**API-Development-Tools**

**HTTP API Development Tools**

**Introduction**

This is a collection of useful resources for building RESTful HTTP+JSON APIs. There are a lot of good tools and entire ecosystems out there! It can be overwhelming not knowing what options are available, so you can use this as a reference starting point.

Contributions are most welcome. Categories are also open to suggestions!

**Table of Contents**

* [API Specification Languages](https://sandysanthosh.github.io/API-Development-Tool/#api-specification-languages)
* [API Specification Tools](https://sandysanthosh.github.io/API-Development-Tool/#api-specification-tools)
* [API Specifications](https://sandysanthosh.github.io/API-Development-Tool/#api-specifications)
* [API Frameworks](https://sandysanthosh.github.io/API-Development-Tool/#api-frameworks)
* [API Client Development Tools](https://sandysanthosh.github.io/API-Development-Tool/#api-client-development-tools)
* [API Documentation](https://sandysanthosh.github.io/API-Development-Tool/#api-documentation)
* [API Clients](https://sandysanthosh.github.io/API-Development-Tool/#api-clients)
* [API Debugging and Mocking](https://sandysanthosh.github.io/API-Development-Tool/#api-debugging-and-mocking)
* [API Design Guides](https://sandysanthosh.github.io/API-Development-Tool/#api-design-guides)
* [API Publishing](https://sandysanthosh.github.io/API-Development-Tool/#api-publishing)
* [API Gateways](https://sandysanthosh.github.io/API-Development-Tool/#api-gateways)
* [API Security](https://sandysanthosh.github.io/API-Development-Tool/#api-security)
* [API Monitoring](https://sandysanthosh.github.io/API-Development-Tool/#api-monitoring)
* [API Testing](https://sandysanthosh.github.io/API-Development-Tool/#api-testing)
* [API Developer Portal](https://sandysanthosh.github.io/API-Development-Tool/#api-developer-portal)
* [JSON Format Standards](https://sandysanthosh.github.io/API-Development-Tool/#json-format-standards)
* [Learning Resources](https://sandysanthosh.github.io/API-Development-Tool/#learning-resources)
* [Blogs](https://sandysanthosh.github.io/API-Development-Tool/#blogs)
* [References](https://sandysanthosh.github.io/API-Development-Tool/#references)

**API Specification Languages**

* [API Blueprint](https://github.com/apiaryio/api-blueprint)
* [JSON Schema](http://json-schema.org/)
* [RAML](https://raml.org/)
* [OpenAPI (formerly known as Swagger)](https://github.com/OAI/OpenAPI-Specification)

**API Specification Tools**

* [Swagger Inspector](https://swagger.io/swagger-inspector/): Test and auto-generate OpenAPI documentation for any API.
* [Swagger Editor](http://editor.swagger.io/): An editor for designing Swagger specifications.
* [Swagger Tools and Integrations](https://swagger.io/open-source-integrations/): A list of libraries and frameworks serving the Swagger ecosystem.
* [OpenAPI Spec Tooling](http://definitions.apievangelist.com/#Tools): A list of libraries and frameworks serving the OpenAPI ecosystem.
* [API Studio](http://apistudio.io/). Write, mock, and share your Swagger specifications online.
* [Dredd](https://github.com/apiaryio/dredd): Validate API documentation written in API Blueprint against its backend implementation.
* [Restlet Studio](https://restlet.com/modules/studio/): Web IDE for API Design.
* [API Spec Converter](https://lucybot-inc.github.io/api-spec-converter/): Convert between different API spec formats.
* [Prism](http://stoplight.io/platform/prism/): Supercharge any OAS file with mocking, transformations, validations, and more.
* [Apimatic](https://www.apimatic.io/): Supports API description formats including Swagger, OAI format, RAML, API Blueprint, IO Docs, WADL, Postman Collections and HAR 1.4 and more
* [Mulesoft Anypoint](https://anypoint.mulesoft.com/): Design and publish enterprise-grade APIs using RAML
* [Sandbox](https://getsandbox.com/): Quick and easy mock RESTful API from API definitions
* [Restunited](https://restunited.com/): Generate SDK, Documentation with Testing and Debugging

**API Specifications**

* [API Commons](http://apicommons.org/): A repository of language-agnostic API specifications / Data Models.
* [APIS.guru](https://apis.guru/openapi-directory/): Directory of API specs in OpenAPI(aka Swagger) 2.0 format.
* [AnyAPI](https://any-api.com/): Documentation and Test Consoles for Public APIs.

