HTTP-Methoden und REST-APIs

Geschätzte benötigte Zeit: **30** Minuten

Ziele

Nach dem Lesen dieses Dokuments sollten Sie in der Lage sein:

- Begriffe im Zusammenhang mit HTTP-Methoden zu definieren
- Richtlinien und Best Practices für das Schreiben von REST-APIs zu erklären

Das Internet basiert auf einer Client-Server-Architektur. Der Endbenutzer interagiert mit dem Client, während die Server die **Dienste**[1] beherbergen, die die Anwendungen, die Geschäftslogik und die Daten betreiben. Clients kommunizieren mit den Servern, um die gewünschte Funktionalität für den Endbenutzer zu erreichen. Daten werden zwischen Client und Server über das Hypertext-Übertragungsprotokoll, besser bekannt als **HTTP**[2], übertragen. Diese Kommunikation erfolgt in der Regel über **APIs**[3]. Im Jahr 2000 wurde eine Reihe von Richtlinien für das Schreiben dieser APIs für eine Client-Server-Architektur entwickelt, die als **REST**[4]-APIs bezeichnet werden.

Das Akronym „REST“ steht für REpresentational State Transfer. Bevor wir diesen Begriff näher erläutern, lassen Sie uns zunächst über HTTP-Methoden und einige Begriffe sprechen.

In einer Client/Server-Architektur bestehen die Anwendungen aus einem oder mehreren Diensten, die auf den Servern residieren. Diese Dienste enthalten **Ressourcen**[5]. Der Client stellt eine **Anfrage**[6] nach einer Ressource über ein **Anfrageobjekt**[7] unter Verwendung einer **Route**[8], die einen **Endpunkt**[9] innerhalb des Dienstes hat. Die Anwendung sendet ein **Antwortobjekt**[10] als **Antwort**[11] an den Client zurück, um dieser Anfrage nachzukommen.

Ein Anfrageobjekt besteht aus drei Teilen: einer **URL**[12], einem **Anfrageheader**[13] und einem **Anfragekörper**[14]. Der Server verwendet die URL, um den Dienst und den Endpunkt innerhalb des Dienstes zu identifizieren, auf den zugegriffen wird. Die URL enthält vier Teile: ein **Protokoll**[15], einen **Hostnamen**[16], einen **Pfad**[17] und eine **Abfragezeichenfolge**[18]. Der Anfrageheader enthält Metadaten über die Ressource des anfragenden Clients, wie den Benutzer **Agent**[19], **Host**[20], **Inhaltstyp**[21], **Inhaltslänge**[22] und welche Art von Daten der Client in der Antwort erwarten sollte.

Der Server antwortet mit einem Antwortobjekt, das aus einem **Header**[23], einem **Körper**[24] und einem **Statuscode**[25] besteht. Der Körper des Antwortobjekts enthält oft eine **JSON**[26] **Nutzlast**[27], um die Daten an den Client zurückzugeben.

Es gibt eine Reihe von HTTP-Methoden, die in der REST-API verwendet werden können und die Interaktion zwischen dem Client und einem Dienst ermöglichen. Die häufigsten Methoden sind **GET**[28], **POST**[29], **PUT**[30], **DELETE**[31] und **PATCH**[32].

Der Name der Methode beschreibt, was mit der Ressource passiert, wenn die Methode angewendet wird. PUT- und DELETE-Methoden führen zu **idempotenten**[33] Daten, wenn dieselbe API-Methode mehrfach aufgerufen wird.

HTTP hat drei Möglichkeiten, Parameter zu übergeben: den **URL-Pfadparameter**[34], den **URL-Abfrageparameter**[35] und den **Header-Parameter**[36]. Die Pfad- und Abfrageparameter werden als Teil der URL übergeben, während der Header-Parameter direkt vom Browser an den Dienst übergeben wird.

Wenn der Dienst eine Anfrage abschließt, gibt er eine Antwort zurück. Ein HTTP-Statuscode sollte Teil dieser Antwort sein. Der HTTP-Statuscode zeigt an, ob die Antwort abgeschlossen wurde oder nicht. Die Kategorien der Antwortcodes sind in der folgenden Tabelle dargestellt.

