

---

記事

[Mihoko Iijima](#) · 2023年2月19日 14m read

## OpenAPI-Suite (OpenAPI 3.0からObjectScriptコードを生成するためのツールセット) : InterSystemsデベロッパーツールコンテスト2023入賞作品のご紹介

開発者の皆さん、こんにちは！

InterSystems デベロッパーツールコンテスト2023 の21の応募作品の中から、[Experts Nomination 第3位に輝いた @Lorenzo Scalese](#) さんの [OpenAPI-Suite \(OpenAPI3.0からObjectScriptコードを生成するためのツールセット\)](#) についてご紹介します。

現時点でIRISはOpenAPI2.0までの対応なのですが、このツールの凄いところは、[OpenAPI3.0に対応している](#)ところです！

提供している機能は、以下の通りです。

1. サーバー側クラスの作成 (OpenAPI2.0と同様に、APIファーストで作成するRESTディスパッチクラスの作成に必要なクラス群を作成します。)
2. HTTPクライアントクラスの作成
3. クライアントプロダクション (ビジネス・サービス、ビジネス・オペレーション、ビジネス・プロセス、Ens.Request、Ens.Response)
4. このツールから生成する機能を指定できるWebインターフェース
5. OpenAPI 1.x、2.x からバージョン3.0への変換ツール

各機能について詳しくは、[@Lorenzo Scales](#) さんが書かれた記事「[OpenAPI Suite - Part 1](#)」「[OpenAPI Suite - Part 2](#)」をご参照ください。

この記事では、1の機能を試してみました。その中でも、以下の対応がとても素晴らしい！と思いました。

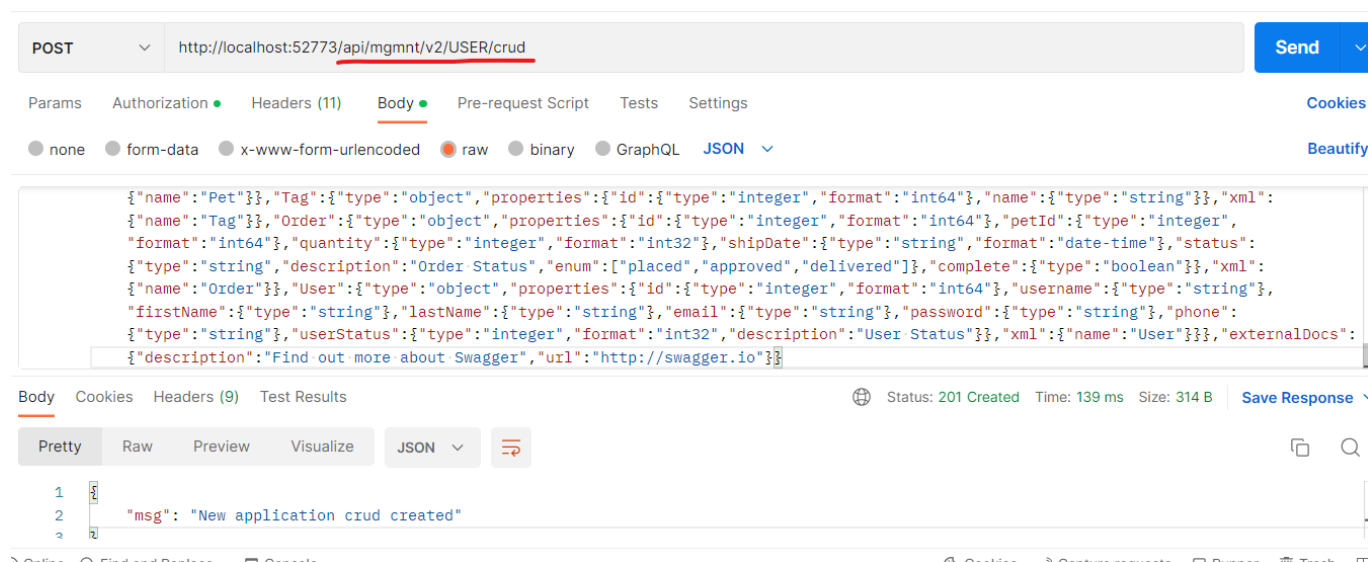
1. OpenAPI 3.0から加わったComponentsオブジェクトの内容に合わせ、クラス定義を自動生成している (指定するパッケージ以下にmodelパッケージができ、そこにインスタンス化できるクラスができています)
2. POSTやPUTの場合、HTTP要求で渡される情報を実装クラス (impl.cls) で処理しやすいように、Bodyで受け取ったJSON文字を1で生成されたクラスのインスタンスに設定するところまで処理してくれています。
  - 。 現時点では、一部手動で修正が必要ですが、そのまま作成されたインスタンスを利用してimpl.clsで %Save()を書いたら保存されました。)