**API Frameworks**

**Ruby**

* [rails-api](https://github.com/rails-api/rails-api): Rails for API only applications.
* [pliny](https://github.com/interagent/pliny): Opinionated template Sinatra app for writing APIs in Ruby.
* [grape](https://github.com/ruby-grape/grape): An opinionated micro-framework for creating REST-like APIs in Ruby.
* [ActiveModel::Serializer](https://github.com/rails-api/active_model_serializers): Brings convention over configuration to your JSON generation.
* [rabl](https://github.com/nesquena/rabl): Generate JSON and XML from any ruby object.
* [jbuilder](https://github.com/rails/jbuilder): Create JSON structures via a Builder-style DSL.
* [roar](https://github.com/trailblazer/roar): Parse and render REST API documents using representers.

**Python**

* [Django REST framework](http://www.django-rest-framework.org/): Toolkit that makes it easy to build Web APIs.
* [Tastypie](https://github.com/django-tastypie/django-tastypie): Webservice API framework for Django.
* [restless](https://github.com/toastdriven/restless): A lightweight REST miniframework for Python.
* [flask-restful](https://github.com/flask-restful/flask-restful): Simple framework for creating REST APIs.
* [Falcon](https://github.com/falconry/falcon): Falcon is a low-level, high-performance Python framework for building HTTP APIs, app backends, and higher-level frameworks.
* [Connexion](https://github.com/zalando/connexion): Swagger/OpenAPI First framework for Python on top of Flask with automatic endpoint validation and OAuth2 support
* [apistar](https://github.com/encode/apistar): A smart Web API framework, designed for Python3.
* [sanic](https://github.com/channelcat/sanic): Sanic is a Flask-like Python 3.5+ web server that’s written to go fast.
* [hug](https://github.com/timothycrosley/hug): hug aims to make developing Python driven APIs as simple as possible, but no simpler.

**Javascript**

* [hapi.js](https://hapijs.com/): Web and services application framework for Node.js.
* [Restify](https://github.com/restify/node-restify): Node.js REST framework specifically meant for web service APIs.
* [Express](https://expressjs.com/): Fast, unopinionated, minimalist web framework for Node.js.
* [sailsjs](http://sailsjs.org/): Realtime MVC Framework for Node.js.
* [Actionhero](https://www.actionherojs.com/): Multi-transport Node.js API server with integrated cluster capabilities and delayed tasks.
* [Baucis](https://github.com/wprl/baucis): To build
* [Koa](http://koajs.com/): Next generation web framework for Node.js
* [Loopback](http://loopback.io/): Node.js framework for creating APIs and easily connecting to backend data sources.
* [Seneca](http://senecajs.org/): A microservices toolkit for Node.js.
* [Feathers](https://feathersjs.com/): Build RESTful and real-time APIs through Socket.io or Primus.
* [Deployd](https://github.com/deployd/deployd): Deployd is the simplest way to build realtime APIs for web and mobile apps
* [Nest](https://github.com/kamilmysliwiec/nest): A modern node.js framework for efficient and scalable web applications built on top of TypeScript

**Go**

* [Go-Json-Rest](https://github.com/ant0ine/go-json-rest): Thin layer on top of net/http that helps building RESTful APIs easily
* [gocrud](https://github.com/manishrjain/gocrud): Go library to simplify creating, updating and deleting arbitrary depth structured data — to make building REST services fast and easy.
* [sleepy](https://github.com/dougblack/sleepy): RESTful micro-framework written in Go.
* [restit](https://github.com/go-restit/restit): Go micro framework to help writing RESTful API integration test.
* [go-relax](https://github.com/codehack/go-relax): Framework of pluggable components to build RESTful API’s.
* [go-rest](https://github.com/ungerik/go-rest): Small and evil REST framework for Go.
* [go-restful](https://github.com/emicklei/go-restful): A declarative highly readable framework for building restful API’s.
* [Goat](https://github.com/bahlo/goat): Minimalistic REST API server in Go.
* [Resoursea](https://github.com/resoursea/api): REST framework for quickly writing resource based services.
* [Zerver](https://github.com/cosiner/zerver): Zerver is a expressive, modular, feature completed RESTful framework.

