



# socialbiz

USER GROUP

## Development Tips and Best Practices for XPages

### Contents:

Page 2	IBM Domino Application Development Futures
Page 14	Responsive Application Development for Xpages
Page 25	Take Your XPages Development to the Next Level
Page 59	From XPages Hero To OSGi Guru: Taking The Scary Out Of Building Extension Libraries
Page 78	Ten Lines Or Less: Interesting Things You Can Do In Java With Minimal Code
Page 99	XPages and JavaServer Faces
Page 129	Be Open - Use WebServices and REST in XPages Application
Page 147	XPages Performance and Scalability
Page 165	Taking XPages Applications from Out-of-the-Box to Outstanding
Page 173	Improve XPages Application Performance with JSON-RPC



Visit SocialBizUG at <https://www.socialbizug.org>

# IBM Domino Application Development Futures

*Eamon Muldoon, IBM*  
*Pete Janzen, IBM*



## Please Note:

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

## Agenda

- 2014 review
- Responsive XPages
- Relational data source for XPages
- Document encryption and signatures
- Open source
- Domino on Bluemix

## 2014 REVIEW

## Notes/Domino/Designer Fixes

- Notes/Domino/Designer 9.0.1 FP 2
  - IE11 support
  - CKEditor 4.3.2 (Domino server)
- Notes/Domino/Designer 9.0.1 FP 3 (yes, delivered in Jan '15 but we wanted you to know :-)
- iOS 8 support for XPages mobile controls - 9.0.1 FP2 IF1
- Dojo 1.9.4
- CKEditor 4.3.2.2 (Domino server & Notes client)
- Critical security fixes
  - TLS1.0 - Patch for Notes/Domino 8.5.1 -> 9.0.1.
  - SHA-2 - Patch for Notes/Domino 9.0.x (requires 9.0 or above)

## OpenNTF Releases



- OpenNTF gives IBM a vehicle to deliver features outside of normal releases
  - New features in XPages Extension Library come back to product
- XPages Extension Library
  - 9 releases in 2014
  - Single Page Application Wizard for XPages
  - Other enhancements will be covered in upcoming slides
- IBM Domino Update Site for Build Management
  - Package of artifacts needed to build XPages

IBM

ConnectED2015

## FUTURES

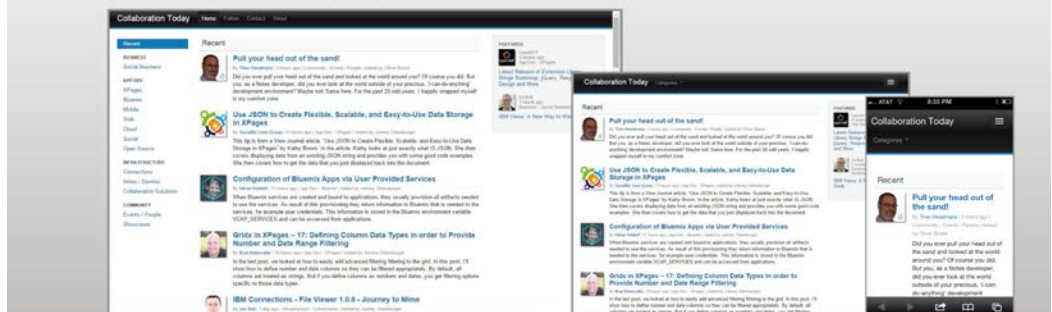
### Responsive XPages

IBM

ConnectED2015

## Develop Once For Desktop, Tablet or Smartphone

- Build web applications which automatically adapt to the screen area of a desktop, tablet or smartphone
- Bootstrap makes it easy to build elegant responsive web apps



IBM

ConnectED2015



## Responsive Web Design With Bootstrap



- Bootstrap (<http://www.getbootstrap.com>)
  - Web Design framework, Responsive since v2.0 (now on v3.3.0)
- Most popular repository on GitHub
- Bootstrap has grown in popularity in the XPages community
- Bootstrap4XPages project: ~3000 downloads in a year

IBM

ConnectED2015

## Born From The Community

- In the beginning – Bootstrap4XPages (B4X) project on OpenNTF
  - Philippe Riand & Mark Leusink
  - Bootstrap 2.3.1, 2.3.2, 3.0.0, 3.1.1, 3.2.0
  - jQuery, Select2, Glyphicons, dbootstrap
  - Themes, renderers, styling
- B4X pulled into the XPages Extension Library
  - New plugin in ExtLib: com.ibm.xsp.theme.bootstrap
  - Bootstrap 3.2.0 only, jQuery, Glyphicons, dbootstrap
  - Two themes, more renderers, more styling, more testing!

IBM

ConnectED2015

## XPages Theme Combo Contribution

- YOUR themes in Domino Designer !
- New Community Driven Feature ...
  - In the class that implements StyleKitFactory also implement StyleKitListFactory
  - Return a list of themes using getThemeIds()

```
/**
 * @author Gary Marjoram
 *
 */
public class MyStyleKitFactory implements StyleKitFactory, StyleKitListFactory {

    @Override
    public String[] getThemeIds() {
        return new String[] { "myThemeId" };
    }
}
```

**XPage Properties**

**Theme Defaults**

Application theme: Platform default

☐ Override on Web: Platform default

☐ Override on Notes: Bootstrap3.2.0 flat

☒ Use mobile theme for XPages with mobile theme:

Mobile theme: OneUI

☐ Override on iOS: OneUI V2

☐ Override on Android: OneUI V2.1

☐ Debug user agent: OneUI V3.0.2

webstandard

Application default

Application default

Application default

IBM

ConnectED2015

## jQuery

- jQuery v2.1.1 contained in XPages Responsive plugin
- Yes, that now means jQuery will ship with Domino!
  - XPages controls still rely on Dojo
- Multiple ways to use it
  - Use it by leveraging Bootstrap theme
  - OR create a theme that adds it as a resource
  - OR simply add it as a resource to an XPage
  - Use jQuery calls in CSJS of your XPage application

## FUTURES

Relational data source for XPages

## Improve Productivity By Bringing Data To The User

- Integrate data from relational databases into the context of your collaborative or workflow driven Domino XPages application
- Create dashboards that allow knowledge workers to quickly access data from disparate systems
- Allow developers to easily integrate relational data into XPages applications with Domino Designer



## Connection Pooling

- Connection pooling improves application responsiveness by establishing connections prior to request for access to RDB
- XPages RDB support provided simple connection pooling
  - Needed something more robust
- Added Apache Commons DBCP (Database Connection Pooling)
  - Apache DBCP support (v 1.4)

```

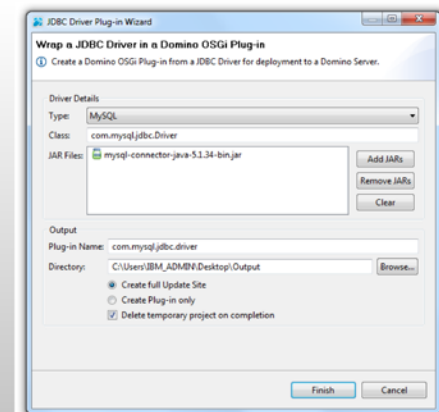
1 <jdbc type="sample">
2   <driver>com.ibm.db2.jcc.DB2Driver</driver>
3   <url>jdbc:db2://myRDBServer:IBM-DB2:SAMPLE/</url>
4   <user>Administrator</user>
5   <password>password</password>
6
7   <sample>
8     <minPoolSize>1</minPoolSize>
9     <maxPoolSize>3</maxPoolSize>
10    <maxConnectionIdle>4</maxConnectionIdle>
11    <maxTimeout>1</maxTimeout>
12    <idleTimeout>300</idleTimeout>
13    <maxLiveTime>0</maxLiveTime>
14    <acquireTimeout>3000</acquireTimeout>
15  </sample>
16 </jdbc>

```




## JDBC Driver Plugin Wizard

- Wraps a JDBC Driver in a Domino OSGi Plug-in
- Complements the relational runtime enhancements made in the same release
- Produces an update site or Plug-in which can then be deployed to a Domino Server
- An OSGi Plug-in is the recommended deployment method when accessing JDBC drivers from XPages applications
  - Registration is automatic
  - Driver is shared, ensuring optimum performance





## FUTURES

Document encryption & signature support for XPages




## Secure Your Data On The Web

- Ensure only the people you want to access the data can access the data using XPages document encryption
- Simplify access using public keys or apply greater control using secret keys
- Ensure authenticity by electronically signing Domino documents from the web



IBM

ConnectED2015

## New Features For XPages Encryption & Signature Support

- New Secret Key Picker Control
  - Populates a picker with the secret encryption keys contained in a user id
- New @Functions
  - @UserSecretKeys()**
    - Extracts secret keys from a user id stored in the Domino ID vault
    - Uses logical default parameter values for quick coding  
Current user id and password, current server as ID Vault server
    - All parameters can be explicitly specified, e.g.  
— @UserSecretKeys(server, password, username)
  - @UserID()**
    - Returns an instance of a UserID object
    - Represents the current user or a specified user

```
<xsp:secretKeyPicker id="secretKeyPicker1"
  for="SecretKeys" pickerIcon="/padlock16x16.png">
  <xe:this.dataProvider>
    <xe:simpleSecretKeyPicker
      secretKeyList="#{javascript:@UserSecretKeys();}" />
    </xe:simpleSecretKeyPicker>
  </xe:this.dataProvider>
</xsp:secretKeyPicker>
```

IBM

ConnectED2015

## Additional Features For XPages Encryption & Signature Support

- New backend classes, methods & properties in C, Java & LotusScript
- New IDVault class
  - Methods for working with IDs (Get or put ID, Get username...)
- New UserID class
  - Method for getting encryption keys
- Other Methods
  - Session class: IDVault.Session.getIDVault()
  - Database class: Database.setUserIDForDecrypt(UserID uid)
  - Document class: Document.encrypt(Optional UserID uid)

IBM

ConnectED2015

# FUTURES

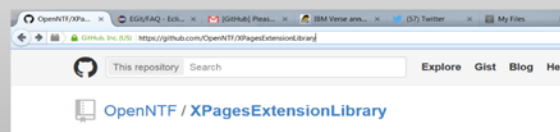
Open Source

IBM

ConnectED2015

## XPages ExtLib Source Repository on GitHub

- Empowering Our Development Community
- XPages source has been available on OpenNTF.org since October 2011
  - Only in a zip file packaged with the binary releases
  - Impediment to community collaboration on the project
- ExtLib source repository available!
  - Provides all the ExtLib runtime and designer features and plug-ins
  - Includes tooling to enable building, localization ... and a P2 Update Site



IBM

ConnectED2015

## XPages ExtLib Source Repository on GitHub

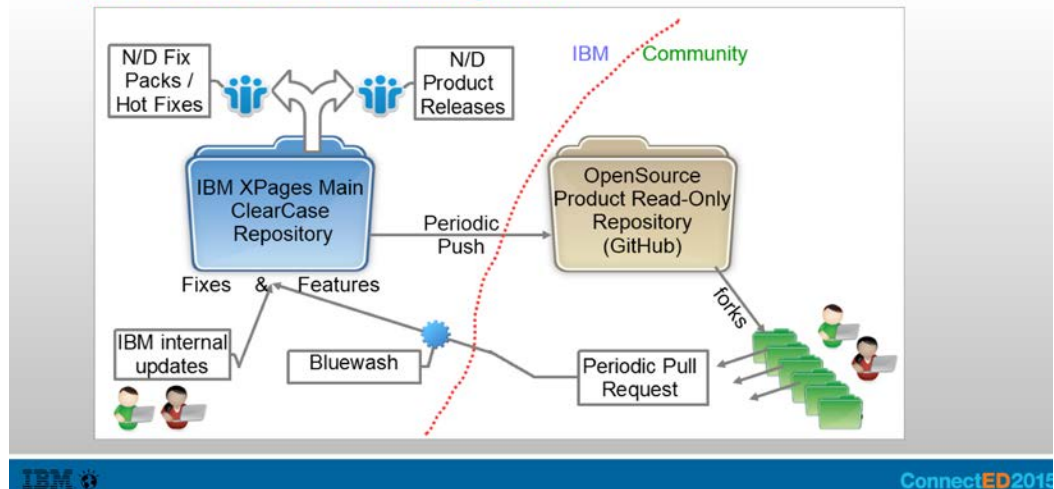
- How does this help YOU ?
- Participating in the ExtLib project means you can...
  - Collaborate on new XPages or Designer extensions
  - Contribute features and bug fixes
  - Influence the direction of the app dev offering
- Participation in the project is easier than ever before !
  - Setting up a ExtLib development environment
    - You need a GitHub account
    - An IDE e.g. Eclipse or Domino Designer

IBM

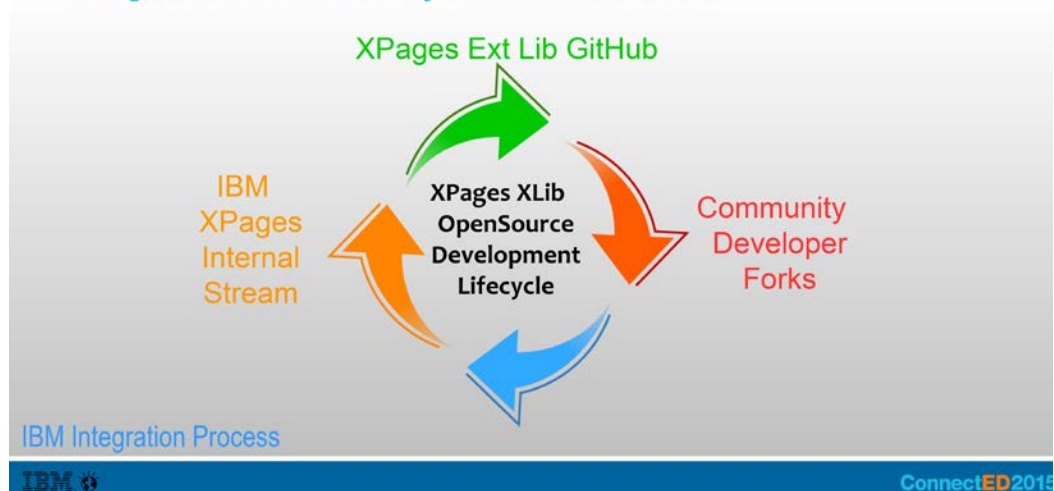
ConnectED2015



## Process for GitHub ExtLib Integration



## XPages ExtLib : Community Contribution Process



## Integrating GitHub Pull Requests

- Code Standards for the XPages Extension Library
- Code submitted via pull requests are merged into core after a code review
- Certain standards are applied:
  - Code must build properly
  - Passing of JUnit tests
  - Tagging of UI strings for localization
  - Etc
- Everything you need is available
  - On GitHub
  - As OpenNTF projects



IBM Domino Update Site for Build Management [Download](#)



# FUTURES

## Domino on Bluemix



IBM

ConnectED2015

## Bluemix - Create & Deploy apps

### About Bluemix

- Run apps in any language
- Built on open standards
- Integration services to systems of record
- Designed for mobile
- Provides DevOps services



### Bluemix services include:

- DevOps
- Big Data
- Mobile
- Cloud Integration
- Security
- Internet of Things
- Business Analytics
- Database
- Web and application



**Compose applications**  
from a rich library of IBM, 3rd party and open source runtimes, services and APIs.



**Deploy and scale**  
new applications and services with infrastructure services from IBM SoftLayer.



**Code with confidence**  
knowing IBM's cloud platform is built on a foundation of open standards.

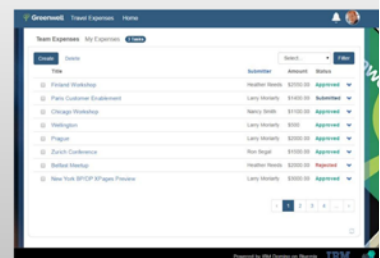
IBM

ConnectED2015

## Release the Power of Domino on IBM Bluemix!



- Customers realize the next generation of LOB apps
  - Customers can access Domino applications on IBM Bluemix
  - Be more responsive to your changing business requirements
  - App.Next → New types of applications that use services like Watson, mobile, social files and more....
  - Leverage your investment in Notes & Domino apps
    - Easier path to modernized applications
    - Integrate your on-premises applications
- Business Partners benefit from new opportunities
  - New routes to market (IBM Cloud Marketplace)
  - Embrace more of the IBM portfolio and 3rd party services
  - Focus on selling the value of your solution, not the supporting infrastructure



IBM

ConnectED2015

## Rapidly Deliver Applications And Services



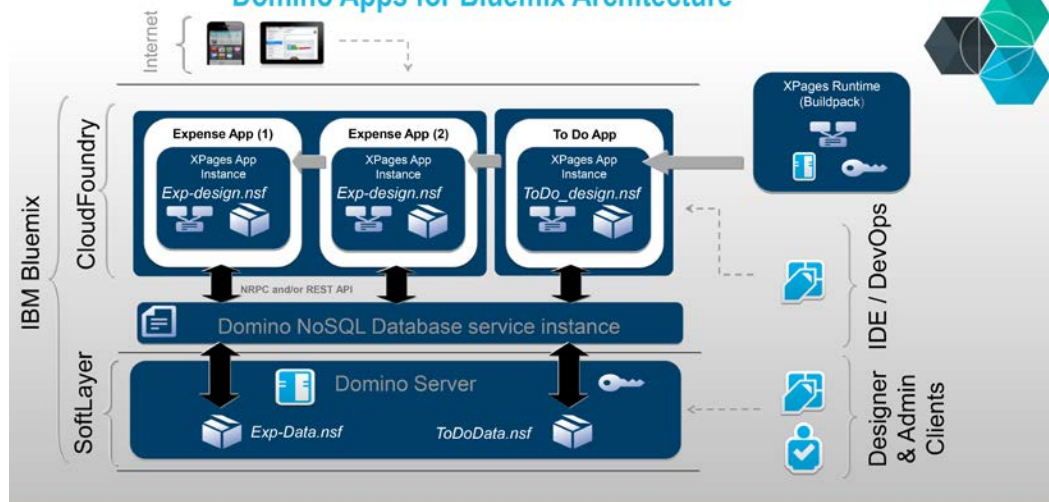
- Developers Can Focus on Developing
  - Bluemix provides a click-and-play environment to build, manage and run Domino apps
  - XPages runtime allows Domino developers to bring their skills to the cloud
  - Domino NoSQL Database service provides secure data store
  - XPages boilerplate allows developers to quickly get an application up and running
  - Use runtimes like: Node.js or Liberty to build apps against Domino
- DevOps - allow the developer to run the entire app
  - Dashboard for monitoring applications
  - Easily scale applications to meet work loads
  - Deliver resilient applications which provide high availability and quickly recover from problems
  - Automate builds with code in RTC or Git



IBM

ConnectED2015





## Domino Apps for Bluemix Architecture



IBM

ConnectED2015

## Domino on Bluemix 2015 Roadmap Directions

- 
 ▪ Bluemix Runtime
  - Domino XPages Runtime for dev, test and production
- 
 ▪ Bluemix Service
  - Domino Data service for dev and test
  - Production ready data service provided by customer, BP or IBM services team
- 
 ▪ Bluemix Boilerplate
  - XPages/Domino Database starter kit with samples to quickly get up and running on Bluemix
- 
 ▪ Domino Designer extensions
  - Build applications for Bluemix
  - Deploy applications to Bluemix



IBM

ConnectED2015

## More Information – Summary

- **OpenNTF** – Open Source Community
  - Code, samples and more: <http://www.openntf.org>
- **Enablement** – Doc, examples, demos
  - Domino Application Development Wiki - <http://www.lotus.com/idd/ddwiki.nsf>
  - NotesIn9 – <http://www.notesin9.com/>
  - Two new Domino application development papers
    - Collaborative and business applications for the connected company → [Link](#)
    - IBM Notes and Domino Applications: A road map for modernization using IBM XPages → [Link](#)
- **Forums** - Got Questions, Need Answers?
  - Stackoverflow - <http://stackoverflow.com/questions/tagged/xpages>
  - XPages Forum - <http://xpages.info/forum>



ConnectED2015

## Engage Online

- **SocialBiz User Group** [socialbizug.org](http://socialbizug.org)
  - Join the epicenter of Notes and Collaboration user groups
- **Social Business Insights blog** [ibm.com/blogs/socialbusiness](http://ibm.com/blogs/socialbusiness)
  - Read and engage with our bloggers
- **Follow us on Twitter**
  - [@IBMConnect](#) and [@IBMSocialBiz](#)
- **LinkedIn** <http://bit.ly/SBComm>
  - Participate in the IBM Social Business group on LinkedIn
- **Facebook** <https://www.facebook.com/IBMConnected>
  - Like IBM Social Business on Facebook



ConnectED2015

## Notices and Disclaimers

Copyright © 2015 by International Business Machines Corporation (IBM). No part of this document may be reproduced or transmitted in any form without written permission from IBM.

**U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.**

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY. IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

**Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com, BrassRing®, Connections™, Domino®, Global Business Services®, Global Technology Services®, SmartCloud®, Social Business®, Kenexa®, Notes®, PartnerWorld®, Prove It®, PureSystems®, Sametime®, Verse™, Watson™, WebSphere®, Worklight®, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).



ConnectED2015

# Responsive Application Development for Xpages

*Brian Gleeson, IBM*  
*Tony McGuckin, IBM*



## Agenda

- Part 1 – Existing Content
  - Bootstrap Intro
  - Bootstrap in XPages
  - Responsive Application Layout
  - Extra Resources
- Part 2 – Back to the future
  - New Controls
  - Future Work
- Part 3 – Wrap Up
  - Best Practices
  - Conclusions





## Solution Identified

- Bootstrap
  - UI framework
  - Open Source
  - Responsive!
- Features
  - Flexible grid-based layout
  - CSS & iconography resources
  - Javascript resources: jQuery
- Very popular, which means
  - Updated regularly
  - Lots of documentation/online support
  - Free & paid addons: themes, templates



getbootstrap.com

IBM

ConnectED2015

## Solution Identified - Bootstrap

- Responsive Web Design
  - Optimised UX on any device
  - Readability, navigation, less scrolling/panning/zooming



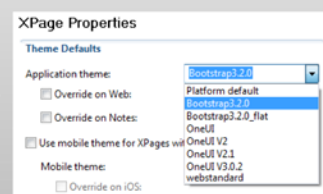
- Benefits
  - Code once for all devices
  - Time & Cost saving
  - Increased reach of application
  - Consistent experience

IBM

ConnectED2015

## Solution Identified - Bootstrap

- Bootstrap in XPages
  - Aug 2013 - Bootstrap4XPages OpenNTF project
  - Nov 2014 - Bootstrap added to Extension Library
    - com.ibm.xsp.theme.bootstrap plugin
  - B4X – 3000+ downloads
  - Bootstrap in Extlib – 827 downloads (r10), 667 downloads (r11) - as of Jan 9th
- Frank's next steps:
  - Install 9.0.1 Extlib, release 10+
  - Change application theme
  - Save, rebuild, refresh

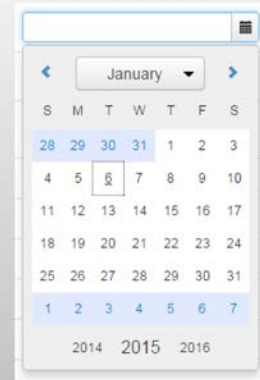
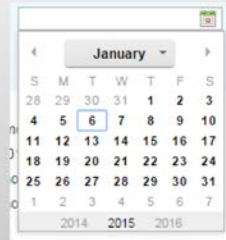


IBM

ConnectED2015

## XPages Controls Before & After Bootstrap

### ■ Date Time Picker

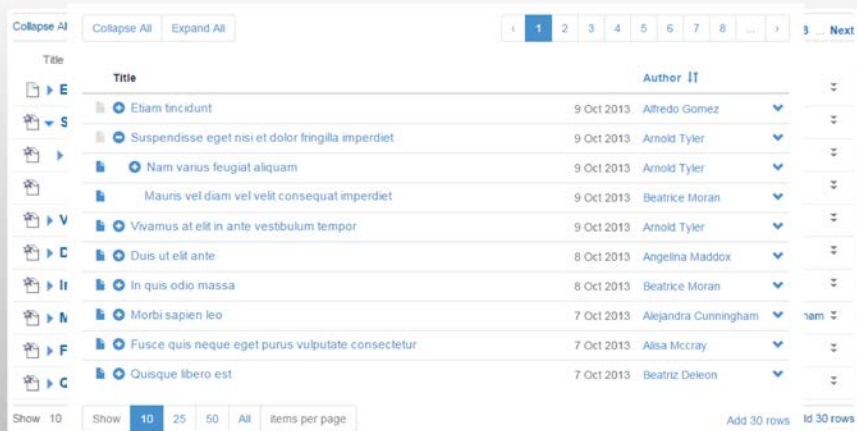


IBM

ConnectED2015

## XPages Controls Before & After Bootstrap

### ■ Data View

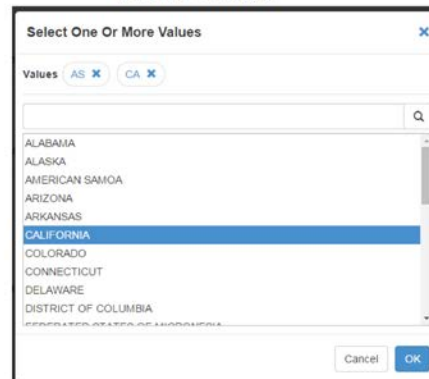


IBM

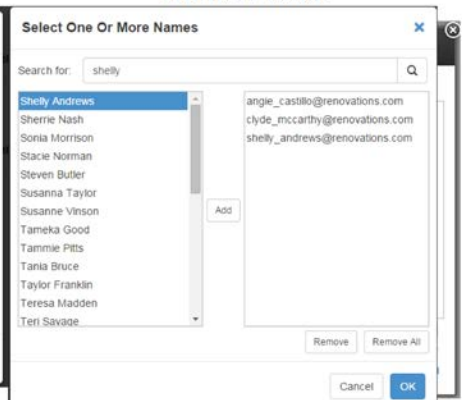
ConnectED2015

## XPages Controls Before & After Bootstrap

### Value Picker



### Name Picker



IBM

ConnectED2015



## Responsive Application Layout

- `<xe:bootstrapResponsiveConfiguration>`
- Configuration for App Layout control
- All standard App Layout properties
- Additional properties
  - fixedNavbar
  - invertedNavbar
  - pageWidth
  - collapseLeftColumn
  - collapseLeftMenuLabel
  - collapseLeftTarget

# DEMO

## Extra Resources - Glyphicons

- 200 icons
- Font format
- Utilised via CSS classes
- Customisable
- How to use them?
  - Built into many XPages controls
  - Add them yourself



```
<xp:panel id="panel1" style="text-align:center;">
  <xp:div id="starIcon" styleClass="glyphicon glyphicon-star"
    style="font-size:50px;color:green">
  </xp:div>
  <xp:div id="starText">Star Icon</xp:div>
  <xp:button id="test1" value="Success">
    <xp:div styleClass="glyphicon glyphicon-ok" style="color:#428bca;">
    </xp:div>
  </xp:button>
</xp:panel>
```



## Extra Resources - jQuery

- jQuery v2.1.1 is in XPages ExtLib
  - Javascript library required by Bootstrap
- Add jQuery to XPage application
  1. Use bootstrap themes: "Bootstrapv3.2.0" or "Bootstrapv3.2.0\_flat"
  2. Extend bootstrap theme
 

```
<theme extends="Bootstrap3.2.0" ... >
...
<resource><content-type>application/x-javascript</content-type>
  <href>/..ibmjspxres/.extlib/responsive/jquery/jquery-2.1.1/jquery-2.1.1.js</href>
</resource>
```
  3. Create new theme, add jQuery resource
  4. Add resource to XPage
 

```
<xp:this.resources>
  <xp:script clientSide="true"
    src="/..ibmjspxres/.extlib/responsive/jquery/jquery-2.1.1/jquery-2.1.1.min.js" >
  </xp:script>
</xp:this.resources>
```

## Extra Resources - jQuery

- Using jQuery in an XPage – CSJS

1. Add to onClientLoad

```
<xp:eventHandler event="onClientLoad" submit="false">
  <xp:this.script><![CDATA[
    $("#{id$='myText'}").html("This is JQUERY!");
  ]]></xp:this.script>
</xp:eventHandler>
```

2. Add in a script block

```
<xp:scriptBlock id="script1">
  <xp:this.value><![CDATA[
    XSP.addOnLoad( function() {
      $("#{id$='myText'}").html("This is JQUERY!");
    });
  ]]>
</xp:scriptBlock>
```

3. Add in eventHandler

```
<xp:button value="Click me" id="button1">
  <xp:eventHandler event="onclick" submit="false"><xp:this.script>
    <![CDATA[
      $("#{id$='myText'}").html("This is JQUERY!");
    ]]>
  </xp:this.script></xp:eventHandler>
</xp:button>
```

# DEMO

## PART 2

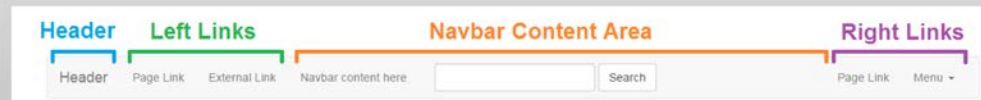
# Back to the Future

IBM

ConnectED2015

### New Controls 1 - Navbar

- Navbar = Responsive navigation header
- Available Properties
  - fixed
  - headerText, headerStyle, headerStyleClass
  - pageWidth
  - navbarLeftLinks
  - navbarRightLinks



IBM

ConnectED2015

# DEMO

IBM

ConnectED2015

## New Controls 2 – Simple Responsive Application Layout

- New configuration for Application Layout
  - Simplified – 1 banner, left/right facets, and content area
  - Responsive

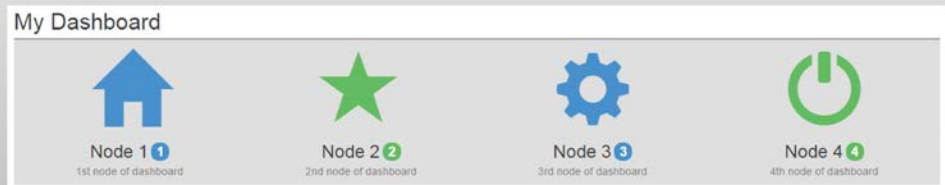


IBM

ConnectED2015

## New Controls 3 – Dashboard

- Responsive Customisable Dashboard
  - Add up to 12 nodes
  - 1 Node = Image, Title, Badge, Description
  - Use an image or glyphicon
  - Customise responsive node sizes



IBM

ConnectED2015

# DEMO

IBM

ConnectED2015

## New Controls 4 - Carousel

- Slideshow control
- Configuration
  - Any number of Slides
  - Customise Responsive Carousel sizes
- Carousel parameters:
  - autoCycle
  - slideInterval
  - isWrapped
  - pauseOnHover
- Slide Content:
  - Image
  - Heading
  - Caption
  - Description
  - Button Link

## New Controls 4 - Carousel

- Slide Content:
  - Image/Background Colour
  - Heading
  - Caption
  - Description
  - Button Link



# DEMO

## Future Work

- Additional Responsive Bootstrap XPages Controls
  - Suggestions welcome!
- More fixes
- Improved Designer tooling
- Release Date for controls demo'ed today: Q1 2015
- Other future plans
  - Bootstrap 3.3.x upgrade
  - Bootstrap 4 – currently pre-beta
  - IE7/IE8 Support?

## PART 3: Wrap Up

## Best Practices

- Use containers to wrap each XPage
 

```
<xp:panel styleClass="container"> OR <xp:panel styleClass="container-fluid">
  <xp:panel styleClass="row">          <xp:panel styleClass="row">
```
- Use grid system – [getbootstrap.com/css/#grid](http://getbootstrap.com/css/#grid)
  - 12 columns
  - .col-xs-X, .col-sm-X, .col-md-X, .col-lg-X
  - .col-xs-offset-Y, .col-sm-offset-Y, .col-md-offset-Y, .col-lg-offset-Y
  - Nested Columns
 

```
<div class="col-sm-9">
  <div class="row">
    <div class="col-xs-8 col-sm-6"></div>
    <div class="col-xs-4 col-sm-6"></div>
```
- Use Bootstrap CSS classes
- Mobile First



## Best Practices

- Extend XPages Bootstrap themes
- Create/reuse your own Bootstrap custom control(s)
- Use CSS media queries

```
/* Extra Small */ @media only screen and (min-width : 480px) { }
/* Small Devices */ @media only screen and (min-width : 768px) { }
/* Medium Devices */ @media only screen and (min-width : 992px) { }
/* Large Devices */ @media only screen and (min-width : 1200px) { }
```

- Responsive images: `<img class="img-responsive">`

– And Image shapes

`<img class="img-rounded">`



`<img class="img-circle">`



`<img class="img-thumbnail">`



IBM

ConnectED2015

## Conclusions – Frank's End Product



- Bootstrap
- Responsive
- XPages
- Glyphicons + jQuery
- New Controls
- Big Raise!



## Conclusions – Take Action!

- Download + install latest XPages ExtLib
- Change application theme
- Play around with Bootstrap in XPages
- Visit [getbootstrap.com](http://getbootstrap.com) for documentation/ideas



- Watch enablement video - <http://www.youtube.com/watch?v=XdWYmPLmrk>
- Read documentation - <http://ibm.biz/BdEMdP>

- Download today's To-Do application
- Contribute!! - <http://github.com/OpenNTF/XPagesExtensionLibrary>



IBM

ConnectED2015

## Resources & Further Reading

- ExtLib - [extlib.openntf.org](http://extlib.openntf.org)
- Bootstrap - [getbootstrap.com](http://getbootstrap.com)

### Community:

- StackOverflow – [stackoverflow.com/questions/tagged/xpages](http://stackoverflow.com/questions/tagged/xpages)
- NotesIn9 (David Leedy) – [notesin9.com](http://notesin9.com)
- Mark Roden - [xomino.com](http://xomino.com)
- John OldenBurger - [xpagesandmore.blogspot.dk/](http://xpagesandmore.blogspot.dk/)

IBM

ConnectED2015

## Engage Online

- SocialBiz User Group [socialbizug.org](http://socialbizug.org)
  - Join the epicenter of Notes and Collaboration user groups
- Social Business Insights blog [ibm.com/blogs/socialbusiness](http://ibm.com/blogs/socialbusiness)
  - Read and engage with our bloggers
- Follow us on Twitter
  - @IBMConnect and @IBMSocialBiz
- LinkedIn <http://bit.ly/SBComm>
  - Participate in the IBM Social Business group on LinkedIn
- Facebook <https://www.facebook.com/IBMConnected>
  - Like IBM Social Business on Facebook

IBM

ConnectED2015

## Notices and Disclaimers

Copyright © 2015 by International Business Machines Corporation (IBM). No part of this document may be reproduced or transmitted in any form without written permission from IBM.

**U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.**

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY. IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

**Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com, BrassRing®, Connections™, Domino®, Global Business Services®, Global Technology Services®, SmartCloud®, Social Business®, Kenexa®, Notes®, PartnerWorld®, Prove It!®, PureSystems®, Sametime®, Verse™, Watson™, WebSphere®, Worklight®, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

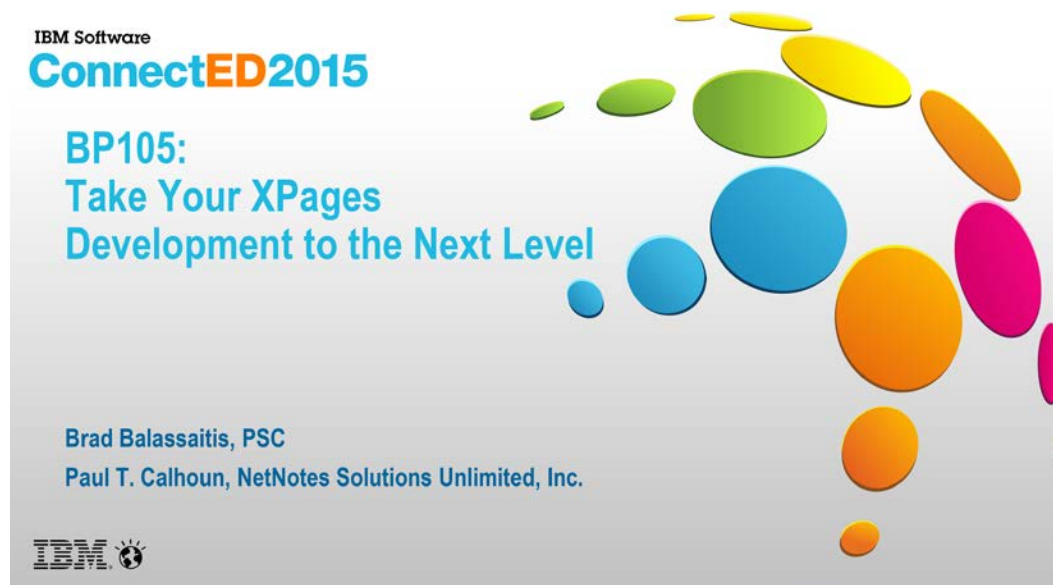
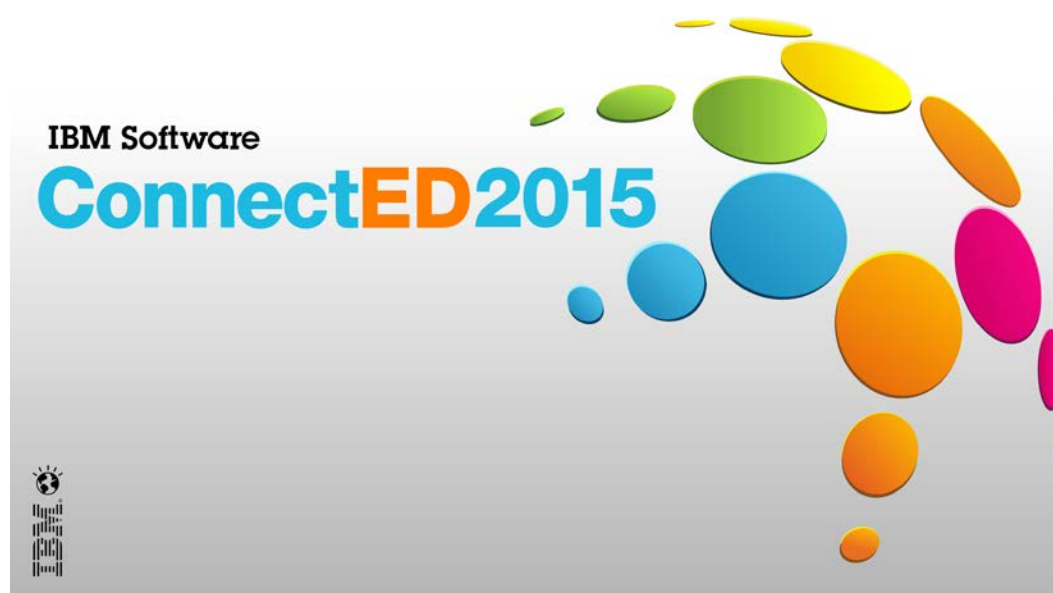
IBM

ConnectED2015

# Take Your XPages Development to the Next Level

*Brad Balassaitis, PSC*

*Paul T. Calhoun, NetNotes Solutions Unlimited, Inc.*



## Your Speakers – Brad Balassaitis

- Senior Consultant for PSC Group
- 2014 and 2015 IBM Champion
- Certified Notes developer since R4
- Blog: [xcellerant.net](http://xcellerant.net)

IBM

ConnectED2015

## Your Speakers – Paul T. Calhoun

- I have been working with Lotus/IBM technologies for the past 25 years.
- I have been certified in both Administration and Development on every major release of Notes/Domino since release 3.
- My focus has always been on training and enablement of Administrators and Developers, but I also provide mentoring and consulting services.
- Most recently as co-owner of QDiligence I have been focused on supporting a hosted XPages solution for public companies that need to gather data from their board of directors.
- When not working, my passion is spending all my spare time with my awesomely wonderful grand-kids. (WARNING: I have pictures ! Lots and Lots of Pictures !!)