IRISには、OpenAPI2.0の仕様でRESTディスパッチクラスを作成する方法が提供されていますが、使用するクラス群とベースURLの作成のみを行っています。

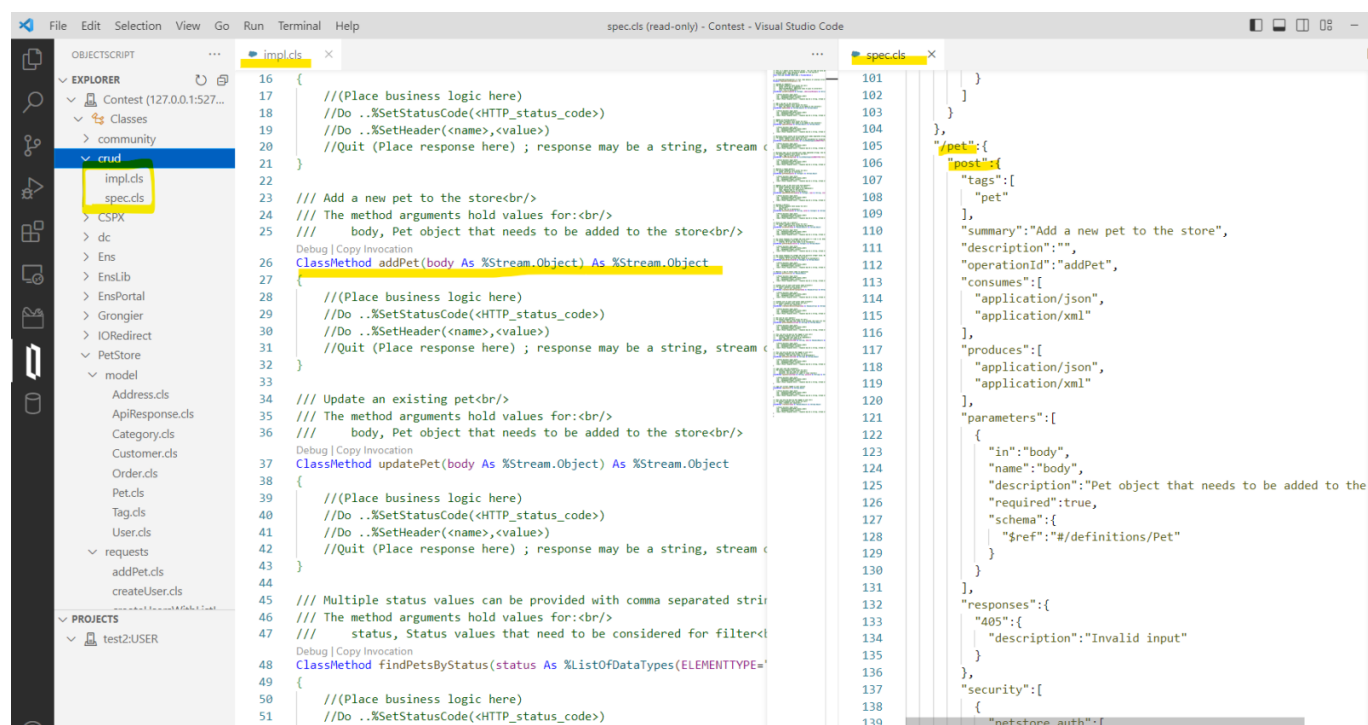
参考までに、IRISが対応しているOpenAPI2.0で作成した内容をご紹介します。

([OpenAPI2.0の仕様で作成されたJSONサンプル](#)を使用しています。)

以下の例では、/curdがベースURLとなり、curdパッケージ以下にクラス群が作成されます。



生成されるクラスは以下の通りです ( 左 : impl.cls、 右 : spec.cls )



ディスパッチクラス ( disp.cls ) は加工不要クラスのため、例ではスタジオでの表示でご紹介しています。

```
crud.dispatch.cls
1 /// This is a sample server Petstore server. You can find out more about Swagger at [http://
2 /// Dispatch class defined by RESTSpec in crud.spec
3 Class crud.dispatch Extends %CSP.REST [ GeneratedBy = crud.spec.cls, ProcedureBlock ]
4 {
5
6 /// The class containing the RESTSpec which generated this class
7 Parameter SpecificationClass = "crud.spec";
8
9 /// Ignore any writes done directly by the REST method.
10 Parameter IgnoreWrites = 1;
11 |
12 /// By default convert the input stream to Unicode
13 Parameter CONVERTINPUTSTREAM = 1;
14
15 /// The default response charset is utf-8
16 Parameter CHARSET = "utf-8";
17
18 XData UriMap [ XMLNamespace = "http://www.intersystems.com/urimap" ]
19 {
20 <Routes>
21 <!-- uploads an image -->
22 <Route Uri="/pet/:petId/uploadImage" Method="post" Call="uploadFile" />
23 <!-- Add a new pet to the store -->
24 <Route Uri="/pet" Method="post" Call="addPet" />
25 <!-- Update an existing pet -->
26 <Route Uri="/pet" Method="put" Call="updatePet" />
27 <!-- Finds Pets by status -->
28 <Route Uri="/pet/findByStatus" Method="get" Call="findPetsByStatus" />
29 }
```

