Artigo
Evgeny Shvarov · Out. 26, 2020 5min de leitura

Open Exchange

Trabalhando com vários projetos ObjectScript simultâneos utilizando VSCode e Docker no InterSystems IRIS

Olá, desenvolvedores!

```
"objectscript.conn" :{
      "ns": "IRISAPP",
      "active": true,
      "docker-compose": {
        "service": "iris",
        "internalPort": 52773
      }
```

Quero compartilhar com vocês um novo recurso bem maneiro que descobri no novo lançamento 0.8 do plugin VSCode ObjectScript de @Dmitry Maslennikov e CaretDev.

O lançamento traz uma nova configuração "docker-compose", que resolve o problema com as portas necessárias para fazer o VSCode Editor se conectar à IRIS. Não era muito conveniente se você tivesse mais de um contêiner Docker com a IRIS em execução na mesma máquina. Agora, esse problema foi resolvido!

Veja abaixo como funciona.

O conceito de usar o Docker localmente para desenvolvimento com a IRIS pressupõe que você tem o dockerfile e o docker-compose.yml no repositório usado para compilar o ambiente do projeto e carregar todo o código ObjectScript no contêiner IRIS obtido pelo Docker Hub. Além disso, você tem o arquivo .vscode/settings.json do VSCode no repositório ao qual você aponta a porta de conexão do servidor web IRIS (junto com outras configurações de conexão, como URL, Namespace e credenciais de login).

A pergunta é: qual é a porta à qual o VSCode deve se conectar?

Você pode usar a porta 52773, que é a porta IRIS padrão para servidores web. Mas, se você tentar iniciar o segundo contêiner do Docker, haverá uma falha, pois você não pode executar dois contêineres do Docker na mesma máquina que esperam conexões pela mesma porta. Mas você pode expor uma porta externa para o contêiner do Docker, e isso pode ser configurado por um arquivo docker-compose.yml. Veja um exemplo (a porta mapeada está em negrito):

```
version: '3.6'
services:
  iris:
    build: .
    restart: always
    ports:
      - 52791:52773
    volumes:
      - ~/iris.key:/usr/irissys/mgr/iris.key
      - ./:/irisdev/app
```

No docker-compose, você deve inserir a mesma porta em .vscode/settings.json:

Mas qual é o problema?

O problema é que, quando você expõe seu projeto como uma biblioteca ou demonstra ç ão e convida pessoas a executar e editar o código com o VSCode, você não quer que elas configurem a porta manualmente e quer que elas clonem o repositório, executem o Docker e tenham a op ç ão de colaborar imediatamente. Eis a pergunta: no docker-compose, a que você deve mapear seu projeto para que não haja conflito com o ambiente de outra pessoa?

Mesmo se você não o expuser a ninguém exceto você mesmo, que porta deve colocar nas configura ç ões de conexão do .vscode/settings.json?

A resposta é: quando você inicia um novo Docker com a IRIS, é exibida a mensagem de erro dizendo que a porta já está sendo usada e que você deve interromper os outros contêineres ou inventar uma nova porta que provavelmente não esteja em uso e tentar usá-la no docker-compose e settings.json.

Que chatice! É uma opera ç ão inútil que gasta tempo demais. Ninguém gosta de fazer isso.

O mesmo ocorre se você expuser a biblioteca.

O alívio chegou com o novo lan ç amento 0.8 do VSCode ObjectScript, em que você pode incluir uma se ç ão docker-compose, que resolve o problema para sempre:

```
"objectscript.conn" :{
      "ns": "IRISAPP",
      "active": true,
      "docker-compose": {
        "service": "iris",
        "internalPort": 52773
      }
```

A se ç ão contém os parâmetros "service" e "internalPort", que informam ao VSCode que, para encontrar a porta de conexão, deve-se verificar o arquivo docker-compose.yml que temos no mesmo repositório, encontrar "iris" na se ç ão "service" e obter a porta, mapeada para a porta interna 52773.

Viva!

O mais maneiro é que agora o Docker tem o modo para docker-compose.yml em que você pode não configurar nenhuma porta no Docker. Você pode deixar o símbolo "-" na porta mapeada, e o Docker usará uma porta disponível aleatoriamente.

```
iris:
    build:
      context: .
      dockerfile: Dockerfile-zpm
    restart: always
    ports:
      - 51773
      - 52773
      - 53773
    volumes:
      - ~/iris.key:/usr/irissys/mgr/iris.key
      - ./:/irisdev/app
```

Viva de novo! Agora, você tem uma op ç ão para não se preocupar mais com as portas do servidor web da IRIS às quais o VSCode se conecta.

Como funciona nesse caso: executamos o docker-compose.yml, o Docker escolhe uma porta aleatória do servidor web e executa a IRIS com ela, o VSCode obtém essa porta pelo Docker e conecta-se à IRIS por ela, e você pode editar e compilar o código imediatamente. Sem nenhuma configura ç ão adicional.

Não tem nada melhor que isso!

