Swagger Petstore - OpenAPI 3.0 v1.0.12

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.

Some useful links:

- The Pet Store repository
- The source API definition for the Pet Store

서버 정보

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

목차

- POST /user - Create user.
- 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} - Update user resource.
- DELETE /user/{username} - Delete user resource.

pet

Everything about your Pets

PUT /pet

Update an existing pet.

Update an existing pet by Id.

인증 요구사항:

- petstore_auth
  - 접근 권한: write:pets, read:pets

요청

요청 본문:

Update an existent pet in the store

요청 본문 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`name`	string	true	-
`category`	object	false	-
`category.id`	integer	false	-
`category.name`	string	false	-
`photoUrls[]`	array<string>	true	-
`tags[]`	array<object>	false	-
`tags[].id`	integer	false	-
`tags[].name`	string	false	-
`status`	string	false	pet status in the store

Content Type: application/json


```
{
  "id": 10,
  "name": "doggie",
  "category": {
    "id": 1,
    "name": "Dogs"
  },
  "photoUrls": [
    "example_photoUrls_1",
    "example_photoUrls_2"
  ],
  "tags": [
    {
      "id": 0,
      "name": "example_name"
    }
  ],
  "status": "example_status"
}
```


Content Type: application/xml

```
<pet>
  <id>10</id>
  <name>doggie</name>
  <category>
    <id>1</id>
    <name>Dogs</name>
  </category>
  <photoUrls>
    <photoUrl>example_photoUrls_1</photoUrl>
    <photoUrl>example_photoUrls_2</photoUrl>
  </photoUrls>
  <tags>
    <tag>
      <id>0</id>
      <name>example_name</name>
    </tag>
  </tags>
  <status>example_status</status>
</pet>
```

Content Type: application/x-www-form-urlencoded

```
id=10
name=doggie
category.id=1
category.name=Dogs
photoUrls[0]=example_photoUrls_1
photoUrls[1]=example_photoUrls_2
tags[0].id=0
tags[0].name=example_name
status=example_status
```

응답

상태 코드: `200`

설명: Successful operation

응답 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`name`	string	true	-
`category`	object	false	-
`category.id`	integer	false	-
`category.name`	string	false	-
`photoUrls[]`	array<string>	true	-
`tags[]`	array<object>	false	-
`tags[].id`	integer	false	-
`tags[].name`	string	false	-
`status`	string	false	pet status in the store

Content Type: application/json


```
{
  "id": 10,
  "name": "doggie",
  "category": {
    "id": 1,
    "name": "Dogs"
  },
  "photoUrls": [
    "example_photoUrls_1",
    "example_photoUrls_2"
  ],
  "tags": [
    {
      "id": 0,
      "name": "example_name"
    }
  ],
  "status": "example_status"
}
```


Content Type: application/xml

```
<pet>
  <id>10</id>
  <name>doggie</name>
  <category>
    <id>1</id>
    <name>Dogs</name>
  </category>
  <photoUrls>
    <photoUrl>example_photoUrls_1</photoUrl>
    <photoUrl>example_photoUrls_2</photoUrl>
  </photoUrls>
  <tags>
    <tag>
      <id>0</id>
      <name>example_name</name>
    </tag>
  </tags>
  <status>example_status</status>
</pet>
```

상태 코드: `400`

설명: Invalid ID supplied

상태 코드: `404`

설명: Pet not found

상태 코드: `422`

설명: Validation exception

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

POST /pet

Add a new pet to the store.

Add a new pet to the store.

인증 요구사항:

- petstore_auth
  - 접근 권한: write:pets, read:pets

요청

요청 본문:

Create a new pet in the store

요청 본문 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`name`	string	true	-
`category`	object	false	-
`category.id`	integer	false	-
`category.name`	string	false	-
`photoUrls[]`	array<string>	true	-
`tags[]`	array<object>	false	-
`tags[].id`	integer	false	-
`tags[].name`	string	false	-
`status`	string	false	pet status in the store

