HTTP API Basics

Course: Information Modelling

HTTP verbs & responses

VERB	CRUD-OPERATION	RESPONSES
POST	create	201 *(created)*, 404 *(not found)*, 409 *(conflict)*
GET	read	200 *(ok)*, 404 *(not found)*
PUT	update (replace)	200 *(ok)*, 204 *(no content)*, 404 *(not found)*, 405 *(method not allowed)*
PATCH	modify (partial)	200 *(ok)*, 204 *(no content)*, 404 *(not found)*, 405 *(method not allowed)*
DELETE	delete	200 *(ok)*, 404 *(not found)*, 405 *(method not allowed)*

*There are a number of other verbs, too, but they are utilized less frequently. Of those less-frequent methods, **OPTIONS** and **HEAD** are used more often than others. Others are CONNECT and TRACE.*

HTTP status codes

HTTP response status codes ***indicate whether*** *a specific* ***HTTP request*** *has* ***been successfully completed****. Responses are grouped in five classes:*

100–199	Informational responses
200–299	**Successful responses**
300–399	Redirection messages
400–499	**Client error responses**
500–599	Server error responses

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Semantic URLs

course: Information Modelling

Use plural nouns for collections

In general, always use the **plural form** to point to collections. In some rare cases, the noun used already implies that it contains many items, e.g. *history*. In this case, the singular form is fine, or one may opt to use another noun, e.g. *events*.

```
/games
/users
/books/{book-id}
/games/{game-id}/users
/history/{timestamp}
/events
/events/{event-id}
```

Use the appropriate parameters

```
GET /posts/{id}/comments?from=2019-0402T04:07:34.0218628Z
```

Use the appropriate parameters

```
GET /posts/{id}/comments?from=2019-0402T04:07:34.0218628Z
```

PATH parameter
resource by id, name, ...
Always required!

Use the appropriate parameters

```
GET /posts/{id}/comments?from=2019-0402T04:07:34.0218628Z
```

PATH parameter
resource by id, name, ...
Always required!

QUERY parameter
filter/search, sort, group, ...

Use the appropriate parameters

```
GET /posts/{id}/comments?from=2019-0402T04:07:34.0218628Z
```

PATH parameter
resource by id, name, ...
Always required!

QUERY parameter
filter/search, sort, group, ...

```
GET /games/{id}/events/{event-id}
GET /games/{id}/events/{timestamp}
POST /games/{id}/events
```

Use the appropriate parameters

```
GET /posts/{id}/comments?from=2019-0402T04:07:34.0218628Z
```

PATH parameter
resource by id, name, …
Always required!

QUERY parameter
filter/search, sort, group, …

```
GET /games/{id}/events/{event-id}
GET /games/{id}/events/{timestamp}
POST /games/{id}/events
    Authentication: …

    {
        …
    }
```

HEADER parameter
context, metadata (describing the request message), authentication, …

Use the appropriate parameters

```
GET /posts/{id}/comments?from=2019-0402T04:07:34.0218628Z
```

PATH parameter
resource by id, name, …
Always required!

QUERY parameter
filter/search, sort, group, …

```
GET /games/{id}/events/{event-id}
GET /games/{id}/events/{timestamp}
POST /games/{id}/events
    Authentication: …

    {
      …
    }
```

HEADER parameter
context, metadata (describing the request message), authentication, …

REQUEST BODY
Used in POST, PATCH, PUT
filter/search, sort: in query!

Sync vs Async communication

course: Information Modeling

Synchronous communication

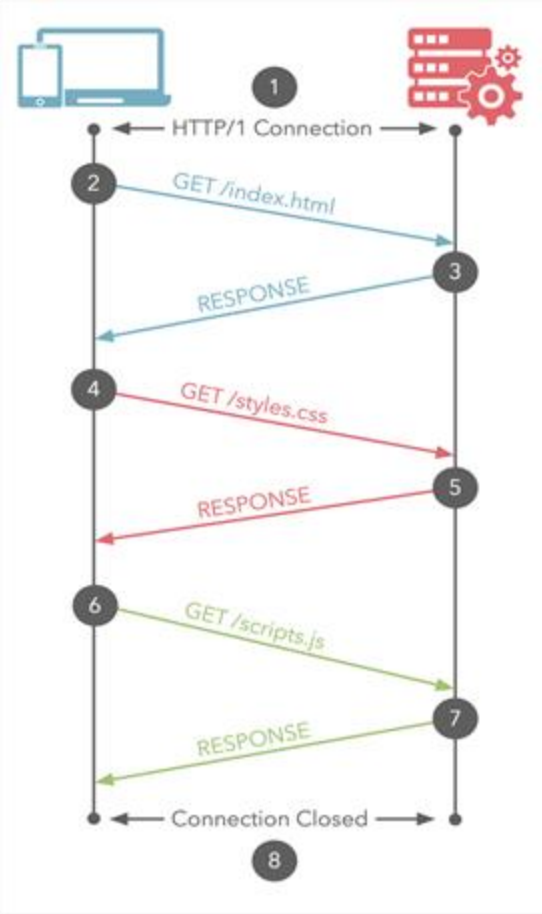
Messages are exchanged in real time

Synchronous communication is a communication type that takes place in **real time** between two or more entities.

Asynchronous communication

Interaction without real-time conversation

With asynchronous communication you do not require an **immediate response** from your peers.




```
http(s):" "//" host [ ":" port ] path [ "?" query ]
```

Synchronous

HTTP

No concurrent requests over a single connection

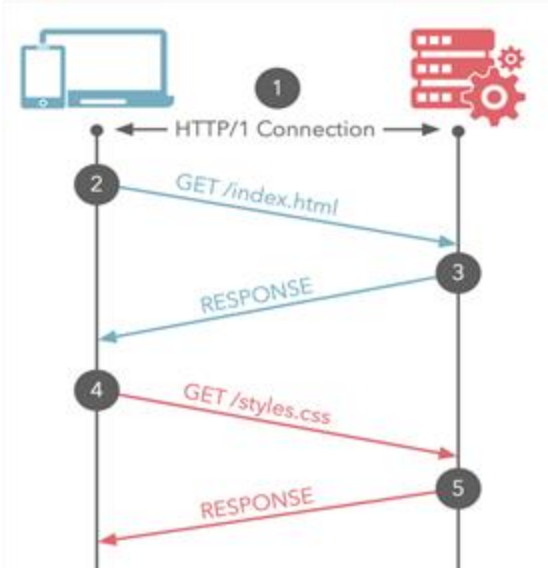
Multiple connections are needed for subsequent requests

