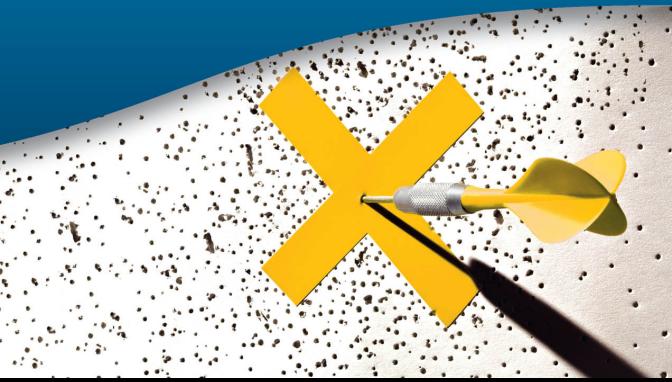
IBM PRESS

# Mastering XPages

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

Second Edition

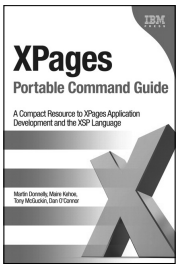
Martin Donnelly ▪ Mark Wallace ▪ Tony McGuckin



FREE SAMPLE CHAPTER

SHARE WITH OTHERS

# Related Books of Interest




## XPages Portable Command Guide

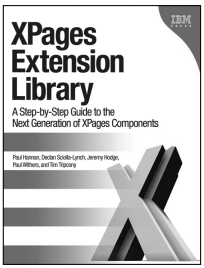
### A Compact Resource to XPages Application Development and the XSP Language

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 Sciolla-Lynch, Jeremy Hodge, Paul Withers, Tim Tripcony

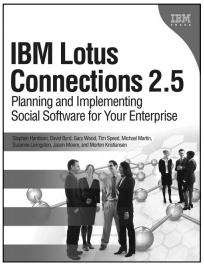
0-13-290181-1

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

# Related Books of Interest



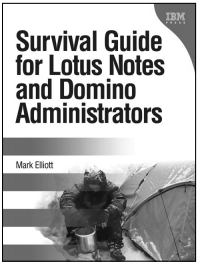

## IBM Lotus Connections 2.5

### 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

ISBN: 0-13-700053-7

In *IBM Lotus Connections 2.5*, a team of IBM Lotus Connections 2.5 experts thoroughly introduces the newest product and covers every facet of planning, deploying, and using it successfully. 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 support 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.

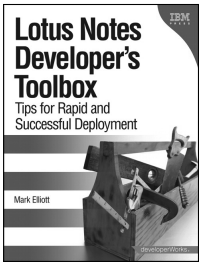
Podcasts

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

IBM Press™

Visit ibmpressbooks.com
for all product information

# Related Books of Interest




## Lotus Notes Developer's Toolbox

### Tips for Rapid and Successful Deployment

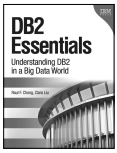
By Mark Elliott

ISBN-10: 0-13-221448-2

*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.

Podcasts

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




### DB2 Essentials

Understanding DB2 in a Big Data World, 3rd Edition

Chong, Liu

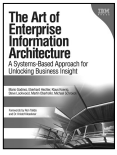
ISBN: 0-13-346190-4


### 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

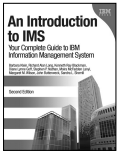



### The Art of Enterprise Information Architecture

A Systems-Based Approach for Unlocking Business Insight

Godinez, Hechler, Koening, Lockwood, Oberhofer, Schroeck

ISBN: 0-13-703571-3

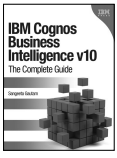



### An Introduction to IMS

Your Complete Guide to IBM Information Management System, 2nd Edition

Klein, Long, Blackman, Goff, Nathan, Lanyi, Wilson, Butterweck, Sherrill

ISBN: 0-13-288687-1




### IBM Cognos Business Intelligence v10

The Complete Guide

Gautam

ISBN: 0-13-272472-3

Sign up for the monthly IBM Press newsletter at
ibmpressbooks.com/newsletters

*This page intentionally left blank*

IBM PRESS

# Mastering XPages

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

2nd Edition

Martin Donnelly
Mark Wallace
Tony McGuckin

IBM Press
Pearson plc
Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris •
Madrid

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

© Copyright 2014 by International Business Machines Corporation. All rights reserved.

Note to U.S. Government Users: Documentation related to restricted right. Use, duplication, or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corporation.

IBM Press Program Managers: Steven M. Stansel, Ellice Uffer
Cover design: IBM Corporation
Associate Publisher: Dave Dusthimer
Marketing Manager: Stephane Nakib
Executive Editor: Mary Beth Ray
Publicist: Heather Fox
Senior Development Editor: Christopher Cleveland
Editorial Assistant: Vanessa Evans
Managing Editor: Kristy Hart
Designer: Alan Clements
Senior Project Editor: Lori Lyons
Technical Reviewers: Dan O'Connor, Paul Stephen Withers
Copy Editor: Apostrophe Editing Services
Indexer: Larry Sweazy, Wordwise Publishing Services
Compositor: Nonie Ratcliff
Proofreaders: Kathy Ruiz, Sarah Kearns
Manufacturing Buyer: Dan Uhrig

Published by Pearson plc

Publishing as IBM Press

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact international@pearsoned.com.

The following terms are trademarks of International Business Machines Corporation in many jurisdictions worldwide: IBM, IBM Press, Notes, Domino, IBM Social Business, IBM SmartCloud, Rational, Lotus, WebSphere, iNotes, LotusScript, developerWorks, and Sametime. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Oracle, Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates. Microsoft, Windows, ActiveX, and Internet Explorer are trademarks of Microsoft Corporation in the United States, other countries, or both. Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Other company, product, or service names may be trademarks or service marks of others.