IBM

ConnectED2015

## Agenda

- Application Responsiveness
- SSJS
- Modifying Component Output
- Java
- Event Handlers
- Custom Controls
- Debugging Tips
- Dojo

IBM

ConnectED2015



# APPLICATION RESPONSIVENESS

IBM

ConnectED2015

## Responsive Applications



IBM

Application Responsiveness

ConnectED2015

## Responsive Applications



IBM

Application Responsiveness

ConnectED2015

### (Not *That* Kind of) Responsive

- Functionally Responsive
- Minimize Server Requests
- Minimize Blocking
- Minimize Page Loading

### Minimize Blocking and Page Loading

- Client-side execution
  - Prevent page submission where possible
- Asynchronous processing

### JSON RPC

- Remote procedure call
  - Client / server interaction
- Asynchronous (not blocking)
- Faster
  - No page submission or refresh
- Simple



## JSON RPC

- Optionally accept parameters
- Optionally send response
- Optionally attach callback
- Caveat: No knowledge of page changes\*

## JSON RPC



## JSON RPC

- Extension library control (since 8.5.2)
- Controls Palette: Data Access > Remote Service  

```
<xe:jsonRpcService id="jsonRpcService1"></xe:jsonRpcService>
```

## JSON RPC

- Simple RPC Method

```
<xe:jsonRpcService id="jsonRpcService1" serviceName="rpcTest">
  <xe:this.methods>
    <xe:remoteMethod name="myMethod">
      <xe:this.script><![CDATA[
        print('Running a remote method');
      ]]></xe:this.script>
    </xe:remoteMethod></xe:this.methods>
  </xe:jsonRpcService>
```

- CSJS to Trigger

```
rpcTest.myMethod();
```

## JSON RPC

- RPC Method

```
<xe:remoteMethod name="ssjsMethod" script="return myFunction();">
</xe:remoteMethod>
```

- CSJS to Trigger and Add Callback

```
rpcTest.ssjsMethod().addCallback(function(response) {
  dg.addNotification(response, {'channel':'info', 'duration': 4000});
});
```

## JSON RPC

- Posted

```
{"params":[],"method":"ssjsMethod","id":1}
```

- Response

```
{"result":"The process has completed.<br><br>This message will self-
destruct.", "id":1 }
```

## JSON RPC

- Tips
  - Add script via property editor (not the blue diamond to compute the value)
  - Call a library function

## Appearing More Responsive

- Use visual indicator ("Loading" image, progress bar, text)
  - Let's user know something is happening
  - Prevents "elevator impatience"

## Appearing More Responsive



## Appearing More Responsive

- Replace view with spinner

```
<xp:button value="Search (Spinner - no delay)" id="button3">
  <xp:eventHandler event="onclick" submit="true"
    refreshMode="partial" refreshId="viewWrapper">
    <xp:this.script><![CDATA[
      var id="#{id:viewPanel1}";
      dojo.place('', dojo.byId(id), "replace");
    ]]></xp:this.script>
  </xp:eventHandler>
</xp:button>
```

## AJAX / Custom REST

- AJAX

- Asynchronously get or post data
- Display update when done

- Custom REST Service

- Process updates server-side
- Return response when done
- Requires R9 or newer Ext Lib ("custom" option not in 8.5.3 UP1)

## AJAX / Custom REST

- AJAX Example

```
dojo.xhr.get({
  url:"REST.xsp/MyData",
  handleAs:"json",
  load: function(data){
    dojo.byId('myDiv').innerHTML = data;
  },
  error: function(msg, args) {
    dojo.style('myDiv', 'color', 'red');
    dojo.byId('myDiv').innerHTML = 'Status: ' + args.xhr.status + '<br />' + msg;
  }
});
```

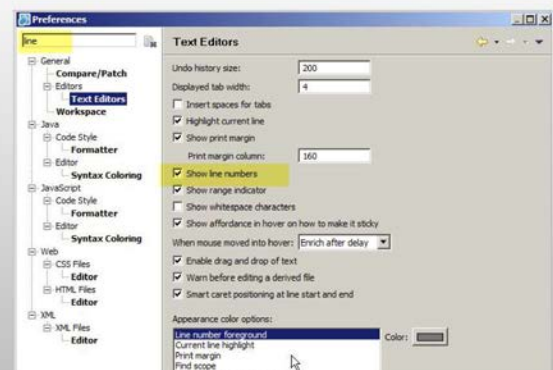
## Mixing Server-Side Logic into Client-Side JavaScript

- Evaluated on server; result rendered into page source
- Component ID:
  - `#{id: }`
- Computed Logic:
  - `#{javascript: }`
- Expression Language
  - `#{currentDocument.fieldName}`
- Does not work in script libraries

# SERVER SIDE JAVASCRIPT

## Quick Tips

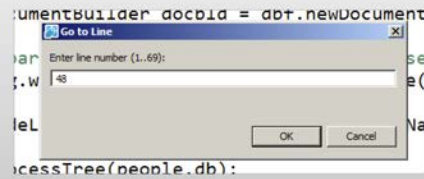
- Enable display of line numbers in Code editors.





## Quick Tips

- Goto specific line number in an open code editor
  - <ctrl> L



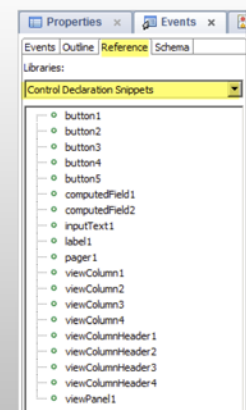
IBM

SSJS

ConnectED2015

## SSJS: Server Side JavaScript

- Every "Component" added to an XPage has a corresponding API
- This API allows developers access to a Components properties
- All Components API's are accessible via the "Resources" tab in the SSJS Editor



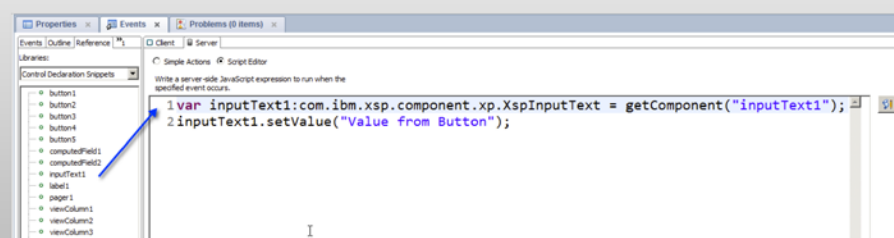
IBM

SSJS

ConnectED2015

## SSJS

- Double clicking the component in the reference window will add the component to the script window and assign it a variable name
- Methods can then be called against those variable names.



IBM

SSJS

ConnectED2015

## Demonstration



IBM

SSJS

ConnectED2015

## SSJS – Using libraries

- Once you have been coding XPages a while
  - Code Patterns will emerge
  - Same code used over and over
  - This is the universe telling you it's time to evolve to using libraries
- Once the library is created
  - It can be re-used easily in multiple applications.
  - Over time you will develop your “standard” library
    - A library added to the beginning of every application development project
- One of the most common libraries needed / used is for validation

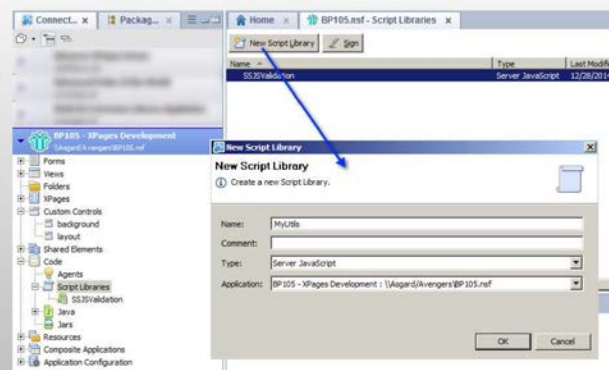
IBM

SSJS

ConnectED2015

## Create the library

- In Domino Designer
  - Expand the Code section
  - Select “Script Libraries
  - Click “New Script Library” action
  - Provide a Name
  - Set Type
    - Type CANNOT be changed once library is created



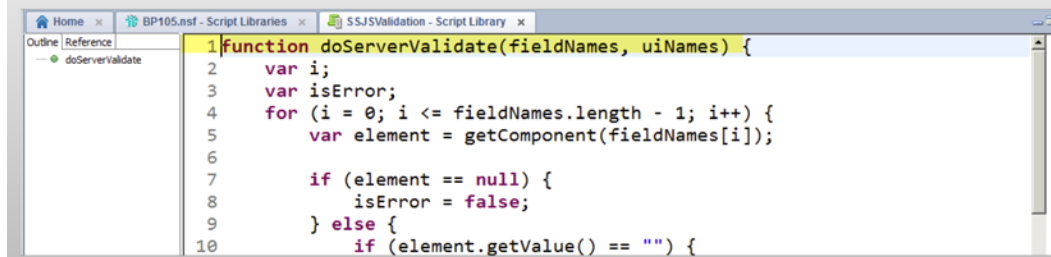
IBM

SSJS

ConnectED2015

## SSJS Libraries are typically a series of functions

- Create the functions that provide the functionality needed from the calling XPage
- Libraries can have multiple functions defined
- Each function is callable separately from the XPage that includes it.



```

1 function doServerValidate(fieldNames, uiNames) {
2     var i;
3     var isError;
4     for (i = 0; i <= fieldNames.length - 1; i++) {
5         var element = getComponent(fieldNames[i]);
6
7         if (element == null) {
8             isError = false;
9         } else {
10            if (element.getValue() == "") {

```

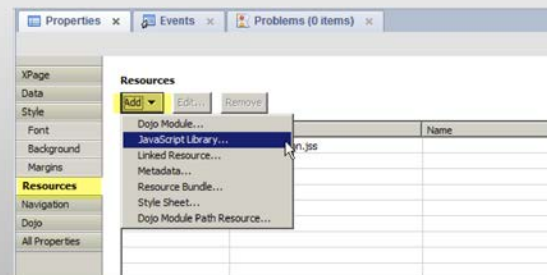
IBM

SSJS

ConnectED2015

## Add the Library to the XPage

- Open the XPage that will use the library
- On the Properties tab select "Resources"
- Click the "Add" button
- Choose "JavaScript Library..." from the list



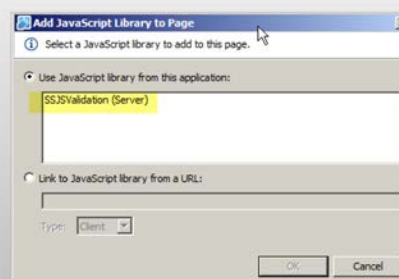
IBM

SSJS

ConnectED2015

## Add library to XPage

- From the dialog
  - Select the SSJS library to add to the XPage



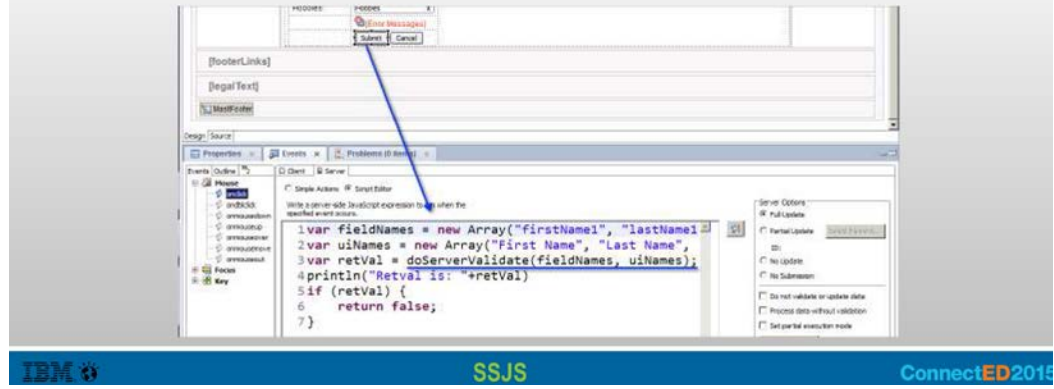
IBM

SSJS

ConnectED2015

## Call the functions of the Library

- Once the Library has been added to the XPage
  - Any of the defined functions are callable from SSJS code editors



## Demo



## MODIFYING COMPONENT OUTPUT

## tagName

- Computed Field (xp:text) or Panel property
- Default is <span> (computed field) or nothing (panel)
- Choose from list or manually enter

rendered	
renderType	
tagName	
data	div
converter	span
value	h1
format	h2
contentType	h3
styling	

## Suppressing Control Output Tag

- disableOutputTag property
- Repeat
  - Prevents <div> tag
  - Note: Cannot use control as partial refresh target
- Panel
  - No effect
- Computed Field
  - Prevents any output

## Suppressing <form> Tag

- createForm property of XPage
 


```
<xp:view xmlns:xp="http://www.ibm.com/xsp/core" createForm="false">
```
- Add your own <xp:form> tags to scope submission
 

```
<form id="view:_id4" method="post" action="/ConnectED.nsf/FormText.xsp"
      class="xspForm" enctype="multipart/form-data">
```
- Use carefully!
  - Client side events will run
  - No server-side events / page submission without a form tag



## Specifying the <body> class

- styleClass property of XPage
  - Adds the class to the <body> tag
  - <body class="myClass xspView tundra">
- styleClass property of Custom Control
  - Adds a <div> with the class

XPage	Property	Value
Data	basics	
Style	data	
Font	dojo	
Background	events	
Margins	styling	
Resources	disableTheme	
Navigation	style	
Dojo	styleClass	myClass 
All Properties	themeld	

## Control Attributes

- Useful for adding HTML5, Bootstrap, jQuery attributes
- attrs
  - name=value pairs to pass through
  - source:
 

```
<xp:inputText id="inputText1"> <xp:this.attrs>
  <xp:attr name="myFirstAttribute" value="value1"></xp:attr>
  <xp:attr name="mySecondAttribute" value="value2"></xp:attr>
</xp:this.attrs></xp:inputText>
```
  - output:
 

```
<input type="text" id="view:inputText1"
  name="view:inputText1" class="xspInputFieldEditBox"
  myFirstAttribute="value1" mySecondAttribute="value2">
```

## attrs - Example

- Example - dijit.TitlePane

▼ TitlePane
 

Declarative

- HTML – Declarative

```
<div id="div0" data-dojo-type="dijit/TitlePane" data-dojo-props="title: 'TitlePane'">
  Declarative
</div>
```

## attrs - Example

- Example - `dijit.TitlePane`

▼ Title Pane

HTML5-style using `attr`

- `<xp:div>` - HTML5 attributes via `attrs`

```
<xp:div id="div2">
  <xp:this.attrs>
    <xp:attr name="data-dojo-props" value="title: 'Title Pane'"></xp:attr>
    <xp:attr name="data-dojo-type" value="dijit.TitlePane"></xp:attr>
  </xp:this.attrs>
  HTML5-style using attr
</xp:div>
```

IBM

Modifying Component Output

ConnectED2015

## attrs

- minimized property – only include attribute name

— Source:

```
<xp:inputText id="inputText1">
  <xp:this.attrs>
    <xp:attr name="myFirstAttribute" value="value1"
      minimized="true"></xp:attr>
  </xp:this.attrs>
</xp:inputText>
```

— Output:

```
<input type="text" id="view:inputText1" name="view:inputText1"
  class="xspInputFieldEditBox" myFirstAttribute>
```

IBM

Modifying Component Output

ConnectED2015

## Themes

- Implement application-wide changes
- Conditionally render stylesheets

```
<resources>
  <script src="/ssjsMyLibrary.jss" clientSide="false"></script>
  <styleSheet href="/myStylesheet.css"
    rendered="#{javascript:context.getUserAgent().isFirefox()}"></styleSheet>
</resources>
```

- Does not work with IE11

— <http://openntf.org/XSnippets.nsf/snippet.xsp?id=detect-ie-11-in-xpages>

IBM

Modifying Component Output

ConnectED2015

## Themes

- Style fields consistently

```
<!-- Set all inputs to bootstrap styling by adding the form-control class -->
<control>
  <name>InputField.EditBox</name>
  <property mode="concat">
    <name>styleClass</name>
    <value>form-control</value>
  </property>
</control>
```

IBM

Modifying Component Output

ConnectED2015

## Converters

- Modify data display
- Several built-in

```
<xp:convertDateTime type="date"></xp:convertDateTime>

<xp:convertNumber type="currency" integerOnly="false"
  currencySymbol="$"></xp:convertNumber>

<xp:convertMask mask="(###) ###-####"></xp:convertMask>
```

IBM

Modifying Component Output

ConnectED2015

## Custom Converters

- Modify data display (not the output tags)
  - data > converter > xp:customConverter
  - SSJS
    - getAsString() - modify data to display
    - getAsObject() - modify data to store
    - Both required; both receive a *value* property to modify and return
  - Can also write in Java
    - <http://tobysamples.wordpress.com/xpages-custom-converters-change-the-data-change-the-world/>

IBM

Modifying Component Output

ConnectED2015

## Custom Renderers

- Modify data and tags
- 1. Create Java class for renderer to extend existing class for component
  - Receives component parameter – use to get value
- 2. Register the renderer in faces-config
- 3. Set the rendererType property of a control to class
  - Or, set it application-wide via theme

## Custom Renderers

- Examples
  - NotesIn9 by Jesse Gallagher
    - <http://www.notesin9.com/2014/10/02/notesin9-156-introduction-to-custom-renderers-in-xpages/>
  - Naveen Maurya
    - <http://www.pipalia.co.uk/xpages-2/creating-custom-renderers-for-xpages-controls/>
  - Keith Strickland
    - [https://www.socialbizug.org/blogs/keithstric/entry/add\\_custom\\_style\\_classes\\_to\\_ibm\\_renderers](https://www.socialbizug.org/blogs/keithstric/entry/add_custom_style_classes_to_ibm_renderers)

# JAVA

## Java: Why?

- We won't debate whether you should or should not learn Java.
  - (You should. Yesterday, today, tomorrow NOW !!!!)
- XPages is built upon a Java framework (JSF/JSP)
  - Even if you NEVER write a single line of Java code, you are executing Java code in the background
- The more you know about the framework(JSF/JSP) the better you will understand how to bend the framework to your will !! (insert evil laugh here)
- If you want to integrate XPages with non-Domino data/architectures then most of the code to do so is implemented in Java
  - JDBC
  - JSON
  - XML

## Java Code in XPages

- Java code can not be added directly to an XPage
  - This is largest barrier to entry for most developers
- Java code can be "called" from SSJS to execute java code methods
- Therefore;
  - Java code does NOT interact with the UI
  - Data (variables) passed to the back end java code comes from data input into the XPage and scope variables set prior to executing the code

## Java code execution from XPages

- The primary design pattern for executing Java code from an XPage is;
  - Call java code from an "event"
    - Like a buttons onClick() event
  - Call an "XAgent"



## What is an XAgent?

- An XAgent is an XPage with the rendering turned off
  - In Java terms, the XPage becomes a servlet
  - This requires the developer to handle the request/response
  - All of the code is in either the "beforeRenderResponse" or "afterRenderResponse" event
- What is this good for?
  - Many JavaScript frameworks require data sources that are only accessible by URL
    - An XAgent that returns JSON or XML can be used in conjunction with JavaScript graphing, charting and grid components
  - Incorporating Third Party Java Libraries
    - JDBC
    - APACHE POI
  - Excel and Word Document processing

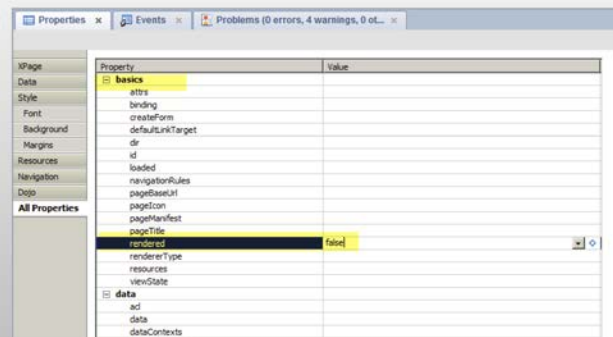
IBM

Java

ConnectED2015

## Create and XAgent

- Create an XPage
  - Set the "rendered" Property to false
  - This disables the output stream



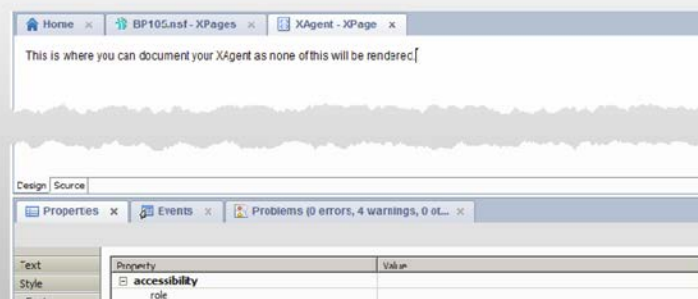
IBM

Java

ConnectED2015

## Tip: Document the XAgent

- Document the XAgent where you would normally place content



IBM

Java

ConnectED2015

## Code in the beforeRenderResponse event

- The code in the beforeRenderResponse event will

- Import Java code if needed
- Get JSF components
- Set the response type
- Call methods of the Java code
- Close the response object

```

1 importPackage(com.util);
2 //get the JSP response
3 //The Faces Context global object provides access to the servlet environment
4 //via the external context
5 var extCont = facesContext.getExternalContext();
6
7 //The servlet's response object provides control to the response output
8 var pageResponse = extCont.getResponse();
9
10 //Set the output stream to stream binary data
11 var pageOutput = pageResponse.getOutputStream();
12
13 //perform java calculations
14 var jcode:ExportToXML = new ExportToXML();
15 jcode.getXML(session,pageOutput);
16
17 //Set the response headers so that the browser displays the "save as/open" dialog
18 pageResponse.setContentType("application/xml");
19 pageResponse.setHeader("Cache-Control", "no-cache");
20 pageResponse.setHeader("Content-Disposition", "attachment; filename=xmliout.xml");
21
22 //Flush and close the output stream
23 pageOutput.flush();
24 pageOutput.close();
25 //Terminate the request processing lifecycle.
26 facesContext.responseComplete();
  
```

IBM

Java

ConnectED2015

## Demo



IBM

ConnectED2015

## EVENT HANDLERS

IBM

ConnectED2015

## Conditional Server-Side Event Execution

- Can have client-side and server-side event handlers on same component event
- return false; from client-side event to prevent server-side event execution
 

```
<xp:eventHandler event="onclick" submit="true" refreshMode="complete">
  <xp:this.script><![CDATA[
    return confirm('Do you want to run the server-side event handler?');]]>
  </xp:this.script>
  <xp:this.action><![CDATA[
    #{javascript:print ('Running server-side event handler')}
  ]]></xp:this.action>
</xp:eventHandler>
```
- Note: If the client-side code displays a prompt or message, server-side event will not wait

## Conditionally Render Event Handler

- Select `<xp:eventHandler>` tag in source and computed rendered property
- Example: Alphabet Bar
- Tip: Combine with conditional style class

## Getting Valid and ID of Component Triggering Event

- this** keyword returns an XspEventHandler (com.ibm.xsp.component.xp.XspEventHandler)
- Component Value
  - `this.getParent().getValue()`
- Component ID
  - `this.getParent.getId()`

## Event Parameters

- Send information to event handling code; can be computed
- Parameter vars available to reference directly in SSJS
 

```
<xp:eventHandler event="onclick" submit="true"
refreshMode="complete">
  <xp:this.parameters>
    <xp:parameter name="myEventParam" value=
      "#{javascript:return session.getEffectiveUserName();}">
    </xp:parameter>
  </xp:this.parameters>
  <xp:this.action>
    <![CDATA[#{javascript:print ('User: ' + myEventParam);}]]>
  </xp:this.action>
</xp:eventHandler>
```

Server Options

☒ Full Update

☐ Partial Update [Select Element...](#)

ID:

☐ No Update

☐ No Submission

☐ Do not validate or update data

☐ Process data without validation

☐ Set partial execution mode

[Select Element...](#) ID:

[Edit Event Parameters...](#)

Name	Value
myEve...	# return session.getEffectiv...

IBM

Event Handlers

ConnectED2015

## Run Client-Side JavaScript After Server-Side Event Handler

- onComplete property (select <xp:eventHandler> tag in source)
  - Attaches a callback
 

```
<xp:button value="Label" id="button1">
  <xp:eventHandler event="onclick" submit="true" refreshMode="partial"
refreshId="myPanel" onComplete="console.log('onComplete');">
    <xp:this.action><![CDATA[#{javascript:// Do Stuff}]]>
  </xp:this.action>
</xp:eventHandler>
</xp:button>
```
- Use case – display growl-style message
- onStart and onError callbacks also available
- Note: Will not run if event performs full page refresh

IBM

Event Handlers

ConnectED2015

## Order of Client-Side Event Execution

- 5 ways to run client JS on an event handler
  - Client JS script, onStart, onComplete, onerror\*, view.postScript (SSJS)
- Conditionally triggered based on event handler refresh mode
  - Full Refresh – client script, view.postScript
  - Partial Refresh – client script, onStart, view.postScript, onComplete
  - No Update – client script, onStart, onComplete
  - No Submission – client script
- Note – onStart, onerror, and onComplete are callbacks related to the server-side event

IBM

Event Handlers

ConnectED2015

## Debugging Java XPages

- If XPages make calls to Java Code elements then that Java code can be debugged using a similar process to debugging agents.
- In order to debug Java code called by an XPage add the following lines to the notes.ini
  - JavaEnableDebug=1
  - JavaDebugOptions=transport=dt\_socket,server=y,suspend=n,address=<port to listen on>
- GOTCHA !!
  - If the previous ports are enabled for java agent debugging then “connecting” to the port above will never work
  - Disable the Java Debug Ports and restart your Notes, Designer client prior to debugging Java called from XPages

## CUSTOM CONTROLS

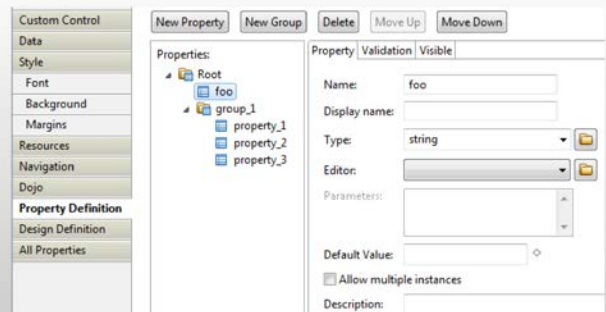
### Custom Controls != Subforms

- Conditional display
  - rendered property
- Custom properties = flexibility for reuse
- Combine with repeat controls for dynamic output



## Custom Properties

- Provide ability to customize each instance
- Data types
- Property editors
- Optionally required
- Single or multi-valued
- Combine into groups



IBM

Custom Controls

ConnectED2015

## Reading Custom Control Properties

- Within the Custom Control
  - `compositeData.propertyName`
- Outside of the Custom Control (SSJS)
  - Add an ID to the custom control instance  
`<xc:myCC propertyName="myValue" id="ccID"></xc:myCC>`
  - Access via `getComponent()`  
`getComponent('ccID').getPropertyMap().getProperty('propertyName');`
- Outside of the Custom Control (JavaScript)
  - `"#{javascript:getComponent('ccID').getPropertyMap().getProperty('propertyName')}`

IBM

Custom Controls

ConnectED2015

## Setting Custom Control Properties

- Within the Custom Control
  - `compositeData.myProperty = 'newValue';`
- Outside of the Custom Control
  - `getComponent('ccID').getPropertyMap().setProperty('myProperty', 'newValue');`

IBM

Custom Controls

ConnectED2015

## Setting Default Custom Property Values Via Theme

- Provides default value if not overridden on a specific instance
- CC
 

```
<xp:view xmlns:xp="http://www.ibm.com/xsp/core" themeId="CC_ThemeID">
  <xp:text escape="true" id="myField1" value="#{javascript:return
    compositeData.foo;}"></xp:text>
</xp:view>
```
- Theme
 

```
<control>
  <name>CC_ThemeID</name>
  <property>
    <name>foo</name>
    <value>bar_ccID</value>
  </property>
</control>
```

IBM

Custom Controls

ConnectED2015

## Setting Default Custom Property Values Via Theme

- Custom Control can have a themeld
- Each instance can have a themeld (but it has no effect)
- Property defined in theme overrides value defined on CC instance

IBM

Custom Controls

ConnectED2015

## Custom Controls as Reusable Fields

- Dynamic field binding – design once and reuse!
  - Cannot compute within EL, but can compute CC property to pass\
- Steps
  - 1. Add a custom property to custom control for fieldName
  - 2. Add a field and set value binding dynamically via EL
 

```
<xp:inputText id="myID"
  value="#{currentDocument[compositeData.fieldName]}">
</xp:inputText>
```
  - 3. On custom control instance, pass in name of field to use
 

```
<xc:ccMyControl fieldName="myFieldName"></xc:ccMyControl>
```

IBM

Custom Controls

ConnectED2015

# DEBUGGING TIPS

IBM

Debugging Tips

ConnectED2015

## Debugging Java XPages

- If XPages make calls to Java Code elements then that Java code can be debugged using a similar process to debugging agents.
- In order to debug Java code called by an XPage add the following lines to the notes.ini
  - JavaEnableDebug=1
  - JavaDebugOptions=transport=dt\_socket,server=y,suspend=n,address=<port to listen on>
- GOTCHA !!
  - If the previous ports are enabled for java agent debugging then "connecting" to the port above will never work
  - Disable the Java Debug Ports and restart your Notes, Designer client prior to debugging Java called from XPages

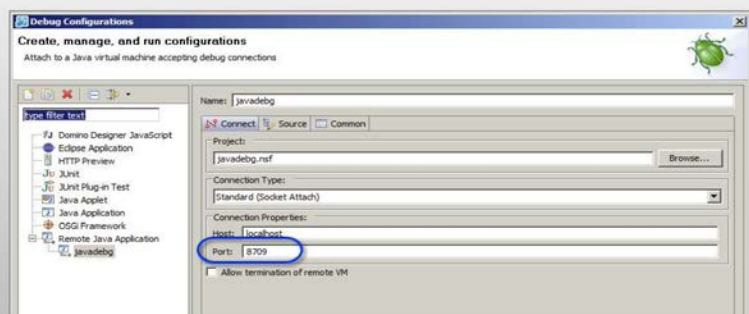
IBM

Debugging Tips

ConnectED2015

## Debug Configuration

- Create a new Debug Configuration to "listen" on the port that was set in the notes.ini file
- Preview the XPage to start the local web preview
- Start the debugger



IBM

Debugging Tips

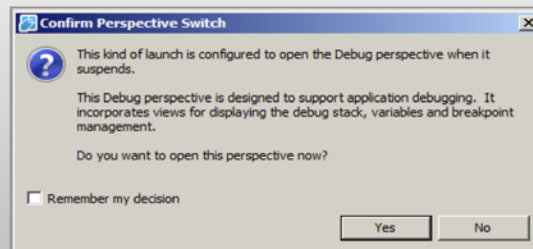
ConnectED2015

## Execute XPage action

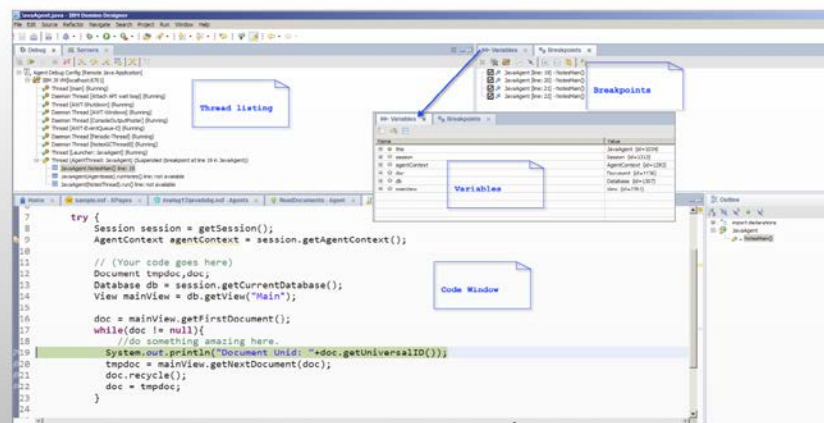
- Set breakpoints in the java code to be debugged
- In the XPage in the browser
  - Execute the action that will call the Java code
- If not already in the Debug perspective you will be prompted to switch to that perspective
- Step through your code reviewing the variable values at each breakpoint
- Clear breakpoints and resume to end
- Dis-connect from Debugger when done.

## Switch to the Debug Perspective

- If not already in the Debug Perspective a dialog will open prompting to switch to that perspective
  - Choose "Yes"
  - Optionally choose "Remember my decision" to suppress this dialog from popping up every time.









## The Debug Perspective



## Debugging

- The Code will stop at the first breakpoint
- From there you can...
  - Inspect variables
  - Step through the code using the icons

Icon	Name	Action
	Resume	This command resumes program execution.
	Suspend	This command suspends program execution.
	Terminate	This command terminates the selected debug target.
	Use Step Filters	This command steps into the highlighted statement applying the current set of step filters.
	Step Into	This command steps into the highlighted statement.
	Step Over	This command steps over the highlighted statement. Execution will continue at the next line either in the same method or, if you are at the end of a method it will continue in the method from which the current method was called. The cursor jumps to the declaration of the method and selects this line.

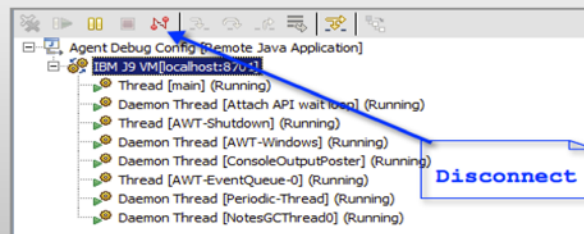
IBM

Debugging Tips

ConnectED2015

## Stop the Debugger

- After the debugging session is complete the debugger is still "listening" on the open port
- In the debug perspective make sure to "Disconnect" from the listening port to turn debugging off



IBM

Debugging Tips

ConnectED2015

## Demo



IBM

Debugging Tips

ConnectED2015



# DOJO

IBM

ConnectED2015

## Dojo Modules

- Many more modules available on server
  - Charting, animation, enhanced grid, etc.
- Dojo Module Resource (xp:dojoModule) adds a dojo.require() to include module
- Example:
 

```
<xp:this.resources>
  <xp:dojoModule name="dojo.fx"></xp:dojoModule>
</xp:this.resources>
```
- Output
 

```
<script type="text/javascript">dojo.require('dojo.fx')</script>
```

IBM

Dojo

ConnectED2015

## Using a Module Not on the Server

- Dojo Module Path
  - Used to make Dojo plugins available to XPages
  - Specify location where additional Dojo files are located

IBM

Dojo

ConnectED2015

## Dijits

- `dojoType` – automatically turns an element into a dijit
  - Must match a Dojo module name
  - `parseOnLoad` will parse and generate dijit
  - May need to set `dojoTheme` to true for styling
- Dojo Attributes
  - Define dijit properties
- `dijit.byId()` (not `dojo.byId()`)
  - Use `"#{id:}"` syntax

## Dijits

- Example - `dijit.TitlePane`

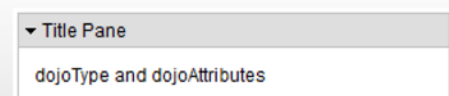


- HTML – Declarative

```
<div id="div0" data-dojo-type="dijit/TitlePane" data-dojo-props="title: 'TitlePane'">
  Declarative
</div>
```

## Dijits

- Example - `dijit.TitlePane`



- `<xp:div>` - `dojoType` and `dojoAttributes`

```
<xp:div id="div1" dojoType="dijit.TitlePane">
  <xp:this.dojoAttributes>
    <xp:dojoAttribute name="title" value="Title Pane"></xp:dojoAttribute>
  </xp:this.dojoAttributes>
  dojoType and dojoAttributes
</xp:div>
```

## Dijits

- Example - dijit.TitlePane



- <xp:div> - HTML5 attributs via attrs

```
<xp:div id="div2">
  <xp:this.attrs>
    <xp:attr name="data-dojo-props" value="title: 'Title Pane'"></xp:attr>
    <xp:attr name="data-dojo-type" value="dijit.TitlePane"></xp:attr>
  </xp:this.attrs>
  HTML5-style using attr
</xp:div>
```

## Dojo Controls

- Many available in extension library / 8.5.3 UP1 / Notes9
- Layout Controls
  - Accordion Container, Border Container, Stack Container
- Display Controls
  - Data Grid, Tab Container
- Input Controls
  - Filtering Select, Name Text Box, Slider, Number Spinner, etc.

# RESOURCES

## Resources

- Sample Data Generation Tool
  - <http://www.generatedata.com/>
- JavaScript Code Beautifier
  - <http://jsbeautifier.org/>
- Notes in Nine
  - <http://www.notesin9.com>
  - Episode Guide
    - <http://www.mindmeister.com/280533435/notesin9>
- TLCC
  - [www.tlcc.com](http://www.tlcc.com)

## Resources

- Head First Books
  - Java
  - JavaScript
  - Programming

## QUESTIONS?

Please remember to fill out evaluations!

