

Welcome Introduction to API Design and Development

Slides: <http://brainwashinc.com/docs/APIDesignDevV7.key.pdf>



Hello



About me...

- IBM, Ericsson, Travelocity
- Author, Online Courses
- iOS, AWS -> API
- I ❤ RESTful JSON
- Jag älskar Sverige!

Prerequisites

This course assumes you:

- This course assumes you are familiar with the concepts of server-client relationship and the core needs of data delivery to/from servers/clients.
- This course will cover the common concepts of API Design and development focusing on RESTful JSON API design.

3

Why RESTful JSON?

- Widely used, supported API structure
- Flexible but defined
- Focused on data (not functionality)
- Can be a facade
- Human Readable
- Cross-platform, language, etc.

4

We teach over 400 technology topics



5

You experience our impact on a daily basis!



6



My pledge to you

I will...

- Make this as interesting and interactive as possible
- Ask questions in order to stimulate discussion
- Use whatever resources I have at hand to explain the material
- Try my best to manage verbal responses so that everyone who wants to speak can do so
- Use an on-screen timer for breaks so you know when to be back

7



Objectives

At the end of this course you will be able to:

- Understand the design and structure of a RESTful API
- Be able to define a RESTful JSON spec (OpenAPI)
- Become familiar with tools related to API design/development
- Understand various concepts and approaches to API design
- Understand concepts related to API testing
- Determine a course of action for developing an API

8

Agenda

- Introduction
- JSON
- Key Aspects of REST
- API Design
- OpenAPI API Specification
- Design Concepts
- Richardson Maturity Model
- Testing and Development
- Development Standards
- Contracts
- Design First vs Code First
- Automated Testing
- Versioning
- Security
- API Design Details
- Automated Testing Details

9

How we're going to work together

- Discussion
- Demo
- Do!

10



Introduction to API Design and Development

Bear Cahill ~ bear@brainwashinc.com

Copyright Brainwash Inc. 2022

11

Understanding APIs in Software Delivery: Introduction

Copyright Brainwash Inc. 2022

12

Understanding APIs in Software Delivery: Introduction

- **Install Insomnia**
 - <https://insomnia.rest/download>

The screenshot shows the Insomnia website homepage on the left and the application interface on the right.

Website (Left):

- Header: Insomnia by Kong, 18,478 users.
- Navigation: Products, Docs, Pricing, Plugins, Login, Get Started for Free.
- Main Content:
 - Download Insomnia**
 - Start building, designing, testing better APIs through spec-first development driven by an APIOps CI/CD pipelines.
 - [Download Insomnia for MacOS](#)
 - By downloading and using Insomnia, I agree to the [Privacy Policy](#) and [Terms](#).
 - What's New? Changelog
 - Not your OS? Download for Windows / Ubuntu or See all downloads.
- Copyright Brainwash Inc. 2022

Application (Right):

- Header: DOCUMENT, POST, base_url, Send, JSON, Auth, Query, Header.
- Left Panel (Endpoints):
 - pet
 - GET /pet/{petId}
 - POST /pet/{petId}
 - DELETE /pet/{petId}
 - POST /pet/{petId}/uploadImage
 - PUT /pet
 - GET /pet/findByStatus
 - GET /pet/findByTags
 - store
 - GET /store/inventory
 - GET /store/order/{orderId}
 - DELETE /store/order/{orderId}
 - POST /store/order
 - user
- Right Panel (Request/Response):

```
POST /pet/{petId} [Set] Send
JSON
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "dogple",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "string"
}
beautify JSON
```

Understanding APIs in Software Delivery: Introduction

- **Insomnia**
 - **Preferences > General**
 - **Uncheck ‘Validate Certificates’**
 - **HTTP Network Proxy**
 - <http://gia.sebank.se:8080/>

The screenshot shows the 'HTTP Network Proxy' configuration dialog.

HTTP Network Proxy

Enable proxy ?

HTTP proxy:

HTTPS proxy:

Copyright Brainwash Inc. 2022

Understanding APIs in Software Delivery: Introduction

- **Optional: Create Github Account**
 - <https://github.com/signup>



Understanding APIs in Software Delivery: Introduction

- **Optional: Install Nodejs/npm**
 - <https://nodejs.org/en/download/>

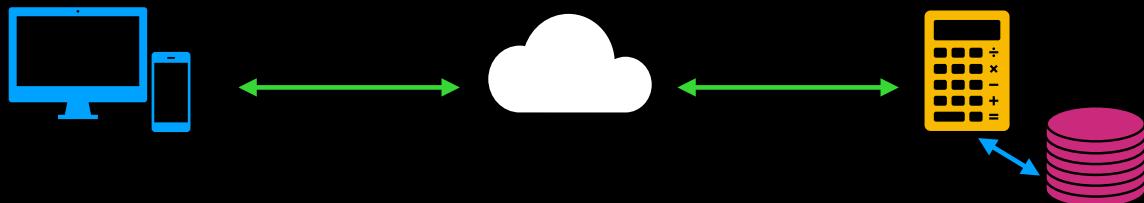


SEB if on VPN:

```
npm config set registry https://repo.sebank.se/artifactory/api/npm/seb-npm/
```

```
npm config set registry https://repo.sebank.se/artifactory/api/npm/seb-npm/
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
```

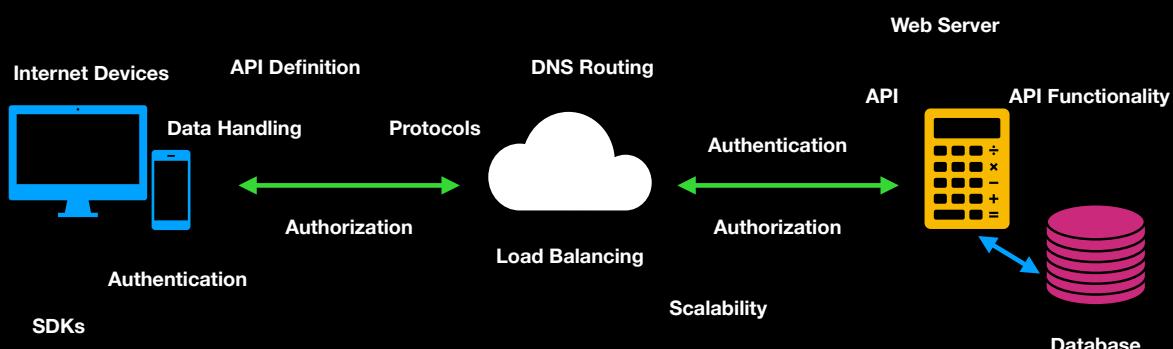
API Design: Introduction



Copyright Brainwash Inc. 2022

17

API Design: Introduction

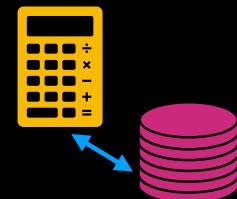


Copyright Brainwash Inc. 2022

18

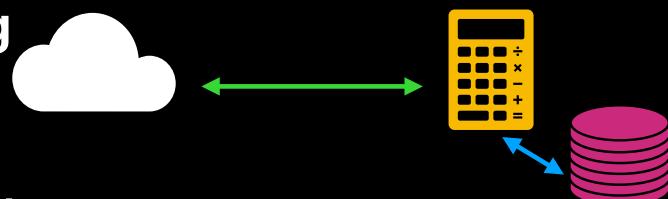
API Design: Introduction

- **API Backend**
 - **Servers (e.g., Apache, Tomcat)**
 - **Web Server Apps**
 - **Processes API Requests**
 - **Functionality**
 - **Java, C#, Python, PHP, etc.**
 - **Data**
 - **Database (e.g., Oracle, MySQL)**



API Design: Introduction

- **API Routing & Security**
 - **DNS**
 - **Load Balancing**
 - **Scalability**
 - **Authentication**
 - **OAuth, Token/JWT**
 - **Authorization**
 - **Permissions**



API Design: Introduction

- **API Client**
 - **Device on Internet**
 - **Understands API**
 - **Endpoints (Nouns, Verbs)**
 - **Data formats (JSON)**
 - **Login/Authentication**
 - **e.g., Token Header**

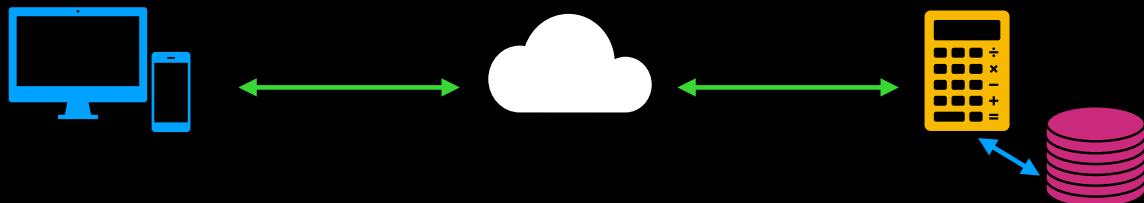


API Design: Introduction



- **Client Communication**
 - **HTTP/s**
 - **JSON, XML, etc.**
 - **API Endpoints**
 - **Provided SDKs**
 - **Generated code**

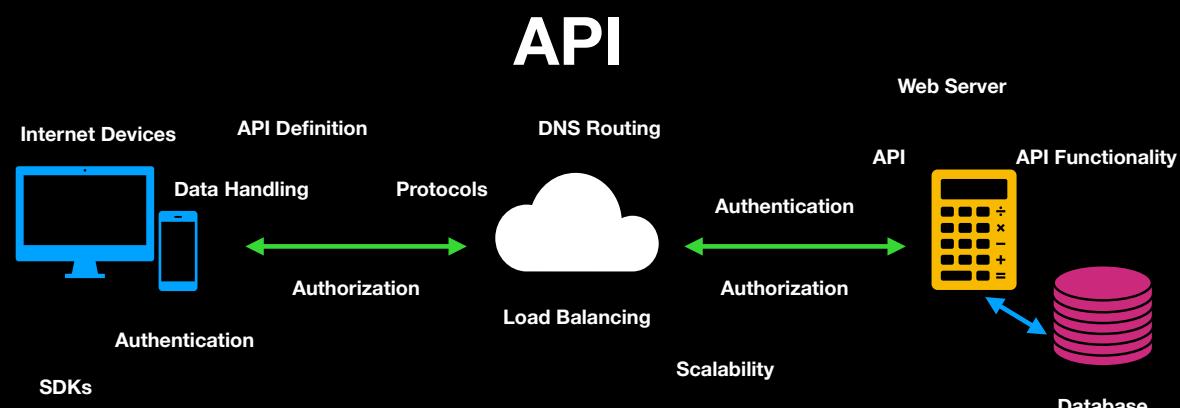
API Design: Introduction



Copyright Brainwash Inc. 2022

23

API Design: Introduction



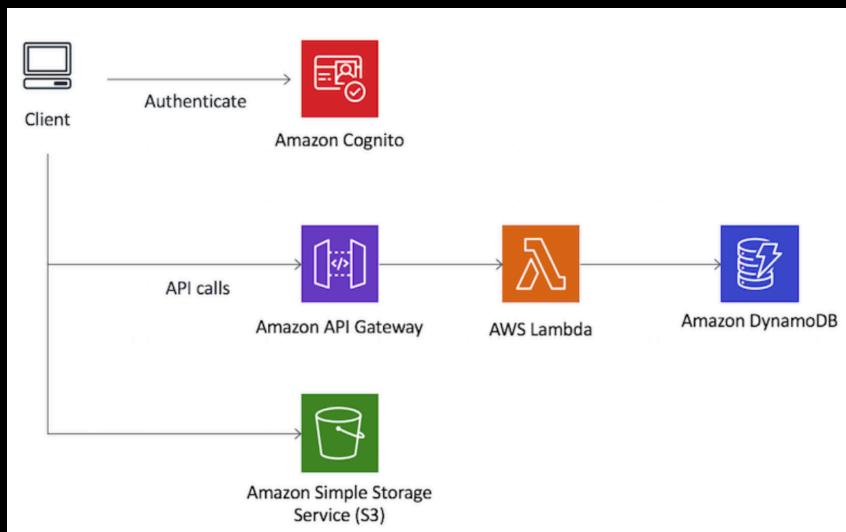
OpenAPI Spec

Copyright Brainwash Inc. 2022

24

API Design: Introduction

- **AWS**



Copyright Brainwash Inc. 2022

25

API Design: Introduction

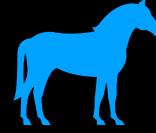


- **Past:**
 - **API Wild West**
 - **URL/Query String, Form Data, etc.**
 - **SOAP/WSDL**
 - **XML, text, etc.**
 - **Everything was proprietary, complex**
 - **Needed a Standard**

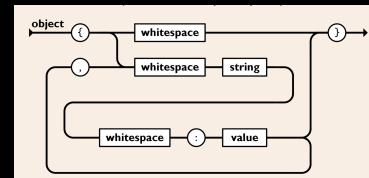
Copyright Brainwash Inc. 2022

26

API Design: Introduction



- **2000:**
 - **Roy Fielding's doctoral dissertation on REST (REpresentational State Transfer)**
- **2001:**
 - **JSON by Douglas Crockford**
 - **JavaScript Object Notation**
 - **[json.org](#)**



API Design: Introduction



- **Early 2000s - Money to be made**
 - **Access data**
 - **Useful functionality, storage**
 - **eBay, Flickr, Amazon, Twitter, etc.**
 - **The standard is key**



- REST (or other) API is expected now
- Libraries/frameworks/SDKs abound
- Variations cause problems
- Downside: better solutions will suffer from inertia



- gRPC - grpc.io
- g = ?
- Remote Procedure Call
- "a client application can directly call a method on a server application on a different machine as if it were a local object"
- GraphQL
 - flexible
 - data efficient
 - Facebook 2012, open-source 2015

gRPC is a modern open source high performance Remote Procedure Call (RPC) framework that can run in any environment. It can efficiently connect services in and across data centers with pluggable support for load balancing, tracing, health checking and authentication. It is also applicable in last mile of distributed computing to connect devices, mobile applications and browsers to backend services.

```
{  
  user(id: 4802170) {  
    id  
    name  
    isViewerFriend  
    profilePicture(size: 50) {  
      url  
      width  
      height  
    }  
    friendConnection(first: 5) {  
      totalCount  
      friends {  
        id  
        name  
      }  
    }  
  }  
}  
  
{  
  "data": {  
    "user": {  
      "id": "4802170",  
      "name": "Zed Gee",  
      "isViewerFriend":  
        "profilePicture":  
          "url": "cdn://  
            "width": 50,  
            "height": 50  
        },  
      "friendConnection":  
        "totalCount":  
          "friends": {  
            "id": "305",  
            "name": "S",  
            "id": "310",  
            "name": "S"  
          }  
    }  
}
```

API Design: Introduction



- **Enter: OpenAPI**
- **Standard**
- **Versioned (e.g., 3.0.1)**
- **<https://swagger.io/specification/>**
- **API OpenAPI Definition**
- **Provides details of implementation**
- **Contract/Standard**
- **Versioned (e.g., 1.1.3)**
- **Starts with Info section**

Copyright Brainwash Inc. 2022

31

OpenAPI Object
This is the root document object of the API.

Fixed Fields

Field Name	Type
openapi	string
info	Info Object
servers	[Server Object]
paths	Paths Object
components	Components Object
security	[Security Requirement Object]
tags	[Tag Object]
externalDocs	External Documentation Object

API Design: Documenting



- **Open API**
- **More doc/spec:**
- **<https://github.com/OAI/OpenAPI-Specification/tree/main/versions>**
- **Various vers**

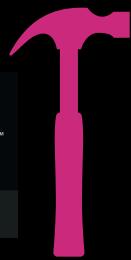
Components Object
Holds a set of reusable objects for different aspects of the OAS. All objects define API unless they are explicitly referenced from properties outside the component.

Fixed Fields

Field Name	Type
schemas	Map[string , Schema Object Reference Object]
responses	Map[string , Response Object Reference Object]
parameters	Map[string , Parameter Object Reference Object]
examples	Map[string , Example Object Reference Object]
requestBodies	Map[string , Request Body Object Reference Object]
headers	Map[string , Header Object Reference Object]

Copyright Brainwash Inc. 2022

32



- **Editor: Info**
- editor.swagger.io
- **Start with version**
 - **openapi: 3.0.1**
- **Notice Errors**
- **Insert > Info**

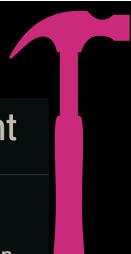
 **Swagger Editor™**
Supported by SMARTBEAR

1 **openapi: 3.0.1**

Errors

Structural error at
should have required property 'info'
missingProperty: info
[Jump to line 0](#)

Structural error at
should have required property 'paths'
missingProperty: paths
[Jump to line 0](#)

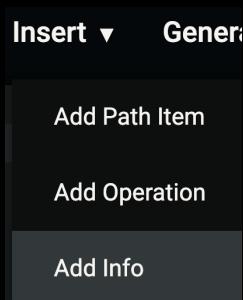


- **Editor**
- **Insert > Info**

- **Enter text in fields**
- **License/Contact**

optional

Click 'Add Info'



Add Info to Document

Title *

REQUIRED. The title of the application.

