Successfully Delivering XPages Projects – All Things Considered

BLUG

Belux Lotus User Group

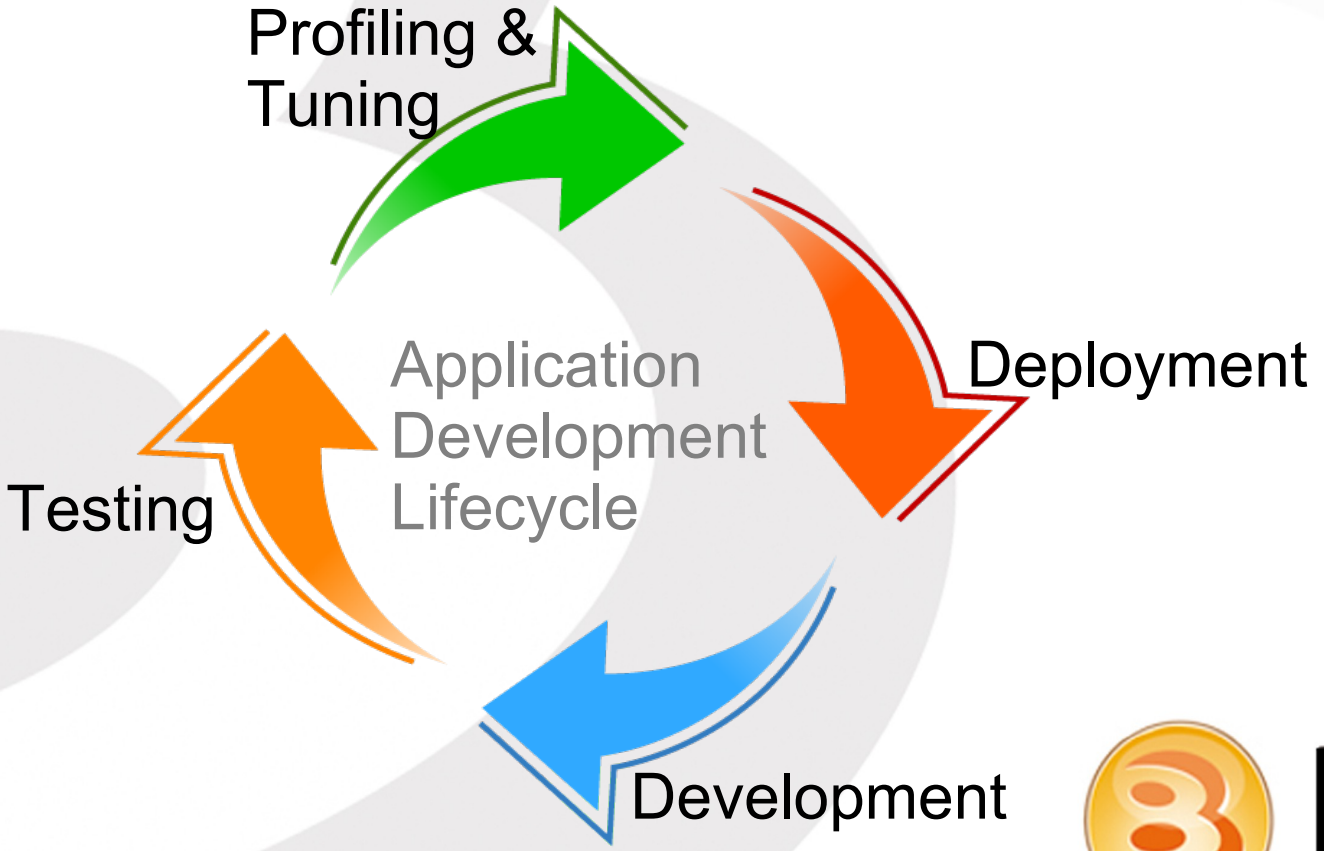
Martin Donnelly
Pete Janzen

Agenda

- Speaker Introduction
- Session Goals
- Team Based Application Development
- Application Testing
- Tuning, Profiling & Debugging
- Deployment Considerations
- Wrap Up

Session Goals

- To examine XPages best practices at critical stages of the application development lifecycle with a view to improving successful project delivery



BLUG
Belux Lotus User Group

Agenda

- Speaker Introduction
- Session Goals
- Team Based Application Development
- Application Testing
- Tuning, Profiling & Debugging
- Deployment Considerations
- Wrap Up

Team Based Application Development

- You are putting together an XPages development team to deliver a new project
  - “How do you organize a team to effectively share code during the development lifecycle?”
    - Comparing code sharing practices
      - Classic Design Source Control
      - XPages Design Source Control
  - “How do you use Domino Designer with a source control system?”
    - What source control systems can Domino Designer integrate with?
    - What operations can be performed?
    - What are the benefits?

Classic Design Source Control "Lock Model"

- Predicated on the principles of "**Lock** – **Modify** – **Unlock**"
  - Only one developer can hold a "Design Lock" on a design element at any given time
    - "Coarse Grained" in the sense that an entire design element is shared out
  - For team development, it requires a centralized "Masterlock" server for lock management
    - Offline development involves "Provisional" locking which can increase the risk of loosing important code changes when subsequently coming back online
  - Code round-tripping is restricted in a team environment when multiple developers are working on the same centralized application
    - Unproductive for compiled resources such as XSP, Custom Java, and Managed Bean class files resulting in unstable application behavior during code round-tripping

Classic Design Source Control

Application Properties - DDB 123

Design Properties

Inheritance

- [x] Inherit design from master template
  - Template name: StdR9Discussion
  - [ ] Refresh design on admin server only
  - Template version is 9.0 (14/12/2012)
- [ ] Database file is a master template
  - Template name:

Options

- [x] Allow design locking
- [x] List in Database Catalog
  - Categories: Discussion
- [x] Show in 'Open Application' dialog
- [ ] Include in multi-database indexing
- [ ] Do not mark modified documents as unread
- [ ] Mark parent note on reply or forward

Basics | Launch | Design | XPages | Advanced

Access Control List to: DDB 123

Basics
Roles
Log
Advanced

Administration server (Master Lock Server)

- ( ) None
- (•) Server: scorpio/renovations

Action: Do not modify Names fields

connect\DDB123.nsf - XPages | allDocuments - XPage

New XPage | Sign

Name	Last Modified	Last Modified By
allDocuments	10/01/2013 09:32:10	Tony McGuckin/Ireland/IBM
authorProfile	14/12/2012 13:22:01	Lotus Notes Template Devel...
byAuthor	14/12/2012 13:22:01	Lotus Notes Template Devel...

Locked by you

XPages Design Source Control “Share Model”

- Predicated on the principles of “**Copy** – **Modify** – **Merge**”
  - Multiple developers can take a copy of a design element at any given time
    - “Fine Grained” in the sense that a copy is taken
      - Subsequent changes are merged back into the base version on a line by line basis
      - Automatically by default, Manually in the case of possible conflicts
  - For team development, it requires a centralized “Source Control” server
    - This is not a Domino server – it is a Source Control server such as Subversion
    - Offline development is unrestricted and does not increase the risk of loosing important code changes when subsequently coming back online as committing changes is done so via fine-grained line-by-line merging

XPages Design Source Control

- Predicated on the principles of "**Copy** – **Modify** – **Merge**"
  - Code round-tripping is unrestricted as developers can do so in their own local preview or standalone development servers
    - Highly productive for compiled resources such as XSP, Custom Java, and Managed Bean class files which avoids instability's in application behavior during code round-tripping
  - Individual developers can commit changes back to the base version in the Source Control server with the added protection of a managed merge process
    - Avoiding code change loss through a decision based merge process
  - A "built" application can be created or updated from the Source Control server base version code at any point in time for sharing with test or deployment teams
    - A "Gatekeeper" should be assigned within the development team to moderate incoming changes into the "built" application

XPages Design Source Control

- Predicated on the principles of "**Copy** – **Modify** – **Merge**"
  - Powerful utilities are provided by the Source Control system allowing developers to take decisive, informed decisions during the development lifecycle
    - Complete Change History of a design element
      - Line by line detail of actual code changes
      - Line by line comparisons between historical versions
      - Rebasing to any particular version in history
      - Modifier/date/time/commentary details for audit trail purposes
  - Failover and backup utilities provide administration with reliability
    - Backup snapshots can be taken at any point in time, even from a live server

Domino Designer Source Control Integration

- Source Control support provided as of Domino Designer version 8.5.3+
  - Domino Designer 8.5.3, and 9.0 are built upon ***Eclipse 3.4.2 / Ganymede***
  - UI features are provided out of the box in Domino Designer for source control integration
    - Developer must install the chosen Source Control system client integration plugins
      - This fully integrates the functionality of that particular system
    - Chosen Source Control integration plugins must be ***Eclipse 3.4.2/Ganymede*** compatible!
  - Apache Subversion project provides numerous freely available, open source implementations
    - Eclipse Subversive, CollabNet, ..., to name two!
      - Eclipse 3.4.2/Ganymede compatible versions are readily available

Domino Designer Source Control Integration

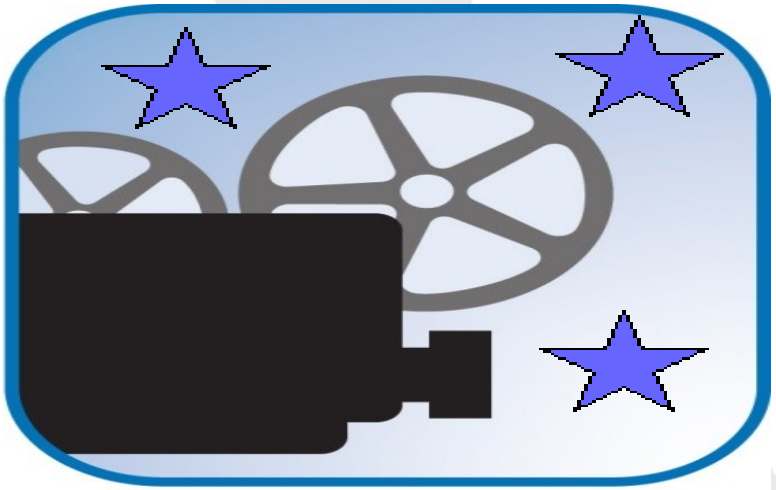
- Source Control support provided in Domino Designer version 8.5.3+
  - Typical setup:
    - *CollabNet Subversion Edge Server*
      - Centralized server providing source control management and storage
      - http://www.collab.net/products/subversion

SubversionEdge.
SMARTER SCM MANAGEMENT

    - *Eclipse Subversive*
      - Client integration / connector plugins installed into Domino Designer
      - http://www.eclipse.org/subversive/
      - http://www.eclipse.org/downloads/download.php?file=/technology/subversive/0.7/builds/Subversive-incubation-0.7.8.I20090904-1300.zip
      - http://community.polarion.com/projects/subversive/download/eclipse/2.0/builds/Subversive-connectors-2.2.0.I20090515-1900.zip

SUBVERSIVE

BLUG
Belux Lotus User Group

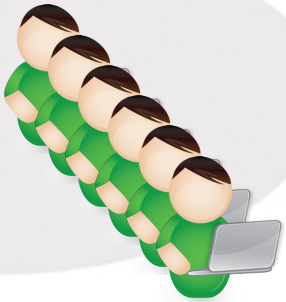


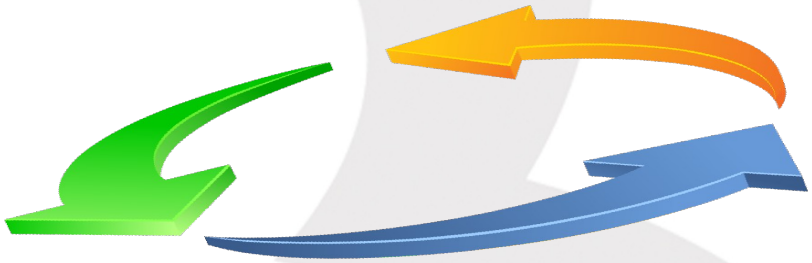
XPages Team Based Development Model

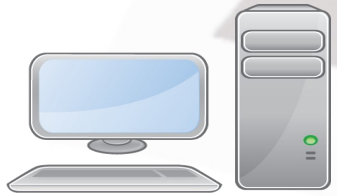
Developers

SVN Server

1. Developers and "Gatekeeper" develop and test in their local clients/servers with On-Disk projects associated with the SVN server code-base

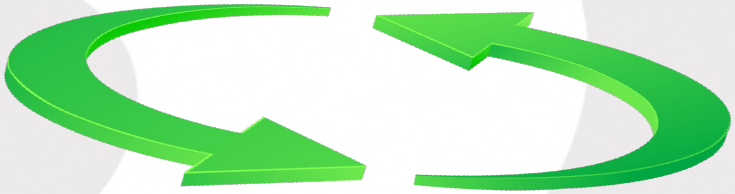




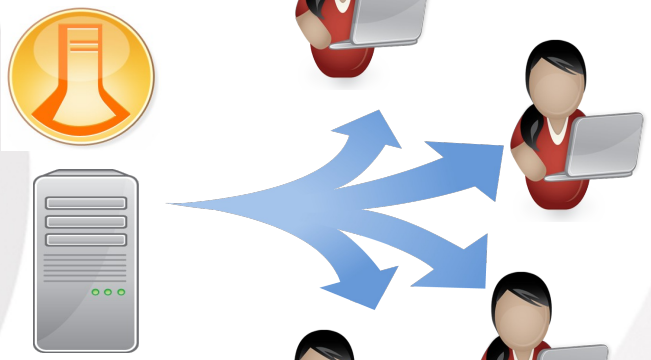


2. Gatekeeper has a local replica of the Domino server database c/w an On-Disk project associated with the SVN server code-base

