Professional Portal Development with Open Source Tools: Java™ Portlet API, Lucene, James, Slide

W. Clay Richardson
Donald Avondolio
Joe Vitale
Peter Len
Kevin T. Smith

Wiley Technology Publishing

Wiley Publishing, Inc.

Professional Portal Development with Open Source Tools: Java™ Portlet API, Lucene, James, Slide

Professional Portal Development with Open Source Tools: Java™ Portlet API, Lucene, James, Slide

W. Clay Richardson
Donald Avondolio
Joe Vitale
Peter Len
Kevin T. Smith

Wiley Technology Publishing

Wiley Publishing, Inc.

Professional Portal Development with Open Source Tools: Java™ Portlet API, Lucene, James, Slide

Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

Library of Congress Cataloging-in-Publication Data:

Professional portal development with open source tools: Java™ Portlet
API, Lucene, James, Slide (Wrox Press) / by W. Clay Richardson . . . [et al.] . .
p. cm.
Includes bibliographical references and index.
ISBN 0-471-46951-3 (PAPER/WEB SITE)
1. Web site development. 2. Open source software. I. Richardson, W. Clay, 1976-
TK5105.888P68 2004
006.7'6--dc22

2003023864

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

1MA/QZ/QS/QU/IN

About the Authors

W. Clay Richardson

W. Clay Richardson is a software consultant specializing in distributed solutions, particularly portal solutions. He has fielded multiple open-source Web and portal solutions, serving in roles ranging from senior architect to development lead. He is a co-author of *More Java Pitfalls,* also published by Wiley & Sons. As an adjunct professor of computer science for Virginia Tech, he teaches graduate-level course-work in object-oriented development with Java. He holds degrees from Virginia Tech and the Virginia Military Institute.

Donald Avondolio

Donald Avondolio is a software consultant with over seventeen years of experience developing and deploying enterprise applications. He began his career in the aerospace industry developing programs for flight simulators, and later became an independent contractor, crafting healthcare middleware and low-level device drivers for an assortment of mechanical devices. Most recently, he has built e-commerce applications for numerous high-profile companies, including The Home Depot, Federal Computer Week, the U.S. Postal Service, and General Electric. He is currently a technical architect and developer on several portal deployments. Don also serves as an adjunct professor at Virginia Tech, where he teaches progressive object-oriented design and development methodologies, with an emphasis on patterns.

Joe Vitale

Joe Vitale has been working with the latest cutting-edge Java technology intensely. His most recent focus has been on Java portals and object-relational mapping tools. One of these projects was writing a content management system that contained role-based authentication of users and the capability for users to upload, delete, and manage files, and secure resources. The whole system was designed to plug right into a portal's interface and enable the portal to directly communicate with it to obtain its resources. Object-relational mapping technologies have also been a focus, using Apache's Object Relational Bridge (OJB).

Peter Len

Peter Len has over seven years' experience performing Web-based and Java application development in a client-server environment. He has designed, coded, and implemented data and Web site components for each aspect of a three-tier architecture. Mr. Len has been developing with Java for over five years and has recently been involved with portal and Web-service development. He holds a master's degree in both international affairs and computer information systems.

Kevin T. Smith

Kevin T. Smith is a technical director and principal software architect at McDonald Bradley, Inc., where he develops security solutions for Web service–based systems. He has focused his career on building enterprise solutions based on open-source tools. He holds undergraduate and graduate degrees in computer science, software systems engineering, and information security. He has taught undergraduate courses in computer science, given technical presentations on Web services and Java programming at numerous technology conferences, and authored several technical books, including *Essential XUL Programming* (Wiley 2001), *More Java Pitfalls* (Wiley 2003), and *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management* (Wiley 2003).

Dedication

This book is dedicated to all those who make the daily sacrifices, and especially to those who have made the ultimate sacrifice, to ensure our freedom and security.

Credits

Authors
W. Clay Richardson
Donald Avondolio
Joe Vitale
Peter Len
Kevin T. Smith

Vice President and Executive Group Publisher
Richard Swadley

Vice President and Executive Publisher
Robert Ipsen

Vice President and Publisher
Joseph B. Wikert

Executive Editorial Director
Mary Bednarek

Editorial Manager
Kathryn A. Malm

Executive Editor
Robert Elliott

Senior Production Editor
Fred Bernardi

Development Editor
Eileen Bien Calabro

Production Editor
William A. Barton

Copy Editor
Luann Rouff

Media Development Specialist
Angela Denny

Permissions Editor
Carmen Krikorian

Project Coordinator
April Farling

Graphic and Layout Technicians
Carrie Foster
Jennifer Heleine
Kristin McMullan
Lynsey Osborn

Quality Control Technicians
John Greenough
Andy Hollandbeck
Brian H. Walls

Text Design & Composition
Wiley Composition Services

Proofreading
Henry Lazarek

Indexing
Tom Dinse

Acknowledgments

I would first like to acknowledge Major Todd DeLong, USA, who had the courage and insight to support an open-source portal solution in the face of overwhelming conventional wisdom, and provided inspiration to this book. Those who know him understand that this is a relatively small measure of his courage. Of course, I could not have had any chance of actually getting this book done without the support of my wonderful wife, Alicia, and daughter, Jennifer. I love both of you more than words can describe. Stephanie, we love you and will never forget you. To my fellow authors, Donnie, Joe, Peter, and Kevin, I appreciate the style, class, and integrity you showed during some very difficult times. I am in a much better place now, and sincerely hope that each of you finds a similar situation for yourself. You are all wonderful talents and it was a great pleasure collaborating with you. I would like to thank Bob Elliott and Eileen Bien Calabro for all their hard work and perseverance working with us on this project. To my leadership, Mark Cramer, Joe Duffy, Jim Moorhead, and Tom Eger, it's wonderful to work for guys who don't talk about "core values," but rather just lead by example. I would like to thank my parents, Bill and Kay, my in-laws, Stephen and Elaine Mellman, my sister, Kari, my brother, Morgan, and my stepfather, Dave, for always being there. I would like to acknowledge my grandmothers, Vivian and Sophie, for being what grandmothers should be.