詳細は【[はじめてのInterSystems IRIS](#)】セルフラーニングビデオ：アクセス編：(REST) APIファーストで作成するRESTディスパッチクラスのビデオをご参照ください。

次は [@Lorenzo Scalese](#) さんの [OpenAPI-Suite](#) の結果をご紹介します。

使用するまでの手順はとても簡単です。

[IPM](#) ( InterSystems Package

Mangaer : 以前はZPMとも呼ばれていました ) の [クライアントツール](#)

を管理ポータルかスタジオからインポートしたら、以下コマンドを実行するだけでツールの準備が整います。

#### クライアントツール

のインポートはどのネームスペースでも大丈夫です。管理ポータルからインポートされる場合は、以下メニューを利用します。

管理ポータル システムエクスプローラ クラス インポート対象ネームスペースを選択 インポートボタンをクリック

RESTディスパッチクラスを置きたいネームスペースに移動し、ZPMコマンドを利用してツールをインストールします。

```
zpm "install openapi-suite"
```

インストール時の画面表示

インストール後、以下URLでツールの専用画面にアクセスできます。

<http://localhost:ポート番号/openapisuite/ui/index.csp>

例 ) <http://localhost:52773/openapisuite/ui/index.csp>

OpenAPI-Suite

Github

## Swagger ObjectScript Code Generator

OpenAPI-Suite is a set of tools to generate ObjectScript from OpenAPI specification. This Web Interface allows to generate code for download purpose and also to generate and compile the code on this server.

Application package name

It must be a valid and non existing ObjectScript package name.

What do you want to generate ?

Namespace

Namespace to install the generated code. Leave empty for code download purpose.

Web Application Name

Optional. Leave empty if you don't want create a web application.

OpenAPI specification

Could be the specification in JSON format or an URL pointing to the OpenAPI specification. Specification version 1.x or 2. are automatically converted to Version 3 before processing.

Download Only

Install On Server

Feature disabled by the server. To enable : Set ^openapisuite.config("web","enable-install-")

初期段階では、画面右下の「Install On Server」ボタンは押せなくなっています。

以下グローバル変数に値をセットすることで、ボタンが押せるようです。

( ZPMコマンドでツールをインストールしたネームスペースでセットしてください )

```
Set ^openapisuite.config("web","enable-install-onserver") = 1
```

準備が完了したら、以下設定します。

「Application package name」にはクラス定義のパッケージ名を ( 例では、PetStore )、「What do you want to generate?」では、「REST Server」を選択します。「NameSpace」にはRESTディスパッチクラスを登録したいネームスペースを選択し、「Web Application Name」には、定義したいベースURL ( ウェブ・アプリケーションパス ) を指定します ( 例では、/pet1 )。

後は、OpenAPIの仕様に基づいて作成さ

れたJSONファイルを指定するだけですが、サンプルでは、<https://petstore3.swagger.io/api/v3/openapi.json>がデフォルトで指定されていますのでこのまま使用します。

最後に、「Install On Server」ボタンをクリックします。

以下のようにログが表示されますので、正常に終了したことを確認してCloseボタンをクリックします。



ツールを通して、ウェブアプリケーションパス : /pet1が作成されている予定です。確認してみましょう。

管理ポータル システム管理 セキュリティ アプリケーション ウェブ・アプリケーション から /pet1があるか確認します。

システム > セキュリティ管理 > ウェブ・アプリケーション > ウェブ・アプリケーションの編集 - (セキュリティの設定)

## ウェブ・アプリケーションの編集

保存 キャンセル

ウェブ・アプリケーション /pet1 の定義を編集:

一般 アプリケーション・ロール マッチング・ロール

名前 /pet1  
必須です。(例. /csp/appname)

