Mastering XPages

A Step-by-Step Guide to XPages Application Development and the XSP Language

Second Edition

Martin Donnelly • Mark Wallace • Tony McGuckin

FREE SAMPLE CHAPTER

SHARE WITH OTHERS
Related Books of Interest

XPages Portable Command Guide
A Compact Resource to XPages Application Development and the XSP Language
By Martin Donnelly, Maire Kehoe, Tony McGuckin, Dan O’Connor
0-13-294305-0
A Practical Primer for XPages Application Development, Debugging, and Performance
Straight from the experts at IBM, this perfect portable XPages quick reference offers fast access to working code, tested solutions, expert tips, and example-driven best practices. Drawing on their unsurpassed experience as IBM XPages lead developers and customer consultants, the authors explore many lesser known facets of the XPages runtime, illuminating these capabilities with dozens of examples that solve specific XPages development problems. Using their easy-to-adapt code examples, you can develop XPages solutions with outstanding performance, scalability, flexibility, efficiency, reliability, and value.

XPages Extension Library
A Step-by-Step Guide to the Next Generation of XPages Components
By Paul Hannan, Declan Scioilla-Lynch, Jeremy Hodge, Paul Withers, Tim Tripcony
0-13-290181-1
The first and only complete guide to Domino development with this library; it’s the best manifestation yet of the underlying XPages Extensibility Framework. Complementing the popular Mastering XPages, it gives XPages developers complete information for taking full advantage of the new components from IBM.
Combining reference material and practical use cases, the authors offer step-by-step guidance for installing and configuring the XPages Extension Library and using its state-of-the-art applications infrastructure to quickly create rich web applications with outstanding user experiences.

Sign up for the monthly IBM Press newsletter at ibmpressbooks.com/newsletters
IBM Lotus Connections 2.5
Planning and Implementing Social
Software for Your Enterprise
By Stephen Hardison, David Byrd, Gary Wood,
Tim Speed, Michael Martin, Suzanne Livingston,
Jason Moore, and Morten Kristiansen

In IBM Lotus Connections 2.5, a team of IBM
Lotus Connections 2.5 experts thoroughly intro-
duces the newest product and covers every facet
of planning, deploying, and using it success-
fully. The authors cover business and technical
issues and present IBM’s proven, best-practices
methodology for successful implementation. The
authors begin by helping managers and technical
professionals identify opportunities to use social
networking for competitive advantage—and by
explaining how Lotus Connections 2.5 places full-
fledged social networking tools at their fingertips.
IBM Lotus Connections 2.5 carefully describes
each component of the product—including
profiles, activities, blogs, communities, easy social
bookmarking, personal home pages, and more.

Survival Guide for
Lotus Notes and
Domino Administrators
By Mark Elliott
ISBN: 0-13-715331-7

Mark Elliott has created a true encyclopedia of
proven resolutions to common problems and
has streamlined processes for infrastructure
support. Elliott systematically addresses sup-
port solutions for all recent Lotus Notes and
Domino environments.

Survival Guide for Lotus Notes and Domino
Administrators is organized for rapid access
to specific solutions in three key areas: client
setup, technical support, and client software
management. It brings together best practices
for planning deployments, managing upgrades,
addressing issues with mail and calendars,
configuring settings based on corporate
policies, and optimizing the entire support
delivery process.

Listen to the author’s podcast at:
ibmpressbooks.com/podcasts

Visit ibmpressbooks.com
for all product information
Related Books of Interest

Lotus Notes Developer’s Toolbox
Tips for Rapid and Successful Deployment
By Mark Elliott

Lotus Notes Developer’s Toolbox will help you streamline and improve every phase of Notes development. Leading IBM Lotus Notes developer Mark Elliott systematically identifies solutions for the key challenges Notes developers face, offering powerful advice drawn from his extensive enterprise experience. This book presents best practices and step-by-step case studies for building the five most common types of Notes applications: collaboration, calendar, workflow, reference library, and website.

Listen to the author’s podcast at: ibmpressbooks.com/podcasts

DB2 Essentials
Understanding DB2 in a Big Data World, 3rd Edition
Chong, Liu
ISBN: 0-13-346190-4

IBM Cognos Business Intelligence v10
The Complete Guide
Gautam

An Introduction to IMS
Klein, Long, Blackman, Goff, Nathan, Lanyi, Wilson, Butterweck, Sherrill
ISBN: 0-13-288687-1

The Art of Enterprise Information Architecture
A Systems-Based Approach for Unlocking Business Insight
Godinez, Hechler, Koenig, Lockwood, Oberhofer, Schroeck

DB2 Developer’s Guide
A Solutions-Oriented Approach to Learning the Foundation and Capabilities of DB2 for z/OS, 6th Edition
Mullins
ISBN: 0-13-283642-4

Sign up for the monthly IBM Press newsletter at ibmpressbooks.com/newsletters
This page intentionally left blank
For my wife and best friend Aileen, survivor of three XPages books ... please don’t let me write any more! Somehow “thanks” can never say enough for all your help, advice, and patience.
—Martin

For Dee, Sam, and Becky. Second time around and a new set of challenges— but the support, encouragement, and patience are still the same.
Go raibh mile maith agat, mo Ghrá.
—Mark

First, a big thank you to Martin and Eamon for giving leadership, motivation, time, and most of all, understanding as we all made our journey through this second edition! Second, sincere thanks and love to my parents for being truly the best parents—and also to my extended family, thanks to all the in-laws and out-laws! Finally, I dedicate this book to Paula and Anna-Rose—always live happy and follow your dreams wherever they take you. All my love!
—Tony
Contents at a Glance

Foreword .................................................. xxiii
Preface .................................................. xxvii
Acknowledgements ................................. xxxiii
About the Author ...................................... xxxv

Part I: Getting Started with XPages
Chapter 1  An Introduction to XPages ................. 3
Chapter 2  Getting Everything You Need .............. 13
Chapter 3  Building Your First XPages Application .. 33

Part II: XPages Development: First Principles
Chapter 4  Anatomy of an XPage ....................... 57
Chapter 5  XPages and JavaServer Faces .............. 157
Chapter 6  Building XPages Application Logic ........ 187

Part III: Data Binding
Chapter 7  Working with Domino Documents .......... 253
Chapter 8  Working with Domino Views .............. 293
Chapter 9  Beyond the View Basics .................... 337
Part IV: Programmability
Chapter 10  Custom Controls ................................. 401
Chapter 11  Advanced Scripting .............................. 443
Chapter 12  XPages Extensibility ............................ 537
Chapter 13  XPages in the Notes Client ................... 607
Chapter 14  XPages Mobile Application Development .... 677
Chapter 15  XPages Unplugged and Debugged .......... 725

Part V: Application User Experience
Chapter 16  XPages Theming ................................. 769
Chapter 17  Application Layout .............................. 849
Chapter 18  Internationalization ............................ 871

Part VI: Performance, Scalability, and Security
Chapter 19  A First Look at Performance and Scalability ... 905
Chapter 20  Advanced Performance Topics ................ 931
Chapter 21  Security ........................................... 1035

Part VII: Appendixes
Appendix A  XSP Programming Reference .................. 1075
Appendix B  XSP Style Class Reference .................... 1081
Appendix C  Useful XPages Sites on the Net ................ 1087
            Index .................................................. 1089
Contents

Foreword ................................................................. xxiii
Preface ................................................................ xxvii
Acknowledgements ................................................... xxxiii
About the Author ....................................................... xxxv

Part I  Getting Started with XPages

Chapter 1  An Introduction to XPages ............................... 3
  XPages Fundamentals .................................................. 3
  Onward and Upward: A Path of Continuous Improvement .......... 4
  The XPages Development Paradigm .................................... 8
    The More Things Change, the More Things Stay the Same ....... 10
  New Horizons ............................................................ 11
  Conclusion ................................................................. 12

Chapter 2  Getting Everything You Need .......................... 13
  Downloads, Versions, and Locations ................................ 13
  Installing Domino Designer .......................................... 14
  Installing Client Fix Packs ........................................... 14
  Client Configuration .................................................. 15
  Quick Tour of Domino Designer ..................................... 16
    Domino Designer Home Page ..................................... 17
    Domino Designer Perspective ..................................... 17
    Creating a New Application ....................................... 19
    Creating an XPage .................................................. 20
    Previewing in the Notes Client .................................... 21
    Previewing in a Web Browser ...................................... 22
    Adding a Control to an XPage .................................... 24
## Contents

XPage Resources .......................... 111
  JavaScript Library .......................... 112
  Style Sheet .......................... 114
  Resource Bundle .......................... 115
  Dojo Module and Dojo Module Path .......... 116
  Generic Head Resource .................. 116
  Metadata Resource ........................ 117
Converters ................................ 118
Validators ................................ 121
Simple Actions .......................... 128
Client-Side Scripting ...................... 134
HTML Tags ................................ 136
Extension Library ......................... 137
  Dynamic Content ........................ 138
  Change Dynamic Content Action ............ 142
  In Place Form .......................... 143
  Dialog, Dialog Context, and Dialog Button Bar .......... 147
  Tooltip Dialog ........................ 150
  JSON RPC Service (Remote Service) .......... 154
Conclusion ................................ 156

Chapter 5  XPages and JavaServer Faces. ................. 157
  What Is JavaServer Faces? .................. 158
  JSF Primer ................................ 159
  How Does XPages Extend JSF? ................. 166
    XML-Based Presentation Tier ................. 169
    Request Processing Lifecycle .................. 169
    User Interface Component Model ............... 170
    Standard User-Interface Components ........... 176
    Value Binding and Method Binding Expression Evaluation .......... 179
    XPages Default Variables .................. 182
Conclusion ................................ 186

Chapter 6  Building XPages Application Logic .............. 187
  Adding Application Logic ................... 187
    Using the xp:eventHandler Tag ................. 190
  Simple Actions .......................... 198
    Change Document Mode ...................... 198
    Confirm Action .......................... 199
    Create Response Document .................. 200
    Delete Document ........................ 201
    Delete Selected Documents ................. 202
    Execute Client Script ...................... 203
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute Script</td>
<td>204</td>
</tr>
<tr>
<td>Modify Field</td>
<td>205</td>
</tr>
<tr>
<td>Open Page</td>
<td>205</td>
</tr>
<tr>
<td>Publish Component Property</td>
<td>207</td>
</tr>
<tr>
<td>Publish View Column</td>
<td>208</td>
</tr>
<tr>
<td>Save Data Sources</td>
<td>209</td>
</tr>
<tr>
<td>Save Document</td>
<td>211</td>
</tr>
<tr>
<td>Set Component Mode</td>
<td>213</td>
</tr>
<tr>
<td>Set Value</td>
<td>214</td>
</tr>
<tr>
<td>Action Group</td>
<td>215</td>
</tr>
<tr>
<td>Send Mail</td>
<td>217</td>
</tr>
<tr>
<td>Change Dynamic Content</td>
<td>219</td>
</tr>
<tr>
<td>Move to Application Page</td>
<td>220</td>
</tr>
<tr>
<td>Using JavaScript with XPages</td>
<td>220</td>
</tr>
<tr>
<td>Server-Side JavaScript</td>
<td>221</td>
</tr>
<tr>
<td>Client JavaScript</td>
<td>240</td>
</tr>
<tr>
<td>Conclusion</td>
<td>250</td>
</tr>
</tbody>
</table>

**Part III Data Binding**

**Chapter 7 Working with Domino Documents**

- Domino Document Data Source                  254
- Creating and Editing Documents              257
- Controlling URL Parameter Usage             258
- Creating Response Documents                258
- Executing Form Logic                        263
- Managing Concurrent Document Updates        266
- Multiple Document Data Sources             272
- Document Data Source Events                 274
- webQuerySaveAgent                          278
- Common Data Source Properties              282
- Miscellaneous Data Source Properties       282
- Working with Domino Documents—Programmatically! 283
  - Simple Actions                           283
  - JavaScript                                284
- Rich Documents                             286
- Conclusion                                 291

**Chapter 8 Working with Domino Views**

- databaseName Property                      295
- View Data Source Filters                   296
- categoryFilter Property                   297
- Full Text Search Properties               299
- parentId Property                         304
### Chapter 9  Beyond the View Basics

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pick a View Control, Any View Control</td>
<td>337</td>
</tr>
<tr>
<td>The View Control: Up Close and Personal</td>
<td>340</td>
</tr>
<tr>
<td>Column Data Like You’ve Never Seen Before</td>
<td>341</td>
</tr>
<tr>
<td>Simple View Panel Make Over</td>
<td>343</td>
</tr>
<tr>
<td>Working with Categories</td>
<td>357</td>
</tr>
<tr>
<td>View Properties and View Panel Properties</td>
<td>366</td>
</tr>
<tr>
<td>Data Table</td>
<td>370</td>
</tr>
<tr>
<td>Building a Mini Embedded Profile View Using a Data Table</td>
<td>376</td>
</tr>
<tr>
<td>Repeat Control</td>
<td>381</td>
</tr>
<tr>
<td>A Repeat Control Design Pattern</td>
<td>383</td>
</tr>
<tr>
<td>Nested Repeats</td>
<td>384</td>
</tr>
<tr>
<td>The Rich Get Richer</td>
<td>386</td>
</tr>
<tr>
<td>Data View</td>
<td>387</td>
</tr>
<tr>
<td>Configuring a Basic Data View Control</td>
<td>389</td>
</tr>
<tr>
<td>Using More Advanced Data View Control Features</td>
<td>392</td>
</tr>
<tr>
<td>Some Fun with the Pager</td>
<td>395</td>
</tr>
<tr>
<td>Conclusion</td>
<td>398</td>
</tr>
</tbody>
</table>

### Part IV  Programmability

#### Chapter 10  Custom Controls

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Divide and Conquer</td>
<td>402</td>
</tr>
<tr>
<td>Getting Started with Custom Controls</td>
<td>403</td>
</tr>
</tbody>
</table>
Create a Renderer and Register It in the Application Configuration (faces-config.xml) ........................................... 551
Quick Test Application to Verify Everything Is OK So Far ........................................... 554
Working with Component Properties ........................................... 555
Component Properties and Attributes ........................................... 555
Adding a Property to a Component ........................................... 556
State Holder: Saving State Between Requests ........................................... 556
Specifying Simple Properties ........................................... 557
Inheriting xsp-config Properties ........................................... 558
Create the Initial xsp-config Definitions ........................................... 562
Create base.xsp-config ........................................... 562
Create an Interface to Match the Group Property Definition in base.xsp-config ........................................... 565
Revisit the Component Properties in Domino Designer ........................................... 568
Specifying Complex Properties ........................................... 568
Complete the xsp-config for the UISpinner Component ........................................... 579
Complete the UI Component Extension, UISpinner ........................................... 588
Complete the Renderer UISpinnerRenderer ........................................... 591
Create a Sample Application Using the UISpinner Component ........................................... 597
Take Your New UI Component Extension for a Test Drive ........................................... 597
Create a Backing Bean ........................................... 597
Register the Backing Bean ........................................... 600
Create the Final Test Application ........................................... 600
Nice Look and Feel ........................................... 604
Test to Ensure That It All Works! ........................................... 604
Where to Go from Here ........................................... 605
XPages Extensibility API Developers Guide ........................................... 605
XPages Extension Library ........................................... 606
IBM developerWorks ........................................... 606
Conclusion ........................................... 606

Chapter 13 XPages in the Notes Client ........................................... 607
Think Inside the Box ........................................... 608
Getting Started with XPages in the Notes Client ........................................... 610
3, 2, 1...Lift Off ........................................... 612
Bookmarks ........................................... 614
Working Offline ........................................... 616
One of These Things Is Not Like the Other ........................................... 619
Other Subtle Differences ........................................... 621
Extended Client-Side JavaScript Commands ........................................... 624
XPages: A Good Notes Citizen ........................................... 632
Introducing enableModifiedFlag and disableModifiedFlag ........................................... 634
Keeping Tabs on Your Client Apps ........................................... 637
Notes Links Versus Domino Links ........................................... 641
Some XPiNC Debugging Tips .......................................................... 645
Optimizing XPages for Notes ......................................................... 649
  Single Copy XPages Design Meets Preload ................................. 652
  XPages RunOnServer ............................................................... 657
XPages and Composite Applications .............................................. 664
  Making a Component of an XPages Application .............................. 664
  Is Anyone Out There? Creating a Component that Listens to Your XPages Component ... 666
  Assembling a Composite Application: Aggregating the XPages Discussion
    Component and Notes Google Widget ........................................ 668
  Hey, This Is a Two-Way Street. A Component May Receive
    and Publish Events ............................................................ 672
Further Adventures with Composite Applications ............................. 675

Chapter 14  XPages Mobile Application Development ............. 677
Getting Started with Mobile Application Development .................... 678
  Safari Browser ................................................................. 680
  Chrome Browser .............................................................. 681
  Firefox Browser ............................................................. 682
  User Agent Device Detection ............................................... 682
  Device Bean ................................................................. 683
Single Page Application Design Pattern ...................................... 685
  Mobile XPage Properties ..................................................... 686
  Single Page Application Control (xe:singlePageApp) ..................... 687
Mobile Application Navigation ............................................... 688
  Navigator ................................................................. 690
  Hierarchical Navigation ................................................... 692
  Context-Sensitive Navigation ............................................... 694
Interacting with a Mobile Application ........................................ 697
  Orientation-Based Interaction ............................................. 697
  Touch-Based Interaction .................................................... 701
  Multitouch-Based Interaction ............................................. 702
Mobile Themes ................................................................. 703
  Data View ................................................................. 704
  Outline ................................................................. 706
  Form Table ............................................................... 706
  Styling XPages Controls for Mobile Applications ....................... 707
Debugging Mobile XPages ........................................................ 710
  Debugging XPages on iOS .................................................... 711
  Debugging XPages with Web Inspector Remote (aka weinre) .......... 713
XPages Mobile Extensions ...................................................... 716
  Infinite Scrolling .......................................................... 717
  Single Page Application Wizard .......................................... 718
Summary ................................................................. 724
Chapter 15  XPages Unplugged and Debugged 725
  Debugging XPages Apps on the Server Side 726
  Printing, Dumping, and Logging 726
  try / catch / finally 729
  Introducing the SSJS Debugger 735
  Using the Java Debugger 748
  Enabling XPages Java Logging 755
  Debugging XPages Apps on the Client Side 759
  CSJS Debuggers 760
  Debugging Dojo 762
  Conclusion 765

Part V  Application User Experience

Chapter 16  XPages Theming 769
  It Used to Be Like That...But Not Anymore 769
  Styling with Style 771
  Setting the Style Property Manually 776
  Understanding How the Style Property Is Used 777
  Computing the Style Property 778
  Styling with Class 779
  Getting Something for Nothing 779
  Understanding How the styleClass Property Is Used 785
  Computing the styleClass Property 788
  Working with Extended styleClass and style Properties 790
  Theming on Steroids! 794
  What Is a Theme? 794
  What Can You Do with a Theme? 795
  Understanding Theme Architecture and Inheritance 796
  Working with a Theme 804
  Theme Resources 814
  Resource Paths 824
  Theme Properties, themeld, Control Definitions, and Control Properties 832
  Conclusion 848

Chapter 17  Application Layout 849
  Divide and Conquer 849
  Application Layout: One Easy Way 850
  Application Layout: Customizing the Content Area 865
  Conclusion 870
Chapter 18  Internationalization ........................................... 871
   Using Localization Options ........................................... 872
   Localization with Resource Bundle Files ......................... 873
   Setting Localization Options ..................................... 874
   Testing a Localized Application .................................. 877
   Working with Translators .......................................... 878
   Merging XPage Changes ............................................. 881
   Gotchas! ............................................................... 883
   Localizing Computed Expressions and JavaScript ................. 885
   Adding a Resource Bundle .......................................... 887
   Localizing Computed Expressions .................................. 889
   Localizing Client-Side JavaScript ................................. 889
   Localizing Script Libraries ...................................... 890
   Server-Side Script Libraries ..................................... 890
   Client-Side Script Libraries ..................................... 891
   International Enablement ......................................... 893
   Locales in XPages .................................................. 894
   Deprecated Locale Codes ......................................... 898
   Localizing Computed Fields ....................................... 900
   Conclusion ......................................................... 901

Part VI  Performance, Scalability, and Security

Chapter 19  A First Look at Performance and Scalability ......... 905
   Golden Rules ....................................................... 906
   Understanding the XPages Request Processing Lifecycle ........ 908
      GET-Based Requests and the XPages Request Processing Lifecycle  909
      POST-Based Requests and the XPages Request Processing Lifecycle  910
   Reducing CPU Utilization ......................................... 912
      GET-Based Versus POST-Based Requests .......................... 912
      Partial Refresh ................................................ 917
      Partial Execution Mode ...................................... 919
   Reducing Memory Utilization .................................... 923
      HTTPJVMMaxHeapSize and HTTPJVMMaxHeapSizeSet Parameters  924
      xsp.persistence.* Properties ................................ 925
      dataCache Property ........................................... 926
   Conclusion ....................................................... 928

Chapter 20  Advanced Performance Topics ......................... 931
   Making Efficient XPages Requests ................................ 931
   Profiling XPages Applications .................................... 932
   Introspecting XPages Requests Using a PhaseListener ........... 949
      The Myths and Realities of the Rendered and Loaded Properties  964
      Using Partial Refresh, Partial Execution, and Dynamic Content  981
Already three years have elapsed since we launched the first edition of this best seller. I say “best seller” because not only has it been a big hit with the community, but it also broke many IBM Press publishing records. I have seen it on a lot of desks while visiting customers, which is a real indicator of its true value! And there is a good reason for this: This book is an exhaustive reference for every XPages developer, from novice to expert. And this release is a worthy successor, with extensive information on all the major new features. Kudos to the gang of authors—it is most definitely an honor for me to be invited to take part in this adventure again.

Three years—even though it allowed my kids to grow by a couple of inches—it still seems like yesterday. But, in the IT space, it feels like decades. A lot of water has flowed under the bridge during this time. The IT landscape evolved toward social, mobile, and cloud development. Let’s look at what happened over the course of this time and what’s coming next.

The first edition of Mastering XPages was based on Notes®/Domino® 8.5.2. This was before the advent of the XPages Extension Library, an asset so important to XPages developers today that life without it seems unthinkable! Not only did this deliver a slew of new capabilities in and of itself, but it allowed us to offer a new application development paradigm to the community whereby XPages features could be delivered outside of the regular Notes/Domino product release cycle. In fact, we made good on this promise by delivering Upgrade Pack 1 for 8.5.3 in December 2011 (just a few months after 8.5.3 itself shipped). More goodies came in 9.0 with the release of a new server-side JavaScript™ debugger in Domino Designer. At the same time, we addressed performance issues with XPages for the Notes client and have continued to upgrade our key components such as Dojo, CKEditor, and XULRunner with each release. In 9.0.1 the XPages runtime has been updated to meet the latest and most stringent accessibility standards and achieve Section 508 compliance. It has also seen its support for mobile application development make significant strides—more on this a little later.
But in a sense, perhaps the biggest achievement for XPages over all this time has been its solidification. From a technical standpoint, we have made the code very robust, performant, and functional. These three intervening years have sealed the adoption of XPages by the community. We have seen many customers moving to XPages—not just to modernize existing applications, but also to create completely new applications from the ground up. I remember the early days when we were looking for successful implementations of XPages. It was a challenge as the technology was so new, but nowadays XPages is widely used for both mission-critical and situational applications. Because of great customer adoption, it now has a large and solid install base around the globe and in all types of organizations, from nonprofits to small and medium enterprises (SMEs) to Fortune 500 corporations. Japan has an “XPages Day” event; an XPages Code-A-Thon has just completed in India...and did you know that Mastering XPages has been translated to Chinese?

