

MODX Revolution Building the Web Your Way

A Journey Through A Content Management Framework

W. Shawn Wilkerson

MODX Revolution Building the Web Your Way: A Journey Through a Content Management Framework

Published by Sanity Press, a Sanity LLC company (sanitypress.com), Daytona Beach, FL 32117. Copyright © 2012 Sanity LLC All rights reserved. The editorial arrangement, analysis, and professional commentary are subject to this copyright notice. No portion of this book may be copied, retransmitted, re posted, duplicated, photocopied, or otherwise reproduced without the express written approval of the author, except by reviewers who may quote brief excerpts in connection with a review. Any unauthorized copying, reproduction, translation, or distribution of any part of this material without permission by the publisher is prohibited and against the law.

Warning and Disclaimer: No information contained in this book should be considered as financial, tax, business, or legal advice. Your reliance upon information and content obtained by you at or through this publication is solely at your own risk. Sanity LLC, Sanity Press nor the author assume no liability or responsibility for damage or injury to you, other persons, or property arising from any use of any product, information, idea, or instruction contained in the content or services provided to you through this book. Reliance upon information contained in this material is solely at the reader's own risk. Sanity LLC, Sanity Press, and the author makes no warranties, either express or implied, regarding the content or its fitness for any particular purpose. The authors have no financial interest in and receive no compensation from manufacturers of products or web sites mentioned in this book.

Trademarks: MODX, MODX Revolution, MODX Evolution are trademarks or registered trademarks of MODX, LLC. All other trademarks are the property of their respective owners. The publisher and the author is not associated with any product or vendor mentioned in this book.

Library of Congress Control Number: 2012914262

ISBN-13: 978-0985853204 (Sanity Press)

ISBN-10: 0985853204

Printed in the United States of America

First Printing: August 2012

07 06 05 04 4 3 2 1

Bulk Sales: Sanity Press offers discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

Sanity Press

1-386-322-7999

sales@sanitypress.com

Abbreviated Contents

Foundations	2
Introducing MODX Revolution	4
Discovering The Heart of MODX Revolution: The Manager	16
Creating Project Templates	46
[[<code>\$Chunks</code>]]:Simplifying Web Projects Using (X)HTML	58
[[<code>*templateVariables</code>]]: Building Interactive Interfaces	74
[[<code>Snippets</code>]], Plug-ins and Widgets: Adding PHP Dynamic Content	98
:filters - The Art of Manipulating Element Output	160
Live Projects	176
Quick Start: Putting the Pieces Together	178
Streamlining Web Projects With SEO, FURLs, and Minify	198
Power Blogging	222
Dynamic JavaScript, CSS, and jQuery Essentials	268
AJAX Content Injection	296
Administration	310
An Overview of Grouping: Category, User and Resource	312
Access Control List Implementation	328
Contexts: Implementing Sites with Multiple Faces	354
Development	378
MODX Revolution API Concepts	380
xPDO: Foundations For Development	420
xPDO: Third-Party Applications	452
Now What?	532
Appendices	536

Contents

Part I: Foundations

Introducing MODX Revolution	4
What is MODX Revolution?	5
Example MODX Revolution Web Projects	7
What Sets MODX Revolution Apart?	8
Simple File Structure	9
Communication Facilitation	10
Streamlined Workflow	11
Project Resources	12
My Current Endeavours	13
Discovering The Heart of MODX Revolution: The Manager	16
The Evolution of MODX Managers	16
Welcome Screen	19
The MODX Revolution Manager	20
Exploring the Menu	21
Dashboard	21
Site Menu	21
Components Menu	23
Security Menu	24
Tools Menu	29
Reports Menu	34
System Menu	37
User menu	42
Support menu	43
Creating Project Templates	46
Concerning Templates	46
MODX Revolution Templates	47
Template Inheritance	47

Naming Conventions	48
Creating a Template	49
Resource Tags	50
System Settings Tags	51
(X)HTML Tags	51
Storing Templates in the Manager	52
Text Editors	52
Example XHTML 1.0 Strict Template	53
W3C Validation	55
Template Storage Alternatives	56
Templates available via Package Management	56
Using Media Sources: Filesystem	56
[[\${Chunks}]]: Simplifying Web Projects Using (X)HTML	58
<hr/>	
Introducing (X)HTML [[\${Chunks}]]	59
Creating a Chunk	59
[[\${siteHeader}]]	60
[[\${jQuery}]]	61
[[\${siteFooter}]]	61
Chunks within Chunks	63
Using Chunks to Streamline Templates	63
Showing the Pieces Together	65
Increasing Readability	67
Using Chunks With Placeholders	68
Using Chunks with Property Sets	69
Overriding a Property Set	70
Chunks Used With Snippets	70
[[\${globalNav}]]	71
[[\${wfNavBar}]]	71
Example [[\${globalNav}]] Output	72
Test Implementations with Duplicate Elements	73
[[*templateVariables]]: Building Interactive Interfaces	74
<hr/>	
What is a Template Variable?	74
Creating Template Variables	75
Place Holder Template Variables	76
Attaching the TV to a Template	77
Understanding Input and Output	77
Selecting The Type of Template Variable Input	78

Template Variable Input Types as of 2.2.2	79
Template Variable Examples	80
Auto-tag	80
Check Box	82
Date	83
Email	84
File	85
Hidden	87
Image	88
Listbox	89
Radio Option	90
Resource List	91
Rich Text Editor	91
Tag	92
Text	93
Textarea	93
URL (Uniform Resource Locator)	94
Forcing the Use of Template Variables	94
Output Template Variables	95
Delimiter	95
HTML Tag	95
Rich Text	96
Custom Forms	96
Database considerations	96
<u>[[Snippets]], Plug-ins and Widgets: Adding PHP Dynamic Content</u>	98
Snippets	99
Why is it called a Snippet?	100
The Dangerous Side to Snippets	100
Verify Snippet Functionality and Output	101
Package Example: Wayfinder	101
Exploring the Directory Structure	102
The Wayfinder Snippet	102
When the Code is All The Documentation We Have	105
Deciphering Snippet Activity	105
Finding the Required Parameters	106
When A Snippet Exists For Evolution And Revolution	107
Snippet Calls	108
Syntax and Caching Implications	108

Use Backticks (`) For Parameters	108
Wayfinder Implementation Samples	109
Global Horizontal Navigation Bar	109
Vertical Navigation	111
Building a Wiki-type Layout	112
Included Navigation Configurations	114
Getting Ready To Build Your Own Snippet	115
Coding Advice	115
MODX Revolution Coding Expectations	117
Using A Basic Structure	118
Part 1: File and Package Information	122
Part 2: Legal Declarations	122
Part 3: Snippet Description and Parameter Documentation	123
Part 4: Functions	123
Part 5: Variable Declaration	124
Part 6: The Workhorse	125
Part 7: Return Value	125
Using the Simplest Solution	126
Adding A Snippet To The Project	128
Steps to Creating a New Snippet	128
Pasting Our Snippet Code	129
Verify Snippet Output	130
Working with Placeholders	132
Simple Example	132
Why Placeholders?	133
Working With Multiple Placeholders and a Single Snippet	134
Using Chunks To Template Snippet Output	137
Create a Snippet to Collect The Data	137
Add An Optional Property Set	141
Building Snippet Frameworks	143
Create A Chunk To Use As A Template	144
Benefits Of Chunk Templates Used For Snippet Output	145
Implementing A Templated Snippet	146
Snippet output	147
Making the Workflow and Snippet Simpler	148
Working With PHP Classes	150
\$modx->runSnippet()	153
Exotic Snippet Calls	154
Plugins	155
Plugin Example: AntiSpam	155

Anti-Spam Output	158
Assigning Priority to Plugins	158
When to use Plugins instead of Snippets	159

:filters - The Art of Manipulating Element Output **160**

Filters defined	161
The Benefits of Filters	161
Implementation Examples	162
Conditional Overrides and Targeting	162
else	162
hide	162
show	162
then	162
Compare Operators	163
eq, is, equals, equalto, isequal, isequalto	163
gt, isgt, greaterthan, isgreaterthan	163
gte, isgte, eg, ge, equalorgreaterthan, greaterthanorequalto	163
input, if	163
lt, islt, lessthan, lowerthan, islessthan, islowerthan	163
lte, islte, le, el, lessthanorequalto, equaltoorlessthan	163
ne, neq, isnot, isnt, notequals, notequalto	163
Content Injection	164
cssToHead	164
htmlToBottom	164
htmlToHead	164
jsToBottom	164
jsToHead	164
toPlaceholder	164
Date Formatting	165
ago	165
date	165
fuzzydate	165
strtotime	165
Element Filters	165
select	165
tag	165
Logic	166
and	166
or	166
Math Operations	166
add, increment, incr	166
divide, div	166
math	166
modulus, mod	166
multiply, mpy	166

subtract, decrement, decr	166
String Manipulation	167
cat	167
cdata	167
default, ifempty, isempty, empty	167
esc, escape	167
ellipsis	167
htmlent, htmlentities	168
ifnotempty, isnotempty, notempty, !empty	168
length, len, strlen	168
lcase, lowercase, strtolower	168
limit	168
md5	169
nl2br	169
notags, striptags, stripTags, strip_tags	169
replace	169
reverse, strrev	169
strip	169
stripString	170
ucase, uppercase, strtoupper	170
ucfirst	170
ucwords	170
urldecode	170
urlencode	170
wordwrap	170
wordwrapcut	170
User Filters	171
isLoggedIn	171
ismember, memberof, mo	171
isnotloggedin	171
userinfo	171
Custom Filter	172
Successful Implementation Considerations	173
Sample Each Stage	173

~~~~~

## *Part II: Live Projects*

~~~~~

Quick Start: Putting the Pieces Together	178
Change System Settings (Chapter 2)	179
Add A Site Template (Chapter 3)	180
Create Chunks (Chapter 4)	181
Create Template Variables (Chapter 5)	183
Install Navigation (Chapter 6)	185

