40 часто задаваемых вопросов и ответов по REST API

23.01.2024

API (Application Programming Interface) расшифровывается как *интерфейс прикладного программирования*. Он служит шлюзом для доступа приложений к некоторым ресурсам других приложений. Преимущество использования API заключается в предоставлении доступа сторонним приложениям, чтобы они не могли получить доступ ко всем данным вашего приложения. Они могут получить доступ только к тем данным, которые вы предоставляете через свой API. Приложение или пользователь, который хочет получить доступ к данным, называется клиентом, а приложение, которое предоставляет данные, — *сервером*.

Сегодня API широко используются в любой архитектуре программного обеспечения. Давайте в этой статье рассмотрим некоторые из наиболее часто возникающих вопросов о REST API. Этот список поможет нам лучше понять как устроена эта технология. Итак начнём!

Что такое REST?

Ответ: REST — это архитектурный дизайн, который определяет некоторые ограничения на работу API. API, которые следуют принципам REST, известны как RESTful API. REST расшифровывается как *Representational State Transfer* (передача данных о состоянии представления).

Это не протокол и не стандарт, а именно архитектура, которая может быть использована для реализации API различными способами. Она обеспечивает высокую гибкость и свободу для разработчиков, поэтому широко используется для разработки API. Вот некоторые принципы архитектуры REST:

- **Separation — Разделение клиента и сервера**: В RESTful API клиент не должен влиять на сервер никаким другим образом, кроме запроса данных через URI (Uniform Resource Identifier). Точно так же сервер не должен каким-либо образом изменять содержимое клиента.
- **Statelessness — Нестационарность**: Когда выполняются два отдельных запроса, они не знают друг о друге. Другими словами, запросы не имеют состояния и не поддерживают его. Если запрос выполнен, он просто завершается. Каждый запрос изолирован от других запросов.
- **Layered Architecture — Многоуровневая архитектура**: Клиент или сервер не знают, куда направляется запрос — непосредственно к источнику или к приложению-посреднику. Их интересует только ответ на запрос.
- **Caching — Кэширование**: Данные или ответ могут быть кэшированы как на стороне клиента, так и на стороне сервера для повышения производительности и масштабируемости. Если к определенному ресурсу часто поступают запросы, то ответ на этот запрос можно кэшировать и использовать при необходимости.

Каковы некоторые ключевые характеристики REST?

Ответ: Ключевыми характеристиками или особенностями REST являются:

- **Flexibility – Гибкость**: Вы можете переехать с одного сервера на другой, и это ничего не изменит, потому что API будет отправлять тот же ответ на конкретный запрос. Кроме того, вы можете добавить столько конечных точек, сколько захотите, для разных типов данных.
- **Scalability – Масштабируемость**: Кэширование улучшает масштабируемость благодаря тому, что ответы сохраняются для последующего использования. Оно снижает нагрузку на сервер, а также уменьшает задержки.
- **Authorization – Авторизация**: С помощью заголовка авторизации вы можете указать учетные данные, которые сервер может использовать для авторизации запроса.
- **Statelessness – Нестационарность**: Это самая важная особенность REST, поскольку она не позволяет запросам знать, что происходит с другими запросами. Запросы изолированы и завершаются, как только они выполнены.

Что такое ресурсы в архитектуре REST?

Ответ: Ресурсы – это объекты, над которыми выполняются различные операции, такие как извлечение, обновление или удаление. Они являются основными строительными блоками архитектуры REST.

Например, если вы рассматриваете интернет-магазин электронной коммерции, то товары, пользователи, а также метаданные считаются ресурсами, поскольку с ними можно работать. Ресурсы могут быть переданы другому приложению через API.

Укажите некоторые преимущества и недостатки REST API.

Ответ: Преимущества REST API заключаются в следующем:

- Простота реализации.
- Ресурсы могут быть легко обработаны.
- Масштабируемость благодаря архитектуре клиент-сервер.
- Поддерживаются различные типы носителей для передачи данных, такие как XML и JSON.

Недостатки:

- Невозможно поддерживать состояние между запросами.
- Из-за многоуровневой архитектуры невозможно узнать истинное происхождение ресурса.
- Не подходит для сложных запросов и запросов.

Дайте определение шаблона REST.

Ответ: Шаблон REST – это утилита или клиент, с помощью которого вы можете получить доступ к API REST. По сути, он скрывает шаблонный код, который вам придется написать, чтобы запросить ресурс у REST API.

Что такое RESTful?

Ответ: RESTful API или сервисы – это интерфейсы, реализующие архитектурный стиль REST*(Representational State Transfer*) и работающие по таким протоколам, как HTTP.