Library of Congress Control Number: 2014930531

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-13-337337-0
ISBN-10: 0-13-337337-1

Text printed in the United States on recycled paper at Courier Westford, Inc. in Westford, Massachusetts.

First Printing: April 1014

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 míle 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

# Contents

# Foreword

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 captured as screen shots and written as numbered figures, using superimposed callouts where appropriate.

## 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 conceptual 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 understand 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 successfully 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 development 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.

# Acknowledgments

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!

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/`. 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`.

> **TIP**
>
> There is also an introductory tutorial available at 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 version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core"
       style="padding-top:20.0px;padding-right:10.0px;padding-left:20.0px">
       <h3>Mobile XPages Applications</h3>
       <xp:br></xp:br>
       User Agent: 
       <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) AppleWeb-
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.

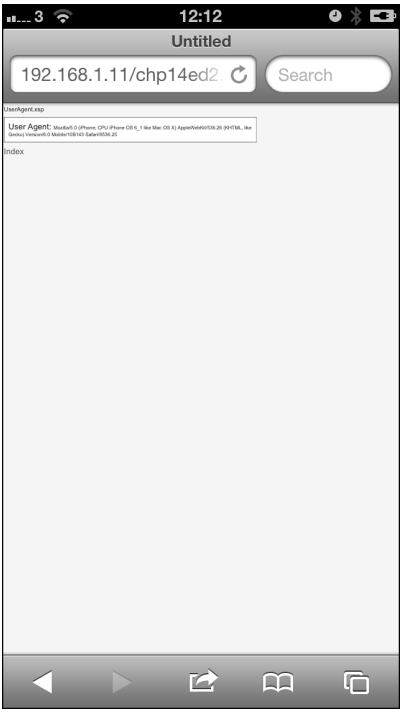



**Figure 14.1** iPhone User Agent

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 WebKit 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.


**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.

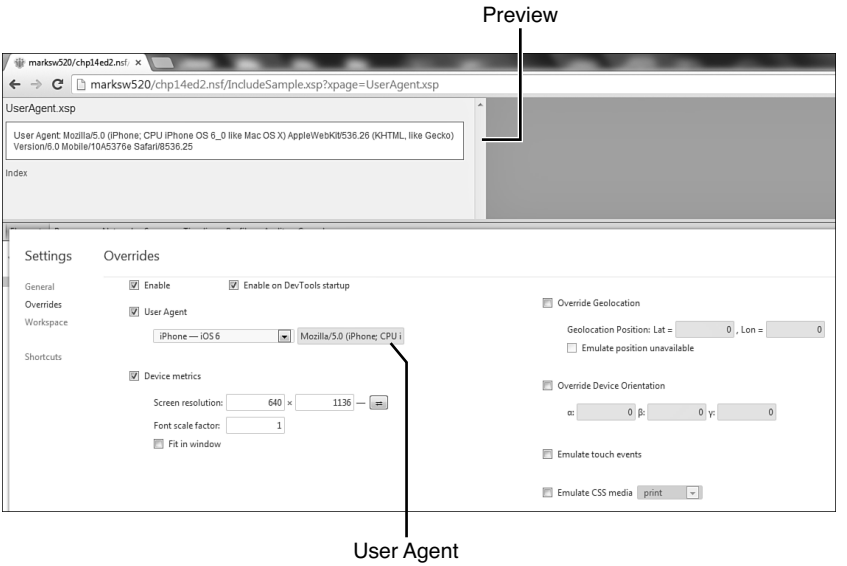



**Figure 14.3** Chrome Developer tools Overrides tab

## 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 version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core"
       style="padding-top:20.0px;padding-right:10.0px;padding-
➥left:20.0px">
       <h3>Mobile XPages Applications</h3>
       <xp:br></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></xp:br>
       Device: 
       <xp:text escape="true" id="computedField2"
              style="color:rgb(128,0,0)">
              <xp:this.value>
<![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 {
```

```
        return "Unknown";
}}]]>
                </xp:this.value>
        </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 version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core">
<style>.desc{position:absolute;left:20em;color:blue;}</style>

<h3>DeviceBean Properties</h3>
deviceBean.isMobile=<xp:text value="#{javascript:deviceBean.
isMobile()}"/>

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

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

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

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

<span class="desc">Version of iPad.</span><br/>
deviceBean.isAndroid=<xp:text value="#{javascript:deviceBean.
isAndroid()}"/>

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

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

<span class="desc">Identifies a device as a Windows Mobile device.
</span><br/>

</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

**DeviceBean Properties**

| Property | Value | Description |
|---|---|---|
| deviceBean.isMobile | false | Identifies a device as a mobile device. |
| deviceBean.isTablet | true | Identifies a device as a tablet. |
| deviceBean.isIphone | false | Identifies a device as an iPhone. |
| deviceBean.isIpad | true | Identifies a device as an iPad. |
| deviceBean.getVersion('ipad') | 4 | Version of iPad. |
| deviceBean.isAndroid | false | Identifies a device as an Android device. |
| deviceBean.isBlackberry | false | Identifies a device as a Blackberry device. |
| deviceBean.isWindows | false | Identifies a device as a Windows Mobile device. |

**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 *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 *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 (`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.

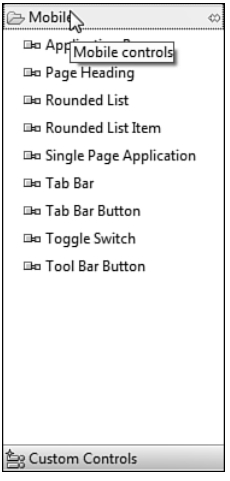



**Figure 14.5** Mobile controls

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.