Content Type: application/json

```
{
  "id": 10,
  "name": "doggie",
  "category": {
    "id": 1,
    "name": "Dogs"
  },
  "photoUrls": [
    "example_photoUrls_1",
    "example_photoUrls_2"
  ],
  "tags": [
    {
      "id": 0,
      "name": "example_name"
    }
  ],
  "status": "example_status"
}
```

Content Type: application/xml

```
<pet>
  <id>10</id>
  <name>doggie</name>
  <category>
    <id>1</id>
    <name>Dogs</name>
  </category>
  <photoUrls>
    <photoUrl>example_photoUrls_1</photoUrl>
    <photoUrl>example_photoUrls_2</photoUrl>
  </photoUrls>
  <tags>
    <tag>
      <id>0</id>
      <name>example_name</name>
    </tag>
  </tags>
  <status>example_status</status>
</pet>
```

Content Type: application/x-www-form-urlencoded

```
id=10
name=doggie
category.id=1
category.name=Dogs
photoUrls[0]=example_photoUrls_1
photoUrls[1]=example_photoUrls_2
tags[0].id=0
tags[0].name=example_name
status=example_status
```

응답

상태 코드: `200`

설명: Successful operation

응답 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`name`	string	true	-
`category`	object	false	-
`category.id`	integer	false	-
`category.name`	string	false	-
`photoUrls[]`	array<string>	true	-
`tags[]`	array<object>	false	-
`tags[].id`	integer	false	-
`tags[].name`	string	false	-
`status`	string	false	pet status in the store

Content Type: application/json

```
{
  "id": 10,
  "name": "doggie",
  "category": {
    "id": 1,
    "name": "Dogs"
  },
  "photoUrls": [
    "example_photoUrls_1",
    "example_photoUrls_2"
  ],
  "tags": [
    {
      "id": 0,
      "name": "example_name"
    }
  ],
  "status": "example_status"
}
```


Content Type: application/xml


```
<pet>
  <id>10</id>
  <name>doggie</name>
  <category>
    <id>1</id>
    <name>Dogs</name>
  </category>
  <photoUrls>
    <photoUrl>example_photoUrls_1</photoUrl>
    <photoUrl>example_photoUrls_2</photoUrl>
  </photoUrls>
  <tags>
    <tag>
      <id>0</id>
      <name>example_name</name>
    </tag>
  </tags>
  <status>example_status</status>
</pet>
```


상태 코드: `400`

설명: Invalid input

상태 코드: `422`

설명: Validation exception

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

GET /pet/findByStatus

Finds Pets by status.

Multiple status values can be provided with comma separated strings.

인증 요구사항:

- petstore_auth
  - 접근 권한: write:pets, read:pets

요청

파라미터:

query 파라미터:

이름	타입	필수 여부	설명
`status`	string	false	Status values that need to be considered for filter (explode: true) (Enum: `available`, `pending`, `sold`)

응답

상태 코드: `200`

설명: successful operation

응답 스키마

응답 형식: 배열

이 응답은 아래 스키마의 배열 형태로 반환됩니다.

배열 아이템 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`name`	string	true	-
`category`	object	false	-
`category.id`	integer	false	-
`category.name`	string	false	-
`photoUrls[]`	array<string>	true	-
`tags[]`	array<object>	false	-
`tags[].id`	integer	false	-
`tags[].name`	string	false	-
`status`	string	false	pet status in the store

Content Type: application/json

```
[
  {
    "id": 10,
    "name": "doggie",
    "category": {
      "id": 1,
      "name": "Dogs"
    },
    "photoUrls": [
      "example_photoUrls_1",
      "example_photoUrls_2"
    ],
    "tags": [
      {
        "id": 0,
        "name": "example_name"
      }
    ],
    "status": "example_status"
  }
]
```


Content Type: application/xml


```
<pet>
  <id>10</id>
  <name>doggie</name>
  <category>
    <id>1</id>
    <name>Dogs</name>
  </category>
  <photoUrls>
    <photoUrl>example_photoUrls_1</photoUrl>
    <photoUrl>example_photoUrls_2</photoUrl>
  </photoUrls>
  <tags>
    <tag>
      <id>0</id>
      <name>example_name</name>
    </tag>
  </tags>
  <status>example_status</status>
</pet>
```


상태 코드: `400`

설명: Invalid status value

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

GET /pet/findByTags

Finds Pets by tags.

