文章
Hao Ma · 一月 15, 2021 阅读大约需 6 分钟

Open Exchange

基于InterSystems IRIS开发的RealWorld应用程序

假设您想编写一些真正的web应用程序，例如medium.com网站的简单克隆。这类应用程序可以在后端使用任何不同的语言编写，也可
以使用前端的任何框架编写。编写这
样一个应用程序有很多方法，你也可以看看这个项目
。它为完全相同的应用程序提供了一堆前端和后端实现。您可以轻松组合它们，任何所选前端应该与任何后端搭配。

我来介绍一下这个使用后端InterSystems IRIS来实现后端的相同的应用程序。

RealWorld项目使用REST并提供预设swagger规范，以及Postman/Newman集合
自动化测试。因此，它有助于实现完全相同的REST
API。幸运的是，InterSystems已经实现了通过swagger规范生成REST API实现的方法。最佳实践在这里。

我实现这个应用程序的步骤是：

- 从swagger规范生成API
- 为应用程序中使用的每个对象类型添加一些持久类，包括
  - Users
  - Articles
  - Comments
- 实现API并用Postman测试
- 最后，用任何前端查看实际效果。

用 docker 启动

你可以自己使用docker 来实验一下。

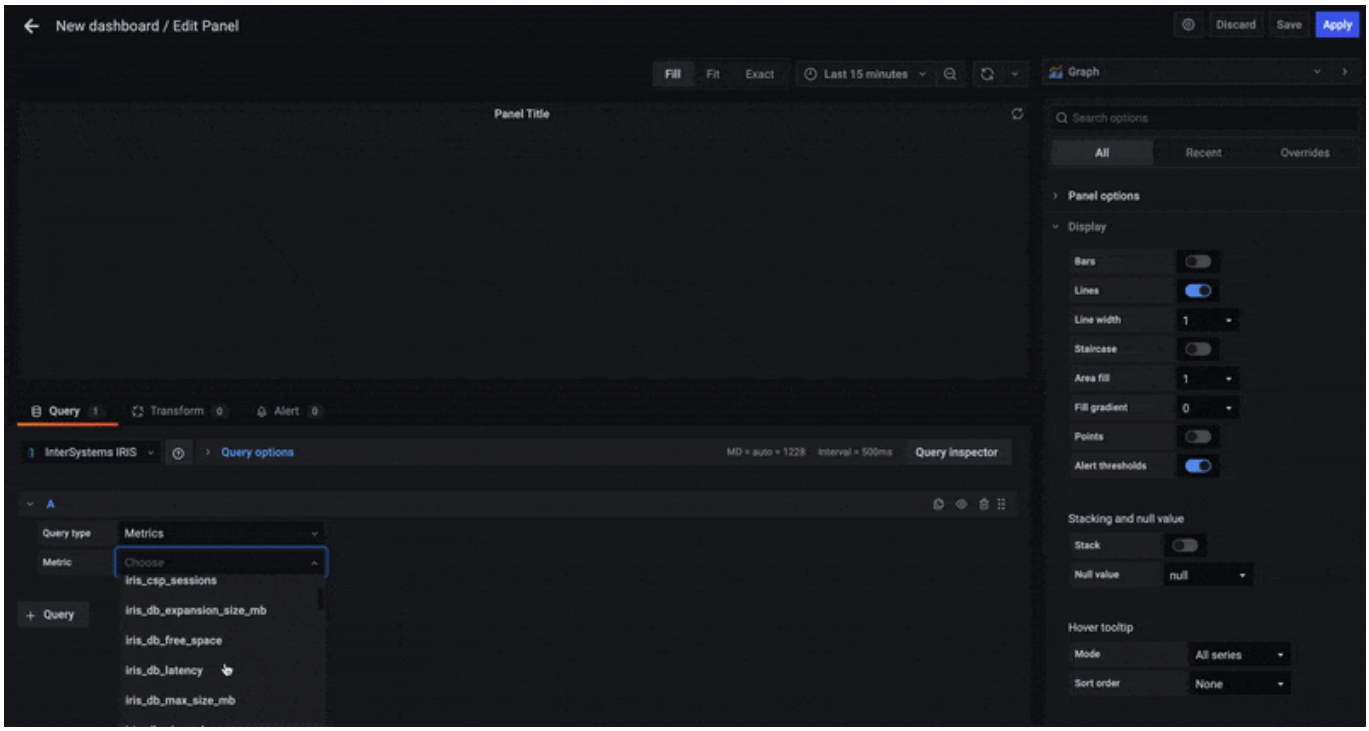
```
// clone github repository
git clone https://github.com/daimor/realworld-intersystems-iris.git
cd realworld-intersystems-iris

// build and run it with docker-compose
docker-compose up -d --build
```

