文章
Michael Lei · 五月 20, 2022 阅读大约需 29 分钟

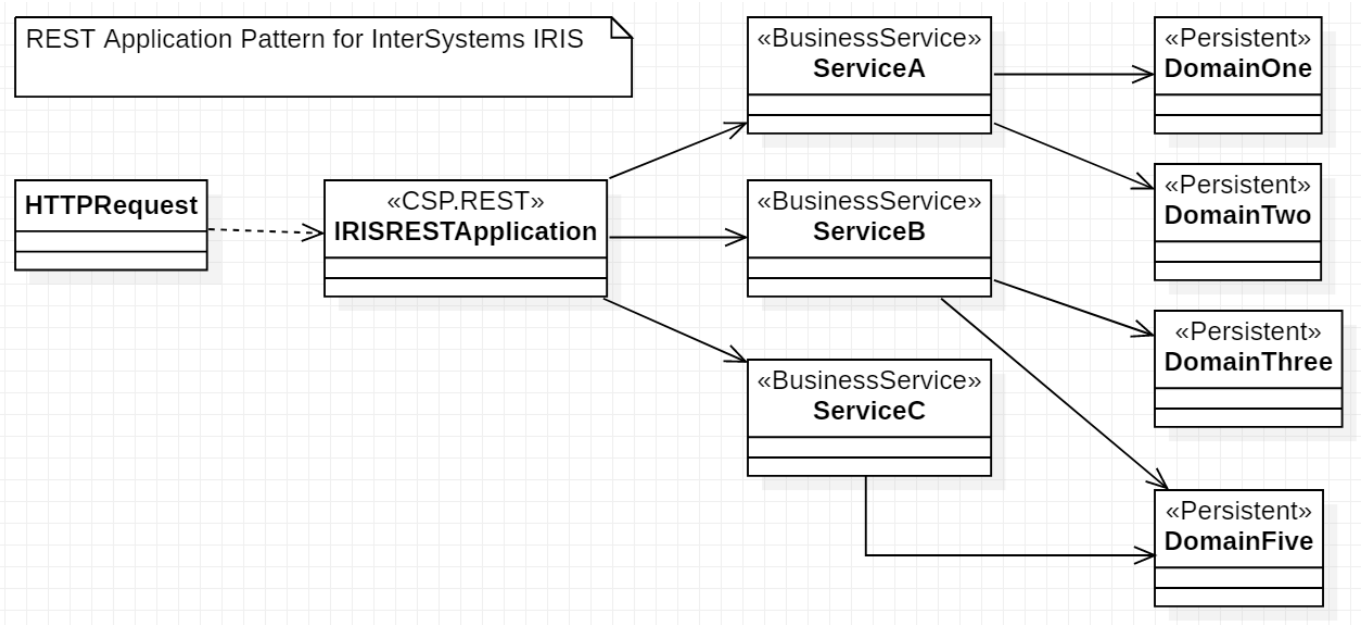
InterSystems IRIS REST API应用程序模式

本文向你推荐一些使用IRIS创建REST API应用程序的模式。

注：所有源代码在https://github.com/yurimarx/movie

类模式到REST应用

首先，请看我对创建IRIS API应用程序所需类的建议:



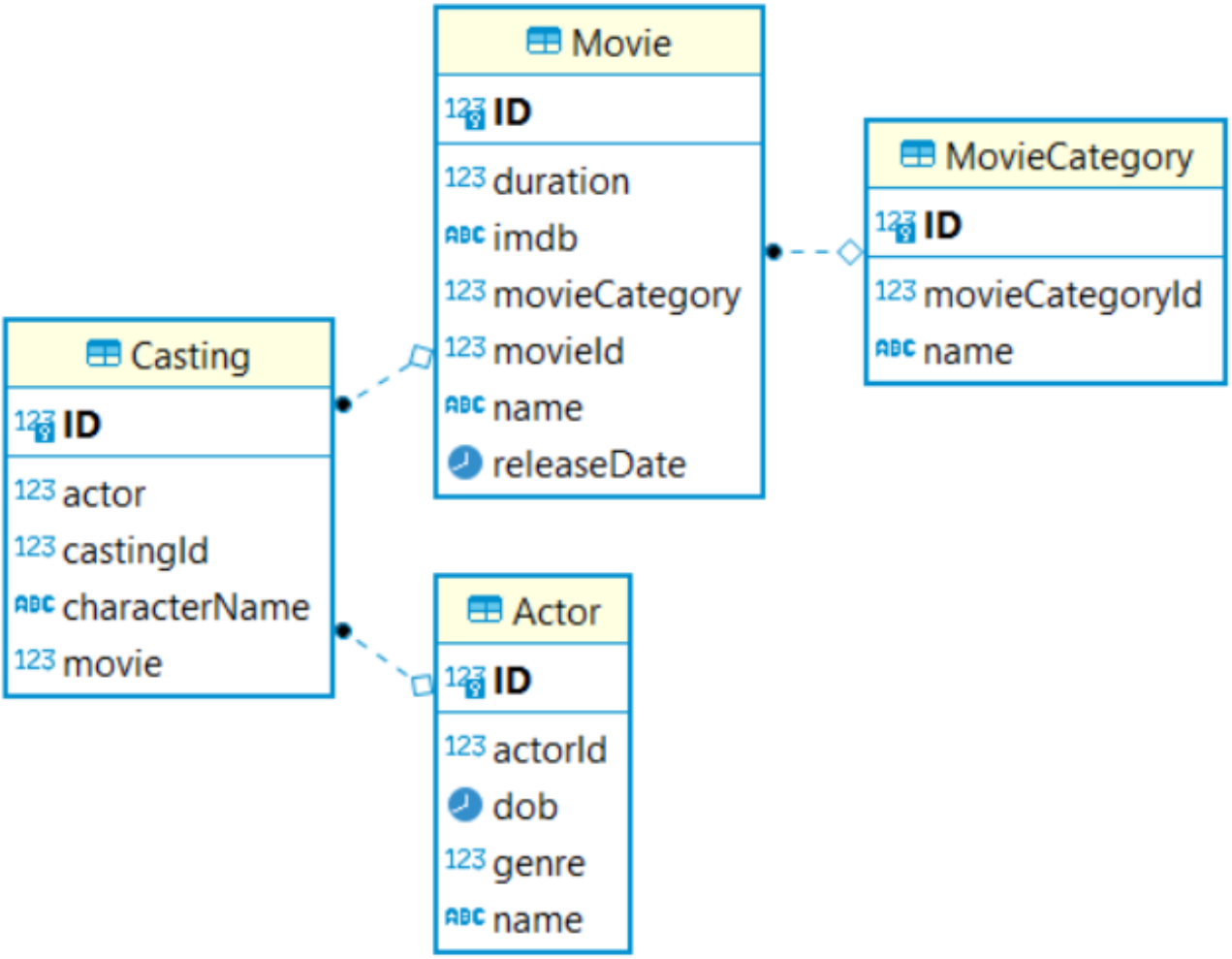
- **IRISRESTApplication**: CSP.REST 类会作为中央控制者来控制业务服务处理的所有REST请求和响应.
- **BusinessService**:
  具有业务主题的类实现。它可以使用一个或多个持久化域类来持久化和查询业务主题要求的数据。
- **Persistent Domain**: 管理SQL表的持久化类.

环境准备

- VSCode;
- Docker Desktop;
- InterSystems ObjectScript Extension Pack.

示例应用的类图

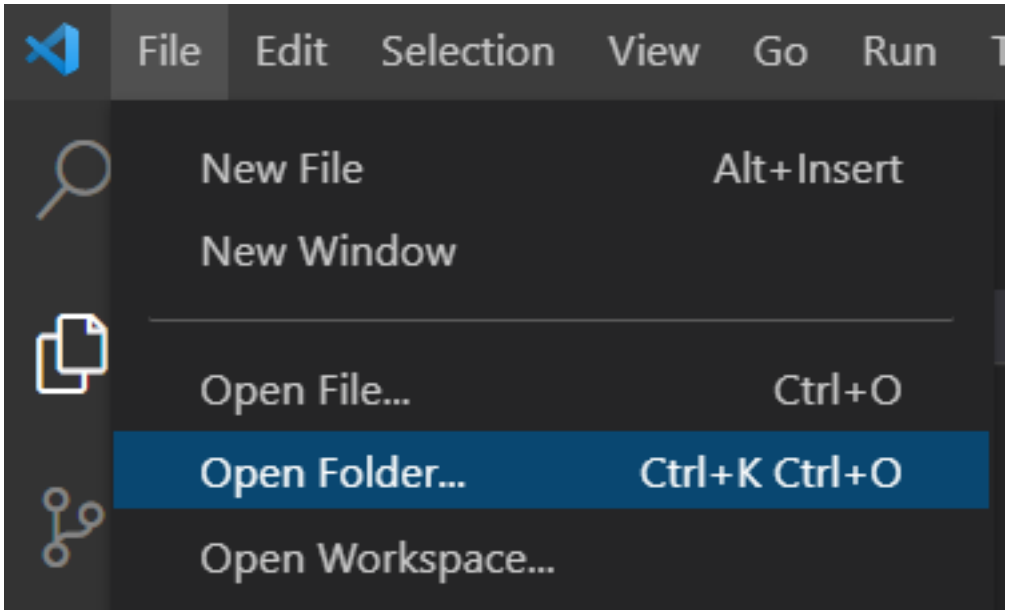
我将创建一个电影目录应用程序来演示文章中建议的模式:



Note: 感谢IRIS API 模版应用 https://openexchange.intersystems.com/package/iris-rest-api-template .
这是本教程的基础.