On the social side, we recently integrated the XPages runtime with the IBM Social Business® Toolkit SDK. This enables XPages applications to extend their reach beyond Notes/Domino and integrate with the broader ICS social platform. By leveraging this SDK, developers can easily integrate social features into their applications and move them to the next level. For example, it becomes easy to collaborate through a community, to query people profiles, to share files, and have a unified search mechanism. This opens up a world of new possibilities, particularly for organizations that have a mix of ICS technologies deployed. From a technical standpoint, because the SDK also targets regular J2EE platforms, the code is the same between all these platforms. This demonstrates one of the key capabilities of XPages for sharing and reusing code across the portfolio.

The XPages mobile story started on OpenNTF.org when a set of dedicated mobile controls was released in 2011. This mobile library was ultimately productized and released in 8.5.3 UP1, along with a “mobilized” version of the Discussion and Teamroom templates. Further improvements have arrived more recently in 9.0.1, including new server-side APIs to assist with device detection and resource management, new mobile debug enablers, and a new common mobile theme based on the IBM OneUI Dojo Extension stylekit (aka IBM IDX). Mobile will definitely be a key focus area in the future: We want XPages to be a technology of choice for writing web-based mobile applications, leveraging the latest and most popular libraries. Despite the proliferation of different devices, with different screen sizes and UI, we aim to ensure that the promise of our write-once-run-everywhere paradigm is a reality in this space. Furthermore, XPages now has the infrastructure to support responsive design—an essential building block for modern mobile applications—and you should expect to see more innovation in this area coming soon in both the core product and on OpenNTF.org. One such example is the recently released Twitter Bootstrap4XPages project on OpenNTF, which makes the great Twitter Bootstrap UI framework available to XPages developers.

To conclude, I would like to talk about increasing access to our application development platform. It is our constant goal to break down barriers to entry and to on-board more and more developers by making our development and deployment story quicker, lighter, and better for
everyone. Think about how the cloud can help us here—for instance, what about a lightweight design-time experience provided via a web-based design tool? This would make XPages immediately available to any developer anywhere! Couple with this a simple facility for instantly deploying the resulting applications to the cloud, and you suddenly have a massively powerful and productive environment. Moreover, by adding tight and seamless integration with IBM SmartCloud® for Social Business, you have a compelling and valuable new model for your enterprise featuring XPages as its shining star.

So the XPages story continues apace, embracing all the new technologies and trends: social, mobile, and cloud computing. What is coming is exciting, and XPages aims to put all of it at your fingertips. Enjoy this book—it marks the gateway to the future.

Philippe Riand
IBM Senior Technical Staff Member
ICS Social Application Development Architect
This page intentionally left blank
Preface

XPages made its official public debut in Notes/Domino version 8.5, which went on general release in January 2009. In the intervening 5 years, there have been five more releases, and that’s not including the XPages Upgrade Pack, which shipped for 8.5.3, or the many XPages Extension Library releases on OpenNTF.org. Even since the first edition of this book was published in 2011, two other XPages books have come to market; the XPages Portable Command Guide and the XPages Extension Library. XPages boasts a vibrant global development community that has adopted, extended, and innovated with the technology in so many different ways to build a veritable myriad of applications. Suffice to say it has been a rapid, eventful, and successful journey so far, and one which the authors of this book have at times struggled to keep pace with! But nonetheless, the goal of this new edition remains the same as before: to provide a single comprehensive guide that enables readers to confidently take on, or actively participate in, real-world XPages application-development projects.

Approach

The first edition of this book is based on XPages in Notes/Domino 8.5.2. This edition is based on XPages in Notes/Domino 9.0.1 and thus is a considerably larger volume than its predecessor, as it needs to cover a lot more ground. As well as wide-ranging updates to the material featured in the first edition, it also adds four new chapters and several hundred extra pages of content. Despite its considerable bulk, it is intended to be accessible to both novice and expert alike, and aims to provide all the help and information needed to get XPages projects built and delivered to the highest standard.

The authors seek to cover all aspects of the XPages development spectrum and to engage the reader with hands-on problems wherever possible. Most chapters come with one or more sample applications that provide plentiful exercises and examples aimed at enabling you to
quickly and efficiently solve everyday development challenges. These resources are located on
the web at www.ibmpressbooks.com/title/9780133373370, so waste no time in downloading
before getting started!

Conventions
Any programming code, markup, or XSP keywords are illustrated in numbered listings using a
fixed width font.

User-interface elements (menus, links, buttons, and so on) of the Notes client, Domino
Designer, or any sample applications are referenced using a bold font.

Visual representations of the design-time experience or runtime features are typically cap-
tured as screen shots and written as numbered figures, using superimposed callouts where appro-
priate.

How This Book Is Organized
This book is divided into seven parts to separately address the many different aspects of XPages
software development in as logical a manner as possible:

Part I, “Getting Started with XPages”: This part gets you familiar with XPages at a con-
ceptual level. It aims to have you up and running quickly on the basics of the design-time tooling
and runtime framework, and to get you comfortable with the overall application development
paradigm.

• Chapter 1, “An Introduction to XPages”: Here, you are introduced to the history of
XPages and given some high-level insights into its design principles in order to under-
stand exactly what it is and what it is not. This is all about giving you the right context
for XPages by defining the problems it solves, the technologies on which it is based, and
where it might go in the future.

• Chapter 2, “Getting Everything You Need”: This chapter concerns itself with the
practical business of obtaining, installing, and configuring Domino Designer and suc-
cessfully walking you through your first “Hello World” XPage! It also focuses on the
XPages Extension Library and how best to integrate it into your XPages development
environment.

• Chapter 3, “Building Your First XPages Application”: This chapter aims to provide a
breadth-first hands-on experience of building a simple web application using the XPages
integrated development environment. This is really just an introductory practical to get
your feet wet and ensure you are comfortable with the basics of the application develop-
ment model before diving any deeper.
Part II, “XPages Development: First Principles”: This part is mostly architectural in nature and aims to give you an appreciation of what’s happening under the XPages hood. This is an essential prerequisite to some of the more advanced topics, like XPages performance and scalability.

- **Chapter 4, “Anatomy of an XPage”:** This chapter examines the XSP markup language and gives a simple example of all the XPages core elements (controls and such) as well as the more important elements contributed to XPages via the XPages Extension Library. It provides a great broad-based view of XPages basics.

- **Chapter 5, “XPages and JavaServer Faces”:** This chapter looks at JavaServer Faces (JSF), which is the web-application development framework on which XPages is based. It looks at some core JSF design points and how XPages leverages and extends the framework.

- **Chapter 6, “Building XPages Application Logic”:** This chapter is a primer for XPages programmability. It introduces the various tools that can be used to implement XPages application logic so that you will be ready to work with the practical examples that are coming down the pike.

Part III, “Data Binding”: This part is really about how XPages reads and writes Notes/Domino data. XPages comes with a library of visual controls that are populated at runtime using a process known as data binding. The mechanics of the data binding process is explored in depth for Notes views and documents.

- **Chapter 7, “Working with Domino Documents”:** This chapter focuses on reading and writing Notes documents via XPages. Advanced use cases are explored and every design property on the Domino document data source is explained and put through its paces using practical examples.

- **Chapter 8, “Working with Domino Views”:** In this chapter, the Domino view data source is dissected and examined, property by property. A section is also dedicated to Domino calendar views, including an in-depth look at how to use REST services to access calendar data. Again, practical exercises are used to drive home the material under discussion.

- **Chapter 9, “Beyond the View Basics”:** Working with Notes/Domino views is a large subject area, so much so that it demands a second chapter to cover all the details. This chapter looks at the various container controls that are available in the standard XPages control library, whose job is to display view data in different formats and layouts in order to support a myriad of customer use cases. This edition includes an in-depth look at the DataView control that was added to the XPages runtime in Notes/Domino 9.0.
Part IV, “Programmability”: This part covers the black art of programming—essentially how to code your applications to do everything from the most basic user operation to writing your own controls that implement completely customized behaviors. This part includes a look at XPages in the Notes client and considers cross-platform application development issues. This edition adds two new chapters in this part focused on XPages debugging and mobile application development.

- **Chapter 10, “Custom Controls”:** This chapter explains the “mini-XPage” design element that is the custom control. It explains how to leverage the custom control in order to “componentize” your application and then maximize the reuse of your XPages development artifacts.

- **Chapter 11, “Advanced Scripting”:** Advanced scripting is an umbrella for many cool topics, like AJAX, Dojo, @Functions, agent integration, managed beans, and so forth. This edition includes a new, extensive field guide that looks at the practicalities of extending the functionality of the rich text editor. This is a must for anyone looking to add pizzazz to their XPages applications.

- **Chapter 12, “XPages Extensibility”:** This chapter explains how to use the XPages extensibility APIs to build and/or consume new controls. This is an amazingly powerful feature that has only recently become available and is well worth exploring once you have mastered XPages fundamentals.

- **Chapter 13, “XPages in the Notes Client”:** XPages in the Notes client initially explains how you can take your XPages web applications offline and then goes on to highlight how you can take advantage of powerful features of the client platform itself, and how to manage applications that run in both environments. The content of this chapter is considerably expanded in this edition to account for many new innovations in this space since Notes/Domino 8.5.2. In particular, it gives in-depth examinations to performance related enhancements.

- **Chapter 14, “XPages Mobile Application Development”:** This new chapter explains how to build Domino mobile applications using XPages. It covers mobile application design patterns and best practices, as well as all the XPages mobile controls. It gives invaluable information as to how best to debug XPages mobile applications and also looks at the very latest XPages mobile extensions available on OpenNTF.org.

- **Chapter 15, “XPages Unplugged and Debugged”:** This new chapter explains the many and varied means of debugging XPages applications—everything from basic printing and logging techniques right through to a thorough exploration of the Server-Side Java Script debugger which was added to Domino Designer 9.0. It also features sections on Java debugging and Client-Side JavaScript debugging.
Part V, “Application User Experience”: This part is all about application look and feel. You learn not just how to make your apps look good and behave well, but how to do so for an international audience! It also includes a new chapter on the enhanced application layout features that were delivered as part of the 9.0 release.

- **Chapter 16, “XPages Theming”**: This chapter teaches you how to manage the appearance and behavior of your application’s user interface. It provides an in-depth look at ad-hoc XPages application styling using cascading style sheets, as well as the main features of the standard XPages UI themes, and explains how to create your own customized themes.

- **Chapter 17, “Application Layout”**: This new chapter describes how to build slick user interfaces quickly using out-of-the-box controls, in particular the Application Layout control that gives the chapter its name.

- **Chapter 18, “Internationalization”**: Read this chapter to learn how your XPages applications can be translated so that they look, feel, and behave as native applications in any geographical locale.

Part VI, “Performance, Scalability, and Security”: Up to this point this book has concentrated on the skills and tools you need to know to develop state-of-the-art collaborative applications. Part VI shifts to deployment and what you need to do to make sure your applications meet customer expectations in terms of performance, scalability, and security.

- **Chapter 19, “A First Look at Performance and Scalability”**: This chapter highlights various tips and tricks that will enable you to tune your XPages application for optimal performance and scalability in various deployment scenarios.

- **Chapter 20, “Advanced Performance Topics”**: This voluminous chapter is new to the second edition and aims to impart all you ever need to know about XPages performance and scalability. Building on the previous chapter, it introduces you to the XPages Toolbox—an essential utility used to profile XPages applications and identify problem performance areas. It also explains key aspects of the XPages request processing lifecycle, and what you need to understand when using partial refresh, partial execute, dynamic content and so forth. This is essential reading for anyone putting XPages applications into production.

- **Chapter 21, “Security”**: Learn about application security issues and considerations and see how XPages integrates with the Domino server and Notes client security models.

Part VII, “Appendixes”

- **Appendix A, “XSP Programming Reference”**: This appendix points to a collection of definitive reference sources that describe all the details of the XSP tags, Java™ and JavaScript classes. It provides examples of how to use these resources to find the information you need.
• **Appendix B, “XSP Style Class Reference”:** This appendix identifies all the standard XPages CSS files and style classes used to build XPages application user interfaces. It’s an essential quick reference for Chapter 16.

• **Appendix C, “Useful XPages Sites on the Net”:** A snapshot of the authors’ favorite XPages websites at the time of writing. This list of sites should help you find whatever it is you need to know about XPages that isn’t found in this book.
One might be forgiven for thinking that a second edition of any book is no major undertaking, but somehow or other the effort required on this occasion has proven every bit as challenging as the first time around! After a long haul, we have finally gotten over the line and we would like to take this opportunity to thank the many people who helped us stay the course.

It is only fitting that we start with our two technical reviewers, Dan O’Connor and Paul Withers. You two did a tremendous job and didn’t just confine your feedback to the new content. Your keen insights and observations have made this a much better book.

A sincere thank you to Eamon Muldoon, whose support on this effort was critical to its success. In fact, most every member of the XPages and Domino Designer teams were leaned on for a bit of help at some stage of the process—so “muchas gracias” to Brian Gleeson, Carlos Parreno Bonano, Darin Egan, Dario Chimisso, Gary Marjoram, Jonathan Roche, Lisa Henry, Máire Kehoe, Padraic Edwards, Paul Hannan, Robert Dignam and Torsten Weigelt. A special mention also goes out to Pete Janzen, Scott Morris, John Woods, and Philippe Riand for their encouragement, support, and advocacy of all things XPages. And also to Jim Quill, whose contributions to the original book remain largely intact in this updated edition.

Needless to say, we remain indebted to all those who helped us with the first edition, namely: Azadeh Salehi, Bill Hume, Brian Bermingham, Brian Leonard, Dan O’Connor, Dave Connolly, Dave Kern, David Taieb, Edel Gleeson, Gearóid O’Treasaigh, Girish P. Baxi, Graham O’Keeffe, Ishfak Bhagat, Jaitirth Shirole, Jeff deRienzo, Jeff Eisen, Jim Cooper, Jim Quill, John Grosjean, John Mackey, Kathy Howard, Lorcan McDonald, Margaret Rora, Matthew Flaherty, Mike Kerrigan, Maureen Leland, Na Pei, Peter Rubinstein, Russ Holden, Santosh Kumar, Simon Butcher, Simon Hewett, Srinivas Rao, Steve Castledine, Steve Leland, Thomas Gumz, Tom Carriker, Willie Doran, Xi Pan Xiao, and Yao Zhang. Apologies to any IBMers accidentally omitted; let us know and we’ll be sure to include you in the reprints!

Acknowledgments
To our friends at IBM Press—in particular Mary Beth Ray, Chris Cleveland, and the Production team—your patience for our many fits and starts over the course of this edition is gratefully appreciated! And on the IBM side of that relationship, we echo those sentiments to Steven Stansel and Ellice Uffer.

Finally, a great big THANK YOU, as always, to our customers and business partners! You are a fantastic community to whom we owe so much for the success of XPages. We hope you enjoy this book and that it helps you in a practical everyday way. Keep building those apps!
About the Authors

The authors of this book have a number of things in common. All three hail from Ireland, work for the IBM Ireland software lab, and have made significant contributions to the development of XPages over the past number of years.

**Martin Donnelly** is a software architect and technical lead for the Domino Designer and XPages teams in IBM Ireland. He has worked on all XPages releases to date and also on a precursor technology known as XFaces. Martin was also a development contributor to such products as the IBM Java Visual editor and IBM Rational® Application Developer. In the 1990s while living and working in Massachusetts, Martin was a lead developer on Domino Designer; this has now gone full circle as he rejoined the Domino Designer team in 2013 to head up the 9.0.1 release. Martin lives in Cork with his wife Aileen, daughters Alison, Aisling, and Maeve, and retired greyhounds Evie and Chelsea. Outside of work his main leisure time pursuits are soccer, fishing, and gardening.

**Mark Wallace** is a software architect in the IBM Ireland software lab. In the past, he worked on the XFaces runtime, which was developed for Lotus® Component Designer and subsequently evolved into the XPages runtime. He has a keen interest in programming models and improving developer productivity. Mark has worked in Lotus and IBM for more than 17 years on various products, and he is currently leading the Social Business Toolkit open source project. Mark lives in Dublin with his wife and two children and spends as much time as possible in the Ireland’s sunny south east enjoying fishing and kayaking with his family.

**Tony McGuckin** is a senior software engineer in the IBM Ireland software lab. Having studied software engineering at the University of Ulster, he began his career with IBM in 2006 and joined the XPages core runtime team shortly thereafter. When not directly contributing to the core runtime, Tony is busy with software research and development for the next generation of application development tooling, most recently focusing on mobile and responsive design. Tony
also spends a lot of time directly engaging with IBM customers as an XPages consultant, where he shows his flair for UI development and his deep understanding of application performance. Tony enjoys spending time with his wife and daughter, and getting out into the great outdoors for hill walking and the occasional chance to do some hunting in the surrounding hillsides of his native County Derry.

**Contributing Author**

**Jim Quill** is a development manager on the IBM Connections team in IBM Ireland. He has been with IBM for 5 years, and in this time he has managed to pack a lot in. He started on the XPages team and helped deliver several 8.5 releases. He then moved to the Connections Mail team where he was the technical lead role for the integration of Connections and Domino. He has most recently just taken up a new challenge—switching to development management. Previous to IBM, Jim enjoyed more than 13 years at Oracle Ireland. There, he worked in areas such as product development and database migration technology, and he was both principal software engineer and technical architect for a number of internal Oracle® support systems. Jim lives in the coastal village of Malahide, north County Dublin, with his wife and four children. When not acting as the kids’ taxi, he continues to play competitive basketball...way past his retirement date.
CHAPTER 14

XPages Mobile Application Development

The mobile features from the XPages Extension Library were promoted into the core XPages runtime in Domino 9.0. This reflects the importance of mobile support in application development as mobile devices (phones and tablets) move to outsell desktop systems and are becoming the norm for how people access web applications. XPages has implemented a Mobile Web Development strategy—that is, it uses web technologies to provide mobile access to your applications. Mobile devices feature powerful web browsers; however, the web interface you have built for desktop clients just won’t cut it for mobile clients. If you have ever accessed the full version of a website from a mobile device, you will have experienced first-hand the type of problems encountered when there is no mobile version of a site. These include

- **Limited resources:** Device processor power, memory, and network bandwidth all tend to be limited on a mobile device.
- **User experience:** Users have particular expectations when using a mobile device—for example, fast response times, navigation to most important features, minimal data entry, UI adapts to device orientation, and many more.
- **Limited functionality:** Users typically need only a subset of functionality and expect applications to reuse functionality from other applications on their mobile device.

The *XPages Extension Library* book provides an introductory description of the XPages mobile controls and the pattern to be used to develop a Create, Read, Update, Delete (CRUD) mobile sample application. The approach this chapter takes is to focus on best practices and design patterns for XPages Mobile Application Development. So even if you are familiar with building mobile applications with XPages, this chapter contains some discussion that you will find interesting. While writing the second edition of this book, Domino 9.0.1 had just been released. It includes some important enhancements for mobile developers, which will be covered in this chapter. For an excellent description of the best practices for Mobile Web Applications, visit [www.w3.org/TR/mwabp/](http://www.w3.org/TR/mwabp/). Some of the best practices outlined in this document
are referred to later in the chapter. Be sure to download the **chp14ed2.nsf** file 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).

**Tip**

There is also an introductory tutorial available at [www-10.lotus.com/ldd/ddwiki.nsf/dx/XPages_Mobile_Controls_Tutorial](http://www-10.lotus.com/ldd/ddwiki.nsf/dx/XPages_Mobile_Controls_Tutorial).

---

### Getting Started with Mobile Application Development

Start with a simple XPage that displays the browser User Agent string. This enables you to detect which device is accessing your application. When you preview the XPage shown in Listing 14.1 in a browser, it displays the User Agent string for your browser, which for Firefox version 18.02 is Mozilla/5.0 (Windows NT 6.1; WOW64; rv:18.0) Gecko/20100101 Firefox/18.0.

#### Listing 14.1  Display User Agent

```xml
<?xml version="1.0" encoding="UTF-8"?>
    style="padding-top:20.0px;padding-right:10.0px;padding-left:20.0px">
    <h3>Mobile XPages Applications</h3>
    <xp:br></xp:br>
    User Agent:&amp;#160;
    <xp:text escape="true" id="computedField1"
        value="#{javascript:context.getUserAgent().getUserAgent()}'" style="color:rgb(128,0,0)"
    ></xp:text>
</xp:view>
```

As I write this, I’m using my home wireless network—so now I can enter the URL for this page into the Safari browser on my iPhone, and I can see the page rendered there. The User Agent string displayed is

Mozilla/5.0 (iPhone; CPU iPhone OS 6_1 like Mac OS X) AppleWebKit-Kit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10B143 Safari/8536.25

Figure 14.1 shows how the page displays on an iPhone.
Straight away I can see problems:

- Typing a URL on my iPhone is painful; I don’t want to have to do a lot of typing using this device.
- I need to pinch and zoom to see the text in the browser. By default I’m seeing the full page with some tiny text at the top, which is not readable.
- This is going to slow down my development if I have to keep switching between my development machine and device to test my changes.
- What if I don’t have an iPhone, an iPad, or an Android device?

The first thing to know is that you don’t need a device to get started with Mobile Application Development. There are a number of alternative options to testing on a real device:

- Using a device emulator, these are typically part of a mobile platform SDK and are available for Mac, Android, Microsoft, and Blackberry devices.
- You can modify the User Agent in your desktop browser.
Most of the demonstrations in this chapter use the technique of overriding the User Agent your desktop browser sends with each request. User Agent spoofing doesn’t provide 100 percent fidelity with the actual device but is a quick way to get your application built before you begin testing on real devices. The remainder of this chapter uses the Safari and Chrome browsers to emulate Apple and Android devices, respectively.

**Safari Browser**

The Windows versions of the Safari browser are available from the Apple support site. The Web-Kit engine used by the Safari for Windows browser is similar to the one on the Apple iPhone and iPad, so this browser is a good option for basic emulation of the Apple mobile devices. Use the following steps to override the User Agent string sent by the browser:

1. If you do not have menus enabled by default, do so via the Show Menu Bar from the General Safari Settings toolbar drop-down.
2. Open Preferences and go to the Advanced tab.
3. Select the option to Show Develop menu in menu bar.
4. Select the User Agent override you want to use from the Develop -> User Agent menu.

Figure 14.2 shows the Safari User Agent choices. You can now select one of these. If you access the XPage in the Safari browser, the page displays the appropriate User Agent string.

<table>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default (Automatically Chosen)</td>
</tr>
<tr>
<td>Safari 5.1.6 — Mac</td>
</tr>
<tr>
<td>Safari 5.1.6 — Windows</td>
</tr>
<tr>
<td>Safari iOS 4.2.2 — iPhone</td>
</tr>
<tr>
<td>Safari iOS 4.3.3 — iPod touch</td>
</tr>
<tr>
<td>Safari iOS 4.3.3 — iPad</td>
</tr>
<tr>
<td>Internet Explorer 9.0</td>
</tr>
<tr>
<td>Internet Explorer 8.0</td>
</tr>
<tr>
<td>Internet Explorer 7.0</td>
</tr>
<tr>
<td>Firefox 4.0.1 — Mac</td>
</tr>
<tr>
<td>Firefox 4.0.1 — Windows</td>
</tr>
<tr>
<td>Opera 11.11 — Mac</td>
</tr>
<tr>
<td>Opera 11.11 — Windows</td>
</tr>
<tr>
<td>Other...</td>
</tr>
</tbody>
</table>

**Figure 14.2** Safari User Agent choices
For example, I selected Safari iOS 4.3.3—iPhone, and the XPage displayed the following User Agent:
Mozilla/5.0 (iPhone; U; CPU iPhone OS 4_3_3 like Mac OS X; en-us)
AppleWebKit/533.17.9 (KHTML, like Gecko) Version/5.0.2 Mobile/8J2 Safari/6533.18.5

**Chrome Browser**

The Chrome browser provides similar functionality as follows:

1. Go to the **Tools** menu and select **Developer tools**.
2. Select the **Settings** (cogged wheel) icon in the bottom-right corner of the **Developer tools** panel.
3. Select the **Overrides** tab.

