WPS
Scripting BFM and HTM

Published on Apr 05, 2008

Scripting BFM and HTM

The Business Flow Manager (BPEL) and Human Task Manager APIs can be scripted. Scripting means invoking the APIs from a scripted language such as Jython without having to code and compile Java code. This can be used for quick tests or even for more advanced administration functions where programs may not be needed.

Both the BFM and HTM expose themselves as EJBs. In order for a scripting client to call an EJB, it needs to run in a suitable environment. The wsadmin command provides just such an environment.

The following can be used to start the appropriate wsadmin environment:

BPE
```
wsadmin -lang jython -wsadmin_classpath <WPSROOT>/ProcessChoreographer/client/bpe137650.jar
```

HTM
```
wsadmin -lang jython -wsadmin_classpath <WPSROOT>/ProcessChoreographer/client/task137650.jar
```

A common start to a Jython program looks like:

```
from javax.naming import InitialContext

context = InitialContext()
home = context.lookup("com/ibm/bpe/api/BusinessFlowManagerHome")
bfm = home.create()
```

List of samples:

- Starting a process
- Creating and Starting a Human Task
- Querying the tables
- Deleteing & Terminating Human Tasks
- Delete Finished Processes
- Find Stopped Activities
- Force retry of Stopped Activities
- Getting the graphics (SVG) for a process
- Getting a list of Invocation Tasks

Starting a process

Here is a script for starting a simple process called **P1** with a String parameter called *input1*. To work with Business Objects in a script, the directory containing the WSDL/XSDs must be added to the wsadmin classpath.

```
from javax.naming import InitialContext

context = InitialContext()
home = context.lookup("com/ibm/bpe/api/BusinessFlowManagerHome")
bfm = home.create()

template = bfm.getProcessTemplate("P1")

input = bfm.createMessage(template.getID(), template.getInputMessageTypeName());
dataObject = input.getObject()
dataObject.setString("input1", "hello")

piid = bfm.initiate(template.getName(), None, input)
```

Creating and Starting a Human Task

The following shows a script that can be used to create an instance of a Human Task. The task is called {http://m1}HT1 and has an input called input1 that is a string.

```
from javax.naming import InitialContext

context = InitialContext()
home = context.lookup("com/ibm/task/api/HumanTaskManagerHome")
htm = home.create()

tkiid = htm.createTask("HT1","http://m1")

cow = htm.createInputMessage(tkiid)
dataObject = cow.getObject()
dataObject.setString("input1", "hello")

htm.startTask(tkiid, cow, None)
```

Querying the tables

The following script queries the existing tasks and displays their Task IDs.

```
from javax.naming import InitialContext

context = InitialContext()
home = context.lookup("com/ibm/task/api/HumanTaskManagerHome")
htm = home.create()

select='DISTINCT TASK.TKIID'
where=None

resultSet = htm.query(select, where, None, None, None)
print resultSet
print resultSet.size()

hasMore = resultSet.first()
while hasMore:
  tkiid = resultSet.getOID(1)
  print tkiid
  hasMore = resultSet.next()
```

Deleteing & Terminating Human Tasks

```
from javax.naming import InitialContext

context = InitialContext()
home = context.lookup("com/ibm/task/api/HumanTaskManagerHome")
htm = home.create()

select='DISTINCT TASK.TKIID'
where='TASK.STATE = TASK.STATE.STATE_READY'

resultSet = htm.query(select, where, None, None, None)

hasMore = resultSet.first()
while hasMore:
  tkiid = resultSet.getOID(1)
  print tkiid
  htm.terminate(tkiid)
  hasMore = resultSet.next()
```

Deleteing Finished Processes

```
from javax.naming import InitialContext

context = InitialContext()
home = context.lookup("com/ibm/bpe/api/BusinessFlowManagerHome")
bfm = home.create()

select='DISTINCT PROCESS_INSTANCE.PIID'
where='PROCESS_INSTANCE.STATE = PROCESS_INSTANCE.STATE.STATE_FINISHED'

resultSet = bfm.query(select, where, None, None, None)

hasMore = resultSet.first()
while hasMore:
  piid = resultSet.getOID(1)
  print piid
  bfm.delete(piid)
  hasMore = resultSet.next()
```

Finding Stopped Activities

In a BPEL process, Activities can be flagged to stop on error. These activities then enter the Stopped state and can be located with a query.

```
from javax.naming import InitialContext

context = InitialContext()
home = context.lookup("com/ibm/bpe/api/BusinessFlowManagerHome")
bfm = home.create()

select='DISTINCT ACTIVITY.AIID'
where='ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED'

resultSet = bfm.query(select, where, None, None, None)

hasMore = resultSet.first()
while hasMore:
  aiid = resultSet.getOID(1)

  activityInstanceData = bfm.getActivityInstance(aiid)
  print '-----'
  print 'Process Instance ID:  ', activityInstanceData.getProcessInstanceID()
  print 'Activity Instance ID: ', activityInstanceData.getID()
  print 'Name:                 ', activityInstanceData.getName()
  print 'Process TemplateName: ', activityInstanceData.getProcessTemplateName()

  hasMore = resultSet.next()
```

Force retry of Stopped Activities

When a BPEL activity has been stopped, it can be restarted with the forceRerty method. It can also be skipped with the forceComplete method.

```
from javax.naming import InitialContext

context = InitialContext()
home = context.lookup("com/ibm/bpe/api/BusinessFlowManagerHome")
bfm = home.create()

select='DISTINCT ACTIVITY.AIID'
where='ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED'

resultSet = bfm.query(select, where, None, None, None)

hasMore = resultSet.first()
while hasMore:
  aiid = resultSet.getOID(1)

  continueOnError = 0
  bfm.forceRetry(aiid, continueOnError)

  hasMore = resultSet.next()
```

Getting the graphics (SVG) for a process

The Business Flow Manager provides a method called `getGraphics()` that returns an XML document that describes an SVG image of the BPEL process template. The following example illustrates retrieving this graphics and saving to a file which can then be opened in an SVG editor/viewer

```
from javax.naming import InitialContext
from java.io import FileOutputStream

context = InitialContext()
home = context.lookup("com/ibm/bpe/api/BusinessFlowManagerHome")
bfm = home.create()
templateName = "TemplateName"
data = bfm.getGraphics(templateName)
fios = FileOutputStream("C:\\file.svg")
fios.write(data)
fios.close()
```

Getting a list of Invocation Tasks

An Invocation Task is one which when created, causes a process or SCA component to be started. In order to launch an invocation Task, you may need to find the list of Task Templates that can be launched. The following script illustrates how to obtain these:

```
from javax.naming import InitialContext

context = InitialContext()
home = context.lookup("com/ibm/task/api/HumanTaskManagerHome")
htm = home.create()

taskTemplateArray = htm.queryTaskTemplates('TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_ORIGINATING', None, None, None)
for x in taskTemplateArray:
  print '----'
  print 'Name:', x.getName(), 'Kind:', x.getKind()
```

See Also:

- Human Task Manager Programming
- Business Flow Manager Programming
- Jython - The Jython/Python programming language
- BFM and HTM Queries - Querying BFM and HTM.

Added by Neil Kolban , last edited by Neil Kolban on Apr 05, 2008 (view change) SHOW COMMENT
Labels: (None)

Info

0 comments