REST-URL

URL *(Uniform Resource Locator — унифицированный указатель ресурса) — это то, что отображается в строке браузера, когда вы заходите на тот или иной сайт.*

Вот несколько примеров URL:

```
https://developer.mozilla.org
https://developer.mozilla.org/ru/docs/Learn/
https://developer.mozilla.org/ru/search?q=URL
```

Каждый из этих URLs могут быть напечатаны в адресной строке браузера, чтобы заставить его загрузить связанную страницу (ресурс).

Структура URL адреса

URL состоит из различных частей, некоторые из которых являются обязательными, а некоторые - факультативными. Рассмотрим наиболее важные части на примере:

```
http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDo
cument
```

`http://`-**это протокол**. Он отображает, какой протокол браузер должен использовать. Обычно это HTTP-протокол или его безопасная версия - HTTPS. Интернет требует эти 2 протокола, но браузеры часто могут использовать и другие протоколы, например `mailto:`
(чтобы открыть почтовый клиент) или `ftp:`
для запуска передачи файлов, так что не стоит удивляться, если вы вдруг увидите другие протоколы.

`www.example.com`- **это доменное имя**. Оно означает, какой веб-сервер должен быть запрошен. В качестве альтернативы может быть использован и IP-адрес, но это делается редко, поскольку запоминать IP сложнее, и это не популярно в интернете.

`:80` -**это порт**. Он отображает технический параметр, используемый для доступа к ресурсам на веб-сервере. Обычно подразумевается, что веб-сервер использует стандартные порты HTTP-протокола (80 для HTTP и 443 для HTTPS) для доступа к своим ресурсам. В любом случае, порт - это факультативная составная часть URL.

`/path/to/myfile.html`-**это адрес ресурса на веб-сервере**. В прошлом, адрес отображал местоположение реального файла в реальной директории на веб-сервере. В наши дни это чаще всего абстракция, позволяющая обрабатывать адреса и отображать тот или иной контент из баз данных.

`?key1=value1&key2=value2` - **это дополнительные параметры**, которые браузер сообщает веб-серверу. Эти параметры - список пар ключ/значение, которые разделены символом `&`
. Веб-сервер может использовать эти параметры для исполнения дополнительных команд перед тем как отдать ресурс. Каждый веб-сервер имеет свои собственные правила обработки этих параметров и узнать их можно, только спросив владельца сервера.

`#SomewhereInTheDocument`-**это якорь на другую часть того же самого ресурса.** Якорь представляет собой вид "закладки" внутри ресурса, которая переадресовывает браузер на "заложенную" часть ресурса. В HTML-документе, например, браузер может переместиться в точку, где установлен якорь; в видео- или аудио-документе браузер может перейти к времени, на которое ссылается якорь. Важно отметить, что часть URL после #, которая также известна как идентификатор фрагмента, никогда не посылается на сервер вместе с запросом.

Виды URL

URL-адреса веб-страниц бывают статические и динамические.

Статический URL представляет собой постоянный адрес, он остается неизменным на протяжении всего времени, пока владелец сайта сам не внесет

в него изменения. Такие адреса не содержат дополнительных параметров. Пример: https://timeweb.com/ru/community/articles/v-chem-sostoit-raznica-mezhdu-domenom-hostingom-i-saytom.

Динамические адреса веб-страниц генерируются в ответ на запросы пользователей и содержат разделители «?», «=», «&», после которых отображаются дополнительные параметры страницы. Такие адреса формируются, к примеру, когда пользователь применяет фильтр, производит сортировку товаров в интернет-магазине, использует поиск по сайту.

URI, URL, URN. Что это, чем отличаются

URI (Uniform Resource Identifier) – это строка символов, которая используется для идентификации какого-либо ресурса по его адресу или по его имени, либо по тому и тому вместе.

URL (Uniform Resource Locator) – это строка символов, которая используется для идентификации какого-либо ресурса, но только по его адресу, по его местоположению.

URN (Uniform Resource Name) – это строка символов, которая используется для идентификации какого-либо ресурса, но только по его имени.

Из вышеперчисленных определений можно сделать вывод, что **URI содержит как URL, так и URN**, которые идентифицируют местоположение и имя ресурса соответственно. URI также может быть просто URL-адресом или просто URN, что позволяет ему идентифицировать только имя или местоположение ресурса.

Что такое REST

Representational State Transfer — это архитектурный стиль взаимодействия компонентов распределённого приложения в сети, способ создания API с помощью протокола HTTP. Технология позволяет получать и модифицировать

данные и состояния удаленных приложений, передавая HTTP-вызовы через интернет или любую другую сеть.

Архитектурный стиль – это набор согласованных ограничений и принципов проектирования, позволяющий добиться определённых свойств системы.