"Gatekeeper"



Domino Server



3. Gatekeeper synchronizes their On-Disk project with the SVN server and moderates incoming changes (accept/reject/modify) before replicating with the Domino server application on an agreed scheduled basis

4. QE / End-users test against the Domino server application knowing its the "built" version c/w latest "approved" development code

Agenda

- Speaker Introduction
- Session Goals
- Team Based Application Development
- Application Testing
- Tuning, Profiling & Debugging
- Deployment Considerations
- Wrap Up

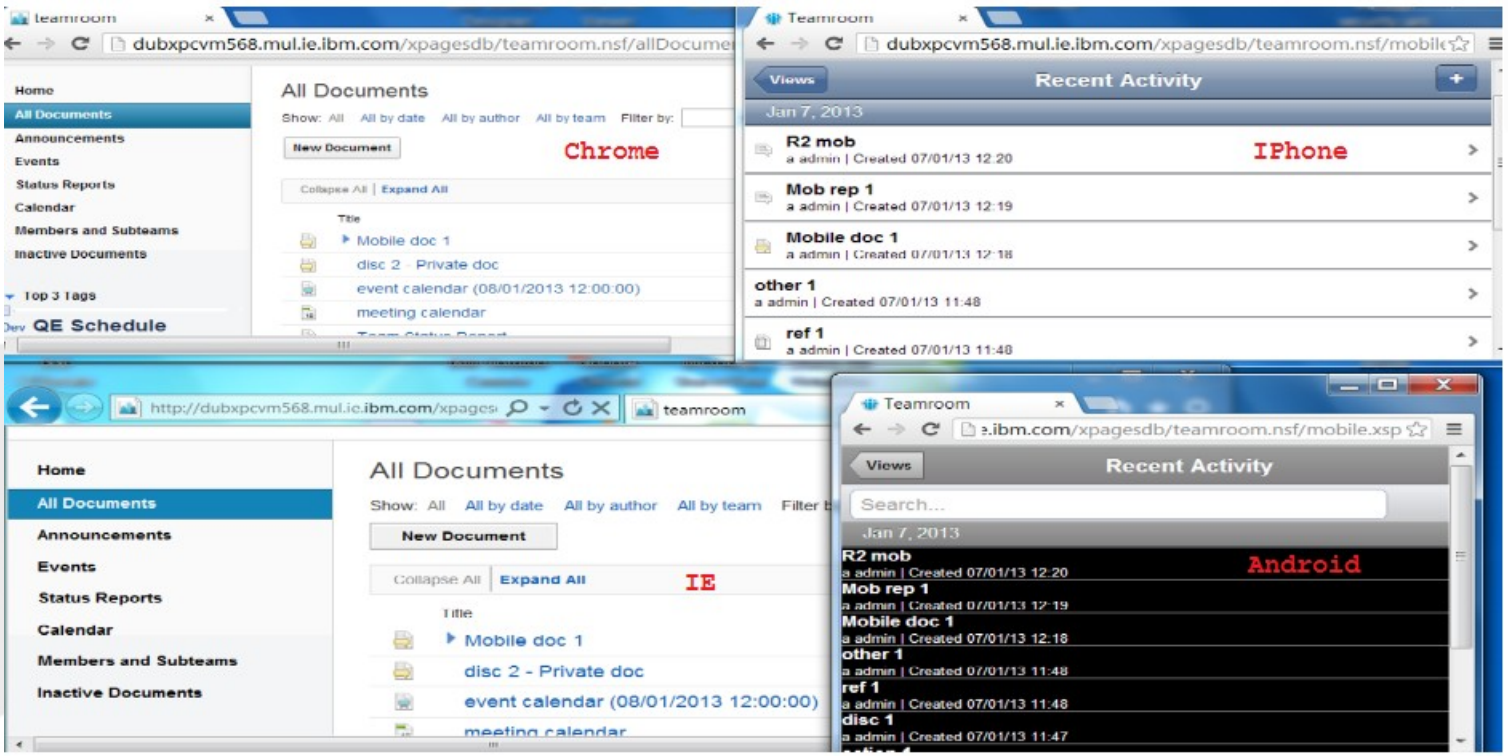
Automated Browser-Driven Testing

When your application is available for functional testing on the web and mobile devices

- Testing across multiple browsers, phones and tablets
- Selenium Overview
- Creating a Selenium Test for an XPage
- Creating a TestNG suite
- Demo
- Pitfalls and Best Practices

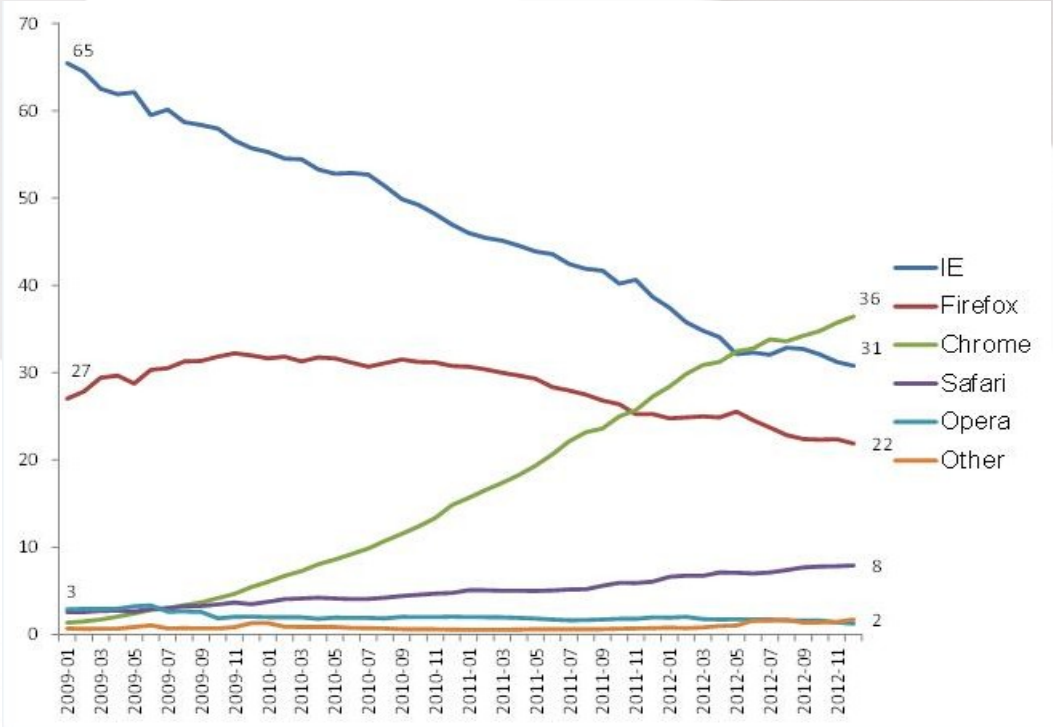
Testing across multiple browsers, phones and tablets

- Does your control render correctly on all browsers ?
- Does the functionality match what you expect on all browsers?
- Mobile Browsers usage is on the increase, is your application ready?



Selenium Overview

- Selenium automates Web browsers
- Selenium 2.0
  - New WebDriver API
  - Modern Browser Support
    - Chrome
    - IE
    - Firefox
    - And more
  - Mobile support
    - Android
    - IOS
- Check http://seleniumhq.org/ for latest browser support



Source:http://gs.statcounter.com

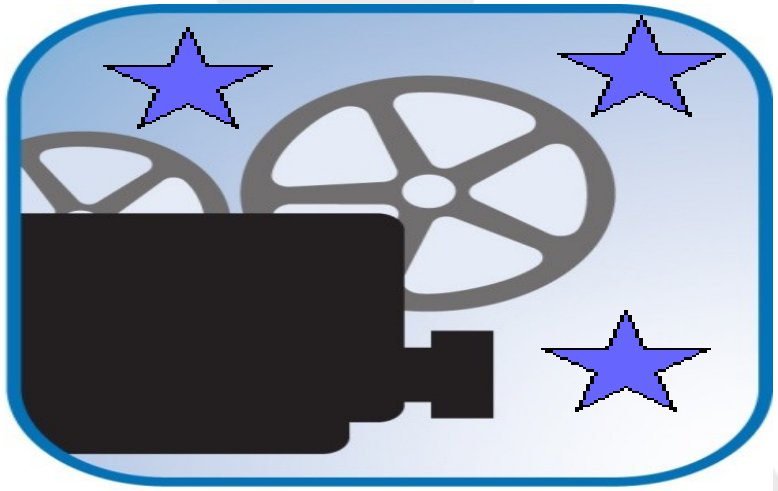
BLUG
Belux Lotus User Group

Creating a Selenium Test for an XPage

- A selenium test can be written in a number of programming languages such as Python, Ruby, Javascript and Java
- Create a WebDriver
  - Launches Browser Instance and drives automation
  - ChromeDriver, FirefoxDriver and more
- Find WebElements on the page using Selenium by class
  - XPath
  - cssSelector
  - linkText
  - and others...
- Call methods on WebElements
  - Click a Button
  - Retrieve text value of html element
- Perform an assertion on a predicted outcome

Create a Example TestNG Test Suite

- TestNG is a test framework used to run tests and report results
- Create a TestNG Test Suite Java class with a method to instantiate and run your Selenium Java class
- Include TestNG annotation (@Test) to mark your method as a TestNG test
- Create TestNG test suite XML file referencing the TestNG method
- Provides results in HTML and XML format. Results can be customised to your own format and layout etc.
- Can be configured to take screenshots when a test fails
- TestNG Annotations
  - BeforeClass
  - BeforeMethod
  - AfterMethod
  - BeforeMethod



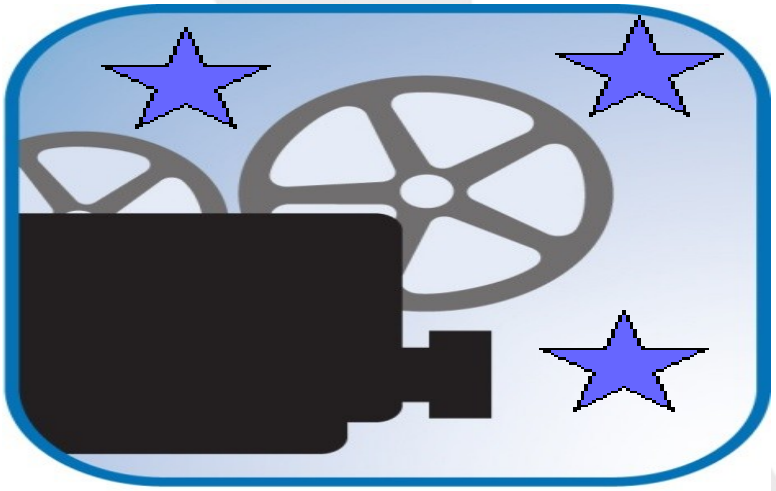
JUnit Overview

JUnit is a unit testing framework for the Java programming language

- Unit Tests ensure Java code is working as intended
  - Create an Object
  - Invoke a method
  - Check the result
- Provides a mechanism to make assertions about the state of your objects
  - AssertTrue – fails if condition is false; passes otherwise
  - AssertEquals – fails if expected and actual objects are not equal
  - And others...
- JUnit plugin is part of IBM Domino Designer ®
- Provides predictable test cases that the XPages team have created to test the core and extension library Java controls
- Extendable for your own specific test needs

Testing a new Java control with XPages JUnit Framework

- Create a new test plugin which depends on JUnit and XPages JUnit Framework plugins as well as core runtime plugins
- Create a TestSuite class and specify test cases to run
- Begin with the BaseGeneratePagesTest generating the .java files from .xsp files
- Create a config.properties file in com.ibm.xsp.test.framework package
  - target.library
  - NamingConvention.package.prefix
  - extra.library.depends.designtime.nonapplication
- Create a xsp.properties in WEB-INF folder
- Create JUnit tests that extend the base tests



Agenda

- Speaker Introduction
- Session Goals
- Team Based Application Development
- Application Testing
- Tuning, Profiling & Debugging
- Deployment Considerations
- Wrap Up

Tuning, Profiling & Debugging

- XPages Toolbox – Profiling Application
  - Runs on the Domino server or the Notes client
  - Provides detailed analysis per request on CPU, Memory, Time costs
- Domino Designer – Code Debuggers
  - ServerSide JavaScript Debugger
  - Java Debugger
- Client/Server Logging
  - JVM Logging (can be controlled using the XPages Toolbox)
- Request Introspection, print, _dump Techniques
  - Introspection techniques can be utilized using lifecycle PhaseListeners
  - print, _dump Techniques can be incorporated into application code

