Jenkins pipeline – Pasar de un archivo XML a un archivo Java.

Para esta POC el objetivo era tener una pipe de Jenkins que a partir de pasarle un archivo .XML como input, este nos lo traduzca a Java y que una vez lo tenga suba ambos archivos a GitHub en sus respectivas carpetas (xml y java).

Para esto, explicare el proceso a seguir. Primero de todo, creamos el repositorio donde estará toda la información, al cual subiremos los archivos nuevos. En este crearemos distintas carpetas, yo he creado una de "tools" para poner los scripts y otras herramientas que vaya a utilizar (por ejemplo el script para pasar de XML a Java). Y además otra carpeta "resources" que dentro tendrá una carpeta para los .xml de entrada y otra para los .java convertidos. (Al final del documento incluire el código tanto de la pipe como del script para pasar de XML a Java).

En la pipe de Jenkins, primero de todo crearemos un parámetro que será el input file. Y el primer Stage será validar este archivo de entrada, comprobaremos tanto la extensión (que sea .xml) y que no sea un archivo vacio. Si ninguno se cumple, será un archivo valido para la pipe. En caso contrario, fallara la pipe.

El segundo paso será crear un directorio en Jenkins donde se guardara el resultado de la pipe, lo llamaremos repo-temp.

Seguidamente, el tercer paso será clonar el repositorio donde tenemos todos los datos, lo clonaremos dentro del directorio de Jenkins que acabamos de crear.

Despues, el siguiente paso es procesar el archivo XML, para pasarlo a Java. Para esto entramos al directorio "repo-temp/tools" donde tenemos el script que convertirá los archivos. Mediante comandos de powershell, lo le pasamos los argumentos necesarios, y ejecutamos el script (le pasamos el input File, la ruta de salida del archivo Java, y la ruta donde queremos mover el input File). Entonces el script genera el nuevo archivo y a la vez mueve los dos archivos a sus correspondientes directorios dentro de "resources".

Una vez se ha realizado todo esto, lo que tenemos que hacer es pushear los cambios al repositorio. Para esto crearemos un nuevo stage donde haremos un git add de las carpetas de resources, donde estarán los nuevos archivos .XML y .Java.

Ademas como ultimo paso tenemos una ant task que crea un build de un microservicio de ejemplo que tengo en el repositorio, simplemente para aprender.

Cuando todo esto se ejecuta correctamente, finaliza la pipe con éxito.

Ahora mostrare los scripts tanto de la pipe como del File Converter. Para leer
Para leerlo con mayor facilidad, podéis entrar al repositorio que he utilizado, donde se puede leer todos los archivos dentro de /tools:
GitHub - RubenMailloBaena/JenkinsPipelineTest