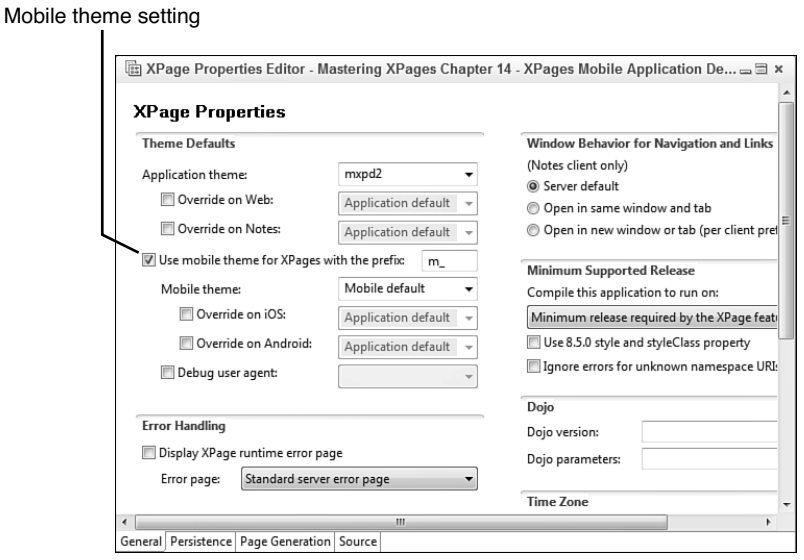



**Figure 14.6** Mobile XPage properties

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

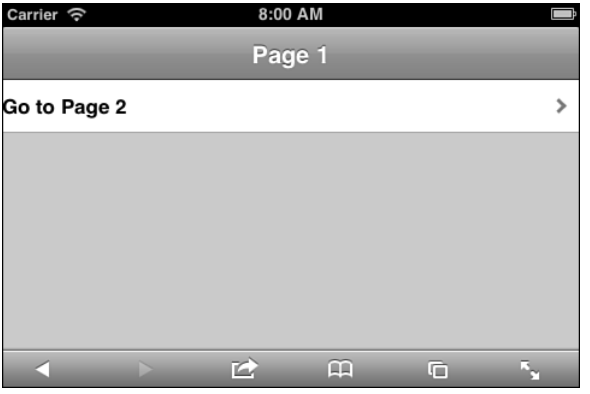



**Figure 14.7** Single Page Application Control on iPhone

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

```
        moveTo="appPage1" transition="flip">
      </xe:djxmLineItem>
    </xe:appPage>
  </xe:singlePageApp>
</xp:view>
```

## Navigator

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 version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core"
       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:djxmHeading>
                     <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:appPage>
              <xe:appPage id="visitorPage" pageName="visitorPage">
                     <xe:djxmHeading back="Home" moveTo="homePage">
                            <xe:this.label>Visitor Info</xe:this.label>
```

```
                </xe:djxmHeading>
            </xe:appPage>
            <xe:appPage id="conservePage" pageName="conservePage">
                <xe:djxmHeading back="Home" moveTo="homePage">
                    <xe:this.label>Conservation</xe:this.label>
                </xe:djxmHeading>
            </xe:appPage>
            <xe:appPage id="educatePage" pageName="educatePage">
                <xe:djxmHeading back="Home" moveTo="homePage">
                    <xe:this.label>Education</xe:this.label>
                </xe:djxmHeading>
            </xe:appPage>
            <xe:appPage id="involvePage" pageName="involvePage">
                <xe:djxmHeading back="Home" moveTo="homePage">
                    <xe:this.label>Get Involved</xe:this.label>
                </xe:djxmHeading>
            </xe:appPage>
            <xe:appPage id="shopPage" pageName="shopPage">
                <xe:djxmHeading back="Home" moveTo="homePage">
                    <xe:this.label>Shop</xe:this.label>
                </xe:djxmHeading>
            </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

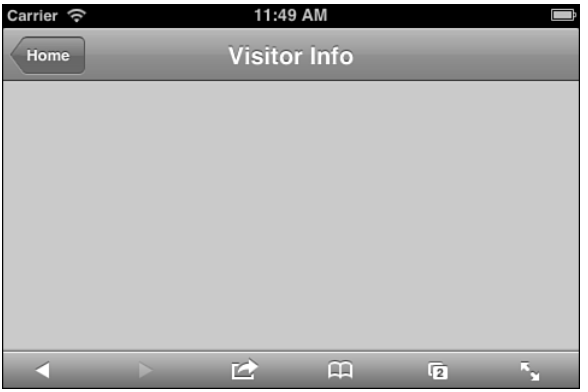



**Figure 14.9** Page Heading with Back button

## 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 version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core"
  xmlns:xe="http://www.ibm.com/xsp/coreex">
  <xe:outline id="outline1">
    <xe:this.treeNodes>
      <xe:basicContainerNode label="Services">
        <xe:this.children>
          <xe:basicLeafNode label="Business" href="#businessPage">
          </xe:basicLeafNode>
          <xe:basicLeafNode label="Training" href="#trainingPage">
          </xe:basicLeafNode>
        </xe:this.children>
      </xe:basicContainerNode>
      <xe:basicContainerNode label="Products">
        <xe:this.children>
          <xe:basicLeafNode label="Software" href="#softwarePage">
          </xe:basicLeafNode>
```

```
          <xe:basicLeafNode label="Systems" href="#systemsPage">
          </xe:basicLeafNode>
        </xe:this.children>
      </xe:basicContainerNode>
      <xe:basicContainerNode label="Support">
        <xe:this.children>
          <xe:basicLeafNode label="Downloads" href="#downloadsPage">
          </xe:basicLeafNode>
        </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 version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core"
  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></xc:Outline>
    </xe:appPage>
    <xe:appPage id="businessPage" pageName="businessPage">
      <xe:djxmHeading back="Services" moveTo="outlinePage">
        <xe:this.label>Business</xe:this.label>
      </xe:djxmHeading>
    </xe:appPage>
    <xe:appPage id="trainingPage" pageName="trainingPage">
      <xe:djxmHeading back="Services" moveTo="outlinePage">
        <xe:this.label>Training</xe:this.label>
      </xe:djxmHeading>
    </xe:appPage>
    <xe:appPage id="softwarePage" pageName="softwarePage">
      <xe:djxmHeading back="Products" moveTo="outlinePage">
        <xe:this.label>Software</xe:this.label>
      </xe:djxmHeading>
    </xe:appPage>
    <xe:appPage id="systemsPage" pageName="systemsPage">
      <xe:djxmHeading back="Products" moveTo="outlinePage">
        <xe:this.label>Systems</xe:this.label>
      </xe:djxmHeading>
```

