In order to start a conversation with one of our agents, I need you to read through and accept our terms and conditions.

What topic are you after today?

- Bank Accounts
- Loans
- Mortgages
- Others
Engage in the Eclipse and Java community this Summer at the Eclipse DemoCamp in Zurich. If you are interested in Open Source, Eclipse Projects, Java and more, this is the event to attend in Switzerland.

During the break and after the talks enjoy the networking, free beer, food and the opportunity to meet the available speakers and project leads.


Contents

1 Location
2 Date and Time
3 Registration
4 Agenda
5 Sponsors

Location

The demo camp will take place in the ETH lecture room HG D1.1 located in the main building of the Swiss Federal Institute of Technology at Rämistrasse 101, 8092 Zurich.

For details see
• Area map
• Floor map

Eclipse Foundation

- Vendor neutral player for open-source
- Intellectual Property Management
- Development Process
- Infrastructure
- Ecosystem Development
Eclipse Foundation

- Working groups: https://www.eclipse.org/org/workinggroups/explore.php
- Projects: https://projects.eclipse.org/
MicroProfile

- Open Tracing 1.3
- Open API 1.1
- Rest Client 1.3
- Config 1.3
- Fault Tolerance 2.0
- Metrics 2.0
- JWT Propagation 1.1
- Health Check 2.0
- CDI 2.0
- JSON-P 1.1
- JAX-RS 2.1
- JSON-B 1.0

MicroProfile 3.0

- New
- Updated
- No change from last release (MicroProfile 2.2)
All the same
OpenAPIs are everywhere
OpenAPIs are everywhere
OpenAPIs are everywhere
Specification GitHub Project

https://github.com/OAI/OpenAPI-Specification
An OpenAPI Specification

OpenAPI v3

- Info
- Servers
- Security
- Paths
- Tags
- ExternalDocs
- Components

JSON or YAML
openapi: 3.0.1
info:
  title: Todo Backend
  version: "1.0"
paths:
  /api/{id}:
    get:
      summary: Get the one todo
      operationId: todoGetOne
      parameters:
        - name: id
          in: path
          description: The id of the todo
          required: true
          schema:
            format: int64
            type: integer
            example: "42"
      responses:
        200:
          description: The requested Todo
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Todo'
openapi: 3.0.1
info:
  title: Todo Backend
  version: "1.0"
paths:
  /api/{id}:
    get:
      summary: Get the one todo
      operationId: todoGetOne
      parameters:
        - name: id
          in: path
          description: The id of the todo
          required: true
          schema:
            format: int64
            type: integer
            example: "42"
      responses:
        200:
          description: The requested Todo
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Todo'
openapi: 3.0.1
info:
  title: Todo Backend
  version: "1.0"
paths:
  /api/{id}:
    get:
      summary: Get the one todo
      operationId: todoGetOne
      parameters:
        - name: id
          in: path
          description: The id of the todo
          required: true
          schema:
            format: int64
            type: integer
            example: "42"
      responses:
        200:
          description: The requested Todo
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Todo'
openapi: 3.0.1
info:
  title: Todo Backend
  version: "1.0"
paths:
  /api/{id}:
    get:
      summary: Get the one todo
      operationId: todoGetOne
      parameters:
        - name: id
          in: path
          description: The id of the todo
          required: true
          schema:
            format: int64
            type: integer
            example: "42"
      responses:
        200:
          description: The requested Todo
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Todo'

request body

parameters (query, path...)

responses
openapi: 3.0.1
info:
  title: Todo Backend
  version: "1.0"
paths:
  /api/{id}:
    get:
      summary: Get the one todo
      operationId: todoGetOne
      parameters:
      - name: id
        in: path
        description: The id of the todo
        required: true
        schema:
          format: int64
          type: integer
          example: "42"
      responses:
        200:
          description: The requested Todo
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Todo'
openapi: 3.0.1
info:
  title: Todo Backend
  version: "1.0"
paths:
  /api/{id}:
    get:
      summary: Get the one todo
      operationId: todoGetOne
      parameters:
      - name: id
        in: path
        description: The id of the todo
        required: true
        schema:
          format: int64
          type: integer
          example: "42"
      responses:
        200:
          description: The requested Todo
          content:
            application/json:
              schema: $ref: '#/components/schemas/Todo'
components:
schemas:
  Todo:
  description: Object representing a Todo
  type: object
  properties:
    id:
      description: id of the entity
      format: int64
      type: integer
      example: "42"
    title:
      description: title of the todo
      type: string
      example: My task
    completed:
      description: whether the todo is completed or not
      type: boolean
      example: "false"
    url:
      description: url associated with the todo
      type: string
    order:
      format: int32
      description: order in the priority list
      type: integer
      example: "10"
components:
schemas:
  Todo:
    description: Object representing a Todo
type: object
    properties:
      id:
        description: id of the entity
        format: int64
        type: integer
        example: "42"
      title:
        description: title of the todo
        type: string
        example: My task
      completed:
        description: whether the todo is completed or not
        type: boolean
        example: "false"
      url:
        description: url associated with the todo
        type: string
      order:
        format: int32
        description: order in the priority list
        type: integer
        example: "10"
Swagger UI

**GET /api/{id}** Get one todo

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The id of the todo</td>
</tr>
</tbody>
</table>

#### id

- required
- integer
- (path)

### Responses

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>The requested Todo</td>
<td>No links</td>
</tr>
</tbody>
</table>

