



Stop (de)bugging me!



BLUG
Benelux Lotus User Group

Who am I

- ▶ **Freelance consultant/ developer**
 - ▶ From the Netherlands
- ▶ **IBM Notes/ Domino**
 - ▶ XPages, web, client, mobile (TeamStudio Unplugged)
- ▶ **OpenNTF Contributor**
 - ▶ XPage Debug Toolbar
 - ▶ Auto Logins for Domino
 - ▶ XPage Multiple File Uploader
- ▶ **IBM Champion for 2 years in a row**



Agenda

- ▶ Debuggers in Designer
 - ▶ Java debugger
 - ▶ SSJS debugger
- ▶ XPage Debug Toolbar™

Debuggers in Designer

- ▶ For Java & SSJS (starting with 9)
- ▶ Based on standards
 - ▶ Java Platform Debugger Architecture (JPDA)
- ▶ ‘Remote’ debugging
 - ▶ Enable on server
 - ▶ Client (Designer/ Eclipse) connects to Server on specified port
 - ▶ Only 1 developer at a time can connect
- ▶ Not recommended for production servers
 - ▶ Impacts performance and security
 - ▶ But that’s not a problem
 - ▶ Since we all never debug applications on production servers



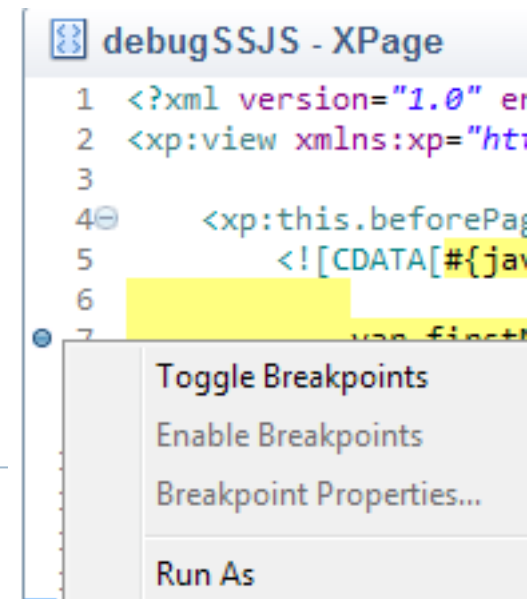
Right?

Java debugger

- ▶ In Designer since 8.5
- ▶ Debug:
 - ▶ Java classes
 - ▶ Managed beans
 - ▶ OSGi plugins
 - ▶ Compiled XPages Java (Local/xsp folder)

Server Side JavaScript debugger

- ▶ New in Domino 9
- ▶ Configured and works (almost) the same as the Java debugger
- ▶ Debug SSJS in the XPage source or a SSJS library
 - ▶ Set breakpoints
 - ▶ Right-click in gutter
 - ▶ Add 'debugger' statement
 - ▶ Don't use this in Designer pre-9 versions!



Activating the debuggers - server

- ▶ Add configuration to notes.ini:

JavaEnableDebug=I

JavascriptEnableDebug=I

JavaDebugOptions=transport=dt_socket,server=y,suspend=n,address=8000

Add this row only if you want to debug SSJS (too)

- ▶ Tip: if you click on the “Debug” icon in the toolbar, you can copy-paste these settings from there

Activating the debuggers - server

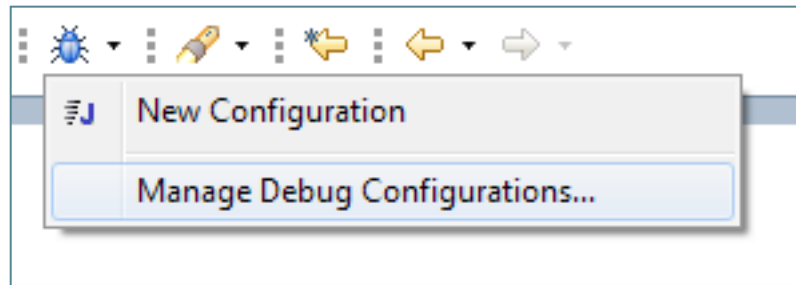
- ▶ Restart the server
- ▶ Watch for this:

```
03/18/2013 09:23:55 PM JUM: WARNING: Remote Java Debugging is enabled, resulting in decreased performance and potentially compromised security
```

- ▶ Port needs to be open(ed) in the firewall
- ▶ Works with local HTTP preview too
 - ▶ Add settings to local notes.ini
- ▶ If the local preview won't start, try starting the local HTTP task from a command prompt
 - ▶ Run “nhttp preview” from your Notes folder

Starting a debug session

- ▶ Click the triangle next to the “bug” icon



(this is Designer 9, icon looks slightly different in 8.5)

- ▶ Select an existing configuration or click “Manage...”

