Swagger Petstore

Host: petstore.swagger.io, **Base URL:** /v2, **Version:** 1.0.0
Author: apiteam@swagger.io

This is a sample server Petstore server. You can find out more about Swagger at http:// swagger.io or on irc.freenode.net, #swagger. For this sample, you can use the api key special-key to test the authorization filters.

Schemes: http

Summary

pet

Everything about your Pets

Operation	Description
POST /pet	Add a new pet to the store
PUT /pet	Update an existing pet
GET /pet/findByStatus	Finds Pets by status
GET /pet/findByTags	Finds Pets by tags
GET /pet/{petId}	Find pet by ID
POST /pet/{petId}	Updates a pet in the store with form data
DELETE /pet/{petId}	Deletes a pet
POST /pet/{petId}/uploadImage	uploads an image

store

Access to Petstore orders

Operation	Description
GET /store/inventory	Returns pet inventories by status
POST /store/order	Place an order for a pet
GET /store/order/{orderId}	Find purchase order by ID
DELETE /store/order/{orderId}	Delete purchase order by ID

user

Operations about user

Operation	Description
POST /user	Create user
POST /user/createWithArray	Creates list of users with given input array
POST /user/createWithList	Creates list of users with given input array
GET /user/login	Logs user into the system
GET /user/logout	Logs out current logged in user session
GET /user/{username}	Get user by user name
PUT /user/{username}	Updated user
DELETE /user/{username}	Delete user

Paths

POST /pet

Add a new pet to the store

Request Body

Pet object that needs to be added to the store

Definition

Properties

id: *integer (int64)*
category: *undefined*

Properties

id: *integer (int64)*
name: *string*

name: *string*
photoUrls: *string[]*

Items: *string[]*

tags: *object[]*

Items: *object[]*

Properties

id: *integer (int64)*
name: *string*

status: *string, enum("available", "pending", "sold")*
pet status in the store

Response

405 Method Not Allowed
Invalid input

PUT /pet

Update an existing pet

Request Body

Pet object that needs to be added to the store

Definition

Properties

id: *integer (int64)*
category: *undefined*

Properties

id: *integer (int64)*
name: *string*

name: *string*
photoUrls: *string[]*

Items: *string[]*

tags: *object[]*

Items: *object[]*

Properties

id: *integer (int64)*
name: *string*

status: *string, enum("available", "pending", "sold")*
pet status in the store

Response

400 Bad Request
Invalid ID supplied

404 Not Found
Pet not found

405 Method Not Allowed
Validation exception

GET /pet/findByStatus

Multiple status values can be provided with comma separated strings

Request Parameters

Name	Description	Type	Data Type
status	Status values that need to be considered for filter	query	string[]

Response

200 OK
successful operation

Definition

Items: *undefined[]*
  Properties
  id: *integer (int64)*
  category: *undefined*
    Properties
    id: *integer (int64)*
    name: *string*
  name: *string*
  photoUrls: *string[]*
    Items: *string[]*
  tags: *object[]*
    Items: *object[]*
      Properties
      id: *integer (int64)*
      name: *string*
  status: *string, enum("available", "pending", "sold")*
    pet status in the store

400 Bad Request
Invalid status value

GET /pet/findByTags

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

Request Parameters

Name	Description	Type	Data Type
tags	Tags to filter by	query	string[]

Response

200 OK
successful operation

Definition

Items: *undefined[]*
  Properties

id: *integer (int64)*
category: *undefined*

PROPERTIES

> **id:** *integer (int64)*
> **name:** *string*

name: *string*
photoUrls: *string[]*

ITEMS: *string[]*

tags: *object[]*

ITEMS: *object[]*

PROPERTIES

> **id:** *integer (int64)*
> **name:** *string*

status: *string, enum("available", "pending", "sold")*
pet status in the store

400 Bad Request
Invalid tag value

GET /pet/{petId}

Returns a single pet

REQUEST PARAMETERS

Name	Description	Type	Data Type
petId	ID of pet to return	path	integer (int64)

RESPONSE

200 OK
successful operation

See section: #/definitions/Pet

400 Bad Request
Invalid ID supplied

404 Not Found
Pet not found

POST /pet/{petId}

Updates a pet in the store with form data

REQUEST PARAMETERS

Name	Description	Type	Data Type
petId	ID of pet that needs to be updated	path	integer (int64)
name	Updated name of the pet	formData	string
status	Updated status of the pet	formData	string

RESPONSE

405 Method Not Allowed
Invalid input

DELETE /pet/{petId}

Deletes a pet

REQUEST PARAMETERS

Name	Description	Type	Data Type
api_key		header	string
petId	Pet id to delete	path	integer (int64)

Response

400 Bad Request
Invalid ID supplied

404 Not Found
Pet not found

POST /pet/{petId}/uploadImage

uploads an image

Request parameters

Name	Description	Type	Data Type
petId	ID of pet to update	path	integer (int64)
additionalMetadata	Additional data to pass to server	formData	string
file	file to upload	formData	file