```
    </xe:appPage>
    <xe:appPage id="downloadsPage" pageName="#downloadsPage">
      <xe:djxmHeading back="Support" moveTo="outlinePage">
        <xe:this.label>Downloads</xe:this.label>
      </xe:djxmHeading>
    </xe:appPage>
  </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 version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core"
       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:appPage>
        </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.

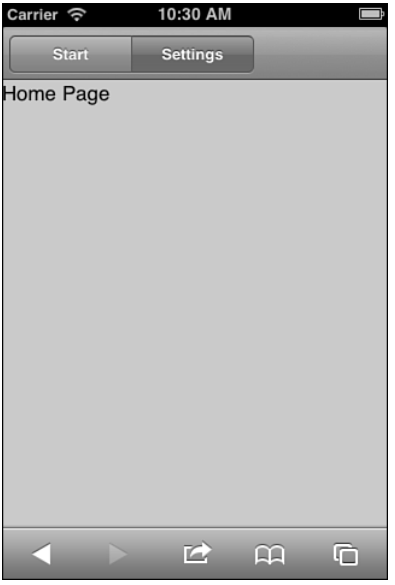



**Figure 14.10** Heading with segmented buttons

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 version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core"
  xmlns:xe="http://www.ibm.com/xsp/coreex">
  <xe:djxmHeading>
    <xe:tabBar id="tabBar1"
      style="background-color:rgb(255,255,255);width:100%;"
```

```
      barType="segmentedControl">
      <xe:tabBarButton id="homeButton" label="Home"
        rendered="#{javascript:compositeData.pageName!='home'}"
        onClick="location.hash='#homePage';">
      </xe:tabBarButton>
      <xe:tabBarButton id="startButton" label="Start"
        onClick="location.hash='#startPage';">
        <xe:this.rendered>
        <![CDATA[#{javascript:compositeData.pageName!='start' &&
compositeData.pageName!='advanced'}]]>
        </xe:this.rendered>
      </xe:tabBarButton>
      <xe:tabBarButton id="settingsButton" label="Settings"
        rendered="#{javascript:compositeData.pageName!='settings'}"
        onClick="location.hash='#settingsPage';">
      </xe:tabBarButton>
      <xe:tabBarButton id="advancedButton" label="Advanced"
        onClick="location.hash='#advancedPage';">
        <xe:this.rendered>
        <![CDATA[#{javascript:compositeData.pageName!='home' &&
          compositeData.pageName!='start' &&
          compositeData.pageName!='advanced'}]]>
        </xe:this.rendered>
      </xe:tabBarButton>
    </xe:tabBar>
  </xe:djxmHeading>
</xp:view>
```

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.

**Listing 14.11** onBeforeTransitionOut Sample XPage

```
<?xml version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core"
xmlns:xe="http://www.ibm.com/xsp/coreex">
  <xe:singlePageApp selectedPageName="firstPage">
    <xe:appPage id="transitionPage" pageName="firstPage">
      <xp:eventHandler event="onBeforeTransitionOut"
        submit="false">
        <xe:this.script>
<![CDATA[
alert("Leaving First page!");
]]>
        </xe:this.script>
      </xp:eventHandler>
      <xe:djxmHeading id="djxmHeading3">
        <xe:this.label>First</xe:this.label>
      </xe:djxmHeading>
      <xe:djxmLineItem label="Second Page" moveTo="secondPage"/>
    </xe:appPage>
    <xe:appPage id="secondPage" pageName="secondPage">
      <xe:djxmHeading id="djxmHeading1">
        <xe:this.label>Second</xe:this.label>
      </xe:djxmHeading>
      <xe:djxmLineItem label="First Page" moveTo="firstPage"/>
    </xe:appPage>
  </xe:singlePageApp>
</xp:view>
```

## 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

```
@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 version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core"
  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 version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core"
  xmlns:xe="http://www.ibm.com/xsp/coreex">
  <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="/lanscape.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 version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core"
xmlns:xe="http://www.ibm.com/xsp/coreex">
  <xe: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>
    <xp:eventHandler event="onOrientationChange" submit="false">
      <xe:this.script>
<![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}";
}
]]>
```

```
      </xe:this.script>
    </xp:eventHandler>
  </xe:singlePageApp>
</xp:view>
```

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

```
label1.innerHTML = "Swiped Me: ";
]]>
        </xe:this.script>
      </xp:eventHandler>
      </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 `xp:metaData` tag with a width of 500. (A further enhancement would be to compute this value based on the specific device.)

**Listing 14.17** Viewport Meta Tag

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

Take, for example, the launch page (`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:

- **`width:`** Viewport width, that is, the width of the page a user sees.
- **`height:`** Viewport height, that is, the height of the page a user sees.

