【MicroProfile開発ガイド】
MicroProfile Open API

2018/12

日本アイ・ビー・エム システムズ・エンジニアリング株式会社

Disclaimer

- この資料は日本アイ・ビー・エム株式会社ならびに日本アイ・ビー・エム　システムズ・エンジニアリング株式会社の正式なレビューを受けておりません。
- 当資料は、資料内で説明されている製品の仕様を保証するものではありません。
- 資料の内容には正確を期するよう注意しておりますが、この資料の内容は2018年12月現在の情報であり、製品の新しいリリース、PTFなどによって動作、仕様が変わる可能性があるのでご注意下さい。
- 今後国内で提供されるリリース情報は、対応する発表レターなどでご確認ください。
- IBM、IBMロゴおよびibm.comは、世界の多くの国で登録されたInternational Business Machines Corporationの商標です。他の製品名およびサービス名等は、それぞれIBMまたは各社の商標である場合があります。現時点でのIBMの商標リストについては、www.ibm.com/legal/copytrade.shtmlをご覧ください。
- 当資料をコピー等で複製することは、日本アイ・ビー・エム株式会社ならびに日本アイ・ビー・エム　システムズ・エンジニアリング株式会社の承諾なしではできません。
- 当資料に記載された製品名または会社名はそれぞれの各社の商標または登録商標です。
- JavaおよびすべてのJava関連の商標およびロゴはOracleやその関連会社の米国およびその他の国における商標または登録商標です。
- Microsoft, WindowsおよびWindowsロゴは、Microsoft Corporationの米国およびその他の国における商標です。
- Linuxは、Linus Torvaldsの米国およびその他の国における登録商標です。
- UNIXはThe Open Groupの米国およびその他の国における登録商標です。

目次

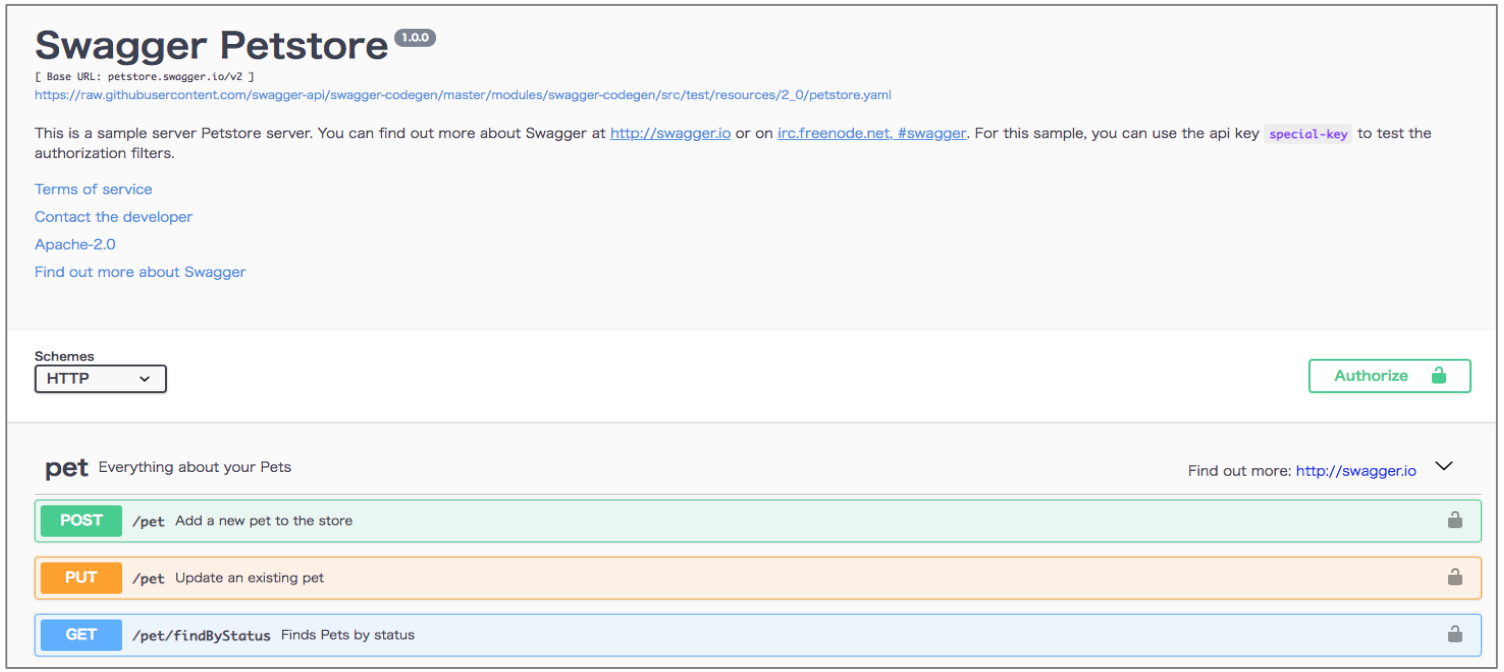
MicroProfile OpenAPI概要

OpenAPI Specification(OAS)とは

Open API Initiativeによって推進されている、API仕様を記述するためのフォーマット規格です。

- ■JSON/YAMLの2つのフォーマットで記述が可能
- ■OpenAPIドキュメントを作成することで、APIの仕様を統一されたフォーマットで記述が可能
- ■2017年7月V3.0.0をリリース
- ■OAS V3.0.0の内容は公開されているGitHubのリポジトリから参照可能

  OpenAPI-Specification(GitHub)
  https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md#documentStructure