**Scala**

* [Colossus](https://github.com/tumblr/colossus): I/O and microservice library for Scala.
* [Finatra](https://twitter.github.io/finatra/): Fast, testable, Scala HTTP services built on Twitter-Server and Finagle.
* [Play](https://www.playframework.com/): The high velocity web framework for Java and Scala.
* [Scalatra](http://www.scalatra.org/): Simple, accessible and free web micro-framework.
* [Skinny Micro](https://github.com/skinny-framework/skinny-micro): Micro-web framework to build servlet applications in Scala.
* [Spray](http://spray.io/): Open-source toolkit for building REST/HTTP-based integration layers on top of Scala and Akka.
* [Akka HTTP](https://github.com/akka/akka-http): The Akka HTTP modules implement a full server- and client-side HTTP stack on top of akka-actor and akka-stream.
* [Swagger Akka HTTP](https://github.com/swagger-akka-http/swagger-akka-http): Swagger-Akka-Http brings Swagger support for Akka-Http Apis.

**Java**

* [Rest.li](http://rest.li/): REST framework using type-safe bindings and asynchronous, non-blocking IO.
* [Dropwizard](http://www.dropwizard.io/1.0.0/docs/): Framework for developing ops-friendly, high-performance, RESTful web services.
* [Jersey](https://jersey.java.net/): RESTful web services in Java.
* [Spring Boot](https://projects.spring.io/spring-boot/): RESTful Web Service using Spring, high-performance and little configuration needed.
* [Metamug Mason](https://github.com/metamug/mason): Create REST APIs with JSP tags and SQL. Edit and hot deploy REST resources on the server.

**Haskell**

* [Scotty](https://github.com/scotty-web/scotty): Micro web framework inspired by Ruby’s Sinatra, using WAI and Warp.
* [Spock](https://github.com/agrafix/Spock): Another Haskell web framework for rapid development.
* [Servant](https://github.com/haskell-servant/servant): A Type-Level Web DSL.
* [Yesod](https://github.com/yesodweb/yesod): The Haskell RESTful web framework.

**Elixir**

* [Phoenix](http://phoenixframework.org/): Framework for building HTML5 apps, API backends and distributed systems.
* [Plug](https://github.com/elixir-plug/plug): A specification and conveniences for composable modules between web applications.

**Erlang**

* [Cowboy](https://github.com/ninenines/cowboy): Small, fast, modular HTTP server written in Erlang.
* [Gen Microservice](https://github.com/videlalvaro/gen_microservice): This library solves the problem of implementing microservices with Erlang.
* [Mochiweb](https://github.com/mochi/mochiweb): Erlang library for building lightweight HTTP servers.

**Postgres**

* [PostgREST](https://github.com/begriffs/postgrest): Serve a RESTful API from any existing PostgreSQL database.
* [pREST](https://github.com/prest/prest): pREST is a way to serve a RESTful API from any databases written in Go.

**MySQL**

* [xmysql](https://github.com/o1lab/xmysql): Generate REST APIs for any MySQL Database.

**PHP**

* [API Platform](https://github.com/api-platform/api-platform): API framework on top of Symfony with JSON-LD, Schema.org and Hydra support
* [Dingo API](https://github.com/dingo/api): A RESTful API package for the Laravel and Lumen frameworks
* [Fractal](https://github.com/thephpleague/fractal): Fractal provides a presentation and transformation layer for complex data output, the like found in RESTful APIs, and works really well with JSON
* [Yii2 Framework](https://github.com/yiisoft/yii2): Provides a whole set of tools to simplify the task of implementing RESTful Web Service APIs

**Miscellaneous**

* [Dream Factory](https://github.com/dreamfactorysoftware/dreamfactory): Turn any database into an API platform.