BLUG
Belux Lotus User Group

XPages Toolbox

- XPages Toolbox – Profiling Application
  - Runs on the Domino server or the Notes client
  - An NSF needs to be installed on the Domino server/Notes client
  - A profiler jar file should be added to the JVM launch options
  - JVM Security policy updates
- Profiles and Measures
  - CPU performance, Memory allocation, Threads, Sessions, Backend Activity
  - Profiling API can also be added into ServerSide JavaScript and/or Custom Java
- Available from OpenNTF.org
  - Free open source project
  - Search for “XPages Toolbox” (*Most recent version is 1.2*)

Request Introspection, print, _dump Techniques

XPages Request Processing Lifecycle

Test XPage Request

Test View Events
Test Simple Lifecycle
Test Process Events

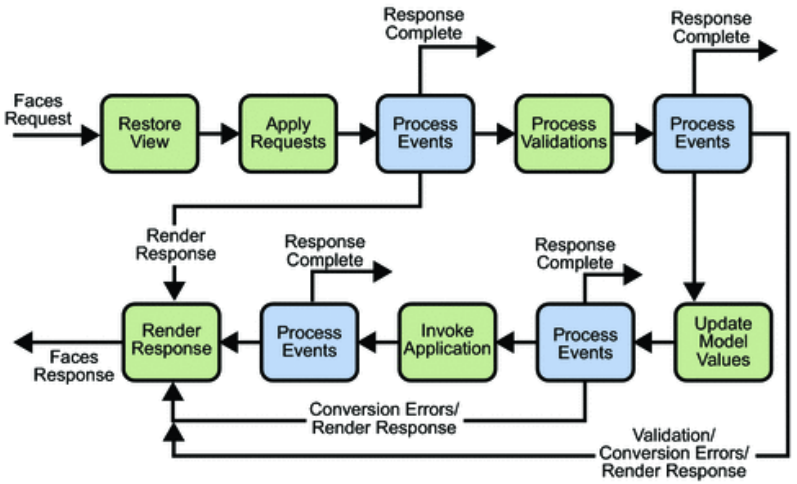
Test Custom Converter
Test Java Converter
Test Process Validations Phase

Test ValueChangeListener
Test ActionListener

Test OnClientLoad Event
Test Inline Events
Test Open Page Simple Action

Test Advanced Lifecycle

Test Rendered 1
Test Rendered 2
Test Loaded And Rendered
Test Create View
Test Create View Repeats
Test Create View Custom In Repeat
Test Create View Client IDs
Test DynamicLoading 1
Test DynamicLoading 2
Test DynamicLoading 3



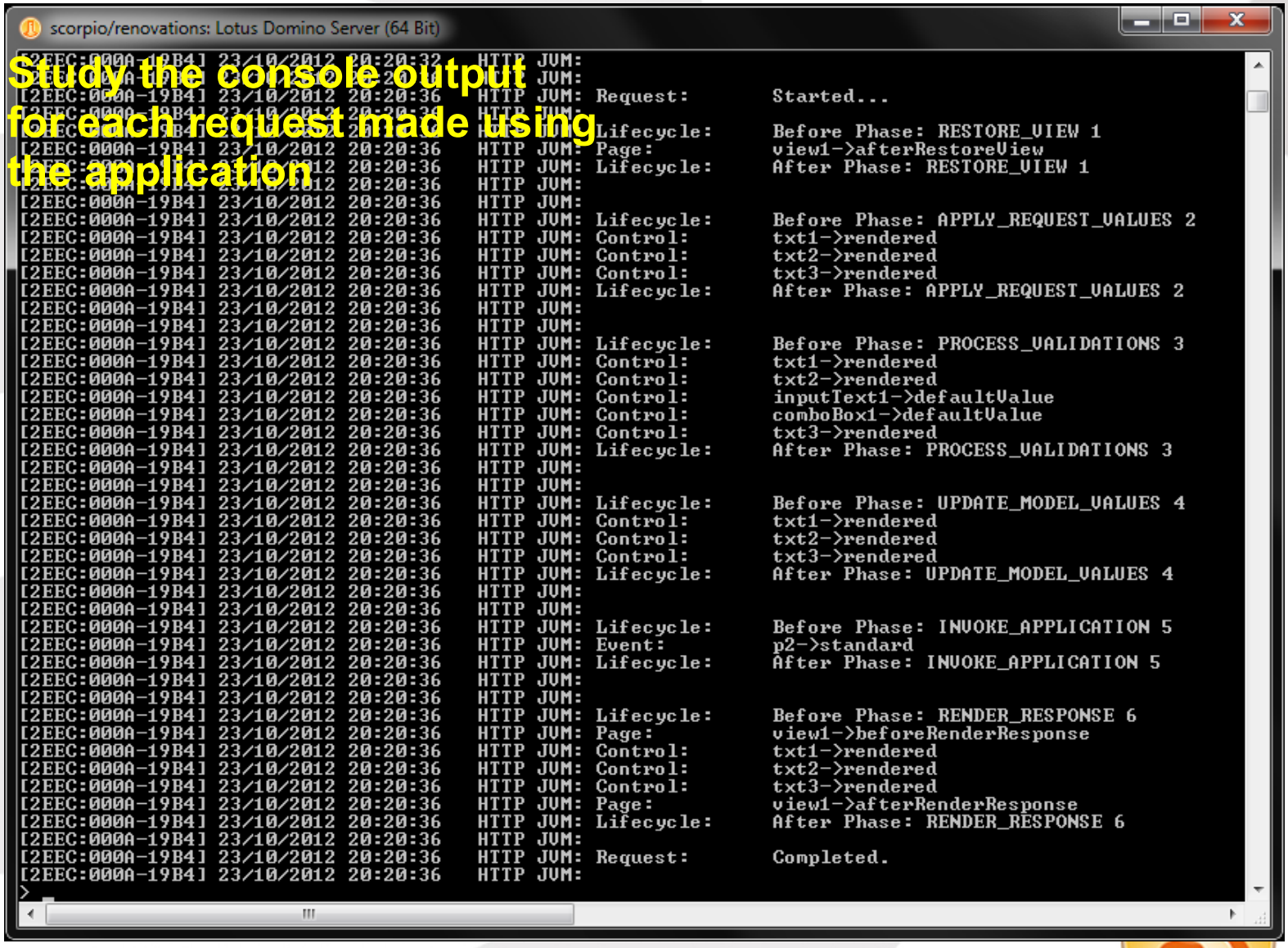
Debug Mode

☑ Debug Mode

Apply

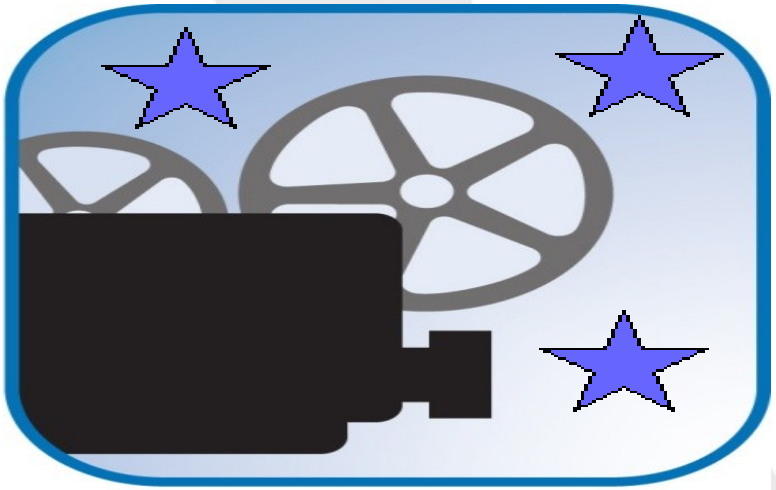
Check this checkbox to enable debug mode

Request Introspection, print, _dump Techniques




```
[2EEC:000A-19B4] 23/10/2012 20:20:32   HTTP JVM:
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM:
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Request:       Started...
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM:
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Lifecycle:     Before Phase: RESTORE_VIEW 1
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Page:          view1->afterRestoreView
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Lifecycle:     After Phase: RESTORE_VIEW 1
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM:
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM:
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Lifecycle:     Before Phase: APPLY_REQUEST_VALUES 2
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Control:       txt1->rendered
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Control:       txt2->rendered
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Control:       txt3->rendered
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Lifecycle:     After Phase: APPLY_REQUEST_VALUES 2
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM:
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM:
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Lifecycle:     Before Phase: PROCESS_VALIDATIONS 3
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Control:       txt1->rendered
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Control:       txt2->rendered
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Control:       inputText1->defaultValue
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Control:       comboBox1->defaultValue
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Control:       txt3->rendered
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Lifecycle:     After Phase: PROCESS_VALIDATIONS 3
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM:
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM:
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Lifecycle:     Before Phase: UPDATE_MODEL_VALUES 4
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Control:       txt1->rendered
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Control:       txt2->rendered
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Control:       txt3->rendered
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Lifecycle:     After Phase: UPDATE_MODEL_VALUES 4
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM:
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM:
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Lifecycle:     Before Phase: INVOKE_APPLICATION 5
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Event:         p2->standard
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Lifecycle:     After Phase: INVOKE_APPLICATION 5
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM:
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM:
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Lifecycle:     Before Phase: RENDER_RESPONSE 6
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Page:          view1->beforeRenderResponse
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Control:       txt1->rendered
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Control:       txt2->rendered
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Control:       txt3->rendered
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Page:          view1->afterRenderResponse
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Lifecycle:     After Phase: RENDER_RESPONSE 6
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM:
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM: Request:       Completed.
[2EEC:000A-19B4] 23/10/2012 20:20:36   HTTP JVM:
>
```

BLUG
Belux Lotus User Group



Agenda

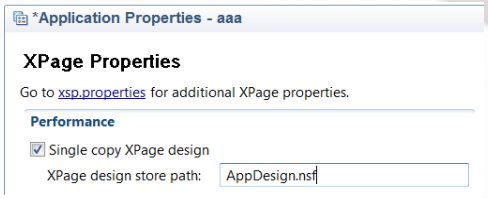
- Speaker Introduction
- Session Goals
- Team Based Application Development
- Application Testing
- Tuning, Profiling & Debugging
- Deployment Considerations
- Wrap Up

Deployment Considerations - Technical challenges

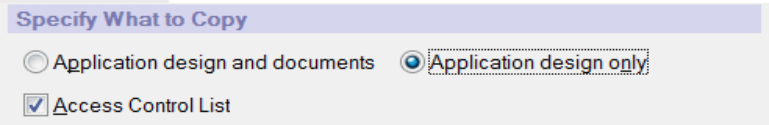
- What is your application usage profile ?
  - Few applications, many users
  - Many applications each with a small number of users
  - A mix of the above...
- Can your applications be tuned to scale and perform well for different profiles ?
- Does the application use extended features like the XPages Extension Library ?
  - Do you also need to use experimental features from OpenNTF ?
  - Can you mix and match extended and experimental feautures
- How do you resolve issues after production applications are deployed ?
  - Can you debug your production application remotely ?
  - Do you know where to find your XPages logs and know how to interpret them?
- Does the application need to run on the Domino Server and Notes client ?
  - If running on Notes client, is it purely for convenience of offlining web apps … or
  - Do you want a fully optimized Notes client XPages application?

Deployment Considerations: Application Usage Profile

- Use Case 1: Lots of application instances based on the same template
  - e.g. a billing template used by all company consultants for every corporate client
    - Individual application instances per customer

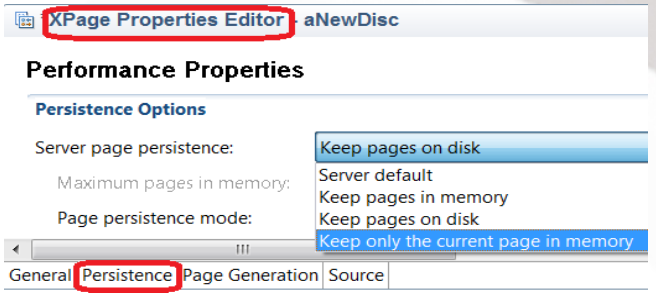


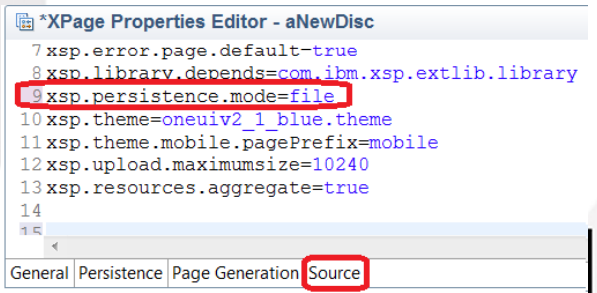
- Potential Issues
  - Each application includes the same design artifacts
    - Redundancy and scalability problems
  - Rolling out design updates is difficult due to the proliferation of NSFs
- Consider the Single Copy XPages Design (SCXD) Feature
  - Create a design-only copy of your application
    - Must be an NSF file, not an NTF
    - Must have a unique name, not a duplicate of any application instance
    - Only applies to XPages design elements and resources
  - Deploy SCXD NSF to Domino server and Notes client, as needed



Deployment Considerations: Application Usage Profile

- Use Case 2: Application used by many concurrent users
  - Application needs to scale well in order to support many users
- Options
  - Consider which XPages persistence options
    - a) Keep pages in memory
    - b) Keep pages on disk
    - c) Keep only the current page in memory
  - Choose the best fit for your usage profile
    - Options b) and c) scale best
    - Option a) is best for performance, e.g. few users using NSF
  - Look at the Java heap size allocated to the Domino server HTTP task
    - NOTES.INI
      - HTTPJVMMaxHeapSizeSet=1
      - HTTPJVMMaxHeapSize=256M
    - The Java memory allocation for the HTTP task
    - 256MB recommended on 32bit servers
    - 1024MB on 64bit servers





