Rest API против Web API: Все, что вам нужно знать

28.04.2023

Интерфейсы прикладного программирования, или API, — это способ взаимодействия компьютерных программ или служб друг с другом. Это взаимодействие обычно происходит через конечную точку API, открытую программой, которую потребляет клиент. В этой статье мы сравним два популярных подхода к созданию API: API с репрезентативной передачей состояния (REST) и Web API.

Что такое REST API?

Вопреки распространенному мнению, REST API — это не протокол. Это архитектура, и это самая популярная архитектура для разработки API. Как мы объясняем в статье GraphQL vs REST: Everything You Need To Know, REST не имеет статического характера, поэтому никакие данные или статус не сохраняются между запросами.

REST также определяет несколько архитектурных ограничений для создания приложений, взаимодействующих через HTTP:

- Клиент-серверная архитектура
- Нестационарность
- Единый интерфейс
- Кэшируемость
- Многослойная системная архитектура
- Код по требованию

REST проще в использовании, чем другие протоколы или архитектуры API. Он также обладает множеством других преимуществ, которые делают его первым выбором для многих разработчиков, создающих API:

- **Разнообразные форматы сообщений:** API REST в основном используются с JSON для сериализации данных, но работают с несколькими форматами сообщений, включая JSON, HTTP, обычный текст и XML. Такой набор возможностей дает ему преимущество перед протоколами типа Service Object Access Protocol (SOAP), которые в основном работают с XML по HTTP, при этом такие варианты, как JSON, значительно легче, гибче за счет поддержки массивов и проще в разборе по сравнению с XML.
- **Методы HTTP:** REST обычно используется с любым из методов GET, POST, PATCH, DELETE или PUT для получения данных и выполнения запросов в зависимости от реализации сервиса. Эти методы возвращают обычные для HTTP коды успеха и неудачи. Другие методы включают OPTIONS, HEAD и TRACE. Эти методы непоследовательны среди сервисов, так как некоторые провайдеры могут реализовать только один метод в соответствии со своими потребностями.
- **Развязанная архитектура:** REST имеет архитектуру клиент-сервер, поэтому его логика отделена от представления — можно одновременно работать над несколькими частями без помех.
- **Масштабируемость:** API REST просты, что делает их удобными в использовании. Однако, если вам необходимо увеличить масштаб, вы можете создать новые конечные точки для включения более сложной логики.
- **Возможность кэширования:** Хотя REST не имеет статического характера, ответ сервера на клиенте можно кэшировать, чтобы избежать повторения избыточных запросов. В ответе сервера обычно содержится информация о том, как должно выполняться кэширование — клиент кэширует запросы в течение определенного периода времени.
- **Безопасность:** В большинстве случаев конечные точки REST открываются через конечные точки HTTPS, что гарантирует, что все коммуникации API защищены с помощью TLS/SSL. REST также поддерживает другие схемы авторизации и аутентификации, такие как OAuth2 и JSON Web Tokens (JWT).

Что такое веб-интерфейс API?

Web API — это просто интерфейс для доступа к ресурсам сервера по протоколу HTTP. Этот термин относится к концепции, а не к конкретной технологии — Web API может быть создан с помощью различных технологий, таких как Java и ASP.NET. Web API используют интерфейс с открытым исходным кодом и задействуют множество клиентских устройств, таких как браузеры, смартфоны, планшеты и ноутбуки. Web API реализуют спецификации протоколов с такими понятиями, как кэширование, версионирование и различные форматы контента. Web API может быть или не быть REST API, в зависимости от того, как он построен. Web API обычно используются в распределенных системах для предоставления услуг на различных устройствах, таких как смартфоны и ноутбуки, и ограничиваются клиентской частью веб-приложения.

Вот два примера широко используемых Web API:

- **API Google:** К ним относятся API YouTube, которые позволяют разработчикам встраивать видеоролики YouTube в свои приложения, такие как веб-сайты, и API Google Maps, который позволяет разработчикам использовать или встраивать Google Maps на веб-страницы с помощью интерфейсов JavaScript или Flash.
- **API-интерфейсы Twitter:** К ним относятся API поиска Twitter, который предоставляет методы для взаимодействия с поиском Twitter, и REST API, который позволяет получить доступ к основным данным Twitter.

Web API осуществляется как взаимодействие между системами. Вот как могут передаваться данные в рамках такого API:

1. Клиентское устройство отправляет запрос на веб-сервер.
2. Веб-сервер получает запрос, обрабатывает его, затем отправляет обратно на клиентское устройство для

выполнения.
3. В результате пользователь получает результат.

К преимуществам Web API относятся:

- **Легкая архитектура:** Web API отлично подходят для устройств с ограниченной пропускной способностью, таких как смартфоны.
- **Описательные заголовки сообщений:** Web API имеют описательные заголовки сообщений, которые могут содержать информацию о типе содержимого, схеме безопасности или способе кэширования.
- **Поддержка всех типов данных:** Тело Web API может использоваться для чего угодно, включая двоичные файлы (видео, изображения, документы), простой XML, JSON и HTML.
- **Ресурсно-ориентированный сервис:** Web API может предоставлять ресурсы в соответствии с архитектурой REST.
- **Простая конфигурация и настройка:** Web API легко настраивать и запускать.