To my technical "posse": Mark "Mojo" Mitchell, Marshall "PAB" Sayen, Mauro "Tre" Marcellino, Scot Schrager, Tom Bachmann, Jon Simasek, Rob Brown, Kim Bell, Kevin McPhilamy, Jon Grasmeder — I look forward to facing more technical challenges with you. To the people I constantly badger for "stuff": Lisa Peters, Chris Reid, Bryan Foster, Steve Tagg, Rick Yard — this has to count for something! Ed, you know how it has to be. I would like to acknowledge those individuals with whom I didn't get to work enough: Arnie Voketaitis, Seth Goldrich, Cliff Toma, Joe Sayen, Kevin Moran, Adam Dean, Ken Pratt, Alex Blakemore, Dave Holberton, Vic Fraenckel, Mike Shea, Jullie Bishop, and many more to whom I will owe a beer for forgetting to mention them. I would like to thank my colleagues at Virginia Tech for their assistance in my career development: Shawn Bohner, Athman Bouguettaya, John Viega, Stephen Edwards, and Tom Sheehan. To Mike Daconta, I appreciate that you gave me my break in writing and that you supported this book, and I can tell now that this lead author thing is harder than it looks! To my duty crew at the Gainesville District VFD: Bob Nowlen, Gary Sprifke, Patrick Vaughn, Marshall Sayen, Gerry Clemente, Javy Lopez, Thomas Mullins, and Brian LaFlamme — we have been through a lot together! Eric Jablow, failing to list you among my mentors in *More Java Pitfalls* was a horrible oversight; I hope I can rectify that here. Matt Tyrrell, despite being a "large cat," you are still like a brother to me. –WCR

I'd also like to thank all of the people I've worked with in the past: Wendong Wang, Arun Singh, Shawn Sherman, Henry Zhang, Bin Li, Feng Peng, Henry Chang., Sanath Shetty, Prabahkar Ramakrishnan, Swati Gupta, Mark Mitchell, Yuanlin Shi, Chiming Huang, Andy Zhang, Chi Luoung, and John Zhang, all of whom I loved working and goofing around with. Additionally, I'd like to thank the members of my current portal development program: Guillermo Suchicital, Susan Hansen, Linda Burchard, Honchal Do, Jae Kim, Johnny Krebs, Steve Brockman, Kevin Mills, Bob Russell, and Arnie Voketaitis. Thanks also to the professors at the Virginia Tech Computer Science/Information Technology Departments: Shawn Bohner, Tarun Sen, Stephen Edwards, John Viega, and all the other dedicated, top-notch staff and instructors.

Last, I wish to thank all of the co-authors, who are fun guys to work with and be around: Kevin, Joe, Peter, and Clay. To all of my family: Mom, Dad, Michael, John, Patricia, Jim, Sue, Reenie, Stephen, Emily, Jack, and Gillian, you guys are great. Thanks also to my friends back in New York: the Wieczorek, Devaney, Howard, Pujols, O'Donohoe, and Keane families; and to those in the open source community who have so generously contributed their hard work and time by crafting all of the fantastic tools described in this book, and have created a software culture of trust and collaboration that have allowed so many others to be productive in their programming endeavors. To my wife, Van, who I love more than anything for her easygoing manner and ability to use power tools and golf drivers better than most men. –DJA

To my wife, Jennifer, and my son, Andrew, thank you for all your love and support throughout this process; without you, I would never have found the energy to complete this. I'd also like to thank the following: the rest of my family, but especially my grandfather and grandmother, Carlo and Annette Vitale; my father, Joseph Vitale; my step-mother, Linda Vitale; and my father and mother-in-law, James and Marlaine Moore, for being helpful and offering encouraging advice. Many thanks also to John Carver, Jeff Scanlon, Brandon Vient, and Aron Lee for their great supporting roles as friends. Of course, I would like to thank all of my co-workers at McDonald Bradley Inc., including Kyle Rice, Danny Proko, Michael Daconta, Ken Bartee, Dave Shuping, Joe Cook, Ken Pratt, Adam Dean, Joon Lee, Maurita Soltis, Keith Bohnenberger, Bill Vitucci, Joe Broussard, Joseph Rajkumar, Theodore Wiatrak, Rebecca Smith, Barry Edmond, and many others who have had a significant, positive influence on me throughout the years. Finally, a special thanks goes to my co-authors for all of their hard work and encouragement. Thank you, all! –JV

This book marks my first participation in the development of a technical book. It has been a great challenge and one that has enlightened me in a number of ways. I would first like to thank my co-authors Clay, Donnie, Joe, and Kevin. They are extremely talented developers and thinkers and I truly appreciate their trust in asking me to help author this book. They are always striving to make a difference in their project work and I have benefited greatly from their help and guidance. I would also like to thank the many great people at my company, McDonald Bradley, who have supported my efforts in writing this book and who were willing to help wherever possible. I would especially like to thank Kevin Moran for his editorial efforts as well as the many years of technical mentorship he has so graciously afforded me. Lastly, I would like to thank my lovely fiancée, Ruby, for her editing, patience, support, and guidance during this effort. May she never stop. – PAL