## Engage Online

- **SocialBiz User Group** [socialbizug.org](http://socialbizug.org)
  - Join the epicenter of Notes and Collaboration user groups
- **Social Business Insights blog** [ibm.com/blogs/socialbusiness](http://ibm.com/blogs/socialbusiness)
  - Read and engage with our bloggers
- **Follow us on Twitter**
  - [@IBMConnect](https://twitter.com/IBMConnect) and [@IBMSocialBiz](https://twitter.com/IBMSocialBiz)
- **LinkedIn** <http://bit.ly/SBComm>
  - Participate in the IBM Social Business group on LinkedIn
- **Facebook** <https://www.facebook.com/IBMConnected>
  - Like IBM Social Business on Facebook



ConnectED2015

## Notices and Disclaimers

Copyright © 2015 by International Business Machines Corporation (IBM). No part of this document may be reproduced or transmitted in any form without written permission from IBM.

**U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.**

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY. IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

**Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com, BrassRing®, Connections™, Domino®, Global Business Services®, Global Technology Services®, SmartCloud®, Social Business®, Kenexa®, Notes®, PartnerWorld®, Prove It®, PureSystems®, Sanetime®, Verse™, Watson™, WebSphere®, Worklight®, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).



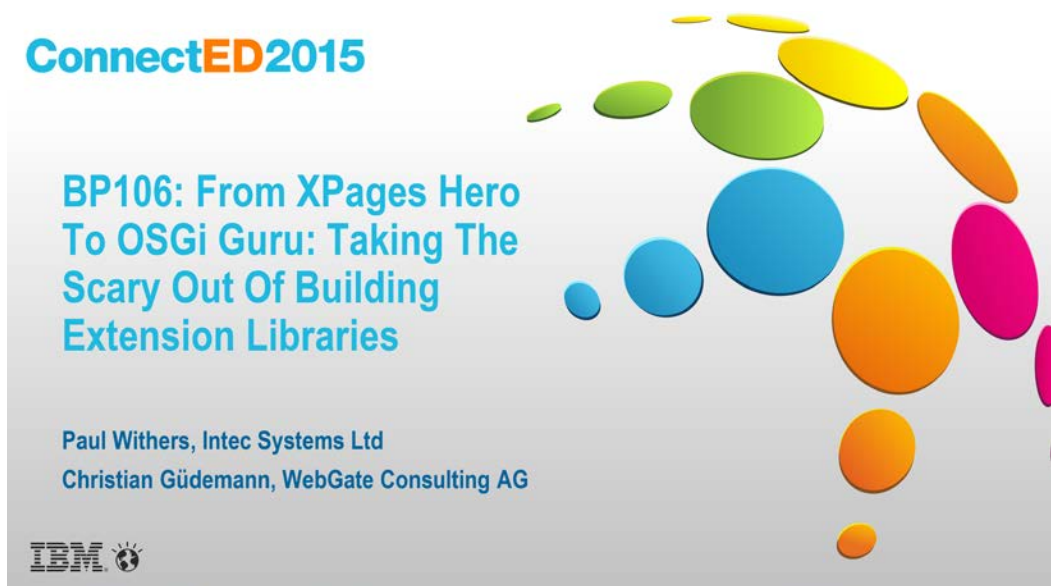
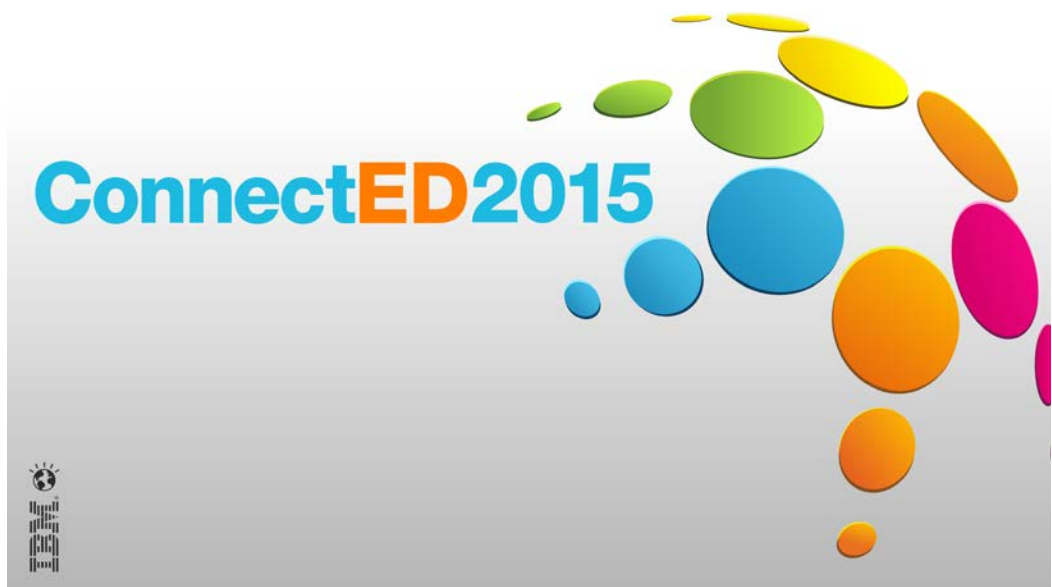
ConnectED2015



# From XPages Hero To OSGi Guru: Taking The Scary Out Of Building Extension Libraries

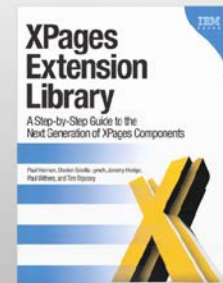
*Paul Withers, Intec Systems Ltd*

*Christian Güdemann, WebGate Consulting AG*



## Paul Withers

- XPages Developer since 2009
- IBM Champion
- OpenNTF Board Member
- Author of XPages Extension Library
- Developer XPages OpenLog Logger
- Co-Developer of OpenNTF Domino API



ConnectED2015

## Christian Güdemann

- IBM Champion
- OpenNTF Chairman
- Architect of XPages Toolkit, POI4XPages, JUnit4XPages and myWebGate
- Notes since Version 2
- Java since Version 1.2
- Eclipse since Version 3
- Freak... don't now when this started, but must be short after I've learned to spell computer



ConnectED2015

## Agenda

- Why?
- Development Environment & Debugging
- Repository Structure / Deployment
- Basic Plugin Structure
- Providing Client-Side Resources
- Providing Third-Party Java Classes
- Providing Components
- Summary



ConnectED2015

## XPages Developers

- [11 types of developer](#) - Stephan Wissel
- [Three types of developer](#) - Niklas Heidloff
- Five tiers of developers - Greg Reeder, in "[A Few Years of XPage Development](#)" series

Extension Library development should be  
the goal of XPages developers

IBM

ConnectED2015

## Why?

- Build Once, Use Anywhere (virtually!)
- Easier to deploy new versions and exploit replication
- Deploy third-party libraries
  - Apache POI etc.
  - JDBC Drivers
- Deploy client-side code server-wide
- More easily avoid Java security exceptions
  - Security tightened in Domino 9
- Required for DOTS (Domino OSGi Tasklet Service) or XPiNC XSP.executeCommand

IBM

ConnectED2015

## Examples Plugins

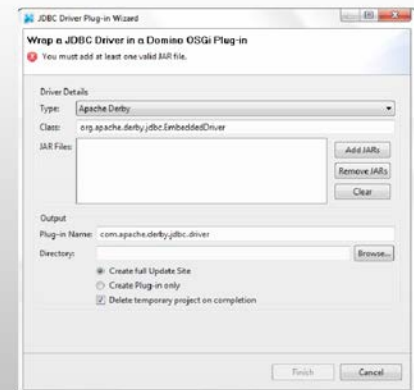
- XPages Extension Library
- IBM SBT
- XPages OpenLog Logger
- OpenNTF Domino API
- Bootstrap4XPages
- XPage Debug Toolbar
- XPages Scaffolding
- XPages Toolkit
- POI4XPages

IBM

ConnectED2015

## JDBC Driver Plugin Wizard

- Enhancement to RDBMS part of Ext Lib
  - Designer Client only functionality
- Requires latest version of ExtLib
- Allows a plugin to be created from one dialog for a JDBC driver
- See 3:15 on video  
<http://www.openntf.org/main.nsf/blog.xsp?permaLink=NHEF-9N7CKD>



IBM

ConnectED2015

## Agenda

- Why?
- Development Environment & Debugging
- Repository Structure / Deployment
- Basic Plugin Structure
- Providing Client-Side Resources
- Providing Third-Party Java Classes
- Providing Components
- Summary

IBM

ConnectED2015

## Configuring the Environment

- Eclipse
- XPages SDK
- Local Domino Server recommended
- See [Installation video](#) from Niklas Heidloff
  - Dates from before Debug Plugin was incorporated into XPages SDK
  - Dates from before Extension Library was Maven-ized
- See [blog series](#) from Paul Withers

IBM

ConnectED2015

## Eclipse for RCP and RAP Developers

- Latest release is [Luna](#)
  - Latest release of XPages SDK fixed to support this release
- Can have multiple versions installed



IBM

ConnectED2015

## XPages SDK

- Gives XPages JREs and Target Platforms
  - JRE → which plugins and jar files are automatically available
  - Target Platform → which application / OS plugins are built for
- From 4:40 in video
- Point Preferences to Domino and Notes installs
- Ensure “Automatically create JRE” ticked
- Tick relevant entry in Java > Installed JREs
- Create and select entry under Plug-in Development > Target Platform

IBM

ConnectED2015

## Debugging

- Debug Plugin now part of XPages SDK install
- Allows you to point Domino server direct to projects in relevant workspace
- From 7:44 in video
- Point Preferences to Domino install
- Configure Domino server for Java debugging, as for Java development
- Run > Debug Configurations
  - Create Debug Configuration
- Connect as with Domino
  - Use Step filters to skip certain packages

IBM

ConnectED2015



## OSGi Configuration

- Allows Domino Server to use plugins directly from Eclipse workspace
  - Direct access to source code (.java files), not compiled code (.class files)
  - Speeds up development / debugging
- Create new OSGi Framework configuration
  - Set as Domino OSGi Framework
  - Set auto-start to false
  - Click Debug – creates pde.launch.ini
  - Issue “res task http” command
  - Obviously will cause problems on networked server!

## When Plugins Are Created / Imported / Amended

- For added / imported plugins
  - Go to OSGi Framework configuration
  - Select the new plugin
  - Click Debug to update config
  - Issue “res task http” command
- If plugin is changed
  - Issue “res task http” command

## Agenda

- Why?
- Development Environment & Debugging
- Repository Structure / Deployment
- Basic Plugin Structure
- Providing Client-Side Resources
- Providing Third-Party Java Classes
- Providing Components
- Summary

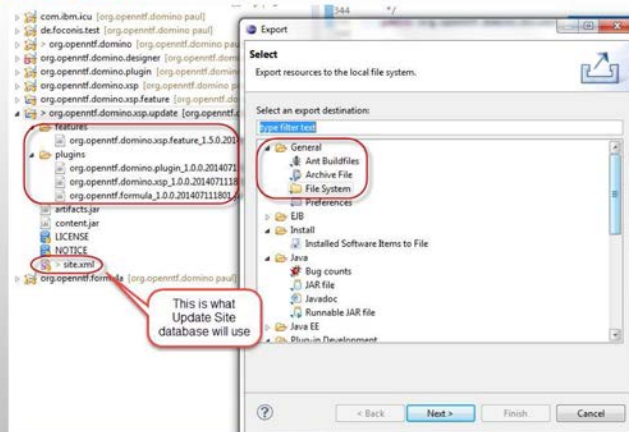
## Structure

- Project inter-relations will differ for Mavenized plugins
  - Maven is XML structure for automatically importing dependencies and building multiple plugins
  - Managed by a "parent" project, see Ext Lib demos or Christian's [blog series](#)
  - Maven has its own learning curve, so we'll skip that for now
- Plugin project
  - This is all that's needed for OSGi framework configuration
- Feature project loads one or more plugin
- Update Site project points to one or more feature
  - Creates plugins and features jars
  - Export as General > File System

IBM

ConnectED2015

## Update Site Project



IBM

ConnectED2015

## Deployment to Server / DDE

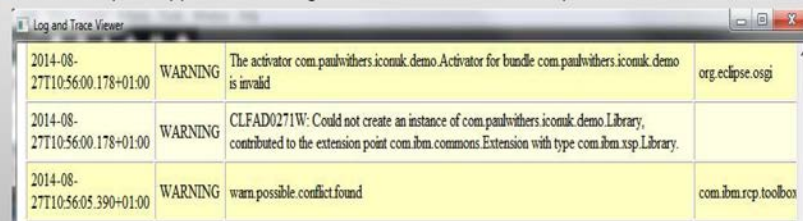
- Server:
  - Run from Eclipse using Domino Debug Plugin
  - Install to remote server as other Ext Libs
    - See Chapter 2 of XPages Extension Library pp28+
- Client
  - Install to DDE as other Ext Libs
  - Every change you make to the component re-install the update
  - Quite laborious for development, but know when you need to re-import and when you don't!
  - Add directly to DDE plugins
    - Best to create separate directory. See blog post by [John Cooper](#) or [slides at end](#)

IBM

ConnectED2015

## Troubleshooting

- Server
  - “tell http osgi diag *com.myplugin.name*” console command confirms any missing dependencies
- Client
  - check Help > Support > View Log and View Trace for errors / print statements

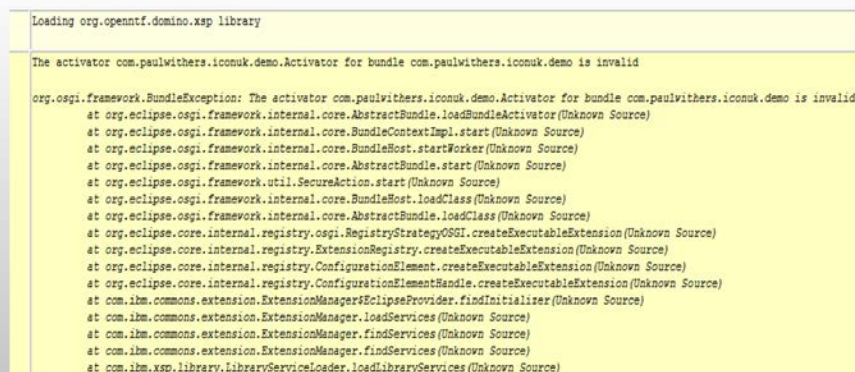


Time	Level	Message	Source
2014-08-27T10:56:00.178+01:00	WARNING	The activator com.paulwithers.iconuk.demo.Activator for bundle com.paulwithers.iconuk.demo is invalid	org.eclipse.osgi
2014-08-27T10:56:00.178+01:00	WARNING	CLFAD0271W: Could not create an instance of com.paulwithers.iconuk.demo.Library, contributed to the extension point com.ibm.commons.Extension with type com.ibm.xsp.Library.	
2014-08-27T10:56:05.390+01:00	WARNING	warn.possible.conflict.found	com.ibm.rcp.toolbox

IBM

ConnectED2015

## Troubleshooting



Message
Loading org.openntf.domino.xsp.library
The activator com.paulwithers.iconuk.demo.Activator for bundle com.paulwithers.iconuk.demo is invalid
org.eclipse.osgi.framework.BundleException: The activator com.paulwithers.iconuk.demo.Activator for bundle com.paulwithers.iconuk.demo is invalid
at org.eclipse.osgi.framework.internal.core.AbstractBundle.loadBundleActivator(Unknown Source)
at org.eclipse.osgi.framework.internal.core.BundleContextImpl.start(Unknown Source)
at org.eclipse.osgi.framework.internal.core.BundleHost.startWorker(Unknown Source)
at org.eclipse.osgi.framework.internal.core.AbstractBundle.start(Unknown Source)
at org.eclipse.osgi.framework.util.SecureAction.start(Unknown Source)
at org.eclipse.osgi.framework.internal.core.BundleHost.loadClass(Unknown Source)
at org.eclipse.osgi.framework.internal.core.AbstractBundle.loadClass(Unknown Source)
at org.eclipse.core.internal.registry.osgi.RegistryStrategyOSGI.createExecutableExtension(Unknown Source)
at org.eclipse.core.internal.registry.ExtensionRegistry.createExecutableExtension(Unknown Source)
at org.eclipse.core.internal.registry.ConfigurationElement.createExecutableExtension(Unknown Source)
at org.eclipse.core.internal.registry.ConfigurationElementHandle.createExecutableExtension(Unknown Source)
at com.ibm.commons.extension.ExtensionManager\$EclipseProvider.findInitializer(Unknown Source)
at com.ibm.commons.extension.ExtensionManager.loadServices(Unknown Source)
at com.ibm.commons.extension.ExtensionManager.findServices(Unknown Source)
at com.ibm.commons.extension.ExtensionManager.findServices(Unknown Source)
at com.ibm.xsp.library.LibraryServiceLoader.loadLibraryServices(Unknown Source)

IBM

ConnectED2015

## Deployment to Development Team / XPINC

- Add to Widget Catalog from Update Site database
  - See XPages Extension Library pp40+
- Best practice is using Desktop Policy, ensures updates automatically deployed



Desktop Settings : Demo Desktop Policy		
Basic   Smart Upgrade   Applications   Widgets   Dial-up Connections   Accounts   Name Servers   SSI		
Widget Settings	How to apply this setting:	
Widget catalog server:	Heracles/Intec-pw_1	Set initial value
Widget catalog application name:	Intec/Widget.nsf_1	Set initial value
Widget catalog categories to install:	Developed Tools_1	Set initial value

IBM

ConnectED2015

## Agenda

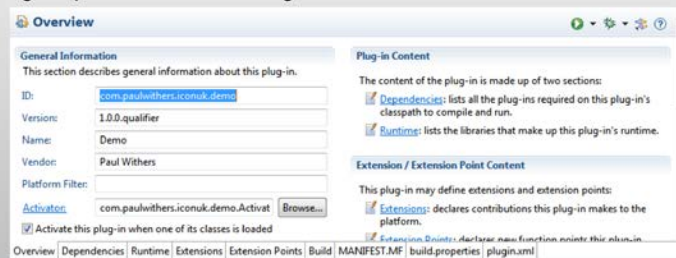
- Why?
- Development Environment & Debugging
- Repository Structure / Deployment
- Basic Plugin Structure
- Providing Client-Side Resources
- Providing Third-Party Java Classes
- Providing Components
- Summary

IBM

ConnectED2015

## Plugin Structure

- See [Extensibility API Developers Guide](#)
- Activator is optional
  - Allows generic code to be run
- Extend `org.eclipse.core.runtime.Plugin`

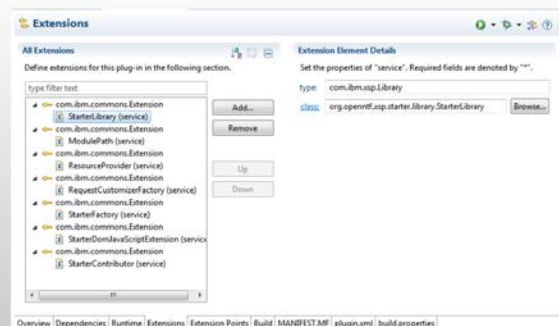


IBM

ConnectED2015

## Extensions Tab

- Extensions load other Java classes
- Use extension point `com.ibm.commons.Extension`
- Use "tell http osgi pt -v com.ibm.commons.Extension" to see types and classes currently loaded



IBM

ConnectED2015

## Library Class

- This is what is selected in Xsp Properties
- Type: `com.ibm.xsp.Library`
- Class: `your.package.Library` extends `AbstractXspLibrary`
- Defines
  - Dependencies
  - Faces-Config files
  - Xsp-Config files

IBM

ConnectED2015


## Contributor Class

- This adds factories
  - Holds server-level maps
  - Load implicit objects (variables)
- Type: `com.ibm.xsp.library.Contributor`
- Class `your.package.Contributor` extends `XspContributor`


IBM

ConnectED2015


## From XSP Starter Kit to Clean Plugin


**Renamed main package**


Updated Activator to run in debug mode  
 Moved Library to main package and renamed  
 Moved Utils class to main package, renamed and cleared down


**Cleaning up Library**

Amending namespace and prefix  
 Clearing down XspConfig Files and FacesConfig files


**Deleted superfluous packages**

NOTE: plugin.xml still needs updating  
 paulswithers authored 30 minutes ago


**Updates to plugin.xml and associated files:**

Overview tab: Renamed plugin and updated Execution Environment  
 Dependencies: Amended version for `com.ibm.xsp.designer`  
 Runtime: Removed deleted packages from Exported Packages area  
 Extensions: Removed deleted extensions  
 Plugin.xml: Deleted commented out extension point

IBM

ConnectED2015



## Agenda

- Why?
- Development Environment & Debugging
- Repository Structure / Deployment
- Basic Plugin Structure
- Providing Client-Side Resources
- Providing Third-Party Java Classes
- Providing Components
- Summary

# PROVIDING CLIENT-SIDE RESOURCES

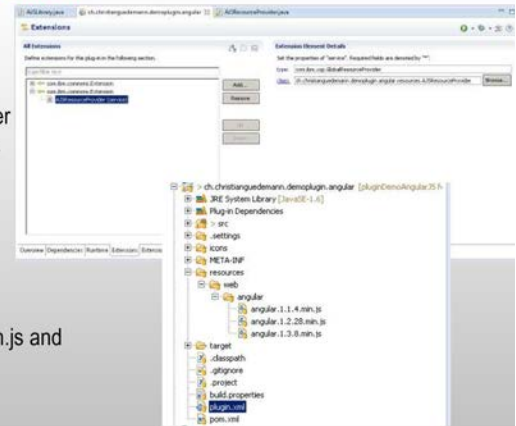
Angular 4 ALL

## Angular for all!

- Angular.JS is popular java script framework to build client-side applications.
- XPages Developers typically distribute Angular to their application by adding the script library to the WebContent folder
- But let me show how easy it is to deploy angular as part of a plugin and imagine how easy it would be to deploy your java script standard components

## Making your plugin to a resource provider

- Time to contribute to an extension
  - com.ibm.commons.Extension
  - Type: com.ibm.xsp.GlobalResourceProvider
  - Class: your.package.ResourcesProvider -> extends BundleResourceProvider
- Add some folders to the plugin
  - resources/web/angular
  - put angular.1.3.8.min.js, angular.1.2.28.min.js and angular.1.1.4.min.js in the folder



IBM

ConnectED2015

## Lets write some code to make the files available as .ibmxspres/.angular/xxx.js

- The following code let the resource provider understand where he can find the .js files:

```
public class AJSResourceProvider extends BundleResourceProvider {

    public static final String RESOURCES_WEB_ANGULAR = "/resources/web/angular/";
    public static final String ANGULAR_PREFIX = ".angular";

    public static ScriptResource ANGULAR_1_1_4 = new ScriptResource("/.ibmxspres/" + ANGULAR_PREFIX + "/angular.1.1.4.min.js", true);
    public static ScriptResource ANGULAR_1_2_28 = new ScriptResource("/.ibmxspres/" + ANGULAR_PREFIX + "/angular.1.2.28.min.js", true);
    public static ScriptResource ANGULAR_1_3_8 = new ScriptResource("/.ibmxspres/" + ANGULAR_PREFIX + "/angular.1.3.8.min.js", true);

    private static Map<String, ScriptResource> ANGULAR_VERSIONS = new HashMap<String, ScriptResource>() {
        private static final long serialVersionUID = 1L;

        {
            put("1.1.4", ANGULAR_1_1_4);
            put("1.2.28", ANGULAR_1_2_28);
            put("1.3.8", ANGULAR_1_3_8);
            put("latest", ANGULAR_1_3_8);
        }
    };

    public AJSResourceProvider() {
        super(AJSActivator.getInstance().getBundle(), ANGULAR_PREFIX);
    }
}
```

IBM

ConnectED2015

## Lets write some code to make the files available as .ibmxspres/.angular/xxx.js

```
@Override
protected URL getResourceURL(HttpServletRequest arg0, String name) {
    String path = RESOURCES_WEB_ANGULAR + name; // $NON-NLS-1$
    int fileNameIndex = path.lastIndexOf('/');
    String fileName = path.substring(fileNameIndex + 1);
    path = path.substring(0, fileNameIndex + 1);
    // see http://www.osgi.org/javadoc/r4v42/org/osgi/framework/Bundle.html
    // #findEntries%28java.lang.String,%20java.lang.String,%20boolean%29
    Enumeration<?> urls = getBundle().findEntries(path, fileName, false/* recursive */);
    if (null != urls && urls.hasMoreElements()) {
        URL url = (URL) urls.nextElement();
        if (null != url) {
            return url;
        }
    }
    return null; // no match, 404 not found.
}

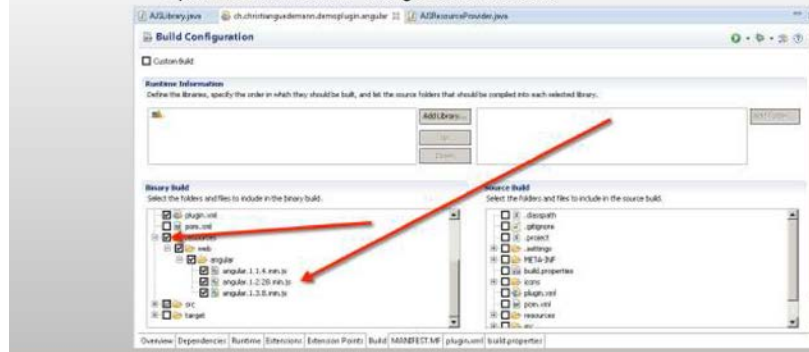
public static ScriptResource getAngularScriptLibrary(String version) {
    if (ANGULAR_VERSIONS.containsKey(version)) {
        return ANGULAR_VERSIONS.get(version);
    }
    return null;
}
```

IBM

ConnectED2015

## Don't forget to export the resources during the build

- You won't believe how many times I was struggling at this point.
  - Open the Manifest.MF and go to the tab build



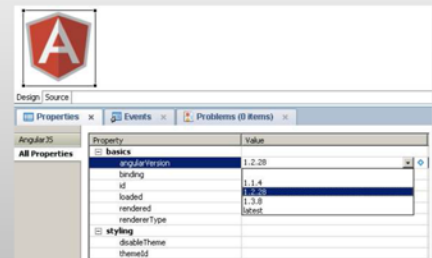
IBM

ConnectED2015

## All done? How can a developer now consume this java script libs?

- He knows the following url statement `/ibmxspres/.angular/angular.1.3.8.min.js` → UGLY
- We can build a custom theme and provide this theme (like the Bootstrap4XPages Project does)

- We build a component and gives the user choice.



IBM

ConnectED2015

## The component (A java representation of an XPages Element)

- The following Code represents the component (Paul will later explain more about)

```
public class UIAngular extends UIComponentBase {

    public static final String COMPONENT_TYPE = "ch.christianguedemann.demoplugin.uiangular"; //$NON-NLS-1$
    public static final String COMPONENT_FAMILY = "ch.christianguedemann.demoplugin.uiangular"; //$NON-NLS-1$
    public static final String RENDERER_TYPE = "ch.christianguedemann.demoplugin.uiangular"; //$NON-NLS-1$

    private String m_AngularVersion;

    public UIAngular() {
        setRendererType(RENDERER_TYPE);
    }

    @Override
    public String getFamily() {
        return COMPONENT_FAMILY;
    }

    public String getAngularVersion() {
        return m_AngularVersion;
    }

    public void setAngularVersion(String angularVersion) {
        m_AngularVersion = angularVersion;
    }

    @Override
    public void restoreState(FacesContext context, Object value) {
        Object[] values = (Object[]) value;
        super.restoreState(context, values[0]);
        m_AngularVersion = (String) values[1];
    }

    @Override
    public Object saveState(FacesContext context) {
        Object[] values = new Object[2];
        values[0] = super.saveState(context);
        values[1] = m_AngularVersion;
        return values;
    }
}
```

IBM

ConnectED2015

### Angular.xsp-config (make the component visible in the DDE)

- Here the .xsp-config file to make the component visible in the DDE (Paul will also explain this later ; )

[illegible]

ConnectED2015

### The renderer (This piece of code brings the angular.js on your page)

- (Yes Paul will also explain this!)

```
public class AngularJSRenderer extends FacesRenderer {

    @Override
    public void encodeBegin(FacesContext context, UIComponent component) throws IOException {
        if (!component.isRendered()) {
            return;
        }
        if (!component instanceof UIAngular) {
            return;
        }
        UIAngular angularJS = (UIAngular) component;
        String version = angularJS.getAngularVersion();
        ScriptResource resource = AJSRResourceProvider.getAngularScriptLibrary(version);
        if (resource == null) {
            throw new FacesException("Angular JS Library with version " + version + " not found!");
        }
        UIViewRootEx rootEx = (UIViewRootEx) context.getViewRoot();
        rootEx.addEncodeResource(resource);
    }
}
```



ConnectED2015

Angular-faces-config.xml -> Instruction for the renderer

```
<faces-config>
  <render-kit>
    <renderer>
      <component-family>ch.christianguedemann.demoplugin.uiangular</component-family>
      <renderer-type>ch.christianguedemann.demoplugin.uiangular</renderer-type>
      <renderer-class>ch.christianguedemann.demoplugin.angular.renderkit.html_extended.AngularJSRenderer</renderer-class>
    </renderer>
  </render-kit>
</faces-config>
```



ConnectED2015



## Demo and Summary

- Imagine that you can in the same way multiply the usage of your brilliant java script code or the CSS style sheet for your cooperate design
- Imagine that you can also customize the look and feel of all Application-Layout based XPages Apps by defining a plugin with some resources and an new Theme
- Imagine how productive your development team can become because you have make your work easy consumable

IBM

ConnectED2015

## Agenda

- Why?
- Development Environment & Debugging
- Repository Structure / Deployment
- Basic Plugin Structure
- Providing Client-Side Resources
- Providing Third-Party Java Classes
- Providing Components
- Summary

IBM

ConnectED2015

## Plugin for Third Party Jars

- Create a separate plugin
  - New > Plug-in from Existing JAR Archives
  - For additional jars
    - Import the jar
    - Add to Build Path
    - Ensure included in Binary Build on build.properties
  - Also blog post by [John Dalsgaard](#)
  - Edit META-INF/MANIFEST.MF and add **.qualifier** to version
    - Always adds a new version to plugin folder of Update Site

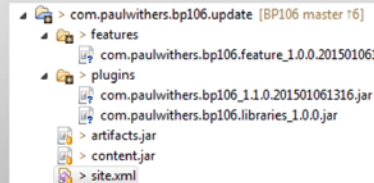
IBM

ConnectED2015



## Including Plugin

- Add as Required Plug-in to plugin.xml
- Click on Properties and tick "Reexport this dependency"
- Add to feature
- Ensure "Unpack the plug-in archive after the installation" is ticked
  - Otherwise DDE will not see the jars



## Agenda

- Why?
- Development Environment & Debugging
- Repository Structure / Deployment
- Basic Plugin Structure
- Providing Client-Side Resources
- Providing Third-Party Java Classes
- Providing Components
- Summary

## Creating Component Plugin

- Take a custom control and make it global
- [NotesIn9 64 by Tim Tripcony](#) #codefortim
- Or code within Eclipse
- [Extensibility API 9.0.1](#)

## Classes for Component

- Component class **DOMINO** / **DDE** (Setters / Adders only)
- .xsp-config to add properties **DDE**
  - See <http://avatar.red-pill.mobi/tim/blog.nsf/d6plinks/TTRY-8DDDEZ> for making Eclipse identify \*.xsp-config files as XML files #codefortim
- Renderer class, if required **DOMINO**
  - Use getRendererType() to find an existing renderer
- faces-config.xml to add renderer **DOMINO**
- Load xsp-config and faces-config.xml in Library class

## Demo Plugin

- Add component for Separator
- Allow properties for:
  - separatorType (New Line / Space)
  - count (integer, defaulting to 1)
- Deploy org.apache.commons.lang3
- Add utility method to convert any object to string detailing properties

## Agenda

- Why?
- Development Environment & Debugging
- Repository Structure / Deployment
- Basic Plugin Structure
- Providing Client-Side Resources
- Providing Third-Party Java Classes
- Providing Components
- Summary

## Links to Demos

- <https://github.com/paulswithers/BP106>
  - Demo database is in notes folder

IBM

ConnectED2015

## Thank You

- |   |  |
|---|--|
| • Paul Withers  | • Christian Güdemann   |
| • <a href="mailto:pwithers@intec.co.uk">pwithers@intec.co.uk</a>                | • <a href="mailto:Christian.guedemann@webgate.biz">Christian.guedemann@webgate.biz</a> |
| • <a href="http://www.intec.co.uk/blog">http://www.intec.co.uk/blog</a>         | • <a href="http://guedebyte.wordpress.com">http://guedebyte.wordpress.com</a>          |
| • <a href="http://twitter.com/paulswithers">http://twitter.com/paulswithers</a> | • <a href="http://twitter.com/guedeWebGate">http://twitter.com/guedeWebGate</a>        |

IBM

ConnectED2015

## Engage Online

- **SocialBiz User Group** [socialbizug.org](http://socialbizug.org)
  - Join the epicenter of Notes and Collaboration user groups
- **Social Business Insights blog** [ibm.com/blogs/socialbusiness](http://ibm.com/blogs/socialbusiness)
  - Read and engage with our bloggers
- **Follow us on Twitter**
  - [@IBMConnect](https://twitter.com/IBMConnect) and [@IBMSocialBiz](https://twitter.com/IBMSocialBiz)
- **LinkedIn** <http://bit.ly/SBComm>
  - Participate in the IBM Social Business group on LinkedIn
- **Facebook** <https://www.facebook.com/IBMConnected>
  - Like IBM Social Business on Facebook

IBM

ConnectED2015

## Notices and Disclaimers

Copyright © 2015 by International Business Machines Corporation (IBM). No part of this document may be reproduced or transmitted in any form without written permission from IBM.

**U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.**

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY. IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

**Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com, BrassRing®, Connections™, Domino®, Global Business Services®, Global Technology Services®, SmartCloud®, Social Business®, Kenexa®, Notes®, PartnerWorld®, Prove It!®, PureSystems®, Sametime®, Verse™, Watson™, WebSphere®, Worklight®, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).



ConnectED2015

## Configuring DDE for Adding Plugin Directly

- Navigate to framework directory inside the Notes Data Directory
- Create a new plugin directory
- Create a .link file
- Inside your newly created text file add the following:
  - path=C:\Program Files (x86)\IBM\Notes\framework\pluginsExt
- Update platform.xml
- Change the transient attribute on the config tag to false
- Replace all the instances of policy="MANAGED-ONLY" to policy="USER-EXCLUDE"



ConnectED2015

## Exporting Plugins Directly to DDE

- Export plugin as a "Deployable Plugin and Fragment"
- Put in newly created directory
- Restart designer



ConnectED2015

# Ten Lines Or Less: Interesting Things You Can Do In Java With Minimal Code

*Julian Robichaux, panagenda*

*Kathy Brown, PSC Group*

IBM Software

**ConnectED2015**

## BP107: Ten Lines Or Less Interesting Things You Can Do In Java With Minimal Code

Julian Robichaux, panagenda

Kathy Brown, PSC Group



### Why are we here?

- Java snippets for IBM Notes® and Domino®
  - 10 lines or less
  - beginner-to-intermediate level stuff
  - hopefully useful
- Integration tips
  - XPages only
  - Gotchas
- Sample database
  - <http://www.runningnotes.net> and/or <http://nsftools.com/blog>



ConnectED2015



## The 10-Line Rules

### Rule #1

The method signature is not part of the line count

```
public String getFoo() {  
    return "foo";  
}
```

This only  
counts as one  
line

IBM

ConnectED2015

## The 10-Line Rules

### Rule #2

A single line with linefeeds added for readability is still just one line

```
public String getFoos() {  
    return "foo" +  
        "foo" +  
        "foofoofoo";  
}
```

This only  
counts as one  
line

IBM

ConnectED2015

## The 10-Line Rules

### Rule #3

We make the rules

```
public String getFoot() {  
    Foot foot = new Foot();  
    foot.addToes(12).makeHairy();  
    return foot.toString();  
}
```

This is as many  
lines as we say  
it is

IBM

ConnectED2015

## The 10-Line Rules

### Rule #4

Error handling and object cleanup has sometimes been omitted for brevity

PLEASE use good error handling in your production code



ConnectED2015

## About the Sample Application

- As you travel, you can upload pictures from your smartphone to this app
- Upon upload, the app will:
  - Read GPS data from the picture
  - Use a web service to obtain the address
  - Create a thumbnail of the photo
  - Embed the thumbnail into a rich text field
- Other functionality:
  - Return info as JSON
  - Return info as a spreadsheet
  - FTP to a server



ConnectED2015

# GETTING STARTED

The basics. Where does this stuff go? How do we access Notes objects?



ConnectED2015

## Where does our Java code live?

- How to add Java design elements to Code > Java and Code > Jars
  - Jars can be copy/pasted, or dragged/dropped, or “Import Jar”
  - Java classes can be copy/pasted, dragged/dropped, or “Create New Class”
- What's the difference?
  - A Jar (Java ARchive) file is a file containing one or more Java classes

## Where does our Java code live?

- NOTE: XPages only, agents are different
  - Agents don't have access to Java design elements
- Just code for now, we'll tackle external libraries later
- Java code can also go in WebContent/WEB-INF/lib or on the host system
  - Paul Calhoun and Paul Della-Nebbia have several slides on benefits of each location in their Java Jumpstart, <http://www.slideshare.net/Teamstudio/java-for-xpages-development>

## Java 1.5

- IBM Notes/Domino 8.5 and 9.0 uses Java 1.6 as a JVM
- Java design elements (for XPages) compile to version 1.5 by default
  - Enhanced for loops
  - Auto-boxing
  - Enums
  - Generics
- Java agents compile to version 1.2 by default
  - Can be changed with notes.ini variable: JavaCompilerTarget
  - Agents might not run on older clients or servers

## Example: Getting Document Attachments [1]

```
public EmbeddedObject getFirstDocAttachment(Document doc)
    throws NotesException {
    Session session = doc.getParentDatabase().getParent();
    Vector atts = session.evaluate("@AttachmentNames", doc);
    String firstFile = (String)atts.get(0);
    if (firstFile.length() > 0) {
        EmbeddedObject eo = doc.getAttachment(firstFile);
        return eo;
    }
    return null;
}
```

Here's one way to get a Notes Session

Everything else is pretty much like LotusScript

IBM

ConnectED2015

## Example: Getting Document Attachments [2]

```
public EmbeddedObject getFirstRichTextAttachment(RichTextItem rtItem)
    throws NotesException {
    Vector atts = rtItem.getEmbeddedObjects();
    for (Iterator iterator = atts.iterator(); iterator.hasNext();) {
        EmbeddedObject eo = (EmbeddedObject) iterator.next();
        if (eo.getType() == EmbeddedObject.EMBED_ATTACHMENT) {
            return eo;
        }
    }
    return null;
}
```

Iterator loop, in case you want to use this in an agent

Notes API still uses Vectors quite often as return objects

IBM

ConnectED2015

## Passing Current Document and Calling Java

```
importPackage(com.lotusphere.bp107_2015);
var doc:NotesDocument = currentDocument.getDocument();
var eo:lotus.domino.EmbeddedObject = TenLinesOrLess.getFirstDocAttachment(doc)
```

JavaScript

Java

```
public static EmbeddedObject getFirstDocAttachment(Document doc)
    throws NotesException {
    Session session = doc.getParentDatabase().getParent();
    Vector atts = session.evaluate("@AttachmentNames", doc);
    String firstFile = (String)atts.get(0);
    if (firstFile.length() > 0) {
        EmbeddedObject eo = doc.getAttachment(firstFile);
        return eo;
    }
    return null;
}
```

IBM

ConnectED2015

# WORKING WITH FILES

Reading, writing, and accessing files. Permissions, considerations, and working with byte arrays directly.