詳細

ネームスペース USER USERのデフォルト・アプリケーション: /csp/user ☐ ネームスペースのデフォルト・アプリケーション

アプリケーション有効 ☒

有効 ☒ REST  
ディスパッチ・クラス PetStore.disp  
必須です。

空のパスをリダイレクト ☐

☐ CSP/ZEN  
☒ アナリティクス ☒ 着信 Web サービス ☐ ログイン CSRF 攻撃を防ぐ

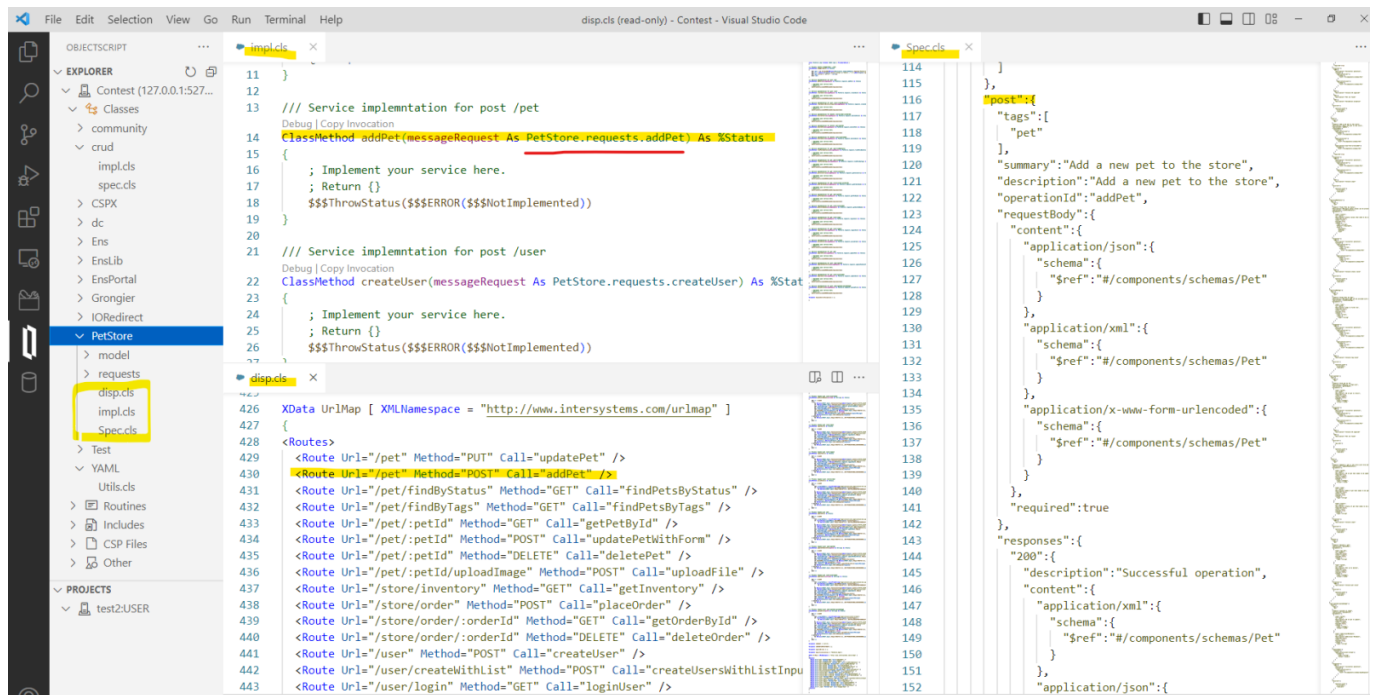
セキュリティの設定  
必要なリソース Group By Id  
許可された認証方法 ☒ 認証なし ☐ パスワード ☐ Kerberos ☐ ログイン Cookie

セッションの設定  
セッションタイムアウト 900 秒 イベントクラス .cls  
セッションにクッキーを使用する 常時 セッションCookieパス /pet1/ セッションCookieのスコープ Strict ユーザCookieスコープ Strict

しっかり作成されていました！

では、入力されたパスに対応するメソッドのマッピングが記載されているdisp ( ディスパッチクラス ) や、コードを実装するimplクラスができていないか確認してみます。