Figure 14.3 shows the Chrome developer tools Overrides tab. You can override the User Agent and also other settings like the device metrics and orientation. The device metrics and orientation and useful for giving you that immediate feedback on how your page will be rendered on the device.
Firefox Browser

There is an add-on for FireFox called User Agent Switcher (addons.mozilla.org/en-US/firefox/addon/user-agent-switcher/), which provides the equivalent functionality. One nice feature of this add-on includes the capability to define your own User Agent string.

User Agent Device Detection

Detecting that your application is being accessed using a mobile device is important because it allows you to use server-side logic to adapt the content for the requesting client. It is preferable to do the adaptation on the server-side because this will improve the user experience and prevent the transfer of unnecessary data. The User Agent is typically used to detect the device in use. Listing 14.2 shows an example of how to detect if the device is an iPhone, iPad, Android, or BlackBerry device.

Listing 14.2  User Agent Device Detection

```xml
<?xml version="1.0" encoding="UTF-8"?
    style="padding-top:20.0px;padding-right:10.0px;padding-
➥left:20.0px">
    <h3>Mobile XPages Applications</h3>
    <xp:br/>
    User Agent:
    <xp:text escape="true" id="computedField1"
        value="#{javascript:context.getUserAgent().getUserAgent()}"
        style="color:rgb(128,0,0)">
    </xp:text>
    <xp:br/>
    Device:
    <xp:text escape="true" id="computedField2"
        style="color:rgb(128,0,0)">
        <![CDATA[#{javascript:var ua = context.getUserAgent().getUserAgent();
            if (ua.indexOf("iPhone") > -1) {
                return "iPhone";
            } else if (ua.indexOf("iPad") > -1) {
                return "iPad";
            } else if (ua.indexOf("Android") > -1) {
                return "Android";
            } else if (ua.indexOf("BlackBerry") > -1) {
                return "BlackBerry";
            } else {]]>}
    </xp:text>
</xp:view>
```


Given the large number of devices in use and that new devices come to the market frequently, this type of coding can become complex. The best practice for mobile applications is to use broader device classification to simplify the process of adapting your content. For example, you might want to generate different content for mobile phones versus tablet devices. In Domino 9.0.1, a new managed bean called the `deviceBean` has been added to the XPages runtime to simplify this process and allow you to implement a device classification strategy.

**Device Bean**

The Device Bean is used to identify the most common mobile and tablet devices—that is, Android; Apple iPhone or iPad; Blackberry; or Windows Mobile devices. The heavy lifting of parsing the User Agent string is handled for you. The most commonly used methods are `deviceBean.isMobile()` and `deviceBean.isTablet()`. For tablet devices, the method `deviceBean.isMobile()` returns false, which means you often see the two values being OR'd to determine if any mobile device is used. Listing 14.3 shows a list of the values available from the Device Bean.

**Listing 14.3  DeviceBean**

```xml
<?xml version="1.0" encoding="UTF-8"?>

    deviceBean.isMobile=<xp:text value="#{javascript:deviceBean.isMobile()}"/>
    <span class="desc">Identifies a device as a mobile device.</span>

    deviceBean.isTablet=<xp:text value="#{javascript:deviceBean.isTablet()}"/>
    <span class="desc">Identifies a device as a tablet device.</span>

    deviceBean.isIphone=<xp:text value="#{javascript:deviceBean.isIphone()}"/>
    <span class="desc">Identifies a device as an iPhone.</span>

    deviceBean.isIpad=<xp:text value="#{javascript:deviceBean.isIpad()}"/>
    <span class="desc">Identifies a device as an iPad.</span>

</xp:view>
```
Figure 14.4 shows the values that are displayed when accessing this page with the User Agent set to that of an iPad.

![DeviceBeanTable.xsp]

**Figure 14.4  iPad Device Bean Values**

Now we have an easy to use technique to identify that a mobile or tablet device is accessing our application we can use this information to adapt the presentation of our application to a form that is suitable for the device being used. The design pattern used to present content suitable for use in mobile applications is the Single Page Application pattern and this is the topic for the next section.
Tip
The device bean is cached for the lifetime of a user session, which means if you are using a browser plug-in to change the user agent \textit{Agent} you actually need to restart the browser for a change to take effect. Starting in Domino 9.0.1 there is a new mobile property in the Xsp properties called \textit{Debug user agent}. This allows you to specify either an iOS or Android User Agent, and the change will take effect the next time you load a page from that application.

Single Page Application Design Pattern
The Single Page Application design pattern was created to address a number of Mobile Application Development best practices related to the conservative use of resources and improving user experience. Consider the following best practices that aim to improve the user experience:

- Bandwidth is typically more constrained on mobile networks; therefore, fewer but larger requests are recommended.
- Application start-time needs to be optimized.
- View switching must be fast and must support bookmarking and back button navigation.

One way to support these best practices is to load the application views either statically or dynamically without requiring a full page reload. Loading additional views is recommended to reduce the number of requests and provide for fast view switching. Associate fragment identifiers with each view, and use these for navigation and bookmarking. XPages includes the Single Page Application Control (\texttt{xe:singlePageApp}), which encapsulates all this functionality for you and provides the standard XPages declarative interface to allow you to configure and control its behavior. This and the other XPages mobile controls are available in their own category—that is, Mobile, within the Controls Palette. Figure 14.5 shows the Mobile controls category.
Before creating any XPages that use the Mobile controls, there is some configuration required to get the pages to render correctly.

**Mobile XPage Properties**

There is a specific theme for use in mobile XPages. This theme is required for the Mobile controls to render with the correct device look and feel. This can be enabled automatically by specifying a prefix for XPages that should use the mobile theme. The standard prefix is `m_` but you can specify your own—for example, `mobile_` is used in the Discussion template. To enable the mobile theme based on a page prefix, use these steps:

1. Go to **Application Configuration > Xsp Properties**.
2. Select the option to **Use mobile theme for XPages with the prefix**.
3. Optionally specify the prefix to use.

Figure 14.6 shows the Mobile XPage properties.

All mobile pages must have the prefix you specified; otherwise, they will include the default theme and won’t render with the device look and feel. Mobile themes will be explored later in this chapter in the section, “Mobile Themes.”

In addition to the mobile theme prefix setting, there are additional mobile properties that can be configured:
• **Mobile theme:** Enables you to specify the mobile theme to use if you want to change from the Mobile default theme.

• **Override on iOS:** Enables users to specify a specific theme for iOS devices.

• **Override on Android:** Enables users to specify a specific theme for Android devices.

• **Debug user agent:** Enables users to override mobile device detection behavior and force all mobile pages to render as either iOS or Android.

Now you are ready to create some mobile specific XPages.

**Single Page Application Control (xe:singlePageApp)**

The Single Page Application Control enables you to define your entire application behavior within a single XPage. The control contains a collection of views that can either be loaded as part of the initial page request or can be dynamically retrieved as needed. Listing 14.4 shows the XPages markup for a mobile page using the Single Page Application control.

**Listing 14.4 Mobile XPage**

```xml
<?xml version="1.0" encoding="UTF-8"?>
    xmlns:xe="http://www.ibm.com/xsp/coreex"
    xmlns:xc="http://www.ibm.com/xsp/custom">
    <xe:singlePageApp id="singlePageApp1" selectedPageName="appPage1">
        <xe:appPage id="appPage1" pageName="appPage1">
            <xe:djxmHeading id="djxmHeading1">
                <xe:this.label>Page 1</xe:this.label>
            </xe:djxmHeading>
            <xe:djxmLineItem id="djxmLineItem1" label="Go to Page 2"
                moveTo="appPage2">
            </xe:djxmLineItem>
        </xe:appPage>
        <xe:appPage id="appPage2" pageName="appPage2">
            <xc:mobile_appPage2/>
        </xe:appPage>
    </xe:singlePageApp>
</xp:view>
```

This XPage includes two Application Pages (xe:appPage), the first of which is included inline in the page. The second Application Page is included via a custom control and this is the recommended pattern to use. The XPage that includes the Single Page Application control can quickly become complex and difficult to maintain if all the pages are included inline. If you look in the Discussion template at the mobile.xsp XPage, you can see another example of this pattern being used. The sample in Listing 14.4 is available in the database that accompanies this chapter in...
two XPage design elements, one named m_SinglePageApplicationControl and one named SinglePageApplicationControl. If you run these samples, only the one with the m_prefix will display correctly with the mobile theme.

**Tip**
The id parameter is required on each Application Page (xe:appPage) control because the default behavior is to dynamically load each page when it is displayed. You can use the preload property to have a page load when its parent is being loaded. This will improve performance for page transition at the cost of a larger initial download.

Figure 14.7 shows the sample running with the mobile theme on an iPhone.

![Figure 14.7 Single Page Application Control on iPhone](image)

It is recommended you review the user interface guidelines for the devices you are targeting to ensure your application fits in on that device. These guidelines provide some basic principles for any mobile application and also specific details on designing for a specific version of a mobile operating system. The next section looks at how the XPages Mobile Controls enable you to build an application that adheres to the best practices in Mobile Application Navigation.

**Mobile Application Navigation**
Users should always know where they are within your application and how to get back to where they have just come from. The views in your application can be organized in different ways. For example, you could implement a navigator that allows the user to jump straight to a particular place in the application. This is best-suited where the application views are organized in a simple
flat list. In a mobile application you may not have the luxury of displaying the navigator all the
time, so the ability to quickly get back to the main navigator is important. Another common
approach is to use a menu-like hierarchical navigation scheme in which users select different
options to navigate through the hierarchy. When using a hierarchical scheme, a Back button is
important to allow users to quickly retrace their steps. The final approach you look at is provid-
ing context-sensitive navigation options. In this case, users will be provided with the navigation
options that logically make sense based on where they are and what they are doing within the
application.

You saw an example of how to do navigation using the Rounded List Item
(xe:djxmLineItem) control earlier in this chapter in Listing 14.4. The Rounded List Item con-
trol enables you to specify the page name to move to but also the transition type to use. The valid
types of transition are

- **slide**: New pages moves in from the side to cover the old page. This is the default.
- **fade**: The old page fades out while the new page fades in.
- **flip**: The old page flips over to display the new page as if the new page were printed on
  the back of the old page.
- **none**: The new page appears immediately without any transition effect.

Listing 14.5 shows how to specify the transition type that will be used when a user changes
the Application Page.

**Listing 14.5  Application Page Transitions**

```xml
<?xml version="1.0" encoding="UTF-8"?>
  xmlns:xe="http://www.ibm.com/xsp/coreex"
  xmlns:xc="http://www.ibm.com/xsp/custom">
  <xe:singlePageApp id="singlePageApp1" selectedPageName="appPage1">
    <xe:appPage id="appPage1" pageName="appPage1">
      <xe:djxmHeading id="djxmHeading1">
        <xe:this.label>Page 1</xe:this.label>
      </xe:djxmHeading>
      <xe:djxmLineItem id="djxmLineItem1" label="Fade to Page 2"
        moveTo="appPage2" transition="fade">
      </xe:djxmLineItem>
    </xe:appPage>
    <xe:appPage id="appPage2" pageName="appPage2">
      <xe:djxmHeading id="djxmHeading2">
        <xe:this.label>Page 2</xe:this.label>
      </xe:djxmHeading>
      <xe:djxmLineItem id="djxmLineItem2" label="Flip to Page 1"
        transition="flip">
      </xe:djxmLineItem>
    </xe:appPage>
  </xe:singlePageApp>
</xp:view>
```
To implement a basic navigator, you need to use the Page Heading (xe:djxmHeading) and Rounded List Item (xe:djxmLineItem) controls. Listing 14.6 shows an XPage that provides a basic navigator. The Page Heading shows users where they are within the application at all times. Each Page Heading includes two additional attributes:

- **back**: Label for the back button
- **moveTo**: Page to move to when the Back button is selected

The combination of these two parameters adds a Back button to each page, which allows users to return to the Home page, which includes the Navigator with a single-click. The main Navigator displays on its own Application Page to optimize real estate on the device.

**Listing 14.6 Navigator XPage**

```xml
<?xml version="1.0" encoding="UTF-8"?>
         xmlns:xe="http://www.ibm.com/xsp/coreex"
         xmlns:xc="http://www.ibm.com/xsp/custom">
  <xe:singlePageApp id="navigationApp" selectedPageName="homePage">
    <xe:appPage id="homePage" pageName="homePage">
      <xe:djxmHeading id="djxmHeading1">
        <xe:this.label>Home</xe:this.label>
        <xe:djxmLineItem label="Visitor Info" moveTo="visitorPage"/>
        <xe:djxmLineItem label="Conservation" moveTo="conservePage"/>
        <xe:djxmLineItem label="Education" moveTo="educatePage"/>
        <xe:djxmLineItem label="Get Involved" moveTo="involvePage"/>
        <xe:djxmLineItem label="Shop" moveTo="shopPage"/>
      </xe:djxmHeading>
    </xe:appPage>
    <xe:appPage id="visitorPage" pageName="visitorPage">
      <xe:djxmHeading back="Home" moveTo="homePage">
        Visitor Info
      </xe:djxmHeading>
    </xe:appPage>
  </xe:appPage>
</xe:singlePageApp>
</xp:view>
```
Figures 14.8 and 14.9 show the Home page with the XPages Navigator and also an Application Page, which includes a Page Heading with an integrated Back button to return to the Home page.

**Figure 14.8** Mobile Navigator
Hierarchical Navigation

Hierarchical navigation can be implemented with the Outline (xe:outline) and associated Node (xe:basicContainerNode and xe:basicLeafNode) controls. Listing 14.7 shows a custom control that supports hierarchical navigation for a Single Page Application. The Outline has a collection of Container Nodes, which in turn have a collection of children that are Basic Nodes and represents the leaves in the tree structure. Selecting one of the Basic Nodes will trigger navigation to the associated Application Page within the Single Page Application.

**Listing 14.7 Outline Custom Control**

```xml
<?xml version="1.0" encoding="UTF-8"?>
          xmlns:xe="http://www.ibm.com/xsp/coreex">
  <xe:outline id="outline1">
    <xe:this.treeNodes>
      <xe:basicContainerNode label="Services">
        <xe:this.children>
          <xe:basicLeafNode label="Business" href="#businessPage"/>
          <xe:basicLeafNode label="Training" href="#trainingPage"/>
        </xe:this.children>
      </xe:basicContainerNode>
      <xe:basicContainerNode label="Products">
        <xe:this.children>
          <xe:basicLeafNode label="Software" href="#softwarePage"/>
        </xe:this.children>
      </xe:basicContainerNode>
    </xe:this.treeNodes>
  </xe:outline>
</xp:view>
```
Listing 14.8 shows the Single Page Application that uses the Outline custom control.

**Listing 14.8 Hierarchical Navigation XPage**

```xml
<?xml version="1.0" encoding="UTF-8"?>
            xmlns:xe="http://www.ibm.com/xsp/coreex"
            xmlns:xc="http://www.ibm.com/xsp/custom">
    <xe:singlePageApp id="navigationApp" selectedPageName="outlinePage">
        <xe:appPage id="outlinePage" pageName="outlinePage">
            <xc:Outline/>
        </xe:appPage>
        <xe:appPage id="businessPage" pageName="businessPage">
            <xe:djxmHeading back="Services" moveTo="outlinePage">
                Business</xe:this.label>
            </xe:djxmHeading>
        </xe:appPage>
        <xe:appPage id="trainingPage" pageName="trainingPage">
            <xe:djxmHeading back="Services" moveTo="outlinePage">
                Training</xe:this.label>
            </xe:djxmHeading>
        </xe:appPage>
        <xe:appPage id="softwarePage" pageName="softwarePage">
            <xe:djxmHeading back="Products" moveTo="outlinePage">
                Software</xe:this.label>
            </xe:djxmHeading>
        </xe:appPage>
        <xe:appPage id="systemsPage" pageName="systemsPage">
            <xe:djxmHeading back="Products" moveTo="outlinePage">
                Systems</xe:this.label>
            </xe:djxmHeading>
    </xe:singlePageApp>
</xp:view>
```
Context-Sensitive Navigation

Context-sensitive navigation means that your application provides users with the navigation options that represent the next logical steps within the application. It is important not to overload users with too many navigation options because this can make your application difficult to use, and mobile application users have a very low tolerance for difficult-to-use applications.

Listing 14.9 shows a mobile XPage with four application pages used for home, start, settings, and advanced functionality. In this scenario, the advanced functionality is only accessible from the Settings page. The intention here is to hide complexity, but another reason to do this is that it might only make sense to make screens available after another operation—for example, after the start functionality.

**Listing 14.9  Context-Sensitive Navigation XPage**

```xml
<?xml version="1.0" encoding="UTF-8"?>
    xmlns:xe="http://www.ibm.com/xsp/coreex"
    xmlns:xc="http://www.ibm.com/xsp/custom">
    <xe:singlePageApp id="contextSensitiveApp"
        selectedPageName="homePage">
        <xe:appPage id="homePage" pageName="homePage">
            <xc:TabBar pageName="home"></xc:TabBar>
            Home Page
        </xe:appPage>
        <xe:appPage id="startPage" pageName="startPage">
            <xc:TabBar pageName="start"></xc:TabBar>
            Start Page
        </xe:appPage>
        <xe:appPage id="settingsPage" pageName="settingsPage">
            <xc:TabBar pageName="settings"></xc:TabBar>
            Settings Page
        </xe:appPage>
        <xe:appPage id="advancedPage" pageName="advancedPage">
            <xc:TabBar pageName="advanced"></xc:TabBar>
            Advanced Page
    </xe:singlePageApp>
</xp:view>
```
The navigation for this application is implemented in the page heading using a segmented button list, as shown in Figure 14.10.

![Figure 14.10 Heading with segmented buttons](image)

The functionality is encapsulated in a custom control, the code for which is demonstrated in Listing 14.10. A Page Heading with a nested Tab Bar is used to implement this navigation strategy. The custom control takes a single parameter that is the name of the page currently displayed. Each Tab Bar button uses a computed rendered property to determine if it should display. The button for the advanced screen displays only when the Settings screen is active.

**Listing 14.10  Context-Sensitive Navigation Custom Control**

```xml
<?xml version="1.0" encoding="UTF-8"?>
         xmlns:xe="http://www.ibm.com/xsp/coreex">
  <xe:djxmHeading>
    <xe:tabBar id="tabBar1"
               style="background-color:rgb(255,255,255);width:100%;"
Another commonly used way to implement this pattern is to use a Tab Bar with a list of icons for navigation displayed at the bottom of the screen.

Now you have seen how to navigate around your application using a number of commonly used patterns. There may be times in which you need to write script that responds to transition events—for example, to prompt the user to perform some action with values entered on the current page. Starting in the 9.0.1 release, XPages also supports new touch-based events:

- **onBeforeTransitionIn**: Triggered before transitioning into a page
- **onAfterTransitionIn**: Triggered after transitioning into a page
- **onBeforeTransitionOut**: Triggered before transitioning out of a page
- **onAfterTransitionOut**: Triggered after transitioning out of a page

The XPage in Listing 14.11 shows how to block transition out of a page.
Interacting with a Mobile Application

Mobile devices introduce some new interaction methods that you need to consider during the development of your application. XPages provides additional mobile events to support the following mobile specific interaction methods:

- Orientation-based
- Touch-based
- Multitouch-based

Orientation-Based Interaction

The optimum layout of the UI can vary depending on the device orientation. Your application should respond to orientation change events and adapt the UI accordingly. If this is not possible,
the UI should be designed to provide a good user experience for each orientation. CSS provides functionality that allows you to control the presentation based on the media type; the `@media` rule can be used to optionally hide content depending on the orientation. Listing 14.12 shows some CSS that defines a style class that enables you to hide elements in portrait orientation. The `@media` rule is used to define different values for the style class based on the current orientation, which hides it when in portrait mode.

**Listing 14.12  Landscape-Only Display**

```css
@media only screen and (orientation:portrait) {
  .landscape-only {
    display: none;
  }
}
@media only screen and (orientation:landscape) {
  .landscape-only {
    display: block;
  }
}
```

Listing 14.13 shows an XPage that uses the CSS in Listing 14.12. The Application Page has a control that displays only in landscape orientation. When you run this sample and switch the device orientation, the Optional Stuff shows or hides automatically.

**Listing 14.13  Landscape-Only Display Using CSS**

```xml
<?xml version="1.0" encoding="UTF-8"?>
         xmlns:xe="http://www.ibm.com/xsp/coreex">
  <xp:this.resources>
    <xp:styleSheet href="/orientation.css"></xp:styleSheet>
  </xp:this.resources>
  <xe:singlePageApp selectedPageName="orientationPage">
    <xe:appPage id="orientationPage" pageName="orientationPage">
      <xe:djxmHeading id="djxmHeading3">
        <xe:this.label>Orientation</xe:this.label>
      </xe:djxmHeading>
      <xe:djxmRoundRectList id="djxmRoundRectList1">
        Required Stuff
      </xe:djxmRoundRectList>
      <xe:djxmRoundRectList id="djxmRoundRectList2" styleClass="landscape-only">
        Optional Stuff
      </xe:djxmRoundRectList>
    </xe:appPage>
  </xe:singlePageApp>
</xp:view>
```
Listing 14.14 shows a further refinement on this technique to define separate style sheets for each orientation and then uses the media attribute on the `xp:styleSheet` tag to determine which style sheet is used.

**Listing 14.14  Landscape-Only Display Using Style Sheets**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
    xmlns:xe="http://www.ibm.com/xsp/coreex">
  <xp:this.resources>
    <xp:styleSheet media="only screen and (orientation:portrait)"
        href="/portrait.css"></xp:styleSheet>
    <xp:styleSheet media="only screen and (orientation:landscape)"
        href="/landscape.css"></xp:styleSheet>
  </xp:this.resources>
  <xe:singlePageApp selectedPageName="orientationPage">
    <xe:appPage id="orientationPage" pageName="orientationPage">
      <xe:djxmHeading id="djxmHeading3">
        <xe:this.label>Orientation</xe:this.label>
      </xe:djxmHeading>
      <xe:djxmRoundRectList id="djxmRoundRectList1">
        Required Stuff
      </xe:djxmRoundRectList>
      <xe:djxmRoundRectList id="djxmRoundRectList2"
        styleClass="landscape-only">
        Optional Stuff
      </xe:djxmRoundRectList>
    </xe:appPage>
  </xe:singlePageApp>
</xp:view>
```

Starting with the 9.0.1 release XPages now includes an `onOrientationChange` event on the Single Page Application control. You can write client-side application logic, which is executed when the user changes the device orientation. Listing 14.15 shows some client-side JavaScript, which is called in response to an orientation change. The `orientation` property can have the following values:
• 0: Portrait mode
• 90: Landscape mode with the screen turned to the left
• -90: Landscape mode with the screen turned to the right
• 180: Portrait mode with the screen upside down

Not all devices support all the modes, for example, iOS devices do not support 180. Also notice in this sample that the initial value is set to Unknown. It is not possible to write a server-side computed expression that computes the current orientation. Because of network latency, the user could have changed the device orientation between requesting the page and it displaying. Therefore, client-side computations need to be used when developing orientation-based logic.

