Swagger Petstore

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](http://swagger.io/irc/). For this sample, you can use the api key `special-key` to test the authorization filters.

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	Required	Schema	Default
BodyParameter	body	Pet object that needs to be added to the store	true	Pet	

Responses

HTTP Code	Description	Schema
405	Invalid input	No Content

Consumes

- application/json
- application/xml

Produces

- application/xml
- application/json

Tags

- pet

Update an existing pet

```
PUT /pet
```

Parameters

Type	Name	Description	Required	Schema	Default
BodyParameter	body	Pet object that needs to be added to the store	true	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

Finds Pets by status

```
GET /pet/findByStatus
```

Description

Multiple status values can be provided with comma separated strings

Parameters

Type	Name	Description	Required	Schema	Default
QueryParameter	status	Status values that need to be considered for filter	true	multi enum (available, pending, sold) array	

Responses

HTTP Code	Description	Schema
200	successful operation	Pet array

HTTP Code	Description	Schema
400	Invalid status value	No Content

Produces

- application/xml
- application/json

Tags

- pet

Finds Pets by tags

```
GET /pet/findByTags
```

Description

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

Parameters

Type	Name	Description	Required	Schema	Default
QueryParameter	tags	Tags to filter by	true	multi string array	

Responses

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

Produces

- application/xml
- application/json

Tags

- pet

Updates a pet in the store with form data

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

Parameters

Type	Name	Description	Required	Schema	Default
PathParameter	petId	ID of pet that needs to be updated	true	integer (int64)	
FormDataParameter	name	Updated name of the pet	false	string	
FormDataParameter	status	Updated status of the pet	false	string	

Responses

HTTP Code	Description	Schema
405	Invalid input	No Content

Consumes

- application/x-www-form-urlencoded

Produces

- application/xml
- application/json

Tags

- pet

Find pet by ID

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

Description

Returns a single pet

Parameters

Type	Name	Description	Required	Schema	Default
PathParameter	petId	ID of pet to return	true	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

Deletes a pet

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

Parameters

Type	Name	Description	Required	Schema	Default
HeaderParameter	api_key		false	string	
PathParameter	petId	Pet id to delete	true	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

uploads an image

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

Parameters

Type	Name	Description	Required	Schema	Default
PathParameter	petId	ID of pet to update	true	integer (int64)	
FormDataParameter	additionalMetadata	Additional data to pass to server	false	string	
FormDataParameter	file	file to upload	false	file	

Responses

HTTP Code	Description	Schema
200	successful operation	ApiResponse

Consumes

- multipart/form-data

Produces

- application/json

Tags

- pet

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	object

Produces

- application/json

Tags

- store

Place an order for a pet

```
POST /store/order
```

Parameters

Type	Name	Description	Required	Schema	Default
BodyParameter	body	order placed for purchasing the pet	true	Order	

Responses

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

Produces

- application/xml
- application/json

Tags

- store

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	Required	Schema	Default
PathParameter	orderId	ID of pet that needs to be fetched	true	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

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	Required	Schema	Default
PathParameter	orderId	ID of the order that needs to be deleted	true	integer (int64)	

Responses

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

Produces

- application/xml
- application/json

Tags

- store

Create user

```
POST /user
```

Description

This can only be done by the logged in user.

Parameters

Type	Name	Description	Required	Schema	Default
BodyParameter	body	Created user object	true	User	

Responses

HTTP Code	Description	Schema
default	successful operation	No Content

Produces

- application/xml
- application/json

Tags

- user

Creates list of users with given input array

```
POST /user/createWithArray
```

Parameters

Type	Name	Description	Required	Schema	Default
BodyParameter	body	List of user object	true	User array	

Responses

HTTP Code	Description	Schema
default	successful operation	No Content

Produces

- application/xml
- application/json

Tags

- user

Creates list of users with given input array

```
POST /user/createWithList
```

Parameters

Type	Name	Description	Required	Schema	Default
BodyParameter	body	List of user object	true	User array	

Responses

HTTP Code	Description	Schema
default	successful operation	No Content

Produces

- application/xml
- application/json

Tags

- user

Logs user into the system

```
GET /user/login
```

Parameters

Type	Name	Description	Required	Schema	Default
QueryParameter	username	The user name for login	true	string	
QueryParameter	password	The password for login in clear text	true	string	

Responses

HTTP Code	Description	Schema
200	successful operation	string
400	Invalid username/password supplied	No Content

Produces

- application/xml
- application/json

Tags

- user

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

Updated user

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

Description

This can only be done by the logged in user.

Parameters

Type	Name	Description	Required	Schema	Default
PathParameter	username	name that need to be updated	true	string	
BodyParameter	body	Updated user object	true	User	

Responses

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

Produces

- application/xml
- application/json

Tags

- user

Get user by user name

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

Parameters

Type	Name	Description	Required	Schema	Default
PathParameter	username	The name that needs to be fetched. Use user1 for testing.	true	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

Delete user

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

Description

This can only be done by the logged in user.

Parameters

Type	Name	Description	Required	Schema	Default
PathParameter	username	The name that needs to be deleted	true	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

Definitions

ApiResponse

Name	Description	Required	Schema	Default
code		false	integer (int32)	
type		false	string	
message		false	string	

Category

Name	Description	Required	Schema	Default
id		false	integer (int64)	
name		false	string	

Order

Name	Description	Required	Schema	Default
id		false	integer (int64)	
petId		false	integer (int64)	
quantity		false	integer (int32)	
shipDate		false	string (date-time)	
status	Order Status	false	enum (placed, approved, delivered)	
complete		false	boolean	false

Pet

Name	Description	Required	Schema	Default
id		false	integer (int64)	
category		false	Category	
name		true	string	
photoUrls		true	string array	
tags		false	Tag array	
status	pet status in the store	false	enum (available, pending, sold)	

Tag

Name	Description	Required	Schema	Default
id		false	integer (int64)	
name		false	string	

User

Name	Description	Required	Schema	Default
id		false	integer (int64)	
username		false	string	

Name	Description	Required	Schema	Default
firstName		false	string	
lastName		false	string	
email		false	string	
password		false	string	
phone		false	string	
userStatus	User Status	false	integer (int32)	