I would like to first thank my co-authors on this project — Clay, Don, Peter, and Joe. I think this book contains some great lessons learned from some of our previous projects. I would also like to express my thanks to Bob Elliott and Eileen Bien Calabro from John Wiley & Sons. Special thanks to Natalie "Bonnie" Schermerhorn for giving me examples for the Llama Web service example. In addition, I can't forget my Southwest Virginia Readability editor, Helen G. Smith, or my Central Virginia Readability editor, Lois G. Schermerhorn. I would also like to recognize the great architecture team on the Virtual Knowledge Base project (as of August 2003): Keith Bohnenberger, Darren Govoni, Eric Monk, Joseph Rajkumar, Joel Sciandra, Maurita Soltis, and Arnie Voketaitis. Mike Daconta, thank you for your support and writing suggestions over the years. To my wife, Gwen, thank you for putting up with me writing on weekends for yet another book project! Last, but not least, I would like to thank God and the following verses, which have affected my life in a powerful way: Philippians 4:4–9 and Romans 8:18–39. Read them — you won't be disappointed. –KTS

Contents

Acknowledgments ix
Introduction xix

Part I: Open Source Portals 1

Chapter 1: The Java Portlet API (JSR 168) 3

Portlet Fundamentals 3
Portlets and Servlets 5
Portal Interactions 6
The Portlet Interface and the GenericPortlet 8
Portlet Life Cycle 8
Portlet Runtime States 8
Portlet Request Handling 9
"There Can Be Only One" 9
ActionRequest 10
RenderRequest 11
GenericPortlet 11
Other Elements of the Java Portlet API 14
PortletConfig 14
PortletURL 14
Portlet Modes 15
Window States 16
Portlet Context 16
Portal Context 16
Portlet Preferences 16
Sessions 18
Calling JSPs and Servlets 19
Portlet Application Structure 21
Security 22
CSS Style Definitions 22
User Information Attributes 24
Portlet Tag Library 28
Portlet Deployment 28
Portlet Application Deployment Descriptor 28
Portlet Declarations 29
Building a Portlet 32
Summary 39

Chapter 2: Searching with Lucene 41

Understanding Search Engine Concepts 41
The Anatomy of a Search Engine 43
The Anatomy of Lucene 48
Analyzer 51
Document 53
Directory 54
Understanding the Lucene Query Syntax 54
Terms 54
Fields 54
Term Modifiers 54
Boolean Operators, Grouping, and Escaping 55
Optimizing Lucene's Performance 57
Summary 57

Chapter 3: Messaging with Apache James 59

Introducing James 59
Working with Mailets and Matchers 60
Bundled Matchers 61
Bundled Mailets 62
Understanding SpoolManager 63
Understanding Repositories 63
File Repositories 64
Database Repositories 64
DBFile Repositories 64
Working with RemoteManager 64
Implementing James 64
Downloading James 64
Installing James 65
Configuring James 65
DNS Server Configuration 66
POP3 Server Configuration 66
SMTP Server Configuration 66
NNTP Server Configuration 67
FetchPOP Configuration 67
RemoteManager Configuration 68
Repository Configuration 68
SpoolManager Configuration 68
Global Server Configuration 68
Creating User Accounts 69

Introducing JavaMail API **70**
The Session Class 71
Message Class 71
Address Class 72
Authenticator Class 72
Transport Class 73
Store and Folder 73
JavaMail in Practice **74**
Sending Messages 74
Receiving Messages 79
Summary **86**

Chapter 4: Object to Relational Mapping with Apache OJB **87**

Exploring Object-to-Relational Mapping Concepts **87**
Understanding OJB Technology Features **88**
Using OJB API Layers **89**
Developing with the PersistenceBroker API 90
The Message Class 93
JDBC Connection Mapping 99
Home.jsp 100
Add.jsp 104
View.jsp 106
Developing with the ODMG API 107
Opening a Database 107
Retrieving Objects 107
Storing Objects 108
Updating Objects 109
Deleting Objects 109
Developing with the JDO API 110
OJB Extras **110**
Verifying an OJB Installation 111
Supported Database Platforms 111
Supported JDBC Data Types 112
Deploying OJB Applications 113
Jar Files 113
Metadata Files 114
JDBC Drivers 114
CLASSPATH Settings 114
OJB Performance 114
Summary **114**

Chapter 5: Content Management with Jakarta's Slide 117

Slide Architecture 117
External Architecture 118
Internal Architecture 120
Transaction Management 122
Transactions 123
Transaction Attributes 123
Namespaces 124
Helpers 124
Stores 132
Domain 132
Slide API Layer 135
Setting Up and Configuring Slide 135
Installing and Running Slide 135
Tomcat Manager 136
The Slide Server 138
Client View 139
Admin View 139
Editor View 139
WebDAV and Slide 139
Windows XP WebDAV Example 140
Summary 143

Chapter 6: Portal Security 145

Core Security Concepts 145
Authentication 145
Authorization 146
Single Sign-On (SSO) 147
Confidentiality 149
Data Integrity 150
Non-repudiation 150
Key Security Standards 150
SSL and TLS 150
XML Encryption 151
XML Signature 151
SAML 151
OASIS Web Services Security (WSS) 153

Building Security Solutions for Your Portal **153**
Web Container Security — Apache Tomcat 154
Server Configuration 155
Application Configuration 165
Programmatic Security Access with JSPs and Servlets 167
Security of the Portlet Container 168
Programmatic Security 169
Portlet Descriptor-Configured Security 170
Beyond the Portal — Secure Back-End Communication 171
Summary **172**