IBM

ConnectED2015

## Basic File IO

- Java keeps improving file IO classes
  - Java 1.5 added Closable interfaces
  - Java 1.7 added java.nio.file classes ("Files" is especially useful)
  - <http://docs.oracle.com/javase/tutorial/essential/io/legacy.html>
- By default, Notes/Domino Java elements compile to **Java 1.5** compatible code
  - You can change to 1.6 in project properties, but this might break things
  - No significant java.io changes in 1.6 anyway: <http://docs.oracle.com/javase/6/docs/technotes/guides/io/enhancements.html>

IBM

ConnectED2015

## Example: Reading a Text File into a String [1]

```
public String readTextFileScanner(String fileName) throws IOException {
    // Scanner can be flaky about sometimes not returning all the data,
    // and can be slower because it's using regex internally
    StringBuilder text = new StringBuilder();
    String crlf = System.getProperty("line.separator");
    Scanner scanner = new Scanner(new FileInputStream(fileName), "UTF-8");
    while (scanner.hasNextLine()){
        text.append(scanner.nextLine() + crlf);
    }
    scanner.close();
    return text.toString();
}
```

**Scanner can be flaky on large strings, and you might be replacing \n with \r\n for linefeeds**

**using the right character set is important!**

IBM

ConnectED2015



## Example: Reading a Text File into a String [2]

```
public String readTextFileReader(String fileName) throws IOException {
    // StringBuilder is faster than StringBuffer because it's not
    // synchronized
    StringBuilder sb = new StringBuilder();
    BufferedReader reader = new BufferedReader(
        new InputStreamReader(new FileInputStream(fileName), "UTF-8"));
    char[] buffer = new char[1024];
    int size = 0;
    while((size = reader.read(buffer)) != -1) {
        sb.append(buffer, 0, size);
    }
    reader.close();
    return sb.toString();
}
```

a tiny bit more work,  
but slightly faster  
and probably more  
reliable

again with the  
character set

## Example: Writing a File to Another File [1]

```
public void copyFileChannel(String fileToRead, String fileToWrite)
    throws IOException {
    // you should add try/finally blocks to make sure streams get closed
    FileInputStream in = new FileInputStream(fileToRead);
    FileOutputStream out = new FileOutputStream(fileToWrite, false);
    // FileChannel is faster but generally uses more memory; you can
    // control this somewhat by using FileChannel.map()
    FileChannel channel1 = in.getChannel();
    FileChannel channel2 = out.getChannel();
    channel1.transferTo(0, channel1.size(), channel2);
    // closing the streams also closes the channels
    in.close();
    out.close();
}
```

FileChannel is fast but uses more  
memory. You can use memory mapping  
(and more code) for better control.

## Example: Writing a File to Another File [2]

```
public void copyFileStream(String fileToRead, String fileToWrite)
    throws IOException {
    // you should add try/finally blocks to make sure streams get closed
    FileInputStream in = new FileInputStream(fileToRead);
    FileOutputStream out = new FileOutputStream(fileToWrite, false);
    byte[] buffer = new byte[1024];
    int size = 0;
    while((size = in.read(buffer)) != -1) {
        out.write(buffer, 0, size);
    }
    in.close();
    out.close();
}
```

Two extra lines of code, but  
easier to stay in a smaller  
memory footprint

## Domino Server Permissions for File (and Network) Access

- Server doc, security fields
  - EITHER: Unrestricted Methods
  - OR: Sign XPages AND Run Restricted
- Dangers of reading/writing files directly to the server
  - No delete access?
  - Forgetting to delete
  - File contention
  - Angry admins

Programmability Restrictions	Who can -
Sign or run unrestricted methods and operations:	ThoseWhoCanRunUnrestricted
Sign agents to run on behalf of someone else:	ThoseWhoCanSignForOthers
Sign agents or XPages to run on behalf of the invoker:	ThoseWhoCanSignForInvoking
Sign or run restricted LotusScript/Java agents:	ThoseWhoCanRunRestricted

IBM

ConnectED2015

## Working Directly with Byte Arrays

- In some cases, we can work directly with byte arrays
  - Potentially avoid file IO and/or restrictions entirely
- Memory considerations
  - It's expensive to hold an entire file in memory
  - Info and a few links about memory at the end of the presentation

IBM

ConnectED2015

## Example: Accessing Embedded Objects [1]

```
EmbeddedObject eo = getFirstDocAttachment(doc);
File tempFile = File.createTempFile("eo-", ".tmp");
eo.extractFile(tempFile.getAbsolutePath());
```

After you're done with the file, you are responsible for deleting it

Easy way to create a temp file in the temp folder

IBM

ConnectED2015

## Example: Accessing Embedded Objects [2]

```
EmbeddedObject eo = getFirstDocAttachment(doc);
InputStream in = new BufferedInputStream(eo.getInputStream());
```

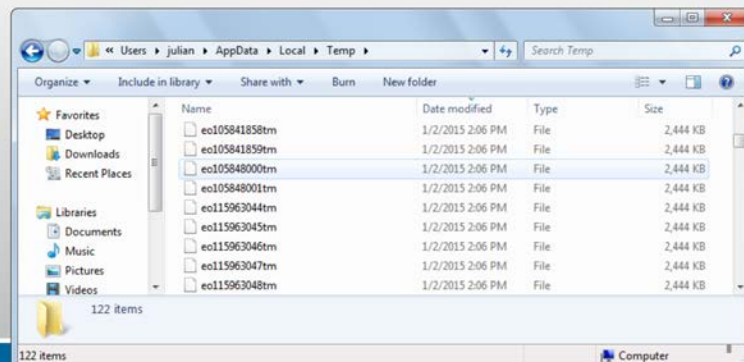
After you're done with the InputStream, you are responsible for closing it

BufferedInputStream supports mark/reset (and it sometimes speeds things up)

This actually creates a temp file in the background

## EmbeddedObject Temp Files

- If your temp directory looks like this, EmbeddedObject.getInputStream() is the culprit
  - Check your code for Exceptions or missing/misplaced InputStream.close()



## Create a Thumbnail and Embed the Image

- Steps:
  - Get the file attachment as an InputStream however you want
  - Resize the image using Java ImageIO
  - Re-attach the image to a rich text field
- We will use a super-special magic technique to attach the image as an embedded image instead of a file attachment
- NOTE: Julian went into more detail about image resizing in a presentation with Mark Myers last year:
  - <http://www.slideshare.net/panagenda/show104-practical-java-30838547>

## Example: Create the Thumbnail Image

InputStream from an EmbeddedObject or file

```
public byte[] createThumbnail(InputStream in, float size)
    throws IOException {
    BufferedImage bi = ImageIO.read(in);
    float scale = Math.min(size/bi.getHeight(), size/bi.getWidth());
    BufferedImage thumb = new BufferedImage(bi.getWidth(),
        bi.getHeight(), bi.getType());
    AffineTransformOp op = new AffineTransformOp
        (AffineTransform.getScaleInstance(scale, scale),
        AffineTransformOp.TYPE_BICUBIC);
    thumb = op.filter(bi, thumb);
    thumb = thumb.getSubimage(0, 0,
        (int)(bi.getWidth()*scale), (int)(bi.getHeight()*scale));
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    ImageIO.write(thumb, "jpg", baos);
    return baos.toByteArray();
}
```

ByteArrayOutputStreams are handy

IBM

ConnectED2015

## Example: Create the Embedded Image in a MIME Field

```
public void embedImageMIME(Document doc, String fieldName,
    byte[] imageData, String mimeType, String fileName)
    throws NotesException {
    MIMEEntity mime = doc.createMIMEEntity(fieldName);
    MIMEHeader hdr = mime.createHeader("MIME-Version");
    hdr.setHeaderValAndParams("1.0");
    Session session = doc.getParentDatabase().getParent();
    Stream stream = session.createStream();
    stream.write(imageData);
    // NOTE: JPG images must be MIME type "image/jpeg", not "image/jpg"
    mime.setContentFromBytes(stream,
        mimeType + "; name=\"" + fileName + "\"",
        MIMEEntity.ENC_IDENTITY_BINARY);
    doc.closeMIMEEntities(true, fieldName);
}
```

you just have to add the data to the MIME field like this

IBM

ConnectED2015

## Example: Create the Embedded Image in a Rich Text Field

```
public void embedImageRichText(RichTextItem rtitem,
    byte[] imageData, String mimeType, String fileName)
    throws NotesException {
    Database db = rtitem.getParent().getParentDatabase();
    Session session = db.getParent();
    Document tempDoc = db.createDocument();
    // for non-image MIME types, this will just be an attachment
    embedImageMIME(tempDoc, "body", imageData, mimeType, fileName);
    boolean isConvert = session.isConvertMime();
    session.setConvertMime(true);
    RichTextItem tempItem = (RichTextItem)tempDoc.getFirstItem("body");
    session.setConvertMime(isConvert);
    rtitem.appendRTItem(tempItem);
    tempDoc.recycle();
}
```

setConvertMime is what changes the MIME data to an embedded image

IBM

ConnectED2015



# THIRD PARTY LIBRARIES

Options for adding and using third party libraries.

IBM

ConnectED2015

## How to Add Third Party Libraries

- You can add them just like we did with our “own” code
  - Drag/drop, copy/paste, import, etc. into Jars, Java or WebContent
  - Sometimes the build path gets lost
    - In Package Explorer, right-click, Build Path > Configure Build Path
- You can create/use an OSGi plugin
  - Here are some tutorials:
    - [http://www-10.lotus.com/ldd/ddwiki.nsf/dx/Creating\\_an\\_XPages\\_Library](http://www-10.lotus.com/ldd/ddwiki.nsf/dx/Creating_an_XPages_Library)
    - [http://www-10.lotus.com/ldd/ddwiki.nsf/dx/Deploying\\_XPage\\_Libraries](http://www-10.lotus.com/ldd/ddwiki.nsf/dx/Deploying_XPage_Libraries)

IBM

ConnectED2015

## How to Add Third Party Libraries

- You can put the JAR files in the /jvm/lib/ext directory
  - Has to be on all servers the app is on
  - Painful to make updates
  - Admins don't like this
- Security
  - Third party code is, well, third party, so use code from trusted sources, like apache.org
  - Might require changes to java.policy or similar

IBM

ConnectED2015



## metadata-extractor Library

- Open-source (Apache 2) library for reading EXIF data from photo files
  - <https://drewnoakes.com/code/exif/index.html>
  - <https://github.com/drewnoakes/metadata-extractor/releases>
  - <http://javadoc.metadata-extractor.googlecode.com/git/2.7.0/index.html>
- Time/date, GPS, camera type, exposure, etc.

## Example: Get GPS Data from a Photo

```
public double[] getLatLong(InputStream in)
    throws IOException, ImageProcessingException {
    Metadata metadata = ImageMetadataReader.readMetadata(in);
    GpsDirectory gpsDir =
        (GpsDirectory)metadata.getDirectory(GpsDirectory.class);
    if (gpsDir != null) {
        GeoLocation gps = gpsDir.getGeoLocation();
        double[] latLong =
            new double[] { gps.getLatitude(), gps.getLongitude() };
        return latLong;
    }
    return null;
}
```

Easy way to populate  
an array

## Generating Barcodes and QR Codes

- ZXing library
  - <https://github.com/zxing/zxing>
  - Java and several other languages
  - Open source, Apache 2.0 licensed
- Create several different types of 1D and 2D barcodes
  - UPC, EAN, Code 128, QR Code, etc.

## Example: Create a QR Code

ISO-8859-1 is preferred for QRCode text

```
public byte[] createQrCode(String text, int size)
    throws IOException, WriterException {
    String charset = "ISO-8859-1";
    Hashtable hints = new Hashtable();
    hints.put(EncodeHintType.CHARACTER_SET, charset);

    MultiFormatWriter qwriter = new MultiFormatWriter();
    BitMatrix qbits = qwriter.encode(text, BarcodeFormat.QR_CODE,
        size, size, hints);
    ByteArrayOutputStream out = new ByteArrayOutputStream();
    MatrixToImageWriter.writeToStream(qbits, "png", out);
    return out.toByteArray();
}
```

You could also write directly to a file, embed this as an image, etc.

# ACCESSING WEB SITES

Making an HTTP connection, getting data, parsing XML

## Connecting to External Websites

- URL classes (HTTP and HTTPS) are built-in to Java
  - Apache HttpComponents are good if you need more fine-grained control
  - <http://hc.apache.org>
- Same security requirements as file access (see earlier slides)
- Additional steps required for proxy servers
  - and possibly HTTPS, if certificates are not allowed

## Maps, Addresses, and Geocoding

- Many online mapping services available
- Be mindful of license restrictions and requirements
  - Sometimes the free services aren't free at all
  - Especially if you're a corporation
- OpenStreetMap is an open-source alternative
  - License: <http://www.openstreetmap.org/copyright>
  - Maps API: <http://wiki.openstreetmap.org/wiki/Develop>
  - Geocoding: <http://wiki.openstreetmap.org/wiki/Nominatim>
  - Usage Policy: [http://wiki.openstreetmap.org/wiki/Nominatim\\_usage\\_policy](http://wiki.openstreetmap.org/wiki/Nominatim_usage_policy)

## Example: Convert GPS Coordinates to Address (reverse geocoding)

```
public String getAddress(double latitude, double longitude) throws IOException {
    URL url = new URL("http://nominatim.openstreetmap.org/reverse?format=xml" +
        "&lat=" + latitude + "&lon=" + longitude + "&zoom=18&addressdetails=1");
    InputStream in = url.openStream();

    try {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        factory.setValidating(false);
        org.w3c.dom.Document domDoc = factory.newDocumentBuilder().parse(in);
        Node node = XPathAPI.selectSingleNode(domDoc,
            "/reversegeocode/result/text()");
        return node.getNodeValue();
    } catch (Exception ignored) { in.close(); }

    return null;
}
```

**Make sure it's "org.w3c.dom.Document"  
and not lotus.domino.Document**

**Weirdly, this closes  
the InputStream  
after parsing**

## XML and XPath

- After your XML data has been parsed, you can:
  - Step through the nodes to look for data (boring)
  - Use XPath to search and extract specific nodes (fun!)
- XPath is a kind of query language for searching through XML
  - Also used with XSL
- Getting started
  - [http://en.wikibooks.org/wiki/XPath/Basic\\_Syntax](http://en.wikibooks.org/wiki/XPath/Basic_Syntax)
  - <http://www.ibm.com/developerworks/xml/tutorials/x-xpath/x-xpath.html>

## OpenStreetMap Output and XPath

```
<?xml version="1.0" encoding="UTF-8"?>
<reversegeocode queryString="format=xml&lat=51.46483333333333&lon=-3.164&zoom=18&addressdetails=1"
  attribution="Data © OpenStreetMap contributors, ODbL 1.0. http://www.openstreetmap.org/copyright" timestamp="Fri, 02 Jan 15
  18:12:42 +0000">
  <result lon="-3.1632088647976" lat="51.46474055" ref="Wales Millennium Centre" osm_id="26584146" osm_type="way"
  place_id="60125628">Wales Millennium Centre, Roald Dahl Plass, Cardiff Bay, Cardiff, Wales, CF10 5AN, United Kingdom</result>
  <addressparts>
    <theatre>Wales Millennium Centre</theatre>
    <pedestrian>Roald Dahl Plass</pedestrian>
    <suburb>Cardiff Bay</suburb>
    <city>Cardiff</city>
    <county>Cardiff</county>
    <state>Wales</state>
    <postcode>CF10 5AN</postcode>
    <country>United Kingdom</country>
    <country_code>gb</country_code>
  </addressparts>
</reversegeocode>
```

Node node = XPathAPI.selectSingleNode(domDoc, "/reversegeocode/result/text()");

IBM

ConnectED2015

## FTP

- For FTP, an easy (and small) library to use is Apache Commons Net
  - <http://commons.apache.org/proper/commons-net/index.html>
- Also supports FTPS, TFTP, and a few other protocols
- SFTP is different
  - try <http://www.jcraft.com/jsch>

IBM

ConnectED2015

## Example: Upload a File via FTP

ftpFilePath like "file.doc" or "/subdir/file.doc"

```
public boolean uploadFtpFile(String server, String ftpFilePath,
  InputStream stream) throws IOException {
  boolean result = false;
  FTPClient ftp = new FTPClient();
  ftp.connect(server);
  if (FTPReply.isPositiveCompletion( ftp.getReplyCode() )) {
    if (ftp.login("anonymous", "anonymous@example.com")) {
      ftp.setFileType(FTP.BINARY_FILE_TYPE);
      result = ftp.storeFile(ftpFilePath, stream);
    }
    ftp.disconnect();
  }
  return result;
}
```

this will fail if ftpFilePath refers to a subdir that doesn't exist, or if the file already exists and you don't have delete rights

IBM

ConnectED2015

## Create a Button on the XPage to upload the attached photo via FTP

```
<xp:button value="FTP File" id="button3" refreshMode="complete">
  <xp:eventHandler event="onClick" submit="true" refreshMode="complete">
    <xp:this.action><![CDATA[#{javascript:try {
importPackage(com.lotusphere.bp107_2015);

var doc:lotus.domino.Document = currentDocument.getDocument(true);
var eo:lotus.domino.EmbeddedObject = TenLinesOrLess.getFirstDocAttachment(doc);

var stream:java.io.InputStream = eo.getInputStream();

var streamIsOpen = true;
TenLinesOrLess.uploadFtpFile("ftp.tardis.who", "/ftp/important.txt", stream);
} catch(e) {
_dump(e);
} finally {
if (streamIsOpen) {
stream.close();
}}}]></xp:this.action></xp:eventHandler></xp:button>
```

## XAGENTS AND OUTPUT

Using XAgents, collecting data, creating JSON, creating a spreadsheet, sending info to the browser

### What is an XAgent?

- What is it?
  - An XPage that is not rendered, but can return JSON or XML or other data (like an agent does) for use in charting, grids, exporting data, and more
- How do you create one? It's as easy as 1-2-3
  - We will show two examples, returning JSON and returning an Excel file
    - Step one - create an XPage
    - Step two - set rendered="false"
    - Step three - add some code to return some data



## JSON in Java

- No native JSON support in Java (until maybe Java 9), must use third-party libraries
- Lots of popular options
  - gson, jackson, etc.
  - see the list at [json.org](http://json.org)
- Libraries that use reflection to create JSON from Objects are nice, but they generally cause AccessControlExceptions with IBM Domino because they use reflection
  - and they're relatively large if you just want to create a JSON string
- json-simple is a nice, small (simple) choice
  - <http://code.google.com/p/json-simple>

IBM

ConnectED2015

### Example: Creating JSON

```
public String createJson() {
    HashMap values = new LinkedHashMap();
    values.put("value1", "one");
    values.put("value2", 2);

    HashMap subValues = new LinkedHashMap();
    subValues.put("subValue1", new Date());
    subValues.put("subValue2", System.currentTimeMillis());
    values.put("value3", subValues);

    return JSONObject.toJSONString(values);
}
```

HashMaps are awesome, and  
LinkedHashMap retains order

Java 1.5 auto-boxing lets us  
use int instead of Integer

You can put  
Maps inside  
of Maps

IBM

ConnectED2015

### The JSON XAgent

```
<xp:view xmlns:xp="http://www.ibm.com/xsp/core" rendered="false">
  <xp:this.afterRenderResponse>
    <![CDATA[#(javascript:
      importPackage(java.lang);
      importPackage(com.lotusphere.bp107_2015);
      var myview:lotus.domino.View = database.getView("travelogvw");
      var json = TenLinesOrLess.createJson(myview);
      var externalContext = facesContext.getExternalContext();
      var writer = facesContext.getResponseWriter();
      var response = externalContext.getResponse();
      response.setContentType("application/json");
      response.setHeader("Cache-Control", "no-cache");
      writer.write(json);
      writer.endDocument()
    )]]>
  </xp:this.afterRenderResponse>
</xp:view>
```

Don't render the XPage

JavaScript for the XAgent

Call the Java code

Set the HTTP header

Send the data

IBM

ConnectED2015

## Spreadsheets and Apache POI

- The go-to Java library for generating Excel spreadsheets is Apache POI
  - <http://poi.apache.org>
- There are already a LOT of examples on how to use POI with XPages
  - Please use your preferred search engine to discover them, we won't play favorites
- But, can we create a spreadsheet with 10 lines or less...?

## Example: Create a Spreadsheet with Apache POI

```
public HSSFWorkbook createSpreadsheet(View view) throws NotesException {
    HSSFWorkbook workbook = new HSSFWorkbook();
    HSSFSheet sheet = workbook.createSheet(view.getName());

    ViewEntryCollection vc = view.getAllEntries();
    for (ViewEntry ve = vc.getFirstEntry(); ve != null; ve = vc.getNextEntry()) {
        HSSFRow row = sheet.createRow( sheet.getPhysicalNumberOfRows() );
        for (int i = 0; i < ve.getColumnValues().size(); i++) {
            row.createCell(i).setCellValue(ve.getColumnValues().get(i).toString());
        }
    }

    return workbook;
}
```

## Export XAgent

- Import different jar
 

```
importPackage(org.apache.poi.hssf.usermodel);
```
- Call the Java code
 

```
var wb:HSSFWorkbook = new HSSFWorkbook();
```
- HTTP header stuff so the browser knows it's a spreadsheet
 

```
pageResponse.setContentType("application/x-ms-excel");
pageResponse.setHeader("Content-Disposition","inline; filename=" +
    fileName);
```
- Send bytes instead of text
 

```
wb.write(pageResponse.getOutputStream());
```

# POTENTIAL PROBLEMS

Security and access and memory

IBM

ConnectED2015

## User Access versus Signer Access on XPages

- Pretty much just like you'd expect
  - If users can't access certain records, they still can't access certain records
  - The Notes Security model is the same
- Use `ExtLibUtil.getCurrentSessionAsSigner()` to get `SessionAsSigner`
- XPages security in general:
  - [http://www-10.lotus.com/ldd/ddwiki.nsf/dx/XPages\\_in\\_the\\_Notes\\_Client-Security](http://www-10.lotus.com/ldd/ddwiki.nsf/dx/XPages_in_the_Notes_Client-Security)

IBM

ConnectED2015

## java.policy File

- The dreaded `java.lang.SecurityException` and `java.security.AccessControlException`
- `AccessController.doPrivileged(new PrivilegedAction() { code here } )`
  - <http://lekkimworld.com/mView.action?entry=1371725987318>
  - Tricky, has to be called from a JAR in /ext or an OSGi bundle (read the whole article)
- Julian's article on java.policy:
  - <http://www.socialbizug.org/blogs/2ec5d0ed-d04e-4b18-9610-9819fcebca79/entry/the-java-policy-file-in-ibm-domino?lang=en-us>

IBM

ConnectED2015

## Memory Errors

- notes.ini variable: HTTPJVMMaxHeapSize
- Make sure you are using a 64-bit version of the Domino server
  - “show server” at the Domino console should say “(64 bit)” for Windows
- Helpful links
  - [http://www-10.lotus.com/ldd/dominowiki.nsf/dx/HTTPJVM\\_Out\\_of\\_memory](http://www-10.lotus.com/ldd/dominowiki.nsf/dx/HTTPJVM_Out_of_memory)
  - [http://www.xpageswiki.com/web/youatnotes/wiki-xpages.nsf/dx/Memory\\_Usage\\_and\\_Performance](http://www.xpageswiki.com/web/youatnotes/wiki-xpages.nsf/dx/Memory_Usage_and_Performance)



ConnectED 2015

## Engage Online

- **SocialBiz User Group** [socialbizug.org](http://socialbizug.org)
  - Join the epicenter of Notes and Collaboration user groups
- **Social Business Insights blog** [ibm.com/blogs/socialbusiness](http://ibm.com/blogs/socialbusiness)
  - Read and engage with our bloggers
- **Follow us on Twitter**
  - [@IBMConnect](https://twitter.com/IBMConnect) and [@IBMSocialBiz](https://twitter.com/IBMSocialBiz)
- **LinkedIn** <http://bit.ly/SBComm>
  - Participate in the IBM Social Business group on LinkedIn
- **Facebook** <https://www.facebook.com/IBMConnected>
  - Like IBM Social Business on Facebook



ConnectED 2015

## Notices and Disclaimers

Copyright © 2015 by International Business Machines Corporation (IBM). No part of this document may be reproduced or transmitted in any form without written permission from IBM.

**U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.**

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS DOCUMENT IS DISTRIBUTED “AS IS” WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY. IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

**Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com, BrassRing®, Connections™, Domino®, Global Business Services®, Global Technology Services®, SmartCloud®, Social Business®, Kenexa®, Notes®, PartnerWorld®, Prove It®, PureSystems®, Sametime®, Verse™, Watson™, WebSphere®, Worklight®, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at: [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).



ConnectED 2015

# THANK YOU

Julian Robichaux

[jrobichaux@panagenda.com](mailto:jrobichaux@panagenda.com)

<http://nsftools.com>

Kathy Brown

[kathy@runningnotes.net](mailto:kathy@runningnotes.net)

<http://runningnotes.net>



This excerpt is from the 2nd Ed. of 'Mastering XPages: A Step-by-Step Guide to XPages Application Development and the XSP Language' by Martin Donnelly, Mark Wallace, Tony McGuckin, published by Pearson/IBM Press, ISBN 978-0-13-337337-0. For more info, please visit the publisher site:  
<http://www.ibmpressbooks.com/store/mastering-xpages-a-step-by-step-guide-to-xpages-application-9780133373370>.

Take advantage of a 40% member discount for SocialBiz members for this book. Simply apply the following code during checkout: SOCIALBIZUG.

## CHAPTER 5

# XPages and JavaServer Faces

As mentioned in the beginning of this book, XPages evolved from a previous runtime technology called XFaces. XFaces was conceived as a way for IBM to provide a universal user-interface programming model that could be adopted across its diverse portfolio of application development platforms. As a runtime technology, it needed to cater to developers of differing skill sets, such as Java/J2EE developers, Domino developers, and so forth. These categorizations are not mutually exclusive, and many organizations contain developers with both sets of skills who might be working on the same projects. In fact, many such developers want to choose which tools to use based on the task they need to accomplish at any given time. (For example, they might need to rapidly create a user interface using a WYSIWYG tool and then switch to using Java to add some complex application logic.)

What IBM set out to achieve with XFaces was to define a programming model that would be suitable for a so-called script-level developer, such as someone who knows how to program using a markup language and JavaScript. This programming model was intended to allow developers to target multiple platforms, such as web and the Eclipse Rich Client Platform (RCP). Also, this programming model was based on the JavaServer Faces (JSF) standard-based web application development framework. Achieving this goal would provide the following benefits to application developers:

- **Learn Once, Write Anywhere:** Developers need only learn one model for development across these platforms. The model must be flexible and powerful to allow programmers to fully exploit and optimize the UI for any particular platform.
- **Write Once, Run Anywhere™:** Developers can create a single set of artifacts that can run across multiple platforms.
- **Provide a script-based programming model:** A model that would be familiar for developers with a Domino Designer (or similar) and dynamic HTML programming background (no Java skills required).

- **Allow artifacts to be shared between Java and Script developers who work on the same project:** For example, script developers create the frontend user interface and Java developers create the backend application logic.
- **Flexibility:** Allows developers to use the most appropriate tool for the task they perform.

As XFaces morphed into XPages, these design points were all retained. This chapter examines the relationship between XPages and JSF. Although one of the goals in XPages is to hide all the Java and J2EE-centric aspects of the JSF programming model, having an understanding of the underlying technology is a major asset for any XPages developer. By understanding how JSF works, and especially the workings of the JSF lifecycle, you learn how your XPages are processed as your application executes. This helps understanding why your application behaves in a particular fashion. Also, both XPages and JSF are designed to be extended. For the Domino Developer, you are no longer restricted to what is provided within the platform as delivered by IBM; it's now possible to extend the platform either to solve a particular problem or as a way to start a new business. Please refer to Chapter 12, "XPages Extensibility," for more information on the options available to extend the XPages runtime.

This chapter is aimed at developers who are interested in extending the XPages runtime using Java by creating new XSP components or developers who are coming from a J2EE background and want to understand how XPages extends JavaServer Faces. This chapter uses the standard JSF terminology when explaining how JSF works and the relationship between JSF and XPages. In JSF parlance, a component is a UI element or what has been previously referred to as a UI control (an edit box or button). JSF uses the terms view and component tree interchangeably. XPages also uses view (remember the root tag of every XPage is the `xp:view` tag) and an XPages' view is, in fact, a component tree. Knowing this means the working definition of XPages can be extended to this: XPages is an XML-based language that can be used to define JSF views, and an XPage is a static representation of a JSF component tree.

Be sure to download the **Chp05Ed2.nsf** files provided online for this book to run through the exercises throughout this chapter. You can access these files at [www.ibmpressbooks.com/title/9780133373370](http://www.ibmpressbooks.com/title/9780133373370).

## What Is JavaServer Faces?

JSF is a component-based, user interface framework for building Java-based web applications. The framework provides the following capabilities:

- A set of reusable user-interface components that can be used to easily create an application frontend or can be used as the starting point to create new custom user interface components
- A Model-View-Controller (MVC) programming model that supports event-driven programming

- A state-full server representation of the user interface that can be synchronized with the client representation
- A mechanism to allow data flow to and from the user interface, including the capability to perform data conversion and data validation
- A framework that can be extended using Java programming techniques

Using the JSF framework as the starting point when creating a web application frees the application developer from having to deal with the stateless nature of HTTP—without the use of a framework, no application state is maintained on the server between requests. The developer can create the required user interface using the standard UI components (a.k.a controls) provided by JSF. Then, the developer can bind these controls to the application data (in the form of Java beans) and then trigger server-side application logic in response to user actions on the application user interface. A Java bean is a reusable Java-based software component (see <http://docs.oracle.com/javase/tutorial/javabeans/> for more details). This type of programming model is familiar to developers of rich client-based applications using technologies such as the Standard Widget Toolkit (SWT); however, at the time JSF was introduced, it was pretty much a new concept for web developers.

The following JSF Primer sidebar provides a basic introduction to JSF and is written with the assumption that you have no knowledge of Java2 Enterprise Edition (J2EE). The relevant J2EE concepts are briefly explained in this sidebar. The JSF lifecycle is also explained in the sidebar; this is a key concept that all XPages developers should understand. For a detailed look at the JSF technology, the authors recommend the following resources:

- *JavaServer Faces Specification*, version 1.1 ([http://docs.oracle.com/cd/E17802\\_01/j2ee/j2ee/javaxserverfaces/1.1/docs/api/](http://docs.oracle.com/cd/E17802_01/j2ee/j2ee/javaxserverfaces/1.1/docs/api/))
- *JavaServer Faces* (O'Reilly)
- *Mastering JavaServer Faces* (Wiley)

## JSF Primer

To run a JSF-based application, you need a Java web container, such as an Apache Tomcat server, and an implementation of the JSF specification. A Java web container is a Java-based server for running Java web applications. JSF 1.1 (which is the version used in the XPages runtime) requires a web container that implements, at a minimum, the Servlet 2.3 and JavaServer Pages 1.2 specifications. (XPages requires support for the Servlet 2.4 specification.) IBM WebSphere Application Server (WAS) and Portal Server support the Servlet and JSP specifications.

A servlet is a Java class that runs in the web container, processes client requests, and generates responses. A servlet is passed parameters that represent the request and response and, in simple cases, all the processing logic can be included within the servlet. Typically, a servlet is defined as the entry point or front controller for a web application. A servlet typically delegates

to request handlers to process the client requests and a presentation tier to generate the responses. Listing 5.1 shows the source code for a simple HTTP servlet. This servlet handles an HTTP GET request and responds with a HTML page that displays the text Hello World. The code to handle the request has access to a request object, which can be used to retrieve information about the request being processed and a response object, which can be used to write the response that is returned to the client.

**Listing 5.1** Sample HTTP Servlet

```
package mxp.chap05;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Sample Servlet
 */
public class SampleServlet extends HttpServlet {

    /**
     * Handle a HTTP GET request.
     */
    protected void doGet(HttpServletRequest request,
                        HttpServletResponse response)
        throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.println("<HTML>");
        out.println("<HEAD>");
        out.println("<TITLE>Hello World</TITLE>");
        out.println("</HEAD>");
        out.println("<BODY>");
        out.println("Hello World");
        out.println("</BODY>");
        out.println("</HTML>");
    }
}
```

JavaServer Pages (JSP) is a presentation logic layer that can generate HTML pages in response to client requests. A JSP page looks like a HTML page, but it contains a mix of static HTML and JSP directives, which can be used to generate dynamic content or performing some processing associated with generating the client response. JSP uses tag libraries to allow special tags to be declared, which can then be invoked by the JSP engine. A JSP implementation comes with a standard tag library called the JavaServer Pages Standard Tag Library (JSTL).

Listing 5.2 shows a sample JSP page that uses the JSF tag library to embed JSF components within an HTML page. Based on what you have learned so far about XSP markup, this sample should be readable. It contains a mix of HTML and JSF tags. The JSF tags cause JSF components to be created and results in a HTML form being created, which contains an edit box that can be used to enter a value and a button that can be used to submit the form.