- application/json

**Example Value**

```json
{
  "id": 42,
  "title": "My task",
  "completed": false,
  "url": "string",
  "order": 10
}
```
Swagger Online Editor

https://editor.swagger.io/
Eclipse IDE plugin: KaiZen-OpenAPI-Editor

https://github.com/RepreZen/KaiZen-OpenAPI-Editor
APICURIO

https://www.apicur.io/
Code generator: OpenAPI-Generator

- **Open Source** (Apache 2.0 License)
- Hosted on **GitHub**: https://github.com/OpenAPI-Tools/openapi-generator
- Java code & mustache templates
- Fork of Swagger-Codegen
MicroProfile OpenAPI
MicroProfile OpenAPI

- specification documentation
- code: annotations & models & programming interfaces (implementation is required)
- to be used on top of JAX-RS
@PATCH
@Path("/{id}")
@Operation(
    operationId = "todoUpdate",
    summary = "Update an existing todo")
@APIResponses(
    value = @APIResponse(
        responseCode = "200",
        description = "The updated Todo",
        content = @Content(
            schema = @Schema(
                implementation = Todo.class))))

public Response update(@RequestBody(
    description = "The todo to update",
    content = @Content(
        schema = @Schema(
            implementation = Todo.class)))) Todo todo,
@PathParam("id") @Parameter(
    description = "The id of the todo",
    name = "id",
    example = "42",
    required = true,
    schema = @Schema(type = SchemaType.INTEGER, format = "int64")) Long id)
@PATCH
@Path("/{id}")
@Operation
  operationId = "todoUpdate",
  summary = "Update an existing todo"
@apiResponses
  value = @ApiResponse
    responseCode = "200",
    description = "The updated Todo",
    content = @Content
      schema = @Schema
        implementation = Todo.class
  )
public Response update(@RequestBody
  description = "The todo to update",
  content = @Content
    schema = @Schema
      implementation = Todo.class
) Todo todo,
@PathParam("id") @Parameter
  description = "The id of the todo",
  name = "id",
  example = "42",
  required = true,
  schema = @Schema(type = SchemaType.INTEGER, format = "int64") Long id)
@PATCH
@Path("/{id}")
@Operation(
    operationId = "todoUpdate",
    summary = "Update an existing todo"
)
@APIResponses(
    value = @APIResponse(
        responseCode = "200",
        description = "The updated Todo",
        content = @Content(
            schema = @Schema(
                implementation = Todo.class))
    ))

public Response update(@RequestBody(
    description = "The todo to update",
    content = @Content(
        schema = @Schema(
            implementation = Todo.class))) Todo todo,
@PathParam("id") @Parameter(
    description = "The id of the todo",
    name = "id",
    example = "42",
    required = true,
    schema = @Schema(type = SchemaType.INTEGER, format = "int64")) Long id)
Models

- Interfaces to represent an OpenAPI specification
- Builder pattern
- Typed, instead of looking at a JSON/YAML tree

package org.eclipse.microprofile.openapi
createOpenAPI()
  .paths(
    createPaths()
      .addPathItem("/api/{id}", createPathItem()
        .GET(
          createOperation()
            .operationId("todoGetOne")
            .summary("Get the one todo")
            .addParameter(createParameter()
              .name("id")
              .in(In.PATH)
              .description("The id of the todo")
              .required(true)
              .schema(createSchema()
                .type(SchemaType.INTEGER)
                .format("int64")
                .example(42))
            .responses(createAPIResponses()
              .addAPIResponse("200", createAPIResponse()
                .description("The requested Todo")
                .content(createContent()
                  .addMediaType("application/json", createMediaType()
                    .schema(createSchema()
                      .ref("#/components/schemas/Todo")))))
          )))
  )
Serving the OpenAPI Spec

- GET http://<host>:<port>/openapi

- Format (JSON or YAML) can be specified with the Accept header

- Document is generated based on:
  - Result returned by OASModelReader.buildModel()
  - Static OpenAPI file
  - Process annotations
  - Filter model via OASFilter
Serving the OpenAPI Spec

**Annotation Based**
- JAX-RS and MP-OpenAPI annotations are leading the document
- Code is verbose

**Static file**
- Spec is easier to write
- A mechanism to keep JAX-RS annotations in sync with the OpenAPI Specification is necessary
Wrapping other parsers

- Some java tools already exist to parse and manipulate OpenAPI specifications:
  - Swagger-Parser
  - KaiZen OpenAPI Parser

- When you write a tool, you would like program against an API to be able to exchange the underlying implementation.

- Model interfaces of MicroProfile OpenAPI can be this layer.

- See https://github.com/OpenAPITools/empoa
2018 demo

THORNTAIL
(a.k.a WildFly Swarm)
https://thorntail.io/

https://openliberty.io/
Demo: todo-backend
Code first approach

Server

Single page application

OpenAPI

Java client

Postman

Jakarta EE

Quarkus
DEMO
At the beginning…

Unblu 4.3 Web API

users

Service to manage all users in the system.

get

GET /api/v1/users/id=string

Read the user for the given id

Required Role

REGISTERED_USER

Required Call Origin

TRUSTED

HTTP Method

GET

Query Parameters

Name | Type | Description
--- | --- | ---
id | string | id of the user which should be returned

Return Type

The user of the id which should be returned. If it could not be found, null is returned.

User
API versioning

7  Unblu version
6
5
4
v1
v2

API version
API versioning

- **7** Unblu version
- **6**
- **5**
- **4**
- **v1**
- **v2**
- **v3** API version
Thank you!

@j2r2b

jmini

Code Examples: https://github.com/jmini/openapi-talk