Part II: How to Build a Portal **173**

Chapter 7: Planning for Portal Deployment **175**

System Requirements **177**
Interface Requirements 177
Operational Requirements 177
Data Requirements 178
Security Requirements 178
Quality Assurance Requirements 179
Software Configuration Management 179
Jakarta's ANT 180
Unit and Load Testing 180
Bug Tracking 181
Continuous Integration 181
Requirements Summary 181
Software Design Methodologies **182**
The Unified Process (UP) 182
Domain Model 184
Software Architecture Document (SAD) 186
Shall Statements and User Stories 189
Class-Responsibility-Collaborator Cards (CRCs) 190
Storyboarding 190
Design Models for Visualization That Are Not in UML 193
Design Decisions **195**
Model 1 Architecture 196
Model 2 Architecture 197
Model 2X Architecture 197
Search Utilities **200**

Content Management **200**
Design Pattern Considerations in Your Portal **201**
Using Java Standards 202
Model-View-Controller (MVC) Pattern 205
Template Method Pattern 205
Memento Pattern 205
Facade Pattern 205
Adapter Pattern 205
Factory Method Pattern 206
Singleton Pattern 206
Front Controller Pattern 206
Intercepting Filter Pattern 206
Client-Side Processing **206**
JavaScript **207**
Server-Side Processing **207**
Java Plug-ins 208
Web Services for Remote Portals (WSRP) **209**
Portal Navigation **210**
Portal Navigation Using Taxonomies 212
Portlet Integration Plan **218**
Summary **220**

Chapter 8: Effective Client-Side Development Using JavaScript **221**

Declaring JavaScript **222**
Validating Data **223**
Adding Functionality **227**
Field Auto-Population 228
Field Auto-Validation 229
Space Management **231**
Multiple-Value Picklists 231
Repeating Values 234
Dynamic Actions **240**
Layering and DHTML **244**
Forms and Layers 244
Movable Layers 250
Summary **253**

Chaper 9: Developing Applications and Workflow for Your Portal **255**

The Portlet Architecture **255**
The Portlet Container 256
Portlet Preferences 258

JSP Tag Library 258
Packaging a Portlet 258
The eXo Portal Platform 259
The eXo Portal 259
Hot Deployment 259
Customization Tool 259
Setup and Installation of eXo 260
Understanding the eXo Directory Structure 260
The Directory Portlet 262
Developing the Directory Portlet 265
The MySQL Database 266
The DirectoryPortlet Class 267
The DirectoryValidator Class 271
The DirectoryView JSP File 272
The DirectoryEdit JSP File 274
The web.xml File 275
The portlet.xml File 275
Deploying the Directory Portlet in eXo 276
Portlet Creation the Model-View-Controller (MVC) Way 277
The MVC Loan Calculator Portlet 279
Web Applications versus Portlet Applications 283
Summary 283

Chapter 10: Portlet Integration with Web Services 285

Basic Concepts 285
Integrating with Traditional Web Services 287
A Simple Example 292
First Approach: SOAP and WSDL Messaging 292
Second Approach: Working with Generated Objects 298
Web Services for Remote Portlets (WSRP) 302
Types of WSRP Services 304
Discovery, Registration, and Deregistration Services 304
Simple WSRP Services — Stateless "View Only" Modes 304
More Complex Services — Interactive WSRP Services 304
Portlet Management Services 305
WSRP Markup Guidelines for Portlet Developers 305
Disallowed XHTML and HTML Tags 305
Cascading Style Sheets (CSS) Style Definitions 305
User Information 310
Summary 310

Chapter 11: Performance Testing, Administering, and Monitoring Your Portal **311**

Continuous Integration **312**
CVS 312
Subversion 315
JUnit 316
AntHill 324
Load Testing **326**
JMeter 327
Portal Requirements/Bug Management and Traceability with Scarab **330**
Scarab 330
Scarab Tasks 331
Portal Administration with JMX **335**
Portal Collaboration with JSPWiki **344**
Summary **346**

Chapter 12: Unifying the Enterprise Application Space Through Web Start **347**

Rich Clients **348**
Java Web Start **349**
Getting Started **351**
Downloading and Installing Java Web Start 351
Configuring the Web Server 351
Creating the JNLP File 352
<jnlp> Attributes 353
<information> Subelement 353
<security> Subelement 353
<resources> Subelement 353
<application-desc> Subelement 354
Application Packaging 354
Client Invocation 354
Code Signing 356
Introductory Application **358**
Using JWS in Portal Implementations **363**
Use in a Web-based Portal 364
Use in a Java Portal 367
Java Swing 367
Java-based Portal Examples 368
Summary **372**

References **373**
Index **375**

Introduction

Portal development projects have become the centerpiece of IT acquisition and development strategy for many organizations. Enterprise integration and Web application developers predictably groan when they hear the word "portal"—nightmares of proprietary APIs, oversold features, and shoddy tool integrations. The authors of this book have been involved in over a dozen production portal efforts over the last several years. In that time, we have dealt with numerous products and frameworks, including some in-house frameworks based on servlets and JSPs. Through all of this, we began to wonder whether these commercial suites were really providing any value. We started to realize that we could put together a framework from open-source products.

We would like to point out that our portal framework is not meant to be an all-or-nothing solution. We present a number of tools that you may use to satisfy your enterprise portal needs, and we demonstrate how to use them, but because portal efforts are largely integration efforts, it would be folly to presume that anyone will drop all of their current systems and pick up our framework.

