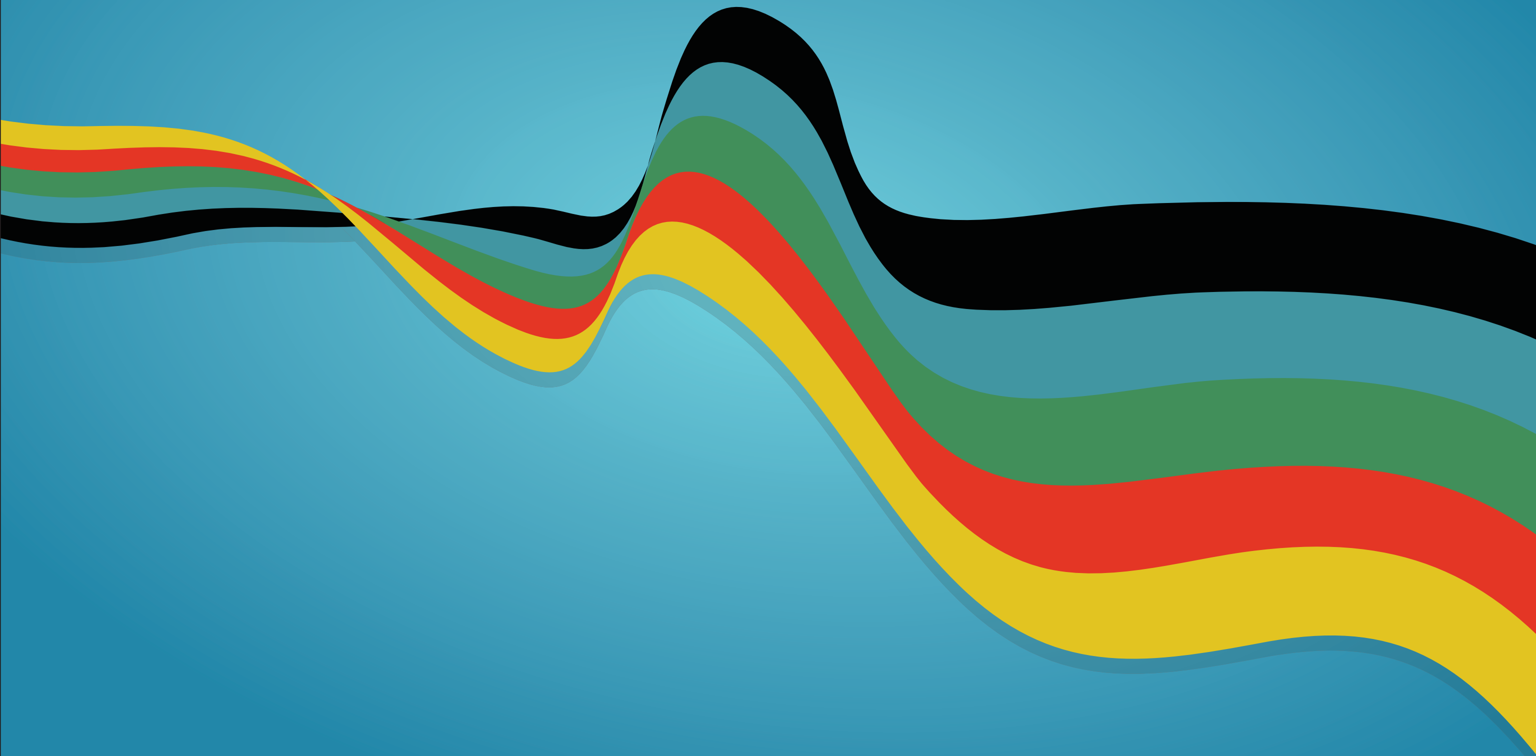
MODX: The Official Guide

Building dynamic websites with the
MODX content management platform.



Bob Ray

Bob Ray

MODX: The Official Guide

"No components should EVER alter core tables or files. That's all I'm gonna say on that one."
Jason Coward (OpenGeek) March 28, 2010

"The mantra we must remember in Open Source is that you get what you give."
Shaun McCormick (splittingred) July 21, 2010

"Unequivocably, there's no way anyone can now claim the documentation for MODX isn't up to snuff."
Ryan Thrash (rthrash) August 3, 2011

Copyright © 2011 Bob Ray. All rights reserved

Publisher: MODX Press
Editor: Haven Stephens
Technical Editor: Shaun McCormick
Cover Design: äkta
Interior Design: Ryan Thrash
Production Coordinator: Michael Hickey
Indexer: Bob Ray

ISBN: 978-0-9836194-0-6

MODX, LLC
25 Highland Park Village, Ste 100-413
Dallas, TX 75205-2789
http://modx.com/press/

42 11 04 10 9 8 7 6 5 3 2 1

Table of Contents

1 |How MODX Works
The MODX Database . . . 2
MODX vs. Third-Party Components . . . 2
The Back End vs. the Front End . . . 5
Basic Building Blocks . . . 6
How MODX Delivers a Web Page. . . . 21
Transport Packages . . . 22
Other MODX Concepts . . . 23
Summary . . . 29
MODX Evolution Notes . . . 30

2 | Installing MODX
Installing the Current Stable Release . . . 36
Installing Add-on Components . . . 52
Friendly URLs, Friendly Aliases . . . 55
Installing MODX on Your Local Machine . . . 62
Porting an Existing Site to MODX . . . 89
Summary . . . 94
MODX Evolution Notes . . . 95

3 | The MODX Manager
The Resource, Element, and File Trees . . . 99
The Top Menu . . . 109
Summary . . . 129
MODX Evolution Notes . . . 130

4 | Resources
Documents . . . 136
Weblinks . . . 176
Symlinks . . . 177
Static Resources . . . 178
Template Variables and Access Permissions. . . . 179
Other Resource Operations . . . 179
Summary . . . 183
MODX Evolution Notes . . . 184

5 | Templates and Template Variables
Templates. . . . 190
Creating a Template. . . . 190
An Example Template . . . 191
Example Template: A Closer Look . . . 192
Deciding What to Put in a Template . . . 198
Using Tags in a Template . . . 199
Template Variables . . . 199
Summary . . . 226
MODX Evolution Notes . . . 227

6 | Chunks and Properties
Introducing Chunks. . . . 230
Creating a Simple Chunk. . . . 230
Using Chunks with Other Tags . . . 232
Other Uses for Chunks . . . 236
Introducing Properties . . . 247
Sending Properties in the Chunk Tag . . . 249
Default Properties Versus Property Sets. . . . 249
Working with Default Properties . . . 252
How Default Properties are Used . . . 255
Property Sets. . . . 256
Summary . . . 263
MODX Evolution Notes . . . 264

7 | Snippets and Plugins
Working with Snippets . . . 268
Working with Plugins. . . . 289
Summary . . . 305
MODX Evolution Notes . . . 306

8 | Advanced Snippet Operations
MODX get*() Methods . . . 310
Object Methods . . . 326
Working with Resources, Elements, and Users . . . 352
Summary . . . 415
MODX Evolution Notes . . . 416

9 | Site Organization and Contexts
Site Organization . . . 419
Contexts. . . . 424
Summary . . . 434
MODX Evolution Notes . . . 435

10 | MODX Security
Security System Overview . . . 438
Security Elements. . . . 442
Working with MODX Security . . . 454
Controlling Access in the Front End . . . 468
Summary . . . 496
MODX Evolution Notes . . . 497

11 | Customizing the MODX Manager
Altering the Top Menu . . . 502
Form Customization . . . 513
Custom Manager Pages. . . . 523
Summary . . . 533
MODX Evolution Notes . . . 533

12 | Using Common Components
Installing Components and Property Sets. . . . 536
Creating an FAQ Page with EZfaq . . . 539
Creating a Contact Page with SPForm . . . 543
Using Breadcrumbs to Show the Current Path . . . 551
Using Wayfinder to Create a Menu . . . 555
Using getResources to Aggregate Content. . . . 565
Using FormIt to Create and Process Forms . . . 571
Summary . . . 594
MODX Evolution Notes . . . 594

13 | Creating Transport Packages
Transport Package Overview . . . 600
A Simple Transport Package. . . . 605
Summary . . . 625
MODX Evolution Notes . . . 626

Appendix
MODX API. . . . 629
PHP Primer . . . 652
MODX Objects Reference . . . 690
Using SMTP to Send Mail in MODX. . . . 696
MODX System Events . . . 697

Acknowledgments

Countless people have been invaluable in the production of this book. First and foremost, I owe a debt of gratitude to the founders and core developers of MODX: Ryan Thrash (rthrash), Jason Coward (OpenGeek), and Shaun McCormick (splittingred). They have developed a first-class Open Source CMS, and this book would not have been possible without their support and patience. Shaun served as the book's technical editor and deserves special credit for ensuring its accuracy. Haven Stephens, the book's editor, played a critical role in eliminating countless errors and making sure that the explanations in the book were as clear as possible.

I would also like to thank the many users of the MODX Forums, both those whose answers have taught me almost everything I know about MODX and those whose questions led to the creation of this book. In particular, I'd like to thank the following people (MODX Forum usernames in parentheses):

Susan Ottwell (sottwell), Jelle Jager (TobyL), (ganeshXL), David Molliere (davidm), (kongondo), (doze), (ZAP), Zaigham Rana (Zi), Andreas Wettainen (mrhaw), Jay Gilmore (smashingred), Shane Sponagle (dev_cw), (coroico), Kyle Jaebker (kylej), Garry Nutting (garryn), Jeff Whitfield (Bravado), (Everett), (flinx), Mike Reid (pixelchutes), Jared Carlow (jaredc), (Dr. Scotty Delicious), (Carsten), Steve Hamblett (shamblett), (NetProphET), Adam Crownoble (aNoble), (netnoise), (sirlancelot), Aaron Wardle (Onesmarthost), (Photowebmax), Erik Bjorn (Sylvaticus), (snop), (护ademyday), Keith Penton (kp52), (charliez), (bob1000), (Chuck), (therebechips), Dimitri Hilverda (Dimmy), (Soshite), Adrian Lawley (Taff), (sinbad), Bruno Perner (Bruno17), (Breezer), (mmjaeger), (AMDbuilder), (sharkbait), (chinesedream) (cipa), (ChuckTrukk), (Paprikas), David Bunker (bunk58), (samba), (paulp), (Byzantium), (SandersDesign), (virtualgadjo), James Ehly (devtrench), Mark Hamstra (Mark H.), (treigh), (hotdiggity), (ottogal), (anso), (pleth), (lossendae), (mconsidine), (Jul), (xdom), Luca Reghellin, (microcipcip), (Pandy06269), Helen Warner (Muggins), Ivan Salcedo (odeclass), Stefan Moser (stefan), Rob Backus (robb), John Hughes, Zuriel Andrusyshyn (zurie).

My apologies to the many others whose names I've omitted either from lack of space or memory lapse.

I'd also like to thank my cat, Cosmo, who spent many hours each day for over two years curled up next to me as I worked on the book at my computer. His comments on the book were invaluable.

Disclaimer of Warranty (Legal Mumbo Jumbo)

All of the information, instructions, and recommendations in this book are provided on a strictly "as is" basis without any warranty, expressed or implied. In particular, any and all warranties of fitness for use or merchantability are disclaimed. Neither MODX nor Bob Ray personally shall be held responsible for any direct, indirect, incidental, or consequential damages that may result from the use of material in this book or anything any reader does as a result of reading any part of the book or any associated web site. It is up to the reader to determine the suitability of any directions or information presented in the book. Furthermore, it is the responsibility of the reader to determine if any information, instructions, or recommendations given in the book are appropriate based upon his or her particular situation. The author and publisher cannot be held accountable for decisions made based upon material in the book.

MODX is an advanced Open Source Content Management System (CMS). If you are not familiar with the term CMS, think of a word processor for web sites, but on steroids. A CMS will usually provide you with a graphical view of your web site, a database that stores the site's content, and one or more editors for creating web pages and populating them with links, lists, headings, and formatted text.

Every CMS tries to provide a convenient and powerful user interface for creating and maintaining web sites, but to some extent, convenience and power are in opposition to one another. Some CMS platforms, like WordPress, put convenience above power. If you want a standard blog with a ready-made look and feel, WordPress provides a surprisingly friendly and intuitive user interface for creating and maintaining one.

MODX, on the other hand, leans more toward the power side of the equation. The learning curve is a little steeper, but MODX allows you much more freedom in the design of your web site, more powerful tools you can use to make the web site unique, and the ability to customize the CMS itself.

MODX is also a powerful Content Management Framework (CMF). This means that the building blocks of MODX can actually be used to create any number of different CMS platforms. We will cover customizing the MODX user interface in later chapters. For the most part, however, the use of MODX as a CMF is beyond the scope of this book, which is mainly about MODX the CMS.

One guiding principle of MODX is to put as few restrictions on the web developer as possible. MODX puts no limits on the CSS and HTML code you use to build your site, and you can easily add PHP or JavaScript code to meet any need. One illustration of this flexibility is that you can take virtually any existing web site and make a MODX web site out of it that looks exactly the same to a front-end visitor. Trying that with many other CMS platforms will leave you cursing and tearing your hair out.

By the time you've finished reading this book, you should be able to use MODX to create new web sites or to port existing web sites to MODX. It was my goal in writing this book to provide all the information you need to use the basic building blocks of MODX to meet your needs as a web designer/developer.

About This Book

Before we dive into MODX, we need to discuss a few things about this book. In the following sections, we'll look at who the book is for, how it is organized, and how the typography of the book can help you understand what you're looking at.

Audience

This book is for anyone who wants to use MODX for web site design, development, and maintenance. In order to get the most out of this book, you should have a basic familiarity with (X)HTML, CSS, and how the two interact in the rendering of web pages.

You will also see some PHP code in the book. You can do a lot with MODX without knowing PHP, but to get the most out of MODX, you really should have at least some knowledge of PHP. If you don't, never fear. PHP is a relatively easy computer language, and the PHP Primer in this book's Appendix will help you get started.

There are many excellent PHP tutorials on the Web, and lots of expert MODX users knew no PHP when they started out. Often, they created a web site with no PHP, and then learned PHP, a little bit at a time, as they began to make their web site more and more interesting and easier to maintain.

This book is perfectly suitable for MODX beginners but contains plenty of in-depth information for power users as well. If you are brand new to MODX, some parts of the book may go over your head the first time you read them. Over time, however, you will grow more familiar with how MODX works, and they will gradually begin to make sense to you.

At first, I tried to put material for beginners at the beginning of the book and save the more advanced information for later. Because of the way MODX works, however, it just wasn't possible. As a result, you'll see things that may confuse you on the first pass. You should be able to ignore them and create a perfectly good web site using the things you do understand. Later, as things make more sense to you, you'll be able to refine your site to take advantage of MODX's more advanced features. The book is also meant to serve as a reference manual for both beginners and experts.

Organization of The Book

Generally, the book is organized on the basis of MODX objects, such as resources, chunks, snippets, plugins, placeholders, etc. If you are new to MODX, you won't be familiar with these at first. After reading Chapter 1, however, you should have a fair idea of what they are and how they work together. Later in the book, we'll look at each MODX object in depth. Because the objects constantly interact with each other, you'll be learning about them all bit-by-bit as we go along.

Conventions

A number of typographical conventions are used throughout the book to make it easier to understand what you're looking at. There are two main typefaces for content in the book: The regular font (used for this sentence) and the code font. The code font looks like this when it appears in a regular font paragraph: `This is the inline code font`. When it appears in a separate paragraph, it looks like this:

```
This is the code snippet font.
```

The code font is used for all (X)HTML, CSS, and JavaScript code and for all MODX tags and their contents. It is also used for file and directory names and for path names and for text to be entered in a field in the MODX Manager.

MODX internal variables such as system event names, settings, and resource/template variable field names will also be in the code font, but to make them distinguishable, they will be in boldface type wherever they appear:

> **`OnManagerFormSave`**, **`site_start`**, **`pagetitle`**.

Key terms will generally be in the regular font in italics when they first appear. Terms referring to MODX objects (snippets, template variables, chunks, plugins, categories, placeholders, etc.) will be in lowercase when they appear in text except at the beginning of a sentence.

Names of resources or elements that you will be creating and editing yourself, will be in italics in the code font when they first appear, then in the regular font in roman type, usually in CamelCase:

> Create a chunk called *`FooterChunk`*. The FooterChunk will contain code for the page footer.

Terms referring to specific areas of the MODX Manager or to form fields in the Manager will generally be in initial caps and in the regular font:

> Create/Edit Resource panel, Long Title, Alias, Package Manager

Shorthand references to Manager commands that use menus for navigation will be in bold and have arrows between them:

> Go to **Tools → Package Management**

Longer descriptions of Manager actions in the text will use initial caps and quotation marks around buttons and menu choices that you are meant to select:

> Select "System" on the Top Menu, then click on "Package Management."

Field names and tab names used in the MODX Manager will be in initial caps when referring to the specific area of the Manager, but in lowercase when discussing them as general objects. They will be enclosed in quotation marks when they are included in directions to click or select them, but used without the quotation marks in general discussion:

> Click on the "Elements" tab.
>
> Chunks are listed on the Elements tab.
>
> Chunks are classified as elements.

The names of specific resources and elements will generally be in CamelCase in the text. They will usually be in the code font and italics when they first appear (especially in directions to create them):

> Create a resource called *`News`*.
>
> Let's create our *`ShowList`* snippet.

Later references to them will be in the regular font in CamelCase:

> The FooterChunk contains all the elements shown in the footer of the page.

They will be enclosed in quotation marks in instructions to click or select them in the Manager:

> Click on the "ShowList" snippet on the "Elements" tab.

Variable names (including snippet properties) will generally start with a lowercase letter and have the first letter of each following word capitalized. They will be in the code font: `$wordLength`, `$fontColor`, `$textHeight`, though there are some exceptions to this rule. Property names, MODX settings, and placeholder names, for example, are often in all lowercase with an underscore character between words: **`site_start`**, **`blocked_until`**. MODX system event names are in CamelCase and begin with a capital letter. All start with "**`On`**": **`OnManagerFormSave`**. Resource field names are in all lowercase, and a few have underscores while most don't: **`pagetitle`**, **`longtitle`**, **`pub_date`**, **`unpub_date`**.

As mentioned earlier, MODX settings and system event names will be in the code font in bold. For resource field names, the caption used in the Manager will be in initial caps in the regular font (e.g., Page Title, Publish Date). The actual field names used internally by MODX (and by users in tags and snippets) will be in the code font and bold (e.g., **`pagetitle`**, **`pub_date`**).

When a MODX tag or example contains a name that the user will supply, the term will be in italics:

```
[[SnippetName? &property1=`value1` &property2=`value2`]]
```

In the above example, the user is expected to replace all the words in italics. This is important because new users sometimes type in the example exactly as it is written and are surprised when it doesn't work.

I've tried to be as faithful to these conventions as possible, but this is a long book with many complicated references. I hope the reader will forgive the inevitable errors in typography.

Because the page width of this book is limited, some lines of PHP and HTML code that should be on a single line are broken into two or more lines. I tried to break them in a way that doesn't harm the user's understanding of what the code does, but it wasn't always possible. The extra lines of code created will always be indented.

The only time the lines must be combined is for Input Option Values of template variables. Here's an example:

```
Red==Red as a rose||Blue==Blue as the sky||
    Green==Green as the grass
```

The lines above must be entered as a single line, but MODX won't let you use a carriage return for Input Option Values anyway, so the need to combine them should be obvious.

In all other cases that I'm aware of, the code can be entered as written and no harm will be done — the PHP and HTML parsers are very forgiving about formatting — but you should feel free to combine the split lines to fit your coding style. Generally, the only negative effect of the split lines will be on the formatting of some raw HTML code you'll see if you select "View Source" in your browser.

About MODX

In the following sections, we'll look at whether MODX is a good choice for you, learn a little bit about the history of MODX, and discuss the various versions of MODX. We'll also take a look at the architecture and key features of MODX.

Is MODX for you?

MODX is designed to be useful to everyone from beginning web designers to advanced Content Management System developers, but it is somewhat difficult for the former. If you are looking for a system that will do everything you want out-of-the-box and let you produce a finished web site without getting your hands dirty, MODX is probably not for you. For a standard blog, for example, WordPress might be a better choice. You install it, select a theme (or use the default one), and begin entering your posts.

Where MODX shines is in cases where you want a unique web site with active pages that you design yourself. Its flexibility and power are unequalled in the CMS world. If you can describe what you want your web site to do, the odds are good that you can do it in MODX.

My first MODX web site, for example, was a site I designed for a local political organization. The site had the usual pages: a page describing the organization's officers, a volunteer page, a page for the local elected officials with links to their web sites, a page containing the organization's bylaws, a page where users could download issues of the organization's newsletter, etc. The site also had a front page with excerpts from news posts, each with a link to the full post.

Next, I added a series of photo galleries for various events and put a random photo from one of the galleries under the menu on the front page that served as a link to the photo's gallery. Then came a fundraising thermometer that showed the current level of contributions.

I set up one user as the photo editor who could easily create new galleries and upload photos in the front end. I set up a news editor who could write new articles that would automatically show up in the news section and be excerpted on the front page. I set up a newsletter editor who could upload new newsletters in .PDF format. Each editor had access to only his or her own section of the site.

I learned MODX, created the site, and wrote simple instructions for each user in a surprisingly short time — less than a week. The site has been rock-solid ever since with almost no intervention on my part.

On one of my next MODX sites, the client wanted a page with seasonal photos and a discount coupon that changed automatically four times a year, with the seasons. I was able to add this in MODX using a simple custom snippet in about an hour. Doing this on most other CMS platforms would have taken much, much longer and would have been extremely frustrating.

In order to use MODX at all, you'll need some basic familiarity with CSS and (X)HTML. It's best if you have created a few web sites before diving into MODX. You don't need to know PHP to use MODX, but it definitely helps. Many MODX users end up learning PHP a little bit at a time as they attempt to get the most out of the CMS.

If you don't know any PHP at all, don't be put off by the examples of PHP code in the book. They are there for advanced users, and you can still do quite a lot with MODX without knowing PHP. The odds are that you will pick it up as you develop your MODX skills and can then go back and read over any sections that confused you at first. There is a PHP Primer in the Appendix at the end of the book that will help get you started.

A Brief History of MODX

In 2004, Ryan Thrash and Raymond Irving started working on what was later to become MODX. The initial MODX project was a combination of the DocVars add-on for the Etomite CMS and Raymond's web-user add-on. There was resistance to the MODX project at Etomite, and the two eventually left that community to work on MODX as a separate CMS.

MODX 0.9.0 began as a fork of Etomite but has developed into an independent CMS platform, and as of the release of MODX 2.0 Revolution, all references to Etomite are gone from the MODX code.

In May of 2005, Jason Coward joined the project, and in 2007, Raymond Irving left (amicably). Jason became the principle architect of the MODX core code. In 2008, Shaun McCormick joined the project and is now primarily responsible for the MODX Manager interface. MODX has an extremely creative and responsive user community, and countless MODX users have contributed to the development process over the last few years.

In 2009, MODX Version 0.9.6 was supplanted by two new versions. The 0.9.6 designation was misleading — by that time, MODX was a very robust and mature product. As of this writing, MODX exists in two forms: MODX 1.x.x Evolution and MODX 2.x.x Revolution.

Versions of MODX

In 2009, both MODX 1.0.0 Evolution and MODX 2.0.0 Revolution were released. If you visit the MODX Forums, you may see them referred to as "Evo" and "Revo" for short.

There are many similarities between the two, especially from the point of view of the user. Under the hood, the two are quite different. We'll look at some of the similarities and differences in the following sections. For now, we'll say that the main components of MODX discussed in this book (chunks, snippets, templates, template variables, plugins, resources, settings, and tags) exist in both versions and play the same role in each.

The MODX Manager user interface has been redesigned for MODX Revolution but is similar enough that most users have little or no trouble making the transition from one to the other.

This book is written primarily from the perspective of MODX Revolution since that is the future of MODX. That said, most of the content in the book will apply equally well to either version, and there is a section at the end of each chapter describing how the content of that chapter applies in MODX Evolution.

MODX Evolution

MODX 1.0.0 Evolution is a mature, stable release of the original MODX codebase with a set of standard add-ons for things like menus, photo galleries, user management, and content aggregation. It provides a fairly seamless upgrade for all legacy MODX sites.

The version numbers of earlier versions of MODX (e.g., 0.9.2, 0.9.6) are somewhat misleading. The version numbers imply that they were beta versions. In fact, they were (and still are) very mature and used in thousands of production web sites.

MODX Evolution has been around longer, and as of this writing, is more familiar to most MODX users than Revolution and has more available add-ons. Evolution also has a smaller footprint and takes somewhat less memory to install and use. Revolution, however, is more robust and more secure. Revolution is a significant technical advance compared to Evolution and represents the future of MODX.

MODX Revolution

MODX 2.0.0 Revolution (which appeared briefly as version 0.9.7) is a complete re-write of the MODX core code and Manager with a new tag syntax and many revolutionary features (hence the name). The Manager is somewhat easier to navigate and has a number of extremely convenient features like drag-and-drop editing and the ability to create new elements and resources and clear the cache in pop-up windows without leaving your current work. Revolution also introduces convenient grids for editing language strings, system settings, and snippet properties.

By far the most "revolutionary" feature in Revolution from the average user's perspective is the Package Management system. In Revolution, add-ons can be downloaded from the

MODX repository and installed with a few clicks in the Package Manager. In Evolution, you have to download a .ZIP file for each add-on, unzip it, and cut and paste code into MODX elements and resources you create yourself — a much more time-consuming and error-prone process.

There is also much for the developer/web programmer to love in MODX Revolution. There is a completely new database API based on xPDO, which provides an efficient and easy-to-use way to read, create, and update all MODX objects. It also allows you to easily pull data from multiple MODX tables (with one query) in a single, elegant line of code.

Which Version Should I Use?

If you are upgrading an existing site that you don't work on very often, MODX Evolution may be a better choice for you. Evolution is also a better choice if you need to install and run your web site with limited memory. As I write this, Revolution needs at least 32M of memory to install (although there are plans to remedy this), so if your web host restricts you to 8M or 16M of memory and you are not allowed to change that, Evolution is the better choice. Most web hosts have higher limits, and many allow you to modify the amount of available memory.

Many of the concepts described in this book are relevant for either version. If you will be working through the examples in the book, however, MODX Revolution is really your only choice. All of the tags and much of the code presented in the book's examples will only work as written in MODX Revolution.

In other situations, the key to making the choice is the available add-ons. At this writing, some of the add-ons available for MODX Evolution have not been ported to Revolution — though most of the important ones have. By the time you read this, there will certainly be more of them available in Revolution, but some less-popular third-party components may still not be available. Many of the standard MODX snippets and plugins have been replaced by faster and better versions for Revolution, and some have also been integrated into the MODX Manager. You should assess your needs and examine the available components (ask in the MODX Forums, if necessary) before making your decision.

Another consideration is the permissions system. MODX Revolution offers much more sophisticated and find-grained control over what users can do and see in the MODX Manager. The price of that control, however, is a steeper learning curve. Once the permissions for Revolution are set correctly, you can usually forget about them, but the initial process can be somewhat frustrating and time-consuming. If you will be the only user of the site, Revolution's default settings will be fine for you. If you have several users with different access rights and the permission system is relatively simple, Evolution may be a better choice. If you need to have users who will belong to more than one user groups and want them to have different capabilities for each group's documents, then Revolution is the better choice.

Assuming that the add-ons you need are available in Revolution (or you are capable of developing or adapting them yourself) and the Revolution permission system meets your needs, I would recommend it for everyone, including beginning MODX users. Revolution is a robust CMS platform and has many features that make it a superior choice.

Architecture and Key Features

The overriding principle in the design of MODX is freedom for the user. To the best of the designers' abilities, MODX puts no restrictions on what you can put in your web site. MODX users are free to use any (X)HTML, CSS, PHP, or JavaScript code to meet their needs.

You could port most existing web sites to MODX simply by pasting the code from each page into a MODX template, moving any PHP code into snippets, and creating an empty document for each page that uses that page's template. Doing this would be a horrible misuse of MODX's power, but the fact that it's possible shows how few restrictions MODX places on the user.

