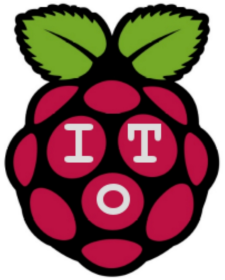
REST: примеры запросов





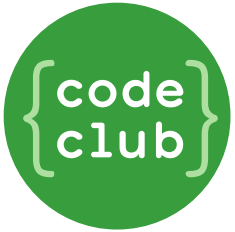


Ruby: работа с HTTP

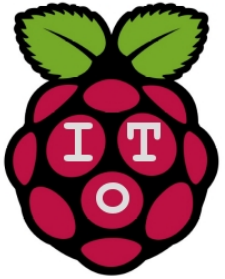


Класс **URI** применяется для манипуляциями с адресом, а класс **Net::HTTP** — для создания запросов, отправки их на сервер и получения ответов.

```
require "net/https"
require "json"

Host = "192.168.43.141"

uri = URI.parse("http://#{host}/json")
http = Net::HTTP.new(uri.host, uri.port)

get_request = Net::HTTP::Get.new(uri.request_uri)
response = http.request(get_request)
text = response.body

hash1 = JSON.parse(text)
p hash1["Sensors"][0]["Pressure"]
```



REST



Где можно применить **REST**?