

Getting Started with IBM WebSphere sMash

Ron Lynn, Karl Bishop,
Brett King



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

© Copyright 2011 by International Business Machines Corporation. All rights reserved.

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

IBM Press Program Managers: Steven M. Stansel, Ellice Uffer

Cover design: IBM Corporation

Associate Publisher: Greg Wiegand

Marketing Manager: Kournaye Sturgeon

Acquisitions Editor: Katherine Bull

Publicist: Heather Fox

Development Editor: Kendell Lumsden

Managing Editor: Kristy Hart

Designer: Alan Clements

Project Editor: Lori Lyons

Copy Editor: Water Crest Publishing

Indexer: Lisa Stumpf

Compositor: Nonie Ratcliff

Proofreader: Apostrophe Editing Services

Manufacturing Buyer: Dan Uhrig

Published by Pearson plc

Publishing as IBM Press

IBM Press offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales

1-800-382-3419

corpsales@pearsontechgroup.com.

For sales outside the U.S., please contact:

International Sales

international@pearson.com.

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both: IBM, the IBM logo, IBM Press, Tivoli, and WebSphere. Java, JavaScript, JDK, JRE, JVM, and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both. Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Other company, product, or service names may be trademarks or service marks of others.

Library of Congress Cataloging-in-Publication Data

Lynn, Ron.

Getting started with IBM Websphere sMash / Ron Lynn, Karl Bishop, and Brett King.

p. cm.

Includes index.

ISBN-13: 978-0-13-701970-0

ISBN-10: 0-13-701970-X

1. Web site development. 2. WebSphere. I. Bishop, Karl. II. King, Brett, 1968- III. Title.

TK5105.8885.W43L96 2010

006.7--dc22

2010030753

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.
Rights and Contracts Department
501 Boylston Street, Suite 900
Boston, MA 02116
Fax (617) 671 3447

ISBN-13: 978-0-13-701970-0

ISBN-10: 0-13-701970-X

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

First printing September 2010

Thank you to all the developers out there who are tired of complex and cumbersome development, and those who had enough desire to do something different and, dare we say it, fun. We wrote this book for you. We hope you enjoy developing in WebSphere sMash using Groovy, PHP, normal Java, and JavaScript with Dojo. None of us can stand to write classic JEE applications anymore. If that doesn't speak to the value of this product, nothing does.

Contents

Introduction	1
Situational Applications	1
Rapid Application Development	1
IBM WebSphere sMash Development Process	2
Available IBM WebSphere sMash Offerings	2
What Is Covered in This Book?	3
Chapter 1 Installing the IBM WebSphere sMash CLI	5
First Things First: Java Development Environment	5
Installing the Command-Line Interface	6
Activating HTTP(S) Proxy Support	8
Test Your IBM WebSphere sMash Installation	9
Getting Started with the Command-Line Interface (CLI)	11
Conclusion	13
Chapter 2 Choose Your Development Environment	15
Introduction	15
AppBuilder	15
Getting Started	16
Sample Applications	16
Creating a New Application	18
Editing Applications	18
Eclipse	21
Sample Applications	21
Creating a New Project	23
Command-Line Interface Environment	27
Sample Applications	28

Creating a New Application	29
Deploying Your Application	30
Conclusion	30
Chapter 3 Your First Handler and Beyond	33
Introduction	33
Application Directory Layout	33
Source Directories	34
Supporting Directories and Files	35
REST	36
REST with the Zero Resource Model (ZRM)	38
Declaring a Dependency	41
Virtual Directories	43
Synchronizing a ZRM Model	43
Event Handling in Groovy	44
Running the Application	44
Explicit Event Handling	46
Event Handling in PHP	49
Event Handling in Java	52
Creating a Client	52
Groovy Templates	52
PHP	56
Dojo	59
Conclusion	61
Chapter 4 Configuration Files Explained	63
Application Configuration	63
Global Context and zero.config	63
Custom Configuration Data	64
Variable Substitution	65
Include Files	66
Handler Configuration	66
Dependency Management with Ivy	69
Ivy Modules	69
Ivy Files	69
Resolution and Resolvers	71
Environment Configuration	74
Useful Information About Your Application	74
Runtime Configuration	75
Response Configuration	75
Command-Line Interface (CLI) Config	77
App Builder Configuration	77
Eclipse Configuration	77
JVM Configuration	78