Add AntiSpam Plugin	185
Add Custom PHP Snippets	186
Edit The Home Page	187
Add Additional Resources	188
Create Subdirectories / Containers	189
Add Additional Pages To A Container	189
Thinking Forward: Structure and Project Direction	190
Adding A Contact Form	191
Adding Site Search (For Static Content Only)	196
<u>Streamlining Web Projects With SEO, FURLs, and Minify</u>	<u>198</u>
Understanding The Search Engine	199
Optimizing a Site For Search Engines	199
Configuring: Friendly URLs	200
Modify .htaccess file	200
Configure MODX Revolution To Use Friendly URLs	202
Check All Templates and Header Chunks For a <base href> Tag	203
Clear the MODX Revolution cache	203
Minified JavaScript / Cascading Style Sheets	204
Installing Minify	204
Match Settings to Server Configuration	205
Testing the Minify Implementation	208
Establishing Groups	210
Forcing JavaScript To The Page Bottom	212
Basic .htaccess Optimizations	213
Strategic Utilization of System Settings and Elements	215
SEO Benefits From the Site Structure	216
Webmaster Resources	217
Creating a Site Map	217
robots.txt	218
humans.txt	218
Dynamic Content	219
<u>Power Blogging</u>	<u>222</u>
Sample Project Requirements	223
 Blogging Foundations	223
Summary of Installed Packages For “Power” Blogging	225
Creating a Blog Implementation	225
General Web Project	225

User Authentication	226
Manually Building a Blog	227
Site Templates	227
Blog Related Chunks	232
Wayfinder Related Chunks	234
Taglister Related Chunks	235
Site - Wide Chunks	235
Individual Author's Resource Page	236
Custom Snippets	236
Screen Shots Taken from 1truth.net	237
Final Notes on Creating a Blog From Scratch	239
Implementing the Articles Add-on	240
Templates	241
Template Variables	252
Creating a Chunk for Category and Other Pages	253
Creating Category Pages	254
Assign Templates to the Article Container	256
Advanced Settings	257
Advanced Settings Highlights	258
Third-Party Services	259
<u>Dynamic JavaScript, CSS, and jQuery Essentials</u>	<u>268</u>
Loading JavaScript into the Resource	269
Forcing JavaScript and HTML to the Bottom of a Resource	269
Forcing CSS, HTML and JavaScript to the Page <head>	270
Using Chunks to Create Version Specific Dynamic Links	271
Laying the JavaScript Foundation	271
Testing the Foundation	272
Implementing Site Wide JavaScript	273
Using Minify's To Combine External JavaScript Files	274
jQuery Foundational Concepts	275
Default document.ready Container	275
MODX Revolution jQuery Implementations	276
Manipulating Hyperlinks	276
Limiting The Number of List Items Displayed	277
Sortable Tables and Working With Runtime Configurations	279
Toggle the Display of Page Content by Element Class	283
Extremely Simple Lightbox	286
Wayfinder Combined With jQuery	289

flashembed()	293
Final Thoughts on Content Injection	294

AJAX Content Injection **296**

AJAX (Asynchronous JavaScript And XML)	297
To AJAX Or Not To AJAX	298
Argument For Implementing AJAX	298
Argument Resistant to AJAX	298
Component Essentials	299
An AJAX Template	299
AJAX Resources	299
Assorted AJAX Implementations	299
Manipulating Snippet Presentation: Extending The Archivist Tree	300
Using \$modx->runSnippet()	304
User Details	305

~~~~~

### ***Part III: Administration***

~~~~~

An Overview of Grouping: Category, User and Resource **312**

Element Categories	313
Resource Group Considerations	315
Adding a Resource to a Resource Group	316
User Groups	317
User Groups: A World Unto Themselves	319
Using the Access Wizard for User and Resource Group Creation	320
Adding Users To A User Group	321
User Roles	322
Policy Templates	323
Policies	325
How the Pieces Work Together	326

Access Control List Implementation **328**

Implementing The Login Package	329
User Account Considerations	331
Implementing a "Trusted User" Read-Only Area	332
Create A Resource Group	332
Create a Resource and add it to the Resource Group	333

Create a User Group	333
Create the Minimum User Role	334
Add a Site Administrator to the User Group	335
Add Additional Users	336
Assign Minimum Roles to Contexts	336
Add Resource Access	337
Verify Settings	338
When Everything is Satisfactory, Click The Save Button.	341
Force All Users to Log Back Into Site	341
Testing the Read-Only Implementation	342
Extending the User Group	343
Add New User Roles to the Project	343
Place Users Within The Newly Created Roles	344
Adjust the minimum User Role for Manager Access	344
Adjust the minimum User Role to access Resources	345
Add Associated Element Categories	345
Attach a Dashboard to the User Group	346
Example Modified Dashboard	350
Targeted Application	351
Contexts: Implementing Sites with Multiple Faces	354
<hr/>	
Laying a Foundation	355
Possible Malfunctions	356
Simple Precautions	356
Understanding Context and User Group Differences	357
Creating a New Context	358
Implementing a Context “Skeleton”	359
Example Site Skeleton Components	359
Global Site Header	361
Utilizing the Site Skeleton Context	362
Prerequisites For Domain Name Specific Contexts	363
Configuring Domain Names on Shared Hosts	365
Hosting Many Web Sites	366
.htaccess Multi-domain Modifications	366
robots.txt and humans.txt Considerations	367
Create or Duplicate a New Context	368
Activating The New Context	368
Subdomain Contexts	369
Forcing the Subdomain to the Site Root	369

.htaccess subdomain Modifications	370
Create or Duplicate a Context for the Subdomain To Use	371
Implement or Edit the domainGateway Plugin	371
domainGateway with Domains and Subdomains	372
Protecting the Manager Context	373
Examples of using Snippets and Chunks with Contexts	374

~~~~~

***Part IV: Development***

~~~~~

MODX Revolution API Concepts	380
Concerning Object Oriented Programming	381
The \$modx Object	382
Benefits of utilizing the MODX Object	382
The Dangerous Side to the MODX Revolution API	383
Verify PHP Functionality and Output	383
\$modx Object Implementation - importing the API	384
Secondary MODX Revolution Classes	385
Avoid Garbage In / Garbage Out Scenarios	386
Retrieving Values From Object Attributes	387
Object Instantiation	388
get(): Directly Accessing Object Attributes	389
toArray(): Returning PHP Arrays of Object Contents	391
toJSON(): Returning JavaScript Object Notation	392
Retrieving Multiple Related Objects	393
getOne(): Accessing a 1:1/0 Relationship	394
getMany(): Accessing a 1:n Relationship	396
Creating an Object Test Bed	398
Creating a Resource Programmatically	403
Establish a Baseline	403
Building an Implementation	405
Updating the User On-The-Fly	407
Mastering the Elements	409
The Small Print	411
All Snippets are treated as functions	411
Resources are Cached as Arrays of Elements	412
The Difference Between Cached and Non Cached Elements	417
Snippets Can Serve as Functions	418

xPDO: Foundations For Development	420
xPDO Overview	421
xPDODriver	422
xPDOGenerator	422
xPDOManager	422
xPDOObject	423
xPDOSimpleObject	423
Schemata	424
Hierarchy Structure	424
The Model	425
The Object	426
The field	427
Relationships	428
Quick Start: Creating a Single Database Lookup Table	429
Step 1: Create a Simple xPDO Schema	429
Step 2: Create a Working Directory	430
Step 3: Upload the Schema to the model/schema directory	430
Step 4: Build the xPDO Model	431
Step 5: Creating Database Table(s) using the xPDO Model	432
Step 6: Instantiate an Object of the Package	434
Step 7: Populate the Table	435
Step 8: Assign a Category (Optional)	436
Packaging Components via Packman	437
Creating a Multi-Table Schema	440
Solutions Diversify As Complexity Increases	440
Lay the Primary Tables Out First	443
Table Normalization	443
Using Names and Variables to Establish Consistency	444
Use the Schema Structure to Clearly Represent Function	445
Establishing The Ground Rules For Relationships	446
Composite Relationships: Planning Cascading Deletions	447
Aggregate: Establishing Non-deleting Relationships	448
Common Solutions to Simple Mistakes	449
Best Examples: Learning from Project Schema	450
xPDO: Third-Party Applications	452
Choosing How To Load The Application	454
Laying the Foundation	455
Create The Application Directory Tree(s)	455

Initialize An xPDO Schema	455
Create a "Parent Class" For the Project	457
Populate Tables With Data	459
Small Beginnings	467
Client Sketch Describing Basic Functionality	468
Retrieving Base Components	469
A Procedural Programming Quagmire	470
discjockey xPDO Schema	471
Discjockey Main Class	476
Extended xPDO Derived Classes	489
MODX Revolution Snippets	494
jQuery Javascript Files	502
Resources	509
Chunk Templates	513
Primary Cascading Style Sheet	516
Screen Shots From discjockey Created For This Chapter	524

Now What? **532**

~~~~~  
***Appendices***  
 ~~~~~

MODX Revolution Terminology **538**

Resource and System Settings **542**

Resource Settings	542
Resource Settings	543
System Settings	544

Primary Object Cheat Sheets **554**

modChunk	555
modPlugin	557
modSnippet	559
modUser	561
modResource	563

The Document Parser And Cache Mechanism **570**

Simplified Parser Sequence of Default Operation	571
Straight From The Core - with additional comments added	572
Cache Mechanism	573
Cached Versus Uncached Snippet Calls	574
Placeholder Considerations	574
Common Errors to Avoid	575

Internet Caching Technologies **576**

JavaScript and CSS Caching	576
HTTP Protocol Caching	577
PHP Caching	578
Internet Proxies	578
Closing Thoughts on Internet Caching Technologie	579

About the Author

W. Shawn Wilkerson has been around most every aspect of the computer and web industries for almost three decades. His client list includes on-line gaming clans, small businesses, ministries and non profits of various sizes, international corporations, and most everything in between.

After two decades in the industry, he returned to school and achieved nine Associate of Science degrees in computer and internet technologies in five years. On the advice of two department chairs and a multitude of professors, he decided to pursue business degrees attached to technology, as the common opinion was the course content in the Computer, Networking, and Internet fields were beneath his skill set and would continue to be so, for quite some time.