Что такое RESTful веб-сервисы?

Ответ: Веб-сервисы RESTful созданы для наилучшей работы в

Интернете. Representational State Transfer (REST) – это архитектурный стиль, который определяет ограничения, такие как унифицированный интерфейс, многоуровневая архитектура и отсутствие статичности, если они применяются к веб-сервису, то вызывают желаемые свойства, такие как производительность и масштабируемость, которые позволяют сервисам лучше всего работать в сети.

Как можно тестировать RESTful веб-сервисы?

Ответ: Чтобы протестировать RESTful веб-сервис, вы можете использовать REST-клиент, например Postman или Thunder Client, и запросить веб-сервис, который вы хотите протестировать. Затем, когда вы получите ответ, поймите его; это ключевая часть.

Если вы хотите протестировать сложный API с большим количеством конечных точек, вам, возможно, придется разделить тестирование и выполнить модульное тестирование, интеграционное тестирование, тестирование производительности и сквозное тестирование.

Назовите некоторые особенности RESTful Web Services.

Ответ: Некоторые из ключевых особенностей RESTful веб-сервисов:

- Поддержка нескольких типов носителей, таких как JSON и XML.
- Масштабируемость
- Изоляция клиента и сервера
- Гибкость

Дайте определение корневым классам ресурсов RESTful.

Ответ: Корневые классы ресурсов — это “простые старые объекты” (POJO), которые либо аннотированы @Path, либо имеют хотя бы один метод, аннотированный @Path или обозначением метода запроса, таким как @GET, @POST, @PUT или @DELETE.

Что такое URI?

Ответ: URI расшифровывается как Uniform Resource Identifier (Единый идентификатор ресурса). Это последовательность символов, используемая для определения местоположения или идентификации ресурсов API или сервиса. Он использует имя или местоположение ресурса для его идентификации, но не полагается на определенный метод или технику.

Что такое безстатусность в REST?

Ответ: Statelessness – это ограничение, применяемое к API, при котором любые два запроса не могут знать, что происходит друг с другом. Другими словами, состояние запросов не поддерживается. Если запрос выполнен, он просто завершается после получения ответа.

Что такое JAX-RS?

Ответ: JAX-RS – это Java API, который позволяет разрабатывать приложения на Java, использующие архитектуру REST. Этот API упрощает разработку REST-приложений на Java.

Что такое ключевые аннотации в API JAX-RS?

Ответ: Аннотации в JAX-RS используются разработчиками для украшения Java-классов с целью определения ресурсов и методов, которые могут быть выполнены над этими ресурсами. К ключевым аннотациям JAX-RS API относятся:

- @GET: Используется для выполнения GET-запросов в HTTP.
- @POST: Используется для выполнения POST-запросов в HTTP.
- @Path: Указывает относительный путь к классу Java.
- @QueryParam: обозначает параметры запроса URI или URL.

Каковы некоторые ключевые особенности APIJAX-RS?

Ответ: К особенностям JAX-RS относятся:

- Кэширование на стороне клиента
- Кэширование на стороне сервера
- Настройка строки запроса
- Аннотации времени выполнения

Как можно настроить приложения JAX-RS?

Ответ: Приложение JAX-RS состоит как минимум из одного класса ресурсов, упакованного в WAR-файл. Базовый URI, с которого ресурсы приложения отвечают на запросы, может быть задан одним из двух способов:

- Используя аннотацию @ApplicationPath в подклассе javax.ws.rs.core.Application, упакованном в WAR
- С помощью тега servlet-mapping в дескрипторе развертывания web.xml WAR

Что такое JAX-WS и JAX-RS?

Ответ: JAX-WS – это Jakarta XML Web Services API, используемый для разработки API с помощью Simple Object Access Protocol (SOAP) – протокола обмена сообщениями на основе XML.

С другой стороны, JAX-RS – это Java API, используемый для создания веб-сервисов с использованием архитектуры REST.

Что такое коды состояния HTTP?

Ответ: Коды состояния — это не что иное, как способ передачи статуса ответа, отправленного сервером клиенту. Они присутствуют в заголовках ответа, отправляемых сервером.

С помощью кодов состояния клиент может определить, был ли запрос неудачным или выполненным, или что-то не так с ответом.

Вот некоторые распространенные коды состояния HTTP:-

- 200 — Означает ключевое слово “OK”. Это означает, что запрос был выполнен, и ответ в порядке.
- 404 — Расшифровывается как “Не найдено”. Это означает, что ресурс отсутствует на сервере или конечная точка не существует.
- 500 — расшифровывается как “Внутренняя ошибка сервера”. Обычно это происходит, когда сервер не может сгенерировать корректный ответ, или же возникает ошибка, которая не является явной.
- 503 — расшифровывается как “Сервис недоступен”. Это означает, что в данный момент сервер не может обработать ни одного запроса, возможно, потому что он мертв или не работает из-за перегрузки запросами. Это также может происходить, когда сервер находится на техническом обслуживании.