This book explains a set of tools at the foundation of an open-source portal framework, and demonstrates how to build your own portal using open-source tools. However, before describing the structure of the book, it makes sense to cover some fundamental concepts addressed therein.

What Is a Portal?

"A portal is your enterprise."

"A portal is a single synergistic access to all your enterprise information, and only the appropriate information."

"A portal is a unique IT strategy that allows me to answer the classic "build versus buy" question — yes!"

"A portal is one of those holes in the side of a ship, like they had on the Love Boat!"

All of these are commonly used as definitions of a portal. Obviously, the first two come from people who wish to sell you on your need for a portal. The third is clearly from a CIO who has funded a portal acquisition, and the last is a popular joke (though wrong because, as all *Love Boat* fanatics know, a **port-hole** is what is in the side of a ship).

The Java community decided to come up with a least common denominator agreement on what a portal is by standardizing on a Portlet API. This standard is known as JSR 168: "Portlet Specification." This book views JSR 168 as the bottom line on portals, but understands the youth of this standard and that the disparities among current portal implementations requires considering portals in a more pragmatic sense.

JSR 168 defines a portal as follows:

> *A portal is a Web-based application that commonly provides personalization, single sign-on, and content aggregation from different sources, and it hosts the presentation layer of information systems. Aggregation is the action of integrating content from different sources within a Web page. A portal may have sophisticated personalization features to provide customized content to users. Portal pages may have different sets of portlets creating content for different users. [JSR168]*

Portals are becoming the new foundation of the Web application platform. The proliferation of Web applications has required software to tie these disparate Web applications together into aggregated applications.

Portals are Web-enabled applications that integrate and deliver information. Figure 1 illustrates an overview of an enterprise portal.

Features of a Portal

The following table provides a list of features commonly found in portal products. It should be noted that the fractious portal market has provided a wide spread of features in each product. In fact, JSR 168, like all standards developed by industry committees, only establishes the minimal required set of portal features.

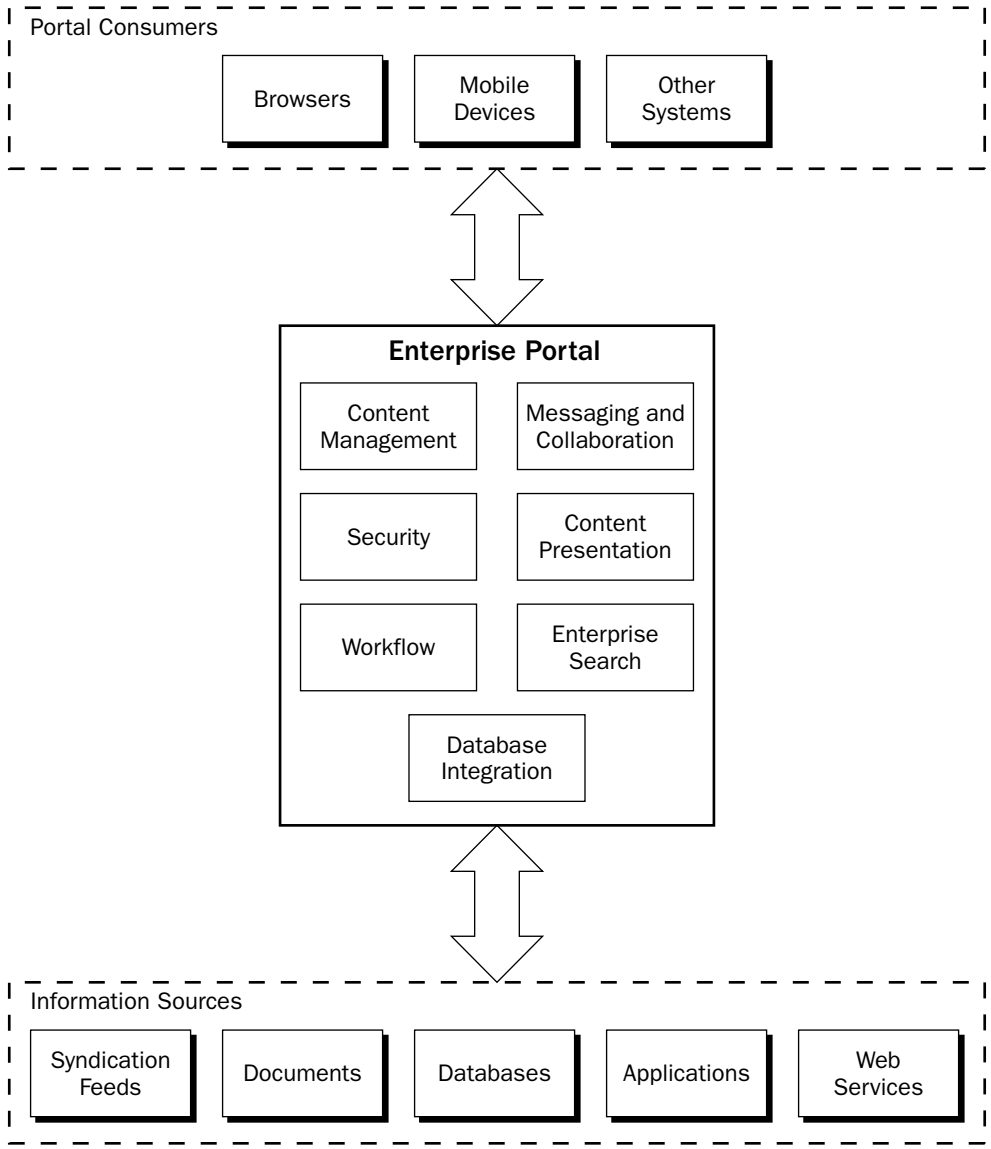
Portal Consumers
Browsers
Mobile Devices
Other Systems
Enterprise Portal
Content Management
Messaging and Collaboration
Security
Content Presentation
Workflow
Enterprise Search
Database Integration
Information Sources
Syndication Feeds
Documents
Databases
Applications
Web Services

