# OpenAPI

Eko Kurniawan Khannedy

# Eko Kurniawan Khannedy

- Technical architect at one of the biggest ecommerce company in Indonesia
- 10+ years experiences
- www.programmerzamannow.com
- youtube.com/c/ProgrammerZamanNow

# Eko Kurniawan Khannedy


- Telegram : @khannedy
- Facebook : fb.com/ProgrammerZamanNow
- Instagram : instagram.com/programmerzamannow
- Youtube : youtube.com/c/ProgrammerZamanNow
- Telegram Channel : t.me/ProgrammerZamanNow
- Email : echo.khannedy@gmail.com

# Sebelum Belajar

- HTTP
- RESTful API

# Agenda

- Pengenalan OpenAPI
- Tipe Data
- Document
- Info
- Component
- Path
- Tag
- Security
- Dan lain-lain

# Pengenalan OpenAPI

# Pengenalan OpenAPI

- OpenAPI merupakan standar spesifikasi, tidak tergantung bahasa pemrograman apapun, untuk membuat dokumentasi RESTful API.
- OpenAPI dibuat agar pengguna RESTful API tidak perlu mengakses kode aplikasi dan membaca dokumen manual (misal dalam bentuk doc, pdf) untuk memahami RESTful API yang dibuat
- OpenAPI bisa menggunakan tool untuk menampilkan secara visual, bahkan untuk membuat kode program client atau server
- https://www.openapis.org/
- https://github.com/OAI/OpenAPI-Specification
- https://swagger.io/specification/

# Document Structure

- OpenAPI memiliki struktur document yang sudah standar
- Namun kita diberi dua opsi untuk membuat document nya, bisa menggunakan JSON atau YAML
- https://www.json.org/
- https://yaml.org/

# OpenAPI Editor

- Document OpenAPI hanya menggunakan JSON atau Yaml, jadi untuk membuat document OpenAPI kita cukup menggunakan Text Editor
- Namun jika kita ingin melihat OpenAPI dalam bentuk visual, kita juga bisa menggunakan Swagger Editor : https://editor.swagger.io/
- Jika menggunakan product JetBrains, bisa menggunakan plugin OpenAPI https://www.jetbrains.com/help/idea/openapi.html
- Jika menggunakan Visual Studio Code, bisa menggunakan plugin OpenAPI https://marketplace.visualstudio.com/items?itemName=42Crunch.vscode-openapi

# Tipe Data

# Tipe Data

- Saat kita membuat RESTful API, sudah dipastikan kita akan membuat request dan response, dimana dalam data request dan response sudah dipastikan terdapat detail data
- Misal jika terdapat data Product, pasti ada id, name, price, dan lain-lain
- Semua detail data tersebut pasti memiliki tipe data
- Kita tidak bisa menggunakan tipe data yang terdapat pada bahasa pemrograman yang digunakan untuk membuat RESTful API, oleh karena itu pada OpenAPI terdapat tipe data general yang bisa digunakan yang dapat dimengerti di semua bahasa pemrograman

# OpenAPI Tipe Data (1)

| type | format | Comments |
| --- | --- | --- |
| integer | int32 | signed 32 bits |
| integer | int64 | signed 64 bits (a.k.a long) |
| number | float | |
| number | double | |
| string | | |

# OpenAPI Tipe Data (2)

| | | |
|---|---|---|
| `string` | `byte` | base64 encoded characters |
| `string` | `binary` | any sequence of octets |
| `boolean` | | |
| `string` | `date` | As defined by `full-date` - RFC3339 |
| `string` | `date-time` | As defined by `date-time` - RFC3339 |
| `string` | `password` | A hint to UIs to obscure input. |

# Document

# Document

- OpenAPI sangatlah sederhana, kita hanya perlu membuat satu file berisi semua data OpenAPI nya
- OpenAPI memiliki struktur yang sudah ditentukan ketika membuat document nya
- Kita bisa menggunakan JSON atau YAML untuk file nya
- https://spec.openapis.org/oas/v3.0.3#openapi-object

# OpenAPI Object (1)

| 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 `openapi` field *SHOULD* be used by tooling specifications and clients to interpret the OpenAPI document. This is *not* related to the API `info.version` 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 `servers` property is not provided, or is an empty array, the default value would be a Server Object with a url value of `/`. |
| paths | Paths Object | ***REQUIRED***. The available paths and operations for the API. |