PetStoreパッケージ以下にしっかりクラス定義が作成されていました！



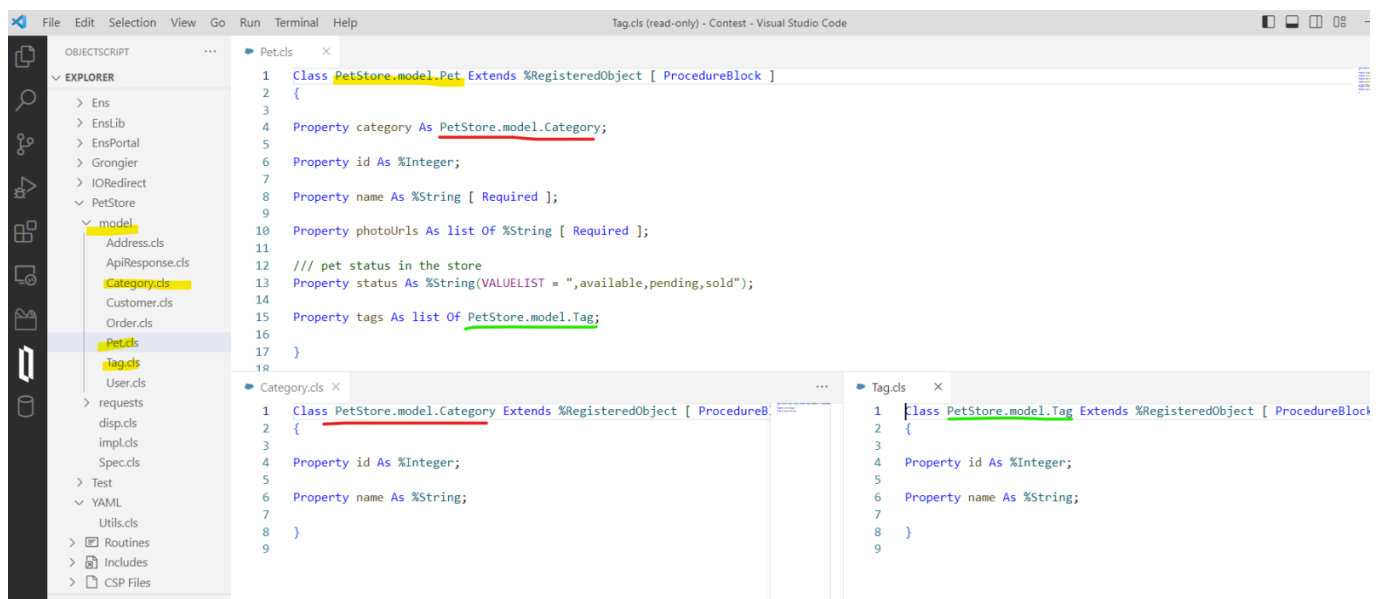
IRISが提供するOpenAPI2.0のクラス群と異なる点は、modelパッケージとrequestsパッケージが含まれている点です。

このツールの中では、OpenAPI3.0のComponentsオブジェクトで指定している情報を、インスタンス生成ができるクラス定義として作成しています。（初期段階では%RegisteredObjectクラスを継承しています）

ComponentsオブジェクトのPetの情報は、以下展開すると参照できます。

Components : PetのSpec

この情報から作成されたクラスは以下の通りです。





インスタンス化ができるクラスで用意されています。

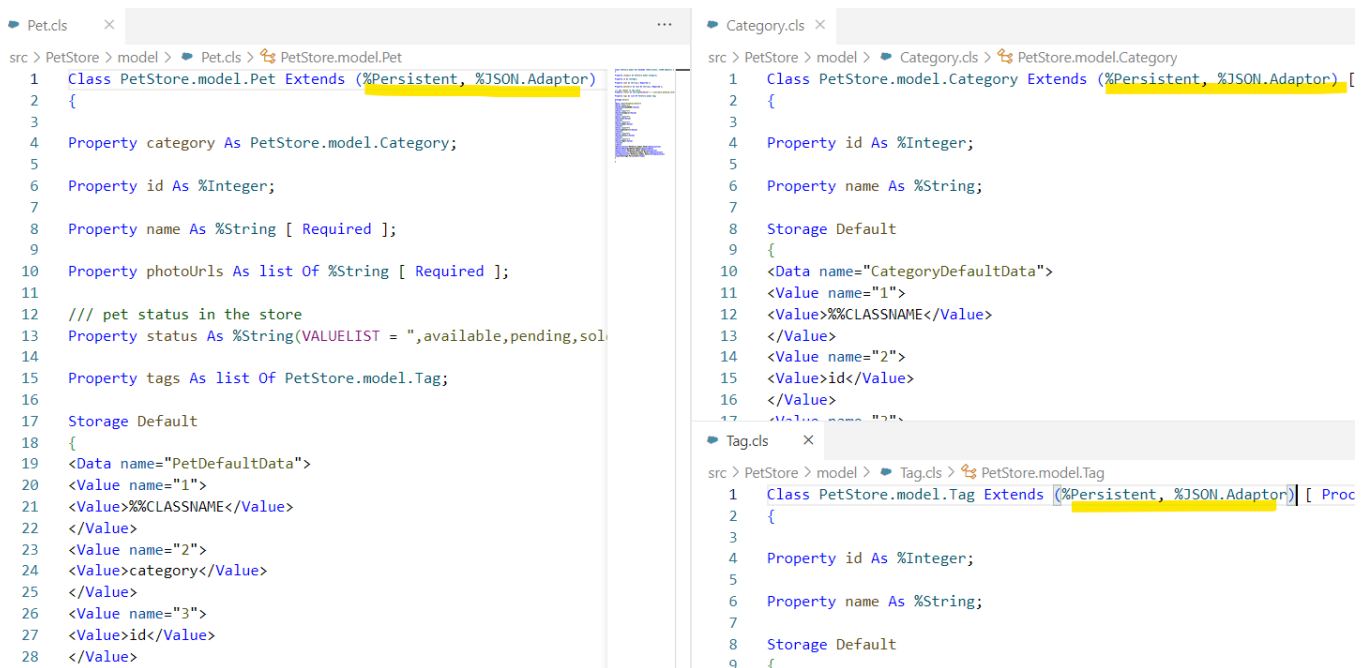
スーパークラスを%RegisteredObjectから%Persistentに変更 + コンパイルを行えば、データベースに登録できる永続クラスとして利用できます。