Starting a debug session - Java

Debug Configurations

Create, manage, and run configurations

Attach to a Java virtual machine accepting debug connections

Name: DebugJava

Name for this configuration (can be any)

Project: stop.nsf

Connection Type: Standard (Socket Attach)

Connection Properties:

Host: localhost

IP Address of your server

Port: 8000

Port that the Debugger listens on

Filter matched 11 of 11 items

Debug Close

Starting a debug session - SSJS

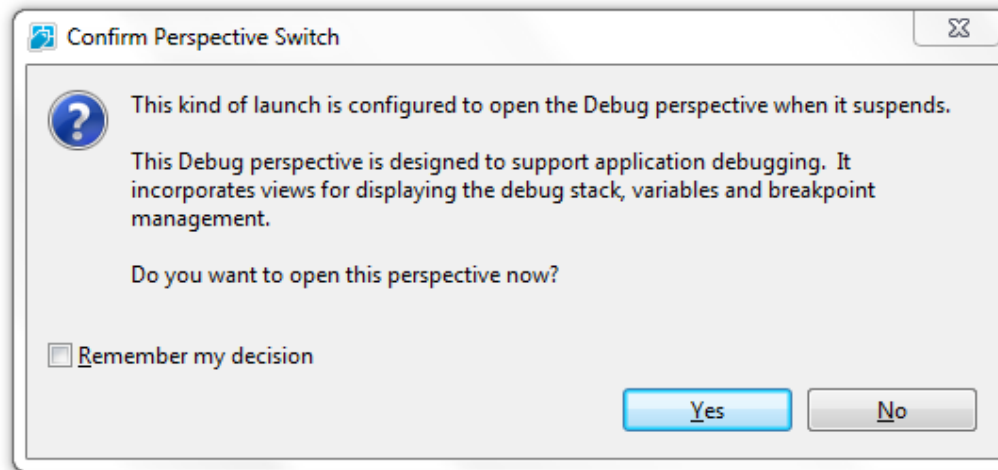
The screenshot shows the Eclipse IDE interface. On the left, the 'Project Explorer' view is visible with a search filter 'type filter text'. The search results list several project types, with 'DebugSSJS' under 'Domino Designer JavaScript' selected. A blue arrow points to this selection. The main area displays the 'Connect' dialog for 'DebugSSJS'. The dialog contains the following information:

- Name:** DebugSSJS
- Connect** tab is active.
- Domino Designer Server Connection Properties**
 - To debug server-side JavaScript, the server must be set up to start in debug mode.
 - Server setup details**
 - To start the server in debug mode, add the following variables to the server's NOTES.INI, then restart the server:
 - JavaEnableDebug=1
 - JavascriptEnableDebug=1
 - JavaDebugOptions=transport=dt_socket,server=y,suspend=n,address=8000
 - The value for "address" must be the port number specified in this configuration.
 - Depending on your environment, you may want to turn off debug mode after you finish debugging.
- Host:** 192.168.137.9
- Port:** 8000
- Debugging behavior**
 - Stop at first line of server-side JavaScript in:
 - All JavaScript
 - Design elements only

At the bottom of the dialog are 'Apply' and 'Revert' buttons. Below the dialog, a status bar indicates 'Filter matched 11 of 11 items'.