In May of 2011, he was awarded a Bachelors in Applied Science in Supervision and Management, which prepares graduates to work for any international corporation on the planet. The BAS also gave insight into the clients perspective, including the uncertainty and fears many of them share when considering the unknown technology universe.

During the Summer of 2012, Shawn began his Masters in Technology Management and expects to graduate in 2014, probably with honors as he has for his ten previous scholastic endeavors.

Over the years, he has programmed in a multitude of languages, beginning in the days of COBOL, RPG, and FORTRAN. Eventually he moved on to C, C++, C##, .net and many others. He finally made the jump to web programming and on a typical day of development, is known to combine as many as six languages into a single application - many of which utilizing reusable code since his move to MODX Revolution in 2004.

He began writing articles for his personal web site (shawnwilkerson.com) and was asked to add it to the official MODX Documentation and eventually joined the official MODX Documentation Team. At the time, he was able to spend quite a few hours in IRC and in the forums helping others work out issues with early versions of MODX Revolution, which didn't go unnoticed.

As a result of his contributions and content on his site and in the official Documentation, a major book publisher contacted him with a concept they were working on for a new line of books. Eventually, this did not work out as the company in two other markets for the publisher, causing them to drop the rest of the projects they had begun under their auspicious - including the three hundred page forerunner of this book.

After spending three decades working with clients and almost two decades working with on-line technologies he has performed every job related to web production from server administration to project management and typically lives in the midst of a few thousand lines of code.

His unique style of writing combines mental imagery and plain common sense.

Dedicated to

Jesus Christ, without whom I would not be alive nor have any purpose.

Victoria Wilkerson, my wife, partner and best friend. No one else could possibly fill your shoes or take the place in my life you have. We have come so very far together and it keeps getting better.

There is truly gold and other very rare minerals to be found in your heart and your life.

Vickie, my princess, I love you so very much - savf.

My kids: Christina, Joshua, Kymberli, Heather, and Chris. My I continue to discover what you need and better serve you as a father, friend, and mentor - for as long as you will have me.

Monica Wilkerson, my mother, who allowed me to purchase my first computer: TI-99/4A, get on line with CompuServe in the 1980s, and who coaxed me into AFJROTC where I would meet the woman who be more to me than simply a wife.

Michael Wilkerson, my brother, for ever holding true to the ideal of family. Maybe now he'll understand how there is "so much to say about a computer program."

Jim and Becky Pelletier, life-time friends and Pastors, who continuously pay a very high cost so others can find their place in a very dark and cruel world. Thank you for taking the hit for wounds caused by others in the "ministry" and restoring as many who will come.

Acknowledgments

For building an amazing platform and allowing the rest of us to use it, I would like to formally thank the following individuals who spend countless hours building, expanding, and extending their platform as far as the technology will go:

Ryan Thrash, MODX, LLC CEO

Shaun McCormick, MODX Senior Developer

Jason Coward, MODX Chief Architect

Jay Gilmore, MODX Director of Marketing and Communications

The MODX Community

Preface

As having had decades of opportunity to fulfill many of the roles related to the computer and web industries, I hope to provide a resource which directly benefits site owners, designers, developers, novices and professionals alike – albeit some as a training manual and others as a quick reference. Keeping in step with MODX Revolution thinking, readers of this work can travel as far as they are willing - individually or as a team. The work should prove beneficial to a very wide audience with varying levels of experience, expertise and needs:

- ◆ Individuals who understand the basics of HTML but have little or no prior experience with Content Management Systems, Frameworks, and/or Platforms
- ◆ Those who wish to have a well rounded tool-set, which can grow with them
- ◆ Individuals who build extensive web applications, but would like to be able to build “into” an existing platform and reduce their overall development cycle
- ◆ System Analysts and Team Leaders looking to build a cohesive Web Team

University professors should be able to use this text as a foundation in which to work with students in their capstone, project management, and other web related classes. The structure of the first section is ideal for this purpose and can be conducted by establishing a goal, assigning each of the corresponding chapters to a team member, and selecting a team leader to “manage” the project taking the position to build an entire project as described in the *Quick Start* chapter. More advanced students could oversee many of these “managers” and their teams.

Developers and designers may even decide to provide a copy of this book to each of the members within their various teams as a basis of interaction. *MODX Revolution: Building The Web Your Way* provides a foundation for each person to better understand their own roles, as well as enabling each of them to perceive that role and its significance within the whole project.

MODX Revolution, by its very design, can be implemented to bridge many of the gaps found throughout the web industry. In many situations, these gaps are simply issues of tech-culture and an inability to communicate. While discussing various aspects of MODX Revolution, suggestions, perspectives, and experience may be found, which could serve in making each member more effective as they fulfil their role in the project.

I personally believe, overall efficiency within a web project team can be directly traced to their ability to inter-operate. Not only should each team member be able to expand their own skill-set, they should also be able to effectively interact with the various other members and components of a given project. The ability to coordinate efforts to where one individual or team can build directly towards the needs of the receiving individual or team can prove invaluable.

Purpose for this book

Essentially, there are very few who will read this work, who I have not had the pleasure of having had spent some time in shoes very similar to theirs. Hence, the main purpose of this work is to allow people to take a few steps in my shoes, sharing some of the experience and understanding I have acquired over the years with MODX Revolution.

This book is intended to introduce topics to the reader and also walk with them just far enough to where they can develop and try new things on their own. By no means should this work be considered the final authority, nor can any book be - especially with MODX Revolution which is extensible in every way. I have spent seven years working with this amazing product and simply desire to help others do the same.

My ultimate goal is to facilitate communication and organization while maintaining flexibility and security -- essentially help others be more productive. I am not here to preach or judge anyone, though I do have my own opinions which occasionally are on the strong side, but usually for good reason. Hopefully, I will meekly and gracefully share insight which is considered practical and valuable.

Where I fail to do this, please forgive me.

Audience and Scope

As with all technical works, we could diverge our attention in many different directions, eventually diluting the material into oblivion. I have had to make some assumptions as to the audience and have had to limit the amount of peripheral conversation to those topics which would directly facilitate the utilization and implementation of MODX Revolution.

As to audience, it is assumed that the reader has *some* experience with web site design and is somewhat familiar with (X)HTML and CSS. Previous experience with Content Management Systems is not required, although previous experience with MODX products may provide a reduced learning curve.

The ability to code in PHP is not necessary to implement MODX Revolution for many web project scenarios, but this knowledge or having access to someone who is willing to learn the API and produce very streamline code will strongly assist in the creation of dynamic applications. In those areas where we move into examples utilizing PHP, this assumption extends to having *some* experience with PHP also, specifically versions 5.2 and above. It is my intention to provide content that is easily understood by a wide range of people, even in **Part IV: Development**.

As this work centers around the implementation and interaction of various web technologies, actual instruction concerning (X)HTML, CSS, and PHP will be left to other venues. Having said this, it should be noted the developers of MODX Revolution went to great lengths in their design

to ensure usability at just about all experience levels. *MODX Revolution: Building The Web Your Way* shares this goal in allowing the reader to get the information they need, without forcing them into areas they are uncomfortable with -- though we will get into more advanced topics as we go along as discussion in the latter parts will focus on implementation and less on explanation.

Each chapter begins with foundational concepts and may very well grow increasingly more complex as the examples and content is presented. Topics previously discussed and referenced in later chapters, will only be mentioned with regard to the relevant information necessary for the current example leaving the reader to return to the earlier content for more information. Feel free to use the index and table of contents to locate the topic where it is discussed in greater detail.

For those who would like to jump right in and start off running, feel free to jump to **Part II: Live Projects**, taking special note of the *Quick Start: Putting the Pieces Together* chapter, which moves very quickly through the various aspects of MODX Revolution, essentially demonstrating the various steps to quickly get a MODX Revolution project up and running.

Part 1: Foundations contains quite a bit of explanation and numerous examples regarding each of the topics necessary to become familiar with MODX Revolution. Each chapter presents a single subject, intentionally building in complexity, while incorporating concepts presented in earlier chapters. This section should be considered a reference on utilizing the Manager and a primer on the implementation of the various Elements utilized in web projects.

Part II: Live Projects moves into real world development examples incorporating jQuery, Ajax, Search Engine optimization, and various other topics for building better projects. In this area, I have endeavored to share concepts garnered from hundreds of projects. The topics presented may occasionally slip into the gray area of personal opinion and experience. Please feel free to treat these topics as introductory and not view them as the “only” methods to accomplish these tasks - especially with SEO.

Part III: Administration introduces the concepts and techniques to effectively administrate a MODX Revolution project. Access Control Lists and other “difficult” topics are presented in a straightforward manner utilizing very familiar terminology to quickly establish concepts usually considered to be advanced simply due to their assumed complexity.

Part IV: Development takes a “stone tossed across water” view of building dynamic PHP applications using MODX Revolution as a foundation. The areas of discovery range from the most commonly used methods in the API, to building third-party applications from scratch - including the xPDO schema and extending the generated classes into full applications.

Regardless of your skill level, this book was conceived with the idea of allowing you to begin at your current level of experience and continue as far as you would want to. The examples chosen for this book are intended to relate easily to a large number of readers. I intentionally refrained from using overly technical examples and verbiage so as to relate to a larger audience.

A word concerning Content Management

When choosing a Content Management application, project leaders should consider a few things: the perspective of the developers, the environment within the community, and how well the software matches the mission and vision of the company. Fortunately for us, the MODX developers purposely build applications which do not limit or overly define our interaction, and the MODX community appears to be set on pushing every boundary they can conceive.

Many Content Management Systems, seem to imply anyone is capable of building a web project. While this is true -- to a point, there is always a cost to master a skill and direct benefits which come *only* from doing so. What many web professionals take for granted, may have had an investment of years of college, combined with frustrated hours, days, or even weeks attempting to learn a new concept. This is magnified by MODX Revolution's ability to streamline work flows and package third-party content, which may inadvertently send the message what we do "is easy".

