

# API

---

## «REST API»

### Node.js - Серверный Javascript

- Application Programming Interface - интерфейс программирования приложений) — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением.

Web-API — используется в веб-разработке, как правило, определенный набор HTTP- запросов, а также определение структуры HTTP-ответов, для выражения которых используют XML или JSON форматы.

# REST

---

- (REpresentational State Transfer) - это архитектура, т.е. принципы построения распределенных гипермедиа систем, того что другими словами называется World Wide Web, включая универсальные способы обработки и передачи состояний ресурсов по HTTP



# Автор идеи

---

**Рой Филдинг**

[vhttps://en.wikipedia.org/wiki/Roy\\_Fielding](https://en.wikipedia.org/wiki/Roy_Fielding)

[vhttps://www.linkedin.com/in/royfielding/](https://www.linkedin.com/in/royfielding/) v

<http://roy.gbiv.com/>

v[Architectural Styles and the Design of Network-based Software Architectures](#) (2000)

# Преимущества

---

- ★ Надёжность (за счёт отсутствия необходимости сохранять информацию о состоянии клиента, которая может быть утеряна);
- ★ Производительность (за счёт использования кэша);
- ★ Масштабируемость;
- ★ Прозрачность системы взаимодействия (особенно необходимая для приложений обслуживания сети);
- ★ Простота интерфейсов;
- ★ Портативность компонентов;
- ★ Лёгкость внесения изменений;
- ★ Способность эволюционировать, приспосабливаясь к новым требованиям (на примере Всемирной паутины).

# Когда использовать REST?

---

- ★ Когда есть ограничение пропускной способности соединения
- ★ Если необходимо кэшировать запросы
- ★ Если система предполагает значительное масштабирование В
- ★ сервисах, использующих А JX

# Критерии REST

---

- ★ Клиент-сервер (Client-Server) Отсутствие
- ★ состояний (Stateless) Кеширование
- ★ ответа (Cacheable) Единый интерфейс
- ★ (Uniform Interface)
- ★ Многоуровневая система (Layered
- ★ System)
  - "Код по требованию" (Code on Demand
  - опционально)

# Идемпотентность

---

С точки зрения RESTful-сервиса, операция (или вызов сервиса) идемпотентна тогда, когда клиенты могут делать один и тот же вызов неоднократно при одном и том же результате. Другими словами, создание большого количества идентичных запросов имеет такой же эффект, как и один запрос. Заметьте, что в то время, как идемпотентные операции производят один и тот же результат на сервере (побочные эффекты), ответ сам по себе может не быть тем же самым (например, состояние ресурса может измениться между запросами).

# HTTP методы. GET

---

используется для получения (или чтения) представления ресурса. В случае “удачного” (или не содержащего ошибок) адреса, GET возвращается представление ресурса в формате XML или JSON в сочетании с кодом состояния HTTP 200 (OK). В случае наличия ошибок обычно возвращается код 404 (NOT FOUND) или 400 (BAD REQUEST).

Безопасный метод. Это означает, что он предназначен только для получения информации и не должен изменять состояние сервера или иметь побочных эффектов.



# HTTP методы. POST

---

запрос наиболее часто используется для создания новых ресурсов. На практике он также используется для создания вложенных ресурсов. Другими словами, при создании нового ресурса, POST запрос отправляется к родительскому ресурсу и, таким образом, сервис берет на себя ответственность на установление связи создаваемого ресурса с родительским ресурсом, назначение новому ресурсу ID и т.п. При успешном создании ресурса возвращается HTTP код 201, а также в заголовке `Location` передается адрес созданного ресурса. POST не является безопасным или идемпотентным запросом.

# HTTP методы. PUT

---

обычно используется для предоставления возможности обновления ресурса. Тело запроса при отправке PUT-запроса к существующему ресурсу URI должно содержать обновленные данные оригинального ресурса (полностью, или только обновляемую часть).

Для создания новых экземпляров ресурса предпочтительнее использование POST запроса. В данном случае, при создании экземпляра будет предоставлен корректный идентификатор экземпляра ресурса в возвращенных данных об экземпляре.

При успешном обновлении возвращается код 200 (или 204 если не был передан какой либо контент в теле ответа).

PUT не безопасная операция, так как в следствии ее выполнения происходит модификация (или создание) экземпляров ресурса на стороне сервера, но этот метод идемпотентен. Другими словами, создание или обновление ресурса посредством отправки PUT запроса - ресурс не исчезнет, будет располагаться там же, где и был

# HTTP методы. DELETE

---

используется для удаления ресурса, идентифицированного конкретным URI (ID).

При успешном удалении возвращается 200 (OK) код HTTP, совместно с телом ответа, содержащим данные удаленного ресурса. Также возможно использование HTTP кода 204 (NO CONTENT) без тела ответа.

Согласно спецификации HTTP, DELETE запрос идемпотентен. Если вы выполняете DELETE запрос к ресурсу, он удаляется. Повторный DELETE запрос к ресурсу закончится также: ресурс удален.