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

인증 요구사항:

- petstore_auth
  - 접근 권한: write:pets, read:pets

요청

파라미터:

query 파라미터:

이름	타입	필수 여부	설명
`tags`	array<string>	false	Tags to filter by (explode: true)

응답

상태 코드: `200`

설명: successful operation

응답 스키마

응답 형식: 배열

이 응답은 아래 스키마의 배열 형태로 반환됩니다.

배열 아이템 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`name`	string	true	-
`category`	object	false	-
`category.id`	integer	false	-
`category.name`	string	false	-
`photoUrls[]`	array<string>	true	-
`tags[]`	array<object>	false	-
`tags[].id`	integer	false	-
`tags[].name`	string	false	-
`status`	string	false	pet status in the store

Content Type: application/json

```
[
  {
    "id": 10,
    "name": "doggie",
    "category": {
      "id": 1,
      "name": "Dogs"
    },
    "photoUrls": [
      "example_photoUrls_1",
      "example_photoUrls_2"
    ],
    "tags": [
      {
        "id": 0,
        "name": "example_name"
      }
    ],
    "status": "example_status"
  }
]
```


Content Type: application/xml


```
<pet>
  <id>10</id>
  <name>doggie</name>
  <category>
    <id>1</id>
    <name>Dogs</name>
  </category>
  <photoUrls>
    <photoUrl>example_photoUrls_1</photoUrl>
    <photoUrl>example_photoUrls_2</photoUrl>
  </photoUrls>
  <tags>
    <tag>
      <id>0</id>
      <name>example_name</name>
    </tag>
  </tags>
  <status>example_status</status>
</pet>
```


상태 코드: `400`

설명: Invalid tag value

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

GET /pet/{petId}

Find pet by ID.

Returns a single pet.

인증 요구사항:

- api_key
- petstore_auth
  - 접근 권한: write:pets, read:pets

요청

파라미터:

path 파라미터:

이름	타입	필수 여부	설명
`petId`	integer	true	ID of pet to return

응답

상태 코드: `200`

설명: successful operation

응답 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`name`	string	true	-
`category`	object	false	-
`category.id`	integer	false	-
`category.name`	string	false	-
`photoUrls[]`	array<string>	true	-
`tags[]`	array<object>	false	-
`tags[].id`	integer	false	-
`tags[].name`	string	false	-
`status`	string	false	pet status in the store

Content Type: application/json

```
{
  "id": 10,
  "name": "doggie",
  "category": {
    "id": 1,
    "name": "Dogs"
  },
  "photoUrls": [
    "example_photoUrls_1",
    "example_photoUrls_2"
  ],
  "tags": [
    {
      "id": 0,
      "name": "example_name"
    }
  ],
  "status": "example_status"
}
```


Content Type: application/xml


```
<pet>
  <id>10</id>
  <name>doggie</name>
  <category>
    <id>1</id>
    <name>Dogs</name>
  </category>
  <photoUrls>
    <photoUrl>example_photoUrls_1</photoUrl>
    <photoUrl>example_photoUrls_2</photoUrl>
  </photoUrls>
  <tags>
    <tag>
      <id>0</id>
      <name>example_name</name>
    </tag>
  </tags>
  <status>example_status</status>
</pet>
```


상태 코드: `400`

설명: Invalid ID supplied

상태 코드: `404`

설명: Pet not found

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

POST /pet/{petId}

Updates a pet in the store with form data.

Updates a pet resource based on the form data.

