

Plataforma XNAT - Pipelines

Instalación de los pipelines de Xnat

<https://wiki.xnat.org/documentation/xnat-administration/configuring-the-pipeline-engine/installing-pipelines-in-xnat>

Prerrequisitos:

- XNat instalado y funcionando
 - en <http://detritus.fundacioace.com:8088/>
 - con el `XNAT_HOME` en `/nas/data/xnat/`
- Los programas Git y Mercurial para descargar código de repositorios.
- El programa `mrimicro` que proporciona la herramienta `dcm2niix` que se usa para convertir DICOM en NIFTI.

Descarga del software

Seguimos el tutorial de arriba que sugiere una forma de instalación desde un directorio temporal, que requiere menos pasos que hacerlo, como indican en otros tutoriales, desde dentro del `XNAT_HOME` (el resultado es equivalente)

Sistema de pipelines

```
$ cd
$ mkdir tmp
$ cd tmp
$ git clone https://github.com/NrgXnat/xnat-pipeline-engine.git
$ cd xnat-pipeline-engine
$ vim gradle.properties
```

```
xnatUrl=http://detritus.fundacioace.com:8088/xnat
siteName=XNAT
adminEmail=osotolongo@fundacioace.com
smtpServer=smtp.fundacioace.com
destination=/nas/data/xnat/pipeline
modulePaths=/nas/daniel/tmp/modules
```

Nota: si no se pone el servidor SMTP correcto, no funcionará la característica de muchos pipelines de enviar un mensaje de correo al usuario diciéndole que un proceso largo ha terminado.

Pipeline que convierte DICOM a NIFTI

```
$ cd
$ cd tmp
$ mkdir modules
$ cd modules
$ hg clone
https://bitbucket.org/nrg_customizations/nrg_pipeline_dicomtonifti
```

Este pipeline usa el dcm2niix para convertir los DICOM en un NIFTI 4D. Por eso el ejecutable dcm2niix debería estar accesible y visible (o sea, en el PATH) del usuario bajo en que se ejecuta XNat. Si no es así, se debe incluir la ruta en el script de configuración:

```
$ cd nrg_pipeline_dicomtonifti/scripts/scripts
$ vim dcm2niix_setup.sh
```

```
#!/bin/bash
```

```
DCM2NIIX_HOME=/nas/software/mricron/
export PATH=$DCM2NIIX_HOME:$PATH
```

Instalación del pipeline

Basta ejecutar desde el directorio xnat-pipeline-engine el programa gradlew (como root):

```
$ cd
$ cd tmp/xnat-pipeline-engine
$ sudo ./gradlew
```

El programa gradlew instala en \$XNAT_HOME/pipeline los módulos/pipelines que estaban en el paquete y otros que son usados desde él, así como varias herramientas para interactuar con XNat desde scripts.

Resultado de la instalación

En \$XNAT_HOME/pipeline se habrán creado varios directorios, entre ellos el catalog, dentro del cual se pueden encontrar los distintos pipelines. En particular, el que interesa para este ejemplo es el dcm2niix.

```
$ ls /nas/data/xnat/pipeline/catalog/
nt-tools          DicomToNifti    FSL_tools       pipeline-tools
birn_tools        Freesurfer      images          utils
commandlineTools freesurfer_tools mricron         validation_tools
dcm2niix          FSL             notifications   xnat_tools
```

Comprobación de la instalación

Accediendo a XNat, en el menú Administer > Pipelines, seleccionar el enlace a “Add pipeline to repository”. En la página que se abre, se debe introducir en el campo “Enter path to pipeline descriptor xml =>” la ruta absoluta a la descripción XML del pipeline que queremos usar:

```
/nas/data/xnat/pipeline/catalog/dcm2nix/DicomToNifti_X.xml
```

Una vez definido el pipeline en XNat, se debe crear un proyecto y, dentro en él, seleccionar los pipelines que se quieren ejecutar sobre las secuencias que se cargarán en él.

Para ello, en el proyecto, se selecciona la pestaña “Pipelines” y se hace click sobre “Add more pipelines”: se debe añadir la línea de la tabla donde ponga “DicomToNifti_X.xml”. En la ventana que se abre, se deben seleccionar las dos opciones de configuración:

```
* Launch pipeline automatically when session is archived
```

Esta opción nos recuerda que el procesamiento se lleva a cabo una vez se cargan las imágenes en el proyecto (o, si se cargan como “pre-archivo”, cuando se selecciona archivar éste).

Para comprobar que la conversión se ha completado, se puede seleccionar “Actions > Download images”. En la página que se carga, se verá en la columna central con los “Scan formats” un apartado “NIFTY”.

Ejemplos de pipelines

A continuación se explica cuál es la estructura de un pipeline y se dan ejemplos de modificación de un pipeline y de construcción de otro.

Estructura de un pipeline

Un *pipeline* se define en un archivo XML dentro de un subdirectorio de `pipeline/catalog` que se crea al instalar en *pipeline engine* (dentro de `/nas/data/xnat`). En el subdirectorio se recomienda la siguiente estructura:

- descripción XML del *pipeline*
- `resources`: subdirectorio con definiciones XML para llamar a programas externos
- `scripts`: subdirectorio con los scripts o programas externos propios del *pipeline*

Ejemplo: el pipeline ...