#### Listing 5.2 Sample JSP with JSF Tags

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<BODY>
  <f:view>
    <h:form id="form1">
      Enter some value:
      <h:inputText
        id="inputText1" value="#{ModelBean.someValue}"/>
      <h:commandButton
        id="commandButton1" action="success" value="Submit"/>
    </h:form>
  </f:view>
</BODY>
```

JSP is the default presentation tier used by the JSF reference implementation. The presentation tier is the layer in an application framework that is responsible for displaying the application data in a human-readable format. The presentation tier defines the JSF component tree (also known as the JSF view), which is the hierarchy of controls that is presented in the user interface. A typical starting point for a JSF-based application is where a user requests a JSP, such as typing a URL like this into a browser:

```
http://somehost/jsfapp/somepage.jsp
```

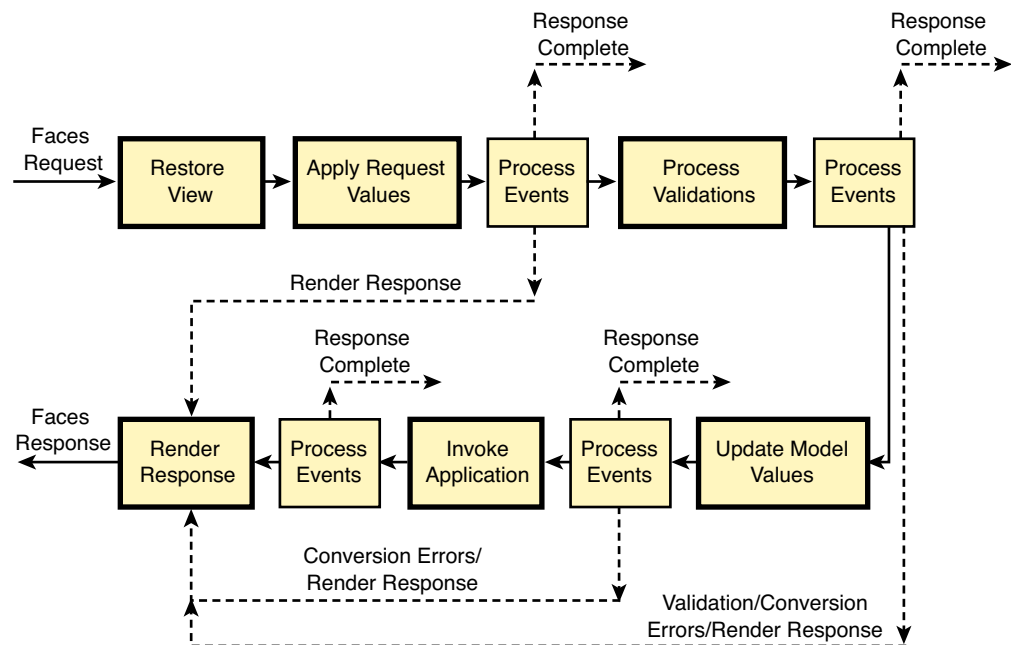
This causes the JSP engine to load and execute the specified JSP. If this page contains JSF components (using the standard JSF tag library), a JSF component tree is also created in addition to the regular JSP processing (the JSF tags are responsible for creating the JSF component tree). The JSF components generate the HTML markup that is presented to the user and the view is cached for the user (see section 2.1.1 of the JSF 1.1. specification, “Non-Faces Request Generates Faces Response”).



Now, if the same page is submitted back to the server, it is handled by the JSF servlet. This servlet is part of the JSF implementation and acts as a front controller for all JSF-based applications. JSF requests are processed in accordance with the rules defined by the JSF request processing lifecycle. The JSF request processing lifecycle consists of a number of well-defined phases that describe how each request is handled and, of course, these phases also apply to XPages. The phases on the standard request processing lifecycle are as follows:

1. Restore View
2. Apply Request Values
3. Process Validations
4. Update Model Values
5. Invoke Application
6. Render Response

Figure 5.1 illustrates how the processing lifecycle operates.



**Figure 5.1** JSF request processing lifecycle

The *Restore View* phase retrieves the JSF view for the request. If no JSF view exists, a new one is created and cached for later use. Maintaining a consistent representation of the JSF view between requests simplifies the programming task for the application developer by simplifying the application logic to focus on the business problem and not having to maintain information about the state of the view.

The *Apply Request Values* phase is used to allow the JSF components to update their state based on the values from the current request; for example, if the component represents an editable value, the component stores the current value. Action and editable components have a special behavior during this phase. If the component `immediate` property is set to `true`, the JSF lifecycle is short circuited. For an action component, the action processing happens at the end of this phase instead of during the lifecycle. For an editable component, the validation processing happens immediately.

The *Process Validations* phase allows any validators associated with components in the view and any built-in server side validation associated with a specific component to be executed. All components that can be used to edit a value and support validation, have an associated property (aptly named `valid`) to indicate whether the current value is valid. When validation errors occur, messages are queued and the `valid` property of the associated component is set to `false`. Validation error messages can be displayed to the end user using the `xp:message` or `xp:messages` tags as described in Chapter 4, “Anatomy of an XPage.” Validation errors typically cause the lifecycle processing to terminate and result in a response being immediately sent back to the end user.

If the *Update Model Values* phase is reached, it is assumed that the values provided in the request are valid (as defined by any validators specified in the view). The current values are stored in the `localValue` property of the associated component. During this phase, the application data is updated with the new values. In the case of an XPages application, the values are written to the Domino document during this phase.

If the *Invoke Application* phase is reached, it is assumed that the application data has been updated. The relevant application logic specified in the view is executed during this phase. In an XPages application, if application logic is associated with a button and that button caused the page to be submitted, it is now that the logic is executed.

The *Render Response* phase generates the response and saves the state of the view. In the XPages case, the response is an HTML page and the rendering is performed using a platform-specific renderkits, and the application developer has control over the state saving (for example, to optimize server performance, he can decide not to save any state). The JSF response rendering model is flexible and is discussed further.

From Figure 5.1, you see that, after certain phases, there is an event-processing operation that can result in the lifecycle being short circuited and the response being rendered. This typically happens if there is a conversion or validation error, which means that data specified by the end user is not valid, so it doesn't make sense to update the application data or to execute any application logic.

Numerous other key concepts in JSF are important to understand before looking at how XPages builds on top of this foundation:

1. Integration with JSP
2. User Interface Component Model
3. Value Binding and Method Binding Expression Evaluation
4. Per-Request State Model
5. Application Integration
6. Rendering Model
7. JSF APIs

JSF implementations must support JSP as the page-description language (the mechanism for defining the JSF component tree). This allows J2EE developers to start creating JSF-based applications using a well-known technology. When JSF tags are added to a JSP page, they cause the JSF component tree to be created when the page is executed.

A JSF user interface is created as a tree of components (known as *controls* in XPages). Components typically are rendered to the user as HTML markup, which produces the application interface. However, not all components render as visual elements in the UI; they can render no markup or just client-side JavaScript and thereby add behavior to the UI. Components can be manipulated on the server, and this can result in changes to the application UI (the Button sample in Chapter 4 shows an example of this). Components can have different types, such as have the ability to trigger application logic, can be a container for other components, can have an associated value, or can edit its associated value. A well-defined data conversion model associated with components allows application data to be converted between the underlying data types and string values and back again. This is essential as data is going to be represented as string values in the HTML markup. There is also a well-defined validation model that allows multiple checks to be performed on user input and prevents application logic executing on invalid data. JSF implementations provide a standard set of user interface components, and these form the basis for the controls you can add to an XPage. Finally, a standard set of data model classes can be used with the standard controls; refer to the JSF Java documentation for the `javax.faces.model` package for more information.

Binding expressions are how application logic and data binding is performed in JSF. Value bindings are used when you need to compute a component property or when you want to bind application data to a component for display and/or editing. JSF uses Expression Language (EL) to specify value binding expressions. EL is fully defined in the JavaServer Pages specification (version 2.0) and the JSF usage only differs in the delimiters used, such as `#{` and `}` instead of `${` and `}` and the fact that EL functions are not supported. EL can be used in XPages applications, as demonstrated in examples in Chapter 4.

Method-binding expressions are a variation on value bindings where parameters can be passed to the method being invoked and the result can be returned. Method bindings invoke

application logic, which in JSF applications is code in Java. XPages additionally supports application logic written in JavaScript. JSF supports an extensible mechanism for resolving binding expression variables and properties. By default, JSF supports Java beans-based property resolution and a well-defined set of variables. This Java-centric approach has been extended in XPages to better support the use of JavaScript and Domino.

During the JSF request processing lifecycle, the request state is represented by a set of JSF objects, such as `FacesContext`, `FacesMessage`, `ResponseStream`, `ResponseWriter`, and `FacesContextFactory`. JSF provides a mechanism to allow built-in request related objects to be available during request processing.

The JSF programming model is Java-based, and there is a well-defined model for the execution of the JSF-based application. XPages provides a dynamic HTML-like programming model (combination of JavaScript and Markup Language) on top of JSF. This can be achieved because JSF provides an extensible mechanism to modify the execution of a JSF application. The application integration APIs in JSF provide access to modify the behavior of how JSF-based applications are executed.

During the execution of a JSF request, the incoming request values need to be decoded at the start of the lifecycle during the *Apply Request Values* phase and subsequently encoded when the response is generated. JSF allows each component to handle the decoding and encoding processes directly. One disadvantage with this approach is that it can tie a component to a particular platform or rendering technology, such as a component that decodes HTTP requests and encodes HTML responses that can't be used with a VoiceML client. To address this problem, JSF supports a model where each component can delegate the encoding and decoding processes to an associated *renderer*. Now, different renderer implementations can be provided for different client types, and JSF provides a simple mechanism to group these renders into a *renderkit* along with the ability to switch between renderkits. This keeps the components platform independent. JSF provides a default HTML renderkit.

The JSF reference implementation comes in two parts:

1. JSF API
2. JSF Implementation

The JSF API is a Java API that consists of interfaces and abstract classes that define the abstractions that make up the JSF engine. JSF allows key parts of the implementation to be extended while still preserving the default behavior. This is achieved by means of a delegation model, where the new extension has the option to execute first and then delegate to the default implementation when appropriate. The JSF API provides abstract Java classes for the modules, which can be extended. JSF also has an XML configuration file format and a mechanism for loading multiple instances of this file. To override a module in the JSF engine, you need to provide your custom implementation and a Faces configuration XML file that specifies that your implementation should be loaded and used instead of the default one. Consider the following quote from the JavaServer Faces specification:

“JSF’s core architecture is designed to be independent of specific protocols and markup. However it is also aimed directly at solving many of the common problems encountered when writing applications for HTML clients that communicate via HTTP to a Java application server that supports servlets and JavaServer Pages (JSP) based applications.”

Although JSF is Java-centric and J2EE-based, the API provides sufficient flexibility to allow the JSF framework to be used in other contexts. So, it is possible to create a non Java-centric programming model on top of JSF and still maintain the benefits of providing a standards-based solution, and this is what has been achieved in XPages.

## How Does XPages Extend JSF?

As previously mentioned, JSF provides a delegation model whereby key modules in the JSF engine can be replaced. To do this, you need to create your own Java class that extends the base class, which defines the module you want to extend. This class must have a constructor that takes a single argument, which is an instance of the class defining the module you are extending. A concrete example of this would be the custom variable resolver that is provided in XPages. The default variable resolver in JSF provides access to a number of built-in variables (see Table 5.1).

**Table 5.1** JSF Default Variables

Name	Value
applicationScope	Map containing the application scope values
cookie	Map containing the cookies for the current request
facesContext	The FacesContext instance for the current request
header	Map containing the HTTP header values for the current request
headerValues	Map containing arrays that contain the header values for the HTTP headers for the current request
initParam	Map containing the initialization parameters for the web application
param	Map containing the request parameters for the current request
paramValues	Map containing arrays that contain the parameter values for request parameters for the current request
requestScope	Map containing the request attributes for the current request
sessionScope	Map containing the session attributes for the current request
view	UIViewRoot of the current component tree

XPages extends these variables to include some additional ones, which are relevant for a Domino application developer, such as the current database. JSF provides a pluggable mechanism to allow what is called a *variable resolver* to be configured for a JSF application. This



variable resolver must provide the default behavior as defined in the JSF specification but can provide additional functionality. To do this, the following two steps are required:

1. An implementation of `javax.faces.el.VariableResolver` must be provided. It either must implement the default behavior or else delegate to the default implementation.
2. The `faces-config.xml` file for the JSF application must be edited to specify the new variable resolver implementation.

The `faces-config.xml` file is the main configuration file for a JSF-based application. It is used to configure the behavior of the application and the JSF runtime. You need to switch to the Java perspective in Domino Designer to perform both of these steps. The `faces-config.xml` file is located in the `\WebContent\WEB-INF` folder. Starting in Domino Designer 9.0, you can show the `faces-config.xml` file in the Applications navigator by editing the Domino Designer navigator preferences.

Listing 5.3 shows the Java code for a variable resolver, which adds support for an additional variable called “magic,” which resolves to a string value “Abracadabra.” This class provides a constructor that takes a single variable, which is an instance of `VariableResolver`; this delegate provides the default behavior. The custom implementation can delegate to this and still provide the default behavior and be compliant with the JSF specification.

---

**Listing 5.3** Sample Variable Resolver

```
package mxp.chap05;

import javax.faces.context.FacesContext;
import javax.faces.el.EvaluationException;
import javax.faces.el.VariableResolver;

/**
 * Sample variable resolver
 */
public class SampleVariableResolver extends VariableResolver {

    private VariableResolver delegate;

    /**
     * Constructor which takes delegate VariableResolver
     */
    public SampleVariableResolver(VariableResolver resolver) {
        delegate = resolver;
    }
}
```

---

```

/**
 * Return the object associated with the specified variable name.
 */
public Object resolveVariable(FacesContext context, String name)
    throws EvaluationException {
    if ("magic".equals(name)) {
        return "Abracadabra";
    }
    return delegate.resolveVariable(context, name);
}

```

---

To get this instance to load, an entry must be added to the `faces-config.xml` specifying that this class as the variable resolver. Listing 5.4 shows what this entry looks like in the `faces-config.xml`.

#### Listing 5.4 Variable Resolver Configuration

---

```

<?xml version="1.0" encoding="UTF-8"?>
<faces-config>
  <application><variable-resolver>
    mxp.chap05.SampleVariableResolver
  </variable-resolver>
</application>
<!--AUTOGEN-START-BUILDER: Automatically generated by IBM Lotus Domino
Designer. Do not modify.-->
<!--AUTOGEN-END-BUILDER: End of automatically generated section-->
</faces-config>

```

---

After these two changes are made, you can now reference the “magic” variable from the `SampleVariableResolver` XPage. Listing 5.5 shows the XSP markup that contains Computed Fields that reference the new “magic” variable.

#### Listing 5.5 Variable Resolver Sample XPage

---

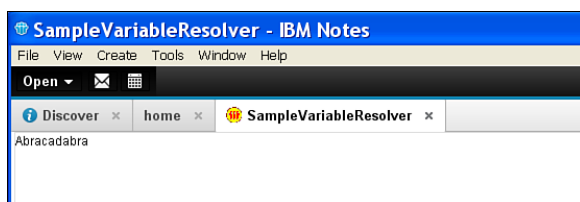
```

<?xml version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core">
  <xp:text escape="true" id="computedField1" value="{magic}">
    </xp:text>
</xp:view>

```

---

When you preview this page, you see the results illustrated in Figure 5.2.



**Figure 5.2** Variable resolver sample preview

## XML-Based Presentation Tier

As mentioned earlier, the default presentation tier in JSF version 1.1 is JSP. There are well-known issues with using JSP and JSF, but the biggest hurdle from the Domino developer perspective is that JSP is a Java-based technology, and not all Domino developers are familiar with Java. Domino developers are, however, familiar with creating HTML markup and, therefore, it was decided to create a new markup-based presentation for JSF. Additionally, JSF developers use the `faces-config.xml` file to configure certain aspects of their application, such as navigation rules and managed beans. In designing the new presentation tier, it was decided to allow the developer to perform most of the application configuration within the XPage itself including page navigation, the configuration of data sources, and the inclusion of application logic. This new presentation tier became the XSP language.

JSF provides the capability for a custom implementation to be provided for the Render Response and Restore View phases of the JSF lifecycle. An abstract Java class called `ViewHandler` can be extended and then this new implementation configured to be the view handler for the JSF application (as demonstrated previously with the custom navigation handler). This mechanism is used in XPages to provide the XSP markup-based presentation tier. So, the first and most important enhancement that XPages provides on top of JSF is the capability to create the JSF view using a markup language. Additionally, XPages provides some custom options for the Restore View phase. The default behavior for saving the state of the JSF view is to walk the component tree and request that each component save its state. The state data is then either stored on the server or serialized into the HTML response and stored on the client. Saving view state on the server has performance implications for the server. Saving state in the response increases the size of the response and, therefore, increases the network traffic. In some cases, there is no need to store the full state of the view (for example, when the page is being used for display only).

## Request Processing Lifecycle

XPages allows you to execute the JSF request processing lifecycle on a portion of the component tree. To do this, use the `execMode` and `execId` properties of the event handler. The `execMode` property allows you to specify that either the complete or partial execution of the lifecycle. When partial execution is specified by setting `execMode="partial"`, only a portion of the component

tree is used when executing the lifecycle. Components that are not part of this subtree are not processed during the lifecycle. The `execId` property specifies the component ID of a control within the pages component tree, which is the root of the subtree to be used when executing in the lifecycle. This allows you to optimize the execution of the lifecycle as a much smaller number of components need to be processed. This is something you will want to do to decrease the load on your server and to improve the performance of your XPages.

XPages also provides an optimization for the Render Response phase of the lifecycle, which either limits or eliminates the response. The event handler has two properties—`refreshMode` and `refreshId`—which specify and control partial refresh (partial or no rendering of the response). When partial refresh is specified by setting `refreshMode="partial"`, only a portion of the component tree contributes to the generated response. The response can also be completely eliminated by setting `refreshMode="norefresh"`. The `refreshId` is used in conjunction with a partial refresh to specify the portion of the component tree, which is used to generate the response, the specified control ID, which should be the root of the subtree that is used. Partial or no refresh is another optimization technique. The responsiveness of your XPages and the end user's experience can be significantly improved by using partial refresh to update just a part of the page and to reduce the number of page reloads.

## User Interface Component Model

JSF uses the term *component* to refer to user interface components or what are known as *controls* in XPages. These components are the user interface elements used to create the application user interface. JSF provides the following:

- A fundamental API for user interface components
- Component behavioral interfaces that allow components to provide specific functionality, such as access to a data model
- A facility to convert data values (for example, to string representation for use in the presentation tier)
- A facility for validating user input

XPages builds on top of the JSF user interface component model to provide the following:

- XPages behavioral interfaces that allow components to contribute to the XPages-specific pages
- XPages converters, which extend the default conversion facility provided by JSF
- XPages validators, which extend the default user validation provided by JSF

## XPages Behavioral Interfaces

The behavioral interfaces are implemented by user-interface components that support XPages-specific behavior. For example, in regular JSF, you must add a tag corresponding to a form

component in the view definition to have a HTML form rendered in the response. For convenience, the standard XPages view root component automatically adds a form to each XPages view. But, what happens now if you want to manually add the form yourself? When you do this, the standard XPages form component automatically disables the automatic form creation by finding the parent, which creates the form and tells it not to automatically create a form. This list describes the XPages behavioral interfaces:

- **FacesAjaxComponent:** Implemented by user-interface components that can handle an AJAX request and return a valid response. The type-ahead component implements this interface and returns the list of suggestions in XML format as the response to an AJAX request.
- **FacesAttrsObject:** Implemented by user-interface components that allow arbitrary extra attributes to be output on their base tag. This is also used to allow attributes to be specified, which are passed through and emitted on the page. All the XPages controls implement this interface. This allows controls to be forward-compatible as attributes, such as Dojo attributes, can be added later without requiring updates to the controls or their renderers.
- **FacesAutoForm:** Implemented by user-interface components that automatically create a form component and is used to ensure that when a form is manually inserted into the view that an automatic form is not created. The XPages view root component implements this interface and normally automatically creates a form for each XPage.
- **FacesComponent:** Implemented by user-interface components that need to perform some initialization before and/or after their children are created or want to build their own children. The repeat component implements this because it builds its own children. The repeat container component (which is the parent for each row of children in a repeat) also implements this interface to ensure the correct row data is available to its children as they are being created.
- **FacesDataIterator:** Implemented by user-interface components that iterate over a value and is used to get information about the data model being used and the rows of data that is displayed. The repeat component implements this.
- **FacesDataProvider:** Implemented by user-interface components that can be configured with a data source. The view root and panel control, among others, implement this and can be configured with a Domino document or view data source.
- **FacesInputComponent:** Implemented by input components and is used to disable validation and to disable the behavior in the Notes client where the user gets prompted if a value is modified and might need to be saved before closing an XPage. The XPages standard input component (described in the next section) implements this interface.



- **FacesInputFiltering:** Implemented by input components that support input filtering and find the correct input filter to be applied. The XPages standard input component implements this interface and supports the filtering of active content.
- **FacesNestedDataTable:** Implemented by user-interface components that render using multiple tables and is used to support AJAX requests that replaces the component rendering. The XPages standard view panel component (described in the next section) implements this interface.
- **FacesOutputFiltering:** Implemented by output components that support output filtering and is used to find the correct output filter to be applied. The XPages standard output component (described in the next section) implements this interface and supports the filtering of active content.
- **FacesPageIncluder:** Implemented by user-interface components that include another XPage and need to perform some initialization before and/or after their children are created or want to build their own children. The **include** component implements this interface because it is used to include another XPage. The standard **include** composite component (described in the next section) also implements this interface because including a Custom Control is a special case of including another XPage.
- **FacesPageProvider:** Implemented by user-interface components that act as the root of a page during the create view phase of the JSF lifecycle. This is only intended for internal use by the XPages page-loading mechanism and must never be implemented by a third party.
- **FacesParentReliantComponent:** Implemented by user-interface components that have a strict child/parent relationship and does not behave correctly if an additional container is inserted between them, and their parent and is used with Custom Controls to force the **include** composite component to remove itself when the children of the Custom Control all rely on the parent. The XPages select item component implements this because it depends on its parent to render the selection it represents.
- **FacesPropertyProvider:** Implemented by the **include** composite component and used in the publishing of composite data. This must not be implemented by third parties.
- **FacesRefreshableComponent:** Implemented by user-interface components that can be refreshed by one of its children in response to an AJAX request. If the component changes its client ID while rendering its children (this is allowed for a `NamingContainer`), the child uses the wrong client ID and the refresh fails. This interface allows the child to get the correct client ID for use in a partial refresh. The XPages standard data component implements this interface.
- **FacesRowAttrsComponent:** Implemented by user-interface components that output an HTML TABLE and TR elements. The attributes (provided via `FacesAttrsObject`) are output on the TABLE element and the row attributes are output on each TR element.

- **FacesRowIndex:** Implemented by user-interface components that support a row index and is used by data sources to compute the components bean ID. The XPages standard data component implements this interface.
- **FacesSaveBehavior:** Implemented by action components which support the save property and is used to check if the data sources on the page should be saved after the corresponding action is performed. The XPages standard command component (described in the next section) implements this interface.
- **FacesThemeHandler:** Implemented by user-interface components that handle setting their own default styles. The XPages standard file download component implements this interface.
- **FacesDojoComponent:** Implemented by user-interface components that support Dojo attributes. The XPages type-ahead component implements this interface.
- **FacesDojoComponentDelegate:** Implemented by user-interface components that support Dojo attributes on behalf of another component. The XPages date time helper component implements this interface.
- **ThemeControl:** Implemented by user-interface components that support style kits. The majority of the XPages components support this.

You could use the behavioral interfaces if you decide to extend XPages (for example, by building your own Java components for XPages). This subject is covered in Chapter 12, “XPages Extensibility.”

## XPages Converters

JSF defines a mechanism to perform conversion to and from the string representation of the data model value. Model values need to be converted to a string representation to be displayed for the user and, when the user edits a value, it is received as a string value and needs to be converted to the correct type for the underlying data model. The `javax.faces.convert.Converter` interface defines the converter behavior. JSF provides a standard set of converters for common data types: various number formats and date/time values. XPages extends two of the standard converters and provides one new converter implementation:

- **DateTimeConverter:** The XPages data/time converter extends the standard JSF date/time converter, but it uses the International Components for Unicode (ICU) libraries for the conversions. For more information on ICU, visit <http://site.icu-project.org>.
- **MaskConverter:** The XPages mask converter applies the specified mask to the string representation of the value being converted. Table 5.2 shows a table listing the supported mask characters.
- **NumberConverter:** The XPages number converter handles the fractional part of integers and can handle the result of XPath.

**Table 5.2** Mask Characters

Mask Character	Description
#	Any valid decimal digit number (uses <code>Character.isDigit</code> )
'	Used to escape any of the special formatting characters
U	All lowercase letters are mapped to uppercase (uses <code>Character.isLetter</code> )
L	All lowercase letters are mapped to lowercase (uses <code>Character.isLetter</code> )
A	Any valid decimal digit or letter (uses <code>Character.isDigit</code> and <code>Character.isLetter</code> )
?	Any letter
*	Anything
H	Any valid hex character (0–9, a–f, or A–F)

### XPages Validators

JSF defines a mechanism to provide the validation (checks) of user inputted values. Although only a single converter may be associated with an input control, multiple validators can be assigned to a control. The reason for this is that the data might need to pass several validation checks before being persisted; for example, the value is required (not empty), the value is a number, or the value is a credit card number. The `javax.faces.validator.Validator` interface defines the validator behavior. Again, JSF provides some standard validators for checking that numbers or strings lie within a specific range. XPages provides some additional validators and some additional interfaces to customize the validator behavior. The following list describes the XPages validators in detail:

- **ClientSideValidator:** Implemented by validators that support client-side validation. Validators that support client-side validation are asked to provide a single line of JavaScript to be included in the rendered response. For XPages validators, this JavaScript references the `xspClientDojo.js` library and emits a call to the appropriate validator method. Listing 5.6 shows the JavaScript that gets included in a page that contains an edit box with a length validator and a submit button. Note the call to attach the length validator to the input control in the HTML page; this associates the length validator with the edit box whose contents it needs to validate.

#### Listing 5.6 Length Validator Client-Side JavaScript

```
<script type="text/javascript">
XSP.addOnLoad(function() {
XSP.attachValidator("view:_id1:inputText1",null,null,new
```

```
XSP.LengthValidator(0,5,"Incorrect length");
XSP.attachEvent("view:_id1:_id4", "view:_id1:button1", "onclick", null,
true, false);
});
</script>
```

- **FacesRequiredValidator:** Implemented by the required validator and used by the XPages standard input component to identify if a required validator has been added to its list of validators.
- **ConstraintValidator:** Validates using the specified regular expression or, if the regular expression is set to one of the predefined keywords, performs the associated standard validation. Table 5.3 shows the predefined keywords the constraint validator supports.

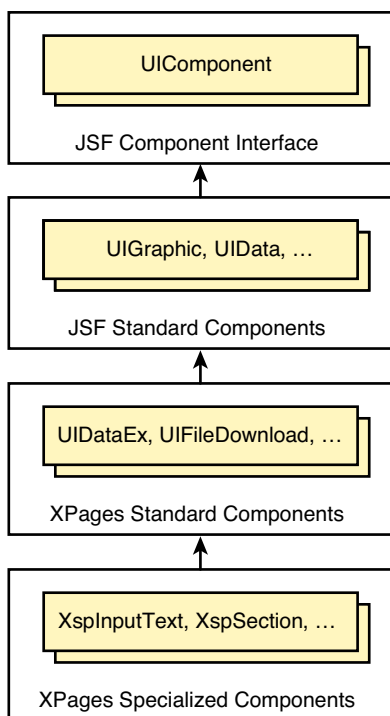
**Table 5.3** Predefined Constraint Checks

Regex	Description
AlphabetOnly	Checks if the value contain only letter characters
DigitOnly	Checks if the value contain only number characters
AlnumOnly	Checks if the value contain only letter and number characters

- **DateTimeRangeValidator:** Validates that a date value lies within the specified time period. Client-side validation and computed properties are supported.
- **DoubleRangeValidatorEx2:** Extends the standard JSF validator to support client-side validation and computed properties.
- **ExpressionValidator:** Enables you to provide custom logic for the client-side and server-side validation.
- **LengthValidatorEx and LongRangeValidatorEx2:** The XPages version of these validators extends the standard JSF validator to support client-side validation and computed properties.
- **ModulusSelfCheckValidator:** Performs a modulus self check (for modulus 10 and 11 only). Client-side validation is not supported. A modulus self check is a standard mechanism for validating identification numbers; for example, modulus 10 (or Luhn algorithm) is a single checksum formula used to validate credit-card numbers.
- **RequiredValidator:** Checks that a value has been specified.

## Standard User-Interface Components

JSF provides a standard set of user interface components which cover the standard control types. Each of these components has a well-defined behavior which is platform independent. The intention is that the JSF standard components would be extended to provide specific implementations for different client platforms. In fact, JSF extends these standard components to provide HTML-specific components. XPages extends the standard components to add XPages-specific behavior and also defines its own completely new standard components. XPages then extends these components to provide the specialized XPages user interface components that are used in Domino Designer and supports the browser and IBM Notes clients. Figure 5.3 shows the hierarchy of user interface components. If you are going to create your own user interface components, you will normally be extending one of the JSF or XPages standard components.



**Figure 5.3** XPages user interface component hierarchy

The following list briefly describes each of the standard user interface components:

- **UICallback:** Represents an area in Custom Control where the user of the Custom Control can add additional content. This component builds its own contents and, after its



children are added, it checks if they are all instances of `NamingContainer` and, if they are, it removes itself from the component hierarchy.

- **UIColumnEx:** Represents a single column of data and expects to have a parent `UIData`. `UIColumnEx` implements `FacesParentRelientComponent` to signal this dependency on its parent.
- **UICommandButton:** Represents a button that, when clicked by the user, can trigger some application logic.
- **UICommandEx2:** Represents a control that, when activated by the user, can trigger some application logic. `UICommandEx2` implements `FacesSaveBehavior`, which means that, when triggered, it can cause the saving of all data sources on the XPage.
- **UIComponentTag:** An abstract component that is extended by specialized XPages components which represent a tag, such as a `div`, `span`, `table`, and so on.
- **UIDataColumn:** Extends `UIColumnEx`, but currently does not add any new behavior.
- **UIDataEx:** Represents a multirow data model. The only allowed children are instances of `UIColumnEx`, which collectively define the presentation a row of data from the model.
- **UIDataIterator:** Like `UIDataEx`, this component represents a multirow data model, but does not have any restriction on what type of children it will have. The children process multiple rows of data, but in a free format rather than the tabular format that `UIData` uses.
- **UIDataPanelBase:** Represents a component that organizes the layout of its children and provides data (it implements `FacesDataProvider`) that is scoped to its children.
- **UIDateTimeHelper:** Used to transform an edit box into a date time picker.
- **UIEventHandler:** Used to handle events on behalf of its parent. It can be configured to handle both client-side events or server-side actions for the component that is its direct parent.
- **UIFileDownload:** Represents a control that can be used to download one or more files.
- **UIFileuploadEx:** Represents a control and can be used to upload a file from a user to the server.
- **UIFormEx:** Represents a HTML form and ensures an XPage doesn't contain nested forms because of the automatic creation of a form elsewhere in the component hierarchy.
- **UIGraphicEx:** Represents a control that displays a graphical image to the user. Currently, the XPages version does not add any new behavior, but it might do so in the future.
- **UIInclude:** Used to support including one XPage within another.
- **UIIncludeComposite:** Used to support including a Custom Control within an XPage.
- **UIInputCheckbox:** Represents a checkbox control.

- **UIInputEx:** Used for controls that display a value and allow that value to be edited. `UIInputEx` adds support for HTML filtering, disabling the validation, and Dojo.
- **UIInputRadio:** Represents a radio button control.
- **UIInputRichText:** Represents a rich text edit control.
- **UIInputText:** Represents an edit box control.
- **UIMessageEx:** Supports the display of error messages for a specific component and adds style kit support.
- **UIMessagesEx:** Supports the display of error messages not related to a specific component and adds theme support.
- **UIOutputEx:** Used to display data model values to the user and adds HTML filtering support.
- **UIOutputLink:** Represents a HTML link.
- **UIOutputText:** Displays a computed value.
- **UIPager:** Used to display a pager control to allow paging through the rows of data associated with the `UIData` or `UIDataIterator` component.
- **UIPagerControl:** Used to display one of the buttons in a pager control, such as first, previous, next, or last buttons.
- **UIPanelEx:** Used as a base component for the controls that are used include an XPage.
- **UIPassThroughTag:** Used whenever a non-xsp tag is added to an XPage. There is no associated xsp tag for this component, but it is used by the page-loading mechanism and appears in the XPages page-translation source code.
- **UIPassThroughText:** Used whenever text added to an XPage. There is no associated xsp tag for this component, but it is used by the page-loading mechanism and appears in the XPages page translation source code.
- **UIPlatformEvent:** Represents a control that can handle a platform event. When the specified platform event occurs, the associated script is executed.
- **UIRepeat:** This component is a `FacesDataIterator` and has two modes of operation: It can either use either a single instances of its children (like a `UIData` component) or it can create one instance of its children for every row of data.
- **UIRepeatContainer:** Used by the `UIRepeat` component when it is creating multiple instances of its children. Each instance of `UIRepeats` children are nested inside a `UIRepeatContainer`, and the container provides access to the row data and index.
- **UIScriptCollector:** Automatically added to the root of an XPages component tree. Its job is to aggregate all the JavaScript code that needs to be included in the generated HTML and to include it within a single script tag at the bottom of the page.

- **UISection:** Represents a container control that displays as a section and can be expanded and collapsed.
- **UISelectItemEx:** Represents a single selection option for a control that allows the user to select from a number of choices, such as a listbox.
- **UISelectItemsEx:** Represents multiple section options for a control that allows the user to select from a number of choices, such as a listbox.
- **UISelectListbox:** A listbox control, which will have nested `UISelectItemEx` or `UISelectItemsEx`, representing the available choices. Depending on whether the listbox is configured for multiple selection, a different specialized XPages component is used, either `XspSelectManyListbox` or `XspSelectOneListbox`.
- **UISelectManyEx:** Represents a control that allows the user to select multiple values from a number of choices.
- **UISelectOneEx:** Represents a control that allows the user to select one value from a number of choices.
- **UITabbedPanel:** Represents a control that contains children which are instances of `UITabPanel` and displays the children as a series of tabs.
- **UITabPanel:** Represents a single tab in a tabbed panel control.
- **UITypeAhead:** A helper component that is used with an edit box to provide type-ahead functionality, such as the ability for the user to start typing in the edit box and see a list of suggestions.
- **UIViewColumn:** Represents a single column in a view control.
- **UIViewColumnHeader:** Represents the header for a single column in a view control.
- **UIViewPager:** Represents a pager in a view control.
- **UIViewPanel:** Represents a view control that can be bound to the data in a Domino view.
- **UIViewRootEx2:** The root component of all XPages component hierarchies.
- **UIViewTitle:** Represents the title of a view control.

You could use one of the standard user interface components as the base class if you were building your own Java components for XPages. This subject is covered in Chapter 12.

## Value Binding and Method Binding Expression Evaluation

JSF supports two types of binding expressions:

- **Value binding:** Computes a value for a property and can support both reading and writing a value
- **Method binding:** Executes some logic

Binding expressions are identified using the `{` and `}` expression delimiters. JSF supports Expression Language (EL) for value and method bindings. JSF defines the `javax.faces.el.ValueBinding` abstract class to represent a value binding expression and `javax.faces.el.MethodBinding` to represent a method binding. The JSF application object is responsible for creating instances of these for use in the JSF processing. XPages extends support for expression binding to include the following:

- Using JavaScript
- Using a special syntax to resolve client IDs
- Support for multipart expressions
- Simple actions

### JavaScript Binding Expressions

A JavaScript binding expression is delimited using `{javascript: and }`. Listing 5.7 shows an example of a JavaScript value binding expression being used to compute the value for a Computed Field. When this code is executed, a string representation of the database property is displayed in the Computed Field. This value binding expression is computed each time the property is accessed. There is an alternative syntax that starts with `$`, which gets evaluated just once when the page is loaded.

#### Listing 5.7 JavaScript Value Binding Expression

```
<xp:text escape="true"
        id="computedField2"
        value="{javascript:database}">
</xp:text>
```

Listing 5.8 shows the syntax for the JavaScript method binding. When the button is clicked, the XPage is submitted and the JavaScript executes. The output from the `print` statement can be seen in the Domino console or Notes trace file.

#### Listing 5.8 JavaScript Method Binding Expression

```
<xp:button value="Execute JavaScript" id="button1" type="submit">
  <xp:this.action>
    <![CDATA[{javascript:print("Executed JavaScript")}]]>
  </xp:this.action>
</xp:button>
```

### Client ID Binding Expressions

An ID binding expression is delimited using `{id: }`. The ID of the user component whose client ID you want to compute is specified in the content of the computed expression, as shown in Listing 5.9. ID expressions are typically used as part of a multipart expression.

---

**Listing 5.9** Client ID Binding Expression

---

```
<xp:text escape="true"
         id="computedField3"
         value="{id:computedField3}">
</xp:text>
```

---

### Multipart Binding Expressions

A multipart expression allows static and dynamic content to be mixed. In Listing 5.10, the value of the Computed Field combines static text, a client ID computed expression, and a JavaScript computed expression.

---

**Listing 5.10** Multipart Value Binding Expression

---

```
<xp:text escape="true"
         id="computedField4"
         value="ID: {id:computedField4} DB: {javascript:database}">
</xp:text>
```

---

### Simple Actions

A simple action is a special type of method binding which is represented by a tag in the XPage and its behavior can be configured using properties. Listing 5.11 shows how to configure a simple ExecuteScript action, which, in turn, invokes a JavaScript method binding.

---

**Listing 5.11** Simple Action Method Binding Expression

---