# OpenAPI Object (2)

| | | |
|---|---|---|
| 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. |

# Kode : OpenAPI Document

```json
{
  "openapi": "3.0.3",
  "info": {},
  "servers": [],
  "paths": {}
}
```

# Info

# Info

- Info merupakan bagian dari informasi metadata tentang API yang kita buat
- Kita bisa memasukkan author, lisensi, dan lain-lain

# Info Object

| | | |
|---|---|---|
| title | `string` | ***REQUIRED***. The title of the API. |
| description | `string` | A short description of the API. CommonMark syntax *MAY* be used for rich text representation. |
| termsOfService | `string` | A URL to the Terms of Service for the API. *MUST* be in the format of a URL. |
| contact | Contact Object | The contact information for the exposed API. |
| license | License Object | The license information for the exposed API. |
| version | `string` | ***REQUIRED***. The version of the OpenAPI document (which is distinct from the OpenAPI Specification version or the API implementation version). |

# Contact Object

| | | |
|---|---|---|
| name | `string` | The identifying name of the contact person/organization. |
| url | `string` | The URL pointing to the contact information. *MUST* be in the format of a URL. |
| email | `string` | The email address of the contact person/organization. *MUST* be in the format of an email address. |

# License Object

| | | |
|---|---|---|
| name | `string` | ***REQUIRED***. The license name used for the API. |
| url | `string` | A URL to the license used for the API. *MUST* be in the format of a URL. |

# Kode : Info

```
"info": {
  "title": "TodoList RESTful API",
  "description": "OpenAPI for TodoList RESTful API",
  "version": "1",
  "contact": {
    "name": "Eko Kurniawan Khanendy",
    "email": "echo.khannedy@gmail.com",
    "url": "https://www.programmerzamannow.com"
  },
  "license": {
    "name": "APACHE 2.0",
    "url": "https://www.apache.org/licenses/LICENSE-2.0"
  }
```

# Server

# Server

- Saat kita membuat API sudah pasti terdapat server RESTful API yang nanti akan kita buat
- Kita bisa memberitahu server yang tersedia di OpenAPI
- Misal, terdapat server development, staging, production dan lain-lain

# Server Object

| | | |
|---|---|---|
| url | `string` | ***REQUIRED***. A URL to the target host. This URL supports Server Variables and *MAY* be relative, to indicate that the host location is relative to the location where the OpenAPI document is being served. Variable substitutions will be made when a variable is named in {brackets}. |
| description | `string` | An optional string describing the host designated by the URL. CommonMark syntax *MAY* be used for rich text representation. |
| variables | Map[`string`, Server Variable Object] | A map between a variable name and its value. The value is used for substitution in the server's URL template. |

# Kode : Server

```
"servers": [
  {
    "url": "https://{environment}.programmerzamannow.com/api/v1",
    "description": "TodoList RESTful API Server",
    "variables": {
      "environment": {
        "default": "dev",
        "description": "Server Environment",
        "enum": [
          "dev",
          "qa",
          "prod"
        ]
```

# External Documentation

# External Documentation

- External documentation merupakan bagian dalam OpenAPI jika kita ingin menambahkan link tambahan dalam OpenAPI
- Bisa menuju link documentation lain, atau mungkin link menuju website

# External Documentation Object

| | | |
|---|---|---|
| description | `string` | A short description of the target documentation. CommonMark syntax *MAY* be used for rich text representation. |
| url | `string` | ***REQUIRED***. The URL for the target documentation. Value *MUST* be in the format of a URL. |

# Kode : External Documentation

```
  "externalDocs": {
    "description": "Youtube Programmer Zaman Now",
    "url": "https://www.youutube.com/c/ProgrammerZamanNow"
  }
}
```

# Path

# Path

- Path merupakan representasi endpoint API di OpenAPI
- Pada Path, kita tidak perlu menuliskan seluruh URL, cukup url di belakang setelah lokasi server

# Path Object

| | | |
|---|---|---|
| /{path} | Path Item Object | A relative path to an individual endpoint. The field name *MUST* begin with a forward slash (`/`). The path is **appended** (no relative URL resolution) to the expanded URL from the `Server Object`'s `url` field in order to construct the full URL. Path templating is allowed. When matching URLs, concrete (non-templated) paths would be matched before their templated counterparts. Templated paths with the same hierarchy but different templated names *MUST NOT* exist as they are identical. In case of ambiguous matching, it's up to the tooling to decide which one to use. |

