aws
Свяжитесь с нами | Поддержка | Русский | Мой аккаунт | Войти | Создание аккаунта AWS
Продукты | Решения | Цены | Документация | Обучение | Партнерская сеть | AWS Marketplace | Обслуживание клиентов | События | Другое
Что такое облачные вычисления? / Раздел концепций в сфере облачных вычислений / Сервисы интеграции приложений / Интеграция приложений / Сети и доставка контента

Что такое RESTful API?

Подробнее о сервисах интеграции приложения

Подробнее о бесплатных предложениях машинного обучения
Бесплатные разработка, развертывание и запуск приложений машинного обучения в облаке

Просмотрите сервисы машинного обучения
Быстрое внедрение инноваций благодаря самому универсальному набору сервисов искусственного интеллекта и машинного обучения

Просмотреть курсы по машинному обучению
Начните проходить курс обучения для специалистов по машинному обучению с использованием контента, созданного экспертами AWS

Читать блоги по машинному обучению
Читайте о последних новостях о продуктах AWS по машинному обучению и рекомендациях

Что такое RESTful API?

RESTful API — это интерфейс,используемые двумя компьютерными системами для безопасного обмена информацией через Интернет. Большинство бизнес-приложений должны взаимодействовать с другими внутренними и сторонними приложениями для выполнения различных задач. Например, чтобы генерировать ежемесячные платежные ведомости, ваша внутренняя бухгалтерская система должна обмениваться данными с банковской системой вашего клиента, чтобы автоматизировать выставление счетов и взаимодействовать с внутренним приложением по учету рабочего времени. RESTful API поддерживают такой обмен информацией, поскольку они следуют безопасным, надежным и эффективным стандартам программного взаимодействия.

Что такое API?

Интерфейс прикладного программирования (API) определяет правила, которым необходимо следовать для связи с другими программными системами. Разработчики внедряют или создают API-интерфейсы, чтобы другие приложения могли программно взаимодействовать с их приложениями. Например, приложение с табелем рабочего времени содержит API, который запрашивает полное имя сотрудника и диапазон дат. Получив эту информацию, интерфейс внутренне обрабатывает табель рабочего времени сотрудника и возвращает количество часов, отработанных за указанный период.

Таким образом, сетевой API функционирует как шлюз между клиентами и ресурсами в Интернете.

Клиенты

Клиенты — это пользователи, которые хотят получить доступ к информации в Интернете. Клиентом может быть человек или программная система, использующая API. Например, разработчики могут создавать программы, которые получают доступ к данным о погоде из метеосистемы. Также получить доступ к этим данным можно из браузера, посетив веб-сайт с информацией о погоде.

Ресурсы

Ресурсы — это информация, которую различные приложения предоставляют своим клиентам. Ресурсы могут быть изображениями, видео, текстом, числами или данными любого типа. Компьютер, который предоставляет ресурсы клиенту, также называется сервером. API позволяет организациям совместно использовать ресурсы и предоставляет веб-службы, обеспечивая безопасность, контроль и аутентификацию. Кроме того, API помогает определить, какие клиенты могут получить доступ к определенным внутренним ресурсам.

Что такое REST?

Representational State Transfer (REST) — это программная архитектура, которая определяет условия работы API. Первоначально REST создавалась как руководство для управления взаимодействиями в сложной сети, такой как Интернет. Архитектуру на основе REST можно использовать для поддержки высокопроизводительной и надежной связи в требуемом масштабе. Ее можно легко внедрять и модифицировать, обеспечивая прозрачность и кросс-платформенную переносимость любой системы API.

Разработчики могут создавать API с использованием нескольких архитектур. API-интерфейсы, соответствующие архитектурному стилю REST, называются REST API. Веб-службы, реализующие архитектуру REST, называются веб-службами RESTful. Как правило, термин RESTful API относится к сетевым RESTful API. Однако REST API и RESTful API являются взаимозаменяемыми терминами.

Ниже приведены некоторые принципы архитектурного стиля REST:

Единый интерфейс

Единый интерфейс является конструктивной основой любого веб-сервиса RESTful. Это указывает на то, что сервер передает информацию в стандартном формате. Отформатированный ресурс в REST называется представлением. Этот формат может отличаться от внутреннего представления ресурса в серверном приложении. Например, сервер может хранить данные в виде текста, но отправлять их в формате представления HTML.