Overriding Configuration Parameters	79
Reverse Proxy Server Configuration	80
Conclusion	80
Chapter 5 Global Context	81
Zones	81
Non-Persistent Zones	81
Persistent Zones	84
Accessing the Global Context	85
Java APIs	86
Groovy APIs	100
PHP APIs	108
Conclusion	120
Chapter 6 Response Rendering	121
Every Conversation Requires a Response	121
Serving Static Files	122
Internationalizing Static Files	122
Serving Dynamic Content	124
PHP Rendering	124
Groovy Rendering	125
Serving Default Files	126
Directory Browsing	127
Custom Rendering States	128
Using Views for Rendering	128
Managing Errors	135
Data Rendering	138
JSON Data Rendering	138
XML Rendering	141
Conclusion	142
Chapter 7 REST Programming	143
What Is REST?	143
Response Codes	145
Request Accept Headers	147
Response Headers	148
REST Handling Within WebSphere sMash	149
Creating a Groovy Resource Handler	150
Creating a PHP Resource Handler	152
Content Negotiation	154
Bonding Resources Together	157
Error Handling and Response Codes	159
Enabling SSL Communication Handlers	160

Testing and Documentation	162
Conclusion	170
Chapter 8 Database Access	171
Introduction	171
Databases Supported in WebSphere sMash	172
Configuration Settings	172
Apache Derby	173
IBM DB2	175
MySQL	175
Oracle	176
Microsoft SQL Server	177
Zero Resource Model	177
Establishing a New ZRM Application	177
Creating a Zero Resource Model	178
Making ZRM Data Available as a Service	181
Adding Data to a Zero Resource Model	182
Loading Data Using a ZRM Test Page	183
Iterative Zero Resource Model Design	184
Database Access with pureQuery	186
Working with pureQuery	186
Simple Query Methods	188
Data Manipulation Statements	191
Prepared Statements	192
Externalizing SQL Statements	194
Connection Pooling	194
Data Access Using Java	195
Data Access in PHP	195
Standard JDBC Database Access	197
Command-Line Database Management	205
Conclusion	206
Chapter 9 Security Model	207
SSL Configuration	209
Enabling Security	213
Application Secret Key	213
Authentication Types	214
Login Form	217
Knowing Your Users	219
Additional Files for Our Application	221
Testing the Secure Application	223
Directory Server Configuration	224
Directory Server User Details	226

OpenID Configuration 228
 Securing Outbound Connections 230
 Conclusion 233

Chapter 10 Event Processing 235

Timers 235
 Application Initialization Using Timers 237
 Kickers 239
 Simple Kicker 240
 File Kicker and Receiver 243
 Events 245
 Custom Events 247
 Conclusion 249

Chapter 11 Framework Components 251

URIUtils 251
 Java APIs 251
 Groovy APIs 255
 PHP APIs 256
 Validators 257
 Active Content Filtering 259
 Assemble Flow 263
 Conclusion 267

Chapter 12 Client-Side Programming with the Dojo Toolkit 269

Enter the Dojo 270
 Enabling Dojo in Your Application 271
 AppBuilder Page Designer 277
 Put a Dojo Face on ZRM and Application Data 279
 DBA—A Complete RIA Using WebSphere sMash and Dojo 282
 Project Creation 283
 Layout Mockup 284
 Initial Page Loading 286
 Application Initialization 288
 Driver Details and Schema Loading 291
 Table Selection and Running SQL 293
 Final Product 294
 Creating Custom Dojo Builds for Performance 294
 Using Non-Supplied Versions of Dojo 295
 Debugging and Best Practices in Dojo Development 296
 Debugging and Logging with Firebug 297
 Code Validation with JSLint 297
 Data Validation with JSONLint 298

Dojo References	298
Conclusion	299
Chapter 13 PHP in WebSphere sMash	301
Why Develop in PHP Using sMash?	301
Adding PHP to Your Application	301
PHP Applications	302
Running PHP Applications in WebSphere sMash	303
PHP to Java Bridge	303
Accessing Java Classes	304
Access Static Java Class Members	304
Example: Using Apache Commons Logging in PHP	305
PHP to Groovy Bridge	308
PHP to Groovy Bridge Example	308
Extending PHP	311
Logger Extension Sample	313
Data Conversion Between PHP and Java in Extensions	315
PHP Arguments to Java Variables	315
Java to PHP Variable Conversion	317
SuperGlobals	317
\$_SERVER	318
\$_GET and \$_POST	318
\$HTTP_RAW_POST_DATA	319
\$_FILES	319
\$_COOKIE	320
\$_REQUEST	320
XML Processing Using PHP and WebSphere sMash	320
WebSphere sMash PHP Extensions	323
WebSphere sMash Utilities	323
URI Utilities	326
Java Extensions	327
Groovy Extensions	328
Remote Connections	329
JSON Utilities	330
Active Content Filtering	331
Cross-Site Request Forgery	331
Login	332
Database Access	332
XML Utilities	346
Conclusion	346
Appendix A Get Started with Groovy	349
Default Imports	350
Dynamic Typing	350

