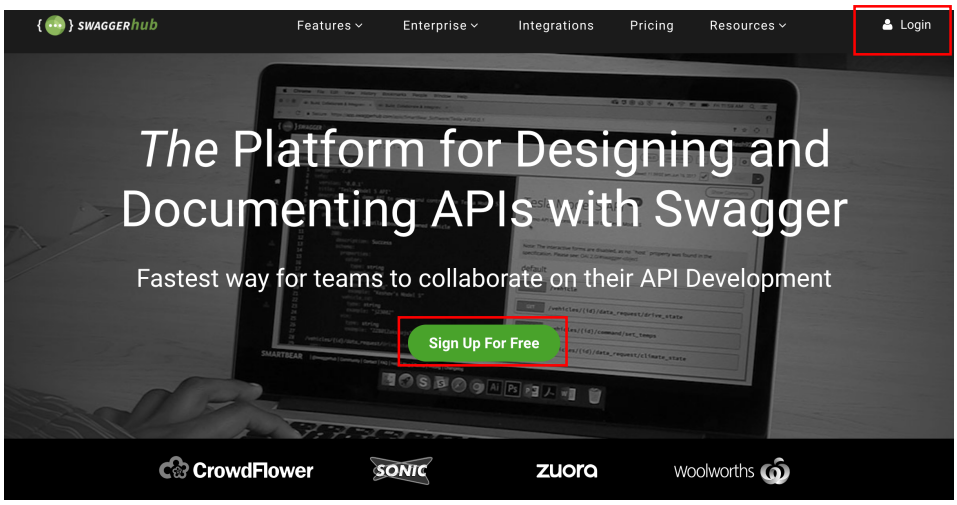
Bonus Lab: Comparing Swagger 2.0 and OpenAPI 3.0

Introduction

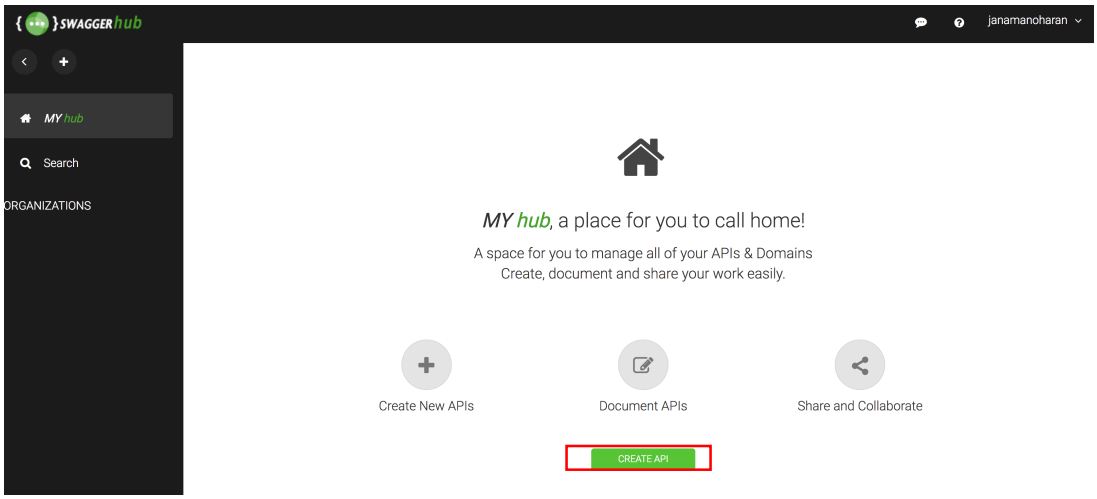
This lab is an optional lab, which will go through some of the major changes in OpenAPI 3.0 from Swagger 2.0. Some of these changes affect the annotations, models, the layout of the OpenAPI documents and the API Explorer user interface. This lab will also explore the ability to convert existing Swagger 2.0 documents to OpenAPI 3.0 documents, while preserving the content of the REST APIs.