Plain text

https://cascadingmedia.com/insites/2015/03/http-2.html

Synchronous

HTTP



No concurrent requests over a single connection

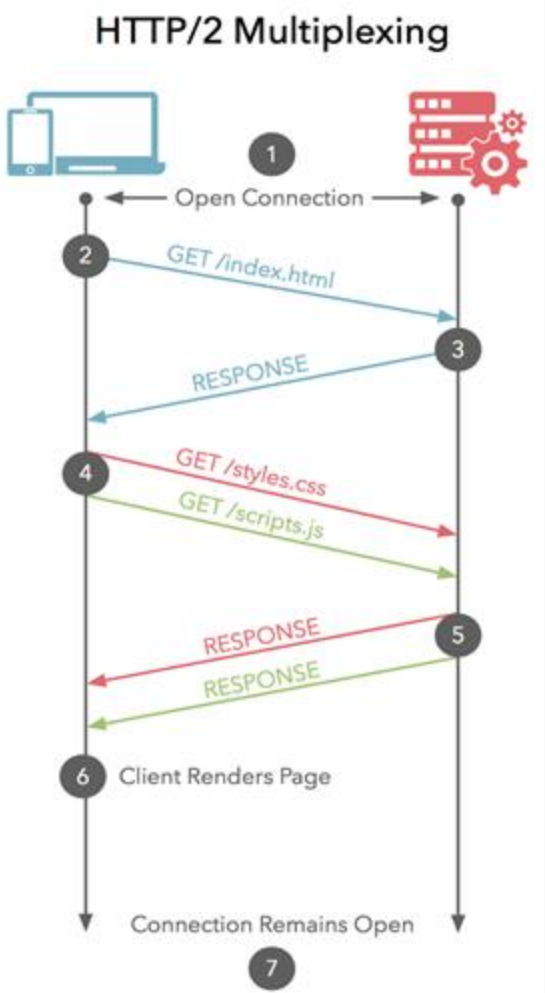
Multiple connections are needed for subsequent requests

Plain text

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="styles.css">
    </head>
    <body>
        <h1>This is a heading</h1>
        <p>This is a paragraph.</p>
    </body>
</html>
```

```
http(s):" "//" host [ ":" port ] path [ "?" query ]
```

https://cascadingmedia.com/insites/2015/03/http-2.html



Asynchronous

HTTP/2

Multiple requests-response messages over a single connection at once

Binary instead of plain text

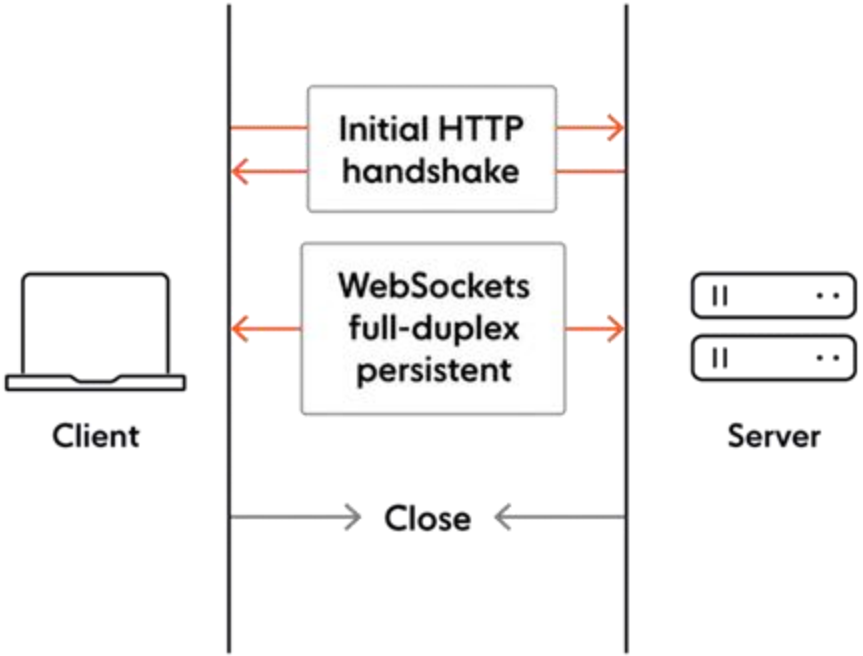
Streaming

Server push

https://cascadingmedia.com/insites/2015/03/http-2.html

Asynchronous

WebSockets



DO NOT USE HTTP-scheme
ws: or **wss:**

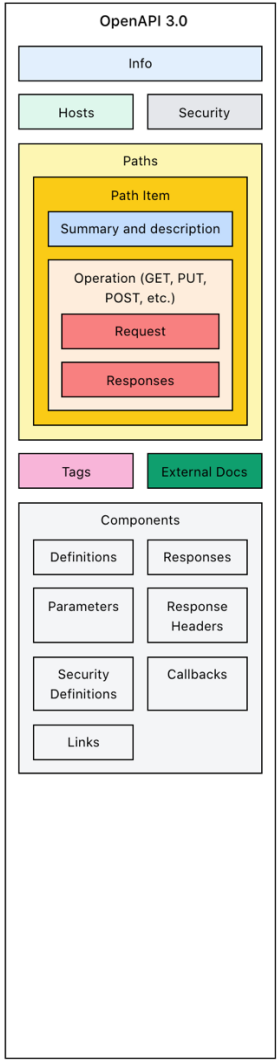
Full-duplex communication over TCP/IP
Data is sent in chunks

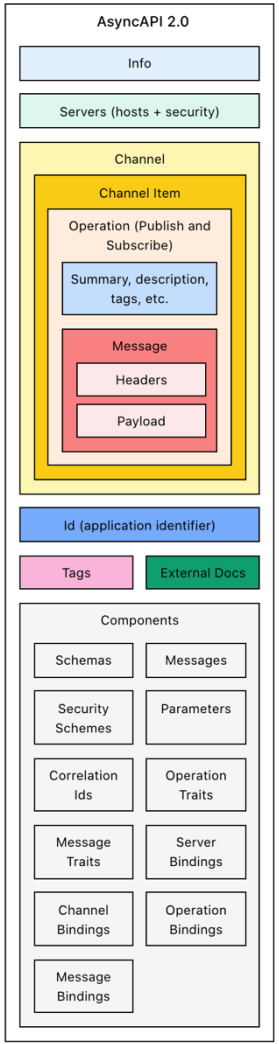
real-time result **feeds**

```
ws(s):" "//" host [ ":" port ] path [ "?" query ]
```

https://sookocheff.com/post/networking/how-do-websockets-work/

OPENAPI





ASYNC API

OpenAPI specs

https://spec.openapis.org/oas/v3.1.0.html

course: Information Modelling

Syntax of OpenAPI description (OAD)

JSON

```json
{
  "anObject": {
    "aNumber": 42,
    "aString": "This is a string",
    "aBoolean": true,
    "nothing": null,
    "arrayOfNumbers": [
      1,
      2,
      3
    ]
  }
}
```

YAML