Throughout this work, many streamlined code examples and implementations will be discussed. If web professionals actually value their clients, they will not inadvertently communicate the simplicity of their involvement, skills, or tools. We should also not be unapproachable or exclusive. Project leaders should endeavor to make the client a part of the project, which is almost always mutually beneficial.

All too often, clients will take over their project and attempt to continue their development in-house -- usually in an effort to save cost. "After all, the web people make it look so easy." I have seen multinational corporations, clubs, and non-profits turn their sites over to a volunteer or their IT department only to watch the web site eventually get banned and removed from most of the search engines. The hard lesson for many of them: anyone can throw together a web project - only a few understand the "laws" of the World Wide Web which can make the site successful.

MODX Revolution can serve as the foundation for a company to become very successful and profitable, while also providing a platform which allows its users to start where they are and grow into new things. Additionally, it facilitates partnership with clients, by possibly involving them in varying levels of the process, helping clients to clearly understand the value of having experts on board to protect and implement their best interests and represent their business to the masses.

I do not pretend or assume to present the final answer in any of the topics discussed in this book. I am simply presenting techniques and concepts which have proven reliable over time. As you progress through these pages you will undoubtedly find multiple methods presented to accomplish similar tasks. It is my hope, you will be encouraged in discovering even more of them for yourself. Once you have acquired a few of these techniques, your production should greatly increase and continue to do so as more techniques are added to your skill-set.

Welcome to MODX Revolution, settle in and get real comfortable. This is a wonderful place to discover the very edges of the web, your skills, and come together as a very effective team.

3

Creating Project Templates

Often entire areas of cities are modeled around a general theme or ethnic group. In some of these, housing sub-divisions are erected using modular homes or cookie cutter homes. The buildings appear to be similar, but each home is actually defined by its occupants and has its own unique characteristics. Essentially, this is the purpose of Templates: to provide a consistent presentation, while allowing content to have freedom of expression.

Concerning Templates

Many modern web sites have been moved to the XHTML 1.x transitional or strict types, in hopes of being easily upgraded as new technologies are introduced. Currently, standards are rapidly moving towards HTML5 and XHTML5, so the temptation to begin implementing areas of the site using these technologies may be expected by clients. For now we will be using XHTML 1 and eventually HTML5 in later chapters. Designers essentially establish a basic foundation and set of boundaries for the site when choosing which Document Type Definition (DTD) to use.

MODX Revolution is standards neutral and as such, is not concerned as to the method utilized by designers to create web content – it simply puts the pieces together. Additionally, MODX Revolution does not create or produce any HTML or any other web language, in and of itself, it is only parsing content referenced through the various Elements and Components added to the respective web site project via the Manager. This affords designers the freedom to utilize any structured markup language currently available, as well as those still in conceptual states - even targeting different technologies towards specific areas of the project. Who knows, someone out there may still require HTML3, which can easily be accomplished with a dedicated Template.

For the cutting edge designers, many modern document type definitions strongly encourage the separation of content from presentation. MODX Revolution Templates enable us to take this one step further, by facilitating the separation of page structure, content, *and* associated styling.

MODX Revolution Templates

MODX Revolution allows any text to be used as a Template and does not require that text to actually mean anything. A web page of complete gibberish can be created by designers, parsed, and displayed for the world to see. There are of course two “very minor” issues which may be of concern to some site administrators and their respective clients: 1) search engines typically ignore gibberish and 2) people usually do not enjoy watching their web browser crash. If having a web site actually visited by search engines or humans is of any significance to the success of the web site project, we would suggest using industry standard (X)HTML to design the page Templates.

Any generation of HTML and XHTML will work within MODX Revolution, though it is the author’s professional opinion that most modern web sites should use XHTML-1.0-Transitional / Strict as the bare minimum. As we are discussing the utilization of a Content Management Framework, there are extremely few reasons to use page frames, and as such we typically avoid frames at all costs. It has also been our experience, Google and other search engines typically do not value frames very highly and in some instances, will ban a site using them from their search index. The examples within this book should adhere to XHTML-1.0-Strict standards or better.

Template Inheritance

MODX Revolution defaults to allowing any Template to be used in any location within a web project. That being said, it should be explained that when a new page is created it will inherit the same Template as the parent document for the new Resource. At the top (root) of the site this infers, the default Template, as defined in the System Settings (under the System menu), will be assigned to all pages created at that level, unless a new Template is manually assigned during the creation of the individual resource. Assuming a parent document is assigned a new Template such as “blueSkies”, via the drop down menu available for that purpose when a resource is created or edited, then its children will also have “blueSkies” as their default Template.

This feature of MODX Revolution functions the way many web projects are built – everything within a directory utilizing the same presentation mechanism. Bearing this in mind, from the outset, could actually reduce the overall production time of a web site. It is by no means an enjoyable experience to manually edit dozens of child resources to change their respective Templates, even if someone were able to do an update query via MySQL based on the **parent**.

The main point here is to plan ahead. It may not be entirely unreasonable to duplicate Templates and use copies of the original in each directory or area of a web site. It is indeed possible, a few

months / years down the line, the project lead decides to have each directory themed differently, which could prove difficult to implement after the web site has “gone live.” Flexibility often needs to be directly proportionate to the project’s complexity and/or client size. Finding an efficient implementation of this flexibility can be a high-wire act – even on good days. Something as simple as the names utilized to refer to a specific Template or other Element can greatly hinder or facilitate a web project.

Naming Conventions

The acceptable *programming* practices associated with MODX Revolution requires the use of naming conventions which provide direct insight into function and purpose. Using this concept across all portions of our web project, even outside of actual programming, should prove highly effective to expedite its progress. By using colorful and descriptive names, non technical and technical users alike can more easily understand how things fit together within the web project.

MODX Revolution allows an optional description for many of the items created within the Manager, do so generously. Also, names and descriptions can be changed at any time.

The second major step for designers is typically to choose a name to refer to a Template. Previously we suggested a Template named “blueSkies”. This may appear to be generic, but it could simply indicate the Template implements a blue theme based on sky imagery. Within the context of a specific web project it may be all that is needed. By understanding the purpose of “blueSkies”, it should become apparent “greenForests” is a green theme centered around forest imagery.

For corporate web projects naming conventions should be considered even more critical. As sections of web projects are dedicated to divisions and their structures, it may become necessary to actually use the 50 character maximum length, as defined within the `MODX_site_Templates` table of the database. Often names such as `hr_intake`, `hr_interview`, `hr_request`, `reps_order`, `reps_expenses`, and `reps_commissions` are used to divide the Templates by department and function. This immediately aides in the site design, in that the valid use of a specific Template at a given location is simply indicated by its name. The Human Relations department can enjoy the simple prefix of `hr_`, whereas all content related to sales representatives can use `reps_`.

Ministries and non / not for profit organizations are usually more complex. Take membership for example. A church can have recognized members within their organization who regularly meet and interact. In terms of a web site, members can be from anywhere and not even directly related to the ministry itself. By implementing different Templates and naming them after the prospective groups, such as `src_` for Sunrise Church and `web_` for the members of a web site it should

help differentiate between the two groups. Imagine if these two member groups were mixed or reversed. Internal church data could be accessed, as well as possibly creating other privacy issues.

It should be noted, that MODX Revolution includes facilities to handle groups of users as well as documents which directly relate to many of these issues. By creating names which communicate purpose and function, it provides an additional tool to verify the correct implementation of a component within the web site.

MODX Revolution Rule #3: Choose names indicating function and purpose.

Creating a Template

The process of creating a Template, for use within a MODX Revolution project, is typically straight-forward. For (X)HTML and CSS gurus this process can take just a few short minutes. Others may take much more time to get their first Template together. Many will revise and adjust a Template until they have it functioning as they have envisioned. Over time, an astute designer will have a hand full of Templates available as a starting point for most sites. Additionally, with experience and a broader understanding of MODX Revolution, these Templates should become more streamlined and standardized in their actual implementation.

Any typical text editor can be used to create Templates – free or commercial. Some will even choose to utilize the Templates included within their hosting package. Others may search the web for hours to find that perfect Template. For users of Adobe Dreamweaver, the process to convert the preexisting Templates to fully-functional MODX Revolution versions is extremely easy. A simple understanding of what is required to get a page working is all that is necessary.

As with many Content Management Systems, MODX Revolution utilizes standard (X)HTML Templates with formatted tags placed in key positions which are dynamically replaced with content and thereby result in a page presentation. Using these tags, designers are provided with immediate access to MODX Revolution configuration settings, resource settings, and user information for utilization within a web page. MODX Revolution provides hundreds of these keywords with the vast majority being optional. To best illustrate Templates, it may be best to look at a working example.

The default install of MODX Revolution provides a minimal Template which will serve nicely to introduce us to two of the types of document tags utilized within Templates and HTML `[[$chunks]]`. Hopefully, this section of anti-dramatic web page code is very familiar with the expected exclusion of the tags specific to MODX Revolution. As with many Content Management Systems, much of the content is defined elsewhere and combined into a single tag.

Listing 03.01: default MODX Revolution Template

```

<html>
  <head>
    <title>[[++site_name]] - [[*pagetitle]]</title>
    <base href="[[++site_url]]" />
  </head>
  <body>
    [[*content]]
  </body>
</html>

```

Resource Tags

Recall that a document in MODX Revolution is called a Resource. This naming convention was selected based on the internationally accepted and standardized Uniform Resource Identifier (URI) nomenclature which specifies: “*a string of characters can be used to identify a resource via its location (URL) and its name (URN)*”. This should help clarify why documents, Static Resources and Weblinks are all referred to as Resources, as each of these can be accessed via the address bar of a web browser.

The default Template code introduces us to the resource tags: `[[*pagetitle]]` and `[[*content]]`. These particular tags are always associated with the current document being displayed in the web browser and originate as a new Resource is created from within the Manager. These values are stored in the `MODX_site_content` table allowing many of the column (field) names in this table to be directly accessed via the `*field_name` formatting demonstrated above. For simplicity, each respective field is consistently named within the Manager, the database, and as Resource tags.