If you are not a power user/developer, you probably won't care what's under the hood in MODX. For those who do care, here is some information about the structure and design of MODX. Don't be put off if some of the following material goes over your head. You can create very impressive MODX web sites without knowing any of it.

xPDO

MODX's data handling is based on xPDO. Created by MODX core developer Jason Coward, xPDO is a PHP-based object-relational bridge that is designed to provide object-oriented access to a variety of underlying database platforms such as MySQL, SQLite, and PostgreSQL.

Both lightweight and robust, xPDO now requires PHP 5.1.1 or higher. It is significantly faster now that it no longer has to support PHP 4. It allows simple file-based result-set caching and custom cache implementations that can be optimized for particular needs. It also supports JSON caching for optimizing Ajax applications.

Using xPDO allows advanced MODX web site developers to easily integrate custom databases into a MODX install using built-in MODX methods. It also provides a relatively simple API that developers can use to obtain information from the MODX database.

At this writing, xPDO supports MySQL and Microsoft SQL databases, with more to follow.

User Interface

The MODX Manager is an Ajax-based user interface implemented with Smarty and ExtJS. In the Manager, users can create and edit content as well as perform a wide variety of administrative tasks like user management, publication scheduling, generating reports, installing add-on components, and adding user-created custom snippets and plugins. The MODX Manager is designed to be productive and intuitive.

In MODX Revolution, (unlike Evolution) many Manager tasks can be performed in pop-up windows, so users can create and update resources and elements and clear the MODX cache without leaving their current location in the Manager. Drag-and-drop functionality also increases productivity and reduces potential errors in the Manager.

Because the Manager is a web-based application, users can manage their MODX site from anywhere they have web access using no tools other than a web browser. MODX supports a wide variety of web browsers.

The Manager can be easily customized, and it can look very different for different users. Users can be restricted to certain areas of the Manager, and the areas they see can be customized for their particular needs and abilities. Multiple editors can be easily plugged into the Manager for use in creating and editing content as well as for managing various kinds of code.

Design Philosophy

MODX is designed around several basic principles. These include security, robustness, speed and efficiency, object-oriented design, core independence, and fine-grained cache control and user management.

Security

MODX is designed to be as secure as possible. All user input is analyzed for potential hazards, and no PHP code is allowed or executed in most MODX resources and elements. PHP code can only appear in snippets and plugins, and its execution is carefully regulated. All PHP code in the system can be contained in the database or in the MODX core, which can be shielded from web access. The user security system is also extremely tight and prevents unauthorized users from accessing sensitive areas of the Manager.

Robustness

MODX is designed from the core outward to be solid and fault-tolerant. Changes are evaluated and tested to ensure that they cause no problems in the operation of the system. MODX sites are generally rock-solid with virtually no downtime due to errors in MODX.

Speed and Efficiency

The developers of MODX are constantly profiling and refactoring MODX objects and their methods to make MODX as fast and efficient as possible. The caching process ensures that frequently accessed content and code are cached for maximum performance.

MODX is also designed to be a lightweight platform. Users install only the components they need for a particular site, and components are designed to be flexible and powerful enough that users can meet their needs with a small number of add-on components.

Object-oriented Design

MODX Revolution is object-oriented from the ground up. Every MODX object is a true PHP object, and all use of objects is through their properties and methods. Chunks, snippets, templates, and template variables, for example, are all subclasses of the `modElement` class. The MODX parser makes full use of the object methods available through the basic class. This makes the MODX core both efficient and easily extendable.

MODX also makes it easy to adhere to the principles of the model-view-controller (MVC) design philosophy by facilitating the separation of content, logic, and presentation.

Core Independence

Another principle is the complete separation of the MODX core from the content of the site. The core can be located in a directory that is not accessible from the Web for security. One MODX core can also support more than one web site.

Central to the separation of the core is a wide variety of built-in ways to extend and tap into the core processes. There is a well-developed API for accessing core methods and a comprehensive list of system events that fire at various points in the operation of the core.

Users can create PHP snippets that use the API to get information from the core and the database. They can also create PHP plugins that listen for the appropriate system event and step in to alter the processing at that point. The main purpose of these capabilities is to give the user the ability to do anything he or she can think of in MODX without touching the core code. When users upgrade to a new version of MODX, none of their custom work has to be done over again. Advanced developers can even create a custom parser class to override or extend the behavior of the MODX parser. Custom elements and resources are also a possibility for users with particular needs.

Fine-grained Cache Control

Another key principle is granular control of the MODX cache. Parts of the core are cached as needed, and various parts of each front-end page can easily be designated as cached or uncached by the user. In MODX Revolution, any element or resource can be designated as cached or uncached. As a result, nothing is cached but content and code that needs to be rapidly accessible. Because of MODX's xPDO base, database result-sets are also cached on an as-needed basis. In addition, there are a number of system settings that control whether certain MODX objects are cached and for how long.

Fine-Grained User Management

Like the caching process, user management is extremely granular in MODX Revolution. Since resources such as documents are in a hierarchical structure on the Resources tab, users can be limited to particular branches of the Resource tree shown on the Resources tab. In addition, individual resources can be placed in resource groups and individual elements can be placed in categories. Specific resource groups and categories can be made accessible only to members of specific user groups. Users can have designated roles that restrict them to specific actions in the Manager such as viewing, saving, editing, publishing, and unpublishing documents. Finally, the MODX Manager can be customized so that each user sees a completely different version of the Manager.

How MODX Works

Using a Content Management System (CMS) is quite a bit different from just creating web pages and linking them together to make a site. A CMS takes a while to learn (and one as flexible and powerful as MODX takes a little longer). The time you spend getting familiar with MODX, however, will pay off many times over. It will make maintaining web sites easier and faster. It will also allow you to easily create web-site features that would be extremely difficult and time-consuming to produce without a CMS backing you up.

While many CMS platforms make life easy by severely restricting what you can do (with structured templates, built-in and add-on components that are hard to modify, hard-coded entity and directory names, etc.), MODX does just the opposite. Letting users do whatever they want is a primary goal of the MODX development team. The flexibility and freedom of MODX come with a price, however. Because there are fewer rules, it's harder to know how to solve any given problem. It's also a little easier to shoot yourself in the foot.

In this chapter, we'll touch on the various parts of MODX and how they work together to let you manage web sites. In later chapters, we'll cover the mechanics of using the MODX Manager, and the details of the various parts of MODX such as web resources, content elements, and files. This chapter is just to get your feet wet, introduce these elements, and give you an overall sense of how MODX does its job.

The MODX Database

If you are coming to MODX from another ***Content Management System*** (***CMS***) such as Drupal or Joomla!, you probably already have some sense of how such systems work. If, instead, you're used to working in straight (X)HTML and CSS, you may not be aware of how a CMS like MODX stores and presents the site's content. New users of MODX sometimes install the MODX sample site then look (unsuccessfully) for the HTML page files.

The reason they don't find the HTML files is that MODX, like most other CMS platforms, stores the page content in a MySQL or Microsoft SQL database (although future versions of MODX will be able to use a variety of other database platforms and will allow you to store web resources and content elements in files rather than the database if you wish).

When a user visits one of your pages, MODX goes to the MODX database, gets the information it needs for the requested page, assembles it, and sends it off to the user's browser for display. This all happens behind the scenes. Many MODX users create sites without ever dealing with the database or the process of saving and retrieving the information. To really make MODX sing, however, you'll eventually want to learn about how some of the behind-the-scenes processes work. For now, we'll just say that you create ***web resources*** (most of which will be ***documents***) in the ***MODX Manager*** (also called the ***back end***) and save them. When a user visits a particular web page at your site, MODX retrieves the appropriate content and sends it off to the user's browser for display. A lot can happen to that content before the user sees it, but we'll get to that later. First, let's look at the differences between MODX and its add-on components and between the back end and the front end of a MODX site.

MODX vs. Third-Party Components

In MODX Revolution, there has been a very determined effort to separate ***MODX*** itself from ***components*** contributed by others that extend or add to the functionality of MODX. A number of ***third-party components*** (also called ***add-ons***, ***3PC***s, ***extras***, ***add-on components***, or just ***components***) were distributed and installed with earlier versions of MODX as a convenience to users. They added functions like custom menus, aggregated content display, text editors, photo galleries, etc. People often assumed that these were part of MODX itself, and when problems occurred, people complained about bugs in MODX. The MODX core developers would have to explain, over and over, that these were not part of MODX.

With MODX Revolution and future versions, only MODX is installed to begin with. It includes just three directories: `/core`, `/connectors`, and `/manager` (a `setup/` directory is present at first but is usually removed after a successful install). Users are also free to move and rename those three directories if they wish or even to install a different or customized manager to administer their sites.

Once you've completed the MODX base install, you can then easily add whatever third-party components meet your needs. This helps keep your MODX site from being loaded down with components you don't need. The download and install process for components is all done in the Manager, and it allows you to browse through the available components, download them, and install them with a few mouse clicks.

If you find you need more components later, new components become available, or new versions of existing components are released, it's a simple matter to browse the package repository and install them. Another advantage of this separation of the base install from third-party components is that the base install can remain "pure." It is unaffected by the installation and upgrading of components, and conversely, updating the base install to a new version of MODX is unlikely to affect any installed components.

Although the base MODX install contains no third-party components, by the time you read this, there may be additional distribution versions of MODX available that include various third-party components tailored to a specific purpose, such as a blog site, a photo gallery site, etc.

MODX Itself

The heart of MODX is in the `/core` directory (although you are free to rename and/or relocate that directory during the install, this is not recommended for new users). The `/core` directory contains the essential parts of the MODX content management engine. You could create a working web site with the tools available in the initial install, but it would have no dynamic menus, no WYSIWYG editor, no content aggregators — none of the things that really make MODX worthwhile for users. In order to gain those features, users need to add third-party components.

MODX Third-Party Components

The official term for anything added to the base MODX install is ***third-party component***. "Third-party component" is kind of a mouthful, so they are often referred to as ***3PCs***, ***extras***, or just ***components*** for short. At this writing, "components" is the more common term, but "extras" is gaining ground. Components are usually installed with a few mouse clicks in the Package Management section of the MODX Manager.

Components are divided into two groups: add-ons and core extensions. ***Add-ons*** do not modify or extend the MODX core but provide extra functionality for MODX. ***Core extensions*** (sometimes referred to as just "***extensions***") actually change or extend the MODX Core.

An extension might, for example, replace the MODX user class with a class that contains the same methods and member variables but adds some new ones (e.g., the user's astrological sign or income level). Another way to think about add-ons and extensions is that add-ons work with MODX and extensions change MODX. Most of the components you might install are add-ons. MODX is designed to make use of a wide variety of add-ons that can use a combination of ***chunks***, ***snippets***, and ***plugins*** (more on these later in this chapter) to do almost anything you want without altering MODX itself.

For most MODX users, the difference between add-ons and core extensions is not important. Users install components that meet their needs and don't really care about what goes on under the hood. For advanced MODX developers, however, the difference is an important one.

Even beginning users of MODX should know that components of all kinds are not officially part of MODX. Third-party components are contributed by volunteers of varying abilities. Some are carefully constructed and highly robust while others are less so. You should also be aware of the term "Sponsored Extras." ***Sponsored Extras*** are third-party components that have been created, edited, or reviewed by the MODX core programmers to assure that they adhere to MODX coding standards and will not interfere with the MODX core.

The Back End vs. the Front End

MODX is divided into two basic parts: the back end and the front end. The ***back end*** is another name for the MODX Manager. The ***front end*** is what visitors to your site see in their browsers.

The Back End

The ***MODX Manager*** (or ***back end***) is where you do the main work of building and managing your site. In the ***Manager***, you can create and edit content. You can also perform a whole array of administrative tasks such as creating users and controlling what those users can do and see at the site. You can create administrative users who can share the workload with you. You can also create and edit any of the various MODX objects such as resources (including documents), templates, template variables, snippets, chunks, plugins, lexicons, namespaces, categories, property sets, system settings, etc.

This list of MODX objects is often daunting for newcomers to MODX, but each of them has a useful part to play in the development of your site, and together, they'll let you do things with your site that you never thought possible. They'll also make your life as a web developer much easier and more productive (trust me). We'll cover each of them in depth, and by the time you finish the book, you should be comfortable with all the ones you need for your site.

The Front End

The ***front end*** is the part that visitors to your site see. There may be parts of your site that only logged-in users can see, but after logging in, they're still in the front end of the site. Depending on how you set things up, front-end users may still be able to add or edit site content and perform specific administrative tasks, but the key difference is that they are not in the ***MODX Manager*** — they are in the ***front end*** of the site. Because they are not in the Manager, what they can do is strictly limited (unless you create and implement code that bypasses the restrictions). This protects the security of your site. It also lets users contribute without learning the Manager interface.

If you have naïve users who need to perform administrative tasks but would be intimidated by the MODX Manager, you have two options. You can customize the MODX Manager to simplify and/or hide the intimidating parts, or you can install or create add-on components that let them perform their administrative tasks in the front end of the site.

Basic Building Blocks

The basic building blocks of a MODX site are web resources, content elements, and files. ***Web resources*** include documents, weblinks, symlinks, and static resources. ***Content elements*** include templates, template variables, chunks, snippets, and plugins. ***Files*** are just plain old files. We'll look at how to manage these things in more depth later in the book, but first, let's take a brief look now at what they are.

Web Resources

Web resources (usually referred to as just ***resources***) are the easiest to define of the MODX objects: They're simply things that can be accessed via a URL. The most commonly used resources, by far, in MODX are documents. In fact, the document is the default resource type. Many MODX users never create a symlink, weblink, or static resource, but everyone creates lots of documents. The most convenient way to create a resource is to right-click somewhere in the Resource tree (on the Resources tab at the left side of the Manager) and hover over Create. This will open a flyout with four choices:

- Create a Document Here
- Create a Weblink Here
- Create a Symlink Here
- Create a Static Resource Here

Clicking on one of the choices will open the Create/Edit Resource panel. On the right side, you'll see the Create/Edit Resource panel showing some of the various resource fields for the resource and several tabs at the top for accessing others. We'll discuss these in more detail later.

You can also create new resources by clicking on "Site" in the Top Menu and selecting the type of resource you want to create. There is also a small icon at the top of the Resource tree that will do the same thing. Doing this will create the resource at the top level of the Resource tree (under the web context icon). Usually, you'll want to put new resources at a particular place in the Resource tree, so right-clicking on the resource or folder that you want as the parent of your new resource is often a better method.

Documents

Before getting into the details of documents, let's take a moment to look at an important distinction that trips up some new MODX users: the difference between a ***document*** and a ***web page***. In MODX, a ***web page*** is what the user sees in the browser window at any given time when visiting your site. The web page is rendered by the browser, which bases the rendering on the information sent to it by MODX. That web page may involve any and all of the basic building blocks we listed above, although the visitors won't know that because all they see is the final rendering.

A ***document***, on the other hand, is a specific MODX object created in the MODX Manager. If you have used a word-processing program such as Microsoft Word, you've already created documents. Those documents had a title, some content, a creation date, an author, etc. Documents in MODX are very similar but are adapted for use in web pages. For example, they have more than one title. There is a short title (Title), a longer title often used as a page heading (Long Title), a title for use in menus (Menu Title), and a title for use in URLs that link to the document (Alias).

These characteristics of a MODX document are called "resource fields" (formerly "***document attributes***"), and they also include a Template, a Publish and/or Unpublish date, a Description, a Summary, etc. Some of these resource fields may be blank, but the document still has them.

There is another distinction we should mention. Many of the resource fields are known by two names. One is the actual name of the field in the MODX database. This is the name that must be used in code or in MODX tags (more on those later). The other is the general name for the field, usually taken from the Create/Edit Resource panel in the Manager where you actually fill in the values for the resource fields. One field, for example, is referred to on the Create/Edit Resource panel as "Summary," but it sets the **`introtext`** field in the MODX database. When we're talking about the general name for the field, we'll use the regular font and capitalize the first letter of each word in the field name (Summary, Menu Title, Long Title). When we're talking about the specific field in the database, we'll use bold type, lowercase, and a different font (**`introtext`**, **`menutitle`**, **`longtitle`**). There are no capital letters in any field name. This difference is not that important now, but will be critical when we talk about setting these fields in later chapters.

In the book, we'll often refer to documents as resources (which they are). The document is the default type of resource and the most often used, but it is not the only one — weblinks, symlinks, and static resources are also resources. So all documents are resources, but not all resources are documents.

When users visit a web page at your site, the URL they are visiting is associated with a single document. The web page does more than just display the document's content, however. It will usually also show the document's content embedded in the template associated with that document (we'll discuss templates in the section below). It may also show the document's Title or some of the other resource fields. If the document is not currently published, the browser may not show it at all.

To complicate things slightly, a web page may be associated in various ways with more than one document. It might show a document that contains the summary fields of a number of other documents with links that will send the viewer to a full version of each document. A blog web page in MODX, for example, might show several different blog posts, each listing the author and date. Each of those posts is the **`content`** field of a separate document, and the author and date come from the **`createdby`** and **`createdon`** resource fields of that particular document. The summary presented for each item normally comes from the **`introtext`** resource field.

Given the complex relationships possible here, you can see why it's important to understand that web pages and documents are very different things in MODX. When we talk about documents in the book, we're always referring to the MODX document object created in the Manager, not to a web page viewed by a site visitor. We'll always refer to a single page displayed in a browser as a "web page."

Weblinks, Symlinks, and Static Resources

A ***weblink*** is a MODX Resource that contains a URL (also called a ***link***) in its **`content`** field. With a weblink, the URL can be to a page at the MODX site or any other site on the Web.

In MODX, a ***link*** can be expressed as a full URL like this:

```
http://yoursite.com/home.html
```

If, instead, it is a link to a page on your own MODX site, it can be expressed as a ***link tag*** (e.g., `[[~1]]`). If your Home page is resource 1 on your site (in other words, its Resource ID number is 1), MODX will replace the link tag with a full URL that will take users to your Home page.

You should use link tags whenever you can because they will take you to the same page even if the title of the page changes or it is moved to another location in the Resource tree.

A ***symlink*** is also a MODX resource that refers to another document, but the Symlink field contains only the Resource ID of the document being referred to, and it must refer to a page on the MODX site.

The main use of weblinks and symlinks is to serve as links in a MODX menu. This is a somewhat advanced topic, and we'll cover it in more detail later in the book.

Static resources are resources that contain a file path in their **`content`** field. Most MODX pages have dynamic content because they contain elements, such as snippets and chunks, which may change over time. That means that the content of the page will be created on the fly before being seen by the site visitor. When you have content that you know won't change unless you edit it yourself, however, you can save it as a file and use a ***static resource*** to display it. The page will display faster and will put less strain on MODX and the Database.

Static Resources may also contain links to files you want to deliver to the visitor such as .PDF files or .DOC files for display or download. You can even use PHP code to create dynamic .PDF files, for example, that the user can view or download by following a link to the static resource. This also allows you to control access to the files so that some users can reach them and others can't.

Content Elements

Content elements (usually referred to as just "elements") in MODX are just what their name suggests: they are MODX objects that create, control, format, organize, and/or contain content. The phrase "content element" is somewhat flexible, and as MODX evolves, new elements are created, and sometimes things that were formerly resource fields may become elements. Sometimes elements can lose that status and be reclassified as something else. The best working definition of content elements is: "things that appear in the Element tree on the Elements tab at the left side of the MODX Manager."

The section below discusses the elements that existed at the time this book was written: templates, template variables, chunks, snippets, plugins, and categories. You should be aware, however, that a few of them may not be found in the Element tree in your version of MODX.

Templates

Most businesses have a standard letterhead they use when sending letters. It gives every letter the same look and saves time since the letter writer doesn't have to type the company name, address, and telephone/fax numbers on each letter. A MODX ***template*** performs the same function. It contains the basic information that will appear on a number of different web pages. It will usually contain the basic (X)HTML for a web page (`DOCTYPE`, `<head>`, `<body>`, etc.) as well as the banner at the top of the page and the footer at the bottom. It may also contain other MODX objects, as we'll see in the section below.

Like the company letterhead, the template will probably contain a header and footer (possibly with images) that will appear on every page that uses the template. Unlike a letterhead, however, a MODX template can also contain other MODX objects. It might contain a document's Title (and/or Long Title) and a menu. These will change on every page, but they will always appear in the same place and with the same styling for every page that uses that template.

When MODX receives a request for a web page from a browser, it finds the appropriate resource (usually a document), then checks to see what template is attached to that resource. The template contains (X)HTML code interspersed with MODX tags. MODX retrieves the template from the database and begins filling it with the appropriate content (by replacing the MODX tags) as the first step in preparing the web page to be returned to the browser.

Beginning MODX users sometimes create more templates than they need because they don't fully understand how flexible MODX templates can be. Some MODX sites (but not all) can get by with a single template even though there are things that will appear on some pages but not others. We'll look at templates in more detail later in the book.

Template Variables

"Template variable" is a confusing phrase for some users because it hasn't always been used consistently. Sometimes, the resource fields we discussed above (Title, Menu Title, Alias, etc.) have been referred to as template variables, even in the official MODX documentation. This is incorrect, however. Those things should be called "resource fields." ***Template variables*** are always created by a user who needs additional resource fields.

Template variables (often called ***TV***s for short) provide a way of extending the list of resource fields. Imagine that you let registered users create some content on your site. When a page created by a user is displayed, you could show the document's Title and the date it was posted by using the document's **`pagetitle`** and **`createdon`** resource fields. What if you also wanted to show the author's astrological sign? You need another resource field. Luckily, you can easily add a TV called "Sign" to hold that information (we'll talk about exactly how to do this in a later chapter). Now, when someone edits the document in the MODX Manager, they'll see an extra field labeled "Sign" with a blank space for entering the user's astrological sign.

Template Variables are very flexible and can be used to hold many things such as text, images, (X)HTML code, dates, directory contents, etc. In fact, we could easily present our "Sign" TV as a drop-down list of the twelve astrological signs or as twelve radio buttons. This is quite easy to do in MODX, and we'll discuss how later in the book.

At this point, you might be wondering how the content of the "Sign" TV gets on the web page. The easiest way is just to place a ***resource tag*** in the template associated with that document (we'll discuss tags in more detail later in this chapter). A resource tag looks like this: `[[*`*`Field/TvName`*`]]`. Both template variables and resource fields can be displayed using resource tags. The following code in the template would display the title, date, and sign, on the page:

```
<p>Post Title: [[*pagetitle]]<br />
Created on: [[*createdon]]<br />
Author's sign: [[*sign]]</p>
```

Template variables are a powerful and underused part of MODX. They can hold strings of text, such as a person's sign, but they can also hold other objects you might want to put on a web page such as drop-down lists, date pickers, and even the content from other documents on the site. We'll talk about how to create them and use them in more detail later in the book.

Chunks

A MODX ***chunk*** is just a piece of reusable content. The key thing to remember about chunks is that they can't contain raw PHP code (that's the job of snippets). Many chunks are just bits of content embedded in (X)HTML code that you want to appear on multiple pages. The banner at the top of a page, the footer at the bottom, and the menu are often contained in chunks. That way, if you edit the chunk, it changes things on every page.

Special-purpose chunks called Tpl chunks are used as mini-templates. They usually contain text and placeholder tags (more on those later) and are used format output. Here is a simple example that could be used to display the name of the current president and vice-president of a company:

```
<h3>Officers</h3>
<p>President: [[+president]]</p>
<p>Vice-president: [[+vice-president]]</p>
```

Typically, code in a snippet would set the values for the two placeholder tags above so that the appropriate names would appear when the chunk is rendered.

Tpl chunks are also used for web forms and as mini-templates for the output of standard add-on components. Tpl chunks are sometimes referred to as "templates," but this is incorrect.

The content of a chunk makes it into a page in various ways. The most common method for displaying chunks is to put a ***chunk tag*** in the template or the **`content`** field of a document. A chunk tag looks like this: `[[$ChunkName]]`. We'll see some other methods of injecting chunk content later in the book, and we'll discuss tags in general a little later in this chapter.

Snippets

A ***snippet*** is simply a piece of executable PHP code. If your users can submit content (such as blog comments) that immediately appears on the site, what happens if a malicious user puts PHP code that erases your hard disk in a comment? In an unprotected site, the first time someone visits the page that shows that comment, it's goodbye site. In MODX, however, nothing happens at all because PHP code in a page's content is stripped out. To be executed, the code must be in a snippet. What appears in the page content, then, is the ***snippet tag*** (sometimes called a ***snippet call***). A snippet tag looks like this:

```
[[SnippetName]]
```

or

```
[[SnippetName? &firstName=`John` &lastName=`Doe`]]
```

In the second example above **`&firstName`** and **`&lastName`** are just pieces of information called ***snippet properties*** (formerly called parameters) that we want to send to the snippet being called. Whatever is returned or printed by the snippet will replace the snippet tag in the page sent to the browser for display.

Important: Note that the values of the snippet properties are enclosed in back-ticks, not single-quotes. This allows you to use single- and double-quotes in your snippet properties (e.g., `&lastName=`O'Connor``). The back-tick is under the ~ at the upper-left on most keyboards.

Using single-quotes, double-quotes, or nothing at all around snippet properties is the most common error made by new users of MODX. If you do that, your snippet simply won't work. Other common errors are forgetting the question mark after the snippet name or the ampersand before every snippet property, and misspelling the name of the snippet or a snippet property (the names are case-sensitive, so type carefully). One last common snippet error to check for is that a Rich Text Editor like TinyMCE may be changing the ampersands in your snippet tags to the "`&amp;`" entity every time you save your work. To fix this, click on the "HTML" button in the editor and convert `&amp;` or `&amp;&amp;` to a single ampersand character inside snippet tags.

Although we often put spaces around equal signs in PHP code to improve readability, we don't do it with snippet tags because it can confuse the parser in some older versions of MODX Evolution.

We'll discuss snippets and snippet tags in detail later in the book. If you don't know PHP, you won't be writing or editing snippets, but you'll still need to understand a little bit about how snippets do their jobs and how to work with snippet properties.

Many beginning MODX users know no PHP at all and have no intention of learning it. Some of them are able to create very sophisticated MODX sites without using PHP code. Many others, however, learn PHP gradually as they continue to use MODX, and some become quite good at it and begin writing their own snippets and plugins. As we said earlier, PHP is not a difficult language, and there is a short MODX PHP Primer at the end of the book to get you started. One of the great things about MODX is that no matter how much you know, there's always more to learn.

Plugins

Plugins are kind of an anomaly among the MODX elements. They often alter content, but you don't display them on a web page using tags. Plugins are used to interrupt the processing of the MODX engine and add custom functions.

There are a number of hooks into the MODX engine (called system events) that let you perform just about any kind of operation at key points in the processing. As MODX goes about its job, it periodically "fires" one of a number of system events. There are events that fire just before a page is rendered in the browser, just after a user attempts to log in, and just before a document or user is saved in the Manager, for example. Each system event has a name, and MODX plugins can listen for a particular system event and act when it fires.

A ***plugin***, then, is just a bit of PHP code that listens for a particular ***system event***. When that event "fires," the plugin's code executes. For example, a plugin can give you access to a document's content just before it is rendered as a web page by having it listen for the `OnWebPagePrerender` event. You could translate the document's content, emphasize key words, strip out HTML comments, turn some items into hyperlinks, etc.

You can also use a plugin to process documents before they're saved in the Manager, process a user's information during the process of logging in, or do any number of other transformations. Plugins are written in PHP code, and you can do anything with them that you can write code for.

The beauty of plugins is that they allow you to put custom processing in place without hacking the MODX core. That means your work will be unaffected by upgrading to newer versions of MODX. Because of the many built-in ways of expanding and extending MODX, you can do almost anything you can think of without touching the MODX core code.

Categories

Categories are basically just labels that you can apply to elements to help you organize them and control access to them. When you create or edit a snippet, plugin, chunk, template, or template variable, you can assign it to an existing category or create a new category for it. When that element is shown in the Element tree in the Manager, it will appear below its category name.

In MODX Revolution, you can assign elements to categories by dragging and dropping them onto the category folder at the bottom of the Element tree.

Note that categorized elements will not show in the Category section of the tree (even though that's where you dragged them). Instead, they'll show under a category folder in their respective sections. For example, a snippet added to the category MyCategory will appear in the Element tree in a MyCategory folder that appears in the Snippet section of the tree.

Resources and files don't have categories. Resources in the Resource tree are organized into a hierarchy under their respective parents and files are organized by the physical directories in which they are stored.

You can have as many categories as you like, and you can have categories within categories. You might, for example, have a category called *MyChunks* to keep the chunks you create separate from the ones used by various MODX components. If you create a lot of chunks, you might have subcategories under MyChunks.