```
<xp:button value="Execute Simple Action" id="button2" type="submit">
  <xp:this.action>
    <xp:executeScript
      script="{javascript:print('Executed Simple Action')}">
    </xp:executeScript>
  </xp:this.action>
</xp:button>
```

---



## XPages Default Variables

Earlier in this chapter, the default JSF variables were listed (see Table 5.1). XPages provides some additional default variables for the Domino application developer. Table 5.4 shows a listing of all the default variables, their values, and a short description of each variable. Refer to the XPage named DefaultVariables in **Chp05Ed2.nsf** to see how this table was generated. At the end of the list, the six new XPages default variables are listed:

- **viewScope:** Map containing the view scope values
- **context:** XspContext instance for the current request
- **database:** Database instance for the current request
- **session:** Session instance for the current request
- **sessionAsSigner:** Session instance with the credentials of the XPage signer
- **sessionAsSignerWithFullAccess:** Session instance with the credentials based on those of the XPager signer and with full administrative access

**Table 5.4** Table of Default Variables

Name	String Value	Description
applicationScope	{ com.sun.faces.OneTimeInitialization=com.sun.faces.OneTimeInitialization, com.sun.faces.ApplicationAssociate=com.sun.faces.application.ApplicationAssociate@51a551a5 com.sun.faces.HTML_BASIC=com.ibm.xsp.renderkit.ReadOnlyRenderKit@45f045f0 javax.servlet.context.tempdir=C:\Users\Mark\AppData\Local\Temp\notes0E3C5E\xsp\chp05ed2.nsf }	Map containing the application scope values
cookie	{ SessionID=javax.servlet.http.Cookie@63a863a8 }	Map containing the cookies for the current request
facesContext	com.ibm.xsp.domino.context.DominoFacesContext@6d346d34	The FacesContext instance for the current request

Name	String Value	Description
header	<pre>{ Cookie=SessionID=81A74A5C8161D376B0373C275 603E4BB8E12AE5B Accept-Encoding=gzip, deflate Accept=text/html,application/xhtml+xml,application/ xml;q=0.9*/;*;q=0.8 Accept-Language=en-ie,en;q=0.7 en-us;q=0.3 User-Agent=Mozilla/5.0 (Windows NT 6.1; WOW64; rv:23.0) Gecko/20100101 Firefox/23.0 Referer=http://localhost/chp05ed2.nsf Connection=keep-alive, Host=localhost }</pre>	Map containing the HTTP header values for the current request
headerValues	<pre>{ Accept=com.ibm.domino.xsp.bridge.http.util. SingleValueEnumeration@43d843d8 Referer=com.ibm.domino.xsp.bridge.http.util. SingleValueEnumeration@443c443c Accept-Encoding=com.ibm.domino.xsp.bridge.http. util.SingleValueEnumeration@441c441c Connection=com.ibm.domino.xsp.bridge.http.util. SingleValueEnumeration@44eb44eb Accept-Language=com.ibm.domino.xsp.bridge.http. util.SingleValueEnumeration@43fa43fa User-Agent=com.ibm.domino.xsp.bridge.http.util. SingleValueEnumeration@43b843b8 Cookie=com.ibm.domino.xsp.bridge.http.util. SingleValueEnumeration@44ca44ca Host=com.ibm.domino.xsp.bridge.http.util.SingleValue Enumeration@43974397 }</pre>	Map containing arrays which contain the header values for the HTTP headers for the current request
initParam	<pre>{ com.sun.faces.forceLoadConfiguration=true com.ibm.xsp.SHARED_CONFIG=true com.sun.faces.verifyObjects=false }</pre>	Map containing the initialization parameters for the web application
param	<pre>{ }</pre>	Map containing the request parameters for the current request

Name	String Value	Description
paramValues	{}	Map containing arrays which contain the parameter values for request parameters for the current request
requestScope	{com.ibm.xsp.SESSION_ID=81A74A5C8161D376B0373C275603E4BB8E12AE5B __xspconvid=null database=chp05ed2.nsf session=CN=MarksW520/O=DEV com.sun.faces.FORM_CLIENT_ID_ATTR=view:_id1 context=com.ibm.xsp.designer.context.ServletXSP-Context@f450f45 com.sun.faces.INVOCATION_PATH=.xsp cookie={ SessionID=javax.servlet.http.Cookie@63a863a8} componentParameters=com.ibm.xsp.application.ComponentParameters@ff80ff8}	Map containing the request attributes for the current request
sessionScope	{__XSP_STATE_BASIC=com.ibm.xsp.application.BasicStateManagerImpl\$ViewHolder@44f444f4 VIEW LIST: !dnms5mob49!/Index __notescontext_publicaccess=com.ibm.domino.xsp.module.nsf.NotesContext\$AccessPrivileges@47544754 xspIsBot=false xsp.sessionData=com.ibm.xsp.designer.context.PersistentSessionData@fb90fb9}	Map containing the session attributes for the current request
view	com.ibm.xsp.component.UIViewRootEx2@753f753f	UIViewRoot of the current component tree
viewScope	{}	Map containing the view scope values
context	com.ibm.xsp.designer.context.ServletXSPContext@f450f45	The XspContext instance for the current request
database	chp05ed2.nsf	The Database instance for the current request

Name	String Value	Description
session	CN=MarksW520/O=DEV	The Session instance for the current request
sessionAsSigner	CN=MarksW520/O=DEV	The Session instance with the credentials of the XPage signer
sessionAsSigner- WithFullAccess	CN=MarksW520/O=DEV	The Session instance with the credentials of the XPage signer and with full administrative access

Chapter 6, “Building XPages Application Logic,” covers XPages default variables and examples of their usage in more detail. A short description of each of the XPages default variables is provided next.

### viewScope

XPages introduces this new scoped variable to supplement the default scoped variables: requestScope, sessionScope, and applicationScope. The viewScope variable allows you to scope your own variables to the lifetime of the associated view, such as XPage. As previously mentioned, the state of a view can be cached between requests so that multiple requests act on the same state of the XPage. The view is restored at the beginning and saved at the end of each request and any view scope variables are saved and restored as part of this process. The viewScope object is a map, so you can add your own variables keyed by name. By default, this map is empty, so you can select whatever names you want without concern for name clashes. The variables you add must be serializable for their state to be saved.

### context

The context variable provides access to the XPages XSPContext object, which is an instance of `com.ibm.xsp.designer.context.XSPContext`. The context object provides XPages-specific contextual information about the current request, such as access to the associated user, timezone, locale, and so on. It also provides numerous utility methods that can be used within your application logic, such as page navigation, HTML filtering, and so on.

### database

The database variable provides access to the Database object, which is an instance of `lotus.domino.Database`. The database object provides access to the current Domino database and supports a wide range of database centric operations. The complete documentation for the

Database class is available in the Java/CORBA Classes section of the IBM Domino Designer Basic User Guide and Reference help document, which is part of the Domino Designer help. Use **Help > Help Contents** to access this documentation.

### **session**

The `session` variable provides access to the `Session` object, which is an instance of `lotus.domino.Session`. The session is assigned credentials based on those of the current user. The session is restricted by the application's ACL and the security tab of the server's Domino Directory entry. The complete documentation for the `Session` class is available in the Java/CORBA Classes section of the IBM Domino Designer Basic User Guide and Reference help document.

### **sessionAsSigner**

The `session` variable provides access to the `Session` object, which is an instance of `lotus.domino.Session`. The session is assigned credentials based on those of the signer of the XPages' design element. The session is restricted by the application's ACL and the Security tab of the server's Domino Directory entry. The complete documentation for the `Session` class is available in the Java/CORBA Classes section of the IBM Domino Designer Basic User Guide and Reference help document.

### **sessionAsSignerWithFullAccess**

The `session` variable provides access to the `Session` object, which is an instance of `lotus.domino.Session`. The session is assigned credentials based on those of the signer of the XPages' design element and allows full administrative access to the application's data. The signer must have permission for full administrative access or this session is not created and will not be available. The complete documentation for the `Session` class is available in the Java/CORBA Classes section of the IBM Domino Designer Basic User Guide and Reference help document.

## **Conclusion**

This concludes the overview of how XPages is built on top of JSF. You learned how XPages extends JSF to add new capabilities and enhanced behaviors while maintaining the JSF standard. As previously mentioned, XPages is currently built with JSF version 1.1, so if you plan to read more about JSF, this is the version to reference.



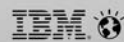
# Be Open - Use WebServices and REST in XPages

*Bernd Hort, assono GmbH*

IBM Software  
**ConnectED2015**

## BP108: Be Open - Use WebServices and REST in XPages

Bernd Hort, assono GmbH



## Agenda

- Introduction / Motivation
- Consume Web Services from Java
- Consume RESTful Web Services from Java
- Consume Web Services from JavaScript
- Consume RESTful Web Services from JavaScript
- dojo Controls
- Questions and Answers

# INTRODUCTION / MOTIVATION

IBM

ConnectED2015

## Bernd Hort assono GmbH

- IBM® Notes® Developer/Designer since Notes 3.3
- OpenNTF Contributor: [assono Framework 2](#)
  - An OOP LotusScript Framework
- Lately specialized in XPages development
- Working for [assono GmbH](#), Germany
- Blog <http://blog.assono.de>
- Twitter [@BerndHort](#)

IBM

ConnectED2015

Remote Systems

Language

Internet

Be Independent Social

Open

API Application  
Programming  
Interface

Standard

IBM

ConnectED2015

# CONSUME WEB SERVICES FROM JAVA

IBM

ConnectED2015

## Web Services in a Nutshell

- Communication between two machines
  - Web Service Consumer and Web Service Provider
- Over a network
  - Mostly using HTTP
- Data encoded as XML
- Messages transferred in SOAP envelops
- Specified by the World Wide Web Consortium (W3C)
- Definition of a Web Service in WSDL
  - Web Service Description Language



IBM

ConnectED2015

## Consume Web Services in Java

- Java EE 6 has built-in classes to consume Web Services:  
[Java API for XML Web Services \(JAX-WS\)](#)
- The Apache project [CXF](#) provides a tool to generate Java files from a WSDL file  
[wsdl2java](#)
- The generated Java classes maps through Java Annotations the Web Service actions to Java methods using Java Architecture for XML Binding - JAXB
- XPages runtime builds on Java EE 6



IBM

ConnectED2015

## Apache CXF – wsdl2java

- [Download](#) CXF from the Apache project site
- Download the wsdl file from the remote system or access it directly from the remote system
- Call the wsdl2java.bat file with the following parameters

-client	To generate a sample Java class calling the Web Service
-exsh true	Enables or disables processing of implicit SOAP headers (i.e. SOAP headers defined in the wsdl:binding but not wsdl:portType section.)
-frontend jaxws21	Currently only supported frontend JAX-WS 2.1
-verbose	Displays comments during the code generation process.
-d {output directory}	The path for the generated Java classes.
-p {package name}	Java package name for the generated Java classes
{wsdl location}	The file path or URI for the WSDL file

IBM

ConnectED2015

## Modification of java.policy file needed!

- JAX-WS uses Java Reflection
- A modification of the java.policy file on the IBM Domino Server is needed to allow a different class loader
  - {Domino Program Path}/jvm/lib/security/java.policy
- The following lines have to be added

```
grant {
    permission java.lang.RuntimePermission "setContextClassLoader";
    permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
};
```



IBM

ConnectED2015

## Basic Authentication

- Basic Authentication on the HTTP level is widely used for securing Web Services
- JAX-WS supports Basic Authentication

```
import javax.xml.namespace.QName;
import javax.xml.ws.BindingProvider;
import javax.xml.ws.Service;

protected static final QName SERVICE_NAME = new QName("{WebServiceTargetNamespace}",
    "{WebServiceName}");

Service service = {YourWebServiceImplementationClass}Service.create(wsdlURL,
    SERVICE_NAME);
port = service.getPort({YourWebServiceClass}.class);

Map<String, Object> ctx = ((BindingProvider) port).getRequestContext();
ctx.put(BindingProvider.USERNAME_PROPERTY, username);
ctx.put(BindingProvider.PASSWORD_PROPERTY, password);
```

IBM

ConnectED2015

## WSDL file protected with Basic Authentication

- At runtime the WSDL definition is needed for the "Service Endpoint" a.k.a. the Web Service URL
- Sometimes the WSDL itself is protected with Basic Authentication
  - Workaround is a Notes application with anonymous access as a WSDL proxy
  - Form with content type "text/xml"

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"[...]>
  <types>[...]</types>
  <message>[...]</message>
  <portType>[...]</portType>
  <binding>[...]</binding>
  <service name="{WebServiceImplementation}Service">
    <port name="{WebService}Port">
      <soap:address location="{URL of Web Service}"></soap:address>
    </port>
  </service>
</definitions>
```

IBM

ConnectED2015

## CONSUME RESTful WEB SERVICES FROM JAVA

IBM

ConnectED2015

## REST in a Nutshell



- Representational state transfer (REST) is an architectural style based on web standards for accessing resources
- RESTful Web Services implements access and modification of resources
- Mostly using HTTP
- Resources may have different representations, e.g. text, xml, json etc.
  - The rest client can ask for specific representation via the HTTP protocol (content negotiation).
- No standard!
- The WSDL equivalent Web Application Description Language (WADL) is mostly optional
  - Hopefully the documentation is good ;-)

IBM

ConnectED2015



## REST in a Nutshell – Nouns and Verbs

- Resources are identified via a URI - Nouns
- Actions on resources uses the HTTP methods - Verbs

Resource	Get	Post	Put	Delete
Collections of resources <a href="http://yourserver/path/sessions">http://yourserver/path/sessions</a>	<b>Lists</b> all resources	<b>Create</b> a new entry in the collection.	<b>Replace</b> the entire collection with another collection.	<b>Delete</b> the entire collection
Single resource <a href="http://yourserver/path/sessions/BP206">http://yourserver/path/sessions/BP206</a>	<b>Retrieve</b> the details of a given resource	<i>Normally not implemented</i>	<b>Replace</b> the addressed member of the collection, or if it doesn't exist, <b>create</b> it.	<b>Delete</b> the specified resource

Also known as create, read, update and delete (CRUD)

## Consume RESTful Web Services in Java

- Java EE 7 has built-in classes to consume RESTful Web Services: [Java API for RESTful Services \(JAX-RS\)](#)
- XPages runtime builds on Java EE 6 😞
- Sun / Oracles reference implementation [Jersey](#) provides all needed Java classes
  - Jersey version 1.71.1 is the last stable version supporting Java 6 and JAX-RS 1.1



## Consume RESTful Web Services in Java - XML

- If the RESTful Web Service supports XML, Jersey just uses Java Architecture for XML Binding - JAXB to map the XML to Java classes
- At runtime Java classes provided with Java Annotations and the same property names are mapped to the data from Web Services



## Example RESTful Web Services in Java

### Representation data as JSON

```
[{
  sessionId: "BP108",
  title: "Be Open - Use Web Services and REST in XPages Applications",
  description: "...",
  speakers: [{
    name: "Bernd Hört",
    company: "assono GmbH"}]
},
{
  sessionId: "Chalk405",
  title: "XPages and Java: Share Your Experiences",
  description: "...",
  speakers: [{
    name: "Bernd Hört",
    company: "assono GmbH"}]
}]
```



ConnectED2015

## Example RESTful Web Services in Java

### Java Class

```
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class Speaker {

    private String name;
    private String company;

    // Getter and setter methods
    // [...]
}

@XmlRootElement
public class Session {

    private String sessionId;
    private String title;
    private String description;
    private Speaker[] speaker;

    // Getter and setter methods
    // [...]
}
```



ConnectED2015

## Example RESTful Web Services in Java

### Java Class for accessing the Web Service

```
import java.net.URI;

import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.UriBuilder;

import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.config.ClientConfig;
import com.sun.jersey.api.client.config.DefaultClientConfig;

public class RESTClient {
    public static Session[] getSessions() {
        ClientConfig config = new DefaultClientConfig();
        Client client = Client.create(config);
        WebResource service = client.resource(getBaseURI());

        return service.path("sessions").accept(MediaType.TEXT_XML).get(Session[].class);
    }

    private static URI getBaseURI() {
        return UriBuilder.fromUri("http://{server}/{applicationpath}").build();
    }
}
```



ConnectED2015

## Consume RESTful Web Services in Java - JSON

- Instead of using JAXB to map XML to Java classes use a JSON to Java mapper
- Jersey 1.71.1 comes with [Jackson](#) 1.9.2 from Codehouse.org
  - Recently moved to <https://github.com/FasterXML/jackson>



## Example RESTful Web Services in Java

- Java Class for accessing the Web Service

```
import [...]  
  
import org.codehaus.jackson.map.ObjectMapper;  
  
public class RESTClient {  
    public static Session[] getSessions() {  
        ClientConfig config = new DefaultClientConfig();  
        Client client = Client.create(config);  
        WebResource service = client.resource(getBaseURI());  
  
        String json = service.path("Sessions").accept(MediaType.APPLICATION_JSON).get(String.class);  
  
        ObjectMapper mapper = new ObjectMapper();  
        sessions = mapper.readValue(json, Session[].class);  
        return sessions;  
    }  
  
    private static URI getBaseURI() {  
        return UriBuilder.fromUri("http://{server}/{applicationpath}").build();  
    }  
}
```

## Modification of java.policy file needed!

- Jersey uses a different class loader
- So another modification of the java.policy file on the IBM Domino Server is needed to allow a different class loader
  - {Domino Program Path}/jvm/lib/security/java.policy
- The following lines needs to be added

```
grant {  
    permission java.lang.RuntimePermission "setContextClassLoader";  
    permission java.lang.RuntimePermission "getClassLoader";  
    permission java.lang.reflect.ReflectPermission "suppressAccessChecks";  
};
```



# CONSUME WEB SERVICES FROM JAVASCRIPT

IBM

ConnectED2015

## Consume Web Services from JavaScript

- Historically is consuming a Web Services the base for all AJAX style techniques
  - The JavaScript class [XMLHttpRequest](#) handles the communication with a server from within a web site
- Astonishingly the support for calling Web Services is still mostly hand coding
  - No efforts are taken because most servers offers REST nowadays
- The SOAP Envelope must be built and send to a server using XMLHttpRequest
- The returning XML has to be parsed

IBM

ConnectED2015

```
<script type="text/javascript">
function soap() {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open('POST', 'https://somesoapurl.com/', true);

    // build SOAP request
    var sr =
    '<?xml version="1.0" encoding="utf-8"?>' +
    '<soapenv:Envelope ' +
    '  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ' +
    '  xmlns:api="http://demo web service" ' +
    '  xmlns:xsd="http://www.w3.org/2001/XMLSchema" ' +
    '  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">' +
    '  <soapenv:Body>' +
    '    <api:some_api_call soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">' +
    '      <username xsi:type="xsd:string">login_username</username>' +
    '      <password xsi:type="xsd:string">password</password>' +
    '    </api:some_api_call>' +
    '  </soapenv:Body>' +
    ' </soapenv:Envelope>';

    xmlhttp.onreadystatechange = function () {
        if (xmlhttp.readyState == 4) {
            if (xmlhttp.status == 200) {
                alert('done - use firebug to see response');
            }
        }
    }
    // Send the POST request
    xmlhttp.setRequestHeader('Content-Type', 'text/xml');
    xmlhttp.send(sr);
    // send request
    // ...
}
</script>
```

IBM

ConnectED2015



## How to get a SOAP Envelope?

- A good way to get the SOAP Envelope needed for a specific Web Service is using a test tool
- Point the test tool to the WSDL to let it generate the test cases
- Invoke the Web Service with sample data and copy the SOAP Envelope
- A good choice is [SoapUI](#)
- Or use the Web Service Explorer from Eclipse
  - Open the Java EE perspective
  - Menu Run\Launch the Web Service Explorer
  - Change to the WSDL page
  - Fill in the sample data
  - Click on "Source"



IBM

ConnectED 2015

## dojo Support

- dojo may not have direct support for consuming SOAP Web Services
- But dojo has a lot of helpers to offer
  - the [dojo](#) base object
    - [dojo.formToJson](#) : Convert a DOM Form to JSON.
    - [dojo.formToObject](#) : Convert a DOM Form to a JavaScript object.
    - [dojo.formToQuery](#) : Convert a DOM Form to a query string.
    - [dojo.objectToQuery](#) : Convert a JavaScript object to a query string.
    - [dojo.queryToObject](#) : Convert a query string to a JavaScript Object
  - [dojo.xhr](#) offers AJAX I/O transports and utility methods
    - [dojo.xhrDelete](#) : Use HTTP DELETE method to make an xhr call.
    - [dojo.xhrGet](#) : Use HTTP GET method to make an xhr call.
    - [dojo.xhrPost](#) : Use HTTP POST method to make an xhr call.
    - [dojo.xhrPut](#) : Use HTTP PUT method to make an xhr call.

dojo

IBM

ConnectED 2015

## dojo Support

- [dojox.rpc](#) communicate via Remote Procedure Calls (RPC) with Backend Servers
- [dojox.rpc.Service](#) is the foundation of most dojox.RPC transportation
- A [Service Mapping Description \(SMD\)](#) is a JSON representation describing web services and it is used by [dojox.rpc.Service](#).
  - Dojo comes with predefined SMDs for accessing
    - Google
    - Twitter
    - Yahoo!
    - Wikipedia

dojox

IBM

ConnectED 2015



# CONSUME RESTful WEB SERVICES FROM JAVASCRIPT

IBM

ConnectED2015

## Consume RESTful Web Services from JavaScript

- Either do it the hard way and use the JavaScript class [XMLHttpRequest](#) directly
- Or use dojo
  - [dojo.xhrDelete](#) : Use HTTP DELETE method to make an xhr call.
  - [dojo.xhrGet](#) : Use HTTP GET method to make an xhr call.
  - [dojo.xhrPost](#) : Use HTTP POST method to make an xhr call.
  - [dojo.xhrPut](#) : Use HTTP PUT method to make an xhr call.

dojo

IBM

ConnectED2015

## dojo.xhrGet

- dojo.xhrGet expects a parameter object with the following properties

Parameter	Description
url	The URL to request data from.
handleAs	The expected format of the data. The currently supported options are: <ul style="list-style-type: none"> <li>•text (default)</li> <li>•json</li> <li>•json-comment-optional</li> <li>•json-comment-filtered</li> <li>•javascript</li> <li>•xml</li> </ul>

IBM

ConnectED2015

## dojo.xhrGet – Optional Parameters

Parameter	Description
<b>sync</b>	Boolean value whether to block the browser while the request is running. Default is false.
<b>preventCache</b>	Boolean value – if set to yes dojo adds a unique query parameter to each request to prevent the browser cache
<b>content</b>	A JavaScript object with name/string value pairs. dojo will encode the values and append it to the query. E.g. ?key1=value1&key2=value2&key3=value3
<b>headers</b>	A JavaScript object of name/string value pairs. The headers are send as part of the request.
<b>timeout</b>	Number of milliseconds to wait until timing out the request. Default is '0', which means infinite (no timeout).
<b>user</b>	For Basic web authentication the username.
<b>password</b>	For Basic web authentication the password.

## dojo.xhrGet – Parameters for Handling the Data

Parameter	Description
<b>load</b>	A JavaScript function invoked when the data is returned from the server. The data will be passed as the first parameter of the function. The format of the data is controlled by the previously mentioned handleAs parameter.
<b>error</b>	A JavaScript function called in case of an error. The first parameter passed to the error function is a JavaScript Error object indicating what the failure was.
<b>handle</b>	A JavaScript function called regardless if the request was successful or not. The first parameter passed to this callback is the response (or error) and the second parameter is the IO args object, from which you can get the status code and determine success or failure.

## dojo.xhrGet - Example

```
var xhrArgs = {
  url: "http://www.example.com/someservice/sessions/BP108",
  handleAs: "json",
  load: function(data){
    dojo.byId("#{id:titleSpan1}").innerHTML = data.title;
  },
  error: function(error){
    dojo.byId("#{id:titleSpan1}").innerHTML = "An unexpected error occurred: "
    + error;
  }
};
dojo.xhrGet(xhrArgs);
```

## dojo.xhrPost

- dojo.xhrPost accepts the same parameter as dojo.xhrGet with the following addition

Parameter	Description
<b>content</b>	A JavaScript object of name/string value pairs. xhrPost will convert this into proper POST format and send it with the post data. Note that this parameter is handled differently from dojo.xhrGet, which encodes it as a query string in url.
<b>form</b>	For posting FORM data, you can provide either the DOM node of your form or the ID of the form. xhrPost will convert this into proper POST format and send it with the post data. If a url is not set in the args to dojo.xhrPost, then it tries to extract the url from the form 'action' attribute.
<b>postData</b>	A string of data you wish to send as the post body. dojo.xhrPost (and dojo.rawXhrPost), do not do any processing of this It is merely passed through as the POST body.

Only one of these parameters can be used in a request.

## dojo.xhrPut

- dojo.xhrPut accepts the same parameter as dojo.xhrGet with the following addition

Parameter	Description
<b>content</b>	A JavaScript object of name/string value pairs. xhrPost will convert this into proper POST format and send it with the post data. Note that this parameter is handled differently from dojo.xhrGet, which encodes it as a query string in url.
<b>form</b>	For posting FORM data, you can provide either the DOM node of your form or the ID of the form. xhrPost will convert this into proper POST format and send it with the post data. If a url is not set in the args to dojo.xhrPost, then it tries to extract the url from the form 'action' attribute.
<b>postData</b>	A string of data you wish to send as the post body. dojo.xhrPost (and dojo.rawXhrPost), do not do any processing of this It is merely passed through as the POST body.

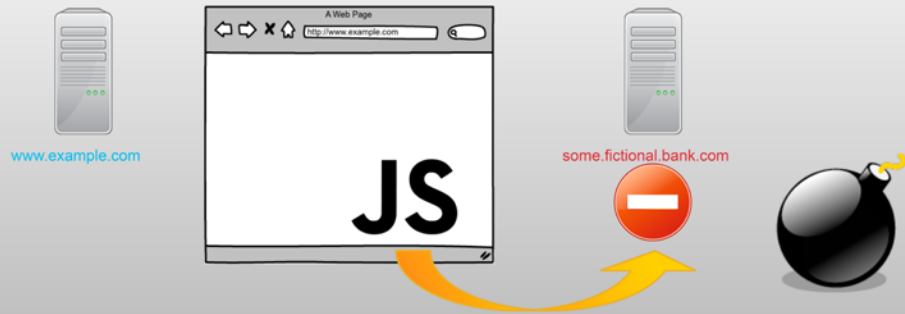
Only one of these parameters can be used in a request.

## dojo.xhrDelete

- dojo.xhrDelete accepts the same parameter as dojo.xhrGet.
- There is no **content** parameter in dojo.xhrDelete because the URI (URL + query params) defines what to delete.

## Same Origin Policy - SOP

- For security reasons a script running in a browser can only access resources from the same server (origin)
- This limitation also applies for Web Services calls



IBM

ConnectED2015

## JSONP – JSON with padding

A solution with side effects

- The Same Origin Policy does not apply to JavaScript tags
  - The JavaScript source could be referenced from any server
- JSONP uses the URI of an embedded JavaScript tag to send a request
- The requested data is returned as a function call to an existing JavaScript function
- This technique is limited to GET requests
- Potential security issue: Whatever data is returned from the “other” server is executed as a JavaScript function in the context of the current browser session
- dojo supports JSONP with the [dojo.io.script](http://dojotoolkit.org/api/1.9/dojo.io.script) package



IBM

ConnectED2015

## JSONP – JSON with padding

A solution with side effects

```
JSON-Data to be returned
{
  sessionId: "BP108",
  title: "Be Open - Use Web Services and REST in XPages Applications",
  description: "...",
  speaker: {
    name: "Bernd Hort",
    company: "assono GmbH"
  }
}
```

```
Call to get data
<script type="application/javascript"
src="http://{someserver}/Sessions/BP108?jsonp=updateSessionDetails">
</script>
```

```
Returned JavaScript
updateSessionDetails({
  sessionId: "BP108", title: "Be Open - Use Web Services and REST in XPages
Applications", description: "...", speaker: {name: "Bernd Hort",
company: "assono GmbH"}
})
```



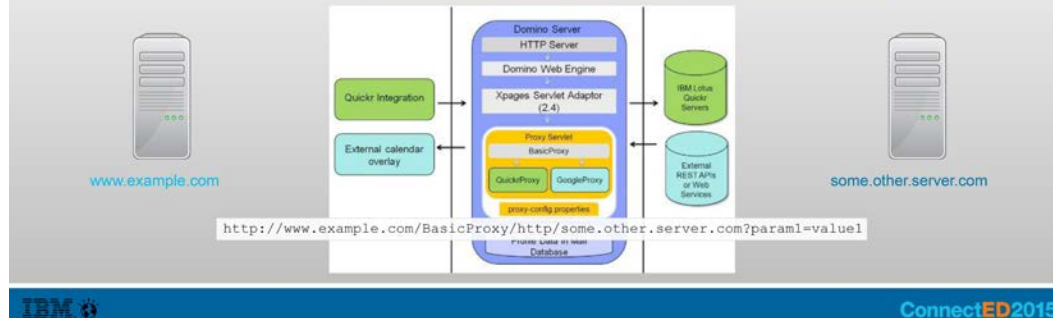
IBM

ConnectED2015



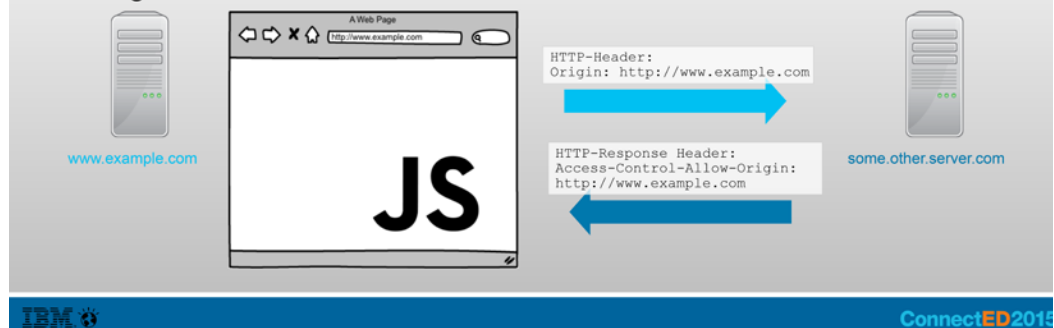
## IBM® iNotes® Proxy Servlet

- In IBM® Lotus® iNotes® 8.5 a proxy servlet was introduced to overcome the SOP
- A request going through the proxy will have the same origin seen from the browsers
- Details can be found in [New features in IBM Lotus iNotes 8.5: Full mode](#)



## Cross-origin resource sharing – CORS

- With Cross-origin resource sharing – CORS the origin is added to the header of the request
- The requesting server returns in the response header if he accepts the origin domain



## Cross-origin resource sharing – CORS

- CORS works on all HTTP actions: GET, POST, PUT, DELETE ...
- CORS is supported in all modern browsers (from [Wikipedia](#))
  - Gecko 1.9.1 (Firefox 3.5, SeaMonkey 2.0, Camino 2.1) and above.
  - WebKit (Initial revision uncertain, Safari 4 and above, Google Chrome 3 and above, possibly earlier)
  - MSHTML/Trident 6.0 (Internet Explorer 10) has native support. MSHTML/Trident 4.0 & 5.0 (Internet Explorer 8 & 9) provides partial support via the XDomainRequest object.
  - Presto-based browsers (Opera) implement CORS as of Opera 12.00 and Opera Mobile 12, but not Opera Mini.



# dojo CONTROLS

IBM

ConnectED2015

## dojo Store API



- The dojo Store API is an abstract API for handling data based on [HTML5/W3C's IndexedDB object store API](#)

Method	Description
<b>get(id)</b>	Retrieves an object by its identifier, returning the object.
<b>query(query, options)</b>	Queries the store using the provided query. Options include <ul style="list-style-type: none"> <li>start - Starting offset</li> <li>count - Number of objects to return</li> <li>sort - Follows the Dojo Data sort definition</li> <li>queryOptions - Follows the Dojo Data queryOptions definition</li> </ul>
<b>put(object, options)</b>	Saves the given object. options.id (optional) indicates the identifier.
<b>add(object, options)</b>	Create a new object. options.id (optional) indicates the identifier.
<b>remove(id)</b>	Delete the object by id..

IBM

ConnectED2015

## dojo Store API (continue)



Method	Description
<b>getIdentity(object)</b>	Returns an object's identity
<b>queryEngine(query, options)</b>	This takes a query and query options and returns a function that can execute the provided query on a JavaScript array. The queryEngine may be replace to provide more sophisticated querying capabilities. The returned query function may have a "matches" property that can be used to determine if an object matches the query.
<b>transaction()</b>	Starts a transaction and returns a transaction object. The transaction object should include: <ul style="list-style-type: none"> <li>commit() - Commits all the changes that took place during the transaction.</li> <li>abort() - Aborts all the changes that took place during the transaction.</li> </ul> Note that a store user might not call transaction() prior to using put, delete, etc. in which case these operations effectively could be thought of as "auto-commit" style actions.
<b>getChildren(object, options)</b>	Returns the children of an object. The options parameter may include the same properties as query options
<b>getMetadata(object)</b>	Returns any metadata about the object. This may include attribution, cache directives, history, or version information. (addresses #3126, #3127)

IBM

ConnectED2015

## dojo.store.JsonRest

- The [dojo.store.JsonRest](#) handles the full HTTP/REST communication with a server based on the HTTP standards using the full range of GET, PUT, POST, and DELETE methods
- It implements all dojo Store API
- A JsonRest store can be instantiated with a URL

```
var store = new dojo.store.JsonRest({
  target: "/Sessions/",
  idProperty: "sessionId"
});
```

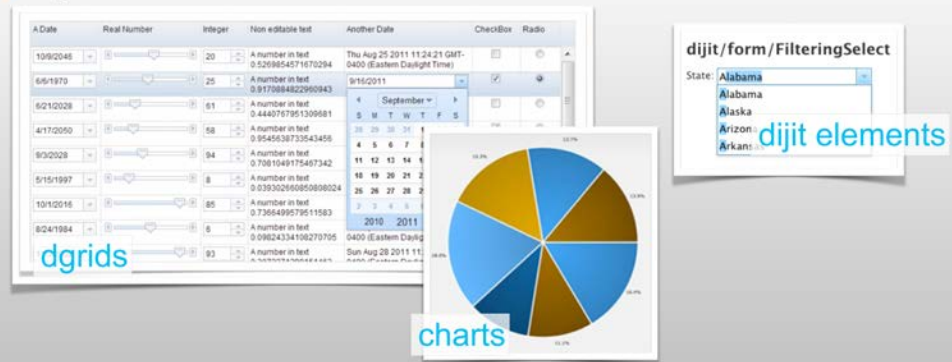
- Paging is supported
- Sorting is supported

IBM

ConnectED2015

## Use of dojo Stores

- dojo Data Stores can be used as data sources for



IBM

ConnectED2015

## XPages Extension Library

- The XPages Extension Library has components that provide or uses RESTful Web Services
- Components that provide RESTful Web Services
  - Domino Data Services
  - RESTService control
  - Custom Database Servlet
  - Custom Wink Service
- Controls that uses RESTful Web Services
  - Dojo Data Grid



IBM

ConnectED2015

## Thank You!

- Get the latest version of this presentation and the sample database from <http://www.assono.de/blog/d6plinks/ibmconnected2015-bp108>



ConnectED2015

## Engage Online

- **SocialBiz User Group** [socialbizug.org](http://socialbizug.org)
  - Join the epicenter of Notes and Collaboration user groups
- **Social Business Insights blog** [ibm.com/blogs/socialbusiness](http://ibm.com/blogs/socialbusiness)
  - Read and engage with our bloggers
- **Follow us on Twitter**
  - @IBMConnect and @IBMSocialBiz
- **LinkedIn** <http://bit.ly/SBComm>
  - Participate in the IBM Social Business group on LinkedIn
- **Facebook** <https://www.facebook.com/IBMConnected>
  - Like IBM Social Business on Facebook



ConnectED2015

## Notices and Disclaimers

Copyright © 2015 by International Business Machines Corporation (IBM). No part of this document may be reproduced or transmitted in any form without written permission from IBM.

**U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.**

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY. IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

**Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, IBM.com, BrassRing®, Connections™, Domino®, Global Business Services®, Global Technology Services®, SmartCloud®, Social Business®, Kenexa®, Notes®, PartnerWorld®, Prove It®, PureSystems®, Sametime®, Versa™, Watson™, WebSphere®, Worklight®, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

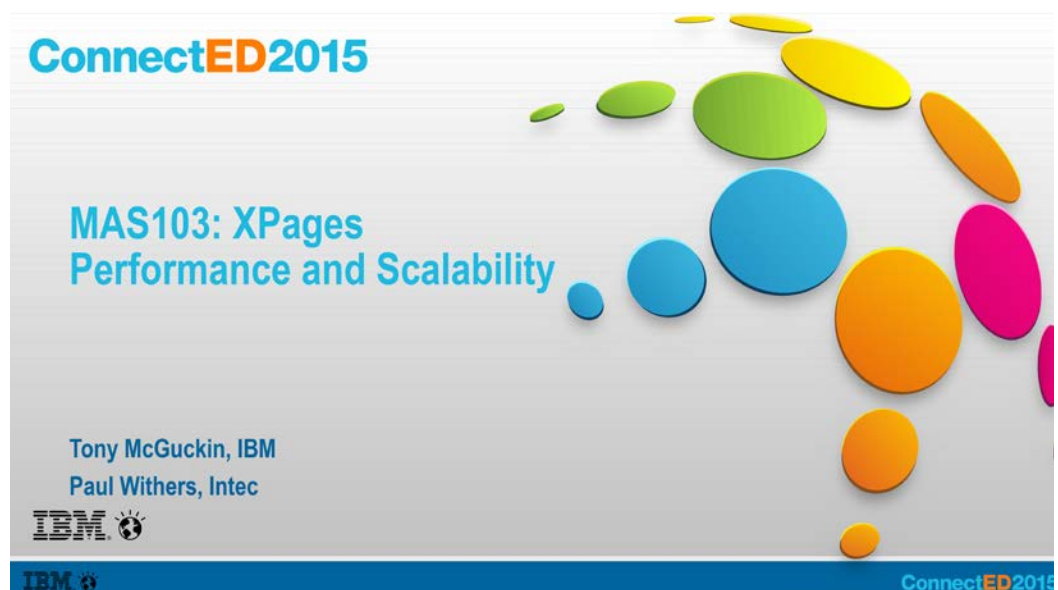
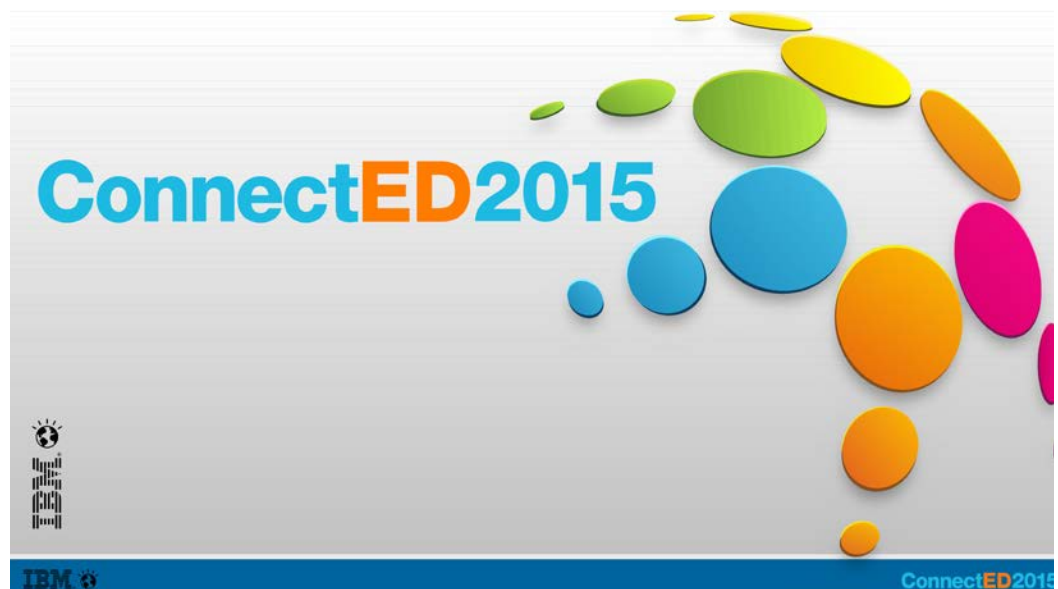


ConnectED2015

# XPages Performance and Scalability

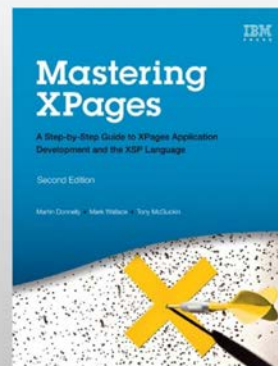
*Tony McGuckin, IBM*

*Paul Withers, Intec*





## Reference material...



This session draws upon material taken from the *Mastering XPages, Second Edition* book from **IBM Press**.

Where applicable the presenters will suggest further reading available from this book about topics in this session.

It is available to order from **Amazon.com** and all good bookshops!

IBM

ConnectED2015

## Tony McGuckin

- Senior Software Engineer / Technical Lead, IBM
  - XPages Core Runtime
  - XPages Bluemix Buildpack / Runtime
- Author Mastering XPages (1<sup>st</sup> and 2<sup>nd</sup> Editions) and XPages Portable Command Guide books
- OpenNTF Contributor



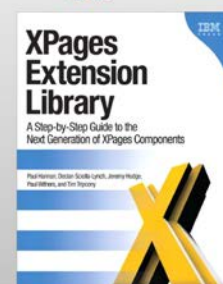
IBM

ConnectED2015

## Paul Withers

- XPages Developer since 2009
- IBM Champion
- OpenNTF Board Member
- Author of XPages Extension Library
- Developer XPages OpenLog Logger
- Co-Developer of OpenNTF Domino API

IBM CHAMPION



IBM

ConnectED2015



# Coding for Performance

IBM

ConnectED2015

## Architect for Performance

- Avoid large pages
  - “Reports” of large numbers of documents
  - Multiple on-the-fly dashboards
  - Data pulled from large variety of sources
  - Less server-side processing, less HTML to push down network
- Design for the web not for printing!
- Understand performance of code
  - You’ve been using Agent Profiling, right?
- μηδέν άγαν
  - Don’t over-complicate a small, simple page!