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


**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 you 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 singe 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

```
<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:eventHandler event="onclick"
          submit="false">
          <xp:this.script>
            <xe:addRows for="dataView1"
              rowCount="10">
            </xe:addRows>
          </xp:this.script>
        </xp:eventHandler>
      </xp:link>
    </xp:this.facets>
    <xe:this.summaryColumn>
      <xe:viewSummaryColumn columnName="CarMake">
      </xe:viewSummaryColumn>
    </xe:this.summaryColumn>
  </xe:dataView>
</xe:appPage>
```

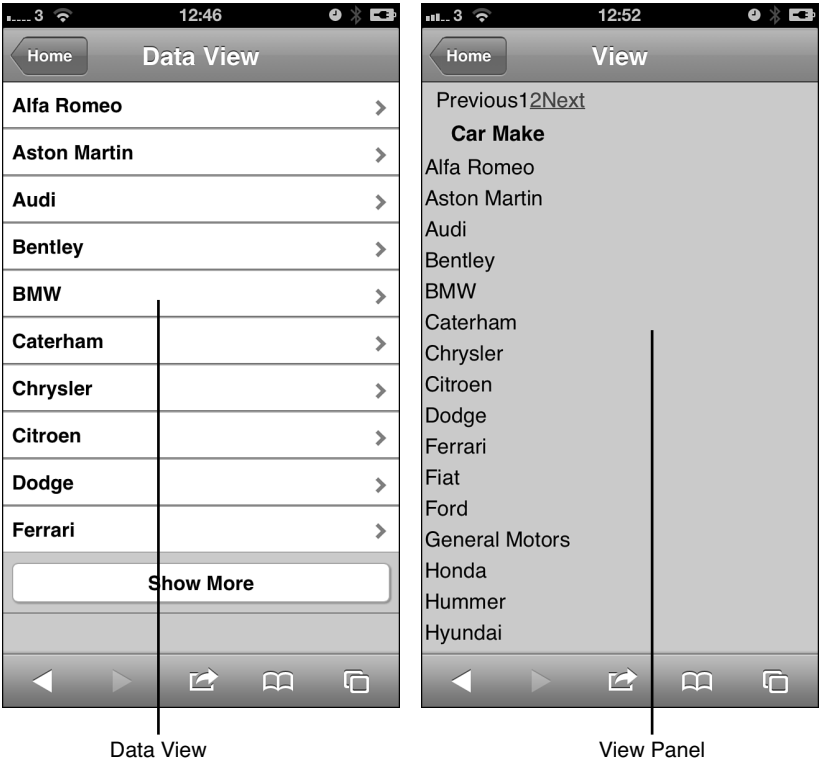



**Figure 14.12** Data View versus View Panel

## 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.

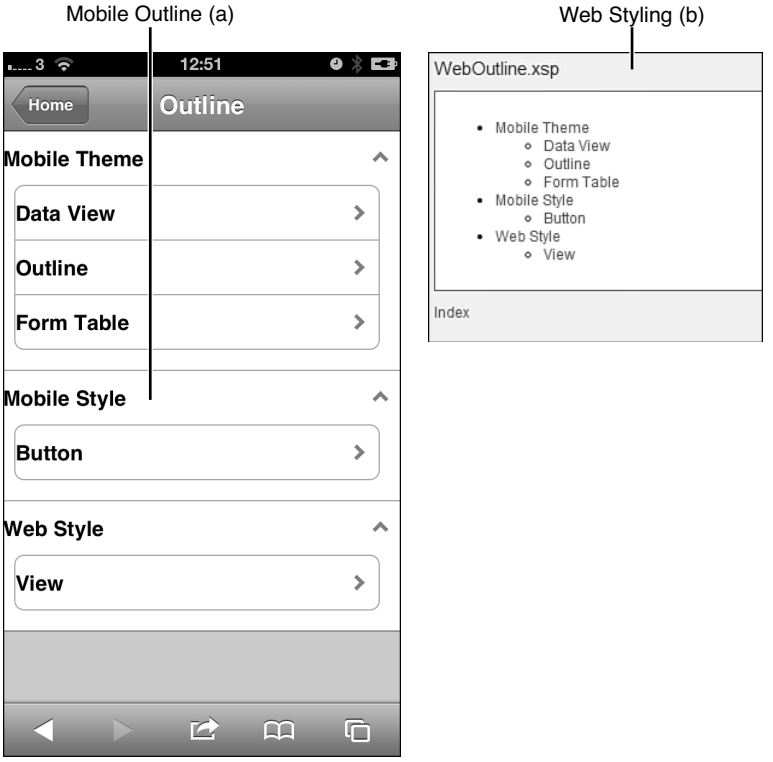



**Figure 14.13** Outline with mobile and web styling

## 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.

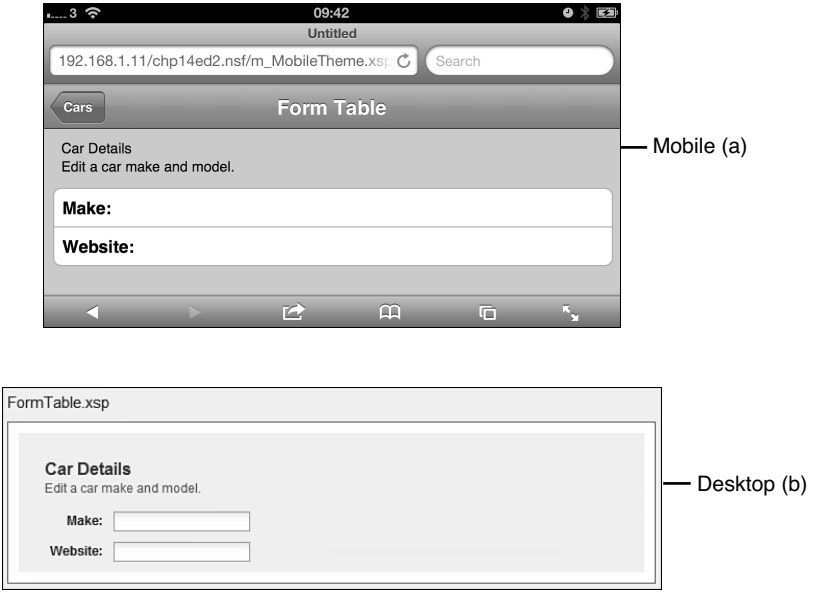



**Figure 14.14** Form Table with mobile and web styling

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

```
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. <![CATA[#{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>
```


**Figure 14.15** Buttons with mobile styling

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 them 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.