5.1 Exploring SwaggerHub

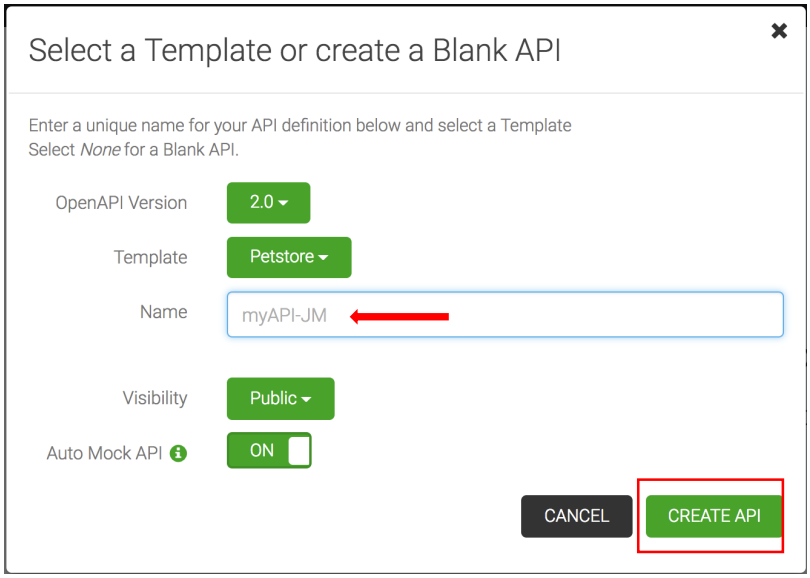
1. Create an account with SwaggerHub, if don't already have one
    a. Go to the SwaggerHub link: https://swaggerhub.com/
    b. Click **Sign Up For Free** to create an account or click **Login** to enter your existing credentials



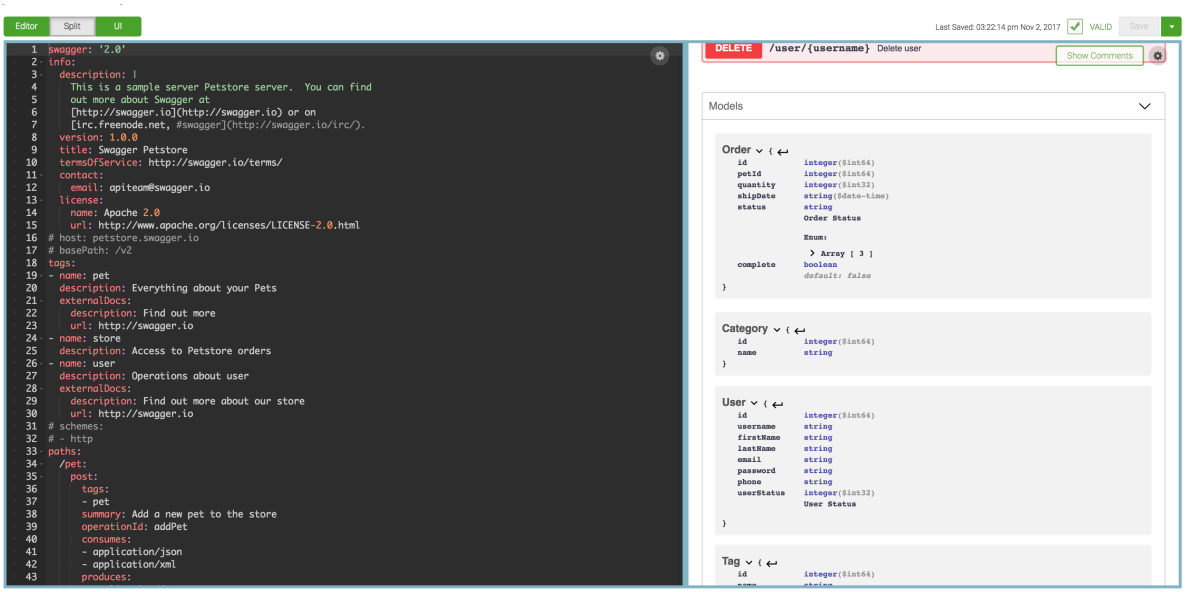
2. Create an API using Swagger 2.0 specifications and explore the REST API
    a. Once logged in, you will be presented with the SwaggerHub dashboard. Let's create an API using Swagger 2.0 specification by clicking the **CREATE API** button



b. In the name field, give a name of your choice for the API. As for the other values, we can leave the default values as is. We want to the Swagger 2.0 specification for now and we will be using the Petstore app as a template for the API. Finish by clicking **CREATE API**



c. SwaggerHub lets you simultaneously see both the Swagger document and the rendered UI, so that it is easier to edit