```yaml
# A comment
anObject:
  aNumber: 42
  aString: This is a string
  aBoolean: true
  nothing: null
  arrayOfNumbers:
    - 1
    - 2
    - 3
```

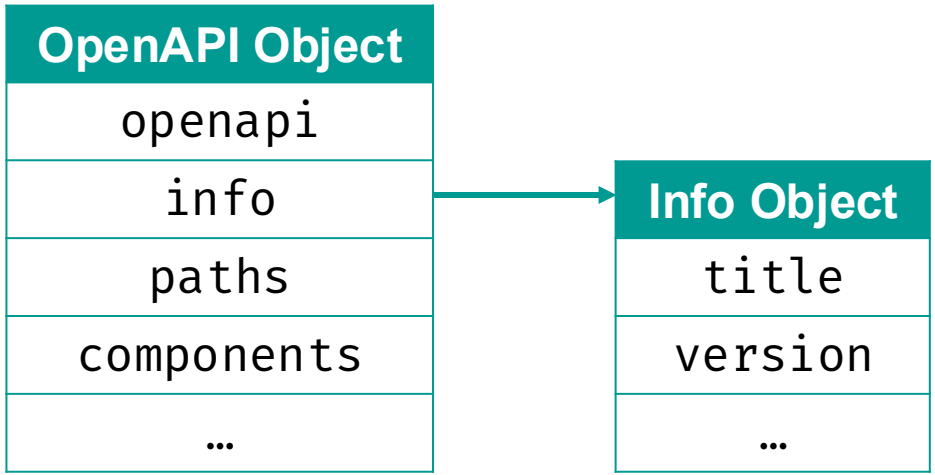
https://swagger.io/docs/specification/

Structure of OAD

OpenAPI Object	
`openapi`	- required
`info`	- required
`paths`	> one required
`components`	
…	

https://swagger.io/docs/specification/

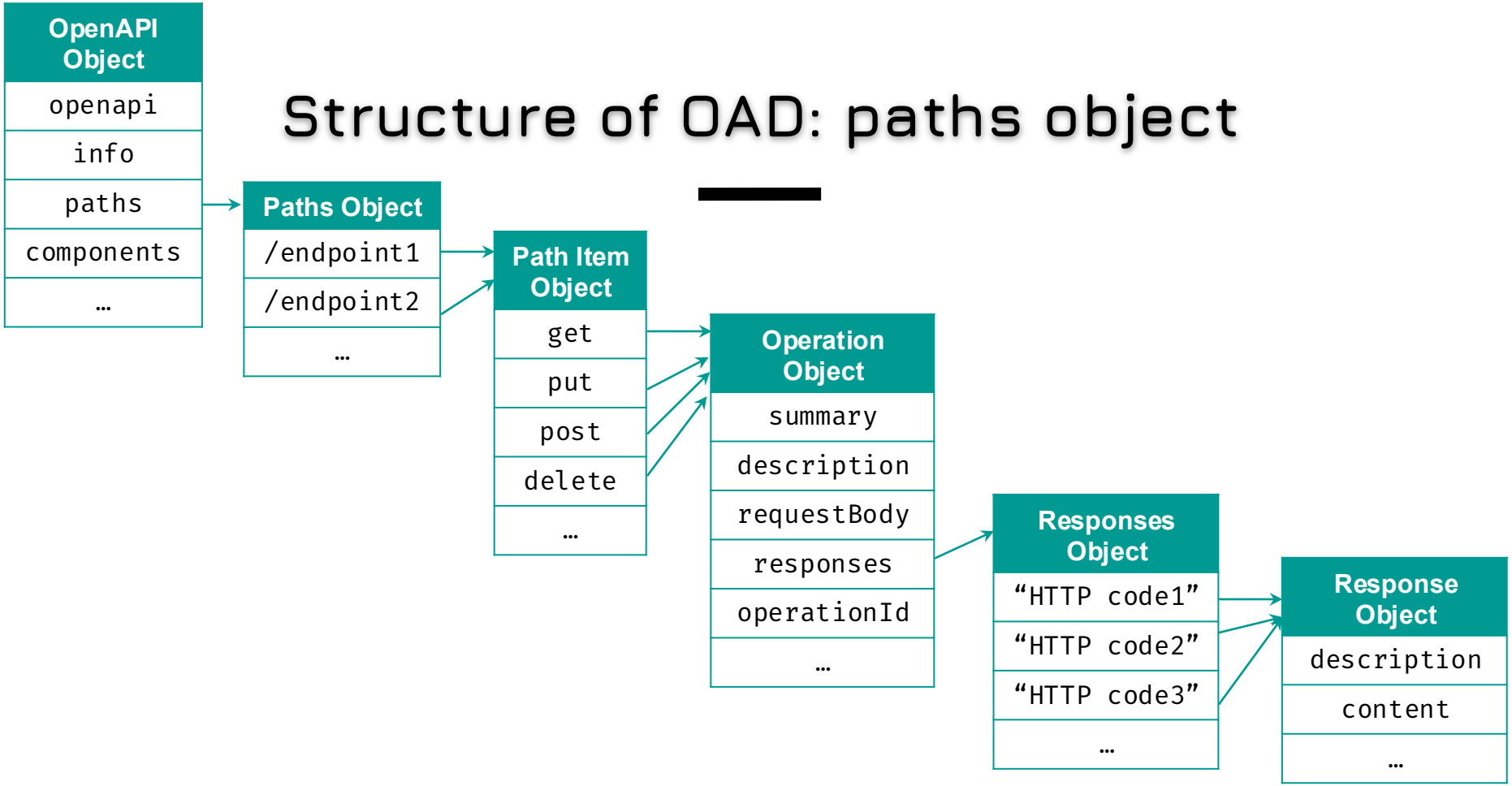
Structure of OAD: info object