**Listing 14.15 onOrientation Change Event**

```xml
<?xml version="1.0" encoding="UTF-8"?>
         xmlns:xe="http://www.ibm.com/xsp/coreex">
    <xe:singlePageApp selectedPageName="orientationChangePage"
                     id="singlePageApp1">
        <xe:appPage id="orientationChangePage"
                    pageName="orientationChangePage">
            <xe:djxmHeading id="djxmHeading3">
                <xe:this.label>OnOrientationChange</xe:this.label>
            </xe:djxmHeading>
            <xp:label value="Unknown" id="label1"></xp:label>
        </xe:appPage>
    </xe:singlePageApp>
</xp:view>

<![CDATA[
var label = document.getElementById("view:_id1:orientationChangePage_content:label1");
if (orientation == 0) {
    label.innerHTML = "Portrait Mode {orientation=0}";
}
else if (orientation == 90) {
    label.innerHTML = "Landscape Mode {orientation=90}";
}
else if (orientation == -90) {
    label.innerHTML = "Landscape Mode {orientation=-90}";
}
else if (orientation == 180) {
    label.innerHTML = "Portrait Mode {orientation=180}";
}
]]>
```
Touch-Based Interaction

Most mobile devices now support the ability for users to interact using a finger or stylus. When designing a UI for a touch-based interaction, controls should be positioned and sized so users can individually select them. Controls that can be selected need to be large enough so that they can be easily selected and it is clear which item is currently selected. Remember part of the screen may be obscured by users’ fingers as they select items, so you need to make sure that you provide feedback (for example, use roll-overs) to indicate what item is currently selected. Also, you need to avoid frustrating users if they cannot easily select a particular option. Listing 14.16 shows an example of interacting with touch-based events when using the mobile switch.

Listing 14.16  onTouchStart and onTouchEnd Events

```xml
<?xml version="1.0" encoding="UTF-8"?>
xmlns:xe="http://www.ibm.com/xsp/coreex">
  <xe:singlePageApp selectedPageName="transitionPage">
    <xe:appPage id="transitionPage" pageName="transitionPage">
      <xe:djxmHeading id="djxmHeading3">
        <xe:this.label>Transition</xe:this.label>
      </xe:djxmHeading>
      <xp:label value="Swipe Me: " id="label1"></xp:label>
      <xe:djxmSwitch leftLabel="ON" rightLabel="OFF" id="djxmSwitch1">
        <xp:eventHandler event="onTouchStart" submit="false">
          <xe:this.script>
            <![CDATA[
              var label1Id = '#{javascript:getClientId("label1")}';
              var label1 = document.getElementById(label1Id);
              label1.innerHTML = "--- Swiping ---";
            ]]>
          </xe:this.script>
        </xp:eventHandler>
        <xp:eventHandler event="onTouchEnd" submit="false">
          <xe:this.script>
            <![CDATA[
              var label1Id = '#{javascript:getClientId("label1")}';
              var label1 = document.getElementById(label1Id);
            ]]>
          </xe:this.script>
      </xe:djxmSwitch>
    </xe:appPage>
  </xe:singlePageApp>
</xp:view>
```
Multitouch-Based Interaction

A multitouch interaction is the ability of the touch screen to detect the presence of multiple points of contact. One of the most common multitouch gestures is pinch-to-zoom, which allows the user to zoom in and out. When designing an XPage that will be accessed on a mobile device, you must take care to ensure users don’t need to zoom when they first view the page. You may have some XPages that you don’t want to or have time to convert to a mobile design, but you can still improve the experience for mobile users by making sure the pages render with the optimum zoom when accessed with a mobile device. The way you do this is to use the viewport meta tag. Listing 14.17 shows how to set the viewport for an XPage using the \texttt{xp:metaData} tag with a width of 500. (A further enhancement would be to compute this value based on the specific device.)

\begin{Verbatim}
\verb|<?xml version="1.0" encoding="UTF-8"?>|
\verb|<xp:view xmlns:xp="http://www.ibm.com/xsp/core">|
\verb|  <xp:this.resources>|
\verb|    <xp:metaData name="viewport" content="width=500">|
\verb|      <xp:metaData>|
\verb|    </xp:this.resources>|
\verb|  <h1>Mobile XPages Applications</h1>|
\verb|  ...|
\verb|</xp:view>
\end{Verbatim}

Take, for example, the launch page (\texttt{index.xsp}) for this chapter’s sample database. Figure 14.11 shows how the page will display first with the default viewport width (which is 980 for an iPhone) and then with the viewport set as shown in Listing 14.17. The XPage is displayed in a more readable and usable manner by changing the width.

The viewport meta tag supports some other attributes that allow you to control how your XPages displays:

- \texttt{width}: Viewport width, that is, the width of the page a user sees.
- \texttt{height}: Viewport height, that is, the height of the page a user sees.
Mobile Themes

• **initial-scale**: Initial zoom scale of the viewport.
• **maximum-scale**: Maximum view scale of the viewport.
• **minimum-scale**: Minimum view scale of the viewport.
• **user-scalable**: Determines if the user is allowed to zoom in and out of the viewport.

![Without Viewport (a)](image1.png) ![With Viewport (b)](image2.png)

**Figure 14.11** With and without viewport

**Mobile Themes**

XPages comes with mobile theme support for iOS and Android devices. As explained earlier, you can configure a mobile theme for use with your mobile XPages by identifying those pages using a special prefix (typically `m_`). The default mobile theme is called Mobile default. When this option is selected, the appropriate theme is automatically selected for you, that is, the theme named iPhone (because of historical reasons) for iOS device or the theme named Android for Android devices. Each theme can cause mobile styles to be applied to selected controls, which give them a native look and feel when they are rendered on a mobile device. You can select an
alternative theme for mobile pages—for example, selecting the OneUI IDX v1.3 mobile theme provides a consistent look and feel between all mobile devices accessing your application. You can also define your own mobile theme; if, for example, you wanted to have your own look and feel irrespective of the mobile device. You can also specify separate themes for iOS and Android devices; again, this flexibility works if your application has its own branding. If you want to switch the style based on the device, you need to add logic within the theme. This is how the One UI theme works, and this will be further explored later in this section. For more detailed information on themes and how to create your own, refer to Chapter 16, “XPages Theming.”

Mobile styling is provided for all the mobile XPages controls and the following Extension Library controls:

- Data View (xe:dataView)
- Outline (xe:outline)
- Form Table (xe:formTable)

The XPage name m_mobileTheme in the sample database that accompanies this chapter includes pages with all these controls.

**Data View**

The Data View control is the alternative to using a regular View Panel control for mobile XPages. The Data View control optimizes the display of the rows of data it displays. Figure 14.12 shows a Data View configured to display the contents of a Domino view. Notice that it efficiently uses the available space. Also, rather than using a traditional pager, it retrieves extra rows on demand and allows the user to scroll through all the retrieved rows.

The Data View control can also navigate to another page within the single page application to display the document associated with a row in the view. As shown in Listing 14.18, this is achieved by setting the pageName property to the hash tag value of the application page to open.

**Listing 14.18  Data View Application Page**

```xml
<xe:appPage id="dataViewPage" pageName="dataViewPage">
  <xe:djxmHeading back="Home" moveTo="controlsPage">
    <xe:this.label>Data View</xe:this.label>
  </xe:djxmHeading>
  <xe:dataView id="dataView1" rows="10" pageName="#formTablePage">
    <xe:this.data>
      <xp:dominoView var="view1" viewName="CarMakes"></xp:dominoView>
    </xe:this.data>
    <xp:this.facets>
      <xp:link text="Show More" escape="true" xp:key="pagerBottom" id="link1">
      </xp:link>
    </xp:this.facets>
  </xe:dataView>
</xe:appPage>
```
Figure 14.12  Data View versus View Panel
Outline

The Outline control was introduced earlier in the chapter as a way to provide navigation within your Mobile application. Figure 14.13 shows the difference between the mobile and web styling. When used in a mobile XPage, the Outline control supports expanding and collapsing of nodes, which makes it suitable providing an application menu.

![Mobile Outline and Web Styling](image)

Figure 14.13 Outline with mobile and web styling

Form Table

The Form Table control is a useful container when you design data entry or display forms. The Form Table has a title and description displayed above the rows of data. Each row in the table has a label and one or more controls to display the row data. Figure 14.14 shows the difference in how the Form Table is styled for a mobile device or desktop web browser. If you inspect the
DOM for the two pages, you’ll notice that for a desktop web browser a table is used, but for a mobile browser, the Form Table Rows are created as div elements with appropriate styling. HTML tables are not a good choice for laying out content when you have limited screen real estate and should be avoided. The common problem when using HTML tables on mobile devices is that they result in excess whitespace, which wastes the valuable device real estate.

So now you’ve seen what you get for free for the default mobile themes, but what about the rest of the controls? The next section looks at the options for styling XPages controls.

Styling XPages Controls for Mobile Applications

Standard XPages controls don’t automatically change their styling when displayed as part of a mobile page using the standard mobile themes. So how do you style a standard XPages control so that it appears well on a mobile device? There are a number of approaches you can use. Take a button as an example use case and explore these options. Listing 14.19 shows an application page with five buttons, and Figure 14.15 shows how they display on a mobile device. The first button (lines 5 through 6) has no styling applied, and if you look at how this is displayed, you’ll see it’s not a good fit for what you would expect on a mobile device. That is, it’s too small and
the styling looks out of place. The second button (lines 7 through 8) has the styleClass set to mblButton, and this causes it to display well on a mobile device. This is a style class that is provided by the mobile theme. Ideally, this is all you would have to do, but unfortunately this is not the case because the style class is different between iOS and Android devices. The third button (lines 9 through 10) shows how to use the Android version of the mobile button style class. So what if you target multiple devices? One option, which is shown in the fourth button (lines 1 through 15) is to compute the appropriate style class to use based on the information from the device bean. This works but is awkward to use. What happens if you use Dojo and your buttons have a dojoType attribute set? This use case is shown in the fifth button (lines 16 through 17) and again no styling is applied so you have work to do, or do you?

Listing 14.19  Button Mobile Styling

```xml
1.  <xe:appPage id="buttonPage" pageName="buttonPage">
2.    <xe:djxmHeading back="Home" moveTo="controlsPage">
3.      <xe:this.label>Button</xe:this.label>
4.    </xe:djxmHeading>
5.    <xp:button value="Standard" id="button1">
6.    </xp:button>
7.    <xp:button value="Mobile" id="button2" styleClass="mblButton">
8.    </xp:button>
9.    <xp:button value="Android" id="button3" styleClass="mblButton_android">
10.   </xp:button>
11.   <xp:button value="Dynamic" id="button4">
12.     <xp:this.styleClass>
13.    <![CDATA[#{javascript:deviceBean.isAndroid() ? "mblButton_android" : "mblButton"}]]>
14.     </xp:this.styleClass>
15.   </xp:button>
16.   <xp:button value="Dojo" id="button5" dojoType="dijit.form.Button">
17.   </xp:button>
18.   <br />
19.   deviceBean.isIphone=
20.   <xp:text value="#{javascript:deviceBean.isIphone()}" />
21.   <br />
22.   deviceBean.isAndroid=
23.   <xp:text value="#{javascript:deviceBean.isAndroid()}" />
24.   <br />
25. </xe:appPage>
```
If instead of using the default mobile theme, you switch to using One UI the situation changes. Figure 14.16 shows how the same five buttons display when the One UI theme is selected as the mobile theme in the XPages Properties. So now things look much better; the standard and Dojo buttons both display well without having had to make any changes. This is the ideal situation; you can just add standard XPages controls and have them display well on mobile devices automatically.
So how does this work? If you inspect the DOM of the mobile page, you can notice that the standard and Dojo buttons are rendered in the page with the style class set to `mblButton` `mblPrimaryButton`. So these style classes are added automatically to all buttons that are rendered when the One UI theme is used. Listing 14.20 shows an extract from the One UI theme file (`oneui_idx_v1.3.theme`). You can see the One UI theme specifies the `lotusBtn` style class is specified for buttons.

**Listing 14.20  One UI Button Theme**

```xml
<!-- Basic Button -->
<control>
  <name>Button</name>
  <property>
    <name>styleClass</name>
    <value>lotusBtn</value>
  </property>
</control>
```

However, this isn’t the full story. There is another theme file used by One UI that is associated with the mobile renderers; this file is called `oneui_idx_v1.3_mobile_renderers_fragment.theme`. If you look at the contents of this file, you will see additional styling configuration, and it is here you see the mobile style classes applied, as demonstrated in Listing 14.21.

**Listing 14.21  One UI Button Mobile Theme**

```xml
<!-- Command Button -->
<control>
  <name>Button.Command</name>
  <property>
    <name>styleClass</name>
    <value> mblButton mblPrimaryButton</value>
  </property>
</control>
```

Using One UI is a good option if you want to have your applications display with a consistent look and feel and to automatically display well on mobile devices. It is recommended that you consider using OneUI by default when developing mobile applications, or indeed create your own theme in preference to adding lots of conditional styling logic within your XPages. Next, look at what to do when things go wrong when you develop a Mobile XPages application.

**Debugging Mobile XPages**

When something goes wrong with your mobile XPages, you need to debug them to diagnose the problem and figure out how to resolve the problem. For client debugging you will likely
have used a browser debugging tool like Firebug for Firefox or Web Inspector for Safari. In this section, you learn about two techniques you can use to debug Mobile XPages. There are other options, for example, there is a recently released Firebug Lite Bookmarklet for iPad, but these techniques described here are the ones favored by the book authors when debugging their Mobile XPages. There are two approaches described in the following sections, both of which rely on viewing the DOM hierarchy for your mobile XPage in real time on another device. The first approach is targeted at iOS mobile development, and the second approach is a more generic solution, which works irrespective of the mobile device you target.

**Debugging XPages on iOS**

If you have a Mac, you can use it to debug your Mobile XPages. The procedure is straightforward using the following steps:

1. Enable the Develop menu in the Advanced preferences.
2. Connect your mobile device to the Mac with a USB cable.
3. A new menu item appears in the Develop menu that enables you to inspect a page on your mobile device.

Figure 14.17 shows an example of the type of menu item that displays if you had an iPhone connected to your Mac.

![Develop menu](image)

**Figure 14.17** Develop menu item

When you select the menu item, the Web Inspector opens. The Web Inspector can be used to view the DOM of the page, which displays within the browser on the iPhone. Figure 14.18 shows the DOM for the `m_Debug.xsp` page.
As you select elements from within the DOM in Web Inspector, the equivalent element within the browser is highlighted. For example, if you select the span element that corresponds to the Toolbar button, the button is highlighted on the device, as shown in Figure 14.19.

In addition to viewing the DOM, you can also display a JavaScript console. Any logging statements that are output using the console JavaScript class can be viewed within Web Inspector. Figure 14.20 shows the JavaScript console with some text that was output when the Toolbar Button in the m_Debug.xsp XPage was clicked.
For more information on the Web Inspector, visit the iOS Developer Library at https://developer.apple.com/library/ios/navigation/.

Debugging XPages with Web Inspector Remote (aka weinre)

If you don’t have a Mac or if you need to debug XPages on an Android or other non-iOS-based device, you can use weinre. The weinre debugger is run as a node.js application, so you need to download and install node.js first. To get the setup to start debugging with weinre, follow these steps:

1. Download and install node (see http://nodejs.org/download/).
2. Install the weinre npm package (use npm -g install weinre).
3. Execute the following command to run weinre: weinre --httpPort <port> --boundHost -all-
4. Add a client-side script tag to each XPage you want to debug; the script tag must load the weinre target script from the server where weinre is running (see Listing 14.22 where weinre is running on a server with ip address 192.168.1.11).
5. Open the weinre client from your desktop browser using the url: http://<host>:<port>/client/.
6. Open the XPage that includes the weinre target script in your mobile browser.
7. Now refresh the weinre client in your desktop browser, and you should see the mobile target listed (as shown in Figure 14.21).
Listing 14.22  Script Tag for weinre Target Script

```xml
<xp:script type="test/javascript"
    src="http://192.168.1.11:8090/target/target-script-min.js"
    clientSide="true">
</xp:script>
```

**Tip**

Select a port for weinre to use that doesn’t conflict with any other server you may be running on the same machine. Also specify either -all- or a specific ip address/hostname as the bound host. Using localhost isn’t sufficient unless the browser you are debugging is running on the same localhost.

Figure 14.21  weinre client
Figure 14.22 shows the DOM for the m_Weinre.xsp page.

The Element tab in the weinre client enables you to view the DOM of the page you are debugging. As you select elements from within the DOM in the weinre client, the equivalent element within the browser is highlighted. For example, if you select the span element that corresponds to the Toolbar button, the button itself is highlighted on the device, as shown in Figure 14.23.
In addition to viewing the DOM, you can also display a JavaScript console. So any logging statements that are output using the console JavaScript class can be viewed within Web Inspector. Figure 14.24 shows the JavaScript console with some text that was output when the Toolbar button in the m_Weinre.xsp XPage was clicked.

For more information on weinre, visit the documentation pages at http://people.apache.org/~pmuellr/weinre/docs/latest/.

XPages Mobile Extensions

As mentioned earlier, at the time of writing, Domino 9.0.1 has just been released and includes many new mobile features. Also included in this release is a series of extension points within the XPages runtime and Domino Designer to enable new mobile features to be delivered outside of the normal Domino release cycle. The Extension Library project on OpenNTF is used as a vehicle to allow new features to be delivered on a continuous basis. These features include enhancements to the XPages runtime—for example, support for new properties or events and also features within Domino Designer to simplify the development of mobile applications. For example, within 6 weeks of the delivery of Domino 9.0.1, a new Extension Library release is available that includes two new features for mobile application development:

- Addition of the infiniteScroll property to the Data View (xe:dataView) control
- Addition of the Single Page Application Wizard

Tip
To use Infinite Scroll and the Single Page Application Wizard you need to install the XPages Extension Library version 901v00_02.x (or higher). The XPages Extension Library is available for free download at http://extlib.openntf.org/.
The samples in this section depend on having these new features so they are included in a separate database called chp14ed2ext.nsf.

Infinite Scrolling

A new property called infiniteScroll has been added to the Data View control for use only in mobile XPages. When infinite scrolling is enabled, as the user scrolls through rows in the Data View control, additional rows will be automatically loaded. Rows are prefetched and added directly to the Data View.

Property:

infiniteScroll

Values:

- **enable**: Enable infinite scrolling in the Data View control. (This overrides the application default.)
- **disable**: Disable infinite scrolling in the Data View control. (This overrides the application default.)
- **auto**: Uses the application setting xsp.progressive.enhancement=[enable|disable]. (This is the default value.)

Listing 14.23 shows a page containing two Data View controls, both of which have infinite scrolling enabled. Each Data View control has 1,000 rows, and when you open the page on a mobile device, you can keep scrolling down, and pages are automatically loaded for you. The sample shows that two data views can be used on the sample page, each with infinite scrolling enabled.

**Listing 14.23** Stacked Data Views with Infinite Scrolling

```xml
<?xml version="1.0" encoding="UTF-8"?>
     xmlns:xe="http://www.ibm.com/xsp/coreex"
     xmlns:xc="http://www.ibm.com/xsp/custom">
  <xp:label value="First Data View" id="label1"></xp:label>
  <xe:dataView id="dataView1" var="viewEntry"
               collapsibleCategory="false" collapsibleDetail="false"
               rows="20"
               collapsibleRows="false"
               style="height:200px; border:2px solid blue; margin: 5px"
               collapsibleDetail="false" collapsibleCategories="false"
               infiniteScroll="true"/>
  <xp:label value="Second Data View" id="label2"></xp:label>
  <xe:dataView id="dataView2" var="viewEntry"
               collapsibleCategory="false" collapsibleDetail="false"
               rows="20"
               collapsibleRows="false"
               style="height:200px; border:2px solid blue; margin: 5px"
               collapsibleDetail="false" collapsibleCategories="false"
               infiniteScroll="true"/>
</xp:view>
```
### TIP

Refer to the release notes for the Extension Library version you have installed for details of any limitations associated with new mobile features. For example, at the time of writing, infinite scrolling works only with mobile views, and it is necessary to disable this for a web view.

### Single Page Application Wizard

This is a new Domino Designer feature that has been delivered as part of the Extension Library. The feature provides a wizard that can guide you through the process of adding a Single Page Application (`xe:singlePageApp`) control to an XPage. The wizard automatically opens every time you add the Single Page Application control to a page. By default the wizard enforces the best practice of having a custom control for each Application Page you need to create. The wizard also enables you to specify how to navigate between the Application pages.
**Tip**

Make sure you have installed the Designer extensions from the updateSiteOpenNTF-designer site that comes with the Extension Library to use this feature.

To start using this new wizard, you need to drop a Single Page Application control onto an XPage. Figure 14.25 shows the Single Page Application Wizard.

![Single Page Application Wizard](image)

**Figure 14.25** Single Page Application Wizard

The wizard has an option to create a custom control for the content of each application page. This is the recommended approach because it helps reduce the complexity of the single page application XPage and results in a more maintainable design. Occasionally, having a custom control per application page can be overkill, so you can turn off this option. The start screen of the wizard also enables you to add the application page by selecting the New button. Figure 14.26 shows the New Application Page dialog, which enables you to create different types of templated application pages.

The following Application Page types are supported:

- General
- Application Navigator
- Document Collection
- Document Viewer
The wizard provides different options to configure each Application Page type. Figure 14.27 shows the general options available for an application page. It will have a page heading by default, and you can optionally use the wizard to add toolbar buttons, a back button, and tab bar buttons.

An application navigator application is a special type of page used to navigate around the single page application. You can configure the same options that are supported by a general page. Figure 14.28 shows the options available for an application navigator page; you can create a set of links that allow users to navigate around the application.
For a document collection application page, the wizard guides you through the process of creating a data source to retrieve the document collection and a data view to display the data. Figure 14.29 shows the options to configure the data source for a document collection page.
Figure 14.30 shows the options to configure the data view used to display the document collection (including an option to enable infinite scrolling).

![Figure 14.30 Document collection Data View configuration](image)

For a document viewer application page, the wizard guides you through the process of creating a data source to retrieve the document and a set of controls to view the fields of the document. Figure 14.31 shows the options to configure the data source for a document viewer page.

Figure 14.32 shows the options to configure the controls used to display the document fields.

**Tip**

At the time of writing, there was no support in the wizard for creating a page to support document editing, but this is something that is planned for the Extension Library.

The final step in the wizard is to specify the default application page to be shown in the single page application. Figure 14.33 shows a screen for the final step in the wizard.