搭建样本应用

1. 在你的文件系统中创建一个movie文件夹。在一个新的VSCode窗口中打开这个文件夹。



2. 在movie 文件夹中创建 Dockerfile 文件来在Docker container实例中运行IRIS社区版. 内容:

Docker file content

3. 在movie 文件夹中创建 docker-compose.yml 文件来让你同时运行你的docker 和其他实例 (不在本例子中, 但是从docker-compose而不是 dockerfile运行是很好的习惯。内容:

Docker composer content

4. 在movie文件夹中创建iris.script文件，在运行IRIS之前做一些操作。这个文件对于做应用程序所需的自定义终端操作非常重要，比如禁用密码过期。内容:

iris.script content

5. 在movie文件夹中创建module.xml文件，使用ZPM安装和运行你的应用程序。这个文件对于应用程序的端点配置和安装swagger-ui（用于使用swagger文件运行和测试你的API的web应用程序）非常重要。内容:

Module.xml content

You can see CSPApplication tag, used to run the application API in the /movie-api URI and enable or disable password to consume the API.

6. 在movie文件夹中创建LICENSE文件，设置你的应用程序的许可证。内容:

LICENSE content

7. 在movie文件夹中创建README.md文件，用markdown语言向用户记录你的应用程序。:

```
## movie-rest-application
This is a sample of a REST API application built with ObjectScript in InterSystems IRIS.
```

8. 在movie文件夹内创建.vscode文件夹。在.vscode文件夹中创建settings.json文件，以配置VSCode和你的IRIS实例之间的服务器连接。内容:

Settings content

9. 在movie文件夹内创建src文件夹，放置你的源代码文件夹和文件。

10. 在src文件夹内创建dc文件夹。当你建立项目到InterSystems开发者社区时，这是个传统，否则就没有必要。

11. 在dc文件夹内创建movie文件夹。这个文件夹将是你的objectscript类的文件夹。

12. 在 src\dc\movie 文件夹中创建我们的第一个类，MovieRESTApp.cls 文件。这个文件将是IRISRESTApplication类。内容：