# Path Item Object (1)

| | | |
|---|---|---|
| $ref | `string` | Allows for an external definition of this path item. The referenced structure *MUST* be in the format of a Path Item Object. In case a Path Item Object field appears both in the defined object and the referenced object, the behavior is undefined. |
| summary | `string` | An optional, string summary, intended to apply to all operations in this path. |
| description | `string` | An optional, string description, intended to apply to all operations in this path. CommonMark syntax *MAY* be used for rich text representation. |

# Path Item Object (2)

| | | |
|---|---|---|
| servers | [Server Object] | An alternative `server` array to service all operations in this path. |
| parameters | [Parameter Object \| Reference Object] | A list of parameters that are applicable for all the operations described under this path. These parameters can be overridden at the operation level, but cannot be removed there. The list *MUST NOT* include duplicated parameters. A unique parameter is defined by a combination of a name and location. The list can use the Reference Object to link to parameters that are defined at the OpenAPI Object's components/parameters. |

# Path Operation

| | | |
|---|---|---|
| get | Operation Object | A definition of a GET operation on this path. |
| put | Operation Object | A definition of a PUT operation on this path. |
| post | Operation Object | A definition of a POST operation on this path. |
| delete | Operation Object | A definition of a DELETE operation on this path. |
| options | Operation Object | A definition of a OPTIONS operation on this path. |
| head | Operation Object | A definition of a HEAD operation on this path. |
| patch | Operation Object | A definition of a PATCH operation on this path. |
| trace | Operation Object | A definition of a TRACE operation on this path. |

# Kode : Path

```
"paths": {
  "/todolist" : {
    "get": {},
    "post": {}
  },
  "/todolist/{todolistId}" : {
    "delete" : {},
    "put" : {}
  }
}
```

# Operation

# Operation

- Setiap Path yang kita buat di OpenAPI, bisa memiliki lebih dari satu Operation
- Hal ini dikarenakan, dalam HTTP, satu URL bisa memiliki beberapa HTTP Method
- Misal url untuk mengambil semua data dan membuat data baru, mungkin url nya sama, yang membedakan adalah HTTP Method nya, GET untuk mengambil data, POST untuk membuat data
- Inti dari API Documentation adalah dokumentasi operation yang terdapat pada RESTful API yang kita buat

# Operation Object (1)

| | | |
|---|---|---|
| tags | [`string`] | A list of tags for API documentation control. Tags can be used for logical grouping of operations by resources or any other qualifier. |
| summary | `string` | A short summary of what the operation does. |
| description | `string` | A verbose explanation of the operation behavior. CommonMark syntax *MAY* be used for rich text representation. |
| externalDocs | External Documentation Object | Additional external documentation for this operation. |
| operationId | `string` | Unique string used to identify the operation. The id *MUST* be unique among all operations described in the API. The operationId value is **case-sensitive**. Tools and libraries *MAY* use the operationId to uniquely identify an operation, therefore, it is *RECOMMENDED* to follow common programming naming conventions. |

# Operation Object (2)

| | | |
|---|---|---|
| parameters | [Parameter Object \| Reference Object] | A list of parameters that are applicable for this operation. If a parameter is already defined at the Path Item, the new definition will override it but can never remove it. The list *MUST NOT* include duplicated parameters. A unique parameter is defined by a combination of a name and location. The list can use the Reference Object to link to parameters that are defined at the OpenAPI Object's components/parameters. |
| requestBody | Request Body Object \| Reference Object | The request body applicable for this operation. The `requestBody` is only supported in HTTP methods where the HTTP 1.1 specification [RFC7231] has explicitly defined semantics for request bodies. In other cases where the HTTP spec is vague, `requestBody` *SHALL* be ignored by consumers. |
| responses | Responses Object | ***REQUIRED***. The list of possible responses as they are returned from executing this operation. |

# Operation Object (3)

| | | |
|---|---|---|
| callbacks | Map[`string`, Callback Object \| Reference Object] | A map of possible out-of band callbacks related to the parent operation. The key is a unique identifier for the Callback Object. Each value in the map is a Callback Object that describes a request that may be initiated by the API provider and the expected responses. |
| deprecated | `boolean` | Declares this operation to be deprecated. Consumers *SHOULD* refrain from usage of the declared operation. Default value is `false`. |
| security | [Security Requirement Object] | A declaration of which security mechanisms can be used for this operation. 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. To make security optional, an empty security requirement (`{}`) can be included in the array. This definition overrides any declared top-level `security`. To remove a top-level security declaration, an empty array can be used. |
| servers | [Server Object] | An alternative `server` array to service this operation. If an alternative `server` object is specified at the Path Item Object or Root level, it will be overridden by this value. |