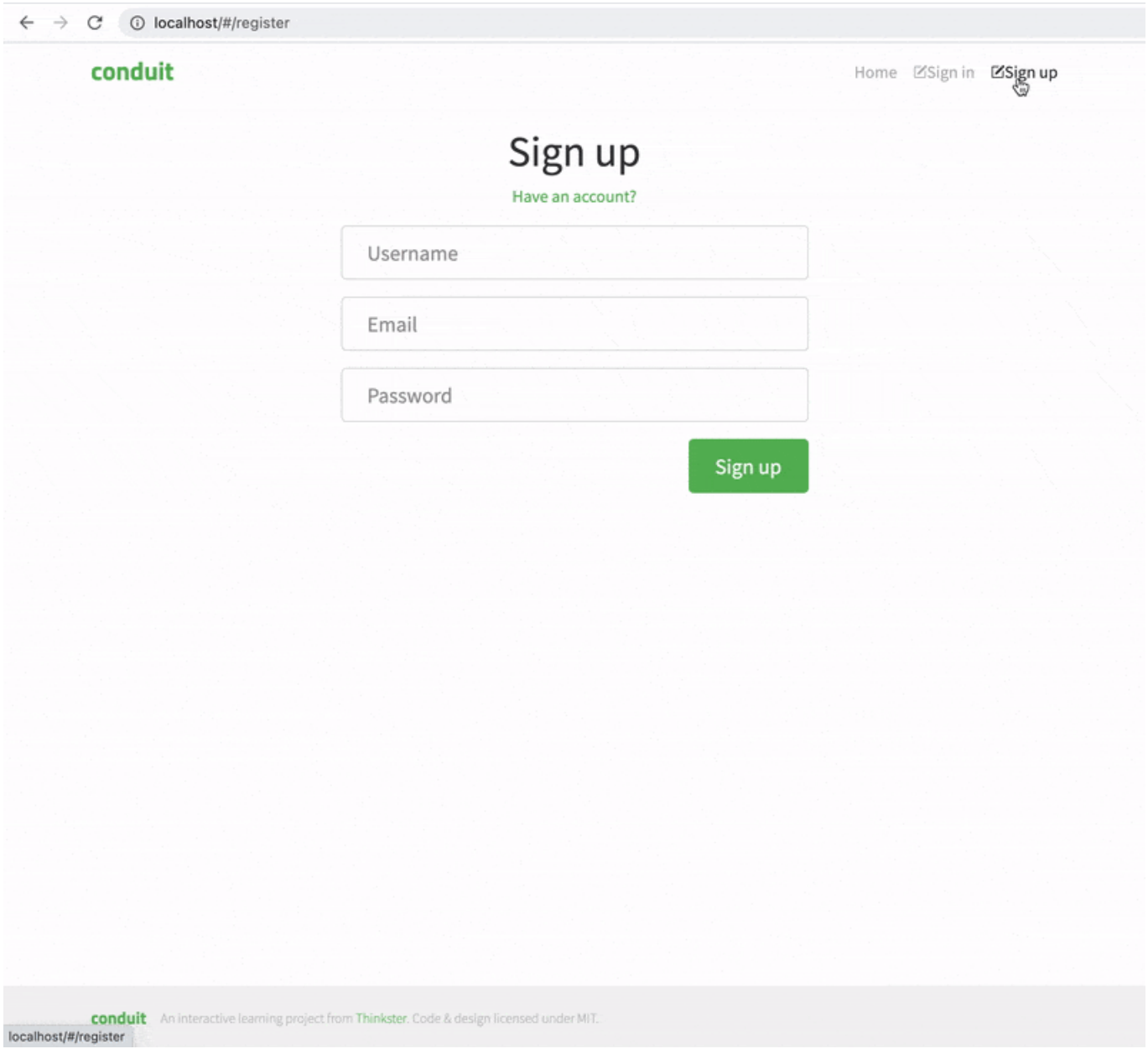
启动后可以通过URL http://localhost:12000/conduit获取IRIS中的REST API，可以用newman测试，需要已安装npm和npx包。

```
APIURL=http://localhost:12000/conduit ./run-api-tests.sh
```

运行Postman的相同测试



可以通过URL http://localhost/访问前端



可以通过zpm运行UnitTest，只需要进入iris会话

```
$ docker-compose exec server iris session iris


Node: 0790684cf488, Instance: IRIS

CONDUIT>zpm
zpm: CONDUIT>test realworld [realworld]    Reload START
[realworld]    Reload SUCCESS
[realworld]    Module object refreshed.
[realworld]    Validate START
[realworld]    Validate SUCCESS
[realworld]    Compile START
[realworld]    Compile SUCCESS
[realworld]    Activate START
[realworld]    Configure START
[realworld]    Configure SUCCESS
[realworld]    Activate SUCCESS
```

```
[realworld]    Test START
Use the following URL to view the result:
http://172.22.0.3:52773/csp/sys/%25UnitTest.Portal.Indices.cls?Index=48&$NAMESPACE=CO
NDUIT
All PASSED
[realworld]    Test SUCCESS
zpm: CONDUIT>
```