MovieRESTApp content

注1：该类扩展了CSP.REST，以作为REST Endpoint使用。

注2：参数chaset是用来对请求和响应进行UTF-8编码的。

注3：CONVERTINPUTSTREAM用于强制要求的内容为UTF-8，如果没有这个参数，你可能会遇到特殊拉丁字母的问题。

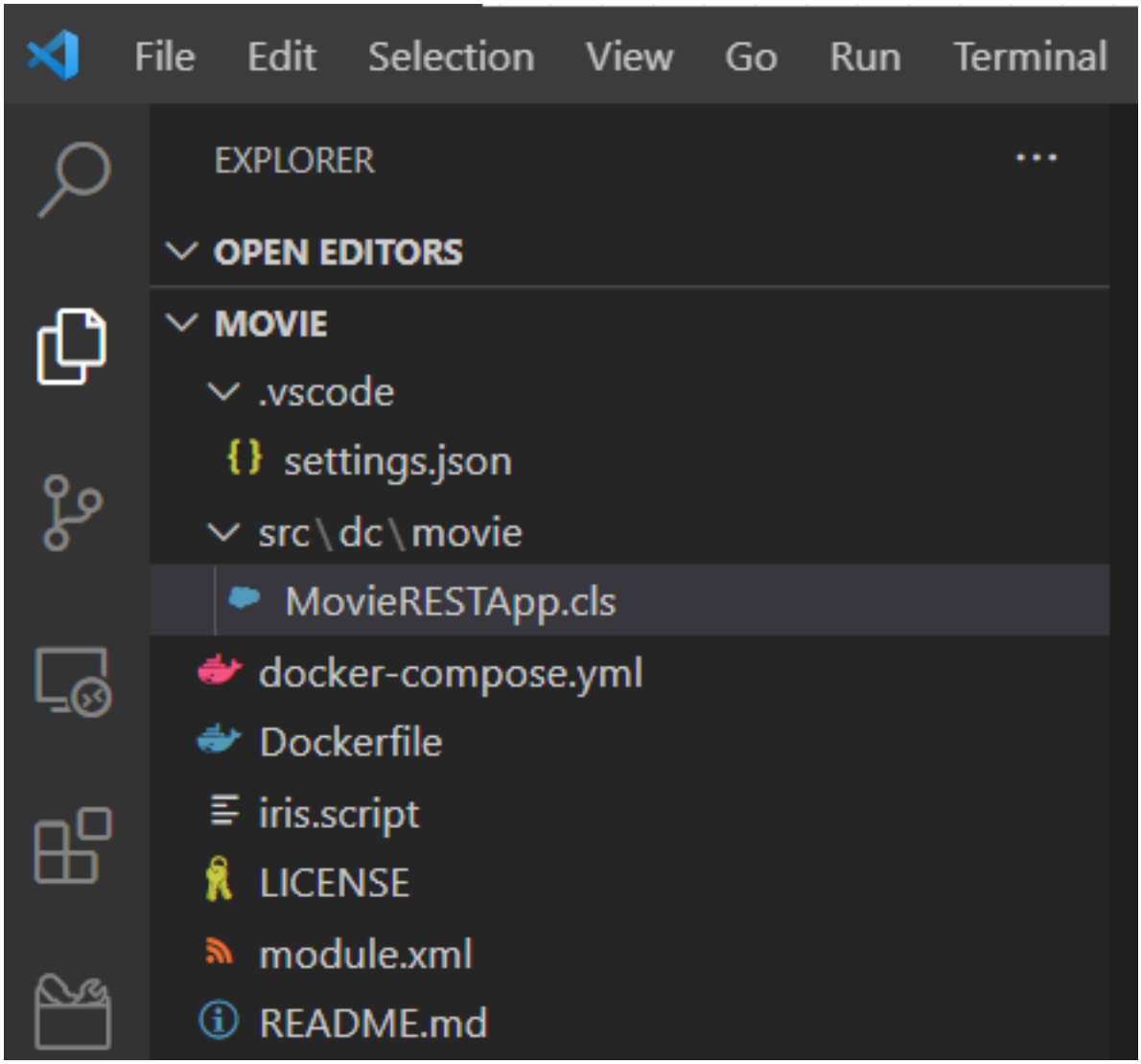
注4：CONTENTTYPE用于使用JSON而不是XML声明内容。

注5：HandleCorsRequest = 1是必要的，它允许你从不同于IRIS服务器的其他服务器上消费API。

注意 6: Routes用于声明每个类方法的API URI。

注7：CSP.REST类的SwaggerSpec允许你生成API swagger（API网络文档）内容。

现在你有以下的文件夹和文件:



13. 打开VSCode终端 (menu Terminal > New Terminal) 并键入:

```
docker-compose up -d --build
```

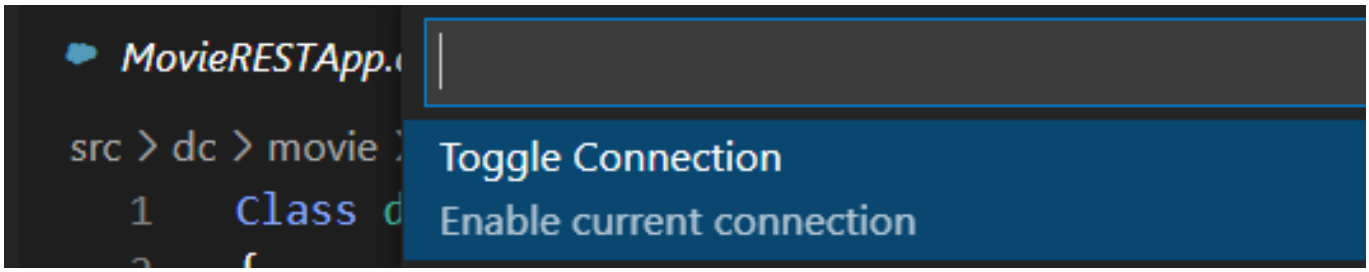
这样就建立一个docker示例并运行。

14. 用Swagger-UI测试API。打开浏览器键入: http://localhost:52773/swagger-ui/index.html. 注意地址栏。