Единый интерфейс накладывает четыре архитектурных ограничения:

1. Запросы должны идентифицировать ресурсы. Это происходит за счет единого идентификатора ресурсов.
2. Клиенты имеют достаточно информации в представлении ресурса, чтобы при желании изменить или удалить ресурс. Сервер выполняет это условие, отправляя метаданные, которые дополнительно описывают ресурс.
3. Клиенты получают информацию о дальнейшей обработке представлений. Сервер реализует это, отправляя описательные сообщения, где содержатся метаданные о том, как клиент может использовать их оптимальным образом.
4. Клиенты получают информацию обо всех связанных ресурсах, необходимых для выполнения задачи. Сервер реализует это, отправляя гиперссылки в представлении, чтобы клиенты могли динамически обнаруживать больше ресурсов.

Отсутствие сохранения состояния

В архитектуре REST отсутствие сохранения состояния относится к методу связи, при котором сервер выполняет каждый клиентский запрос независимо от всех предыдущих запросов. Клиенты могут запрашивать ресурсы в любом порядке, и каждый запрос либо изолирован от других запросов, либо его состояние не сохраняется. Это конструктивное ограничение REST API подразумевает, что сервер может каждый раз полностью понять и выполнить запрос.

Многоуровневая система

В многоуровневой системной архитектуре клиент может подключаться к другим авторизованным посредникам между клиентом и сервером и по-прежнему получать ответы от сервера. Серверы также могут передавать запросы другим серверам. Вы можете спроектировать свою веб-службу RESTful для работы на нескольких серверах с несколькими уровнями (безопасностью, приложениями и бизнес-логикой), совместно выполняющих клиентские запросы. Эти уровни остаются невидимыми для клиента.

Емкость кэша

Веб-службы RESTful поддерживают кэширование, то есть процесс сохранения некоторых ответов на клиенте или на посреднике для сокращения времени ответа сервера. Например, вы заходите на веб-сайт с общими изображениями верхнего и нижнего колонтитулов на каждой странице. Каждый раз, когда вы посещаете новую страницу веб-сайта, сервер должен повторно отправлять одни и те же изображения. Чтобы избежать этого, клиент кэширует или сохраняет эти изображения после первого ответа, а затем использует изображения из кэша. Веб-службы RESTful управляют кэшированием с помощью ответов API, которые определяют себя как кэшируемые или некэшируемые.

Код по запросу

В архитектурном стиле REST серверы могут временно расширять или настраивать функциональные возможности клиента, передавая код программного обеспечения. Например, когда вы заполняете регистрационную форму на каком-либо веб-сайте, ваш браузер сразу же выделяет все допущенные ошибки (например, неверные номера телефонов). Это происходит благодаря коду, отправленному сервером.

В чем заключаются преимущества RESTful API?

RESTful API обладает следующими преимуществами:

Возможность масштабирования

Системы, реализующие REST API, могут эффективно масштабироваться благодаря оптимизации взаимодействия между сервером и клиентом по REST. Отсутствие сохранения состояния снимает нагрузку с сервера: серверу не нужно сохранять информацию о предыдущих запросах клиента. Отлаженное кэширование частично или полностью устраняет некоторые взаимодействия между клиентом и сервером. Перечисленные функции предполагают масштабируемость и не ограничивают пропускную способность, что может привести к снижению производительности.

Гибкость

Веб-службы RESTful поддерживают полное разделение клиента и сервера. Они упрощают и разделяют различные серверные компоненты, чтобы каждая часть могла развиваться независимо. Изменения платформы или технологии в серверном приложении не влияют на клиентское приложение. Возможность разделения функций приложения на уровни еще больше повышает гибкость. Например, разработчики могут вносить изменения в уровень базы данных, не переписывая логику приложения.

Независимость

REST API не зависит от используемой технологии. Вы можете создавать как клиентские, так и серверные приложения на разных языках программирования, не затрагивая структуру API. Также можно изменить базовую технологию на любой стороне, не влияя на обмен данными.

Как работает RESTful API?

