

Property Based API Testing

EDD Workshop Session

Patrick Pfeiffer Johannes Brüggmann

March 6, 2025

- 1 OpenAPI Basics
- 2 Property-Based-Testing Basics
- 3 Applying PBT to OpenAPI /-Generator

Section 1

OpenAPI Basics

OpenApi: About

- API description language on top of JSON schema
- intended for REST and JSON RPC APIs with JSON or XML payloads
- language independent API specification written in JSON or YML
- plenty of generators and tools for individual language bindings
- client and server stubs generation for models and api

OpenApi: Building Blocks

- General info block (title, license, server, tags, ...)
- Operations (HTTP-verb + Path, request and responses, media-types)
- Schemata (named and anonymous models, parameters)
- Security (OAuth 2.0, HTTP BasicAuth, ...)

Swagger Editor

File Edit Insert Generate Server Generate Client About

```

1 openapi: 3.0.3
2 info:
3   title: Swagger Petstore - OpenAPI 3.0
4   description: |
5     This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about
6     Swagger at https://swagger.io or on the third iteration of the pet store, at https://petstore.swagger.io.
7     You may want to review the pet endpoint to learn more about the code.
8     Third-party libraries can be used to generate the code from the OpenAPI definition.
9   termsOfService: https://petstore.swagger.io/terms
10  servers:
11    - url: https://petstore.swagger.io/v2
12      description: Swagger Petstore - OpenAPI 3.0
13  paths:
14    /pet:
15      get:
16        description: Get a pet by ID
17        parameters:
18          - name: petId
19            in: path
20            required: true
21            schema:
22              type: integer
23        responses:
24          200:
25            description: Pet found
26            schema:
27              $ref: '#/components/schemas/Pet'
28          404:
29            description: Pet not found
30      post:
31        description: Create a new pet
32        parameters:
33          - name: body
34            in: body
35            required: true
36            schema:
37              $ref: '#/components/schemas/Pet'
38        responses:
39          201:
40            description: Pet created
41            schema:
42              $ref: '#/components/schemas/Pet'
43          400:
44            description: Invalid input
45    /pet/{petId}:
46      get:
47        description: Get a pet by ID
48        parameters:
49          - name: petId
50            in: path
51            required: true
52            schema:
53              type: integer
54        responses:
55          200:
56            description: Pet found
57            schema:
58              $ref: '#/components/schemas/Pet'
59          404:
60            description: Pet not found
61      post:
62        description: Update a pet by ID
63        parameters:
64          - name: petId
65            in: path
66            required: true
67            schema:
68              type: integer
69          - name: body
70            in: body
71            required: true
72            schema:
73              $ref: '#/components/schemas/Pet'
74        responses:
75          200:
76            description: Pet updated
77            schema:
78              $ref: '#/components/schemas/Pet'
79          400:
80            description: Invalid input
81          404:
82            description: Pet not found
83      delete:
84        description: Delete a pet by ID
85        parameters:
86          - name: petId
87            in: path
88            required: true
89            schema:
90              type: integer
91        responses:
92          204:
93            description: Pet deleted
94    /pet/{petId}/uploadImage:
95      post:
96        description: Upload an image to a pet
97        parameters:
98          - name: petId
99            in: path
100           required: true
101           schema:
102             type: integer
103          - name: image
104            in: body
105            required: true
106            schema:
107              type: string
108              format: binary
109        responses:
110          200:
111            description: Image uploaded
112            schema:
113              type: string
114              format: binary
115          400:
116            description: Invalid input
117          404:
118            description: Pet not found
119    /store/inventory:
120      get:
121        description: Get pet inventory
122        responses:
123          200:
124            description: Pet inventory
125            schema:
126              type: object
127              additionalProperties:
128                type: integer
129    /store/order:
130      post:
131        description: Create an order
132        parameters:
133          - name: body
134            in: body
135            required: true
136            schema:
137              $ref: '#/components/schemas/Order'
138        responses:
139          201:
140            description: Order created
141            schema:
142              $ref: '#/components/schemas/Order'
143          400:
144            description: Invalid input
145    /store/order/{orderId}:
146      get:
147        description: Get an order by ID
148        parameters:
149          - name: orderId
150            in: path
151            required: true
152            schema:
153              type: integer
154        responses:
155          200:
156            description: Order found
157            schema:
158              $ref: '#/components/schemas/Order'
159          404:
160            description: Order not found
161      delete:
162        description: Delete an order by ID
163        parameters:
164          - name: orderId
165            in: path
166            required: true
167            schema:
168              type: integer
169        responses:
170          204:
171            description: Order deleted
172    /store/order/{orderId}/complete:
173      post:
174        description: Complete an order by ID
175        parameters:
176          - name: orderId
177            in: path
178            required: true
179            schema:
180              type: integer
181        responses:
182          201:
183            description: Order completed
184            schema:
185              $ref: '#/components/schemas/Order'
186          400:
187            description: Invalid input
188          404:
189            description: Order not found
190    /user:
191      post:
192        description: Create a new user
193        parameters:
194          - name: body
195            in: body
196            required: true
197            schema:
198              $ref: '#/components/schemas/User'
199        responses:
200          201:
201            description: User created
202            schema:
203              $ref: '#/components/schemas/User'
204          400:
205            description: Invalid input
206    /user/{username}:
207      get:
208        description: Get a user by username
209        parameters:
210          - name: username
211            in: path
212            required: true
213            schema:
214              type: string
215        responses:
216          200:
217            description: User found
218            schema:
219              $ref: '#/components/schemas/User'
220          404:
221            description: User not found
222      post:
223        description: Update a user by username
224        parameters:
225          - name: username
226            in: path
227            required: true
228            schema:
229              type: string
230          - name: body
231            in: body
232            required: true
233            schema:
234              $ref: '#/components/schemas/User'
235        responses:
236          200:
237            description: User updated
238            schema:
239              $ref: '#/components/schemas/User'
240          400:
241            description: Invalid input
242          404:
243            description: User not found
244      delete:
245        description: Delete a user by username
246        parameters:
247          - name: username
248            in: path
249            required: true
250            schema:
251              type: string
252        responses:
253          204:
254            description: User deleted
255    /user/{username}/login:
256      post:
257        description: Login a user by username and password
258        parameters:
259          - name: username
260            in: path
261            required: true
262            schema:
263              type: string
264          - name: password
265            in: body
266            required: true
267            schema:
268              type: string
269        responses:
270          200:
271            description: User logged in
272            schema:
273              type: string
274          400:
275            description: Invalid input
276          404:
277            description: User not found
278    /user/{username}/logout:
279      post:
280        description: Logout a user by username
281        parameters:
282          - name: username
283            in: path
284            required: true
285            schema:
286              type: string
287        responses:
288          200:
289            description: User logged out
290            schema:
291              type: string
292          400:
293            description: Invalid input
294          404:
295            description: User not found
296  components:
297    schemas:
298      Pet:
299        type: object
300        properties:
301          id:
302            type: integer
303          name:
304            type: string
305          status:
306            type: string
307          tag:
308            type: string
309      Order:
310        type: object
311        properties:
312          id:
313            type: integer
314          petId:
315            type: integer
316          quantity:
317            type: integer
318          shipDate:
319            type: string
320          complete:
321            type: boolean
322      User:
323        type: object
324        properties:
325          id:
326            type: integer
327          username:
328            type: string
329          password:
330            type: string
331          email:
332            type: string
333          phone:
334            type: string
335          userStatus:
336            type: integer
337      Tag:
338        type: object
339        properties:
340          id:
341            type: integer
342          name:
343            type: string
344    responses:
345      default:
346        type: object
347        properties:
348          code:
349            type: integer
350          message:
351            type: string
352          errors:
353            type: array
354            items:
355              type: object
356              properties:
357                code:
358                  type: integer
359                message:
360                  type: string
361          headers:
362            type: object
363            properties:
364              X-Rate-Limit:
365                type: integer
366              X-Expires-After:
367                type: string
368              X-Total-Count:
369                type: integer
370    securitySchemes:
371      api_key:
372        type: apiKey
373        in: header
374        name: api-key
375      basic:
376        type: http
377        scheme: basic
378      bearer:
379        type: http
380        scheme: bearer
381  security:
382    - api_key: []
383    - basic: []
384    - bearer: []
385  tags:
386    - name: pet
387      description: Operations on your pets
388    - name: store
389      description: Operations on the store
390    - name: user
391      description: Operations on the user
392    - name: tag
393      description: Operations on the tag
394  externalDocs:
395    description: Find out more about Swagger
396    url: https://swagger.io
397  swagger: 3.0.3