Debug perspective

- ▶ Designer automatically asks to open **Debug perspective** when it suspends
 - ▶ Breakpoint or 'debugger' statement in SSJS



- ▶ Change behaviour in Preferences > Run/Debug > Perspectives
 - ▶ Tip: switch perspectives using Ctrl-F8 in Designer
-

▶ Demo

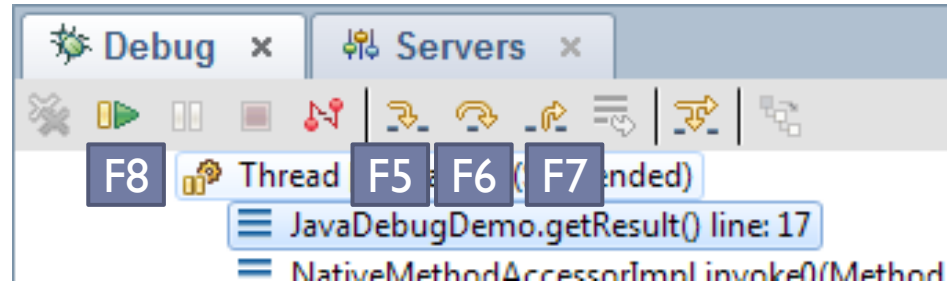
Source not found?

The screenshot illustrates the process of resolving a 'Source not found' error in Eclipse. The main IDE window shows a red error message 'Source not found.' in the editor area, with a button 'Edit Source Lookup Path...' below it. An arrow points from this button to the 'Edit Source Lookup Path' dialog box, which is open and shows a 'Source Lookup Path' containing a 'Default' folder. Another arrow points from the 'Add Source' button in the 'Edit Source Lookup Path' dialog to the 'Add Source' dialog box. The 'Add Source' dialog box has 'Java Project' selected. A third arrow points from the 'Add Source' dialog to the 'Project Selection' dialog box, which is open and shows a list of projects: 'stop.nsf' (checked) and 'openLog.nsf' (unchecked). The 'Project Selection' dialog also has checkboxes for 'Add exported entries of selected projects.' and 'Add required projects of selected projects.' both checked.



Keyboard shortcuts

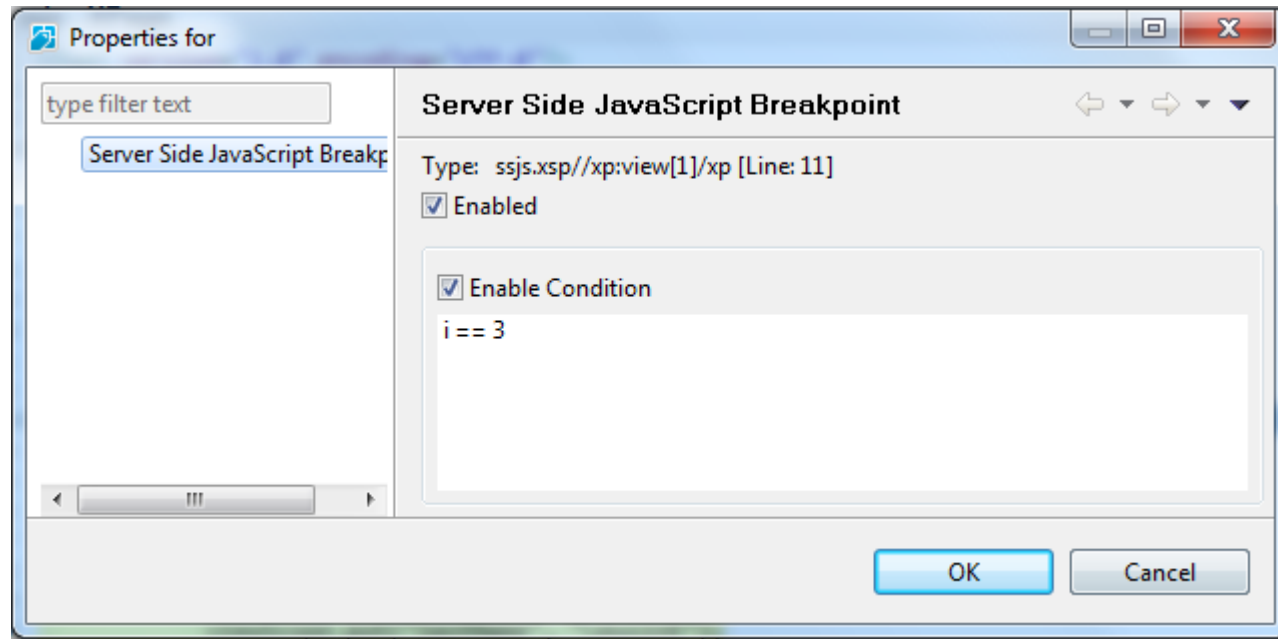
- ▶ F5 Step into
- ▶ F6 Step over
- ▶ F7 Step exit
- ▶ F8 Continue