Figure 1

Feature	Description
Aggregation of content	The capability to tie different content fragments into one consistent and interoperating view.
Customized views	Customization commonly refers to having different views based on the role of the person in the organization. For example, HR personnel would have one view, while Finance personnel would have a different one — each customized for their specific job needs.
Personalized content	Personalization takes customization one step further and provides the capability for individual users to customize their view to suit their individual needs. For example, Joe may work in HR, but be specifically responsible for hiring Java developers, so he would want to personalize his view for that task, and perhaps link to content in the Java domain, such as Apache.
Unified security model	Users have an account for their time card application, their HR system, and so on. This provides not only *single sign-on*, but also an enterprise-wide security policy based on role.
Collaboration features	While some portals may provide sophisticated tools for chat, application sharing, common whiteboards, conferencing, and so on, collaboration is mainly about developing communities of interest, whereby people can share common expertise and insight on a particular set of data. For example, a finance user may want to comment on a particular division's report, providing advice or asking questions regarding particular investment decisions.
Localization	Localization involves customizing content to the locale in which it is being presented. This could involve character sets, language, currency exchange, date format, and so on.
Internationalization	Internationalization support enables an application or portal to be developed in such a way that it can be easily localized. Depending on the design of the portal, this can be very hard or very easy.
Workflow	Workflow supports the user's ability to seamlessly move through a set of tasks across multiple data sources and applications. For example, a user may need to pull data from three sources (catalog, inventory, and partner inventory), and update two others (shipping and invoicing) as part of completing a supply chain transaction.
Web services access	Web services have become the latest development in the trend toward interoperability. They provide a strong capability to both access partner systems (business to business) and be consumed by consumer applications (business to consumer). A portal should be able to both consume Web services and provide them.
Self-service	A recent trend in portals, particularly those for external consumption, has been toward users being able to provide self-service. The idea is that it should be easy for a user to provide and access sufficient information to conduct transactions with minimal or no support from other people.

Feature	Description
Client agnostic processing	A portal should be able to service not only many different browsers (all but the most junior Web developers understand how difficult this can be), but also different devices (such as mobile devices) and platforms (such as other applications).

As mentioned before, this table does not encompass all of the features to be found in any portal suite available on the market. Nor is this a minimal set of features expected in any portal. Instead, this list is meant to cut through the buzzwords and provide an understanding of the kind of features that a portal can provide to your enterprise.

Components of a Portal Framework

In addition to having a set of conceptual features, a portal framework also contains a set of commonly found components. Again, this list is neither comprehensive nor limiting, but it provides a solid overview of the kind of components that are available in many portals. The following table describes these common enterprise components:

Enterprise Component	Description
Content management	Content management suites vary greatly in what they do in terms of access control, content markup, presentation, revisioning, and so on. However, most portals maintain some capability to publish and maintain their content.
Syndication access	A newswire provides a good analogy for a syndication feed. Essentially, Web sites produce and update lists of new content. Portals render that content and regularly check for updates from the syndication feed. There are several feed standards, including OCS, RSS 0.9, RSS 1.0, RSS 2.0, and so on. These are usually cached locally on the server for all clients with an interest in that given feed.
Mail	Mail support varies greatly in portals. Almost all provide some support, even if it's only as simple as mailing back user name and password information. More sophisticated implementations enable a Web-based mail client or even e-mail subscriptions to particular content.
Search Engine	While portals go a long way in providing a road map to the information users need to find, users will still need the capability to search through the content sources for things that are too fine-grained for the road map. Portals often exhibit wide disparities in their search engine capabilities, with some having a search tool built into the portal, while others simply support plugging in another tool. This can lead to confusion on the part of developers about which search engine capabilities are actually part of the software they acquired, and which need to be acquired.

Table continued on following page

Enterprise Component	Description
Database Access	Most portals provide a "database browser" portal, which wraps SQL calls into a browsing interface. Others also come with sophisticated object-to-relational mapping suites that can be used to tie databases seamlessly into the portal framework.
Collaboration tools	Threaded, searchable discussion forums are the most common collaboration tools that come with a portal. However, many also include live chat rooms, whiteboard tools, and so on.

This table gives you an appreciation for some of the components commonly found in portal suites. However, we should probably provide you with a formal description of a portlet.

What Is a Portlet?

"Portlets are little windows into your enterprise."

"Portlets are those boxes you see on a Web page."

"Isn't that what they call those portable restrooms you see at construction sites?"

In fact, portlets are reusable user interface Web components that provide a view of an information system. These components provide a markup **fragment,** as they are called by JSR 168, which enables them to be aggregated into larger **portal pages**. This definition, as well as the definition of portal containers and their relationships to servlets and JSP, is covered in detail in Chapter 1.

A Brief History of Enterprise Portals

In the beginning, there were research papers, and those research papers were good. They were expense to print and share, however, so they were shared electronically over the network now called the Internet. The number of users on this network was relatively small, and the places where papers could be found were rather well known, as the users were generally sophisticated.

Tools such as Mosaic and Netscape Navigator were developed to make it even easier for less sophisticated users to browse information online. As the ease of use and publication increased, so did the amount of data. Soon, the Internet was open to commercial ventures, and data of all types (regrettably, in some cases) was soon being shared across this information superhighway.

