Artigo
Heloisa Paiva · Set. 6, 2022 4min de leitura

Interoperabilidade - passo a passo para criar e conectar business hosts

Recentemente comecei a estudar interoperabilidade e achei a documentação oficial muito útil para entender como funciona, mas ainda tive dificuldade para implementar sozinha. Com ajuda dos meus colegas de trabalho, consegui criar uma Demo de um sistema e aprender na prática. Por isso, decidi escrever esse post para ajudar outros a "colocar a mão na massa" e passar adiante a ajuda que tive.

Introdução

Primeiramente, vamos relembrar alguns conceitos:

- Interoperabilidade - o significado da palavra não é tão difícil quanto sua pronúncia - é basicamente o nome para a "magia" que recebe e entrega todo tipo de informação de um sistema a outro.
- Business hosts - se a interoperabilidade é a magia, os business hosts são a cartola do mágico - há os business services, que reconhecem e recebem informação, e as enviam como mensagens bem estruturadas para business processes ou business operations. Os business operations realizam as operações desejadas (como sugere o nome) e entregam a mensagem. Os business processes controlam o fluxo das mensagens: definem onde elas vão e como são passadas.
- Adaptadores - os adaptadores são classes que podemos usar para reconhecer e manipular todo tipo de informação que tenhamos que lidar. Na prática, colocamos os as classes como parâmetros e propriedades para acessar seus métodos e propriedades.

Preparando para começar a criar a produção

É mais fácil começar de maneira simples - vamos utilizar apenas Services e Operations de início - digamos que haja um Service que recebe um tipo de mensagem facilmente reconhecida pela única Operation com a qual estamos trabalhando.

O desenvolvimento é mais fácil quando o propósito da produção e suas partes estão muito claras. Se quiser, pode ser de grande ajuda desenhar um diagrama ou escrever os passos que deseja que o código realize.

Por exemplo:

Comece perguntando "o que tenho que fazer?" - na minha Demo, precisei manipular uma tabela SQL - dou informação com o Título e Autor de um livro e insiro numa tabela.

"Então, o que preciso que a produção faça?" - ela deve receber uma mensagem contendo o Título e o Autor e fazer um SQL INSERT

"Ok, e como isso acontece?" - o Business Service (BS) vai receber o Título e o Autor e passar para o Business Operation (BO). O BO implementa o código SQL.

"Agora que tenho os caminhos que a informação vai seguir, preciso entender a mensagem. O que ela é?" - tem muitas formas pelas quais posso enviar os dados. Posso colocá-los num arquivo, enviar numa aplicação REST, ou até num email. Vamos utilizar o arquivo para começar da maneira mais simples. Meu BS receberá o arquivo, lerá e enviará suas informações ao BO, que realizará a query.

Começando o trabalho

You can start by the part you're most confident in coding.

Você pode começar pela parte que se sente mais confiante para "codar".

- O Business Service

```
Class Demo.Books.BS.FileService Extends Ens.BusinessService
{

Parameter ADAPTER = "EnsLib.File.InboundAdapter";

Method OnProcessInput(pInput As %Stream.Object, Output pOutput As %RegisteredObject) As %Status
  {
      Set tSC = $$$OK

      Try
      {
          // Reads the first line of the recieved File
          Set tLine = pInput.ReadLine()

          // Sets the properties of the request based on the data recieved
          Set tRequest = ##class(Demo.Books.BO.Register.Request).%New()
          Set tRequest.Title = $PIECE(tLine, ",", 1)
          Set tRequest.Author = $PIECE(tLine, ",", 2)

          // Sends to operation
          Set tSC = ..SendRequestSync("Demo.Books.BO.Operation", tRequest, .tResponse)

          Throw:$$$ISERR(tSC)
      }
      Catch (tEx)
      {
          Set:'$$$ISERR(tSC) tSC = tEx.AsStatus()
      }

      Quit tSC
  }

}
```

Eu comecei pelo BS. Agora, fica claro para mim que preciso implementar a classe de Request para que guarde os dados e a Operation, para onde vão os dados.