You can see from these two examples that some important enhancements for mobile application development are being made available outside of the normal Domino release cycle.
Figure 14.31  Document Viewer data source configuration

Figure 14.32  Document Viewer Fields configuration
Summary

This concludes the chapter on XPages mobile application development. So what does the future hold? Responsive web design is emerging as a new approach for building web interfaces with an optimal viewing experience across mobile and desktop browsers and beyond. Bootstrap for XPages is already available as an extension Library, and this provides functionality to allow you to build responsive user interfaces. For more information on using Bootstrap with XPages, visit: http://bootstrap4xpages.com/. One UI is expected to also evolve in this direction. So in the future (if you are not already there) the lines between developing for the desktop and developing for mobile will blur, and the techniques you will use now for building Mobile XPages applications will become the mainstream techniques you will use.
This page intentionally left blank
Index

SYMBOLS
@Commands, 479
@DbColumn() function, 10
@DbLookup() function, 10
@Functions, 239, 479
@Random function, 481
| (pipe), 87

A
access control lists. See ACLs
Accessibility category (controls), 77
accessing
ACL levels, 1037
administrator access, 1035
anonymous, 44, 619
CSJS, 1059
documents, 74, 268
forms, 1040-1041
Full Access Administrator field, 1054
HTML directories, 824-825
Notes client, 607-608
panels, 99
pass-by-references, 417
profiles, 620
programmability restrictions, 1053-1056
Property Broker, 1050
Public Access, 1041, 1065-1066
reader access lists, 1047
remote servers, 653
restricting, 1042-1045
security, 1035. See also security
servers, 1038
values, 75
View Panel, preventing, 1042
views, 326-336, 1041-1042
Access Key Validator editor, 584
accounts
Run On Server, 658
testing, 659
ACF (Active Content Filtering), 1062-1065
ACLs (access control lists), 22, 35
anonymous user access, 554
application layer of security, 1037-1039
Internet name and passwords, 1039
panels, 100
replication, 619
security, implementing, 1051-1052
action buttons, adding, 48
action group simple actions, 215-216
action property, 256
actions
adding, 52
clipboard, disabling, 1041
debugging, 739
event handlers, 674
New Group, 413
New Property, 413
receive events, 673
simple, 11, 128-134
action group, 215-216
change document mode, 198-199
change dynamic content, 219-220
clients, 189
confirm, 199-200
create response document, 200-201
delete document, 201-202
delete selected documents, 202-203
execute client script, 203-204
Domino documents, 283
execute script, 204
logic, 198-218
modify field, 205
move to application page, 220
open page, 205-207
publish component property, 207
publish view column, 208-209
save component mode, 213-214
save data sources, 209-211
save document, 211-213
send mail, 217-218
servers, 189
set value, 214-215
actions property, 215
Active Content Filtering. See ACF
Add Action button, 50
AddDatabaseToWorkspace, updating, 632
adding
action buttons, 48
actions, 52
Application Layout control, 850-865
applications to lists, 651
behavior, 78
Button controls, 50
comments, 1067
Computed Fields, 25
data sources, 45
dependencies, 625
documents, 73
fields, 38
folders, 630
Java source code, 543-544
logic, 187-197
mark up, 136
node items, 858
NSF components, 670
Package Explorer, 541
place names, 857
profile blocks, 948-949
properties to components, 556
resource bundle files, 887-888
server logic, 244-245
strings, 882
type ahead to edit boxes, 78
UI Spinner controls, 554
add-ons, User Agent Switcher, 682
Add Simple Action dialog box, 50
afterPhase() method, 958
Agent Contents section, 326
Agent.getParameterDocID() method, 482
agent.runOnServer() method, 489
agent.runWithDocumentContext() method, 483, 488
agents
building blocks, 482-489
FT Search Test, 325
jAgent, 484-487
to run on behalf of someone else, 1055
to run on behalf of the invoker, 1055
User Agents
iPhone, 678
modifying, 679
resources, 827-831
Safari, 680-681
Switcher add-on, 682
viewing, 678
Aggregate Container, 849
aggregate containers, 403, 432-433
aggregating components, 671
Resource Aggregation, enabling, 907
AJAX (Asynchronous JavaScript and XML), 444
partial refresh, 444-460
doing-it-my-way style, 453
out-of-the-box style, 445-452
performance, 460
partialRefreshGet utility functions, 454-456
partialRefreshPost utility functions, 458
alert() function, 622, 646
alerts
ECLS, 187
security exceptions, 1061
alert() statement, 759
algorithms, lazy-load/preemptive, 1021
aliases, 824
All Documents view, 338, 345
Data Tables, 372
structure of, 383
allDocumentsView control, 494
allDocuments XPage, previewing, 865
allocating
JVM heapspace, 1021
memory, 924. See also memory
allowDeletedDocs property, 256
Allow Document Locking feature, 269
All Properties section (Properties tab), 558
ALL statement, 757
alpha XPages, markup for, 435
alwaysCalculateLast, 397
analyzing
Eclipse Memory Analyzer, 1019
memory, 1005, 1008-1019
performance, 932. See also performance
anonymous access, 44, 554, 619, 620
Anonymous names, 1051
APIs (application programming interfaces), 6
current platforms, detecting, 830
document locking, 269
Dojo. See Dojo
Java classes, 1078
JSF, 165
reading directions, detecting, 831
REST service, 327-330
Apple iPhones. See iPhones
Application Layout control
adding, 850-865
Content area, customizing, 865-870
overview of, 849-850
Application Level themes, 797
Application Management menu, 29
application programming interfaces. See APIs
Application Properties, editing, 874
applications
building, 33-34
building blocks, dividing, 402-403
clients, debugging, 759-764
components, formatting, 664-666
configuring, 22, 853
navigation/link behavior, 641
optimizing, 906
content, viewing, 541
CRUD, 47-53
Custom Controls, 402. See also Custom Control
debugging, 645-648, 726-759
Domino Designer, creating, 19
from external web browsers, invoking, 1058
Firefox Lite, 648
formatting, 540
forms, 36-41
frameworks, scripting, 443-444
Hello World themes, 805
IBM OneUI themes, 604, 1086
inline styles, performance, 779-786
instances, creating, 34
internalization, 871, 872-886
JSF, 158. See also JSF
launch options, modifying, 642
layouts, 849-850. See also Application Layout control libraries, 750
lists, adding, 651
loading, optimizing, 650
localization configuring, 894-897
testing, 877-878
logic. See logic
memory models, 1005-1008
MIME, formatting, 641
Mobile Application Development, 677-678. See also Mobile Application Development
Notes client. See Notes client overview of, 3-4
previewing, 35
privileges, configuring, 1057
processing, 614
quick testing, 554
resources, managing, 815
sample documents, 36
SCXD, 652-654
security, 1035-1049. See also security formatting, 1049-1051
layers, 1037-1039
transitions, 689
translations, loading, 893
UISpinner components, 597-604
views, 36-41-47
applicationScope variable, 166
Application Theme. See also themes
Override on Notes setting, 809
Override on Web setting, 809
applLayout XPage, 860, 864
Apply button, 960
applying compositeData objects, 421-427
compression, 1026-1027
date converters, 344
Dojo widgets, 468-470
images to columns, 348-351
immediate property, 916
inline CSS styles, 771. See also CSS
Java, debugging, 748-755
JavaScript, 220-250
JSON RPC service (remote service), 246-250
libraries, applications, 750
partial refresh, 906
profilers, 932-949
Property Definitions, Custom Controls, 411-421
readOnly property, 914
REST service, 327-330
security, 1049-1053
sessionAsSigner property, 1067-1068
Single Page Application Wizard, 719
style property, 777-778
styles classes, 779-793
Custom Controls, 407
modifying, 774
Notes client, 775
themes, 779-793, 795, 804-807
Toolbox (XPages), 932-933
translators, 878-881
XPages Extension Library, 26-32
xp:eventHandler tags, 190
XSP client script libraries, 245-246
Apply Request Values, 163
architecture
Custom Controls, 433
runtime, 1005
themes, 796-797
areas, Content, 865-870
Array class, 235
artifacts updating, 630
XPages Toolbox project, 932
xsp-config, inheriting, 558-562
ascending order, sorting results, 324
assigning access control to panels, 99
Form fields, 40
Asynchronous JavaScript and XML. See AJAX
AttachmentDialog.js source code, 525
attacks, 1062. See also security attributes configuring, 63
maps, implementing, 555
styleClass, 587
UI component properties, 555
<aattribute> tag, 555
authentication name-and-password, 1037
server layer, 1036-1037
sessions, 1037
troubleshooting, 636
AUTHOR access level, 1037, 1041
author fields, 30
autoformatting tags, 50
automating documents, 283-286
available locales, 897, 899
avoiding backend getDocument() API methods, 907
computed rendered property expressions, 907
string concatenation, 887
B
Back button, 691. See also navigating
backend
classes, Java, 10
getDocument() API
methods, 907
profilers, 946-948
backgrounds
colors, 99
modifying, 771
Background tab, 771
backing beans
formatting, 597-599
registering, 599-600
backups, translations, 883
bandwidth, Single Page
Application design patterns, 685
banners, anonymous users, 620
bars
Custom Controls, previewing, 404
defining, 410
base .xsp-config files
creating, 562-565
interfaces, creating, 565-568
Basic Application configuration, 853
Basic Node items, 861
Basics category (controls), 77
bce property, 217
beans
backing
formatting, 597-599
registering, 599-600
Device Beans, 683-685
managed, 490-496
beforePhase() method, 958
behavior
adding, 78
components, modifying, 557-558
Custom Controls, themes, 795
Data Tables, 370
interfaces, JSF, 170-173
Notes client, 632-633
optimizing, 918
override, <control> element, 838
tabs, managing, 638
View control, 640
benefits
of JSF, 157
of XPages, 4
beta XPages, markup for, 438
bidirectional resources, themes, 831-832
BiDi support, indented categorization with, 363
binary plug-ins, importing source code, 752
binding, 770
Computed Fields, 370
controls, 159
data, 72
Data Tables, 370
expressions, 164, 180
methods, 587
titleLabel, 426
values, 179
View controls to data sources, 41
binding property, 214
blank application templates, 540
blocks
profiles, 945, 948-949, 967
special_profile(), 969
bodyHtml property, 217
bodyPlain property, 217
bookmarks, Notes client, 614-616
Boolean
Check Box editor, 584
class, 235
editors, 416
Value editor, 584
bootstrapping costs, 614
BorderPane element, 469
bottlenecks, 938. See also performance
breakpoints
conditional, 742-745
configuring, 738
Java, 745-747
bridges
NSF classloaders, 1058
XSP.executeCommand, 1059
browsers
applications
invoking from external, 1058
launching, 1050
previewing, 860
IE, Notes client, 608
Mobile Application Development
Chrome, 681
Firefox, 682
Safari, 680-681
processing, 918
reloading, 445
User Agents
modifying, 679
viewing, 678
bug fixes, 8
building
applications, 33-34
creating instances, 34
CRUD, 47-53
forms, 36-41
views, 36-41-47
blocks
agents, 482-489
dividing, 402-403
in-memory documents,
482-489
profile documents,
482-489
scripting, 478-489
Custom Controls, 427-432
UI components, 544-554
built-in translations
Dojo, 893
runtime, 893
built-in variables, JSF, 166
<bundle> element, 818
properties, 818
resources, 818
bundles
properties, exporting, 873
resource files, 115, 876
adding, 887-888
editing, 878
exporting, 878-880
importing, 880-881
localization, 873-874
button bars, dialog boxes, 147-151
Button control, 49
  Style properties panel, 786
Button controls, adding, 50
buttons, 83, 708. See also specific buttons
buyMostRecent XPage, previewing, 865
byTagViewCc control, 319
C
  caching
defCache property, 926-928
  enabling, 787
  resource bundles, 890
  views, 318-322
CAE (Composite Application Editor), 669, 671
calculating entry counts, 397
calendar data, accessing views, 326-336
calling front-end Java classes, 624
cancel property, 213
capabilities, vertical, 1004
captionStyleClass property, 793
Cascading Style Sheets. See CSS
categories
  of controls, 76
  filters, 298
  hierarchies, 360
  indented categorization with BiDi support, 363
  properties, modifying, 557
  rendering, 360
View Panel, applying, 357-360
  views, 313, 364
categorization, 297
categoryColumn property, 389
categoryFilter property, 295, 297-299
  bookmarks, 615
  <category> tag, 559
  cc property, 217
CDATA, 67
delimiters, 67
central processing units.
  See CPUs
change document mode, 198-199
Change Dynamic Content Action control, 142-144
changeDynamicContentAction() method, 993
change dynamic content simple action, 219-220
Character Set Type Picker editor, 584
charset property
  <linkResource> element, 820
  <script> element, 819
checkboxes, 90
groups, 92-93
checked values, 90
checking
  for documentID at runtime, 1066
  Public Access, 1066
security, 1057-1058
child elements, <resource>
  element support, 816
  children, panels, 98-101
Chinese translation, 880
Chrome, Mobile Application Development, 681
CKEditor, 286, 290
classes
  Array, 235
  Boolean, 235
  com.ibm.xpages.
    PreviewBean, 492
  com.ibm.xsp.designer.context.
    XSPUserAgent, 829
  Date, 235
  DirectoryUser, 236
  Document, 284
  DominoViewEntry, 1077
dummy, creating, 571
Java, 10, 538
  API programming
    references, 1078
declaring managed beans, 492
  formatting, 546, 626
  programming references, 1076-1077
JavaScript, 1078-1080
libraries, internationalization, 894
Math, 235
NotesXspDocument, 236
NotesXspViewEntry, 236
Number, 235
Object, 235
paths, extending Dojo, 466-468
ProfilerAggregator, 949
renderers, creating, 539
String, 235
styles, 1081-1083
  applying, 779-793
  CSS files, 1081-1082
  wrappers, 1079
classes.css, 782
classloaders, NSF, 1058
cleaning projects, 874
clients
  applications, debugging, 759-764
  Domino Designer, configuring, 15
  fix packs, installing, 14-15
  ID binding expressions, 181
  JavaScript, 240
    adding server logic, 244-245
      applying XSP client script
        libraries, 245-246
          including server data in, 243-244
  JSON RPC service (remote service), 154-156
  Notes client, 361-365. See also Notes client
  RCP (Rich Client Platform), 17
  Sametime, 15
  servers, synchronizing, 159
  simple actions, 189
toolbars, 647
VoiceML, 165
weinre debuggers, 714
client-side
  scripting, 134-136
  script libraries, 891-893
validation, 127. See also validators
Client Side Event editor, 584
Client-Side JavaScript. See CSJS
clientSide property, <script> element, 819
clipboard, disabling actions, 1041
closing windows, 634
code
AttachmentDialog.js, 525
client-side/server-side,
combining, 444
deprecated locale, 898-900
ImageDialog.js, 527
Java. See Java
monitoring, 963
collections, documents, 45, 708
colors
backgrounds, 99, 771
radio buttons, 91
columnClasses property, 793
columnName property, 203, 390
columns
calendars, 328
formatting, 341-343
headers, 105
images, applying to, 348-351
repeat control, 107, 108
sorting, 307, 351-357
View Panel, 348-351
views, 102, 348
categories, 364
navigating, 40
selecting, 43
sorting, 323-326
viewing, 47
combining
CD/MIME, 644
client-side/server-side
code, 444
searching and sorting, 323
Combo Box editor, 584
combo boxes, 88-90
editors, 416
com.ibm.xpages.PreviewBean class, 492
com.ibm.xsp.designer.context.XSPUserAgent class, 828-829
command controls, 82
commands
extended CSJS, 624-632
log(), 759
comments
adding, 1067
markup, 266
common properties, document
data sources, 282
comparing Domino links/Notes
links, 641-644
compatibility, CD/MIME, 644
complex properties, 64-66, 847
interfaces, 577
link controls, 913
specifying, 568-579
complex-type property, 569
complex values, 66-67
ComponentBindingObject
interfaces, 577
<component-class> tag, 549
<component-extension> tag, 549
<component-family> tag, 549
components
aggregating, 671
applications, formatting,
664-666
behavior, modifying, 557-558
Domino Designer, revising
properties in, 568
examples, 539
formatting, 158
JSF. See JSF
models, Uls, 170
NSF, adding, 670
properties, adding, 556
runtime architecture, 1006
trees, processing, 918
types, referencing, 570
Uls. See Uls
wiring, 671
<component> tag, 549
<component-type> tag, 549
Composite Application Editor,
installing, 14
composite applications, Notes
client, 664-676
compositeData objects,
421-428
composite patterns, 545
compression, 1026-1027
computed expressions,
localization, 885-890
Computed Fields
adding, 25
binding, 370
configuring, 79
controls, 94, 790
localization, 900
modifying, 840
panels, 99
partialRefreshField element,
452
partialRefreshGetField, 456
computeDocuments property, 256
computed properties, 67-71
computed rendered property
expressions, 907
computed values, style
property, 778
computeWithForm property, 256,
654-656, 1040, 1046-1047
computing styleClass property,
788
concatenation, avoiding
strings, 887
concurrencyMode property, 256,
266
concurrent document updates,
managing, 266-272
conditional breakpoints,
debugging, 742-745
condition property, 215
<dojoModule> element, 819
CONFIG statement, 757
Configuration Wizard dialog box,
853
configuring
ACF, 1064
Application Layout control,
853
Application Page options, 720
applications, 22, 853
navigation/link behavior,
641
optimizing, 906
attributes, 63
bookmarks, 614
breakpoints, 738
controls, rich text, 287
Data properties, 66
Data View control, 389-391, 722
dir property, 893
Domino Designer, 15
EDITOR access levels, 1039
Event Parameters, 460-462
forms, accessing, 1040-1041
infinite scrolling, 717-718
JavaScript, debugging, 764
lang property, 893
links, 644
locales, 894-897
localization, 874-876
mobile properties, 686
pagers, 395-398
performance preferences, 651
plug-ins, 29, 625
privileges, 1057
programmability restrictions, 1053-1056
Public Access, 1065-1066
RAM, 1023
readonly property, 1046
rendered property, 1046
requests, scalability, 1004-1034
scability, 1007, 1020-1034
SCXD, 652-654
serialization, 1032
sessionAsSigner property, 1067-1068
State Management Layer, 1027-1032
style property manually, 776-777
themes, 797-803, 807-810
Toolbox (XPages), 932-933
variables, resolvers, 158
XPages Extension Library, 28.xsp-config configuration files, 538, 547-550
confirm action, 199-200
confirm() function, 622
connecting
Data Tables, 371
data sources, 75-76
data tables, 105-106
deleting, troubleshooting, 738
Domino servers
output, 960
p2 Std button, 963
Test Phases button, 962
servers, viewing, 959
Constructors, public no-args, 556
Container Controls palette, View control, 42
containers, 98-111
aggregate, 403
Change Dynamic Content
Action, 142-144
data tables, 105-106
Dynamic Content, 138-141
include page, 109
layouts, 403
panels, 98-101
repeat control, 106-109
sections, 111
tabbed panels, 110
tables, 101-102
views, 102-104
content
ACF, 1062-1065
applications, viewing, 541
drop targets, 861
dynamic, 138-141
Dynamic Content control, 992-1004
rich text, managing, 641
rows, 370
themes, creating, 807
views, 41, 309-312
XML, adding, 67
Content area, customizing, 865-870
ContentLangDirection property, 289
ContentPane element, 469
Index

D

data, separation, 770
databaseName property, 256, 295-296
databases
  searching, 299-304
  XPages, defining, 58
database variable, 182
data binding, 41, 72
dataCache property, 295, 318-322, 906, 926-928
data category (controls), 76
data context, 75. See also context
data conversion, 81
Data palette, 46
Data properties, configuring, 66
Data Source Picker editor, 585
data sources, 10, 73-76
  adding, 45
  context, 75-76
  documents, 73-74, 254-282
  common properties, 282
  creating, 45
  events, 274-278
  multiple, 272-274
  webQuerySaveAgent property, 278-281
  views, 74-75. See views
Data Tables, 370-381
  All Documents view, 372
  connecting, 371
  previewing, 378
  Profile view, 376-381
  data tables, containers, 105-106
dataTableStyleClass property, 793
dataTableStyle property, 793
Data View control, 339, 387-394
  configuring, 389-391
  document collection, 722
  facets, 394
  mobile themes, 704-705
  optimizing, 392-394
  properties, 389
Date class, 235
dates
  converters, applying, 344
  expiration, default cache, 787
DateTimeConverter, 173
Date Time Picker control, 463
date/time picker controls, 80-82
debugBean object, 951
Debug Configurations dialog box, 737, 742
debugging, 725
  actions, 739
  applications, 726, 759-764
  conditional breakpoints, 742-745
  connecting, troubleshooting, 738
CSJS, 760-762
debugger statements, 741
Dojo, 762-764
EL, 951
features, 741
Java
  applying, 748-755
  breakpoints, 745-747
  enabling logging, 755-759
JavaScript, configuring, 764
Mobile Application Development, 710-716
PhaseListeners, 958
plug-ins, 752
printing, 726-729
Request Introspection, 967
SSJS, 735-740
topics, viewing, 732
try/catch/finally blocks, 729-735
Web Inspector, 713-716
XPages
  iOS, 711-713
  Toolbox, 933
XPiNC tips, 645-648
debug property, 951
declaring
  managed beans, 490
  name/value properties, 833
<resource> elements, 817
decode() method, 596
default cache expiration dates, 787
default encryption keys, 1041
default languages, 874. See also internalization
default locales, 897, 899
<default-prefix> tag, 549
default read access, 1040
default scripting language. See JavaScript
default themes, 807
default variables, 166, 182-183
defining
  complex properties, 569
  fields, 38
  navigation rules, 50
  properties, 414
  tags, 539
toolbars, 287
View controls, 43
XPages, 58-59
definitions
  controls, 838-839
  xsp-config
    inheriting, 558-562
    creating, 562-568
delegation models, JSF, 166
delete document simple action, 201-202
delete selected documents simple action, 202-203
deleting
calendar entries, 334
documents, 52-53
scripts, tags, 1064
strings, 883
delimiters, CDATA, 67
deliveryPriority property, 217
deliveryReport property, 217
deltaex option, 1032
delta option, 1032
Dependencies tab, 625
DEPOSITOR access level, 1037
deprecated locale codes, 898-900
<description> tag, 559
design
Custom Controls, 401-402, 432-440
Discussion application, 34
patterns
application layouts, 849-850
formality, 850
programmability restrictions, 1054
Repeat control, 383
resources, 11
SCXD, 652-654
security
configuring privileges, 1057
layers, 1039-1046
Single Page Application
design patterns, 685-688
themes, 555
DESIGNER access level, 1037
<linkResource> tag, 559, 583
design-time visualization, 9
desktop browsers. See also
browsers
Domino server consoles,
viewing, 959
User Agents, modifying, 679
detecting
current platforms, 830
reading directions, 831
requests, 815
User Agents devices, 682-683
Debug user agent property, 685
development, 3, 8-11
IDEs, 16
interfaces, 769-771
Lotus Notes Template
Development ID file, 1050
menus, items, 711
Mobile Application
Development, 677-678. See also
Mobile Application
Development
models, 11
deviceBean.isMobile() method, 683
deviceBean.isTablet() method, 683
Device Beans, 683-685
devices
emulators, 679
User Agents, detecting,
682-683
differences, Notes client, 621-624
directories
Dojo, 826
Global themes, 825-826
HTML, 824-825
DirectoryUser class, 236
dir property
configuring, 893
<linkResource> element, 820
dirty documents
navigating, 633
saving, 633
disableModifiedFlag property,
634-637
disableValidators property, 196
disabling
printing, 1041
view filters, 306
Discussion application, 33. See also applications
banners, anonymous users, 620
creating, 34
forms, 36-41
Notes client, starting, 610
replication, 618
views, 36-41-47
disk persistence, 907, 1020-1026
displayAs property, 341
displaying
controls, 93
User Agents, 678
<display-name> tag, 549, 559
dividng
building blocks, 402-403
processing, 982
DOCTYPE, 61
documentation, IBM OneUI
themes, 1086
document-centric data models, 41
Document class, 284
documentld property, 206, 256,
1066
Document Object Model. See DOM
documents, 253. See also
forms
accessing, 74, 268
Allow Document Locking
feature, 269
anonymous user access, 554
change document mode, 198-199
collections, 45, 708
controls, 10
data sources, 73-74, 254-282
common properties, 282
definition, 256
events, 274-278
multiple, 272-274
webQuerySaveAgent
property, 278-281
default read access, 1040
deleting, 52-53
dirty
navigating, 633
saving, 633
editing, 257-258, 1046
fields, viewing, 265
formatting, 38, 257-258, 1046
form logic, executing, 263-266
hierarchies, 311
individual, viewing, 45
in-memory, building blocks, 482-489
JavaScript, 284-286
keyView, 324
managing, 40
privileges, configuring, 1057
profile, building blocks, 482-489
programming, 283-286
replication, 617
response, creating, 258-263
rich, 286-291
sample, 36-37
saving, 83
security layers, 1046-1048
signing, 1048, 1051
summary data, 40
UNIDs, 39
updating, managing, 266-272
URLs, controlling parameter
usage, 258
View Panel, linking, 346-348
views
building, 41-47
retrieving, 315-317
Document Type Definition. See DTD
doing-it-my-way style, partial refresh, 453
Dojo, 455, 463
built-in translations, 893
category (controls), 77
class paths, extending, 466-468
debugging, 762-764
deprecated language codes, 898
folders, viewing, 474
integration, 463-478
modules, 116
Modules, resource types, 815
plug-ins, 826
rich text editors, 286
Toolkit, 463
when an XPage is not an XPage, 475-478
widgets, 466-468
customizing, 470-475
mxpd.data.
ViewReadStore, 476
mxpd.ui.ViewTree, 476
standard, 468-470
dojoIntegration, 466, 469, 471
<dojoModule> element, 818
properties, 819
resources, 464-466, 819
dojoParseOnLoad property, 464
dojo.require() statement, 465
dojoTheme property, 464, 827
dojoType property, 466
DOM (Document Object Model), 188, 239, 715
Domino Designer, 4-5
applications, creating, 19
client fix packs, installing, 14-15
clients, configuring, 15
components, revising
properties in, 568
controls, adding, 24-26
Data View control, 387-394
downloading, 13-14
Home Page, 16
identify bots, 141
installing, 14
navigating, 16-26
Notes Client, previewing in, 21
Package Explorer
adding, 541
enabling, 473
plug-ins, debugging, 752
registries, 570
styles, 773
viewing, 16
Views entry, 37
web browsers, previewing in, 22-24
XPages
creating, 20-21
Extension Library, installing, 29
Domino Internet ID, working
offline password, 619
Domino servers. See also servers
output, 960
p2 Std button, 963
Test Phases button, 962
viewing, 959
DominoViewEntry class, 1077
Double Value editor, 585
downloading
Domino Designer, 13-14
files, 97
drag-and-drop, views, 43
drop targets, 861
DTD (Document Type Definition), 60
dummy classes, creating, 571
_dump() method, 8, 646, 728
dumpObject() method, 760
dumps
memory, 1009
viewing, 1017
XML, 1011
sessions, 1030
duration, default cache expiration
dates, 787
Dynamic Content control, 138-141, 974, 992-1004
Dynamic View Panel, 359