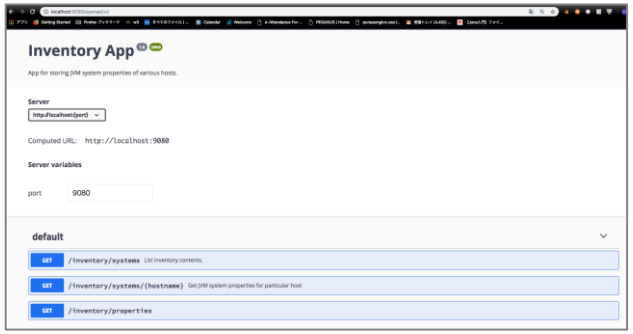

```yaml
title: Sample Pet Store App
description: This is a sample server for a pet store.
termsOfService: http://example.com/terms/
contact:
  name: API Support
  url: http://www.example.com/support
  email: support@example.com
license:
  name: Apache 2.0
  url: http://www.apache.org/licenses/LICENSE-2.0.html
version: 1.0.1
```

```yaml
/pets:
  get:
    description: Returns all pets from the system that the user has access to
    responses:
      '200':
        description: A list of pets.
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/pet'
```

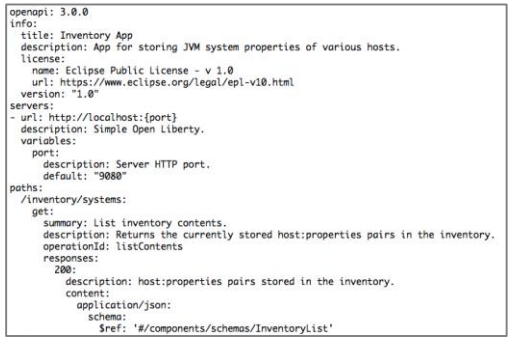
MicroProfile OpenAPIとは

MicroProfile OpenAPIは JAX-RSのエンドポイントに、アノテーションを付与することにより、OASv3.0.0(前ページ紹介)に準拠したyaml、json形式のAPI仕様を統一した実装方法で出力可能となります。

UI画面



yaml表示画面



```
openapi: 3.0.0
info:
  title: Inventory App
  description: App for storing JVM system properties of various hosts.
  license:
    name: Eclipse Public License - v 1.0
    url: https://www.eclipse.org/legal/epl-v10.html
  version: "1.0"
servers:
- url: http://localhost:{port}
  description: Simple Open Liberty.
  variables:
    port:
      description: Server HTTP port.
      default: "9080"
paths:
  /inventory/systems:
    get:
      summary: List inventory contents.
      description: Returns the currently stored host:properties pairs in the inventory.
      operationId: listContents
      responses:
        200:
          description: host:properties pairs stored in the inventory.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/InventoryList'
```

MicroProfile の機能

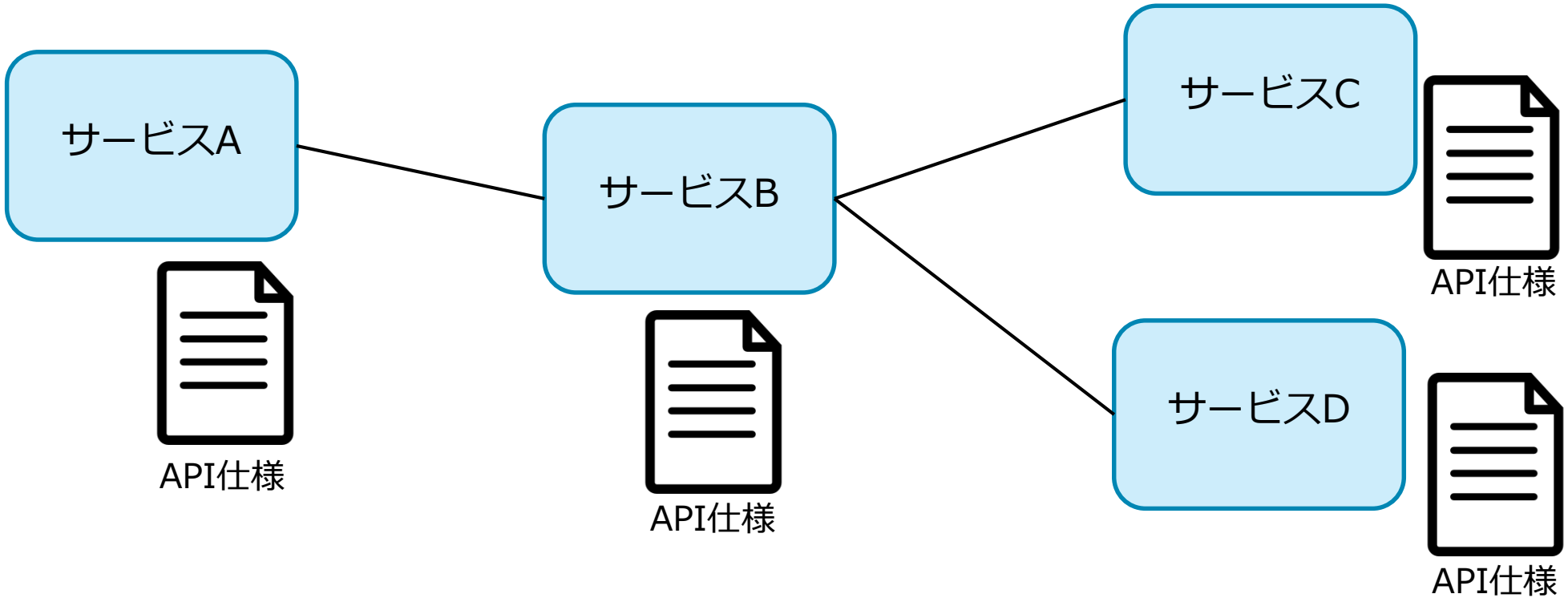
JAX-RS	CDI	JSON-P	JSON-B
REST APIおよびクライアント開発	依存性注入とライフサイクル管理	JSONデータの生成と解析	JSONデータとJavaオブジェクトのマッピング

