OpenLog Logger for XPages

IMPORTANT NOTE: Java error logging calls will need amending for when migrating to 6.0.0. All OpenLogItem static calls have been replaced with com.paulwithers.openLog.OpenLogUtil methods.

Variants of OpenLogItem.logError() and OpenLogItem.logEvent() need changing to OpenLogUtil.logError() and OpenLogUtil.logEvent(). Any other method calls to OpenLogItem will need amending to OpenLogUtil.getOpenLogItem(). Using DDE's Search functionality will allow a “find and replace”.

SSJS error / event logging requires no amendments.

Introduction

This code began as a Java class in XPages Help Application. It was subsequently extracted and significantly enhanced. Now **OpenLog** is easier and more flexible than ever before to use in XPages.

- It is Apache licensed
- It can be used in your managed beans and other Java classes, as it has ever since XPages Help Application.
- It can be used directly from SSJS with as little as *openLogBean.addError(e,this);*.
- From SSJS all caught errors on the page are logged out together, at the end of the request.
- Only two method names are used from SSJS, one to add an error, one to add an event, making it easy to pick up
- From SSJS you only need to pass the information you wish. There is no need to pass nulls or empty strings.
- From SSJS only one unique error per component is logged, regardless of how many times the error is encountered during the refresh lifecycle. That is not available from any other current logging mechanism.
- In SSJS, if you use a custom error page, uncaught errors will also be logged.
- Uncaught errors will be logged for the page the error occurs on, not your custom error page.
- This database includes full documentation of how to log errors.
- You can define the OpenLog path and a variety of other variables without needing to change the code.
- Those variables can be defined in xsp.properties on the NSF or the server. You can also define a backup in the notes.ini. If nothing is set defaults will be used.
- The functionality and code are available as an OSGi plugin (the best practice approach) but can also be included in individual NSFs.

Configuration

Configuration settings can be defined in the xsp.properties of the database or server or in the server's notes.ini. They are retrieved in that order.

Set the OpenLog path by using xsp.openlog.filepath, defaulting to "OpenLog.nsf"

Set debug level if errors occur in OpenLogItem class using xsp.openlog.debugLevel. The default is 2, to log just full details. 1 will log just the message, 0 will log nothing.

By default SSJS errors will also be added to a FacesMessage, so displayed if a partial refresh occurs. The error also includes the control id the error relates to, if an id has been defined (best practice). To suppress this set xsp.openlog.displayError=false

xsp.openlog.genericErrorMessage can be used to define a generic error message to be displayed to the users e.g. "Please contact IT, an error has occurred". If xsp.openlog.genericErrorMessage returns an empty string, the default, then the full error is returned

xsp.openlog.email can be used to send logs to an OpenLog database that is set up as a mail-in database. The log document is created in the server's mail.box and saved with the email address as the recipient.

xsp.openlog.suppressEventStack can be used to suppress stack traces in Event logs. Full stack traces are still recorded for errors. NOTEThe stack trace is the only method I know of to identify which XPages lifecycle phase an Event / Error is logged for, e.g. you may get a message logged for Restore View phase because a variable is not available, but the page displays fine because the relevant variable is available in Render Response.

xsp.openlog.expireDate can be used to set the ExpireDate field used by auto-archiving. This should be a number of days, e.g. 30, to auto-expire the log at 30 days from the date logged.

xsp.openlog.includeQueryString can be used to include any query string values (e.g. ?open&login) in the "Agent" field in the OpenLog log.

xsp.openlog.suppressEventControl can be used to suppress the ID of the control triggering an Event when logging from SSJS.

xsp.openlog.templateFilepath can be used to define a template filepath. If no database exists at the OpenLog path, a new database will be created based on the template.

Implementation

The code can be implemented either within an NSF or as an OSGi plugin. If you are using OpenNTF Domino API, all the functionality is already included in that product. Instead of OpenLogUtil, use XspOpenLogUtil. Instead of OpenLogUtil.getOpenLogItem(), use XspOpenLogUtil.getXspOpenLogItem(). All other method names are unchanged.

The **Examples** area provides some examples you can look at. "Error On Load" and "Two Errors" will re-route to the error page.

One important note on XPages custom error pages. **Never use XPages events on an error page.** For example, a Section control on an XPage adds events to expand / collapse the section. The section will work as normal if the error page is triggered from a full refresh. But if the error page is triggered from a partial refresh, the XSP Command Manager only prints HTML to the browser, it doesn't attach events. So your section will not work. **Error** is the custom error page in this application and gives an example of an XPage with only standard HTML.

In An NSF

The Java classes and faces-config.xml code to enable the openLogBean managed bean

and Phase Listener can be placed into an NSF.

1.Add the src/com.paulwithers.openLog package to the WebContent/WEB-INF folder.