E
ECL (Execution Control List), 612
Notes client, 1059-1062
security, 1058
workstation ECL security layers, 1048-1049
Eclipse
history of, 16-17
Memory Analyzer, 1019
plug-ins, 5
Events tab, 188
Events view, 427
   partial refresh, 445
examples
   interface components, 539
   of themes, 796
exceptions
   null pointers, testing, 746
   security alerts, 1061
   selecting, 747
   themes, levels of inheritance exceeded, 814
   throwing, 730
execId property, 194, 446, 921
execMode property, 194
execute client script simple action, 203-204
executeCommand() method, 624, 626
execute script simple action, 204
executing
   application logic, 187
   form logic, 263-266
   Notes client, 608
   partial execution, 919-922, 986-992
   queries, 302, 324
   replication, 617
   restricted network operations, 1046
Execution Control List. See ECL
execution security alert. See ESA
existing applications, 616. See also applications
expandLevel property, 295, 310-312
expiration dates, default cache, 787
Export dialog box, 879
exporting
   property bundles, 873
   resource bundle files, 878-880
Expression Language. See EL
expressions
   binding, 164
   computed, localization, 885-890
   computed rendered property, avoiding, 907
EL, 372
   formula language, 482
   JavaScript, binding, 180
   method-binding, 165
   validation, 418
extended CSJS commands, 624-632
extendedRTE.xsp
   axtargetUrl configuration, 530
   InputRichTextConverter configuration, 534
   source markup, 506
extending
   Dojo class paths, 466-468
   java.policy files, 1069-1071
   JSF (JavaServer Faces), 166-186
   styleClass property, 790-793
   style property, 790-793
   themes, 795
extensibility, 537-538
   component examples, 539
   properties, specifying complex, 568-579
UI component extensions
   building, 544-554
   creating, 540-544
   customizing renderers, 591-596
   implementing UISpinner, 588-591
   properties, 555-562
   sample application creation, 597-604
UI controls, creating, 538-539
XPages Extensibility API Developers Guide, 605
xsp-config files
   creating initial definitions, 562-568
   UISpinner components, 579-587
Extensible HTML. See XHTML
Extensible Markup Language. See XML
extensions
   MIME, 641
Mobile Application Development, 716-722
Single Page Application Wizard, 718-722
UI components
   creating, 540-544
   customizing, 540
   customizing renderers, 591-596
   implementing UISpinner, 588-591
   properties, 555-562
   sample application creation, 597-604
XPages Extension Library, 7, 11, 26-32
external web browsers, 1058
extraColumns property, 389
Facets, 103
  Application Layout control, 862, 868
  Data View control, 394
property, 341
failover procedures, bookmarks, 615
features
  CKEditor, 290
  computeWithForm, 654-656
Data Tables, 370
debugging, 741
Event Parameters, 460-462
installing, 14
managed beans, 490-496
Run On Server, 656-657
View Panel, 340-341
XPages Extension Library, 28
fields
  author, 30
  computed, 94
Computed Fields
    binding, 370
    localization, 900
defining, 38
encryption, 1048
Form, assigning, 40
formatting, 38
Full Access Administrator, 1054
hidden, 264
properties, 37
reader, 1047
Required, 423
Tags, 36
viewing, 265
field studies, fulfilling customer requirements, 496-535
file-handling controls, 95-97
File New Replica dialog box, 617
File Replication menu, 616
files
CSS, 1081-1082
downloading, 97
faces-config.xml, 490
Full Session XML Dump, 1013
JAR, 1057
java.policy, 1069-1071
Lotus Notes Template Development ID, 1050
NSFs, 253
Part Session XML Dump, 1011
resource bundle, 876
.tld, 549
WAR, 541
.xsp-config. See also
  .xsp-config files
    formatting, 538, 547-550
    UISpinner components,
    579-587
filters
  ACF, 1062-1065
  views
data source, 296-309
  disabling, 306
Finish Single Application Page Wizard, 724
Firebug
  invoking, 177
  navigating, 761
Firebug Lite, integration, 648
Firefox
  Mobile Application Development, 682
  Notes client, 608
User Agents, viewing, 678
fix packs, installing clients, 14-15
folders
  adding, 630
  Dojo, viewing, 474
  Java source files, 543
Local, 544
Font tab (Style properties panel), 771
foreign languages, 874. See also
  languages
  for loops, 937
  formality, 850
formatting
  applications, 540
  components, 664-666
  Domino Designer, 19
  Notes client, 619-621
  security, 1049-1051
  backgrounds, 771
  backing beans, 597-599
  bookmarks, 614
  classes
    Java, 626
    UIs component extensions,
    545-547
  colors, radio buttons, 91
columns, 341-343
components, 158
controls using repeat control
to, 107
Custom Controls, 402,
  403-411
documents, 38, 253, 257-258,
  1046
data sources, 255
  response, 258-263
dummy classes, 571
fields, 38
folders, Java source files, 543
Google search widgets,
  666-668
HTML, 59
indexes, 300
infinite scrolling, 717-718
interfaces
  base .xsp-config files,
  565-568
  Java, 570
Java classes, 546
Javadoc format, 1077
MIME, 641
pagers, 395-398
panels, 98-101
projects, 629
renderers, 539, 551-553
Repeat control, 106-109, 383
rich text, 79-80
SCXD, 652-654
sections, 111
style sheets, 114-115
tabbed panels, 110
tables, 101-102
tags, specification, 547-550
TeamRoom application
instances, 327
text, 286
themes. See themes
topics, 36
UIs
component extensions, 540-544
controls, 538-539
update sites, 630
View Panel, 343-345
views, 102-104
windows, Notes client, 632
XML syntax, 62-63
XPages, 20-21
.xsp-config files, 538, 547-550
base, 562-565
definitions, 562-568
Form fields, assigning, 40
formName property, 256
forms, 36-41
accessing, 1040-1041
AUTHOR access level, 1041
computeWithForm feature, 654-656
logic, executing, 263-266
metadata, creating documents based on, 38
Form Table control, mobile
themes, 706
formula language, 479, 482
for property, 219
Forum View control, 339
forwarding, disabling, 1041
foundation layers, 3, 164
fragments, URLs, 996
frameworks
accounts, 659
applications, scripting, 443-444
from property, 217
FTSearchTest agent, 325
fulfilling customer requirements, 496-535
Full Access Administrator field, 1054
Full Session XML Dump file, 1013
full text search properties, 299-304
fulltree option, 1032
functionality
internationalization, 893
mobile applications, 677
User Agent Switcher, 682
functions. See also methods
getDbColumn(), 10
dbLookup(), 10
Random(), 481
alert(), 622, 646
confirm(), 622
_dump(), 646
print(), 647
_profile(), 945
server layer security, 1036
G
gadgetUrl property, 217
Garbage Collectors (JVMs), 1021
Generate Heap Dump button, 1011
generating HTML pages, 161
Generic File Picker editor, 585
generic heads, 116
German translations, 878
getAllEntriesByKey() method, 309
getBrowser() method, 829
getBrowserVersion() method, 829
getBrowserVersionNumber() method, 829
getClientId() method, 444
getColumnValue() method, 321
getColumnValues() method, 343
getCommandList() method, 626
getComponentAsString() method, 224
getComponent() method, 444
getComponentsAsString() method, 224
getDatabasePath() function, 472
getDocument() method, 284
getFamily() method, 284
getFamily() methods, 547
GET requests, 906, 908-910
CPU utilization, reducing, 912-917
introspection, 969
Network tab, 999
getStyleKitFamily() method, 591
getters, properties, 578
getUserAgent() method, 829
getValue() method, 558
getVersion() method, 829
getVersionNumber() method, 829
ggetViewAsString() method, 224
Global directory, themes, 825-826
global objects, 227-240
Google search widgets, creating, 666-668
gotchias, localization, 883
grant statements, 1070
groups
checkboxes, 92-93
multiple simple actions, 131
properties
Custom Controls, 430-432
matching definitions, 565-568
radio buttons, 93
.xsp-config tags, 559
<tag> tag, 559
<group-type-ref> tag, 559
<group-type> tag, 559
GZip persisted files option, 1026
H
handlers property, 195
hardcoding strings, 872
hasEntry() method, 829
headers
columns, 105
variable, 166
headerValues variable, 166
headings, segmented buttons, 695
heapspace, allocating JVM, 1021
Hello World application, 19, 160.
See also applications
ACLs (access control lists), 23
themes, 805
Help Index, 479
help() method, 761
help property, 213
hidden fields, 264
hidden text, viewing, 648
hiding
controls, 99
sections, 1048
Hierarchical Dominators link, 943
hierarchies
categories, 360
Mobile Application Development, 692-694
tags, 311
user interfaces, 176
history, 4-8
of Eclipse, 17
Home Page, Domino Designer, 16
href element, 816
hreflang property,
<linkResource> element, 820
href property
<linkResource> element, 820
<styleSheet> element, 820
HTML (Hypertext Markup Language), 9, 59
directory, 824-825
generating, 161
markup, receiving, 787
styles, viewing, 785
tags, 136
text, formatting, 286
themes, 770
htmlFilter property, 1063
<html> tags, 61
htmlUrl property, 217
HTTP (Hypertext Transfer Protocol), 160, 658
httpEquiv property, <metaData> element, 821
HTTPJVMMaxHeapSize parameter, 924
HTTPJVMMaxHeapSizeSet parameter, 924
hyperlinks, 84. See also links
Hypertext Markup Language. See HTML
Hypertext Transfer Protocol. See HTTP
I
I18n. See internalization
IBM
developerWorks, 13, 606
OneUI themes, 604, 1086
Rational Application Developer, 3
support, 7
iconColumns property, 389
id attributes in Custom Controls, 410
identifiers, locale, 771
identify bots, 141
IDEs (Integrated Development Environments), 16, 57
IDs, Lotus Notes Template Development ID file, 1050
IE (Internet Explorer)
CSJS, debugging, 762
Notes client, 608
ignoreRequestParams property, 256, 295, 305-306
ImageDialog.js source code, 527
Image File Picker editor, 585
images, 94
backgrounds, 771
columns, applying to, 348-351
controls, 95, 743
immediate property, 196, 458, 916
partial execution mode, 922
implementing
ACLs, security, 1051-1052
attributes, maps, 555
JSF, 164
LargeSmallStepInterface, 573
navigators, 690-691
renderers, customizing, 591-596
standard methods, 547
StateHolder interfaces, 556
UISpinner class, 588-591
importance property, 217
importing
plug-ins, 753
resource bundle files, 880-881
source code, 752
import statements, 628
Inbound stage, 982
include page containers, 109
indented categorization with BiDi
support, 363
indexes
creating, 300
Help Index, 479
individual documents, viewing, 45
infinite looping, 812
infinite scrolling, 717-718
infoboxes, 37
document IDs, 39
Run On Server, 659
INFO statement, 757
inheritance
themes, 796-797, 812-814
.xsp-configure properties, 558-562
initParam variable, 166
inline CSS styles. See also CSS
applying, 771
performance, 779-786
in-memory documents, building blocks, 482-489
iNotes calendar control, 330-336
in place editing, 137, 143-147
input controls, readonly property, 1045
inputText2 XSP markup, 480
installing
client fix packs, 14-15
Domino Designer, 14
extended command plug-ins, 630
features, 14
libraries, 749
plug-ins, 29
update sites, 630
XPages
Extension Library, 29
Toolbox, 932
XULRunner plug-ins, 609
instances
creating, 34
documents, 285
multiple, Custom Controls, 430-432
Teamroom application, creating, 327
JavaServer Faces. See JSF
JavaServer Pages. See JSP
Java Specifications Request. See JSR
Java Virtual Machines. See JVMs
JBoss, 3
JCP (Java Community Process), 4
JSF (JavaServer Faces), 3, 63, 157-158
APIs (application programming interfaces), 165
behavioral interfaces, 170-173
client ID binding expressions, 181
components, extending, 545
converters, 173
default variables, 166, 182-183
extending, 166-186
implementing, 164
JavaScript binding expressions, 180
lifecycles, customizing, 169
method-binding expressions, 179
multipart binding expressions, 181
overview of, 158-159
referencing, 165
request, 165
request processing lifecycles, 169-170
simple actions, 181
UIs, 164, 176-179
user interface component model, 170
validators, 174-175
value binding, 179
XML-based presentation tiers, 169
JSF Primer, 159-166
JSON (JavaScript object notation). 454
JSON RPC service (remote service), 154-156, 246-250
JSP (JavaServer Pages), 9, 161
Custom Controls, 401
engines, loading, 161
JSR (Java Specifications Request), 4
JVMs (Java Virtual Machines), 1009
Garbage Collectors, 1021
heapspace, allocating, 1021
RAM persistence, 1021
keys
encryption, 1041
property, 295, 306-309
keysExactMatch property, 295, 306-309
keyView documents, 324
labels, 94
controls, 94
naming, 408
Landscape mode, 700
lang property, configuring, 893
Language Direction Picker editor, 585
Language Picker editor, 585
language property, 289
languages, 9
deprecated codes, 898
formula, 479, 482
internationalization, 871. See also internationalization translations, 872
XUL, 608
LargeSmallStepInterface, 573
lastSubmit property, 245
launching. See starting layers
foundation, 3, 164
security, 1035-1049
applications, 1037-1039
design, 1039-1046
documents, 1046-1048
resources, 1036
servers, 1036-1037
workstation ECL, 1048-1049
SSL, 1037
State Management Layer, 1020, 1027-1032
Layout Container, 849
layoutContainer design pane, 855
layouts
applications, 849. See also Application Layout control container patterns, Custom Controls, 433-440
containers, 403
design patterns, 849-850
panels, 98-101
regions, 1047
tables, 101-102
views, 102-104, 338
lazy-load/preemptive algorithms, 1021
LCD (Lotus Component Designer), 4
levels
ACLs, accessing, 1037
of inheritance, themes, 812-814
libraries
@Functions, 479
applications, 750
installing, 749
internationalization classes, 894
JavaScript Library, 112-114
JSF tag, 161
resources, 824
Runtime, 894
scripts, 224
client-side, 891-893
localization, 890-893
server-side, 890-891
signing to run on behalf of someone else, 1055
system, 227-240
t.l.d files, 549
XPages Extension Library, 7, 11, 137-156, 606
Application Layout control, 851
applying, 26-32
Change Dynamic Content Action control, 142-144
dialog boxes, 147-151
Dynamic Content control, 138-141
JSON RPC service (remote service), 154-156
in place editing, 143-147
tooltip dialog box, 150-153
licenses, Domino Designer, 14
lifecycles
Inbound/Outbound stages, 982
JSF, customizing, 169
phases, 983
processing, 162
requests
JSF, 169-170
processing, 908-912
limiting server-side execution, 906
Link control, 639
link property, 638
$linkResource element, 818
properties, 820
resources, 820
links, 83-85
behavior, 641
configuring, 644
controls, 913
documents, View Panel, 346-348
Domino links/Notes links, comparing, 641-644
GET/POST requests, 914
Hierarchical Dominators, 943
resource types, 815
View control, 46
link target property, 639
listboxes, controls, 86-88
listings
(3.1) View control XSP markup, 44
(3.2) XSP button markup, 49
(4.1) sample HTML, 59
(4.2) sample XML, 60
(4.3) sample XHTML, 60
(4.4) XML documents with XHTML and XForms, 61
(4.5) setting Data properties, 66
(4.6) configuring ID property using xp:this, 66
(4.7) using CDATA sections with xp:this, 67
(4.8) computing values dynamically, 68
(4.9) computing values when pages load, 70
(4.10) complete computed values sample, 71
(4.11) data binding to a Notes document field, 72
(4.12) Domino document sample, 74
(4.13) Domino view sample, 75
(4.14) data context sample, 76
(4.15) edit box bound to a Notes document field, 78
(4.16) adding type ahead to edit boxes, 78
(4.17) multiline edit box bound to Notes document fields, 79
(4.18) rich text control bound to a Notes document field, 79
(4.19) date/time picker sample, 80
(4.20) date only, time only, and date plus time sample, 81
(4.21) using event handlers to submit XPages, 82
(4.22) responses to button clicks, 83
(4.23) opening pages with links, 84
(4.24) navigation rule sample, 85
(4.25) listbox sample, 86
(4.26) computed listbox sample, 87
(4.27) combo box sample, 88
(4.28) computed combo box sample, 90
(4.29) checkbox sample, 90
(4.30) radio button sample, 91
(4.31) checkbox group sample, 92
(4.32) radio button group sample, 93
(4.33) label sample, 94
(4.34) computed field sample, 95
(4.35) image sample, 82
(4.36) file upload sample, 96
(4.37) file download sample, 97
(4.38) panel sample, 99
(4.39) ACL sample, 100
(4.40) table sample, 102
(4.41) view sample, 103
(4.42) view with two pagers, 104
(4.43) data table sample, 105
(4.44) repeat sample, 106
(4.45) repeat data table columns sample, 107
(4.46) repeat radio buttons sample, 109
(4.47) include page sample, 110
(4.48) tabbed panel sample, 110
(4.49) section sample, 111
(4.50) Script Library sample, 113
(4.51) style sheet sample, 115
(4.52) resource bundle sample, 116
(4.53) Dojo module sample, 116
(4.54) generic head resource sample, 117
(4.55) metadata resource sample, 117
(4.56) converter sample, 119-120
(4.57) validator sample, 122-127
(4.58) current document sample, 129
(4.59) simple action sample, 115
(4.60) action group sample, 131-133
(4.61) confirm sample, 136
(4.62) client-side scripting sample, 135
(4.63) HTML sample, 136
(4.64) dynamic content sample, 138
(4.65) Change Dynamic Content Action sample, 142
(4.66) in place form sample, 144
(4.67) in place form sample markup, 145
(4.68) dialog sample - opening, 147
(4.69) dialog sample - xe:dialog tag, 149
(4.70) tooltip dialog sample, 151
(4.71) tooltip dialog content, 152
(4.72) JSON RPC service sample, 154
(5.1) sample HTTP servlet, 160
(5.2) sample JSP with JSF tags, 161
(5.3) sample variable resolver, 167
(5.4) variable resolver configuration, 168
(5.5) variable resolver sample XPage, 168
(5.6) length validator client-side JavaScript, 174
(5.7) JavaScript value binding expressions, 180
(5.8) JavaScript method binding expressions, 180
(5.9) client ID binding expressions, 181
(5.10) multipart value binding expressions, 181
(5.11) simple action method binding expressions, 181
(5.12) XPages sample to display current date/time, 190
(5.13) server simple action updating current date/time, 191
(5.14) server JavaScript updating current date/time, 192
(5.15) client simple action updating current date/time, 193
(5.16) JavaScript rendered by xp:executeClientScript, 193
(5.17) client JavaScript updating current date/time, 194
(5.18) renderkit-specific client script handlers, 195
(5.19) using immediate property for cancel buttons, 197
(5.20) change document mode to editable, 199
(5.21) confirm before deleting documents, 200
(5.22) create response document, 201
(5.23) deleting documents selected in views, 203
(5.24) executing client scripts, 204
(5.25) executing server scripts, 204
(5.26) modify field being invoked after page has loaded, 205
(5.27) open documents for editing, 206
(5.28) publishing column values as component property, 207
(5.29) publishing column values as component properties, 209
(5.30) saving two documents, 210
(5.31) saving documents one at a time, 211
(5.32) change component mode to edit and view mode, 214
(5.33) setting values in the viewScope, 215
(5.34) using action groups, 216
(5.35) send mail simple action, 218
(5.36) change dynamic simple action, 219
(5.37) Hello World XPage, 221
(5.38) ViewUtils script library, 223
(5.39) changing pass-through text, 225
(5.40) including facets in view inspector outline, 226
1110 Index