Once elements are placed in a category, you can hide them or control what users can do with them by creating Element Category Access ACL entries in the Manager (see Chapter 10 for more details on MODX security).

If you don't use categories to control access to elements, they are just there for your convenience in organizing your elements and making them easier for you to find when you want to edit them. Categories are completely optional. Some people don't use them at all. Others, especially those with very complex sites, use lots of them. As with so many things in MODX, it's up to you.

Other Elements

The following elements are not found in the Element tree, but they serve to store and present content on your site. They include placeholders, links, settings, tags, and files.

Placeholders, Links, and Settings

Placeholders are not shown in the Element tree in the Manager, but they qualify as elements because they hold content and can be displayed using resource tags. The reason they don't show in the Element tree is that their value is always set in snippets or plugins rather than being entered in the Manager. A snippet or plugin sets a placeholder by using the following bit of PHP code:

```
$modx->setPlaceholder('placeholderName','value');
```

Once the placeholder has been set, its value will replace any instance of the corresponding ***placeholder tag***:

```
[[+placeholderName]]
```

Many MODX snippets set placeholders. You can display the values set for those placeholders with placeholder tags. The tags can go in a resource's content field, in a chunk, or in a template variable or template. If you have a snippet (let's call it *SetAuthor*), for example, that sets a placeholder called `author_name` to the name of the author of the current document, you could put the following snippet tag in your template: `[[SetAuthor]]`. The snippet tag would produce no output because your snippet neither prints anything nor returns anything — it just sets the placeholder. Anywhere in the content of your document, then, you could insert a placeholder that would be replaced by the author's name:

```
The author of this document is [[+author_name]].
```

Note that for the `[[+author_name]]` tag to work, the snippet tag must come before it in the code so that the placeholder will be set.

MODX does set a couple of placeholders for you on every page request that you can use to display the ID or username of the currently logged-in user (if any).

```
ID of the current user: [[+modx.user.id]]
Username of the current user: [[+modx.user.username]]
```

Links are a MODX shorthand for a URL to a page at the site and are based on one of the resource fields — the ***Resource ID***. Most of the time, links are used to refer to documents. The Resource ID of a document is sometimes called the document ID or document identifier, but the correct term is "Resource ID." The ID is shown in parentheses in the Resource tree in the Manager following the resource's name. You can place a link pretty much anywhere by using the ***link tag***:

```
[[~##]]
```

In the code above, you would replace ## with the Resource ID of the resource (document) you want a link to. That may sound complicated, but it's really pretty simple. Let's look at an example. Suppose you have a page about aardvarks that has a Resource ID of 12. You would see the number 12 in parentheses next to the name of the document in the Resource tree (on the Resources tab) in the Manager. Anywhere you would put **`http://mysite.com/aardvark.html`**, you could put `[[~12]]` instead. The name of the document might change, and its location on your site might change, but `[[~12]]` will always provide a reliable link to it because its Resource ID will never change. You should always use a link tag when creating a link to any page at your site.

Like placeholders, settings don't show in the Element tree but can be displayed using tags (***setting tags***, to be precise). A ***setting*** is essentially a variable whose value is available across the MODX site. They include system settings, context settings, and user settings (more on these later in this chapter). All are displayed with a setting tag:

```
[[++setting_name]]
```

You can use setting tags freely to display or use the values of any system, context, or user settings. Some system settings are critical to the operation of MODX, however, so don't change a system setting unless you're sure of what it does.

Tags

We've already seen some MODX ***tags*** in this chapter, but let's look at them a little more closely. All MODX tags are replaced by the object they represent. Here is a quick list showing what each tag will be replaced by:

- **Chunk tag** — Contents of the chunk
- **Resource tag** — Value of the resource field or template variable it represents
- **Snippet tag** — Output or return value of the PHP code in the snippet
- **Link tag** — The URL of the resource it refers to
- **Placeholder tag** — The value of the placeholder (often set in a snippet or plugin)
- **Setting tag** — The value of the context, user, or system setting
- **Language tag** — A particular language string from the current lexicon

In MODX Revolution and beyond, all tags begin with `[[` and end with `]]`. We've discussed most of the tags, but Table 1-1 gives a summary of the new tag style used in MODX Revolution and later and the tags used in earlier versions of MODX:

Table 1-1: Old and New Tag Styles

Tag Type	MODX Evolution (old)	MODX Revolution
Resource	`[*`*`ResourceField/TvName`*`*]`	`[[*`*`ResourceField/TvName`*`]]`
Chunk	`{{`*`ChunkName`*`}}`	`[[$`*`ChunkName`*`]]`
Snippet	`[[`*`SnippetName`*`]]`	`[[`*`SnippetName`*`]]`
Setting	`[(`*`SettingName`*`)]`	`[[++`*`SettingName`*`]]`
Placeholder	`[+`*`PlaceholderName`*`+]`	`[[+`*`PlaceholderName`*`]]`
Link	`[~`*`ResourceId`*`~]`	`[[~`*`ResourceId`*`]]`
Language	No Tag	`[[%`*`LanguageStringKey`*`]]`

Note that those symbols after the opening braces (`$`, `+`, `++`, `*`, `~`, and `%`) are called ***tokens*** in MODX. They tell MODX what kind of tag is being processed.

New users of MODX often don't realize that tags can be used almost anywhere in MODX and that you can nest them. A snippet tag can contain a chunk tag as a property, and the chunk could contain resource tags that might contain link tags.

Here's an example of a setting tag nested inside a link tag. It will display a link to the site's Home page:

```
[[~[[++site_start]]]]
```

Let's break that down and look first at the outer tag, which is a link tag.

`[[~X]]` is a link tag where *X* is the Resource ID of the document you want to link to. It will be replaced by the URL associated with that document. In other words, the tag `[[~12]]` will be replaced by the URL of the document whose Resource ID is 12.

There is nothing special about the Home page of a MODX site. The **`site_start`** system setting can point to any page on the site and MODX treats the page like any other when it is rendered.

`[[++site_start]]` is a setting tag that will be replaced by the Resource ID of your site's Home page. So if the Resource ID of your site's Home resource is 1 (and it often is), this tag will be replaced by the number 1.

So `[[~[[++site_start]]]]` will be replaced by `[[~1]]` which will be replaced by something like **`http://`***`yoursite.com`***`/home.html`**. For an actual link to your Home page, you'd want something like the following (X)HTML code:

```
<a href="[[~[[++site_start]]]]">Home</a>
```

When creating nested tags in MODX, always count the number of left and right brackets. The two numbers must be equal, and in MODX Revolution, each must be an even number.

For the more technically minded, here is a specification of the MODX tag syntax:

`[[` — opening brackets.

`!` — (optional) do-not-cache flag.

elementToken — token identifying the element type if it's not a snippet:

> **no token** — snippet.
> **`$`** — chunk.
> **`*`** — resource field/template variable (TV).
> **`+`** — placeholder.
> **`++`** — setting.
> **`~`** — link.
> **`%`** — language.

`elementName` — name of element (e.g., Wayfinder, MyChunk).

`@propertysetName` — (optional) property set identifier.

`:`*`modifierName=`*`` ` ``*`modifierData`*`` ` ``:…` — (optional) one or more output modifiers.

`?` — indicates that properties are coming; required if there are properties.

**`&`*`propertyName=`*`` ` ``*`propertyValue`*`` ` `` &…`** — properties prefixed with `&`

`]]` — closing brackets.

Here is a complex tag example with every possible type of tag element:

```
[[!getResources@propset1:default=`No resources found.`? &parents=`1`
&sortby=`RAND()`]]
```

The tag above tells MODX to process the getResources snippet uncached specifying `propset1` to override the default properties. The `:default` output modifier will produce a message if getResources returns nothing. Two properties, **`&parents`** and **`&sortby`** will be sent to the snippet and will override any other properties with the same names.

Files

In MODX, ***files*** are simply files that exist within the MODX site. As we've seen in the sections above, much of the content and format of a MODX web site is contained in documents, chunks, templates, and other MODX objects that exist in the MODX database. Files still play a role, however. You will probably have some files of your own. You might have image files, CSS files, JavaScript files, or other files that are used in your site. In addition, many MODX components have **`readme.txt`** files, class files, CSS files, example files, etc.

The files used by components are generally found in one of two places. Files that need to be web-accessible are located below this directory (where *`component_name`* is the name of the component):

```
assets/components/component_name
```

Files that don't need to be web-accessible are located below this directory:

```
core/components/component_name
```

The files for the Wayfinder snippet, for example, will be in these two directories:

```
assets/components/wayfinder/
core/components/wayfinder/
```

In MODX Revolution, the **`/core`** directory containing all of the critical MODX PHP files can be located outside of the **`public_html`** directory so that its files can't be accessed directly from the Web. This makes MODX much more secure. A good security principle for files is that files containing executable code (e.g., executable PHP files) or sensitive material like Social Security Numbers should go under the relocated **`/core`** directory so they can't be reached directly via the Web.

New users of MODX are often confused about where to put their own files. The official MODX answer is an emphatic, but somewhat unhelpful, "anywhere you want." It is a guiding principle of MODX not to restrict users in any way unless it's absolutely necessary for MODX to function correctly (and MODX is designed so that there are very few restrictions). As long as you know where the files are and you tell MODX where they are, anything goes.

Some users like to put the files in, or just under, the root directory for faster loading in **`/images`**, **`/css`**, **`/js`**, etc. Another good place to put them is somewhere under the MODX **`/assets`** directory (i.e., **`assets/images`**, **`assets/css`**, **`assets/js`**). Properly written third-party components always refer to the assets directory by the `MODX_ASSETS_PATH` setting, which you can set.

The **`/assets`** directory is never used directly by the MODX core. Because of the way the **`/assets`** directory is handled by MODX, you can be confident that it won't be touched when you upgrade your MODX installation to a new version. If all of your personal files are below the **`/assets`** directory, you can be certain that they won't be touched by an upgrade. Placing them there also makes it possible to rename, relocate, or re-upload the other MODX directories without worrying about your own files.

How MODX Delivers a Web Page

Now that you have some sense of the basic building blocks MODX uses to create a web page, let's look briefly at how MODX does the job.

When a user's browser requests a web page from the site, MODX first checks to make sure that the page exists. If it does not, the user is forwarded to the site's `Error` (page not found) page. Next, MODX checks to make sure that there are no security restrictions that would prevent the current user from viewing the page. If there are, the user is forwarded to either the `Error` page, or the site's `Unauthorized` page (depending on the security settings) and processing stops.

Assuming that there are no security restrictions that would prevent the user from seeing the page, MODX retrieves the document associated with that URL and checks to see what template is associated with that document. It then gets the template, and the MODX ***document parser*** goes to work.

The document parser loads the template into its own internal text editor. The template will contain tags identifying the various building blocks. The document parser replaces those tags with the things they represent. For example, a chunk tag will be replaced by the contents of the chunk and a snippet tag will be replaced by the output of the snippet. Once that replacement has been made, there may be some tags left in the page (the chunk might contain some tags, or the snippet might put some tags in the output it returns). The snippet might also pull content from another snippet or chunk or from a file and insert it into the output it returns.

The parser continues replacing tags until there are no more left. If any plugins are listening for the events that fire during this process, they may also alter the page content at various points in the process. In addition, if some of the content is cached, the parser will get the content from the MODX cache instead of the database. When the process is complete and there are no more tags to process, MODX delivers the page to the web browser for display. If the resulting page contains a reference to a file to be included (such as a CSS file, a JavaScript file, or an image file), the browser will handle that appropriately and then, finally, show the page to the user.

Transport Packages

Transport packages are new in MODX Revolution and are one of the best reasons to upgrade to the new version. A ***transport package*** is a .ZIP file that can contain files, MODX objects, and PHP scripts packaged for automatic installation in a MODX site. The most common transport package is used to install a specific third-party component, but transport packages can contain core extensions, updates, templates, simple collections of files, or even an entire MODX site.

In earlier versions of MODX, in order to install a third-party component, you needed to download the component, unzip the files in the appropriate directory at the MODX site, create elements, and cut-and-paste code from the component files into the elements. Sometimes the process was not very well-documented and it was easy to make mistakes that would make the component unusable. With Transport packages, you go to Package Management in the MODX Manager, search for available packages, download them, and click on "Install." All the work is done for you and done without errors.

Even better, Transport packages can interact with the user during the install process, so you can often set configurations and preferences during the install. The SPForm package, for example, creates a simple, spam-proof contact form for your site. During the install, you're asked for your return email address and you specify whether you want the sample `Contact` page installed. If you say yes, the package automatically installs all the necessary resources, elements, and files. It creates a `Contact Us` page and a `Thank You` page. At the end of the installation, you have a working contact form on your site that automatically appears in your menu.

The downside of Transport packages is that they are more work for developers who have to learn how to create them. This means that it may be a while before some of the existing components for MODX are available as Transport packages. Fortunately, the old installation method will still work as long as the component has been rewritten to work with MODX Revolution.

If you try to install a MODX add-on component in Revolution the old-fashioned way by downloading and cutting and pasting, make sure it has been rewritten to work with MODX Revolution.

Other MODX Concepts

In this section, we'll look at some other MODX objects: namespaces, the lexicon, topics, settings, workspaces, properties, and the MODX cache. With the exception of properties, they are less likely to be used by beginning users than the topics we discussed earlier in the chapter, but it's still useful to know a little bit about them.

Namespaces

A ***namespace*** is like a category tag for MODX components. It allows component developers to identify the various MODX objects that go with their component, such as topics in the MODX lexicon, resources, elements, and transport packages. The namespace also contains the path where the component can be found. For low-level development, the most important use of namespaces is with the MODX lexicon. We'll explain that when we discuss the lexicon in the next section.

If you are not developing components for distribution to other users, you probably won't need to know anything about namespaces. We'll discuss namespaces in more detail in later chapters.

The MODX Lexicon, Namespaces, and Topics

In MODX, the ***lexicon*** is just a dictionary of words and phrases that can be used in the MODX Manager or in web pages at your site. Entries in the lexicon (called ***language strings***) are read from ***language files***. Each language string has a key (used to retrieve it), a namespace, a language, and a topic. That way, third-party components can load the language strings they need into the lexicon and their keys can be used to retrieve them.

The term "language" is somewhat misleading since what constitutes a language in MODX is actually based on a particular culture rather than a language. That is, you could specify British English or American English as the language to be loaded into the lexicon. You could even have a "beginning user" language and an "advanced user" language if you wanted to. In some cases, you will see the word "culture" instead of "language." The system setting that controls the lexicon strings used in the front end of a MODX site, for example, is called **`culture_key`**.

As you might imagine, the lexicon is a great tool for internationalizing your site. You can change the language used in the MODX manager just by changing the lexicon used without directly altering the content of any panels, forms, or menus.

When you are using some part of the Manager, each word or phrase on the screen has its own lexicon entry. They all come from the language specified by the `manager_language` setting in the System Settings grid. If you change this system setting and the appropriate language is available, all the text in the Manager will change to the new language (you may have to log out and log back in to have this take effect).

Note that using the lexicon doesn't mean that MODX will translate your site for you. It will just replace language tags on a web page or Manager page with the appropriate content from whatever language is currently loaded in the lexicon.

A namespace, when used with the lexicon, is just a group of language strings used for a specific purpose. All the strings used with a particular add-on component, for example, will usually be in their own namespace. A ***topic*** is just a subdivision of a namespace.

The MODX ***core*** namespace contains all the language strings used in the Manager. The core namespace has a different topic for each part of the Manager (e.g., a snippet topic, a chunk topic, etc.). Language strings from the snippet topic, for example, are used as prompts when you are creating or editing a snippet in the Manager.

MODX Revolution only loads the topics it needs for a specific page to make things faster and more efficient. When you are editing a chunk in the Manager, for example, MODX loads the chunk topic and knows that it only has to search that topic for the language strings on the page.

If you don't like the wording of any string printed in the Manager or by a component, you can go to the lexicon grid in the Lexicon Management section of the Manager and change it. Your changes will survive upgrades to MODX or to any add-on components.

We'll discuss the lexicon in more detail later in the book.

System Settings

A ***system setting*** is a site-wide variable available across a particular MODX site. Sometimes a system setting contains a string of text such as the site name (contained in the **`site_name`** system setting). Others contain a value that controls some aspect of MODX such as the number of times a user can enter incorrect login information before being blocked (**`failed_login_attempts`**) or for how many minutes the user will be blocked after exceeding the number of failed attempts (**`blocked_minutes`**). System settings are divided into areas for your convenience in finding them.

Every system setting has a description that explains what it does and what the acceptable values are, and you can search for system settings by any key word in their names or descriptions. All system settings can be changed in the System Setting grid in the Manager. As you might guess, if you have no idea what a system setting does, it's best not to modify it. System settings are overridden by context settings and context settings are overridden by user settings — more on those later.

If you go to **System → System Settings** in the Manager Top Menu, you will see the System Settings grid. You can also edit any of the system settings there. Sometimes, you want to show a system setting on a web page. To do that, you simply put a setting tag on the page where you want the information to appear:

```
[[++site_name]]
```

The setting tag above would be replaced by the value of the **`site_name`** setting in the System Settings grid.

Context Settings and Workspaces

Contexts are a new concept in MODX Revolution. Many MODX users will not deal with contexts at all. Contexts are somewhat difficult to explain but are very useful in certain situations. The main thing to understand about contexts is that a context has its own resources and settings. If you want to host two different sites within one MODX site that share the same MODX database, you could create a new context for the second site. The second site could use a different language, for example, but would still be able to use the MODX core and any installed components. It could also have the same users but with different privileges and settings (e.g., the same user could have a different personal Home page in each context).

When you are using the MODX Manager, you are in the `mgr` context. The default front-end context is called `web` in MODX Revolution, and you can see it at the top of the Resource tree. All resources in that context will appear below `web`. For many sites, this is the only context you need.

If you are using an early release of MODX Revolution, you probably won't see workspaces at all. In later releases, each ***workspace*** will provide an area where you can work on the settings, lexicons, components, and resources for a particular context. MODX Revolution has a "core workspace," but since there is only one, you probably won't know it's there.

To modify or create context settings for a particular context, go to **System → Contexts**, then either click on "Create New" or right-click on an existing context and select "Update Context," then click on the "Context Settings" tab. Any context settings you create here will override any system settings with the same key (name). Creating new contexts complicates things in many ways, so be sure you really need a new context before creating one.

User Settings

User settings are just system settings that only apply to a particular user. When you create a new user at your site, that user has no user settings. All the site's system settings apply to that user (unless overridden by context settings). If you add user settings for that user that have the same names as system settings, those user settings will override both system settings and context settings for that user alone. To create user settings for a particular user, go to **Security → Manage Users**, right-click on a user and select "Update User," then click on the "Settings" tab.

Suppose that the Manager Language system setting (**`manager_language`**) is set to *`en`* in the System Settings grid. All the strings used in the Manager will be in English. If you have a user that speaks German, though, you can create a user setting for that user called **`manager_language`** and set it to *`de`* so that user will see nothing but German in the Manager. The user setting will override the system setting.

A user could also have a particular Home page on the site. If you create a **`site_start`** user setting and set its value to the Resource ID of a particular page on the site, when a user is logged in, any page that contains a MODX tag with a link to the Home page of the site will show a link to that user's particular Home page:

```
[[~[[++site_start]]]]
```

User Setting names don't have to match any of the system settings. You can add any settings you need. You could add user settings that stored a user's height and weight, for example, and they could be displayed on any page that the user visited at the site using a standard setting tag. For that particular example, though, it is more likely that you would put that information in an extended field of the user profile.

Properties

Most of the elements described earlier in this chapter can have properties. ***Properties*** are a lot like the settings we described in the three previous sections. They are a collection of variables each with a name (technically called the ***key***), a value, and a description. Like settings, they can be created and edited in a grid in the Manager. Properties are most often used with snippets, but are also used with plugins, chunks, templates, and template variables.

This might make properties sound redundant, but there is an important difference between properties and settings. Settings are available to every page in the site. Properties, in contrast, are attached to individual MODX elements (chunks, snippets, plugins, templates, and template variables), so their scope is limited to the element they are attached to. This makes them a more efficient choice when you want a value available in a single element or a group of elements but not across the whole site.

When properties are attached to a snippet or plugin, the property values can contain information used in the element. When properties are attached to another element, the values are available in that element via placeholder tags.

Properties exist in three forms. They can be defined in an element tag, in the element's default properties, or in a property set. In all three cases, the properties serve the same function: to provide a set of key/value pairs that can be used with specific elements.

When sent in a tag, properties are always preceded by an ampersand character and their values are surrounded with back-ticks. The following example shows two properties sent in a snippet tag:

```
[[SnippetName? &color=`red` &size=`large`]]
```

In the example above, the value of the `color` property is set to "`red`" and the value of the `size` property is set to "`large`." Many standard MODX snippets use properties to provide information the snippet needs to do its job and to control the actions of the snippet.

A snippet like Wayfinder that produces a menu, for example, might have a property that determines whether the menu shows unpublished resources. That property might look like this in the snippet tag:

```
&showUnpublished=`1`
```

In addition to being sent in tags, properties can be created and edited in the **Tools → Property Sets** section of the Manager. They can also be created and edited on the Properties tab of any element they are attached to. The second method is usually a much better choice if you intend to use the property set with a particular element. If you just want to override a few default properties, it's easiest just to send them as properties in the element tag.

In earlier versions of MODX, properties sent in a tag were referred to as *parameters*. You may still see that term used in older documentation and MODX forum posts.

We'll discuss properties and all their uses in much more detail later in the book.

The MODX Cache

MODX keeps much of the content of a site in the ***MODX cache***. This makes web page access much faster because MODX doesn't have to query the database for the content. Very few users will deal with the MODX cache directly, but you still need to know a little about it because you can control what content is cached.

The rule for what to cache is simple: Let MODX cache content that doesn't change often. If you have a web page that shows the current time and the page is cached, every visitor to the site will see the time that the page was first visited instead of the current time. This will continue until the MODX cache is cleared. Then a new time will be cached until the next time the cache is cleared.

Caching is controlled in two ways. The caching of MODX resources is controlled by two checkboxes on the Page Settings tab of the Create/Edit Resource panel in the Manager. If you uncheck the "Cacheable" checkbox, MODX will get a new version of the page every time someone visits it. If the "Empty Cache" checkbox is checked (which it is by default), the cache will be cleared when you save the page.

For all MODX tags (including those for snippets, chunks, and template variables), the format of the tag controls whether caching will be in effect. Putting an exclamation point after the opening braces of the tag will make the object uncached; leaving it out will make it cached. The exclamation point means "get me a fresh version of this." If the contents of a chunk seldom change, for example, you can let MODX cache it by using this tag:

```
[[$ChunkName]]
```

You can clear the MODX cache manually whenever you change the chunk's content. Snippets often produce different output at different times. In those cases, you'll want to have them uncached:

```
[[!SnippetName]]
```

Remember that the caching of different things in MODX Revolution is independent. For our example above about a page that shows the current time, the time would probably be calculated, formatted, and displayed by a snippet (let's call it *ShowTime*). If the rest of the page never changed, you could leave the page itself cached but call the snippet with this tag:

```
[[!ShowTime]]
```

That way, the page content would be cached, but the snippet's output would always be fresh because the snippet is called uncached.

The cache is also a source of confusion for some users because if the content is cached, changes they make may not show up when they view the site. MODX may be showing you the old cached content. Remember, too, that your browser may be showing you a cached version of a page you have visited before. When you make changes and can't see them, or correct a problem but still see it, clear the MODX site cache and your browser cache and check again.

Summary

By now you should have a basic idea of how MODX stores the information for a web site and transforms that information into finished web pages. Don't worry if some of the details aren't completely clear. We'll be covering all the topics in more detail later in the book.

Although you could create a fairly polished site using the techniques described in this chapter, we've only just scratched the surface of the power and flexibility MODX brings to web design.

Once users master the basic techniques involved in creating a MODX web site, they soon discover that they want more. They begin thinking about their site (or sites) in new ways and considering things like allowing subsets of users to access certain parts of the site (or certain functions in the Manager), letting users create and edit pages in the front end without logging in to the Manager, customizing pages for individual users, or automatically highlighting certain key terms on various pages. They wonder if what they're thinking of is possible in MODX, and the answer is almost always "yes."

MODX Evolution Notes

If you are using a pre 2.0.0 version of MODX, most of the sections of this chapter will still apply. The core of MODX has been completely rewritten for MODX Revolution. From the user's point of view, however, it still has much in common with previous versions. There have been a few minor changes in the database, but the table names and structures of the tables are unchanged for the most part.

General Differences

MODX Evolution and earlier versions still store information in the database and still have a front end and a back end (MODX Manager). The Manager interface is different, but most of the same functions are there and most users switching from one to the other have little trouble finding the familiar areas of the Manager.

As we mentioned earlier in this chapter, there is still a distinction between MODX itself and the third-party components that extend and expand it. In MODX Evolution, however, the distinction is somewhat blurred by the fact that a number of third-party components are included in the base install. There is also an option to install a complete sample MODX site, which includes a template, user permissions, user logins, and a working blog.

Terminology

If you are using a version of MODX prior to Version 1.0.0 (e.g., MODX 0.9.6), there is also a difference in the terminology used for various resources and components between the two versions (in fact, the meanings of the terms "resource" and "element" are different).

In earlier versions of MODX, documents were historically called "documents" and elements such as snippets, chunks, and files were referred to as "resources." That terminology has shifted to match that of MODX Revolution, but not all the documentation has caught up.

In current versions of MODX (both Evolution and Revolution), documents are classified as resources; chunks, snippets, plugins, templates, and template variables are classified as elements; and files are just files.

The new terminology does a better job of matching the way the resources and elements are actually used and how they exist in the database and in the MODX code. This switch sounds confusing, but in practice, people have little trouble making the transition.

Navigation

Navigation in the Manager is a little more cumbersome in Evolution. In order to edit a chunk, for example, you have to go to **Elements → Manage Elements → Chunks** and then select the chunk name. In Revolution, you simply expose the Element tree (by clicking on the "Elements" tab) and click on the chunk you want to edit.

Content Elements

As for the elements themselves, they are largely unchanged. Chunks, snippets, plugins, placeholders, links, and system settings are essentially the same, although there are minor changes in the tags used to insert them (see the table earlier in this chapter), and there is no language tag in Evolution.

Core Code

The code that MODX uses to deliver a document has been revamped in Revolution, but on the surface, the process is much the same. The Evolution parser still cycles through the content replacing the tags until they're all replaced. One change is that all tags can be cached or uncached in Revolution. In Evolution, only snippets can be designated as cached or uncached. In addition, the caching process has been made much more consistent in Revolution.

Language

MODX Evolution has no lexicon or topics. It simply has language files that are pulled in explicitly in the code of components or the Manager. Loading of these files is not automatic as it is in Revolution, and creating and changing language strings in Evolution involves editing the physical language files in a text editor — there is no option to override the strings in the Manager. Evolution also has no namespaces.

System Settings

System settings are essentially the same in Evolution, but editing them is more cumbersome. In Revolution, they all appear in the System Settings grid, organized by areas, and can be searched for using key terms. In Evolution, they are spread across five different tabs with no search function. Individual settings can sometimes be difficult to find. The system settings in Evolution are reached by going to **Tools → Configuration** on the Top Menu.

Context, Settings, Workspaces, and Add-ons

Evolution has no context settings, user settings, or workspaces, and probably the most important difference in terms of convenience, no transport packages. In Evolution, you must install third-party components by manually copying files to the correct locations, creating new chunks, snippets, and documents, and pasting the code from the source files directly into them. Usually, the third-party component will come with a `readme.txt` file explaining how to perform the installation. It's easy to make a mistake during this process, and it's often difficult to discover what mistake you have made.

In Revolution, you can install any third-party component that has a transport package by selecting it in the Repository, waiting a few seconds for it to download, and then clicking on the "Install" button. All this is done without leaving the Manager. In Revolution, transport packages can automatically create any necessary elements, tags, resources, system settings, etc. Transport packages can also interact with the user to allow you to set configuration options during the install.

Properties and Caching

There are properties in Evolution, but in practice, they are only used with snippets. They are almost always sent in snippet tags. Snippets can have default properties in Evolution, but they are somewhat cumbersome and are seldom used.

The MODX cache performs the same function in both Evolution and Revolution. The only differences are that in Revolution more elements can be designated as cached or uncached, the tag usage for caching varies slightly, and the Revolution cache is more consistent.