Что такое методы HTTP?

Ответ: HTTP-методы используются для выполнения определенного типа действий над конкретным ресурсом API. Например, если вы хотите получить список фильмов из API коллекции фильмов, то вы можете использовать метод GET, предоставляемый HTTP. Если вы хотите обновить данные, вы можете использовать метод POST, предоставляемый HTTP.

Часто используемые методы HTTP выглядят следующим образом:

- **GET**: Запросы, использующие GET, должны только получать данные.
- **POST**: Обновляет ресурс, отправляя на сервер только что обновленный ресурс.
- **DELETE**: удаляет указанный ресурс.
- **PATCH**: Частично изменяет ресурс.

Как работает базовая аутентификация HTTP?

Ответ: Аутентификация — это процесс проверки подлинности клиента для обеспечения безопасности данных. В HTTP аутентификация работает через заголовок авторизации, который отправляется клиентом.

Заголовок авторизации состоит из имени пользователя/идентификатора и пароля клиента, которые затем проверяются сервером, и доступ предоставляется.

Важно отметить, что при использовании HTTP-аутентификации канал, по которому передаются учетные данные, должен быть зашифрован и защищен.

Защитить канал можно с помощью уровня SSL, который интегрирован в HTTPS. Поэтому при работе с учетными данными рекомендуется использовать HTTPS, а не простой HTTP.

Каковы основные компоненты HTTP-запроса?

Ответ: HTTP-запрос состоит из следующих компонентов:

- **Строка запроса**: Это первая строка в любом запросе,

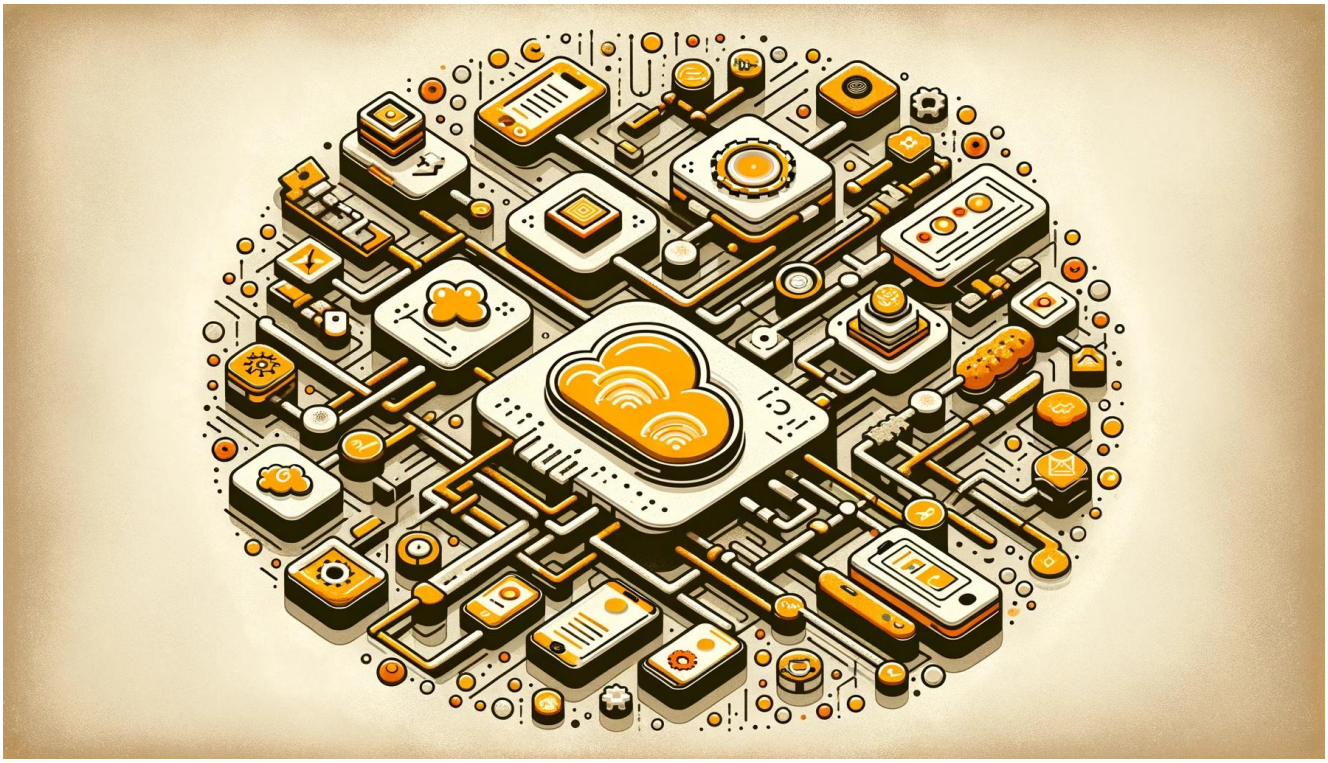
которая состоит из метода HTTP, пути или конечной точки и номера версии HTTP.