默认使用Vue前端，但也能运行Angular和React

```
web=angular docker-compose up -d --build web

web=react docker-compose up -d --build web

web=vue docker-compose up -d --build web
```

通过ZPM安装

InterSystems IRIS部分（后端）可以通过ZPM安装

```
USER>zpm
zpm: USER>install realworld [realworld]    Reload START
[realworld]    Reload SUCCESS
[realworld]    Module object refreshed.
[realworld]    Validate START
[realworld]    Validate SUCCESS
[realworld]    Compile START
[realworld]    Compile SUCCESS
[realworld]    Activate START
[realworld]    Configure START
[realworld]    Configure SUCCESS
[realworld]    Activate SUCCESS
zpm: USER>
```

它将创建/conduit`Web应用程序，只要设置正确的端口，也能用newman进行测试。

```
APIURL=http://localhost:52773/conduit ./run-api-tests.sh
```

可以用ZPM进行UnitTest

```
zpm: USER>test realworld [realworld]    Reload START
[realworld]    Reload SUCCESS
[realworld]    Module object refreshed.
[realworld]    Validate START
[realworld]    Validate SUCCESS
[realworld]    Compile START
[realworld]    Compile SUCCESS
[realworld]    Activate START
[realworld]    Configure START
[realworld]    Configure SUCCESS
[realworld]    Activate SUCCESS
[realworld]    Test START
Use the following URL to view the result:
http://172.17.0.2:52773/csp/sys/%25UnitTest.Portal.Indices.cls?Index=4&$NAMESPACE=USE
R
```

```
All PASSED
[realworld]    Test SUCCESS
```

备注

在开发这个项目的过程中，我遇到了一些问题。

- %JSON.Adaptor
  - 它在导入全新对象时效果非常好。但如果需要部分更新现有对象，则%JSONImport对于来源JSON中应该有的必需字段无效。
    - 所以我没用%JSONImport更新对象，而是用了一个从传入的JSON到对象的简单集（如果定义了值）。
  - 只能导出到字符串、流和输出设备。无法导出到原生JSON
    - API需要返回被另一个对象（属性被命名为返回对象的类型）包装的任何对象。并用%JSONExportToString解决了这个问题，对于数组，将其转换为原生JSON
  - 忽略空集合属性（如：数组和列表）的导出。虽然应用程序可能期望得到字段的空数组，但它根本没有得到任何字段
    - 这个问题没解决。太棘手，只能在%JSON.Adapter端解决。
- %REST - REST实现的生成器及REST实现本身
  - 即使没有任何更改，ṣpec类的任何编译都会更新ïmpl类。因此，切记保持生成的部分（如：方法名、参数列表和变量名）不变，否则将会被下一次ṣpec编译重写，这可能会在构建用于生产的应用程序时发生。
  - REST可以有/̀users/̀端点，也能获取/̀users/请求，在这种情况下，两者效果相同。但如果只定义了第一种方式，REST就不能识别第二种方式。
    - 要解决这个问题，必须修改swagger规范，只复制带新端点/̀users/的/̀users/̀
  - Swagger规范定义了参数化请求的默认值，而生成器忽略了
    - 在方法/生成器的代码中手动设置默认值可能会重写方法定义，而参数的设置默认值将被删除。所以可能在部署后破坏实现。
  - %REST.REST中的方法不可用于覆盖，仅由d̈isp类使用，且将被ṣpec类的编译完全重写。
    - 无法访问实例的OnPreDispatch方法，也就无法进行更多控制，如：检查访问
  - Swagger规范定义了哪些端点是公共的，哪些需要授权。%REST生成器无法使用。
    - API必须用JWT来授权请求，且必须手动检查哪个端点需要检查访问。超出了%OAuth2实现的范围，在IRIS中使用JWT也很麻烦。
  - 在ïmpl类中生成的方法应该返回原生JSON对象、流或字符串。但我认为如果它也能接受%JSON.Adaptor对象就很好。

无论如何，实现这样的应用程序是非常有趣的。至少知道了可以用IRIS来实现。

这个应用程序使用了IRIS的这些特性

- 原生JSON + %JSON.Adaptor
- REST，及其遵守swagger规范的实现生成器
- OAuth2的JWT
- 容器化

竞赛

这个项目正在参加InterSystems全栈竞赛，如果您喜欢请投票。

#Angular #JSON #OAuth2 #React #REST API #Vue.js #竞赛 #InterSystems IRIS
在 InterSystems Open Exchange 上检查相关应用程序

源

URL:https://cn.community.intersystems.com/post/%E5%9F%BA%E4%BA%8Eintersystems-iris%E5%BC%80%E5%8F%91%E7%9A%84realworld%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F