My API

Description

A short description of the application. Com

Best API ever

Version *

REQUIRED. The version of the OpenAPI d

1.0

Terms of Service

A URL to the Terms of Service for the API. I

<https://www.example.com/>



- **Text inserted**

```

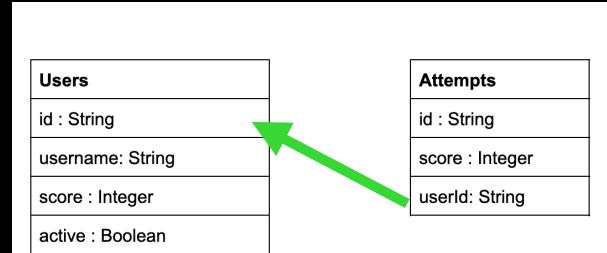
1  openapi: 3.0.1
2  info:
3    title: My API
4    version: '1.0'
5    description: Best API ever
6    termsOfService: https://www.example.com/
7    contact:
8      email: bear@brainwashinc.com
9      name: Bear Cahill
10     url: http://www.example.com/contact
11    license:
12      name: My License
13      url: http://www.example.com/license

```

- **End of lab**



- **Consider the data represented**
- **Fetch users**
 - **.../users**
- **Fetch 1 user**
 - **.../users/<id>**
- **Attempts too.**
- **User's attempts**
 - **.../users/<id>/attempts**
- **What is returned?**



API Design: JSON



- **RESTful JSON**
- **JSON**
 - **JavaScript Object Notation**
 - json.org
 - **Arrays, Dictionaries of values**
 - **Represented in text**
 - **Parseable across platforms, languages, etc.**

```
value
object
array
string
number
"true"
"false"
"null"

object
'{ ws '}'
'{ members '}'

members
member
member ',' members

member
ws string ws ':' element

array
'[' ws ']'
 '[' elements ']'
```

API Design: JSON



- **Data represented in JSON**
- .../users/1234abc

```
{
  "id": "1234abc",
  "username": "bear",
  "score": 55,
  "active": true
}
```

Users
id : String
username: String
score : Integer
active : Boolean

API Design: JSON

- **JSON**
 - **Object name (User) not included**
 - **Arrays []**
 - **Dictionary/Object { }**
 - **Keys are Strings**
 - **Values vary**
 - **Strings, Ints, Bool, Dictionaries, Array**



```
value
object
array
string
number
"true"
"false"
"null"

object
'{ ws '}'
'{ members '}'

members
member
member ',' members

member
ws string ws ':' element

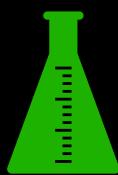
array
'[' ws ']'
 '[' elements ']'
```

API Design: JSON



- **JSON**
 - **RESTful JSON - very common combo**
 - **SDKs often provide parsing**
 - **Sometimes built into languages**
 - **JS, Swift (Codeable)**

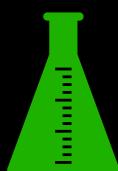
API Design: JSON



- **Lab**
 - <https://jsonlint.com/>
 - **Create an array of Elements**
 - **Each element has 4 members:**
 - **id of type String**
 - **username of type String**
 - **score of type Integer**
 - **active of type Boolean**
 - **See next slide for hints**

Users
id : String
username: String
score : Integer
active : Boolean

API Design: JSON



- **Lab**
 - **json.org has details**
 - **Arrays are within [and]**
 - **Elements are within { and }**
 - **Element members are key-value pairs**
 - "id" : "test"
 - **Members and Array elements are comma separated**
 - **See next slide for example**

API Design: JSON



- **Lab**

```
[ {  
    "id": "1234abc",  
    "username": "bear",  
    "score": 55,  
    "active": true  
}, {  
    "id": "xyz123",  
    "username": "schmeb",  
    "score": 550,  
    "active": false  
}, {  
    "id": "qwerty",  
    "username": "jed",  
    "score": 553,  
    "active": true  
}]
```

- **End of lab**

Copyright Brainwash Inc. 2022

43

API Design: Design



- **Key Concepts**
- **HTTP/s for communication**
- **State is...**
 - **Value(s) of data at a given time**
 - **Cookies, Parameters, etc. represent state**
- **REST Statelessness**
 - **Nothing stored by server (or possibly even sent by client) about state**

Copyright Brainwash Inc. 2022

44



- **Key Aspects**
 - **Nouns - resource based**
 - e.g., User, .../users
 - **Verbs to URIs**
 - e.g., GET, POST
 - **Stateless - each request has it all**
 - **Scaleable - doesn't need same server for next request**



- **Key Aspects**
 - **Nouns - resource based**
 - e.g., Users
 - <domain>/users
 - <http://www.example.com/api/users>
 - Array of User objects (JSON): []
 - <domain>/users/<id>
 - <http://www.example.com/api/users/234>
 - 1 User object: { }

API Design: Design

• Key Aspects

- Nouns - resource based
 - Relationships in URL
 - Attempts relates to Users
 - /users/qwerty/attempts - returns 2 comments

```
{  
  "users": [{  
    "id": "1234abc",  
    "username": "bear",  
    "score": 55,  
    "active": true  
  }, {  
    "id": "xyz123",  
    "username": "schmeb",  
    "score": 550,  
    "active": false  
  }, {  
    "id": "qwerty",  
    "username": "jed",  
    "score": 553,  
    "active": true  
  }],  
  "attempts": [{  
    "id": "a1",  
    "score": 5,  
    "userId": "qwerty"  
  }, {  
    "id": "bb2",  
    "score": 3,  
    "userId": "xyz123"  
  }, {  
    "id": "999",  
    "score": 8,  
    "userId": "qwerty"  
  }]  
}
```

API Design: Design

- Verbs - CRUD
 - POST = Create
 - .../users
 - GET = Read
 - PATCH = Update
 - .../users/1234abc
 - {"score":56}
 - DELETE = Delete

```
{  
  "users": [{  
    "id": "1234abc",  
    "username": "bear",  
    "score": 55,  
    "active": true  
  }, {  
    "id": "xyz123",  
    "username": "schmeb",  
    "score": 550,  
    "active": false  
  }, {  
    "id": "qwerty",  
    "username": "jed",  
    "score": 553,  
    "active": true  
  }],  
  "attempts": [{  
    "id": "a1",  
    "score": 5,  
    "userId": "qwerty"  
  }, {  
    "id": "bb2",  
    "score": 3,  
    "userId": "xyz123"  
  }, {  
    "id": "999",  
    "score": 8,  
    "userId": "qwerty"  
  }]  
}
```

API Design: Design

- **Verbs - CRUD**
 - **GET - read**
 - **.../users - all**
 - **.../users/1 - user with id = 1**
 - **.../attempts - all**
 - **.../attempts/1 - one**
 - **.../users/1/attempts**

```
{
  "users": [
    {
      "id": "1234abc",
      "username": "bear",
      "score": 55,
      "active": true
    },
    {
      "id": "xyz123",
      "username": "schmeh",
      "score": 550,
      "active": false
    },
    {
      "id": "qwerty",
      "username": "jed",
      "score": 553,
      "active": true
    }
  ],
  "attempts": [
    {
      "id": "al",
      "score": 5,
      "userId": "qwerty"
    },
    {
      "id": "bb2",
      "score": 3,
      "userId": "xyz123"
    },
    {
      "id": "999",
      "score": 8,
      "userId": "qwerty"
    }
  ]
}
```

Copyright Brainwash Inc. 2022

49

- **Lab: GET** API Design: Design
- **<https://my-json-server.typicode.com/typicode/demo>**
- **View db.json file**
- **Via browser:**
 - **GET posts, single post, comments, single comment, comments for 1 post**



```
{
  "posts": [
    {
      "id": 1,
      "title": "Po"
    },
    {
      "id": 2,
      "title": "Po"
    },
    {
      "id": 3,
      "title": "Po"
    }
  ],
  "comments": [
    {
      "id": 1,
      "body": "so",
      "postId": 1
    },
    {
      "id": 2,
      "body": "so"
    }
  ]
}
```

How to

1. Create a repository on GitHub (<https://github.com/<your-username>/<your-repo>>)
2. Create a `db.json` file
3. Visit <https://my-json-server.typicode.com/<your-username>/<your-repo>> to access your server

No registration. Nothing to install.



Sponsor

Share

Fork

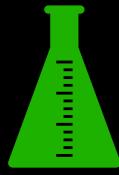
demo

by typicode

Available resources

• posts³

API Design: Design



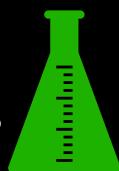
- **Lab: GET - Different db.json**
- **<https://my-json-server.typicode.com/bearc0025/api>**
- **View db.json file**
- **Via browser:**
 - **GET users, 1 user, attempts, 1 attempt, etc.**

```
{
  "users": [
    {
      "id": "1234abc",
      "username": "bear",
      "score": 55,
      "active": true
    },
    {
      "id": "xyz123",
      "username": "schmeb",
      "score": 550,
      "active": false
    },
    {
      "id": "qwerty",
      "username": "jed",
      "score": 553,
      "active": true
    }
  ],
  "attempts": [
    {
      "id": "a1",
      "score": 5,
      "userId": "qwerty"
    }
  ]
}
```

Copyright Brainwash Inc. 2022

51

API Design: Design



- **Lab: OpenAPI Server**
- **[editor.swagger.io > Insert > Add Servers](#)**
- **Fill out form**
- **Click Add Servers (bottom right)**

Insert ▾ Generate Server ▾

Add Path Item
Add Operation
Add Info
Add External Documentation
Add Tag Declarations
Add Tags To Operation
Add Servers

URL *
REQUIRED. A URL to the target host. This URL supports Server relative to the location where the OpenAPI document is being served (brackets).

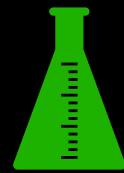
Description
An optional string describing the host designated by the URL. Could be used for documentation.

Copyright Brainwash Inc. 2022

52

API Design: Design

- **Lab: OpenAPI Server**
- **Results**



```
servers:  
  - url: https://my-json-server.typicode.com/bearc0025/api  
    variables: {}  
    description: User API
```

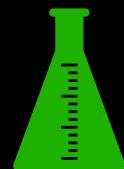
Servers

https://my-json-server.typicode.com/bearc0025/api - User API ▾

- **End of lab**

API Design: Design

- **Lab: OpenAPI Add Path**
- **Insert > Add Path Item**
- **Fill in form**
- **Click Add Path**



Insert ▾ Generate Server ▾

Add Path Item

Add Path

Path *
REQUIRED. The path to add.
/users

Summary
Enter a summary of the path.
user level operations

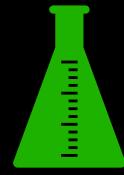
Description
An optional, string description
create and read users

Creates:

```
paths:  
  /users:  
    summary: user level operations  
    description: create and read users
```

API Design: Design

- **Lab: OpenAPI Add Path Method: GET**
- **Insert > Add Operation**
- **Fill in form**
- **Click Add Operation**



Creates:

Insert ▾ Generate Server

Add Path Item

Add Operation

Add Operation to Document

Path *
REQUIRED. The path to add the operation to.
`/users`

Operation *
REQUIRED. Select an operation.
`get`

Summary
Add a short summary of what the operation does.
`fetch users`

Description
A verbose explanation of the operation behavior. Commonly used to describe how an operation works.
`read users from server`

Operation ID
Unique string used to identify the operation. The id MUST be unique across all operations in the API.
`fetchUsers`

Tags
A list of tags for API documentation control. Tags can be used to filter and group operations.
`users`

```
paths:  
/users:  
summary: user level operations  
description: create and read users  
get:  
summary: fetch users  
description: read users from server  
operationId: fetchUsers  
responses:  
default:  
description: Default error sample response  
tags:  
- users
```

Copyright Brainwash Inc. 2022

55

- ## API Design: Design
- **Lab: OpenAPI Add Path Method: GET**
 - **Notice updated documentation**
 - **Execute GET**



users

GET /users fetch users

GET /users fetch users

read users from server

Parameters

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' '\n"https://my-json-server.typicode.com/learncode025/api/users"\n--accept "/*"\n'
```

Request URL

<https://my-json-server.typicode.com/learncode025/api/users>

Server response

Code Details

200

Response body

```
[{"id": "123456c", "username": "test", "score": 500, "active": true}, {"id": "xy123", "username": "test", "score": 500, "active": false}, {"id": "werty", "username": "test", "score": 500, "active": true}]
```

Download

- **End of lab**

Copyright Brainwash Inc. 2022

56

API Design: Design

- **Verbs - Create, Update**
 - **POST = Create**
 - **PATCH = Update**
 - **PUT**
 - **If exists, replace**
 - **Otherwise, create**
 - **Idempotent - multiple requests, same result**

Copyright Brainwash Inc. 2022

57

```
{
  "users": [
    {
      "id": "1234abc",
      "username": "bear",
      "score": 55,
      "active": true
    },
    {
      "id": "xyz123",
      "username": "schmeb",
      "score": 550,
      "active": false
    },
    {
      "id": "qwerty",
      "username": "jed",
      "score": 553,
      "active": true
    }
  ],
  "attempts": [
    {
      "id": "al",
      "score": 5,
      "userId": "qwerty"
    },
    {
      "id": "bb2",
      "score": 3,
      "userId": "xyz123"
    },
    {
      "id": "999",
      "score": 8,
      "userId": "qwerty"
    }
  ]
}
```

API Design: Design

- **Verbs - CRUD**
 - **POST = Create**
 - **GET = Read**
 - **PATCH = Update**
 - **DELETE = Delete**
- **POST/PUT/PATCH**
 - **Overlap**
 - **Require HTTP body**

Copyright Brainwash Inc. 2022

58

```
{
  "users": [
    {
      "id": "1234abc",
      "username": "bear",
      "score": 55,
      "active": true
    },
    {
      "id": "xyz123",
      "username": "schmeb",
      "score": 550,
      "active": false
    },
    {
      "id": "qwerty",
      "username": "jed",
      "score": 553,
      "active": true
    }
  ],
  "attempts": [
    {
      "id": "al",
      "score": 5,
      "userId": "qwerty"
    },
    {
      "id": "bb2",
      "score": 3,
      "userId": "xyz123"
    },
    {
      "id": "999",
      "score": 8,
      "userId": "qwerty"
    }
  ]
}
```

API Design: Design

- **Query String**
 - **Additional parameters:**
 - **Filtering**
 - **Searching**
 - **Paging**
 - **Embedding**
 - **etc.**
 - **.../api/users/1?_embed=attempts**

Copyright Brainwash Inc. 2022

59

```
{
  "users": [
    {
      "id": "1234abc",
      "username": "bear",
      "score": 55,
      "active": true
    },
    {
      "id": "xyz123",
      "username": "schmeb",
      "score": 550,
      "active": false
    },
    {
      "id": "qwerty",
      "username": "jed",
      "score": 553,
      "active": true
    }
  ],
  "attempts": [
    {
      "id": "a1",
      "score": 5,
      "userId": "qwerty"
    },
    {
      "id": "bb2",
      "score": 3,
      "userId": "xyz123"
    },
    {
      "id": "999",
      "score": 8,
      "userId": "qwerty"
    }
  ]
}
```

- **For more... PluralSight**



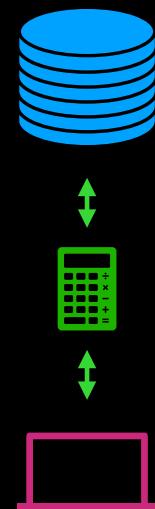
➤ [APIs- Application Programming Interface: Executive Briefing](#)

Dan Appleman

Copyright Brainwash Inc. 2022

60

API Design: Design Concepts

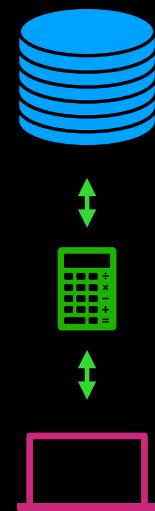


API Design: Design Concepts

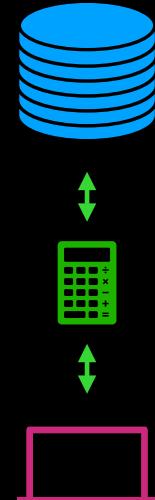
- **Separation of Concerns**
 - **Gathering Data** - e.g., from database
 - **Formatting Data** - e.g., JSON
 - **Delivering Data** - e.g., HTTP, streaming
 - **Security**
 - **Scalability**
 - **Reliability**
 - **Etc.**



- **Separation of Concerns**
 - **Gathering Data**
 - When requested by client
 - **Very important on the server**
 - Good database design
 - Cache if possible
 - Efficient queries



- **Separation of Concerns**
 - **Formatting Data**
 - When sending/returning to client
 - **Important to client**
 - **Directed by server (one-for-all)**



API Design: Design Concepts

- **Separation of Concerns**
 - **Delivering Data**
 - How it is sent: protocol, etc.
 - Important to client
 - Directed by server
 - Usually the same for all clients

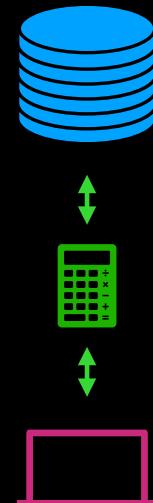
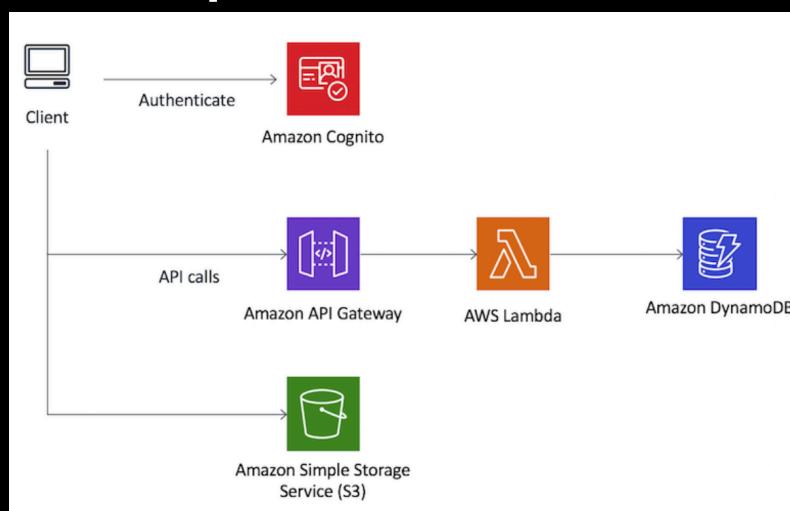


Copyright Brainwash Inc. 2022

65

API Design: Design Concepts

- **AWS - Separation of Concerns**



Copyright Brainwash Inc. 2022

66



- **Operations: Responses**
 - **Default Response**
 - **Specified by Status Code (e.g., 200, 404, etc.)**
- **Includes:**
 - **Description**
 - **Content**
 - **Content Type**
 - **Schema**

```
get:
  summary: fetch users
  description: read users from server
  operationId: fetchUsers
  responses:
    default:
      description: Default error sample response
    tags:
      - users
```



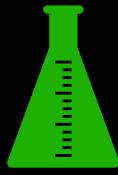
- **For all 200-299 responses**
- **application/json content**
- **Schema is an object**
- **With properties**

Users
id : String
username: String
score : Integer
active : Boolean

```
get:
  summary: fetch users
  description: read users from server
  operationId: fetchUsers
  responses:
    '2XX':
      description: successful fetch
      content:
        application/json:
          schema:
            type: object
            properties:
              id:
                type: string
                example: "abc123"
              username:
                type: string
                example: "bear"
              score:
                type: integer
                example: 55
              active:
                type: boolean
                example: true
```

API Design: Design

- **Lab: OpenAPI Add GET Response**
- **NOTE: We'll make this an array later.**