# Kode : Operation

```
"paths": {
  "/todolist": {
    "get": {
      "summary": "Get All Todolist",
      "description": "Only will return active todolist, complete todolist or deleted todolist will be removed",
      "responses": {}
    },
    "post": {
      "summary": "Create new Todolist",
      "description": "Create new active todolist",
      "responses": {}
    }
  },
```

# Parameter

# Parameter

- Parameter adalah data yang dikirim tidak melalui Request Body
- Operation bisa memiliki parameter lebih dari satu
- OpenAPI mendukung beberapa jenis parameter, yaitu Query Parameter, Path Variable, Header, dan Cookie
- Kita bisa menambahkan parameter pada Operation, sehingga pengguna bisa tahu bahwa ada parameter yang perlu dikirim ketika memanggil Operation tersebut

# Parameter Object (1)

| | | |
|---|---|---|
| name | `string` | ***REQUIRED***. The name of the parameter. Parameter names are *case sensitive*.<br>• If `in` is `"path"`, the `name` field *MUST* correspond to a template expression occurring within the path field in the Paths Object. See Path Templating for further information.<br>• If `in` is `"header"` and the `name` field is `"Accept"`, `"Content-Type"` or `"Authorization"`, the parameter definition *SHALL* be ignored.<br>• For all other cases, the `name` corresponds to the parameter name used by the `in` property. |
| in | `string` | ***REQUIRED***. The location of the parameter. Possible values are `"query"`, `"header"`, `"path"` or `"cookie"`. |
| description | `string` | A brief description of the parameter. This could contain examples of use. CommonMark syntax *MAY* be used for rich text representation. |

# Parameter Object (2)

| | | |
|---|---|---|
| required | `boolean` | Determines whether this parameter is mandatory. If the parameter location is `"path"`, this property is ***REQUIRED*** and its value *MUST* be `true`. Otherwise, the property *MAY* be included and its default value is `false`. |
| deprecated | `boolean` | Specifies that a parameter is deprecated and *SHOULD* be transitioned out of usage. Default value is `false`. |
| allowEmptyValue | `boolean` | Sets the ability to pass empty-valued parameters. This is valid only for `query` parameters and allows sending a parameter with an empty value. Default value is `false`. If `style` is used, and if behavior is `n/a` (cannot be serialized), the value of `allowEmptyValue` *SHALL* be ignored. Use of this property is *NOT RECOMMENDED*, as it is likely to be removed in a later revision. |

# Kode : Parameter (1)

```
"get": {
  "summary": "Get All Todolist",
  "description": "Only will return active todolist, complete todolist or deleted todolist will be removed",
  "responses": {},
  "parameters": [
    {
      "name": "include_done",
      "description": "Include done todolist in the result",
      "required": false,
      "in": "query"
    },
    {
      "name": "name",
      "description": "Search todolist by name",
      "required": false,
      "in": "query"
    }
```

# Kode : Parameter (2)

```
"/todolist/{todolistId}": {
  "delete": {
    "summary": "Delete todolist",
    "description": "Delete todolist permanently",
    "responses": {},
    "parameters": [
      {
        "name": "todolistId",
        "description": "Todolist Id for deleted",
        "in": "path",
        "required": true
      }
```

# Schema

# Schema

- Saat kita membuat parameter, kita mungkin ingin memberitahu tentang tipe data untuk parameter tersebut
- Parameter memiliki attribute bernama schema, dimana schema sebenarnya sangat kompleks, kita akan bahas secara bertahap
- Dimulai dari simple schema, misal tipe data yang sebelumnya sudah kita bahas

# JSON Schema Specification

- Schema menggunakan JSON Schema untuk mendefinisikan struktur datanya
- JSON Schema bisa berisikan tipe sederhana, seperti number, string, boolean dan lain-lain, atau bahkan tipe data kompleks, seperti object dan array (yang akan kita bahas nanti)
- https://json-schema.org/
- https://json-schema.org/draft/2020-12/json-schema-core.html