Index

!empty (output modifier), 172
{core_path}, 349, 411, 524, 527
$_COOKIE variable, 699
$_GET array, 528
$_GET parameters, 646
$_GET variable, 528, 646
$_output variable in plugins, 300-301
$_POST variable, 699
$_SERVER['HTTP_REFERER'] variable, 550
$_SESSION variable, 301, 699
$cacheFlag variable, 631-632, 633, 699
$classKey argument for getObject, 311-312
$key and $value pairs, 344-345
$modx variable, 310, 629, 699
$modx->resource variable, 152, 312-313, 318, 353-355, 387-388
  in plugins, 290
  reference variable for, 312-313
  using get() with, 327, 355, 359
  using getMany() with, 359
  using getObjectGraph() with, 363-364
  using getOne() with, 359
  using getTVValue() with, 376, 378
  using joinGroup() with, 332
  using leaveGroup() with, 332
  using save() with, 344
  using set() with, 344
  using toArray() with, 327, 355
$modx->user variable, 318, 381-389, 481
  checking login status with, 385, 469, 481, 490, 592
  checking user group membership with, 386
  setting placeholders for, 345, 384
  using get() with, 323, 327, 345, 381-382, 388-389, 592-593
  using getOne() with, 384, 592-593
  using hasSessionContext() with, 469, 481, 490, 592
  using isMember() with, 385
  using joinGroup() with, 332
  using leaveGroup() with, 332
$output variable (PHP), 656
$output variable in snippets, 176, 272
$resource variable, 311-313, 354
  assignment by reference, 312-313
  in plugins, 297-298
  in transport packages, 622
  using fromArray() with, 638
  using get() with, 320, 355, 637
  using getCollection() with, 313
  using getContent() with, 651
  using getMany() with, 318, 321
  using getObject() with, 313, 318, 320, 637
  using getObjectGraph() with, 323, 363
  using getOne() with, 318, 321, 390
  using getTVValue() with, 331
  using hasChildren() with, 356-357
  using joinGroup() with, 332
  using newObject() with, 340, 398-399, 638
  using save() with, 339, 340, 399
  using set() with, 340, 639
  using setContent() with, 340, 399, 651
  using toArray() with, 355, 370-373
$scheme argument for makeUrl(), 646
$scriptProperties array, 366-367, 374
$scriptProperties variable, 366-367, 374
$value variable, 344-345
$xpdo variable, 310, 325, 405, 629
(anonymous) user, 104, 381-383, 387, 460, 469, 474-475, 481
--ff-only merge (Git), 81, 88
.DOC files as static resources, 178
.htaccess file, 46-47, 59-60, 426
  configuring for FURLs, 59
  Furls and, 59
  renaming, 59
  RewriteBase, 60
.PDF files as static resources, 178
3PCS (see: add-on components)
404 (page-not-found) error, 50
500 server error, 46
= operator (PHP), 661
=& operator (PHP), 312, 354, 388
=== operator (PHP), 336
@ BINDINGS, 220-226
@CHUNK binding, 222-223
@DIRECTORY binding, 225
@EVAL binding, 225-226
@FILE binding, 224
@INHERIT binding, 207, 221-222
@RESOURCE binding, 224
@SELECT binding, 225
[[*content]] tag, 49, 145-146
[[*content]] tag in template, 194
_build directory (transport packages), 602, 614, 623

A

aborting transport packages, 620
about MODX, vi-xiii, 122
absolute URL in makeUrl() method, 646
Access Control Lists, 441, 449-454
  context access entries, 449-451
  creating entries, 449-454
  element category access entries, 453-554
  entries, 438
  resource group access entries, 451-453, 458-459
Access Controls menu, 115
  Access Policies tab, 115
  Category Access tab, 116
  Context Access tab, 115-116
  Flush All Sessions, 117
  Flush Permissions, 117
  Policy Templates tab, 115
  Resource Group Access tab, 115-116
  Resource Groups tab, 117
  Roles tab, 115
  User Groups, 115
  User Groups tab, 115-116
access policies, 440, 443
  Administrator, 443, 444-445
  context access, 444-445
  creating, 115, 443-445
  duplicating, 444, 456, 459
  editing, 115, 444
  EditorAdmin, 457, 458
  EditorElement, 462, 463
  EditorResource, 459
  Element, 443, 444-445
  element category, 444-445
  FeView, 471, 472
  FeViewTemplate, 486-487
  for context access ACL entries, 450-451
  for element category access ACL entries, 453
  for resource group access ACL entries, 452, 458-459
  Load, 471
  Load List and View, 444, 459
  Load Only, 443, 475

manager action, 444-445
object, 444-445
Resource, 443, 444-445, 458, 483, 484
resource group access ACL entries, 444-445
ResourceViewOnly, 460, 460
types of, 444-445
View, 471

access policy templates, 115, 443
FeViewTemplate, 471

`access_permissions` permission, 465

ACL entries, 438, 441
context access, 116, 449-451, 462-463
creating, 449-454
Element Category Access, 116, 453-554
Resource Group Access, 116, 451-453, 458-459

actions, 126, 502-508
creating, 524-525
resource/create, 514
resource/update, 514

Actions tree, 505-506

active user field, 479

active users, 479

`add` (output modifier), 172

Add Category dialog, 117, 462

add package, 537

Add Permission to Template dialog, 486-487

Add Resource Group dialog, 458, 483

Add User Group dialog (form customization), 518

Add User To Group dialog, 447-448, 472

add-on components
at `http://modx.com`, 537
commonly used, 535-594
downloading, 122
in Evolution, 96
installing, 52-54, 536-538
namespaces, 349
overview, 4
PackMan, 600
with no transport package, 538

`add_children` permission, 459

`addEventListener()` method, 640

adding
(see also: attaching and assigning)
chunks to categories in transport packages, 621-623
elements to categories in transport packages, 621-623
properties to property sets, 257-258
resources to resource groups, 117, 179
snippets to categories in transport packages, 621-623
tables to database, 409-411
template variables to resource groups, 219
template variables to templates, 219
template variables to templates in code, 402-404
Top Menu items, 126
users to user groups, 115, 447-448

`addMany()` method, 342, 402, 621, 637

`addOne()` method, 341-342, 403, 404, 406, 408, 409, 497, 637

`addPackage()` method, 630

admin Super User, 100, 101, 106, 442-444
accidentally giving users rights as, 457, 464-465
accidentally hiding menu items from, 512
accidentally hiding resources from, 439
accidentally hiding TVs from, 519, 521
accidentally removing permissions for, 444
advisability of adding to user groups, 447, 450
authority level, 450
changing username for security, 420
custom menu for, 510-513
default ACL entries for, 442
detecting in tags, 156
form customization rules and, 516
hiding menu items from others, 510-513
hiding resources from others, 433, 439, 458-459
hiding template variables from others, 483-484
inheriting permissions, 450
minimum role of, 450
previewing from Manager, 382, 468
restricting access for others, 466
restricting package installation to, 620
`tree_root_id` for, 467-468

Administrator policy, 443, 444-445

Administrator user group, 450-451

AdminOnly category, 461

Advanced distribution of MODX, 420-421

advanced snippet operations, 309-414

advanced transport package tutorial, 611-625

aggregate alias, 319

`alias` (Resource Alias) resource field, 142

AllDocs resource group, 455, 458

`allow_private` permission, 486

`alt` context, 426-433, 432

`and` (output modifier), 168

anonymous user, 104, 381-383, 387, 460, 469, 474-475, 481

Apache Friends, 62

Apache `httpd.conf` file, 60

`apache/bin` directory, 430

API documentation link, 129

Appendix, 629-715

architecture of MODX, x-xiii

arithmetic operators in PHP, 660

array functions in PHP, 678

array of file paths for transport packages, 623-624

`array_merge()` PHP function, 375, 384, 388

arrays in PHP, 668-669

assets (in Update Action dialog), 506

`assets` directory, 20

`assets` directory (transport packages), 623

`assets` directory for transport packages, 602-604

`assets_url` system setting, 193, 527

assigning
(see also: adding and attaching)
chunks to categories in code, 401-402
chunks to categories in transport packages, 621-623
resources to resource groups, 117, 179
resources to resource groups in code, 332, 408, 408
roles to users, 115
snippets to categories in transport packages, 621-623
template variables to resource groups, 219
templates to categories in code, 401-402

users to user groups, 115, 447-448
users to user groups in code, 332, 406-408
assignment by reference, 312-313
assignment operators in PHP, 661
associative arrays (PHP), 669
attaching
(see also: adding and assigning)
elements to categories in transport packages, 621-623
plugins to events in code, 408
properties to property sets, 257-258
property sets to elements, 251
property sets to elements in code, 409
property sets to plugins, 293-295
snippets to categories in code, 401-402
system events to plugins, 292, 292
template variables to templates, 203, 219
template variables to templates in code, 402-404
attachment content disposition, 178
audience for this book, ii
authenticating users, 302, 384-385, 469
authority numbers, 440, 446
auto-tag template variable input type, 208-209
`automatic_alias` system setting, 143, 409
avoiding the error page, 475

B
back end, 5
contexts, 441
system events, 699-713
versus front end, 5
back-ticks, 12
BadWords Manager plugin, 296-299
BadWords plugin, 290-295
BadWords tutorial, 290-295
BadWordsMgr tutorial, 296-299
base href tag, 429
`base_url` context setting, 428
`base_url` system setting, 428
BaseTemplate, 49
basic building blocks of MODX, 6-20
Batcher add-on component, 504-507
Batcher namespace path, 505
`beginTransaction()` method, 405, 630
block comments in PHP, 654
blogging with getResources, 571
boolean variables in PHP, 655
branches (Git), 70-72
BreadCrumbs
add-on component tutorial, 551-555
CSS, 553-555
installing, 552
properties, 553
browser support in MODX, 36-37
bug fixes in Git, 84
bug reporting link, 129
build scripts
for transport packages, 601, 605-610
MyComponent example, 625
running outside of MODX, 624-625
`build.config.php` file (Revolution development version), 87
`build.config.php` file (transport packages), 624
`build.transport.php` file (transport packages), 600-601, 621-623
building blocks of MODX, 6-20
building the development version of Revolution, 87-88
built-in functions in PHP, 675-679
bulk actions in form customization, 523

C
cache
`clear_cache` permission, 503
clearing, 28, 65, 111
control, xiii
control in Evolution, 187
control with tags, 166
empty cache checkbox, 150, 150
exclamation point in tags, 166
in Evolution, 33
output (viewing), 180
overview, 28-29
system events, 715
tags and, 166
cacheable resource field, 150
cached tags, 150, 166
calling processors in code, 409
cancel button, 455
Captcha plugin, 300-302
caption TV field, 205
`cat` (output modifier), 169
categories, 105
`addMany()` method, 402
AdminOnly, 461
assigning chunks to in code, 401-402
assigning elements to in transport packages, 621-623
assigning snippets to in code, 401-402
assigning templates to in code, 401-402
creating, 105, 106
creating in code, 343, 401-402
defined, 14
in transport packages, 621-623
manager system events, 712
placing elements in, 105
protected, 453
removing, 106
uses for, 14
category TV field, 205
`cdata` (output modifier), 171
ChangeEmail custom hook for FormIt tutorial, 591-594
`changelog.txt` (transport packages), 614-615
changelogs (transport packages), 602
ChangePassword snippet, 481
`changePassword()` method, 640, 650
`changeWebUserPassword()` method, 640
changing
(see also: modifying and renaming)
component menu items, 504-505
current resource alias in code, 344
language strings used in the Manager, 164
the site name, 49
Top Menu items, 504-505
character set, 63
CheatSheet component tutorial, 523-533
checkbox template variable input type, 209-210
checking user's login status, 481
checkout (Git), 80, 88
CheckPermission snippet, 487-488
`checkPreview()` method, 640
`checkSession()` method, 640
Children related object, 318, 321, 324, 357, 364
Chmod directory dialog, 108
chunk tags, 11-12, 157, 194
chunk tags in Evolution, 264-265

chunks
- assigning to categories in code, 401-402
- assigning to categories in transport packages, 621-623
- creating, 230
- creating in code tutorial, 400
- defined, 11, 230
- displaying, 17
- displaying with a snippet, 237
- editing in Evolution, 264-265
- emailTpl, 585
- `getChunk()` method, 368-370
- in snippet properties, 236-237
- ItemTpl, 242, 284
- manager system events, 705-706
- object reference, 692
- OuterTpl, 242, 284
- overview, 230
- relatedUserTpl, 389
- RowTpl, 242, 284
- `save_chunk` permission, 449, 453
- sending properties in tag, 249
- Tpl, 241-244, 473
- tutorial, 230-235
- TV-based, 244-247
- uses for, 232-244, 238-241
- using tags with, 232
- using to store data, 238-241
- using with snippets, 236-237
- working with, 229-247

class variables in PHP, 683
`class_key` (Class Key) resource field, 151
classes
- creating in PHP, 682-686
- in PHP, 679-686
- modUser, 650-651
- modX, 640-649
- xPDO, 630-637
- xPDOObject, 637-639

`cleanDocumentIdentifier()` method, 640
`clear_cache` permission, 503
clearing MODX, 65
cloning a fork at GitHub, 78-79
cloning Revolution development version, 73-75
CMPs, 523-533
CodeMirror editor, 269
coding style (PHP), 672-673
command-line client for Git, 73
comments
- HTML, 531
- in PHP, 653-655
- PHP block, 654
- PHP single-line, 654

commit messages (editing in Git), 85
`commit()` method, 4-6, 630
commits (Git), 71
common MODX add-on components, 535-594
common snippets in Evolution, 595-596
comparison operators in PHP, 661-662
components
- changing menu item for, 504-505
- CheatSheet, 523-533
- namespaces, 349
- overview, 4
- Top Menu item, 504-505

Components Top Menu item, 109, 113, 508
composite alias, 319
conditional output modifiers
- `Reference`, 167-169
- `replacing with snippets`, 280-283

conditionals in PHP, 664-668
config file still writable warning, 45
`config.core.php` files, 420
`config.inc.php` file
- Evolution, 95
- Revolution, 41, 420-423

configuration check
- Evolution, 96
- Revolution, 45, 110

configuring
- `.htaccess` file for FURLS, 59
- CodeMirror editor, 269
- File Manager in Evolution, 131
- Git editor for commit messages, 85
- Git for MODX development, 76-81
- modules in Evolution, 131
- MODX Evolution, 131
- SMTP for Gmail, 586, 696
- SPForm, 548-551
- XAMPP, 62-63

Confirm Registration resource, 477
conflicting permissions, 440
`connect()` method, 631
connection information, 42
`connectors` directory, 420
constants
- `ECHO`, 635
- `FILE`, 635
- `HTML`, 635
- `MODE_NEW`, 299, 699
- `MODE_UPD`, 299, 699
- `modSystemEvent::MODE_NEW`, 299, 699
- `modSystemEvent::MODE_UPD`, 299, 699
- `modX::LOG_LEVEL_INFO`, 625
- `modX::SESSION_STATE_EXTERNAL`, 644
- `modX::SESSION_STATE_INITIALIZED`, 644
- `modX::SESSION_STATE_UNAVAILABLE`, 644
- `modX::SESSION_STATE_UNINITIALIZED`, 644
- `MODX_ASSETS_PATH`, 603, 614
- `MODX_ASSETS_URL`, 603
- `MODX_CONFIG_KEY`, 601
- `MODX_CORE_PATH`, 601, 603, 630, 634
- `MODX_PROCESSORS_PATH`, 422
- `REDIRECT_HEADER`, 648
- `REDIRECT_META`, 648
- `REDIRECT_REFRESH`, 648
- `xPDO::LOG_LEVEL_DEBUG`, 635
- `xPDO::LOG_LEVEL_ERROR`, 635
- `xPDO::LOG_LEVEL_FATAL`, 635
- `xPDO::LOG_LEVEL_INFO`, 617, 618, 635
- `xPDO::LOG_LEVEL_WARN`, 635
- `XPDO_CLI_MODE`, 625
- `xPDOTransport::ABORT_ON_VEHICLE_FAIL`, 620
- `xPDOTransport::ACTION_INSTALL`, 618
- `xPDOTransport::ACTION_UNINSTALL`, 618
- `xPDOTransport::ACTION_UPGRADE`, 618
- `xPDOTransport::PACKAGE_ACTION`, 617
- `xPDOTransport::PRESERVE_KEYS`, 606, 610, 621-622
- `xPDOTransport::RELATED_OBJECT_ATTRIBUTES`, 621-623
- `xPDOTransport::RELATED_OBJECTS`, 621-623
- `xPDOTransport::UNIQUE_KEY`, 606, 610, 621-622
- `xPDOTransport::UPDATE_OBJECT`, 606, 610, 621-622

constraint field (form customization), 518, 521
constraints in form customization,

515, 521-522
constructors in PHP, 684
contact form
  creating with FormIt tutorial, 581-587
  creating with SPForm tutorial, 543-551
Container (`isfolder`) resource field, 146
`container_suffix` system setting, 57, 146
`content` (Resource Content) resource field, 145-146
Content Disposition (`content_dispo`) resource field, 151
content disposition of attachments, 178
content elements defined, 9
content elements in Evolution, 31
content tag, 49, 145-146
content tag in template, 194
content types, 125
`content_dispo` (Content Disposition) resource field, 151
`content_type` (Content Type) resource field, 150
context access ACL, 115-116
context access ACL entries, 449-451, 456-457, 462-463
context access policy, 444-445
`context` property for link tags, 163
context settings, 25-26
  `base_url`, 428
  `default_template`, 429, 429, 432
  `http_host`, 429, 431
  in Evolution, 32
  `site_start`, 428
  `site_url`, 429
contexts, 424-434
  `alt`, 426-433, 432
  back end, 441
  `base_url` setting, 428
  creating, 25-26, 125
  creating a second, 426-433
  `default_template` setting, 429
  defined, 25-26
  front end, 441
  Gateway plugin, 432
  `http_host` setting, 429
  in Evolution, 32, 435
  in separate directories, 426-431
  `initialize()` method, 426-427
  manager system events, 713
  `mgr`, 25-26, 424-425, 441, 450-451, 456, 458, 459, 460, 483
  protected, 439
  security permissions, 433
  settings, 428-429
  sharing core between sites, 425-426, 433
  `site_start` setting, 428
  `site_url` setting, 429
  two in the same directory, 431-432
  uses for, 424
  `web`, 25-26, 424-425, 432, 441, 446-447, 457, 460, 468, 472, 484
contributing
  a bug fix with Git, 84
  a new feature in Git, 86
  to MODX development, 75-89
contributor workflow in Git, 83-89
controller, 506
controlling
  access in the front end, 468-474
  plugins with template variables, 295
  program flow (PHP), 670-672
  snippets with template variables, 204
  text color with a TV tutorial, 377-379
conventions (typographical in *MODX: The Official Guide*), iii-v
converting a site to MODX, 89-94
converting a template to MODX, 90
core
  directory, 20, 349, 411, 421-422, 425-426, 433, 524
  directory (moving), 421-422
  directory (renaming), 421-422
  directory (transport packages), 602-603, 623
  independence in MODX, xii-xiii
  lexicon, 504-505
  `MODX_CORE_PATH` constant, 601, 603, 630, 634
  `modx_core_path` system setting, 642
`core/packages` directory, 537
`count()` PHP function, 678
cPanel, 429
Create Access Policy dialog, 471
Create Directory dialog, 108
Create Directory Here, 108
Create Namespace dialog, 126
Create New Element Here, 106
Create New Profile dialog (form customization), 517-518
Create New Set dialog (form customization), 518
Create New Setting dialog, 466-467, 491
`create` permission, 453
Create Resource Group Dialog, 448, 455, 470, 482
Create Role dialog, 440, 471
Create User Group dialog, 447, 471
Create/Edit Resource panel, 135-154
  access controls tab, 179
  `alias` (Resource Alias) resource field, 142
  altering with form customization, 513-523
  Cacheable (`cacheable`) resource field, 150
  Class Key (`class_key`) resource field, 151
  Container (`isfolder`) resource field, 146
  `content` (Resource Content) resource field, 145-146
  Content Disposition (`content_dispo`) resource field, 151
  Content Type (`content_type`) resource field, 150
  Created By (`createdby`) resource field, 152
  Created On (`createdon`) resource field, 152
  creating a new tab with form customization, 522
  Deleted By (`deletedby`) resource field, 153
  Deleted On (`deletedon`) resource field, 153
  Description (`description`) resource field, 142
  Edited By (`editedby`) resource field, 153
  Edited On (`editedon`) resource field, 153
  Empty Cache checkbox, 150
  Freeze URI (`uri_override`) resource field, 154
  Hide From Menus (`hidemenu`) resource field, 145
  hiding tabs with form customization, 520
  `id` (Resource ID) resource field, 138
  `introtext` (Summary) resource field, 143
  `isfolder` (Container) resource field, 146
  Link Attributes (`link_attributes`) resource field, 143

Long Title (`longtitle`) resource field, 142
Menu Index (`menuindex`) resource field, 145
Menu Title (`menutitle`) resource field, 144
modx-page-settings tab, 520
modx-panel-resource-tv tab, 520, 522
modx-resource settings tab, 520
modx-resource-access-permissions tab, 520
`pagetitle` (Resource Title) resource field, 141
Parent Resource (`parent`) resource field, 144
Publish Date (`pub_date`) resource field, 148
Published (`published`) resource field, 141
Published checkbox, 141
Published On (`publishedon`) resource field, 147
Resource Alias (`alias`) resource field, 142
Resource Content (`content`) resource field, 145-146
Resource ID (`id`) resource field, 139
Resource Title (`pagetitle`) resource field, 141
Rich Text (`richtext`) resource field, 147
Searchable (`searchable`) resource field, 149
setting field default values form customization, 523
setting labels form customization, 523, 523
Summary (`introtext`) resource field, 143
`template` (Uses Template) resource field, 140
template variables tab, 179
Unpublish Date (`unpub_date`) resource field, 149
URI (`uri`) resource field, 154
`uri_override` (Freeze URI) resource field, 154
Uses Template (`template`) resource field, 140

Create/Edit User panel, 114
access permissions tab, 114
general information tab, 114
settings tab, 114

`createdby` (Created By) resource field, 152

CreatedBy related object, 320, 323, 388, 390

`createdon` (Created On) resource field, 152

CreatedResources related object, 324

`createPackage()` method (transport packages), 606, 608

`createVehicle()` method (transport packages), 606, 610, 622

creating
a second context, 426-433
Access Control List entries, 449-454
access permissions, 114-115
access policies, 115, 443-445
ACL entries, 449-454
actions, 524-525
branch with Git, 84
categories, 106
categories in code, 343, 401-402
chunks, 230
chunks in code, 339
chunks in code tutorial, 400
classes in PHP, 682-686
contact form with FormIt tutorial, 581-587
contact form with SPForm tutorial, 543-551
content types, 125
context access ACL entries, 456-457
contexts, 25-26, 125
custom hooks for FormIt, 588-594
custom output modifiers, 174-176
custom permissions, 486-487
custom validators for FormIt, 578-579
default properties, 253
directories, 108
documents, 136
elements, 106
elements in code tutorial, 400
error (404 page-not-found) page, 50
FAQ page with EZfaq snippet tutorial, 539-542
fork of MODX at GitHub, 78-79
forms with FormIt tutorial, 571-594
lexicon files, 412, 412
lexicon strings (entries) for CheatSheet component, 526-527
Login page, 472-473
menu with Wayfinder tutorial, 555-565
messages for users, 127
MIME types, 125
MODX database, 40
MODX objects in code, 338-344, 397-409
namespaces, 126, 411
new Create/Edit Resource panel tabs with form customization tutorial, 522
objects by calling processors, 409
properties for template variables, 219
property sets, 119, 256, 261-262, 279, 538-539
resource group access ACL entries, 458-459
resource groups, 117, 448
resources, 6, 50, 103, 113, 135-154
resources by calling processor, 409
resources in code, 339
resources in code tutorial, 398
roles, 115, 440, 446
simple class in PHP tutorial, 682-686
snippets in code, 340-341
snippets in code tutorial, 400
static resources, 103, 113, 178
symlinks, 103, 113
template variables, 201
template variables in code tutorial, 402-404
template variables in Evolution, 227
templates, 190
templates in code tutorial, 400
Top Menu actions, 524-525
Top Menu items, 508-510
transport package tutorial, 599-625
transport packages, 599-626
unauthorized page, 50
user extended fields, 115
user groups, 115, 115, 447, 447
user profile in code, 341-342, 404-405
user settings, 114
users in code, 404
virtual hosts, 429-431
weblinks, 103, 113

credits (MODX), 122

criteria, 311-313

criteria for queries, 363

CSS
file for SPForm, 547
for BreadCrumbs, 553-555
Wayfinder, 559

cURL, 52, 536
enabling in XAMPP, 536

php_curl.dll file, 536
current resource
changing alias in code, 344
children, 315, 315
creator, 388
editor, 388
getting children, 360-361
getting ID of in code, 315, 327-328
getting in code, 312, 318, 327-328
getting parent information, 359
getting TV values for, 331, 376, 378
getting TV values in code, 376
in Evolution, 307
publisher, 389
related objects, 387-389
toArray() method, 328, 356
users related to, 387-389
current user, 16, 318, 327, 381-383, 384, 469, 485, 492-493, 592-593
currently logged-in users, 111
custom database table schemas, 409-411
custom database tables, 409-411
custom hooks
for FormIt, 588-594
variables available for FormIt, 590
custom manager page tutorial, 523-533
custom manager pages in Evolution, 533
custom menus, 513
Custom output modifiers, 174-176
custom permissions, 510-512
creating, 486-487
custom preHooks for FormIt, 591-593
custom Top Menu items, 508-509
custom validator properties for FormIt, 578-579
custom validators for FormIt, 578-579
customizing the Create/Edit Resource panel with form customization, 513-523
customizing the MODX Manager, 501-533

D
data storage in chunks, 238-241
database
adding custom tables to, 409-411
deleting objects in code, 344
dumping, 65-66
exporting, 65-66
getting chunks from, 311
getting related objects, 359-361
getting resources from, 311
getting system settings from, 380
getting template variables from, 377-379
getting user extended fields from, 397
getting user groups from, 406-408
getting user profiles from, 382-383
getting users from, 311, 392-393
host, 42
importing, 65-66
location of properties, 250
location of property sets, 250
modifying MODX objects in code, 338-344
modx_actions table, 506
name, 42
password, 40
queries, 631-632, 635-638
removing objects in code, 344
sharing between sites, 425
site_content table, 137
table prefix, 425
username, 40
users table, 479
database tables
adding to database, 409-411
custom, 409-411
overhead, 121
site_snippets, 250
date (output modifier), 172
date and time functions in PHP, 679
date template variables
formatting codes, 215
input type, 210
output type, 214-215
date() PHP function, 679
DayErrorPage resource, 303
DBAPI in Evolution, 416
debugging PHP code, 686-690
decr (output modifier), 172
decrement (output modifier), 172
default
collation, 63
database, 42
lexicon file, 348
permissions, 442
security settings, 442
template variable output type, 215
default (output modifier), 172, 567
default properties, 249-255
creating, 253
editing, 251
elements, 249-255
in Evolution, 306
of snippets, 276-277
overriding, 256
removing, 255
uses for, 255
using with plugins, 293-295
versus property sets, 249-251
working with, 249-251
Default Value TV field, 207
default.inc.php lexicon file, 348
default_template context setting, 429, 432
defined constants (transport packages), 605-607
deletedby (Deleted By) resource field, 153
DeletedBy related object, 390
deletedon (Deleted On) resource field, 153
deleting
(see also: removing)
branches (Git), 82
categories, 106
default properties, 255
directories, 109
files, 109
MODX objects in code, 344
property sets, 260
resources, 102-103, 182
resources from resource groups, 448
resources from resource groups in code, 332
users from user groups, 447-448
users from user groups in code, 332
delimiter template variable output type, 216
description resource field, 142
description TV field, 205
design philosophy of MODX, xi
develop branch (Git), 71
development version of Evolution, 97
development version of Revolution, 70-89
build.config.php file, 87
building, 87-88
cloning, 73-75
transport.core.php file, 87
updating, 88-89
dialogs