GStrings and Heredocs 351

Embedded Quotes 352

Getters and Field Pointers 352

Parentheses and Method Pointers 353

Return Statements 354

Exception Handling 354

Safe Dereferencing 355

Operator Overloading 355

Boolean Evaluation 356

Closures 357

Lists 358

Maps 361

Ranges 362

Looping 363

Optional Parameters 365

Index 367

Introduction

IBM® WebSphere® sMash is a platform for the rapid development and deployment web applications using popular web technologies. It quickly enables developers to go from concept to production in a fraction of the time required by traditional platforms and web application models. Developers can use the dynamic scripting languages Groovy and PHP to speed development and still have Java™ as the underlying system language for extension development. Add to this easy development of Representational State Transfer (REST) services and rich AJAX interfaces. This is a platform that excels at quickly getting Web 2.0 applications built. To speed deployment, the runtime environment is integrated, so there is no “server environment” to deploy to. Applications can be built to run standalone on any machine. This makes IBM WebSphere sMash an ideal platform for the development of situation applications.

Situational Applications

Situational applications are defined by Wikipedia as follows:

A **situational application** is “good enough” software created for a narrow group of users with a unique set of needs. The application typically (but not always) has a short life span, and is often created within the group where it is used, sometimes by the users themselves. As the requirements of a small team using the application change, the situational application often also continues to evolve to accommodate these changes. Although situational applications are specifically designed to embrace change, significant changes in requirements may lead to an abandonment of the situational application altogether—in some cases, it is just easier to develop a new one than to evolve the one in use.

Given the constantly changing business landscape, situational applications provide the ideal instrument for constant adaptation. As a platform for situational application, WebSphere sMash provides an ideal mix of features and flexibility allowing for rapid application development.

Rapid Application Development

The use of dynamic scripting languages is well acknowledged to improve software developer productivity. Groovy and PHP both have many build-in and reusable components and allow developers to create function with fewer lines of code. The use of dynamic scripting languages along with the integrated runtime environment also speeds the develop-compile-test loop by removing the need for compilation. Developers need to make changes only to their scripts, and IBM WebSphere sMash detects and executes the changed code. This sort of rapid development means that applications can be continuously tested and available to testers or end users even while an application is still being developed, extended, or evolved. Applications can also be

rapidly changed as the situation dictates. This type of continuous application adaptation fits the constantly changing business landscape perfectly.

IBM WebSphere sMash Development Process

IBM WebSphere sMash has been developed using a community-driven commercial development process. This means that most of the development efforts are done in the open and are transparent to all parties through a community-based website. The community website hosts the IBM WebSphere sMash incubator project, Project Zero. Project Zero is the technology for which the IBM WebSphere sMash product is derived. IBM encourages everyone to participate by contributing to the open discussions around the product development, features, functions, and bugs. The community website provides tools to facilitate participation. The tools include forums, a wiki, a bug-reporting tool, a developers blog, and access to the source repository. You can even download the latest stable and experimental versions of the product. This begs the question: Which version should I use?

Available IBM WebSphere sMash Offerings

IBM WebSphere sMash is available in four different offerings. The offerings range from fully supported with an IBM commercial license and extended features to bleeding-edge nightly builds of the latest and greatest code. The four offerings are summed up in Table 1.1.

Table 1.1 IBM WebSphere sMash Offerings

Offering	Features	Availability
IBM WebSphere sMash	Stable, production version of WebSphere sMash with a Standard IBM Commercial license.	Available for purchase from www.ibm.com/software/webservers/smash/
IBM Reliable Transport Extension for WebSphere sMash	Stable, production version of WebSphere sMash with a Standard IBM Commercial license. This version extends WebSphere sMash with features for reliable messaging and communication.	Available for purchase from www.ibm.com/software/webservers/smash/
IBM WebSphere sMash Development Edition	Stable, community version of WebSphere sMash with a license that allows for development and limited deployment.	Available for free from www.projectzero.org/download/
Project Zero	Experimental version of WebSphere sMash with features not yet available in the stable versions.	Available for free from www.projectzero.org/download/latest.php

From this list of offerings, it should be fairly easy to match up what level of investment you're looking for. Most developers start with the IBM WebSphere sMash Development Edition unless they're looking for the cutting-edge enhancements or would like to peek into the future. When the developers have established that WebSphere sMash is their preferred platform for development, the business would then invest in the production-ready and supported versions.

What Is Covered in This Book?