```yaml
openapi: 3.1.0
info:
  title: A minimal OpenAPI Description
  version: 0.0.1
paths: {} # No endpoints defined
```

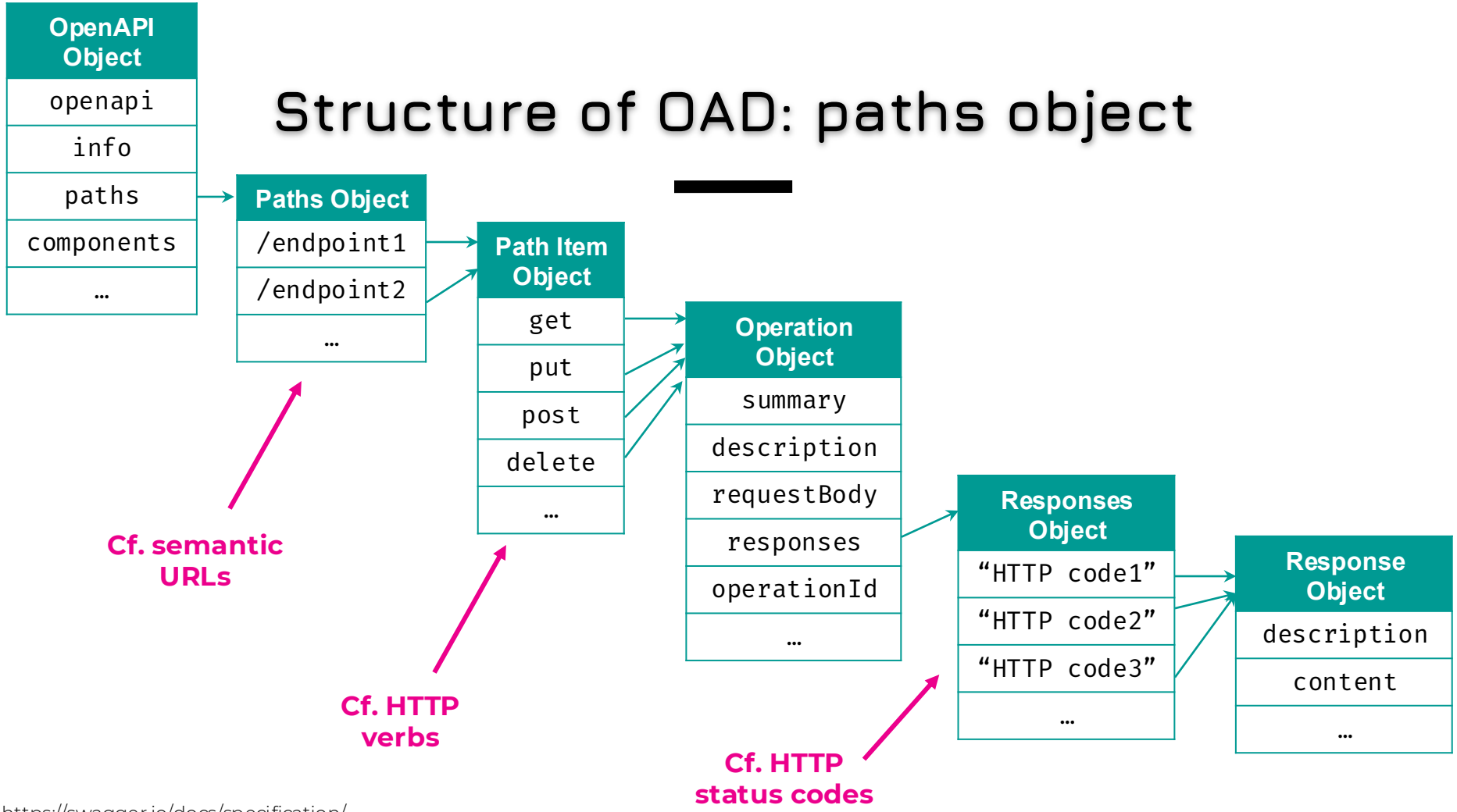
https://swagger.io/docs/specification/

Structure of OAD: paths object



https://swagger.io/docs/specification/

Structure of OAD: paths object



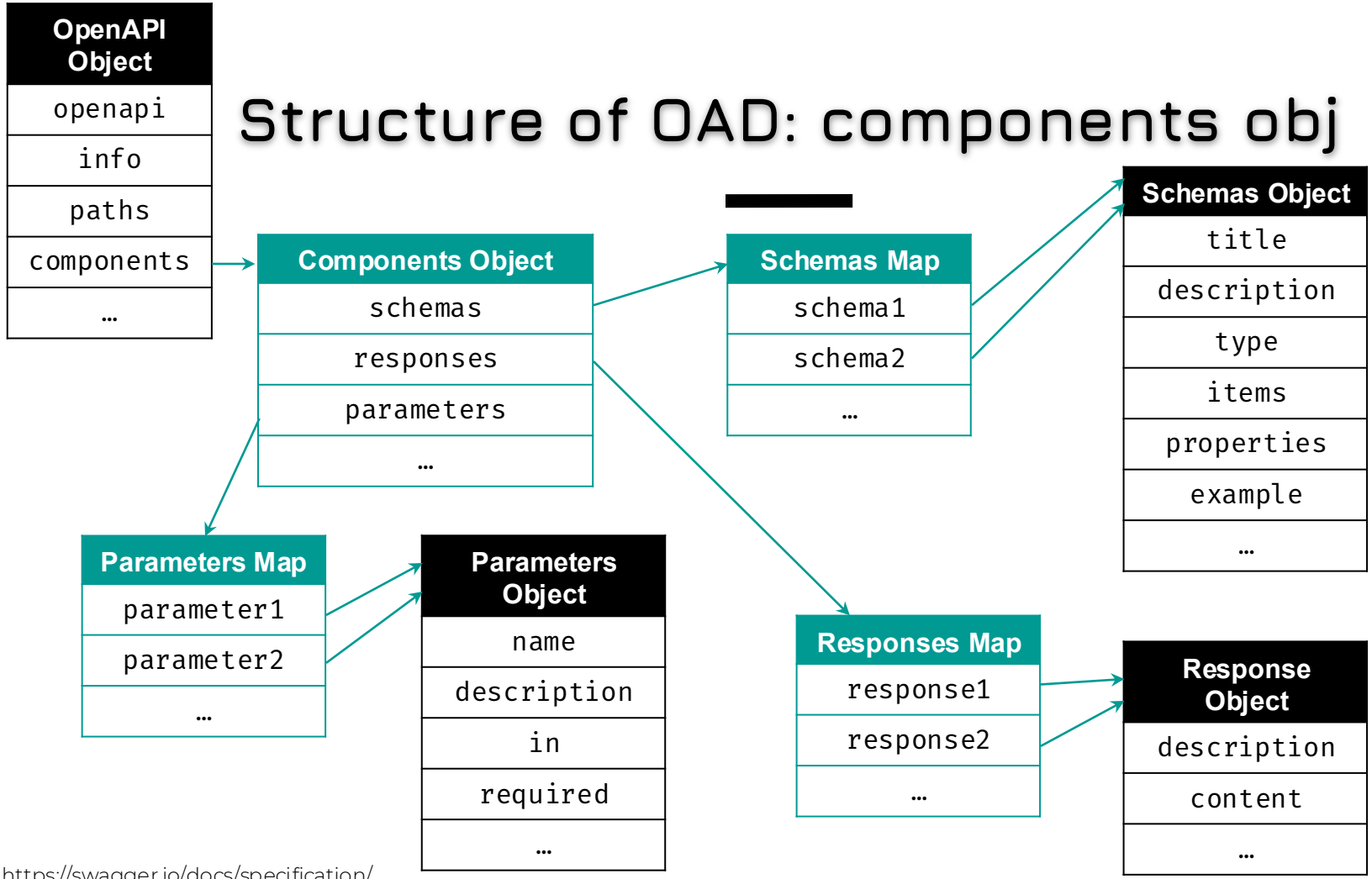
https://swagger.io/docs/specification/

Example (TicTacToe)