Add Category, 117, 462
Add Permission to Template, 486-487
Add Resource Group, 458, 483
Add User Group dialog (form customization), 518
Add User to Group, 447-448, 472
Chmod directory, 108
Create Access Policy, 471
Create Directory, 108
Create Namespace, 126
Create New Profile dialog (form customization), 517-518
Create New Set dialog (form customization), 518
Create New Setting, 466-467, 491
Create Resource Group, 448, 455, 470, 482
Create Role, 440, 471
Create User Group, 447, 471
New Category, 462
New Property Set, 119
PhpMyAdmin File Download, 65-66
Rename Directory, 109
Rename File, 109
Update Action, 505
Update Menu, 510
Update User Role, 448
Upload Files, 108
User Group Context Access, 449-451, 456-457

directories
_build (transport packages), 602, 614
apache/bin, 430
assets, 20
chmod, 108
connectors, 420
core, 20, 349, 411, 421-422, 425-426, 433, 524
core/packages, 537
creating, 108
deleting, 108
htdocs, 63
manager, 420
moving manager, 423
moving the core, 421-422
public_html, 59, 422, 602
removing, 108
renaming, 109
renaming manager, 423
renaming the core, 421-422
root, 420
setting permissions, 108
setup, 420
XAMPP, 63

directory names in transport packages, 601
directory permissions, 41, 108
disabling
CodeMirror, 269
news feed, 110
permissions, 115, 444, 452, 463, 465-466, 514
plugins, 296
RSS feeds, 110
security feed, 110
snippets, 92

displaying
a chunk with a snippet, 237
chunks, 17
language strings, 17
lexicon strings with code, 347-351, 413
links, 17
placeholders, 17
resource content, 49, 145-146, 193
resource fields, 17
settings, 17
snippet output, 17, 272
snippet tags (without executing), 330
template variables, 11, 17, 200, 203, 375-376
user information, 16, 380-396

Ditto, 565, 570
div (output modifier), 172
divide (output modifier), 172
documentation link, 122
documentObject, 307
DocumentRoot, 430, 431
documents
(see also: resources)
Create, 103
creating, 6, 136
defined, 7
edit_document permission, 465
fields, 7, 135-154, 138
in Evolution, 185, 397
Private Document, 470
Private Document2, 486
publish_document permission, 449
recently edited (Manager Home screen), 110
recently edited (Report menu)), 121
save_document permission, 453, 465
using tags in, 154-167

double quotes in PHP, 657-659
Download Extras button, 122
downloading MODX, 38
downloading transport packages, 53, 122
dropdown list menu template variable input type, 210
dumping database, 65-66
Duplicate Resource, 103, 181
duplicating access policies, 444, 456, 459

E

ECHO constant, 635
echo statement in PHP, 676
Edit Element, 106
edit permission, 453
Edit Resource, 103
edit_document permission, 465
edit_tv permission, 514
editedby (Edited By) resource field, 153
EditedBy related object, 323, 388
editedon (Edited On) resource field, 153
editing
access policies, 115, 444
chunks in Evolution, 264-265
commit messages (Git), 85
default properties, 251
elements in Evolution, 184
files, 108-109
files in Evolution, 130
object policies, 465-466
property sets, 251
property sets on the tools menu, 260
resources, 103, 135-154
system settings, 50

Editor role, 458, 459, 460, 463, 484
EditorAdmin policy, 456, 457
EditorElement policy, 462, 463
EditorResource policy, 459
editors
CodeMirror, 269
TinyMCE, 147

Editors user group, 455
eg (output modifier), 168
el (output modifier), 168
element categories, 105
Element Category Access ACL entries,

116
element category policies, 444-445
Element policy, 443, 444-445
Element tree, 104-106
  Create New Element Here, 106
  Edit, 106
  Quick Create, 106
  right-click menu, 106
Element tree icons, 104
`element_tree` permission, 465
elements
  attaching property sets to, 251
  attaching property sets to in code, 409
  creating, 106
  creating in code tutorial, 400
  default properties, 249-255
  defined, 9
  editing, 106
  `element_tree` permission, 465
  hiding, 461-463
  in Evolution, 31
  protected, 439
  Quick Update, 106
  removing, 106
  `save_element` permission, 443
`ellipsis` (output modifier), 171, 567
`else` (output modifier), 168
`else` statement in PHP, 665-667
`elseif` statement in PHP, 665
email hook for FormIt, 581
email hook properties for FormIt, 587
email template variable input type, 210
`emailsender` system setting, 651
`emailsubject` system setting, 651
emailTpl chunk, 585
`empty` (output modifier), 172
Empty Cache checkbox, 150
`empty()` PHP function, 335-336
enabling
  cURL, 52
  cURL in XAMPP, 536
  `mod_rewrite`, 60
  news feed, 110
  RSS feeds, 110
  security feed, 110
enhanced hello snippet, 273-275
eof marker (PHP), 671
`eq` (output modifier), 168
`equalorgreatherthan` (output modifier), 168
`equals` (output modifier), 168
`equalto` (output modifier), 168
`equaltoorlessthan` (output modifier), 168
error log, 121, 690
error page, 50
  avoiding, 475
error page unpublished warning, 110
`error_page` system setting, 302, 475
`errorCode()` method, 631
`errorInfo()` method, 631
errors
  404 (page-not-found), 50
  server 500, 46
  xPDO log, 635
`esc` (output modifier), 170
`escape` (output modifier), 170
event model, 300-301
events (see: system events)
Evolution
  cache, 33
  cache control, 187
  chunk tags, 264-265
  common snippets, 595-596
  config file, 95
  configuration check, 96
  content elements, 31
  context settings, 32
  contexts, 32, 435
  custom manager pages, 533
  DBAPI, 416
  default properties, 306
  development version, 97
  documentObject, 307
  documents, 185
  editing chunks, 264-265
  editing elements, 130, 184
  editing files, 130
  elements, 31
  file and directory permissions, 95
  form customization, 533
  FURLs, 97
  `getChildIds()` method, 416
  `getChunk()` method, 416
  `getLoginUserID()` method, 307
  `getLoginUserName()` method, 307
  `getParentIds()` method, 416
  getting system settings, 416
  installing, 42, 96
  installing add-on components, 96, 595
  installing snippets, 306
  lexicon, 32
  lexicon strings, 416
  missing Manager functions in, 133
  module manager system events, 712
  modules, 533
  overview, viii
  `parseChunk()` method, 416
  PHX, 187
  placeholders, 306
  plugins, 307
  properties, 32, 186, 265, 306
  Resource tree, 184
  roles, 446, 498
  `runSnippet()` method, 416
  security system, 497-498
  snippet properties, 595
  snippets, 306
  system settings, 32
  tags, 17, 306
  templates, 185
  Top Menu, 130-132
  transport packages, 626
  user information in, 307
  user settings, 32
  `userLoggedIn()` method, 307
  using SMTP, 596
  versus Revolution, vii-x
example template, 191
`exec()` method, 631
`explode()` PHP function, 678
exporting property sets, 261
extended user fields, 115
extended user fields in code, 393-397
ExtJS, 523, 532
extras (see: add-on components)
EZfaq
  installing, 539-540
  properties, 542
  snippet tutorial, 539-542
  styling FAQ page, 541-542

F

false in PHP, 656-657
`fclose()` PHP function, 678
feature branches (Git), 71
features of MODX, x-xiii
`feof()` PHP function, 671
fetch (Git), 80-81, 88
FeView policy, 471, 472
FeViewer role, 471, 472
FeViewTemplate access policy, 471, 486-487

fgets() PHP function, 671, 678
field default values (setting with form customization), 523
field labels (setting with form customization), 523
field rules in form customization, 515
fields
(see also: resource fields)
constraint field in form customization, 518, 521
extended user (in code), 393-397
getting user extended fields in code, 397
hiding with form customization, 522
of template variables, 205-208
file
(see also: files)
functions in PHP, 678-679
paths for transport packages, 623-624
pointers in PHP, 655
resolvers (transport packages), 612-614, 623
validators (transport packages, 619
vehicles (transport packages), 614-615
file and directory permissions, 41, 69
FILE constant, 635
file template variable input type, 211
File tree, 106-109
chmod directory, 108
icons, 107
right-click menu, 108-109
upload files, 108
file-not-found page, 50
file_exists() PHP function, 678
file_get_contents() PHP function, 606, 609, 615, 678
file_put_contents() PHP function, 678
file_tree permission, 443, 465
filemanager_path system setting, 467-468
filemanager_path tutorial, 467-468
filemanager_path user setting, 467-468
files
(see also: file)
.DOC as static resources, 178
.htaccess, 46-47, 426
.PDF as static resources, 178
Apache httpd.conf, 60
build.config.php (Revolution development version), 87
build.config.php (transport packages), 624
build.transport.php (transport packages), 600-601, 621-623
config.core.php, 420
config.inc.php, 420-423
config.inc.php for Evolution, 95
creating lexicon files, 412
default.inc.php, 348
deleting, 109
editing, 108-109
editing in Evolution, 130
Evolution config, 95
file_tree permission, 443, 465
filemanager_path setting, 467-468
hosts, 430
ht.access, 46-47
httpd-vhosts.conf, 430-431
httpd.conf, 60
in MODX, 19
index.php (components), 505-506
index.php (for CheatSheet component), 525
index.php (MODX), 38, 56, 61, 142, 165
index.php (separate contexts), 426-428
index.php for CheatSheet component, 527
install.script.php (transport packages), 616-620
lexicon, 348, 412
lexicon file locations (transport packages), 611-612
loading lexicon files in snippets, 412
locations (transport packages), 601-604
locations for MODX, 20
locations for XAMPP, 63
modresponse.class.php, 290
MODX, 19
MODX config, 41, 41, 420-423
MODX schema, 318-319
modx.class.php, 601
modx.mysql.schema.php, 318-319
names in transport packages, 601
permissions, 41
permissions in Evolution, 95
php.ini, 536
php_curl.dll, 536
placement, 38-39
readme.txt (transport packages), 602, 614-615
removing, 109
renaming, 109
schema (MODX), 318-319
setting permissions, 69, 108
SPForm CSS, 547
SPForm lexicon, 548
transferring in transport packages, 612-615
transport package, 601-604
transport package lexicon, 603
transport.core.php (Revolution development version), 87
uploading, 108
user.input.html (transport packages), 614-615
filters, 167
firing system events, 289-290
float variables in PHP, 655
flush all sessions, 117, 442, 457-458, 462
flush permissions, 117, 442, 457-458, 462, 474
fonts used in *MODX: The Official Guide*, iii-v
fopen() PHP function, 678
for loop (PHP), 671-672
foreach loop (PHP), 670
ForgotPassword snippet, 480
Form Customization, 513-523
bulk actions, 523
constraints, 515, 521-522
creating new Create/Edit Resource panel tabs with, 522
field rules, 515
hiding fields with, 522
hiding tabs with, 520
hiding template variables with, 514
in Evolution, 533
Manager panel, 117
moving TVs to a new tab with, 520
overview, 514-516
profiles, 516
rule sets, 516
rules, 515
sets, 515
setting field default values with, 523
setting field labels with, 523
setting template variable labels with, 523
tab rules, 515
TV rules, 515
form processing, 571-594

formatting codes for date template variables, 215
FormIt
built-in hooks, 580-582
built-in validators, 574-578
contact form tutorial, 581-587
custom hook tutorial, 588-594
custom preHooks, 591-593
custom validator properties, 578-579
custom validators, 578-579
email and redirect hooks, 580
email hook properties, 587
FormItAutoResponder hook, 580, 587-588
hooks and preHooks, 579-582
properties, 573-574
snippet tutorial, 571-594
spam hook, 587
tpl chunks, 585, 587
uses for, 571, 581
using SMTP with, 586
validators, 574-578
variables available in custom hooks, 590
FormItAutoResponder hook, 587-588
FormItAutoResponder properties, 587-588
FormItRetriever snippet, 592
formProcAllowedReferers (SPForm), 550
forums link, 127-128
`fputs()` PHP function, 679
`fread()` PHP function, 678
Freeze URI (`uri_override`) resource field, 154
friendly alias path, 57
friendly aliases, 58
Friendly URLs (see:FURLs)
`fromArray()` method, 340-341, 638
`fromJSON()` method, 638
front end, 5
contexts, 441
controlling access in, 468-474
hiding template variables in (tutorial), 484-485
versus back end, 5
front-end system events, 714-715
full URL in `makeUrl()` method, 646
functions
(see: PHP functions)
PHP, 673-679
FURLs
alias field, 142
friendly alias path, 57
friendly aliases, 58
in Evolution, 97
in Revolution, 55-61
prefixes, 57
RewriteBase, 60
suffixes, 57
testing, 60
`fwrite()` PHP function, 678

G
Gateway plugin, 432
`ge` (output modifier), 168
generating a GitHub SSH key, 77
generic methods, 310-314
`get()` and `set()` methods in PHP, 684-685
`get()` method, 327, 355, 391, 629, 638
`get*()` methods, 310-327
`getActiveChildren()` method, 640
`getAggregates()` method, 631
`getAllChildren()` method, 640
`getAncestry()` method, 631
`getAttribute()` method, 631
`getAuthenticatedUser()` method, 640-641
`getCacheManager()` method, 631, 641
`getCachePath()` method, 631
`getChildIds()` method, 315, 357, 641
`getChildIds()` method in Evolution, 416
`getChunk()` method, 237, 317, 368-370, 389, 488, 528, 642
`getChunk()` method in Evolution, 416
`getCollection()` method, 313, 362-363, 373, 390, 392-393, 403, 407, 631-632
`getCollectionGraph()` method, 324-325, 364-365, 391, 397, 631-632
`getComposites()` method, 631-632
`getConfig()` method, 642
`getContent()` method, 328, 651
`getContext()` method, 642
`getCount()` method, 633
`getDebug()` method, 633
`getDebugBacktrace()` method, 633
`getDocGroups()` method, 642
`getDocument()` method, 642
`getDocumentChildren()` method, 642
`getDocumentChildrenT-Vars()` method, 642
`getDocumentIdentifier()` method, 642
`getDocumentMethod()` method, 642
`getDocumentObject()` method, 642
`getDocuments()` method, 642
`getEventMap()` method, 642
`getFields()` method, 633
`getFullTableName()` method, 642
`getIterator()` method, 313, 633
`getKeywords()` method, 642
`getLoginUserID()` method, 642
`getLoginUserID()` method (Evolution), 307
`getLoginUserName()` method, 643
`getLoginUserName()` method (Evolution), 307
`getLoginUserType()` method, 643
`getManager()` method, 633
`getManagerPath()` method, 643
`getMany()` method, 321, 359-361, 373, 638
`getMany()` versus `getOne()`, 321
`getMETATags()` method, 643
`getMicroTime()` method, 633
GetMyComponent add-on package, 625
`getObject()` `$classKey` argument, 311-312
`getObject()` method, 310-313, 354, 390, 629, 633
`getObjectGraph()` method, 321-324, 363-364, 384, 388, 396, 633-634
`getOne()` method, 320, 348, 358-360, 388, 390, 392, 639
`getOne()` versus `getMany()`, 321
`getOption()` method, 314, 367-368, 490, 527, 634
`getPackage()` method, 634

getPageInfo() method, 643
getParent() method, 643
getParentIds() method, 315-316, 358, 643
getParentIds() method in Evolution, 416
getParser() method, 643
getPK() method, 634
getPKType() method, 634
getPlaceholder() method, 316, 643
getProperties() method, 370
getRegisteredClientScripts() method, 643
getRegisteredClientStartupScripts() method, 643
getRequest() method, 643
getResources
- blogging with, 571
- properties, 567-570
- snippet, 565-571
- tag, 196
- tutorial, 566-567
- uses for, 565, 571

getResponse() method, 643
getSelectColumns() method, 393
getService() method, 412
getSessionState() method, 644
getSettings() method, 644, 650
getTableName() method, 634
getTemplateVar() method, 644
getTemplateVarOutput() method, 644
getTemplateVars() method, 644
getting
- children of current resource, 360-361
- children of current resource in code, 315
- chunks from database, 311
- current resource in code, 312-313, 318, 327-328
- current resource related objects, 388
- current user, 381-383
- data from a chunk tutorial, 238-241
- ID of current resource, 315, 327-328
- MODX version information in code, 645
- properties in code, 365-368
- related objects, 359-361
- resources from database, 311
- system settings in Evolution, 416
- system settings with `getOption()`, 380
- template variables from database, 377-379
- TV values for current resource in code, 331, 376
- user extended fields from database, 397
- user groups from database, 406-408
- user information in code, 380-396
- user profile with `getCollection()`, 393
- user profile with `getObjectGraph()`, 322-325, 388-389
- user profile with `getOne()`, 382, 392
- user profiles from database, 382-383
- users from database, 311, 392-393

getTree() method, 644
getTVValue() method, 378
getUser() method, 644
getUserDocGroups() method, 644
getUserGroupNames() method, 650
getUserGroups() method, 650
getUserInfo() method, 645
getValue() method, 330, 652
getVersionData() method, 645
getWebUserInfo() method, 645
Git
- `--ff-only` merges, 81, 88
- Bash, 73
- branches, 70-72
- bug fixes in, 84
- checkout, 80, 88
- client, 72
- cloning MODX development version, 73-75
- commits, 71
- contributing a new feature, 86
- contributor workflow, 83-89
- creating a remotes, 80
- deleting branches, 82
- deleting remotes branches, 82
- develop branch, 71
- development version of Revolution, 75-89
- editing commit messages, 85
- feature branches, 71
- fetch, 80-81, 88
- GUI, 73
- guidelines for using with MODX, 83-84
- HEAD, 71
- hotfix branch, 71
- installing, 73
- master branch, 71
- merge, 81, 88
- origin remote, 80
- push, 82
- release branches, 72
- remotes, 80
- Revolution repository, 72
- `SHA1`, 71
- switching branches, 72
- tracking branches, 80
- upstream remote, 80
- workflow, 83-86

Git for transport packages, 604-605
GitHub, 70-75
- cloning a fork of MODX, 78-79
- creating an account, 76-78
- creating an SSH key, 77
- forking MODX, 78-79

Gmail (using SMTP with), 585-586, 696
graph methods, 321-325
greaterthanorequalto (output modifier), 168
gt (output modifier), 168
gte (output modifier), 168
guidelines for Git with MODX, 83-84

H

handler, 526
handler for menu items, 503-504
handleRequest() method, 645
handling clicks in the Top Menu, 507
hasChildren() method, 329, 356-357
hasPermission() method, 485, 488, 645
hasSessionContext() method, 384-385, 469, 481, 490, 650
hasTemplate() method, 331
hasTemplateVar() method, 331
HEAD (Git), 71
hello snippet, 269-270
hello snippet (enhanced), 273-275
Hello World tutorial (PHP), 653
Hello2 snippet, 605
help, 122
- API documentation link, 129
- MODX, 122
- MODX API documentation link,

630
MODX documentation, 122
MODX Wiki, 128
MySQL, 122
PHP, 122, 652
searching MODX forums, 127-128
xPDO, 122
hidden template variable input type, 211
HiddenTVs resource group, 482, 484
`hide` (output modifier), 168
Hide From Menus checkbox, 145
Hide From Menus checkbox (and Wayfinder), 560
`hidemenu` (Hide From Menus) resource field, 145
HidePages snippet tutorial, 469
hiding
Create/Edit Resource panel fields with form customization tutorial, 522
Create/Edit Resource panel tabs with form customization tutorial, 520
elements tutorial, 461-463
pages with a snippet tutorial, 468-470
resources from menus, 145
resources in the Manager, 438-439
resources with ACL entries tutorial, 470-474
resources with custom permissions tutorial, 485-488
tabs with form customization, 520
template variables in the front end tutorial, 484-485
template variables in the Manager with ACLs tutorial, 481-484
template variables with form customization, 514
template variables with form customization tutorial, 520
Top Menu items, 510-512
history of MODX, vi-viii
home page, 48
previewing, 111
Home screen of the Manager, 109
Home Top Menu choice, 109
hooks
built-in for FormIt, 579-582
custom for FormIt, 588-594
email for FormIt, 581
FormIt custom preHooks, 591-593
FormIt `setValue()` method, 592
math for FormIt, 579-580
properties for FormIt email hook, 587
properties for FormIt spam hook, 587
redirect for FormIt, 581
`setValue()` method (FormIt), 592
variables available in FormIt custom hooks, 590
hooks and preHooks for FormIt, 579-582
host (database), 42
hosts file, 430
hotfix branch (Git), 71
how MODX delivers a web page, 21
how MODX works, 1
`ht.access` file, 46-47
`htdocs` directory, 63
HTML comments, 531
`HTML` constant, 635
HTML tag template variable output type, 216
htmlarea template variable input type, 211
`htmlent` (output modifier), 170
`htmlentities` (output modifier), 170
http URL in `makeUrl()` method, 646
`http://api.modx.com`, 630
`http://modx.com/extras`, 537
`http_host` context setting, 429, 431
`http_host` system setting, 429
`http_referer` variable, 550
`httpd-vhosts.conf` file, 430-431
`httpd.conf` file, 60
https URL in `makeUrl()` method, 646

I

icons
Element tree, 104
File tree, 107
Resource tree, 101
id (resource ID) field, 340
`id` (Resource ID) resource field, 139
id user field, 381
`if` (output modifier), 168
`if` statement in PHP, 664-666
`ifempty` (output modifier), 172
`ifnotempty` (output modifier), 172
image template variable input type, 211-212
image template variable output type, 217
`implode()` PHP function, 634, 678
importing
HTML resources, 118
property sets, 261
resources, 118
`inc` (output modifier), 172
`increment` (output modifier), 172
`index.php` file
components, 505-506
for CheatSheet component, 525, 527
MODX, 38, 56, 61, 142, 165
separate contexts, 426-428
info chunk tutorial, 233-235
Info panel, 111
`initialize()` method (contexts), 426-427
`initialize()` method (MODX), 600-601, 645
`innerJoin()` method, 393
InnoDB storage engine, 405
`input` (output modifier), 168
input filters, 176
input option values TV field, 206
input type TV field, 206
input types for TVs, 208-213
`insideManager()` method, 645
`install.script.php` file (transport packages), 616-620
installer still present warning, 47
installing
add-on components, 52-54, 536-538
add-on components in Evolution, 595
add-on components with no transport package, 538
BreadCrumbs, 552
Evolution add-on components, 96
EZfaq, 539-540
Git, 73
packages, 52-54
snippets in Evolution, 306
snippets with Package Manager, 268
transport packages, 54
transport packages manually, 537-538
XAMPP, 62-63
installing MODX Evolution, 42, 96
add-on components, 96
configuration check, 96

development version, 97
file and directory permissions, 95
FURLs, 96, 97
installing MODX Revolution, 35
config file, 41
configuration check (Revolution), 45
connection information, 42
database host, 42
database name, 42
file placement, 38-39
installer still present warning, 47
localhost, 64
`register_globals`, 45-47
running setup, 41
summary screen, 43
under XAMPP, 64
instanceof in PHP, 337-338
integer variables in PHP, 655
interacting with the user in transport packages, 616-620
internationalization for transport packages, 611-612
intersect objects
modElementPropertySet, 409
modPluginEvent, 408
modResourceGroupResource, 408
modTemplateVarTemplate, 403
modUserGroupMember, 406
introduction, i-xiii
`introtext` (Summary) resource field, 143
`invokeEvent()` method, 301, 645, 698
invoking system events, 289-290
`is` (output modifier), 168
is MODX for you?, vi-vii
is user logged in?, 384-385
`is_array()` PHP function, 335
`is_object()` PHP function, 337
`isAuthenticated()` method, 650
`isBackend()` method, 645
`isempty` (output modifier), 172
`isequal` (output modifier), 168
`isequalto` (output modifier), 168
`isfolder` (Container) resource field, 146, 356
`isFrontend()` method, 645
`isgt` (output modifier), 168
`isgte` (output modifier), 168
`islessthan` (output modifier), 168
`isloggedin` (output modifier), 173
`islowerthan` (output modifier), 168
`islte` (output modifier), 168
`ismember` (output modifier), 169
`isMember()` method, 386-387, 650
`isMemberOfWebGroup()` method, 646
`isNew()` method, 639
`isnot` (output modifier), 168
`isnotempty` (output modifier), 172
`isnotloggedin` (output modifier), 173
`isnt` (output modifier), 168
`isset()` PHP function, 334-335
ItemTpl chunk, 242, 284

J
JavaScript, 526, 643, 647
with Wayfinder, 559
`joinGroup()` method, 408
`joinGroup()` method (resource object), 332
`joinGroup()` method (user object), 332, 650
JSON, 395, 631-632

K
K&R style (PHP), 672-673
key, 247
Kongondo, 555

L
language (see: lexicon)
language files (see: lexicon files)
language strings (see: lexicon strings)
language tags, 163-164
languages, 349
`lastInsertId()` method, 634
`lcase` (output modifier), 169
`le` (output modifier), 168
`leaveGroup()` method (resource object), 332
`leaveGroup()` method (user object), 332, 650
`len` (output modifier), 170
`length` (output modifier), 170
`lessthan` (output modifier), 168
`lessthanorequalto` (output modifier), 168
lexicon
core, 504-505
`default.inc.php` file, 348
for CheatSheet component, 526-527
for Top Menu items, 504-505, 508
in Evolution, 32
overview, 23-24
SPForm, 548
using in snippets tutorial, 411-413
working with in code, 347-351, 411-413
lexicon files
creating, 412
`default.inc.php`, 348
for transport packages, 603, 611-612
loading in code, 412
location of, 348, 412
SPForm, 548
lexicon keys
for CheatSheet component, 527
for Top Menu items, 504-505, 508
Lexicon Management, 124-125
lexicon strings
displaying, 17
displaying with code, 413
in code, 347-351, 413
in Evolution, 416
overview, 23-24
prefixes, 350
uses for, 347
lexicon topics, 23-24, 349-351, 506
lexicon topics for custom manager pages, 524
`lexicon()` method, 646
`lexicon->load()` method, 350, 412
`lexicon_entries` table, 348
`limit` (output modifier), 171
limiting access to elements tutorial, 461-463
limiting permissions in the Manager tutorial, 464-467
link tag properties, 163
link tags, 16, 161-163, 197
`context` property, 163
`scheme` property, 163
`link_attributes` (Link Attributes) resource field, 143
links, 16
bug report, 129
displaying, 17
MODX API documentation, 129, 630
MODX forums, 127-128
MODX wiki, 128
`list` permission, 459
listbox (multi-select) template variable

input type, 212
listbox (single-select) template variable input type, 212
ListData chunk, 242, 284
load headers, 506, 514
Load List and View policy, 444, 459
Load Only policy, 443, 475
`load` permission, 449, 459, 475
Load policy, 471
`loadClass()` method, 606, 608, 609, 634-635
loading lexicon topics, 350
localhost default database, 42
location
 of `config.core.php` files, 420
 of `config.inc.php` file, 41
 of Evolution config file, 95
 of lexicon files, 348, 412
 of properties in database, 250
 of property sets in database, 250
 of Revolution config file, 41
lock variable for editing TV field, 206
locks, 112
`log()` method, 635
`log()` method (transport packages), 617
`log()` method options, 635
`logEvent()` method, 646
logged-in users list, 111
logging out, 113
logical operators in PHP, 663-664
Login page
 creating, 472-473
 `loginResourceId` property, 474
 tutorial, 472-473
Login snippet, 472-473
login status, 384-385, 469
login tests, 302
`loginResourceId` property (Login), 474
`logManagerAction()` method, 646
`longtitle` (Long Title) resource field, 142
`lowercase` (output modifier), 169
`lowerthan` (output modifier), 168
`lte` (output modifier), 168
`ltrim()` PHP function, 676

