

openapi: 3.0.3

info:

title: Swagger Petstore - OpenAPI 3.0

description: |-

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, with time, 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/OAS 2.0 version of Petstore, then click [here](https://editor.swagger.io/?url=https://petstore.swagger.io/v2/swagger.yaml). Alternatively, you can load via the `Edit > Load Petstore OAS 2.0` menu option!

Some useful links:

- [The Pet Store repository](https://github.com/swagger-api/swagger-petstore)

- [The source API definition for the Pet Store](https://github.com/swagger-api/swagger-petstore/blob/master/src/main/resources/openapi.yaml)

termsOfService: http://swagger.io/terms/

contact:

email: apiteam@swagger.io

license:

name: Apache 2.0

url: http://www.apache.org/licenses/LICENSE-2.0.html

version: 1.0.11

externalDocs:

description: Find out more about Swagger

url: http://swagger.io

servers:

- url: https://petstore3.swagger.io/api/v3

tags:

- name: pet

description: Everything about your Pets

externalDocs:

description: Find out more

url: http://swagger.io

- name: store

description: Access to Petstore orders

externalDocs:

description: Find out more about our store

url: http://swagger.io

- name: user

description: Operations about user

paths:

/api/pet:

PUT:

tags:

- pet

summary: Update an existing pet

description: Update an existing pet by Id

operationId: updatePet

requestBody:

description: Update an existent pet in the store

content:

application/json:

```
    schema:
      $ref: '#/components/schemas/Pet'
  application/xml:
    schema:
      $ref: '#/components/schemas/Pet'
  application/x-www-form-urlencoded:
    schema:
      $ref: '#/components/schemas/Pet'
  required: true
responses:
  200:
    description: Successful operation
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Pet'
      application/xml:
        schema:
          $ref: '#/components/schemas/Pet'
  400:
    description: Invalid ID supplied
  404:
    description: Pet not found
  422:
    description: Validation exception
security:
  - petstore_auth:
    - write:pets
    - read:pets
```

POST:

tags:

- pet

summary: Add a new pet to the store

description: Add a new pet to the store

operationId: addPet

requestBody:

description: Create a new pet in the store

content:

application/json:

schema:

\$ref: '#/components/schemas/Pet'

application/xml:

schema:

\$ref: '#/components/schemas/Pet'

application/x-www-form-urlencoded:

schema:

\$ref: '#/components/schemas/Pet'

required: true

responses:

200:

description: Successful operation

content:

application/json:

schema:

\$ref: '#/components/schemas/Pet'

application/xml:

schema:

\$ref: '#/components/schemas/Pet'

```
400:
  description: Invalid input
422:
  description: Validation exception
security:
  - petstore_auth:
      - write:pets
      - read:pets
/api/pet/findByStatus:
  GET:
    tags:
      - pet
    summary: Finds Pets by status
    description: Multiple status values can be provided with comma
separated strings
    operationId: findPetsByStatus
    parameters:
      - name: status
        in: query
        description: Status values that need to be considered for filter
        required: false
        explode: true
        schema:
          type: string
          default: available
          enum:
            - available
            - pending
            - sold
```

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'

400:

description: Invalid status value

security:

- petstore_auth:
 - write:pets
 - read:pets

/api/pet/findByTags:

GET:

tags:

- pet

summary: Finds Pets by tags

description: Multiple tags can be provided with comma separated strings. Use tag1, tag2, tag3 for testing.

operationId: findPetsByTags

parameters:

```
- name: tags
  in: query
  description: Tags to filter by
  required: false
  explode: true
  schema:
    type: array
    items:
      type: string
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'
  400:
    description: Invalid tag value
security:
  - petstore_auth:
      - write:pets
      - read:pets
```

/api/pet/{petId}:

GET:

tags:

- pet

summary: Find pet by ID

description: Returns a single pet

operationId: getPetById

parameters:

- name: petId

in: path

description: ID of pet to return

required: true

schema:

type: integer

format: int64

responses:

200:

description: successful operation

content:

application/json:

schema:

\$ref: '#/components/schemas/Pet'

application/xml:

schema:

\$ref: '#/components/schemas/Pet'

400:

description: Invalid ID supplied

404:

description: Pet not found

security:

- api_key: []
- petstore_auth:
 - write:pets
 - read:pets

POST:

tags:

- pet

summary: Updates a pet in the store with form data

description: ''

operationId: updatePetWithForm

parameters:

- name: petId
 - in: path
 - description: ID of pet that needs to be updated
 - required: true
 - schema:
 - type: integer
 - format: int64
- name: name
 - in: query
 - description: Name of pet that needs to be updated
 - schema:
 - type: string
- name: status
 - in: query
 - description: Status of pet that needs to be updated
 - schema:
 - type: string

responses:

400:

description: Invalid input

security:

- petstore_auth:
 - write:pets
 - read:pets

DELETE:

tags:

- pet

summary: Deletes a pet

description: delete a pet

operationId: deletePet

parameters:

- name: api_key
 - in: header
 - description: ''
 - required: false
 - schema:
 - type: string
- name: petId
 - in: path
 - description: Pet id to delete
 - required: true
 - schema:
 - type: integer
 - format: int64

responses:

400:

```
        description: Invalid pet value
security:
  - petstore_auth:
      - write:pets
      - read:pets
/api/pet/{petId}/uploadImage:
  POST:
    tags:
      - pet
    summary: uploads an image
    description: ''
    operationId: uploadFile
    parameters:
      - name: petId
        in: path
        description: ID of pet to update
        required: true
        schema:
          type: integer
          format: int64
      - name: additionalMetadata
        in: query
        description: Additional Metadata
        required: false
        schema:
          type: string
    requestBody:
      content:
        application/octet-stream:
```

```
      schema:
        type: string
        format: binary
    responses:
      200:
        description: successful operation
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ApiResponse'
    security:
      - petstore_auth:
          - write:pets
          - read:pets
/api/store/inventory:
  GET:
    tags:
      - store
    summary: Returns pet inventories by status
    description: Returns a map of status codes to quantities
    operationId: getInventory
    responses:
      200:
        description: successful operation
        content:
          application/json:
            schema:
              type: object
              additionalProperties:
```

```
        type: integer
        format: int32
    security:
      - api_key: []
/api/store/order:
  POST:
    tags:
      - store
    summary: Place an order for a pet
    description: Place a new order in the store
    operationId: placeOrder
    requestBody:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Order'
        application/xml:
          schema:
            $ref: '#/components/schemas/Order'
        application/x-www-form-urlencoded:
          schema:
            $ref: '#/components/schemas/Order'
    responses:
      200:
        description: successful operation
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Order'
```

400:

description: Invalid input

422:

description: Validation exception

/api/store/order/{orderId}:

GET:

tags:

- store

summary: Find purchase order by ID

description: For valid response try integer IDs with value ≤ 5 or > 10 . Other values will generate exceptions.

operationId: getOrderById

parameters:

- name: orderId

in: path

description: ID of order that needs to be fetched

required: true

schema:

type: integer

format: int64

responses:

200:

description: successful operation

content:

application/json:

schema:

\$ref: '#/components/schemas/Order'

application/xml:

schema:

\$ref: '#/components/schemas/Order'

400:

description: Invalid ID supplied

404:

description: Order not found

DELETE:

tags:

- store

summary: Delete purchase order by ID

description: For valid response try integer IDs with value < 1000.
Anything above 1000 or nonintegers will generate API errors

operationId: deleteOrder

parameters:

- name: orderId

in: path

description: ID of the order that needs to be deleted

required: true

schema:

type: integer

format: int64

responses:

400:

description: Invalid ID supplied

404:

description: Order not found

/api/user:

POST:

tags:

- user

summary: Create user

description: This can only be done by the logged in user.

operationId: createUser

requestBody:

description: Created user object

content:

application/json:

schema:

\$ref: '#/components/schemas/User'

application/xml:

schema:

\$ref: '#/components/schemas/User'

application/x-www-form-urlencoded:

schema:

\$ref: '#/components/schemas/User'

responses:

default:

description: successful operation

content:

application/json:

schema:

\$ref: '#/components/schemas/User'

application/xml:

schema:

\$ref: '#/components/schemas/User'

/api/user/createWithList:

POST:

tags:

- user

summary: Creates list of users with given input array
description: Creates list of users with given input array
operationId: createUsersWithListInput

requestBody:

content:

application/json:

schema:

type: array

items:

\$ref: '#/components/schemas/User'

responses:

200:

description: Successful operation

content:

application/json:

schema:

\$ref: '#/components/schemas/User'

application/xml:

schema:

\$ref: '#/components/schemas/User'

default:

description: successful operation

/api/user/login:

GET:

tags:

- user

summary: Logs user into the system

description: ''

operationId: loginUser

parameters:

- name: username
 - in: query
 - description: The user name for login
 - required: false
 - schema:
 - type: string
- name: password
 - in: query
 - description: The password for login in clear text
 - required: false
 - schema:
 - type: string

responses:

200:

description: successful operation

headers:

X-Rate-Limit:

description: calls per hour allowed by the user

schema:

type: integer

format: int32

X-Expires-After:

description: date in UTC when token expires

schema:

type: string

format: date-time

content:

application/xml:

```

        schema:
          type: string
      application/json:
        schema:
          type: string
    400:
      description: Invalid username/password supplied
/api/user/logout:
  GET:
    tags:
      - user
    summary: Logs out current logged in user session
    description: ''
    operationId: logoutUser
    parameters: []
    responses:
      default:
        description: successful operation
/api/user/{username}:
  GET:
    tags:
      - user
    summary: Get user by user name
    description: ''
    operationId: getUserByName
    parameters:
      - name: username
        in: path

```

description: 'The name that needs to be fetched. Use user1 for testing. '

required: true

schema:

type: string

responses:

200:

description: successful operation

content:

application/json:

schema:

\$ref: '#/components/schemas/User'

application/xml:

schema:

\$ref: '#/components/schemas/User'

400:

description: Invalid username supplied

404:

description: User not found

PUT:

tags:

- user

summary: Update user

description: This can only be done by the logged in user.

operationId: updateUser

parameters:

- name: username

in: path

description: name that need to be deleted

```
    required: true
    schema:
      type: string
requestBody:
  description: Update an existent user in the store
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/User'
    application/xml:
      schema:
        $ref: '#/components/schemas/User'
    application/x-www-form-urlencoded:
      schema:
        $ref: '#/components/schemas/User'
  responses:
    default:
      description: successful operation
DELETE:
  tags:
    - user
  summary: Delete user
  description: This can only be done by the logged in user.
  operationId: deleteUser
  parameters:
    - name: username
      in: path
      description: The name that needs to be deleted
      required: true
```

```
      schema:
        type: string
    responses:
      400:
        description: Invalid username supplied
      404:
        description: User not found
components:
  schemas:
    Order:
      type: object
      properties:
        id:
          type: integer
          format: int64
          example: 10
        petId:
          type: integer
          format: int64
          example: 198772
        quantity:
          type: integer
          format: int32
          example: 7
        shipDate:
          type: string
          format: date-time
        status:
          type: string
```

```
    description: Order Status
    example: approved
    enum:
      - placed
      - approved
      - delivered
    complete:
      type: boolean
  xml:
    name: order
Customer:
  type: object
  properties:
    id:
      type: integer
      format: int64
      example: 100000
    username:
      type: string
      example: fehguy
    address:
      type: array
      xml:
        name: addresses
        wrapped: true
      items:
        $ref: '#/components/schemas/Address'
  xml:
    name: customer
```

Address:

type: object

properties:

street:

type: string

example: 437 Lytton

city:

type: string

example: Palo Alto

state:

type: string

example: CA

zip:

type: string

example: '94301'

xml:

name: address

Category:

type: object

properties:

id:

type: integer

format: int64

example: 1

name:

type: string

example: Dogs

xml:

name: category

User:

type: object

properties:

id:

type: integer

format: int64

example: 10

username:

type: string

example: theUser

firstName:

type: string

example: John

lastName:

type: string

example: James

email:

type: string

example: john@email.com

password:

type: string

example: '12345'

phone:

type: string

example: '12345'

userStatus:

type: integer

description: User Status

format: int32

```
        example: 1
xml:
  name: user
Tag:
  type: object
  properties:
    id:
      type: integer
      format: int64
    name:
      type: string
xml:
  name: tag
Pet:
  required:
    - name
    - photoUrls
  type: object
  properties:
    id:
      type: integer
      format: int64
      example: 10
    name:
      type: string
      example: doggie
    category:
      $ref: '#/components/schemas/Category'
    photoUrls:
```

```
    type: array
    xml:
      wrapped: true
    items:
      type: string
      xml:
        name: photoUrl
  tags:
    type: array
    xml:
      wrapped: true
    items:
      $ref: '#/components/schemas/Tag'
  status:
    type: string
    description: pet status in the store
    enum:
      - available
      - pending
      - sold
  xml:
    name: pet
ApiResponse:
  type: object
  properties:
    code:
      type: integer
      format: int32
  type:
```

```

        type: string
    message:
        type: string
    xml:
        name: '##default'
requestBodies:
    Pet:
        description: Pet object that needs to be added to the store
        content:
            application/json:
                schema:
                    $ref: '#/components/schemas/Pet'
            application/xml:
                schema:
                    $ref: '#/components/schemas/Pet'
    UserArray:
        description: List of user object
        content:
            application/json:
                schema:
                    type: array
                    items:
                        $ref: '#/components/schemas/User'
securitySchemes:
    petstore_auth:
        type: oauth2
        flows:
            implicit:
                authorizationUrl: https://petstore3.swagger.io/oauth/authorize

```

scopes:

write:pets: modify pets in your account

read:pets: read your pets

api_key:

type: apiKey

name: api_key

in: header