```yaml
openapi: 3.1.0
info:
  title: Tic Tac Toe
  description: |
    This API allows writing down marks on a Tic Tac Toe board and requesting the state of
    the board or of individual squares.
  version: 1.0.0
paths: # Whole board operations
  /board:
    get:
      summary: Get the whole board
      description: Retrieves the current state of the board and the winner.
      responses:
        "200":
          description: "OK"
          content: ...
```

https://swagger.io/docs/specification/

Structure of OAD: components obj



https://swagger.io/docs/specification/

Example components (TicTacToe)

```yaml
components:
  parameters:
    rowParam:
      description: Board row (vertical coordinate)
      name: row
      in: path
      required: true
      schema:
        $ref: '#/components/schemas/coordinate'
  schemas:
    coordinate:
      type: integer
    mark:
      type: string
      enum:
        - .
        - X
        - O
```

Cf. use the appropriate parameters

(Many fields deliberately left out for the sake of clarity, see complete example at https://learn.openapis.org/examples/v3.1/tictactoe.yaml)

Mapping to OpenAPI specs

course: Information Modelling

Entity types

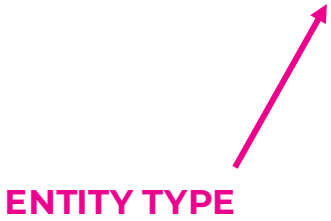
course: Information Modelling

Mapping an entity to a component definition

```yaml
components:
  schemas:
    Member:
      type: object
```

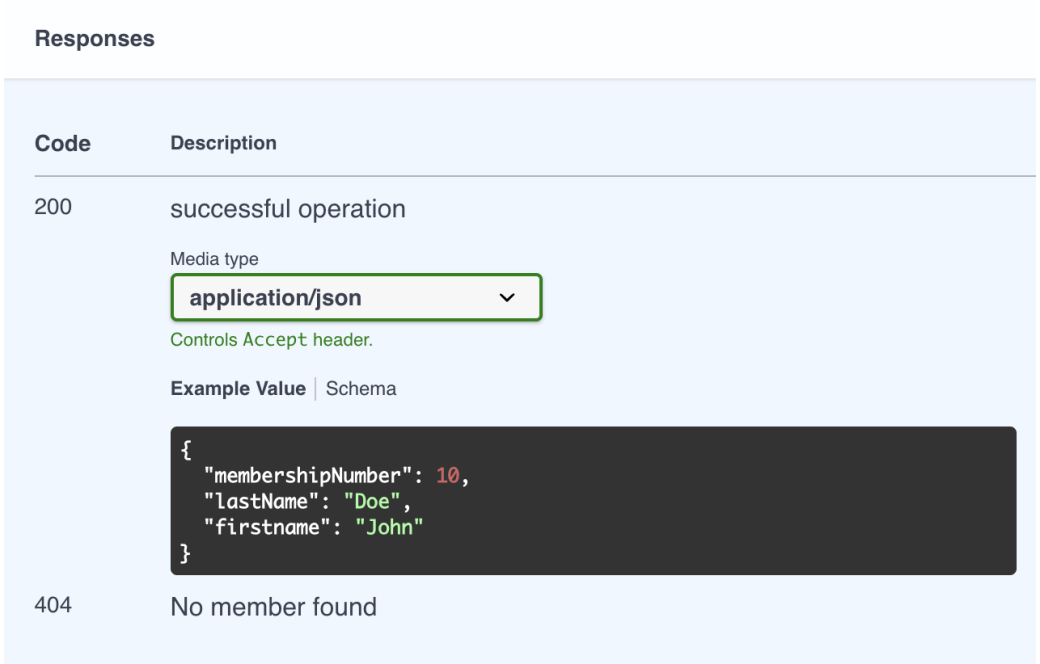
Mapping an entity to a component definition

```yaml
components:
  schemas:
    Member:
      type: object
```



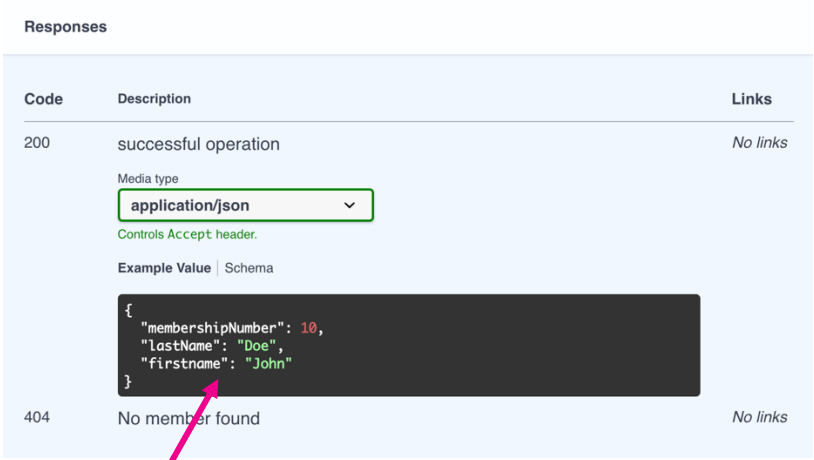
Creating entities

```yaml
/members/{membershipNumber}:
  get:
    summary: Find member by membershipnumber
    description: Returns a single member
    operationId: getMemberByMembershipNumber
    responses:
      '200':
        description: successful operation
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Member'
          application/xml:
            schema:
              $ref: '#/components/schemas/Member'
      '404':
        description: No member found
```



Creating entities

```yaml
/members/{membershipNumber}:
  get:
    summary: Find member by membershipnumber
    description: Returns a single member
    operationId: getMemberByMembershipNumber
    responses:
      '200':
        description: successful operation
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Member'
          application/xml:
            schema:
              $ref: '#/components/schemas/Member'
      '404':
        description: No member found
```

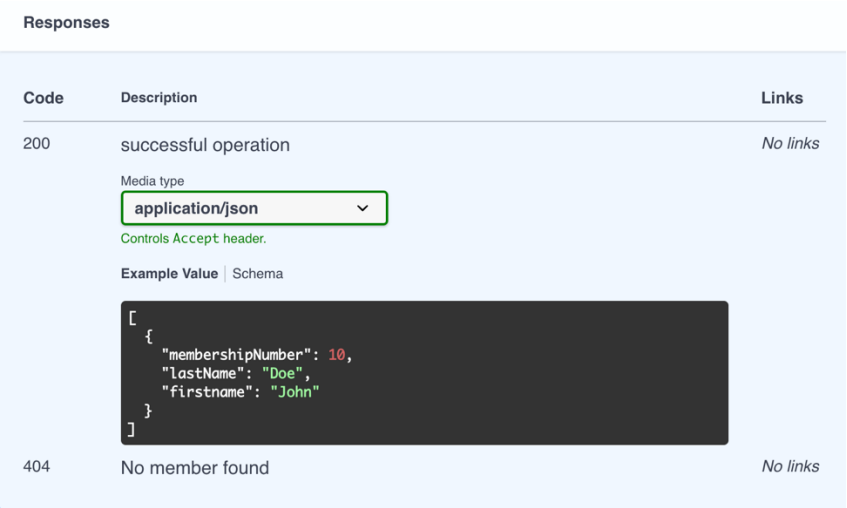


ENTITY

Creating entity sets