IBM

ConnectED2015

## Cache for Performance

- Cache data in application / session / view / request scopes as appropriate
  - XSnippet for caching header resources (SSJS / Java)  
<http://openntf.org/XSnippets.nsf/snippet.xsp?id=set-cache-headers-on-an-xpage>
  - XSnippet for Keyword caching bean (Java) <http://openntf.org/XSnippets.nsf/snippet.xsp?id=keyword-caching-bean>
  - XSnippet for refreshing individual scoped variables (Java)  
<http://openntf.org/XSnippets.nsf/snippet.xsp?id=refresh-applicationscope-variables-with-individual-timeout>
  - Add code to keyword / config document save to update scopes

IBM

ConnectED2015

## Languages for Performance

- Hierarchy of Languages
  - Literal content > Expression Language > SSJS
  - SSJS is stored as string, parsed and evaluated at runtime
  - “Custom” allows combining all three in one component’s value
    - Less HTML emitted
    - Not picked up by in-built localization
- Some SSJS code can be converted to Expression Language
  - E.g. `#database.title` instead of `#javascript:database.getTitle()`
- Java?
  - `#myBean.myProperty` = Expression Language
  - `#javascript:myBean.runMyMethod()` = SSJS, but just one line

## Compute for Performance

- Compute On Page Load where possible
  - `$javascript:session.getEffectiveUserName()` because it *cannot* change during page’s life
- Lazy Load
  - Easier with Java Beans
    - Can be triggered from Expression Language
  - Possible with SSJS
- Compute using **`view.isRenderingPhase()`** where appropriate
  - Code will only calculate **once** during partial refresh
  - Good for properties only affecting HTML output – rendered, style, styleClass, value property of Computed Field components

## Load for Performance

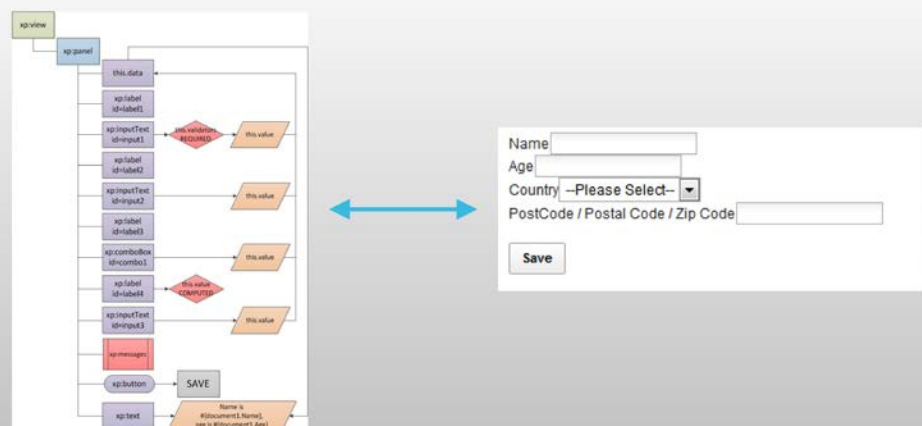
- Use dataContexts for duplicate properties, especially “rendered”
  - E.g. `#userIsAdmin` (Expression Language) pointing to dataContext computing **on page load** whether user has admin role
    - dataContexts set to compute dynamically refresh a number of times
    - If bound to panels, get cleaned up during Invoke Application phase <http://avatar.red-pill.mobi/tim/blog.nsf/d6plinks/TTRY-9CD3JN>
    - Use **if (`view.isRenderingPhase()`)** to avoid errors and only run during Render Response
- Use Dynamic Content control
  - Only the live “rendered” area is in the Component Tree

## Scope for Performance

- Scope data to the lowest scope
  - If used on one page only, viewScope
  - If used on multiple pages for one user, sessionScope
- Scope data to smallest component
  - dataContexts allow variables to be "scoped" to XPages, Custom Controls or Panels
  - Same applies to dominoView / dominoData / dataObjects variables

## Refresh for Performance

## Component Tree?!



## Partial Refresh – Step 1: Submission

- Browser → Server
- In-built Partial Refresh
  - Full web page submitted
- XSP.partialRefreshGet()
  - Nothing submitted
  - Server-side processing ignores any user action since last refresh
- XSP.partialRefreshPost()
  - By default, full web page submitted
  - Can override with options {clearForm: true, additionalFields: ['#{inputText1}', '#{inputText2}']}
  - Only send inputText1 and inputText2 content

IBM

ConnectED2015

## Partial Refresh – Step 1: Submission

- Combine optimised XSP.partialRefreshPost and in-built Partial Refresh
- See XSnippet from Per Henrik Lausten extending work by Sven Hasselbach
  - <http://openntf.org/XSnippets.nsf/snippet.xsp?id=optimized-partial-refresh-with-support-for-server-side-actions>

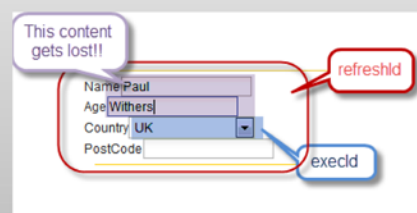
```
<xp:button id="button1">
  <xp:eventHandler event="onclick" submit="false" id="submitEventHandler" refreshMode="partial" refreshId="somePart">
    <xp:this.action><![CDATA[#{javascript:someServerSideAction();}]]></xp:this.action>
    <xp:this.script>
      <![CDATA[
        XSP.partialRefreshPost(
          #{id:gridSection},{
            clearForm: true,
            additionalFields: [#{id:inputText01},
                              #{id:inputText02} ],
            submitId: '#{id:submitEventHandler}'
          }
        );
      ]]]>
    </xp:this.script>
  </xp:eventHandler>
</xp:button>
```

IBM

ConnectED2015

## Partial Refresh – Step 2: Execution (High Level)

- Server → Server
- By default full component tree is processed
- execMode="partial" execId="processId"
  - Only the component with that ID and all descendants are processed
  - User content outside that execId is ignored



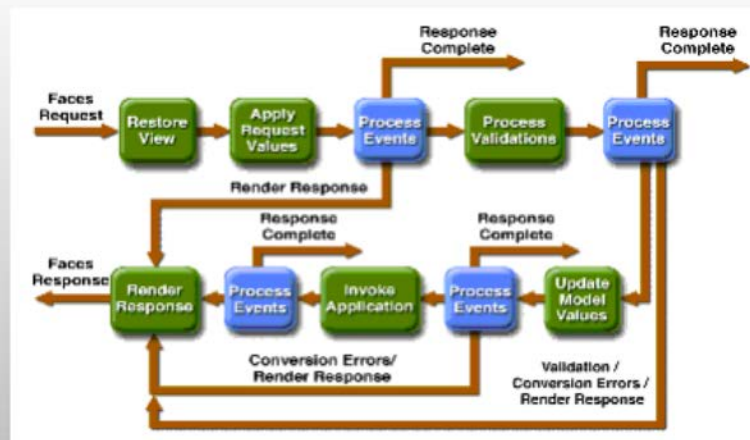
IBM

ConnectED2015

## Partial Refresh – Step 3: Refresh

- Server → Browser
- refreshMode="partial" refreshId="processId"
- Only HTML for component with that ID and all descendants is passed to browser

## XPages Lifecycle



## Six Phases

- **Restore View**
  - Component Tree retrieved
- **Apply Request Values**
  - this.setSubmittedValue(stringFromBrowser) runs on all or excld
  - Any event logic runs if immediate="true"
- **Process Validation**
  - Any converters applied for all or excld
  - Any validators applied for all or excld



## Six Phases

- **Update Model Values**
  - `this.setValue(this.setSubmittedValue())` runs on all or `execld`
  - `this.setSubmittedValue(null)` runs on all or `execld`
  - e.g. `dominoDocument.replaceItemValue(this.getValue())`
- **Invoke Application**
  - Any event logic run
- **Render Response**
  - HTML rendered and state saved
  - Only event that runs during page load

## Debugging XPages Lifecycle

- Always, always, **ALWAYS** include a Display Errors control in refresh area
  - Otherwise validation / conversion errors will not be displayed on browser
- Doc Brown: "Marty, it's perfect, you're just not thinking fourth dimensionally"
  - When does server-side code run?
  - What HTML is passed to the browser?
- Use DebugBean XSnippet to inspect phases
  - <http://openntf.org/XSnippets.nsf/snippet.xsp?id=xrpl-isphase-api>

## XSP.partialRefreshGet()

- No updates to component tree
  - `beforeRestoreView`
  - `afterRestoreView`
  - `beforeRenderResponse`
  - `afterRenderResponse`

## Basic Partial Refresh

- **Validation fails**
  - beforeRestoreView
  - afterRestoreView
  - beforeApplyRequestValues
  - afterApplyRequestValues
  - beforeProcessValidations
  - beforeRenderResponse
- Event logic does not run

## No Validation

- Conversion still honoured
  - If failed, skips from ProcessValidations to RenderResponse
  - Event Logic not run
- If no conversion errors, all phases run
- Values passed from submittedValue to value
- Component tree updated

## Immediate="true"

- Same as basic XPage processing with validation failure
  - beforeRestoreView
  - afterRestoreView
  - beforeApplyRequestValues
  - afterApplyRequestValues
  - beforeRenderResponse
  - afterRenderResponse
- Event logic run in ApplyRequestValues phase
- Component value never goes past submittedValue
- Data model not updated

# Developing for Performance

IBM

ConnectED2015

## Developing for Performance

The XPages Toolbox

- The “**XPages Toolbox**” → XPages based Application (*The “XPages Swiss Army Knife”*)
  - Runs on the Domino server or the Notes client
  - **XPagesToolbox.nsf** needs to be installed on the Domino server or XPINC client
  - A **profiler.jar** file needs to be added to the *JVM launch options* & *JVM java.policy* updated
- Should be used regularly during development / testing cycles to:
  - Profile CPU performance & Memory usage (per request or periodically) / Backend usage
  - Create Java Heap Dumps / XML Memory Dumps
  - Production use only for problem resolution - **sensitive data collection** capabilities
- Available from OpenNTF.org
  - Free open source project / Search for “XPages Toolbox”
  - Full *readme.pdf* instructions within the project download files

IBM

ConnectED2015

## Developing for Performance

Using the XPages Toolbox

- Provides insight into custom **SSJS** code using **Profile Blocks**

```
__profile("blockIdentifier", "optionalInformation"){

    // profile my custom code...
    var nd:NotesDocument = document1.getDocument();
    ....

    __profile("blockIdentifier", "optionalInformation"){

        // profile my nested profile block...
        var x = nd.getItemValueString("x");
        ....

    }

}
```

IBM

ConnectED2015

## Developing for Performance

Using the XPages Toolbox

- Provides insight into custom **Java** code using **Profile Block Decorator**

```
private static final ProfilerType pt = new ProfilerType("MyBlock");

public void myMethod() {
    if(Profiler.isEnabled()) {
        ProfilerAggregator pa = Profiler.startProfileBlock(pt, null);
        long startTime = Profiler.getCurrentTime();
        try { _myMethod(); } finally {
            Profiler.endProfileBlock(pa, startTime);
        }
    } else { _myMethod(); }
}

private void _myMethod() { // real implementation of myMethod ... }
```

IBM

ConnectED2015

## Developing for Performance

Using the XPages Toolbox

- Profile Blocks in SSJS / Java are **Non-Invasive** and **Supportive**
  - Leave the custom Profile Blocks in your SSJS / Java Code
- No **negative** performance impact on any application even if the XPages Toolbox is not installed on a server or XPiNC client
- Therefore supporting you for future **profiling & maintenance** tasks

IBM

ConnectED2015

## Developing for Performance

The XPages Toolbox



Demo:

**Profile XPages Request using Wall and CPU Profilers...**  
**Perform CPU and Wall time intensive tasks...**  
**Analyze profiling results and identify issues in the XPages Toolbox!**

XPages Toolbox									
<div> <a href="#">Home</a> <a href="#">CPU Profiler</a> <a href="#">Backend Profiler</a> <a href="#">Runtime Monitoring</a> <a href="#">Session Dumps</a> <a href="#">Periodic Snapshots</a> <a href="#">Threads</a> <a href="#">Logging</a> </div>									
<div> <a href="#">Start CPU Time Profiler</a> <a href="#">Start Wall Time Profiler</a> <a href="#">Stop Profiler</a> <a href="#">Reset Profiler</a> <a href="#">Save Snapshot</a> <a href="#">Delete Snapshots</a> </div>									
<div> <b>Hierarchical All</b>  <b>Hierarchical Dominators</b>  <b>By Count</b>  <b>By Total Time</b>  <b>By Specific Time</b>  <b>By Average Time</b> </div>									
<div> Show: 10   25   50   100 entries Expanded All   Collapse All  Snapshot Type Content Count Total time Max time Avg time Min time Specific time  Thu Nov 14 10:38:24 GMT 2013 </div>									
<div> XPages Request http://localhost:8080/MyApp/MyApp.xsp 1 9028 9028 9028 9028 0  XSP Phases RESTORE_VIEW 1 16 16 16 16 0  XSP Restore view RestCPUWallTime.xsp 1 16 16 16 16 0  XSP Restore view state RestCPUWallTime 1 16 16 16 16 16  XSP Phases INVOKES_APPLICATION 1 9812 9812 9812 9812 0  JavaCodeExecution println("Performing CPU 1 9812 9812 9812 9812 9812 </div>									
<div> Resources  Copyright 2012 </div>									

IBM

ConnectED2015

# Persist for Scalability

IBM

ConnectED2015

## Persistence Summary

- HTTP is **stateless**, XPages is **stateful** by default
  - `xsp.session.transient=true` makes XPage stateless
- For each browser session, for every XPage component trees need storing
- `xsp.persistence.tree.maxviews=4` – default number of trees stored in memory
- `xsp.persistence.file.maxviews=16` – default number of trees stored on disk
- `xsp.session.timeout=30` – default inactivity timeout for **browser stateful session**
- (`xsp.application.timeout=30` – default applicationScope timeout)
- **Recommendation:** reduce timeout and use Keep Session Alive

IBM

ConnectED2015

## Persistence Summary

- `xsp.persistence.mode=basic`
  - Keep pages in memory: best for quick retrieval, few browser sessions
- `xsp.persistence.mode=file`
  - Keep pages on disk: best for lots of browser sessions, slower retrieval
- `xsp.persistence.mode=fileex`
  - Keep latest page in memory, rest on disk: best for lots of users, quick retrieval of latest page

IBM

ConnectED2015



## Persistence Summary

- `xsp.persistence.file.gzip`
  - Before storing to disk, gzip the content: smaller files, but slower to store / retrieve
- `xsp.persistence.dir.xspstate`
  - Default folder for storing pages to disk
  - Similar options for default location for file uploads and attachments

## Page Persistence

- `xsp.persistence.viewstate=fulltree`
  - Default, whole tree is persisted and update
- `xsp.persistence.viewstate=delta`
  - Only stores changes since page first loaded
  - Valid if pages are stored in memory
- `xsp.persistence.viewstate=deltaex`
  - Stores full component tree for current page, deltas for others
  - Valid if pages are only stored in memory

## Page Persistence

- `xsp.persistence.viewstate=nostate`
  - No component tree is stored, similar to `xsp.session.transient=true`
- Page persistence settings can be set on individual XPages
  - Great for large, readonly XPages with no paging, state not persisted
  - Go to next page from **default**, not go to next page from **current**
  - Toggle show detail from **default**, not toggle show detail from **current**
    - Unless `showDetailsOnClient=true`
  - **Recommendation:** set `xsp.persistence.viewstate=nostate` on XAgents

# Developing for Scalability

IBM

ConnectED2015

## Developing for Scalability

Surface Zero - Analysing Memory Usage

**CPU**, **Wall Time**, and **Backend** profiling gives insight into the *vertical* processing costs for any given XPage request

But what about *vertical* and *horizontal* memory costs?

- Requires analysis using JVM Heap Dumps
- Requires analysis using JVM System Dumps
- Requires analysis using XML Session Dumps

IBM

ConnectED2015

## Developing for Scalability

Analysing Memory Usage – JVM Heap & System Dumps

- Generate a **JVM Heap** or **System Dump** of the HTTP task JVM memory
  - A button in the XPages Toolbox generates a JVM Heap Dump
  - XSP Commands on the Domino Server console allow spontaneous Heap / System Dump creation
    - `tell http xsp heapdump` (triggers `com.ibm.jvm.Dump.HeapDump()`)
    - `tell http xsp systemdump` (triggers `com.ibm.jvm.Dump.SystemDump()`)
- Analyze Heap / System Dumps using the **Eclipse Memory Analyzer**
  - <http://www.eclipse.org/mat/>
  - Also install Eclipse Extension: **IBM Diagnostic Tool Framework for Java Version 1.1**  
<http://www.ibm.com/developerworks/java/jdk/tools/dfj.html>
- Heap Dumps automatically occur in production when certain conditions occur
  - Eg: By default when an XPage fails due to the server running out of JVM Memory
    - `java.lang.OutOfMemoryError`

IBM

ConnectED2015

## Developing for Scalability

Analysing Memory Usage – XML Session Dumps

- Generate **XML Session Dumps** of the HTTP task JVM memory
  - Two options available under the **XPages Toolbox** → **Session Dumps Tab**
    - ✓ **Full XML Session Dump**
    - ✓ **Partial XML Session Dump**
- Analyze **XML Session Dumps** using any **Browser/XML/Text Editor**
  - **Caution required on production systems as sensitive data is collected within the XML Session Dump file**

IBM

ConnectED2015

## Developing for Scalability

Heap/System/XML Session Dumps

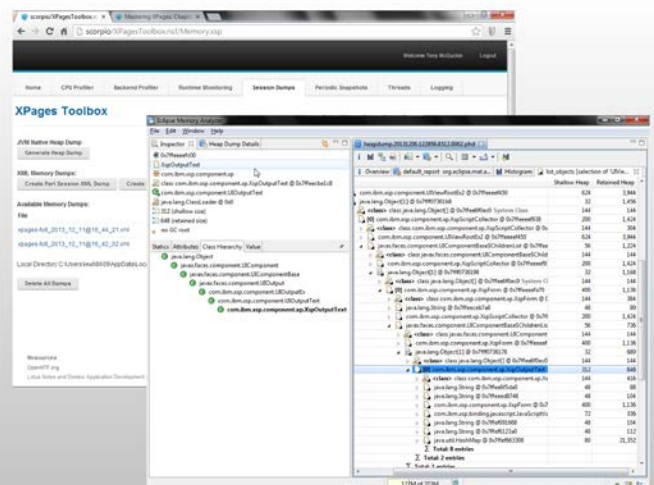


Key elements:

Make XPages Requests...

Generate Heap/System/XML Session Dumps...

Analyze memory usage in each type of Dump format to identify issues!

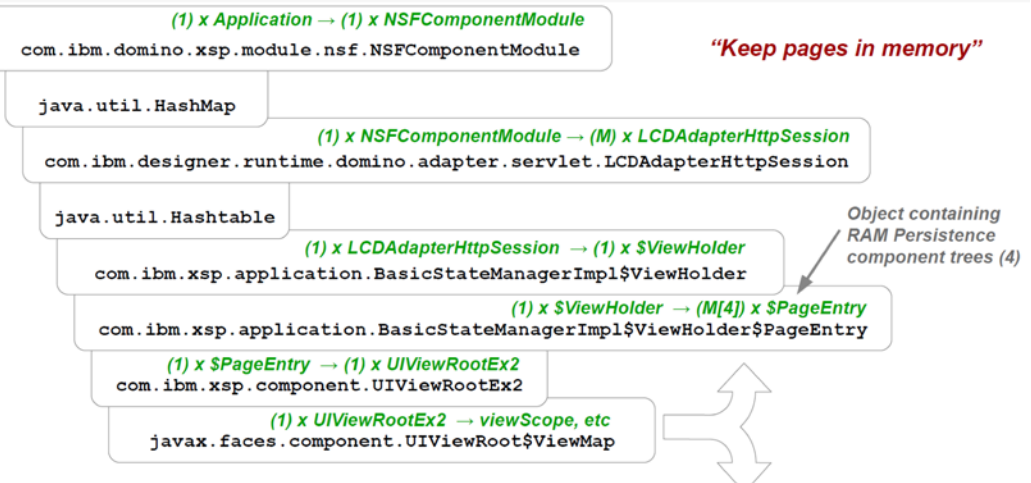


IBM

ConnectED2015

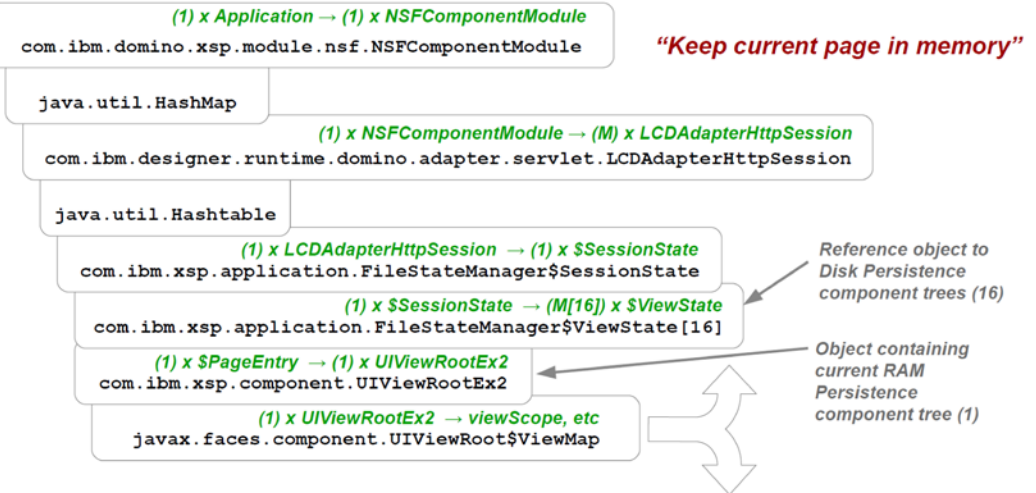
## Developing for Scalability

Heap/System Dumps – QQL / Navigating XSP Objects Entry Point...



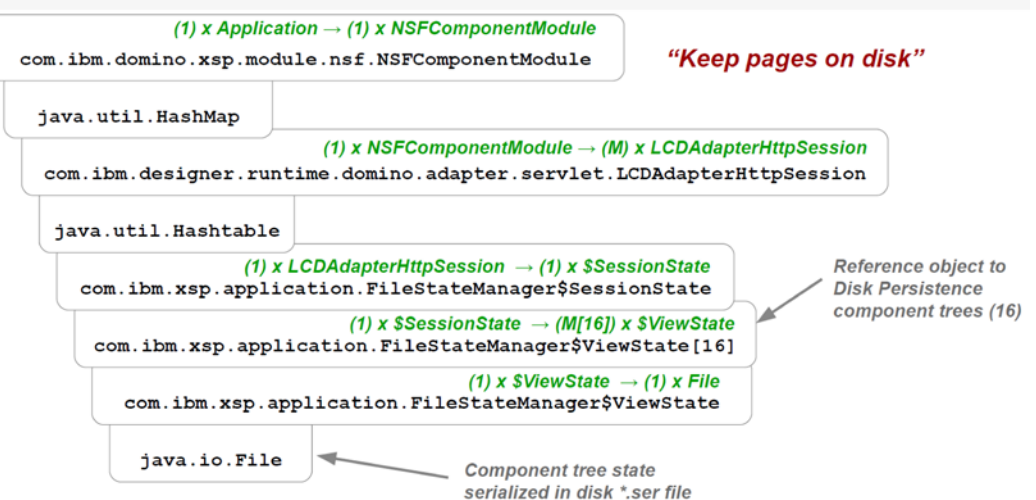
## Developing for Scalability

Heap/System Dumps – OQL / Navigating XSP Objects Entry Point...















## Developing for Scalability

Heap/System Dumps – OQL / Navigating XSP Objects Entry Point...





## Other Great XPages Sessions On The Way

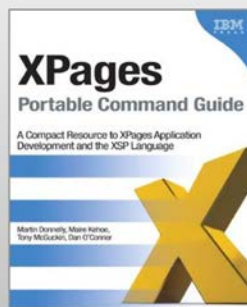
 -CHALK202: App Dev Trends and Directions: A Conversation with the Inventors of XPages.	 -BP107: Ten Lines Or Less: Interesting Things You Can Do in Java With Minimal Code
 -BP108: Be Open - Use WebServices And REST in XPages Applications	 -BTE102: The Future of Web Development - Write Once, Run Everywhere with AngularJS and Domino
 -BP106: From XPages Hero To OSGi Guru: Taking The Scary Out Of Building Extension Libraries	 -MAS103: XPages Performance and Scalability
 -AD101: IBM Domino App Dev Futures	 -CHALK405: XPages and Java: Share your Experience
 -CHALK201: IBM Domino Application Development	 -BP105: Take Your XPages Development to the Next Level
 -CHALK102: OpenNTF Domino API: The Community API	 -AD302: Responsive Application Development for XPages

IBM

ConnectED2015

## Technical Education

- IBM Press Books and eBooks
  - Three best-selling publications



IBM

49

ConnectED2015

## More Information – Summary

- OpenNTF – Open Source Community  
<http://www.openntf.org>
- XPages.info – One Stop Shopping for XPages  
<http://xpages.info>
- XPages Forum – Got Questions, Need Answers?  
<http://xpages.info/forum>
- Domino Application Development Wiki  
<http://www.lotus.com/idd/ddwiki.nsf>
- Stackoverflow  
<http://stackoverflow.com/questions/tagged/xpages>



IBM

50

ConnectED2015



## Engage Online

- **SocialBiz User Group:** [socialbizug.org](http://socialbizug.org)
  - Join the epicenter of Notes and Collaboration user groups
- **Social Business Insights blog:** [ibm.com/blogs/socialbusiness](http://ibm.com/blogs/socialbusiness)
  - Read and engage with our bloggers
- **Follow us on Twitter**
  - @IBMConnect and @IBMSocialBiz
- **LinkedIn:** <http://bit.ly/SBComm>
  - Participate in the IBM Social Business group on LinkedIn
- **Facebook:** <https://www.facebook.com/IBMConnected>
  - Like IBM Social Business on Facebook

IBM

ConnectED2015

## Notices and Disclaimers

Copyright © 2015 by International Business Machines Corporation (IBM). No part of this document may be reproduced or transmitted in any form without written permission from IBM.

**U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.**

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY. IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

**Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com, BrassRing®, Connections™, Domino®, Global Business Services®, Global Technology Services®, SmartCloud®, Social Business®, Kenexa®, Notes®, PartnerWorld®, Prove It!®, PureSystems®, Sametime®, Verse™, Watson™, WebSphere®, Worklight®, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

IBM

ConnectED2015

# Taking XPages Applications from Out-of-the-Box to Outstanding



*Brad Balassaitis*

*Kathy Brown*

The User Interface (UI) is a critical component for modernizing an application. Users won't use your application if it doesn't provide a modern look-and-feel similar to the other

consumer applications that they use frequently. Internal users may not be given a choice, but user adoption and application success is heavily influenced by the application interface.

Out of the box, you can create professional-looking applications with OneUI, but your apps will look like every other app created with OneUI. Many developers have turned to using other frameworks like Twitter Bootstrap and KendoUI to create more modern-looking apps, but just like with OneUI, those apps look like every other app created with the entry-level examples of those frameworks.

How can you modernize your application and still give it the WOW factor, without spending hundreds of hours just on UI (with no guarantee that it will look good)? UI themes. With predefined CSS and plugins included, a theme can transform your application. UI themes are typically responsive, so your XPages applications will look great on the desktop, mobile devices, and tablets.

## UI Frameworks and Themes

Let's start by talking about the differences between UI frameworks and themes.

A **UI framework** is a collection of components with a pre-defined look-and-feel. Some well-known JavaScript-based examples include Twitter Bootstrap, KendoUI, Sencha Ext JS, and Foundation. Major JavaScript libraries such as jQuery and Dojo also include many UI components.

These frameworks are a good starting point for building an interface because good programmers are not necessarily good UI designers. Often, the focus is put on the application logic, and the UI doesn't get the same level of attention. Instead of

enhancing the application, it detracts from it. We shoot for web 2.0 but end up with web two-point-ewww.

UI frameworks help by providing a well-designed, standardized look and feel with predefined CSS styling for framework components. You can even customize the look and feel by overriding the default styling as needed. This is a big leap forward from creating your own styling completely from scratch.

This may sound like what you think of as a **theme**, but there is a distinct difference – themes are a fleshed-out UI implementation, containing styling (CSS) and layout for the entire page's look and feel. They can also include additional resources (such as plugins) that are part of the interface. The resources all work together and look good together. A theme can be built on an existing framework or it could be created by a UI designer from scratch.

You could think of a framework as the materials for building a house and the theme as the blueprint.

## OneUI

If you've developed XPages applications, then you're most likely familiar with OneUI – a standardized web application UI that was developed by IBM and used on many of their sites.

OneUI is an example of a UI theme that's built into the XPages platform. It includes a number of stylesheets that define the look and feel and uses many Dojo widgets for UI components. It's designed to work with the Application Layout control to make it very easy for you to easily implement an application-wide page structure.

Figure 1 shows what a form looks like with the application layout control but no theme enabled in the application. Figures 2 and 3 show the same page with OneUI v2.1 and OneUI v3.2 enabled, respectively.

- Tab 1
- Tab 2
- Tab 3

Save

Field 1		Field 6	
Field 2		Field 7	
Field 3		Field 8	
Field 4		Field 9	
Field 5		Field 10	

Figure 1: XPage Form Table with Application Layout and No Theme

Figure 2: XPage Form Table with Application Layout and OneUI v2.1

Figure 3: XPage Form Table with Application Layout and OneUI v3.2

## Advantages of OneUI

- If you don't have a UI already designed or the time (or skills) for design, OneUI makes it easy to implement a consistent, application-wide look and feel.
- Assuming you are using the same version of OneUI as IBM, your application will also look consistent with other IBM products such as IBM Connections.
- There are numerous simple color schemes readily available because they come installed on the Domino server.
- It's designed to handle browser inconsistencies; if you look at the source of a OneUI theme file, you'll see many conditional references to browser-specific stylesheets.
- The application layout control is designed to work with OneUI, and it provides property panel options to easily add or remove key layout regions around the main content area (banner, footer, left column, right column, etc.) and add components to each region.
- It is designed to handle right-to-left screen layout as well as left-to-right.
- Data Views are feature-rich data display controls that are designed to have a better look with OneUI on a desktop as well as on a mobile device.
- Form Tables are designed to work with OneUI styling, so they allow you to easily set up a form layout structure with consistent styling.

## Disadvantages of OneUI

- For several years now, it has not been a modern-looking interface; it just does not compare to readily-available current frameworks.
- The layout is not responsive, so it does not adapt well to different screen sizes.
- The forms, in particular, do not look appealing.
- The form layout is table-based and adjusts to some extent based on screen size, but the columns often end up different widths within a given section and are very hard to keep in line across different sections if you break the form up into several tables based on the layout needs of different sections.
- Customization can be difficult beyond the basics of color and logo.
- Most OneUI applications look the same.



There are clear advantages to having a UI theme, but we need something that's more modern-looking and also provides a responsive layout.

Enter Twitter Bootstrap.

## Bootstrap

There are many options available, but we're going to focus on Twitter Bootstrap because it is by far the most popular UI framework, having been used to build millions of websites. It is the most popular project on GitHub; its star count is more than double the next-highest project. Its popularity has allowed it to mature very quickly and means that there are many resources available to get started and to get help with any problems you face because others have undoubtedly faced them before.

### Advantages of Bootstrap

- The layout is a fluid grid structure, which is far more flexible than a table-based layout structure.
- Bootstrap was designed to be responsive. Its mobile-first design provides the capability to build applications where one implementation can work well on any device size.
- It is based on jQuery, the most popular JavaScript library in the world.
- The styling is well-designed, with built-in typography and many UI controls including navigation bars, buttons, toggles, badges, dialogs, and much, much more.
- A flexible icon font (Glyphicons) is built-in to Bootstrap.

All of these things are already designed to handle browser inconsistencies, so they're widely usable. Twitter Bootstrap is also easy to get up-to-speed on.

## Bootstrap Themes

Bootstrap is an extremely strong foundation on which to build an application interface. But the challenge is getting beyond the default layout that most developers use when first getting started; for a time, there were many, many websites that implemented Bootstrap and all looked virtually the same because they all had the black top navigation bar as shown in Figure 4.

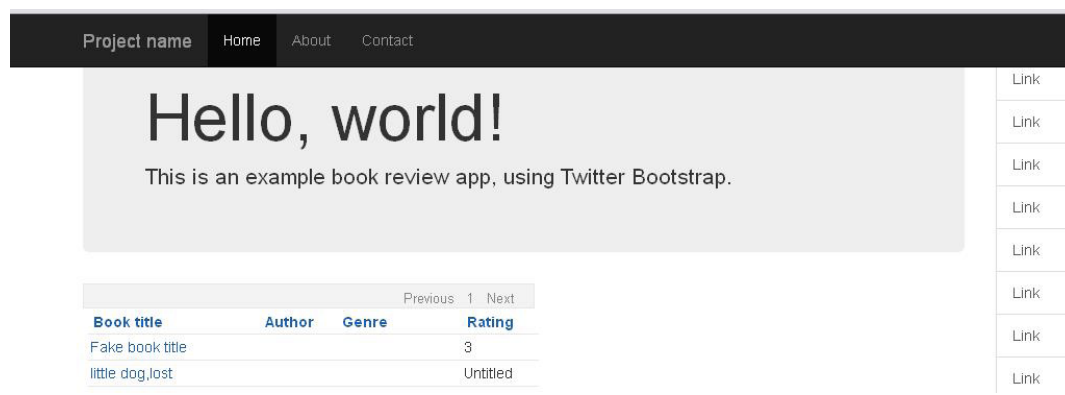


Figure 4: Basic Bootstrap

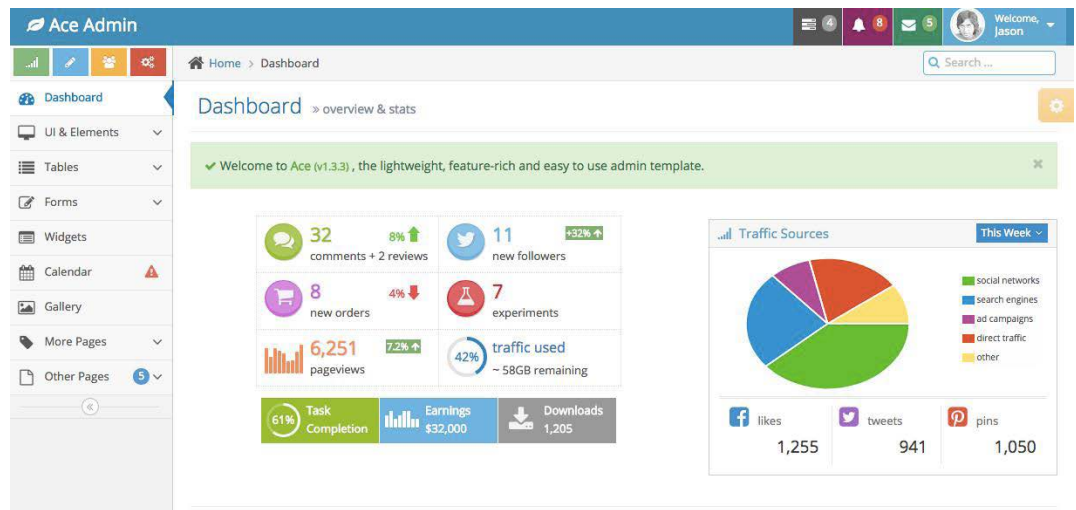
A framework is a great start, but it's not a finished UI design. Since programming and UI design are very different skillsets, many developers are not also good UI designers. You can easily spend many, many hours on UI design and still have it not look very good if you're doing it from scratch.

If you don't have a corporate standard UI design or a good designer available, a great solution is a Bootstrap theme.

Themes provide professionally-designed full-page layouts. They include the CSS for the look and feel and are often designed to use Bootstrap's responsive UI features, so a single UI can work well on any screen or device.

They provide a consistent user experience from page-to-page and often include plugins for additional features. They generally provide sample pages for you to use as a model. Without using a Bootstrap theme, you could use Bootstrap's styling and add your own plugins, however you will have to load them individually and ensure they work well together and don't conflict. A Bootstrap theme will have done that work for you.

There are hundreds of free and paid themes available to transform your XPages application. Because it's Bootstrap, it's responsive, so those XPages applications will look great on the desktop, mobile devices, and tablets as you can see in Figure 5.



## Bootstrap-Related UI Features Available in (or Coming Soon to) XPages

If you've heard of Bootstrap4XPages or attended any of the ConnectED sessions by IBM's XPages development team, you may be wondering how all of these fit in with Bootstrap features that are available in XPages.

Bootstrap4XPages is an OpenNTF project that was added to the Extension Library in November 2014.

It adds Bootstrap resources (CSS, jQuery, icon fonts) to the application and uses custom renderers to modify XPages applications using the Application Layout control to render Bootstrap elements. It makes it very easy to add Bootstrap to a OneUI application and render the application.

There are enhancements coming soon, as mentioned in Brian Gleeson's session at ConnectED on Responsive Application Development for XPages:

- **Navbar control:** creates a Bootstrap navigation bar and has properties to define the header, links, and content and choose whether it's fixed or inverted
- **Dashboard control:** lets you easily display and format glyphs and label text with badges
- **Carousel:** displays a slideshow of images and contains properties to customize it

The main difference between Bootstrap4XPages and a custom theme is that a theme is a fully-designed page layout. Although Bootstrap4XPages makes adding Bootstrap even simpler and has a growing library of controls, you still need to

design your UI with Bootstrap4XPages. Your source code will still use the OneUI class and style names, which can be confusing and difficult to customize.

## Up Next: Implementing a Professionally-Designed UI Theme

In this part of Taking XPages Applications from Out-of-the-Box to Outstanding, we've covered the difference between a framework and a professionally-designed theme. We've also discussed why you may want to use a theme. In the next part of this article, we will discuss how to implement a Bootstrap theme in your application. There is more involved than just picking a theme and installing it. We'll cover tips and tricks we learned to integrate a theme with your application. ●

---

Brad Balassaitis is an IBM Champion for Collaboration Solutions (2014 and 2015) and a Senior Consultant in the Collaboration practice at PSC Group. He has 18 years of experience developing innovative Notes/Domino applications, primarily working with XPages over the past several years. He has presented sessions on XPages at IBM Connect, user groups, and on webinars. He has also contributed several tutorials to NotesIn9.com along with a database with several Dojo Data Grid samples in XPages toOpenNTF. Brad blogs frequently about XPages at Xcelerant.net, and you can also find him on Twitter at @Balassaitis.

---

---

Kathy Brown is a Senior Consultant with PSC Group, LLC. Kathy is located in New Hampshire and has worked with IBM Notes since 1995. She has presented at events such as Lotusphere, THE VIEW Developer conference, and numerous user group conferences around the world. She also writes for various industry publications, including SocialBizUg.org's Developer Edition newsletter. Kathy blogs about IBM Notes (and other things) at her website [www.runningnotes.net](http://www.runningnotes.net). She is an IBM Champion, a twitter addict (15 accounts and counting), and proud to be a Nerd Girl!