Statuscode-Bereich	**Bedeutung**
200-299	Alles ist in Ordnung
300-399	Ressource wurde verschoben
400-499	Client-seitiger Fehler
500-599	Server-seitiger Fehler

Wie bereits erwähnt, ist REST eine Reihe von Richtlinien. Es gibt fünf Anforderungen, damit eine API als RESTful betrachtet werden kann, plus ein optionales Kriterium:

1. Die API nutzt eine Client-Server-Architektur, die aus Ressourcen besteht, die über HTTP verwaltet und bereitgestellt werden.
2. Die Kommunikation zwischen Client und Server ist **zustandslos**[37].
3. Daten sind **cachable**[38], um die Leistung auf der Client-Seite zu verbessern.
4. Die Schnittstelle wird in einem Standardformat übermittelt, sodass die angeforderten Ressourcen, die auf dem Server gespeichert sind, von der Darstellung, die an den Client gesendet wird, getrennt sind. Die an den Client gesendete Darstellung enthält genügend Daten, damit der Client die Darstellung manipulieren kann.
5. Anfragen und Antworten kommunizieren über verschiedene Schichten, wie **Middleware**[39]. Der Client und der Server kommunizieren oft nicht direkt miteinander.
6. (Optional) Ressourcen sind in der Regel statisch, können jedoch auch ausführbaren Code enthalten. Der Code sollte nur ausgeführt werden, wenn der Client dies anfordert.

Wenn der Client eine Anfrage stellt, muss er auch Informationen über seinen Zustand an den Server übermitteln. Jede Kommunikation zwischen dem Client und dem Server sollte alle Informationen enthalten, die zur Durchführung der Anfrage erforderlich sind. Der Client, nicht der Dienst, verwaltet den Zustand. Jede Anfrage muss die erforderlichen Informationen enthalten, damit der Server die Anfrage versteht. Wenn der Benutzer beispielsweise einen Datensatz in einer Datenbank anzeigt und dieser Datensatz aktualisiert werden muss, muss der Client auch senden, welcher Datensatz aktualisiert werden muss. Der Server weiß nicht, welcher Datensatz derzeit angezeigt wird.

Ein `@app.route()`-Methode wird verwendet, um einen RESTful-Dienst zu definieren. Diese Methode nimmt zwei Parameter: die Route von der URL zu dem Dienst, auf den zugegriffen wird, und einen optionalen HTTP-Methodenparameter wie POST, GET usw. Der Routenparameter kann Variablen enthalten, wie z. B. `<username>`. Der Wurzel einer Route wird durch '/' dargestellt. Wenn Sie beispielsweise möchten, dass die Route www.mywebsite.com/accounts ist, müssen Sie nur '/accounts' als Routenparameter in der `@app.route()`-Funktion angeben.

REST-APIs haben die folgenden Merkmale:

- Ressourcenbasiert; das heißt, sie beschreiben Mengen von Ressourcen
- Enthalten nur Substantive, keine Verben
- Verwenden Singularsubstantive, wenn sie auf eine einzelne Ressource verweisen, oder Pluralsubstantive, wenn sie auf eine Sammlung von Ressourcen verweisen
- Werden immer durch URLs identifiziert

NICHT RESTful APIs	**RESTful-Äquivalente**
GET http://api.myapp.com/getUser/123	GET http://api.myapp.com/users/123
POST http://api.myapp.com/addUser	POST http://api.myapp.com/users
GET http://api.myapp.com/removeUser/123	DELETE http://api.myapp.com/users/123

URL-Formatrichtlinien

- Sollten einen Schrägstrich '/' verwenden, um eine hierarchische Beziehung in der Verzeichnisstruktur anzuzeigen
- Sollten einen abschließenden Schrägstrich vermeiden, z. B. /resource/
- Sollten Bindestriche verwenden, nicht Camel Case, z. B. /my-resource, nicht /myResource
- Sollten keinen Unterstrich '_' in der URL verwenden, z. B. /my-resource, nicht /my_resource
- Sollten Kleinbuchstaben verwenden
- Sollten keinen Punkt '.' in einer URL verwenden
- Können mehrere untergeordnete Ressourcen und IDs in der URL enthalten, z. B. GET /resource/{id}/subordinate/{id}