```
get:  
  summary: fetch users  
  description: read users from server  
  operationId: fetchUsers  
  responses:  
    '2XX':  
      description: successful fetch  
      content:  
        application/json:  
          schema:  
            type: object  
            properties:  
              id:  
                type: string  
                example: "abc123"  
              username:  
                type: string  
                example: "bear"  
              score:  
                type: integer  
                example: 55  
              active:  
                type: boolean  
                example: true
```

Responses

Code	Description
2XX	successful fetch

Media type

application/json

Controls Accept header.

Example Value | Schema

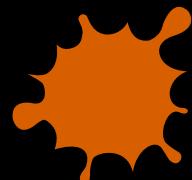
```
{  
  "id": "abc123",  
  "username": "bear",  
  "score": 55,  
  "active": true  
}
```

- **End of lab**

Copyright Brainwash Inc. 2022

69

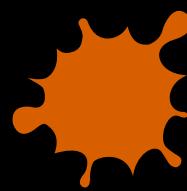
API Design: Design Concepts



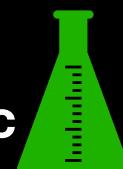
- **Errors**
 - **HTTP**
 - **Response Codes**
 - **1xx: Informational**
 - **2xx: Success**
 - **3xx: Redirection**
 - **4xx: Client Error**
 - **5xx: Server Error**
 - **<https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>**

Copyright Brainwash Inc. 2022

70



- **Errors - Best Practices**
 - "Helpful" information is best
 - **Readable message**
 - **Meaningful codes (http and internal)**
 - **Hints/Details when possible and appropriate**

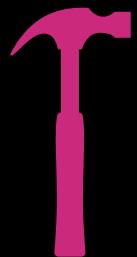


- **Lab**
- **If you have a git account. Create a public repo...**
- **<https://my-json-server.typicode.com/>**
- **Demo data @ <https://my-json-server.typicode.com/bearc0025/api/db>**

How to

1. Create a repository on GitHub (`<your-username>/<your-repo>`)
2. Create a `db.json` file
3. Visit <https://my-json-server.typicode.com/<your-username>/<your-repo>> to access your server

API Design: Design Concepts



- Now that we have two...
- Add Another Server
- [editor.swagger.io](#)
- Insert > Add Servers

The screenshot shows the Swagger Editor interface. On the left, a sidebar menu has 'Insert' selected, and 'Add Servers' is highlighted. The main area is titled 'Add Servers' and contains a 'Server' object definition. It includes fields for 'URL *' (set to <https://my-json-server.typicode.com/bearc0025/api>), 'Description' (set to 'My GIT server'), and 'Server Variables' (a map between variable names and values).

- Add your server or mine:
 - <https://my-json-server.typicode.com/bearc0025/api>

Copyright Brainwash Inc. 2022

73

API Design: Richardson Maturity Model



Copyright Brainwash Inc. 2022

74

API Design: Richardson Maturity Model

- **Breakdown** <https://martinfowler.com/articles/richardsonMaturityModel.html>
- **Four Levels:**
 - **Level 0: Uses HTTP**
 - **Level 1: Resources (nouns)**
 - **Level 2: HTTP Verbs**
 - **Level 3: Hypermedia Controls (HATEOAS = Hypertext As The Engine Of Application State)**



Copyright Brainwash Inc. 2022

75

API Design: Richardson Maturity Model

- **Level 0: Uses HTTP**
 - **Mostly just HTTP plus anything...**
 - **XML, YAML, JSON, etc.**
 - **RPC**
 - **Might be one endpoint for all requests**
 - **Content specifies details for request and response**
 - **And/or GET query string**



Copyright Brainwash Inc. 2022

76

API Design: Richardson Maturity Model

- **Level 1: Resources (nouns)**
 - **Various endpoints**
 - e.g., /posts, /posts/1
 - **Body/GET still specifies details**
 - **Request, Response**
 - **Various formats: XML, etc.**



API Design: Richardson Maturity Model

- **Level 2: HTTP Verbs**
 - **GET**
 - **Same request, same results**
 - **Cacheable**
 - **POST (create)**
 - **201 Created**
 - **409 Conflict**
 - **PUT, PATCH, DELETE, etc.**



API Design: Richardson Maturity Model

- **Level 3: Hypermedia Controls (HATEOAS = Hypertext As The Engine Of Application State)**
 - **What to do next and how**
 - **Can be dynamic**
 - **Client doesn't need knowledge**



```
{  
    "departmentId": 10,  
    "departmentName": "Administration"  
    "locationId": 1700,  
    "managerId": 200,  
    "links": [  
        {  
            "href": "10/employees",  
            "rel": "employees",  
            "type" : "GET"  
        }  
    ]  
}
```

API Design: Richardson Maturity Model

- **HATEOAS**
 - **No official OpenAPI format**
 - **Common practice includes:**
 - **href: URI**
 - **rel: relation type**
 - **type: expected media resource type (e.g., JSON or <https://restfulapi.net/hateoas/>)**
 - **See [RFC5988](#) and Hypertext Application Language (HAL)**



```
{  
    "departmentId": 10,  
    "departmentName": "Administration",  
    "locationId": 1700,  
    "managerId": 200,  
    "links": [  
        {  
            "href": "10/employees",  
            "rel": "employees",  
            "type" : "GET"  
        }  
    ]  
}
```

API Design: Components



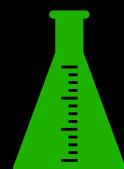
- **Many sections for reuse**
- **Note: schemas**
 - **Can define schemas**
 - **Reference in Operations**
- **Also, responses**
 - **Reference schemas**
 - **Reference in Operations**
- **<https://swagger.io/specification/>**

Copyright Brainwash Inc. 2022

Fixed Fields		
Field Name	Type	Description
schemas	Map[string, Schema Object Reference Object]	An object to hold reusable Schema Objects.
responses	Map[string, Response Object Reference Object]	An object to hold reusable Response Objects.
parameters	Map[string, Parameter Object Reference Object]	An object to hold reusable Parameter Objects.
examples	Map[string, Example Object Reference Object]	An object to hold reusable Example Objects.
requestBodies	Map[string, Request Body Object Reference Object]	An object to hold reusable Request Body Objects.
headers	Map[string, Header Object Reference Object]	An object to hold reusable Header Objects.
securitySchemes	Map[string, Security Scheme Object Reference Object]	An object to hold reusable Security Scheme Objects.
links	Map[string, Link Object Reference Object]	An object to hold reusable Link Objects.
callbacks	Map[string, Callback Object Reference Object]	An object to hold reusable Callback Objects.

81

API Design: Components



- **Create a components section (bottom)**
- **Add a schemas section**
- **Add a schema in it named ‘FullUser’**
- **Copy under “schema” in the ‘2XX’ response**
 - **Fix indenting**
- **Replace ‘2XX’ schema with ref**

```
responses:  
  '2XX':  
    description: successful fetch  
    content:  
      application/json:  
        schema:  
          $ref: '#/components/schemas/FullUser'
```

```
components:  
schemas:  
  FullUser:  
    type: object  
    properties:  
      id:  
        type: string  
        example: "abc123"  
      username:  
        type: string  
        example: "bear"  
      score:  
        type: integer  
        example: 55  
      active:  
        type: boolean  
        example: true
```

- **End of lab**

Copyright Brainwash Inc. 2022

82

API Design: Design

- **Lab**
- **<https://pokeapi.co/>**
- **Fetch data and drill down into more**
- **Try with limit and offset**

The RESTful Pokémon API
Serving over 60,000,000 API calls each month!

All the Pokémon data you'll ever need in one place, easily accessible through a modern RESTful API.

Check out the docs!

Try it now!

https://pokeapi.co/api/v2/pokemon/ditto

Need a hint? Try `pokemon/ditto`, `pokemon/1`, `type/3`, `ability/4`, or `pokemon?limit=100&offset=200`.

Direct link to results: <https://pokeapi.co/api/v2/pokemon/ditto>

Resource for ditto

```
abilities: [ { ability: { name: "limber" }, url: "https://pokeapi.co/api/v2/ability/7/" } ]
```

Copyright Brainwash Inc. 2022

83

API Design: Design

- **Lab**
- **<https://swapi.dev/>**
- **Fetch data and drill down into more**
- **Try with page query string parameter**

<https://swapi.dev/api/> [people?page=5](https://swapi.dev/api/people?page=5)

The Star Wars API
(what happened to swapi.co?)

All the Star Wars data you've ever wanted:
Planets, Spaceships, Vehicles, People, Films and Species
From all SEVEN Star Wars films
Now with The Force Awakens data!

Try it now!

https://swapi.dev/api/people/1/

Need a hint? try `people/1/` or `planets/3/` or `starships/9/`

Result:

```
{ "name": "Luke Skywalker", "height": "172", "mass": "77", "hair_color": "blond", "skin_color": "fair", "eye_color": "brown", "birth_year": "19BBY", "gender": "male", "homeworld": "https://swapi.dev/api/planets/1/", "species": [ "https://swapi.dev/api/species/1/" ], "films": [ "https://swapi.dev/api/films/1/", "https://swapi.dev/api/films/2/", "https://swapi.dev/api/films/3/", "https://swapi.dev/api/films/4/", "https://swapi.dev/api/films/5/", "https://swapi.dev/api/films/6/", "https://swapi.dev/api/films/7/" ], "vehicles": [ "https://swapi.dev/api/starships/1/", "https://swapi.dev/api/starships/2/" ], "starships": [ "https://swapi.dev/api/starships/1/" ], "species": [ "https://swapi.dev/api/species/1/" ] }
```

Copyright Brainwash Inc. 2022

84

API Design: Testing and Development



- **Test**
 - **Tools: ReadyAPI, Postman, Insomnia, StopLight, Paw, etc.**
 - **Mock endpoints**

The screenshot shows the Postman interface with a collection of tests. The 'Tests' tab is selected, showing a green status indicator. Below the tabs are buttons for Authorization, Pre-request Script, Tests (green), and Variables. A note says: "These tests will execute after every request in this collection. [Learn more about Postman's execution order.](#)" The 'Tests' section contains a code snippet:

```
1 pm.test("Status code is 200", function () {  
2     pm.response.to.have.status(200);  
3});
```

Below the code, it says: "Test scripts are written in JavaScript, and are run after the response is received." There is a link: "Learn more about tests scripts". The right side of the interface shows a list of test results for various API endpoints, including GET, DELETE, PATCH, and POST requests, with some failing and some passing.

Copyright Brainwash Inc. 2022

85

API Design: Testing and Development



- **Accuracy/Functionality**
- **Performance - scalability**
- **Errors - fail well**
- **Automate when possible**
 - **Including CI/CD**

The screenshot shows the ReadyAPI interface. At the top, there are tabs for Form, Code, Preview, Mocks, and 14 Problems. The 'Routes' tab is selected, showing a table of routes with columns for Method, Path, and Mock URL. The routes listed are:

Method	Path	Mock URL
GET	/pets	http://127.0.0.1:3106/pets
POST	/pets	http://127.0.0.1:3106/pets
GET	/pets/{petId}	http://127.0.0.1:3106/pets/{petId}
GET	/employee	http://127.0.0.1:3106/employee
POST	/employee	http://127.0.0.1:3106/employee
PUT	/employee	http://127.0.0.1:3106/employee
PATCH	/employee	http://127.0.0.1:3106/employee
DELETE	/employee	http://127.0.0.1:3106/employee

On the right, it says "powered by Prism". Below the routes, there is a 'Logs' tab.

Copyright Brainwash Inc. 2022

86

API Design: Testing and Development



- **Previous steps: Doc, Security, Versioning, etc.**
- **Audience: public, private**
- **Once published, no changes (new vers)**
 - **Maybe not even bug fixes if it changes how the API works**
- **Monitor**
 - **Logs, errors, user input**

API Design: Dev Standards



- **Contract communicates details**
- **Maintaining the contract helps keep all parties consistent/correct**
- **Changes should be driven via the contract**
 - **Centralized update for all**



- **API Spec**
 - **Swagger/OpenAPI**
 - **Endpoints (paths)**
 - **Methods (verbs)**
 - **Requests bodies**
 - **Response bodies**
 - **Model**
 - **Parameters**



- **Benefits**
 - **Source of Truth**
 - **Contract with agreement**
 - **Communication**
 - **Reliable, dependable, predictable**
 - **Otherwise - opposite of above**
- **Remember: An API is meant to be used - don't make it difficult**



- **Richardson Maturity Model**
 - **Use HTTP, resources, verbs and (possibly) Hypermedia**
 - **Security - OAuth, SSL**
 - **Documentation**
 - **Readily available**
 - **Updated**
 - **Versioning - in URL, header, both**



- **Filtering/Sorting/Search/Limiting/Pagination**
 - **Query string parameters**
 - **Route alias for complex/common**
 - `/reports/yesterday`
 - **Embed/expand**
- **Caching**
 - **Custom, ETag (hash version of resource) or Last-Modified**
 - **304 Not Modified response**



- **POST/PUT/PATCH - return resource**
- **JSON**
 - **Including PUT/POST/PATCH http body**
 - **Not key-value: name=bear&ver=1.1**
 - **Content-type: application/json**
- **Camel case vs Snake case**
 - **Camel for JavaScript and similar**
 - **camelCase vs snake_case**



- **HEAD**
- **Identical to GET but no body for response**
- **Meta info in headers per resource**
 - **Last mod, length, content type**
- **Can be used for URL validity tests**
- **Cacheable**



```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Type: text/html; charset=UTF-8
Date: Wed, 08 May 2013 10:12:29 GMT
ETag: "780602-4f6-4db31b2978ec0"
Last-Modified: Thu, 25 Apr 2013 16:13:23 GMT
Content-Length: 1270
```

- **OPTIONS** API Design: Dev Standards
- **API options (Allow/Content-type)**
- **Doesn't initiate data fetch/storage**
- **May apply to server or specific resource**
- **Non-Cacheable**
- **It's not a true resource (GET)**
- **<https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>**



```
HTTP/1.1 200 OK
Allow: GET,HEAD,POST,OPTIONS,TRACE
Content-Type: text/html; charset=UTF-8
Date: Wed, 08 May 2013 10:24:43 GMT
Content-Length: 0
```

9.2 OPTIONS

The OPTIONS method represents a request for information about the communication options available on the request/response chain identified by the Request-URI. This method allows the client to determine the options and/or requirements associated with a resource, or the capabilities of a server, without implying a resource action or initiating a resource retrieval.

Responses to this method are not cacheable.

API Design: Dev Standards

- **TRACE**
- **Should reflect the message received back to client as the body**
- **Allows client to see what/how is being received on the other end**
- **See <https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>**



Copyright Brainwash Inc. 2022

97

API Design: Contracts

- **OpenAPI Documentation**
- **<https://swagger.io/specification/>**
- **Various sections and types**

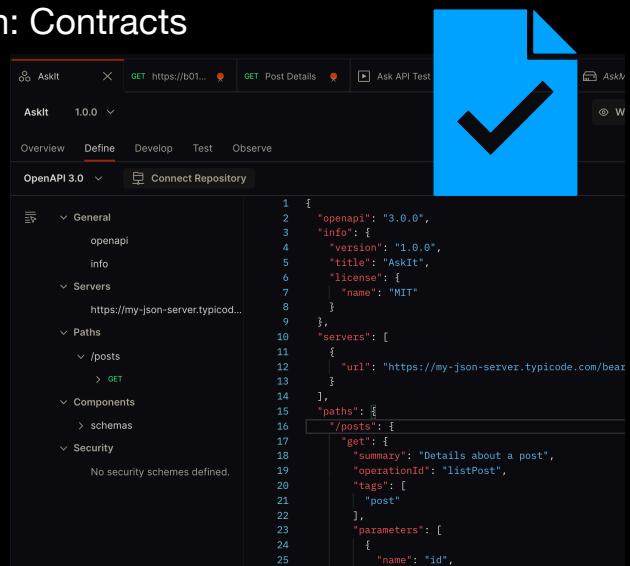
Field Name	Type	Description
openapi	string	REQUIRED. This string MUST be the semantic version number of the OpenAPI Specification version that the OpenAPI document uses. The <code>openapi</code> field SHOULD be used by tooling specifications and clients to interpret the OpenAPI document. This is <i>not</i> related to the API <code>info.version</code> string.
info	Info Object	REQUIRED. Provides metadata about the API. The metadata MAY be used by tooling as required.
servers	[Server Object]	An array of Server Objects, which provide connectivity information to a target server. If the <code>servers</code> property is not provided, or is an empty array, the default value would be a Server Object with a <code>url</code> value of <code>/</code> .
paths	Paths Object	REQUIRED. The available paths and operations for the API.
components	Components Object	An element to hold various schemas for the specification.
security	[Security Requirement Object]	A declaration of which security mechanisms can be used across the API. The list of values includes alternative security requirement objects that can be used. Only one of the security requirement objects need to be satisfied to authorize a request. Individual operations can override this definition. To make security optional, an empty security requirement (()) can be included in the array.
tags	[Tag Object]	A list of tags used by the specification with additional metadata. The order of the tags can be used to reflect on their order by the parsing tools. Not all tags that are used by the Operation Object must be declared. The tags that are not declared MAY be organized randomly or based on the tools' logic. Each tag name in the list MUST be unique.
externalDocs	External Documentation Object	Additional external documentation.

Copyright Brainwash Inc. 2022

98

API Design: Contracts

- **Tools**
 - **SwaggerUI/Hub**
 - **Insomnia**
 - **OpenAPI Generator**
 - **Postman**
 - **PAW**
 - **Stoplight**
 - **Swashbuckle**
- **Can help generate or consume**



The screenshot shows the AskIt application interface. At the top, there are tabs for 'Overview', 'Define' (which is selected), 'Develop', 'Test', and 'Observe'. Below that is a 'Connect Repository' button. The main area displays an OpenAPI 3.0 specification. The code is as follows:

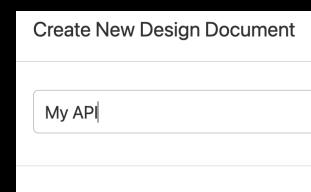
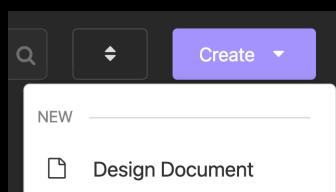
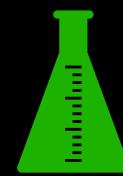
```
1  {
2     "openapi": "3.0.0",
3     "info": {
4         "version": "1.0.0",
5         "title": "AskIt",
6         "license": {
7             "name": "MIT"
8         }
9     },
10    "servers": [
11        {
12            "url": "https://my-json-server.typicode.com/bear"
13        }
14    ],
15    "paths": {
16        "/posts": {
17            "get": {
18                "summary": "Details about a post",
19                "operationId": "listPost",
20                "tags": [
21                    "post"
22                ],
23                "parameters": [
24                    {
25                        "name": "id",
26                    }
27                ]
28            }
29        }
30    }
31 }
```

Copyright Brainwash Inc. 2022

99

API Design: Insomnia

- **Create API in Insomnia**
 - **Open Insomnia**
 - **Click Create > Design Document**
 - **Give it a name and click Create**



Copyright Brainwash Inc. 2022

100

API Design: Insomnia

- **Paste your texts from the Swagger editor into the Insomnia editor**



The screenshot shows the Insomnia API editor interface. On the left, there's a sidebar with sections for INFO, SERVERS, and PATHS. The PATHS section is expanded, displaying a JSON schema for a POST endpoint. The schema includes fields for openapi, info, title, version, description, servers, paths, and responses. The code is numbered from 1 to 21. The main area shows the raw JSON code:

```
1  openapi: 3.0.0
2  info:
3    title: My API
4    version: '1.0'
5    description: test with typicode
6    servers:
7      - url: https://my-json-server.typicode.com/bearc0025/api
8      variables: {}
9      description: github api
10   paths:
11     /posts:
12       summary: post level operations
13       description: fetch or add new posts
14       get:
15         summary: fetch all posts
16         description: return all posts from server
17         operationId: fetchPosts
18       responses:
19         default:
20           description: Default error sample response
21
```

Copyright Brainwash Inc. 2022

101

API Design: Insomnia

- **Test on the right**



The screenshot shows the Insomnia API editor interface. At the top, it says "My API 1.0 OAS3". Below that is a "test with typicode" section. Under "Servers", the URL "https://my-json-server.typicode.com/bearc0025/api - github api" is listed. In the "default" dropdown, a "GET /posts" request is selected. The description for this request is "fetch all posts" and the response is "return all posts from server". Below the requests, there are sections for "Parameters" and "Try it out".

- **End of lab**

Copyright Brainwash Inc. 2022

102

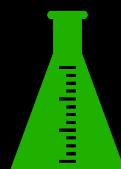
API Design: Contracts

- **For more... PluralSight**

➤ Designing RESTful Web APIs

Shawn Wildermuth

API Design: RESTful API Basics



- **Lab: Local Typicode JSON Server**
 - **Local server**
 - **Same data as Typicode db.json file**
 - **Add server in OpenAPI design**

API Design: RESTful API Basics

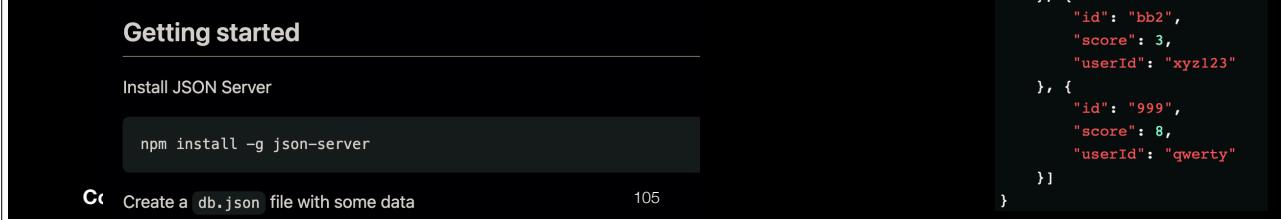
- **Lab: github.com/typicode/json-server**
- **npm install -g json-server**
- **Create db.json file with contents like previous test server**
- **<https://my-json-server.typicode.com/bearc0025/api/db>**

Getting started

Install JSON Server

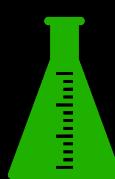
```
npm install -g json-server
```

C Create a db.json file with some data 105



```
{ "users": [ { "id": "1234abc", "username": "bear", "score": 55, "active": true }, { "id": "xyz123", "username": "schmeb", "score": 550, "active": false }, { "id": "qwerty", "username": "jed", "score": 553, "active": true } ], "attempts": [ { "id": "al", "score": 5, "userId": "qwerty" }, { "id": "bb2", "score": 3, "userId": "xyz123" }, { "id": "999", "score": 8, "userId": "qwerty" } ] }
```

- ## API Design: RESTful API Basics
- **Lab**
 - **json-server --watch db.json**
 - **Or: npx json-server -w db.json**
 - **In same folder as db.json file**
 - **<http://localhost:3000/users>**



Bear-MBP:json-server bearc2022\$ json-server --watch db.json

```
\{^_/\ hi!
```

Loading db.json
Done

Resources
<http://localhost:3000/users>
<http://localhost:3000/attempts>

Home
<http://localhost:3000>

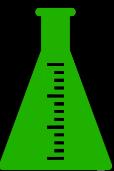
localhost:3000/users/1234abc

```
{
  "id": "1234abc",
  "username": "bear",
  "score": 55,
  "active": true
}
```

Copyright Brainwash Inc. 2022 106



API Design: RESTful API Basics



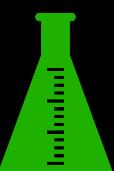
- **Lab**
- **http://localhost:3000/users/qwerty/attempts**

```
[  
  {  
    "id": "a1",  
    "score": 5,  
    "userId": "qwerty"  
  },  
  {  
    "id": "999",  
    "score": 8,  
    "userId": "qwerty"  
  }]
```

Copyright Brainwash Inc. 2022

107

API Design: RESTful API Basics



Typicode - fake server -> typicode/demo (repo)

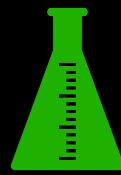
Typicode - fake server -> myaccount/myrepo (repo)

Local typicode server
-> Local db.json

Browser, OpenAPI editor, curl, Insomnia, Postman, etc.

108

API Design: RESTful API Basics



- **Lab**
- **Add Server to Swagger editor**
- **Via menu or edit text**

```
Insert ▾ Gener...
Add Path Item
Add Operation
Add Info
Add External Doc
Add Tag Declaration
Add Tags To Operation
Add Servers
```

```
servers:
- url: http://localhost:3000
  variables: {}
  description: Local User API
- url: https://my-json-server.typicode.com/bearc0025/api
  variables: {}
  description: User API
```

Servers

- ✓ http://localhost:3000 - Local User API
- https://my-json-server.typicode.com/bearc0025/api - User API

API Design: RESTful API Basics



- **Lab**
- **Test against local server**

Request URL
http://localhost:3000/users

Server response

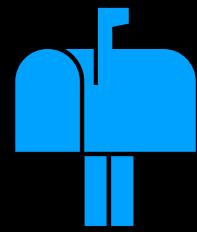
Code Details

200 Response body

```
[  
  {  
    "id": "1234abc",  
    "username": "bear",  
    "score": 55,  
    "active": true  
  },  
  {  
    "id": "xyz123",  
    "username": "schmeb",  
    "score": 550,  
    "active": false  
  },  
  {  
    "id": "qwerty",  
    "username": "jed",  
    "score": 553,  
    "active": true  
  }]
```

Download

- **End of lab**



- **CRUD: Create**
- **POST HTTP Method**
- **Usually on base path e.g. users**
- **HTTP Body with data: JSON**
 - **Server creates ID**
 - **Request Body without ID**

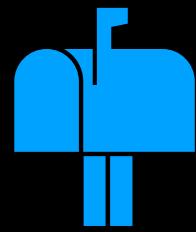
```
{
  "username": "bear",
  "score": 55,
  "active": true
}
```



- **https://swagger.io/specification/#parameter-object**
- **Similar to Response/schema**
- **Not per-HTTP code**
- **Can be defined in Components**

Request Body Object	
Describes a single request body.	
Fixed Fields	
Field Name	Type
description	string
content	Map[string, Media Type Object]
required	boolean

API Design: RESTful API Basics



- **Lab: Add /users POST**
- **Add POST operation**
 - **/users**
 - **post**

Add Operation to Document

Path *
REQUIRED. The path to add the operation to.

Operation *
REQUIRED. Select an operation.

Summary
Add a short summary of what the operation does.

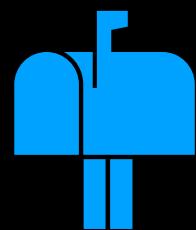
Description
A verbose explanation of the operation behavior. Consider

Operation ID
Unique string used to identify the operation. The id must be unique and operationId to uniquely identify an operation, therefore

Tags
A list of tags for API documentation control. Tags can be used for dynamic generation of Swagger API documentation.

```
post:  
  summary: create user  
  description: store user on server  
  operationId: createUser  
  responses:  
    default:  
      description: Default error sample response  
  tags:  
    - users
```

API Design: RESTful API Basics



- **Lab**
- **Without a Request Body section, the documentation doesn't specify what to send**
- **Add Request Body (typed out on next slide)**

```
post:  
  summary: create user  
  description: store user on server  
  operationId: createUser  
  responses:  
    default:  
      description: Default error sample response  
  tags:  
    - users
```

API Design: RESTful API Basics

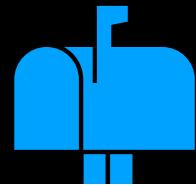
- **Lab**
- **requestBody:**
 - **Starts section**
- **required: true**
- **content:**
 - **application/json**
 - **schema**
 - **object**

```
post:  
  summary: create user  
  description: store user on server  
  operationId: createUser  
  requestBody:  
    required: true  
    content:  
      application/json:  
        schema:  
          type: object  
          properties:  
            username:  
              type: string  
              example: bear  
            score:  
              type: integer  
              example: 55  
            active:  
              type: boolean  
              example: true
```

API Design: RESTful API Basics

- **Lab**
- **properties:**
 - **Object data**
- **<name> e.g., title**
- **type:**
 - **String, Integer, etc.**
 - **See <https://swagger.io/specification/#data-types>**
- **Can be required (otherwise, optional)**

```
post:  
  summary: create user  
  description: store user on server  
  operationId: createUser  
  requestBody:  
    required: true  
    content:  
      application/json:  
        schema:  
          type: object  
          properties:  
            username:  
              type: string  
              example: bear  
            score:  
              type: integer  
              example: 55  
            active:  
              type: boolean  
              example: true
```



- **Lab**
- **Test in documentation**
- **Verify response**

- **End of lab**

Copyright Brainwash Inc. 2022

117

POST /users create user

store user on server

Parameters

No parameters

Request body required

application/json

```
{ "username": "bear", "score": 55, "active": true }
```



- **Lab: Add Request to POST**
- **Add PostNewUser to schemas:**
- **Add ref in post method requestBody:**

```
requestBody:  
  required: true  
  content:  
    application/json:  
      schema:  
        $ref: '#/components/schemas/PostNewUser'
```

```
PostNewUser:  
  type: object  
  properties:  
    username:  
      type: string  
      example: bear  
    score:  
      type: integer  
      example: 55  
    active:  
      type: boolean  
      example: true
```

- **End of lab.**

Copyright Brainwash Inc. 2022

118

API Design



- **OpenAPI: Schemas**
 - **Also supports:**
 - **Wildcards contents: image/***
 - **anyOf, oneOf, allOf**
 - **Files**
 - **Multipart POSTs**
 - **Form Data**

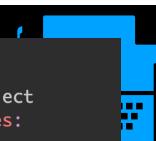
```
Dog: # "Dog" is a value for the pet_type property
      aliof: # Combines the main "Pet" schema with "Dog"
              - $ref: '#/components/schemas/Pet'
              - type: object
                properties:
                  bark:
                    type: boolean
                  breed:
                    type: string
                    enum: [Dingo, Husky, Retriever, Shepherd]
```

Copyright Brainwash Inc. 2022 119

```
requestBody:
  description: A JSON object containing pet information
  content:
    application/json:
      schema:
        oneOf:
          - $ref: '#/components/schemas/Cat'
          - $ref: '#/components/schemas/Dog'
          - $ref: '#/components/schemas/Hamster'

examples:
  hamster: # <--- example name
    summary: An example of a hamster
    value:
      # vv Actual payload goes here vv
      name: Ginger
      petType: hamster
```

API Design



- **OpenAPI: Schemas**
 - **FullUser is the same as PostNewUser plus id**
 - **Define FullUser as PostNewUser and id**

```
schemas:
  FullUser:
    type: object
    properties:
      id:
        type: string
        example: abc123
      username:
        type: string
        example: bear
      score:
        type: integer
        example: 55
      active:
        type: boolean
        example: true
  PostNewUser:
    type: object
    properties:
      username:
        type: string
        example: bear
      score:
        type: integer
        example: 55
      active:
        type: boolean
        example: true
```

Copyright Brainwash Inc. 2022

120



- **OpenAPI: Schemas**
 - Set the schema to be ‘allOf:’
 - Array (“-“)
 - Ref to FullUser
 - “type: object” like now but just id

```

schemas:
  FullUser:
    allOf:
      - $ref: '#/components/schemas/PostNewUser'
      - type: object
        properties:
          id:
            type: string
            example: abc123

```

- **End of lab**



- **OpenAPI: Request Bodies**
 - Can be defined in Components
 - Referenced in operations

```

post:
  summary: create user
  description: store user on server
  operationId: createUser
  requestBody:
    required: true
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/PostNewUser'

post:
  summary: create user
  description: store user on server
  operationId: createUser
  requestBody:
    required: true
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/PostNewUser'

components:
  requestBodies:
    UserPostBody:
      description: new user request
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/PostNewUser'

```

- **End of lab**

API Design



- **OpenAPI: Responses**
 - Can be defined in Components
 - Referenced in operations

```
post:  
  summary: create user  
  description: store user on server  
  operationId: createUser  
  requestBody:  
    $ref: '#/components/requestBodies/UserPostBody'  
  responses:  
    2XX:  
      description: successful fetch  
      content:  
        application/json:  
          schema:  
            $ref: '#/components/schemas/FullUser'
```

```
post:  
  summary: create user  
  description: store user on server  
  operationId: createUser  
  requestBody:  
    $ref: '#/components/requestBodies/UserPostBody'  
  responses:  
    2XX:  
      $ref: '#/components/responses/UserRespBody'  
    default:  
      description: Default error sample response  
  tags:  
    - users
```

```
components:  
  responses:  
    UserRespBody:  
      description: user response  
      content:  
        application/json:  
          schema:  
            $ref: '#/components/schemas/FullUser'
```

- **End of lab**

Copyright Brainwash Inc. 2022

123

API Design



- **OpenAPI: Responses**
 - Arrays
 - /users GET returns an array
 - That can be another schema in another response use in /users GET

```
schemas:  
  UserArray:  
    type: array  
    items:  
      $ref: '#/components/schemas/FullUser'  
  
responses:  
  UserArrayResponseBody:  
    description: user array response  
    content:  
      application/json:  
        schema:  
          $ref: '#/components/schemas/UserArray'
```

```
/users:  
  summary: user level operations  
  description: create and read users  
  get:  
    summary: fetch users  
    description: read users from server  
    operationId: fetchUsers  
    responses:  
      2XX:  
        $ref: '#/components/responses/UserArrayResponseBody'  
      default:  
        description: Default error sample response
```

- **End of lab**

Copyright Brainwash Inc. 2022

124



- **Components**
 - **Reusable items**
 - **Request bodies, responses, schemas, parameters, etc.**
 - **Can become the bulk of your spec**
 - **Helps keep the paths/operations:**
 - **Concise**
 - **Description**
 - **Many references**



- **Server Variables**
 - **Allow for various values**
 - **Ports**
 - **Schemes**
 - **Versions**

API Design



- **Server Variables**
 - **Insert > Add Servers**
 - **Variables within {}**

```
{scheme}://www.example.com:{port}/{version}
```

Server Variables
A map between a variable name and its value. The value is used for substitution in URLs.

Variable Name *	Default *
The name of the server variable.	REQUIRED. The default value is used if no other value is provided. An alternate value is resolved by substituting the variable name with its value. This value MUST be present.
scheme	https
Enum	
An enumeration of strings. Values must be unique and are from a limited set.	
Enum Value	http
Enum Value	https

Copyright Brainwash Inc. 2022

127

API Design



- **Server Variables**

```
- url: '{scheme}://www.example.com:{port}/{version}'  
variables:  
  scheme:  
    default: https  
    enum:  
      - http  
      - https  
    description: 'protocol scheme'  
  version:  
    default: v2  
    enum:  
      - v2  
      - v1  
    description: 'version of api'  
  port:  
    default: '8080'  
    enum:  
      - '8080'  
      - '8000'  
    description: 'server port'
```

Copyright Brainwash Inc. 2022

Servers
{scheme}://www.example.com:{port}/{version} - variables

Computed URL: https://www.example.com:8080/v2

Server variables

scheme	https
version	v2
port	8080 8000

128



API Design



- **OpenAPI: Paths and Parameters**
 - **May be templated:**
 - **/users/{userId}**
 - **/users/{userId}/attempts**
 - **OpenAPI specifies parameters**
 - **name**
 - **in: path, query, header, cookie**
 - **schema: typically integer or string**

```
parameters:  
- name: userId  
  in: path  
  required: true  
  description: Para  
  schema:  
    type : integer  
    format: int64  
    minimum: 1
```

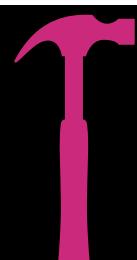
API Design



- **OpenAPI: Parameters**
 - **in:**
 - **path - URL**
 - **query - Query String**
 - **header - request header**
 - **cookie - Cookie**

```
parameters:  
- name: userId  
  in: path  
  required: true  
  description: Parameter  
  schema:  
    type : integer  
    format: int64  
    minimum: 1
```

API Design



- **Lab: /users/{userId}**
 - **Add /users/{userId}**
 - **Add GET method**
 - **Include id parameter of type Integer**

Add Path

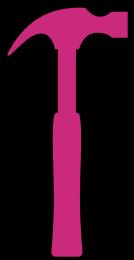
Insert ▾ Generat

Add Path Item

Path *	REQUIRED. The path to add.
/users/{userId}	
Summary	Enter a summary of the path.
specific user operation	
Description	An optional, string description, int
fetch, update, delete user	

```
/users/{userId}:  
  summary: specific user operation  
  description: fetch, update, delete user
```

API Design



- **Lab: /users/{userId}**
- **Add GET Operation**

Insert ▾ Generate
Add Path Item
Add Operation

Add Operation to Document

Path *
REQUIRED. The path to add the operation to.
`/users/{userId}`

Operation *
REQUIRED. Select an operation.
`get`

Summary
Add a short summary of what the operation does.
`fetch user`

Description
A verbose explanation of the operation behavior. Consider adding examples.
`fetch user by id`

Operation ID
Unique string used to identify the operation. The id MUST be unique across all operations in the API to uniquely identify an operation, therefore it is recommended to use the operation name.
`fetchUser`

Tags
A list of tags for API documentation control. Tags can be used for dynamic generation of API documentation.
`users`

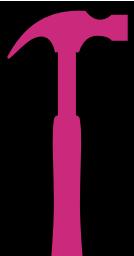
Tag *
REQUIRED. The name of the tag.
`users`

```
/users/{userId}:
  summary: specific user operation
  description: fetch, update, delete user
  get:
    summary: fetch user
    description: fetch user by id
    operationId: fetchUser
    responses:
      default:
        description: Default error sample response
    tags:
      - users
```

Copyright Brainwash Inc. 2022

133

API Design



- **Lab: /users/{userId}**
- **Add GET parameters section (type in editor)**
- **Try it out**

```
/users/{userId}:
  summary: specific user operation
  description: fetch, update, delete user
  get:
    summary: fetch user
    description: fetch user by id
    operationId: fetchUser
    parameters:
      - name: userId
        in: path
        required: true
        schema:
          type: integer
          minimum: 0
          example: 1
```

Request URL
`http://localhost:3000/users/1234abc`

Server response

Code	Details
200	Response body
	<pre>{ "id": "1234abc", "username": "bear", "score": 55, "active": true }</pre> <p>Edit Download</p>

134



- **OpenAPI: Parameters**
 - **query - Query String**
 - **e.g., /users?offset=20**

```
- in: query
  name: offset
  schema:
    type: integer
  description: The number of items to skip
```



- **OpenAPI: Parameters**
 - **header - request header**
 - **e.g., X-Request-ID:**
77e1c83b-7bb0-437b-bc50-a7a58e5660ac

```
paths:
  /ping:
    get:
      summary: Checks if the server is alive
      parameters:
        - in: header
          name: X-Request-ID
          schema:
            type: string
            format: uuid
            required: true
```



- **OpenAPI: Parameters**
 - **cookie - Cookie**
 - e.g., **Cookie: debug=0;**
csrftoken=BUSe35dohU3O1MZvDCUOJ

```
parameters:
  - in: cookie
    name: debug
    schema:
      type: integer
      enum: [0, 1]
      default: 0
  - in: cookie
    name: csrftoken
    schema:
      type: string
```



- **OpenAPI: Parameters**
 - **Required: true/false**
 - **schema**
 - **Used for primitive types**
 - **Arrays**
 - **Serialized (to string) objects**
 - **content for more complex serialization**
 - See <https://swagger.io/docs/specification/serialization/>

```
parameters:
  - name: userId
    in: path
    required: true
    description: Parameter
    schema:
      type : integer
      format: int64
      minimum: 1
```

API Design



- **OpenAPI: Parameters**
 - **May have default values**
 - Not needed for required values
 - **May have min/max**