The User Perspective

It quickly became important to find a way for users to easily find what they were looking for in this storm of information available on the Internet. This imperative resulted in the first generation of **search engines**. Search engines did make it easier to quickly find resources and services on the Internet, but it was still hard to ensure that you got the correct and best data available on a given topic. Effective Web research was still at the mercy of the sophistication of the users.

What if someone who knew the Internet pretty well could put together a road map? Someone did, and called it Yahoo!. It provides a category list through which users can drill down to the information and Web sites that they need. If you want something such as Atlanta Braves statistics, for example, you can find them under a treelike mapping whose topics range from broad to narrow: Directory > Recreation > Sports > Baseball > Major League Baseball (MLB) > Teams > Atlanta Braves. Ultimately, it would provide you with the relevant pages — in this case, links to a number of sports Web sites that maintain an Atlanta Braves page. In addition, you would find a link to sites that enable you to buy tickets at Turner Field. You could have guessed that such sites existed, and you probably could find them eventually, but the search engine enables you to quickly access all of the relevant information (in theory) on the Atlanta Braves. (This shows the ability of a human to have a better understanding of what you are actually seeking than a machine, which gathers its information from a few words and a number of sophisticated algorithms.)

Now, you can bookmark this page and return to it at any time (using the same computer) to find your information on the Atlanta Braves. But what if you want to check the score from last night's game, and get any news from the Braves' clubhouse? Even if you could get that information, you would still want it delivered without asking for it, and many sports sites do have individualized pages for every team in major sports. This is known as **customization** — providing customized content for particular users based on their interest (Braves fan). Moreover, you can get customized information about all of your interests, saving you a tremendous amount of time.

This desire for speed and customization is what fed the development of My Yahoo! It is a portal that enables you to personalize your view with a certain look and feel, with content defined by you — your sports teams, your stocks, your news, and your links. This was clearly a breakthrough in that you could actually cater the Web to your interests, at least as a starting point.

The Business Perspective

It quickly became apparent to businesses that there was real value in getting information to customers, partners, and employees easily and efficiently. The easier the communication became, the quicker the money changed hands. In addition, informational portals began to appear for advertising and subscription purposes. An example of this was ESPN, which provided great sports information. By becoming the preferred place to receive sports information on the Internet, it was able to sell premium content and advertising.

Aside from a being a new media channel, organizations began to recognize an internal value to Web-enabling their applications, databases, and so on. They realized that as their employees became more proficient with the Web browser, training costs for using many of their applications could be decreased, as the company apps would behave similarly to the commercial apps (for example, ordering a book from Amazon). Furthermore, they realized the lower maintenance costs of having one application on the desktop (the Web browser which was standard with the operating system — no statement about the Microsoft lawsuit is implied here) and centrally managing the rest of the application on the server side. This enabled patches, upgrades, and so on to be installed on one machine and affect all of the users. In addition, the test environment of that one machine was much easier to control (because it was easier to match the configuration for only one machine than for one machine per person in your organization). This reduces the number of configuration incompatibilities, and conflicts dramatically wane.

Figure 2 illustrates how organizational applications interacted prior to Web-enabling them.

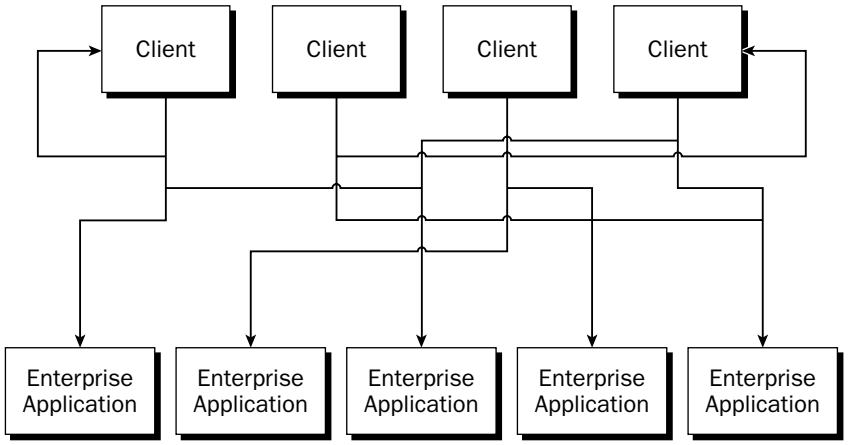

Figure 2

The interdependencies between applications and the subsequent impact of these interdependencies on the organization were quite real, complex, and problematic. However, Figure 3 illustrates what things looked like after Web-enabling the applications.

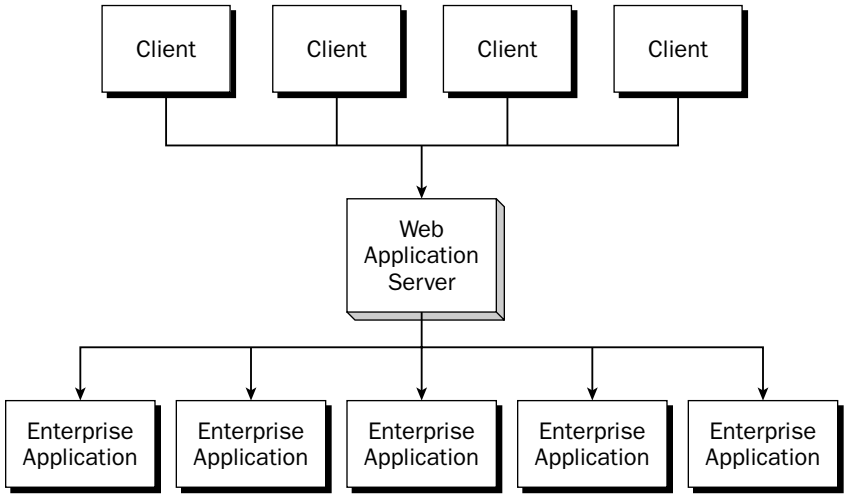

Figure 3

As you can see, Web-enabling the applications cleared up the spaghetti dependency diagram, and the similarities to the portal diagram in Figure 1 are unavoidable.