**API Client Development Tools**

**General**

* [OpenAPI extension for Microsoft Visual Studio Code](https://marketplace.visualstudio.com/items?itemName=42Crunch.vscode-openapi): Microsoft VS Code IDE has an OpenAPI extension that includes OpenAPI / Swagger file creation, editing, navigation, code snippets, IntelliSense, security static analysis.
* [Swagger CodeGen](https://github.com/swagger-api/swagger-codegen): Generate client libraries automatically from a Swagger-compliant server.
* [sdks.io](https://sdks.io/): Explore Automatically Generated SDKs.
* [AutoRest](https://github.com/Azure/autorest): Generate client libraries for RESTful web services
* [Unirest](http://unirest.io/): Lightweight HTTP-Request client libraries for several languages
* [OpenAPI Generator](https://github.com/openapitools/openapi-generator): A community fork of Swagger Codegen to automatically generate API clients, server stubs and documentation for REST APIs given an OpenAPI/Swagger spec.

**Ruby**

* [Net::HTTP](https://apidock.com/ruby/Net/HTTP): An HTTP client API for Ruby.
* [faraday](https://github.com/lostisland/faraday): Simple, but flexible HTTP client library, with support for multiple backends.
* [rest-client](https://github.com/rest-client/rest-client): Simple HTTP and REST client for Ruby
* [heroics](https://github.com/interagent/heroics): Ruby HTTP client for APIs represented with JSON schema.
* [blanket](https://github.com/inf0rmer/blanket): A Ruby API wrapper.
* [nestful](https://github.com/maccman/nestful): Ruby HTTP/REST client.

**Java**

* [Retrofit](https://square.github.io/retrofit/): A type-safe HTTP client for Android and Java.

**Javascript**

* [Restangular](https://github.com/mgonto/restangular): Restangular is an AngularJS service that simplifies common GET, POST, DELETE, and UPDATE requests with a minimum of client code

**.NET**

* [Refit](https://github.com/paulcbetts/refit): The automatic type-safe REST library for .NET Core, Xamarin and .NET
* [WebAnchor](https://github.com/mattiasnordqvist/Web-Anchor): Web Anchor provides type-safe, testable and flexible access to web resources.

**API Documentation**

* [ReDoc](https://github.com/Rebilly/ReDoc): OpenAPI/Swagger-generated API Reference Documentation.
* [Swagger UI](https://github.com/swagger-api/swagger-ui): Dynamically generate documentation from a Swagger-compliant API.
* [Slate](https://github.com/lord/slate): Static site generated documentation for your API.
* [prmd](https://github.com/interagent/prmd): JSON Schema tooling: scaffold, verify, and generate documentation from JSON Schema documents.
* [Aglio](https://github.com/danielgtaylor/aglio): An API Blueprint renderer with theme support that outputs static HTML.
* [Apiary](https://apiary.io/): Collaborative design, instant API mock, generated documentation, integrated code samples, debugging and automated testing.
* [Readme](https://readme.io/): API Documentation Hosting.
* [Embed curl](https://www.embedcurl.com/): Embeddable curl commands on the web.
* [Gelato](https://gelato.io/): Create developer portals for your API.
* [API Docs](https://api-docs.io/): Hosted public API documentation for OAS (Swagger) and RAML specs.
* [Docula](https://docu.la/): Documentation using Github and OpenAPI standards.
* [Docbox](https://github.com/tmcw/docbox): REST API documentation generator, using Markdown.
* [widdershins](https://github.com/Mermade/widdershins): REST API documentation generator from OpenAPI 3.0 / Swagger 2.0 / AsyncAPI 1.x / Semoasa 0.1.0 definition

**API Clients**

**Hosted**

* [ExtendsClass](https://extendsclass.com/rest-client-online.html): Web-based HTTP client.
* [JSON Generator](http://www.json-generator.com/): Generate and host mock JSON data.

**Desktop**

* [Firecamp](https://firecamp.app/): API Studio for WebSocket, Rest API and GraphQL.
* [HTTPie](https://httpie.org/): Command line HTTP client.
* [HttpMaster](https://www.httpmaster.net/): Desktop tool for REST API testing.
* [Paw](https://paw.cloud/): REST client for Mac.
* [Postman](https://www.getpostman.com/): Desktop API testing tool.
* [Insomnia](https://insomnia.rest/): REST API client for Mac, Windows, and Linux.
* [Jsonium](http://jsonium.org/): Free REST client for JSON over HTTP protocols
* [I’m only Resting](http://www.swensensoftware.com/im-only-resting): Windows GUI tool for REST API testing

**API Debugging and Mocking**

**Hosted**

* [Beeceptor](https://beeceptor.com/): An HTTP-proxy for rest APIs - inspect and build mock APIs.
* [MockBin](https://mockbin.com/): Generate mock HTTP endpoints.
* [RequestBin](https://requestb.in/): Inspect and debug webhook POST requests.
* [httpbin](http://httpbin.org/): Templated responses for testing various scenarios for HTTP requests.