また、POST要求では、以下の形式でJSONオブジェクトが送付されます。

```
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "Dog"
  },
  "name": "Hachi",
  "photoUrls": [
    "https://x.gd/1pbYK"
  ],
  "tags": [
    {
      "id": 0,
      "name": "Middle"
    }
  ],
  "status": "available"
}
```

このJSONをそのままPet、Tag、Categoryクラスのインスタンスに当てはめることができればいいので、%JSON.Adaptorを追加で継承させるとよさそうです。

ということで、Pet、Tag、Categoryに手を加えるとしたら、スーパークラスの設定を %RegisteredObject から %Persistent と %JSON.Adaptor の多重継承に変えたらよさそうです。



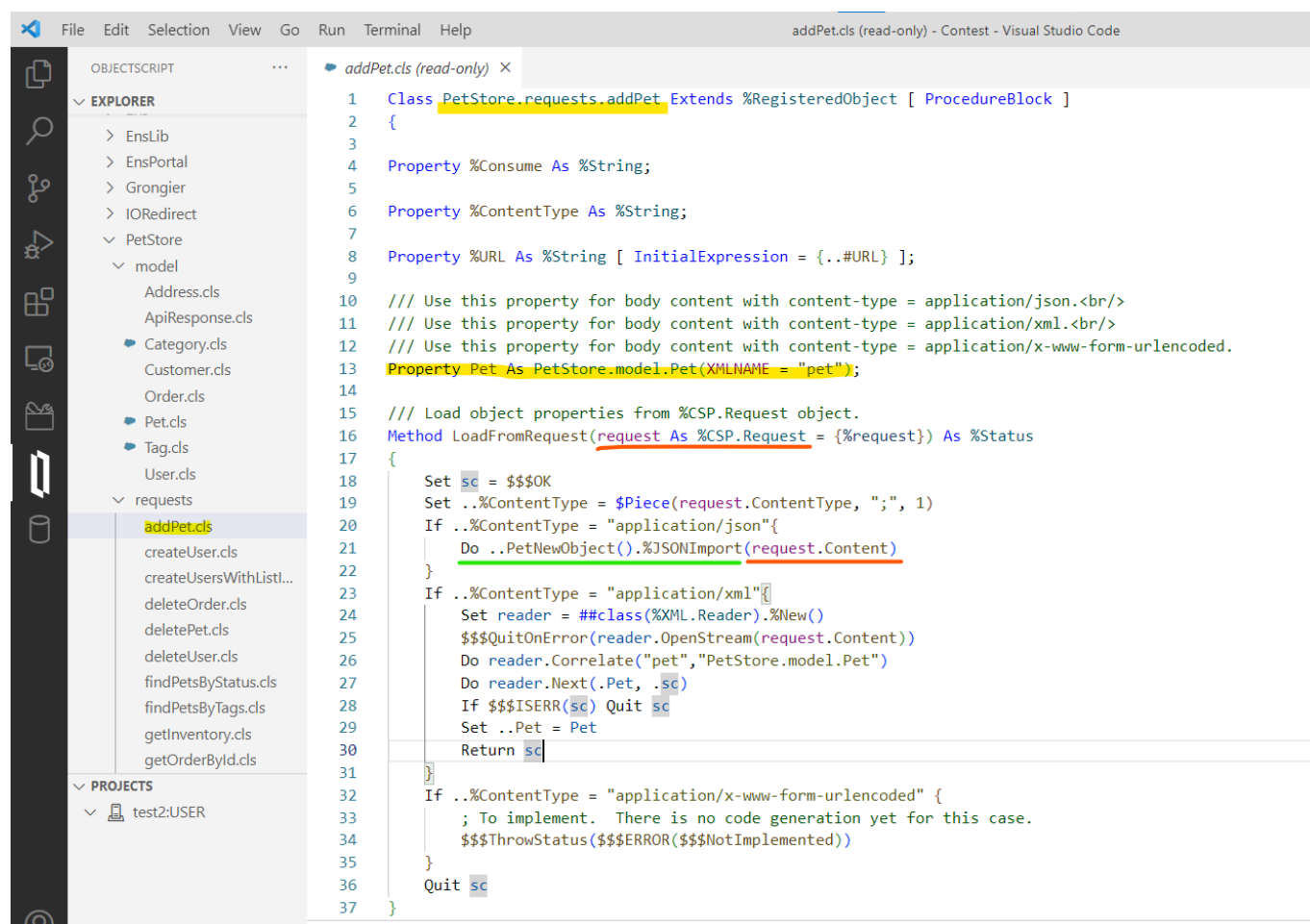
```
src > PetStore > model > PetStore.model.Pet
1 Class PetStore.model.Pet Extends (%Persistent, %JSON.Adaptor)
2 {
3
4 Property category As PetStore.model.Category;
5
6 Property id As %Integer;
7
8 Property name As %String [ Required ];
9
10 Property photoUrls As list Of %String [ Required ];
11
12 /// pet status in the store
13 Property status As %String(VALUELIST = ",available,pending,sold");
14
15 Property tags As list Of PetStore.model.Tag;
16
17 Storage Default
18 {
19 <Data name="PetDefaultData">
20 <Value name="1">
21 <Value>%%CLASSNAME</Value>
22 </Value>
23 <Value name="2">
24 <Value>category</Value>
25 </Value>
26 <Value name="3">
27 <Value>id</Value>
28 </Value>
```

```
src > PetStore > model > CategoryStore.model.Category
1 Class PetStore.model.Category Extends (%Persistent, %JSON.Adaptor) [
2 {
3
4 Property id As %Integer;
5
6 Property name As %String;
7
8 Storage Default
9 {
10 <Data name="CategoryDefaultData">
11 <Value name="1">
12 <Value>%%CLASSNAME</Value>
13 </Value>
14 <Value name="2">
15 <Value>id</Value>
16 </Value>
17 <Value name="3">
```

```
src > PetStore > model > TagStore.model.Tag
1 Class PetStore.model.Tag Extends (%Persistent, %JSON.Adaptor) [ Proc
2 {
3
4 Property id As %Integer;
5
6 Property name As %String;
7
8 Storage Default
9 {
```