(11.6) dojo.require() statement, 465
(11.7) XPage markup showing extended button control, 465
(11.8) dojoIntegration markup, 469
(11.9) dojoIntegration markup, 471
(11.10) getDatabasePath() function, 472
(11.11) mxpd.data.ViewReadStore widget, 476
(11.12) mxpd.ui.ViewTree widget, 476
(11.13) markup for previewHandler, 477
(11.14) inputTexts XSP markup, 480
(11.15) @Random function, 481
(11.16) session.evaluate() function, 481
(11.17) agent.runWithDocumentContext() method, 483
(11.18) jAgent agents, 486
(11.19) agent.runWithDocumentContext() method, 488
(11.20) agent.runOnServer() method, 489
(11.21) faces-config.xml file, 490
(11.22) com.ibm.xpages.PreviewBean class, 492
(11.23) moreLink link in allDocumentsView control, 494
(11.24) standardRTE.xsp source markup, 504
(11.25) extendedRTE.xsp source markup, 506
(11.26) mxpd2.xsp-configuration source markup, 511
(11.27) mxpd2.component.InputRichText.java source markup, 517
(11.28) response HTML envelopes, 522
(11.29) response HTML envelopes, 522
(11.30) FacesAjaxComponent interface, 523
(11.31) AttachmentDialog.js source code, 525
(11.32) ImageDialog.js source code, 527
(11.33) extendedRTE.xsp showing axTargetUrl configuration, 530
(11.34) extendedRTE.xsp showing InputRichTextConverter configuration, 534
(12.1) implement standard methods, 547
(12.2) xsp-config file for tag specification, 548
(12.3) simple renderer implementation, 552
(12.4) registering renderers, 553
(12.5) simple properties, 557
(12.6) <group> snippet form bas .xsp-config, 570
(12.7) <group-type-ref> snippet from uispinner.xsp-config, 560
(12.8) MinMaxInput snippet, 560
(12.9) base .xsp-config, 563
(12.10) MinMaxPair.java, 566
(12.11) uispinner.xsp-config, 567
(12.12) complex properties, 569
(12.13) setting complex type properties, 569
(12.14) LargeSmallStepInterface.java, 570
(12.15) complex property, mx:this syntax, 572
(12.16) LargeSmallStepImp.java, 574
(12.17) uispinner.xsp-config (Final), 579
(12.18)UISpinner.java, 588
(12.19) UISpinnerRenderer.java, 592
(12.20) SpinnerBean.java, 599
(12.21) updated faces-config.xmo with managed beans, 600
(12.22) xSpinText.xsp, 600
(13.1) renderkit-specific properties, 620
(13.2) Wordsworth quiz, 623
(13.3) extended command extension point service, 625
(13.4) extended command extension point service, 627
(13.5) extended commands called using CSJS, 631
(13.6) link target property, 639
(13.7) client print-to-console debugging, 647
(13.8) Firebug Lite, 648
(13.9) computeWithForm, 654
(13.10) logic to compute runtime platforms, 661
(13.11) simple action for Search control, 666
(13.12) CSJS for Search control, 666
(13.13) JavaScript, receive events, 674
(13.14) HTML markup emitted for receive events, 675
(14.1) displaying User Agents, 678
(14.2) User Agent device detection, 682
(14.3) Device Beans, 683
(14.4) mobile XPage, 687
(14.5) application page transitions, 689
(14.6) Navigator XPage, 690
(14.7) Outline custom control, 691
(14.8) hierarchical navigation XPage, 693
(14.9) context-sensitive navigation XPage, 694
(17.1) XSP markup for
layoutContainer XPage, 855
(17.2) XSP markup for
layoutContainer XPage, 858
(17.3) XSP source markup for
appLayout XPage, 859
(17.4) XSP markup for
layoutContainer XPage, 862
(17.5) XSP markup for
layoutContainer XPage, 863
(17.6) XSP markup for
allDocuments XPage, 866
(17.7) XSP markup for
byMostRecent XPage, 867
(17.8) XSP markup for
layout Custom Control, 868
(17.9) XSP markup for
facetMiddle, 870
(17.10) XSP markup for
MiddleColumn, 870
(18.1) German translations, 878
(18.2) Chinese translation, 880
(18.3) translated resource
bundle after string changes, 882
(18.4) resource bundle adding
strings, 882
(18.5) resource bundle
removing strings, 883
(18.6) label without ID, 884
(18.7) two labels without IDs, 884
(18.8) resource bundle for two
labels without IDs, 885
(18.9) computed expressions
and JavaScript, 886
(18.10) using resource bundles
from CSJS, 890
(18.11) programmatically
loading resource bundles, 890
(18.12) referencing loaded
resource bundles, 891
(18.13) JavaScript class
representation of resource
bundles, 892
(18.14) JavaScript class
representation of resource
bundles, 892
(18.15) switching locales
programmatically, 894
(18.16) Arabic page, 896
(19.1) XSP markup for
linkReply link control, 913
(19.2) XSP markup for
viewTopic Custom Control, 915
(19.3) XSP markup for
buttonNewTopic control, 917
(19.4) XSP markup for
buttonSave, 921
(19.5) XSP markup,
dataCache sets, 928
(20.1) permission grant for
Toolbox to run Java code, 933
(20.2) SSJS code performing
CPU-intensive tasks, 936
(20.3) SSJS code performing
non-CPU-intensive tasks, 941
(20.4) SSJS calling arbitrary
backend utility functions, 944
(20.5) view, document, agent
operations, 945
(20.6) Java agents called from
SSJS code, 948
(20.7) Apply button XSP
markup and SSJS code, 951
(20.8) faces-config.xml, 952
(20.9) debugBean.java source
code, 954
(20.10) debugBeanPhaseListener.
java source code, 956
(20.11) testPhasesAndEventHandler
XPage, 960
(20.12) testRenderedAndLoaded1
XPage, 965
(20.13) Say hello! button, 971
(20.14) testRenderedAndLoaded2
XPage, 975
(20.15) Say hello oncemore!
button, 983
(20.16) Say hello onelastime!
button, 988
(20.17) XSP markup of
GET-Based, 998
(20.18) GET links in
testDynamicContent HTML
markup, 996
(20.19) XSP markup for
testMemoryDump XPage, 1009
(20.20) modified
source markup for
testMemoryDump, 1016
(20.21) testMemoryDump
with rendered=false, 1017
(20.22) Temporary Pages
Location, 1023
(20.23) Server and
Application Level XSP
properties for maximum
views, 1025
(20.24) XSP markup for
testPersistence1, 1029
(20.25) XSP markup for
testPersistence2, 1029
(21.1) check access to
underlying view, 1042
(21.2) XPages acl and
aclEntry syntax, 1043
(21.3) XPages acl property
element, 1044
(21.4) check ACL access
level, 1046
(21.5) hiding sections, 1048
(21.6) executing restricted
network operations, 1056
(21.7) sample ACF custom
configuration, 1064
(21.8) using sessionAsSigner,
1068
(21.9) java.policy file, 1069
(21.10) grant declaration for
Java code, NSF, 1071
(21.11) grant declaration for Java code, via SSJS in NSF, 1071
<theme> element, inheritance path, 814
lists
applications, adding, 651
field properties, 37
modifying, 1052
properties, 63
reader access, 1047
loaded property, 195, 219, 256, 295, 316, 964-980, 1043-1046
<bundle> element,
<dojoModule> element, 819
<linkResource> element, 820
<metaData> element, 821
<script> element, 819
<styleSheet> element, 820
loading
applications
optimizing, 650
translations, 893
Java, Notes client, 1059
JSP engines, 161
restrictions, 1041
Load Java code permission, 612
locales
deprecated locale codes, 898-900
identifiers, 771
programmatically, switching, 894
sensitive data, 893
in XPages, 894-897
Local folders, 544
localization, 870
applications, testing, 877-878
computed expressions/
JavaScript, 885-890
computed fields, 900
configuring, 874-876
control IDs, 884
CSJS, 889
gotchas, 883
options, 872-886
resource bundle files, 873-874
script libraries, 890-893
strings, 876
locations, Java source code folders, 544
locking, Allow Document Locking feature, 269
log() command, 759
logging out, 1037
logic, 187
adding, 187-197
client-side simple actions, refreshing using, 192-193
CSJS, refreshing using, 193-194
event handler properties, 194-197
forms, executing, 263-266
servers, adding, 244-245
server-side simple actions, refreshing using, 190-191
simple actions, 198-218
SSJS, 191-192, 733
xp:eventHandler tags, applying, 190
logs
Java, enabling, 755-759
strings, 959
troubleshooting, 645
loops
for, 937
infinite, 812
Lotus Component Designer. See LCD
Lotus Notes Template Development ID file, 1050
LotusScript, 10

M
Main Topic form, 36
field definitions, 38
hidden fields, 264
managed beans, 490-496. See also backing beans declaring, 490
MANAGER access level, 1037
managing
Application Management menu, 29
applications, Notes client, 637-641
CSS style sheets, 782
documents, 40, 266-272
Dojo, 463
interface themes, 795
properties, 538, 835
rich text, 641
state, 1007
State Management Layer, 1020, 1027-1032
tabs, 638
themes, resources, 815
translated strings, 873
windows, 638
maps
attributes, implementing, 555
objects, 1080
Margins tab, 771
markSubjectConfidential property, 217
markup
adding, 136
for alpha XPages, 435
for beta XPages, 438
buttons, 49
comments, 266
controls, 77
Custom Controls, 405
Data Tables, 374
documents
data source, 254
locking, 271
dojointegration, 469, 471
editing, 64
EL data binding, 372
extendedRTE.xsp source, 506
extension point service, 625
HTML, receiving, 787
inputText2, 480
for layoutContainer controls, 436
mxpd2.component.
InputRichText.java source, 517
mxpd2.xsp-config source, 511
for omega XPages, 439
for previewHandler, 477
for rich text control, 287
showing extended button control, 465
Index 1115

Index

Single Page Application, 685, 687-688
themes, 703-710
   Data View control, 704-705
   Form Table control, 706
   Outline control, 706
   styling controls, 707-710
User Agents, device detection, 682-683
mobile applications, programming, 678-685
mobile controls, 685
Single Page Application design patterns, 686-687
styles, 707-710
models
   components, UIs, 170
   development, 11
   memory, 1005-1008
   object, 221-227
   relational data, 41
Model-View-Controller. See MVC
mode property, 198, 213
modes, partial execution, 919-922, 986-992
modify field simple action, 205
modifying
   application launch options, 642
   backgrounds, 771
   component behavior, 557-558
Compu...
New Property action, 413
New Script Library dialog box, 62
New Source Folder dialog box, 543
New Themes dialog box, 805
New Topic button, 36
New XPage dialog box, 20
NO ACCESS access level, 1037, 1065
nodes, adding items, 858
non-primitive properties, 568. See also complex properties
nostate option, 1032
Notes client, 607-608
applications formatting, 619-621
managing, 637-641
searching design elements, 634
behavior, 632-633
bookmarks, 614-616
composite applications, 664-676
differences, 621-624
disableModifiedFlag property, 634-637
ECLs, 1059-1062
enableModifiedFlag, 634-637
extended CSJS commands, 624-632
installing, 14
Java, loading, 1059
launch options, 612-614
Notes links versus Domino links, 641-644
optimizing, 649-664
overview of, 608-609
previewing in, 21, 25
rendering, 361-365
security, 1058-1062
starting, 610-612
styles, applying, 775
stylingWithExtendedClasses XPage, 793
View Panel, rendering, 361-365
working offline, 616-619
XPiNC debugging tips, 645-648
Notes/Domino building blocks, 478-489
agents, 482-489
in-memory documents, 482-489
profile documents, 482-489
Notes Storage Files. See NSFs
NotesXspDocument class, 236
NotesXspViewEntry class, 236
NSFs (Notes Storage Files), 253
classloaders, 1058
components, adding, 670
SCXD, 653
NullPointerException, 748
null pointers, testing exceptions, 746
Number class, 235
NumberConverter, 173
Number Format editor, 586
number spinners, 539

Object class, 235
object models, 221-227
objects
compositeData, 421-428
debugBean, 951
global, 227-240
maps, 1080
MYAPP, 733
sessionAsSigner, 1067
sessions, 1037
viewScope, 907
XMLHttpRequest, 455
Object Technology International. See OTI
offline (working), Notes client, 616-619
OFF statement, 757
omega XPages, markup for, 439
onclick events, 321
onComplete events, 458
onComplete property, 455
OnError events, 458
onError property, 455
OneUI Application configuration, 854
onStart events, 458
onStart property, 455
opening sample documents, 37
OpenNTF.org, 6, 7
open page simple action, 205-207
operating systems. See OSs
optimizing. See also performance
applications, loading, 650
behavior, 918
Data View control, 392-394
mobile application navigation, 690
Notes client, 649-664
performance, 15
requests, 931-949
scalability, 906, 1007, 1020-1034
start-time, 685
themes, 794-847
views, 337
options
Application Page, 720
applications adding logic, 189
configuring, 853
delta, 1032
deltaex, 1032
ESA, 1061
Events view, 446
forms, accessing, 1040-1041
fulltree, 1032
GZip persisted files, 1026
localization, 872-886
MyProfile, 619
nostate, 1032
Partial Update, 446
preload, 652
Server Options, 988
SSJS conditional breakpoints, 744
views, accessing, 1041-1042
Oracle JDeveloper, 3
orientation-based interaction, 697-701
orientation property, 699
OSs (operating systems), 1020
OTI (Object Technology International), 17
Index

Outbound stage, 982
outerStyleClass property, 791
Outline control, mobile themes, 706
outlines, 21
Outline view, 402
out-of-the-box style, partial refresh, 445-452
output, Domino server consoles, 960
override behaviors, <control> element, 838
Override on Notes setting, 809
Override on Web setting, 809
override property, 839
overriding
    rendering, methods, 547
    User Agents, Safari, 680-681

