REST-URL

REST

Representational State Transfer — это архитектурный стиль взаимодействия компонентов распределённого приложения в сети. Архитектурный стиль – это набор согласованных ограничений и принципов проектирования, позволяющий добиться определённых свойств системы. Технология REST позволяет получать данные и состояния удаленных приложений, работать с этими данными и модифицировать их.

URL обозначает Uniform Resource Locator.

URL — система унифицированных адресов электронных ресурсов, или единообразный определитель местонахождения ресурса (файла).

Структура URL

Согласно упомянутому стандарту, URL имеет следующую структуру:

<схема>://<логин>:<пароль>@<хост>:<порт>/?<параметры>#<якорь>

Далее приведено краткое описание элементов структуры:

- **Схема** – определяет **протокол** передачи данных и соответствующее обращение к ресурсу. Основные протоколы передачи данных:
  - **HTTP** – это протокол передачи данных. Протокол HTTP используется при пересылке веб-страниц между компьютерами, подключенными к одной сети.

- **HTTPS** – более безопасный вариант протокола HTTP, использующий шифрование. Предназначен для защищенной передачи личных данных пользователей. Использование данного протокола подкрепляется сертификатом SSL (secure sockets layer), который владелец сайта может сгенерировать самостоятельно (самоподписной сертификат, Подробнее о сертификатах читайте в статье Цифровые сертификаты безопасности .
- **FTP** – это протокол передачи файлов со специального файлового сервера на компьютер пользователя. FTP дает возможность абоненту обмениваться двоичными и текстовыми файлами с любым компьютером сети.
- **MAILTO** – протокол для передачи электронных писем. Используется как форма обратной связи. При переходе по ссылке, содержащей данный протокол, откроется почтовый клиент.

- **Логин и пароль** – эти данные в протоколе http, как правило, не указываются. Они определяют права пользователя на доступ к определенной странице сайта.
- **«Собака» (@)** – специальный разделитель. Без логина и пароля его не указывают.
- **Хост** – это доменное имя, иногда IP-адрес ресурса, к которому обращается пользователь. Домены бывают нескольких уровней.

  Домены первого (верхнего) уровня также называют доменной зоной. Например, в доменном имени "safe-surf.ru", .ru – это домен первого уровня. Все домены верхнего уровня можно разделить на две группы:

  Национальные или географические домены. Они определяют принадлежность сайта к той или иной стране или географической территории. Например, домен .ru принадлежит России, .kz - Казахстану, .ua – Украине и пр.

  Домены общего пользования. Они могут устанавливать принадлежность сайта к определенной категории или виду деятельности. Например .com – коммерческие, .info – информационные, .biz – для бизнеса, .org – некоммерческие, .travel – туризм и пр.

  Домен safe-surf.ru является доменным именем второго уровня. Здесь .ru - это домен верхнего (первого) уровня. Собственное имя web-сайта – safe-

surf находится на втором месте от окончания полного имени. Именно поэтому такие домены называются доменами второго уровня.

Второй и все последующие уровни домена должны быть уникальны в группе своего родительского домена. Иначе говоря, в сети Интернет может быть только один домен второго уровня safe-surf в домене верхнего уровня .ru.

Домены третьего уровня регистрируются у организаций или пользователей, владеющих доменами второго уровня. Например, для страницы safe-surf.ru может быть создан домен для форума – forum.safe-surf.ru. Получилось доменное имя третьего уровня, где forum является доменом третьего уровня в зоне safe-surf.ru.

- **Порт** – указывается довольно редко. Например, при обращении по протоколу http к сайту или любому файлу, размещенному на странице, автоматически присваивается одно из значений – 80 или 8080.
- **URL-путь** – путь к файлу или конкретному разделу сайта.
- **Параметры** – запрос с определёнными параметрами, которые передаются на сервер методом GET. Чтобы разделять параметры применяют знак &, например: ?параметр1=значение1&параметр2=значение2
- **Якорь –** это уникальная строка, состоящая из букв и/или цифр. Она ссылается на определённую область открываемого веб-документа. С его помощью можно сделать так, чтобы пользователь попал к определенному месту на странице.

Отличия URI и URL

URI (*Uniform Resource Identifier*) **–** унифицированный идентификатор ресурса.