M

`mail()` PHP function, 550, 696
`makeUrl()` method, 481, 488, 490, 646
Manage Users menu, 114
Manager, 99-132
 configuration check (Revolution), 110
 customizing, 501-533
 Element tree, 104-106
 File tree, 106-109
 Home screen, 109
 Resource tree, 100-104
 trees, 99-109
manager access system events, 700-713
manager action policy, 444-445
manager action rules, 449
manager actions, 126, 502-508
Manager Actions Report, 121
`manager` directory, 420
Manager menu (see Top Menu)
manager users, 440-441, 446-447
manager users in Evolution, 497-498
`manager_language` system setting, 612
`manager_language` user setting, 447
`manager_url` system setting, 531
manually installing transport packages, 537-538
many-to-many relationships (see: intersect objects)
master branch (Git), 71
`math` (output modifier), 172
math hook for FormIt, 579-580
`md5`, 342
`md5` (output modifier), 173
member pages with `tree_root_id` tutorial, 489-496
member role, 475
Member template, 490-491
member variables in PHP, 683
`memberof` (output modifier), 169
MemberPages plugin, 492-493
MemberPages snippet, 490-491
Members resource, 489
menu (see Top Menu)
menu item action, 503-508
menu item handler, 503-504
menu tree, 503-513
Menu Update dialog, 503
menuindex (and Resource tree sorting), 101
`menuindex` (Menu Index) resource field, 145
menus (custom), 513
`menutitle` (Menu Title) resource field, 144
merge (Git), 81, 88
`mergeChunkContent()` method, 646
`mergeDocumentContent()` method, 646
`mergePlaceholderContent()` method, 646
`mergeSettingsContent()` method, 646
`messageQuit()` method, 647
methods
 `addEventListener()`, 640
 `addMany()`, 342, 402, 621, 637
 `addOne()`, 341-342, 403, 404, 406, 407, 408, 409, 637
 `addPackage()`, 630
 `beginTransaction()`, 405, 630
 `changePassword()`, 640, 650
 `changeWebUserPassword()`, 640
 `checkPreview()`, 640
 `checkSession()`, 640
 `cleanDocumentIdentifier()`, 640
 `commit()`, 406, 630
 `connect()`, 631
 `createPackage()` (transport packages), 606, 608
 `createVehicle()` (transport packages), 606, 610
 `errorCode()`, 631
 `errorInfo()`, 631
 `exec()`, 631
 `fromArray()`, 340-341, 638
 `fromJSON()`, 638
 generic, 310-314
 `get()`, 327, 355, 391, 629, 638
 `getActiveChildren()`, 640
 `getAggregates()`, 631
 `getAllChildren()`, 640
 `getAncestry()`, 631
 `getAttribute()`, 631
 `getAuthenticatedUser()`, 640-641
 `getCacheManager()`, 631, 641
 `getCachePath()`, 631
 `getChildIds()`, 315, 357, 641
 `getChunk()`, 237, 317, 368-370, 389, 488, 528, 642
 `getCollection()`, 313, 362-363, 373, 390, 392-393, 403, 407, 631-632
 `getCollectionGraph()`, 324-325, 364-365, 391, 397, 632

getComposites(), 631-632
getConfig(), 642
getContent(), 328, 651
getContext(), 642
getCount(), 633
getDebug(), 633
getDebugBacktrace(), 633
getDocGroups(), 642
getDocument(), 642
getDocumentChildren(), 642
getDocumentChildrenTVars(), 642
getDocumentIdentifier(), 642
getDocumentMethod(), 642
getDocumentObject(), 642
getDocuments(), 642
getEventMap(), 642
getFields(), 633
getFullTableName(), 642
getIterator(), 313, 633
getKeywords(), 642
getLoginUserID(), 642
getLoginUserName(), 643
getLoginUserType(), 643
getManager(), 633
getManagerPath(), 643
getMany(), 321, 359-361, 373, 638
getMETATags(), 643
getMicroTime(), 633
getObject(), 310-313, 354, 390, 629, 633
getObjectGraph(), 321-324, 363-364, 384, 388, 396, 633-634
getOne(), 320, 358-360, 384, 388, 390, 392, 639
getOption(), 314, 367-368, 380, 490, 527, 634
getPackage(), 634
getPageInfo(), 643
getParent(), 643
getParentIds(), 315-316, 358, 643
getParser(), 643
getPK(), 634
getPKType(), 634
getPlaceholder(), 316, 643
getProperties(), 370
getRegisteredClientScripts(), 643
getRegisteredClientStart-upScripts(), 643
getRequest(), 643
getResponse(), 643
getSelectColumns(), 393
getService(), 412
getSessionState(), 644
getSettings(), 644, 650
getTableName(), 634
getTemplateVar(), 644
getTemplateVarOutput(), 644
getTemplateVars(), 644
getTree(), 644
getTVValue(), 378
getUser(), 644
getUserDocGroups(), 644
getUserGroupNames(), 650
getUserGroups(), 650
getUserInfo(), 645
getValue(), 330, 652
getVersionData(), 645
getWebUserInfo(), 645
graph, 321-325
handleRequest(), 645
hasChildren(), 329, 356-357
hasPermission(), 485, 488, 645
hasSessionContext(), 384-385, 469, 481, 490, 650
hasTemplate(), 331
hasTemplateVar(), 331
initialize() (contexts), 426-427
initialize() (MODX), 600-601, 645
innerJoin(), 393
insideManager(), 645
invokeEvent(), 301, 645, 698
isAuthenticated(), 650
isBackend(), 645
isFrontend(), 645
isMember(), 386-387, 650
isMemberOfWebGroup(), 646
isNew(), 639
joinGroup(), 408
joinGroup() (user object), 332, 650
lastInsertId(), 634
leaveGroup() (resource object), 332
leaveGroup() (user object), 332, 650
lexicon(), 646
lexicon->load(), 350, 412
loadClass(), 606, 608, 609, 634-635
log(), 635
log() (transport packages), 617
logEvent(), 646
logManagerAction(), 646
makeUrl(), 481, 488, 490, 646
mergeChunkContent(), 646
mergeDocumentContent(), 646
mergePlaceholderContent(), 646
mergeSettingsContent(), 646
messageQuit(), 647
modUser class, 650-651
modX class, 640-649
newObject(), 339, 397, 398, 399, 400, 403, 404, 406, 407, 408, 409, 635
newObject() (transport packages), 606
newQuery(), 313, 393, 631-632, 636
object, 326-332
parseChunk(), 647
parseSchema(), 410
prepare(), 636
process(), 328-329
processElementTags(), 646
putChunk(), 647
putVehicle() (transport packages), 606
putVehicle() (transport packages), 610
query(), 636
reg*(), 647
regClientCSS(), 378, 527, 647
regClientHTMLBlock(), 647
regClientScript(), 647
regClientStartupHTMLBlock(), 647
regClientStartupScript(), 647
registerNamespace() (transport packages), 606, 608
related object, 317-319
reloadConfig(), 647
remove(), 344, 639
remove() (user object), 651
removeAllEventListener(), 648
removeCollection(), 636
removeEventListener(), 648
removeObject(), 636
removeObject() (transport packages), 618
renderOutput(), 330-331, 652
resolve() (transport packages), 612-614
rollback(), 405, 636
runSnippet(), 373-375, 648
save(), 339, 343-344, 397, 398, 400, 404, 639
save() (user object), 651
sendEmail(), 651
sendError(), 648
sendErrorPage(), 469, 648
sendForward(), 302, 648

sendRedirect(), 488, 648
sendUnauthorizedPage(), 469, 490, 648
set(), 340, 397, 399, 404, 639
set() (transport packages), 606
setAttribute(), 636
setContent(), 340, 399, 400, 651
setContent() (transport packages), 606
setDebug(), 636, 648
setLogLevel(), 636
setLogLevel() (transport packages), 625
setLogTarget(), 636
setLogTarget() (transport packages), 625
setOption(), 637
setPackageAttributes() (transport packages), 615
setPlaceholder(), 344-347, 591, 593, 648-649
setPlaceholders(), 346-347, 649
setService(), 643
setValue() (FormIt hooks), 592
setValue() (TV), 404, 414
sortby() for xPDO queries, 631-632
stripTags(), 649
switchContext(), 649
toArray(), 328, 355, 370-373, 384, 639
toJSON(), 639
toPlaceholder(), 346, 649
toPlaceholders(), 346-347, 384, 388-389, 649
unsetPlaceholder(), 347, 649
unsetPlaceholders(), 347, 649
userLoggedIn(), 649
where() for xPDO queries, 631-632
writeSchema(), 410
xPDO class, 630-637

mgr context, 25-26, 424-425, 450-451, 456, 458, 459, 460, 483
mgr context permissions, 441
MIME types, 125
minimum role
for context access ACL entries, 450-451
for element category access ACL entries, 453
for resource group access ACL entries, 452, 458-459
mo (output modifier), 169
mod (output modifier), 172
mod_rewrite, 60
modChunk object reference, 691
MODE_NEW constant, 299, 699
MODE_UPD constant, 299
MODE_UPS constant, 699
modElementPropertySet intersect object, 409
MODExt, 532-533
modifying
(see also: changing and renaming)
MODX objects in code, 338-344
the Top Menu tutorial, 507-512
modPlugin object reference, 692-693
modPluginEvent intersect object, 408
modPrincipal object, 319
modResource object reference, 691
modResourceGroupResource intersect object, 408
modresponse.class.php file, 290
modSnippet object reference, 692
modSystemEvent::MODE_NEW constant, 299, 699
modSystemEvent::MODE_UPD constant, 299, 699
modTemplate object reference, 693
modTemplateVar object reference, 693-694
modTemplateVarTemplate intersect object, 403
modules
in Evolution, 533
manager system events (Evolution), 712
modulus (output modifier), 172
modUser class, 650-651
modUser object, 381
modUser object reference, 694
modUserGroup object, 318-319
modUserGroupMember intersect object, 319, 406
modUserProfile object reference, 695
MODX
.htaccess file, 426
about, vi-xiii
About Report, 122
add-on components (overview), 4
Advanced distribution, 420-421
API documentation link, 129, 630
API reference, 629-652
architecture, x-xiii
assets directory, 20
back end, 5
BaseTemplate, 49
basic building blocks, 6-20
browser support, 36-37
build.config.php file (Revolution development version), 87
building the development version of Revolution, 87-88
cache (overview), 28-29
cache control, xiii
categories defined, 14
chunk object reference, 692
chunks, 229-247
chunks defined, 11
class methods, 640-649
common add-on components, 535-594
components (overview), 4
config file, 41, 420-423
configuration check (Evolution), 96
configuration check (Revolution), 45
contexts, 424-434
contexts defined, 25-26
contributing to development, 75-89
core directory, 20, 349, 411, 425-426, 433, 524
core directory (moving), 421-422
core independence in MODX, xii-xiii
core/packages directory, 537
creating contexts, 25-26
creating resources, 6
credits, 122
customizing the Manager, 501-533
database, 1, 40
database creation, 40
database host, 42
database name, 42
database password, 40
database username, 40
deleting resources, 102-103
delivering a web page, 21
design philosophy, xi
documentation link, 122
documents defined, 7
downloading, 38
Element tree, 104-106
error log, 121, 690
error page, 50
event model, 300-301
Evolution, viii
Evolution tags, 17
file and directory permissions, 41
files, 19
forums (searching), 127-128

forums link, 127-128
front end, 5
get*() methods, 310-327
getting version information in code, 645
help, 122
history, vi-viii
home page, 48
index.php file, 38, 56, 61, 142, 165
index.php file (separate contexts), 426-428
initialize() method, 600-601
installing, 35
is it for you?, vi-vii
key features, x-xiii
lexicon (overview), 23-24
lexicon strings, 23-24
links, 16
Manager, 99-132
Manager trees, 99-109
menu (see Top Menu),
mgr context, 25-26, 424-425
moving site, 64-70
multiple sites sharing core, 433
namespaces defined, 23
News panel, 110
object-oriented design, xii
objects reference, 690-695
parser, 21
placeholders defined, 15
plugin object reference, 692-693
plugins defined, 13
porting an existing site to, 89-94
processors, 409
property sets defined, 27-28
related objects, 359-361
resource object reference, 691
resources, 7, 135-183
Revolution, viii-ix
Revolution development version, 70-89
Revolution objects, 680-682
Revolution tags, 17-19
robustness, xiii
root directory, 420
RSS feeds, 110
running setup, 41
schema, 318-319
Security panel, 110
security system, 437-496
security system elements, 442-453
server requirements, 36-37
setup, 41
sharing core between sites, 425-426, 433
sharing database between sites, 425
site name, 49
site organization, 419-423
snippet object reference, 692
snippet properties defined, 12, 12
snippets defined, 12-13
speed and efficiency, xii
supported browsers, 36-37
system events, 289-290
system events Quick Reference, 696-715
system settings defined, 25
table prefix, 425
tag syntax specification, 19
tags, 17-19, 49, 145-146, 154-176
tags in documents, 154-167
template object reference, 693
template variable object reference, 693-694
template variables defined, 10
templates (overview), 10
Top Menu, 109-128
Tpl chunks defined, 12
Traditional distribution, 420-423
transport packages defined, 22
unauthorized page, 50
user interface, xi
user object reference, 694
user profile object reference, 695
user settings, 26
versions, vii-x
web context, 25-26, 424-425
Wiki link, 128
workspaces, 25-26

MODX Evolution, viii
cache, 33
cache control, 187
chunk tags, 264-265
common snippets, 595-596
config file, 95
content elements, 31
contexts, 32, 435
custom manager pages, 533
DBAPI, 416
default properties, 306
development version, 97
documentObject, 307
documents, 185
editing chunks, 264-265
editing elements, 130, 184
editing files, 130
elements, 31
file and directory permissions, 95
form customization, 533
getChildIds() method, 416
getChunk() method, 416
getLoginUserID() method, 307
getLoginUserName() method, 307
getParentIds() method, 416
getting system settings, 416
installing, 42, 96
installing add-on components, 96, 595
installing snippets, 306
lexicon, 32
lexicon strings, 416
manager users, 497-498
missing Manager functions in, 133
modules, 533
output modifiers, 187
parseChunk() method, 416
PHX, 187
placeholders, 306
plugins, 307
properties, 32, 186, 265, 306
Resource tree, 184
roles, 446, 498
runSnippet() method, 416
security system, 497-498
snippet properties, 595
snippets, 306
system settings, 32
tags, 306
templates, 185
Top Menu, 130-132
transport packages, 626
user information in, 307
user settings, 32
userLoggedIn() method, 307
using SMTP, 596
web users, 497-498

MODX Manager, 99-132
configuration check, 110
customizing, 501-533
Element tree, 104-106
File tree, 106-109
Home screen, 109
trees, 99-109

MODX Motors resources, 557-558
modX object, 310, 699
MODX objects (creating in code), 338-344, 397-409
MODX Objects reference, 690-695
modx.class.php file, 601

modx.mysql.schema.php file, 318-319
modx.user.id placeholder, 381
modx.user.username placeholder, 381
MODX: The Official Guide
- audience for, ii
- introduction to, i-xiii
- organization of, iii
- typographical conventions, iii-v

modX::LOG_LEVEL_INFO constant, 625
modX::SESSION_STATE_EXTERNAL constant, 644
modX::SESSION_STATE_INITIALIZED constant, 644
modX::SESSION_STATE_UNAVAILABLE constant, 644
modX::SESSION_STATE_UNINITIALIZED constant, 644
modx_actions database table, 506
MODX_ASSETS_PATH constant, 603, 614
MODX_ASSETS_URL constant, 603
modx_base_path system setting, 642
modx_base_url system setting, 642
modx_charset system setting, 192
MODX_CONFIG_KEY constant, 601
MODX_CORE_PATH constant, 601, 603, 630, 634
modx_core_path system setting, 642
MODX_PROCESSORS_PATH constant, 422
moving
- a site to MODX, 89-94
- local site to remote server, 64-70
- MODX manager directory, 423
- the MODX core directory, 421-422
- TV to a new tab with form customization tutorial, 520

mpy (output modifier), 172
multi-language sites, 434
multiple security permissions, 440
multiply (output modifier), 172
my-new-tab, 522
my_allow_private permission, 486-487
my_view_user permission, 510
MyComponent example transport package build script, 625
MyISAM storage engine, 405
MyMenu custom menu item, 509
MySQL help, 122
mysysgit, 73

N
name of MODX database, 42
namespace path, 349
namespace path (Batcher), 505
namespaces, 23, 411, 506
- cheatsheet, 524
- creating, 126
- defined, 23
- for components, 349

ne (output modifier), 168
neq (output modifier), 168
nested tags, 18, 164-165, 197
New Category dialog, 42
new feature in Git, 86
new MODX objects in code, 339
New Property Set dialog, 119
New Role button, 115
new_document_in_root permission, 439
new_user permission, 449
newObject() method, 339, 397, 398, 399, 400, 403, 404, 406, 407, 408, 409, 635
newObject() method (transport packages), 606, 621
newQuery() method, 313, 393, 631-632, 636
News resource group, 459
news RSS feed, 110
news summary with getResources tutorial, 566-567
NightErrorPage resource, 303
nl2br (output modifier), 171
notags (output modifier), 170
notempty (output modifier), 172
notequals (output modifier), 168
notequalto (output modifier), 168
null variable in PHP, 655
number template variable input type, 212

O
object methods, 326-332
object policies, 444-445
object policies (editing), 465-466
Object policy, 459
object variables in PHP, 655
object-oriented design of MODX, xiii
object-oriented programming in PHP, 679-686
ObjectDemo tutorial, 353
objects in PHP, 679-686
OnBeforeCacheUpdate system event, 715
OnBeforeChunkFormDelete system event, 706
OnBeforeChunkFormSave system event, 297, 705
OnBeforeDocFormDelete system event, 704, 704
OnBeforeDocFormSave system event, 296, 703
OnBeforeEmptyTrash system event, 705
OnBeforeManagerLogin system event, 301, 700
OnBeforeManagerLogout system event, 701
OnBeforeManagerPageInit system event, 701
OnBeforeModFormDelete system event, 712
OnBeforeModFormSave system event, 712
OnBeforePluginFormDelete system event, 708
OnBeforePluginFormSave system event, 707
OnBeforeResourceGroupRemove system event, 705
OnBeforeSaveWebPageCache system event, 715
OnBeforeSnipFormDelete system event, 707
OnBeforeSnipFormSave system event, 706
OnBeforeTempFormDelete system event, 709
OnBeforeTempFormSave system event, 708
OnBeforeTVFormDelete system event, 710
OnBeforeTVFormSave system event, 709
OnBeforeUserActivate system event, 711
OnBeforeUserFormDelete system event, 702, 711
OnBeforeUserFormSave system event, 702, 710
OnBeforeWebLogin system event,

714
OnBeforeWebLogout system event, 715
OnBeforeWUserFormDelete system event, 711
OnBeforeWUserFormSave system event, 711
OnCacheUpdate system event, 715
OnCategoryBeforeRemove system event, 712
OnCategoryBeforeSave system event, 712
OnCategoryRemove system event, 712
OnCategorySave system event, 712
OnChunkBeforeSave system event, 706
OnChunkFormDelete system event, 706
OnChunkFormPrerender system event, 705
OnChunkFormRender system event, 705
OnChunkFormSave system event, 706
OnChunkRemove system event, 706
OnChunkSave system event, 706
OnContextBeforeRemove system event, 713
OnContextBeforeSave system event, 713
OnContextFormPrerender system event, 713
OnContextFormRender system event, 713
OnContextRemove system event, 713
OnContextSave system event, 713
OnContextUpdate system event, 713
OnCreateDocGroup system event, 704
OnDocFormDelete system event, 704
OnDocFormPrerender system event, 703
OnDocFormRender system event, 703
OnDocFormSave system event, 703
OnDocPublished system event, 703
OnDocUnPublished system event, 703
OnEmptyTrash system event, 705
OnFileManagerUpload, 702
OnFriendlyURLSettingsRender system event, 713
OnHandleRequest system event, 432, 714
OnInterfaceSettingsRender system event, 713
Online panel, 111
OnLoadWebDocument system event, 714
OnLoadWebPageCache system event, 715
OnLogPageHit system event, 715
OnManagerAuthentication system event, 700
OnManagerChangePassword system event, 701
OnManagerCreateGroup system event, 701
OnManagerDeleteUser system event, 702, 711
OnManagerLogin system event, 700
OnManagerLoginFormPrerender system event, 700
OnManagerLoginFormRender system event, 301, 700
OnManagerLogout system event, 701
OnManagerPageInit system event, 701
OnManagerSaveUser system event, 702, 710
OnMiscSettingsRender, 713
OnModFormDelete system event, 712
OnModFormPrerender system event, 712
OnModFormRender system event, 712
OnModFormSave system event, 712
OnPageNotFound system event, 302, 304, 715
OnPageUnauthorized system event, 714-715
OnParseDocument system event, 714
OnPluginBeforeRemove system event, 708
OnPluginBeforeSave system event, 707
OnPluginEventBeforeRemove system event, 708
OnPluginEventRemove system event, 708
OnPluginFormDelete system event, 708
OnPluginFormPrerender system event, 707
OnPluginFormRender system event, 707
OnPluginFormSave system event, 708
OnPluginRemove system event, 708
OnPluginSave system event, 707
OnResourceBeforeSort system event, 704
OnResourceDuplicate system event, 704
OnResourceGroupBeforeSave system event, 705
OnResourceGroupRemove system event, 705
OnResourceGroupSave system event, 705
OnResourceSort system event, 704
OnResourceToolbarLoad system event, 704
OnResourceTVFormPrerender system event, 704
OnResourceTVFormRender system event, 704
OnResourceUndelete system event, 704
OnRichTextBrowserInit system event, 712
OnRichTextEditorInit system event, 712
OnRichTextEditorRegister system event, 712
OnSiteRefresh system event, 715
OnSiteSettingsRender, 713
OnSnipFormDelete system event, 707
OnSnipFormPrerender system event, 706
OnSnipFormRender system event, 706
OnSnipFormSave system event, 707
OnSnippetBeforeRemove system event, 707

OnSnippetBeforeSave system event, 706
OnSnippetRemove system event, 707
OnSnippetSave system event, 707
OnTempFormDelete system event, 709
OnTempFormPrerender system event, 708
OnTempFormRender system event, 708
OnTempFormSave system event, 709
OnTemplateBeforeRemove system event, 709
OnTemplateBeforeSave system event, 708
OnTemplateRemove system event, 709
OnTemplateSave system event, 709
OnTemplateVarBeforeRemove system event, 710
OnTemplateVarBeforeSave system event, 709
OnTemplateVarRemove system event, 710
OnTemplateVarSave system event, 710
OnTVFormDelete system event, 710
OnTVFormPrerender system event, 709
OnTVFormRender system event, 709
OnTVFormSave system event, 710
OnUserActivate system event, 711
OnUserBeforeRemove system event, 702, 711
OnUserBeforeSave system event, 702, 711
OnUserChangePassword system event, 701
OnUserFormDelete system event, 702, 711
OnUserFormPrerender system event, 710
OnUserFormRender system event, 493, 710
OnUserFormSave system event, 493, 702, 711
OnUserGroupBeforeRemove system event, 701
OnUserGroupBeforeSave system event, 701
OnUserGroupRemove system event, 701
OnUserGroupSave system event, 701
OnUserNotFound system event, 700
OnUserRemove system event, 702, 711
OnUserSave system event, 702, 711
OnUserSettingsRender system event, 713
OnWebAuthentication system event, 700
OnWebAuthentification system event, 714
OnWebChangePassword system event, 714
OnWebCreateGroup system event, 714
OnWebDeleteUser system event, 714
OnWebLogin system event, 714
OnWebLogout system event, 715
OnWebPageCompete system event, 714
OnWebPageInit system event, 714
OnWebPagePrerender system event, 290, 292, 714
OnWebSaveUser system event, 714
OnWUserFormDelete system event, 711
OnWUserFormPrerender system event, 711
OnWUsrFormRender system event, 711
OnWUsrFormSave system event, 711
OOP in PHP, 679-686
operators
- = (PHP), 661
- =& (PHP), 312, 354, 388
- === (PHP), 336
- arithmetic (PHP), 660
- assignment (PHP), 661
- comparison (PHP), 661-662
- logical (PHP), 663-664
- PHP, 272-273, 659-664
- ternary (PHP), 281-282, 667-668

optimize database tables, 121
organization of the book, iii
origin remote (Git), 80
OuterTpl chunk, 242, 284
output modifiers
- !empty, 172
- add, 172
- and, 168
- cat, 169
- cdata, 171
- Conditional, 167-169
- Custom, 174-176
- date, 172
- decr, 172
- decrement, 172
- default, 172, 567
- div, 172
- divide, 172
- eg, 168
- el, 168
- ellipsis, 171, 567
- else, 168
- empty, 172
- eq, 168
- equalorgreatherthan, 168
- equals, 168
- equalto, 168
- equaltoorlessthan, 168
- esc, 170
- escape, 170
- Evolution (PHX), 187
- ge, 168
- greaterthanorequalto, 168
- gt, 168
- gte, 168
- hide, 168
- htmlent, 170
- htmlentities, 170
- if, 168
- ifempty, 172
- ifnotempty, 172
- inc, 172
- increment, 172
- input, 168
- is, 168
- isempty, 172
- isequal, 168
- isequalto, 168
- isgt, 168
- isgte, 168
- islessthan, 168
- isloggedin, 173
- islowerthan, 168
- islte, 168
- ismember, 169

isnot, 168
isnotempty, 172
isnotloggedin, 173
isnt, 168
lcase, 169
le, 168
len, 170
length, 170
lessthan, 168
lessthanorequalto, 168
limit, 171
lowercase, 169
lowerthan, 168
lte, 168
math, 172
md5, 173
memberof, 169
mo, 169
mod, 172
modulus, 172
mpy, 172
multiply, 172
ne, 168
neq, 168
nl2br, 171
notags, 170
notempty, 172
notequals, 168
notequalto, 168
reverse, 171
select, 168
show, 168
String, 169-171
strip, 170
strip_tags, 170
striptags, 170
strlen, 170
strrev, 171
strtolower, 169
strtotime, 173, 567
strtoupper, 170
subtract, 172
tag, 171
then, 168
ucase, 170
ucfirst, 170
ucwords, 170
uppercase, 170
use with properties, 174
userinfo, 173, 566
wordwrap, 171
wordwrapcut, 171

output modifiers Reference, 167-175
output type TV field, 208
output types for template variables, 214-218
overhead of database tables, 121
overriding default properties, 256
overriding properties, 276

P

pack() method (transport packages), 607
package attributes for transport packages, 615
Package Manager, 22, 52-54, 122-123
add package, 537
cURL and, 536
installing snippets, 268
Search Locally for Packages, 53, 123, 537, 610
troubleshooting, 536
packages (see: transport packages)
PackMan add-on component, 600
page count snippet tutorial, 413-414
page settings, 136
page-not-found page, 50
page-not-found plugin, 302-305
page-not-found snippet, 302-304
PageNotFound tutorial, 302-305
pagetitle (Resource Title) resource field, 141
parameters (see also: properties)
parameters (with $_GET), 646
parent (Resource Parent) resource field, 144, 358
Parent related object, 318, 321, 363
parseChunk() method, 647
parseChunk() method in Evolution, 416
parser, 21
parseSchema() method, 410
password user field, 342, 381, 640, 650
passwords in code, 342
pasting code into in PHP, 659
path to namespace, 349
PBKDF2, 342
Peoples add-on component, 392
permissions, 443
access_permissions, 465
add_children, 459
allow_private, 486
clear_cache, 503
conflicting, 440
create, 453
custom, 510-512
default settings, 442
disabling, 115, 444, 452, 463, 465-466, 514
edit, 453
edit_document, 465
edit_tv, 514
element_tree, 465
file_tree, 443, 465
for contexts, 433
list, 459
load, 449, 459, 475
mgr context, 441
my_allow_private, 486-487
my_view_user, 510
new_document_in_root, 439
new_user, 449
publish, 466
publish_document, 449
remove, 466
removing, 445
resource_tree, 465
save, 449, 453
save_chunk, 449, 453
save_document, 453, 465
save_element, 443
save_tv, 514
undelete, 466
unpublish, 466
view, 453, 459
view_tv, 514
view_user, 510
web context, 441
permissions system, 437-496
permissions system overview, 438-441
PHP
and smart quotes, 659
arithmetic operators, 660
arrays, 668-669
assignment operators, 660
associative arrays, 669
boolean variables, 655
built-in functions, 675-679
class variables, 683
classes, 679-686
coding style, 672-673
comments, 653-655
comparison operators, 661-662
conditionals, 664-668
constructors, 684
date and time functions, 679
debugging, 686-690
double quotes, 657-659
echo statement, 676

else statement, 665-667
elseif statement, 665
eof marker, 671
false, 656-657
file functions, 678-679
file pointers, 655
float variables, 655
for loop, 671-672
foreach loop, 670
get() and set() methods, 684-685
getting help, 652
help, 122
if statement, 664-666
instanceof, 337-338
integer variables, 655
K&R style, 672-673
logical operators, 663-664
member variables, 683
null variable, 655
object variables, 655
object-oriented programming, 679-686
objects, 679-686
online documentation, 652
OOP, 679-686
operators, 272-273
pasting code into, 659
print statement, 676
program flow, 670-672
quoted strings, 657-659
reserved words, 657
server requirements, 36-37
simple arrays, 668-669
single quotes, 657-659
string functions, 676-677
string variables, 655
switch statement, 666-667
ternary operator, 281-282, 667-668
true, 656-657
tutorial, 652-690
user-defined functions, 673-675
validators in transport packages, 619-620
variables, 272, 655-657
while loop, 671