La descripción del *pipeline* se hace en un documento XML con la siguiente estructura:

Ejemplo: modificación del `//pipeline//` ...

Ejemplo: limpieza de las series en una sesión DICOM

Aquí se describirá el *pipeline* CleanMRSession que lleva a cabo la selección de los scans de una serie DICOM de forma análoga a la descrita en la página [Usar la API de XNAT](#).

Este *pipeline* consta de los siguientes archivos:

- CleanMRSession.xml
- resources/cleanMRSession.xml
- scripts/cleanMRSession.sh (script Bash)

CleanMRSession.xml

El código, completo, es así:

```
<?xml version="1.0" encoding="UTF-8"?>
<Pipeline xmlns="http://nrg.wustl.edu/pipeline"
xmlns:xi="http://www.w3.org/2001/XInclude"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://nrg.wustl.edu/pipeline ..\schema\pipeline.xsd"
xmlns:fileUtils="http://www.xnat.org/java/org.nrg.imagingtools.utils.FileUtils">
  <name>RunCommand</name>
  <location>CleanMRSession</location>
  <description>Leave only valid series in a session</description>

<!-- Standard input parameters -->
  <documentation>
    <authors>
      <author>
        <lastname>Rodriguez-Perez</lastname>
        <firstname>Daniel</firstname>
      </author>
    </authors>
    <version>0</version>
    <input-parameters>
      <parameter>
        <name>scanIDs</name>
        <values><schemalink>xnat:mrSessionData/scans/scan/ID</schemalink></values>
        <description>The scan ids of all the scans of the
session</description>
      </parameter>
      <parameter>
        <name>sessionID</name>
        <values><schemalink>xnat:mrSessionData/ID</schemalink></values>
        <description>Xnat session ID</description>
      </parameter>
      <parameter>
        <name>sessionName</name>
        <values><schemalink>xnat:mrSessionData/label</schemalink></values>
```

```

        <description>Xnat session label</description>
    </parameter>
    <parameter>
        <name>projectID</name>
    <values><schemalink>xnat:mrSessionData/project</schemalink></values>
        <description>Project</description>
    </parameter>
    <parameter>
        <name>subjectID</name>
    <values><schemalink>xnat:mrSessionData/subject_ID</schemalink></values>
        <description>Subject ID</description>
    </parameter>
</input-parameters>
</documentation>

<xnatInfo appliesTo="xnat:mrSessionData"/>

<outputFileNamePrefix>^concat(/Pipeline/parameters/parameter[name='workdir']/values/unique/text(),'/LOG')^</outputFileNamePrefix>

<!-- Pipeline loop -->
    <loop id="series"
    xpath="^/Pipeline/parameters/parameter[name='scanIDs']/values/list^"/>

<!-- Other computed parameters -->
    <parameters>
        <parameter>
            <name>workdir</name>
            <values>
                <unique>^concat(/Pipeline/parameters/parameter[name='builddir']/values/unique/text(),'/',/Pipeline/parameters/parameter[name='sessionName']/values/unique/text())^</unique>
            </values>
        </parameter>
    </parameters>

    <steps>
<!-- Download session series -->
        <step id="MKDIRS" description="Make DICOM folders"
        workdirectory="^/Pipeline/parameters/parameter[name='workdir']/values/unique/text()^">
            <!-- mkdir -p $WORKDIR/RAW/series -->
            <resource name="mkdir" location="commandlineTools" >
                <argument id="p"/>
                <argument id="dirname">
                    <value>^concat(/Pipeline/parameters/parameter[name='workdir']/values/unique/text(),'/RAW/',PIPELINE_LOOPON(series))^</value>
                </argument>
            </resource>
        </step>
        <step id="DOWNLOAD" description="Download scan DICOMs"