- O Business Operation

Request

```
Class Demo.Books.BO.Register.Request Extends (Ens.Request, %XML.Adaptor)
{

Parameter RESPONSECLASSNAME = "Ens.Response";

Property Title As %String;

Property Author As %String;

}
```

A classe de Request será simples assim; apenas vou utilizá-la para guardar os dados. O %XML.Adaptor é para que a Request fique visível facilmente no Portal de Administração para lidar com erros.

Operation

```
Class Demo.Books.BO.Operation Extends Ens.BusinessOperation
{

Property Adapter As EnsLib.SQL.OutboundAdapter;

Parameter ADAPTER = "EnsLib.SQL.OutboundAdapter";

Parameter INVOCATION = "Queue";

Method Register(pRequest As Demo.Books.BO.Register.Request, Output pResponse As Ens.Response) As %Status
 {
     // I could implement the method here, but to be more organized I created an Abstract Class for it.
     Quit ##class(Demo.Books.BO.Register.Method).Execute(##this, pRequest, .pResponse)
 }

// This map recognizes the type of message was sent in the request and executes the method specified.

XData MessageMap
 {
<MapItems>
 <MapItem MessageType = "Demo.Books.BO.Register.Request">
   <Method>Register</Method>
  </MapItem>
 </MapItems>
 }

}
```

O BO tem um mapa de mensagens para dar o encaminhamento adequado para cada tipo de mensagem que recebe. Por exemplo, se no BS, no método SendRequestSync, no argumento Request, eu tivesse utilizado um tipo diferente, digamos "Demo.Books.BO.SearchTable.Request", eu poderia criar um outro MapItem com esse tipo de mensagem no MessageType, referenciando um método Search em <Method>.

Method

```
Class Demo.Books.BO.Register.Method [ Abstract ]
{

ClassMethod Execute(pHost As Demo.Books.BO.Operation, pRequest As Demo.Books.BO.Register.Request, Output pResponse As Ens.Response) As %Status
  {
      Set tSC = $$$OK

      Try
      {
          Set tSC = pRequest.NewResponse(.pResponse)

          Throw:$$$ISERR(tSC)

          Set tSC = pHost.Adapter.ExecuteUpdate(.pRow, "INSERT books (title, author) VALUES (?,?)", pRequest.Title, pRequest.Author)

          Throw:$$$ISERR(tSC)
      }
      Catch (tEx)
      {
          Set:'$$$ISERR(tSC) tSC = tEx.AsStatus()
      }

      Quit tSC
  }

}
```

Aqui, implementamos tudo que a Operation deve fazer. É melhor implementar o método em uma outra classe, porque após recompilar o BO, talvez seja necessário recomeçar a produção para trabalhar com as novas mudanças.

Configurações no Portal de Administração

Finalmente, para que tudo funcione, siga o caminho:

Portal de Administração > Interoperabilidade > Lista > Produções > Novo

O portal criará uma classe com as informações da Produção·

Então, você pode adicionar um Service com a classe criada e estabelecer o caminho do Arquivo (onde vão os arquivos com inputs) e o caminho de trabalho.

Também pode adicionar uma Operation com a classe criada e especificar suas configurações, se necessário.

Observações

- O Business Operation poderia receber uma Synchronous Request (requisição síncrona), o que significaria que o Service só responderia depois que o BO já houvesse retornado sua mensagem, porque a resposta do Service depende da resposta do BO, por exemplo, se tivesse que realizar uma operação de pesquisa na tabela. Como a produção só faz uma operação de INSERT, o BS só necessita enviar a informação e o BO só precisa inseri-la; não é necessária uma resposta, então poderíamos utilizar uma Asynchronous Request (requisição assíncrona).
- Não é o escopo deste post discutir especificações de adaptadores como os de File ou SQL, esse artigo foca em dar apenas uma visão geral do desenvolvimento e passos para compreender melhor a prática.

- Sinta-se à vontade para entrar em contato - adoraria ajudar em qualquer maneira possível!

#Innovatium #Interoperabilidade #InterSystems IRIS

URL de origem:https://pt.community.intersystems.com/post/interoperabilidade-passo-passo-para-criar-e-conectar-business-hosts