Article
Yuri Marx · Nov 10, 2020 5m read

Creating a custom interoperability business service using HTTP Adapter

The productions are components developed with InterSystems IRIS Interoperability module to promote integrations between systems, data sources/targets, web services, API, message channels, etc. Productions are composed by:

1) Business services to get/ingest/get data events or requests;

2) Business operations to send or persist data to repositories, systems, API, web services, etc; and

3) BPL - BPEL flows to orchestrate, mediate, compose and route data from business services to business operations.

In this article I will show you how to create a custom business service to get a file from a multipart HTTP request and enable it into a production using class comments, see:

```
Class dc.upload.UploadService Extends Ens.BusinessService
{ //
extends Ens.BusinessService to create a custom Business service using Object S
cript
  //
This class receive a file from a multipart http request and save to the folder config
ured
// into folder parameter
  //Choose an adapter to get data from a source of data
//HTTP.InboundAdapter allows you get data from an http request
Parameter ADAPTER = "EnsLib.HTTP.InboundAdapter";
  //
custom parameter to allows production user set destination folder to multipart file
uploaded
Property Folder As %String(MAXLEN = 100);
  //
when you set parameter Folder to SETTINGS parameter, the production IRIS inter
face create a field
//to the user fills
//so the user will inform host path for the uploaded file
Parameter SETTINGS = "Folder,Basic";
  //
This method is mandatory to have a business service. It receives the multipart file
```

```
into pInput
//and returns a result to the caller using pOutput
Method OnProcessInput(pInput As %GlobalBinaryStream, pOutput As
%RegisteredObject) As %Status
{
    //try to do the actions
    try {
        Set reader = ##class(%Net.MIMEReader).%New()
//creates a MIMEReader to extract files from
//multipart requests
        Do reader.OpenStream(pInput) //reader open the file
          Set tSC = reader.ReadMIMEMessage(.message)
//the reader put the file uploaded into a MIME Message

//Get Header obtains headers from the request and the multipart file, like content-
type
//or content disposition
        //the content disposition have 3 headers:
// Content-Disposition: form-data; name="file"; filename="filename.ext"
        //This split content-disposition header into 3 parts
        Set filenameHeader = $PIECE(message.GetHeader("CONTENT-
DISPOSITION", .header),";",3)
        //get filename header value
        Set filename = $EXTRACT(filenameHeader, 12, $LENGTH(
filenameHeader)-1)

//Headers are not more needed. It clean the header to remains only the file conten
t to be saved
        Do message.ClearHeaders()
          //create a file object to save the multipart file
        Set file=##class(%Stream.FileBinary).%New()

//points the file to folder informed into folder parameter, plus upload filename from
header
        Set file.Filename=..Folder_filename
        //save body message (the file content) to file object
        Do file.CopyFromAndSave(message.Body)

        //return a sucess message to the user/caller
```

```
        Set pOutput = "File "_filename_
" uploaded with success to: ".. Folder_filename
        Set tSC=$$$OK

    //returns error message to the user
    } catch e {
        Set tSC=e.AsStatus()
        Set pOutput = tSC
    }
      Quit tSC
}
  }
```

I created an application into the Open Exchange to show you this business class. To test this code, use it. Follow the steps:

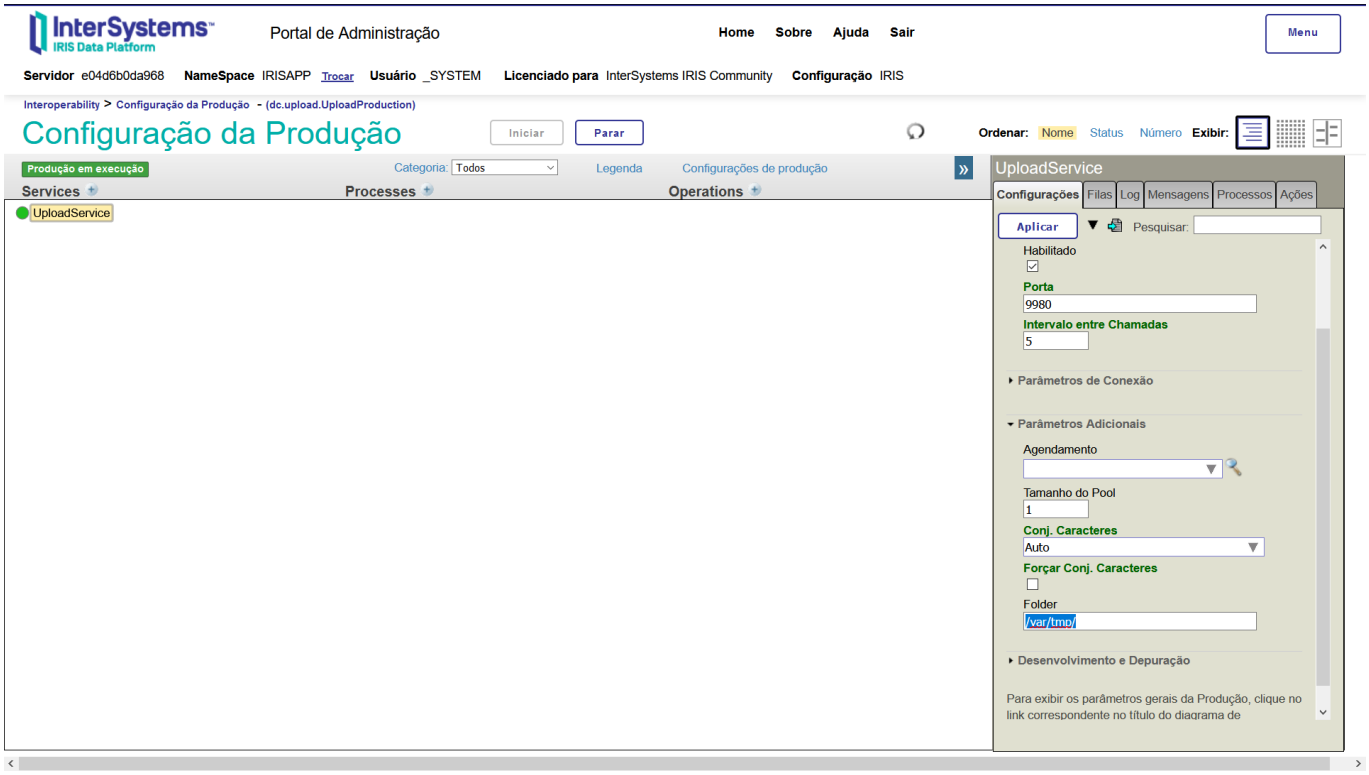
1) Go to: https://openexchange.intersystems.com/package/upload-adapter