- ▶ Strange issue with the F6 key
 - ▶ With the default Designer/ Eclipse binding it doesn't work
 - ▶ If you create your own binding to F6 it does
 - ▶ Do this in File > Preferences > Type "keys" in filter

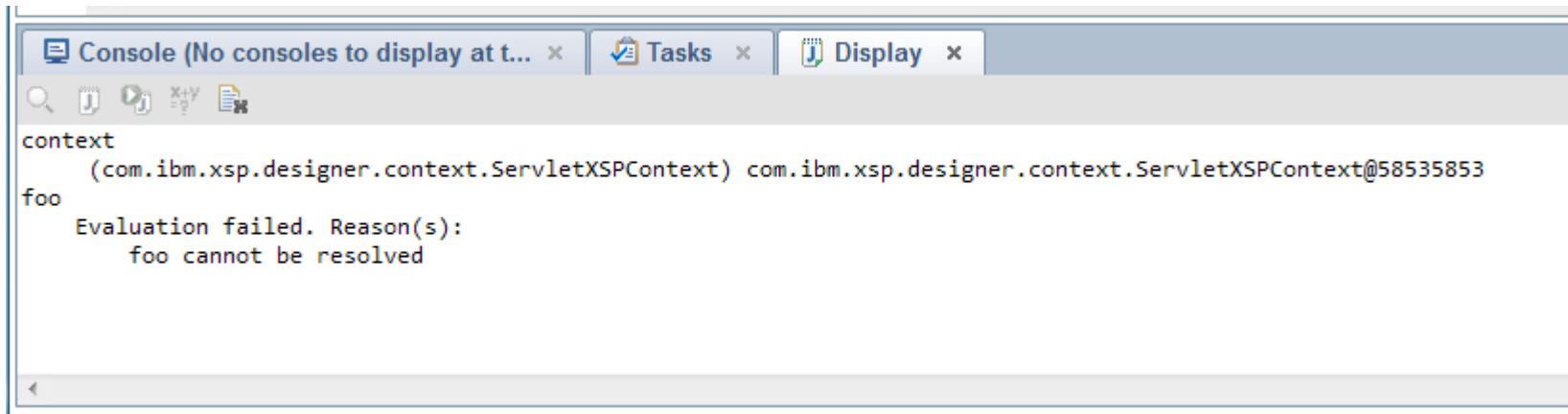
Conditional breakpoints

- ▶ Set breakpoint
- ▶ Right-click on breakpoint
- ▶ Click “Breakpoint properties”



Display view

- ▶ Window > Show Eclipse View > Display
- ▶ Run an expression in the context of a breakpoint and view the result
- ▶ Only works with Java debugger, not SSJS debugger
- ▶ Has code completion