```
parameters:  
  - in: cookie  
    name: debug  
    schema:  
      type: integer  
      enum: [0, 1]  
      default: 0  
  - in: cookie  
    name: csrfToken  
    schema:  
      type: string
```

```
parameters:  
  - name: userId  
    in: path  
    required: true  
    description: Parameter  
    schema:  
      type: integer  
      format: int64  
      minimum: 1
```

API Design



- **OpenAPI: Parameters**
 - **May have an example(s)**
 - **May be an enum**

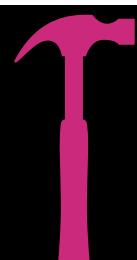
```
parameters:  
  - in: query  
    name: limit  
    schema:  
      type: integer  
      minimum: 1  
    example: 20
```

```
parameters:  
  - in: query  
    name: status  
    schema:  
      type: string  
      enum:  
        - available  
        - pending  
        - sold
```



- **OpenAPI: Parameters**
 - **Parameters shared by multiple methods may be defined on the path level.**
 - **e.g., Used by GET and DELETE**

```
/posts/{postId}:
  summary: individual post operations
  description: fetch, delete, update
  parameters:
    - in: path
      required: true
      name: postId
      schema:
        type: integer
        minimum: 0
      example: 5
```



- **Lab: /users/{userId}**
 - **Put parameters section on path level**
 - **Applies to all operations for this path**

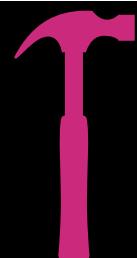
```
/users/{userId}:
  summary: specific user operation
  description: fetch, update, delete user
  parameters:
    - name: userId
      in: path
      required: true
      schema:
        type: string
      example: 1234abc
```



- **OpenAPI: Parameters**
- **May be in components**
- **Referenced with \$ref**

```
components:
  parameters:
    UserId:
      name: userId
      in: path
      required: true
      schema:
        type: string
      example: 1234abc
```

```
/users/{userId}:
  summary: specific user operation
  description: fetch, update, delete user
  parameters:
    - $ref: '#/components/parameters/UserId'
```



- **Lab: /users/{userId}**
- **Put parameter in parameters section of components**
- **Refer from operation /user/{userId}**

```
components:
  parameters:
    UserId:
      name: userId
      in: path
      required: true
      schema:
        type: string
      example: 1234abc
```

```
/users/{userId}:
  summary: specific user operation
  description: fetch, update, delete user
  parameters:
    - $ref: '#/components/parameters/UserId'
```

- **End of lab**



- **OpenAPI: Parameters - Serialization**
 - For transmitting data structures/objects
 - **Keywords:**
 - **style** - how values are delimited
 - Various by parameter location (e.g., path)
 - **explode (boolean)**- specifies if values should be separate parameters (e.g., arrays)



- **Path Parameters**
 - **Styles:**
 - **simple (default)**
 - **label**
 - **matrix**
 - **Explode**
 - **true**
 - **false (default)**



- **Styles: simple - primitive**
- **Looks like /rates/{id}**
- **Parameter as we've seen it before**
- **Required**
- **Schema type: integer**

r.typicode.com/bearc0025/api/rates/1

```
/rates/{id}:
  summary: review stuff
  description: operations on item
  get:
    summary: fetch data
    description: returns data from db
    parameters:
      - in: path
        required: true
        name: id
        schema:
          type: integer
```



- **Styles: simple - array**
- **Array of integers**
- **explode true: /rates/{id*}**
- **explode false: /rates/{id}**
- **URL the same**

on-server.typicode.com/bearc0025/api/rates/1,2,4

```
/rates/{id*}:
  summary: review stuff
  description: operations on item
  get:
    summary: fetch data
    description: returns data from db
    parameters:
      - in: path
        style: simple
        explode: false
        required: true
        name: id*
        schema:
          type: array
          items:
            type: integer
```



- **Styles: simple - object**
 - **Object**
 - **explode: false**
 - **Comma separated**

```
-json-
icode.com/bearc0025/api/rates/id,1,text,this%20post
```

```
/rates/{item}:
  summary: review stuff
  description: operations on item
  get:
    summary: fetch data
    description: returns data from db
    parameters:
      - in: path
        style: simple
        explode: false
        required: true
        name: item
        schema:
          type: object
          properties:
            id:
              type: integer
              example: 1
            text:
              type: string
              example: this post
```



- **Styles: simple - object**
 - **Object**
 - **explode: true**
 - **Key/Value pairs**

```
.com/bearc0025/api/rates/id=1,text=this%20post
```

```
/rates/{item*}:
  summary: review stuff
  description: operations on item
  get:
    summary: fetch data
    description: returns data from db
    parameters:
      - in: path
        style: simple
        explode: true
        required: true
        name: item*
        schema:
          type: object
          properties:
            id:
              type: integer
              example: 1
            text:
              type: string
              example: this post
```



- **Styles: label - primitive/array**
 - **Dot-prefixed (aka Label Expansion)**
 - **{.item}**

Request URL

```
https://my-json-server.typicode.com/bearc0025/api/rates/.1
```

- **{.item*} for array**

URL

```
://my-json-server.typicode.com/bearc0025/api/rates/.1.2
```

Copyright Brainwash Inc. 2022

151

```
/rates/{.item}:
  summary: review stuff
  description: operations on item
  get:
    summary: fetch data
    description: returns data from item
    parameters:
      - in: path
        style: label
        explode: false
        required: true
        name: .item
        schema:
          type: integer
```



- **Styles: label - object**
 - **explode: false**

URL

```
/my-json-
typicode.com/bearc0025/api/rates/.id.1.text.this%20post
```

- **explode: true**

URL

```
my-json-
picode.com/bearc0025/api/rates/.id=1.text=this%20post
```

Copyright Brainwash Inc. 2022

152

```
parameters:
  - in: path
    style: label
    explode: false
    required: true
    name: .item
    schema:
      type: object
      properties:
        id:
          type: integer
          example: 1
        text:
          type: string
          example: this post
```



- **Styles: matrix**
 - **Primitive**

Request URL

```
https://my-json-server.typicode.com/bearc0025/api/rates/;item=1
```

- **Array - Explode**
 - **False**

Request URL

```
https://my-json-server.typicode.com/bearc0025/api/rates/;item=1,2
```

- **True**

Request URL

```
https://my-json-server.typicode.com/bearc0025/api/rates/;item=1;item=2
```



- **Styles: matrix**
 - **Object**
 - **Explode false**

Request URL

```
https://my-json-
server.typicode.com/bearc0025/api/rates/;item=id,1;text,this%20post
```

- **Explode true**

Request URL

```
https://my-json-
server.typicode.com/bearc0025/api/rates/;id=1;text=this%20post
```



- **Query Parameters**
 - **Styles**
 - **form**
 - **spaceDelimited**
 - **pipeDelimited**
 - **deepObject**

- **Query Parameter Styles**
 - **form**

Request URL

```
https://my-json-server.typicode.com/bearc0025/api/rates?item=1
```

```
/rates:  
summary: review stuff  
description: operations on item  
get:  
summary: fetch data  
description: returns data from db  
parameters:  
- in: query  
name: item  
style: form  
explode: true  
schema:  
type: integer
```

• Array

- **Explode false**

Request URL

```
https://my-json-server.typicode.com/bearc0025/api/rates?item=1,2
```

- **Explode true**

Request URL

```
https://my-json-server.typicode.com/bearc0025/api/rates?item=1&item=2
```



- **Query Parameter Styles**
- **spaceDelimited - (explode false)**

Request URL

```
https://my-json-server.typicode.com/bearc0025/api/rates?item=1%202
```

- **pipeDelimited - (explode false)**

Request URL

```
https://my-json-server.typicode.com/bearc0025/api/rates?item=1|2
```

- **For explode = true, looks like form**

Copyright Brainwash Inc. 2022



- **Query Parameter Styles**
- **deepObject (Object)**
 - **%5B = [**
 - **%5D =]**
 - **?item[id]=1&item[text]=this post**

Request URL

```
https://my-json-server.typicode.com/bearc0025/api/rates?  
item%5Bid%5D=1&item%5Btext%5D=this%20post
```

Copyright Brainwash Inc. 2022

158

API Design



- **Header, Cookie Serialization**
- See <https://swagger.io/docs/specification/serialization/>

Header Parameters

Header parameters always use the `simple` style, that is, comma-separated values. This corresponds to the `{param_name}` URI template. An optional `explode` keyword controls the object serialization. Given the request header named `X-MyHeader`, the header value is serialized as follows:

style	explode	URI template	Primitive value X-MyHeader = 5	Array X-MyHeader = [3, 4, 5]	Object X-MyHeader = {"role": "admin", "firstName": "Alex"}
simple *	false *	(id)	X-MyHeader: 5	X-MyHeader: 3,4,5	X-MyHeader: role=admin,firstName=Alex
simple	true	(id*)	X-MyHeader: 5	X-MyHeader: 3,4,5	X-MyHeader: role=admin,firstName=Alex

* Default serialization method

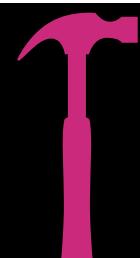
Cookie Parameters

Cookie parameters always use the `form` style. An optional `explode` keyword controls the array and object serialization. Given the cookie named `id`, the cookie value is serialized as follows:

style	explode	URI template	Primitive value id = 5	Array id = [3, 4, 5]	Object id = {"role": "admin", "firstName": "Alex"}
form *	true *		Cookie: id=5		
form	false	id=(id)	Cookie: id=5	Cookie: id=3,4,5	Cookie: id=role=admin,firstName=Alex

* Default serialization method

API Design



- **Lab: Swagger DELETE with ID**
 - **Move {id} parameter to components**
 - **(If haven't already)**

```
/posts/{postId}:
  summary: individual post operations
  description: fetch, delete, update
  parameters:
    - $ref: '#/components/parameters/PostId'
```

API Design

- **Lab: Swagger DELETE with ID**
 - Add DELETE operation/method to /users/{userId} (Insert -> Add Operation)
 - Test

Add Operation to Document

Path *
REQUIRED. The path to add the operation to.
`/users/{userId}`

Operation *
REQUIRED. Select an operation.
`delete`

Summary
Add a short summary of what the operation does.
`delete a user`

Description
A verbose explanation of the operation behavior. Commonly used to describe the effect of the operation.
`remove user from server`

Operation ID
Unique string used to identify the operation. The id MUST be unique across all operations. It is recommended to use the operationId to uniquely identify an operation, therefore, the value of operationId is `deleteUser`.

Tags
A list of tags for API documentation control. Tags can be used to group operations together, so that they can be displayed together.

Tag *
REQUIRED. The name of the tag.
`users`

```
delete:  
  summary: delete a user  
  description: remove user from server  
  operationId: deleteUser  
  responses:  
    default:  
      description: Default error sample response  
  tags:  
    - users
```

- **End of Lab**

Copyright Brainwash Inc. 2022

161

API Design

- **Lab: Swagger PATCH with ID**
 - Add patch operation to /users/{userId}
 - Very similar to post on /users
 - NOTE: Be sure to add the requestBody
 - Test

Path *
REQUIRED. The path to add the operation to.
`/users/{userId}`

Operation *
REQUIRED. Select an operation.
`patch`

```
patch:  
  summary: update a user  
  description: store changes on server  
  operationId: updateUser  
  requestBody:  
    $ref: '#/components/requestBodies/UserPostBody'  
  responses:  
    default:  
      description: Default error sample response  
  tags:  
    - users
```

Copyright Brainwash Inc. 2022

162

API Design



- **Lab: Swagger PATCH with ID**
 - Update a user and verify the changes made are stored and existing values are otherwise not changed

The screenshot shows the Swagger UI interface for a PATCH request. On the left, a "Response body" panel displays a JSON object with fields: "username": "bear", "score": 55, "active": true, and "id": "i1HYeyQ". In the center, a "Request body" panel shows a JSON object with the same fields, except "active" is set to false. A green arrow points from the "active: true" field in the response to the "active: false" field in the request. On the right, another "Response body" panel shows the updated JSON object with "active" set to false. The "id" field remains "i1HYeyQ".

Copyright Brainwash Inc. 2022

163

API Design: Design First



Copyright Brainwash Inc. 2022

164

API Design: Design First

- **Best for stability/maintainability**
- **Requires commitment**
 - **Keep up to date later**
- **Ideally know all data/endpoints**
- **Mocks/test**
- **Client feedback is earlier**
- **Benefits from code generators**
- **Code can be questionable**



API Design: Design First

- **What could go wrong?**
 - **Reality - backend/db isn't as expected when implemented**
 - **Client needs aren't served**
 - **Large/many queries for little data**
 - **Good for exposing data to unknown clients and uses**



API Design: Code First



API Design: Code First

- **Best for speed**
- **Flexible**
- **Doesn't solve problems that don't exist**
- **Uncovers problems without design**
- **Documentation suffers**
- **Harder to maintain/communicate**



API Design: Code First

- **What could go wrong?**
 - **Code doesn't map to endpoints, parameters**
 - **Doc/communication fails**
 - **API solidification causes delays**
 - **Good for in-house data and quick updates internally (especially for small, tight teams)**



API Design: Automated Testing



- **Benefits**
 - **Agnostic**
 - **Isolated**
 - **Repeatable**
 - **Fast**



- **Focus On What You're Testing**
 - **Functionality = Unit Tests**
 - **GUI = UI Tests**
 - **API = Endpoints, Format, etc.**
- **Errors**
 - **Proper codes**
 - **Messages**
 - **Graceful failures**



- **Tools**
 - **Mock servers**
 - **Postman, Insomnia, Paw, etc.**
- **Fully Fleshed out OpenAPI**
 - **Endpoints & Methods**
 - **Request Bodies**
 - **Responses**
 - **Components/Model**
- **SDK Generation if applicable**



- **Insomnia**

- **"Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any javascript testing framework."**



Chai Assertion Library



- **Insomnia**

- **<https://www.chaijs.com/api/bdd/>**

BDD

.lengthOf(n[, msg])

- @param {Number} n
- @param {String} msg _optional_

Asserts that the target's `length` or `size` is equal to the given number `n`.

```
expect([1, 2, 3]).to.have.lengthOf(3);
expect('foo').to.have.lengthOf(3);
expect(new Set([1, 2, 3])).to.have.lengthOf(3);
expect(new Map([('a', 1), ('b', 2), ('c', 3)])).to.have.lengthOf(3);
```

The BDD styles are `expect` and `should`. Both use the same chainable language to construct assertions, but they differ in the way an assertion is initially constructed. Check out the [Style Guide](#) for a comparison.

API Reference

Language Chains

The following are provided as chainable getters to improve the readability of your assertions.

Chains

- `to`
- `be`
- `been`
- `is`
- `that`
- `which`
- `and`
- `has`
- `have`
- `with`
- `at`
- `of`
- `same`
- `but`
- `does`
- `still`
- `also`

API Design: Automated Testing



- **Insomnia**
 - **Create Test Suite**
 - **Add tests and 'expects'**

New Suite

Tests Failed 1/1

Fetch All Users

[GET] fetch all users

Failed Fetch All Users 337 ms

expected 'bear' to equal 'bearasdf'

Returns 200

New Suite 1

Tests Passed 1/1

Find by Status Array

pet / [GET] Finds Pets by status

Passed Find by Status Array

```
const response1 = await insomnia.send();
const body = JSON.parse(response1.data);
expect(body).to.be.an('array');
expect(body).to.have.lengthOf.above(1);
const item = body[0];
expect(item).to.be.an('object');
expect(Object.keys(item)).to.have.lengthOf(4);
expect(item).to.have.property('id');
expect(item).to.have.property('username');
expect(item).to.have.property('score');
expect(item).to.have.property('active');
expect(item.username).to.equal('bear');
```

```
const response = await insomnia.send();
const body = JSON.parse(response.data);
expect(body).to.be.an('array');
expect(body).to.have.lengthOf.above(50);
const item = body[0];
expect(item).to.be.an('object');
expect(Object.keys(item)).to.have.lengthOf.above(4);
expect(item.id);
```

Copyright Brainwash Inc. 2022

API Design: Automated Testing



- **Insomnia**
 - **Create Test Suite**
 - **Add tests and 'expects'**

```
const response1 = await insomnia.send();

expect(response1.status).to.equal(200, "not 200");

const body = JSON.parse(response1.data);

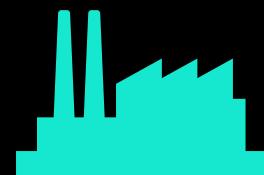
expect(body).to.be.an('array');
expect(body).to.have.lengthOf.greaterThan(0);
expect(body[0]).to.have.property('id');
expect(body[0]).to.have.property('postId');
```

API Design: Automated Testing

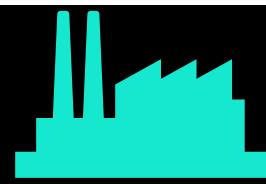


- **Insomnia and other tools can...**
 - **Lint your spec**
 - **Look for and trap small issues with big effects in syntax and similar**
 - **Run Tests**
 - **CI/CD on checkin, automated**
 - **Define config**
 - **Generate Code, Documentation**

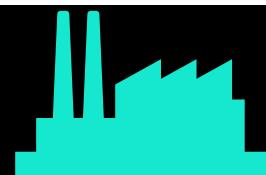
API Design: Legacy Systems



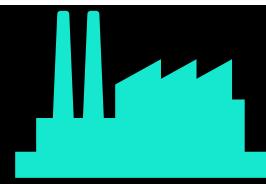
- **What is the Legacy system?**
- **How to map to REST nouns**
 - **What are the data?**
 - **Tables? Columns? Join?**
 - **e.g., Report generation -> /report**
- **Focus on API, state (not functionality)**
- **API might be a facade**
- **Treat the client and server as equals**



- **Is the Legacy system based on database tables?**
 - **Nouns may be based on tables**
 - **Verbs convert straight from CRUD**
 - **Functionality can be minimal**
 - **Authentication always a concern**



- **Is the Legacy system based on functionality?**
 - **Nouns may be based on results**
 - **May only have GET for some endpoints (e.g., /reports)**
 - **Authentication always a concern**



- **Is the Legacy system based on functionality?**
 - **May use route aliases for grouping or common requests**
 - **e.g., /reports/today, /reports/yest**





- **Versioning**
 - **In URL**
 - **Easy, clear**
 - **Requires URL changes for clients**
 - **...example.com/api/v2/reports/34**



- **In Header**
 - **Might be better for HATEOAS**
 - **Limits clients - requires header**
 - **Accept Header:**
 - **e.g., Accept: application/vnd.example+json;version=1.0**
 - **Custom Header:**
 - **e.g., Accept-version: v1**

API Design: Versioning



- **Both**
 - **Major in URL - defaults to latest minor**
 - **Allows for updates**

API Design: Versioning



- **Backward Compatible**
 - **Old versions -> old code**
 - **Or new code's version of functionality**
 - **Behind the scenes implementation**

API Design: Versioning



- **Deprecated version/endpoint**
 - **Freeze code**
 - **Reliable**
 - **Easier to isolate and update/fix**

API Design: Security by Design



API Design: Security by Design

- **Consider security as 1st class entity**
- **SSL**
- **HTTP Basic Auth (header)**
- **API keys, secrets**
 - **Dangerous, Changes**
 - <http://www.omdbapi.com/?apikey=<api key>&t=memento>
- **OAuth**
 - **Authentication, Authorization**
 - **JWT (Tokens)**
 - **ID**
 - **Access**



Copyright Brainwash Inc. 2022

