Article
Yuri Marx · Nov 18, 2020 7m read

Open Exchange

Creating a PEX Business Operation

PEX is a new InterSystems IRIS feature to allows extends IRIS with existent features from Java or .NET.

It is possible create interoperability inbound and outbound adapters, business services (integrate from external to internal) and operations (integrate internal to external).

To create a PEX component it is necessary import .NET (InterSystems.EnsLib.PEX.*) or Java (com.intersystems.enslib.pex.*) packages and extends or implements the properly class.

In this article I will use my OCR Service. It has a PEX business Operation, see the code:

```
public class OcrOperation extends BusinessOperation {
    // Connection to InterSystems IRIS
  private IRIS iris;
    @Override
  public void OnInit() throws Exception {
     iris = GatewayContext.getIRIS();
  }

  @Override
  public Object OnMessage(Object request) throws Exception {
     IRISObject req = (IRISObject) request;
     String filePath = req.getString("FileName");
     String ocrText = doOcr(filePath);
     IRISObject response = (IRISObject)(iris.
classMethodObject("Ens.StringContainer","%New",ocrText));
     return response;
  }

  public String doOcr(String filePath) {
       File tempFile = new File(filePath);
       String ocrText = "";
       try {

        if (tempFile.toString().contains(".pdf")) {
           ocrText = extractTextFromPDF(tempFile);
        } else {
```

```
            ocrText = extractTextFromImage(tempFile);
          }
           return ocrText;
        } catch (IllegalStateException | IOException | TesseractException e) {
          return e.getMessage();
        }
      }
      private String extractTextFromPDF(File tempFile) throws
IOException, TesseractException {
         String ocrText = "";
         // Load file into PDFBox class
        PDDocument document;
        document = PDDocument.load(tempFile);
         // Extract images from file
        PDFRenderer pdfRenderer = new PDFRenderer(document);
        StringBuilder out = new StringBuilder();
         ITesseract tesseract = new Tesseract();
        tesseract.setDatapath("/usr/share/tessdata/"); //directory to trained models
        tesseract.setLanguage("por"); // choose your language/trained model
         for (int page = 0; page < document.getNumberOfPages(); page++) {
          BufferedImage bim = pdfRenderer.
renderImageWithDPI(page, 300, ImageType.RGB);
           // Create a temp image file
          File temp = File.createTempFile("tempfile_  + page, ".png");
          ImageIO.write(bim, "png", temp);
           String result = tesseract.doOCR(temp);
          out.append(result);
           // Delete temp file
          Files.delete(temp.toPath());
           ocrText = out.toString();
        }
         return ocrText;
      }
      private String extractTextFromImage(File tempFile) throws
TesseractException {
         ITesseract tesseract = new Tesseract();
        tesseract.setDatapath("/usr/share/tessdata/"); //directory to trained models
        tesseract.setLanguage("eng+por"); // choose your language/trained model
         return tesseract.doOCR(tempFile); //call tesseract function doOCR()
```

```
                        //passing the file to be processed with OCR technique
    }
    @Override
   public void OnTearDown() throws Exception {
      // TODO Auto-generated method stub

   }
```

For operations you implement OnInit to initiate Java connection with IRIS and initiate other Java resources, if necessary, and OnTearDown to release resources allocated. Finally you need implement OnMessage, with your source code to the PEX operation, including the code thats uses Java frameworks to implement something, in my case de logic to extract text from images using Tess4J framework. This framework uses Google Tesseract to OCR.

To Java connection works, it is necessary configure a Java Gateway into the production with the business operation, to allows create a Java proxy context. See my production sample:

```
Class dc.ocr.OcrProduction Extends Ens.Production
{
 XData ProductionDefinition
{
<Production Name="dc.ocr.OcrProduction" LogGeneralTraceEvents="false">
 <Description></Description>
 <ActorPoolSize>2</ActorPoolSize>
 <Item Name="OcrService" Category="" ClassName=
"dc.ocr.OcrService" PoolSize="1" Enabled="true" Foreground="false"
Comment="" LogTraceEvents="false" Schedule="">
 </Item>
 <Item Name="JavaGateway" Category="" ClassName=
"EnsLib.JavaGateway.Service" PoolSize="1" Enabled="true"
Foreground="false" Comment="" LogTraceEvents="false" Schedule="">
   <Setting Target="Host" Name="ClassPath">
.:/usr/irissys/dev/java/lib/JDK18/*:/opt/irisapp/*:/usr/irissys/dev/java/lib/gson/*:/usr/i
rissys/dev/java/lib/jackson/*:/jgw/ocr-pex-1.0.0.jar</Setting>
   <Setting Target="Host" Name="JavaHome">
/usr/lib/jvm/java-8-openjdk-amd64/</Setting>
 </Item>
 <Item Name="OcrOperation" Category="" ClassName=
"EnsLib.PEX.BusinessOperation" PoolSize="1" Enabled="true"
Foreground="false" Comment="" LogTraceEvents="false" Schedule="">
   <Setting Target="Host" Name="%gatewayPort">55555</Setting>
   <Setting Target="Host" Name="%remoteClassname">
community.intersystems.pex.ocr.OcrOperation</Setting>
   <Setting Target="Host" Name=
```

```
"%gatewayExtraClasspaths">
.:/usr/irissys/dev/java/lib/JDK18/*:/opt/irisapp/*:/usr/irissys/dev/java/lib/gson/*:/usr/irissys/dev/java/lib/jackson/*:/jgw/ocr-pex-1.0.0.jar</Setting>
  </Item>
</Production>
}
  }
```

It is important config the classpath to the Java JAR used by your Java PEX app, and set the correct port and Java PEX class name.

See all details into github to my OCR Service. It can be used as a start point to your PEX app.

https://openexchange.intersystems.com/package/OCR-Service

#Interoperability #Java #InterSystems IRIS
Check the related application on InterSystems Open Exchange

Source URL:https://community.intersystems.com/post/creating-pex-business-operation