URL (*Uniform Resource Locator*) **–** унифицированный определитель местонахождения ресурса.

Различия:

- **URL** – это локатор, тогда как **URI** является идентификатором.
- **URL** специфичен для веб-ресурсов и есть универсальный **URI**, который содержит **URL**.
- **URL** должен иметь протокол, такой как http, ftp, tel и т. Д., тогда как **URI** не должен иметь протокол.
- **URI** включает в себя **URL** и URN.

Сходства:

- И **URL**, и **URI** могут использоваться для относительных и абсолютных ссылок.
- И **URL**, и **URI** могут принимать параметры запроса.
- Согласно определению W3C, **URL** и **URI** – это одно и то же, и они могут быть взаимозаменяемыми.

Требования к архитектуре REST

Существует 6 обязательных ограничений для построения распределённых REST-приложений по Филдингу.

Выполнение этих ограничений обязательно для REST-систем. 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 изменениям, переносимость, отслеживаемость и надёжность.

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

Обязательными условиями-ограничениями являются:

1. **Разграничение клиента и сервера**

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, хранящего данные, повышает переносимость кода клиентского интерфейса на другие платформы, а упрощение серверной части улучшает масштабируемость. Наибольшее же влияние на всемирную паутину, пожалуй, имеет само разграничение, которое позволяет отдельным частям развиваться независимо друг от друга, поддерживая потребности в развитии интернета со стороны различных организаций.

2. **Нет сохранения состояния**

Протокол взаимодействия между клиентом и сервером требует соблюдения следующего условия: в период между запросами клиента никакая информация о состоянии клиента на сервере не хранится (Stateless protocol или «протокол без сохранения состояния»). Все запросы от клиента должны быть составлены

так, чтобы сервер получил всю необходимую информацию для выполнения запроса. Состояние сессии при этом сохраняется на стороне клиента. Информация о состоянии сессии может быть передана сервером какому-либо другому сервису (например, в службу базы данных) для поддержания устойчивого состояния, например, на период установления аутентификации. Клиент инициирует отправку запросов, когда он готов (возникает необходимость) перейти в новое состояние.

Во время обработки клиентских запросов считается, что клиент находится в переходном состоянии. Каждое отдельное состояние приложения представлено связями, которые могут быть задействованы при следующем обращении клиента.

3. **Кэширование**

Как и во Всемирной паутине, клиенты, а также промежуточные узлы, могут выполнять кэширование ответов сервера. Ответы сервера, в свою очередь, должны иметь явное или неявное обозначение как кэшируемые или некэшируемые с целью предотвращения получения клиентами устаревших или неверных данных в ответ на последующие запросы. Правильное использование кэширования способно частично или полностью устранить некоторые клиент-серверные взаимодействия, ещё больше повышая производительность и масштабируемость системы.

4. **Единый интерфейс**

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

К унифицированным интерфейсам предъявляются следующие четыре ограничительных условия:

Идентификация ресурсов

Все ресурсы идентифицируются в запросах, например, с использованием URI в интернет-системах. Ресурсы концептуально отделены от представлений, которые возвращаются клиентам. Например, сервер может отсылать данные из базы данных в виде HTML, XML или JSON, ни один из которых не является типом хранения внутри сервера.

Манипуляция ресурсами через представление

Если клиент хранит представление ресурса, включая метаданные — он обладает 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 указан в списке MIME-типов.

Гипермедиа как средство изменения состояния приложения (HATEOAS)

Клиенты изменяют состояние системы только через действия, которые динамически определены в гипермедиа на сервере (к примеру, гиперссылки в гипертексте). Исключая простые точки входа в приложение, клиент не может предположить, что доступна какая-то операция над каким-то ресурсом, если не получил информацию об этом в предыдущих запросах к серверу. Не существует универсального формата для предоставления ссылок между ресурсами, Web Linking (RFC 5988 -> RFC 8288) и JSON Hypermedia API Language Архивная копия от 27 июня 2014 на Wayback Machine являются двумя популярными форматами предоставления ссылок в REST HYPERMEDIA сервисах.

5. **Многоуровневая система (слои)**

Клиент обычно не способен точно определить, взаимодействует ли 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 счёт балансировки нагрузки и распределённого кэширования. Промежуточные узлы также могут подчиняться политике безопасности с целью обеспечения конфиденциальности информации.