```
GET /api/rest/contracts/status?contractId=42 HTTP/1.1  
Authorization: Basic ZXh0ZXJuYWwtc2Vydm1jZW5hbWU6YXBpa2V5
```

- **Authentication vs Authorization**
 - **Client needs to be authenticated**
 - **e.g., username + password**
 - **Results in authorization**
 - **e.g., token**
 - **Has certain permissions (and not others)**



Copyright Brainwash Inc. 2022

190

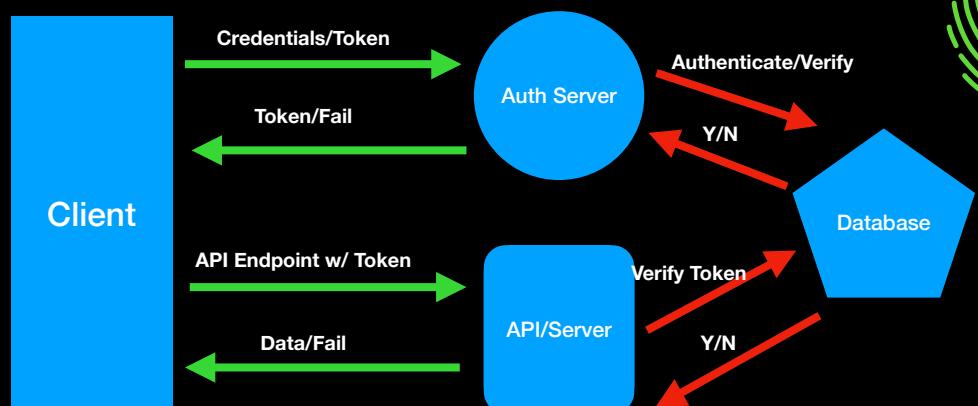
API Design: JSON Web Tokens

- **JWT**
- <https://jwt.io/introduction>
 - `<header>.<payload>.<signature>`
- <https://openid.net/developers/jwt/>
- **Authentication/Authorization**
 - **Access - http header**
 - **Expiration - refresh**
 - **SDKs often handle bulk**
 - **Authorize by user/role**



API Design: Security by Design

- **Authenticate and Authorize**



API Design: Security by Design

- **securityDefinitions** is now **securitySchemes** in components for reuse

```
components:  
  securitySchemes:  
  
    BasicAuth:  
      type: http  
      scheme: basic  
  
    BearerAuth:  
      type: http  
      scheme: bearer  
  
    ApiKeyAuth:  
      type: apiKey  
      in: header  
      name: X-API-Key  
  
    OpenID:  
      type: openIdConnect  
      openIdConnectUrl: https://example.com/.well-known/openid-config  
  
    OAuth2:  
      type: oauth2  
      flows:  
        authorizationCode:  
          authorizationUrl: https://example.com/oauth/authorize  
          tokenUrl: https://example.com/oauth/token  
          scopes:  
            read: Grants read access  
            write: Grants write access  
            admin: Grants access to admin operations
```

API Design: Security by Design

- **Example: Cognito**



```
components:  
  securitySchemes:  
    MyAppAuth:  
      type: "apiKey"  
      name: "token"  
      in: "header"  
      x-amazon-apigateway-authType:  
        "cognito_user_pools"
```

```
paths:  
  /v2/alarm:  
    get:  
      responses:  
        "200":  
          description: "200 response"  
          content:  
            application/json:  
              schema:  
                $ref: "#/components/schemas/Alarm"  
      security:  
        - MyAppAuth: []
```

Security



- **Security Scheme Object**
- **securitySchemes in Components**
 - **Supported schemes:**
 - **HTTP authentication**
 - **API key (header, cookie, query)**
 - **Mutual TLS (server & client certs)**
 - **OAuth2 (implicit*, password, client credentials, authorization code)**

* deprecated

Security



- **Fields**
 - **type : string**
 - **Values: apiKey, http, mutualTLS, oauth2, openidConnect**
 - **description : string**
 - **Text description**
 - **Each type has their own fields...**

Security

- **apiKey fields:**
 - **name**
 - **The name of the header, cookie or query string parameter where the API key is passed.**
 - **in**
 - **Where the API key is passed: query, header, cookie**

```
type: apiKey  
name: api_key  
in: header
```



Security

- **http fields:**
 - **scheme**
 - **HTTP authorization used**
 - **Values should be in IANA registry**
 - **e.g., Basic, Bearer, etc.**
 - **bearerFormat (for http bearer scheme)**
 - **Hint for token format (e.g., JWT)**

```
type: http  
scheme: bearer  
bearerFormat: JWT
```





- **oauth2 fields:**
 - **flows (Object)**
 - **implicit**
 - **password**
 - **clientCredentials**
 - **authorizationCode**
 - **The above 4 fields are flow objects...**



- **flow fields:**
 - **authorizationURL (implicit and authorizationCode)**
 - **tokenURL (password, clientCredentials, authorizationCode)**
 - **refreshURL (all)**
 - **scopes (all) - [string,string]**
 - **e.g., "write:pets": "modify..."**
 - **Example...**

Security



```
type: oauth2
flows:
  implicit:
    authorizationUrl: https://example.com/api/oauth/dialog
    scopes:
      write:pets: modify pets in your account
      read:pets: read your pets
  authorizationCode:
    authorizationUrl: https://example.com/api/oauth/dialog
    tokenUrl: https://example.com/api/oauth/token
    scopes:
      write:pets: modify pets in your account
      read:pets: read your pets
```

Security



```
securitySchemes:
  api_key:
    type: apiKey
    name: api_key
    in: header
  petstore_auth:
    type: oauth2
    flows:
      implicit:
        authorizationUrl: http://example.org/api/oauth/dialog
      scopes:
        write:pets: modify pets in your account
        read:pets: read your pets
```

Security

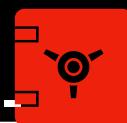


- **Security Requirement for Operation**
 - **List required security schemes**
 - **If multiple, all must be satisfied**
 - **Refs Security Schemes in Components**
 - **For oauth2 and openIdConnect**
 - **May include required scopes**
 - **For others**
 - **May include role names**

```
security:  
- petstore_auth:  
  - write:pets  
  - read:pets
```

```
security:  
- MyAppAuth: []
```

Security

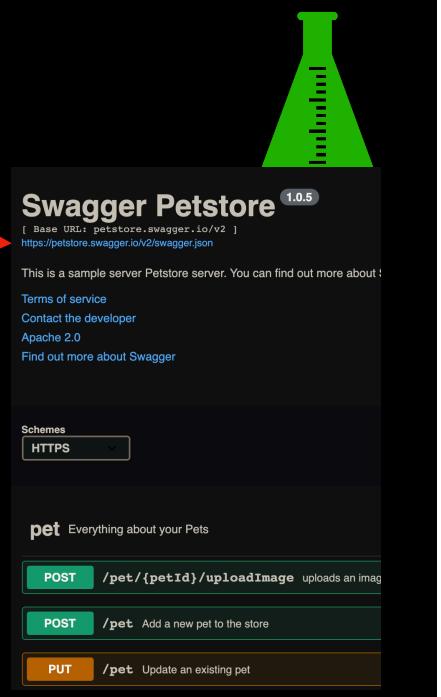


- See <https://github.com/OAI/OpenAPI-Specification/blob/main/versions/3.1.0.md#security-scheme-object>

Security Scheme Object			
Defines a security scheme that can be used by the operations.			
Supported schemes are HTTP authentication, an API key (either as a header, a cookie or mutual TLS (use of a client certificate), OAuth2's common flows (implicit, password, code) as defined in RFC6749 , and OpenID Connect Discovery. Please note that a flow is deprecated by OAuth 2.0 Security Best Current Practice. Recommended for most flows with PKCE.			
Fixed Fields			
Field Name	Type	Applies To	Notes
type	string	Any	REQUIRED. The type are "apiKey", "http", "openIdConnect".
description	string	Any	A description for security scheme, can be used for rich text rendering.
name	string	any	REQUIRED. The name of the security scheme.

API Design

- **Lab: Pet Store API**
 - <https://petstore.swagger.io/>
 - Load JSON
 - Copy text to jsonlint.com
 - Validate and review



A screenshot of the Swagger Petstore 1.0.5 interface. At the top, it shows the base URL: petstore.swagger.io/v2 and the JSON file: https://petstore.swagger.io/v2/swagger.json. Below this, there's a navigation bar with links to Terms of service, Contact the developer, Apache 2.0, and Find out more about Swagger. A dropdown menu for Schemes shows 'HTTPS' selected. The main content area is titled 'pet' with the subtitle 'Everything about your Pets'. It lists three operations:

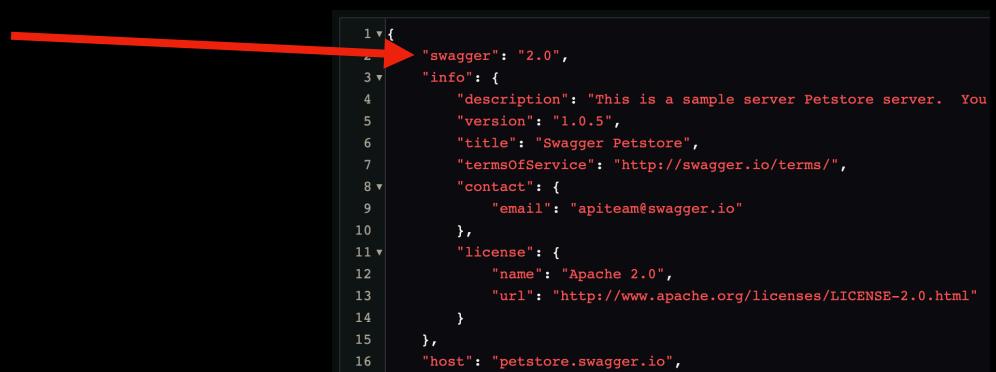
- POST /pet/{petId}/uploadImage - uploads an image
- POST /pet - Add a new pet to the store
- PUT /pet - Update an existing pet

Copyright Brainwash Inc. 2022

205

API Design

- **Lab: Pet Store API**
 - Notice that it's swagger 2.0 JSON
 - But largely similar in content



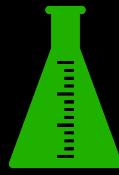
```
1 v{  
2   "swagger": "2.0",  
3   "info": {  
4     "description": "This is a sample server Petstore server. You can find out more about SWAGGER at http://swagger.io",  
5     "version": "1.0.5",  
6     "title": "Swagger Petstore",  
7     "termsOfService": "http://swagger.io/terms/",  
8     "contact": {  
9       "email": "apiteam@swagger.io"  
10    },  
11    "license": {  
12      "name": "Apache 2.0",  
13      "url": "http://www.apache.org/licenses/LICENSE-2.0.html"  
14    }  
15  },  
16  "host": "petstore.swagger.io",  
17  "basePath": "/v2",  
18  "schemes": ["https"],  
19  "tags": [  
20    {"name": "pet", "description": "Operations about pet"},  
21    {"name": "category", "description": "Operations about category"},  
22    {"name": "tag", "description": "Operations about tag"},  
23    {"name": "user", "description": "Operations about user"},  
24    {"name": "store", "description": "Operations about store"}  
25  ],  
26  "paths": {  
27    "/pet": {  
28      "get": {  
29        "summary": "Find pet by ID",  
30        "description": "Returns a single pet",  
31        "parameters": [  
32          {"name": "petId", "in": "path", "required": true, "type": "string"}  
33        ],  
34        "responses": {  
35          "200": {  
36            "description": "A pet model in application/json format",  
37            "schema": {  
38              "$ref": "#/definitions/Pet"  
39            }  
40          }  
41        }  
42      },  
43      "post": {  
44        "summary": "Updates an existing pet",  
45        "description": "Updates an existing pet by ID",  
46        "parameters": [  
47          {"name": "body", "in": "body", "required": true, "schema": {  
48            "$ref": "#/definitions/Pet"  
49          }  
50        },  
51          {"name": "petId", "in": "path", "required": true, "type": "string"}  
52        ]  
53      },  
54      "put": {  
55        "summary": "Deletes a pet",  
56        "description": "Deletes a existing pet",  
57        "parameters": [  
58          {"name": "petId", "in": "path", "required": true, "type": "string"}  
59        ]  
60      },  
61      "delete": {  
62        "summary": "Deletes multiple pets by id",  
63        "description": "Deletes a list of pets",  
64        "parameters": [  
65          {"name": "ids", "in": "body", "required": true, "schema": {  
66            "type": "array",  
67            "items": {  
68              "type": "string"  
69            }  
70          }  
71        }  
72      }  
73    },  
74    "/pet/findByStatus": {  
75      "get": {  
76        "summary": "Finds Pet by status",  
77        "description": "Search for pet by status",  
78        "parameters": [  
79          {"name": "status", "in": "query", "type": "string", "enum": ["available", "pending", "sold"]}  
80        ],  
81        "responses": {  
82          "200": {  
83            "description": "An array of found Pet objects",  
84            "schema": {  
85              "type": "array",  
86              "items": {  
87                "$ref": "#/definitions/Pet"  
88              }  
89            }  
90          }  
91        }  
92      }  
93    },  
94    "/pet/findByTags": {  
95      "get": {  
96        "summary": "Finds Pet by tags",  
97        "description": "Search for pet by tags",  
98        "parameters": [  
99          {"name": "tags", "in": "query", "type": "string", "enum": ["cat", "dog", "fish"]}  
100        ],  
101        "responses": {  
102          "200": {  
103            "description": "An array of found Pet objects",  
104            "schema": {  
105              "type": "array",  
106              "items": {  
107                "$ref": "#/definitions/Pet"  
108              }  
109            }  
110          }  
111        }  
112      }  
113    },  
114    "/pet/{petId}/uploads": {  
115      "post": {  
116        "summary": "uploads an image",  
117        "description": "uploads an image",  
118        "parameters": [  
119          {"name": "file", "in": "body", "type": "file"}  
120        ]  
121      }  
122    },  
123    "/pet": {  
124      "post": {  
125        "summary": "Add a new pet to the store",  
126        "description": "Add a new pet to the store",  
127        "parameters": [  
128          {"name": "body", "in": "body", "schema": {  
129            "$ref": "#/definitions/Pet"  
130          }  
131        }  
132      }  
133    },  
134    "/pet/{petId}": {  
135      "put": {  
136        "summary": "Update an existing pet",  
137        "description": "Update an existing pet",  
138        "parameters": [  
139          {"name": "body", "in": "body", "schema": {  
140            "$ref": "#/definitions/Pet"  
141          }  
142        },  
143          {"name": "petId", "in": "path", "required": true, "type": "string"}  
144        ]  
145      },  
146      "delete": {  
147        "summary": "Deletes a pet",  
148        "description": "Deletes a pet",  
149        "parameters": [  
150          {"name": "petId", "in": "path", "required": true, "type": "string"}  
151        ]  
152      }  
153    },  
154    "/user": {  
155      "post": {  
156        "summary": "Creates a new user",  
157        "description": "Creates a new user",  
158        "parameters": [  
159          {"name": "body", "in": "body", "schema": {  
160            "$ref": "#/definitions/User"  
161          }  
162        }  
163      }  
164    },  
165    "/user/login": {  
166      "get": {  
167        "summary": "Logs user into the system",  
168        "description": "Logs user into the system",  
169        "parameters": [  
170          {"name": "username", "in": "query", "type": "string"},  
171          {"name": "password", "in": "query", "type": "string"}  
172        ]  
173      }  
174    },  
175    "/user/logout": {  
176      "get": {  
177        "summary": "Logs out current user session",  
178        "description": "Logs out current user session",  
179        "parameters": []  
180      }  
181    },  
182    "/user/{userId}": {  
183      "get": {  
184        "summary": "Get user by user ID",  
185        "description": "Get user by user ID",  
186        "parameters": [  
187          {"name": "userId", "in": "path", "required": true, "type": "string"}  
188        ]  
189      },  
190      "put": {  
191        "summary": "Update user",  
192        "description": "Update user",  
193        "parameters": [  
194          {"name": "body", "in": "body", "schema": {  
195            "$ref": "#/definitions/User"  
196          }  
197        },  
198          {"name": "userId", "in": "path", "required": true, "type": "string"}  
199        ]  
200      },  
201      "delete": {  
202        "summary": "Delete user",  
203        "description": "Delete user",  
204        "parameters": [  
205          {"name": "userId", "in": "path", "required": true, "type": "string"}  
206        ]  
207      }  
208    },  
209    "/store/order": {  
210      "post": {  
211        "summary": "Place an order",  
212        "description": "Place an order",  
213        "parameters": [  
214          {"name": "body", "in": "body", "schema": {  
215            "$ref": "#/definitions/Order"  
216          }  
217        }  
218      }  
219    },  
220    "/store/order/{orderId}": {  
221      "get": {  
222        "summary": "Get order by ID",  
223        "description": "Get order by ID",  
224        "parameters": [  
225          {"name": "orderId", "in": "path", "required": true, "type": "string"}  
226        ]  
227      },  
228      "put": {  
229        "summary": "Update an existing order",  
230        "description": "Update an existing order",  
231        "parameters": [  
232          {"name": "body", "in": "body", "schema": {  
233            "$ref": "#/definitions/Order"  
234          }  
235        },  
236          {"name": "orderId", "in": "path", "required": true, "type": "string"}  
237        ]  
238      },  
239      "delete": {  
240        "summary": "Deletes an existing order",  
241        "description": "Deletes an existing order",  
242        "parameters": [  
243          {"name": "orderId", "in": "path", "required": true, "type": "string"}  
244        ]  
245      }  
246    }  
247  },  
248  "definitions": {  
249    "Pet": {  
250      "type": "object",  
251      "properties": {  
252        "id": {  
253          "type": "string",  
254          "format": "int64"  
255        },  
256        "category": {  
257          "type": "string",  
258          "format": "string"  
259        },  
260        "name": {  
261          "type": "string",  
262          "format": "string"  
263        },  
264        "photoUrls": {  
265          "type": "array",  
266          "items": {  
267            "type": "string",  
268            "format": "string"  
269          }  
270        },  
271        "tags": {  
272          "type": "array",  
273          "items": {  
274            "type": "string",  
275            "format": "string"  
276          }  
277        },  
278        "status": {  
279          "type": "string",  
280          "format": "string"  
281        }  
282      }  
283    },  
284    "User": {  
285      "type": "object",  
286      "properties": {  
287        "id": {  
288          "type": "string",  
289          "format": "int64"  
290        },  
291        "username": {  
292          "type": "string",  
293          "format": "string"  
294        },  
295        "firstName": {  
296          "type": "string",  
297          "format": "string"  
298        },  
299        "lastName": {  
300          "type": "string",  
301          "format": "string"  
302        },  
303        "email": {  
304          "type": "string",  
305          "format": "string"  
306        },  
307        "password": {  
308          "type": "string",  
309          "format": "string"  
310        },  
311        "userStatus": {  
312          "type": "string",  
313          "format": "string"  
314        }  
315      }  
316    },  
317    "Order": {  
318      "type": "object",  
319      "properties": {  
320        "id": {  
321          "type": "string",  
322          "format": "int64"  
323        },  
324        "petId": {  
325          "type": "string",  
326          "format": "int64"  
327        },  
328        "quantity": {  
329          "type": "integer",  
330          "format": "int32"  
331        },  
332        "shipDate": {  
333          "type": "string",  
334          "format": "date-time"  
335        },  
336        "status": {  
337          "type": "string",  
338          "format": "string"  
339        },  
340        "name": {  
341          "type": "string",  
342          "format": "string"  
343        },  
344        "email": {  
345          "type": "string",  
346          "format": "string"  
347        }  
348      }  
349    }  
350  }  
351 }
```

Copyright Brainwash Inc. 2022

206

API Design

- **Lab: Pet Store API**
 - Copy JSON into [editor.swagger.io](#)
 - Review the generated docs
 - Test