Deployment Considerations: Other xsp.properties options

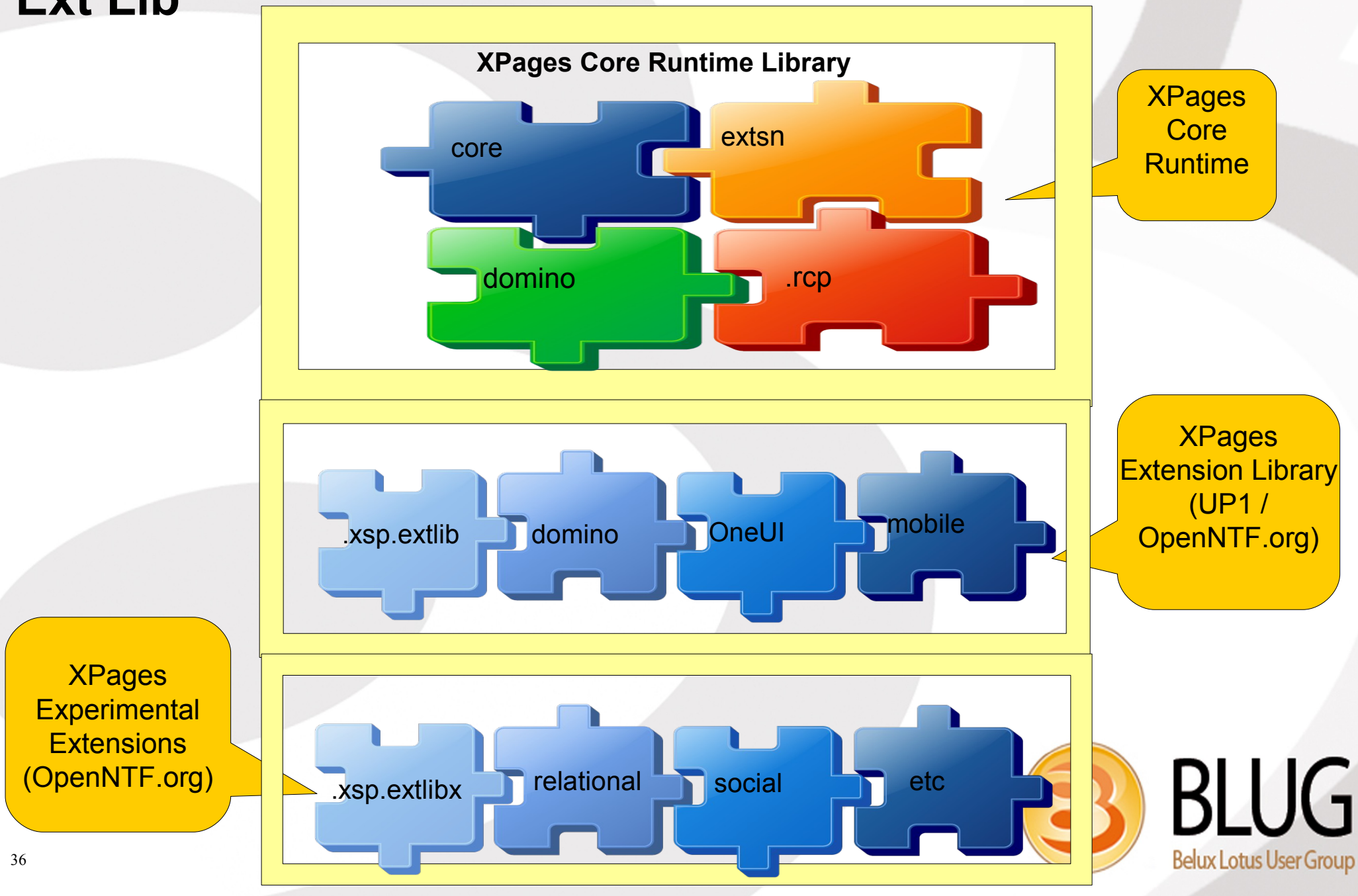
xsp.properties file contains settings to control XPages runtime behaviors

Can be applied on a server-wide, per app, per page or per request basis

Some other useful examples are summarized as follows:

- More persistence options
  - xsp.persistence.file.maxviews,
  - xsp.persistence.dir.xspstate
- Server timeouts
  - xsp.application.timeout=30min
  - xsp.session.timeout=30min
- Network Payload Management
  - xsp.compress.mode=gzip - network files are smaller (enabled by default)
  - xsp.resources.aggregate - means fewer requests for CSS, JS and image files
- Browser expiration for web resources (CSS, JS, images etc)
  - xsp.expires.global=10days

Deployment Considerations: XPages Core, UP1 and Ext Lib



BLUG
Belux Lotus User Group

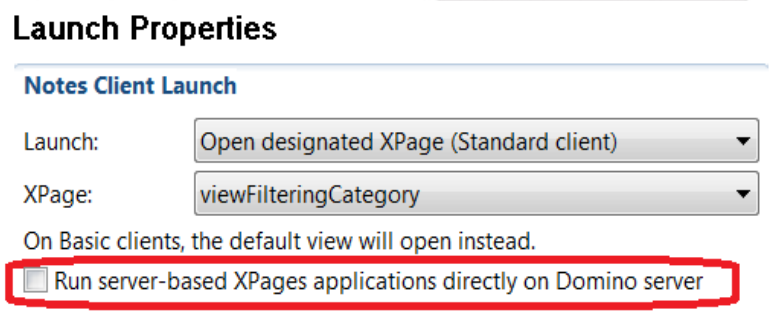
Deployment Considerations: XPages Core, UP1 and Ext Lib

Managing your XPages runtime configuration

- How can the XPages Upgrade Pack and Extension Library co-exist?
  - e.g. I have deployed XPages 8.5.3 plus the latest Extension Library
    - How do I upgrade to UP1 ?
      - You must first uninstall the Extension Library
      - Then run the UP1 add-on installer
    - Can I use OpenNTF experimental components (e.g. RDBMS) with UP1 ?
      - No … at least, not all builds
    - How do I upgrade 9.0 SE ?
      - Uninstall the Extension Library
      - Run the 9.0 SE installer
  - Can I install Upgrade Pack 1 into 853 with Fix Pack releases ?
  - I have XPages 8.5.3 UP1, How do I upgrade 9.0 SE ?
  - Can Upgrade Packs and OpenNTF releases be interchangeable in future?

Notes/Domino 9.0
Deployment Performance Considerations

- XPages applications can be preloaded on both Notes client and web server
- New Single Copy XPages Design option
  - Combine Single Copy XPages Design with Preload for production applications
  - Deploying a local design-only SCXD NSF means all web resources are resolved locally
  - Form design elements required by **computeWithForm** logic resolved locally
  - Reduced network traffic from SXCD + Preload greatly enhances XPiNC performance !!!
- New XPiNC RunOnServer option if Notes optimizations are not a priority



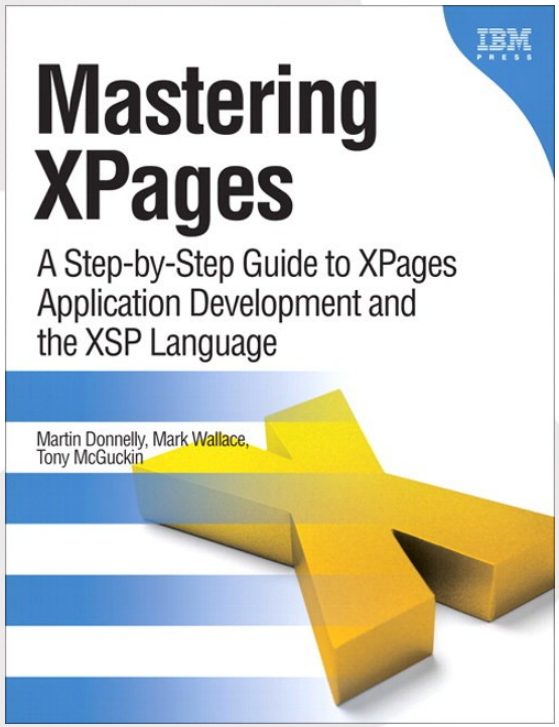
BLUG
Belux Lotus User Group

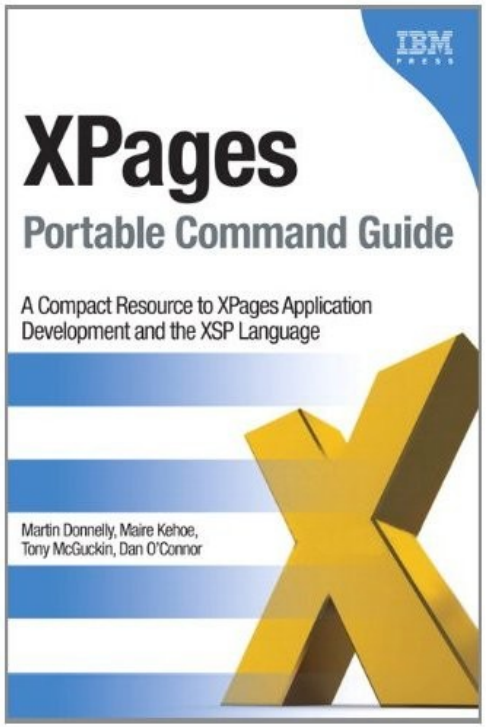
Agenda

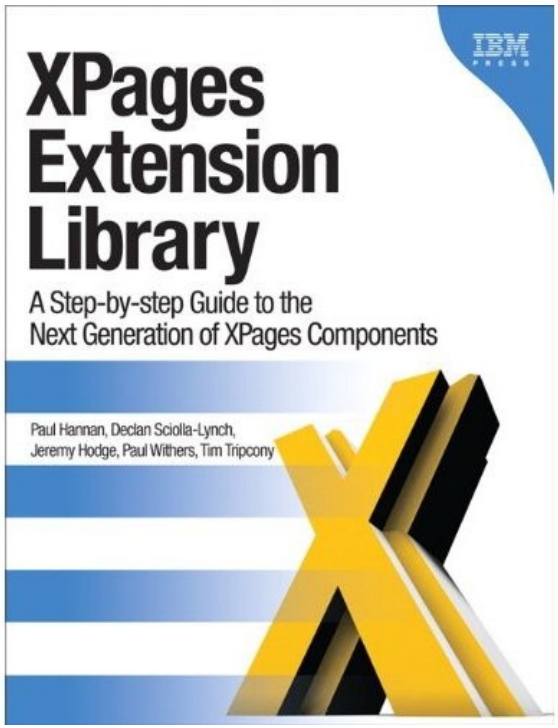
- Speaker Introduction
- Session Goals
- Team Based Application Development
- Application Testing
- Tuning, Profiling & Debugging
- Deployment Considerations
- Wrap Up

Technical Education

- IBM Press Books and eBooks
  - Three major publications over the past two years
  - All available for evaluation in the bookstore in the Solutions Expo







BLUG
Belux Lotus User Group

Q & A

Contact Information:
martin_donnelly@ie.ibm.com
peter_janzen@us.ibm.com

BLUG
Belux Lotus User Group

Backup Slides

Selenium Slides

BLUG
Belux Lotus User Group

Pitfalls and Best Practices

- Selenium Grid - Allows to scale for large test suites and multiple environments
- Selenium doesn't support browser dialogs. AutoIT and java.awt.Robot can be used to workaround this
- IE6 and IE7 can be the most problematic
- When dealing with iframes be mindful at what level on the page the webDriver is running.
  - webDriver.switchTo().defaultContent();
  - webDriver.switchTo().frame(String frameId);
- Trick the browser into believing you are running in a mobile browser by using User-Agent manipulation
  - Android - Mozilla/5.0 (Linux; U; Android 4.0.3; ko-kr; LG-L160L Build/IML74K) AppleWebkit/534.30 (KHTML, like Gecko) Version/4.0 Mobile Safari/534.30
  - IOS - Mozilla/5.0 (iPhone; CPU iPhone OS 5_0 like Mac OS X) AppleWebKit/534.46 (KHTML, like Gecko) Version/5.1 Mobile/9A334 Safari/7534.48.3

Selenium Test

- Connect13Example.java – Example XPages Selenium Test

Connect13Example.java

```
1  package xpages.connect13.example;
2  import org.openqa.selenium.By;
7
8  public class Connect13Example {
9
10     //Server & Application paths
11     private String server = "http:\\localhost\\";
12     private String application = "discuss.nsf\\simpleXPageTest.xsp";
13
14     //WebDriver object
15     private WebDriver webDriver;
16
17     @Test(description="Connect app test")
18     public void runConnectTest() {
19
20         //Set the path to the Chrome driver before we create it
21         System.setProperty("webdriver.chrome.driver", "c:\\Selenium\\chromedriver.exe");
22
23         //Create the webDriver
24         webDriver = new ChromeDriver();
25
26         //Load the XPage
27         webDriver.get(server+application);
28
29         //Find the button on the page using an xpath locator
30         WebElement theButton = webDriver.findElement(By.xpath("//button[@id='view:_id1:button1']"));
31         //click it
32         theButton.click();
33
34         //Find the computed field on the page, using a cssSelector locator this time.
35         WebElement theCompField = webDriver.findElement(By.cssSelector("span[id*='computedField1']"));
36
37         //And get the text value it contains
38         String actualResult = theCompField.getText();
39
40         //Verify the result is as expected
41         String expectedResult = "World";
42         assert(actualResult.equals(expectedResult)):"Verify computed field value is '" + expectedResult + "'. Found '" + actualResult + "'.";
43
44         webDriver.close();
45     }
```