**Desktop**

* [Json-Server](https://github.com/typicode/json-server) Full fake REST API with zero coding.
* [Mockoon](https://mockoon.com/): Desktop API mocking tool.
* [Postman](https://www.getpostman.com/docs/postman/mock_servers/setting_up_mock): Desktop API client and mocking tool.

**API Design Guides**

* [Google API Design Guide](https://cloud.google.com/apis/design/)
* [PayPal API Style Guide](https://github.com/paypal/api-standards/blob/master/api-style-guide.md)
* [Heroku Platform HTTP API Design Guide](https://github.com/interagent/http-api-design)
* [Haufe API Style Guide](http://work.haufegroup.io/api-style-guide/)
* [Microsoft REST API Guidelines](https://github.com/Microsoft/api-guidelines/blob/master/Guidelines.md)
* [18F API Standards](https://github.com/18f/api-standards)
* [The RESTed NARWHL](https://www.narwhl.com/)
* [White House Web API Standards](https://github.com/whitehouse/api-standards)
* [Zalando REST API Guidelines](https://zalando.github.io/restful-api-guidelines/)
* [API Stylebook Design Guidelines](http://apistylebook.com/design/guidelines/)
* [API Stylebook Design Topics](http://apistylebook.com/design/topics/)
* [Adidas-group API Design Guide](https://github.com/adidas-group/api-guidelines)
* [Azure API Design](https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design)

**API Publishing**

* [Mashape](https://www.mashape.com/): API Marketplace.

**API Gateways**

* [Ambassador API Gateway](https://www.getambassador.io/): Ambassador is a specialized control plane that translates Kubernetes annotations to Envoy configuration. All traffic is directly handled by the high-performance Envoy Proxy.
* [AWS API Gateway](https://aws.amazon.com/api-gateway/): Traffic management, authorization and access control, monitoring, and API version management.
* [API Umbrella](https://apiumbrella.io/): Proxy that sits in front of your APIs.
* [APIAxle](https://github.com/apiaxle/apiaxle/): Proxy that sits in front of your APIs.
* [APIGrove](https://apigrove.github.io/apigrove/): API manager built in Java on top of Fuse ESB.
* [Apigee127](https://github.com/apigee-127/a127-documentation/wiki/What-is-Apigee-127): nodejs based API Gateway
* [Pushpin](http://pushpin.org/): Proxy for both request/response or streaming (long poll) of responses
* [Strongloop](https://github.com/strongloop/microgateway): nodejs based API Gateway
* [Fusio](http://www.fusio-project.org/): PHP based open source API management platform
* [Camel](https://camel.apache.org/): Empowers you to define routing and mediation rules in a variety of domain-specific languages, including a Java-based fluent API, Spring or Blueprint XML configuration files, and a Scala DSL.
* [HAProxy](http://www.haproxy.org/): Reliable, high Performance TCP/HTTP load balancer.
* [OpenResty](https://openresty.org/): Fast web application server built on top of Nginx.
* [Tengine](http://tengine.taobao.org/): A distribution of Nginx with some advanced features.
* [Tyk](https://tyk.io/): Open-source, fast and scalable API gateway, portal and API management platform.
* [Vulcand](https://github.com/vulcand/vulcand): Programmatic load balancer backed by Etcd.
* [Zuul](https://github.com/Netflix/zuul): An edge service that provides dynamic routing, monitoring, resiliency, security, and more.
* [Kong](https://getkong.org/): An open-source management layer for APIs, delivering high performance and reliability.
* [Janus](https://github.com/hellofresh/janus): A lightweight API Gateway written in Go by [Hello Fresh](https://engineering.hellofresh.com/).
* [fabio](https://github.com/fabiolb/fabio): A fast, modern, zero-conf load balancing HTTP(S) router for deploying microservices managed by [consul](https://www.consul.io/) by eBay.
* [Traefik](https://github.com/containous/traefik): Træfik (pronounced like traffic) is a modern HTTP reverse proxy and load balancer written in Go.
* [Oathkeeper](https://github.com/ory/oathkeeper): OIdentity & Access Proxy (IAP) that authorizes HTTP requests based on sets of rules. Integrates with ORY Hydra.