# Коды состояний HTTP

[\(полный список\)](#)

## 1xx: Information

★ 100: Continue

## 2xx: Success

★ 200: OK

★ 201: Created

★ 202: Accepted

★ 204: No Content

## 3xx: Redirect

★ 301: Moved Permanently

★ 307: Temporary Redirect

## 4xx: Client Error

★ 400: Bad Request

★ 401: Unauthorized

★ 403: Forbidden

★ 404: Not Found

## 5xx: Server Error

★ 500: Internal Server Error

★ 501: Not Implemented

★ 502: Bad Gateway

★ 503: Service Unavailable

★ 504: Gateway Timeout

# Формирование URI

---

## Это не REST

- ★ /api/users/do\_some
- ★ /api/users/13?action=delete
- ★ /api/users/13?delete=1
- ★ /api/users/13/remove
- ★ /api/getUsers
- ★ /api/v1/users.get

# Лучшие практики

---

- ★ Использовать для описания базовых URL существительные во множественном числе
- ★ Использование более конкретных, четких имен а не абстрактных ( `/news`, `/videos` , а не `/items` )
- ★ Сложную логику описывать за счет дополнительных параметров, т.е. прятать за знаком «?»
- ★ Пример пагинации ( `/users?limit=25&offset=50` )
- ★ Фильтрация ответа (`/friends?fields=id,name,picture`)
- ★ Поддержка нескольких форматов (json, xml)

# Примеры URI ресурсов

---

Добавление нового клиента в систему:

★ POST <http://www.example.com/customers>

Получить данные клиента с идентификатором клиента №33245:

★ GET <http://www.example.com/customers/33245>

*Тот же URI будет использоваться для PUT и DELETE для обновления и удаления соответственно.*

создание нового продукта:

★ POST <http://www.example.com/products>

для чтения, обновления, удаления продукта 66432, соответственно:

★ GET | PUT | УДАЛИТЬ <http://www.example.com/products/66432>

# Примеры URI ресурсов

---

создания нового заказа для клиента:

★ **POST** <http://www.example.com/orders>  
*но это, возможно, вне контекста клиента*

**POST** <http://www.example.com/customers/33245/orders>

*для конкретного клиента*

список заказов, созданных или принадлежащих клиенту # 33245:

★ **GET** <http://www.example.com/customers/33245/orders>

усложним, добавить позицию для заказа №8769 (которая для клиента №33245):

★ **POST** <http://www.example.com/customers/33245/orders/8769/lineitems>



# Примеры URI ресурсов

---

получение списка заказа по номеру без знания номера конкретного клиента.:

★ GET <http://www.example.com/orders/8769>

*пройти один уровень глубже в иерархии, вернуть только первую позицию в том же порядке*

GET <http://www.example.com/customers/33245/orders/8769/lineitems/1>

# Примеры URI ресурсов

---

★ Ограничение с помощью заголовка диапазона:

*Когда выполняется запрос для целого ряда элементов с использованием HTTP-заголовка вместо параметров строки запроса*

**Range: items=0-24**

★ Ограничение по параметрам Query-String:

*offset - это начальный номер позиции (соответствует первой цифре в строке элементов для заголовка Range), а limit - максимальное количество возвращаемых элементов*

**GET <http://api.example.com/resources?offset=0&limit=25>**

★ При этом сервер должен отвечать заголовком Content-Range, чтобы указать, сколько элементов возвращается и сколько еще не отображено

**Content-Range: items 0-24/66**

# Примеры URI ресурсов

---

## ★ Ограничение с помощью заголовка диапазона:

*Когда выполняется запрос для следующего ряда элементов с использованием HTTP-заголовка вместо параметров строки запроса*

**Range: items=15-27**

## ★ Передача параметров Query-String:

*offset - это начальный номер позиции (соответствует первой цифре в строке элементов для заголовка Range), а limit - максимальное количество возвращаемых элементов*

**GET <http://api.example.com/resources?offset=0&limit=25>**

## ★ При этом сервер должен отвечать заголовком Content-Range, чтобы указать, сколько элементов возвращается и сколько еще не отображено

**Content-Range: items 0-24/66 (\*)** *если общее количество элементов на данный момент не известно*

# Примеры URI ресурсов

---

## Сортировка и фильтрация отключение/ версионирование

### ★ фильтрация:

GET [http://www.example.com/users?filter="name::todd|city::denver|title::grand poobah"](http://www.example.com/users?filter=name::todd|city::denver|title::grand poobah)

*Разделитель двойной двоеточия («::») отделяет имя свойства от значения сравнения*

### ★ сортировка:

GET [http://www.example.com/users?sort=last\\_name|first\\_name|-hire\\_date](http://www.example.com/users?sort=last_name|first_name|-hire_date)

*Для каждого свойства - сортировка имени в порядке возрастания, и для каждого свойства, префиксного типа («-») сортировки в порядке убывания. Отделите каждого имени свойства вертикальной полосой («/»).*

### ★ Версионирование/отключение ресурса:

**Content-Type: application/json; version=1 (406 NOT ACCEPTABLE)**

**Deprecated: true (200 OK)**

# Обработка ошибок

---

- ★ Не нужно помещать все ошибки логики/сервера в статус код HTTP
- ★ Описание ошибки должно быть четким и лаконичным
- ★ Отдавайте корректный HTTP статус код (RFC 2616) Всегда
- ★ отдавайте ответ в запрошенном формате
- ★ Если статус код позволяет отдавать тело сообщения — необходимо его расширить за счет краткого описания проблемы.

**ВСЕМ СПАСИБО ЗА  
ВНИМАНИЕ!**

---

**Добавляйтесь в друзья**

Иващенко Андрей

[https://twitter.com/andr\\_ivas](https://twitter.com/andr_ivas)

<https://vk.com/andr.ivas>