Config	Fault Tolerance	Health Check	Health Metrics	JWT Propagation
ポータビリティを向上させる構成の外部化	障害に対処するための堅牢な動作	サービスの稼働確認の共通フォーマット	サービス状況をモニタリングするためのエンドポイント	インターオペラブルな認証とロールベースのアクセス制御

Open Tracing	Open API	Rest Client
複数サービスに跨る分散トレース	Open API仕様（Swagger）対応	タイプセーフなREST API呼び出し

利用シナリオ

複数のサービスを連携させて一つのアプリケーションを開発するマイクロサービス・アーキテクチャーにおいては、多数のサービスが公開するAPIの仕様を開発者間で共有することが必要となります。そのため、統一された方式でAPI仕様を公開する仕組みが求められますが、MicroProfile Open APIによりその標準化が可能となります。



フィーチャーの有効化

WAS Liberty上でMicroProfile OpenAPIの機能を有効化するためには、該当サーバーのserver.xmlにmpOpenAPI-1.0フィーチャーを設定します。

当該フィーチャーが含まれるMicroProfile-1.4、 MicroProfile-2.0を設定することも可能です。

```
<server description="new server">

    <!-- Enable features -->
    <featureManager>
         <feature>mpOpenAPI-1.0</feature>
    </featureManager>
</server>
```

mpOpenAPI-1.0フィーチャーを有効化することにより、以下のフィーチャーも暗黙的ロードにより自動的に有効化されます。

- jaxrs-2.0
- jaxrs-2.1
- jaxrsClient-2.0
- mpConfig-1.2
- servlet-3.0
- servlet-3.1
- servlet-3.1

＊MicroProfileとは異なるIBM独自のフィーチャーopenapi-3.0もありますが
API仕様を公開する仕組みをより標準化させるため、MicroProfile OpenAPIを利用することを推奨しています。
openapi-3.0の使用方法は以下Knowledge Centerのサイトを参照してください。

https://www.ibm.com/support/knowledgecenter/ja/SS7K4U_liberty/com.ibm.websphere.wlp.zseries.doc/ae/twlp_api_openapi.html

MicroProfile OpenAPIの使用方法

アノテーション

MicroProfile OpenAPIでは様々なアノテーションを提供します。

アノテーションを活用することで、RESTfulAPIの概要、詳細、パラメータ、完了コードなど様々なAPI仕様に必要な情報を提供することが可能となります。

```
@GET
@Path("/{hostname}")
@Produces(MediaType.APPLICATION_JSON)
@APIResponses(
    value = {
        @APIResponse(
            responseCode = "404",
            description = "Missing description",
            content = @Content(mediaType = "text/plain")),
        @APIResponse(
            responseCode = "200",
            description = "JVM system properties of a particular host.",
            content = @Content(mediaType = "application/json",
            schema = @Schema(implementation = Properties.class))) })
@Operation(
    summary = "Get JVM system properties for particular host",
    description = "Retrieves and returns the JVM system properties from the system "
    + "service running on the particular host.")
public Response getPropertiesForHost(
    @Parameter(
        description = "The host for whom to retrieve the JVM system properties for.",
        required = true,
        example = "foo",
        schema = @Schema(type = SchemaType.STRING))
    @PathParam("hostname") String hostname) {
    // Get properties for host
    Properties props = manager.get(hostname);
    if (props == null) {
        return Response.status(Response.Status.NOT_FOUND)
                       .entity("ERROR: Unknown hostname or the system service may "
                           + "not be running on " + hostname)
                       .build();
    }
```

Open APIで提供しているアノテーション/そのパラメータは次ページを参照してください。

アノテーション一覧

MicroProfile OpenAPIの主要アノテーションを以下表に記載しています。

アノテーション	概要
@APIResponses	複数のレスポンスに関する内容を定義
@APIResponse	レスポンスに関する内容を定義
@Operation	特定のパスのAPIについての内容を定義
@Parameter	パラメータに関する内容を定義
@Content	スキーマとメディアタイプを定義
@Schema	入出力のデータ型を定義

＊MicroProfile OpenAPIにはその他多くのアノテーションが存在します。
MicroProfile OpenAPI Specification(アノテーション一覧):
http://download.eclipse.org/microprofile/microprofile-open-api-1.0/microprofile-openapi-spec.html#_quick_overview_of_annotations
OpenLiberty MicroProfile OpenAPI (アノテーション詳細説明):
https://openliberty.io/docs/ref/microprofile/2.0/#class=org/eclipse/microprofile/openapi/annotations/Components.html&package=org/eclipse/microprofile/openapi/annotations/package-frame.html

APIResponseアノテーション

アノテーション	クラス
@APIResponses	org.eclipse.microprofile.openapi.annotations.responses.APIResponses
@APIResponse	org.eclipse.microprofile.openapi.annotations.responses.APIResponse

使用場所	パラメータ	型	説明
@APIResponses	value	APIResponse[]	@APIResponseの配列を定義
@APIResponse	content	Content[]	@Contentによりメディアタイプとスキーマを定義
	description	String	レスポンスに関する説明を定義
	headers	Header[]	レスポンスヘッダーの追加情報を配列で定義
	name	String	レスポンスの名前を定義
	ref	String	レスポンスオブジェクトクラスの定義
	responsecode	String	レスポンスコードを定義