Begriffe und Definitionen

Begriff	**Beschreibung**
1.Dienst	Eine Softwarekomponente, die Teil einer Anwendung ist und einen bestimmten Zweck erfüllt. Im Allgemeinen nimmt ein Dienst Eingaben von einem Client oder einem anderen Dienst entgegen und erzeugt eine Ausgabe.
2.HTTP	Das Protokoll, das von einer Client-Server-Architektur im Internet zum Abrufen oder Austauschen von Ressourcendaten verwendet wird.
3.API	Eine Anwendungsprogrammierschnittstelle ist eine Reihe von Definitionen und Protokollen, die es zwei Diensten ermöglichen, miteinander zu kommunizieren. Die API fordert Daten an, die zwischen einer Ressource und den Ergebnissen, die von dieser Ressource zurückgegeben werden, ausgetauscht werden können.
4.REST	Eine Reihe von architektonischen Richtlinien, die beschreiben, wie man eine Schnittstelle (API) zwischen zwei Komponenten, normalerweise einem Client und einem Server, schreibt, die beschreiben, wie diese Komponenten miteinander kommunizieren. REST steht für REpresentational State Transfer. REST beschreibt eine standardisierte Methode zur Identifizierung und Manipulation von Ressourcen. REST stellt sicher, dass die zwischen Client und Server ausgetauschten Nachrichten selbstbeschreibend sind und definieren, wie der Client mit dem Server interagiert, um auf Ressourcen auf dem Server zuzugreifen.
5.Ressource	Eine Ressource ist das grundlegende Konzept einer RESTful API. Es handelt sich um ein Objekt, das einen definierten Typ, zugehörige Daten, Beziehungen zu anderen Ressourcen und eine Reihe von Methoden hat, die darauf operieren können. Eine Ressource wird häufig im JSON-Format definiert, kann aber auch in XML vorliegen.
6.Anfrage	Eine Anfrage wird von einem Client an einen Host auf einem Server gestellt, um auf eine Ressource zuzugreifen. Der Client verwendet Teile einer URL, um die benötigten Informationen von der Ressource zu bestimmen. Die häufigsten Anfragemethoden sind GET, POST, PUT, PATCH und DELETE, aber auch HEAD, CONNECT, OPTIONS, TRACE und PATCH.
7.Anfrageobjekt	Enthält die HTTP-Anfragedaten. Es besteht aus drei Teilen: einer URL, einem Header und einem Body.
8.Route	Die Kombination aus einer HTTP-Methode und dem Pfad zur Ressource vom Wurzelpfad aus.
9.Endpunkt	Der Standort der Ressource, der von einer REST-API auf dem Server angegeben wird. Er wird normalerweise über die URL in der HTTP-Methode der API identifiziert.
10.Antwortobjekt	Enthält die HTTP-Antwortdaten als Reaktion auf eine Anfrage. Es enthält einen Header, einen Body und einen Status.
11.Antwort	Eine Antwort wird von einem Server erstellt und an einen Client gesendet, um entweder dem Client die angeforderte Ressource bereitzustellen, dem Client mitzuteilen, dass die angeforderte Aktion abgeschlossen ist, oder den Client darüber zu informieren, dass ein Fehler bei der Verarbeitung der Anfrage aufgetreten ist.
12.URL	Ein "Uniform Resource Identifier" wird synonym mit dem Begriff URL verwendet. Sie sind Teil einer RESTful API, die den Endpunkt der angeforderten Ressource lokalisiert und die Daten enthält, wie dieser Endpunkt manipuliert werden sollte. Der Client gibt eine HTTP-Anfrage unter Verwendung der URI/URL ab, um die Ressource zu manipulieren. Sie sollten aus vier Teilen bestehen: dem Hostnamen, dem Pfad, dem Header und einer Abfragezeichenfolge.
13.Anfrage-Header	Informationen, die dem Server über die abgerufene Ressource oder den anfordernden Client übermittelt werden. Beispiele sind: • Methode mit Endpunkt: POST /car-reviews • User-agent: Der Typ des Browsers, den der Client verwendet. • Host: Ein Computer in einem Netzwerk, der mit anderen Hosts kommuniziert. • ContentType: Der Medientyp einer Ressource wie Text, Audio oder ein Bild. • Content length: Die Anzahl der Bytes an Daten, die in einer Antwort gesendet werden. • Accept-Encoding: Erwartetes Rückformat der Daten, z.B. application/json • Verbindungsinformationen
14.Anfrage-Body	Stellt die Daten bereit, die an den Server übermittelt werden.
15.Protokoll	Gibt dem Dienst an, wie die Daten zwischen dem Server und dem Client übertragen werden sollen.
16.Hostname	Der Name eines Geräts in einem Netzwerk, das auch oft als Site-Name bezeichnet wird.