P
p2 Std button, 963
Package Explorer
    Domino Designer, adding, 541
    enabling, 473
    resource bundle files
        exporting, 879
        importing, 881
packages
    Java, 492, 629
    statements, 628
pageName property, 346
Pager
    elements, 44
    partialExecute property, 923
    properties, 373, 395-398
    viewing, 102
pagerBottom* property, 389
pagerTop* property, 389
palettes, controls, 21
panels, 57, 98-101
    loaded/rendered properties, 967
Pager property, 373
Style properties, 771
tabbed, 110
View Panel, 340-341
    applying images to columns, 348-351
categories, 357-360
    formatting, 343-345
    linking documents, 346-348
    navigating, 345
    properties, 366-370
    rendering Notes client, 361-365
    sorting, 351-357
paraphraphs, 32
parameters
    Event Parameters, 460-462
    events, 447
    HTTPJVMMaxHeapSize, 924
    HTTPJVMMaxHeapSizeSet, 924
    querystring, 450
    suspend, 749
parameters property, 196, 206, 219
    searchValue, 293
    params property, 455
    paramValues variable, 166
    param variable, 166
    parentId property, 200, 256, 295, 304
    parseVersion() method, 829
    partialExecute property, 923
    partial execution mode, 919-922, 986-992
partial refresh, 137
AJAX, 444-460
    out-of-the-box style, 445-452
    performance, 460
    applying, 906
    doing-it-my-way style, 453
    out-of-the-box style, 445-452
    performance, 917-918
    request introspection, 981-986
    Request Introspection (XPages), 981-986
    partialRefreshField element, 447, 452
    partialRefreshGetField Computed Field, 456
    partialRefreshGet utility functions, 454-456
partialRefreshPost utility functions, 458
Partial Update option, 446
Part Session XML Dump file, 1011
pass-by-references, accessing, 417
passwords
    Internet name and, 1039
    name-and-password authentication, 1037
    server layer security, 1037
    working offline, 619
Password Value editor, 586
paths
    classes, extending Dojo, 466-468
databases, 296
Dojo module, 116
Java Build Path editor, 491
resources, themes, 824
patterns
    aggregate containers, Custom Controls, 432-433
    composite, 545
design
    application layouts, 849-850
    formality, 850
    layout container, Custom Controls, 433-440
    Repeat control, 383
    SCXD, 652
    Single Page Application design, 685-688
pausing debug mode, 744
performance, 905, 931
    bootstrapping costs, 614
    CPUs, reducing utilization, 912-922
    Dojo debuggers, 764
    inline CSS styles, 779-786
    memory, 923-928. See also memory
    optimizing, 15, 649-664
    partial execution mode, 919-922
    partial refresh, 460, 917-918
requests
  Dynamic Content controls, 992-1004
  loaded/rendered property, 964-958
  optimizing, 931-949
  partial execution, 986-992
  partial refresh, 981-986
  PhaseListeners, 949-1004
  processing lifecycles, 908-912
  scalability, 1004-1034
  XPiNC, 656-657
permissions
  Load Java code, 612
  programmability restrictions, 1053-1056
  runtime, 1062
  views, 1041
  XPages Toolbox, 932-933
persistence
  Disk, 1020-1026
  disk, 907
  RAM, 1021-1022, 1024-1026
  xsp.persistence.* properties, 925-926
  perspectives, switching debug, 739
  PhaseListeners, requests, 949-1004
phases
  lifecycles, 983
  RENDER_RESPONSE, 970, 982
  RESTORE_VIEW, 970, 982
  testing, 972
pickers
  Partial Execution control, 989
pipe (), 87
Place Bar actions list, 858
place names, adding, 857
Platform Level, themes, 797
platforms
  current, detecting, 830
  support, 620
PLATO Notes, 4
plug-ins
  configuring, 625
  debugging, 752
  Dojo, 826
  Eclipse, 5
  extended command, installing, 630
  importing, 753
  installing, 29
  viewing, 631
  XULRunner, 609
policies, java.policy files, 1069-1071
Portraits mode, 700
ports, 1037. See also connecting
postback requests, 322
postNewDocument property, 256
postOpenDocument property, 256
  events, 731
postOpenView property, 295, 316
POST requests, 906, 908, 910-912
  CPU utilization, reducing, 912-917
  introspection, 972
postSaveDocument property, 256
prefixes
  Custom Controls, 407
  mobile themes, 686
  specifying, 61
preload options, 652
presentations
  modifying, 778
  separation between data/structure, 770
  themes, enabling, 841
  tiers
    JSP, 161
    XML-based, 169
  view columns, 366
preventing Scala expression, 217
preventing View Panel access, 1042
previewHandler, markup for, 477
previewing
  allDocuments XPage, 865
  applications, 35
  applyLayout Xpage, 860, 864
buyMostRecent XPage, 865
Chinese translations, 881
custom expressions, 889
Custom Controls, 404, 409
Data Tables, 378
German translations, 878
locales/timezones, 235
Main Topic form, 36
in Notes Client, 21, 25
properties, 834, 836
pseudo translations, 877
repeat control, 108, 382
resource themes, 823
styles, 781
themes, 813
titles, 426
View control, 44
views, 43
in web browsers, 22-24, 25
XPages, 751
Preview in Notes option, 21
print() function, 647
printing
disabling, 1041
Println() statement, 961
print() method, 8
privileges, configuring, 1057
Problems view, 425
processing
  applications, 614
  browsers, 918
  components, trees, 918
  dividing, 982
  events, 446
  lifecycles, 162, 169-170
  lifecycles phases, 983
  real-time views, 950
  requests, lifecycles, 908-912
Process Validations, 163
(Profile) function, 945
ProfilerAggregator class, 949
profilers, 932-949
backend, 946-948
CPU, 936-942
wall time, 942-946
Say hello one last time!
  request, 990
  snapshots, 969, 977
profiles
anonymous access, 620
blocks, 945, 948-949, 967
building blocks, 482-489
testRenderedAndLoaded
XPage, 968
Profile view, Data Tables,
376-381
programmability restrictions,
1053-1056
programming
client-side/server-side code,
combining, 444
documents, 283-286
locales, switching, 894
mobile applications, 678-685
references, 1075
Java API classes, 1078
Java classes, 1076-1077
JavaScript pseudo classes,
1078-1080
XSP tags, 1075
resource bundles, loading,
890-891
projects
cleaning, 874
formatting, 629
prompt() function, 622
properties. See also specific
properties
Application Properties,
editing, 874
<bundle> element, 818
bundles, exporting, 873
complex, 64-66, 847
link controls, 913
specifying, 568-579
components
adding, 556
revising in Domino
Designer, 568
computed, 67-71
controls, 76
Custom Controls, 885-886
customization, 289
customizing, 413
Data, configuring, 66
Data View control, 389
defining, 414
document data sources, 282
<dojoModule> element, 819
escape, 69
event handlers, 194-197
fields, 37
getters, 578
groups
Custom Controls, 430-432
matching definitions,
565-568
<linkResource> element, 820
managing, 538
<metaData> element, 821-823
Mobile controls, 686-687
navigation, 50
previewing, 834, 836
removeRepeat, 107
rendered, 136, 964
renderkit-specific, 620
resource types, 815
<script> element, 819
sheets, 21
simple, 63-65, 557-558
State Management Layer,
1028
styles, 772
themeld, 835-837
themes, 832-848
control definitions,
838-839
types, modifying, 562
UI components, 555-562, 555
values
computing, 68
managing, 835
View Panel, 366-370
views, 366-370
.xsp-config files, inheriting,
558-562
.xsp-config tags, 559
.xsp.persistence.*
Properties editors, 857
properties property, 638
Properties view, 412
Property Broker access, 1050
<property-class> tag, 559
Property Definitions, 411-421
<property> element, 832
<property-extension> tag, 559
<property-name> tag, 559
Property tabs, 415-417
<property>tag, 556, 559
pseudo translations, viewing, 877
Public Access, 1065-1066
checking, 1066
configuring, 1065-1066
users, 1041
public no-args constructors, 556
publish component property
simple action, 207
publishing events, 672-675
publish view column simple
action, 208-209
queries, 299, 302. See also
searching
executing, 302, 324
multiterm full text, 303
queryNewDocument property,
256
queryOpenDocument property,
256
queryOpenView property, 295,
316, 318
querySaveDocument property,
256
querystring parameters, 450
quick testing applications, 554
radio buttons
controls, 91
groups, 93
RAM (random access memory),
1005
configuring, 1023
Disk persistence, selecting,
1020-1021
optimizing, 906
persistence, 1021-1022,
1024-1026
RCP (Rich Client Platform), 17, 157, 609, 620
cpinstall.properties, 756
READER access level, 1037, 1042
reader access lists, 1047
reader fields, 1047
reading directions, detecting, 831
readMarkClass property, 793
readOnly property, 906, 914
readonly property, 1045-1046
real property, 820
real-time views, processing, 950
rebuilding projects, 874
receiving events, 672-675
HTML markup, 787
reducing CPUs cycles, 1004
utilization, 912-922
memory utilization, 923-928
redundancy Application Layout control, 849
inline styles, 780
theme resources, 815
references JSF, 165
programming, 1075
Java API classes, 1078
Java classes, 1076-1077
JavaScript pseudo classes, 1078-1080
XSP tags, 1075
style classes, 780
tags, 73
types and components, 570
XSL style class, 1081
Reference tab, 227
reformatting tags, 50
refreshId property, 196, 918, 984
refreshing Event Parameters, 460-462
image controls, 743
partial refresh applying, 906
performance, 917-918
request introspection, 981-986
refreshMode property, 196, 984
RegExp class, 235
regions, layouts, 1047
registering
backing beans, 599-600
renderers, 551-553
registries, Domino Designer, 570
Regular Expression editor, 586
relational data models, 41
relationships, HTML tags, 61
Release Line Picker editor, 586
releases, XPages Extension Library, 27
reloading browsers, 445
remote access, debugging Web Inspector, 713-716
remote servers, 653
removeRepeat property, 107
removing
scripts, tags, 1064
strings, 883
rendered property, 136, 196, 964-980, 1045-1046
<bundle> element, 818
<dojoModule> element, 819
<linkResource> element, 820
<metaData> element, 821
<script> element, 819
<styleSheet> element, 820
renderers
classes, 539
creating, 551-553
customizing, 591-596
render-independent properties, 555. See also properties
rendering
categories, 360
HTML, 59
memory, 1018
methods, overriding, 547
Notes client, 361-365
XULRunner, 609
renderkits, Run ON Server, 661
renderkit-specific properties, 620
Render Response, 163
RENDER_RESPONSE phase, 969, 982
Repeat control, 381-387
formatting, 383
nested repeats, 384-386
rich text, 386-387
repeat control, 106-109
Repeat controls, 339
replica IDs, moving, 296
replication, working offline, 616
Reply button, 429
replyButton control, 427
reports
backend profilers, 947
CPU profilers, 938-940
wall time profilers, 944
Request Introspection (XPages), 949-1004
Dynamic Content controls, 992-1004
loaded/rendered properties, 964-980
partial execution, 986-992
partial refresh, 981-986
requestParameterPrefix property, 295, 313
requestParamPrefix property, 256, 274
requests, 912-917. See also Request Introspection (XPages) detecting, 815
GET, 906, 908, 909-910
CPU utilization, reducing, 912-917
Network tab, 999
JSF, 165
loaded/rendered properties, 964-980
optimizing, 931-949
partial refresh, 981-986
PhaseListeners, 949-1004
POST, 906, 908, 910-912
postback, 322
processing lifecycles, 169-170, 908-912
scalability, 1004-1034
security, checking, 1047
state management between, 555
saving, 556
requestScope variable, 166, 231
Required field, 423
requirements, fulfilling customer, 496-535
reserved areas. See facets
Resource Aggregation, enabling, 907
<resource> elements, 823
child element support, 816
within mxpd themes, 816
rendered running Notes client, 830
rendered to iPhone, 830
types, support, 818
resources, 111-117
<bundle> element, 818
bundle files, 115, 876
adding, 887-888
editing, 878
exporting, 878-880
importing, 880-881
localization, 873-874
composite applications, 675-676
design, 11
dojoModule, 464-466
Dojo module/Dojo module
path, 116
<dojoModule> element, 819
generic heads, 116
JavaScript Library, 112-114
JSF, 159
libraries, 824
<linkResource> element, 820
metadata, 117
<metaData> element, 821
mobile applications, 677
<script> element, 819
security layers, 1036
<styleSheet> element,
style sheets, 114-115, 781
support, 770
themes. See themes
types, support, 815
websites, 1087
<resources> element, 823
Resource Servlet, 824
response documents, creating, 258-263
restoreState() method, 556, 558,
1020, 1033
Restore View, 163, 169
RESTORE_VIEW phase, 969,
982
restoring state, 556, 578
restricting. See also security
access, 1042-1045
loading, 1041
operations, security,
1056-1057
programmability restrictions,
1053-1056
search results, 301
REST service, 327-330
results
computewithForm feature,
655
of full text searches, 300
multiple key filter, 309
restricting, 301
sorting, 324
retrieving documents, 315-317
reusable user-interface
components, 158
reuse, inheriting xsp-config
properties, 558
rey property, <linkResource>
element, 820
Riand, Philippe, 932
Rich Client Platform. See RCP
rich documents, 286-291
rich text
controls, 79-80
managing, 641
Repeat controls, 386-387
Rich Text Editor, 497, 1063
right property, 1044
rights, users, 1053-1056
Rounded List Item, 689
rowClasses property, 793
rows
Data Tables, 370
property, 312
rules
ACF, 1064
navigating, defining, 50
navigation, 85
performance, 906-907
Run as Web user, 1037
running
CPU profilers, 936-942
unrestricted methods,
1054-1055, 1057
XPages, restricting access,
1042-1045
Run Non-CPU Intensive Task
button, 942
Run On Server, 657-664
Run Some Task button, 943
runtime
architecture, 1005
built-in translations, 893
Custom Controls, 421
documentID, checking for,
1066
ebraska, queryOpenView
property, 318
logging levels, 755-756, 758
Notes client, 608
permissions, 1062
script library, 234
stacks, 31
testDynamicContent XPage,
993
Runtime library, 894
S
Safari, Mobile Application
Development, 680-681
Sametime clients, 15
c粉碎Bundles() method, 891
sample documents, 36, 37
save component mode simple
action, 213-214
data sources simple action,
209-211
data document simple action,
211-213
saveLinkAs property, 256, 644
c粉碎 property, 196
c粉碎State() method, 556, 558,
1020
saving
dirty documents, 633
documents, 83
MIME documents, 643
state, 556, 578
translations, 883
Say hello! button, 970
Say hello once more! button, 983
Say hello one last time! button, 987
scalability, 905
configuring, 1007
memory
analyzing, 1008–1019
models, 1005–1008
optimizing, 906, 1007,
1020–1034
requests, 1004–1034
scheme property, <metaData>
element, 821
scope
property, 256, 295, 316
rules for Custom Controls,
402
of themes, 795
Script Editor, 69, 587
<script> element, 818
properties, 819
resources, 819
script libraries, 224
script property, 203
scripts, 443
AJAX
partial refresh, 444–460
partialRefreshGet utility
functions, 454–456
partialRefreshPost utility
functions, 458
applications
frameworks, 443–444
building blocks, 478–489
agents, 482–489
in-memory documents,
482–489
profile documents,
482–489
client-side, 134–136
Dojo
customizing widgets,
470–475
dojotype property, 466
dojoModule resources,
464–466
integration, 463–478
standard widgets, 468–470
when an XPage is not an
XPage, 475–478
widgets, 466–468
Event Parameters, 460–462
fulfilling customer
requirements, 496–535
libraries
client-side, 891–893
localization, 890–893
server-side, 890–891
signing to run on behalf of
someone else, 1055
managed beans, 490–496
servers, 189
tags, removing, 1064
scrolling, 717–718. See also
navigating
SCXD (Single Copy XPages
Design), 652–654
Search dialog box, 634
searchExactMatch property, 295,
302
searchFuzzy property, 295, 302
searching
components, formatting,
664–666
design elements, 634
full text search properties,
299–304
Google search widgets,
creating, 666–668
multiterm full text queries,
303
sorting, combining, 323
searchList property, 295, 303
searchMaxDocs property, 295
search property, 295, 300
searchQuery property, 672–675
searchValue parameter, 293
searchVariants property, 295, 302
sections, 111, 1047
hiding, 1048
secure sockets layer. See SSL
security, 1035
ACF, 1062–1065
ACLs, implementing,
1051–1052
alerts, exceptions, 1061
applications, formatting,
1049–1051
applying, 1049–1053
checking, 1057–1058
design, configuring privileges,
1057
documents, locking, 269
ECLs, 1058, 1059–1062
Java JVM vulnerabilities, 612
java.policy files, 1069–1071
layers, 1035–1049
Notes client, 1058–1062
passwords, working offline,
619
programmability restrictions,
1053–1056
Public Access, 1065–1066
restricted operations,
1056–1057
sessionAsSigner property,
1067–1068
XPages
signing, 1052–1053
Toolbox, 934
XULRunner, 1058
segmented buttons, headings
with, 695
selecting
Disk persistence/RAM,
1020–1021
exceptions, 747
View controls, 337–340
XULRunner, 608
selection controls, 85
send mail simple action, 217–218
serialization, 1027, 1032
Server Options, 988
servers
accessing, 1038
applications, debugging,
726–759
clients
configuring, 15
synchronizing, 159
data in JavaScript clients, 243-244
logic, adding, 244-245
Run On Server, 657-664
security layers, 1036-1037
simple actions, 189
working offline, 617. See also working offline
server-side execution, limiting, 906
Server-Side JavaScript. See SSJS
server-side processing, dividing, 982
server-side script libraries, 890-891
servlets, HTTP, 160
sessionAsSigner property, 1067-1068
sessionAsSignerWithFullAccess variable, 186
Session Dumps tab, 1008
session.evaluate() function, 481
sessions
authentication, 1037
dumps, 1030
variables, 186
sessionScope variable, 166
setRenderType() method, 547
setValue() method, 558
set value simple action, 214-215
SEVERE statement, 757
Shape Type Picker editor, 586
sharing consistent styles, 780
sheets, properties, 21
showCheckBox property, 341
showContent() method, 993
Show View dialog box, 542
signing agents
to run on behalf of someone else, 1055
to run on behalf of the invoker, 1055
documents, 1048, 1051
programmability restrictions, 1053
script libraries to run on behalf of someone else, 1055
sessionAsSigner property, 1067-1068
unrestricted methods, 1054
XPages, 1052-1053
simple actions. See actions, simple
simple properties, 63-65, 557-558
Single Copy XPages Design. See SCXD
Single Page Application
controls, 687-688
design patterns, 685-688
controls, 687-688
Mobile controls, 686-687
Single Page Application Wizard, 716, 718-722
skins
customizing, 289
property, 289
snapshots, wall time profilers, 969, 977
Solid State Drives. See SSDs
sortColumn property, 295
sorting
columns, 307, 323-326
results, 324
searching, combining, 323
View Panel, 351-357
sortOrder property, 295
source code. See also code
AttachmentDialog.js, 525
ImageDialog.js, 527
importing, 752
Java, adding, 543-544
Source view, 63
special_profile() method, 969
specification, creating tags, 547-550
specifying
complex properties, 568-579
IBM OneUI themes, 604
prefixes, 61
simple properties, 557-558
speeding up browser processing, 918. See also performance
src property
<bundle> element, 818
<script> element, 819
SSDs (Solid State Drives), 1023
SSJS (Server-Side JavaScript), 8, 221, 260
conditional breakpoints, 742-745
debugging, 735-740
Java logging, enabling, 755-759
resource types, 815
sessionAsSigner property, 1067
troubleshooting, 742
try/catch/finally blocks, 729-735
SSL (secure sockets layer), 1037
Stack Overflow, 7
stacks
runtime, 31
troubleshooting, 645
standalone installations, 15. See also installing
standard methods, implementing, 547
standardRTE.xsp source markup, 504
standards, 3
standard user-interface components, 176-179
Standard Widget Toolkit. See SWT
starting
applications, modifying
launch options, 642
mobile applications, 678-685
Notes client, 610-612
bookmarks, 614-616
launch options, 612-614
startKeys property, 295, 310
start-time, optimizing, 685
Start Wall Time Profiler button, 943, 967
state
managing, 538, 1007
renderers, registering, 553
requests, saving, 556
restoring, 556, 578
saving, 578
StateHolder interfaces, 556, 577
State Management Layer, 1020, 1027-1032
statements
alert(), 759
ALL, 757
CONFIG, 757
debugger, 741
dojo.require(), 465
grant, 1070
import, 628
INFO, 757
OFF, 757
packages, 628
PrintIn(), 961
SEVERE, 757
WARNING, 757
storage
Java source files, 543
NAS, 1023
NSFs, 253
rich text content, 79
String class, 235
strings
adding, 882
concatenation, avoiding, 887
deleting, 883
hardcoding, 872
localization, 876
localizing, 887
logs, 959
modifying, 882
translated, managing, 873
String Value editor, 586
structures
All Documents view, 383
separation between data/presentations, 770
themes, architecture, 796-797
XPages Extension Library, 30-32
Style Class editor, 586
styleClass property
attributes, 587
computing, 788
extending, 790-793
formatting, 785-787
<linkResource> element, 820
<style> Editor, 65, 586
Style properties panel, 771
Button control, 786
style property
applying, 777-778
computed values, 778
extending, 790-793
<linkResource> element, 820
manually configuration, 776-777
styles. See also themes
classes, 1081-1083
applying, 779-793
CSS files, 1081-1082
controls, 76
Custom Controls, 407
inline CSS, performance, 779-786
mobile controls, 707-710
modifying, 774
Notes client, applying, 775
pagers, 395-398
partial refresh
doing-it-my-way style, 453
out-of-the-box, 445-452
previewing, 781
properties, 772
resources, libraries, 824
themes
CSS, 771-778
overview of, 794
<styleSheet> element, 818
properties, 820
resources, 819
style sheets, 114-115
stylingWithExtendedClasses
XPage, 793
subcategories, viewing, 357-360
subject property, 217
Submit button, 48
submitLatency property, 245
submit property, 196
summaryColumn property, 389
summary data, documents, 40
support
ACF, 1062
Application Page types, 719
BiDi, indented categorization
with, 363
<bundle> element, properties, 818
CRUD, 47-53
Custom Controls, 770
<dojoModule> element, properties, 819
foreign languages, 874
IBM, 7
<linkResource> element, properties, 820-823
<metaData> element, properties, 821-823
properties, 555
<resource> elements, 816, 818
resources, 770, 815
<script> element, properties, 819
styleClass property, 788
<styleSheet> element, properties, 820
suspend parameter, 749
switching
debug perspectives, 739
locales programmatically, 894
views, 685
SWT (Standard Widget Toolkit), 159, 623
sync processes, working offline, 616
syntax
complex properties, 569
tags, 9
XML, 62-63
system libraries, 227-240
T
tabbed panels, 110
tables, 101-102
data, containers, 105-106
Data Tables, 370-381
documents, accessing, 74
tabManagement.xsp, 639
tabs
Background, 771
Dependencies, 625
Font, 771
managing, 638
Margins, 771
Network, GET requests, 999
Property, 415-417
Reference, 227
Session Dumps, 1008
Validation, 417-419
Visible, 419-421
tag library definition files. See .tld files
<tag-name> tag, 549
tags, 9
client-side scripting, 134
clouds, view category filtering, 298
controls, 76-98
data sources, 73-76
defining, 539
editing, 65
hierarchies, 311
HTML, 136, 1086
<html>, 61
JSF, 161
mx:uspinner, 554
programming references, 1075
properties, modifying, 557
reformatting, 50
scripts, removing, 1064
simple actions, 130
specification, creating, 547-550
syntax, 9
<theme>, 807
:view>, 63
<x:acl>, 1042
<x:acIEntry>, 1043, 1044
xp:actionGroup, 215
XPages, 72-73
xp:callback, 436
xp:changeDocumentMode, 198
xp:changeDynamicContentAction, 219
xp:confirm, 199
xp:createResponse, 200
xp:deleteDocument, 201
xp:deleteSelectedDocuments, 202
xp:dominoDocument, 254
xp:eventhandler, 189
xp:executeClientScript, 203
xp:executeScript, 204
xp:modifyField, 205
xp:openPage, 206
xp:publishViewColumn, 208
xp:repeat, 432
xp:save, 210
xp:savedocument, 211
xp:sendMail, 217
xp:setComponentMode, 213
xp:setValue, 214
xp:view, 411
xp:viewColumn, 348
.xsp-config files, 547, 549
XSP markup, 59-73
Tags field, 36
Tag View control, updating, 322
target property, 206, 640
<bundle> element, 818
<dojoModule> element, 819
<linkResource> element, 820
<metaData> element, 821
<script> element, 819
<styleSheet> element, 820
targets, drop, 861
Teamroom application, 327
templates
blank application, 540
Custom Controls, 433
Discussion, 33
Lotus Notes Template Development ID file, 1050
testAdvancedLifecycle XPage, 960, 962
TestCalendarOutline view, 327
testCPUvsWallTime XPage, 937
testDynamicContent XPage, 993
testing, 936. See also profilers accounts, 659
applications, 554
creating with UISpinner components, 597-604
localization, 877-878
for null pointer exceptions, 746
phases, 972
translation, 872
testLoadedAndRendered,
971-975
testMemoryDump XPage, 1009, 1014
testPhasesAndEventHandler XPage, 960
Test Phases button, 962
testRenderedAndLoaded1 XPage, 965, 967
testRenderedAndLoaded2 XPage, 975
text
entering, 36
formatting, 286
full text search properties, 299-304
hidden, viewing, 648
MIME, formatting, 641
rich controls, 79-80
Repeat controls, 386-387
ThemeControl interface, 173, 587
themeld property, 835-837, 844
theme property, 809
themes, 769
Application Layout control, 854
Application Level, 797
applying, 795, 804-807
architecture, 796-797
configuring, 797-805, 807-810
CSS, 771-778
<empty>, 810-812
Global directory, 825-826
HTML directory, 824-825
IBM OneUI, 604, 1086
inheritance, 796-797, 812-814
inline CSS styles, performance, 779-786
interfaces
development, 769-771
managing, 795
mobile, 686
Mobile Application Development, 703-710
Data View control, 704-705
Form Table control, 706
Outline control, 706
styling controls, 707-710
naming, 805
optimizing, 794-847
overview of, 794
Platform Level, 797
previewing, 813
properties, 832-848
controls, 839-848
definitions, 838-839
themeld, 835-837
resources, 814-821
bidirectional, 831-832
CSS, 825
Dojo plug-ins, 826
dojoTheme property, 827
paths, 824
previewing, 823
User Agents, 827-831
style classes, applying,
779-793
styleClass property
computing, 788
extending, 790-793
formatting, 785-787
Themes editor, 806
<theme> tag, 807
throwing exceptions, 730
time, data/time picker controls,
80-82
Time Zone Picker editor, 586
Tips and Tools, 298
titleLabel bindings, 426
title property, 424
<linkResource> element, 820
titles, 422
applications, 540
previewing, 426
.tld (tag library definition) files, 549
toolbars
bookmarks, invoking directly
in, 615
clients, 647
customizing, 288
defining, 287
Medium, 287
ToolBar (XPages), 932-933
Session Dumps tab, 1008
tools, 4
debugging, 725, 759. See also
debugging
Dojo, 463. See also Dojo
Domino Designer, 13-14. See
also Domino Designer
Eclipse, 17
Eclipse Memory Analyzer,
1019
new features, 4
partialRefreshGet utility
functions, 454-456
partialRefreshPost utility
functions, 458
profilers, 932-949
Toolbox (XPages), 932-933
troubleshooting, 648
tooltip dialog box, 150-153
topics
formatting, 36
Main Topic form, 36
Notes client, formatting, 632
samples, 36
viewing, debugging, 732
top-level XPages, Application
Layout control, 852
to property, 217
toString() method, 728
touch-based events, 696
touch-based interaction, 701-702
transitions, applications, 689
translations, 872. See also
internalization
applications, loading, 893
backups, 883
built-in
Dojo, 893
runtime, 893
Chinese, 880
German, 878
languages, 872
viewing, 877
translators, applying, 878-881
trees
components
processing, 918
state persistence, 925
JSF, 161
troubleshooting
ACF, 1062-1065
authentication, 636
bugs, 8
debugging
correcting, 738
SSJS, 742
iPhone User Agents, 679
memory, 1018, 1025
tools, 648
validators, 128
XPages Toolbox, 934
XPInc debugging tips,
645-648
try/catch/finally blocks,
debugging, 729-735
two views, 312-313
type ahead, enabling, 78
TypeAhead control, 463
type property, 207, 208, 1044
<linkResource> element, 820
<script> element, 819
types
Application Page, 719
components, referencing, 570
of containers, 98
items, naming, 857
of links, 84
properties, modifying, 562
<resource> elements, support,
818
resources, 815
of selection controls, 85

U
UIComponentBase class, 545
UIComponent class, 545
UIs (user interfaces), 10
component extensions
classes, 545-547
creating, 540-544
customizing renderers,
591-596
implementing UISpinner, 588-591
properties, 555-562
quick testing, 554
sample application creation, 597-604
components
building, 544-554
elements, 539
JSF, 176-179
models, 170
controls, creating, 538-539
JSF, 164
reusable user-interface components, 158
themes, 769. See also themes applying style classes, 779-793
computing styleClass property, 788
CSS, 771-778
development, 769-771
extending styleClass property, 790-793
formatting styleClass property, 785-787
optimizing, 794-847
UISpinner class, 545, 551, 579-582, 588-591
UISpinnerRenderer.java, implementing, 591-596
uispinner.xsp-config, updating, 566
unchecked values, 90
UNID (universal id), 39
uniform resource identifiers. See URIs
uniform resource locators. See URLs
universal id. See UNID
unreadMarksClass property, 793
unrestricted methods
running, 1054-1055
signing, 1057
Update Model Values, 163
update sites
formatting, 630
installing, 630
updating
AddDatabaseToWorkspace, 632
documents, managing, 266-272
Events view, 446
TagView control, 322
uispinner.xsp-config, 566
Upgrade Pack, 8
uploading files, 96
URIs (uniform resource identifiers), 407
URLs (uniform resource identifiers) bookmarks, 615
documents, controlling parameter usage, 258
fragments, 996
Notes client launch, 612
use cases, CPU profilers, 937
use hash feature, 138
User Agents
iPhone, 678
Mobile Application Development, 682-683
modifying, 679
resources, 827-831
Safari, 680-681
Switcher add-on, 682
viewing, 678
user interfaces. UIs
usernames, 1037. See also authentication; security
users
credentials, 1041
Public Access, 1041
rights, 1053-1056
Run as Web, 1037
servers, accessing, 1038
utilization
CPUs, reducing, 912-922
memory, reducing, 923-928

V
validateAllFields property, 245
validation
enableModifiedFlag property, 636
events, 446
Validation tabs, 417-419
validators, 121-127, 174-175
ValueBindingObjectImpl interfaces, 577
ValueBindingObject interfaces, 577
ValueHolder interfaces, 596
value property, 205, 207, 214
values
accessing, 75
binding, 179
complex, 66-67
computed, style property, 778
converters, 118-120
dataCache property, 928
iPad Device Bean, 684
properties
computing, 68
managing, 835
variables
currentDocument, 129
default, JSF, 166, 182-183
requestScope, 231
XPagesPreload, 650
var property, 198, 201, 205, 206, 211, 256, 295
<bundle> element, 818
versions, 4
caches, 787
Domino Designer, 13
Windows, Safari, 680-681
XULRunner, 610
vertical capabilities, 1004
videos, composite applications, 675-676
View controls, 42, 639
behavior, 640
columns, formatting, 341-343
drag-and-dropping, 314
links, 46
navigating, 340-370
previewing, 44-45
selecting, 337-340
viewing
applications, content, 541
calendar REST service, 332
categories, 357-360
controls, 99
data binding helper dialog box, 41
documents
collections, 45
individual, 45
Domino Designer, 16
Domino server consoles, 959
fields, 265
folders, Dojo, 474
hidden text, 648
iPhone User Agents, 678
memory dumps, 1017
pagers, 102
plug-ins, 631
topics, debugging, 732
translations, 877
User Agents, 678
views
categories, 313
categoryFilter property, 297-299
columns, 102
categories, 364
navigating, 40
selecting, 43
sorting, 323-326
viewing, 47
content, 309-312
Countries, 89
CPU profilers, 937
databaseName property, 294-296
data sources, 74-75, 296-309
Data Tables, 370-381. See also Data Tables
Data View control, 387-394
documents, retrieving, 315-317
Dynamic View Panel, 359
Events, 427
expandLevel property, 310-312
facets, 103
filters, disabling, 306
full text search properties, 299-304
ignoreRequestParams property, 305-306
iNotes calendar control, 330-336
keysExactMatch property, 306-309
keys property, 306-309
loaded property, 316
multiple, 312-313
navigating, 35
optimizing, 337
Outline, 402
Pager property, 395-398
permissions, 1041
postOpenView property, 316
previewing, 43
Problems, 425
Profile, Data Tables, 376-381
properties, 366-370, 412
public access, 1065
queryOpenView property, 316
real-time processing, 950
Repeat control, 381-387
formatting, 383
nested repeats, 384-386
rich text, 386-387
requestParameterPrefix property, 313
Restore View, 163
scope property, 316
Source, 63
startKeys property, 310
summary data, 40
switching, 685
when view is not a view, 314
viewScope object, 907
viewScope variable, 182
viewStyleClass property, 793
$view> tag, 63
viewTopic Custom Control, 793
View Trace menu, 646
view variable, 166
visibility, views, 1042. See also viewsVisible tabs, 419-421
VoiceML, 165

W
W3C (World Wide Web Consortium), 59
wall time profilers, 942-946
Say hello one last time!
request, 990
snapshots, 969, 977
WAR (web-application archive) files, 541
warnings, dialog boxes, 621
WARNING statement, 757
WAS (WebSphere Application Server), 5, 159, 608
web, differences between clients, 621-624
web-application archive. See WAR files
web browsers, previewing in, 22-24, 25
Web Inspector, 711
debugging, 713-716
xsp.persistence.viewstate XSP property, 1032
XSP.publishEvent() function, 624
XSP style classes, 1082-1086
XSP tags, programming references, 1075
XSPUrl classes, 236
XSPUserAgent class, 236
XUL (XML User Interface Language), 608
XULRunner, 608
   CSJS, accessing, 1059
   security, 1058
   version dialog box, 609

Z
zip files, 1026. See also compression