Sample Selenium XPage

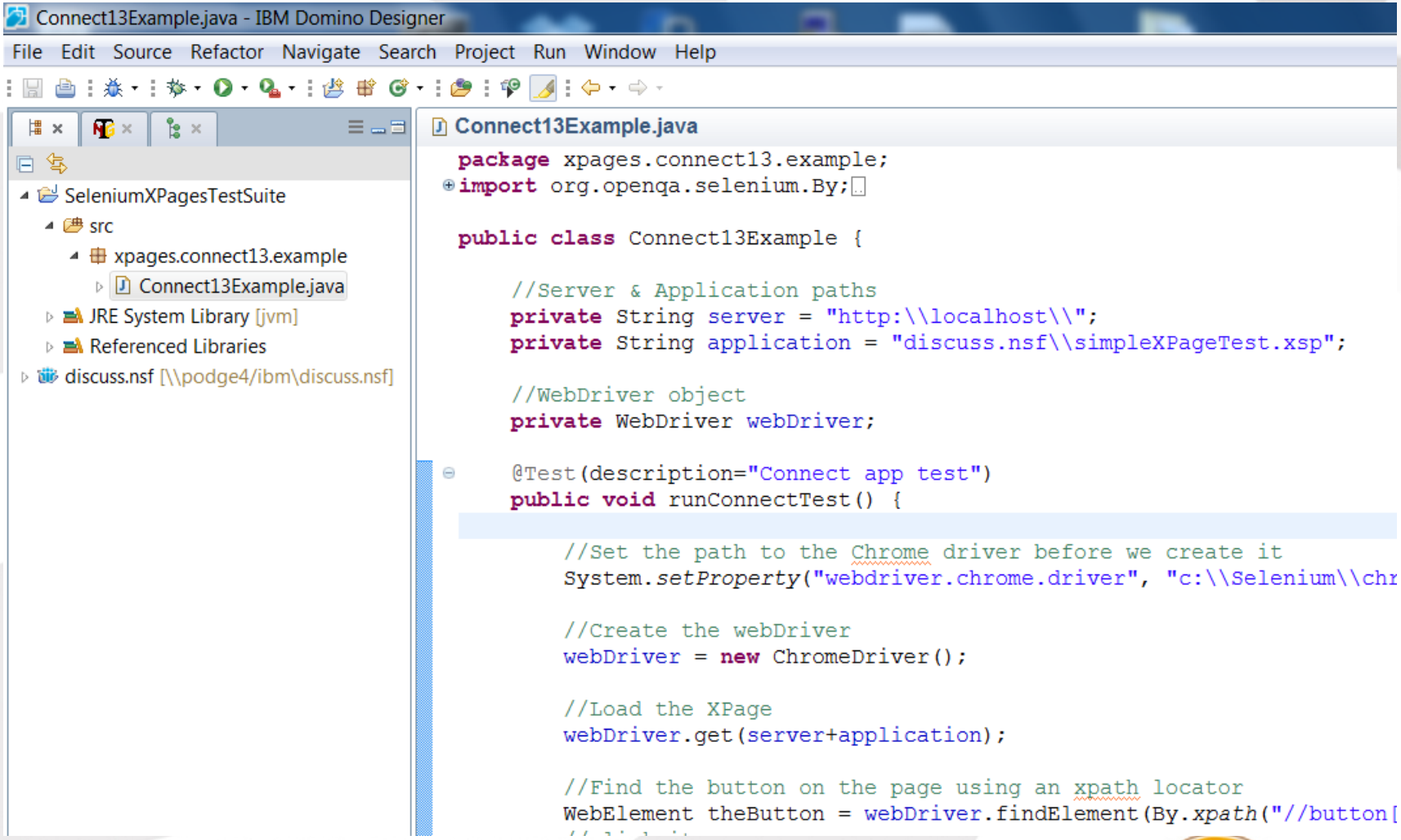
- Sample XPage – Used for simple Selenium Test to grab a value from a computed field and check it matches what is expected

simpleXPageTest - XPage

```
<?xml version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core">
    <xp:br></xp:br>
    <xp:button value="Click me" id="button1">
        <xp:eventHandler event="onclick" submit="true"
            refreshMode="partial" refreshId="panel1">
            <xp:this.action>
            <![CDATA[#{javascript:
                        var value=getComponent("computedField1").getValue();
                        if(value == "<h1>Hello</h1>")
                            getComponent("computedField1").setValue("<h1>World</h1>");
                        else
                            getComponent("computedField1").setValue("<h1>Hello</h1>");  }
            ]]>
            </xp:this.action>
        </xp:eventHandler>
    </xp:button>
    <xp:br></xp:br>
    <xp:panel
        style="border-color:rgb(0,128,255);border-style:solid;border-width:medium"
        id="panel1">
        <xp:text escape="false" id="computedField1">
            <xp:this.value><![CDATA[<h1>Hello</h1>]]>
        </xp:this.value>
        </xp:text>
    </xp:panel>
    <xp:br></xp:br>
</xp:view>
```

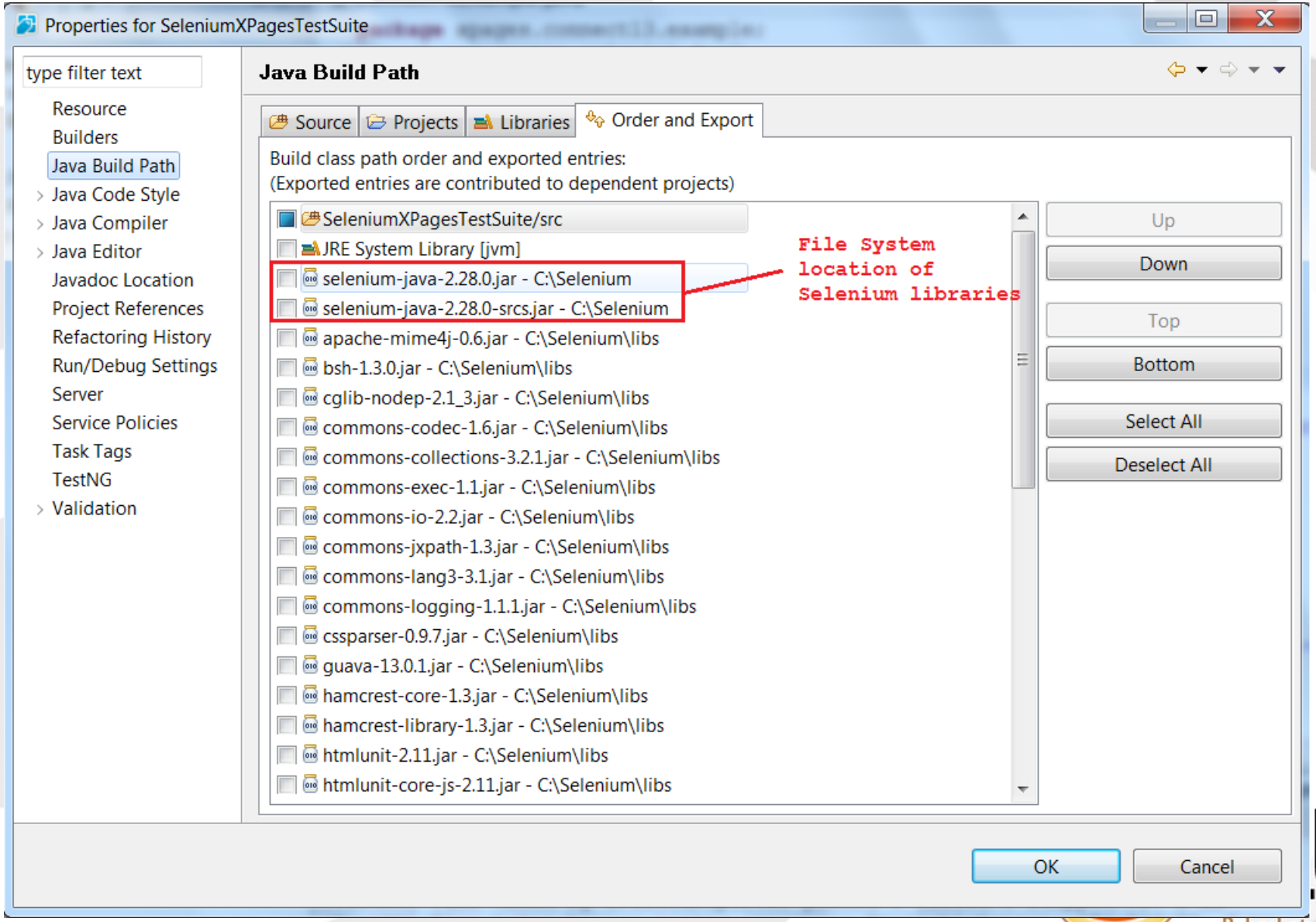
SeleniumXPagesTestSuite Project

- Example Selenium Project open in the Java perspective



XPagesSeleniumTestSuite Project Build Path

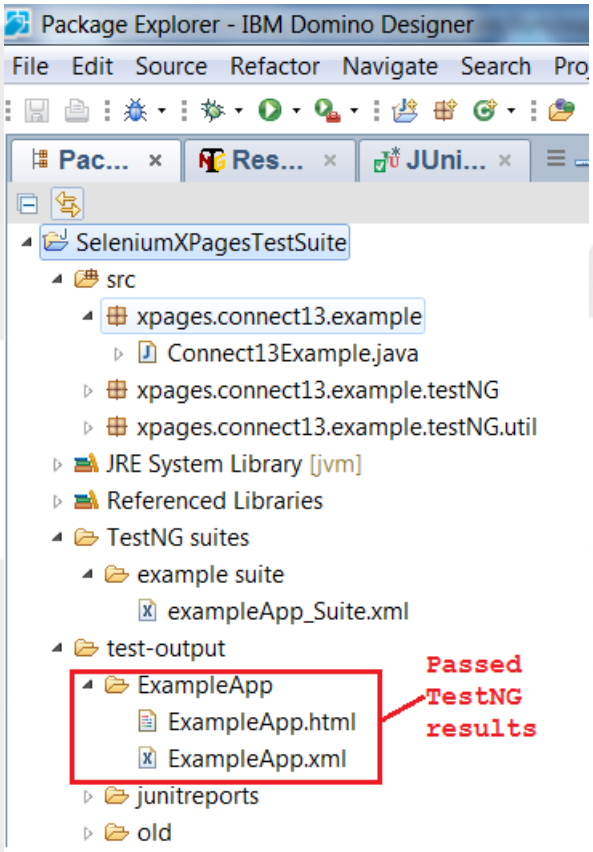
- Shows the build path configuration pointing to the Selenium Libraries