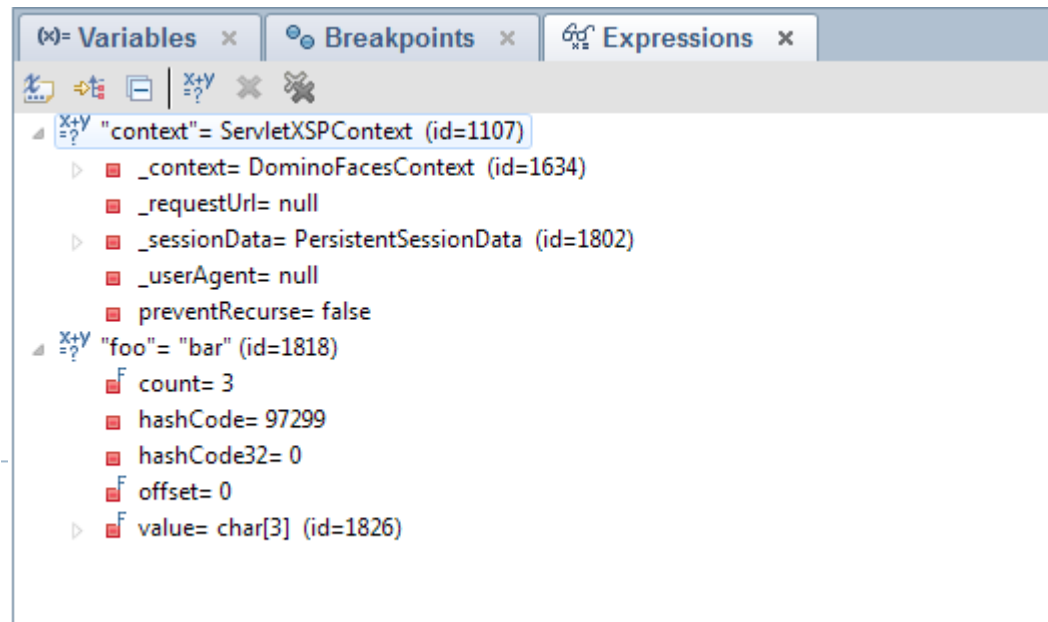


The screenshot shows the Eclipse IDE's 'Display' view. The title bar includes 'Console (No consoles to display at t... x', 'Tasks x', and 'Display x'. The view contains the following text:

```
context
  (com.ibm.xsp.designer.context.ServletXSPContext) com.ibm.xsp.designer.context.ServletXSPContext@58535853
foo
  Evaluation failed. Reason(s):
    foo cannot be resolved
```

Expressions view

- ▶ Window > Show Eclipse View > Expressions
- ▶ Inspect the current state of objects
- ▶ 2 ways to add:
 - ▶ Right-click on expression in 'Display' view
 - ▶ Type in manually

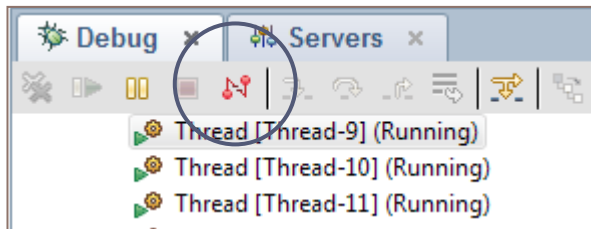


From SSJS to Java

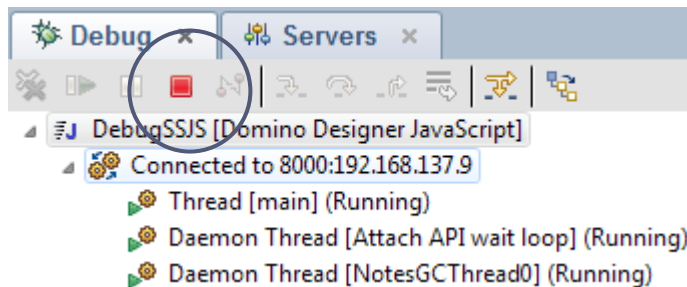
- ▶ You can debug Java code when you're debugging SSJS code
- ▶ But **NOT** using “Step into” (F5)
 - ▶ It will only stop on breakpoints in the Java class