**Figure 14.16** Buttons with mobile styling using One UI

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

```
<!-- 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

```
<!-- 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.

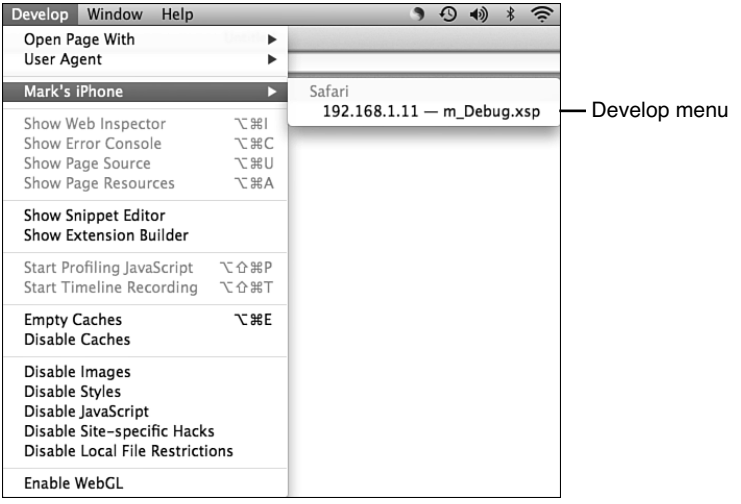



**Figure 14.17** Develop menu item

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.


**Figure 14.18** Web Inspector

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.

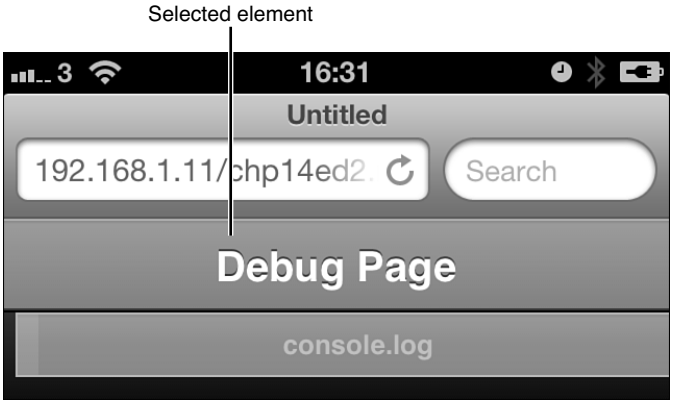



**Figure 14.19** Selected element

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.

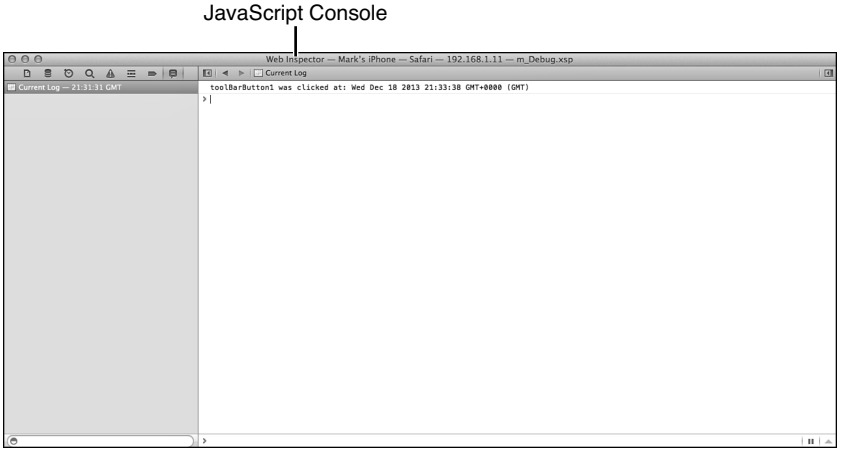



**Figure 14.20** JavaScript console

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

```
<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.

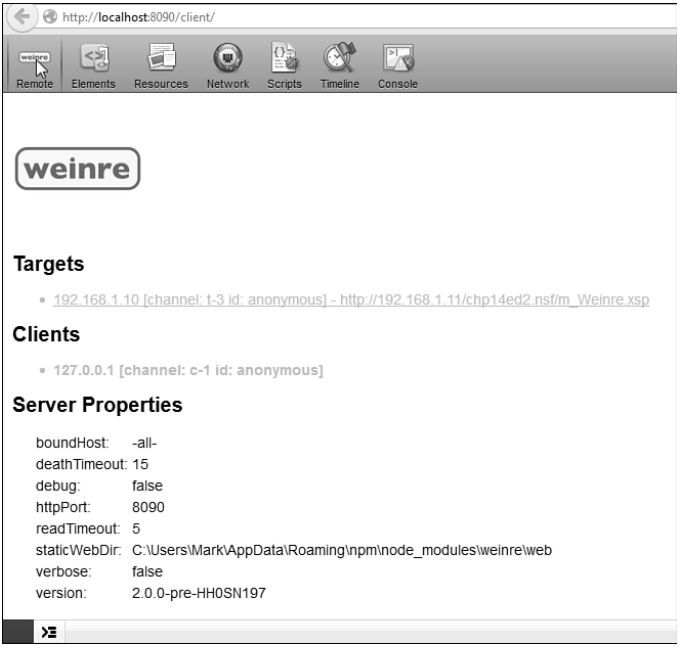



**Figure 14.21** weinre client

Figure 14.22 shows the DOM for the **m_Weinre.xsp** page.


**Figure 14.22** Web Inspector

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.


**Figure 14.23** Selected element

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.

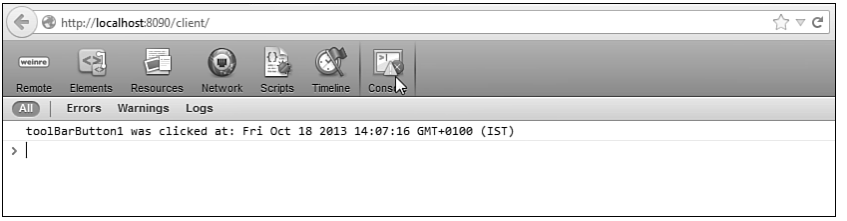



**Figure 14.24** JavaScript console

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/.

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