Базовый принцип работы RESTful API совпадает с принципом работы в Интернете. Клиент связывается с сервером с помощью API, когда ему требуется какой-либо ресурс. Разработчики описывают принцип использования REST API клиентом в документации на API серверного приложения. Ниже представлены основные этапы запроса REST API:

1. Клиент отправляет запрос на сервер. Руководствуясь документацией API, клиент форматирует запрос таким образом, чтобы его понимал сервер.
2. Сервер аутентифицирует клиента и подтверждает, что клиент имеет право сделать этот запрос.
3. Сервер получает запрос и внутренне обрабатывает его.
4. Сервер возвращает ответ клиенту. Ответ содержит информацию, которая сообщает клиенту, был ли запрос успешным. Также запрос включает сведения, запрошенные клиентом.

Сведения о запросе и ответе REST API могут немного различаться в зависимости от того, как разработчики проектируют API.

Что содержит клиентский запрос RESTful API?

API RESTful требует, чтобы запросы содержали следующие основные компоненты:

Уникальный идентификатор ресурса

Сервер присваивает каждому ресурсу уникальный идентификатор ресурса. В случае со службами REST сервер идентифицирует ресурсы с помощью универсального указателя ресурсов (URL). URL указывает путь к ресурсу. URL аналогичен адресу веб-сайта, который вы вводите в браузере для посещения веб-страницы. URL также называется адресом запроса и четко указывает серверу, что требуется клиенту.

Метод

Как правило, разработчики реализуют RESTful API с помощью протокола передачи гипертекста (HTTP). Метод HTTP сообщает серверу, что ему необходимо сделать с ресурсом. Ниже приведены четыре распространенных метода HTTP:

GET

Клиенты используют GET для доступа к ресурсам, расположенным на сервере по указанному URL. Они могут кэшировать запросы GET и отправлять параметры в запросе RESTful API, чтобы сообщить серверу о необходимости фильтровать данные перед отправкой.

POST

Клиенты используют POST для отправки данных на сервер. При этом они включают в запрос представления данных. Отправка одного и того же запроса POST несколько раз имеет побочный эффект — многократное создание одного и того же ресурса.

PUT

Клиенты используют PUT для обновления существующих на сервере ресурсов. В отличие от POST, отправка одного и того же запроса PUT несколько раз дает один и тот же результат в веб-службе RESTful.

DELETE

Клиенты используют запрос DELETE для удаления ресурса. Запрос DELETE может изменить состояние сервера. Однако если у пользователя нет соответствующей аутентификации, запрос завершается ошибкой.

Заголовки HTTP

Заголовки запросов — это метаданные, которыми обмениваются клиент и сервер. Например, заголовок запроса указывает формат запроса и ответа, предоставляет информацию о статусе запроса и т. д.

Данные

Запросы REST API могут включать данные для успешной работы POST, PUT и других методов HTTP.

Параметры

Запросы RESTful API могут включать параметры, которые предоставляют серверу более подробную информацию о необходимых действиях. Ниже приведены некоторые типы параметров:

- Параметры пути, которые определяют детали URL.
- Параметры запроса, которые запрашивают дополнительную информацию о ресурсе.
- Параметры cookie, которые быстро аутентифицируют клиентов.

Что такое методы аутентификации RESTful API?

Веб-служба RESTful должна аутентифицировать запросы для последующей отправки ответа. Аутентификация — это процесс подтверждения личности. Например, для подтверждения личности можно использовать удостоверение личности или водительские права. Точно так же клиенты службы RESTful должны подтвердить свою личность серверу, чтобы установить доверие.

RESTful API поддерживает четыре распространенных метода аутентификации:

HTTP-аутентификация

HTTP определяет некоторые схемы аутентификации, которые можно использовать при реализации REST API. Ниже представлены две такие схемы:

Базовая аутентификация

При базовой аутентификации клиент отправляет имя пользователя и пароль в заголовке запроса. Он кодирует их с помощью метода кодирования base64, который преобразует пару имя пользователя–пароль в набор из 64 символов для безопасной передачи.

Аутентификация носителя

Аутентификация носителя — это процесс предоставления управления доступом носителю токена. Как правило, токен носителя представляет собой зашифрованную строку символов, которую сервер генерирует в ответ на запрос входа в систему. Клиент отправляет токен в заголовках запроса для доступа к ресурсам.