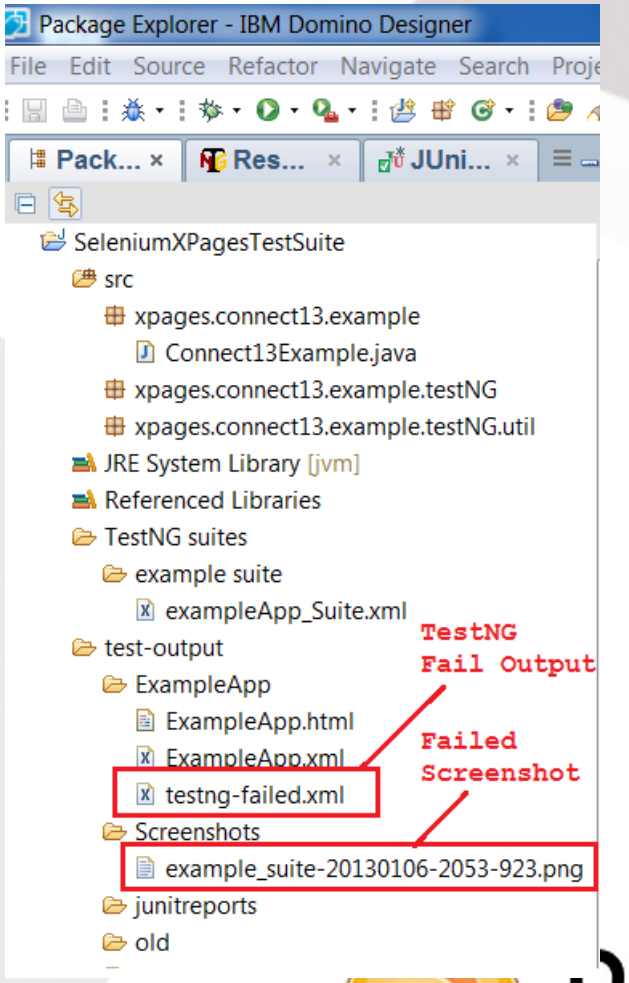
TestNG Test Output

- Test Output from a passed and failed Selenium test

Passed TestNG Output

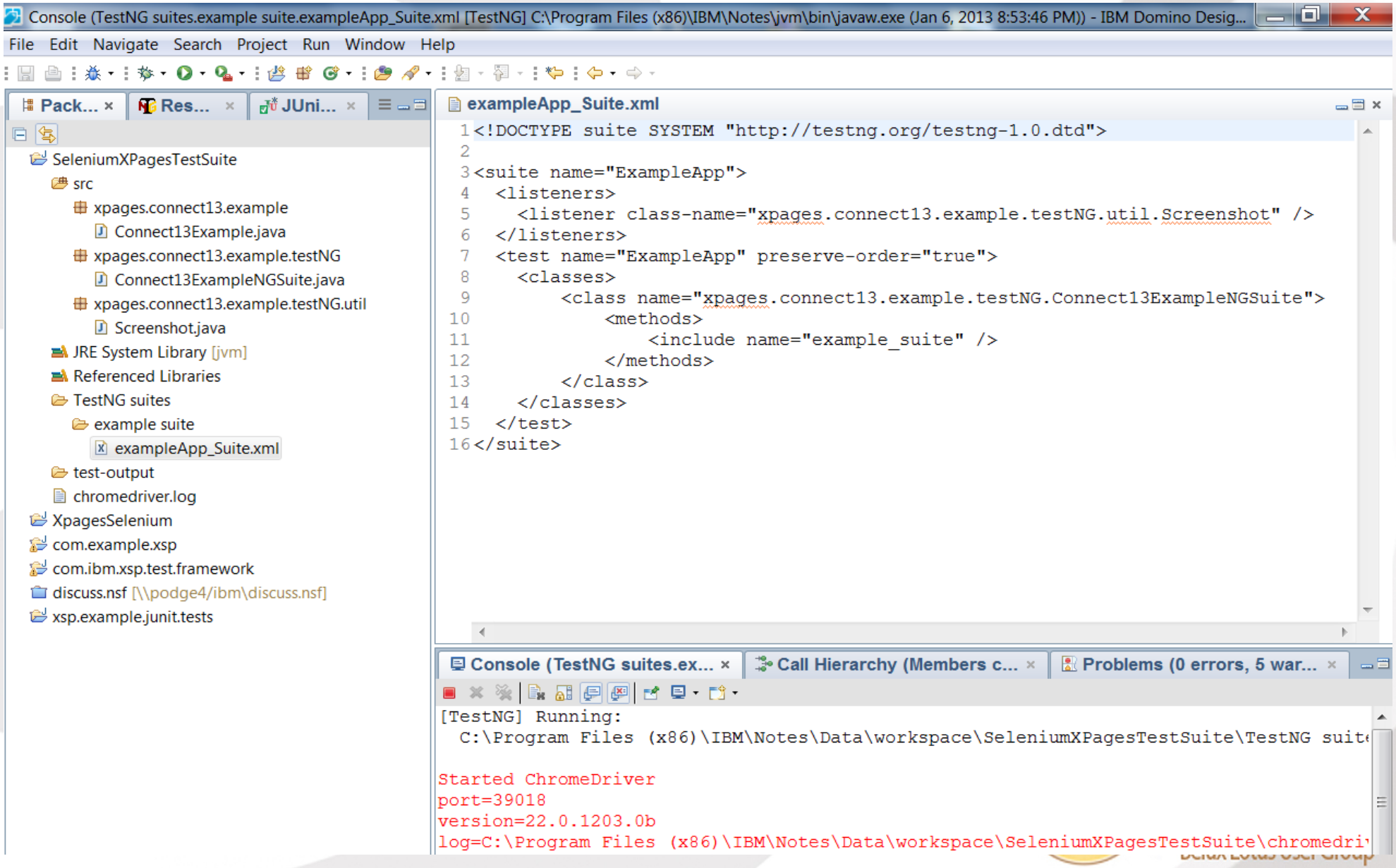


Failed TestNG Output



Selenium + TestNG Project

- SeleniumXPagesTestSuite with added TestNG example suite



JUnit

XPages JUnit Framework

- Provides predictable test cases that the XPages team have created to test the core and extension library Java controls
- Suite of test cases that can be applied to your library of controls
  - Verify xsp-config files can be parsed
  - Verify set methods correspond to property names
  - Verify controls can be serialized
  - And others...
- Utility to generate .java files from .xsp files
- Utilities for creating the JSF control tree and rendering HTML to the .xsp file
- Extendable for your own specific test needs
- Ideally requires some knowledge of the xsp-config file format
  - http://www-10.lotus.com/ldd/ddwiki.nsf/dx/XPages_configuration_file_format

Testing a new Java control with XPages JUnit Framework

- BaseLabelsLocalizedTest is for testing a predefined list of properties which should always be localizable when defined in a control
  - label
  - title
  - alt
  - and others..
- Add new “mainTitle ” property that we want to be localized
- Add new “title” property that we don't want to be localized in this test case
- Extend BaseLabelsLocalizedTest as new ExampleLabelsLocalizableTest
- Add new localizable property (“mainTitle”) to list of properties to be tested as localizable
- Change xsp-config to omit “title” property as localizable

Pitfalls and Best Practices

- Start to run JUnit tests from the beginning of development cycle even if you haven't written any yet!
- Always begin by generating .java files if testing .xsp files

```
//test that generates .java files from .xsp files
// at the start, as other tests depend on it to pass.
// - BaseGeneratePagesTest
suite.addTestSuite(BaseGeneratePagesTest.class);
```

- Do not check in the contents of the gen/ folder to source control, though you can check in the gen/ folder itself.
- Don't edit the test framework project itself. Create a subclass and update the behavior there
- If a test fails, read the test description and read the failure message. If needs be debug into source code itself.
- Learn from existing ExtLib tests in .control package available on OpenNTF

XPages Example JUnit Suite

- ExampleTestSuite to run the base set of tests from the XPages JUnit test framework

*ExampleTestSuite.java

```
1  package xsp.example.test;
2
3  import java.util.List;
69
70 public class ExampleTestSuite extends TestSuite{
71
72  public static final long SUITE_VERSION = 39;
73
74     public static List<Class<?>> getTestClassList() {
75         TestClassList suite = new TestClassList();
76
77         // Put the test that generates .java files from .xsp files
78         // at the start, as other tests depend on it to pass.
79         // - BaseGeneratePagesTest
80         suite.addTestSuite(BaseGeneratePagesTest.class);
81
82         // .setup
83         // - TestSetupTest
84         suite.addTestSuite(TestSetupTest.class);
85         // - BaseSuiteSetupTest
86         suite.addTestSuite(ExampleSuiteSetupTest.class);
87
88         // - BaseLabelsLocalizableTest
89         suite.addTestSuite(ExampleLabelsLocalizableTest.class);
90
91         // .control
92         // Should have a test verifying control rendered output is as expected
93         // .lifecycle tests
94         // - RegisteredDecodeTest
95         suite.addTestSuite(RegisteredDecodeTest.class);
```

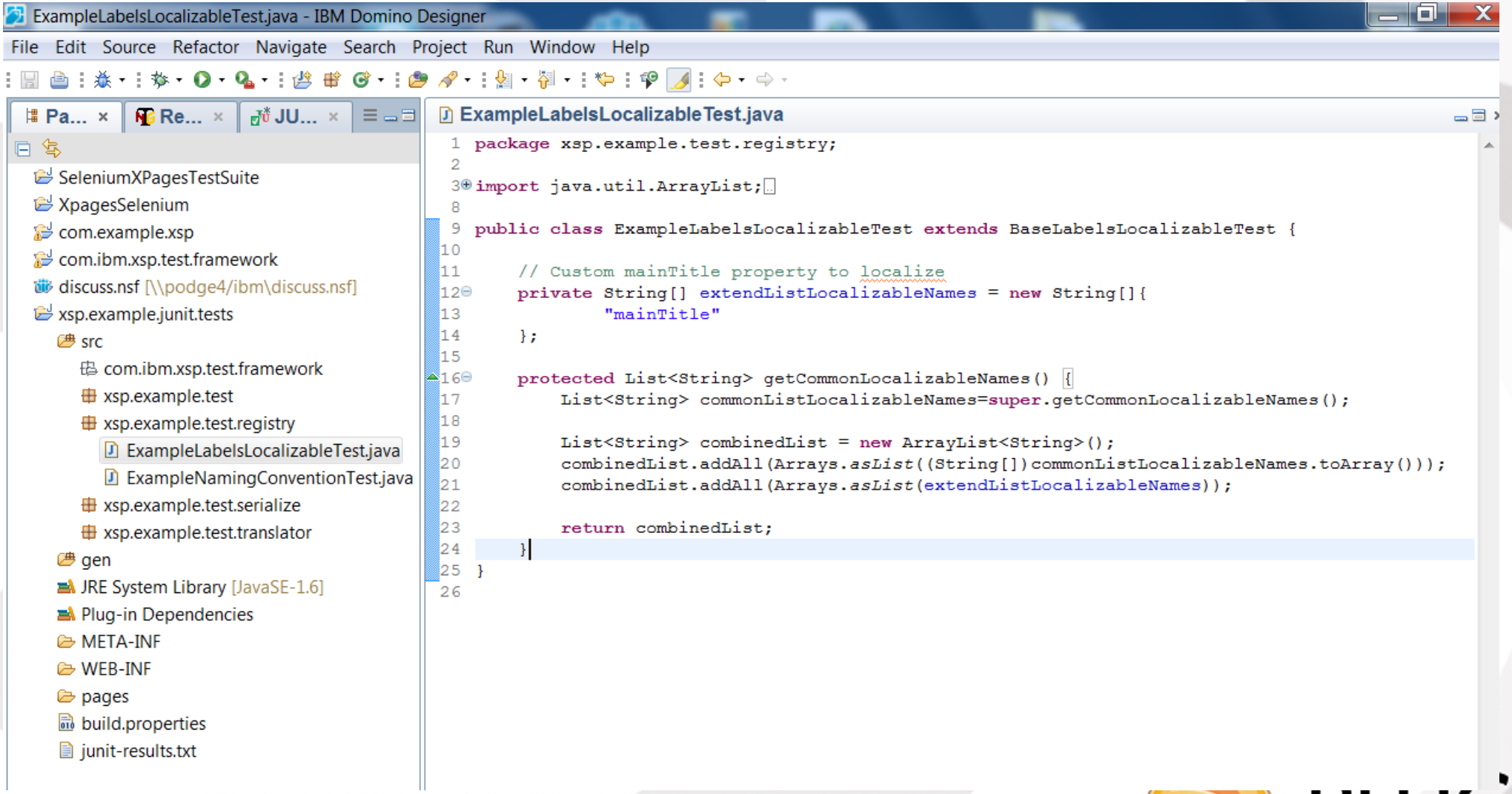
Base XPages Junit Framework test

Extended Junit test

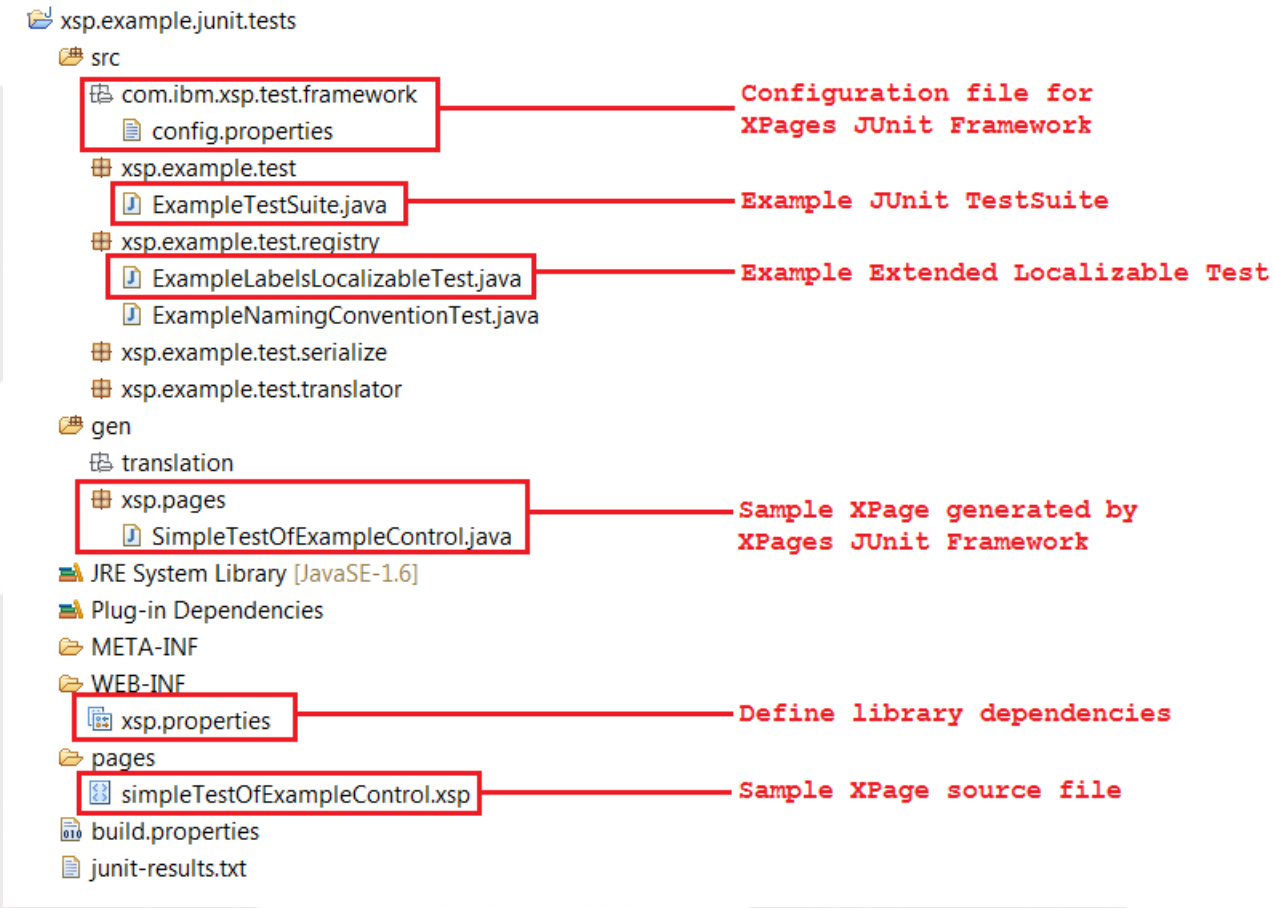
Office Padraic (Network) | Smart Insert | 90 : 9