Operationアノテーション

アノテーション	クラス
@Operation	org.eclipse.microprofile.openapi.annotations.Operation

パラメータ	**型**	**説明**
summary	String	API動作の概要を定義
description	String	API動作の詳細を定義
operationId	String	Operationを識別するための文字列IDを定義
deprecated	boolean	非推奨のAPIとする場合はtrueを定義
hidden	boolean	非表示にする場合はtrueを定義

Parameterアノテーション

アノテーション	クラス
@Parameter	org.eclipse.microprofile.openapi.annotations.parameters.Parameter

パラメータ	**型**	**説明**
name	String	パラメータの名前を定義
description	String	パラメータの詳細を定義
example	String	パラメータの例を定義
schema	Schema	パラメータの型を定義
required	boolean	パラメータを必須とする場合はtrueを定義
deprecated	boolean	非推奨のパラメータとする場合はtrueを定義
hidden	boolean	非表示にする場合はtrueを定義

Content/Schemaアノテーション

アノテーション	クラス
@Content	org.eclipse.microprofile.openapi.annotations.media.Content

パラメータ	型	説明
mediaType	String	そのオブジェクトが適用されるメディアタイプ
schema	Schema	リクエストボディに対する型を定義

アノテーション	クラス
@Schema	org.eclipse.microprofile.openapi.annotations.media.Schema

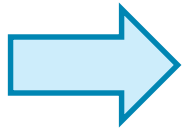
パラメータ	型	説明
implementation	String	スキーマを実装するjavaクラスを定義
type	SchemaType	スキーマの型を定義
name	String	スキーマの名前を定義
description	String	スキーマの説明内容を定義

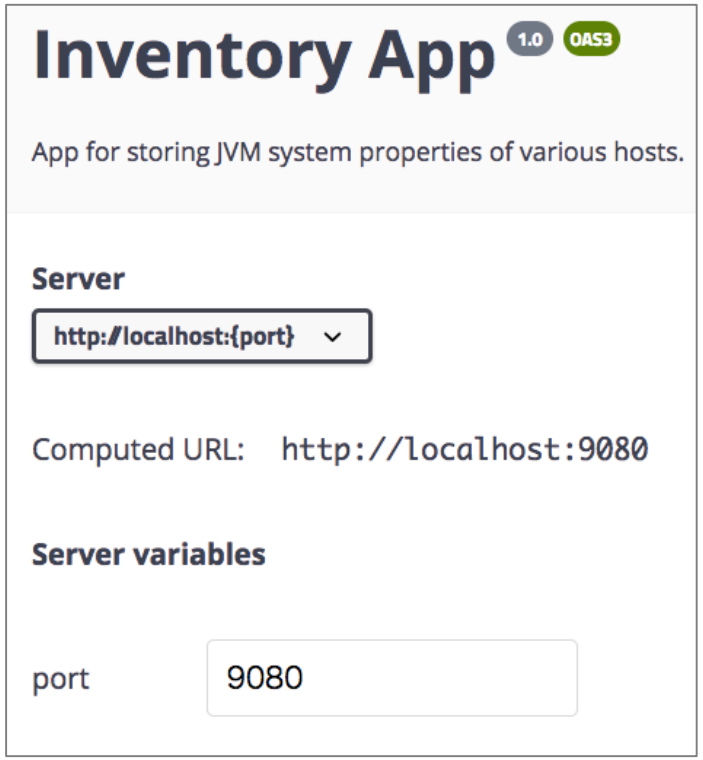
その他のAPI仕様記述方法

MicroProfile OpenAPI を活用してAPI仕様を記述するにはアノテーションだけでなく、以下2通りの機能を活用することも可能です。

- OASFilterインターフェースの実装(アノテーションよりこちらの実装が優先されます)
  - filterメソッドの中で各API仕様の要素に対応したメソッドを使用して値を設定するように実装することでAPI仕様を記述できます。

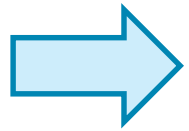
```
@Override
public void filterOpenAPI(OpenAPI openAPI) {
  // tag::oasfactory[]
  openAPI.setInfo(
      OASFactory.createObject(Info.class).title("Inventory App").version("1.0")
                .description(
                    "App for storing JVM system properties of various hosts.")
                .license(
                    OASFactory.createObject(License.class)
                              .name("Eclipse Public License - v 1.0").url(
                                  "https://www.eclipse.org/legal/epl-v10.html")));
```





- Static Open Fileの活用(アノテーションによる実装が優先されます)
  - yamlまたはjson形式のファイルを作成して、API仕様を記述できます。

```
1 openapi: 3.0.0
2 info:
3   title: Inventory App
4   description: App for storing JVM system properties of various hosts.
5   license:
6     name: Eclipse Public License - v 1.0
7     url: https://www.eclipse.org/legal/epl-v10.html
8   version: 1.0
```



それぞれの活用方法は次ページ以降に記載しています。

その他のAPI仕様記述方法 ー OAS Filterの使用方法

OASFilterインターフェースのfilterメソッドを実装してフィルタークラスを作成・登録することにより、API仕様上の情報を補完／追加することが可能です。

–例)アプリケーションのタイトル、APIのURL、ポート番号、追加レスポンス詳細説明など

*filterメソッドの種類は下記URLを参照

https://openliberty.io/docs/ref/microprofile/2.0/#class=org/eclipse/microprofile/openapi/OASFilter.html&package=org/eclipse/microprofile/openapi/package-frame.html

–作成したフィルタークラスを使用するにはMETA-INFディレクトリー配下にmicroprofile-config.propertiesファイルを作成し、フィルタークラスの以下定義が必要となります。

```
mp.openapi.filter = <FilterClassName>
```