Ключи API

Ключи API — это еще один вариант аутентификации REST API. При таком подходе сервер генерирует уникальное значение и присваивает его первому клиенту. Всякий раз, когда клиент пытается получить доступ к ресурсам, он использует для верификации уникальный ключ API. Ключи API менее надежны: поскольку клиент должен передавать ключ, повышается вероятность его кражи.

OAuth

OAuth сочетает в себе пароли и токены для безопасного входа в любую систему. Сначала сервер запрашивает пароль, а затем дополнительный токен для завершения процесса авторизации. Он может проверять токен в любое время, а также через определенный период времени в соответствии с областью и сроком действия.

Что содержит ответ сервера RESTful API?

Принципы REST требуют, чтобы ответ сервера содержал следующие компоненты:

Строка состояния

Строка состояния содержит трехзначный код состояния, который сообщает об успешном или неудачном выполнении запроса. Например, коды 2XX указывают на успешное выполнение, а коды 4XX и 5XX — на ошибки. Коды 3XX указывают на перенаправление URL.

Ниже приведены некоторые распространенные коды состояния:

- 200: общий ответ об успешном выполнении
- 201: ответ об успешном выполнении метода POST
- 400: неверный запрос, который сервер не может обработать
- 404: ресурс не найден

Текст сообщения

Текст ответа содержит представление ресурса. Сервер выбирает подходящий формат представления на основе содержания заголовков запроса. Клиенты могут запрашивать информацию в форматах XML или JSON: они определяют запись данных в виде обычного текста. Например, если клиент запрашивает имя и возраст человека по имени Джон, сервер возвращает представление JSON в следующем формате:

'{"name":"John", "age":30}'

Заголовки

Ответ также содержит заголовки или метаданные об ответе. Они дают более подробный контекст ответа и включают такую информацию, как название сервера, кодировка, дата и тип контента.

Как AWS может помочь в управлении RESTful API?

Шлюз API Amazon – это полностью управляемый сервис для разработчиков, предназначенный для создания, публикации, обслуживания, мониторинга и обеспечения безопасности API в любых масштабах. API Gateway позволяет создавать RESTful API для приложений двусторонней связи в реальном времени.

С помощью API Gateway можно:

- Предоставить пользователям высокую производительность при запросах и ответах API.
- Разрешить доступ к API с помощью AWS Identity and Access Management (IAM) и Amazon Cognito, которые обеспечивают встроенную поддержку OAuth.
- Запускать несколько версий одного и того же API одновременно с помощью API Gateway, что позволяет быстро дорабатывать, тестировать и запускать новые версии.
- Отслеживать метрики производительности и информацию о запросах API, задержке данных и частоте ошибок из API Gateway.

Начните работу со шлюзом API, воспользовавшись нашим пошаговым руководством и создав аккаунт AWS уже сегодня.

Следующие шаги на AWS

Дополнительные ресурсы по продукту

Подробнее о сервисах интеграции приложений »

Зарегистрировать бесплатный аккаунт

Получите мгновенный доступ к уровню бесплатного пользования AWS.

Зарегистрируйтесь для получения доступа к уровню бесплатного пользования API Gateway »

Начало разработки в консоли

Начните разработку в Консоли управления AWS.

Вход »

Подробнее об AWS
Что такое AWS?
Что такое облачные вычисления?
Инклюзивность, многообразие и равенство AWS
Что такое DevOps?
Что такое контейнер?
Что такое озеро данных?
Безопасность облака AWS
Новые возможности
Блоги
Пресс-релизы

Ресурсы для работы с AWS
Начало работы
Обучение и сертификация
Библиотека решений AWS
Центр архитектуры
Вопросы и ответы по продуктам и техническим темам
Аналитические отчеты
Партнеры AWS

Разработчики на AWS
Центр разработчика
Пакеты SDK и инструментарий
.NET на AWS
Python на AWS
Java на AWS
PHP на AWS
JavaScript на AWS

Поддержка
Свяжитесь с нами
Получение помощи специалиста
Обратиться в службу поддержки
AWS re:Post
Центр знаний
Обзор AWS Support
Юридическая информация
Работа в AWS

Создать аккаунт AWS