```yaml
/members:
  get:
    summary: Find all members
    description: Returns all members
    operationId: getAllMembers
    responses:
      '200':
        description: successful operation
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/MemberSet'
          application/xml:
            schema:
              $ref: '#/components/schemas/MemberSet'
      '404':
        description: No member found
```

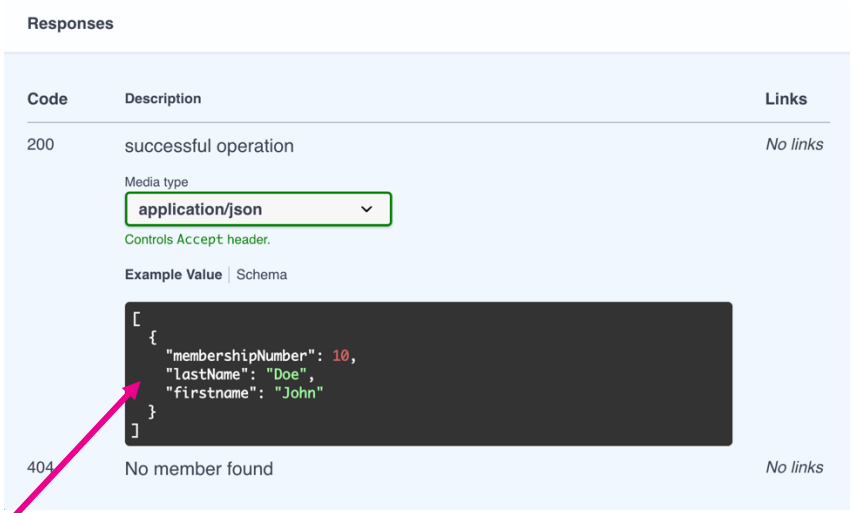
```yaml
components:
  schemas:
    Member:
      type: object
    MemberSet:
      type: array
      items:
        $ref: '#/components/schemas/Member'
```



Creating entity sets

```yaml
/members:
  get:
    summary: Find all members
    description: Returns all members
    operationId: getAllMembers
    responses:
      '200':
        description: successful operation
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/MemberSet'
          application/xml:
            schema:
              $ref: '#/components/schemas/MemberSet'
      '404':
        description: No member found
```

```yaml
components:
  schemas:
    Member:
      type: object
    MemberSet:
      type: array
      items:
        $ref: '#/components/schemas/Member'
```



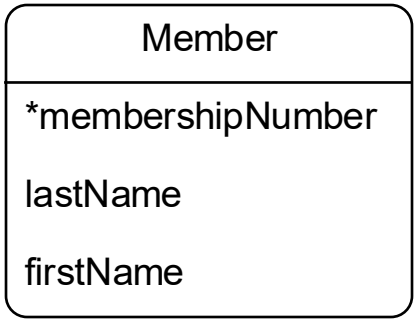
ENTITY SET

Attributes

course: Information Modelling

Attributes: basic pattern

```yaml
components:
  schemas:
    Member:
      type: object
      required:
        - membershipNumber
        - lastName
        - firstName
      properties:
        membershipNumber:
          type: integer
          format: int32
        lastName:
          type: string
        firstName:
          type: string
```



Supported types & formats: https://swagger.io/docs/specification/data-models/data-types/

Optional attributes

```yaml
components:
  schemas:
    Member:
      type: object
      required:
        - membershipNumber
        - lastName
        - firstName
      properties:
        membershipNumber:
          type: integer
          format: int32
        lastName:
          type: string
        firstName:
          type: string
        title:
          type: string
```

All object properties are **optional by default**.

Derived attributes

```yaml
components:
  schemas:
    Member:
      type: object
      required:
        - membershipNumber
        - lastName
        - firstName
        - dateOfBirth
      properties:
        membershipNumber:
          type: integer
          format: int32
        lastName:
          type: string
        firstName:
          type: string
        title:
          type: string
        dateOfBirth:
          type: string
          format: date
        age:
          type: integer
          format: int32
```

```yaml
components:
  schemas:
    Member:
      type: object
      required:
        - membershipNumber
        - lastName
        - firstName
        - dateOfBirth
      properties:
        membershipNumber:
          type: integer
          format: int32
        lastName:
          type: string
        firstName:
          type: string
        title:
          type: string
        dateOfBirth:
          type: string
          format: date
        age:
          type: integer
          format: int32
```

Derived attributes

We cannot define things that are calculated in an OpenAPI spec. Because an OpenAPI spec merely gives you an **overview of what you can expect to receive** from a certain piece of code or a certain endpoint.

Multivalued attributes

```yaml
components:
  schemas:
    Member:
      type: object
      required:
        - membershipNumber
        - lastName
        - firstName
        - dateOfBirth
      properties:
        membershipNumber:
          type: integer
          format: int32
        lastName:
          type: string
        firstName:
          type: string
        title:
          type: string
        dateOfBirth:
          type: string
          format: date
        age:
          type: integer
          format: int32
```

```yaml
phoneNumbers:
  type: array
  items:
    type: string
```

Composite attributes

There is **no out of the box solution**. Composite attributes consist of multiple properties = they are in effect an **object**.

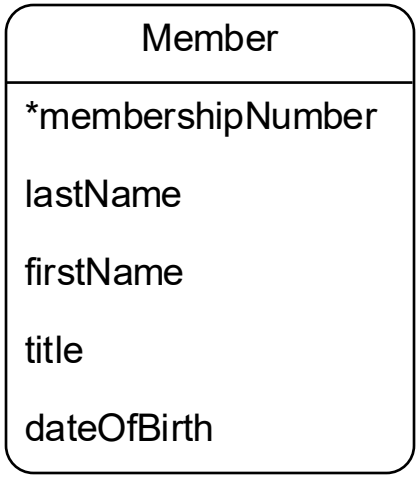
So if we wanted to do this, we would need new entities and relations, and be using **composition** instead! (cf. Later)

Identity

course: Information Modelling

Key attributes

MembershipNumber is unique.
There is no way to specify that in the **property.**




```yaml
components:
  schemas:
    Member:
      type: object
      required:
        - membershipNumber
        - lastName
        - firstName
        - dateOfBirth
      properties:
        membershipNumber:
          type: integer
          format: int32
        lastName:
          type: string
        firstName:
          type: string
        title:
          type: string
        dateOfBirth:
          type: string
          format: date
        age:
          type: integer
          format: int32
```

Key attributes in path

But, every **path** leads to a certain entity (set).

We can use the key attribute inside the path.