However, while this appears to be a single, consolidated unit, that is only the illusion of the hardware and network connections. Typically, the integration followed a popular technique at the time — integration by user. Essentially, the user had a front page with links to each of the applications, which the user had to access to find or process a particular piece of information. If users were lucky, they could log in to

each of them with the same user name and password, but there was still no concept of a **single sign-on,** or a consistent security model. Furthermore, users were treated as if they were all the same. The data they required and the applications they could access were uniform across the enterprise.

This deficiency led to the development of the enterprise portal, which sought to bring the same concepts to an individual enterprise. Connecting all the apps into one consistent network — available to all of the users based on their role — obviated the need for repetitive data-hunting, and for cutting and pasting information into other applications until that process yielded the appropriate results.

The Technology Perspective

As soon as you identify a need, you will find a software company willing to oversell their product as the solution to that need, including a bunch of needs that you didn't know you had. When the first enterprise portal products came out, they had entirely proprietary APIs, and an interesting mix of feature sets. For example, some had personalization features, others strong content management, while others excelled at enabling workflow. Many had good **enterprise application integration** features. Eventually, though, they all ended up providing a core feature set.

Along the way, J2EE began its meteoric rise through the server-side world. It provided strong enterprise application development and integration features. It also had a substantial impact on Web development activities with the servlet and JSP specifications.

Quickly, though, the portal capabilities became **technical discriminators,** and provided nonstandard extensions to J2EE. These extensions ranged from very close approximations of the standard components, with limited wrappers, to full-blown rewrites of presentation logic code. Some of these extensions even claimed to be predecessors to standard APIs — despite having been released after the standard was widely adopted.

The problem was that the portal implementations were starting to fracture the J2EE application base, which would defeat the driver behind its success — the portability of enterprise applications. IBM recognized the problem and proposed JSR 162, but Sun disagreed with their approach and proposed JSR 167. Ultimately, Sun and IBM agreed to combine their two proposals into JSR 168. JSR 168 evolved slowly as a result of the need to resolve these differences, as well as to compensate for the emerging OASIS standard on Web Services for Remote Portals (WSRP).

> *Web Services for Remote Portals (WSRP) is an OASIS standard that views the portal and Web service interaction from a completely different angle. It conceptualizes what are known as* ***forward-facing Web services****, which are Web services that are data-centric and provide a specification for how to render that data. These Web services would like a contract to which they can adhere and know how they will be presented to users. This is discussed further in Chapter 10.*

Challenges in Building Portals

Unfortunately, many portal development efforts have been highly fragmented, which has resulted in applications that are difficult to navigate and frustrate users. In fact, the authors of this book first worked together to save a project locked in a textbook example of this phenomenon. Many of these

problems occurred because of poor design and improper development strategies that involved the application of proprietary solutions that were difficult to deploy and maintain.

Additionally, improperly designed portal projects have forced organizations to absorb development costs that were disproportionate to the value added. Many of these efforts have resulted in the dilution of a corporation's image and have decreased customer loyalty.

The fact that so many portal failures have occurred should come as no surprise to those who have been involved in portal development programs. Portal application developments are arduous. They require the successful amalgamation of many different technologies and applications that must satisfy a disparate end-user community.

Are Portals Here to Stay?

IDC estimates the market for enterprise portal platforms will reach more than 2.6 billion dollars by 2006. Despite the overall depression in tech spending, Gartner estimates that portal software spending grew 59 percent to $709 million dollars in 2001. Delphi Group estimates that the Portal Market in 2003 will reach $957 million worldwide, and reach $1.14 billion in 2004. Nearly all Web development efforts are being geared toward integration Web portals. This means that a large portion of Java Web developers will be using portal software. Much like JBoss and Tomcat provide open-source options for those who develop Java Web applications, the current portal options have been limited, requiring individual experience in a wide variety of open-source projects.

The money flow clearly indicates that IT investments will continue in portal software. Therefore, developers need to be well versed in this area — not only to provide organizations with the capability to save significant amounts of money by leveraging open-source products, but to keep their own career options open.

What Is Wrong with Web Applications?

There is nothing wrong with conventional Web applications. As a matter of fact, portals are primarily based on them. For a lot of cases, all that needs to be developed is a Web application; done properly, it can easily merge into a portal environment later.

Where developers can run afoul is by ignoring the new portal standards available and writing their own aggregation mechanisms. If, as a developer, you build your own nonstandard portal implementation, you prevent the portability of your application.

However, note that the solution needed is often a simple Web application, which has no requirements for a portal. Suppose, for example, you need a new billing application. You should not assume you need a portal and that, within that portal, you will find the solution to making your application. Instead, you can build the application as necessary, realizing that doing integration-friendly things is good software practice, just as it is good practice to build what customers need, not what you want to give them.

We believe that the concept of the portal as a desktop replacement is still a bit far-fetched and unrealistic, because a Web browser is only *one* of the tools most people use on a daily basis. While centrally managing