2.Add the XML for the Phase Listener and Managed Bean to your database's faces-config.xml, also in the WebContent/WEB-INF folder. Ensure it is outside the AUTOGEN section. Anything within that section gets removed when the NSF is built.

3.Ensure the WebContent/src folder is added as a source folder, by right-clicking and selecting Build Path > Use as source folder.

As An OSGi Plugin

Alternatively, the functionality is also available as an OSGi plugin. Just import the updatesite.zip into an Update Site database on your server and issue a *restart task http* command to the server. The steps are exactly the same as installing the Extension Library.

In Designer select File > Application > Install. Choose *Search for new features to install,* adding the same Update Site as a zip/jar location. Again, the same steps as installing the Extension Library.

In each NSF, in Application Properties on the Advanced tab (Designer 8.5) or Xsp Properties on the Page Generation tab (Designer 9.0) enable the com.paulwithers.openLog.library.StarterLibrary extension library.

If using for XPiNC, users will also need to install the library. The recommended method is via a Widget Catalog.

Follow Chapter 2 of XPages Extension Library book or see the video for full details.

Using from Java

Using OpenLog from Java is even easier than ever.

The OpenLogItem is now accessible from a utility class. Just import com.paulwithers.openLog.OpenLogUtil (similar to the utility class name that has been in use in OpenNTF Domino API for some time). Method names haven't changed.

- ***OpenLogUtil.logError(Throwable)***: pass the Exception / Throwable from your try/ catch block.
- ***OpenLogUtil.logError(Session, Throwable)***: as above, but you can pass a different session, if required.
- ***OpenLogUtil.logError(Session, Throwable, String, Level, Document)***: full control, but only worth using if you want to pass a different session. It calls logErrorEx, so if you are using the default session, it's advisable to just call OpenLogItem.logErrorEx().
- ***OpenLogUtil.logErrorEx(Throwable, String, Level, Document)***: first param is an Exception / Throwable or null; second param is a String custom error message; third param is a java.util.logging.Level object - Level.SEVERE, Level.WARNING, Level.INFO, Level.CONFIG, Level.FINE, Level.FINER, or Level.FINEST; fourth param is a Document object or null.
- ***OpenLogUtil.logEvent(Throwable, String, Level, Document)***: params as above.
- ***OpenLogUtil.logEvent(Session, Throwable, String, Level, Document)***: params as above, plus the ability to pass a different session.

To access the OpenLogItem to control settings at a granular level, use **OpenLogUtil.getOpenLogItem()**.

If you wish to pass a message back to the browser, you can use **OpenLogUtil.getOpenLogItem().addFacesMessage(String componentName, String message)**. This checks **xsp.openlog.displayError** and also **xsp.openlog.genericErrorMessage**.

Using from SSJS

Why have I added this? You'll notice there's no error handling in the SSJS in XPages Help Application. Matt White's OpenLogXPages script library in TaskJam is excellent and I've used it extensively. But it's not Apache-licensed, so I couldn't use it in XPages Help Application. So I've added a managed bean and code to aggregate all error or event messages and output in one go prior to rendering the response.

In addition it will log to OpenLog any error triggered because you do not have a try/catch block, but only if you have a custom error page in the application. This is because the error is retrieved from requestScope.error. If you use the standard error page or use no error page at all, requestScope.error is never encountered when an XPage is rendered. Note, however, that if you do not have a try/catch block only one error per page will be logged. The XSP Command Manager aborts as soon as an uncaught error is encountered.

To log a caught error, the following options are available:

- ***openLogBean.addError(Exception, Component)***: pass the error object from your try/catch and the component. To avoid hard-coding, use ***this*** from a property or ***this.getParent()*** from an event (so
  ```
  try{
    ...
  } catch(e) {
    openLogBean.addError(e,this);
  }
  ```
  If no component is available, pass *null*.
- ***openLogBean.addError(Exception, Component, int)***: the extra parameter here is an integer, 1 to 7, corresponding to the Java Level logged. 1 is the most severe, 7 the least.
- ***openLogBean.addError(Exception, Component, int, string)***: the extra parameter here is the UNID of a NotesDocument to pass. It is not good practice to store Notes objects in a bean, so only pass the UNID. The code will attempt to retrieve the document from the current database. If that cannot be done the UNID will be added to the error logging message instead.
- ***openLogBean.addError(Exception, string, Component)***: the new second parameter is an extra message as a string. This provides the equivalent to logErrorEx. If there is no error, pass *null*.
- ***openLogBean.addError(Exception, string, Component, int)***: passing error object, additional message, component and error level.
- ***openLogBean.addError(Exception, string, Component, int, string)***: passing error object, additional message, component, error level and NotesDocument

UNID.