ExampleLabelsLocalizableTest.java

- Extended JUnit Test to add mainTitle property as localizable

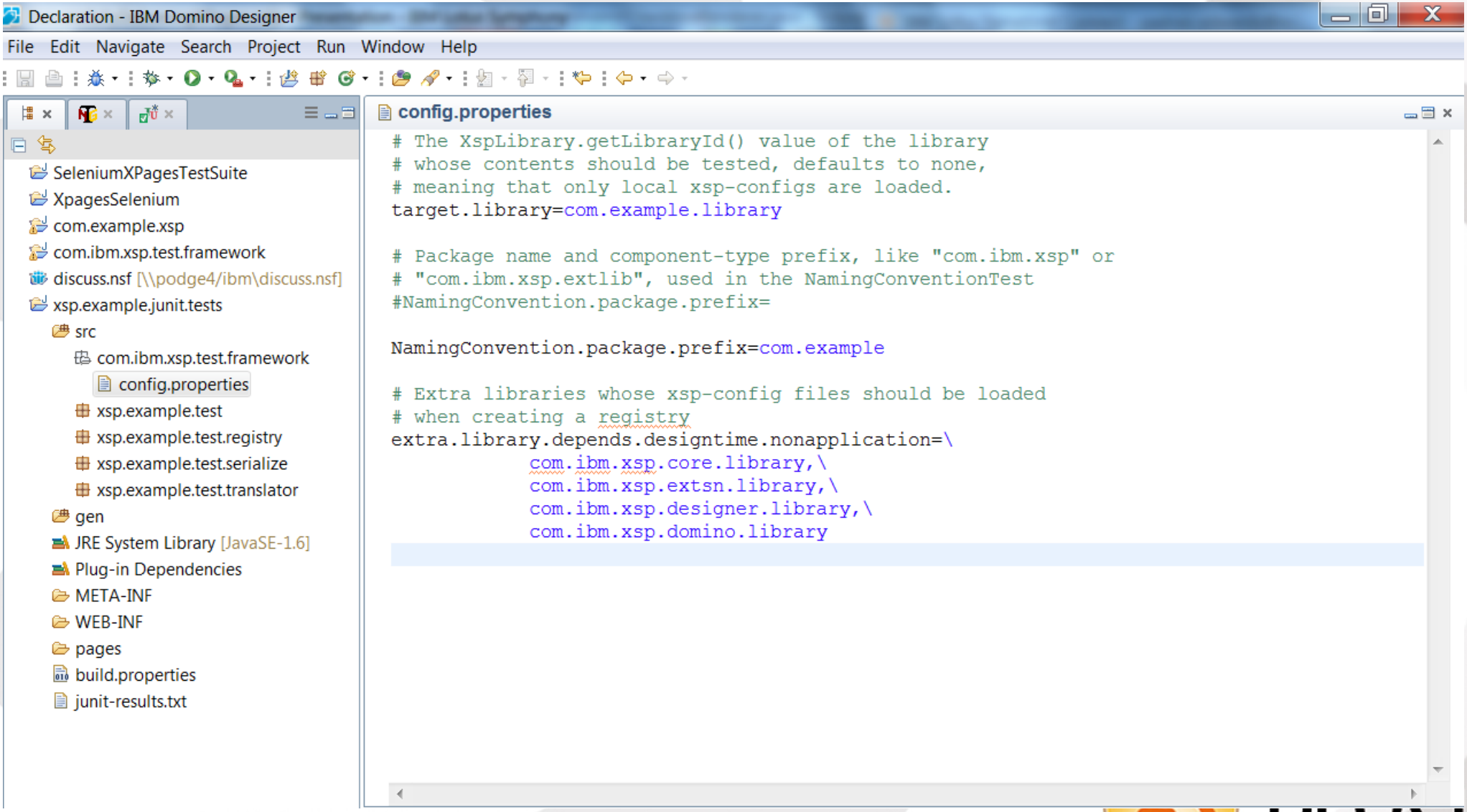


Example XPages JUnit Project Pieces



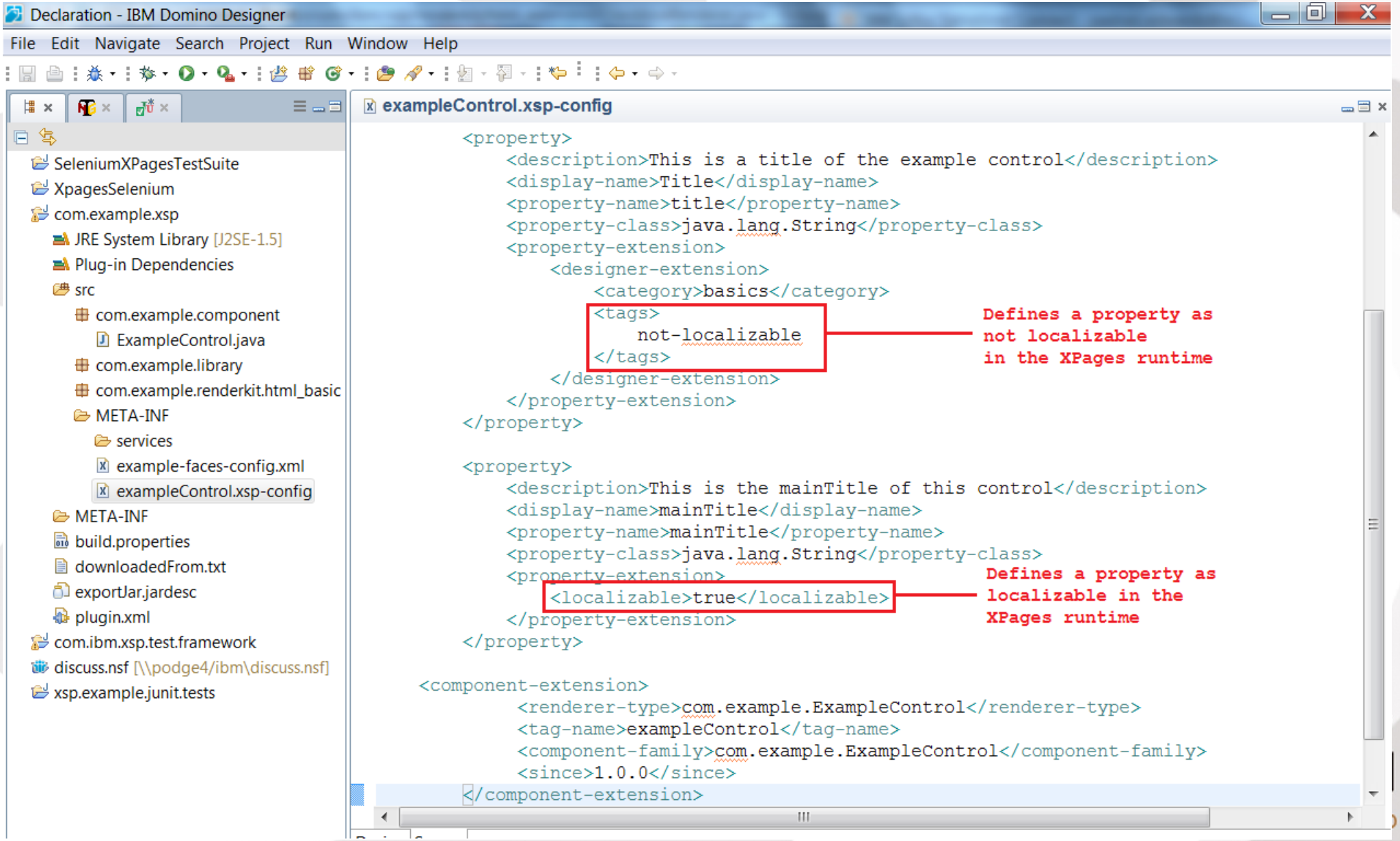
Config.properties

- Properties file needed in your test plugin to configure XPages JUnit framework

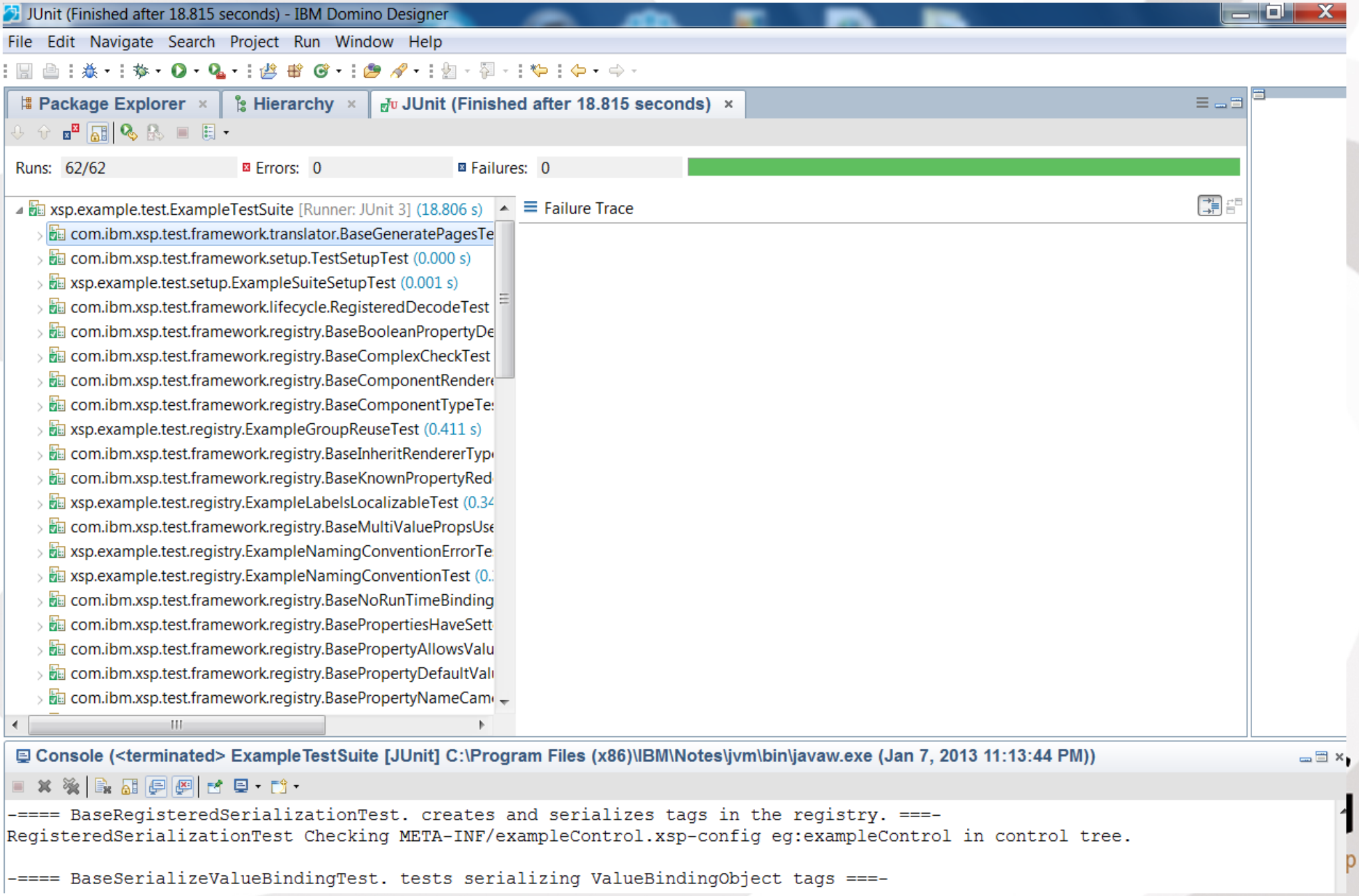


Localizable and not-localizable tags for properties

- Properties file needed in your test plugin to configure XPagesJUnit framework

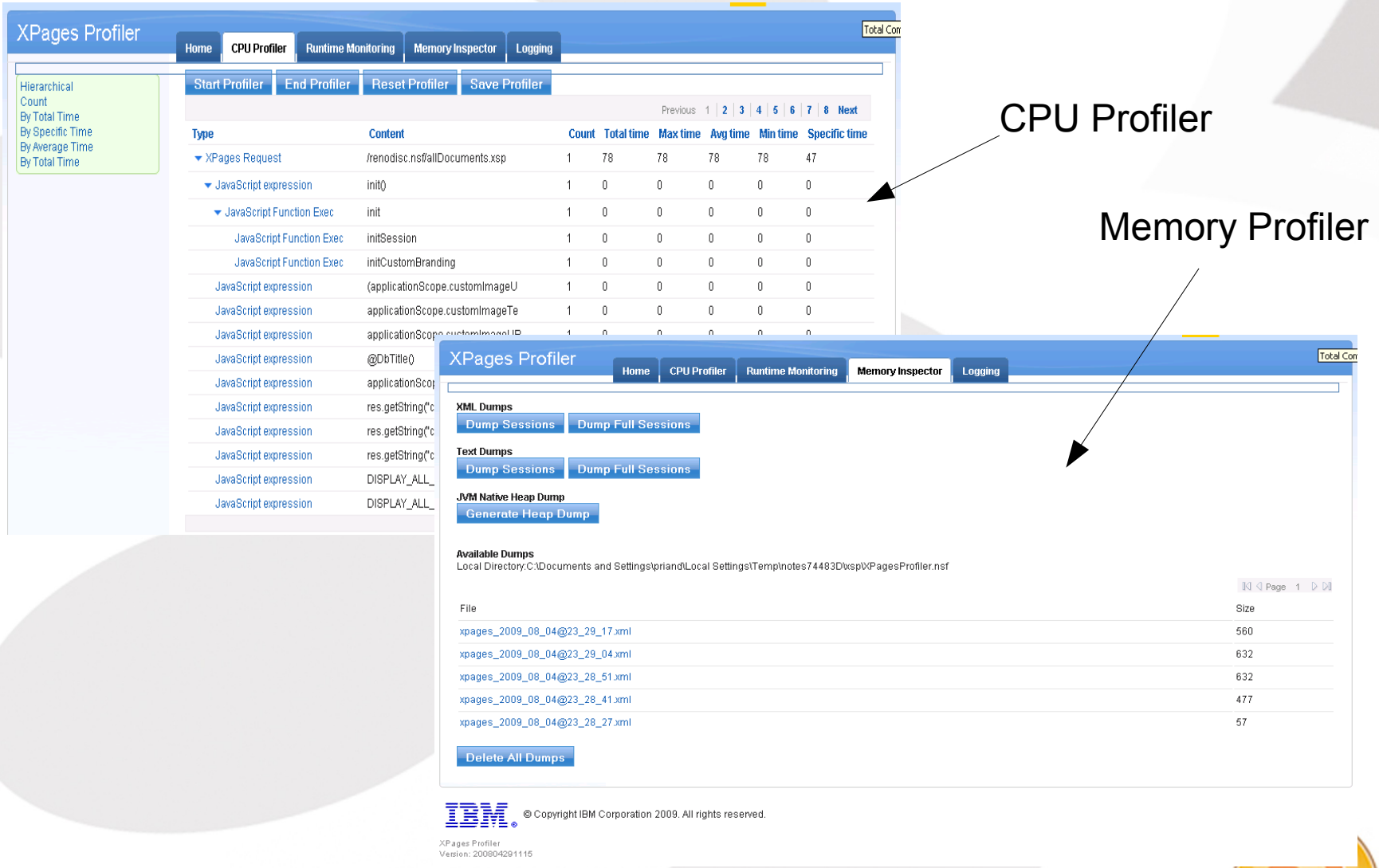


Passing JUnit ExampleTestSuite



XPages Toolbox

XPages Toolbox



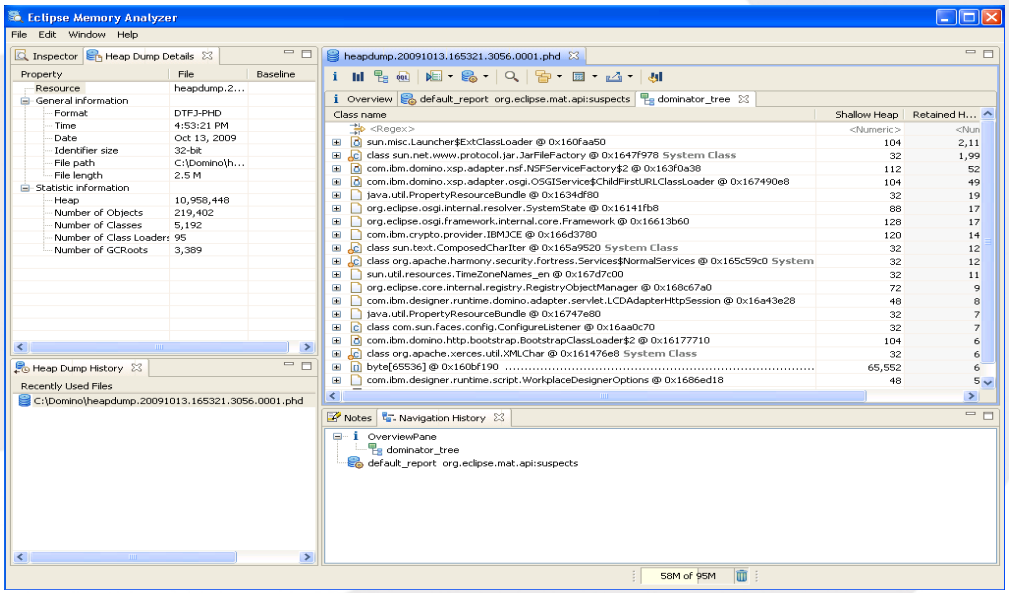
CPU Profiler

Memory Profiler

BLUG
Belux Lotus User Group

XPages Toolbox

- Generate a heap dump of the JVM running in the HTTP task
  - A button in the XPages profiler generates the heap dump
  - A new command from the Domino console
    - `tell http xsp heapdump` (triggers `com.ibm.jvm.Dump.HeapDump()`)
    - `tell http xsp javadump` (triggers `com.ibm.jvm.Dump.JavaDump()`)
- Analyze the heap dump using the Eclipse memory analyzer
  - **http://www.eclipse.org/mat/**



Deployment

BLUG
Belux Lotus User Group

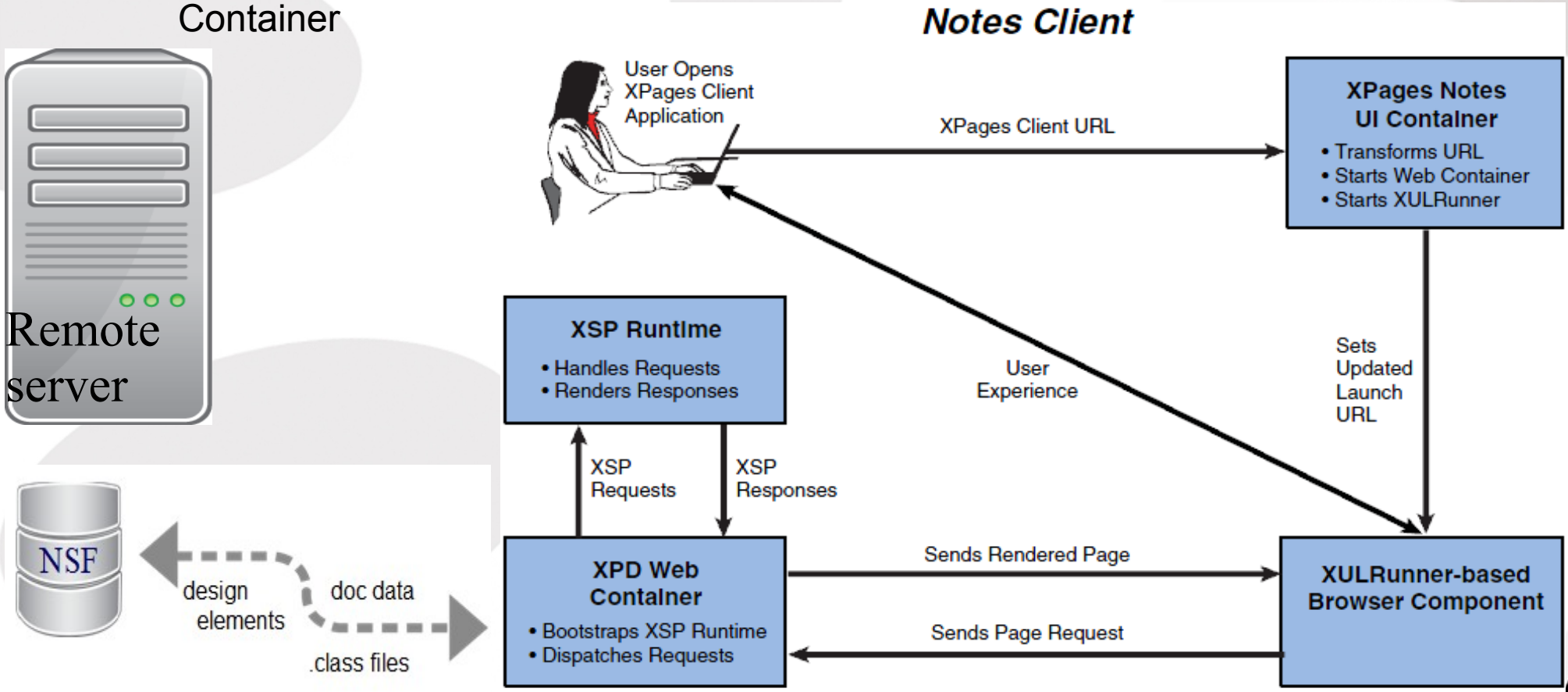
Deployment Considerations: Application Load Time

- XPages applications can be slow to load when first opened by user
  - What happens when...
    - You load your first XPages application in a Notes session ?
    - User loads first XPages application on Domino server ?
  - Why is it slower ?
    - XPages runtime must bootstrap
      - A lot of Java class loading...
    - The application launch page plus any other dependent custom controls, libraries must run
      - Potentially a lot of Java class loading...
- XPages applications can be preloaded on both Notes client and web server
  - e.g. update NOTES.INI manually or by provisioning policy updates
    - XPagesPreload=1
      - Bootstraps just the core XPages runtime
    - XPagesPreloadDB=db.nsf/myPage.xsp, serverName!! another.nsf/myPage.xsp
      - Loads specific application(s)
  - Preloading effectively simulates a warm start, i.e.
    - Same initial load time as when user opens app for 2nd time

BLUG
Belux Lotus User Group

