API

Swagger Petstore’s

Table of Contents

Overview

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.

MARKDOWN HINT

Heading 3

Text attributes

standard *italic*, *italic*, **bold**, **bold**, `monospace`.

Horizontal rule

Bullet list

- apples
- oranges
- pears

Numbered list

1. apples
2. oranges
3. pears

Link

A link.

Tables

Column1	Collumn2	Collumn3
default	center	right
cell1	cell2	cell3
cell1	cell2	cell3

Version information

Version : 1.0.0

Contact information

Contact Email : apiteam@swagger.io

License information

License : Apache 2.0
License URL : http://www.apache.org/¬licenses/¬LICENSE-2.0.html
Terms of service : http://swagger.io/¬terms/

URI scheme

Host : petstore.swagger.io
BasePath : /v2
Schemes : HTTP

Tags

- pet : Everything about your Pets
- store : Access to Petstore orders
- user : Operations about user

Paths

Add a new pet to the store

```
POST /pet
```

Parameters

Type	Name	Description	Schema
Body	**body** *required*	Pet object that needs to be added to the store	Pet

Responses

HTTP Code	Description	Schema
405	Invalid input	No Content

Consumes

- `application/json`
- `application/xml`

Produces

- `application/xml`
- `application/json`

Tags

- pet

Security

Type	Name	Scopes
oauth2	**petstore_auth**	write:pets,read:pets

Example HTTP request

Request path

```
/pet
```

Request body

```json
{
  "id" : 0,
  "category" : {
    "id" : 0,
    "name" : "string"
  },
  "name" : "doggie",
  "photoUrls" : [ "string" ],
  "tags" : [ {
    "id" : 0,
    "name" : "string"
  } ],
  "status" : "string"
}
```

Update an existing pet

```
PUT /pet
```

Parameters

Type	Name	Description	Schema
Body	**body** *required*	Pet object that needs to be added to the store	Pet

Responses

HTTP Code	Description	Schema
400	Invalid ID supplied	No Content
404	Pet not found	No Content
405	Validation exception	No Content

Consumes

- `application/json`
- `application/xml`

Produces

- `application/xml`
- `application/json`

Tags

- pet

Security

Type	Name	Scopes
oauth2	**petstore_auth**	write:pets,read:pets

Example HTTP request

Request path

```
/pet
```

Request body

```
{
  "id" : 0,
  "category" : {
    "id" : 0,
    "name" : "string"
  },
  "name" : "doggie",
  "photoUrls" : [ "string" ],
  "tags" : [ {
    "id" : 0,
    "name" : "string"
  } ],
  "status" : "string"
}
```

Finds Pets by status

```
GET /pet/findByStatus
```

Description

Multiple status values can be provided with comma separated strings

Parameters

Type	Name	Description	Schema
Query	**status** *required*	Status values that need to be considered for filter	< enum (available, pending, sold) > array(multi)

Responses

HTTP Code	Description	Schema
200	successful operation	< Pet > array
400	Invalid status value	No Content

Produces

- `application/xml`
- `application/json`

Tags

- pet

Security

Type	Name	Scopes
oauth2	**petstore_auth**	write:pets,read:pets

Example HTTP request

Request path

```
/pet/findByStatus
```

Request query

```
{
  "status" : "string"
}
```

Example HTTP response

Response 200

```
[ {
  "id" : 0,
  "category" : {
    "id" : 0,
    "name" : "string"
  },
  "name" : "doggie",
  "photoUrls" : [ "string" ],
  "tags" : [ {
    "id" : 0,
    "name" : "string"
  } ],
  "status" : "string"
} ]
```

Finds Pets by tags

```
GET /pet/findByTags
```

operation.deprecated

Description

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

Parameters

Type	Name	Description	Schema
Query	**tags** *required*	Tags to filter by	< string > array(multi)

Responses

HTTP Code	Description	Schema
200	successful operation	< Pet > array
400	Invalid tag value	No Content