```

```
workdirectory="^concat(/Pipeline/parameters/parameter[name='workdir']/values/unique/text(), '/RAW/', PIPELINE_LOOPON(series))^">
    <!-- XnatDataClient -username $USER -password $PASSWORD -
useAbsolutePath -batch -method GET
"$HOST/data/experiments/$SESSIONID/scans/$SERIES/resources/DICOM/files" -->
    <resource name="XnatDataClient" location="xnat_tools">
        <argument id="user">
<value>^/Pipeline/parameters/parameter[name='user']/values/unique/text()^</value>
        </argument>
        <argument id="password">
<value>^/Pipeline/parameters/parameter[name='pwd']/values/unique/text()^</value>
        </argument>
        <argument id="absolutePath"/>
        <argument id="batch"/>
        <argument id="method">
            <value>GET</value>
        </argument>
        <argument id="remote">
<value>^concat('"/Pipeline/parameters/parameter[name='host']/values/unique/text(), '/data/experiments/', /Pipeline/parameters/parameter[name='sessionID']/values/unique/text(), '/scans/', PIPELINE_LOOPON(series), '/resources/DICOM/files")^</value>
        </argument>
    </resource>
</step>
<!-- Check and delete unfit series -->
    <step id="CLEAN" description="Run clean script"
workdirectory="^concat(/Pipeline/parameters/parameter[name='workdir']/values/unique/text(), '/RAW')^" continueOnFailure="true" >
    <resource name="cleanMRSession"
location="CleanMRSession/resources">
        <!-- cleanMRSession RAW/$series -->
        <argument id="user">
<value>^/Pipeline/parameters/parameter[name='user']/values/unique/text()^</value>
        </argument>
        <argument id="password">
<value>^/Pipeline/parameters/parameter[name='pwd']/values/unique/text()^</value>
        </argument>
        <argument id="URL">
<value>^concat('"/Pipeline/parameters/parameter[name='host']/values/unique/text(), '/data/experiments/', /Pipeline/parameters/parameter[name='sessionID']/values/unique/text(), '/scans/')^</value>
        </argument>
        <argument id="series">
            <value>^PIPELINE_LOOPON(series)^</value>
        </argument>
```

```

        </resource>
    </step>
<!-- Delete all files -->
    <step id="RMDIRS" description="Remove all DICOMs"
workdirectory="~/Pipeline/parameters/parameter[name='workdir']/values/unique
/text()^">
        <!-- rm -rf $WORKDIR/RAW/series $WORKDIR/CLS/series -->
        <resource name="rm" location="commandlineTools" >
            <argument id="r"/>
            <argument id="f"/>
            <argument id="file">
<value>^concat(/Pipeline/parameters/parameter[name='workdir']/values/unique/
text(), '/noRAW')^</value>
            </argument>
        </resource>
    </step>
<!-- The end -->
</steps>
</Pipeline>

```

cleanMRSession.xml (descripción de cleanMRSession.sh)

El código, completo, es así:

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- -->
<Resource xmlns="http://nrg.wustl.edu/pipeline">
    <name>cleanMRSession.sh</name>
    <location>CleanMRSession/scripts</location>
    <commandPrefix>bash</commandPrefix>
    <type>Executable</type>
    <description>Moves admissible series from one folder to
another</description>
    <estimated_time>00:01:00</estimated_time>
    <input>
        <argument id="user">
            <name>u</name>
            <description>User</description>
        </argument>
        <argument id="password">
            <name>p</name>
            <description>Password</description>
        </argument>
        <argument id="URL">
            <name>U</name>
            <description>URL</description>
        </argument>
        <argument id="series">
            <description>Series</description>

```

```
</argument>  
</input>  
</Resource>
```

cleanMRSession.sh (script que realiza el borrado)

El código, completo, es así:

```
#!/bin/bash  
  
#NOTA: requires dicom3tools' dcmkey  
  
ARGS=( "$@" )  
  
#parse arguments  
USER=""  
PASSWORD=""  
URL=""  
SERIES=""  
for ((n=0; n<${#ARGS[@]}; n++)) ; do  
  case "${ARGS[$n]}" in  
    -u)  
      let n=n+1  
      USER="${ARGS[$n]}"  
      ;;  
    -p)  
      let n=n+1  
      PASSWORD="${ARGS[$n]}"  
      ;;  
    -U)  
      let n=n+1  
      URL="${ARGS[$n]}"  
      ;;  
    -*) echo "Skipping option ${ARGS[$n]}" >&2  
      ;;  
    *)  
      SERIES="${ARGS[$n]}"  
      ;;  
  esac  
done  
  
echo "USER=$USER"  
echo "PASSWORD=$PASSWORD"  
echo "URL=$URL"  
echo "SERIES=$SERIES"  
  
#temporaries  
TMP_PRO=$(mktemp)
```



```
#Tipos de estudios
cat > $TMP_PRO <<.EOF
ep2d_bold_p2_resting_state
ep2d_fid_basic_bold_p2_AP
ep2d_fid_basic_bold_p2_PA
asl_3d_tra_iso_3.0_highres
DTIep2d_diff_mddw_48dir_p3_AP
DTIep2d_diff_mddw_4b0_PA
t1_mprage_sag_p2_iso
t2_space_dark-fluid_sag_p2_iso
.EOF

#Archivos de la misma serie ($4 único), con fechas y descripciones,
filtrados por tipo de estudio
DCM="$(ls "$SERIES" | head -n 1)"
dckey -k "AcquisitionDate" "$SERIES/$DCM" 2>&1 | grep -v Error &&\
dckey -k "SeriesDescription" "$SERIES/$DCM" 2>&1 | grep -v Error | grep -v -
f $TMP_PRO &&\
$HOME/pipeline/xnat-tools/XnatDataClient -u $USER -p $PASSWORD -r
"$URL/$SERIES" -m DELETE

#Dejar todo limpio
rm -f $TMP_PRO
```

From:
<http://detritus.fundacioace.com/wiki/> - Detritus Wiki

Permanent link:
http://detritus.fundacioace.com/wiki/doku.php?id=neuroimagen:xnat_pipelines&rev=1572392487

Last update: 2020/08/04 10:45