- **Заголовки**: HTTP-заголовки используются для предоставления метаданных о запросе.
- **Тело (необязательно)**: Этот компонент присутствует только для некоторых методов запроса. Он не требуется для GET-запросов, но необходим для POST-запросов. Это фактическое сообщение запроса.

Каковы основные компоненты HTTP-ответа?

Ответ: Ответ HTTP состоит из следующих компонентов:

- **Статус**: Это код состояния HTTP, который отправляется сервером.
- **Заголовки**: Как и запросы, ответы также имеют соответствующие заголовки, которые предоставляют полезную информацию об ответе.
- **Сообщение**: Это фактические данные, которые сервер отправляет клиенту для запроса определенного ресурса.



В чем разница между REST и AJAX?

Ответ: AJAX — это клиент, с помощью которого вы можете получить доступ к RESTful API. Он используется для отправки асинхронных запросов с помощью JavaScript.

REST, или Representational State Transfer, — это архитектура, которая может быть реализована для создания RESTful API. Короче говоря, для отправки HTTP-запросов можно использовать AJAX, который служит в качестве клиента, но если вы хотите реализовать RESTful API, то вам нужно использовать архитектуру REST.

В чем разница между SOAP и REST?

Ответ: **Representational State Transfer,** или REST, — это архитектура с минимальными ограничениями для создания API. SOAP, или **Simple Object Access Protocol,** — это протокол с жесткими требованиями для реализации API.

REST более гибкий и простой в использовании, чем SOAP. В SOAP используется обмен сообщениями на основе XML, в то время как в

REST можно использовать множество типов передачи данных, таких как JSON, XML и т. д. По сравнению с SOAP, REST более легкий и быстрый.

Веб-сервисы SOAP имеют встроенную систему безопасности, что является одним из преимуществ использования SOAP перед REST, но дополнительные функции также делают его сложным и тяжелым в использовании.

В чем разница между PUT и POST?

Ответ: POST — это метод HTTP-запроса, который отправляет некоторые данные на сервер. Если вы сделаете несколько POST-запросов для определенного ресурса, то это может привести к побочным эффектам для ваших данных. Например, если вы хотите добавить статью в коллекцию, если вы сделаете несколько POST-запросов, в коллекцию будет добавлено несколько статей, что приведет к появлению лишних статей.

PUT — это метод HTTP-запроса, который отправляет данные на сервер для определенного ресурса, но обновляет их только один раз. Если вы отправите несколько PUT-запросов для определенного ресурса, никаких побочных эффектов не возникнет, а данные будут добавлены только один раз. При PUT, если ресурс не существует, создается новый, а если он существует, то обновляется существующий.

PUT является идемпотентным, а POST — нет.

Что такое полезная нагрузка?

Ответ: Полезная нагрузка в REST API — это просто тело запроса, отправленного клиентом на сервер. Это данные, которые вы хотите отправить на сервер и получить ответ.

Какой максимальный размер полезной нагрузки может быть отправлен в методах Post?

Ответ: Протокол HTTP не устанавливает ограничений по умолчанию. Ограничение может зависеть от максимального предела клиента или сервера, в зависимости от того, что является минимальным.

Какие лучшие практики необходимо соблюдать при создании URI?

Ответ: Некоторые из ключевых моментов, которые необходимо учитывать при создании URI, следующие:

- Избегайте использования расширений файлов
- Будьте последовательны во всех URI
- Разделяйте URI на домены и поддомены для различных наборов ресурсов
- Используйте дефис или подчеркивание для разделения слов в предложениях, встроенных в URI
- Используйте прямую косую черту для обозначения иерархии ресурсов
- Кодируйте URI с помощью правильной кодировки
- Старайтесь сделать URI человекочитаемым

Что такое идемпотентные методы?

Ответ: Идемпотентные HTTP-методы оказывают одинаковое воздействие на сервер, несмотря на отправку нескольких одинаковых запросов. Например, если вы отправите несколько одинаковых запросов DELETE для определенного ресурса, ресурс не будет меняться при каждом запросе; он будет обновляться

так, как будто был отправлен только один запрос.