Produces

- `application/xml`
- `application/json`

Tags

- pet

Security

Type	Name	Scopes
oauth2	**petstore_auth**	write:pets,read:pets

Example HTTP request

Request path

```
/pet/findByTags
```

Request query

```
{
  "tags" : "string"
}
```

Example HTTP response

Response 200

```
[ {
  "id" : 0,
  "category" : {
    "id" : 0,
    "name" : "string"
  },
  "name" : "doggie",
  "photoUrls" : [ "string" ],
  "tags" : [ {
    "id" : 0,
    "name" : "string"
  } ],
  "status" : "string"
} ]
```

Updates a pet in the store with form data

```
POST /pet/{petId}
```

Parameters

Type	Name	Description	Schema
Path	**petId** *required*	ID of pet that needs to be updated	integer (int64)
FormData	**name** *optional*	Updated name of the pet	string
FormData	**status** *optional*	Updated status of the pet	string

Responses

HTTP Code	Description	Schema
405	Invalid input	No Content

Consumes

- `application/x-www-form-urlencoded`

Produces

- `application/xml`
- `application/json`

Tags

- pet

Security

Type	Name	Scopes
oauth2	**petstore_auth**	write:pets,read:pets

Example HTTP request

Request path

```
/pet/0
```

Request formData

```
"string"
```

Find pet by ID

```
GET /pet/{petId}
```

Description

Returns a single pet

Parameters

Type	Name	Description	Schema
Path	**petId** *required*	ID of pet to return	integer (int64)

Responses

HTTP Code	Description	Schema
200	successful operation	Pet
400	Invalid ID supplied	No Content
404	Pet not found	No Content

Produces

- `application/xml`
- `application/json`

Tags

- pet

Security

Type	Name
apiKey	**api_key**

Example HTTP request

Request path

```
/pet/0
```

Example HTTP response

Response 200

```
{
  "id" : 0,
  "category" : {
    "id" : 0,
    "name" : "string"
  },
  "name" : "doggie",
  "photoUrls" : [ "string" ],
  "tags" : [ {
    "id" : 0,
    "name" : "string"
  } ],
  "status" : "string"
}
```

Deletes a pet

```
DELETE /pet/{petId}
```

Parameters

Type	Name	Description	Schema
Header	**api_key** *optional*		string
Path	**petId** *required*	Pet id to delete	integer (int64)

Responses

HTTP Code	Description	Schema
400	Invalid ID supplied	No Content
404	Pet not found	No Content

Produces

- `application/xml`
- `application/json`

Tags

- pet

Security

Type	Name	Scopes
oauth2	**petstore_auth**	write:pets,read:pets

Example HTTP request

Request path

```
/pet/0
```

Request header

```
"string"
```

uploads an image

```
POST /pet/{petId}/uploadImage
```

Parameters

Type	Name	Description	Schema
Path	**petId** *required*	ID of pet to update	integer (int64)
FormData	**additionalMetadata** *optional*	Additional data to pass to server	string
FormData	**file** *optional*	file to upload	file

Responses

HTTP Code	Description	Schema
200	successful operation	ApiResponse

Consumes

- `multipart/form-data`

Produces

- `application/json`

Tags

- pet

Security

Type	Name	Scopes
oauth2	**petstore_auth**	write:pets,read:pets

Example HTTP request

Request path

```
/pet/0/uploadImage
```

Request formData

```
"file"
```

Example HTTP response

Response 200

```
{
  "code" : 0,
  "type" : "string",
  "message" : "string"
}
```

Returns pet inventories by status

```
GET /store/inventory
```

Description

Returns a map of status codes to quantities

Responses

HTTP Code	Description	Schema
200	successful operation	< string, integer (int32) > map

Produces

- `application/json`

Tags

- store

Security

Type	Name
apiKey	**api_key**

Example HTTP request

Request path

```
/store/inventory
```

Example HTTP response

Response 200

```
"object"
```

Place an order for a pet

```
POST /store/order
```

Parameters

Type	Name	Description	Schema
Body	**body** *required*	order placed for purchasing the pet	Order

Responses

HTTP Code	Description	Schema
200	successful operation	Order
400	Invalid Order	No Content

Produces

- `application/xml`
- `application/json`

Tags

- store

Example HTTP request

Request path

```
/store/order
```

Request body

```
{
  "id" : 0,
  "petId" : 0,
  "quantity" : 0,
  "shipDate" : "string",
  "status" : "string",
  "complete" : true
}
```