Web API против REST API

Теперь давайте сравним эти два API более подробно.

Сходство архитектуры

Веб- и REST API имеют некоторые архитектурные сходства — давайте рассмотрим их.

- **Нестационарность:** HTTP-запросы происходят изолированно и по своей сути являются статичными, поскольку каждый запрос содержит достаточно информации для его выполнения. Несколько запросов связаны друг с другом только посредством общей информации, такой как cookies

или идентификатор сессии. Отсутствие синхронизации состояния снижает сложность и повышает производительность, поскольку серверу не нужно отслеживать запросы клиентов. Одновременные запросы также могут быть масштабированы на несколько серверов.

- **Многоуровневая архитектура:** Они оба поддерживают многоуровневую архитектуру, в которой развертывание API, аутентификация запросов и хранение могут происходить на нескольких серверах.
- **Ресурсно-ориентированная:** В ресурсно-ориентированных архитектурах ресурсы сопоставляются с унифицированными идентификаторами ресурсов (URI). Как Web, так и REST API ориентированы на ресурсы, поскольку они предоставляют ресурсы через URI.
- **Возможность кэширования:** В REST и Web API запросы, которые возвращают одну и ту же информацию при каждом вызове, кэшируются. Например, вызов OPTION на конечной точке будет кэшироваться, поскольку результат будет одинаковым независимо от того, сколько раз он будет вызван. Это свойство, известное как идемпотентность, является хорошей основой для определения того, когда данные можно кэшировать. Идемпотентность всегда учитывается в REST, хотя и не так часто в Web API. Идемпотентный вызов API — это такой вызов, результаты которого никогда не изменятся — независимо от того, сколько раз он будет вызван — даже если на сервере что-то изменится. Примерами идемпотентных методов являются GET, HEAD и OPTIONS.

Различия в архитектуре

Хотя Web API и REST API имеют схожие архитектурные модели, у них также есть некоторые ключевые различия.

- **Координация на стороне клиента и сервера:** REST API имеют слабо связанную архитектуру, что позволяет вести

независимую разработку на стороне клиента и сервера. В Web API изменения между клиентом и сервером координируются более тонко.
- **Интерфейс:** В зависимости от деталей реализации, REST API, как правило, используют стандартные интерфейсы. Web API используют пользовательские интерфейсы, в зависимости от поставщика API.

Связь

Web API достаточно гибкие, чтобы использовать любой стиль коммуникации, в то время как REST API в основном используются с JSON, XML и обычным текстом. Эти параметры означают, что REST API хорошо подходят для передачи текстовых данных, таких как операции создания, чтения, обновления и удаления (CRUD) базы данных, но более ограничены, когда речь идет о двоичных данных. Web API предлагают гораздо лучшие возможности для сервисов, требующих двоичных данных — например, сервисов потоковой передачи музыки или видео — поскольку они поддерживают больше форматов сообщений.

Примеры использования

Хотя во многих случаях эти форматы API взаимозаменяемы, есть несколько сценариев, в которых один лучше другого:

- **Облачные сервисы и приложения:** Благодаря своей природе REST API используются в облачных сервисах, поскольку компоненты без статических данных могут масштабироваться и перераспределяться в соответствии с изменениями. Облачные сервисы и метрики обычно лучше всего представлять в виде REST API, поскольку в этом случае нет необходимости в пользовательском коде.
- **Потоковые сервисы:** Веб-интерфейсы API имеют лучшую поддержку и низкие накладные расходы при использовании двоичных данных на устройствах с ограниченной памятью

или пропускной способностью, поэтому они лучше всего подходят для сервисов, требующих потоковой передачи данных.

- **Работа с базами данных (CRUD):** Функциональность CRUD проще и легче реализовать через REST API, чем через Web API.

REST API трудно управлять сложными запросами, которые требуют доступа к ресурсам, не расположенным в простой иерархии. Это связано с тем, что его URI ссылаются на ресурсы, то есть управление такой ситуацией включает в себя манипуляции с путями URI, параметрами запроса и телом запроса, что противоречит цели REST. В этом случае предпочтительнее использовать Web API API, поскольку он позволяет настраивать систему и имеет широкую поддержку URI-ответов и заголовков запросов.

Заключение

Web API и REST API используются для создания приложений, которые предоставляют ресурсы и обмениваются данными по протоколу HTTP. В то время как REST описывает архитектурные ограничения над унифицированным интерфейсом, веб-интерфейсы API в целом представляют собой концепцию, которая может быть RESTful, в зависимости от реализации. Как Web API, так и REST API являются легкими форматами, которые взаимозаменяемы во многих ситуациях. Однако, по сравнению с REST API, Web API обеспечивают более настраиваемый опыт и поддержку большего количества типов сообщений, а также поддерживают сложные взаимодействия между серверами и клиентами, работающими с двоичными данными.