*パッケージ名含む

*具体的な実装方法例はサンプルコードを参照してください。

その他のAPI仕様記述方法 ー Static Open Fileの使用方法

コード上にAPI仕様を記述するだけではなく、事前に作成した下記条件に当てはまるOpenAPI文書を利用してAPI仕様を記述することが可能です。

- 「openapi.yml」、「openapi.yaml」、「openapi.json」のいずれかの名前である
- src/main/webapp/META-INF ディレクトリーに配置

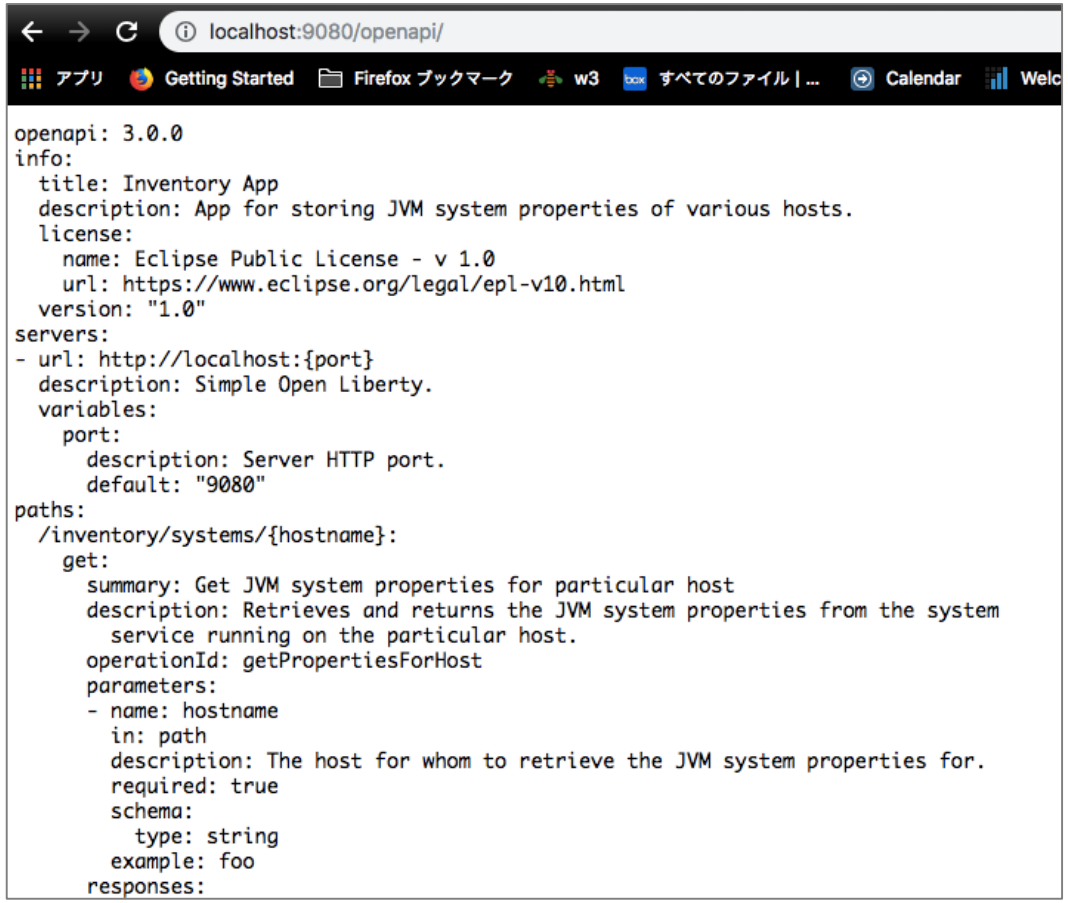
OpenAPIアノテーションとOpenAPI文書の優先順位

- OpenAPIアノテーションが優先される
  - アノテーションを付与していない部分の補完を行う
- OpenAPI文書がAPI仕様として完成している場合/アノテーションを無視したい場合
  - META-INFディレクトリー 配下のmicroprofile-config.propertiesファイルに「mp.openapi.scan.disable = true」を設定