この後は、受け取ったBodyの情報をデータベースに更新する処理が必要です。

この処理の大枠を記載してくれているのが、requestパッケージ以下クラスです。以下例では、/petに対してPOST  
要求を行うときに動作するクラス ( PetStore.requests.addPet.cls ) の中身です。



```
1 Class PetStore.requests.addPet Extends %RegisteredObject [ ProcedureBlock ]
2 {
3
4 Property %Consume As %String;
5
6 Property %ContentType As %String;
7
8 Property %URL As %String [ InitialExpression = {..#URL} ];
9
10 /// Use this property for body content with content-type = application/json.<br/>
11 /// Use this property for body content with content-type = application/xml.<br/>
12 /// Use this property for body content with content-type = application/x-www-form-urlencoded.
13 Property Pet As PetStore.model.Pet(XMLNAME = "pet");
14
15 /// Load object properties from %CSP.Request object.
16 Method LoadFromRequest(request As %CSP.Request = {%request}) As %Status
17 {
18     Set sc = $$$OK
19     Set ..%ContentType = $Piece(request.ContentType, ";", 1)
20     If ..%ContentType = "application/json"{
21         Do ..PetNewObject().%JSONImport(request.Content)
22     }
23     If ..%ContentType = "application/xml"{
24         Set reader = ##class(%XML.Reader).%New()
25         $$$QuitOnError(reader.OpenStream(request.Content))
26         Do reader.Correlate("pet", "PetStore.model.Pet")
27         Do reader.Next(.Pet, .sc)
28         If $$$ISERR(sc) Quit sc
29         Set ..Pet = Pet
30         Return sc
31     }
32     If ..%ContentType = "application/x-www-form-urlencoded" {
33         ; To implement. There is no code generation yet for this case.
34         $$$ThrowStatus($$$ERROR($$$NotImplemented))
35     }
36     Quit sc
37 }
```