6. **Код по запросу (необязательное ограничение)**

REST может позволить расширить функциональность клиента за счёт загрузки кода с сервера в виде апплетов или скриптов. Филдинг утверждает, что дополнительное ограничение позволяет проектировать архитектуру, поддерживающую желаемую функциональность в общем случае, но, возможно, за исключением некоторых контекстов.

Идентификатором в REST является URI

Унифицированный Идентификатор Ресурса — URI

URI — последовательность символов, идентифицирующая физический или абстрактный ресурс, который не обязательно должен быть доступен через сеть Интернет, причем, тип ресурса, к которому будет получен доступ, определяется контекстом и/или механизмом.

Например:

перейдя по http://example.com — мы попадем на http-сервер ресурса идентифицируемого как example.com

в то же время ftp://example.com — приведет нас на ftp-сервер того же самого ресурса

или например http://localhost/ — URI идентифицирующий саму машину откуда производится доступ

В современном интернете, чаще всего используется две разновидности URI — URL и URN.

Основное различие между ними — в задачах:

URL — Uniform Resource Locator, помогает найти какой либо ресурс

URN — Uniform Resource Name, помогает этот ресурс идентифицировать

Упрощая: URL — отвечает на вопрос: «Где и как найти что-то?», URN — отвечает на вопрос: «Как это что-то идентифицировать».

Синтаксис Унифицированных Идентификаторов Ресурсов (URI)

Схема или протокол

http:// это пример протокола (схемы). Тут описывается какой протокол браузер должен использовать. Обычно это HTTP протокол или его безопасная версия - HTTPS. Интернет требует один из этих двух, но браузеры также знают как работать с некоторыми другими, например mailto: (чтобы открыть почтовый клиент) или ftp: для работы с передачей файлов.

Популярные схемы:

Владелец (имя хоста)

www.example.com - это доменное имя, идентификатор ответственного за это пространство имён. Идентифицирует, какой именно Веб-сервер получает запрос. Альтернативно, можно просто использовать IP address, но поскольку это не так удобно, то этот способ используется не часто.

Порт

:80 - это порт сервера. Он идентифицирует технические "ворота", которые нужны для доступа к ресурсу на сервере. Обычно порт не указывается, т.к. существуют общепринятые нормы о стандартных портах для HTTP (80 для HTTP и 443 для HTTPS). В других случаях обязательно нужно указывать.

Путь

/path/to/myfile.html - это путь к ресурсу на Веб-сервере. Изначально путь типа этого указывал на физическое место файла на сервере, но сейчас всё чаще это псевдоним или описание некоторого абстрактного ресурса.

Строка запроса (query string)

?key1=value1&key2=value2 - это дополнительные параметры, предоставляемые Веб-серверу. Это список пар "ключ=значение", разделённых символом & . Веб-сервер может использовать эти параметры как дополнительные инструкции, что именно сделать с ресурсом перед отправкой его пользователю. Каждый Веб-сервер имеет свои правила насчёт параметров, и единственный надёжный способ узнать как конкретный Веб-сервер обрабатывает эти параметры - это спросить того, кто контролирует Веб-сервер.

Фрагмент

#SomewhereInTheDocument - это "якорь" на другую часть ресурса. Якорь представляет собой что-то вроде "закладки" внутри ресурса, давая браузеру указание показать содержимое с определённого места. В HTML-документе, к примеру, браузер будет скроллить к точке где якорь определён, а на аудио/видео-документе браузер попытается перейти на время, указанное в якоре. Важно что часть, начинающаяся с # - никогда не пересылается серверу в запросе.

HTTP Request-Response Structure

В основе HTTP - клиент-серверная структура передачи данных. Клиент формирует запрос (request) и отправляет на сервер; на сервере запрос обрабатывается, формируется ответ (response) и передается клиенту.

Структура HTTP запроса

HTTP запрос состоит из трех основных частей:

1. строка запроса (request line),
2. заголовок (message header) и
3. тело сообщения (entity body). Тело сообщения не является обязательным параметром. Между заголовком и телом есть пустая разделительная строка.