PHP functions
array(), 678
array_merge(), 375, 384, 388
built-in, 675-679
count(), 678
date(), 679
empty(), 335-336
explode(), 678
fclose(), 678
feof(), 671
fgets(), 671, 678
file_exists(), 678
file_get_contents(), 606, 609, 615, 678
file_put_contents(), 678
fopen(), 678
fputs(), 679
fread(), 678
fwrite(), 678
implode(), 634, 678
is_array(), 335
is_object(), 337
isset(), 334-335
ltrim(), 676
mail(), 550, 696
online documentation, 652
rtrim(), 676
str_ireplace(), 677
str_replace(), 330, 609, 677
strftime(), 173, 215, 679
string, 676-677
stripos(), 677
stristr(), 677
strlen(), 676
strpos(), 677
strstr(), 676
strtolower(), 676
strtotime(), 173, 679
strtoupper(), 676
substr(), 677
time(), 679
trim(), 240, 609, 676
unlink(), 679
user-defined, 673-675

PHP operators, 272-273, 659-664
=, 661
=&, 312, 354, 388
===, 336
arithmetic, 660
assignment, 660
comparison, 661-662
logical, 663-664
ternary, 281-282, 667-668

PHP primer, 652-690
php.ini file, 536
php_curl.dll file, 536
phpinfo, 121
PHPMyAdmin
accessing, 63
creating database, 40
exporting database, 65-66
File Download dialog, 65-66
importing database, 65-66

PHX in Evolution, 187
PHX in Revolution, 167-175
placeholder tags, 15, 161
placeholders
defined, 15
in Evolution, 306
modx.user.id, 381
modx.user.username, 381
setting in code, 344-347, 383-384
setting in snippets, 272-273, 274-275
Wayfinder, 562

PluginDemo chunk, 291
PluginDemo resource, 291
plugins, 289-304
attaching property sets to, 293-295
attaching system events to, 292
attaching to events in code, 408
Captcha, 300-302
controlling with template variables, 295
debugging, 686-690
default properties, 293-295
defined, 13, 267, 289
disabling, 296
executing on create versus update resource, 299
Gateway, 432
in Evolution, 307
lexicon strings in, 347-351
listening to multiple system events, 298
manager system events, 707-708
MemberPages, 492-493
object reference, 692-693
page-not-found, 302-305
property sets with, 293-295
sanity checking in, 332-338
uses for, 13-14, 289, 302, 432
using property sets with, 293-295
working with, 289-305
working with in code, 408

policies
Administrator, 443, 444-445
context access, 444-445
duplicating, 444, 456
editing, 444
EditorAdmin, 456, 457, 458
EditorElement, 462, 463
EditorResource, 459
Element, 443, 444-445
element category, 444-445

FeView, 471, 472
FeViewTemplate, 486-487
for context access ACL entries, 450-451
for element category access ACL entries, 453
for resource group access ACL entries, 452, 458-459
Load, 471
Load List and View, 444, 459
Load Only, 443, 475
manager action, 444-445
object, 444-445, 459
Resource, 443, 444-445, 458, 483, 484
resource group access, 444-445
ResourceViewOnly, 460
types of, 444-445
View, 471

policy templates, 115, 443
FeViewTemplate, 471

porting
a template to MODX, 90
an existing site to MODX, 89-94
content to MODX, 91

prefixes for FURLs, 57
PreFormIt custom preHook for FormIt, 591-593
`prepare()` method, 636
`PRESERVE_KEYS` in transport packages, 606, 610, 621-622
Preview Site, 111
previewing resources, 103, 183
`print` statement in PHP, 676
priority of properties, 276
Private Document, 470
Private Document 2 resource, 486
Private resource group, 470, 472
PrivateViewers user group, 471
`process()` method, 328-329
processed content of template variables, 377
`processElementTags()` method, 646
processors, 409
Profile related object, 318, 382-384, 391-397
creating in code, 341-342, 404
getting with `getCollection()`, 393
getting with `getObjectGraph()`, 322-325, 388-389
getting with `getOne()`, 382, 392
object reference, 695
setting placeholders for, 383

Profile snippet, 396, 480
profiles in form customization, 516
program flow (PHP), 670-672
properties, 229, 247-263
adding to property sets, 257-258
BreadCrumbs, 553
`context` for link tags, 163
default, 249-255
EZfaq, 542
FormIt, 573-574
FormIt email hook, 587
FormIt spam hook, 587
FormItAutoResponder, 587-588
getResources, 567-570
getting in code, 365-368
in Evolution, 32, 186, 265, 306
in property sets, 250-251
in snippet tags, 270-271
key, 247
`loginResourceId` (Login), 474
overriding, 276
overview, 247-248
priority of, 276
`scheme` for link tags, 163
sending in chunk tags, 249
snippet, 195, 276-277
SPForm, 548-551
`startId` (Wayfinder), 558-559
`takeMeBack` (SPForm), 550-551
use with output modifiers, 174
using with plugins, 293-295
`validate` (FormIt), 578
value, 247
Wayfinder, 558-559, 561, 564-565
where stored, 250
working with, 365-368

properties of snippets tutorial, 270-280
property sets, 229
adding properties to, 257-258
and default property sets tutorial, 252-263
attaching to elements in code, 409
attaching to plugins, 293-295
attaching to snippets, 279
creating, 119, 256, 261-262, 279, 538-539
defined, 27-28
editing, 251
editing on the tools menu, 259
exporting, 261
getting in code, 370
importing, 261
properties in, 250-251
removing, 260
specifying in element tags, 259
specifying in snippet tags, 279
SPForm, 548-551
Tools menu, 119
use with snippets, 278-280
using with plugins, 293-295
versus default properties, 249-251
Wayfinder, 562-563
where stored, 250
with snippets tutorial, 270-280

protected
categories, 453
contexts, 439
elements, 439
resources, 451-453

protection of MODX objects, 438-439
`pub_date` (Publish Date) resource field, 148
`public_html` directory, 59, 422, 602
Publication Report, 120
Publish Date (`pub_date`) resource field, 148
`publish` permission, 466
`publish_default` system setting, 50
`publish_document` permission, 449
Published checkbox, 141
Published resource field, 141
PublishedBy related object, 389, 390-391
`publishedon` (Published On) resource field, 147
PublishedResources related object, 324
publishing resources, 103, 182
purging resources, 102
push (Git), 82
`putChunk()` method, 647
`putVehicle()` method (transport packages), 606, 610, 622

Q

queries
`innerJoin()` method, 393
`sortby()` method, 631-632
`where()` method, 362-363, 393, 631-632, 638

query criteria, 311-313, 363
`query()` method, 636

Quick Create
elements, 106
resources, 103, 182
static resources, 103
symlinks, 103
weblinks, 103
Quick Reference to system events, 696-715
Quick Update
elements, 106
resources, 103, 181
QuickEmail extra, 479
quoted strings in PHP, 657-659

R

radio options template variable input type, 212
raw content of template variables, 376
`readme.txt` file (transport packages), 602, 614-615
recaptcha hook for FormIt, 581-582
recently edited documents (Manager Home screen), 110
recently edited documents (Report menu), 121
recipientArray for SPForm, 549-550
redirect hook for FormIt, 581
`REDIRECT_HEADER` constant, 648
`REDIRECT_META` constant, 648
`REDIRECT_REFRESH` constant, 648
redirecting users
based on custom permissions, 485-488
based on login status, 481, 490
based on `tree_root_id` setting, 489-496
in FormIt, 579-581, 586
in SPForm, 544-545, 550-551
on login, 385, 473-474
on page-not-found, 302-305, 475
on registration, 477-480
with `sendForward()`, 302-303, 648
with `sendRedirect()`, 481, 488, 490, 494-495, 648
reference variables, 312-313
refresh resource, 182
`reg*()` methods, 647
`regClientCSS()` method, 378, 527, 647
`regClientHTMLBlock()` method, 647
`regClientScript()` method, 647
`regClientStartupHTMLBlock()` method, 647
`regClientStartupScript()` method, 647
Register resource, 477
Register snippet, 394, 396, 476-480
`register_globals`, 45-47, 110
`register_globals` is on warning, 45-47, 110
Registered user, 471
Registering users tutorial, 476-480
`registerNamespace()` method (transport packages), 606, 608
related objects, 317-319, 359
available with `getMany()`, 359
available with `getOne()`, 359
Children, 318, 321, 324, 357, 364
CreatedBy, 320, 323, 388, 390
CreatedResources, 324
defined, 317-319
DeletedBy, 390
EditedBy, 323, 388
in transport packages, 621-623
Parent, 318, 321, 359-361, 363
Profile, 318, 322-325, 341, 382-384, 388-389, 391-397, 404
PublishedBy, 389, 390-391
PublishedResources, 324
TemplateVars, 324
relatedUserTpl chunk, 389
release branches (Git), 72
`reloadConfig()` method, 647
Remove Category, 106
`remove` permission, 466
`remove()` method, 344, 639
`remove()` method (user object), 651
`removeAllEventListener()` method, 648
`removeCollection()` method, 636
`removeEventListener()` method, 648
`removeObject()` method, 636
`removeObject()` method (transport packages), 618
removing
(see also: deleting)
branches (Git), 82
categories, 106
default properties, 255
directories, 108
elements, 106
files, 109
leading and trailing spaces, 240
locks, 112
MODX objects in code, 344
permissions, 445
property sets, 260
resources, 102-103, 182
resources from resource groups, 448
resources from resource groups in code, 332
roles, 446
system settings, 124
Top Menu items, 126, 503-504, 508-510
transport packages, 54-55
users from user groups, 115, 116, 447-448
users from user groups in code, 332
Rename Directory dialog, 109
Rename File dialog, 109
renaming
(see also: changing and modifying)
`.htaccess` file, 59
directories, 108
files, 109
MODX `manager` directory, 423
the `core` directory, 421-422
Top Menu items, 504-505
`renderOutput()` method, 330-331, 652
replacing conditional output modifiers with a snippet tutorial, 280-283
reporting bugs, 129
reports, 120-122
Manager Actions, 121
Publication Schedule, 120
requirements
browsers, 36-37
database, 36-37
operating system, 36-37
PHP, 36-37
web servers, 36-37
reserved words in PHP, 657
reset user password in Manager, 127
ResetPassword snippet, 480
`resolve()` method (transport packages), 612-614, 619
resolvers (transport packages), 609
Resource Alias (Alias) resource field, 142
Resource Content (`content`) resource field, 145-146
resource fields, 7, 135-154

alias (Alias), 142
cacheable (Cacheable), 150
class_key (Class Key), 151
content (Resource Content), 145-146
content_dispo (Content Disposition), 151
content_type (Content Type), 150
createdby (Created By), 152
createdon (Created On), 152
deletedby (Deleted By), 153
deletedon (Deleted On), 153
description (Description), 142
displaying, 17
editedby (Edited By), 153
editedon (Edited On), 153
hidemenu (Hide From Menus), 145
hiding in Create/Edit Resource panel with form customization, 522
id (Resource ID), 139, 340
introtext (Summary), 143
isfolder (Container), 146, 356
link_attributes (Link Attributes), 143
longtitle (Long Title), 142
menuindex (Menu Index), 145
menutitle (Menu Title), 144
pagetitle (Resource Title), 141
parent (Parent Resource), 144
parent (Resource Parent), 358
pub_date (Publish Date), 148
published (Published), 141
publishedon (Published On), 147
quick reference, 138-154
Resource ID (id), 139
Resource Title (pagetitle), 141
richtext (Rich Text), 147
searchable (Searchable), 149
template (Uses Template), 140
unpub_date (Unpublish Date), 149
uri (URI), 154
uri_override (Freeze URI), 154

Resource Group Access ACL entries, 116, 451-453, 458-459
resource group access policies, 444-445
resource groups, 117, 438, 448
AllDocs, 455, 458
assigning resources to, 179
assigning template variables to, 220
creating, 448
HiddenTVs, 482, 484
News, 459
Private, 470, 472
ViewOnly, 459, 460

Resource ID (id) resource field, 139
resource list template variable input type, 212
resource manager system events, 703-705
Resource policy, 443, 444-445, 458, 483, 484
resource tags, 11, 154-156
Resource Title (pagetitle) resource field, 141
Resource tree, 100-104
deleting resources, 103
Duplicate Resource, 103
Edit Resource, 103
Evolution, 130
icons, 101
menuindex (sorting by), 101-102
purging resources, 102
Quick Update Resource, 103
right-click menu, 103
sorting, 101-102
type style, 102
View Resource, 103

resource/create action, 514
resource/update action, 514
resource_tree permission, 465
resources, 135-183
(see also: documents)
assigning to resource groups, 117
assigning to resource groups in code, 408
changing alias in code, 344
Confirm Registration, 477
Create, 103
creating, 6, 135-154
creating in code tutorial, 398
creating with processor, 409
deleting, 102-103, 182
duplicating, 103, 181
edit_document permission, 465
editing, 135-154
Evolution Resource tree, 130
fields, 138
for Wayfinder tutorial, 557-558
getTVValue() method, 378
hiding in the Manager, 438-439, 454-460
hiding with ACL entries tutorial, 470-474
hiding with custom permissions tutorial, 485-488
importing, 118
in Evolution, 185, 397
joinGroup() method, 332
leaveGroup() method, 332
Member, 489
object reference, 691
ObjectDemo, 352
page settings tab, 136
preview, 103
previewing, 183
Private Document, 470
Private Document2, 486
protected, 438-439, 451-453
publish permission, 466
publish_document permission, 449
publishing, 103, 182
Quick Create, 103, 182
Quick Update, 181
recently edited (Manager Home screen), 110
recently edited (Report menu), 121
Register, 477
remove permission, 466
removing from resource groups, 448
removing from resource groups in code, 332
save_document permission, 453, 465
site_content table, 137
static, 178
symlinks, 177
Thank You (SPForm), 544-545
Thanks for Registering, 477
toArray() method, 370-373
Unauthorized, 487
undelete permission, 466
undeleting, 103
unpublish permission, 446
users related to, 389-391
users related to current resource, 387-389
using tags in, 154-167
viewing cache output, 180
weblinks, 176-177
working with in code tutorial, 352-365

ResourceViewOnly policy, 460
retrieving (see: getting)
reverse (output modifier), 171
Revolution
architecture, x-xiii
building the development version,

87-88
  cloning the development version, 73-75
  development version, 70-89
  objects, 680-682
  overview, viii-ix
  roles in, 446
  tags, 17-19
  updating the development version, 88-89
  versus Evolution, vii-x

Revolution development version
  building, 87-88
  cloning, 73-75
  creating a fork, 78-79
  updating, 88-89

RewriteBase, 60

rich text
  manager system events, 712
  template variable input type, 213
  template variable output type, 217

`richtext` (Rich Text) resource field, 147

richtext editor, 147

right-click menu
  Element tree, 106
  File tree, 108-109
  Resource tree, 103

robustness of MODX, xiii

roles
  assigning to users, 115
  authority numbers, 440, 446
  creating, 440, 446
  Editor, 458, 459, 460, 463, 484
  FeViewer, 471, 472
  form customization, 515
  in Evolution, 446, 498
  in Revolution, 446
  in security permissions, 449
  member, 475
  minimum, 450-453
  minimum role for context access ACL entries, 450-451
  minimum role for element category access ACL entries, 453
  minimum role for resource group access ACL entries, 452, 458-459
  overview, 446
  removing, 446
  Revolution versus Evolution, 440
  working with in code, 408

`rollback()` method, 405, 636

`root` directory, 420

RowTpl chunk, 242, 284

RSS feeds, 110, 110, disabling

`rtrim()` PHP function, 676

rule sets in form customization, 516

running setup, 41

running transport package build script outside of MODX tutorial, 624-625

`runSnippet()` method, 373-375, 648

`runSnippet()` method in Evolution, 416

S

sanity checking, 332-338, 393

`save` permission, 449, 453

`save()` method, 339, 343-344, 397, 398, 400, 404, 639
  element objects, 400
  overview, 343-344
  reference, 639
  resource objects, 339
  user objects, 404, 651

`save_chunk` permission, 449, 453

`save_document` permission, 453, 465

`save_element` permission, 443

`save_tv` permission, 514

schema file (MODX), 318-319

schema for custom database files, 410-411

scheme argument for `makeUrl()`, 646

`scheme` property for link tags, 163

scriptProperties (see: `$scriptProperties`)

Search Locally for Packages, 53, 123, 537, 610

searchable resource field, 149

searching MODX forums, 127-128

searching the site, 112

security
  Access Control Lists, 441, 449-454
  access policies, 440, 443-445
  adding users to user groups, 447-448
  Administrator policy, 444-445
  authority numbers, 440
  context access ACL entries, 449-451
  context access policy, 444-445
  creating ACL entries, 449-454
  creating context access ACL entries, 456-457
  creating resource group access ACL entries, 458-459
  creating roles, 446
  creating user groups, 447, 448
  default settings, 442
  editing object policies, 465-466
  element category access policy, 444-445
  Element policy, 444-445
  elements, 442-453
  `hasPermission()` method, 485
  hiding elements tutorial, 461-463
  hiding pages with a snippet tutorial, 468-470
  hiding resources in the Manager tutorial, 454-460
  hiding resources with ACL entries tutorial, 470-474
  hiding resources with custom permissions tutorial, 485-488
  hiding template variables in the front end tutorial, 484-485
  hiding template variables in the Manager with ACLs tutorials, 481-484
  in Evolution, 497-498
  limiting permissions in the Manager tutorial, 464-467
  manager action policies, 444-445
  member pages with `tree_root_id` tutorial, 489-496
  MemberPages plugin, 492-493
  MemberPages snippet, 490-491
  minimum role, 450-453, 456-459
  minimum role for context access ACL entries, 449
  minimum role for element category access ACL entries, 453
  minimum role for resource group access ACL entries, 452, 458-459
  object policies, 444-445
  overview, 438-441
  permissions, 440, 443
  permissions for contexts, 433
  policies for context access ACL entries, 450-451
  policies for element category access ACL entries, 453
  policies for resource group access ACL entries, 452, 458-459
  protected categories, 453
  protected resources, 451-453
  protection, 438-439
  removing roles, 446
  reorganizing site for, 420-423

resource group access policies, 444-445
Resource policy, 444-445
Revolution versus Evolution, 440
roles in, 449
RSS feed, 110
system, 437-496
Top Menu, 113-117
user roles, 448
working with, 454-496

Security menu
Access Controls, 115
Flush All Sessions, 117
Flush Permissions, 117
Manage Users, 114

`select` (output modifier), 168
`sendEmail()` method, 651
`sendError()` method, 648
`sendErrorPage()` method, 469, 648
`sendForward()` method, 302, 648
`sendRedirect()` method, 488, 648
`sendUnauthorizedPage()` method, 469, 490, 648
SEO-friendly URLs, 55-61
se`OnChunkBeforeRemove`, 706
server 500 error, 46
server requirements
browsers, 36-37
database, 36-37
for MODX, 36-37
operating system, 36-37
PHP, 36-37
web servers, 36-37

`set()` method, 340, 397, 399, 404, 639
`set()` method (transport packages), 606
`setAttribute()` method, 636
`setContent()` method, 340, 399, 400, 651
`setContent()` method (transport packages), 606
`setDebug()` method, 636, 648
`setLogLevel()` method, 636
`setLogLevel()` method (transport packages), 625
`setLogTarget()` method, 636
`setLogTarget()` method (transport packages), 625
`setLogTarget()` method options, 636
`setOption()` method, 637
setPackageAttributes array (transport packages), 610
`setPackageAttributes()` method (transport packages), 615
`setPlaceholder()` method, 344-347, 591, 593, 648-649
`setPlaceholders()` method, 346-347, 649
sets in form customization, 516
`setService()` method, 643
setting
directory permissions, 108, 108
field default values with form customization tutorial, 523
field labels with form customization tutorial, 523
file and directory permissions, 69
file permissions, 108
package attributes for transport packages, 615
placeholders in code, 344-347, 383-384
placeholders with a snippet tutorial, 274-275
template variable labels with form customization tutorial, 523
TV default values with form customization tutorial, 517-519
user password, 127

setting tags, 25, 159-161
settings
(see also: context settings)
(see also: system settings)
(see also: user settings)
defined, 16
displaying, 17
in Evolution, 32
working with in code, 380

setup, 41
`setup` directory, 420
`setup/includes` directory, 420
`setValue()` method (FormIt hooks), 592
`setValue()` method (TV), 404, 414
`SHA1` (Git), 71
sharing
core between sites, 425-426
core with multiple contexts, 433
database between sites, 425

`show` (output modifier), 168
showing (see: displaying)
ShowList category, 242, 284
ShowList tutorial, 283-288
simple arrays (PHP), 668-669
simple filter plugin tutorial, 290-295
simple snippet tutorial, 268-270
simple transport package tutorial, 605-610
single quotes in PHP, 657-659
single-line comments in PHP, 654
site
Clear Cache, 65, 111
Logout, 113
New Document, 113
New Static Resource, 113
New Symlink, 113
New Weblink, 113
Remove Locks, 112
searching, 112
View, 111

site cache (clearing), 65, 111
site name (changing), 49
site organization, 419-423
Site panel, 111
Site Preview, 111
Site Schedule, 120
`site_content` table, 137
`site_name` system setting, 49, 109, 193, 651
`site_snippets` database table, 250
`site_start` system setting, 197, 428, 455, 553
`site_url` context setting, 429
`site_url` system setting, 193, 429
sites (multi-language), 434
smart quotes in PHP, 659
Smarty, 523, 532
SMTP
settings for Gmail, 586, 696
system settings for, 696
using in Evolution, 596
using with FormIt, 586
using with SPForm, 550

snippet default properties, 249-255, 276-277
snippet properties, 249-263, 269-271
as PHP variables, 270-271
defined, 12, 267
in Evolution, 186, 595
in snippet tags, 195, 270-271
using chunks in, 236-237
using template variables in, 204

snippet tags, 12-13, 157-158, 195
display rather than execute, 330

SnippetDemo tutorial, 270-280
snippets
- `$output` variable, 272
- `addOne()` method, 403
- advanced operations, 309-414
- assigning resources to resource groups in, 408
- assigning to categories in code, 401-402
- assigning to categories in transport packages, 621-623
- attaching plugins to events in, 408
- attaching property sets, 279
- attaching property sets to elements in, 409
- ChangePassword, 481
- CheckPermission, 487-488
- creating categories in, 401-402
- creating chunks in, 400
- creating elements in, 400
- creating in code tutorial, 400
- creating objects in, 338-344
- creating template variables in, 402-404
- creating templates in, 400
- debugging, 686-690
- default properties, 249-255, 276-277
- defined, 12-13, 267
- disabling, 92
- displaying lexicon strings, 413
- displaying output, 17
- displaying output of, 272
- Ditto, 565, 570
- extended user fields in, 393-397
- EZfaq tutorial, 539-542
- ForgotPassword, 480
- FormIt, 571-594
- FormItRetriever, 592
- getResources, 565-571
- getting template variables in, 377-379
- getting user information in, 380-396
- hello snippet tutorial, 269-270
- Hello2, 605
- HidePages, 469
- hiding pages with, 468-470
- in Evolution, 306
- installing with Package Manager, 268
- lexicon strings in, 347-351
- loading lexicon files, 412
- Login, 472-473
- manager system events, 706-707
- MemberPages, 490-491
- object reference, 692
- ObjectDemo, 353
- page-not-found, 302-304
- Profile, 396, 480
- properties, 12, 249-263, 267, 269-271, 276-277
- properties tutorial, 270-280
- property sets, 250-151, 256-263
- Register, 394, 396, 476-480
- ResetPassword, 480
- return value, 272
- sanity checking in, 332-338
- setting placeholders in, 274-275
- SPForm, 543-551
- SPFResponse, 548-549
- StudentTeacher, 387
- UpdateProfile, 396, 405, 480
- uses for, 12, 267, 302, 338, 370, 380-381, 387, 468
- using the lexicon in, 411-413
- using Tpl chunks in, 370-373
- working with, 268-288
- working with roles, 408
- working with user groups in, 406-408

sockets, 536
sort order TV field, 206
`sortby()` method for xPDO queries, 631-632
sorting
- and MySQL 5.0.51, 145
- in `getCollection()`, 362-361, 631-632
- in `GetMany()`, 639
- in getResources, 569
- in Wayfinder, 145, 564
- in xPDO queries, 631-632
- Recently Edited Documents report, 121
- Resource tree, 101-102
- snippet output, 284-287
- system events grid, 697
- template variables, 206

source directory (transport packages), 601, 614, 624
spam hook for FormIt, 581-582, 587
spam hook properties for FormIt, 587
specifying property sets in element tags, 259
specifying property sets in snippet tags, 279
`spf_block_prompt`, 547
`spf_inline_prompt`, 547
SPForm
- contact form tutorial, 543-551
- Contact page, 545
- CSS file, 547
- formProcAllowedReferers, 550
- installing, 543
- lexicon files, 548
- multiple recipients for, 549-550
- overview, 544
- properties, 548-551
- property sets for, 548-551
- recipientArray, 549-550
- setting options, 548-551
- snippet, 543-551
- `spf_block` prompt, 547
- `spf_inline` prompt, 547
- `takeMeBack` property, 550-551
- thank you page, 544-545
- Tpl chunks for, 545
- translating to other languages, 548
- using SMTP with, 550

SPFResponse snippet, 548-549
SQLSTATE string, 631
static resources, 178
- creating, 103, 178
- for downloadable files, 178
- overview, 9
- Quick Create, 103

storing/retrieving data in a chunk tutorial, 238-241
`str_ireplace()` PHP function, 677
`str_replace()` PHP function, 330, 609, 677
`strftime()` PHP function, 173, 215, 679
String
- functions in PHP, 676-677
- output modifiers, 169-171
- template variable output type, 218
- variables in PHP, 655