```
swagger: "2.0"
info:
  description: "This is a sample server Petstore server. You can find out more about
    Swagger at [http://swagger.io](http://swagger.io) or on [irc.freenode.net, #swagger]
    (http://swagger.io irc). For this sample, you can use the api key 'special-key'
    to bypass authorization filters."
  version: "1.0.0"
  title: "Swagger Petstore"
  termsOfService: "http://swagger.io/terms/"
  contact:
    email: "apiteam@swagger.io"
  license:
    name: "Apache 2.0"
    url: "http://www.apache.org/licenses/LICENSE-2.0.html"
  host: "petstore.swagger.io"
  basePath: "/v2"
  tags:
    - name: "pet"
      description: "Everything about your Pets"
      externalDocs:
        description: "Find out more"
        url: "http://swagger.io"
    - name: "store"
      description: "Access to Petstore orders"
      externalDocs:
        description: "Find out more about our store"
        url: "http://swagger.io"
  schemes:
    - https
    - http
  paths:
    /pet:
      post:
        tags:
          - pet
        summary: "Add a new pet to the store"
        description: ""
        operationId: "addPet"
        consumes:
          - application/json
          - application/xml"
```

pet

Swagger Editor documentation

Find out more: <http://swagger.io>

Method	Path	Description
POST	/pet	Add a new pet to the store
PUT	/pet	Update an existing pet
GET	/pet/findByStatus	Finds Pets by status
DELETE	/pet/findByTags	Finds Pets by tags
GET	/pet/{petId}	Find pet by ID
POST	/pet/{petId}	Updates a pet in the store with form data
DELETE	/pet/{petId}	Deletes a pet
POST	/pet/{petId}/uploadImage	uploads an image
store Access to Petstore orders		
GET	/store/inventory	Returns pet inventories by status
POST	/store/order	Place an order for a pet
GET	/store/order/{orderId}	Find purchase order by ID
DELETE	/store/order/{orderId}	Delete purchase order by ID

- End of lab

Copyright Brainwash Inc. 2022

207

API Design

- Videos:
 - Post assessment survey questions



[APIs- Application Programming Interface: Executive Briefing](#)

Dan Appleman



[Designing RESTful Web APIs](#)

Shawn Wildermuth



[REST Fundamentals](#)

Howard Dierking



[Introduction to OAuth2, OpenID Connect and JSON Web Tokens \(JWT\)](#)

Dominick Baier

Copyright Brainwash Inc. 2022

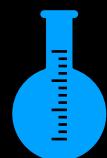
208



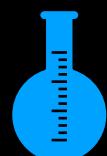
- **Principles**
 - **Types of testing related to API:**
 - **UI**
 - **Functional**
 - **Security**
 - **Performance/Load**
 - **Error**
 - **Validation - end-to-end**



- **Principles**
 - **Types of testing related to API:**
 - **UI**
 - **Creating Requests**
 - **Data format**
 - **Receiving Responses**
 - **May be client (not yours)**
 - **May be SDK**



- **Principles**
 - **Types of testing related to API:**
 - **Functional**
 - **Back end**
 - **Code logic**
 - **Request Processing**
 - **Data Gathering, Formatting**
 - **Response Creation**



- **Principles**
 - **Types of testing related to API:**
 - **Security**
 - **Authentication**
 - **Authorization**
 - **Access Prevention**



- **Principles**
 - **Types of testing related to API:**
 - **Performance/Load**
 - **Process Speed**
 - **Scalability**
 - **Cost**
 - **DOS Security**
 - **Response Time**



- **Principles**
 - **Types of testing related to API:**
 - **Error**
 - **Fail Gracefully**
 - **Appropriate Status/Error Codes**
 - **Messages**
 - **No Secrets**
 - **Logging**

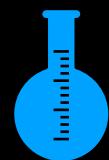


- **Best Practices**
 - **Use Realistic Data**
 - **Mock Data**
 - **Real Data**
 - **All Sizes**
 - **Include Errors**



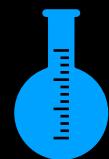
- **Best Practices**
 - **Test Success and Fail**
 - **Don't just test that it works**
 - **Wild Testing**





- **Best Practices**
 - **Drive Tests with Data**
 - **Map test data to outcomes**
 - **Repeatable**
 - **Easily Extendable**
 - **Fail Conditions Added**

(100, 0.5)
(200, 0.75)
(300, 0.85)
(0, 0.0)



- **Best Practices**
 - **Logging/Analytics**
 - **Store API traffic**
 - **Analyze for changes**
 - **Helps with bug research**



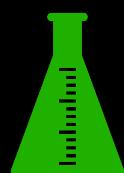
- **Best Practices**
 - **Performance Minded**
 - **Functional Tests can be reused**
 - **Performance**
 - **Scalability**
 - **Security**



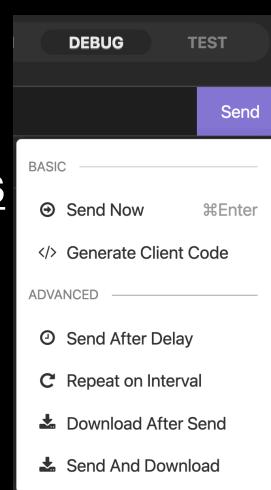
- **Best Practices**
 - **Automation**
 - **Time Saving**
 - **Reliable/Repeatable**
 - **Agnostic to other software/processes**
 - **Right tool for the right job**
 - **Insomnia, Postman, Stoplight, etc.**
 - **CI/CD**



- **Integration Testing**
 - End-to-End
 - Doesn't have to be last
 - May take more time than expected
 - Maybe (automated) UI driven
 - SDK
 - Client Agnostic



- **Testing with Insomnia**
 - Debug > Ctrl+Send menu
 - Delay, Repeat, etc.
 - Plugins
 - <https://insomnia.rest/plugins>
 - CLI
 - <https://insomnia.rest/product/automated-testing>



API Design: Automated Testing

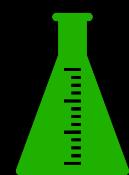


- **Automated Testing with Insomnia**
- **inso CLI**
 - **Lint your spec**
 - **Look for and trap small issues with big effects in syntax and similar**
- **Run Tests**
 - **CI/CD on checkin, automated**
 - **Define config**
 - **Generate Code, Documentation**

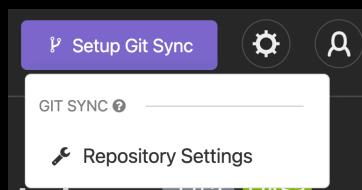
Copyright Brainwash Inc. 2022

223

API Automated Testing



- **CI/CD with Insomnia**
 - **Create git repo and not URL**
 - **In Insomnia select Setup 'Git Sync' > Repository Settings**
 - **Fill out the form**



Configure Repository

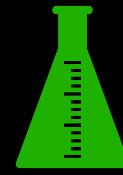
Git URI (https)

Author Name <input type="text" value="Name"/>	Author Email <input type="text" value="Email"/>
Username <input type="text" value="MyUser"/>	Authentication Token <input type="text" value="88e7ee63b254e4b0bf047559eafe86ba9dd49507"/>

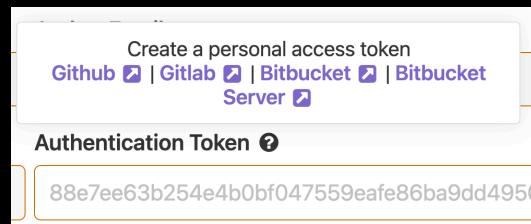
Copyright Brainwash Inc. 2022

224

API Automated Testing



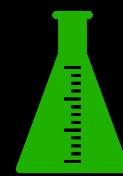
- Token
 - Click ? For links to instructions
 - Sync with Github or similar



Copyright Brainwash Inc. 2022

225

API Automated Testing



- **Insomnia CLI**
 - **Install CLI:**
 - <https://www.npmjs.com/package/insomnia-inso>

```
npm install -g insomnia ins
```

```
MacBook-Pro:~ perry2008$ npm install --global insomnia-insc
npm WARN deprecated har-validator@5.1.0: This library is no longer supported
npm WARN deprecated request@2.88.2: Request has been deprecated, see https://github.com/request/request#issues-3142
See https://nodejs.org/api/modules.html#modules_sealing_for_details.

npm WARN deprecated http-signature@0.11.0: This version has been deprecated and is no longer supported or maintained by its authors. It is now marked as 'deprecated' and no longer receives security updates or promises now, please switch to that.
npm WARN deprecated har-validator@5.1.0: This version has been deprecated and is no longer supported or maintained by its authors. It is now marked as 'deprecated' and no longer receives security updates or promises now, please switch to that.
npm WARN deprecated har-validator@5.0.1: This version has been deprecated and is no longer supported or maintained by its authors. It is now marked as 'deprecated' and no longer receives security updates or promises now, please switch to that.
npm WARN deprecated har-validator@4.0.1: This version has been deprecated and is no longer supported or maintained by its authors. It is now marked as 'deprecated' and no longer receives security updates or promises now, please switch to that.
npm WARN deprecated har-validator@3.5.0: This version has been deprecated and is no longer supported or maintained by its authors. It is now marked as 'deprecated' and no longer receives security updates or promises now, please switch to that.
npm WARN deprecated har-validator@3.3.1: This version has been deprecated and is no longer supported or maintained by its authors. It is now marked as 'deprecated' and no longer receives security updates or promises now, please switch to that.
npm WARN deprecated har-validator@3.2.1: This version has been deprecated and is no longer supported or maintained by its authors. It is now marked as 'deprecated' and no longer receives security updates or promises now, please switch to that.
npm WARN deprecated har-validator@3.1.0: This version has been deprecated and is no longer supported or maintained by its authors. It is now marked as 'deprecated' and no longer receives security updates or promises now, please switch to that.
npm WARN deprecated har-validator@3.0.0: Please upgrade to @mapbox/node-pre-gyp; the non-scoped node-pre-gyp package is deprecated and only the @mapbox scoped package is supported.
11 releases in the future
npm WARN deprecated @mapbox/node-pre-gyp@2.1.0: This version has been deprecated and is no longer supported or maintained
npm WARN deprecated har-validator@2.0.0: Har-validator@2.0.0 and earlier versions of Har-validator's latest version is now available as @mapbox/insomnia-insc on npm

added 64 packages, and audited 495 packages in 1s

64 packages are looking for funding
  run `npm fund` for details

22 vulnerabilities (18 moderate, 9 high, 3 critical)

To address issues that do not require attention, run:
  npx audit fix

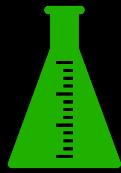
Some issues need review, and may require choosing
a different dependency:
  npx audit --audit-only fix for details.
  Bear-MDP@~ bear@2008 npm audit

  npx audit This command requires an existing lockfile.
```

Copyright Brainwash Inc. 2022

226

API Automated Testing



- **Insomnia CLI**
 - **Execute: `inso run test`**
 - **Select suite**
 - **Select environment**
 - **View Results**

```
Bear-MBP:~ bearc2020$ inso run test
✓ Select a document or unit test suite · Test 1 - uts_276215
? Select an environment ...
OpenAPI env env_env_65edc1
```

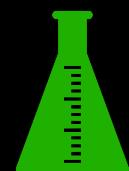
```
Bear-MBP:~ bearc2020$ inso run test
? Select a document or unit test suite ...
Swagger Petstore 1.0.2 spc_6d22e0
| New Suite 1 uts_07beec
My API spc_e4343f
| Test 1 uts_276215
```

```
Bear-MBP:~ bearc2020$ inso run test
✓ Select a document or unit test suite · Test 1 - uts_276215
✓ Select an environment · OpenAPI env - env_env_65edc1

Test 1
✓ Returns 200 (1278ms)

1 passing (1s)
```

API Automated Testing



- **Insomnia CLI**
 - **Note the suite and env identifiers/names**
 - **Run tests directly**

```
Bear-MBP:~ bearc2020$ inso run test
✓ Select a document or unit test suite · Test 1 - uts_276215
✓ Select an environment · OpenAPI env - env_env_65edc1

Test 1
✓ Returns 200 (1278ms)

1 passing (1s)
```

```
Bear-MBP:~ bearc2020$ inso run test uts_276215 --env env_env_65edc1

Test 1
✓ Returns 200 (1144ms)

1 passing (1s)
```

API Automated Testing



- **Insomnia CLI**
- **inso lint spec**

```
Bear-MBP:~ bearc2020$ inso lint spec  
? Select an API Specification ...  
Swagger Petstore 1.0.2 spc_6d22e0  
My API spc_e4343f
```

```
Bear-MBP:~ bearc2020$ inso lint spec  
✓ Select an API Specification · My API - spc_e4343f  
No linting errors. Yay!
```

API Automated Testing



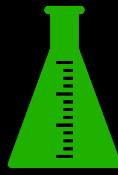
- **Insomnia CLI**
- **inso scripts**
- **Config files**
- **Specify a name and what to run**

```
scripts:  
_ mylint: inso lint spec "My API"
```

- **Run with script (or w/o if name is unique)**

```
Bear-MBP:insomnia bearc2020$ inso mylint  
No linting errors. Yay!  
Bear-MBP:insomnia bearc2020$ inso script mylint  
No linting errors. Yay!
```

API Automated Testing



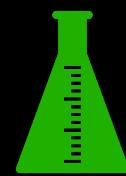
- **Insomnia CI/CD Tests**
 - e.g., **Github Actions**
 - <https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions>
 - **"New Workflow"**

A screenshot of a GitHub repository page for 'bearc0025/insomniaapi'. The page shows tabs for Code, Issues, Pull requests, and Actions. The Actions tab is selected. Below the tabs, there are buttons for Workflows and All workflows. A red arrow points from the 'New Workflow' button in the Workflows section towards the 'New workflow' button in the screenshot.

Copyright Brainwash Inc. 2022

231

API Automated Testing



- **Choose template**
 - **"Set up this workflow"**
 - **Review file**

Choose a workflow template

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging

Skip this and [set up a workflow yourself](#) →

Workflows made for your repository (Suggested)

A screenshot of the GitHub 'Set up this workflow' template interface. It shows a 'Simple workflow' template by GitHub. The template code is as follows:

```
echo Hello, world!
```

The 'Simple workflow' template has a note: "Start with a file with the minimum necessary structure." and a 'Set up this workflow' button.

```
1 # This is a basic workflow to help you get started with Actions
2
3 name: CI
4
5 # Controls when the workflow will run
6 on:
7   # Triggers the workflow on push or pull request events but only for the master branch
8   push:
9     branches: [ master ]
10    pull_request:
11      branches: [ master ]
12
13 # Allows you to run this workflow manually from the Actions tab
14 workflow_dispatch:
15
16 # A workflow run is made up of one or more jobs that can run sequentially or in parallel
17 jobs:
18   # This workflow contains a single job called "build"
19   build:
20     # The type of runner that the job will run on
21     runs-on: ubuntu-latest
22
23 # Steps represent a sequence of tasks that will be executed as part of the job
24 steps:
25   # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
26   - uses: actions/checkout@v2
27
28 # Runs a single command using the runners shell
29 - name: Run a one-line script
30   run: echo Hello, world!
31
32 # Runs a set of commands using the runners shell
33 - name: Run a multi-line script
34   run:
35     echo Add other actions to build,
36     echo test, and deploy your project.
```

Copyright Brainwash Inc. 2022

232

API Automated Testing

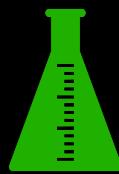


- [**https://docs.insomnia.rest/inso-cli/continuous-integration**](https://docs.insomnia.rest/inso-cli/continuous-integration)
- **Copy 'steps' from Insomnia example at site above**
- **Replace command parameters with your project names/ids**

```
# .github/workflows/test.yml
name: Test

jobs:
  linux:
    name: Validate API spec
    runs-on: ubuntu-latest
    steps:
      - name: Checkout branch
        uses: actions/checkout@v1
      - uses: kong/setup-inso@v1
        with:
          inso-version: 2.4.0
      - name: Lint
        run: inso lint spec "Designer Demo" --ci
      - name: Run test suites
        run: inso run test "Designer Demo" --env UnitTest --ci
      - name: Generate declarative config
        run: inso generate config "Designer Demo" --type declarative --ci
```

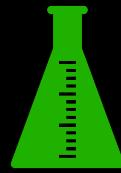
API Automated Testing



- **Paste steps into template**
 - **Optionally remove pull request run in 'on' section**
 - **On push...**
 - **Checkout**
 - **Install**
 - **Lint**
 - **Tests**
 - **Generate config**

```
1 # This is a basic workflow to help you get started with Actions
2
3 name: CI
4
5 # Controls when the workflow will run
6 on:
7   # Triggers the workflow on push or pull request events but only for the master branch
8   push:
9     branches: [ master ]
10
11 # Allows you to run this workflow manually from the Actions tab
12 workflow_dispatch:
13
14 # A workflow run is made up of one or more jobs that can run sequentially or in parallel
15 jobs:
16   # This workflow contains a single job called "build"
17   build:
18     # The type of runner that the job will run on
19     runs-on: ubuntu-latest
20
21 # Steps represent a sequence of tasks that will be executed as part of the job
22 steps:
23   - name: Checkout branch
24     uses: actions/checkout@v1
25   - uses: kong/setup-inso@v1
26     with:
27       inso-version: 2.4.0
28   - name: Lint
29     run: inso lint spec "My API" --ci
30   - name: Run test suites
31     run: inso run test uts_276215 --env "OpenAPI env" --ci
32   - name: Generate declarative config
33     run: inso generate config "My API" --type declarative --ci
```

API Automated Testing



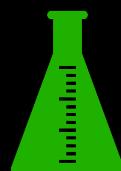
- **Change the name of the file**

insomniaapi / .github / workflows / `blank.yml` in master

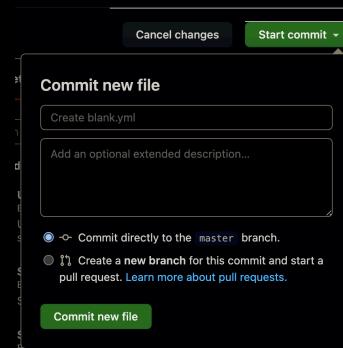
- **When saved, .github/workflows with that file will be added to your repo**



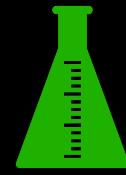
API Automated Testing



- **Commit the file**
- **Add name and comments as you like**
- **'Commit new file'**
- **Pull in Insomnia to keep in sync!**



API Automated Testing



- **Go back to 'Actions' and see your action is in progress**

● new github action

CI #7: Commit 9c97b1b pushed by bearc0025

- **Click it for details**