JENKINS CODE:

```
pipeline {
  agent any

  parameters{
    base64File 'inputFile'
  }

  environment {
    REPO_URL = 'https://github.com/RubenMailloBaena/JenkinsPipelineTest.git'
    PUSH_URL = '@github.com/RubenMailloBaena/JenkinsPipelineTest.git'
    BRANCH = 'main'
    USERNAME = 'RubenMailloBaena'
    EMAIL = 'rubenmaillo2003@gmail.com'
    ANT_DIR = 'repo-temp/tools'
    XML_DIR = '/repo-temp/resources/XML'
    JAVA_DIR = '/repo-temp/resources/Java'
    VALID_FORMATS = '.txt,.xml'
  }

  stages {
    stage('Validating Input File'){
      steps{
        withFileParameter('inputFile'){
          powershell '''
            $input = "$env:inputFile"
            $length = (Get-Item $input).Length
            $filename = "$env:inputFile_FILENAME"
            $ext = [System.IO.Path]::GetExtension($filename).ToLower()

            if($length -eq 0){
              Write-Host "[ERROR]: The file content can't be empty: $input"
              exit 1
            }
```

```
            $validExts = "$env:VALID_FORMATS".Split(',')
            if(-not $validExts.Contains($ext)){
              Write-Host "[ERROR]: Invalid file format: $ext --> allowed formats: $validExts"
              exit 1
            }

            Write-Host "Input File Validated!"
          '''
        }
      }
    }

    stage('Prepare Workspace') {
      steps {
        script {
          powershell '''
            if (Test-Path -Path repo-temp) {
              Remove-Item -Path repo-temp -Recurse -Force
            }
            mkdir repo-temp
          '''
        }
      }
    }

    stage('CLonar Repositorio') {
      steps {
        dir('repo-temp'){
          git branch: "${env.BRANCH}", url: "${env.REPO_URL}"
        }
      }
    }
```

```
    stage('Process Input File'){
      steps{
        withFileParameter('inputFile'){
          dir ('repo-temp/tools'){
            script{
              powershell '''
                $filename = "$env:inputFile_FILENAME"
                $inputFilePath = "$env:inputFile"

                $extension = [System.IO.Path]::GetExtension($filename).ToLower()

                if($extension -eq ".xml"){ #CONVERTIMOS EL XML EN JAVA
                  $inputFileDestination = Join-Path -Path "$env:WORKSPACE$env:XML_DIR" -ChildPath
"$filename"

                  $filenameWithoutExt = [System.IO.Path]::GetFileNameWithoutExtension($filename)
                  $javaOutputFile = Join-Path -Path "$env:WORKSPACE$env:JAVA_DIR" -ChildPath
"$filenameWithoutExt.java"

                  javac XMLToJavaConverter.java
                  java XMLToJavaConverter $inputFilePath $javaOutputFile $inputFileDestination

                  if($LASTEXITCODE -ne 0){
                    Write-Host "[ERROR]: Java conversion failed. Exiting"
                  }
                  else{
                    Write-Host "[INFO]: Input File: $filename"
                    Write-Host "[INFO]: Output Java File: $filenameWithoutExt.java"
                    Write-Host "[INFO]: Converted in Path: $javaOutputFile"
                    Write-Host "[INFO]: Input XML moved to Path: $inputFileDestination"
                  }
                }
                else{ #MOSTRAMOS EL CONTENIDO DEL TXT EN CONSOLA
                  Write-Host "Input file is a .txt, printing $filename content:"
                  Get-Content $inputFilePath
```

```
                }
                '''
            }
        }
    }
}

stage('Push Files to Repository'){
    steps{
        dir('repo-temp'){
            withCredentials([string(credentialsId: 'github_push', variable: 'GITHUB_TOKEN')]){
                script{
                    powershell '''
                        #EVITAR ADVERTENCIAS DE LF/CRLF
                        git config core.autocrlf true

                        git config user.name "$env:USERNAME"
                        git config user.email "$env:EMAIL"

                        git add resources/XML/*.xml
                        git add resources/Java/*.java

                        $env:GIT_URL = "https://${env:GITHUB_TOKEN}$env:PUSH_URL"

                        git commit -m "Add input XML and converted Java files"
                        git push --set-upstream $env:GIT_URL $env:BRANCH
                    '''
                }
            }
        }
    }
}
```

```
    stage('Compilar con Ant task'){
      steps{
        withFileParameter('inputFile'){
          dir(env.ANT_DIR){
            script{
              def antHome = tool name: 'Ant Installation', type: 'hudson.tasks.Ant$AntInstallation'
              def mavenHome = tool name: 'Maven Installation', type: 'hudson.tasks.Maven$MavenInstallation'
              def mvnCmd = "${mavenHome}\\bin\\mvn.cmd"

              bat """
                set INPUT_FILE=${inputFile}
                set MVN_CMD=${mvnCmd}
                "${antHome}\\bin\\ant.bat" -Dinput.file=%INPUT_FILE% -Dmvn.cmd=%MVN_CMD%
              """
            }
          }
        }
      }
    }
  }
}
```