Disconnect

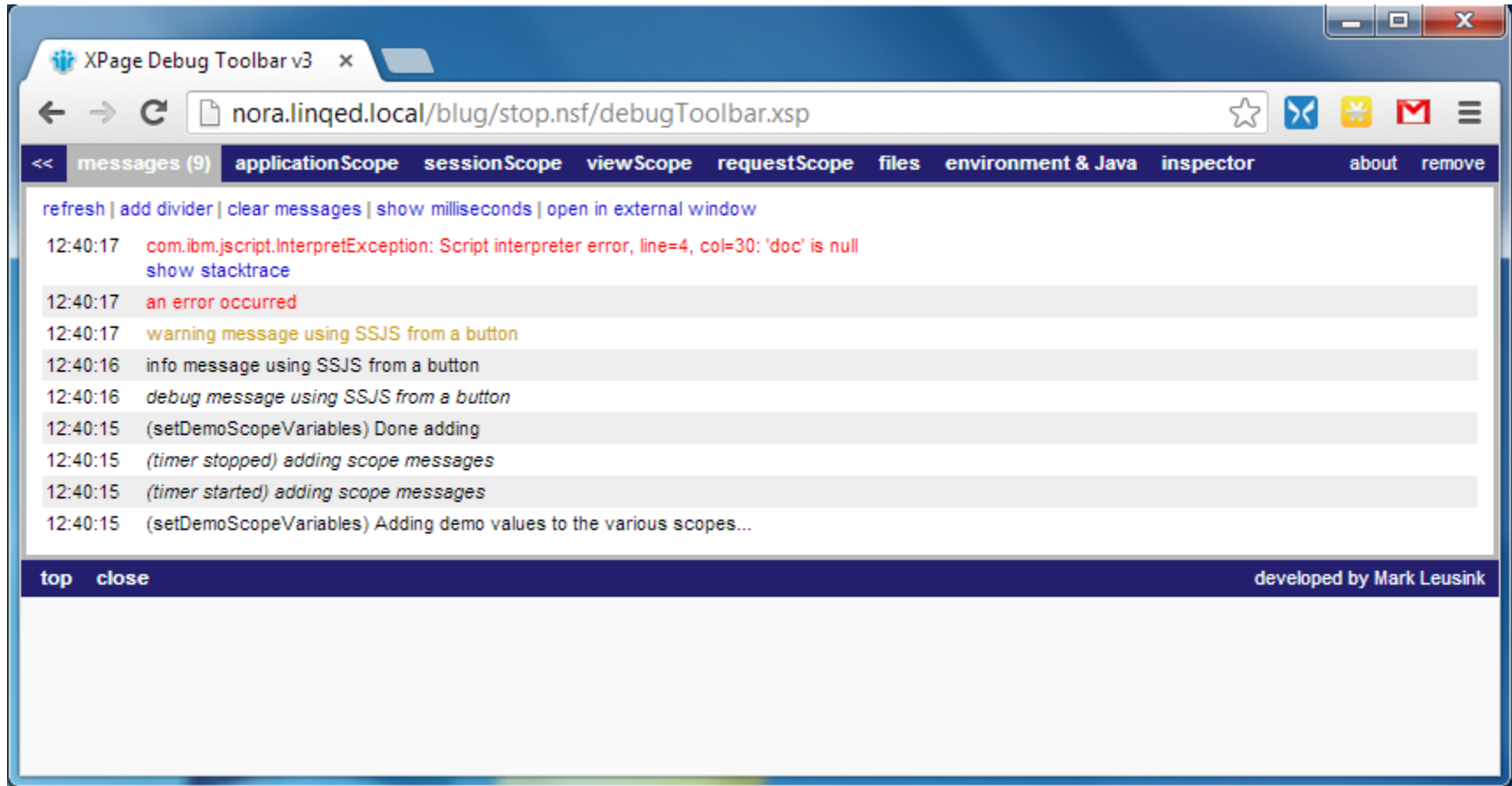
- ▶ When you're done: disconnect your debugging session
 - ▶ Java debugger



- ▶ SSJS debugger



XPage Debug Toolbar



Background/ features

- ▶ Tool for XPage developers
- ▶ Free download from OpenNTF
 - ▶ Or directly from GitHub
- ▶ Features:
 - ▶ Log debug messages
 - ▶ Using `print()` or `_dump()` for debug/ log messages is NOT very convenient
 - ▶ You need access to server console
 - ▶ Messages from everyone (and server messages) in one stream
 - ▶ Makes administrators unhappy
 - ▶ View contents of scopes
 - ▶ Log file reader
 - ▶ API Inspector