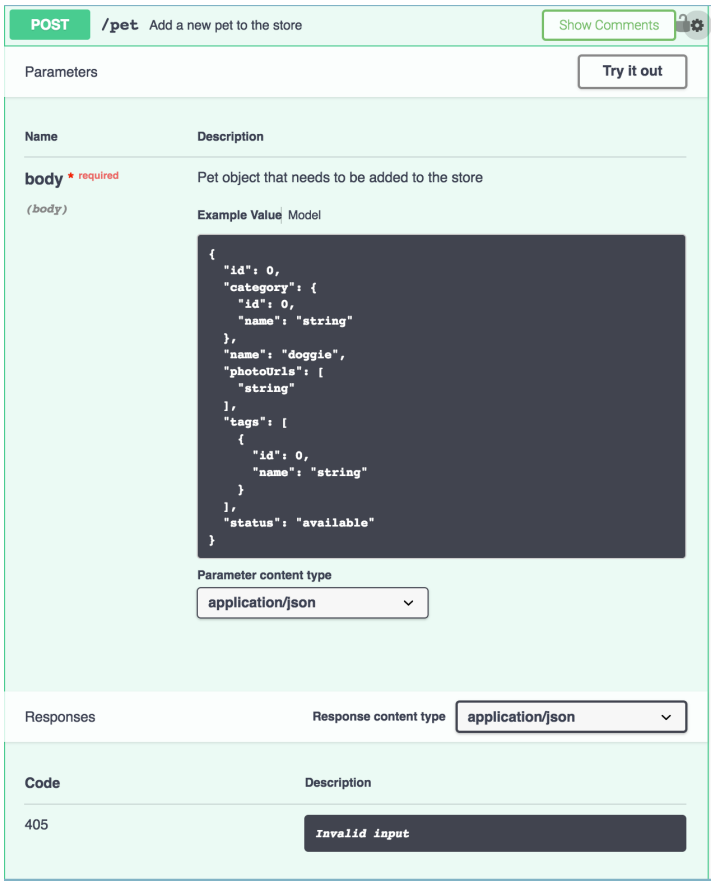


d. Let's take a look at an example; the first method in the API is a post method which will add a Pet to the petstore.

   i. In the Swagger document, all the information about the method is documented. Such information includes the name, description, parameters, possible response values and security schemes.

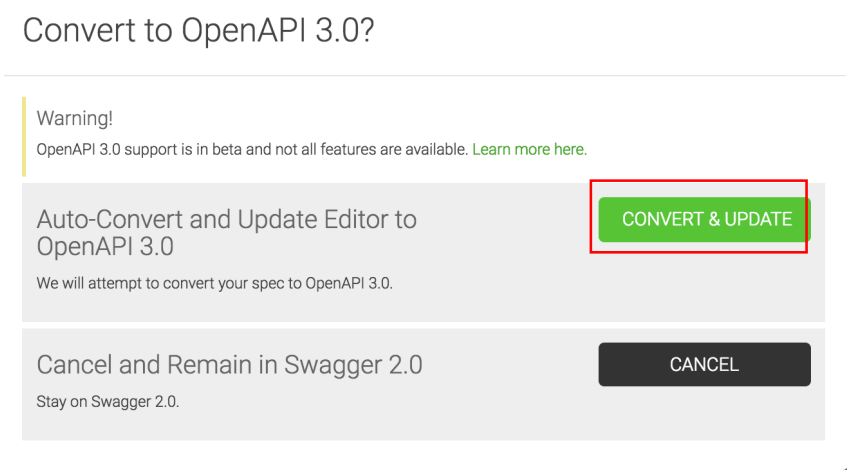
```yaml
paths:
  /pet:
    post:
      tags:
      - pet
      summary: Add a new pet to the store
      operationId: addPet
      consumes:
      - application/json
      - application/xml
      produces:
      - application/json
      - application/xml
      parameters:
      - in: body
        name: body
        description: Pet object that needs to be added to the store
        required: true
        schema:
          $ref: '#/definitions/Pet'
      responses:
        405:
          description: Invalid input
      security:
      - petstore_auth:
        - write:pets
        - read:pets
```

   ii. In the UI, all this information is nicely rendered.



3. Convert the Swagger 2.0 document to a OpenAPI 3.0 document
   a. In the top left corner, right above the UI view, click the dropdown menu to reveal the option to convert the document. Click **Convert to OpenAPI 3.0**