2) Into your terminal/cmd execute: *git clone* https://github.com/yurimarx/upload-adapter.git

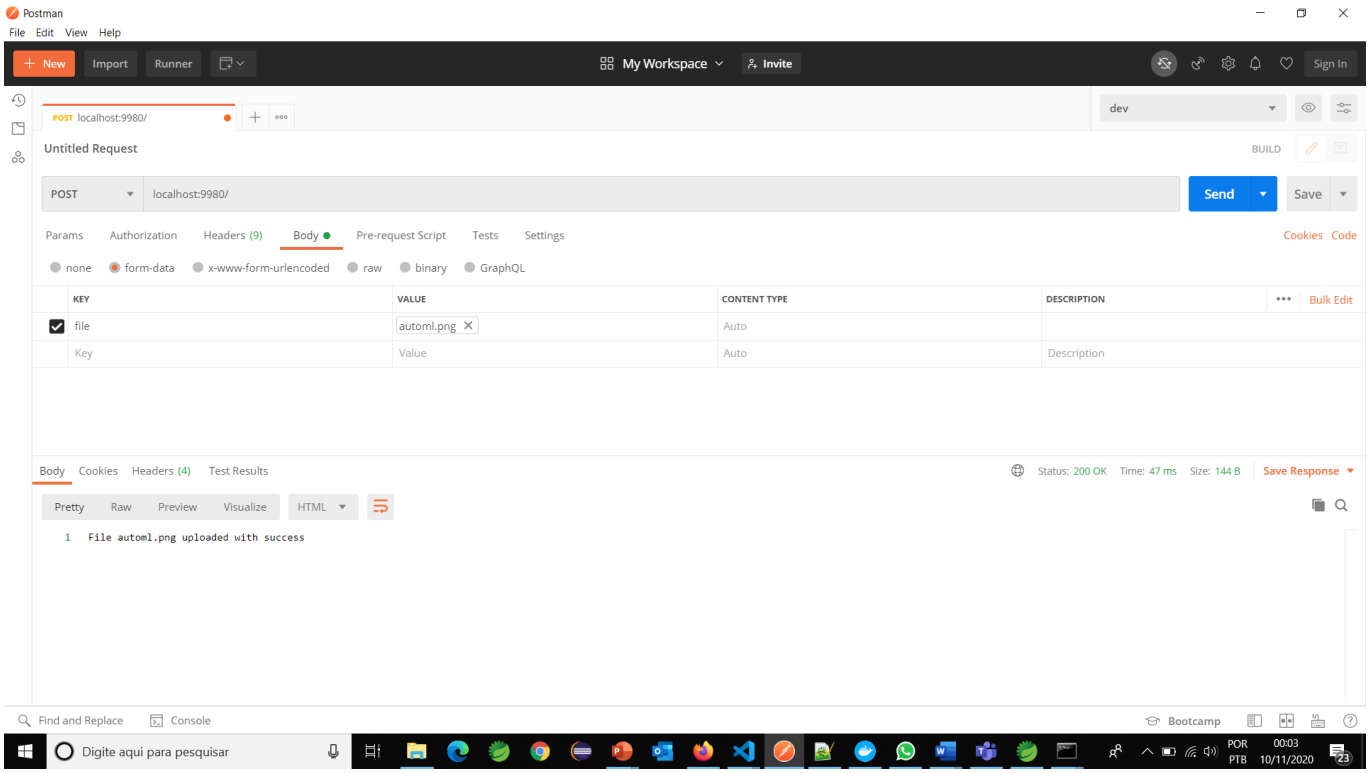
3) Into your terminal/cmd execute: *docker-compose build* and after execute *docker-compose up -d*

4) Open the production

5) Set the host destination folder to the uploaded files and start the production. See:



6) Now Open Postman or create a multipart request into a form pointing to localhost:9980/ using POST with a form-data file attribute. See sample:



7) Check the upload into your IRIS docker instance.

Now, with this business service, you have alternatives to send a file to your docker instance using postman or a html form and compose this feature with BPL and business operations. Fantastic!

PS: a loved ObjectScript MIMEReader and MIME* classes.

#Interoperability #InterSystems IRIS

Source URL:https://community.intersystems.com/post/creating-custom-interoperability-business-service-using-http-adapter