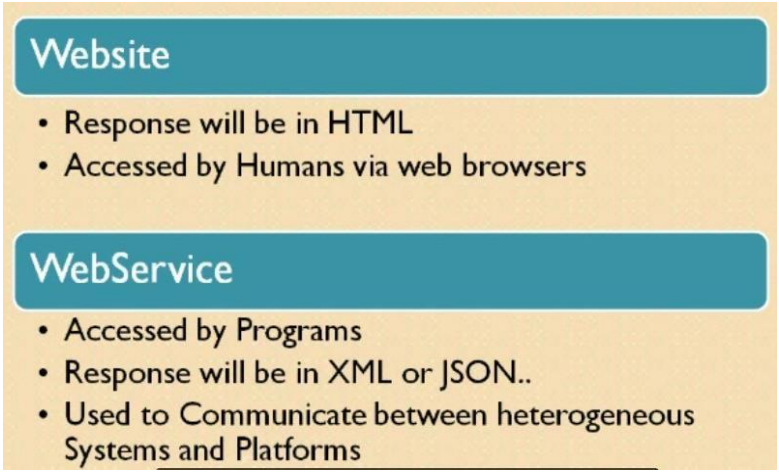
Überblick Webservice, API, REST

Web-Services sind der Grundpfeiler moderner verteilter Anwendungen. Mit Hilfe von **WebServices** schaffet man **Schnittstellen**, die beliebigen Clients erlauben, Dienste der Server-Anwendung in Anspruch zu nehmen.

Unterschied Website und Webservice:



API:
Ein API stellt Entwicklern Daten zur Verfügung und nimmt Daten entgegen. Es steht für „application programming interface“. Es kann

- Daten in einer Datenbank ablegen, und Endpunkte bereitstellen nämlich
- zum Lesen,
- Anlegen,
- Bearbeiten und Löschen.

Der Vorteil von APIs: als deren Nutzer muss man sich nicht damit auseinandersetzen, wo die Daten gespeichert werden und wie die Datenbank dahinter aussieht. Man schickt JSON an HTTP(s)-Endpunkte und bekommt JSON-Daten zurück. Wie die Endpunkte heißen und wie die Anfrage und die Antwort aussehen, beschreibt die API-Dokumentation. Die wird hier nun beschrieben:

APIs verständlich zu dokumentieren:
Um diese APIs sinnvoll einsetzen zu können, wenn z.B. Entwicklerteams an getrennten Komponenten arbeiten und über APIs kommunizieren, benötigt der Entwickler ein Format für die Dokumentation der Daten. Dafür gibt es den meistverbreiteten Standard „OpenAPI-Specification“, abgekürzt OAS.

Ursprünglich wurde das Format unter dem Namen „**Swagger**“ entwickelt. Als Swagger zum Quasi-Industriestandard wurde, entschied sich die Entwicklerfirma „SmartBear“, der deren geistiges Eigentum an die „OpenAPI Initiative“ zu übergeben. Dieser Gemeinschaft unter dem Dach der Linux-Foundation gehören etwa 30 Unternehmen als Mitglieder an. Große Firmen verwenden OAS wie z.B. der Modeversandhändler Zalando und die deutsche Fluglinie Lufthansa.

Kein vernünftiger Programmierer wird heute direkt über manuell zusammengebaute Nachrichten mit einem Web-Service kommunizieren. **Stattdessen verwendet man APIs**, die die gesamte Kommunikation zwischen Client und Server transparent durchführen. Ob für diese Kommunikation dann JSON, XML oder ein völlig anderes Format eingesetzt wird, spielt dabei keine Rolle. Tatsächlich gibt es APIs, bei denen das Nachrichtenprotokoll beliebig ausgetauscht werden kann, ohne dass sich für den Programmierer etwas ändert.

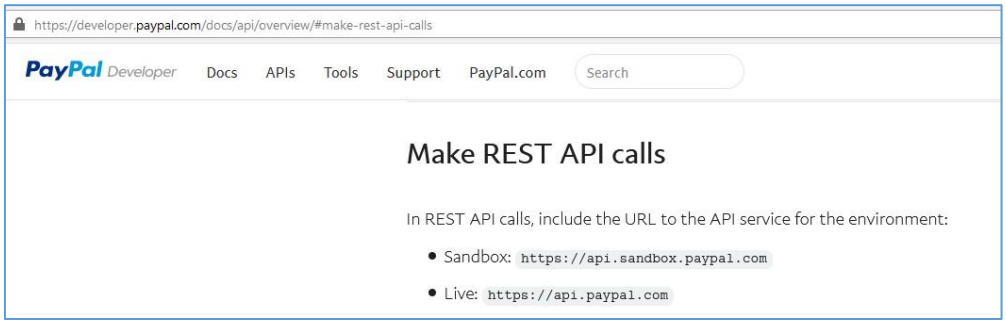
Aktuell unterscheidet man insbesondere **zwei Arten von Web-Services**. Und zwar

- **SOAP und REST**

Beide haben gemeinsam, dass zwei Parteien miteinander kommunizieren. Ein Client und ein Server. Bei einem auf **SOAP** basierten Web-Service kann der Datenaustausch zwischen Server und Client nur mithilfe von **XML Daten** geschehen wohingegen wir bei einem REST Web-Service bei der Auswahl des Datenformats im Wesentlichen frei sind.