Deployment Considerations: Remote Apps Running in Notes

- Although NSF resides on a remote server, the app runs in the Notes client
  - Incurs network overhead as everything must execute in the ***local*** XPD Web Container



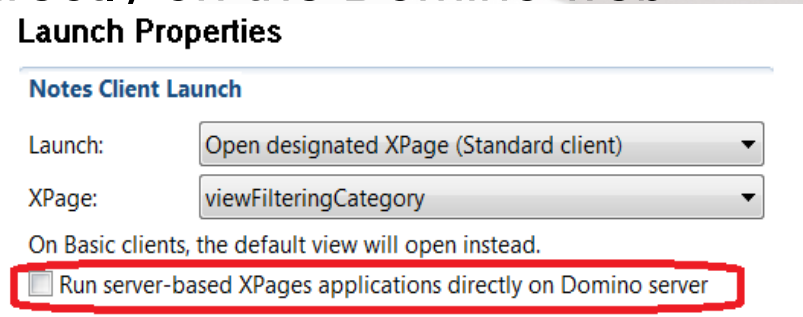
Deployment Considerations: Remote Apps Running in Notes

- Combine Single Copy XPages Design with Preload for production applications
  - Preload means that all required Java classes are preloaded when app is launched
  - Deploying a local design-only SCXD NSF means all web resources are resolved locally
    - Images, CSS, JavaScript files all loaded locally
  - Any Form design elements required by **computeWithForm** logic are resolved locally
    - A Form element is arbitrarily large, e.g. includes all subforms and any other children
    - Large Computed Forms + High Latency Networks = Poor XPages performance
  - Reduced network traffic from SXCD + Preload greatly enhances XPiNC performance !!!

- SXCD helps when many apps need preloading
  - What happens when I have **many instances** of the same application?
    - XPagesPreloadDB=AppA.nsf/home.xsp, AppB.nsf/home.xsp, … AppN.nsf/home.xsp
  - Not practical to constantly add and maintain NOTES.INI with new app names
  - All app instances share a common design …
    - and it is the Design elements that need to preload !
  - Just preload the SXCD NSF

Deployment Considerations: Remote Apps Running in Notes

- Use new XPinC RunOnServer option if Notes optimizations are not a priority
- Choose option to run remote applications directly on the Domino web server
  - New in IBM Notes 9.0 Social Edition
  - Server must be running http task
  - User must have Domino web account
  - Offline access is not a requirement
- Supports XPiNC custom behaviours, look and feel
  - Notes bookmarking
  - Notes client context menus, e.g. Open In Designer
  - EnableModifiedFlag/DisableModifiedFlag
- XPiNC apps can be coded conditionally according to execution mode
- Cannot fully participate in Notes composite applications



```
<xp:this.rendered><![CDATA[#{javascript:!context.isRunningContext("Notes")}]]></xp:this.rendered>

<xp:label.rcp rendered="false"></xp:label.rcp>
```