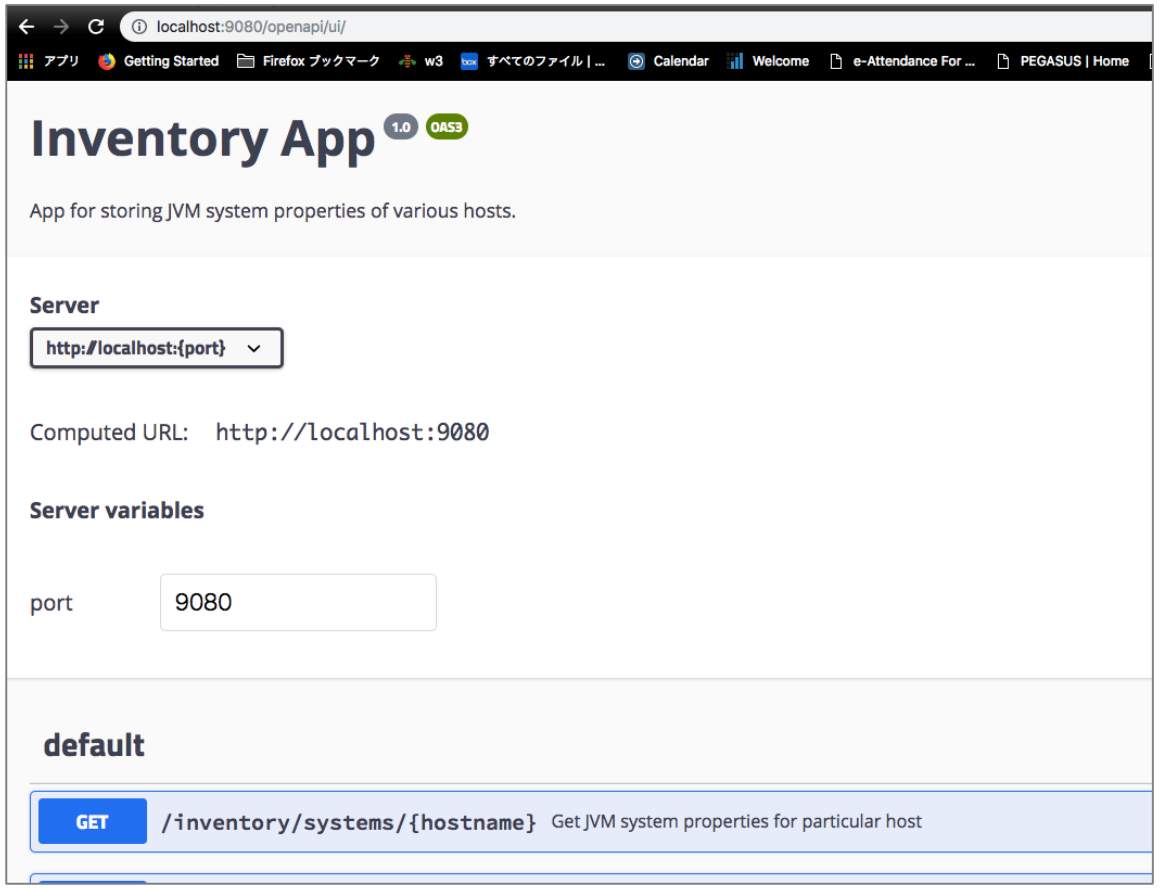
Open API仕様出力方法

下図２通りの形式でAPI仕様の出力が可能です。
*/openapiエンドポイントにおいてデフォルトではyaml形式での出力になります。
JSON形式で出力したい場合はクエリパラメータ「format=json」を付与してください。

http://<ドメイン名>:<ポート番号>/openapi



http://<ドメイン名>:<ポート番号>/openapi/ui



サンプルコード

本章ではOpen LibertyのWebサイトで紹介されているサンプルコードを活用し、Open APIの使用法を紹介します。

Open Liberty MicroProfile OpenAPI：https://github.com/openliberty/guide-microprofile-openapi

紹介するサンプルコードのクラスファイル/yamlファイル：

InventoryResource.class (package：io.openliberty.guides.inventory)
- /systems および /systems/{hostname} エンドポイントに対する処理ロジック

InventoryList.class (package：io.openliberty.guides.model)
- レスポンスパラメータクラス

src/main/webapp/META-INF/openapi.yaml
- API仕様を補完/実装するためのyaml or jsonファイル

InventoryOASFileter.class (package：io.openliberty.guides.filter)
- API仕様を補完/実装するためのOASFilterインターフェースの実装クラス

/systems パスのAPIに対するAPI仕様をアノテーション、Static Open File、OASFilterクラスを活用してOASv3.0.0に準拠した出力実装を行う例を紹介します。

サンプルコード：APIResponseアノテーション使用例

InventoryResource.class(/{hostname} エンドポイント実装部分)

```
@GET
@Path("/{hostname}")
@Produces(MediaType.APPLICATION_JSON)
@APIResponses(
    value = {
        @APIResponse(
            responseCode = "404",
            description = "Missing description",
            content = @Content(mediaType = "text/plain")),

        @APIResponse(
            responseCode = "200",
            description = "JVM system properties of a particular host.edit!",
            content = @Content(mediaType = "application/json",
            schema = @Schema(implementation = Properties.class)))
    })
```

- APIResponsesアノテーション

JAX-RSのリソース・メソッドに対して指定します。

valueパラメータ内に、レスポンスコード毎に@APIResponseを配列で定義します。

- APIResponseアノテーション

responseCodeパラメータに指定されたコード毎のレスポンスについての説明を追記できます。

・contentパラメータにでは@Contentを使用してmediaTypeを指定

・schemaパラメータでは@Schemaを使用してimplementationにBeanクラスを指定するとAPI仕様上にレスポンスデータの構造を表現することが可能

サンプルコード：Operation/Parameterアノテーション使用例

InventoryResource.class(/{hostname} エンドポイント実装部分)

```
@Operation(
    summary = "Get JVM system properties for particular host",
    description = "Retrieves and returns the JVM system properties from the system "
     + "service running on the particular host.",
    deprecated = false,
    hidden = false)
public Response getPropertiesForHost(
    @Parameter(
       description = "The host for whom to retrieve the JVM system properties for.",
       example = "foo",
       schema = @Schema(type = SchemaType.STRING))
    @PathParam("hostname") String hostname) {
    // Get properties for host
    Properties props = manager.get(hostname);
    if (props == null) {
        return Response.status(Response.Status.NOT_FOUND)
                       .entity("ERROR: Unknown hostname or the system service may "
                            + "not be running on " + hostname)
                       .build();
   }


    //Add to inventory to host
    manager.add(hostname, props);
    return Response.ok(props).build();
}
```

- Operationアノテーション

エンドポイントに対する説明を定義します。

deprecatedパラメータをtrueに定義すると非推奨APIとして表示することが可能、またhiddenパラメータをtrueにセットするとAPI文書に表示されなくなります。

- Parameterアノテーション

リクエストパラメータについて定義します。

exampleパラメータを定義すると、定義内容がパラメータの例として表示されます。また、schemaに関しては当該パラメータの型を定義します。

サンプルコード：Schemaアノテーション使用例

@Schemaアノテーションを付与したクラスをリソースクラス側のschemaパラメータに指定することで、レスポンスデータのスキーマとして出力することができます。自動で対象クラスのフィールド名やgetterメソッド名を基にAPIのレスポンスデータのサンプルをJSON形式で生成して出力することが可能です。