It should be noted, that many of the fields are optional and may return empty if the field is, in fact, empty or comprised of a NULL value. Other fields are reportedly used for internal MODX Revolution functions and may not have a corresponding Resource tag, but are still available programmatically. For the time being, having access to the `[[*pagetitle]]` and `[[*content]]` will provide the majority of page content within most sites. The minimum Resource tag required to make a web page functional within MODX Revolution is the `[[*content]]` tag. All other resource tags are optional, though some are highly beneficial.

System Settings Tags

MODX Revolution is capable of running multiple domains, sites, and other derivatives within a single instance. It should be noted that the primary difference between System Tags and Resource Tags is an issue of scope.

System Tags are uniformly the same across an entire Context (defined in Appendix A), with each Context capable of having its own set of System Tags as it overrides the base System Settings. Resource Tags are always related to that Resource – regardless of which Context it is found in.

In the last chapter, we were introduced to the concept of System Settings noting that the default installation of MODX Revolution has approximately 150 System Settings. We demonstrated a method to change the name of a web project and indicated that the new name is stored as a variable in the `MODX_system_settings` table of the database. Many of these settings can be directly accessed following the naming convention demonstrated with `[[++site_name]]`, as all Systems Settings are prefixed with `++`.

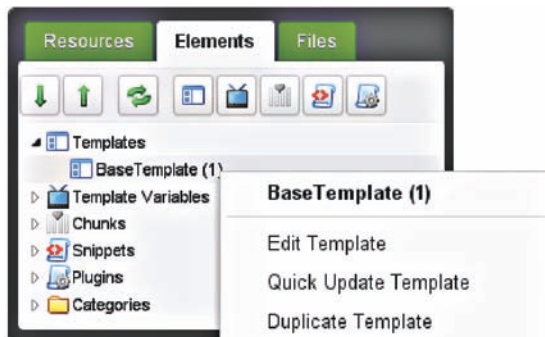
An additional difference between the System Setting Tag and the Resource Tag, is that none of the System Tags need to be included within a document for it to function. Both `[[++site_name]]` and `[[++site_url]]` could simply be replaced with straight text: Shawn's Website and `http://www.shawnwilkerson.com/` respectively.

(X)HTML Tags

The majority of the default Template is straight W3C - brand HTML, which is typical of many of the Templates implemented within MODX Revolution Sites. Any HTML or (X)HTML tag specified by the W3C is usable, *keeping in mind that not all web browsers are created equally*. Periodic testing of Templates with test content should be mandatory, as well as validation testing. Web sites such as `http://browsershots.org/` are great for letting designers see what a client sees. The W3C provides an on-line validation system, which can be found by visiting `http://validator.w3.org/`. There are also a host of other sites, in addition to similar functionality of many IDEs, which will validate and analyze the final page as well. It should be noted: the default Template should be edited, as it does not validate in its present form.

All Templates should implement the `<base href>` declaration, either with `<base href="[[++site_url]]" />`, `<base href="http://domain.com" />`, or `<base href="/" />`.

Storing Templates in the Manager



The default mechanism for working with Templates is located within the MODX Revolution Manager. To access this feature we first need to log into the website by visiting yourDomain.com/manager. Once inside the Manager, access the **Elements** tab located in the center of the Site Component and Resource Trees (left side of the display area).

Next we open the Templates tree by clicking the arrow next to the Templates option. In a default

install, this should scroll down and provide an interface similar to the one indicated. By right-clicking the “BaseTemplate” option, we are presented with a context menu providing each of the options available regarding Templates. Go ahead and left-click “Edit” in the context menu or “BaseTemplate” in the Element Tree which will take us immediately to an Editor where we can view the default Template. This can also be accomplished by simply left clicking the name itself.

Text Editors

By default, MODX Revolution includes a plain text editor, as shown when viewing the “BaseTemplate.” Initially, I would recommend leaving the default Template untouched, until you master creating MODX Revolution Templates. Optionally, the CodeMirror Package can be installed via the Package Manager in the System Menu, which provides some basic functionality.

If you prefer an editor on your computer and find yourself limited to notepad or some other simplified editor, visit the MODX Development Environment information page, located at <http://rtfm.MODX.com/display/revolution21/Setting+up+a+Development+Environment> to find a list of editors tested and used by the members of the MODX community.

Syntax highlighting is the main feature of interest here, but additional features like code completion, code suggestion, verification etc., are also nice. The more features the better, especially if they can be turned on or off, as it makes reading and editing code much easier during those 18+ hour days. Please keep in mind, it may be wise to create a local backup of the default Template, as well as any other Templates created for web projects.

For future reference, the Template name is used within the Manager for display purposes only. The identification number displayed next to the Template name in the Element tree, is the only reference stored in the `MODX_site_content` table in the database. This means a Template can be renamed as much as desired without interrupting the web site or inadvertently breaking pages due to a naming mismatch.

Example XHTML 1.0 Strict Template

To get started we will begin by creating a new, somewhat typical, MODX Revolution Template for use on our site. We will be needing access to the Manager, so feel free to keep a Manager tab open in your web browser. Select **Elements** to move into the Element tree, right-click **Templates** and select **Create a New Template** and enter the following, saving it when finished:

Listing 03.02: RevoBook Template

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
      charset=utf-8" />
    <title>[[++site_name]]'s [[*longtitle]]</title>
    <base href="[[++site_url]]" />
    <link rel="canonical" href="[[~[[*id]]]]"/>
    <meta name="author" content="w. Shawn wilkerson" />
    <meta name="copyright" content="Copyright ©2010 w. Shawn
      wilkerson, All Rights Reserved" />
    <meta name="date" content="[[*publishedon]]" />
    <meta name="description" content="[[*description]] by w.
      Shawn wilkerson" />
    <meta name="keywords" content="word1, word2, word3" />
    <link type="text/css" rel="stylesheet" href="revo.css" />
  </head>
  <body>
    <div id="header">
    </div>
    <div id="container">
      <div id="mainContent">
        [[*content]]
      </div>
      <br class="clearfloat" />
      <div id="footer">
      </div>
    </div>
  </body>
</html>

```

Listing 03.03: RevoBook HTML5 Template

```

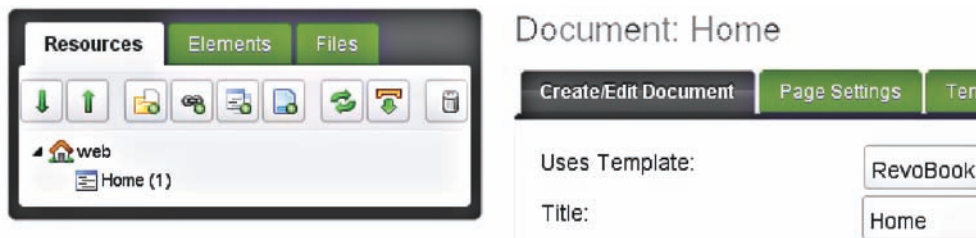
<!DOCTYPE html>
<html>
<head>
  <base href="[[++site_url]]"/>
  <meta charset="utf-8"/>
  <title>[[*pagetitle]]</title>
  <meta name="author" content="w. Shawn wilkerson" />
  <meta name="revised" content="w. wilkerson, [[*publishedon]]"/>
  <link type="text/css" rel="stylesheet" href="revo5.css" />
  <link rel="canonical" href="[[~[[*id]]]]"/>
  <!--[if IE]>
  <script src="//html5shiv.googlecode.com/svn/trunk/html5.js">
  </script>
  <![endif]-->
</head>
<body>
<header>
  <nav>
    [[- global navigation - this is to be a MODX Revo Comment ]]
  </nav>
  <div id="header_middle">