# Kode : Schema

```
{
  "name": "include_done",
  "description": "Include done todolist
  "required": false,
  "in": "query",
  "schema": {
    "type": "boolean",
    "default": true
  }
},
```

```
{
  "name": "name",
  "description": "Search todolist by na
  "required": false,
  "in": "query",
  "schema": {
    "type": "string"
  }
}
```

# JSON Schema Validation

- Schema mendukung JSON Schema Validation
- Hal ini membuat kita bisa memberitahu validasi yang diperlukan ketika pengguna membaca OpenAPI kita
- Dengan kita tambahkan Schema Validation, pengguna RESTful API kita bisa tahu validation yang kita buat tanpa harus kita buat dokumentasi secara terpisah
- https://json-schema.org/draft/2020-12/json-schema-validation.html

# Kode : Schema Validation

```
{
  "name": "include_done",
  "description": "Include done t
  "required": false,
  "in": "query",
  "schema": {
    "type": "boolean",
    "default": true,
    "nullable": true
  }
},
```

```
{
  "name": "name",
  "description": "Search todolist by
  "required": false,
  "in": "query",
  "schema": {
    "type": "string",
    "nullable": false,
    "maxLength": 100,
    "minLength": 1
  }
```

# Request Body

# Request Body

- OpenAPI juga mendukung dokumentasi untuk request body
- Dengan ini, kita memberi tahu tentang schema request body yang harus dikirim ketika menggunakan RESTful API kita
- Request body mendukung schema, sehingga kita memberi tahu bentuk schema format data, bahkan validasi yang diperlukan

# Request Body Object

| | | |
|---|---|---|
| description | `string` | A brief description of the request body. This could contain examples of use. CommonMark syntax *MAY* be used for rich text representation. |
| content | Map[`string`, Media Type Object] | ***REQUIRED***. The content of the request body. The key is a media type or [media type range]appendix-D) and the value describes it. For requests that match multiple keys, only the most specific key is applicable. e.g. text/plain overrides text/* |
| required | `boolean` | Determines if the request body is required in the request. Defaults to `false`. |

# Media Type Object

| | | |
|---|---|---|
| schema | Schema Object \| Reference Object | The schema defining the content of the request, response, or parameter. |
| example | Any | Example of the media type. The example object *SHOULD* be in the correct format as specified by the media type. The `example` field is mutually exclusive of the `examples` field. Furthermore, if referencing a `schema` which contains an example, the `example` value *SHALL override* the example provided by the schema. |
| examples | Map[ `string`, Example Object \| Reference Object] | Examples of the media type. Each example object *SHOULD* match the media type and specified schema if present. The `examples` field is mutually exclusive of the `example` field. Furthermore, if referencing a `schema` which contains an example, the `examples` value *SHALL override* the example provided by the schema. |
| encoding | Map[`string`, Encoding Object] | A map between a property name and its encoding information. The key, being the property name, *MUST* exist in the schema as a property. The encoding object *SHALL* only apply to `requestBody` objects when the media type is `multipart` or `application/x-www-form-urlencoded`. |

# Kode : Request Body

```
"post": {
  "summary": "Create new Todolist",
  "description": "Create new active todolist",
  "requestBody": {
    "required": true,
    "content": {
      "application/json": {
        "schema": {}
      }
    }
  },
```

# Object Schema

# Object Schema

- Sebelumnya kita hanya membuat schema sederhana, seperti schema tipe data boolean atau string
- Spesifikasi di JSON Schema juga mendukung pembuatan schema berupa objek, yaitu data yang memiliki attribute lebih dari satu

# Kode : Object Schema

```
"application/json": {
  "schema": {
    "type": "object",
    "properties": {
      "name": {
        "type": "string",
        "required": true,
        "minLength": 1
      },
      "priority": {
        "type": "integer",
        "required": true,
        "minimum": 1,
```

# Array Schema

# Array Schema

- Selain tipe data object, Schema juga mendukung tipe data array
- Saat membuat tipe data array, kita juga bisa menentukan tipe data items yang terdapat di array, bisa tipe data sederhana, bisa tipe data kompleks seperti object atau array lagi

# Kode : Array Schema

```
"properties": {
  "name": {
    "type": "string",
    "required": true,
    "minLength": 1
  },
  "tags": {
    "type": "array",
    "items": {
      "type": "string",
      "minLength": 1
    }
  },
```