인증 요구사항:

- petstore_auth
  - 접근 권한: write:pets, read:pets

요청

파라미터:

path 파라미터:

이름	타입	필수 여부	설명
`petId`	integer	true	ID of pet that needs to be updated

query 파라미터:

이름	타입	필수 여부	설명
`name`	string	false	Name of pet that needs to be updated
`status`	string	false	Status of pet that needs to be updated

응답

상태 코드: `200`

설명: successful operation

응답 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`name`	string	true	-
`category`	object	false	-
`category.id`	integer	false	-
`category.name`	string	false	-
`photoUrls[]`	array<string>	true	-
`tags[]`	array<object>	false	-
`tags[].id`	integer	false	-
`tags[].name`	string	false	-
`status`	string	false	pet status in the store

Content Type: application/json

```
{
  "id": 10,
  "name": "doggie",
  "category": {
    "id": 1,
    "name": "Dogs"
  },
  "photoUrls": [
    "example_photoUrls_1",
    "example_photoUrls_2"
  ],
  "tags": [
    {
      "id": 0,
      "name": "example_name"
    }
  ],
  "status": "example_status"
}
```


Content Type: application/xml


```
<pet>
  <id>10</id>
  <name>doggie</name>
  <category>
    <id>1</id>
    <name>Dogs</name>
  </category>
  <photoUrls>
    <photoUrl>example_photoUrls_1</photoUrl>
    <photoUrl>example_photoUrls_2</photoUrl>
  </photoUrls>
  <tags>
    <tag>
      <id>0</id>
      <name>example_name</name>
    </tag>
  </tags>
  <status>example_status</status>
</pet>
```


상태 코드: `400`

설명: Invalid input

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

DELETE /pet/{petId}

Deletes a pet.

Delete a pet.

인증 요구사항:

- petstore_auth
  - 접근 권한: write:pets, read:pets

요청

파라미터:

header 파라미터:

이름	타입	필수 여부	설명
`api_key`	string	false	

path 파라미터:

이름	타입	필수 여부	설명
`petId`	integer	true	Pet id to delete

응답

상태 코드: `200`

설명: Pet deleted

상태 코드: `400`

설명: Invalid pet value

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

POST /pet/{petId}/uploadImage

Uploads an image.

Upload image of the pet.

인증 요구사항:

- petstore_auth
  - 접근 권한: write:pets, read:pets

요청

파라미터:

path 파라미터:

이름	타입	필수 여부	설명
`petId`	integer	true	ID of pet to update

query 파라미터:

이름	타입	필수 여부	설명
`additionalMetadata`	string	false	Additional Metadata

요청 본문:

요청 본문 스키마

타입: string

포맷: binary

Content Type: application/octet-stream

응답

상태 코드: `200`

설명: successful operation

응답 스키마

이름	타입	필수 여부	설명
`code`	integer	false	-
`type`	string	false	-
`message`	string	false	-

Content Type: application/json

```
{
  "code": 0,
  "type": "example_type",
  "message": "example_message"
}
```

상태 코드: `400`

설명: No file uploaded

상태 코드: `404`

설명: Pet not found

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

---

store

Access to Petstore orders

GET /store/inventory

Returns pet inventories by status.

Returns a map of status codes to quantities.

인증 요구사항:

- api_key

요청

응답

상태 코드: `200`

설명: successful operation

응답 스키마

타입: object

Content Type: application/json

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

POST /store/order

Place an order for a pet.

Place a new order in the store.

요청

요청 본문:

요청 본문 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`petId`	integer	false	-
`quantity`	integer	false	-
`shipDate`	string	false	-
`status`	string	false	Order Status
`complete`	boolean	false	-

Content Type: application/json

```
{
  "id": 10,
  "petId": 198772,
  "quantity": 7,
  "shipDate": "example_shipDate",
  "status": "approved",
  "complete": false
}
```

Content Type: application/xml

```
<order>
  <id>10</id>
  <petId>198772</petId>
  <quantity>7</quantity>
  <shipDate>example_shipDate</shipDate>
  <status>approved</status>
  <complete>False</complete>
</order>
```

Content Type: application/x-www-form-urlencoded

```
id=10
petId=198772
quantity=7
shipDate=example_shipDate
status=approved
complete=False
```

응답

상태 코드: `200`

설명: successful operation

응답 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`petId`	integer	false	-
`quantity`	integer	false	-
`shipDate`	string	false	-
`status`	string	false	Order Status
`complete`	boolean	false	-

Content Type: application/json


```
{
  "id": 10,
  "petId": 198772,
  "quantity": 7,
  "shipDate": "example_shipDate",
  "status": "approved",
  "complete": false
}
```


상태 코드: `400`

설명: Invalid input

상태 코드: `422`

설명: Validation exception

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

GET /store/order/{orderId}

Find purchase order by ID.