Viel wichtiger ist aber REST.

Beispiel: Eine API ist wie eine URL die einen JSON-Response zurückgibt. Hier für Paypal, um Paypal im eignem Webshop anbieten zu können.



Representational State Transfer (REST)

REST ist kein Web-Service-Protokoll. REST ist weder eine konkrete Technologie noch ein offizieller Standard. Es handelt sich vielmehr um einen **Softwarearchitekturstil**, bestehend aus Leitsätzen und bewährten Praktiken für netzwerkbasierte Systeme.

Das **REST-Paradigma** entwickelte sich aus dem **1994 von Roy Fielding** entworfenen HTTP Object Model. Fielding entwickelte seine Idee von einem einheitlichen Konzept über die Jahre weiter, bis er 2000 den REST-Architekturstil im Rahmen seiner Dissertation veröffentlichte. Fielding war zuvor auch an der Spezifikation des Hypertext-TransferProtokolls (HTTP) beteiligt.

RESTful APIs werden von vielen Websites eingesetzt, unter anderen von Google, Amazon, Twitter und LinkedIn.

Tatsächlich handelt es sich bei REST um ein Architekturmodell für verteilte Anwendungen, dass man, vielleicht ohne es zu wissen, bereits kennt. Der bekannteste Vertreter des REST-Architekturmodells ist das World Wide Web.

Ausschlaggebend für die „**RESTfulness**“ (als **RESTful** bezeichnet man eine Anwendung, die den Prinzipien des REST-Modells folgt.) einer Architektur ist dabei das Vorhandensein von Ressourcen

- die über eine einheitliche Syntax (URLs) adressierbar sind
- sowie einheitlicher Operationen, die auf diese Ressourcen angewendet werden können.

Obwohl dieses Prinzip allen Websites zugrunde liegt, ist es insbesondere im Zusammengang mit Web-Services interessant.

Die **technische Grundlage** des **REST-Modells** ist das **http-Protokoll**.

Es definiert eine **Reihe von Verben**, die die Art einer Anfrage beschreiben. Neben dem häufig verwendeten **GET** und **POST** existieren **PUT** und **DELETE**. Betrachtet man diese Verben genauer, erkennt man gewisse Parallelen zu Operationen, die in vielen Anwendungen sehr häufig durchgeführt werden:

- POST für Erstellen,
- GET für Lesen,
- PUT für Aktualisieren und DELETE für Löschen.

Beispiele für mögliche Endpunkte:

Endpunkt	HTTP-Methode	Funktion
/magazines	GET	gibt alle Magazine zurück
/magazines	POST	erstellt ein Magazin
/magazines/{id}	GET	gibt das Magazin mit der angegebenen ID zurück
/magazines/{id}	PUT	aktualisiert den kompletten Datensatz eines Magazins
/magazines/{id}	PATCH	aktualisiert Teile des Datensatzes eines Magazins
/magazines/{id}	DELETE	löscht den Datensatz eines Magazins

Möchte man den Server also beispielsweise anweisen, etwas zu löschen, so könnte man – statt wie gewohnt – einer POST- oder GET-Anfrage auch eine DELETE-Anfrage benutzen. Doch dann fragt sich, wie man dem Server mitteilen soll, was er zu löschen hat. Auch hierfür hat das HTTP-Protokoll (oder vielmehr das Web) bereits die Antwort: Jeder „Ressource“, die der Server verwaltet, wird eine eigene URL zugeordnet.

Beispiel:
Möchte man also beispielsweise den Kunden mit der Nummer 94213 aus der Datenbank löschen, so könnte man dem Server eine DELETE-Anfrage an die URL

http://www.mustersoft.de/kunden/94213

senden. Möchte man stattdessen nur die persönlichen Daten des Kunden aktualisieren, so würde man eine PUT-Anfrage starten. Allerdings wäre es damit natürlich nicht getan, denn die neuen Kundendaten müssen dem Server ja in irgendeiner Form mitgeteilt werden. Hierzu

macht REST keine genauen Vorschriften. In der Praxis verwendet man allerdings häufig XML. Prinzipiell kann man aber auch HTML oder jedes beliebige andere Format wählen. Da REST hier keine besonderen Anforderungen stellt, muss natürlich gewährleistet sein, dass Client und Server dieselbe Sprache sprechen bzw. verstehen. Das REST-Modell zeichnet sich vor allem dadurch aus, dass es leicht zu verstehen ist und auf Konzepten aufbaut, die sich immer wiederholen. Web-Services, die diesem Muster folgen, werden deshalb immer beliebter. So bietet beispielsweise Amazon seine WebServices inzwischen sowohl in einer SOAP- als auch in einer REST-Variante an.