InventoryResource.class (/inventory/systems パス実装部分)

```
@GET
@Produces(MediaType.APPLICATION_JSON)
@APIResponse(
    responseCode = "200",
    description = "host:properties ・・・",
    content = @Content(
        mediaType = "application/json",
        schema = @Schema(
            type = SchemaType.OBJECT,
            implementation = InventoryList.class)))
```

SystemData.class

```
@Schema(name="SystemData", description="・・・")
public class SystemData {

    @Schema(required = true)
    private final String hostname;

    @Schema(required = true)
    private final Properties properties;

    public SystemData(String hostname,
                      Properties properties) {
        this.hostname = hostname;
        this.properties = properties;
    }

    public String getHostname() {
        return hostname;
    }

    public Properties getProperties() {
        return properties;
    }
}
```

InventoryList.class

```
@Schema(name="InventoryList", description="・・・")
public class InventoryList {

    @Schema(required = true)
    private List<SystemData> systems;

    public InventoryList(List<SystemData> systems) {
        this.systems = systems;
    }

    public List<SystemData> getSystems() {
        return systems;
    }

    public int getTotal() {
        return systems.size();
    }
}
```

出力イメージ

Example Value | Model

```
{
  "systems": [
    {
      "hostname": "string",
      "properties": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      }
    }
  ],
  "total": 0
}
```

サンプルコード：OAS Filterの使用例

OASFilterインターフェース実装クラス

```
public class InventoryOASFilter implements OASFilter {

  @Override
  public APIResponse filterAPIResponse(APIResponse apiResponse) {
    if ("Missing description".equals(apiResponse.getDescription())) {
      apiResponse.setDescription("Invalid hostname or the system service may not "
          + "be running on the particular host.");
    }
    return apiResponse;
  }

  @Override
  public void filterOpenAPI(OpenAPI openAPI) {
    openAPI.setInfo(
        OASFactory.createObject(Info.class).title("Inventory App").version("1.0")
                  .description(
                      "App for storing JVM system properties of various hosts.")
                  .license(
                      OASFactory.createObject(License.class)
                                .name("Eclipse Public License - v 1.0").url(
                                    "https://www.eclipse.org/legal/epl-v10.html")));

    openAPI.setServers(Arrays.asList(
        OASFactory.createObject(Server.class).url("http://localhost:{port}")
                  .description("Simple Open Liberty.").variables(
                      OASFactory.createObject(ServerVariables.class)
                                .addServerVariable("port",
                                    OASFactory.createObject(ServerVariable.class)
                                              .description(
                                                  "Server HTTP port.")
                                              .defaultValue("9080")))));
  }
}
```

- filterAPIResponse() メソッド

Open API文書上のresponses要素に対応したAPIResponseクラスが引数として渡されるので、それが保持する情報を補完または更新します。

- filterOpenAPI() メソッド

Open API文書上のopenapi要素に対応したOpenAPIクラスが引数として渡されるので、それが保持する情報を補完または更新します。

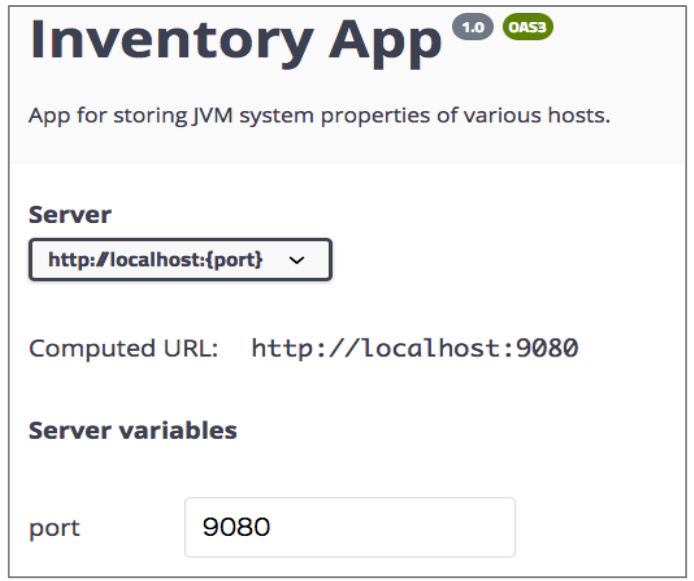
サンプルコード：Static Open File使用例

META-INF/openapi.yamlにより、各エンドポイントを束ねるアプリケーションとしての情報を
この例では記述しています。

src/main/webapp/META-INF/openapi.yaml

```yaml
openapi: 3.0.0
info:
  title: Inventory App
  description: App for storing JVM system properties of various hosts.
  license:
    name: Eclipse Public License - v 1.0
    url: https://www.eclipse.org/legal/epl-v10.html
  version: 1.0.0
servers:
- url: http://localhost:{port}
  description: Simple Open Liberty.
  variables:
    port:
      default: "9080"
      description: Server HTTP port.
```