For valid response try integer IDs with value <= 5 or > 10. Other values will generate exceptions.

요청

파라미터:

path 파라미터:

이름	타입	필수 여부	설명
`orderId`	integer	true	ID of order that needs to be fetched

응답

상태 코드: `200`

설명: successful operation

응답 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`petId`	integer	false	-
`quantity`	integer	false	-
`shipDate`	string	false	-
`status`	string	false	Order Status
`complete`	boolean	false	-

Content Type: application/json


```
{
  "id": 10,
  "petId": 198772,
  "quantity": 7,
  "shipDate": "example_shipDate",
  "status": "approved",
  "complete": false
}
```


Content Type: application/xml


```
<order>
  <id>10</id>
  <petId>198772</petId>
  <quantity>7</quantity>
  <shipDate>example_shipDate</shipDate>
  <status>approved</status>
  <complete>False</complete>
</order>
```


상태 코드: `400`

설명: Invalid ID supplied

상태 코드: `404`

설명: Order not found

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

DELETE /store/order/{orderId}

Delete purchase order by identifier.

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

요청

파라미터:

path 파라미터:

이름	타입	필수 여부	설명
`orderId`	integer	true	ID of the order that needs to be deleted

응답

상태 코드: `200`

설명: order deleted

상태 코드: `400`

설명: Invalid ID supplied

상태 코드: `404`

설명: Order not found

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json


```
{
  "code": "example_code",
  "message": "example_message"
}
```


---

user

Operations about user

POST /user

Create user.

This can only be done by the logged in user.

요청

요청 본문:

Created user object

요청 본문 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`username`	string	false	-
`firstName`	string	false	-
`lastName`	string	false	-
`email`	string	false	-
`password`	string	false	-
`phone`	string	false	-
`userStatus`	integer	false	User Status

Content Type: application/json

```
{
  "id": 10,
  "username": "theUser",
  "firstName": "John",
  "lastName": "James",
  "email": "john@email.com",
  "password": "12345",
  "phone": "12345",
  "userStatus": 1
}
```

Content Type: application/xml

```
<user>
  <id>10</id>
  <username>theUser</username>
  <firstName>John</firstName>
  <lastName>James</lastName>
  <email>john@email.com</email>
  <password>12345</password>
  <phone>12345</phone>
  <userStatus>1</userStatus>
</user>
```

Content Type: application/x-www-form-urlencoded

```
id=10
username=theUser
firstName=John
lastName=James
email=john@email.com
password=12345
phone=12345
userStatus=1
```

응답

상태 코드: `200`

설명: successful operation

응답 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`username`	string	false	-
`firstName`	string	false	-
`lastName`	string	false	-
`email`	string	false	-
`password`	string	false	-
`phone`	string	false	-
`userStatus`	integer	false	User Status

Content Type: application/json

```
{
  "id": 10,
  "username": "theUser",
  "firstName": "John",
  "lastName": "James",
  "email": "john@email.com",
  "password": "12345",
  "phone": "12345",
  "userStatus": 1
}
```


Content Type: application/xml


```
<user>
  <id>10</id>
  <username>theUser</username>
  <firstName>John</firstName>
  <lastName>James</lastName>
  <email>john@email.com</email>
  <password>12345</password>
  <phone>12345</phone>
  <userStatus>1</userStatus>
</user>
```


상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json


```
{
  "code": "example_code",
  "message": "example_message"
}
```

POST /user/createWithList

Creates list of users with given input array.

Creates list of users with given input array.

요청

요청 본문:

요청 본문 스키마

요청 형식: 배열

이 응답은 아래 스키마의 배열 형태로 반환됩니다.