Austausch von Daten: Web-Services sind Dienste im Web mit denen man Rohdaten austauschen kann. Diese Rohdaten können von unterschiedlichem Format sein. Sehr beliebte Formate sind das XML und das JSON Format.

Beispiel:
Facebook bietet eine eigene URL an, über die man auf deren Services zugreifen kannst. Nämlich https://developers.facebook.com. Über diese URL hat man Zugriff auf Services, mit denen man sich bei Facebook anmelden, posten und Freundeslisten auslesen kann. Damit können deine Programme also alles das machen, was du normalerweise mit Maus und Tastatur per Hand durchführst, wenn du bei Facebook eingeloggt bist.

Genau genommen sprechen wir hier bereits von einer speziellen Art von Web-Service. Nämlich einem sogenannten **RESTful Web-Service**.

Die Sammlung aller REST Services auf einem Server nennt man REST API.

Toolunterstützung

REST basiert auf eingeführten Standards, für die es bereits eine Unzahl von Tools gibt. REST fähig sind Web Server wie Apache oder IIS, Web Container wie Tomcat und Proxy Server wie Squid. Spezialisierte REST Tools sind nicht notwendig. Es ist durchaus denkbar, dass zukünftig Bibliotheken, Frameworks oder Tools die Entwicklung von REST Anwendungen gezielt unterstützen.

REST nutzt HTTP für eine **zustandslose Client-Server-Kommunikation**, d. h., ein Client sendet Anfragen (**Requests**) an einen Server, der diese bearbeitet und dann Antworten (**Responses**) zurücksendet. Man unterscheidet zwischen einem REST-Server, der Ressourcen bereitstellt, und REST-Clients, die auf diese Ressourcen zugreifen. Dazu **besitzt jede Ressource eine ID**, die in der Regel **als URI** (Uniform Resource Identifier) modelliert ist.

Zur Adressierung eines REST-Service (auch Ressource genannt) dient eine URI, die aus Server und Port sowie einem Basispfad und einem Pfad der Ressource besteht:

http://server:port/basePath/resourcePath/

URI oder URL?
Heißt es nun URI oder URL?
Eine URL ist das Akronym für „Universal Resource Identifier“.

Eine URL ist eine „Uniform Resource Locator“.
Die Unterscheidung ist eigentlich nur noch von historischer Bedeutung. Alle neuen Spezifikationen verwenden den Begriff „URI“. Damit wird nicht nur eine eindeutige ID festgelegt, sondern auch gleichzeitig die „Adresse“ im Sinne des Protokolls, Server- oder Domainnamens.
Beispiel: http://www.ietf.org/rfc/rfc2396.txt oder
ftp://ftp.is.co.at

Bei einer **RESTful API** handelt es sich um eine Programmierschnittstelle, die HTTP-Anfragen verwendet, um per GET, PUT, POST und DELETE auf Daten zuzugreifen.

Damit wird der darunter registrierte REST-Service angesprochen, der die gewünschte Aktion ausführt. Es werden z. B. neue Datensätze angelegt oder Informationen zu bestehenden Datensätzen abgefragt. Die Kommunikation basiert auf http und stützt sich vor allem auf die vier Operationen POST, GET, PUT und DELETE:

- POST – Erzeugt neue Daten, d.h. eine neue Ressource.
- GET – definiert einen Lesezugriff auf eine oder mehrere Ressourcen. GET ist das „Arbeitspferd“ des World Wide Web. Mehr als 95% aller Interaktionen im WWW haben lesenden Charakter.
- PUT – Verändert eine existierende Ressource. Falls diese noch nicht existiert, kann eine Ressource auch neu erzeugt werden.
- DELETE – löscht eine Ressource.

Weshalb die Bezeichnung REpresentational State Transfer verwendet wird, wird aus dem folgenden Szenario deutlich. Ein Web Browser fordert eine Seite, oder allgemeiner eine Ressource über eine URL an. Ein HTML Dokument, welches eine Repräsentation der Ressource darstellt, wird vom Server zum Client übertragen. Das HTML Dokument kann Links enthalten, die auf weitere Ressourcen im Web verweisen. Navigiert der Client zu einer neuen Seite, so verändert er seinen Zustand, er wechselt oder macht einen Transfer zu einem neuen Zustand durch. Über Repräsentationen wird ein Transfer von einem Status in einen anderen Status durchgeführt.

Frameworks für das Implementieren von REST-Schnittstellen:

Es existieren viele Frameworks.

- Restify Plattform: JavaScript/Node.js mcavage.me/node-restify
- Slim Plattform: PHP www.slimframework.com
- Laravel https://laravel.com/ ein mächtiges PHP-Framework
- Lumen Micro-Framework, von Laravel abstammend https://lumen.laravel.com/