```
build
Started 28s ago
> ⏪ Set up job
> ⏪ Checkout branch
> ⏪ Install NodeJS
-> ⏪ Install Inso
17  npm WARN deprecated non-pkg-peer@0.15.8: P1
    dependency is deprecated package will receive updates
18  npm WARN deprecated request@2.88.2: request
    ○ Lint
    ○ Run test suites
    ○ Generate declarative config
```

```
build
succeeded now in 37s
> ⏪ Set up job
> ⏪ Checkout branch
> ⏪ Install NodeJS
-> ⏪ Install Inso
> ⏪ Lint
> ⏪ Run test suites
> ⏪ Generate declarative config
> ⏪ Complete job
```

- **Actions shows status**

Copyright Brainwash Inc. 2022

237

✓ new github action

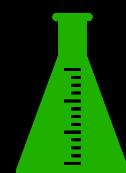
CI #7: Commit 9c97b1b pushed by bearc0025

- **End of lab**

Copyright Brainwash Inc. 2022

238

API Automated Testing



- **Make a change in Insomnia**
- **Commit and Push**
- **Verify action runs**

● test update for action

CI #8: Commit 15e7abf pushed by bearc0025

- **Verify success (or failure)**

✓ test update for action

CI #8: Commit 15e7abf pushed by bearc0025

[bearc0025/insomniaapi] CI workflow run

CI: All jobs have failed

[View workflow run](#)

✖ CI / build Failed in 14 seconds

1

Tutorials

Jetty Web API Servlet



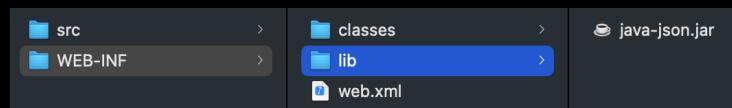
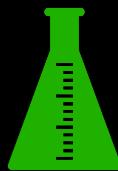
- **Lab: Java Server**
- **Note: This is not meant to be an exhaustive step-by-step for configuring Jetty but may be helpful:**
 - **Jetty: <https://www.eclipse.org/jetty/>**
 - **Unzip**
 - **Set env var:**
 - **JETTY_HOME=/Users/bearc2020/Downloads/jetty-home-11.0.5**
 - **Start server:**
 - **java -jar start.jar**
 - **Load <http://localhost:8080/>**



- **On same level as jetty-home-<ver>**
 - **Create jetty-base dir on same level as jetty-home...**
 - **Run:**
 - **java -jar start.jar jetty.base=/path/to/your/jetty-base --add-module=server,http,deploy,annotations**
 - **Change dir to your jetty-base and run:**
 - **java -jar /path/to/your/jetty-home/start.jar --add-module=server,http,deploy,annotations**

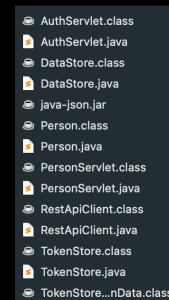
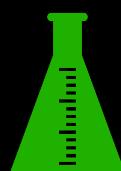
API Design: Java Server and Client

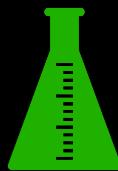
- **Under the jetty-base/webapps dir**
- **Create a PersonServlet dir**
- **Inside PersonServlet create src and WEB-INF dirs**
 - **src is for source code but can be located anywhere**
 - **WEB-INF is for configuration, classes and libraries**



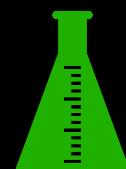
API Design: Java Server and Client

- **Download java-json.jar from here:**
- **[java-json/java-json.jar.zip](#)**
- **Copy that to the src and WEB-INF/lib directories**
- **Optional: Copy various source files into the src directory**





- **Compile the various .java files e.g.,**
- **javac -cp <path to jetty home>/lib/jetty-jakarta-servlet-api-5.0.2.jar:./java-json.jar:.. <filename>.java**
 - NOTE: Not all java files require the servlet api and json jar files.
- **Copy the class files into WEB-INF/classes**
 - AuthServlet.class
 - DataStore.class
 - Person.class
 - PersonServlet.class
 - RestApiClient.class
 - TokenStore.class

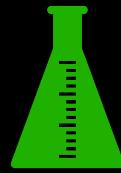


- **Create the web.xml file under the WEB-INF dir**
- **Boilerplate...**

```
<web-app
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1">
```

API Design: Java Server and Client

- **Servlet(s)...**



```
<servlet>
    <servlet-name>PersonServlet</servlet-name>
    <servlet-class>PersonServlet</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>PersonServlet</servlet-name>
    <url-pattern>/people/*</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>AuthServlet</servlet-name>
    <servlet-class>AuthServlet</servlet-class>
</servlet>

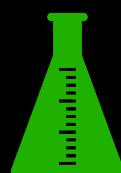
<servlet-mapping>
    <servlet-name>AuthServlet</servlet-name>
    <url-pattern>/auth</url-pattern>
</servlet-mapping>
```

Copyright Brainwash Inc. 2022

247

API Design: Java Server and Client

- **CORS (Cross-Origin Resource Sharing) and end...**



```
<filter>
    <filter-name>cross-origin</filter-name>
    <filter-class>org.eclipse.jetty.servlets.CrossOriginFilter</filter-class>
    <init-param>
        <param-name>allowedOrigins</param-name>
        <param-value>*</param-value>
    </init-param>
    <init-param>
        <param-name>allowedMethods</param-name>
        <param-value>GET,POST</param-value>
    </init-param>
    <init-param>
        <param-name>allowedHeaders</param-name>
        <param-value>X-Requested-With,Content-Type,Accept,Origin,Authorization</param-value>
    </init-param>
</filter>

<filter-mapping>
    <filter-name>cross-origin</filter-name>
    <filter-pattern>/*</filter-pattern>
</filter-mapping>

</web-app>
```

Copyright Brainwash Inc. 2022

248

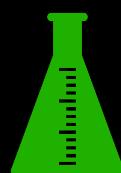
API Design: Java Server and Client

- **With the env var, you can start jetty from other dirs.**
- **Restart jetty from the jetty-base dir:**
 - `java -jar $JETTY_HOME/start.jar`
 - Load: <http://localhost:8080/>
- **Load the web app at:**
 - <http://localhost:8080/PersonServlet/people/Kevin>



API Design: Java Server and Client

- **Create HTML client and place in webapps/root directory**
- **Load with:**
 - <http://localhost:8080/ApiClient.html>



API Client

Token:

Name:

Password:

API Design: Java Server and Client

- **DataStore is initialized with each start with preconfigured data:**



```
personMap.put("Ada", new Person("Ada", "Ada Lovelace was the first programmer.", 1815, "pwa"));
personMap.put("Kevin", new Person("Kevin", "Kevin is the author of HappyCoding.io.", 1986, "pkw"));
personMap.put("Stanley", new Person("Stanley", "Stanley is Kevin's cat.", 2007, "pws")); }
```

- **Users can be fetched with GET without login**
- **Logged in users can Save changes to own account**
- **Logged in users can create new users**

C# .NET Server





- **C# .NET tutorial**
- **<https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-5.0&tabs=visual-studio-code>**

Visual Studio Visual Studio Code Visual Studio for Mac

- Visual Studio Code ↗
- C# for Visual Studio Code (latest version) ↗
- .NET 5.0 SDK ↗

Java Standalone Server Using Spark





- **Java JDK**
- <https://docs.oracle.com/en/java/javase/>
- **Verify JAVA_HOME is set in env**
- **Maven**
- <https://maven.apache.org/download.cgi>
- <https://maven.apache.org/install.html>
- **Verify maven bin dir is in PATH in env**



- **Create Java Maven Project**
- **Create pom.xml**
 - mvn archetype:generate -DgroupId=com.brainwashinc.demos.hellorestjava -DartifactId=hello-rest -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4 -DinteractiveMode=false
- **Build Package**
 - cd hello-rest; mvn clean package
- **Run**
 - **java -cp target/hello-rest-1.0-SNAPSHOT.jar com.brainwashinc.app.App**

```
Bear-MBP:hello-rest bearc2020$ java -cp target/hello-rest-1.0-SNAPSHOT.jar com.brainwashinc.app.App
Hello World!
```



- **Spark**
- "A micro framework for create web applications"
- Add to project: <http://sparkjava.com/download>
- Add to pom.xml

```
<dependency>
    <groupId>com.sparkjava</groupId>
    <artifactId>spark-core</artifactId>
    <version>2.9.3</version>
</dependency>
```



- **Spark**
- Update App class
 - Imports

```
import static spark.Spark.*;
import spark.Request;
import spark.Response;
import spark.Route;

public class App
{
    public static void main( String[] args )
    {
        get("/hello", (req, res) -> "Hello World");
        // System.out.println( "Hello World!" );
    }
}
```

- Route

- Build: mvn clean package (error: next slide)

Java Standalone Server



- **Spark**
- **Error**

```
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.8.0:compile (default-compile) on project hello-rest: Compilation failure
[ERROR] /Users/Shared/Downloads/pomgen/hello-rest/src/main/java/com/brainwashinc/app/App.java:[16,34]
lambda expressions are not supported in -source 7
[ERROR]   (use -source 8 or higher to enable lambda expressions)
```

- **Update pom.xml maven to 1.8**

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

Java Standalone Server



- **Spark**
- **Run: e.g.,**
- **mvn exec:java -Dexec.mainClass="com.brainwashinc.app.App"**

```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
```

- **Update pom.xml**

```
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
    <version>1.7.21</version>
</dependency>
```

- **Rebuild, Rerun**

Java Standalone Server



- **Output with port: 4567**

```
[Thread-1] INFO org.eclipse.jetty.util.log - Logging initialized @1265ms to org.eclipse.jetty.util.log.Slf4jLog
[Thread-1] INFO spark.embeddedserver.jetty.EmbeddedJettyServer - == Spark has ignited!
[Thread-1] INFO spark.embeddedserver.jetty.EmbeddedJettyServer - -> Listening on 0.0.0.0:4567
[Thread-1] INFO org.eclipse.jetty.server.Server - jetty-9.4.31.v20200723; built: 2020-07-23T17:57:36.812Z; git: 450ba27947e13e66baa8cd1ce7e85a4461cacc1d; jvm 17+35-LTS-2724
[Thread-1] INFO org.eclipse.jetty.server.session - DefaultSessionIdManager workerName=node0
[Thread-1] INFO org.eclipse.jetty.server.session - No SessionScavenger set, using defaults
[Thread-1] INFO org.eclipse.jetty.server.session - node0 Scavenging every 60000ms
[Thread-1] INFO org.eclipse.jetty.server.AbstractConnector - Started ServerConnector@6becc6b5{HTTP/1.1, (http/1.1)}{0.0.0.0:4567}
[Thread-1] INFO org.eclipse.jetty.server.Server - Started @1354ms
```

- Hit with browser or curl:
 - curl http://localhost:4567/hello

Hello World

Java Standalone Server



- Change the main
 - Add a path for /user
 - Takes an id and returns the parameters

```
path("/user", () -> {
    get("/:id", (req, res) -> req.params());
});
```

- Rebuild; rerun
- Hit with browser or curl:
 - curl http://localhost:4567/user/123

{:id=123}



- **Change the main**
- **Prefix that path with a v1 path**

```
path("/v1", () -> {
    path("/user", () -> {
        get("/:id", (req, res) -> req.params());
    });
});
```

- **Rebuild; rerun**
- **Hit <http://localhost:4567/v1/user/123>**

{:id=123}



- **Add a before and after**

```
path("/v1", () -> {
    before("/*", (q, a) -> System.out.print("Received api call\n"));
    path("/user", () -> {
        get("/:id", (req, res) -> req.params());
    });
    after("/*", (q, a) -> System.out.print("Received api done\n"));
});
```

- **Rebuild; rerun**
- **Hit <http://localhost:4567/v1/user/123>**
- **See log output**

Received api call
Received api done



- **Add other methods**

```
path("/v1", () -> {
    before("/*", (q, a) -> System.out.print("Received api call\n"));
    path("/user", () -> {
        post("", (req, res) -> "add user");
        put("/:id", (req, res) -> "replace user: " + req.params());
        delete("/:id", (req, res) -> "delete user: " + req.params());
        get("/:id", (req, res) -> req.params());
    });
    after("/*", (q, a) -> System.out.print("Received api done\n"));
});
```

- **Rebuild; rerun**



- **Of course you need actually functionality for the methods...**

```
path("/v1", () -> {
    before("/*", (q, a) -> System.out.print("Received api call\n"));
    path("/user", () -> {
        post("", (req, res) -> {
            // functionality goes here
            return "add user";
        });
        put("/:id", (req, res) -> "replace user: " + req.params());
        delete("/:id", (req, res) -> "delete user: " + req.params());
        get("/:id", (req, res) -> req.params());
    });
    after("/*", (q, a) -> System.out.print("Received api done\n"));
});
```



- **Query those methods**
 - curl -X POST <http://localhost:4567/v1/user>
 - curl -X PUT <http://localhost:4567/v1/user/1>
 - curl -X GET <http://localhost:4567/v1/user/1>
 - curl -X DELETE <http://localhost:4567/v1/user/1>
- **Spark documentation:**
 - <http://sparkjava.com/documentation>

Python Standalone Server Using Flask



Python-Flask Standalone Server



- **Install python**
 - <https://www.python.org/downloads/>
- **Make a project directory (e.g., pyflaskrestful)**
- **Create and active an environment**
 - `python3 -m venv .venvrest`
 - `source .venvrest/bin/activate`
- **Install flask and flask-restful**
 - `pip3 install flask`
 - `pip3 install flask-restful`

Copyright Brainwash Inc. 2022

269

Python-Flask Standalone Server



- **Create an app.py file**
 - **Add imports**
 - **Create API object**

```
# using flask_restful
from flask import Flask, jsonify, request
from flask_restful import Resource, Api

app = Flask(__name__)
# API object
api = Api(app)
```

Copyright Brainwash Inc. 2022

270

Python-Flask Standalone Server



- **Create a Resource and add it to the API**
- **Add the run statement**

```
# class for a resource
class UserRoot(Resource):
    def get(self):
        return [{'userId': '123'}, {'userId': '1234'}]

# add the resource(s)
api.add_resource(UserRoot, '/user')

# driver
if __name__ == '__main__':
    app.run(debug = True)
```

Copyright Brainwash Inc. 2022

271

Python-Flask Standalone Server



- **Start the process: python3 app.py**
 - **Uses 127.0.0.1:5000**

```
(.venvrest) Bear-MBP:pyflaskrestful bearc2020$ python app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 255-296-312
```

Copyright Brainwash Inc. 2022

272



- **Hit the endpoint**
 - **curl http://127.0.0.1:5000/user**

```
Bear-MBP:javaserverSpark bearc2020$ curl http://127.0.0.1:5000/user
[
    {
        "userId": "123"
    },
    {
        "userId": "1234"
    }
]
```



- **Create a User level resource**
 - **Takes userId from path**
 - **Add it below the existing UserRoot**

```
class User(Resource):
    def get(self, userId):
        return {'userId': userId}

# add the resource(s)
api.add_resource(UserRoot, '/user')
api.add_resource(User, '/user/<userId>')
```



- **Hit the new endpoint**
 - **curl http://127.0.0.1:5000/user/123a**

```
Bear-MBP:javaserverSpark bearc2020$ curl http://127.0.0.1:5000/user/123a
{
    "userId": "123a"
}
```



- **Create post method to UserRoot**
 - **Get the form data from the request**

```
class UserRoot(Resource):
    def post(self):
        data = request.form['username']
        return {'username':data, 'userId':'123'}, 201
```

- **Hit the endpoint with data**

```
curl -X POST http://127.0.0.1:5000/user -d "username=bear"
```

```
Bear-MBP:javaserverSpark bearc2020$ curl -X POST http://127.0.0.1:5000/user
-d "username=bear"
{
    "username": "bear",
    "userId": "123"
```



- **Change the post endpoint to expect a JSON body**
- **Return data including body data**

```
def post(self):
    data = request.json
    return {"username":data["username"], "userId":"123"}
```



- **Hit the endpoint with JSON data and content-type header**
- `curl -X POST http://127.0.0.1:5000/user -d '{"username":"bear","email":"test@wx.com"}' -H "Content-Type: application/json"`

```
Bear-MBP:javaserverSpark bearc2020$ curl -X POST http://127.0.0.1:5000/user -d
'{"username":"bear","email":"test@wx.com"}' -H "Content-Type: application/json"
{
    "username": "bear",
    "userId": "123"
}
```

```

# using flask_restful
from flask import Flask, jsonify, request
from flask_restful import Resource, Api

app = Flask(__name__)
# API object
api = Api(app)

# class for a resource
class UserRoot(Resource):
    def post(self):
        data = request.json
        # return data
        return {"username":data["username"], "userId":"123"}
    # data = request.form['username']
    # return {'username':data, 'userId':'123'}, 201

    def get(self):
        return [{'userId': '123'}, {'userId': '1234'}]

class User(Resource):
    def get(self, userId):
        return {'userId': userId}

# add the resource(s)
api.add_resource(UserRoot, '/user')
api.add_resource(User, '/user/<userId>')

# driver
if __name__ == '__main__':
    app.run(debug = True)

```

Python-Flask Standalone Server

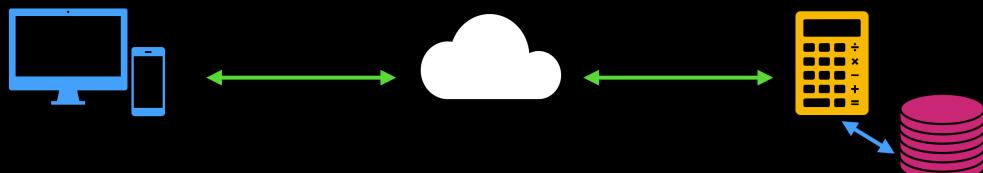
Final Full Code



279

RESTful API Design and Development

The End



Copyright

Thank you!

Continue Learning on Pluralsight

<https://app.pluralsight.com/channels/details/fcdbda2afd3a-4a44-ad09-c057fea85ce0?s=1>

The channel provides you with ways of learning beyond the instructor led API training.

Mans Böll • 56h 28m • 4 Members • Company - SEB [Edit Channel](#)

+ Add content [Add section](#) ...

Continue the learning (7h 41m)
Section with courses to continue the upskilling in API

APIS- Application Programming Interface: Executive Briefing
Course Dan Appleman - Beginner - May 26, 2020 - 30m 56s

REST Fundamentals
Course Howard Dierking - Intermediate - Dec 14, 2020 - 2h 40m

Designing RESTful Web APIs
Course Shawn Whitehead - Beginner - Aug 5, 2019 - 2h 7m

Introduction to OAuth2, OpenID Connect and JSON Web Tokens (JWT)
Course Dominick Baier - Intermediate - Jun 26, 2013 - 2h 24m

> Java (Optional) (12h 59m)
Section for API development related to Java and relevant frameworks

DevelopIntelligence
A PLURALSIGHT COMPANY

Thank you!

If you have additional questions,
please reach out to me at:
bear@brainwashinc.com

DevelopIntelligence
A PLURALSIGHT COMPANY