배열 아이템 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`username`	string	false	-
`firstName`	string	false	-
`lastName`	string	false	-
`email`	string	false	-
`password`	string	false	-
`phone`	string	false	-
`userStatus`	integer	false	User Status

Content Type: application/json

```
[
  {
    "id": 10,
    "username": "theUser",
    "firstName": "John",
    "lastName": "James",
    "email": "john@email.com",
    "password": "12345",
    "phone": "12345",
    "userStatus": 1
  }
]
```


응답

상태 코드: `200`

설명: Successful operation

응답 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`username`	string	false	-
`firstName`	string	false	-
`lastName`	string	false	-
`email`	string	false	-
`password`	string	false	-
`phone`	string	false	-
`userStatus`	integer	false	User Status

Content Type: application/json

```
{
  "id": 10,
  "username": "theUser",
  "firstName": "John",
  "lastName": "James",
  "email": "john@email.com",
  "password": "12345",
  "phone": "12345",
  "userStatus": 1
}
```


Content Type: application/xml


```
<user>
  <id>10</id>
  <username>theUser</username>
  <firstName>John</firstName>
  <lastName>James</lastName>
  <email>john@email.com</email>
  <password>12345</password>
  <phone>12345</phone>
  <userStatus>1</userStatus>
</user>
```


상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json


```
{
  "code": "example_code",
  "message": "example_message"
}
```

GET /user/login

Logs user into the system.

Log into the system.

요청

파라미터:

query 파라미터:

이름	타입	필수 여부	설명
`username`	string	false	The user name for login
`password`	string	false	The password for login in clear text

응답

상태 코드: `200`

설명: successful operation

응답 스키마

타입: string

Content Type: application/xml

Content Type: application/json

상태 코드: `400`

설명: Invalid username/password supplied

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

GET /user/logout

Logs out current logged in user session.

Log user out of the system.

요청

응답

상태 코드: `200`

설명: successful operation

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

GET /user/{username}

Get user by user name.

Get user detail based on username.

요청

파라미터:

path 파라미터:

이름	타입	필수 여부	설명
`username`	string	true	The name that needs to be fetched. Use user1 for testing

응답

상태 코드: `200`

설명: successful operation

응답 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`username`	string	false	-
`firstName`	string	false	-
`lastName`	string	false	-
`email`	string	false	-
`password`	string	false	-
`phone`	string	false	-
`userStatus`	integer	false	User Status

Content Type: application/json

```
{
  "id": 10,
  "username": "theUser",
  "firstName": "John",
  "lastName": "James",
  "email": "john@email.com",
  "password": "12345",
  "phone": "12345",
  "userStatus": 1
}
```


Content Type: application/xml


```
<user>
  <id>10</id>
  <username>theUser</username>
  <firstName>John</firstName>
  <lastName>James</lastName>
  <email>john@email.com</email>
  <password>12345</password>
  <phone>12345</phone>
  <userStatus>1</userStatus>
</user>
```


상태 코드: `400`

설명: Invalid username supplied

상태 코드: `404`

설명: User not found

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

PUT /user/{username}

Update user resource.

This can only be done by the logged in user.

요청

파라미터:

path 파라미터:

이름	타입	필수 여부	설명
`username`	string	true	name that need to be deleted

요청 본문:

Update an existent user in the store

요청 본문 스키마

이름	타입	필수 여부	설명
`id`	integer	false	-
`username`	string	false	-
`firstName`	string	false	-
`lastName`	string	false	-
`email`	string	false	-
`password`	string	false	-
`phone`	string	false	-
`userStatus`	integer	false	User Status

Content Type: application/json

```
{
  "id": 10,
  "username": "theUser",
  "firstName": "John",
  "lastName": "James",
  "email": "john@email.com",
  "password": "12345",
  "phone": "12345",
  "userStatus": 1
}
```

Content Type: application/xml

```
<user>
  <id>10</id>
  <username>theUser</username>
  <firstName>John</firstName>
  <lastName>James</lastName>
  <email>john@email.com</email>
  <password>12345</password>
  <phone>12345</phone>
  <userStatus>1</userStatus>
</user>
```

Content Type: application/x-www-form-urlencoded

```
id=10
username=theUser
firstName=John
lastName=James
email=john@email.com
password=12345
phone=12345
userStatus=1
```

응답

상태 코드: `200`

설명: successful operation

상태 코드: `400`

설명: bad request

상태 코드: `404`

설명: user not found

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

DELETE /user/{username}

Delete user resource.