# Example

# Example

- OpenAPI mendukung example data
- Example data merupakan data contoh yang bisa kita tambahkan baik itu di Parameter, Request Body dan Response Body

# Kode : Example di Parameter

```
},
{
  "name": "name",
  "description": "Search todolist by name",
  "required": false,
  "in": "query",
  "schema": {
    "type": "string"
  },
  "examples": {
    "java" : {
      "value": "Java"
    },
    "php" : {
      "value": "PHP"
    }
```

# Kode : Example di Request Body

```
"application/json": {
  "examples": {
    "java": {
      "value": {
        "name": "Belajar Pemrograman Java",
        "tags": ["Java","Pemrograman"],
        "priority": 3
      }
    },
    "php": {
      "value": {
        "name": "Belajar Pemrograman PHP",
        "tags": ["PHP","Pemrograman"],
        "priority": 2
      }
```

# Response Body

# Response Body

- Selain Request Body, kita juga mendokumentasikan Response Body di OpenAPI
- Dengan demikian, kita bisa memberi tahu format Response Body yang kita kembalikan pada RESTful API yang kita buat
- Yang menarik di OpenAPI, kita bisa memberi tahu format response body sesuai dengan response status nya, misal kita bisa tentukan untuk response status 200, 400, 500, dan lain-lain

# Response Body Object

| | | |
|---|---|---|
| HTTP Status Code | Response Object \| Reference Object | Any HTTP status code can be used as the property name, but only one property per code, to describe the expected response for that HTTP status code. A Reference Object can link to a response that is defined in the OpenAPI Object's components/responses section. This field *MUST* be enclosed in quotation marks (for example, "200") for compatibility between JSON and YAML. To define a range of response codes, this field *MAY* contain the uppercase wildcard character `X`. For example, `2XX` represents all response codes between `[200-299]`. Only the following range definitions are allowed: `1XX`, `2XX`, `3XX`, `4XX`, and `5XX`. If a response is defined using an explicit code, the explicit code definition takes precedence over the range definition for that code. |

# Response Object

| | | |
|---|---|---|
| description | `string` | ***REQUIRED***. A short description of the response. CommonMark syntax *MAY* be used for rich text representation. |
| headers | Map[`string`, Header Object \| Reference Object] | Maps a header name to its definition. [RFC7230] states header names are case insensitive. If a response header is defined with the name `"Content-Type"`, it *SHALL* be ignored. |
| content | Map[`string`, Media Type Object] | A map containing descriptions of potential response payloads. The key is a media type or [media type range]appendix-D) and the value describes it. For responses that match multiple keys, only the most specific key is applicable. e.g. text/plain overrides text/* |
| links | Map[`string`, Link Object \| Reference Object] | A map of operations links that can be followed from the response. The key of the map is a short name for the link, following the naming constraints of the names for Component Objects. |

# Kode : Response Body

# Tag

# Tag

- Tag merupakan fitur tambahan, dimana kita bisa grouping beberapa Path menjadi satu Tag yang sama
- Misal ketika kita membuat OpenAPI untuk RESTful API Toko Online, kita grouping dengan Tag Product, Customer, Payment, Order, dan lain-lain

# Kode : Tag

```
"paths": {
  "/todolist": {
    "get": {
      "tags": ["Todolist"],
      "summary": "Get All Todolist",
      "description": "Only will return active todolist, complete
      "responses": {...},
      "parameters": [...]
    },
    "post": {
      "tags": ["Todolist"],
      "summary": "Create new Todolist",
      "description": "Create new active todolist",
      "requestBody": {"required": true...},
      "responses": {}
    }
  },
```

# Component

# Component

- Component merupakan bagian dalam OpenAPI untuk menyimpan object yang bisa digunakan ulang
- Misal, saat nanti kita membuat spek untuk Request Body atau Response Body, dibanding kita buat satu persatu, jika ada beberapa yang sama, lebih baik kita buat dalam Component, sehingga bisa digunakan di beberapa Endpoint API
- Ada banyak jenis component, ada schema, request, response, parameter, header, dan lain-lain

# Component Object (1)

| | | |
|---|---|---|
| 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. |

# Component Object (2)

| | | |
|---|---|---|
| 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. |

# Kode : Component

```
"components": {
  "schemas": {
    "Todolist": {
      "type": "object",
      "properties": {
        "id": {
          "type": "string"
        },
        "name": {
          "type": "string"
        },
        "tags": {
          "type": "array",
          "items": {
            "type": "string"
          }
```

```
},
"parameters": {
  "todolistId": {
    "in": "path",
    "name": "todolistId",
    "schema": {
      "type": "string",
      "required": true
    }
  }
}
```