**API Security**

* [Online OpenAPI/Swagger File Security Audit](https://apisecurity.io/tools/audit/): Free online static analysis of API contract files. Upload the file and get the report.
* [API Security checklist](https://github.com/shieldfy/API-Security-Checklist): Checklist of the most important security countermeasures when designing, testing, and releasing your API.
* [Ory Hydra](https://github.com/ory/hydra): OAuth2 server with OpenID Connect written in Go.
* [Approov](https://approov.io/): Mobile App Attestation to ban API abuse by verifying the software that is accessing your API. User authentication is not enough!
* [42Crunch](https://42crunch.com/): API Security platform that includes static analysis, dynamic testing (scan), and runtime protection (API firewall.)

**API Monitoring**

* [Runscope](https://www.runscope.com/): API Performance Monitoring.
* [Ping-API](https://ping-api.com/): Automated API Testing.

**API Testing**

* [Assertible](https://assertible.com/): Continuously test and monitor your APIs after deployments and across environments.
* [Pyresttest](https://github.com/svanoort/pyresttest): YAML based REST testing and API microbenchmarking tool
* [OWASP Zaproxy](https://github.com/zaproxy/zaproxy): A tool to test your API for known security vulnerabilities, with a great CI integration.
* [vREST NG](https://ng.vrest.io/): An enterprise ready application for automated API Testing. It is very simple and intuitive application.

**API Developer Portal**

* [Tyk](https://tyk.io/features): API Developer Portal on top of API gateway, make your API gateway easier to be used by developers.
* [APIMATIC](https://apimatic.io/developer-experience-portal): Instantly build an API Portal with SDKs, Live Code Samples, Test Cases, API Transformation and language specific Docs & Reference - tailored for your API.
* [DWOLLA](https://developers.dwolla.com/): Simple and powerful API for ACH transfers.
* [Fusio](https://www.fusio-project.org/): Fusio is an open source API management platform which helps to build and manage REST APIs. Fusio provides all tools to quickly build an API from different data sources yet it is possible to create complete customized responses.
* [Gravitee](https://gravitee.io/): Open source API platform, unleash the power of your APIs.
* [Wicked Haufe](https://wicked.haufe.io/): Open Source API Management. A full stack open source solution to API Management, based on Mashape’s Kong, node.js, and docker.

**JSON Format Standards**

* [HAL](http://stateless.co/hal_specification.html)
* [JSONAPI](http://jsonapi.org/faq/)
* [JSON Schema](http://json-schema.org/)
* [Hydra](http://www.hydra-cg.com/)
* [Ion](https://github.com/ionwg/ion-doc)

**Learning Resources**

* [REST in Practice](http://shop.oreilly.com/product/9780596805838.do)
* [Roy Fielding’s dissertation on REST](https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm)
* [Best Practices for Designing a Pragmatic RESTful API](http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api)
* [How to Design a REST API](https://blog.octo.com/en/design-a-rest-api/)
* [Automated API Development](https://yos.io/2016/04/27/automated-api-development/)
* [Nordic APIs](http://nordicapis.com/)
* [Undisturbed REST](https://www.mulesoft.com/sites/default/files/resource-assets/ebook-UndisturbedREST_v1.pdf)
* [Build APIs You Won’t Hate](https://leanpub.com/build-apis-you-wont-hate)
* [Irresistible APIs](https://www.manning.com/books/irresistible-apis)
* [How to build an API](https://apiary.io/how-to-build-api)
* [API University](https://www.programmableweb.com/api-university)
* [RESTful Web Services](http://shop.oreilly.com/product/9780596529260.do)
* [RESTful Web APIs](http://shop.oreilly.com/product/0636920028468.do)
* [The Ten Essentials for Good API Documentation](https://alistapart.com/article/the-ten-essentials-for-good-api-documentation)