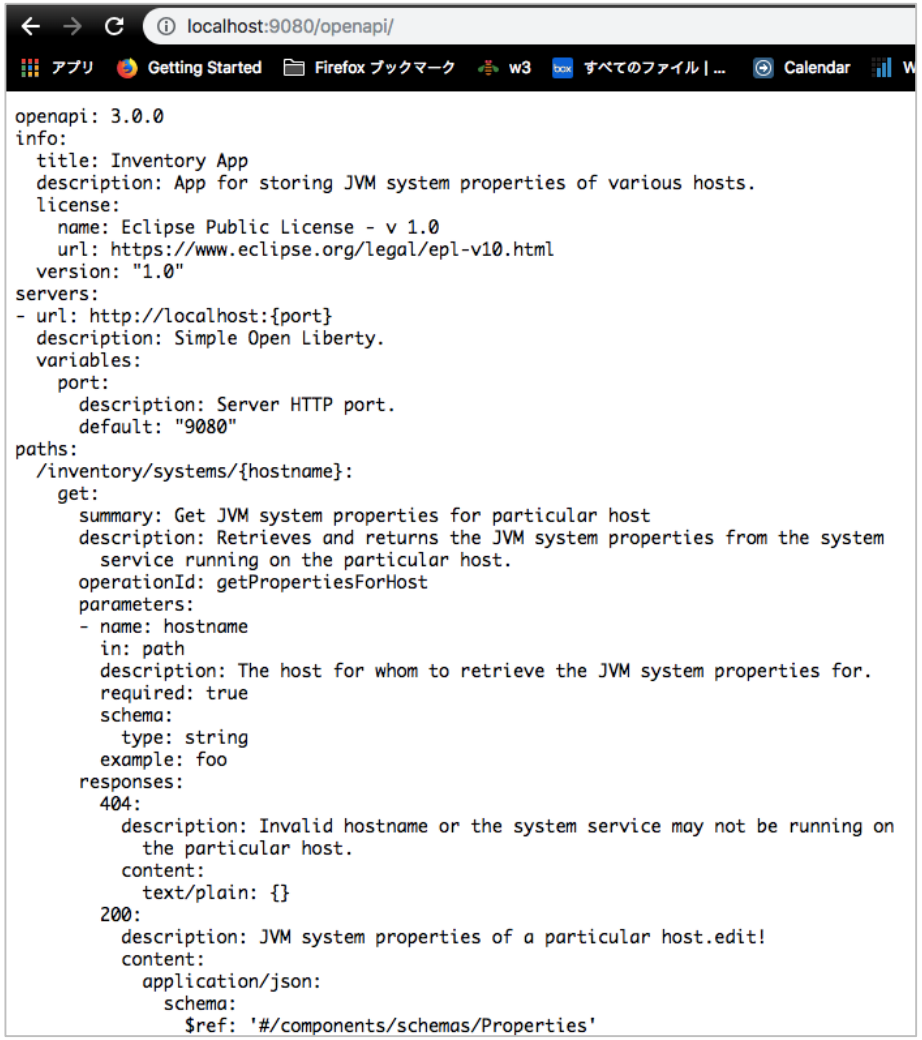
出力イメージ




```yaml
openapi: 3.0.0
info:
  title: Inventory App
  description: App for storing JVM system properties of various hosts.
  license:
    name: Eclipse Public License - v 1.0
    url: https://www.eclipse.org/legal/epl-v10.html
  version: "1.0"
servers:
- url: http://localhost:{port}
  description: Simple Open Liberty.
  variables:
    port:
      description: Server HTTP port.
      default: "9080"
```

サンプルコード：API仕様出力例

- サンプルコードで記述したAPI仕様の出力イメージ




```
openapi: 3.0.0
info:
  title: Inventory App
  description: App for storing JVM system properties of various hosts.
  license:
    name: Eclipse Public License - v 1.0
    url: https://www.eclipse.org/legal/epl-v10.html
  version: "1.0"
servers:
- url: http://localhost:{port}
  description: Simple Open Liberty.
  variables:
    port:
      description: Server HTTP port.
      default: "9080"
paths:
  /inventory/systems/{hostname}:
    get:
      summary: Get JVM system properties for particular host
      description: Retrieves and returns the JVM system properties from the system
        service running on the particular host.
      operationId: getPropertiesForHost
      parameters:
      - name: hostname
        in: path
        description: The host for whom to retrieve the JVM system properties for.
        required: true
        schema:
          type: string
        example: foo
      responses:
        404:
          description: Invalid hostname or the system service may not be running on
            the particular host.
          content:
            text/plain: {}
        200:
          description: JVM system properties of a particular host.edit!
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Properties'
```

```
  /inventory/systems:
    get:
      summary: List inventory contents.
      description: Returns the currently stored host:properties pairs in the inventory.
      operationId: sample
      parameters:
      - name: abc
        in: query
        description: List inventory contents.
        schema:
          type: string
        example: sample
      responses:
        200:
          description: host:properties pairs stored in the inventory.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/InventoryList'
  /inventory/properties:
    get:
      operationId: getProperties
      responses:
        default:
          description: default response
          content:
            application/json:
              schema:
                type: object
                additionalProperties:
                  type: string
components:
  schemas:
    InventoryList:
      required:
      - systems
      type: object
      properties:
        systems:
          type: array
          items:
            $ref: '#/components/schemas/SystemData'
        total:
          type: integer
      description: POJO that represents the inventory contents.
```

参考URL

- OpenLiberty – Documenting RESTfulAPIs OpenAPI
  https://openliberty.io/guides/microprofile-openapi.html
- MicroProfile OpenAPI Specification
  http://download.eclipse.org/microprofile/microprofile-open-api-1.0/microprofile-openapi-spec.pdf

- OpenLiberty – Annotation一覧
  https://openliberty.io/docs/ref/microprofile/2.0/#class=org/eclipse/microprofile/openapi/annotations/Components.html&package=org/eclipse/microprofile/openapi/annotations/package-frame.html
- Github — MicroProfile OpenAPI サンプルコード
  https://github.com/openliberty/guide-microprofile-openapi
- IBM Knowledge Center – MicroProfile OpenAPI1.0を利用したREST API資料の生成
  https://www.ibm.com/support/knowledgecenter/ja/SSEQTP_liberty/com.ibm.websphere.wlp.doc/ae/twlp_mpopenapi.html

End of File