`strip` (output modifier), 170
`strip_tags` (output modifier), 170
`stripos()` PHP function, 677
`striptags` (output modifier), 170
`stripTags()` method, 649
`stristr()` PHP function, 677
`strlen` (output modifier), 170
`strlen()` PHP function, 676
`strpos()` PHP function, 677
`strrev` (output modifier), 171
`strstr()` PHP function, 676
`strtolower` (output modifier), 169

strtolower() PHP function, 676
strtotime (output modifier), 173, 567
strtotime() PHP function, 173, 679
strtoupper (output modifier), 170
strtoupper() PHP function, 676
StudentTeacher tutorial, 387
styling Wayfinder menus, 558-559
substr() PHP function, 677
subtract (output modifier), 172
suffixes for FURLs, 57
Summary (introtext) resource field, 143
suPHP, 47
suPHP_ConfigPath, 47
Support menu, 127-128
  Bugs link, 129
  Forums link, 127-128
  Wiki link, 128
supported browsers, 36-37
SVN for transport packages, 604-605
switch statement in PHP, 666-667
switchContext() method, 649
switching branches (Git), 72
symlinks, 9, 177
  creating, 103
  Quick Create, 103
system events, 289-290, 696-715
  back end, 699-713
  cache, 715
  category manager events, 712
  chunk manager events, 705-706
  context manager events, 713
  firing, 289-290
  front-end events, 714-715
  invoking, 289-290
  manager action, 700-713
  module events (Evolution), 712
  multiple events in a plugin, 298
  OnBeforeCacheUpdate, 715
  OnBeforeChunkFormDelete, 706
  OnBeforeChunkFormSave, 705
  OnBeforeChunkSave, 297
  OnBeforeDocFormDelete, 704, 704
  OnBeforeDocFormSave, 296, 703
  OnBeforeEmptyTrash, 705
  OnBeforeManagerLogin, 301, 700
  OnBeforeManagerLogout, 701
  OnBeforeManagerPageInit, 701
  OnBeforeModFormDelete, 712
  OnBeforeModFormSave, 712
  OnBeforePluginFormDelete, 708
  OnBeforePluginFormSave, 707
  OnBeforeResourceGroupRemove, 705
  OnBeforeSaveWebPageCache, 715
  OnBeforeSnipFormDelete, 707
  OnBeforeSnipFormSave, 706
  OnBeforeTempFormDelete, 709
  OnBeforeTempFormSave, 708
  OnBeforeTVFormDelete, 710
  OnBeforeTVFormSave, 709
  OnBeforeUserActivate, 711
  OnBeforeUserFormDelete, 702, 711
  OnBeforeUserFormSave, 702, 710
  OnBeforeWebLogin, 714
  OnBeforeWebLogout, 715
  OnBeforeWUserFormDelete, 711
  OnBeforeWUserFormSave, 711
  OnCacheUpdate, 715
  OnCategoryBeforeRemove, 712
  OnCategoryBeforeSave, 712
  OnCategoryRemove, 712
  OnCategorySave, 712
  OnChunkBeforeSave, 706
  OnChunkFormDelete, 706
  OnChunkFormPrerender, 705
  OnChunkFormRender, 705
  OnChunkFormSave, 706
  OnChunkRemove, 706
  OnChunkSave, 706
  OnContextBeforeRemove, 713
  OnContextBeforeSave, 713
  OnContextFormPrerender, 713
  OnContextFormRender, 713
  OnContextRemove, 713
  OnContextSave, 713
  OnContextUpdate, 713
  OnCreateDocGroup, 704
  OnDocFormDelete, 704
  OnDocFormPrerender, 703
  OnDocFormRender, 703
  OnDocFormSave, 703
  OnDocPublished, 703
  OnDocUnPublished, 703
  OnEmptyTrash, 705
  OnFileManagerUpload, 702
  OnFriendlyURLSettingsRender, 713
  OnHandleRequest, 432, 714
  OnInterfaceSettingsRender, 713
  OnLoadWebDocument, 714
  OnLoadWebPageCache, 715
  OnLogPageHit, 715
  OnManagerAuthentication, 700
  OnManagerChangePassword, 701
  OnManagerCreateGroup, 701
  OnManagerDeleteUser, 702, 711
  OnManagerLogin, 700
  OnManagerLoginFormPrerender, 700
  OnManagerLoginFormRender, 301, 700
  OnManagerLogout, 701
  OnManagerPageInit, 701
  OnManagerSaveUser, 702, 710
  OnModFormDelete, 712
  OnModFormPrerender, 712
  OnModFormRender, 712
  OnModFormSave, 712
  OnPageNotFound, 302, 304, 715
  OnPageUnauthorized, 714-715
  OnParseDocument, 714
  OnPluginBeforeRemove, 708
  OnPluginBeforeSave, 707
  OnPluginEventBeforeRemove, 708
  OnPluginEventRemove, 708
  OnPluginFormDelete, 708
  OnPluginFormPrerender, 707
  OnPluginFormRender, 707
  OnPluginFormSave, 708
  OnPluginRemove, 708
  OnPluginSave, 707
  OnResourceBeforeSort, 704
  OnResourceDuplicate, 704
  OnResourceGroupBeforeSave, 705
  OnResourceGroupRemove, 705
  OnResourceGroupSave, 705
  OnResourceSort, 704
  OnResourceToolbarLoad, 704
  OnResourceTVFormPrerender, 704
  OnResourceTVFormRender, 704
  OnResourceUndelete, 704
  OnRichTextBrowserInit, 712
  OnRichTextEditorInit, 712
  OnRichTextEditorRegister, 712
  OnSiteRefresh, 715
  OnSiteSettingsRender, 713
  OnSnipFormDelete, 707
  OnSnipFormPrerender, 706
  OnSnipFormRender, 706
  OnSnipFormSave, 707
  OnSnippetBeforeRemove, 707
  OnSnippetBeforeSave, 706
  OnSnippetRemove, 707
  OnSnippetSave, 707
  OnTempFormDelete, 709

OnTempFormPrerender, 708
OnTempFormRender, 708
OnTempFormSave, 709
OnTemplateBeforeRemove, 709
OnTemplateBeforeSave, 708
OnTemplateRemove, 709
OnTemplateSave, 709
OnTemplateVarBeforeRemove, 710
OnTemplateVarBeforeSave, 709
OnTemplateVarRemove, 710
OnTemplateVarSave, 710
OnTVFormDelete, 710
OnTVFormPrerender, 709
OnTVFormRender, 709
OnTVFormSave, 710
OnUserActivate, 711
OnUserBeforeRemove, 702, 711
OnUserBeforeSave, 702, 711
OnUserChangePassword, 701
OnUserFormDelete, 702, 711
OnUserFormPrerender, 710
OnUserFormRender, 493, 710
OnUserFormSave, 493, 702, 711
OnUserGroupBeforeRemove, 701
OnUserGroupBeforeSave, 701
OnUserGroupRemove, 701
OnUserGroupSave, 701
OnUserNotFound, 700
OnUserRemove, 702, 711
OnUserSave, 702, 711
OnUserSettingsRender, 713
OnWebAuthentication, 700
OnWebAuthentification, 714
OnWebChangePassword, 714
OnWebCreateGroup, 714
OnWebDeleteUser, 714
OnWebLogin, 714
OnWebLogout, 715
OnWebPageCompete, 714
OnWebPageInit, 714
OnWebPagePrerender, 290, 292, 714
OnWebSaveUser, 714
OnWUserFormDelete, 711
OnWUserFormPrerender, 711
OnWUsrFormRender, 711
OnWUsrFormSave, 711
plugin manager events, 707-708
Quick Reference, 696-715
resource manager events, 703-705
rich text manager events, 712
snippet manager events, 706-707
system setting events, 713
template manager events, 708-709
template variable manager events, 709-710
user manager events, 710-711

System Info, 121
System menu, 122-126
Actions, 126
Content Types, 125
Contexts, 125
Lexicon Management, 124-125
Namespaces, 126
Package Manager, 122-123
System Settings, 123-124

system settings
assets_url, 527
automatic_alias, 143, 409
base_url, 428
container_suffix, 57, 146
defined, 25
editing, 50, 123-124
emailsender, 651
emailsubject, 651
error_page, 302, 475
filemanager_path, 467-468
FURLs, 55-61
getting with getOption(), 380
http_host, 429
in Evolution, 32
manager system events, 713
manager_language, 612
manager_url, 531
modx_base_path, 642
modx_base_url, 642
modx_charset, 192
modx_core_path, 642
not in system setting grid, 159
publish_default, 50
removing, 124
RSS feeds, 110
site_name, 49, 109, 193, 651
site_start, 197, 428, 455, 553
site_url, 193, 429
tree_root_id, 468, 468, 490
udperms_allowroot, 439
unauthorized_page, 475

T

tab rules in form customization, 515
table prefix, 425
tag (output modifier), 171
tag syntax specification, 19
tags, 11, 12-13, 18, 49, 90, 145-146, 150, 154-176
base href, 429
cached, 150, 166
chunk, 11-12, 157, 194
content, 49, 145-146
content (in template), 194
getResources, 196
in Evolution, 306
in templates, 199
language, 163-164
link, 16, 161-163, 197
MODX Evolution, 17
MODX Revolution, 17-19
nested, 18, 164-165, 197
placeholder, 15, 161
quick reference, 154
resource, 11, 154-156
setting, 25, 159-161
snippet, 12-13, 157-158, 195
syntax definition, 18
uncached, 150, 166
uses for, 154-167
using in documents, 154-167
using with chunks, 232
Wayfinder, 195

takeMeBack property for SPForm, 550-551
target directory (transport packages), 601, 614, 624
template (Uses Template) resource field, 140
template variable fields, 205-208
caption, 205
category, 205
default value, 207
description, 205
input option values, 206
input type, 206
lock variable for editing, 206
output type, 208
sort order, 206
variable name, 205

template variables, 199-226, 200, uses for
@ BINDINGS, 220-226
@CHUNK binding, 222-223
@DIRECTORY binding, 225
@EVAL binding, 225-226
@FILE binding, 224
@INHERIT binding, 207, 221-222
@RESOURCE binding, 224
@SELECT binding, 225
access permissions tab, 220
assigning to resource groups, 219
attaching to templates, 203, 219

attaching to templates in code, 402-404
auto-tag input type, 208-209
checkbox input type, 209-210
controlling plugins with, 295
controlling snippets with, 204
creating, 201
creating in code tutorial, 402-404
creating in Evolution, 227
creating properties for, 219
date input type, 210
date output type, 214-215
default output type, 215
default value, 203
defined, 10
delimiter output type, 216
displaying, 11, 17, 200, 203, 375-376
dropdown list menu input type, 210
`edit_tv` permission, 514
email input type, 210
fields of, 205-208
file input type, 211
getting in code, 377-379
getting processed output of, 377
getting raw content of, 376
`getValue()` method, 330
`hasTemplate()` method, 331
hidden input type, 211
hiding in the Manager, 481-484
hiding with form customization, 514, 520
HTML tag output type, 216
htmlarea input type, 211
image input type, 211-212
image output type, 217
in snippet properties, 204
input types, 208-213
listbox (multi-select) input type, 212
listbox (single-select) input type, 212
manager system events, 709-710
moving to new tab with form customization, 520
number input type, 212
object reference, 693-694
output types, 214-218
overview, 10
properties tab, 219
radio options input type, 212
`renderOutput()` method, 330
resource list input type, 212
rich text input type, 213
rich text output type, 217
`save_tv` permission, 514
setting captions (labels) with form customization, 523
setting default values with form customization, 517-519
`setValue()` method, 414
string output type, 218
template access tab, 219
text input type, 213
textarea (mini) input type, 213
textarea input type, 213
URL output type, 218
uses for, 10, 200, 204, 296, 377
using tags to display, 203
using to control snippets, 204
`view_tv` permission, 514
working with in code, 377-378

template variables tab (Create/Edit Resource panel), 179

templates, 189-199
assigning to categories in code, 401-402
attaching template variables to, 203, 219
attaching template variables to in code, 402-404
content tag, 193
creating, 190
creating in code tutorial, 400
example, 191
`hasTemplateVar()` method, 331
in Evolution, 185
manager system events, 708-709
Member, 490-491
object reference, 693
overview, 10, 10
porting to MODX, 90
using tags in, 199
what to include, 198

TemplateVars related object, 324

ternary operator (PHP), 281-282, 667-668

text template variable input type, 213

textarea (mini) template variable input type, 213

textarea template variable input type, 213

Thank You page (SPForm), 544-545

Thanks for Registering resource, 477

`then` (output modifier), 168

third-party components (see: add-on components)

`time()` PHP function, 679

TinyMCE, 147

Title (`pagetitle`) resource field, 141

`toArray()` method, 328, 355, 384, 639

`toJSON()` method, 639

Tools menu, 117-120
Error Log, 121
Import HTML, 118
Import Resources, 118
Manager Actions Report, 121
Property Sets, 119
Reports, 120-122
System Info, 121

Top Menu, 109-128
actions, 502-508
Actions tree, 505-506
adding menu items, 126
altering, 502-513
Components, 109, 113, 508
creating actions, 524-525
creating items, 508-510
custom items, 508-509
editing items, 126
Evolution, 130-132
handler, 503-504
handling menu clicks, 507
hiding items, 510-512
Home, 109
lexicon keys, 508
menu tree, 503-513
modifying, 126
order of items, 507-508
removing items, 126, 503-504, 508-510
renaming items, 504-505
reordering, 507-508
Security, 113-117
Site, 111
Support, 127-128
System, 122-126
Tools, 117-120
Update Menu Dialog, 503
User, 126-127

topics (lexicon), 23-24, 349-351, 506, 524

`toPlaceholder()` method, 346, 649

`toPlaceholders()` method, 346-347, 384, 388-389, 649

Tpl chunks, 241-244, 283-288, 473
defined, 12
for FormIt, 585, 587

for SPForm, 545
tutorial, 283-288
Wayfinder, 558-559, 560-562
working with in code, 370-373
tracking branches (Git), 80
Traditional distribution of MODX, 420-423
transactions, 405-406
transfer local site to remote server, 64-70
transferring files in transport packages, 612-615
transport packages, 22
(see also: add-on components)
_build directory, 601, 614, 623
aborting, 620
array of file paths, 623-624
assets directory, 602-604, 623
build scripts, 601, 605-610
build.config.php file, 624
build.transport.php file, 621-623
categories, 621-623
changelog.txt, 614-615
changelogs, 602
core directory, 602-603, 623
createVehicle() method, 606, 610, 622
creating, 599-626
defined, 22
defined constants, 605-607
directory names in, 601
downloading, 53, 122
file locations, 601-604
file names in, 601
file paths, 623-624
file resolvers, 612-614, 623
file validators, 619
file vehicles, 614-615
Git and, 604-605
in Evolution, 626
install.script.php file, 616-620
installing, 54
installing manually, 53-55, 537-538
interacting with the user in, 616-620
internationalization, 611-612
lexicon files, 603, 611-612
log() method, 617
MyComponent example script, 625
newObject() method, 621
not downloadable through package management, 53-55, 537
pack() method, 607
package attributes, 615
PHP validators, 619-620
PRESERVE_KEYS, 606, 610, 621-622
putVehicle() method, 606, 610, 622
QuickEmail, 479
readme.txt file, 602, 614-615
related objects in, 621-623
removing, 54-55
resolve() method, 612-614, 619
resolvers, 609
running build script outside of MODX, 624-625
setLogLevel() method, 625
setLogTarget() method, 625
setPackageAttributes, 615
setPackageAttributes array, 610
source directory, 601, 614, 624
SVN, 604-605
target directory, 601, 614, 624
transferring files, 612-614
uninstalling, 54-55
UNIQUE_KEY, 606, 610, 621-622
UPDATE_OBJECT, 606, 610, 621-622
updating, 54
user.input.html file, 614-615
uses for, 599
validate() method, 620
validators, 609, 619-620, 619-620
version control, 604-605
xPDOTransport::RELATED_OBJECT_ATTRIBUTES, 621-623
xPDOTransport::RELATED_OBJECTS, 621-623
transport.core.php file (Revolution development version), 87
tree_root_id
redirecting users based on, 489-496
system setting, 468, 490
tutorial, 466-468
user setting, 447, 466-468, 489
trim() PHP function, 240, 609, 676
troubleshooting
admin Super User permissions, 444
Apache startup, 430
debugging code, 686-690
disappearing menu items, 512
disappearing resources, 100, 439
disappearing TVs, 519, 521
email, 479, 550
FURLs and mod_rewrite,
image TVs, 217
mail not sent, 479
MODX API methods, 629-652
MODX forum searches, 127-128
Package Manager and cURL, 52, 536
pasting PHP code, 659
register_globals, 110
Server 500 error, 46
sorting, 145
system events, 696-715
true in PHP, 656-657
tutorials
advanced transport package, 611-625
BadWords, 290-295
BadWordsMgr, 296-299
BreadCrumbs add-on component, 551-555
ChangeEmail custom hook for FormIt, 591-594
CheatSheet component, 523-533
contact form with FormIt, 581-587
contact form with SPForm, 543-551
controlling text color with a TV, 377-379
creating a menu with Wayfinder, 555-565
creating a simple object class in PHP, 682-686
creating a transport package, 599-625
creating elements in code, 400
creating FAQ page with EZfaq snippet, 539-542
creating forms with FormIt, 571-594
creating new Create/Edit Resource panel tabs with form customization, 522
creating resources in code, 398
creating template variables in code, 402-404
custom manager page, 523-533
enhanced hello snippet, 273-275
filemanager_path, 467-468
FormIt custom hook, 588-594
FormIt snippet, 571-594
getResources, 566-567
Hello World (PHP), 653
HidePages snippet, 469
hiding a TV with form customization, 520
hiding Create/Edit Resource panel fields with form customization, 522
hiding Create/Edit Resource panel tabs with form customization, 520
hiding elements, 461-463

hiding pages with a snippet, 468-470
hiding resources in the Manager, 454-460
hiding resources with ACL entries, 470-474
hiding resources with custom permissions, 485-488
hiding template variables in the front end, 484-485
hiding template variables in the Manager with ACLs, 481-484
info chunk, 233-235
limiting access to elements, 461-463
limiting element permissions, 461-463
limiting permissions in the Manager, 464-467
Login page, 472-473
member pages with `tree_root_id`, 489-496
modifying the Top Menu, 507-512
moving TV to a new tab with form customization, 520
news summary with getResources, 566-567
ObjectDemo, 353
page count snippet, 413-414
PageNotFound, 302-305
PHP, 652-690
properties of snippets, 270-280
property sets and default properties, 252-263
Registering users, 476-480
replacing conditional modifiers with a snippet, 280-283
running transport package build script outside of MODX, 624-625
setting field default values with form customization, 523
setting field labels with form customization, 523
setting placeholders with a snippet, 274-275
setting template variable labels with form customization, 523
setting TV default values with form customization, 517-519
ShowList, 283-288
simple chunk, 230-235
simple filter plugin, 290-295
simple snippet, 268-270
simple transport package, 605-610
SnippetDemo, 270-280
storing/retrieving data in a chunk, 238-241
StudentTeacher, 387
Tpl chunk, 283-288
`tree_root_id`, 466-468
TV-based chunks, 244-247
user extended fields in code, 393-397
using `isMember()` to select content, 387
using the lexicon in snippets, 411-413
working with resources in code, 352-365
YesNoDemo, 280-283
TV rules in form customization, 515
TV-based chunks tutorial, 244-247
TVs (see: template variables)
two contexts in same directory, 431-432
type style of Resource tree, 102
typographical conventions in *MODX: The Official Guide*, iii-v

U

`ucase` (output modifier), 170
`ucfirst` (output modifier), 170
`ucwords` (output modifier), 170
`udperms_allowroot` system setting, 439
`uft8_unicode_ci`, 63
unauthorized page, 50
unauthorized page unpublished warning, 110
Unauthorized resource, 487
`unauthorized_page` system setting, 475
uncached tags, 150, 166
`undelete` permission, 466
undeleting resources, 103
`UNIQUE_KEY` in transport packages, 606, 610, 621-622
unix timestamp, 173
`unlink()` PHP function, 679
`unpub_date` (Unpublish Date) resource field, 149
Unpublication Report, 120
Unpublish Date (`unpub_date`) resource field, 149
`unpublish` permission, 466
unpublished error page warning, 110
unpublished unauthorized page warning, 110
Unregistered user, 471
`unsetPlaceholder()` method, 347, 649
`unsetPlaceholders()` method, 347, 649
Update Action dialog, 505
Update Menu dialog, 503, 510
Update User Group panel, 115
Update User Role dialog, 448
`UPDATE_OBJECT` in transport packages, 606, 610, 621-622
UpdateProfile snippet, 396, 405, 480
updating
development version of Revolution, 88-89
MODX objects in code, 338-344
transport packages, 54
user groups, 115, 115-116
upgrading (see: updating)
upload files, 108
Upload Files dialog, 108
`uppercase` (output modifier), 170
upstream remote (Git), 80
`uri` (URI) resource field, 154
`uri_override` (Freeze URI) resource field, 154
URL scheme in `makeUrl()` method, 646
URL template variable output type, 218
user
(see also: user profile)
(see also: users)
anonymous, 104, 381-383, 387, 460, 469, 474-475, 481
class methods, 650-651
current, 16, 318, 327, 381-383, 469, 485, 492-493, 592-593
extended fields, 115
Messages, 127
passwords in code, 342
recent resources edited by, 127
reset password in Manager, 127
settings, 114
Top Menu, 126-127
user extended fields in code tutorial, 393-397
user fields
active, 479
id, 381
password, 342, 381, 640, 650
username, 381
User Group Context Access dialog, 449-451, 456-457

user groups
adding users to, 115, 447-448
Administrator, 450-451
checking membership in code, 386-387
creating, 115, 447
custom menus for, 513
Editors, 455
PrivateViewers, 471
removing users from, 115
updating, 115, 115-116
working with in code, 406-408
user id field, 381
user information
displaying, 16, 380-396
getting in code, 380-396
in Evolution, 307
user interface, xi
User menu
Messages, 127
profile, 127
Recent Resources, 127
Reset Password, 127
user profile, 127, 318-319, 322-325, 382-384
creating in code, 341-342, 404-405
getting with `getCollection()`, 393
getting with `getObjectGraph()`, 322-325, 388-389
getting with `getOne()`, 382
object reference, 695
setting placeholders for, 382-383
user roles, 448
user settings, 26
creating, 114
`filemanager_path`, 467-468
in Evolution, 32
`manager_language`, 447
`tree_root_id`, 447, 466-468, 489
user-defined functions in PHP, 673-675
`user.input.html` file (transport packages), 614-615
`userinfo` (output modifier), 173, 566
userinfo fields, 173
`userLoggedIn()` method, 649
`userLoggedIn()` method (Evolution), 307
username user field, 381
users
active, 479
adding to user groups, 447
anonymous, 104, 381-383, 387, 460, 469, 474-475, 481
assigning roles to, 115
assigning to user groups, 447-448
authenticating, 302, 469
checking login status in code, 384-385, 469, 481
checking user group membership in code, 386-387
creating access permissions, 114-115
creating in code, 404
current user, 16, 318, 327, 381-383, 469, 485
database table, 479
interacting with in transport packages, 616-620
`isMember()` method, 386-387
`joinGroup()` method, 332, 408
`leaveGroup()` method, 332
list of currently logged in, 111
logged in or not, 469, 481
manager, 440-441, 446-447
manager (in Evolution), 497-498
manager system events, 710-711
object reference, 694
passwords in code, 342
redirecting based on custom permissions, 485-488
redirecting based on login status, 481, 490
redirecting based on `tree_root_id` setting, 489-496
redirecting in FormIt, 579-581, 586
redirecting in SPForm, 544-545, 550-551
redirecting on login, 385, 473-474
redirecting on page-not-found, 302-305, 475
redirecting on registration, 477-480
redirecting with `sendForward()`, 302-303, 648
redirecting with `sendRedirect()`, 481, 488, 490, 494-495, 648
Registered, 471
registering, 476-480
related to another resource, 389-391
related to current resource, 387-389
removing from user groups, 116, 447-448
removing from user groups in code, 332
resetting password, 127
setting password, 127
Unregistered, 471
web, 440-441, 446-447
web (in Evolution), 497-498
working with in code, 380-396
uses for
categories, 14
chunks, 232-244, 238-241
contexts, 424
default properties, 255
FormIt, 571, 581
getResources, 565, 571
lexicon strings, 347
plugin properties, 293-295
plugins, 13-14, 289, 302, 338, 432
snippet properties, 236-237
snippets, 12, 267, 302, 338, 370, 380-381, 387, 468
tags, 154-167
template variables, 10, 200, 204, 296, 377
Tpl chunks, 370
transport packages, 599
Wayfinder, 555
Uses Template resource field, 140
using
a content management system, 2
chunks in snippet properties, 236-237
chunks to store data, 238-241
chunks with snippets, 236-237
Git for transport packaged, 604-605
`isMember()` to select content tutorial, 387
lexicon in snippets tutorial, 411-413
lexicon strings in code, 347-351
properties with plugins, 293-295
property sets with plugins, 293-295
SMTP in Evolution, 596
SMTP with FormIt, 586
SMTP with SPForm, 550
tags in documents, 154-167
tags in templates, 199
tags to display template variables, 203
tags with chunks, 232
template variables in snippet properties, 204
template variables to control snippets, 204
Tpl chunks in snippets, 370-373
`utf-8`, 63

V

`validate` property for FormIt, 578
`validate()` method (transport

packages), 620
validating FormIt forms, 574-578
validators (FormIt), 574-578
validators (transport packages), 609, 619-620
value, 247
variable name TV field, 205
variables
  $_COOKIE, 699
  $_GET, 528, 646
  $_output, 300-301
  $_POST, 699
  $_SERVER['HTTP_REFERER'], 550
  $_SESSION, 301, 699
  $cacheFlag, 631-632, 633, 699
  $key, 344-345
  $modx, 310, 629, 699
  $modx->resource, 152, 290, 312-313, 318, 327-328, 332, 353-355, 387-388
  $modx->user, 318, 323, 327, 345, 381-389, 384, 385, 388-389, 469, 481, 490, 592, 592-593
  $output, 272, 656
  $resource, 311-313, 318, 354
  $resource (assignment by reference), 312-313
  $resource in plugins, 297-298
  $scriptProperties, 366-367, 374
  $value, 344-345
  $xpdo, 310, 325, 405, 629
  assignment by reference, 312-313
  available in FormIt custom hooks, 590
  in PHP, 655-657
  PHP string, 655
variables in PHP, 272
version control, 72-73
version control for transport packages, 604-605
version information in code, 645
versions of MODX, vii-x
view permission, 453, 459
View policy, 471
View Resource, 103, 180
View Resource panel
  cache output tab, 180
  changes tab, 180
  general tab, 180
view_tv permission, 514
view_user permission, 510
viewing cache output, 180
ViewOnly resource group, 459, 460
virtual domain, 431
virtual hosts, 429-431

W

warnings
  config file still writable, 45
  installer still present, 47
  register_globals is on, 45-47, 110
  unpublished error page, 110
  unpublished unauthorized page, 110
Wayfinder
  book, 555
  CSS, 559
  hide from menus checkbox, 560
  Javascript with, 559
  placeholders, 562
  properties, 558-559, 561, 564-565
  property set for, 562-563
  startId property, 558
  styling menus, 558-559
  tag, 195
  Tpl chunks, 558-559, 560-562
  tutorial, 555-565
  uses for, 555
web context, 25-26, 424-425, 432, 446-447, 457, 460, 468, 472, 484
web context permissions, 441
web page parsing, 21
web resources, 6
web users, 440-441, 446-447
web users in Evolution, 497-498
weblinks, 8-9, 176-177
  creating, 103
  Quick Create, 103
where() method for xPDO queries, 362-363, 393, 631-632, 638
which version of MODX to choose, vii-x
while loop (PHP), 671
Wiki, 128
wordwrap (output modifier), 171
wordwrapcut (output modifier), 171
workflow with Git, 83-86
working with
  BreadCrumbs,
  categories in code, 401-402
  chunks, 229-247
  default properties, 252-255
  extended user fields in code, 393-397
  EZfaq, 539-542
  form customization rules, 517-522
  FormIt, 571-594
  getChunk(), 368-370
  getResources, 565-571
  MODX security, 454-496
  plugins, 289-305
  properties in code, 365-368
  resources in code tutorial, 352-365
  runSnippet(), 373-375
  set(), 399
  setContent(), 399
  settings in code, 380
  snippets, 268-288
  SPForm, 543-555
  template variables in code, 377-378
  the current user in code, 381-383
  the lexicon in code, 411-413
  the Resource tree, 100-104
  Tpl chunks in code, 370-373
  user groups in code, 406-408
  users in code, 380-396
  Wayfinder, 555-565
workspaces, 25-26
writeSchema() method, 410

X

XAMPP
  configuring, 62-63
  downloading, 62
  enabling cURL, 536
  file locations, 63
  installing, 62-63
  mod_rewrite, 60
xPDO
  class methods, 629-652
  for custom database tables, 409-411
  help, 122
  logging errors, 635
  overview, x-xi
  transactions, 405-406
  xPDO object, 310
xPDO queries
  criteria, 311-313
  innerJoin() method, 393
  newQuery() method, 313, 393, 631-632, 635
  sortby() method, 631-632
  where() method, 362-363, 393, 631-632, 638
xPDO::LOG_LEVEL_DEBUG constant, 635
xPDO::LOG_LEVEL_ERROR constant, 635

`xPDO::LOG_LEVEL_FATAL` constant, 635
`xPDO::LOG_LEVEL_INFO` constant, 617, 618, 635
`xPDO::LOG_LEVEL_WARN` constant, 635
`XPDO_CLI_MODE` constant, 625
xPDOFileVehicle, 614-615
xPDOObject class, 637-639
`xPDOTransport::ABORT_ON_VEHICLE_FAIL` constant, 620
`xPDOTransport::ACTION_INSTALL` constant, 618
`xPDOTransport::ACTION_UNINSTALL` constant, 618
`xPDOTransport::ACTION_UPGRADE` constant, 618
`xPDOTransport::PACKAGE_ACTION` constant, 617
`xPDOTransport::PRESERVE_KEYS` constant, 606, 610, 621-622
`xPDOTransport::RELATED_OBJECT_ATTRIBUTES` constant, 621-623
`xPDOTransport::RELATED_OBJECT_ATTRIBUTES` in transport packages, 621-623
`xPDOTransport::RELATED_OBJECTS` constant, 621-623
`xPDOTransport::RELATED_OBJECTS` in transport packages, 621-623
`xPDOTransport::UNIQUE_KEY` constant, 606, 610, 621-622
`xPDOTransport::UPDATE_OBJECT` constant, 606, 610, 621-622

Y

YesNoDemo tutorial, 280-283