```yaml
/members/{membershipNumber}:
  get:
    summary: Find member by membershipnumber
    description: Returns a single member
    OperationId: getMemberByMembershipNumber
    parameters:
      - name: membershipNumber
        in: path
        description: Number of the Member you are searching for
        required: true
        schema:
          type: string
    responses:
      '200':
        description: successful operation
        content:
          application/json:
            Schema:
              $ref: '#/components/schemas/Member'
```

Weak entities

course: Information Modelling

Weak entities

- VGC:

Semantic URI's

/members/{membershipNo}

/members/{membershipNo}/subscriptions

/members/{membershipNo}/subscriptions/{startDate}

Semantic URI's

/members/{membershipNo}

/members/{membershipNo}/subscriptions

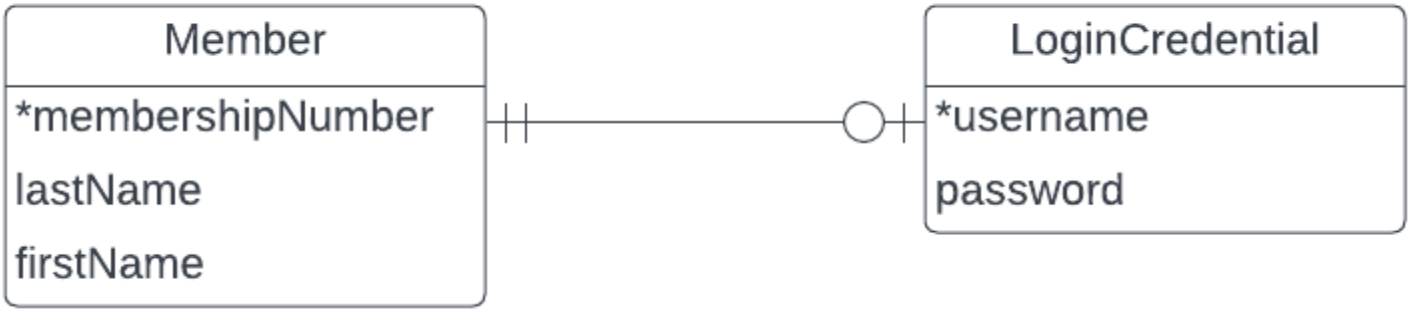
/members/{membershipNo}/subscriptions/{startDate}

~~/subscriptions/{startDate}/members/{membershipNo}~~

Relationships

course: Information Modelling

One-to-one



- All members are defined
- Some members may choose to use the self-service website and therefore create a username/password combo

One-to-one

```yaml
components:
  schemas:
    Member:
      type: object
      required:
        - membershipNumber
        - lastName
        - firstName
        - dateOfBirth
      properties:
        membershipNumber:
          type: integer
          format: int32
        lastName:
          type: string
        firstName:
          type: string
        title:
          type: string
        dateOfBirth:
          type: string
          format: date
        age:
          type: integer
          format: int32
        loginCredential:
          $ref: '#/components/schemas/LoginCredential'
```

```yaml
LoginCredential:
   type: object
   required:
     - username
     - password
   properties:
     username:
       type: string
     password:
       type: string
       format: password
     member:
       $ref: '#/components/schemas/Member'
```

Only to serve as an example,
don't actually make user data retrievable in your API...

One-to-one total participation

```yaml
components:
  schemas:
    Member:
      type: object
      required:
        - membershipNumber
        - lastName
        - firstName
        - dateOfBirth
      properties:
        membershipNumber:
          type: integer
          format: int32
        lastName:
          type: string
        firstName:
          type: string
        title:
          type: string
        dateOfBirth:
          type: string
          format: date
        age:
          type: integer
          format: int32
        loginCredential:
          $ref: '#/components/schemas/LoginCredential'
```

```yaml
LoginCredential:
    type: object
    required:
      - username
      - password
      - member
    properties:
      username:
        type: string
      password:
        type: string
        format: password
      member:
        $ref: '#/components/schemas/Member'
```

One-to-many

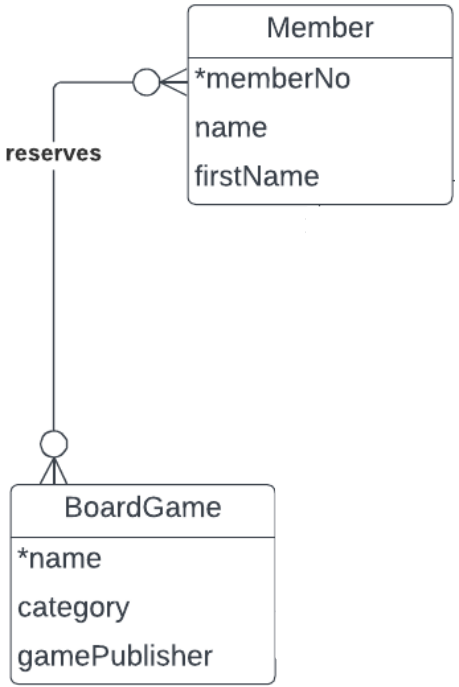
- A Member has one or more subscriptions
- Here, Subscription is a weak entity ➔ identified by memberNo and startDate

One-to-many

```yaml
components:
  schemas:
    Member:
      type: object
      required:
        - membershipNumber
        - lastName
        - firstName
        - dateOfBirth
        - subscriptions
      properties:
        membershipNumber:
          type: integer
          format: int32
        lastName:
          type: string
        firstName:
          type: string
        title:
          type: string
        dateOfBirth:
          type: string
          format: date
        age:
          type: integer
          format: int32
        subscriptions:
          type: array
            items:
              $ref: '#/components/schemas/subscription'
```

```yaml
Subscription:
   type: object
   required:
     - startDate
     - endDate
     - fee
     - member
   properties:
     startDate:
       type: string
       format: date
     endDate:
       type: string
       format: date
     fee:
       type: number
       format: double
     member:
       $ref: '#/components/schemas/member'
```

Many-to-many


Member
*memberNo
name
firstName
reserves
BoardGame
*name
category
gamePublisher

Many-to-many

```
BoardGame:
   type: object
   required:
     - name
     - category
     - gamePublisher
   properties:
     name:
       type: string
     category:
       type: string
     gamePublisher:
       type: string
BoardGameSet:
   type: array
   items:
     $ref: '#/components/schemas/BoardGame'
```

```
/boardgames/{name}/members:
  get:
    summary: Find members that reserved a certain game
    description: Returns all members that reserved a certain game
    OperationId: getMembersByBoardGame
    parameters:
      - name: name
        in: path
        description: name of the game you are looking at
        required: true
        schema:
          type: string
```

Many-to-many