Архитектура: REST - требования

Существует шесть обязательных ограничений для построения распределенных **REST-приложений**

Выполнение этих ограничений обязательно для **REST-системы**. Накладываемые ограничения определяют работу сервера в том, как он может обрабатывать и отвечать на запросы клиентов. Действуя в рамках этих ограничений, система приобретает такие желательные свойства как производительность, масштабируемость, простота, способность к изменениях, переносимость, обслуживаемость и надежность.

Если сервис-приложение нарушает любое из этих ограниченных условий, данную систему нельзя считать **REST-системой**

1. Модель клиент-сервер

Первым ограничением применимым к гибридной модели, является приведение архитектуры к модели клиент-сервер. Разграничение потребностей является принципом, лежащим в основе данного накладываемого ограничения. Отделение потребности интерфейса клиента от потребностей сервера (хранившего данные) повышает переносимость кода клиентского интерфейса на другие платформы, а упрощение серверной части улучшает масштабируемость.

2. Отсутствие состояния

Протокол взаимодействия между клиентов и сервером требует соблюдения следующего условия, в период между запросами клиента ни какая информация о состоянии клиента на сервере не храниться (протокол без сохранения состояния). Все запросы от клиента должные быть составлены так, чтобы сервер получил всю необходимую информации для выполнения запроса. Состоянии сессии при этом сохраняется на стороне клиента. Информация о состоянии сессии может быть передана на сервер какому-либо другому сервису для поддерживания устойчивого состояния (на пример на период

установления аутентификации). Клиент инициирует отправку запросов, когда он готов (возникает необходимость) перейти в новое состояние.

3. Кеширование

Как и во всемирной паутине, клиенты а также промежуточные узлы, могут выполнять кеширование ответов сервера. Ответы сервера в свою очередь, должны иметь явно или не явное обозначение как кэшируемые или некэшируемые с целью предотвращения получения клиентами устаревших или неверных данных в ответах на последующие запросы. Правильное использование кеширования способно частично или полностью устранить некоторые клиент-серверные взаимодействия, еще больше повышая производительность и масштабируемость.

4. Единообразие интерфейса

Наличии унифицированного интерфейса является фундаментальным требованием дизайна REST-сервисов. Унифицированные интерфейсы позволяют каждому из сервисов развиваться независимо.

Предъявляться следующие четыре ограничительных условия:

- **Идентификация ресурсов** - все ресурсы идентифицируются в запросах, на пример с использованием URI в интернет системах. Ресурсы концептуально отделены от представлений, который возвращают клиентам. Например, сервер может отсылать данные из базы данных в виде HTML, XML, JSON, но не один и которых не является типом хранения внутри сервера.
- **Манипуляция ресурсами через представление** - если клиент хранит представление ресурса, включая метаданные - он обдает достаточной информацией для модификации или удаления ресурса
- **Самоописываемые сообщения -** каждое сообщение содержит достаточно информации, чтобы понять, каким образом его обрабатывать. К примеру обработчик сообщения (parser) необходимый для извлечения данных может быть указана в списке MIME-типов
- **Гипермедиа как средство изменения состояние приложения** - Клиенты изменяют состояние системы только через действия, которые динамические определены в гипермедиа на сервере. Исключая простые точки входа в приложения, клиент не может предположить что для доступа какая-то операция над каким-то ресурсом, если не получил информации об этом в предыдущих запроса к серверу.

5. Слои

Клиент обычно не способен точно определить, взаимодействует он напрямую с сервером или же с промежуточным узлом, в связи с иерархической структурой сетей (подразумевая, что такая структура образует слои). Применение промежуточных серверов способно повысить масштабируемость за счет балансировки нагрузки и распределенного кеширования. Промежуточные узлы также могут подчиняться политики безопасности с целью обеспечения конфиденциальности информации.

6. Код по требования (необязательное ограничение)

REST может позволить расширить функциональность клиента за счет загрузки кода с сервера в виде апплетов или скриптов.

HTTP Request-Response Structure

HTTP (Hyper Text Transfer Protocol)- это основа для передачи данных в Интернете. Передача данных начинается с запроса, отправленного от клиента, и заканчивается ответом, полученным от веб-сервера.

- **Запросы (HTTP Requests)** — сообщения, которые отправляются клиентом на сервер, чтобы вызвать выполнение некоторых действий. Зачастую для получения доступа к определенному ресурсу. Основой запроса является HTTP-заголовок.
- **Ответы (HTTP Responses)** — сообщения, которые сервер отправляет *в ответ* на клиентский запрос.

Структура HTTP-запросов и ответов

Структура HTTP-запросов:

1. Стартовая строка

Стартовая строка состоит из трёх элементов:

1. **Метод HTTP**, глагол (например, `GET`, `PUT` или `POST`) или существительное (например, `HEAD` или `OPTIONS`), описывающие требуемое действие.

Например, `GET` указывает, что нужно доставить некоторый ресурс, а `POST` означает отправку данных на сервер (для создания или модификации ресурса, или генерации возвращаемого документа).