Example HTTP response

Response 200

```
{
  "id" : 0,
  "petId" : 0,
  "quantity" : 0,
  "shipDate" : "string",
  "status" : "string",
  "complete" : true
}
```

Find purchase order by ID

```
GET /store/order/{orderId}
```

Description

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

Parameters

Type	Name	Description	Schema
Path	**orderId** *required*	ID of pet that needs to be fetched	integer (int64)

Responses

HTTP Code	Description	Schema
200	successful operation	Order
400	Invalid ID supplied	No Content
404	Order not found	No Content

Produces

- `application/xml`
- `application/json`

Tags

- store

Example HTTP request

Request path

```
/store/order/0
```

Example HTTP response

Response 200

```
{
  "id" : 0,
  "petId" : 0,
  "quantity" : 0,
  "shipDate" : "string",
  "status" : "string",
  "complete" : true
}
```

Delete purchase order by ID

```
DELETE /store/order/{orderId}
```

Description

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

Parameters

Type	Name	Description	Schema
Path	**orderId** *required*	ID of the order that needs to be deleted	integer (int64)

Responses

HTTP Code	Description	Schema
400	Invalid ID supplied	ErrorMessage
404	Order not found	No Content

Produces

- `application/xml`
- `application/json`

Tags

- store

Example HTTP request

Request path

```
/store/order/0
```

Example HTTP response

Response 400

```
{
  "longMessage" : "string",
  "shortMessage" : "string"
}
```

Create user

```
POST /user
```

Description

This can only be done by the logged in user.

Parameters

Type	Name	Description	Schema
Body	**body** *required*	Created user object	User

Responses

HTTP Code	Description	Schema
default	successful operation	No Content

Produces

- `application/xml`
- `application/json`

Tags

- user

Example HTTP request

Request path

```
/user
```

Request body

```
{
  "id" : 0,
  "username" : "string",
  "firstName" : "string",
  "lastName" : "string",
  "email" : "string",
  "password" : "string",
  "phone" : "string",
  "userStatus" : 0
}
```

Creates list of users with given input array

```
POST /user/createWithArray
```

Parameters

Type	Name	Description	Schema
Body	**body** *required*	List of user object	< User > array

Responses

HTTP Code	Description	Schema
default	successful operation	No Content

Produces

- `application/xml`
- `application/json`

Tags

- user

Example HTTP request

Request path

```
/user/createWithArray
```

Request body

```
[ {
  "id" : 0,
  "username" : "string",
  "firstName" : "string",
  "lastName" : "string",
  "email" : "string",
  "password" : "string",
  "phone" : "string",
  "userStatus" : 0
} ]
```

Creates list of users with given input array

```
POST /user/createWithList
```

Parameters

Type	Name	Description	Schema
Body	**body** *required*	List of user object	< User > array

Responses

HTTP Code	Description	Schema
default	successful operation	No Content

Produces

- `application/xml`
- `application/json`

Tags

- user

Example HTTP request

Request path

```
/user/createWithList
```

Request body

```
[ {
  "id" : 0,
  "username" : "string",
  "firstName" : "string",
  "lastName" : "string",
  "email" : "string",
  "password" : "string",
  "phone" : "string",
  "userStatus" : 0
} ]
```

Logs user into the system

```
GET /user/login
```

Parameters

Type	Name	Description	Schema
Query	**password** *required*	The password for login in clear text	string
Query	**username** *required*	The user name for login	string

Responses

HTTP Code	Description	Schema
200	successful operation **Headers** : `X-Rate-Limit` (integer (int32)) : calls per hour allowed by the user. `X-Expires-After` (string (date-time)) : date in UTC when token expires.	string
400	Invalid username/password supplied	No Content

Produces

- `application/xml`
- `application/json`

Tags

- user

Example HTTP request

Request path

```
/user/login
```

Request query

```
{
  "password" : "string",
  "username" : "string"
}
```

Example HTTP response

Response 200

```
"string"
```

Logs out current logged in user session

```
GET /user/logout
```

Responses

HTTP Code	Description	Schema
default	successful operation	No Content

Produces

- `application/xml`
- `application/json`

Tags

- user

Example HTTP request

Request path

```
/user/logout
```

Get user by user name

```
GET /user/{username}
```

Parameters

Type	Name	Description	Schema
Path	**username** *required*	The name that needs to be fetched. Use user1 for testing.	string