  </div>
</header>
<section class="main_content">
  <div id="container_wide">
    [[*content]]
  </div>
</section>
<br class="clear">
<footer>
  [[ - global footer element will go here]]
</footer>
</body>
</html>

```

W3C Validation

Before we do anything with this Template, we should probably take a minute to validate it. Using the left mouse button, click on the Resources tab, and then on Home. We are now going to change the Template for the Home page to use RevoBook instead of BaseTemplate. Simply use the drop down bar to Select the new Template. If it does not appear, there is a very distinct possibility the Template was not saved and the creation process will need to be repeated. Once the new Template is in place, save the document and select **View**.



Before panic sets in, remember that we have not began creating web pages or content and the default “Home” is indeed empty, hence, the bland, white, empty page. Copy the URL from the web browser address bar and use the W3C validator by visiting <http://validator.w3.org/>. Note: HTML5 validators are still a work in process. The page should correctly validate to XHTML Strict and provide a real friendly green “Passed” rating. Without any errors or warnings in our Template, this is a good starting point to begin working on the rest of the site.

Anyone who wants to live on the edge can simply type their name or whatever comes to mind in the content area of the Home Resource, save the document and then **View** the page. You should now have content appearing on the front of the web site. If you want to go all the way and leave nothing for the return trip, validate the web page with the new content and see if you continue to achieve the coveted green “Passed” recognition from the W3C. At this stage it may very well be prudent to print out the RevoBook Template and the source code from the preview of the “Home” page. Take interest in the changes between the various `[[*resource]]` and `[[++system]]` tags used in the RevoBook Template and the final page parsed by MODX Revolution. It also might be useful to use landscape page-layout and a color printer, if your editor supports this functionality.

In the web projects at my company, the use of any on-line based Rich Text Editors is strictly frowned upon – at least as far as designers and developers are concerned. Clients are another matter. There are three main reasons for this: 1) Any updates, overwrites, deletions, and other activities to sites are currently permanent. Once an Element is saved there is no going back. 2) If another Manager user (or ourselves) accesses and destroys our work of genius, we would have to resort to pulling it from a database backup – assuming there is one. 3) Rich Text Editors have a tendency to “fix” correct code. Note: versioning is in future plans for MODX Revolution.

To solve many of these issues before they turn into a perfect storm, we have established a company policy which requires all additions to the web site to be stored on a local system and backed up to a local server, before being put into MODX Revolution. There are many different methods used to have a local version of a web site. We would simply suggest: `type_name-date.ext` (template_Main-201104020.xhtml) or building sub-directories matching those used on the site or even using a directory to match the type of element being worked on (`templates\name-date.ext`). For us, each individual item is stored with a date after the name and before the extension and a combination of these solutions. Feel free to develop your own to suit your needs.

We also have another question which may eventually need to be answered, namely which Template is going to serve as the default Template of the site. Currently, the `BaseTemplate` is the default Template site wide. This can be remedied in a couple of ways:

1. `Duplicate` the default Template and rename the copy to something like MODX Default, followed by subsequently over writing the current `BaseTemplate` with our chosen default Template, and optionally renaming the Template as needed
2. Change the default Template setting in the `System | Settings` interface by using the drop-down menu.

Template Storage Alternatives

While slightly outside of the purview of this chapter, we should take a moment to mention a non-standard method of storing live Templates on a server.

Templates available via Package Management

We should probably mention, a number of Templates are conveniently available via Package Management located in the System menu. Click on `Download Extras` and select the `Front-End Templates` option near the bottom of the list. Be careful to honor the licenses attached to any MODX Revolution Addon.

Using Media Sources: Filesystem

Introduced in MODX Revolution 2.2 was the ability to store Elements using technologies and APIs from all over the web. Each medium can be defined in Media Sources and subsequently utilized through each Elements creation process. In regards to Templates, simply select the `Is Static` checkbox, and use the `Static File` (MODX Browser) to locate the file. This allows us to bypass using the Manager for subsequent modifications to the Template, as they can be simply uploaded via the IDE software, FTP, SFTP, SCP, etc.

You could also utilize your own cloud space to store and share Templates and other Elements.

Summary

MODX Revolution Templates can easily be imported via the Manager and provide a consistent presentation, while allowing content to have freedom of expression. Users may design web content using any available W3C standard, as well as any Document Type Definition (DTD) recognizable by web browsers. Multiple W3C standards can even be used within the same site. It is very important each Template utilizes the `<base href=" " />` tag.

Child documents will inherit the Template used by their parents, with the top level pages using the default Template for the site (or as defined via the context the Template is to be used in). By default, Templates can be utilized by any resource located anywhere within a web project and only need the `[[*content]]` tag to function. Many of the hundreds of MODX Revolution system and resource settings are available can greatly enhance and simplify the web design process, while predominantly remaining optional.

The names utilized to create Templates and the other components within a web project should speak to the function and purpose of the given item. Many of the MODX Revolution tags are so well named, that supporting documentation is seldom needed to ascertain their role within a web site. This habit should be developed and extensively exercised throughout the entire project.

Lastly, during this chapter we have been exposed to the vast majority of issues and topics related to Templates, while simultaneously weaving other topics into the discussion. In the process we managed to sneak something past many of you. Back in our RevoBook Templates, we stacked multiple Revolution tags together to create a variable which is changed on every page. This variable simply creates a link to the current page and is parsed from the inside out. First MODX Revolution gets the current document id, and then it creates a full path to it using the documents name: `[[~[[*id]]]]`.

This implementation represents a small sampling of the complexity which can be implemented with MODX Revolution using just its tags. Hopefully, this may be reason enough to explore these tags and to discover the power waiting to be found within MODX Revolution.

Index Of Code Listings

Preface	xxi
Introducing MODX Revolution	4
Discovering The Heart of MODX Revolution: The Manager	16
Creating Project Templates	46
Listing 03.01: default MODX Revolution Template	50
Listing 03.02: RevoBook Template	53
Listing 03.03: RevoBook HTML5 Template	54
[[\$Chunks]]: Simplifying Web Projects Using (X)HTML	58
Listing 04.01: jQuery Chunk which loads the library	61
Listing 04.02: our site footer displayed on every page	61
Listing 04.03: loading the jQuery Beauty of Code JavaScript	62
Listing 04.04: calling Chunks from within another Chunk	63
Listing 04.05: base MODX Revolution Template	64
Listing 04.06: the Template combined with Chunk output	65
Listing 04.07: [[\$youTubeVideos]] Chunk using Chunk Parameters	68
Listing 04.08: [[\$youTubeVideos]] Chunk call replacing YouTube's Embed Code	69
Listing 04.09: [[\$youTubeVideos]] using a Property Set	69
Listing 04.10: [[\$youTubeVideos]] with Property Set and parameter overrides	70
Listing 04.11: establishing global navigation	71
Listing 04.12: Wayfinder template Chunk used for the global navigation	71
Listing 04.13: global navigation menu created by the Wayfinder Snippet	72
[[*templateVariables]]: Building Interactive Interfaces	74
Listing 05.01: sample output when Text Area is assigned as output type	92
[[Snippets]], Plug-ins and Widgets: Adding PHP Dynamic Content	98

Listing 06.01: Wayfinder Snippet	103
Listing 06.02: globalNav Chunk	109
Listing 06.03: wfNavBar Chunk used as template for Wayfinder Row	110
Listing 06.04: establishing a simple vertical menu	111
Listing 06.05: replacing other Snippets with Wayfinder	112
Listing 06.06: wfListChildOrSibs Chunk	113
Listing 06.07: wfStartItemWithHeaders	113
Listing 06.08: implementing a menu style included with Wayfinder	114
Listing 06.09: getIPAddress.php	118
Excerpt 06.10: getIPAddress.php	122
Excerpt 06.11: getIPAddress.php	123
Excerpt 06.12: getIPAddress.php	123
Excerpt 06.13: getIPAddress.php	124
Excerpt 06.14: getIPAddress.php	124
Excerpt 06.15: getIPAddress.php	125
Excerpt 06.16: getIPAddress.php	125
Listing 06.17: visitorIPAddress.php	126
Listing 06.18: returnIPAddress.php	127
Listing 06.19: template_MODXSection.html	131
Listing 06.20: placeHolderIPAddress.php	132
Listing 06.21: placeHolderIPAddress.php	132
Listing 06.22: toPlaceholderAuthor.php	134
Listing 06.23: toPlaceholdersAuthor.php	134
Listing 06.24: templateAuthors.xhtml	136
Listing 06.25: showYouTube.php	138
Excerpt 06.26: showYouTube.php modified to use direct input from array	139
Excerpt 06.27: showYouTube.php with instantiated array values	143
Listing 06.28: template_youtube.xhtml Chunk	144
Listing 06.29: template_youtube.html5 compliant Chunk	145
Listing 06.30: showYouTube Snippet with template_youtube Chunk output	147
Listing 06.31: showYouTubeSimple.php	148
Listing 06.32: example.snippet.php	151
Listing 06.33: example.class.php	152
