Learning Objectives: REST APIs and HTTP methods

Learners will be able to...

- **enumerate the common verbs of RESTful APIs: Create, Read, Update, Delete and match with the HTTP methods: GET, POST, PUT, PATCH, DELETE**
- **identify which methods are generally safe to retry**

info

Make Sure You Know

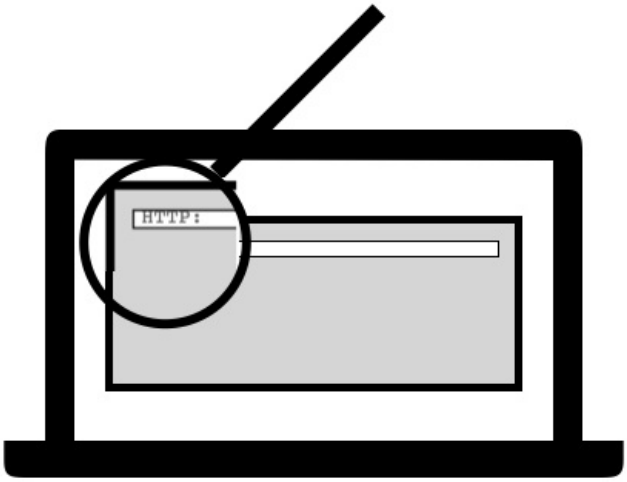
Students should be familiar with the Javascript programming language

Limitations

We will not cover basic programming skills.

HTTP Request Methods

As we briefly discussed in the previous assignment, Hypertext Transfer Protocol (HTTP) enables communication between clients and servers. The **HTTP Protocol** provides a set of **request** methods that are used to specify the action required. The method names are case-sensitive, they must be specified in uppercase.



The methods also share a set of common features:
- **safe** - a request is considered safe if it does not alter the server. All **read-only** requests are safe. Another term that is used for these safe requests is **immutable** which means that they do not change data.
- **idempotent** - each successive request has the same effect on the server as the previous request. That means if there are multiple of the exact same request, only the first one changes the data. All **safe** requests are by definition **idempotent**. The **DELETE** method is **idempotent**, the first time you call it, the item will be deleted but subsequent calls with the same item to delete will not change the server. The **PUT** and **PATCH** requests are also implemented as **idempotent**.
- **cacheable** - a **cacheable** response is one that can be stored and used later. The methods **GET** and **HEAD** return responses that can be cacheable.

The HTTP Request methods

These are sent from the client to the server. You can find more information about each request here.

Name	Action

GET	Used to retrieve the specified resource.
HEAD	Similar to **GET** but does not include the response body.
POST	Sends data to the server.
PUT	Can create a new resource or modifies an existing resource.
DELETE	Deletes the specified resource.
CONNECT	Creates a two way connection with the requested resource.
OPTIONS	Returns available communication options for the specified URL or server.
TRACE	Can be used for debugging, often returns the message that was sent.
PATCH	A **PATCH** request contains instructions for how to modify a resource.

The **REST APIs** we are covering in this assignment mostly use the **GET**, **POST**, **PUT**, **DELETE**, and **PATCH** requests.

What is a REST API?

As discussed in the previous assignment, an API provides a way for one program to communicate with another. Representational State Transfer (REST) API or RESTful API is an architecture that allows for large-scale reliable communication over the Internet using the HTTP methods explained on the previous page.

Main features of the Restful APIs

A Uniform Interface

The uniform interface applies to the way information is requested, the requests look the same whether they come from a computer or a mobile phone. There are four constraints that govern this interface.
1. The request identifies the resource using a **URI - Uniform Resource Identifier**. More information about URIs in the previous assignment.
1. Clients receive enough information about a resource to be able to modify or delete it. The server sends metadata to fulfill this requirement.
1. The server provides information to the client about how to process the message, for example information about the media type.
1. The server sends the client information about other required related resources using hyperlinks.

Stateless System

Each client request to the server is stateless, no information about a request is stored by the server to be used in a later request. Requests can be made in any order, they are not dependent on a previous one having been completed.

Layered system

A layered system refers to the fact that a request can be relayed through multiple authorized intermediaries and still receive a response from the server. An example of this would be an authentication layer between the client and the server.

Supports Caching

To improve server response time, data such as images, can be cached on the client or an intermediary. This saves the download time for an image that is used multiple times, for example a logo that might appear on

multiple pages. The RESTful API provides the means to identify a resource as cacheable or noncacheable, only resources that don't change should be cached.

Code on Demand

The server can send code to be executed on the client side to extend or customize functionality. For example, code that can error check data that is entered on the client side before it is sent to the server.

Advantages of a RESTful API

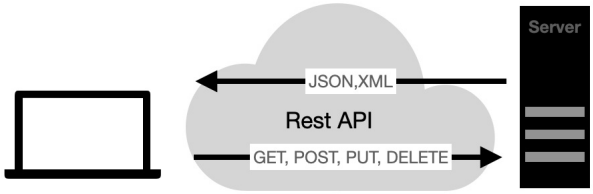
Scalability is one of the benefits you reap when using the REST API. The well implemented caching scheme reduces the number of client-server interactions and statelessness reduces server load because state data does not need to be saved and the server is not waiting for sequential requests. The server's response is independent of the way the data is stored on the server. For example, data may be returned in JSON or XML format, the response headers include information about the format of the response.

Restful APIs

CRUD - Create, Read, Update and Delete

The CRUD acronym represents the four basic functions an application needs to perform. In a RESTful environment they are performed using the following HTTP methods:

You **create** a new record using a **POST** request
You **read** a record using a **GET** request
To **update** a record you use a **POST** or **PUT** request.
To **delete** a record you use a **POST** or **DELETE** request.



For example if you were a gift shop owner and you decided to sell a new item you would **create** a new record with the item name, the cost, and the number of items in stock. You would use **read** if you wanted to check how many items were in stock. You would **update** the record after a sale to change the inventory value. If you decided to no longer carry an item you would **delete** the record for the item.

You can use the Javascript Fetch API to do these things. On the following two pages you will find examples.

API Sandbox - Time API

In the panel on the left you can view all the APIs available for the Time API application.

Try these things out:

- Click on the first **Get** `/api/Time/current/zone`, an API that will get you the current time for a time zone. If you click on the **Try it out** button you can enter a time zone ID from the leftmost column in this list. Then click **Execute** and it will show you the curl command you can use, the URL, the Response body and more.
- Click on the **Post** `/api/Conversion/Translate`, an API that translates a date into a specified language. Click the **Try it out** button and edit the value to the right of the `languageCode`, you can find a list of IANA language codes here. Click **Execute** and view the return data. Try entering an illegal code and see what happens.

Access public REST APIs

Getting information using Javascript

In the upper left-hand panel you can see the HTML file with the embedded Javascript code that will query a web API that **estimates a person's age** based on their **name**.

This is done by using the Javascript `fetch` command.

Try it out:

Type a name into the prompt box on the bottom left, the results are returned in **JSON** format.

Post request with data

In this example you use the `fetch` command to **POST** a string to analyze and it returns the result. You are posting this text to a sentiment analysis engine, more information here.

To run it paste the following commands into the terminal window in the lower left:

```
cd sentiment
npm i
nodejs sentiment.js
```

At the prompt, type in a sentence to analyze for sentiment. For example: *I love ice cream*.

To run it again paste the following at the terminal prompt:

```
nodejs sentiment.js
```