Некоторые из идемпотентных методов включают:

- PUT
- DELETE
- GET
- HEAD
- OPTIONS

Что такое Postman?

Ответ: Postman — это инструмент для разработки, модификации и тестирования API. Он предоставляет множество возможностей для быстрого создания и тестирования API без необходимости установки клиента.

Что такое заголовки Cache-Control?

Ответ: Заголовок Cache-Control состоит из инструкций или директив для настройки кэширования в браузерах и серверах. Он указывает браузеру или серверу, что кэшировать и как долго это должно кэшироваться перед запросом через сеть.

Заголовок Cache-Control включает в себя следующие директивы:-

- max-age
- no-cache
- частный
- публичный
- no-store
- неизменяемый

Дайте определение обмена сообщениями в RESTful Web Services.

Ответ: Обмен сообщениями в веб-сервисах RESTful означает, что клиент отправляет HTTP-запрос на сервер, на который сервер отвечает HTTP-ответом. Это взаимодействие между клиентом и сервером называется обменом сообщениями.

В чем разница между монолитной архитектурой, SOA и архитектурой микросервисов?

Ответ: В монолитной архитектуре все управляется в одном месте. Клиентская часть, сервер, а также база данных управляются из одного места. Именно поэтому она известна как монолитная, ведь слово “монолит” означает единый блок или камень.

SOA расшифровывается как Service-Oriented Architecture. В этой архитектуре различные аспекты приложения управляются различными сервисами, которые также являются программным обеспечением. Таким образом, это комбинация нескольких сервисных программных модулей. Интеграция является ключевой частью этой архитектуры.

Архитектура микросервисов похожа на SOA, но, в отличие от SOA, в ней есть несколько автономных программ, которые общаются друг с другом с помощью API. В отличие от монолитной архитектуры, здесь все автономно и в какой-то степени независимо.

Как работает микросервисная архитектура?

Ответ: В микросервисной архитектуре приложения делятся на более мелкие подразделения, которые независимы друг от друга и

работают сами по себе, но взаимодействуют друг с другом с помощью четко определенного набора API.

К преимуществам микросервисной архитектуры относятся гибкость, гибкость, масштабируемость, независимые технологии, многократно используемые сервисы и простота развертывания.

Что такое CRUD?

Ответ: CRUD расшифровывается как Create (создание), Read (чтение), Update (обновление), Delete (удаление). Это операции, которые можно выполнять над определенным ресурсом. API, поддерживающий все эти операции, называется CRUD API. Это самые основные операции, которые могут быть выполнены API над ресурсом.

Что такое кэширование?

Ответ: Кэширование – это техника хранения ответа или запроса на клиенте или сервере для последующего использования.

Ответы обычно кэшируются на клиенте, потому что если клиент делает один и тот же запрос несколько раз за короткий промежуток времени, то нет смысла снова запрашивать ответ по сети и тратить пропускную способность.

Для чего используется @RequestMapping?

Ответ: Это аннотация в spring framework, которая используется для отображения веб-запросов на определенные классы обработчиков и/или методы обработчиков.

Что делает @PathVariable?

Ответ: Аннотация @PathVariable в spring framework используется для извлечения значения переменных шаблона и присвоения их значения переменной метода.

Что такое HttpMessageConverter.

Ответ: Когда HTTP-запрос (или его часть) нужно преобразовать в тип, необходимый в качестве аргумента для метода-обработчика, или когда значение, возвращаемое методом-обработчиком, нужно каким-либо образом преобразовать для создания HTTP-ответа, используются конвертеры HTTP-сообщений.

Какие инструменты необходимы для тестирования вашего Web API?

Ответ: Ниже перечислены некоторые инструменты, которые могут помочь вам в тестировании API:

- Postman
- Rest Assured
- Rest Sharp
- Katalon
- ReadyAPI
- Apigee

Заключение

В завершение, список из 40ка часто задаваемых вопросов и ответов по REST API представляет собой ценный ресурс для всех, кто стремится глубже понять эту важную технологию. REST API играет ключевую роль в современной разработке программного обеспечения, обеспечивая эффективное и гибкое взаимодействие между различными системами и приложениями. Понимание основ, принципов и лучших практик работы с REST API поможет разработчикам создавать мощные, надежные и масштабируемые приложения. Надеемся, что ответы на эти вопросы помогут вам уверенно навигировать по миру REST API и использовать его потенциал для достижения ваших технических и бизнес-целей.

Переведено с сайта Geekflare