# Reference

# Reference

- OpenAPI memiliki fitur reference, dimana dengan reference kita bisa membuat reference ke data component yang sudah kita buat
- Hal ini lebih baik, daripada kita buat component yang sama berkali kali pada beberapa path, misal jika terdapat response body yang sama, lebih baik kita gunakan reference, jika memiliki parameter yang sama, lebih baik kita gunakan reference

# Kode : Reference pada Parameter

```
"/todolist/{todolistId}": {
  "delete": {
    "tags": [...],
    "summary": "Delete todolist",
    "description": "Delete todolist permanently",
    "responses": {},
    "parameters": [
      {
        "$ref": "#/components/parameters/todolistId"
      }
    ]
  },
  "put": {
    "parameters": [
      {
        "$ref": "#/components/parameters/todolistId"
      }
```

# Kode : Reference pada Response Body

```
"/todolist": {
  "get": {
    "tags": [...],
    "summary": "Get All Todolist",
    "description": "Only will return active todolist, com
    "responses": {
      "200": {
        "description": "Get Todolist Success",
        "content": {
          "application/json": {
            "schema": {
              "type": "array",
              "items": {
                "$ref": "#/components/schemas/Todolist"
              }
```

```
"post": {
  "tags": [...],
  "summary": "Create new Todolist",
  "description": "Create new active todolist",
  "requestBody": {"required": true...},
  "responses": {
    "200": {
      "description": "Success create todolist",
      "content": {
        "application/json": {
          "schema": {
            "$ref": "#/components/schemas/Todolist"
          }
        }
```

# Security

# Security

- OpenAPI mendukung dokumentasi untuk Security RESTful API
- Dengan ini kita bisa memberitahu pengguna mekanisme Security apa yang kita gunakan di RESTful API kita
- Ada banyak format security yang didukung oleh OpenAPI, seperti apiKey (query param, header, cookie), http (basic auth, bearer token), oauth2 dan openIdConnect
- Sebelum menggunakan fitur Security, kita perlu membuat requirement Security terlebih dahulu di Component, baru kita bisa gunakan di Operation

# Security Object (1)

| Field Name | Type | Applies To | Description |
| --- | --- | --- | --- |
| type | `string` | Any | ***REQUIRED***. The type of the security scheme. Valid values are `"apiKey"`, `"http"`, `"oauth2"`, `"openIdConnect"`. |
| description | `string` | Any | A short description for security scheme. CommonMark syntax *MAY* be used for rich text representation. |
| name | `string` | `apiKey` | ***REQUIRED***. The name of the header, query or cookie parameter to be used. |
| in | `string` | `apiKey` | ***REQUIRED***. The location of the API key. Valid values are `"query"`, `"header"` or `"cookie"`. |

# Security Object (1)

| | | | |
|---|---|---|---|
| scheme | `string` | `http` | ***REQUIRED***. The name of the HTTP Authorization scheme to be used in the Authorization header as defined in [RFC7235]. The values used *SHOULD* be registered in the IANA Authentication Scheme registry. |
| bearerFormat | `string` | `http` (`"bearer"`) | A hint to the client to identify how the bearer token is formatted. Bearer tokens are usually generated by an authorization server, so this information is primarily for documentation purposes. |
| flows | OAuth Flows Object | `oauth2` | ***REQUIRED***. An object containing configuration information for the flow types supported. |
| openIdConnectUrl | `string` | `openIdConnect` | ***REQUIRED***. OpenId Connect URL to discover OAuth2 configuration values. This *MUST* be in the form of a URL. |

# Kode : Security

```
},
"components": {
  "securitySchemes": {
    "TodolistAuth": {
      "type": "apiKey",
      "in": "header",
      "name": "X-API-Key"
    }
  },
```

# Kode : Security di Operation

```
"/todolist": {
  "get": {
    "security": [
      {
        "TodolistAuth": []
      }
    ],
    "tags": [...],
    "summary": "Get All Todolist",
    "description": "Only will return a
    "responses": {...},
    "parameters": [...]
  },
```

```
"post": {
  "security": [
    {
      "TodolistAuth": []
    }
  ],
  "tags": [...],
  "summary": "Create new Todolist
  "description": "Create new acti
  "requestBody": {
    "required": true,
    "content": {...}
  },
```