FILE CONVERTER CODE:

```
import org.w3c.dom.*;


import java.io.File;

import java.io.FileWriter;


import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;


//Documentation: https://zetcode.com/java/dom/


public class XMLToJavaConverter{


  //EL STRING DE INPUT DEBE ESTAR SIEMPRE EN PRIMERA POSICION PARA VALIDAR QUE TIENE UNA REGION DE INPUT OBLIGATORIA

  private static String[] permittedLabels = {"input", "output"};


  //pos 1. Nombre del atributo que indica el tipo de varaible

  //pos 2. ---

  private static String[] permittedAttributes = {"varType"};


  //Escribir en Lower Case

  private static String[] permitedVarTypes = {"string", "boolean", "integer", "float", "double"};


  private static boolean hasInputRegion = false;

  private static boolean lastTypeWasString = false;


  private static final int MAX_VARIABLE_ATTRIBUTES = 1;

  private static final int MAX_VARIABLE_CHILDS = 1;


  //en el arg0 entra el XML y en el arg1 el path de salida del .java y el arg2 donde queremos mover el input XML

  public static void main(String[] args){


    if (args.length < 3) {

      System.err.println("Usage: XMLToJavaConverter <input XML path> <output Java path> <move XML path>");

      System.exit(1);

    }
```

```
    try{
        String inputFilePath = args[0];
        File xmlFile = new File(inputFilePath);

        //Cargamos el XML usando DOM
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document document = builder.parse(xmlFile);

        //Creamos el archivo Java
        document.getDocumentElement().normalize();
        String javaClassContent = generateJavaClass(document.getDocumentElement());

        //Escribimos el contenido Java a un archivo nuevo
        String outputJavaFile = args[1];
        FileWriter writer = new FileWriter(outputJavaFile);
        writer.write(javaClassContent);
        writer.close();

        //Movemos el archivo XML a su carpeta correspondiente
        xmlFile.renameTo(new File(args[2]));
    }
    catch(Exception e){
        e.printStackTrace();
        System.exit(1);
    }
}

// Método para generar la clase Java a partir del XML
private static String generateJavaClass(Element rootElement) throws InvalidXMLException{

    StringBuilder result = new StringBuilder();

    //CREATE CLASS
    String className = rootElement.getNodeName();
```

```
result.append("public class " + UpperFirstLetter(className) + "{\n\n");

NodeList regionsList = rootElement.getChildNodes();
NodeList variablesList;
String variablesResult = "";
String functionsResult = "";

//BUCLE DE REGIONES (INPUT / OUTPUT)
for(int i=0; i<regionsList.getLength(); i++){

  Node node = regionsList.item(i);

  if(isValidRegion(node.getNodeName())){

    variablesList = node.getChildNodes();

    //BUCLE DE VARIABLES (PARAM1 / RESULT2)
    for(int j=0; j<variablesList.getLength(); j++){

      node = variablesList.item(j);

      if(isValidVariable(node)){

        //VALIDAMOS QUE EL EL ATRIBUTO Y LA VARAIBLE SON CORRECTOS; Y MAPEAMOS EL NOMBRE
        String varType = mapVariableTypes(isValidVariableType(node));
        String nodeName = node.getNodeName();

        //attribute (ej. private String param1)
        variablesResult += "\tprivate " + varType + " " + nodeName;

        //attribute value (ej. = "Valor1";) Si es String añadimos comillas.
        if(lastTypeWasString)
          variablesResult += " = \"" + node.getTextContent().replace(" ","") + "\";\n";
        else
          variablesResult += " = " + node.getTextContent().replace(" ","") + ";\n";
```

```
                //getter
                functionsResult += "\n\tpublic " + varType + " get" + UpperFirstLetter(nodeName) + "(){\n"
                + "\t\treturn " + nodeName + ";\n"
                + "\t}\n";

                //setter
                functionsResult += "\n\tpublic void set" + UpperFirstLetter(nodeName) + "(" + varType + " " + nodeName +"){\n"
                + "\t\tthis." + nodeName + " = " + nodeName + ";\n"
                + "\t}\n\n";
            }
        }
    }
  }

  //COMPROBAMOS QUE TENEMOS EL LABEL DE INPUT NECESARIO
  checkIfHasInputField();

  result.append(variablesResult);