Response

200 OK
successful operation

See section: #/definitions/ApiResponse

GET /store/inventory

Returns a map of status codes to quantities

Response

200 OK
successful operation

Definition

Properties

POST /store/order

Place an order for a pet

Request body

order placed for purchasing the pet

Definition

Properties

id: *integer (int64)*
petId: *integer (int64)*
quantity: *integer (int32)*
shipDate: *string (date-time)*
status: *string, enum("placed", "approved", "delivered")*
Order Status
complete: *boolean (default:"false")*

Response

200 OK
successful operation

See section: #/definitions/Order

400 Bad Request
Invalid Order

GET /store/order/{orderId}

For valid response try integer IDs with value >= 1 and <= 10. Other values will generated exceptions

Request parameters

Name	Description	Type	Data Type
orderId	ID of pet that needs to be fetched	path	integer (int64) (1 ≤ x ≤ 10)

Response

200 OK
successful operation

See section: #/definitions/Order

400 Bad Request
Invalid ID supplied

404 Not Found
Order not found

DELETE /store/order/{orderId}

For valid response try integer IDs with positive integer value. Negative or non-integer values will generate API errors

Request parameters

Name	Description	Type	Data Type
orderId	ID of the order that needs to be deleted	path	integer (int64) (1 ≤ x < +∞)

Response

400 Bad Request
Invalid ID supplied

404 Not Found
Order not found

POST /user

This can only be done by the logged in user.

Request body

Created user object

Definition

Properties

id: *integer (int64)*
username: *string*
firstName: *string*
lastName: *string*
email: *string*
password: *string*
phone: *string*
userStatus: *integer (int32)*
User Status

Response

default undefined
successful operation

POST /user/createWithArray

Creates list of users with given input array

Request body

List of user object

Definition

ITEMS: *undefined[]*

PROPERTIES

id: *integer (int64)*
username: *string*
firstName: *string*
lastName: *string*
email: *string*
password: *string*
phone: *string*
userStatus: *integer (int32)*
User Status

RESPONSE

default undefined
successful operation

POST /user/createWithList

Creates list of users with given input array

REQUEST BODY

List of user object

DEFINITION

ITEMS: *undefined[]*

PROPERTIES

id: *integer (int64)*
username: *string*
firstName: *string*
lastName: *string*
email: *string*
password: *string*
phone: *string*
userStatus: *integer (int32)*
User Status

RESPONSE

default undefined
successful operation

GET /user/login

Logs user into the system

REQUEST PARAMETERS

Name	Description	Type	Data Type
username	The user name for login	query	string
password	The password for login in clear text	query	string

RESPONSE

200 OK
successful operation

DEFINITION

400 Bad Request
Invalid username/password supplied

GET /user/logout

Logs out current logged in user session

Response

default undefined
successful operation

GET /user/{username}

Get user by user name

Request Parameters

Name	Description	Type	Data Type
username	The name that needs to be fetched. Use user1 for testing.	path	string

Response

200 OK
successful operation

See section: #/definitions/User

400 Bad Request
Invalid username supplied

404 Not Found
User not found

PUT /user/{username}

This can only be done by the logged in user.

Request Parameters

Name	Description	Type	Data Type
username	name that need to be updated	path	string

Request Body

Updated user object

Definition

Properties

id: *integer (int64)*
username: *string*
firstName: *string*
lastName: *string*
email: *string*
password: *string*
phone: *string*
userStatus: *integer (int32)*
User Status

Response

400 Bad Request
Invalid user supplied

404 Not Found
User not found

DELETE /user/{username}

This can only be done by the logged in user.

Request Parameters

Name	Description	Type	Data Type
username	The name that needs to be deleted	path	string

Response

400 Bad Request
Invalid username supplied

404 Not Found
User not found

Definitions

Order

Definition

Properties

> **id:** *integer (int64)*
> **petId:** *integer (int64)*
> **quantity:** *integer (int32)*
> **shipDate:** *string (date-time)*
> **status:** *string, enum("placed", "approved", "delivered")*
> Order Status
> **complete:** *boolean (default:"false")*

Category

Definition

Properties

> **id:** *integer (int64)*
> **name:** *string*

User

Definition

Properties

> **id:** *integer (int64)*
> **username:** *string*
> **firstName:** *string*
> **lastName:** *string*
> **email:** *string*
> **password:** *string*
> **phone:** *string*
> **userStatus:** *integer (int32)*
> User Status

Tag

Definition

Properties

> **id:** *integer (int64)*
> **name:** *string*

Pet

Definition

Properties

> **id:** *integer (int64)*
> **category:** *undefined*
> **Properties**
> > **id:** *integer (int64)*
> > **name:** *string*
>
> **name:** *string*

photoUrls: *string[]*
  Items: *string[]*
tags: *object[]*
  Items: *object[]*
    Properties
    id: *integer (int64)*
    name: *string*
status: *string, enum("available", "pending", "sold")*
  pet status in the store

ApiResponse

Definition

Properties

code: *integer (int32)*
type: *string*
message: *string*