Responses

HTTP Code	Description	Schema
200	successful operation	User
400	Invalid username supplied	No Content
404	User not found	No Content

Produces

- `application/xml`
- `application/json`

Tags

- user

Example HTTP request

Request path

```
/user/string
```

Example HTTP response

Response 200

```
{
  "id" : 0,
  "username" : "string",
  "firstName" : "string",
  "lastName" : "string",
  "email" : "string",
  "password" : "string",
  "phone" : "string",
  "userStatus" : 0
}
```

Updated user

```
PUT /user/{username}
```

Description

This can only be done by the logged in user.

Parameters

Type	Name	Description	Schema
Path	**username** *required*	name that need to be updated	string
Body	**body** *required*	Updated user object	User

Responses

HTTP Code	Description	Schema
400	Invalid user supplied	No Content
404	User not found	No Content

Produces

- `application/xml`
- `~~application~~/json`

kjlk jlk

Tags

- user

Example HTTP request

Request path

```
/user/string
```

Request body

```
{
  "id" : 0,
  "username" : "string",
  "firstName" : "string",
  "lastName" : "string",
  "email" : "string",
  "password" : "string",
  "phone" : "string",
  "userStatus" : 0
}
```

Delete user

```
DELETE /user/{username}
```

Description

This can only be done by the logged in user.

Parameters

Type	Name	Description	Schema
Path	**username** *required*	The name that needs to be deleted	string

Responses

HTTP Code	Description	Schema
400	Invalid username supplied	No Content
404	User not found	No Content

Produces

- `application/xml`
- `application/json`

Tags

- user

Example HTTP request

Request path

```
/user/string
```

Definitions

ApiResponse

Name	Description	Schema
code *optional*	**Example** : `0`	integer (int32)
message *optional*	**Example** : `"string"`	string
type *optional*	**Example** : `"string"`	string

Category

Name	Description	Schema
id *optional*	**Example** : `0`	integer (int64)
name *optional*	**Example** : `"string"`	string

ErrorMessage

An error message with a long and a short description

Name	Description	Schema
longMessage *required*	A long error description **Example** : `"string"`	string
shortMessage *required*	A short error description **Example** : `"string"`	string

Order

Name	Description	Schema
complete *optional*	**Default** : `false` **Example** : `true`	boolean
id *optional*	**Example** : `0`	integer (int64)
petId *optional*	**Example** : `0`	integer (int64)
quantity *optional*	**Example** : `0`	integer (int32)
shipDate *optional*	**Example** : `"string"`	string (date-time)
status *optional*	Order Status **Example** : `"string"`	enum (placed, approved, delivered)

Pet

Name	Description	Schema
category *optional*	**Example** : `"Category"`	Category
id *optional*	**Example** : `0`	integer (int64)

Name	Description	Schema
name *required*	**Example** : `"doggie"`	string
photoUrls *required*	**Example** : `[ "string" ]`	< string > array
status *optional*	pet status in the store **Example** : `"string"`	enum (available, pending, sold)
tags *optional*	**Example** : `[ "Tag" ]`	< Tag > array

Tag

Name	Description	Schema
id *optional*	**Example** : `0`	integer (int64)
name *optional*	**Example** : `"string"`	string

User

Name	Description	Schema
email *optional*	**Example** : `"string"`	string
firstName *optional*	**Example** : `"string"`	string
id *optional*	**Example** : `0`	integer (int64)
lastName *optional*	**Example** : `"string"`	string
password *optional*	**Example** : `"string"`	string
phone *optional*	**Example** : `"string"`	string
userStatus *optional*	User Status **Example** : `0`	integer (int32)
username *optional*	**Example** : `"string"`	string

Security

petstore_auth

Type : oauth2
Flow : implicit
Token URL : http://petstore.swagger.io/¬oauth/¬dialog

Name	Description
write:pets	modify pets in your account
read:pets	read your pets

api_key

Type : apiKey
Name : api_key
In : HEADER