  result.append(functionsResult);
  result.append("}");
  return result.toString();
}

//VALIDAMOS QUE TENEMOS LA REGION INPUT
private static void checkIfHasInputField() throws InvalidXMLException{
  if(!hasInputRegion)
    throw new InvalidXMLException("[ERROR]: The input .xml does not contain an '" + permittedLabels[0] + "' region/field");
}

//VALIDAMOS QUE LA REGION ESTE BIEN ESCRITA Y SI ES EL INPUT, QUE ES OBLIGATORIO
private static boolean isValidRegion(String nodeName) throws InvalidXMLException{
  if(Character.isLetter(nodeName.charAt(0))){
    if(!isPermitted(nodeName, permittedLabels, true, true)) //ESTA DENTRO DE LOS NOMBRES PERMITIDOS
      throw new InvalidXMLException("[ERROR]: Contains an invalid region name --> " + nodeName);
    return true;
  }
```

```
    return false;
  }

  //VALIDAMOS QUE LA VARAIBLE Y ATRIBUTO DEL XML SEA CORRECTO
  private static boolean isValidVariable(Node node) throws InvalidXMLException{
    if(Character.isLetter(node.getNodeName().charAt(0))){

      if(node.getChildNodes().getLength() != MAX_VARIABLE_CHILDS) //VALIDAMOS QUE LAS VARIABLES SOLO TENGAN UN SOLO HIJO
        throw new InvalidXMLException("[ERROR]: Variables can't have more than " + MAX_VARIABLE_CHILDS +" childs or be empty! --> " + node.getNodeName());

      if(node.getAttributes().getLength() != MAX_VARIABLE_ATTRIBUTES) //VALIDA QUE SOLO TENGAMOS UN ATTRIBUTO
        throw new InvalidXMLException("[ERROR]: Variables can't have more than " + MAX_VARIABLE_ATTRIBUTES + " attributes or be empty! --> " + node.getNodeName());

      return true;
    }
    return false;
  }

  //VALIDAMOS QUE ES UN TIPO DE VARIABLE QUE EXISTE
  private static String isValidVariableType(Node node) throws InvalidXMLException{

    String attributeName = node.getAttributes().item(0).getNodeName();

    if(!isPermitted(attributeName, permittedAttributes, true, false)) //ATRIBUTO QUE EXISTE
      throw new InvalidXMLException("[ERROR]: Invalid attribute name: --> " + attributeName);

    String varType = node.getAttributes().getNamedItem(attributeName).getTextContent();

    if(!isPermitted(varType, permitedVarTypes, true, false)) //TIPO DE VARIABLE QUE EXISTE
      throw new InvalidXMLException("[ERROR]: Invalid variable type: --> " + varType);

    return varType.toLowerCase();
  }
```

```
  //VALIDAMOS SI LOS NOMBRES ESTAN DENTRO DE LA COLECCION INDICADA, SI VENIMOS DE LAS REGIONES; PODEMOS VALIDAR SI ESTA EL INPUT OBLIGATORIO

  //Y TENEMOS OTRO BOOL PARA INDICAR SI DEBEMOS ESCRIBIR TODO IGUAL, O SI PODEMOS IGNORAR LAS MAYUSCULAS.

  private static boolean isPermitted(String inputName, String[] permittedCollection, boolean capsSensitive, boolean checkForInputRegion) {


    if (checkForInputRegion && !hasInputRegion) {

      String permittedLabel = permittedCollection[0];

      hasInputRegion = capsSensitive ? permittedLabel.equals(inputName) : permittedLabel.equalsIgnoreCase(inputName);


      if (hasInputRegion)

        return true;

    }


    for (String permitted : permittedCollection) {

      boolean isMatch = capsSensitive ? permitted.equals(inputName) : permitted.equalsIgnoreCase(inputName);


      if (isMatch)

        return true;

    }


    return false;

  }


  //MAPEAMOS LOS NOMBRES DE LOS TIPOS DE VARIABLES (de string a String, de integer a int, etc)

  private static String mapVariableTypes(String varType){


    lastTypeWasString = false;


    switch (varType) {

      case "string":

        lastTypeWasString = true;

        return UpperFirstLetter(varType);


      case "integer":

        return "int";
```

```
            default:
                return varType;
        }
    }

    //PONEMOS EN MAYUSCULA LA PRIMERA LETRA DE LA PALABRA
    private static String UpperFirstLetter(String word){
        if(word.isBlank() || word.isEmpty()) return "";

        String c1 = word.substring(0,1).toUpperCase();
        return c1 + word.substring(1);
    }

    //EXCEPCION PARA XML INVALIDOS
    private static class InvalidXMLException extends Exception{
        public InvalidXMLException(String message){
            super(message);
        }
    }
}
```