```
                summary="Activity" infiniteScroll="enable"
                value="#{javascript:1000}">
                <xe:this.summaryColumn>
                        <xe:viewSummaryColumn
                                value="#{javascript:'First Data View -
➥'+viewEntry}">
                        </xe:viewSummaryColumn>
                </xe:this.summaryColumn>
        </xe:dataView>
        <xp:label value="Second Data View" id="label2"></xp:label>
        <xe:dataView id="dataView2" var="viewEntry" collapsibleRows="true"
                collapsibleDetail="true" columnTitles="true"
                rows="20" showCheckbox="false"
                style="height:200px; border:2px solid #008abf; margin: 5px;"
                infiniteScroll="enable" value="#{javascript:1000}">
                <xe:this.summaryColumn>
                        <xe:viewSummaryColumn
                                value="#{javascript:'Second Data View -
➥'+viewEntry}">
                        </xe:viewSummaryColumn>
                </xe:this.summaryColumn>
        </xe:dataView>
        <xe:tabBar id="tabBar1">Mastering XPages</xe:tabBar>
</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 on to an XPage. Figure 14.25 shows the Single Page Application Wizard.

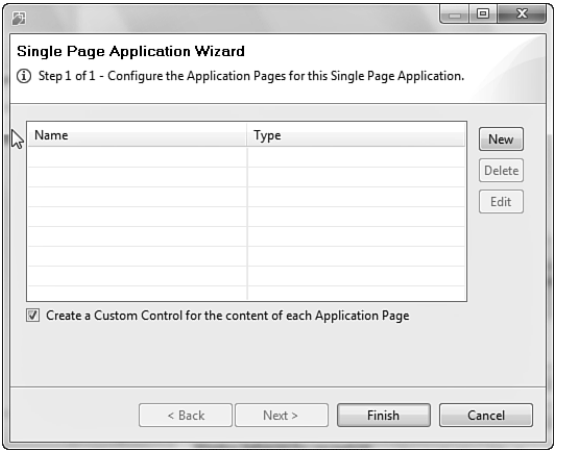



**Figure 14.25** Single Page Application Wizard

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

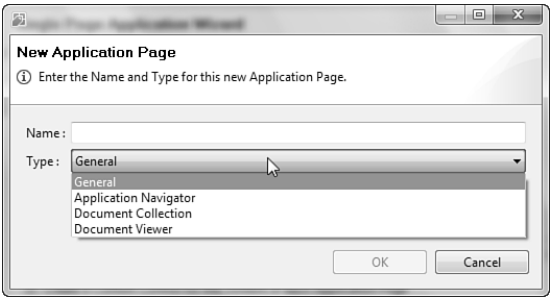



**Figure 14.26** New Application Page dialog

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.

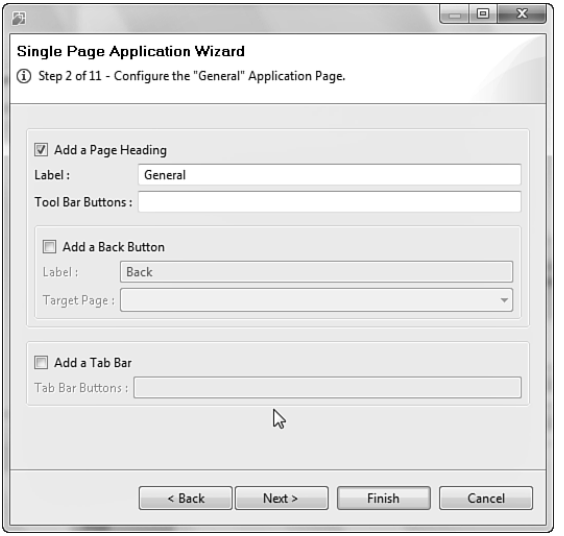



**Figure 14.27** General Application Page options

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.

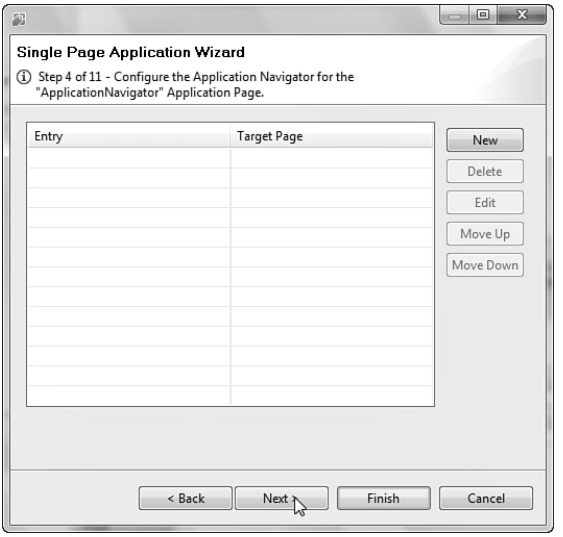



**Figure 14.28** Application Navigator Page options

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.