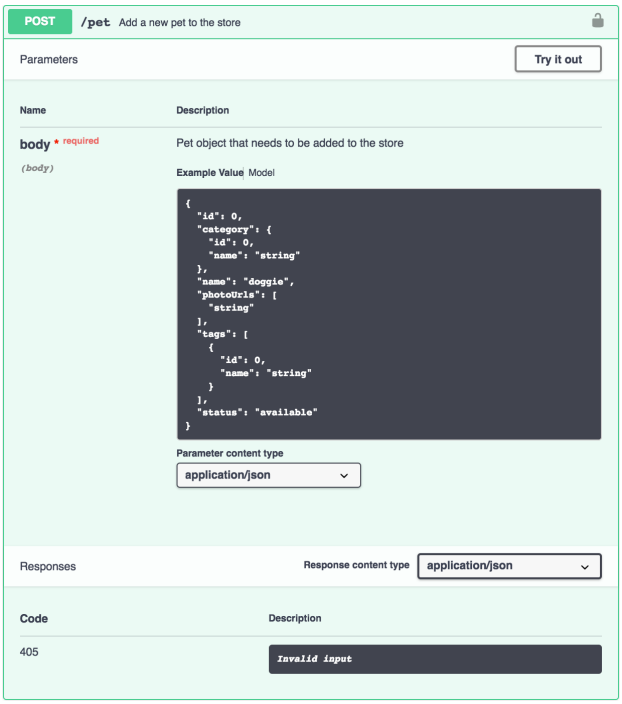
   b. Click **Convert & Update** to convert the document to OpenAPI 3.0. Even if the document is converted, you still view and edit the Swagger 2.0 version of the document.



   c. Let's revisit the same example from before, but now, in the OpenAPI 3.0 version of document.
      i. Here is the documentation for the post method; the documentation for the method is much shorter now than it was for the Swagger 2.0 version. In the next section of the lab, we will explore how this was possible.

```
paths:
  /pet:
    post:
      tags:
        - pet
      summary: Add a new pet to the store
      operationId: addPet
      responses:
        '405':
          description: Invalid input
      security:
        - petstore_auth:
            - 'write:pets'
            - 'read:pets'
      requestBody:
        $ref: '#/components/requestBodies/Pet'
```

ii. Here is the rendered UI for that method



In the next section of the lab, we will explore some of the changed that were made from Swagger 2.0 to OpenAPI 3.0.

5.2 Exploring Changes from Swagger 2.0 to OpenAPI 3.0

1. The root document object has changed from **swagger** to **openapi**. This definition allows us to specify which version the document will be in

```yaml
swagger: '2.0'
```

```yaml
openapi: 3.0.0
```

2. In both versions, objects models can be created and referenced later in other places of the document such as parameters, and responses

```yaml
definitions:
  Order:
    type: object
    properties:
      id:
        type: integer
        format: int64
      petId:
        type: integer
        format: int64
      quantity:
        type: integer
        format: int32
      shipDate:
        type: string
        format: date-time
      status:
        type: string
        description: Order Status
        enum:
        - placed
        - approved
        - delivered
      complete:
        type: boolean
        default: false
    xml:
      name: Order
```

In OpenAPI 3.0, the **definitions** field has been changed to **components**. Furthermore, the models can now be divided into which area of a method it applies. So, for example, you can define the models that are specific to a response, and others that are specific to a parameter

```yaml
components:
  schemas:
    Order:
      type: object
      properties:
        id:
          type: integer
          format: int64
        petId:
          type: integer
          format: int64
        quantity:
          type: integer
          format: int32
        shipDate:
          type: string
          format: date-time
        status:
          type: string
          description: Order Status
          enum:
            - placed
            - approved
            - delivered
        complete:
          type: boolean
          default: false
      xml:
        name: Order
    Category:
      type: object
      properties:
        id:
          type: integer
          format: int64
        name:
```

3. The content field has also been added in the OpenAPI 3.0 specification. Content will describe the structure of its parent field. The content can be used to describe elements such as Schemas, responses, parameters, examples and request bodies.
   Here is a sample response in which content is used to describe the different ways the response body can look

```yaml
responses:
  '200':
    description: successful operation
    content:
      application/json:
        schema:
          type: array
          items:
            $ref: '#/components/schemas/Pet'
      application/xml:
        schema:
          type: array
          items:
            $ref: '#/components/schemas/Pet'
```

4. In Swagger 2.0, body parameters can be documented using the parameter object. But in OpenAPI 3.0, body parameters are document using the Request Body object.

```yaml
  post:
    tags:
      - pet
    summary: Updates a pet in the store with form data
    operationId: updatePetWithForm
    parameters:
      - name: petId
        in: path
        description: ID of pet that needs to be updated
        required: true
        schema:
          type: integer
          format: int64
    responses:
      '405':
        description: Invalid input
    security:
      - petstore_auth:
          - 'write:pets'
          - 'read:pets'
    requestBody:
      content:
        application/x-www-form-urlencoded:
          schema:
            type: object
            properties:
              name:
                description: Updated name of the pet
                type: string
              status:
                description: Updated status of the pet
                type: string
```

5. These are some of the major changes from Swagger 2.0 to OpenAPI 3.0. Take some time to switch between the two versions of the document to see how certain fields have changed. To switch between the files, click the version in the top right corner, right above the Editor view

Congratulations! You have successfully completed the bonus lab!