Excerpt 06.34: The template for the calling the another Snippet	153
Listing 06.35: [abbreviated] ObfuscateEmail plugin originally by Aloysius Lim	156
:filters - The Art of Manipulating Element Output	160
Listing 07.01: filterCheckLastLog Custom Filter Snippet	172

Quick Start: Putting the Pieces Together	178
Listing 08.01: base MODX Revolution Template	180
Listing 08.02: siteheader XHTML Chunk	181
Listing 08.03: jQuery XHTML Chunk	181
Listing 08.04: globalNav XHTML	182
Listing 08.05: siteFooter XHTML	182
Listing 08.06: javascript Chunk	182
Listing 08.07: wfNavBar Chunk used as template for Wayfinder Row example	185
Listing 08.08: visitorIPAddress.php	186
Listing 08.09: example contact form for use with Formit	192
Listing 08.10: example formitStandardEmail Template Chunk	193
Listing 08.11: example landing page for use with Formit	193
Streamlining Web Projects With SEO, FURLs, and Minify	198
Listing 09.01: .htaccess file	201
Excerpt 09.02: (X)HTML base href should be considered required	203
Excerpt 09.03: edited config.php for use with Minify	205
Excerpt 09.04: typical external css file linked to a page	207
Excerpt 09.05: minified external css file linked to a page	207
Excerpt 09.06: typical css file	208
Excerpt 09.07: minified css file	208
Listing 09.08: from chapter 3 to load our syntax highlighter JavaScript	210
Listing 09.09: /min/grouspConfig.php	211
Listing 09.10: Minified JavaScript group call	211
Listing 09.11: javascriptToFooter Snippet	212
Listing 09.12: load required JavaScript libraries (can be placed at Template top)	212
Listing 09.13: Page Output of javascriptToFooter	212
Listing 09.14: example .htaccess file for use with MODX Revolution	213
Excerpt 09.15: related page header content	215
Listing 09.16: wfNavBar Chunk used as template for Wayfinder Row example	216
Listing 09.17: example MODX Revolution robots.txt file	218
Listing 09.18: example MODX Revolution humans.txt implementation	218
Listing 09.19 Sample Link for Search engines to be fed information	219
Listing 09.20 Same jQuery implementation to Ajaxify a fall through link	219
Power Blogging	222
Listing 10.02: base_blogContent.xhtml Template	227
Listing 10.01: base_template.xhtml	230
Listing 10.03: blogCategoryPage Chunk uses the baseTemplate site Template	232

Listing 10.04: blogPosting Chunk (template for blog posts on non blog pages)	233
Listing 10.05: categoryList Chunk (comma separated list category Resource IDs)	233
Listing 10.06: globalNav Chunk (provides links to only top of the site)	234
Listing 10.07: wfListChildOrSibs Chunk (used on main /Authors page)	234
Listing 10.08: wfNavBar Chunk (used for global navigation)	234
Listing 10.09: wfRightMenu Chunk (used for a right menu category link)	234
Listing 10.10: tagListerListItems Chunk (item links to tag content in right menu)	235
Listing 10.11: siteFooter Chunk	235
Listing 10.12: siteHeader Chunk	235
Listing 10.13: Resource contents for individual author	236
Listing 10.14: getUserID	236
Listing 10.15: template_articles_baseTemplate.html for non blog content	241
Listing 10.16: template_articles_categoryPage.html	244
Listing 10.17: template_articles_blogPage.html	247
Listing 10.18: chunk_articles_ArticleRowTpl.html	253
Listing 10.19: In-line template_for use with Tweets	262
Listing 10.20: In-line template_for use with Tweets point to our app	265

Dynamic JavaScript, CSS, and jQuery Essentials **268**

Listing 11.01: injectJSLinksToBottom Snippet	269
Listing 11.02: injectHTMLToBottom Snippet	269
Listing 11.03: injectCSSLinkstoHead Snippet	270
Listing 11.04: injectHTMLToHead Snippet	270
Listing 11.05: injectJSLinksToHead Snippet	270
Listing 11.06: jQuery Chunk	271
Listing 11.07: jQueryTools Chunk	271
Listing 11.08: loads all required JavaScript (placed at Template top)	271
Listing 11.09: Page Output of injectJSLinksToBottom	272
Listing 11.10: googleAnalytics.js using asynchronous page tracking	273
Listing 11.11: site.js to handle various link scenarios (requires jQuery)	273
Listing 11.12: groupsConfig.php (located in Minify's min directory)	274
Listing 11.13: jQuery asking "if the document is ready for it to function"	275
Excerpt 11.14: from site.js	276
Listing 11.15: page content (X)HTML via Wayfinder output	277
Listing 11.16: (jQuery) limitNumberOfItems.js	277
Listing 11.17: Sample Table Data	280
Listing 11.18: Link DataTables jQuery file	281
Listing 11.19: \$tablesort Chunk (Page Specific Configuration attached to id)	281
Listing 11.20: page content of Click Event filters and portfolio items	284

Listing 11.21: inject external JavaScript to page bottom	284
Listing 11.22: jquery.manipulateByClass.js	285
Listing 11.23: \$lightbox Chunk Configuration Container	287
Listing 11.24: LightBox Sample Page Content	288
Listing 11.25: wfKwicksOuter Wayfinder Outer Template:	290
Listing 11.26: wfKwicksItem Wayfinder Row / Item Template	290
Listing 11.27: jquery.kwicksconfiguration.js file	291
Listing 11.28: kwicks.css	292
Listing 11.29: HTML Container hold the video linked to the id="intel"	293
Listing 11.30: flowplayer Implementation	293
Listing 11.31: flowplayer Implementation (W3C compliant code)	293
Listing 11.32: Samples of using Output Filters for content injection	294
Listing 11.32: Grouping Injected Items To a "Single" Placeholder	294
Listing 11.33: Example jQuery Chunk to allow Injection Grouping	294
Listing 11.34 Template Aspect of Injection Grouping	294

AJAX Content Injection **296**

Listing 12.01: excerpt from Power Blogging base_template.xhtml	300
Listing 12.02: Chunk ajaxArchives derived from Listing 12.01	300
Listing 12.03: revised excerpt from Listing 10.02 base_template.xhtml	300
Listing 12.04: ajaxResource (with ID of 38 linked to AJAX)	301
Listing 12.05: getMonths.php	301
Listing 12.06: revised archives <div> from base_template.xhtml	302
Listing 12.07: Chunk ajaxArchives	303
Listing 12.08: ajaxArchives Snippet (wrapper for Archivist)	304
Listing 12.09: ajaxResource content for implementing AJAX wrapper Snippet	304
Listing 12.10: userDetails.xhtml	305
Listing 12.11: userDetails.php Snippet	306

An Overview of Grouping: Category, User and Resource **312**

Access Control List Implementation **328**

Listing 14.01 File Widget - can be of any type delivered by the web server	347
Listing 14.02 HTML Widget - Can contain most any (X)HTML syntax	348

Contexts: Implementing Sites with Multiple Faces **354**

Listing 15.01: Sample Site Template Using Context Variables:	360
Listing 15.02: getContextName Snippet	360

Listing 15.03 Establish Global Top Navigation for each Context	361
Listing 15.04 Establish a generic cross-Context Site Footers	361
Listing 15.05: muti-domain .htaccess file	366
Listing 15.06: robots.txt content for a MODX Revolution Resource / Chunk	367
Listing 15.07: humans.txt content for a MODX Revolution Resource / Chunk	367
Listing 15.08: domainGateway Plugin	368
Listing 15.09: subdomain and domain modified .htaccess file	370
Listing 15.10: domainGateway Plugin with subdomains	371
Listing 15.11 Streamlined Gateway plugin	372
Listing 15.12: Optional .htaccess Category / directory to subdomain redirect	372
Listing 15.13 Language based contexts and Protecting the manager	373
Listing 15.14: contextNames Chunk (may require anonymous user access)	374
Listing 15.15: Login Snippet call using a Chunk to provide Context names	374
Listing 15.16: Example Wayfinder Call Across Contexts	374
Listing 15.17: getContextName Snippet - returns the current Context Name	374
Listing 15.18: getSessionContexts - returns all of the Contexts for this user	374

MODX Revolution API Concepts 380

Listing 16.01: PHP 5 class structure implementing the MODX Revolution API	384
Listing 16.02: Accessing the \$modx object in a PHP Snippet	384
Listing 16.03: PHP Snippet to access the current user object	388
Listing 16.04: PHP Snippet to access users via a runtime parameter - the user id	388
Listing 16.05: PHP Snippet to access any user by other table field(s)	388
Listing 16.06: PHP Snippet to specify a user or default to current user	388
Listing 16.07: the format to access MODX Objects by attributes	388
Listing 16.08: PHP Snippet targeting specific columns in the Object's table	389
Listing 16.09: Example string / array results of Listing 16.08	389
Listing 16.10: PHP Snippet accessing extended fields demonstrating 1:1 relation	389
Listing 16.11: PHP array representation of optional extended fields (ksorted)	390
Listing 16.12: PHP Snippet using toArray();	391
Listing 16.13: PHP array representation of an anonymous user object	391
Listing 16.14: PHP Snippet using toJSON();	392
Listing 16.15: JSON representation of a typical user (note: table order retained)	392
Listing 16.16: PHP Unnecessary traditional-style code	393
Listing 16.17: PHP Snippet to access the profile of an instantiated user object	394
Listing 16.18: PHP array representation of a typical user's profile (ksorted)	395
Listing 16.19: PHP Snippet to access the settings of an instantiated user object	396
Excerpt 16.20: PHP excerpt from moduser.class.php demonstrating getMany()	396
Listing 16.21: Chunk to be used as an output template	397

Listing 16.22 Snippet which processes a collection for a user	397
Listing 16.23: PHP Snippet to display user related information	398
Listing 16.24: Output from running testbedUser on an Anonymous User	402
Listing 16.25: PHP Snippet to return a Resource in a PHP array format	403
Listing 16.26: PHP array representation of a 2.2.4 MODX Revolution Resource	404
Listing 16.27: PHP Snippet to create children under a Parent with ID of 2	405
Listing 16.28: Shorter PHP array providing identical functionality to full version	406
Listing 16.29: PHP Snippet to quickly allow users to select their gender	407
Listing 16.30: PHP Snippet a single function to create 5 types of MODX Objects	409
Listing 16.31: Original PHP Snippet	411
Listing 16.32: Same Snippet from the MODX Revolution core/cache/snippets	411
Listing 16.33: Cache Representation of a Resource in MODX 2.2.1	412
Listing 16.34: excerpt demonstrating non cached content in Revolution 2.1.3	417
Listing 16.35: PHP Function Snippet - function_displayColumns	418
Listing 16.36: excerpt from another PHP Snippet returning an array	418

xPDO: Foundations For Development **420**

Listing 17.01: /core/model/schema/modx.mysql.schema.xml base structure	424
Listing 17.02: model declaration excerpt from modx.mysql.schema.xml	425
Listing 17.03: excerpt of assorted object declarations	426
Listing 17.04: a single field from the modResource Object	427