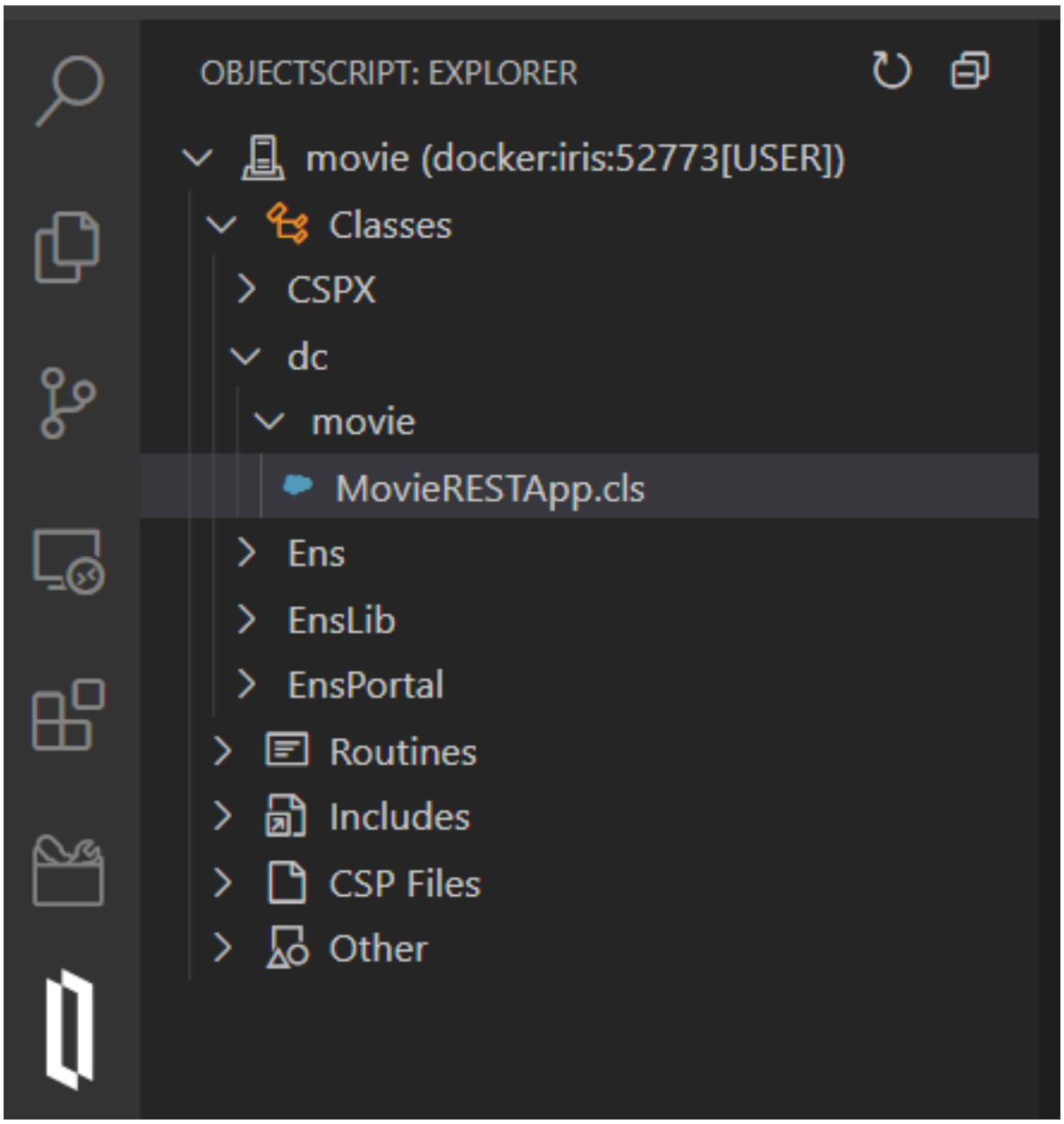
在VSCode和 IRIS 之间建立联系

1. 点击ObjectScript 栏 (VSCode footer)

2. 在顶部选择Toogle 链接:

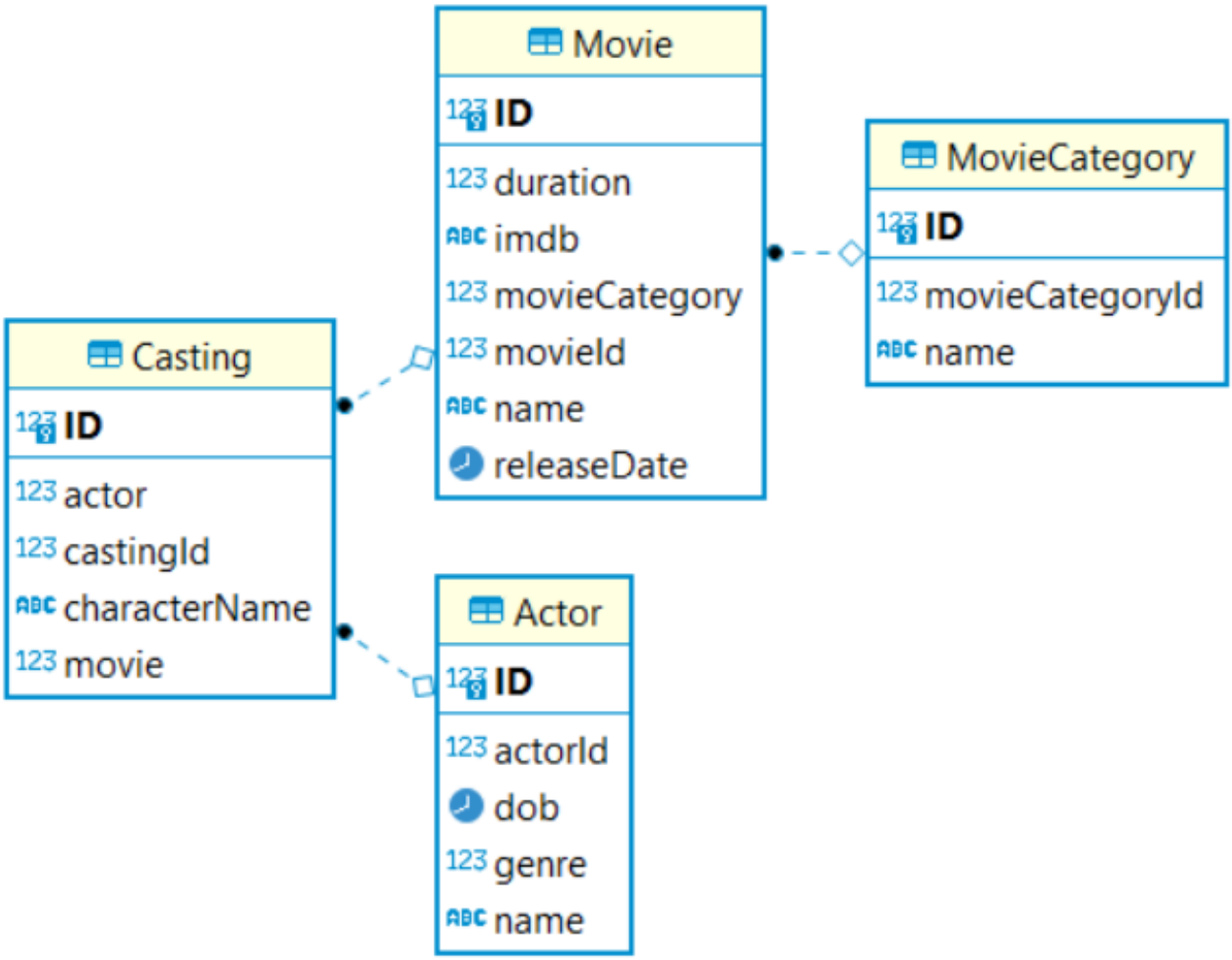


3. 在ObjectScript Explorer里检查链接状态 (你能看到文件夹和创建的类):



Movie 目录应用的持久化类

在这个部分我们会创立持久化域的类来存储和查询业务数据.查看DBeaver图:



1. 在src\dc\movie 目录内创建文件folder模型.

2. 在model目录内创建Actor.cls 文件，写以下内容:

```
Class dc.movie.model.Actor Extends (%Persistent, %JSON.Adaptor)
{
 Parameter %JSONREFERENCE = "ID";
 Property actorId As %Integer [ Calculated, SqlComputeCode = { set {*}={
%%ID}}, SqlComputed ];
Property name As %VarString(MAXLEN = 120); Property dob As %Date;
Property genre As %Integer(VALUELIST = ",1,2");
 }
```

3. 在model目录内创建 Movie.cls 文件，写内容:

```
Class dc.movie.model.Movie Extends (%Persistent, %JSON.Adaptor)
{
 Parameter %JSONREFERENCE = "ID";
 Property movieId As %Integer [ Calculated, SqlComputeCode = { set {*}={
%%ID}}, SqlComputed ];
Property name As %VarString(MAXLEN = 120);
```

```
Property releaseDate As %Date;
Property duration As %Integer;
Property imdb As %String(MAXLEN = 300);
Property movieCategory As dc.movie.model.MovieCategory;
ForeignKey MovieCategoryFK(movieCategory) References dc.movie.model.
MovieCategory();

}
```