LoadFromRequest()メソッドに注目すると、引数requestにHTTP要求を操作できる%requestが渡されるようになって  
います ( IRISのRESTディスパッチクラスの内部処理では、HTTP要求は%CSP.Requestクラスのインスタ  
ンスに収納されます )。

request.Content

上記実行でBodyに指定されたJSONオブジェクトを取得できます。

```
Do ..PetNewObject().%JSONImport(request.Content)
```

と記述がありますが、このクラス内にPetNewObject()メソッドがないため、以下のように修正しています ( Petの  
インスタンスを生成し、プロパティPetに設定した後、受信したJSONをインスタンスに割り当てるための処理に  
変えています )。

```
set ..Pet=##class(PetStore.model.Pet).%New()
do ..Pet.%JSONImport(request.Content)
```

これで、HTTP要求を受信したときの処理は終了です。残りは、impl.clsの実装です。

初期段階では、以下の状態で何も記載されていません。

```
ClassMethod addPet(messageRequest As PetStore.requests.addPet) As %Status
```



```
{  
  ; Implement your service here.  
  ; Return {}  
  $$$ThrowStatus( $$$ERROR( $$$NotImplemented ) )  
}
```

引数messageRequestには、1つ前の流れで確認した PetStore.requests.addPet クラスのインスタンスが指定されています ( このインスタンスのPetプロパティにHTTP要求で受信したJSONオブジェクトから作成したインスタンスが設定されています。 )

以下のように書き換え、コンパイルを行ったら修正は終了です！

```
ClassMethod addPet(messageRequest As PetStore.requests.addPet) As %Status  
{  
  ; Implement your service here.  
  ; Return {}  
  //$$$ThrowStatus( $$$ERROR( $$$NotImplemented ) )  
  set status=$$$OK  
  set status=messageRequest.Pet.%Save()  
  return status  
}
```

POST要求をテストしてみます。

The screenshot shows the OpenAPI-Suite interface. At the top, a POST request is configured for the URL `http://localhost:52773/pet1/pet`. The 'Body' tab is selected, showing a JSON payload for adding a pet. The payload includes an ID, category, name, photo URLs, tags, and status. The 'Send' button is visible in the top right. Below the request configuration, the response is displayed with a status of 200 OK, a time of 9 ms, and a size of 258 B. The response is shown in a 'Text' view.

```
1 {  
2   "id": 0,  
3   "category": {  
4     "id": 0,  
5     "name": "Dog"  
6   },  
7   "name": "Hachi",  
8   "photoUrls": [  
9     "https://x.gd/1pbYK"  
10  ],  
11  "tags": [  
12    {  
13      "id": 0,  
14      "name": "Middle"  
15    },  
16  ],  
17  "status": "available"  
18 }
```

Body Cookies Headers (9) Test Results Status: 200 OK Time: 9 ms Size: 258 B Save Response

Pretty Raw Preview Visualize Text

Online Find and Replace Console Cookies Capture requests Runner Trash

PetもCategoryもTagもしっかりデータ登録できました！

```
select id,name,category->name,photourls,status from PetStore_model.Pet
```

Row count: 1 Performance: 0.002 seconds 325 global references 2308 commands executed 0 disk read later

id	name	name	photoUrls	status
0	Hachi	Dog	<a href="https://x.gd/1pbYK">https://x.gd/1pbYK</a>	available

1 row(s) affected

```
select * from PetStore_model.Tag
```

Row count: 1 Performance: 0.002 seconds 324 global referen

ID1	id	name
1	0	Middle

少し加工が必要ですが、OpenAPI3.0のComponentsオブジェクトからクラス定義が自動生成されるところが、とても便利だと思いました！

ぜひ皆さんもお試しく下さい。

#REST API #開発環境 #InterSystems IRIS #InterSystems IRIS for Health

ソースURL:<https://jp.community.intersystems.com/post/openapi-suite%E3%88openapi-30%E3%81%8B%E3%82%89objectscript%E3%82%B3%E3%83%BC%E3%83%89%E3%82%92%E7%94%9F%E6%88%90%E3%81%99%E3%82%8B%E3%81%9F%E3%82%81%E3%81%AE%E3%83%84%E3%83%BC%E3%83%AB%E3%82%BB%E3%83%83%E3%83%88%E3%83%89%E3%83%89%E3%83%87%E3%83%99%E3%83%AD%E3%83%83%E3%83%91%E3%83%BC%E3%83%84%E3%83%BC%E3%83%AB%E3%82%B3%E3%83%B3%E3%83%86%E3%82%B9%E3%83%882023%E5%85%A5%E8%B3%9F%E4%BD%9C%E5%93%81%E3%81%A>

[E%E3%81%94%E7%B4%B9%E4%BB%8B](#)