OpenLog Logger for XPages

Introduction

This code began as a Java class in XPages Help Application. It has now been 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 backup debug level (in case of problems creating an OpenLog document) 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 "OpenLog.nsf" and "0" are the defaults.
- The functionality and code can be added to an NSF. Alternatively, it is available as an Extension Library.

Set the OpenLog path by using xsp.openlog.filepath in the xsp.properties of the database or server or in the server's notes.ini. They are retrieved in that order.

Set the default debug level by using xsp.openlog.debugLevel in the same places. This defines what will be logged to the server console if an OpenLog document cannot be created - nothing (0), the error message (1), or the error message and full stack trace (2). OpenLogLogger.nsf has an NSF's code and demo XPages. See that database's xsp.properties for implementation. See **Using from Java** for full documentation on accessing it in Java code.

See **Using from SSJS** for full documentation of how to use it from SSJS.

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.

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.

Implementation

The code can be implemented either within an NSF or as an OSGi plugin.

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 class is now static, so there is no need to create an object before calling any method. Just import com.paulwithers.openLog.OpenLogItem (a new package name, for forwards compatibility) Method names haven't changed. But I've fixed OpenLogItem.LogErrorEx().

- ***OpenLogItem.logError(Throwable)***: pass the Exception / Throwable from your try/catch block.
- ***OpenLogItem.logError(Session, Throwable)***: as above, but you can pass a different session, if required.
- ***OpenLogItem.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().
- ***OpenLogItem.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.
- ***OpenLogItem.logEvent(Throwable, String, Level, Document)***: params as above.
- ***OpenLogItem.logEvent(Session, Throwable, String, Level, Document)***: params as above, plus the ability to pass a different session.

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().