Excerpt 17.05: of a mutual aggregate one-to-many relation (User to Resource)	428
Excerpt 17.06: of a composite relation which will be deleted with the user	428
Listing 17.07: statelookup xPDO schema file	429
Listing 17.08: PHP Snippet to create xPDO Models - requires \$packageName	431
Listing 17.09: PHP Snippet to create Database tables from xPDO Model	432
Listing 17.10: Success response from \$manager->createObjectContainer()	433
Listing 17.11: statelookup.snippet.php	434
Listing 17.12: excerpt from statelookup xPDO schema file	434
Listing 17.13: populateStateTable by specifying object attribute = array value	435
Listing 17.14: populateStateTable with array keys matching table names	436
Listing 17.15: discjockey primary object schema - building a foundation	441
Listing 17.16: typical xPDO object declaration	445
Listing 17.17: xPDO composite cascading relationship of an Event object	447
Listing 17.18: xPDO aggregate loose relationship	448
Listing 17.19: typical xPDO object declaration	449
Excerpt 17.20: from discjockey schema - displaying relationship ownership	450

xPDO: Third-Party Applications **452**

Listing 18.01: initializeModel PHP Snippet	456
Listing 18.02: initiateModel Snippet Call	456
Listing 18.03: discjockey.class.php (in its earliest rendition)	457
Listing 18.04: Simple discjockey instantiation test	458
Listing 18.04: importGenreCategories PHP Snippet - New Data	459
Listing 18.05 importSongs Snippet to import songs from a PHP array	463
Listing 18.06 importPlaylists Snippet to import old playlists from PHP array	464
Listing 18.07 discjockey.mysql.schema.xml	471
Listing 18.08 discjockey.class.php	476
Listing 18.09 djplaylist.class.php	489
Listing 18.10 djsong.class.php implemented to provide helper functions	491
Listing 18.11 discjockey.ajax.php	494
Listing 18.12 createPlaylist.php (presumably a Manager user creation)	497
Listing 18.13 discjockey.snippet.php	500
Listing 18.14 dj_jquery.main.js	502
Listing 18.15 dj_jquery.createplaylist.js	503
Listing 18.16 discjockey.template.xhtml	508
Listing 18.17 mainpage-events.xhtml	509
Listing 18.18 createPlaylists.xhtml	510
Listing 18.19 dj_filterby.chunk.tpl	513
Listing 18.20 dj_playlistrow.chunk.tpl	513
Listing 18.21 dj_playlistsection.chunk.tpl	513
Listing 18.22 dj_rowboat.listevents.chunk.tpl	514
Listing 18.23 dj_rowboat.listplaylistsbyeventid.chunk.tpl	514
Listing 18.24 dj_rowboat.listsongs.chunk.tpl	514
Listing 18.25 dj_rowboat.pickevent.chunk.tpl	514
Listing 18.26 dj_showsong.chunk.tpl	515
Listing 18.27 dj_showsongplaylist.chunk.tpl	515
Listing 18.28 dj_showsongpopularity.chunk.tpl	515
Listing 18.29 discjockey.css	516

Now What? 532

MODX Revolution Terminology 538

Resource and System Settings 542

Primary Object Cheat Sheets 554

The Document Parser And Cache Mechanism 570

Listing Appendix D01: Excerpt from modResponse from MODX Revolution 2.2.4 572

Internet Caching Technologies 576

Listing Appendix E01: Forcing a page to never be cached using HTTP meta tags 577

Listing Appendix E02: Forcing a page to never be cached using PHP 577

Listing Appendix E03: removing a site from a Squid Proxie Serve 579

Note: All chapters are intentionally listed whether or not they contain code listings.

Index

Symbols

` 107
 \$modx->setDebug(true) 448
 .htaccess Optimizations 208

A

Access Wizard 311
 addMany() 448
 Articles 234
 Feedburner 253
 feed.rss 253
 Templates 235
 Template Variables 246
 Auto-tag 79. *See* Template Variables

B

Backticks 107

C

Cache 129
 cache_default 546
 cache_disabled 546
 Categories. *See* Manager: Categories
 Category 304
 Check Box 81. *See* Template Variables
 Chunks 58
 Caching 58
 Creating 58
 nesting 62
 With Snippets 69
 Coding Advice 114
 Coding Expectations 116
 Components Menu 23

D

Date 82. *See* Template Variables
 Designers 11

Developers 10
 Document Parser 14, 107, 398, 517, 543

E

Email 83

F

Feedburner 253
 File 84
 filters 159
 Benefits 160
 Compare Operators 162
 Conditional Overrides 161
 Custom Filter 171
 Date Formatting 163, 164
 defined 160
 Element Filters 164
 list 163
 add 165
 ago 163, 164
 and 165
 cat 166
 cdata 166
 cssToHead 163
 date 163, 164
 decr 165
 decrement 165
 default 166
 div 165
 divide 165
 eg 162
 el 162
 ellipsis 166
 else 161
 !empty 167
 empty 166
 eq 162
 equalorgreaterthan 162
 equals 162
 equalto 162
 equaltoorlessthan 162
 esc 166
 escape 166

fuzzydate 163, 164
ge 162
greaterthan 162
greaterthanorequalto 162
gt 162
gte 162
hide 161
htmlent 167
htmlentities 167
htmlToBottom 163
htmlToHead 163
if 162
ifempty 166
ifnotempty 167
incr 165
increment 165
input 162
is 162
isempty 166
isequal 162
isequalto 162
isgreaterthan 162
isgt 162
isgte 162
islessthan 162
isloggedin 170
islowerthan 162
islt 162
islte 162
ismember 170
isnot 162
isnotempty 167
isnotloggedin 170
isnt 162
jsToBottom 163
jsToHead 163
lcase 167
le 162
len 167
length 167
lessthan 162
lessthanorequalto 162
limit 167
lowercase 167
lowerthan 162
lt 162
lte 162
math 165
md5 168
memberof 170
mo 170
mod 165
modulus 165
mpy 165
multiply 165
ne 162
neq 162
nl2br 168
notags 168
notempty 167
notequals 162
notequalto 162
or 165
replace 168
reverse 168
select 164
show 161
strip 168
stripString 169
strip_tags 168
striptags 168
stripTags 168
strlen 167
strev 168
strtolower 167
strtotime 163, 164
strtoupper 169
subtract 165
tag 164
then 161
toPlaceholder 163
ucase 169
ucfirst 169
ucwords 169
uppercase 169
urldecode 169
urlencode 169
userinfo 170

- wordwrap 169
- wordwrapcut 169
- Logic 165
- Math Operations 165
- String Manipulation 166
- Targeting 161
- Form Customization 95
- Friendly URLs
 - <base href> 198
 - Site Templates 198
 - [[++site_url]] 198
- fromArray() 374
- fromJSON() 374
- FURLs. *See* Friendly URLs

G

- get() 371
- getOne() 376
- getService() 436
- getSettings() 378

H

- Hidden 86
- humans.txt 213

I

- Image 87
- Import Resources 30
 - modDocument 30
 - modStaticResource 30
- include() 436
- include_once() 436

J

- JavaScript 207
 - javascriptToFooter Snippet 207
 - Page Bottom 207
- jQuery 270
 - DataTables 275
 - Manipulating Hyperlinks 270

L

- loadClass() 436

M

- Manager. *See* Site Menu
 - Categories 304
 - Components Menu 23
 - Home Button 21
 - Reports Menu 31
 - Error Log 35
 - Manager Actions 35
 - Site Schedule 34
 - System info 36
 - Security Menu 24
 - Access Controls 25
 - Flush All Sessions 29
 - Flush Permissions 29
 - Form Customization 26
 - Manage Users 24
 - Resource Groups 26
 - Site Menu 21
 - Logout 23
 - New Document 23
 - Preview 21
 - Remove Locks 22
 - Search 22
 - Static Resource 23
 - Symlink 23
 - Weblink 23
 - Support menu 43
 - System Menu 37
 - Actions 42
 - Contexts 37
 - Lexicon Management 41
 - Namespaces 42
 - Package Management 40
 - Considerations 39
 - installation 37
 - uninstall 39
 - System Settings 40
 - Tools Menu 29
 - Import HTML 33
 - Import Resources 30
 - Property Sets 33
 - User menu 42

Media Source 85, 87
 Media Sources 31, 331
 meta keywords 28
 Minify 199
 Groups 205
 Implement 202
 installing 199
 Testing 203

MODX Revolution API 263
 \$modx->regClientCSS(\$externalStyleSheet) 264
 \$modx->regClientHTMLBlock(\$src) 263
 \$modx->regClientScript(\$javascript) 263
 \$modx->regClientStartupHTMLBlock(\$src) 264
 \$modx->regClientStartupScript(\$javascript) 264

P

Plugins 98
 AntiSpam 154
 Property Sets 33
 Attach Element 141
 Benefits 142
 Implementing 145
 Snippet 140

R

Radio Option 89
 Reports Menu 34
 Error Log 35
 Manager Actions 35
 Site Schedule 34
 System info 36
 require() 436
 require_once() 436
 Resource List 90
 Rich Text Editor 90
 robots.txt 213

S

schema. *See* xPDO schema
 Search. *See* Site Menu
 Security Menu 24
 Access Controls 25
 Flush All Sessions 29

Flush Permissions 29
 Form Customization 26
 Manage Users 24
 Resource Groups 26
 Site Map 212
 Site Menu 21
 Clear Cache 21
 Logout 23
 New Document 23
 Remove Locks 22
 Static Resource 23
 Symlink 23
 View 21
 Weblink 23
 Snippets 97, 379, 451
 advice 115
 Archivist 219
 AJAX 294
 Articles 234
 Backticks 107
 Basic Structure 117
 Benefits Of Chunk Templates 144
 Cached 129
 Coding Advice. *See* Coding Advice
 Coding Expectations. *See* Coding Expectations
 Creating 127
 Creating Templates 143
 Crumbz 219
 Danger 99, 365
 Dangerous Side 99, 365
 Expectations 116
 flow chart 114
 FormIt 219, 368
 FormItCountryOptions 368
 FormItStateOptions 368
 getPage 219, 230
 getResourceField 220, 223
 getResources 80, 219
 getResourcesTag 226
 getUserID 220, 230, 238
 GoogleSiteMap 212
 Implementing A Templated Snippet 145
 injectCSSLinkstoHead 264
 injectHTMLToBottom 263

- injectHTMLToHead 264
- injectJSLinksToBottom 263
- injectJSLinksToHead 264
- Login 220
- Login Package 320
- Package 100
- PackMan 39, 433
- Parser Considerations. *See* Document Parser
- Pasting 128
- PHP Classes 149
- Quip 219
- QuipCount 227
- QuipReply 222
- Rampart 220
- randomImages 220
- Rowboat 451
- RowBoat 400
- SimpleSearch 192
- Simplest Solution 125
- streamlining 147
- Structure 117
- taglister 80, 81
- tagLister 219, 223
- UltimateParent 220, 223
- userDetails 300
- Verify 100, 365
- Wayfinder 100, 220
 - Horizontal 108
 - predefined navigation 113
 - Vertical 110
 - Wiki-type 111
- Support menu 43
- System Menu 37
 - Actions 42
 - Content Types 41
 - Contexts 42
 - Lexicon Management 41
 - Namespaces 42
 - Package Management 37
 - Considerations 39
 - Download Extras 38
 - Removing Packages 39
 - Upgrading Packages 39

T

- Tag 91
- tagCloud 79
- Template Variables
 - Auto-tag 79
 - Check Box 81
 - Date 82
 - Email 83
 - File 84
 - Media Source 85
 - Hidden 86
 - Image 87
 - Media Source 87
 - Listbox 88
 - Number 89
 - Radio Option 89
 - Columns 89
 - Resource List 90
 - Rich Text Editor 90
 - Tag 91
 - Textarea 92
- toArray() 373
- toJSON() 374
- Tools Menu 29
 - Import HTML 33
 - Import Resources 30
 - Media Sources 31
 - Property Sets 33

U

- User Groups
 - Access Wizard 311
- User Roles
 - Access Permissions 312, 326

W

- Wayfinder 100

X

- xPDODriver 404
- xPDOGenerator 404
- xPDOManager 404

- xPDO Relationships 410
 - Aggregate 410
 - Composite 410
 - Inheritance 410
- xPDO schema 406
 - field 406, 409
 - attributes 409
 - dbtype 409
 - default 409
 - index 409
 - indexgrp 409
 - key 409
 - null 409
 - phptype 409
 - precision 409
 - model 406
 - baseClass 407
 - defaultEngine 407
 - package 407
 - phpdoc-package 407
 - phpdoc-subpackage 407
 - platform 407
 - tablePrefix 407
 - version 407
 - object 406, 408
 - class 408
 - extends 408
 - table 408