```
BoardGame:
   type: object
   required:
     - name
     - category
     - gamePublisher
   properties:
     name:
       type: string
     category:
       type: string
     gamePublisher:
       type: string
BoardGameSet:
   type: array
   items:
     $ref: '#/components/schemas/BoardGame'
```

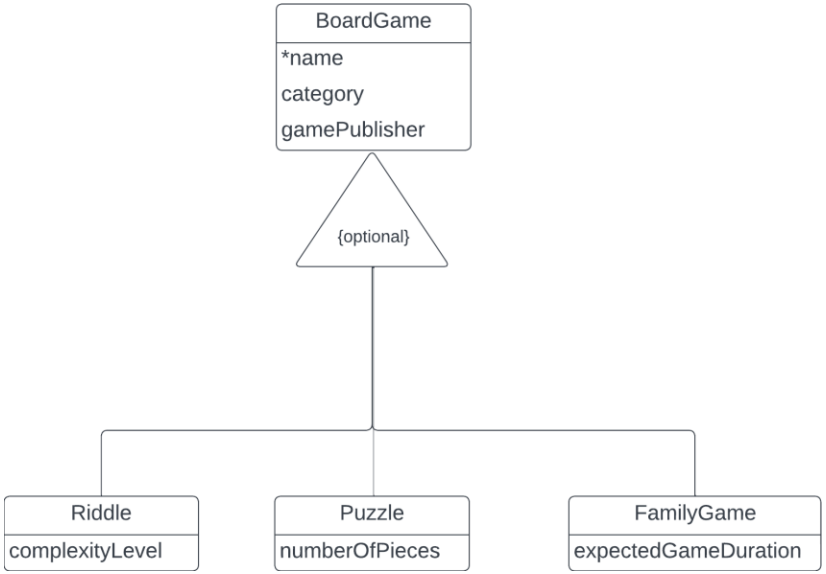
```
/boardgames/{name}/members:
  get:
    summary: Find members that reserved a certain game
    description: Returns all members that reserved a certain game
    OperationId: getMembersByBoardGame
    parameters:
      - name: name
        in: path
        description: name of the game you are looking at
        required: true
        schema:
          type: string
```

```
/members/{membershipnumber}/boardgames:
```

Inheritance

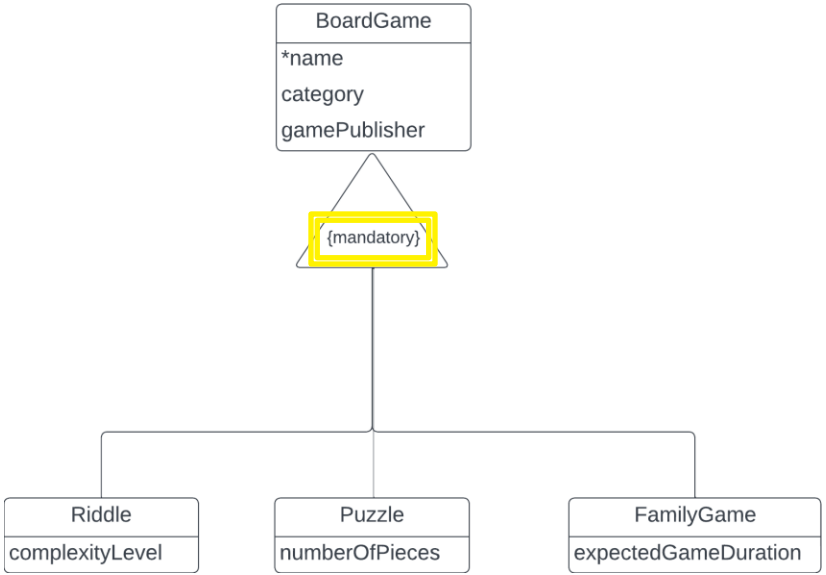
course: Information Modelling

Inheritance: attributes



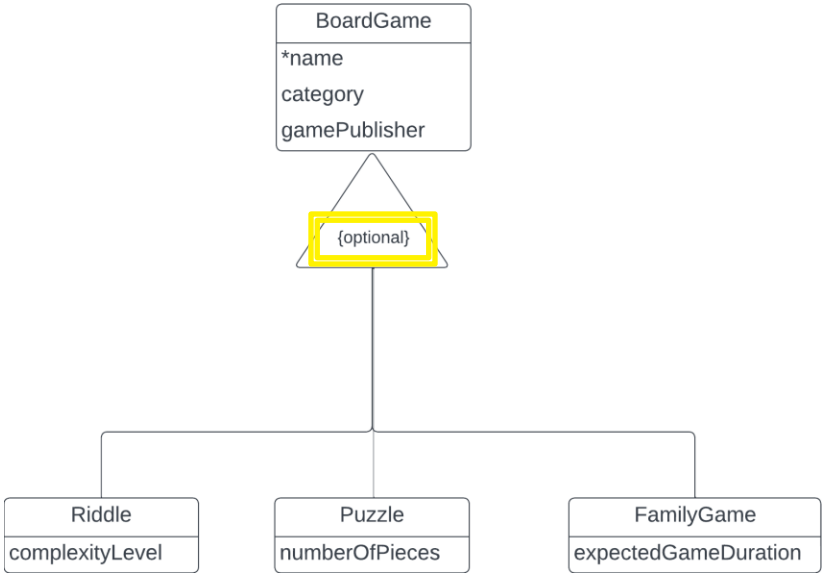

```yaml
components:
  schemas:
    Boardgame:
      type: object
      required:
        - name
        - category
        - gamePublisher
      properties:
        name:
          type: string
        category:
          type: string
        gamePublisher:
          type: string
    Riddle:
      allOf: # inherits all properties of BoardGame
      - $ref: '#/components/schemas/BoardGame'
      - type: object
        # all other properties specific to a Riddle
        properties:
          complexityLevel:
            type: integer
            format: int32
```

Inheritance: output




```yaml
/boardgames:
  post:
    summary: Add new boardgame
    operationId: createBoardGame
    tags:
      - boardgames
    requestBody:
      content:
        application/json:
          schema:
            oneOf:
              - $ref: '#/components/schemas/Riddle'
              - $ref: '#/components/schemas/Puzzle'
              - $ref: '#/components/schemas/FamilyGame'
    responses:
      '201':
        Description: added successfully
          content:
            application/json:
              schema:
                oneOf:
                  - $ref: '#/components/schemas/Riddle'
                  - $ref: '#/components/schemas/Puzzle'
                  - $ref: '#/components/schemas/FamilyGame'
```

Inheritance: output




```yaml
/boardgames:
  post:
    summary: Add new boardgame
    operationId: createBoardGame
    tags:
      - boardgames
    requestBody:
      content:
        application/json:
          schema:
            oneOf:
              - $ref: '#/components/schemas/BoardGame'
              - $ref: '#/components/schemas/Riddle'
              - $ref: '#/components/schemas/Puzzle'
              - $ref: '#/components/schemas/FamilyGame'
    responses:
      '201':
        Description: added successfully
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/BoardGame'
```