---

# Improve XPages Application Performance with JSON-RPC



*Brad Balassaitis, PSC Group*

One of the best-kept secrets of XPages is JSON-RPC control. It is the ideal solution for application developers who want:

- Their XPages applications to be more efficient and run faster
- To easily implement client-side and server-side JavaScript interaction

## Historically Executing Server-Side Code from Client-Side Code

If you've ever needed to trigger server-side JavaScript (SSJS) from client-side JavaScript in XPages, especially in earlier versions, you probably used a kludgy workaround like this:

1. Add a Button control to the page with the required SSJS logic
2. Set the button to always be hidden via CSS
3. Use client-side JavaScript from any event handler to get a handle to the hidden button and trigger its click() event

Even worse, if you had to pass a value from client-side JavaScript to the server side to be available when the SSJS runs, you may have used a hacktastic solution similar to mine:

1. Add a Hidden Input control to the page and bind it to a scope variable
2. Use client-side JavaScript to put a value into the hidden field
3. Use client-side JavaScript to trigger a partial refresh on that Hidden Input control to push the value to the component tree on the server
4. Use client-side JavaScript to trigger the click() event of a hidden button control with the required SSJS logic

If you're still using methods like these to implement client-side and server-side code interaction, I have good news: There's a much better way! JSON-RPC is a much cleaner method for making applications perform more efficiently.



## JSON-RPC Explained

The Remote Service control uses JSON-RPC to allow you to set up simple, efficient interactions between client-side and server-side JavaScript.

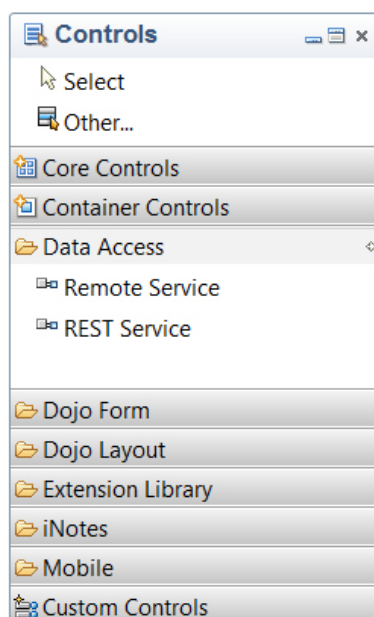
JSON-RPC is a specification that was created to be a simpler alternative to SOAP. Nearly 10 years old, it has been at version 2 since 2010, after which it was added to XPages. JSON-RPC is defined by [JSONRPC.org](http://JSONRPC.org) as a “stateless, light-weight remote procedure call (RPC) protocol.” But don’t let that seemingly simple description belie the tremendous benefits that it can provide.

A remote procedure call is a mechanism for client-side code to trigger a procedure to run on the server. JavaScript Object Notation (JSON) is the format of the communication between the client and server (see [json.org](http://json.org) for more information). Putting the two sides together, JSON-RPC provides a way for client-side code to remotely trigger server-side code and pass information back and forth in the form of JSON.

If you’d like to hear the role of a JSON-RPC client explained in detail by animated robots, by all means, [check out this YouTube video](#).

## JSON-RPC in XPages

JSON-RPC is available in XPages via the Remote Service control, which can be found in the Data Access drawer of the Controls palette, as shown in Figure 1.



**Figure 1** Use JSON-RPC via the Remote Service control in the Controls palette

When you drag the control onto an XPage or custom control and switch to the Source view, you'll see that it creates an `<xe:jsonRpcService>` tag:

```
<xe:jsonRpcService id="jsonRpcService1"></xe:jsonRpcService>
```

This control has been a part of the Extension Library since an 8.5.2 version of the library was released in December 2010. It is also part of the 8.5.3 Upgrade Pack 1 installation and is built into IBM Notes 9.x.

## Benefits of JSON-RPC

Let's look at the many benefits to using JSON-RPC via the Remote Service control.

### ***It Enables Client-Side to Server-side Code Execution and Communication***

Client-side JavaScript can be used to trigger SSJS logic to run on the server. You can pass parameters to the remote method and return a response to the client-side code.

### ***It is Asynchronous***

JSON-RPC does not block the user's interaction with the application because the remote procedure is triggered asynchronously. Therefore, the user can continue working on the page regardless of how long the remote procedure takes to complete.

You can execute a remote procedure quietly or you can attach a callback function so that you can run client-side JavaScript after the remote procedure has finished and, optionally, returned a value.

### ***It is Faster***

There is no HTTP POST or form submission when a remote procedure is triggered. JSON-RPC doesn't need information other than the method you're calling, the parameters you pass to it, and any return value that is created.

Unlike the normal XPages partial Refresh model, the response from the server does not send back any HTML output because it does not require an update to the page.

### ***It is Easy to Use***

JSON-RPC is already designed to be a simplified version of SOAP, but the Remote Service control makes it even easier! The communication between client-side and server-side JavaScript is performed behind the scenes via JSON, and you don't have to worry about how to format or handle it.

You also don't have to write extra code on the client or the server to handle the communication. The control provides the client, and the Domino server provides

the remote environment in which the methods can be executed. In addition, the control hides some of the requirements, such as the version specification and ID, keeping it very simple.

The control automatically creates a JavaScript variable that you can use to reference it from anywhere on the page.

### Important Caveat

The biggest performance advantage of JSON-RPC is also a caveat that you need to keep in mind.

Because the form is not submitted, any changes made to the form since the last submissions are not available to the SSJS logic running in the remote procedure. Also, if you make any changes to the current page in the SSJS procedure, they will be lost.

#### Note

This is true for all intents and purposes. However, it is possible to work around this caveat and save any changes made by a JSON-RPC method. This challenge and solution were documented and described brilliantly (of course) by [Tim Tripcony](#).

## Getting Started with JSON-RPC

To get started with JSON-RPC, we'll set up a remote method that puts a message into a scope variable. This is not otherwise possible from client-side JavaScript (without several extra steps), but it's a good simple example of client-side code triggering server-side code. Follow these steps:

1. Drag and drop a Remote Service control onto an XPage
2. In the Properties view, select All Properties
3. Under basics, define the serviceName. This will be the name by which you will refer to the RPC with client-side JavaScript.
4. Also under basics, click in the Value column next to methods, and click the + button to add a method. You should see `remoteMethod[0]` under methods in the properties, as shown in Figure 2.



Design
Source

Properties x
Data x
Events x
Progress x

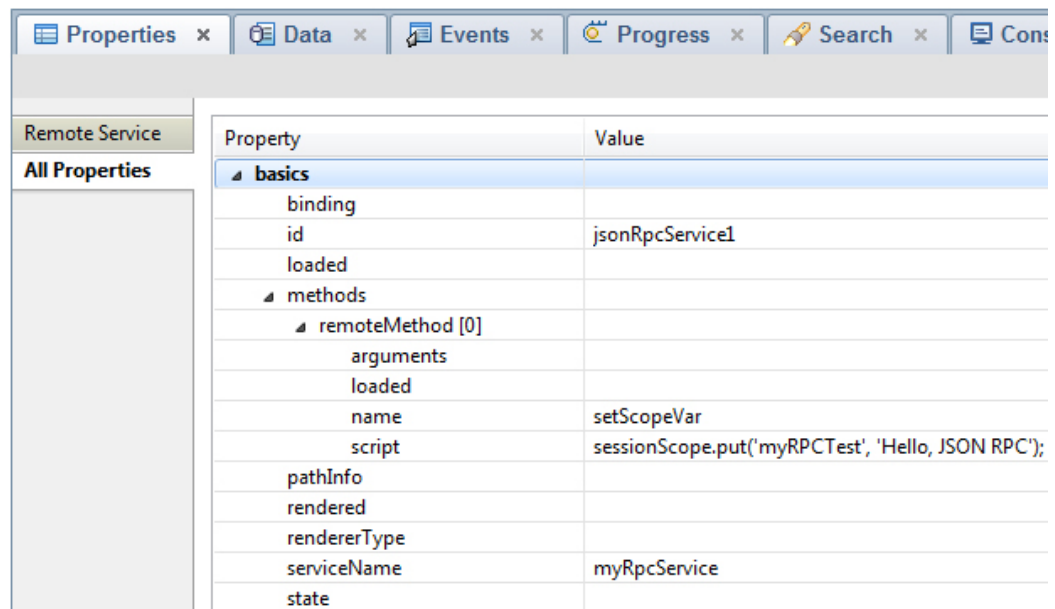
Remote Service
All Properties

Property	Value
▲ basics	
binding	
id	jsonRpcService1
loaded	
▲ methods	
▲ remoteMethod [0]	
arguments	
loaded	
name	
script	
pathInfo	
rendered	
rendererType	
serviceName	myRpcService
state	
▲ styling	
disableTheme	
themeld	

**Figure 2** Setting up a Remote Service

Next, implement the remote method as follows:

1. Give it a name and select the script property
2. Click the external property editor icon (not the blue diamond) and add a line of SSJS that sets a scope variable, as shown in Figure 3



Property	Value
binding	
id	jsonRpcService1
loaded	
methods	
remoteMethod [0]	
arguments	
loaded	
name	setScopeVar
script	sessionScope.put('myRPCTest', 'Hello, JSON RPC');
pathInfo	
rendered	
rendererType	
serviceName	myRpcService
state	

**Figure 3** Add SSJS that sets a scope variable in the remote method

Here's the source code that's generated for the control:

```
<xe:jsonRpcService id="jsonRpcService1"
  serviceName="myRpcService">
  <xe:this.methods>
    <xe:remoteMethod name="setScopeVar"
      script="sessionScope.put ('myRPCTest', 'Hello, JSON RPC');">
    </xe:remoteMethod>
  </xe:this.methods>
</xe:jsonRpcService>
```

The control creates an `<xe: jsonRpcService>` tag. The tag has an `<xe:this.methods>` tag that includes one or more `<xe:remoteMethod>` tags, each of which contains a remote procedure.

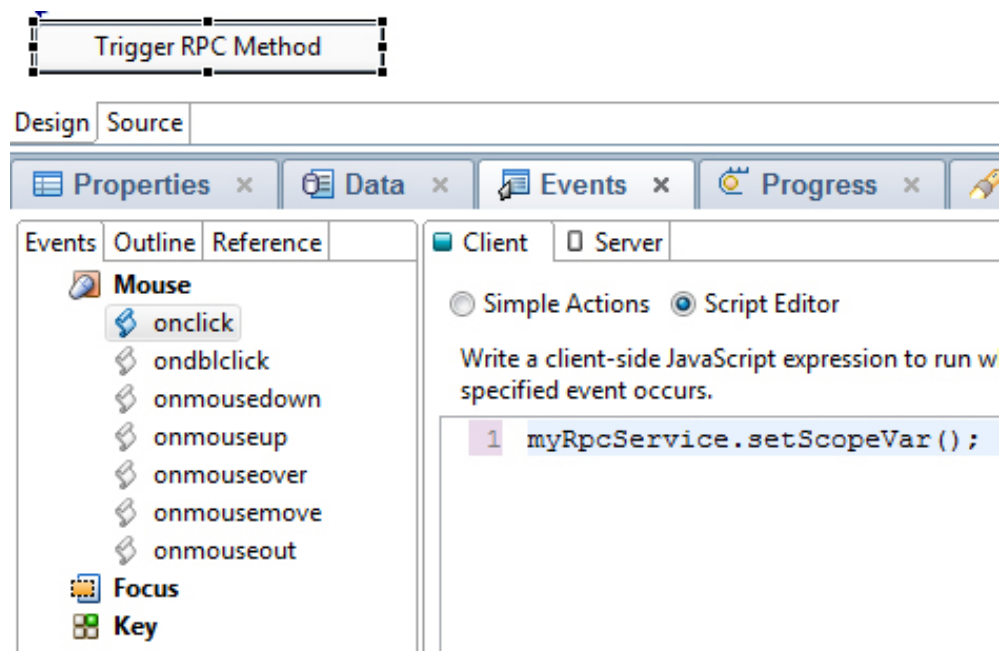
Next, we'll need to trigger the method. To call a remote method, use this general form:

**`serviceName.methodName()`**

Following are the steps to call a remote method:

1. Add a button to the page and switch to the Client tab of the Events view for the button
2. Add the following statement to trigger the method (see Figure 4):  
**`myRpcService.setScopeVar();`**





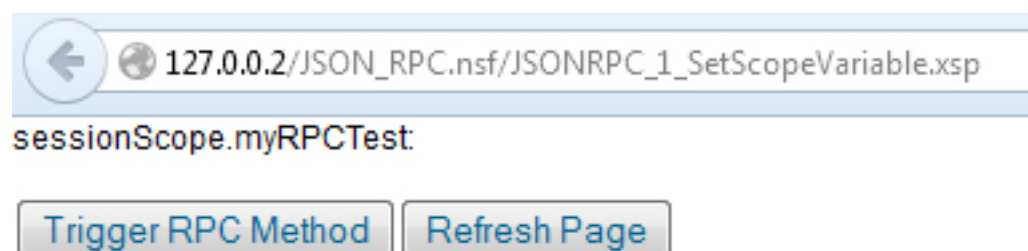
**Figure 4** Add a client-side JavaScript statement to trigger the remote method

3. In order to easily verify that the remote method works, add a Computed Value control to the page that displays the value of the scope variable.
4. Finally, since the RPC method does not refresh the page, add a second button that performs a full-page refresh when clicked.

#### Note

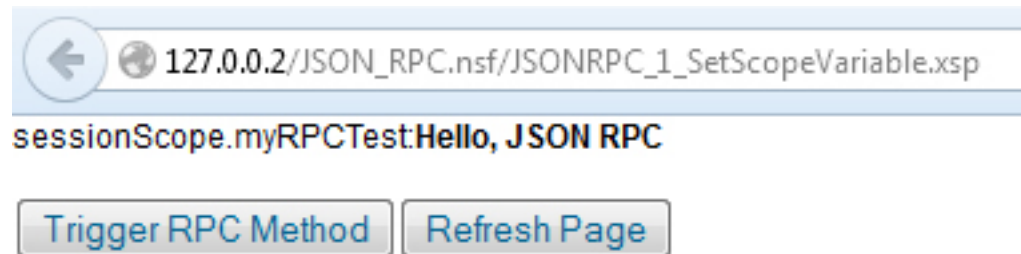
The first button could have just as easily refreshed the page, but to get the most benefit from JSON-RPC, I recommend you think about using it without refreshing the page or submitting anything.

Figure 5 shows what my test page looks like when I first open it:



**Figure 5** A test page showing the addition of the RPC method and a button to refresh the page

Once I click the Trigger RPC Method button and then the Refresh Page button, I see the scope variable that was set by the RPC method, as shown in Figure 6.



**Figure 6** Displaying the scope variable set by the RPC method shows when you click on the Trigger RPC Method and Refresh Page buttons

Following is the entire source of that test page:

```
<?xml version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core" xmlns:xe="http://
www.ibm.com/xsp/coreex">

  <xe:jsonRpcService id="jsonRpcService1" serviceName="myRpcService">
    <xe:this.methods>
      <xe:remoteMethod name="setScopeVar"
        script="sessionScope.put('myRPCTest', 'Hello, JSON RPC');">
      </xe:remoteMethod>
    </xe:this.methods>
  </xe:jsonRpcService>

  sessionScope.myRPCTest:
  <xp:text escape="true" id="computedField1" value="{sessionScope.
  myRPCTest}" style="font-weight:bold">
  </xp:text>

  <br />
  <br />

  <xp:button value="Trigger RPC Method" id="button1">
    <xp:eventHandler event="onclick" submit="false">
      <xp:this.script><![CDATAmyRpcService.setScopeVar();]</xp:this.script>
    </xp:eventHandler>
  </xp:button>
  <xp:button value="Refresh Page" id="button2">
```

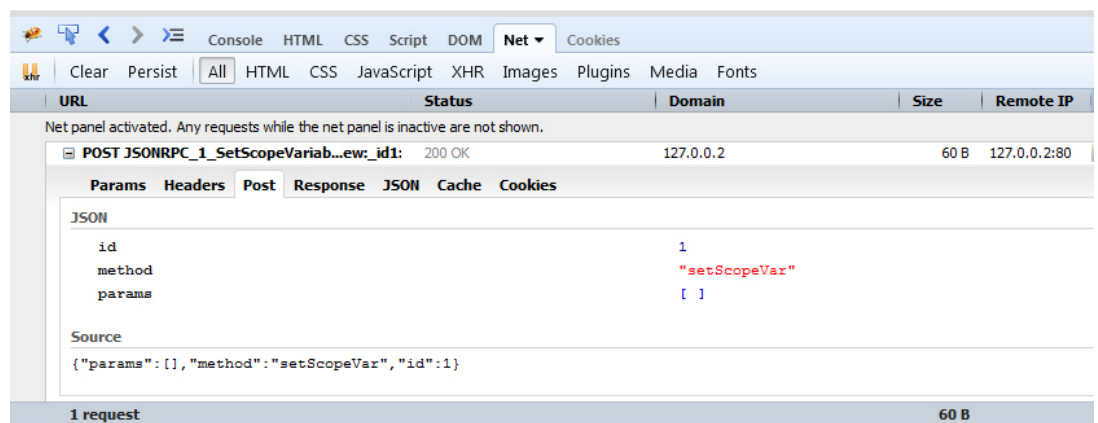
```
<xp:eventHandler event="onclick" submit="true"
  refreshMode="complete">
</xp:eventHandler>
</xp:button>

</xp:view>
```

Granted, this isn't an eye-popping example, but it shows how easy it is to remotely trigger server-side code.

## Monitoring JSON-RPC Calls

Even though triggering the remote procedure does not cause the page to be submitted, a call is still made in order to execute the method. You can see this on the Net tab of your browser developer tools as seen in Figure 7.



**Figure 7** Remote method request with no parameter

As mentioned earlier, the request is minimal — all that is sent out is an ID, a method name, and an array of parameters. (In this example, no parameters are sent along with the request, but the next example includes parameters.)

## External Property Editor vs. Blue Diamond

The difference between adding the code for a remote method via the external property editor icon and adding it by clicking the blue diamond to compute the value is subtle but very important.

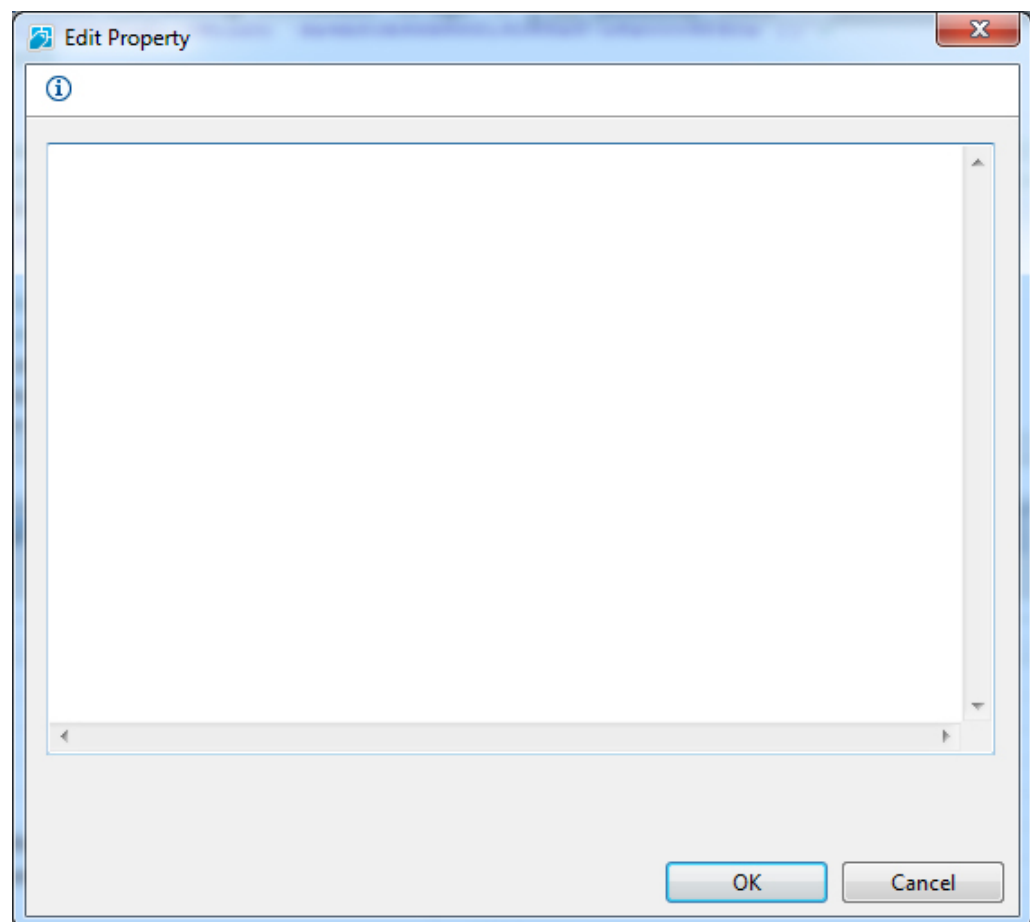
When you enter the code via the external property editor, it is passed directly through to the `<xe:this.script>` tag as follows:

```
<xe:this.script><![CDATA[ // Your code here
```

However, when you click the blue diamond to compute the value, the code is wrapped within `<![CDATA#javascript: ]`, which causes it to be parsed differently, creating problems that are not easy to track down. For example, parameter values are not accessible:

`<xe:this.script><![CDATA#javascript: // Your code here`

The downside is that the property editor has no script help, so it's just like writing code in a plain text editor, as Figure 8 shows.



**Figure 8** Using the property editor is like using a plain text editor—there is no script help

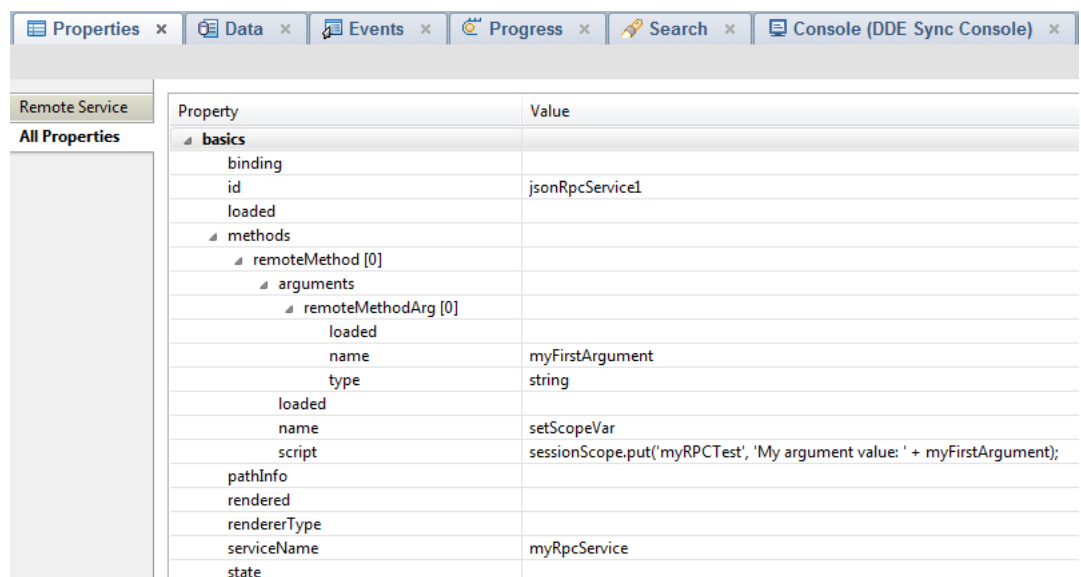
The computed value has the normal script editor that you would expect. Since it's not advisable to write a lot of code in the property editor, it is easier to keep the code in the property editor simple and just call a script library function.

## Passing Arguments

Next, we'll look at a method for adding arguments to a remote procedure call,

starting with our previous example. To let the remote method know to expect an argument, perform the following steps:

1. Select the Remote Service control and go to **All Properties** in the Properties view
2. Under **basics > methods > remoteMethod[0]**, click **arguments** and click the + button to add an argument
3. Set the name of the argument and (optionally) define the type to be **string**, **boolean**, **number**, or **list**
4. To use the argument, simply refer to it by name within the remote method script. In this example, I've updated it to put the value of the argument into the scope variable.
5. When calling the remote method from client-side JavaScript, add the value to be passed into the argument, as shown in Figure 9



Property	Value
basics	
binding	
id	jsonRpcService1
loaded	
methods	
remoteMethod [0]	
arguments	
remoteMethodArg [0]	
loaded	
name	myFirstArgument
type	string
loaded	
name	setScopeVar
script	sessionScope.put('myRPCTest', 'My argument value: ' + myFirstArgument);
pathInfo	
rendered	
rendererType	
serviceName	myRpcService
state	

**Figure 9** Add the argument to pass when calling the remote method from client-side JavaScript

Update the client-side JavaScript to call the method to pass a string value to the remote method:

```
myRpcService.setScopeVar('Hello');
```

Click the Trigger RPC Method button and then the Refresh Page button to see the scope variable that was set along with the value that was passed to the remote method, as shown in Figure 10.



sessionScope.myRPCTest.My argument value: Hello

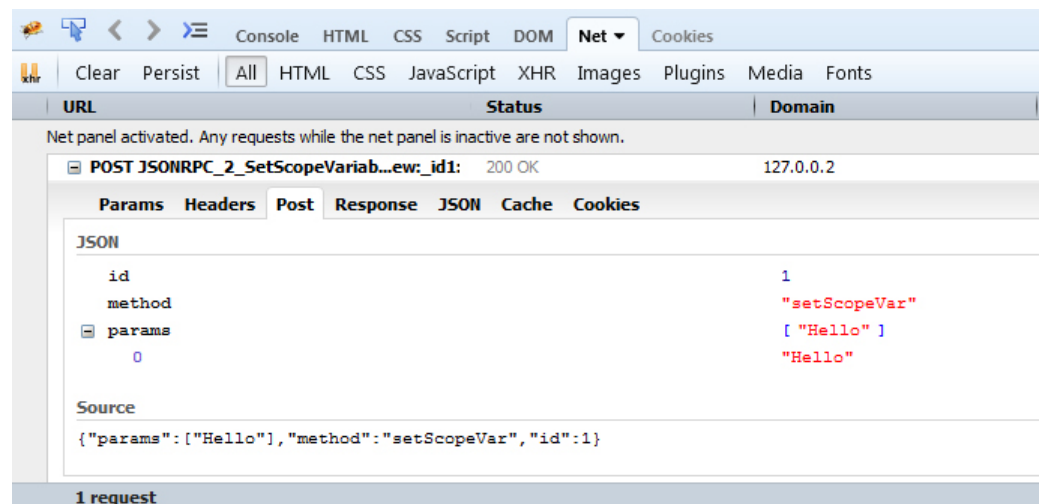
Trigger RPC Method Refresh Page

**Figure 10** See the scope variable and the value that was passed to the remote method by clicking the Trigger RPC Method and Refresh Page buttons

This is the full source of the Remote Service control now. Adding an argument added an `<xe:remoteMethodArg>` tag within an `<xe:this.arguments>` tag under the `<xe:remoteMethod>` tag

```
<xe:jsonRpcService id="jsonRpcService1" serviceName="myRpcService">
  <xe:this.methods>
    <xe:remoteMethod name="setScopeVar"
      script="sessionScope.put('myRPCTest', 'My argument value: ' +
        myFirstArgument);">
      <xe:this.arguments>
        <xe:remoteMethodArg name="myFirstArgument" type="string">
        </xe:remoteMethodArg>
      </xe:this.arguments>
    </xe:remoteMethod>
  </xe:this.methods>
</xe:jsonRpcService>
```

Figure 11 shows what the request looks like in Firebug. You can see that the argument was passed to the remote method as part of the params object.



**Figure 11** The request as it appears in Firebug, including the argument passed to the remote method

## Passing Dynamic Arguments

The example of passing arguments was simplistic because it passed a hard-coded value. However, you can easily modify it to pass a dynamic value by updating the client-side JavaScript that calls the remote method. You can pass any value that you can retrieve from the page via client-side JavaScript and you can even pass values that need to be retrieved from server-side JavaScript using expression language (EL) syntax.

For example, if you have a repeat control with a collection named **myRepeat** and a property named **myProperty**, you could pass/retrieve it in client-side JavaScript with this syntax:

```
'#{javascript: myRepeat.myProperty}'
```

Then your call to the remote method would look like this:

```
myRpcService.setScopeVar('#{javascript: myRepeat.myProperty}');
```

This syntax tells the server to evaluate the expression while generating the page and put the value into the generated client-side JavaScript, so it will only work for values that can be computed at the time the page is loaded (or refreshed).

You can call remote methods from outside an XPages component, but you cannot evaluate server-side expressions into client-side JavaScript without an XPages component.

As mentioned previously, the RPC method does not know anything about changes that have been made since the page was last posted, but that does not preclude you from passing an updated value to the method. Just get a handle to a field in client-side JavaScript, read the value, and pass it in to the remote method. If you want to get an Edit Box component's value, you could use EL syntax to get the generated ID for the field in client-side JavaScript so you can get a handle to the field and retrieve its value.

Continuing with the previous example, if you wanted to pass in the current value of an Edit Box component named **myField** to put in the scope variable, you could use a statement like this when calling the remote method:

```
myRpcService.setScopeVar(document.getElementById("#{id: myField}").value);
```

## Adding a Callback Function and Accepting a Return Value

The examples shown thus far open the door for plenty of uses for the Remote Service control, but it becomes even more useful when you add a callback function. The callback runs when the Remote Service is finished, and it allows you to accept a return value and then take further action based on it.

The general syntax for attaching a callback and retrieving the response is as follows:

```
serviceName.methodName().addCallback(function(response){
    // do stuff
});
```

You do not have to accept a response — the callback function can take no parameters. Here's an example of the callback function in action:

```
01 <xe:jsonRpcService id="jsonRpcService1" serviceName="myRpcService">
02   <xe:this.methods>
03     <xe:remoteMethod name="rpcGetUserName"
04       script="return context.getUser().getCommonName();">
05   </xe:remoteMethod>
06 </xe:this.methods>
07 </xe:jsonRpcService>
08
09 <div id="responseDiv" style="height:100px; width:200px;"></div>
10 <br />
11
12 <xp:button value="Trigger RPC Method" id="button1">
13   <xp:eventHandler event="onclick" submit="false">
14     <xp:this.script><![CDATA[
15       myRpcService.rpcGetUserName().addCallback(function(response){
16         console.log('RPC Response: ' + response);
17         document.getElementById('responseDiv').innerHTML = 'RPC
18           Response: ' + response;
19       });
20     ]></xp:this.script>
21   </xp:eventHandler>
22 </xp:button>
```

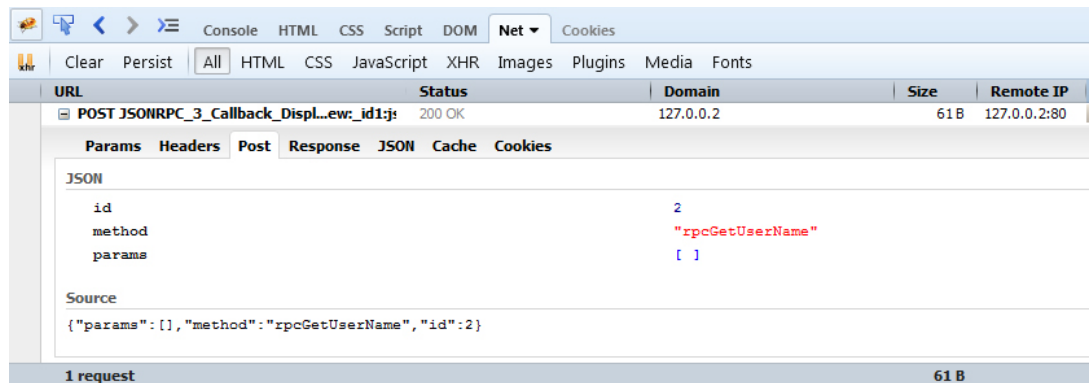
Lines 01-07 define a remote method that gets the current user name and returns it.

Line 09 defines a pass-thru HTML <div> tag where the response will be displayed.

Lines 14-19 are the client-side script that calls the remote method and adds a callback function that accepts the response and displays it both in the browser console and in the div shown in line 09.

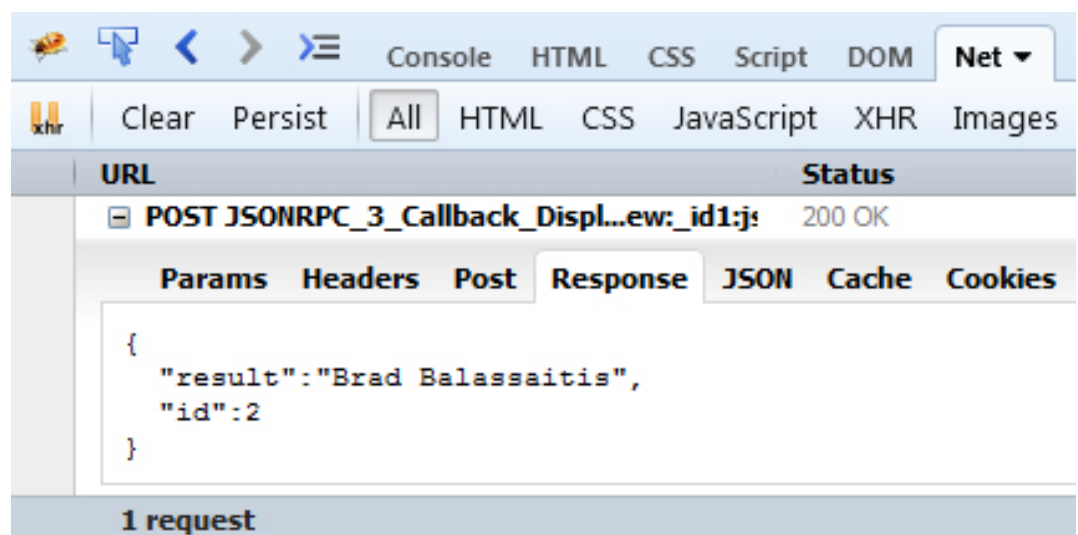
Pretty simple! Now we can asynchronously trigger server-side methods, pass parameters, accept a response, and trigger additional logic with a callback.

Figure 12 shows the remote method request in Firebug. The method is being called with no parameters because none were defined for this remote method.



**Figure 12** Remote method request with a callback function in Firebug

Since there was a return value, we can also see data populated in the request response, as shown in Figure 13.



**Figure 13** See data populated in the remote method response on the Response tab in Firebug

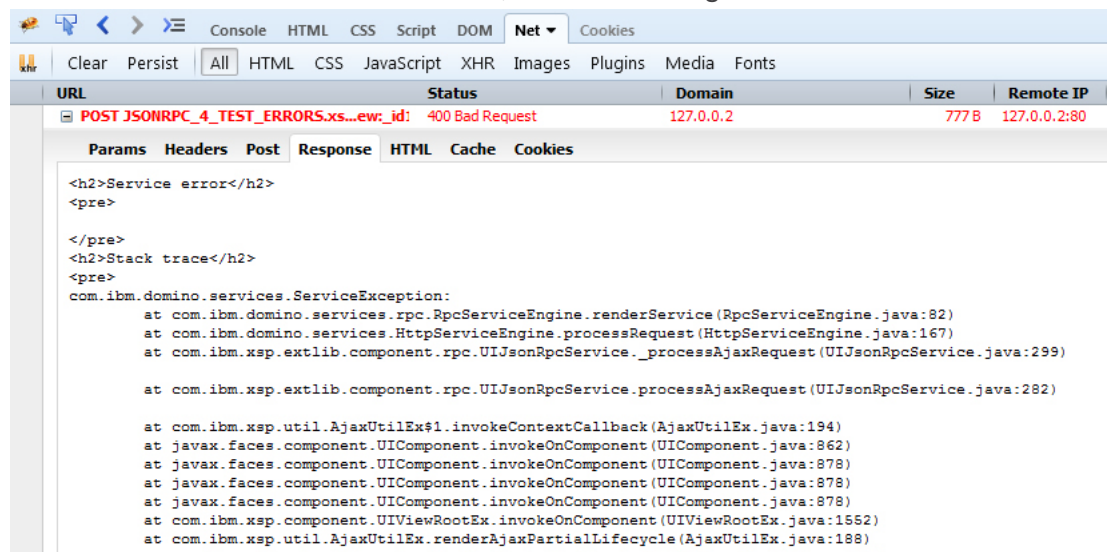
## Suggestions for Logic in an RPC Callback

To help get the gears turning for potential uses, you might consider doing the following within the callback:

- Display a response in the browser console, in an alert, as an inline message on the page, or — even better — as a growl-style message that floats up on the screen and does not block user interaction.
- Launch a dialog box to display a related document
- Trigger a partial refresh if something behind the scenes needs to update part of the page

## Handling Errors

If an error is thrown in the remote method, it will show in the server console but not on the page. Take care to handle errors in your remote method so that they fail gracefully and return a response that you can check for in your callback. Otherwise, the only way you'll know that there's an error is if you check the request in the browser tools and see an error status, as shown in Figure 14.



**Figure 14** An error in the remote method shows up as an HTTP 400 error in the browser tools

## A Success Story

I had an application for which I used a Remote Service to pass a value from client-side JavaScript to SSJS and then run client-side code so that a dialog box could be opened to display a document based on the unique identifier (UNID) in the scope variable. I replaced a method as I described at the beginning of this article



with an RPC call that passes the UNID, uses SSJS to store it in a scope variable, and then uses client-side JavaScript within the callback to open the dialog box to display the correct document based on the UNID in the scope variable.

The original logic ran in an average of 75.7 ms while the RPC call ran in an average of 20.1 ms (73.4% faster). And this comparison is only of the time it takes to get a value into a scope variable!

### Remotely Useful

Another great use that I've found for Remote Services is viewing a list of documents and displaying one in a slideout panel when a row is clicked. This can be done smoothly with JSON-RPC without requiring a page refresh for the following reasons:

- The slideout panel contains a document data source where the UNID is computed to read from a scope variable
- When a row is clicked, a client-side JavaScript event handler passes the UNID to a remote method
- The remote method simply puts the UNID into a scope variable

In a callback attached to the remote method execution, the slideout panel is opened and an **XSP.partialRefreshGet()** is triggered to update the document in the slideout

Even with the partial refresh, it's more efficient because it's not executing a POST.

Another variation is opening up a document in a dialog during the callback.

### Conclusion

By now, you are no doubt starting to see some of the many ways that the Remote Service control can be useful in your applications.

JSON-RPC = simplicity, speed, and client-server code interaction. ●