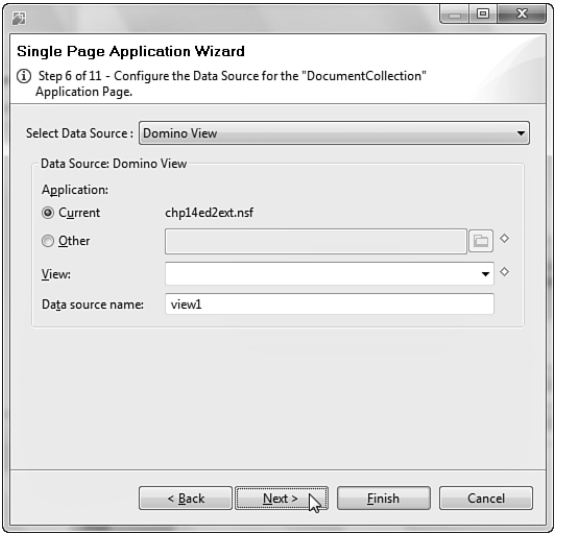



**Figure 14.29** Document collection data source configuration

Figure 14.30 shows the options to configure the data view used to display the document collection (including an option to enable infinite scrolling).

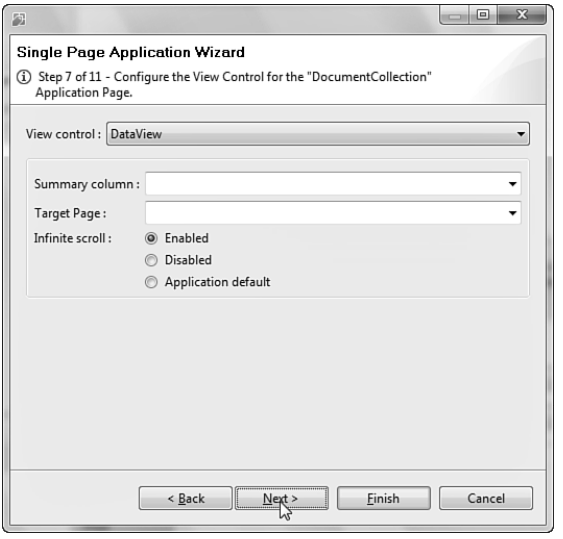



**Figure 14.30** Document collection Data View configuration

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.

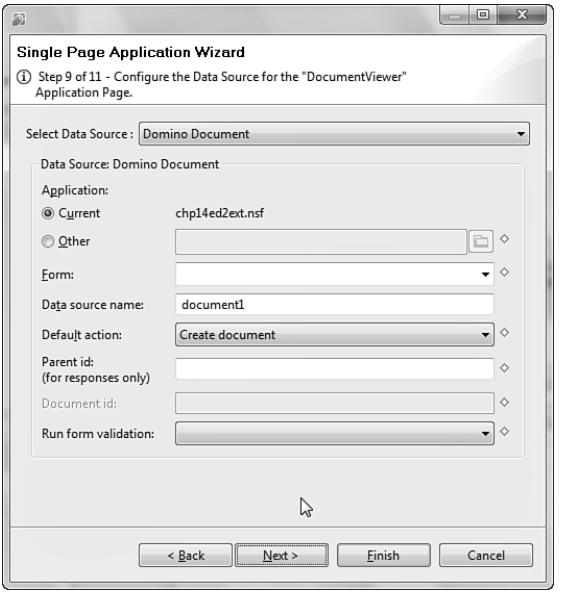



**Figure 14.31** Document Viewer data source configuration

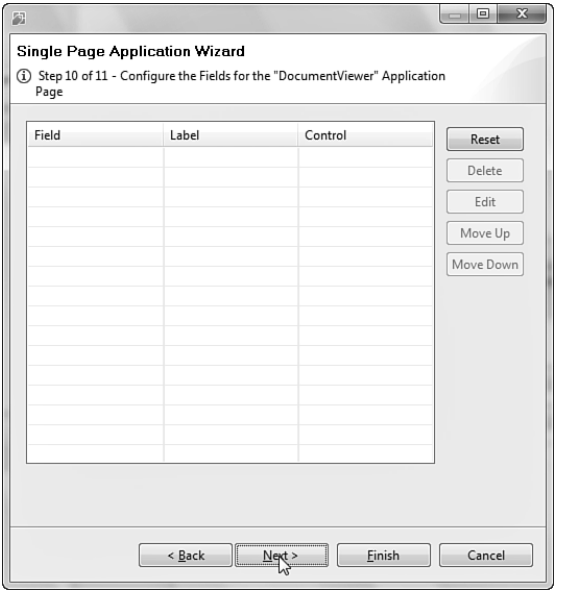



**Figure 14.32** Document Viewer Fields configuration

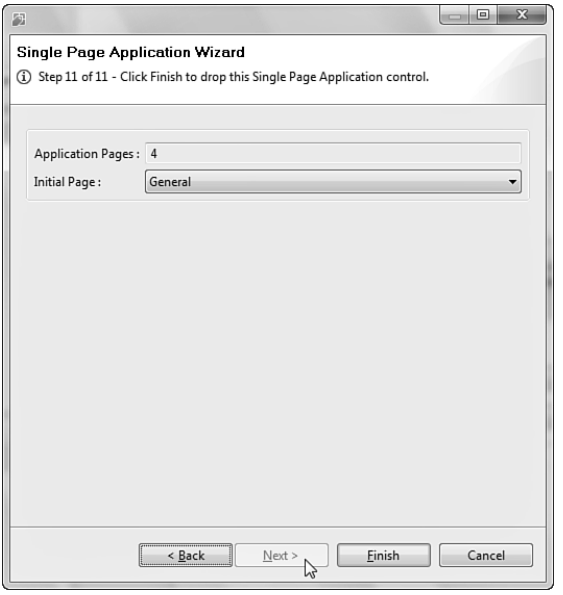



**Figure 14.33** Finish Single Application Page Wizard

## 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



## A

## B

## C

## D

## E

## F

## G



## H

## J

## K



## L

## M

## N

## O

## P

## Q



## R

## S

## T

## U

## V

## W

## X

## Z