4. 在model目录内创建MovieCategory.cls文件，写下如下内容：

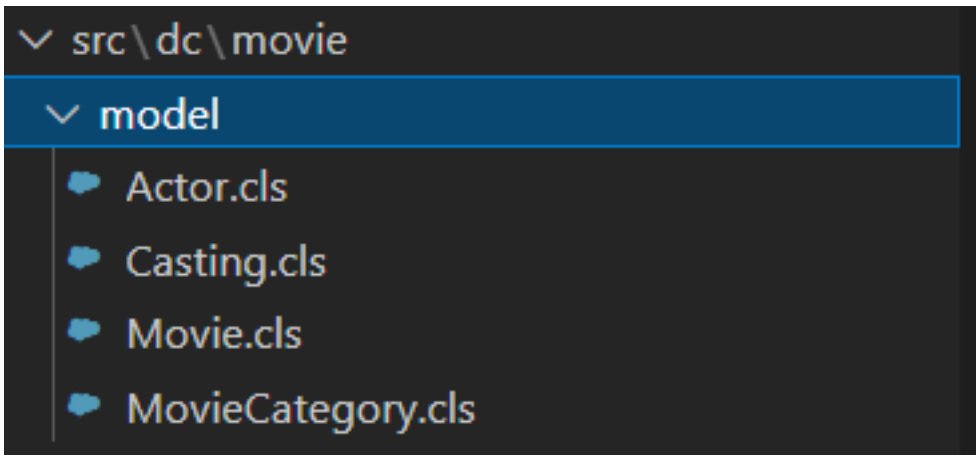
```
 Class dc.movie.model.MovieCategory Extends (%Persistent, %JSON.Adaptor)
{
  Parameter %JSONREFERENCE = "ID";
Property movieCategoryId As %Integer [ Calculated,
SqlComputeCode = { set {*}={%%ID}}, SqlComputed ];
Property name As %VarString(MAXLEN = 120);

}
```

5. 在model目录内创建Casting.cls文件，写下如下内容：

```
 Class dc.movie.model.Casting Extends (%Persistent, %JSON.Adaptor)
{
  Parameter %JSONREFERENCE = "ID";
  Property castingId As %Integer [ Calculated, SqlComputeCode = { set {*}={
%%ID}}, SqlComputed ];
Property movie As dc.movie.model.Movie;
ForeignKey MovieFK(movie) References dc.movie.model.Movie();
Property actor As dc.movie.model.Actor;
ForeignKey ActorFK(actor) References dc.movie.model.Actor();
Property characterName As %String(MAXLEN = 100);
Index CastingIndex On (movie, actor) [ Unique ];
  }
```

查看创建的文件:



注1: 参数 %JSONREFERENCE = "ID" 允许在JSON回复内返回ID 值.

注2: 属性actorId 作为 %Integer [Calculated, SqlComputeCode = { set {*}={%%ID}}, SqlComputed]
和一些其他的类似的属性被用来返回class+id 到 JSON响应中去.

注3: (VALUELIST = "1,2") 设置可能的值到1或者 2。

注4: 外健 MovieFK(movie) 参考 dc.movie.model.Movie() 和类似的用来创建SQL 外键参照.

注5: 在 (movie, actor) [Unique]上建立索引CastingIndex和类似的用来不允许在合并On (movie 和 actor)时重复值

注6: 我使用 Camel Case 做属性名称因为这是JSON属性命名的最佳实践.

业务服务类到Classes to the Movie Catalog Application

在本节中，我们将创建具有业务逻辑的类（做持久性、查询和计算的方法）。

1. 在src/dc/movie中创建服务文件夹。

2. 在服务文件夹中创建CrudUtilService.cls文件。写下内容:

CrudUtilService content

3. 在服务文件夹中创建MovieService.cls文件。编写内容:

MovieService content

4. 在服务文件夹中创建MovieCategoryService.cls文件。编写内容:

MovieCategoryService content

5. 在服务文件夹中创建ActorService.cls文件。编写内容:

ActorService content

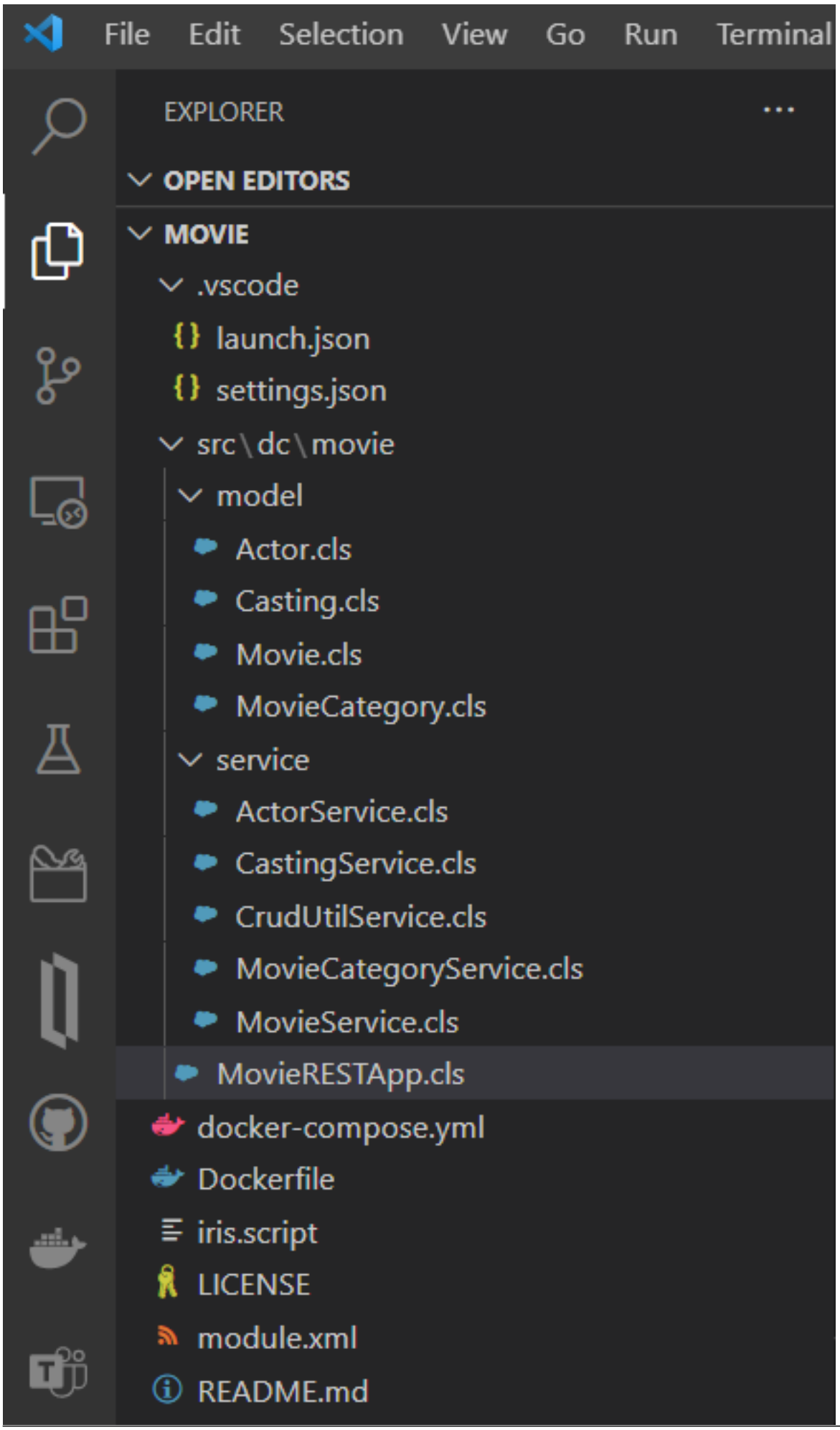
6. 在服务文件夹中创建CastingService.cls文件。编写内容：

CastingService content

7. 更新MovieRESTApp.cls文件，创建所有新服务类方法的路径。编写内容:

MovieRESTApp updated content

8. 最终项目的文件和文件夹是:


File
Edit
Selection
View
Go
Run
Terminal
EXPLORER
OPEN EDITORS
MOVIE
.vscode
launch.json
settings.json
src\dc\movie
model
Actor.cls
Casting.cls
Movie.cls
MovieCategory.cls
service
ActorService.cls
CastingService.cls
CrudUtilService.cls
MovieCategoryService.cls
MovieService.cls
MovieRESTApp.cls
docker-compose.yml
Dockerfile
iris.script
LICENSE
module.xml
README.md

9. 访问 http://localhost:52773/swagger-ui/index.html测试你的新方法

注 1: REST路径是以/id为复数的业务主题，当我们需要将id传递给Camel 和Case的情况下，将路径传递到.

注 2: 我们使用动词GET进行查询，POST用于新记录，PUT用于更新记录，DELETE用于删除记录。

注 3: 在<Route Url="/movies/casting/:id" Method="GET" Call="GetMovieCasting" /> 我使用/casting表示第二个目的（获得Movie和Casting）。这个方法运行ToJSON()，因为它是一个动态数组([])和动态项目({})。

注 4: 我创建了 CrudUtilService ，遵循 "避免重复 "的原则，做通用CRUD方法的小工具。

谢谢观赏!

#REST API #教程 #InterSystems IRIS

源URL:https://cn.community.intersystems.com/post/intersystems-iris-rest-api%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F%E6%A8%A1%E5%BC%8F