LotusDomino

Using data from Lotus Domino

There are several ways to access data stored in Lotus Notes / Domino.

ReadViewEntries

ReadViewEntries is a standard Lotus Domino URL command that can be used to retrieve entries of a view in XML format. This has been available since Lotus Domino 5.0.2.

More info can be found in the official Lotus documentation , in an article at IBM Developerworks and in an old IBM Redbook.

There are some online samples here.

Typical usage is as part of a generator in a pipeline.

Example pipeline snippet:

```
  <map:match pattern="internal/faqs/*">
    <map:generate type="file" src=
"http://domino.company.com/faqdatabase.nsf/faqs-by-topic?readviewentries&amp;RestrictToCategory={1}&amp;
Start=1&amp;Count=300"/>
    <map:transform src="xsl/domino/faq-view2xhtml-snippet.xsl"/>
    <map:serialize type="xml"/>
  </map:match>
```

Some remarks:

- ReadViewEntries provides only *read* access to data

Toolkit for Java/CORBA

This toolkit includes java classes to access data on a Lotus Domino server. Escpecially the NCSO.jar file is important because it allows remote access to the Domino server (and your Cocoon server is probably not running on the same machine as your Domino server).

The toolkit can be downloaded from the Lotus Toolkits and Drivers site. The documentation is available here.

Typical usage is as Java code in your own component, Java code in an XSP, or in flowscript.

Example flow snippet:

```
importPackage(Packages.lotus.domino);

function writelog(userid, message)
{
  var s = null ;
  var db = null ;
  var logdoc = null ;

  try
  {
    s = Packages.lotus.domino.NotesFactory.createSession("domino.company.com", "userid", "password");
    db = s.getDatabase(null, "mydatabase.nsf") ;

    logdoc = db.createDocument();
    logdoc.replaceItemValue("Form", "Log");
    logdoc.replaceItemValue("logUserid", userid) ;
    logdoc.replaceItemValue("logAction", message);
    logdoc.save() ;
  }
  catch(e)
  {
  }
  finally
  {
    try
    {
      if(logdoc != null)
      {
        logdoc.recycle() ;
      }
      if(db != null)
      {
        db.recycle() ;
      }
      if(s != null)
      {
        s.recycle() ;
      }
    }
    catch(e)
    {
    }
  }
}
```

Some remarks:

- The Java toolkit allows both *read* and *write* access.
- The Java toolkit uses DIIOP (Corba) for remote connections. Performance is not great.
- You need to call the recycle() method to clean up memory. There's no decent memory management.
- The jar files of this toolkit are also included in recent Lotus Notes clients.