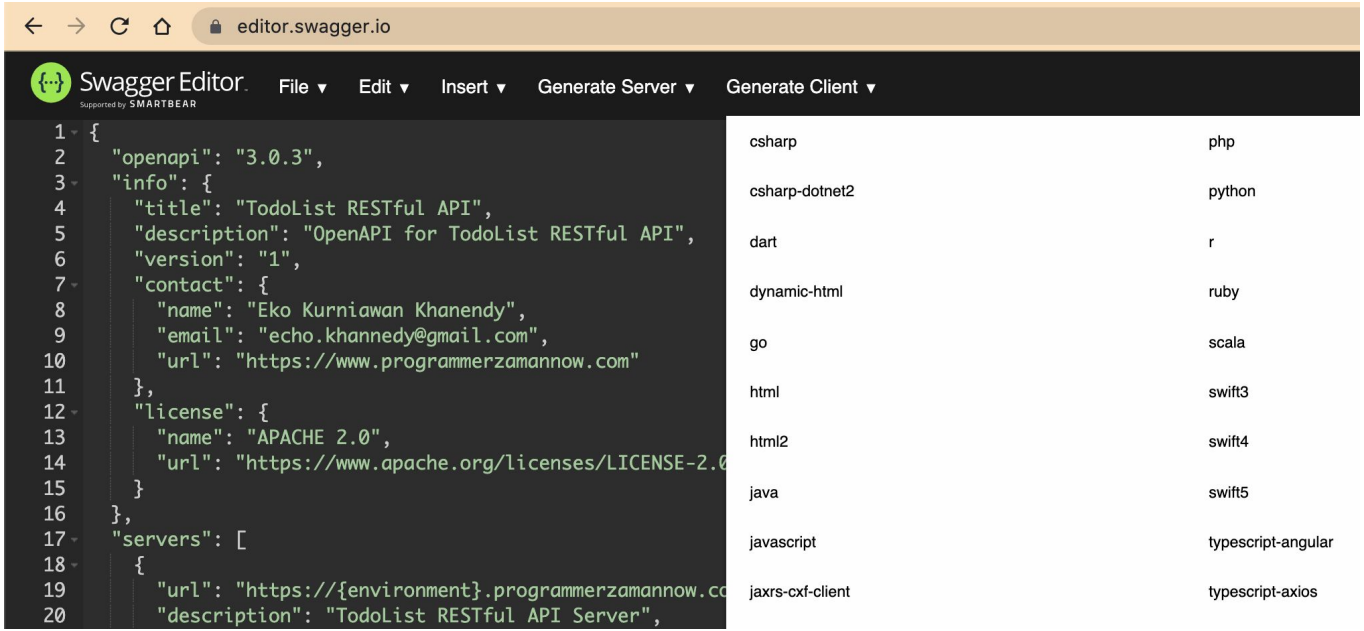
```
"/todolist/{todolistId}": {
  "delete": {
    "security": [
      {
        "TodolistAuth": []
      }
    ],
    "tags": [
      "Todolist"
    ],
    "summary": "Delete todolist",
    "description": "Delete todolist p
    "responses": {},
    "parameters": [
      {
```

# Code Generator

# Code Generator

- OpenAPI adalah spesifikasi yang standard, dan banyak diadopsi oleh banyak perusahaan
- Bahkan kita bisa melakukan code generator sesuai dengan bahasa pemrograman yang kita inginkan
- Contohnya kita bisa menggunakan https://editor.swagger.io/ untuk membuat code client atau server sesuai dengan OpenAPI file yang kita buat
- Hal ini bisa mempermudah ketika kita butuh membuat kode client atau server yang sesuai dengan OpenAPI spec yang sudah dibuat

# Swagger Code Generator



# Penutup

# Penutup

- Masih banyak fitur yang terdapat di OpenAPI, silahkan pelajari langsung di halaman websitenya
- Mulai sekarang selalu gunakan OpenAPI untuk membuat dokumentasi RESTful API
- Buatlah OpenAPI sebelum kita membuat aplikasi RESTful API
- OpenAPI bisa digunakan sebagai kontrak kesepakatan antara yang akan membuat RESTful API dan yang akan menggunakan RESTful API