E você pode fazer o mesmo com o seguinte modelo que enviei recentemente de acordo com o novo recurso do VSCode ObjectScript 0.8, que atualizou o settings.json e o docker-compose.yml. Para testar o modelo, execute os comandos abaixo no terminal (testados no Mac). Você também precisa do git e do Docker Desktop instalados.

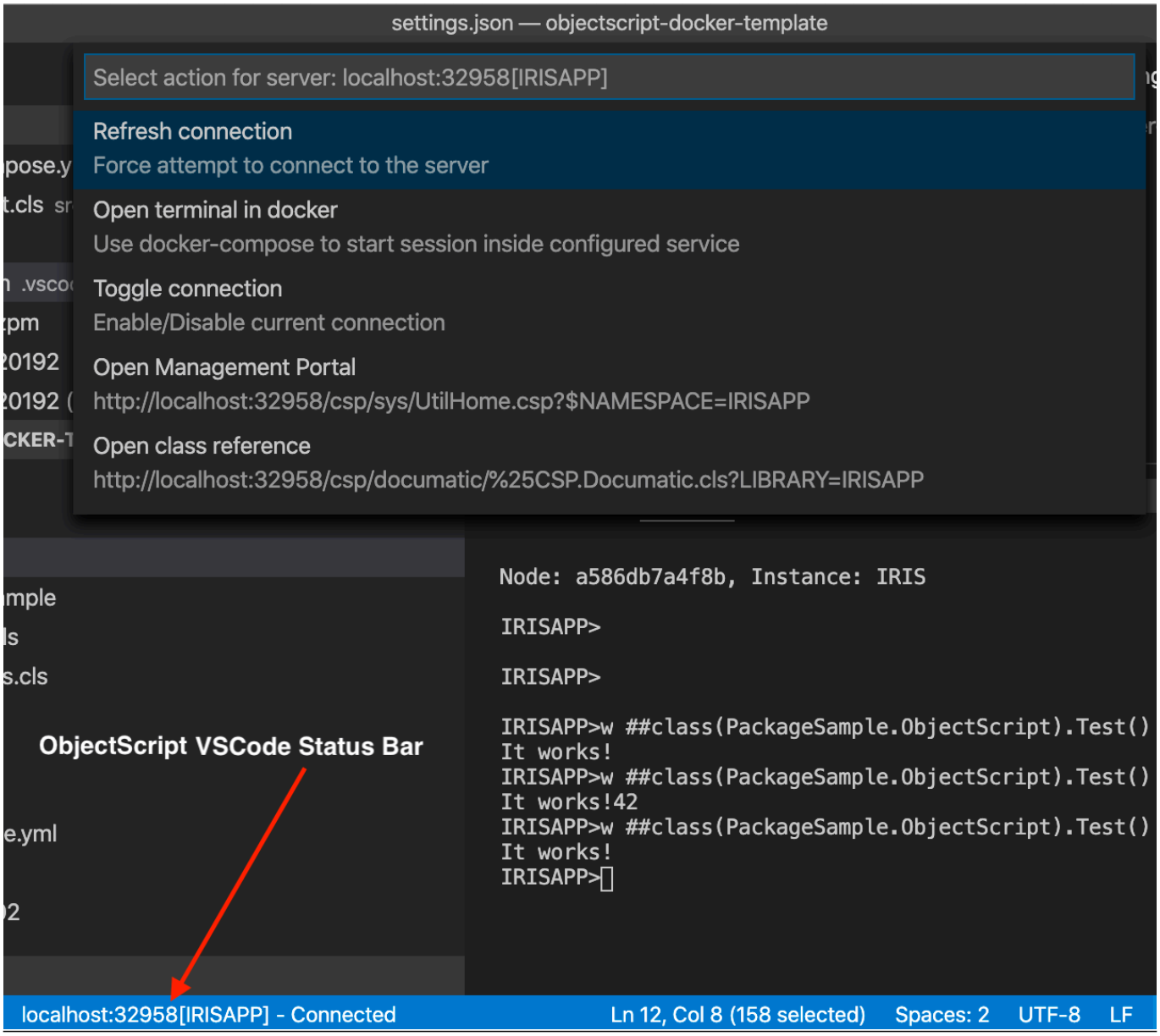
```
$ git clone https://github.com/intersystems-community/objectscript-docker-template.git

$ cd objectscript-docker-template

$ docker-compose up -d
```

Abra essa pasta no VSCode (é preciso ter o plugin VScode ObjectScript instalado):

Verifique se o VSCode está conectado – clique na linha de status do VSCode:



Reconecte o VSCode, se necessário.

Você pode abrir o IRIS Terminal no VSCode, se necessário, com o menu ObjectScript.

Concluído! Agora, você pode executar, compilar e depurar o código!

Viva a programa ç ão!

#Docker #ObjectScript #VSCode #InterSystems IRIS #Open Exchange
Confira o aplicativo relacionado no InterSystems Open Exchange

URL de origem: https://pt.community.intersystems.com/post/trabalhando-com-v%C3%A1rios-projetos-objectscript-simult%C3%A2neos-utilizando-vscode-e-docker-no