2. **Цель запроса**, обычно URL, или абсолютный путь протокола, порт и домен обычно характеризуются контекстом запроса. Формат цели запроса зависит от используемого HTTP-метода. Это может быть

   - Абсолютный путь, за которым следует `'?'` и строка запроса. Это самая распространённая форма, называемая *исходной формой* (*origin form*) . Используется с методами `GET`, `POST`, `HEAD`, и `OPTIONS`. `POST / HTTP 1.1` `GET /background.png HTTP/1.0` `HEAD /test.html?query=alibaba HTTP/1.1` `OPTIONS /anypage.html HTTP/1.0`

   - Полный URL - *абсолютная форма* (*absolute form*) , обычно используется с `GET` при подключении к прокси. `GET http://developer.mozilla.org/ru/docs/Web/HTTP/Messages HTTP/1.1`

   - Компонента URL "authority", состоящая из имени домена и (необязательно) порта (предваряемого символом `':'`), *называется authority form*. Используется только с методом `CONNECT` при установке туннеля HTTP. `CONNECT developer.mozilla.org:80 HTTP/1.1`

   - Форма звёздочки (*asterisk form)*, просто "звёздочка" (`'*'`) используется `с методом OPTIONS` и представляет сервер. `OPTIONS * HTTP/1.1`

3. **Версия HTTP**, определяющая структуру оставшегося сообщения, указывая, какую версию предполагается использовать для ответа.

2. Заголовки

Заголовки запросов HTTP имеют стандартную для заголовка HTTP структуру: не зависящая от регистра строка, завершаемая (`':'`) и значение, структура которого определяется заголовком. Весь заголовок, включая значение, представляет собой одну строку, которая может быть довольно длинной.

Существует множество заголовков запроса. Их можно разделить на несколько групп:

- *Основные заголовки (General headers*), например, Via (en- US), относящиеся к сообщению в целом

- *Заголовки запроса (Request headers*), например, `User-Agent`, `Accept-Type`, уточняющие запрос (как, например, `Accept-Language`), придающие контекст (как `Referer`), или накладывающие ограничения на условия (like `If-None`).
- *Заголовки сущности*, например `Content-Length`, относящиеся к телу сообщения. Как легко понять, они отсутствуют, если у запроса нет тела.

3. Тело

Последней частью запроса является его тело. Оно бывает не у всех запросов: запросы, собирающие (fetching) ресурсы, такие как `GET`, `HEAD`, `DELETE`, или `OPTIONS`, в нем обычно не нуждаются. Но некоторые запросы отправляют на сервер данные для обновления, как это часто бывает с запросами `POST` (содержащими данные HTML-форм).

Тела можно грубо разделить на две категории:

- Одноресурсные тела (Single-resource bodies), состоящие из одного отдельного файла, определяемого двумя заголовками: `Content-Type` и `Content-Length`.
- Многоресурсные тела (Multiple-resouce bodies), состоящие из множества частей, каждая из которых содержит свой бит информации. Они обычно связаны с HTML-формами.

Структура HTTP-ответов:

1. Строка статуса

Стартовая строка ответа HTTP, называемая строкой статуса, содержит следующую информацию:

1. *Версию протокола*, `обычно HTTP/1.1`.
2. *Код состояния (status code)*, показывающая, был ли запрос успешным. Примеры: `200`, `404` или `302`
3. *Пояснение (status text*). Краткое текстовое описание кода состояния, помогающее пользователю понять сообщение HTTP.

Пример строки статуса: `HTTP/1.1 404 Not Found.`

2. Заголовки

Заголовки ответов HTTP имеют ту же структуру, что и все остальные заголовки: не зависящая от регистра строка, завершаемая двоеточием (`':'`) и значение, структура которого определяется типом заголовка. Весь заголовок, включая значение, представляет собой одну строку.

Существует множество заголовков ответов. Их можно разделить на несколько групп:

- *Основные заголовки (General headers*), например, Via (en- US), относящиеся к сообщению в целом.
- *Заголовки ответа (Response headers*), например, `Vary` и `Accept-Ranges`, сообщающие дополнительную информацию о сервере, которая не уместилась в строку состояния.
- *Заголовки сущности (Entity headers*), например, `Content-Length`, относящиеся к телу ответа. Отсутствуют, если у запроса нет тела.

3. Тело

Последней частью ответа является его тело. Оно есть не у всех ответов: у ответов с кодом состояния, например, `201` или `204`, оно обычно отсутствует.

Тела можно разделить на три категории:

- Одноресурсные тела (Single-resource bodies), состоящие из отдельного файла известной длины, определяемые двумя заголовками: `Content-Type` и `Content-Length`.
- Одноресурсные тела (Single-resource bodies), состоящие из отдельного файла неизвестной длины, разбитого на небольшие части (chunks) с заголовком Transfer-Encoding (en-US), значением которого является `chunked`.
- Многоресурсные тела (Multiple-resouce bodies) состоящие из многокомпонентного тела, каждая часть которого содержит свой сегмент информации. Они относительно редки.