17.Pfad	Der Pfad identifiziert den Standort der Ressource im Dienst und ihren Endpunkt. Zum Beispiel: https://www.customerservice/customers/{customer_id}
18.Abfragezeichenfolge	Der Teil einer URL, der die Abfrage enthält.
19.User-agent	Der Typ des Browsers, den der Client verwendet.
20.Host	Ein Computer in einem Netzwerk, der mit anderen Hosts kommuniziert.
21.Inhaltstyp	Der Medientyp einer Ressource wie Text, Audio oder ein Bild.
22.Inhaltlänge	Die Anzahl der Bytes an Daten, die in einer Antwort gesendet werden.
23.Antwort-Header	Enthält Metadaten zur Antwort, wie einen Zeitstempel, Caching-Kontrolle, Sicherheitsinformationen, Inhaltstyp und die Anzahl der Bytes im Antwort-Body.
24.Antwort-Body	Die Daten von der angeforderten Ressource werden an den Client zurückgesendet.
25.Antwort-Status	Der Rückgabecode, der das Ergebnis des Status der Anfrage an den Client kommuniziert.
26.JSON	"JavaScript Object Notation" ist ein Format zum Speichern und Transportieren von Daten, normalerweise als Möglichkeit, Daten von einem Dienst auf einem Server an den Client zu senden. Es besteht aus Schlüssel-Wert-Paaren und ist selbstbeschreibend. Das Format der JSON-Daten ist dasselbe wie der Code zum Erstellen von JavaScript-Objekten, was die Umwandlung dieser Daten in JavaScript-Objekte erleichtert, kann aber in jeder Programmiersprache geschrieben werden. JSON hat drei Datentypen: Skalare (Zahlen, Zeichenfolgen, Booleans, null), Arrays und Objekte.
27.Nutzlast	Die Nutzlast sind die Daten im Body einer Antwort, die aufgrund einer API-Anfrage von einem Server an den Client transportiert werden.
28.GET	Die GET-Methode wird als Anfrage verwendet, die eine Darstellung einer Ressource abruft. GET() sollte niemals eine Ressource ändern und nur eine Darstellung der angeforderten Ressource zurückgeben.
29.POST	HTTP-Methode, die Daten an den Server sendet, um eine Ressource zu erstellen und den Statuscode 201_CREATED zurückgeben sollte.
30.PUT	HTTP-Methode, die eine Ressource aktualisiert oder eine vorhandene ersetzt. Mehrmaliges Aufrufen von PUT hintereinander hat keine Nebeneffekte, während POST dies hat. Es sollte einen 200_OK-Code zurückgeben, wenn die Ressource existiert und aktualisiert werden kann, oder einen 404_NOT_FOUND-Code, wenn die Ressource nicht existiert.
31.DELETE	HTTP-Methode, die eine Ressource löscht und 204_NO_CONTENT zurückgibt, wenn die Ressource existiert und vom Server gelöscht werden kann, oder wenn die Ressource nicht gefunden werden kann, was bedeutet, dass sie bereits gelöscht wurde.
32.PATCH	HTTP-Methode, die partielle Modifikationen an einer Ressource anwendet.
33.Idempotent	Beschreibt ein Element einer Menge, das unverändert bleibt, wenn mehrere identische Anfragen gestellt werden. PUT- und DELETE-Methoden führen zu idempotenten Daten, wenn dieselbe API-Methode mehrfach aufgerufen wird.
34.URL-Pfadparameter	Wird vom Client als Variable im Pfad der URL an die Operation übergeben.
35.URL-Abfrageparameter	Enthält Schlüssel-Wert-Paare, normalerweise im JSON-Format, und wird durch ein '?' vom Pfad getrennt. Wenn es mehrere Schlüssel-Wert-Paare gibt, sollten sie durch ein '&' getrennt werden. Die Abfrage kann verwendet werden, um einen Filter zu übergeben, der auf die von der Operation zurückgegebenen Ergebnisse angewendet werden soll.
36.Header-Parameter	Enthält zusätzliche Metadaten zur Abfrage, wie die Identifizierung des Clients, der die Operation aufruft.
37.Zustandslos	Alle Anfragen von einem Client an einen Server nach Ressourcen erfolgen isoliert voneinander. Der Server ist sich des Anwendungsstatus auf dem Client nicht bewusst, sodass diese Informationen mit jeder Anfrage übermittelt werden müssen.
38.Cachebar	Die Fähigkeit, Daten auf dem Client zu speichern, sodass diese Daten in einer zukünftigen Anfrage verwendet werden können.
39.Middleware	Software, die zwischen Anwendungen, Datenbanken oder Diensten sitzt und es diesen unterschiedlichen Technologien ermöglicht, miteinander zu kommunizieren.

Skills Network