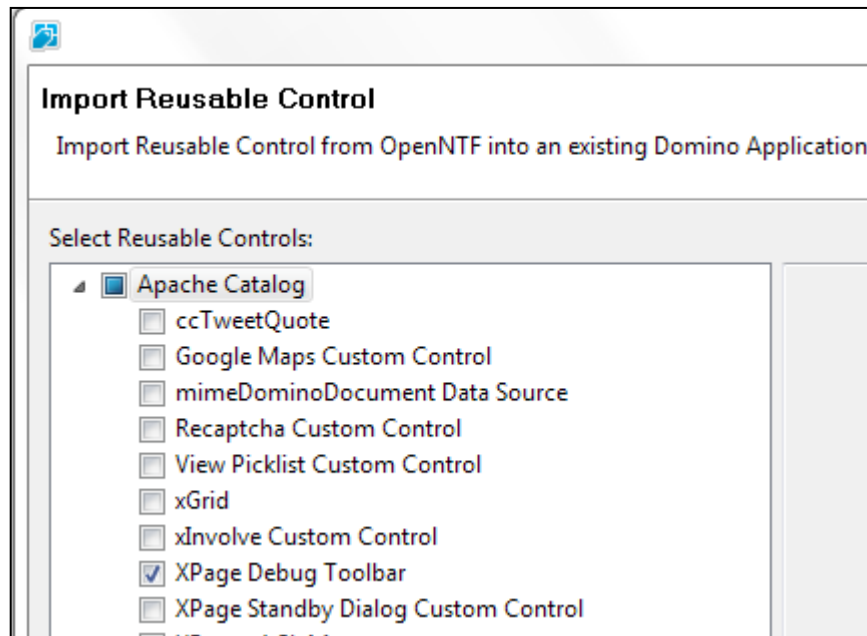


Components

ccDebugToolbar	Custom control	
xpDebugToolbar	SSJS library	
eu.linqed.debugtoolbar.DebugToolbar	Java class	
eu.linqed.debugtoolbar.Message	Java class	
debugToolbarConsole	XPage	optional
debugToolbarErrorPage	XPage	optional
dBar	Managed bean	
Events	View	OpenLog integration
Event	Form	OpenLog integration

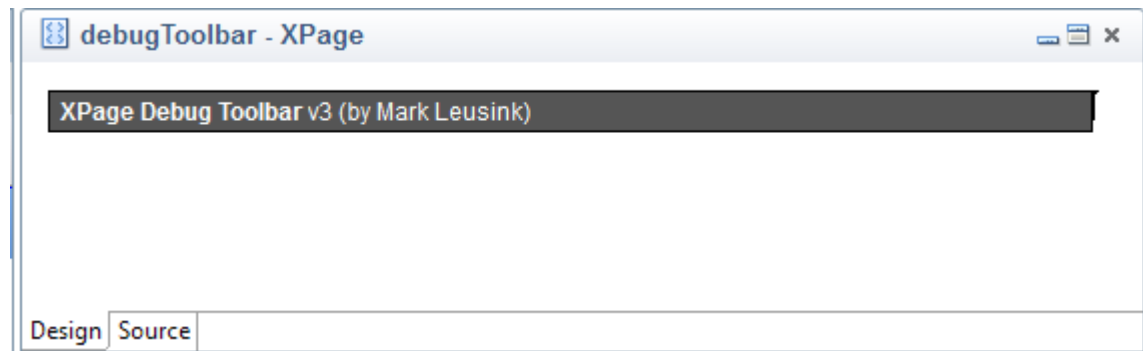
Installation

- ▶ 2 ways to install:
 - ▶ Download from OpenNTF and copy to your application
 - ▶ Use the OpenNTF Import/ export tool



Installation

- ▶ Add the (dBar) managed bean
- ▶ Add the **ccDebugToolbar** custom control to your XPage
- ▶ Change the default settings using custom control properties
 - ▶ collapseTo
 - ▶ defaultCollapsed
 - ▶ color



Scope contents

- ▶ Shows contents of application-, session-, view- and requestScope
- ▶ Remove a variable
 - ▶ Or clean an entire scope
- ▶ You can modify values through the Inspector

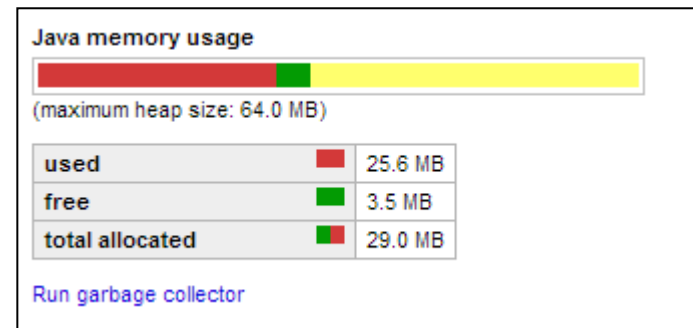
Environment variables

- ▶ Information about the current:

- ▶ User
- ▶ Browser
- ▶ Server
- ▶ Database
- ▶ Request

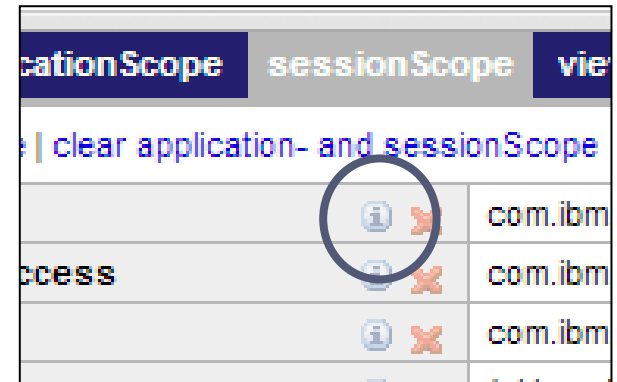
- ▶ Java heap size

- ▶ Maximum heap size
- ▶ Allocated
- ▶ Used
- ▶ Free



Inspector

- ▶ View classes for controls on the current page
 - ▶ And change them
- ▶ View any variable:
 - ▶ Scoped variables
 - ▶ Click on the (i) to open a variable in the Inspector
 - ▶ Managed beans
 - ▶ Shows all properties and methods
 - ▶ Any XPage runtime object:
 - ▶ database
 - ▶ session
 - ▶ context
 - ▶ view
- ▶ Drill-down to see what value/ object is returned



Messages

- ▶ Add message logging calls to your application
- ▶ **Developer / session specific** messages
 - ▶ Cleared when browser is closed
- ▶ Add messages using:
 - dBar.debug(“message”)
 - dBar.info(“message”)
 - dBar.warn(“message”)
 - dBar.error(“message”)
 - dBar.dump(<object>)
- ▶ Specify a context using an optional 2nd parameter:
 - dBar.info(“message”, “context”)

Messages

- ▶ `dBar.error()` function accepts **'error'** objects:

```
try {  
    var doc = null;  
    var id = doc.getUniversalID();  
} catch (e) {  
    dBar.error(e);  
}
```

- ▶ Add a **divider**: `dBar.addDivider()`;

Messages

- ▶ Messages can also be written from Java
 - ▶ `DebugToolBar.get().info("message");`
 - ▶ `DebugToolBar.get().warn("message");`
- ▶ Or catch an Exception
 - ▶ `DebugToolBar.get().error(e);`

Log messages to documents

- ▶ Create **documents** for all dBar calls
 - ▶ In a configured database
- ▶ All required code already built-in
- ▶ Uses the **OpenLog** (form/fields) format
 - ▶ Use the OpenLog database to view them
- ▶ **Log level** can be set: log only messages with a certain log level (or 'higher')
 - ▶ warn = warn + error
 - ▶ debug = debug + info + warn + error

Log messages to documents

- ▶ Configure using **managed properties**

- ▶ logDbPath current, logdb.nsf
- ▶ logEnabled true, false
- ▶ logLevel debug, info, warn, error

- ▶ **Logging will continue even if the toolbar isn't displayed !**

Best practices

- ▶ Add **[debug]** role to the ACL
- ▶ Set **loaded property** of the ccDebugToolbar only for [debug] role:
 - ▶ loaded=“#{javascript:context.getUser().getRoles().contains('[debug'])}”
- ▶ Enable logging to documents
 - ▶ logLevel = “warn” (errors and warnings only)
- ▶ Leave everything in when moving to production

Resources

- ▶ XPage Debug Toolbar on OpenNTF

<http://www.openntf.org/internal/home.nsf/project.xsp?action=openDocument&name=XPage%20Debug%20Toolbar>

- ▶ XPage Debug Toolbar on GitHub

<https://github.com/markleusink/XpageDebugToolbar>

- ▶ OpenNTF Import/ export tool

<http://www.openntf.org/internal/home.nsf/project.xsp?action=openDocument&name=Import%20and%20Export%20for%20Designer>

- ▶ OpenLog

<http://www.openntf.org/internal/home.nsf/project.xsp?action=openDocument&name=OpenLog>

Thank you !

- ▶ Twitter: markleusink
- ▶ Skype: mark_leusink
- ▶ Blog: <http://linqed.eu>
- ▶ Email: m.leusink@linqed.eu