To log an event, the following options are available:

- ***openLogBean.addEvent(string, Component)***: the first parameter is a message, the second parameter is the component to log against.
- ***openLogBean.addEvent(string, Component, int)***: the extra parameter is a warning level, 1 to 7, as in addError().
- ***openLogBean.addEvent(string, Component, int, string)***: the extra parameter is the UNID of a NotesDocuemnt to pass, as in addError().

Source Control

The source code for the projects is available on GitHub at https://github.com/paulswithers/openlogjava and on OpenNTF Stash at .

Files Included

LICENCE

NOTICE

OpenLogLogger.nsf – database comprising Java classes and additions to faces-config.xml that need to be copied into an NSF. The database also includes XPages with examples and documentation about calling the code from either SSJS or Java.

com.paulwithers.openLog.update.zip – Update Site for OSGi plugin.

com.paulwithers.openLog.src.zip – Source code for plugin, feature and update site projects.

portlist.xml – XML to import Java classes and faces-config.xml into an NSF

OpenLog Logger for XPages Documentation – this file.

Enhancements / Fixes in M2

Full details of enhancements / fixes included in M2 can be found in the GitHub project: https://github.com/paulswithers/openlogjava/issues?milestone=1&page=1

#1 Fix for SSJS errors if EL object does not exist.

#2 Add SSJS errors automatically to facesMessage. The method can be used from Java as well – OpenLogItem.addFacesMessage(String, String). A property org.openlog.displayError=false can be used to suppress this.

#3 Add property to hold a generic error message – xsp.openlog.genericErrorMessage. The message will only be logged once per request from SSJS. This uses OpenLogItem.addFacesMessage(String, String), so Java calls can also use that method.

#4 Add code from org.openntf.domino to prevent infinite loops. The infinite loop should only happen if current database or current session is null, so should not occur in NSF version. But the check has been added for safety.

#5 Ensure generic error message is only logged once per request.

Enhancements / Fixes in M3

Full details of enhancements / fixes included in M3 can be found in the GitHub project: https://github.com/paulswithers/openlogjava/issues?milestone=2&state=closed

#8 Fix for various uncaught exceptions that had different Java classes for error.getCause() or error.getCause().getCause().

#9 Fix for error message greater than 32k (happens if error is in a large SSJS library)

#10 Enhancement to logging to an OpenLog that's a mail-in database. Use xsp.openlog.email for the email address to mail to. You should not need an OpenLog database on the server, because the message is created in the mail.box. (Domino automatically handles re-directing mail.box to mail1.box etc if multiple mail boxes are in use on the server.)

Enhancements / Fixes in M4

Full details of enhancements / fixes included in M4 can be found in the GitHub project: https://github.com/paulswithers/openlogjava/issues?milestone=4&state=closed and https://github.com/OpenNTF/org.openntf.domino/issues/75

#15, #17 Added try/catch blocks to processUncaughtException to handle any unexpected error classes.

#16 Catching NotesException and java.io.IOException classes.

Org.openntf.domino #75 allow suppression of stack traces.

Enhancements in M5

Full details of enhancements included in M5 can be found in the GitHub project: https://github.com/paulswithers/openlogjava/issues?milestone=5&state=closed

#18 Change from requestScope to sessionScope to ensure errors are logged if the developer uses context.redirectToPage() in catch block.

#19 Manage getting the component from eventHandler in the OpenLogErrorHolder, so developers can just pass *this* instead of *this* from properties and *this.getParent()* from eventHandlers. Note: *this.getParent()* will still work, so no enforced changes, just simpler conversion of existing error logging when applying OpenLog to the application.

#20 Allow setting of an xsp.property to automatically add an ExpireDate field to logs for today + xsp.property value. This allows you to use standard Domino auto-archiving.

Enhancements in M5.1

Full details of the fix included in M5.1 can be found on the GitHub project: https://github.com/paulswithers/openlogjava/issues?milestone=5.1&q=is%3Aclosed

#21 Cannot access mail.box as anonymous.

Enhancements in M6

Full details of enhancements included in M5 can be found in the GitHub project: https://github.com/paulswithers/openlogjava/issues?milestone=6&state=closed

#22 Fix for error thrown if historyUrls is empty

#23 Add the user message in OpenLogItem.addFacesMessage()

#24 Add the user message in OpenLog's database views

#25 Configuration option to remove “Event logged for” in database views.

#26 Fix for error when invalid component id for partial refresh

#27 Fix for erroneous reporting of database source of error in plugin

#28 Configuration option to include full URL in logs

#29 Fix for ParseException not captured in uncaught exceptions

#30 Add OpenLogUtil class as Java entry point, for best practice

#31 Use enum for “Error” and “Event”

#32 Configuration option to auto-create database based on template filepath