```

Swagger Petstore - OpenAPI 3.0

This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about Swagger at <https://swagger.io>. In the third iteration of the pet store, we've switched to the design first approach. You can now help us improve the API whether it's by making changes to the definition itself or to the code. That way, we can improve the API in general, and expose some of the new features in OAS3.

If you're looking for the Swagger 2.0/GSW 2.0 version of Petstore, then click [here](#). Alternatively you can load in the [Edit > Load Petstore OAS 2.0](#) menu option!

Save your work:

- The [Pet Store repository](#)
- The [source API definition for the Pet Store](#)

Terms of service
Contact the developer
Apache 2.0
[Find out more about Swagger](#)

Swagger

[Swagger Petstore](#)

Authorize

pet Everything about your Pets [Find out more >](#)

Method	Path	Description
PUT	/pet	Update an existing pet
POST	/pet	Add a new pet to the store
GET		

Figure 1: Editieren von OpenApi-Dateien in Swagger Editor.

A1: Modelliere in OpenAPI

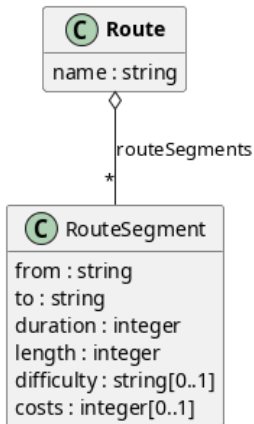


Figure 2: Klassendiagramm für das Modell "Route".

Section 2

Property-Based-Testing Basics

PBT: Basics

$$\exists P : S \rightarrow \{\top, \perp\} \implies P(s) = \top, \forall s \in S \quad (1)$$

Section 3

Applying PBT to OpenAPI /-Generator

Folie 3