1. В **строке запроса** указывается метод передачи, версия протокола HTTP и URL, к которому должен обратиться сервер. Заголовки содержат тело

сообщения, передаваемые параметры и другие сведения. В теле сообщения могут находиться передаваемые в запросе данные.

HTTP методы

Для того, чтобы указать серверу на то, какое действие мы хотим произвести с ресурсом, используется тип HTTP-запроса, который также называется **HTTP метод**. Существует несколько HTTP методов, которые описывают действия с ресурсами. Наиболее часто используемыми являются GET и POST.

Метод GET

Метод GET запрашивает информацию из указанного источника и не влияет на его содержимое. Запрос доступен для кеширования данных и добавления в закладки. Длина запроса ограничена (макс. длина URL - 2048).

Метод POST

Метод POST используется для отправки данных, что может оказывать влияние на содержимое ресурса. В отличие от метода GET запросы POST не могут быть кешированы, они не остаются в истории браузера и их нельзя добавить в закладки. Запросы POST не ограничиваются в объеме.

Другие методы

2. Заголовки запроса

- **Host** — ВАЖНО. Домен, по которому мы переходим
- **Cookie** — куки, отдаваемые браузером серверу
- **User-Agent** — браузер, которым делаем запрос
- **Accept-Language** и **Accept-Encoding** — принимаемые браузером язык и кодировка
- **Referer** — предыдущая страница
- **Authorization** — реквизиты для базовой авторизации (логин, пароль)

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-resource bodies), состоящие из множества частей, каждая из которых содержит свой бит информации. Они обычно связаны с HTML-формами .

Структура HTTP ответа сервера состоит из:

1. Строки состояния HTTP ответа, в которой сервер указывает версию HTTP протокола и код состояния.
2. Нуля или нескольких полей HTTP заголовка, разделенных между собой символом CRLF.
3. Пустой строки (в этой строке должен быть только символ CRLF), эта строка обозначает окончание полей заголовка.
4. Необязательное тело HTTP сообщения.

Пример:

1.Строка состояния HTTP ответа сервера

Первая строка в HTTP ответе – это строка состояния, иначе Status-Line. Она состоит из версии протокола HTTP, числового кода состояния HTTP сервера и поясняющей фразы. Окончание строки состояния в HTTP ответе является символ CRLF.

Пример строки состояния HTTP ответа сервера:

Да, кстати, строка состояния — это параметр HTTP.

Код состояния HTTP ответа и поясняющая фраза

Вообще, правильно говорить элемент кода состояния или Status-Code – это целочисленный трехразрядный код результата понимания и удовлетворения запроса клиента. Грубо говоря, это число, которое показывает то, как сервер понял запрос клиента.

Статус-коды ответа

По группам

100 — 199 Информационные

200 — 299 Успешные

300 — 399 Редиректы

400 — 499 Клиентские ошибки

500 — 599 Серверные ошибки

2. Поля заголовка HTTP ответа

Поля заголовка HTTP ответа необходимы серверу для того, чтобы передать дополнительную информацию клиенту, которая не может быть помещена в строку состояния. Поля заголовка HTTP ответа помогают клиенту правильно обработать HTTP сообщение сервера. Так же поля заголовка HTTP ответа могут содержать дополнительную информацию о сервере и о дальнейшем доступе к ресурсу, указанному в URI (URI в HTTP).

Заголовки ответа

Cache-Control — сервер говорит браузеру, как кэшировать данные

Content-Type — тип и подтип содержимого, а также кодировка и приложение для открытия содержимого. Примеры:

Content-Type: text/html; charset=UTF-8 — Тип текст, подтип HTML. Кодировка UTF-8

Content-Type: image/gif — Тип картинка, подтип GIF

Content-Type: application/pdf — Тип «открываемый внешним приложением», подтип PDF

Content-Disposition — говорит браузеру, скачивать или открывать документ как веб-страницу

Location — заголовок для редиректа

Set-Cookie — сервер передает браузеру куку

WWW-Authenticate — выводит окно с базовой аутентификацией

Content-Encoding — сервер говорит браузеру, сжата ли страница и в каком виде

Server и **X-Powered-By** — технологии, на которых работает сайт

4. **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-resource bodies), состоящие из многокомпонентного тела, каждая часть которого содержит свой сегмент информации. Они относительно редки.