This can only be done by the logged in user.

요청

파라미터:

path 파라미터:

이름	타입	필수 여부	설명
`username`	string	true	The name that needs to be deleted

응답

상태 코드: `200`

설명: User deleted

상태 코드: `400`

설명: Invalid username supplied

상태 코드: `404`

설명: User not found

상태 코드: `default`

설명: Unexpected error

응답 스키마

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

Content Type: application/json

```
{
  "code": "example_code",
  "message": "example_message"
}
```

스키마

API에서 사용되는 데이터 모델 스키마입니다.

Order

이름	타입	필수 여부	설명
`id`	integer	false	-
`petId`	integer	false	-
`quantity`	integer	false	-
`shipDate`	string	false	-
`status`	string	false	Order Status
`complete`	boolean	false	-

예시:


```
{
  "id": 10,
  "petId": 198772,
  "quantity": 7,
  "shipDate": "example_shipDate",
  "status": "approved",
  "complete": false
}
```


XML 예시:


```
<order>
  <id>10</id>
  <petId>198772</petId>
  <quantity>7</quantity>
  <shipDate>example_shipDate</shipDate>
  <status>approved</status>
  <complete>False</complete>
</order>
```


Category

이름	타입	필수 여부	설명
`id`	integer	false	-
`name`	string	false	-

예시:

```
{
  "id": 1,
  "name": "Dogs"
}
```

XML 예시:

```
<category>
  <id>1</id>
  <name>Dogs</name>
</category>
```

User

이름	타입	필수 여부	설명
`id`	integer	false	-
`username`	string	false	-
`firstName`	string	false	-
`lastName`	string	false	-
`email`	string	false	-
`password`	string	false	-
`phone`	string	false	-
`userStatus`	integer	false	User Status

예시:

```
{
  "id": 10,
  "username": "theUser",
  "firstName": "John",
  "lastName": "James",
  "email": "john@email.com",
  "password": "12345",
  "phone": "12345",
  "userStatus": 1
}
```

XML 예시:

```
<user>
  <id>10</id>
  <username>theUser</username>
  <firstName>John</firstName>
  <lastName>James</lastName>
  <email>john@email.com</email>
  <password>12345</password>
  <phone>12345</phone>
  <userStatus>1</userStatus>
</user>
```

Tag

이름	타입	필수 여부	설명
`id`	integer	false	-
`name`	string	false	-

예시:

```
{
  "id": 0,
  "name": "example_name"
}
```

XML 예시:

```
<tag>
  <id>0</id>
  <name>example_name</name>
</tag>
```

Pet

이름	타입	필수 여부	설명
`id`	integer	false	-
`name`	string	true	-
`category`	-	false	-
`photoUrls[]`	array<string>	true	-
`tags[]`	array<->	false	-
`status`	string	false	pet status in the store

예시:

```
{
  "id": 10,
  "name": "doggie",
  "photoUrls": [
    "example_photoUrls_1",
    "example_photoUrls_2"
  ],
  "tags": [
    {
      "id": 0,
      "name": "example_name"
    }
  ],
  "status": "example_status"
}
```

XML 예시:

```xml
<pet>
  <id>10</id>
  <name>doggie</name>
  <photoUrls>
    <photoUrl>example_photoUrls_1</photoUrl>
    <photoUrl>example_photoUrls_2</photoUrl>
  </photoUrls>
  <tags>
    <tag>
      <id>0</id>
      <name>example_name</name>
    </tag>
  </tags>
  <status>example_status</status>
</pet>
```

ApiResponse

이름	타입	필수 여부	설명
`code`	integer	false	-
`type`	string	false	-
`message`	string	false	-

예시:

```json
{
  "code": 0,
  "type": "example_type",
  "message": "example_message"
}
```

XML 예시:

```xml
<##default>
  <code>0</code>
  <type>example_type</type>
  <message>example_message</message>
</##default>
```

Error

이름	타입	필수 여부	설명
`code`	string	true	-
`message`	string	true	-

예시:

```
{
  "code": "example_code",
  "message": "example_message"
}
```