This book covers a wide range of topics of interest to software developers. The intent of this book is to supplement and expand upon the information available on the projectzero.org website and in the product documentation. Each chapter includes concrete examples and demonstrations of how to use the technology presented. You can find much of the code from this book at the IBM Press Books website: ibmpressbooks.com/sMash. We open the book by discussing installation and development environments. From our own experience as software engineers, we see that these are important aspects for the adoption of any new development environment. After we learn about installation and the available development environments, we move into a chapter that outlines a complete application. This chapter is intended to stand by itself and give you a quick introduction to the complete WebSphere sMash development cycle. After that, we dive into the details of many different aspects of WebSphere sMash. This book is by no means comprehensive. We've included what we believe to be important aspects to get you going and give hints into where there's more to be uncovered. We hope you enjoy this book and find it to be useful in learning WebSphere sMash.

This page intentionally left blank

Installing the IBM WebSphere sMash CLI

In the Introduction, we talked about the four different IBM WebSphere sMash offerings. For the purposes of this book, any version should be appropriate. So, feel free to download the latest Project Zero build or the latest IBM WebSphere sMash Developer Edition. Before we do that, though, let's prepare our Java development environment.

First Things First: Java Development Environment

Prior to installing the sMash CLI, you must ensure that you have a functional Java SE Development Kit (JDK™) version 5 or 6 installed. It is important to remember that you cannot use the Java SE Runtime Environment (JRE™) to develop with sMash. The AppBuilder development environment requires libraries found only in the JDK and not in the JRE. This is a common problem with first-time users. You may choose to use either the JDK from Sun or IBM. They can be found in the locations in Listing 1.1.

Listing 1.1 Locations of Supported JDKs

IBM: <http://www.ibm.com/developerworks/java/jdk/>
Sun: <http://java.sun.com/javase/downloads/index.jsp>

After you have downloaded a JDK, install it per the instructions provided for your operating system. The following steps show the configuration using the Linux® command line. For Windows® users, you need to open a command prompt and alter the statements appropriately. When installed, you can verify your JDK installation by running the commands in either Listing 1.2 or Listing 1.3.

Listing 1.2 Linux Java Version Check

```
$ java -version
java version "1.6.0"
Java(TM) SE Runtime Environment (build pxi3260sr3-20081106_07(SR3))
IBM J9 VM (build 2.4, J2RE 1.6.0 IBM J9 2.4 Linux x86-32 jvmsi3260-
20081105_25433 (JIT enabled, AOT enabled)
J9VM - 20081105_025433_lHdSMr
JIT - r9_20081031_1330
GC - 20081027_AB)
JCL - 20081106_01
```

Listing 1.3 Windows Java Version Check

```
C:\>java -version
java version "1.6.0"
Java(TM) SE Runtime Environment (build pwi3260sr2-20080818_01(SR2))
IBM J9 VM (build 2.4, J2RE 1.6.0 IBM J9 2.4 Windows XP x86-32
jvmwi3260-20080816
_22093 (JIT enabled, AOT enabled)
J9VM - 20080816_022093_lHdSMr
JIT - r9_20080721_1330ifx2
GC - 20080724_AA)
JCL - 20080808_02
```

The output shown in the listings indicates that I am running the IBM JDK version 1.6.0 on a Linux x86 platform. If running the Sun JDK, a similar output will be shown. The important thing is that you get JDK version information back and not some random error. Now that we have the JDK installed and have ensured that it is working, we can move on to downloading and installing WebSphere sMash.

Installing the Command-Line Interface

The installation for IBM WebSphere sMash is very easy. At a high level, you simply need to download the zip file and unzip it, set up a proxy server if needed, and test the installation. IBM WebSphere sMash is highly modular. It uses the ivy dependency manager (see <http://ant.apache.org/ivy/>) to manage dependencies for each application. Every application has tremendous flexibility to include only the modules the application needs. Developers also have the ability to create reusable modules for other applications to use.

For the purposes of this book, it is fine to download the latest stable WebSphere sMash Development Edition driver or the experimental Project Zero driver. You can download either from the Project Zero website (see Figure 1.1).



Figure 1.1 Project Zero website

The website, projectzero.org, will change over time, but you can find the download link to retrieve the zip file from the site. Remember, it is <http://projectzero.org>.

After you have the zip file on your computer, you need to unzip it using whatever extraction tool you have available. Unzip the contents into a directory of your choosing. The zip file has a base of zero, which you want to retain. In these samples, I am using a top-level directory called `development` located under my home directory. In the following, replace the location shown with your preferred directory:

```
unzip ~/Downloads/zero_<version>.zip -d ~/development/
```

Of course, if you're using a different operating system, your unzip command and paths will look different. After the files are extracted, you should end up with a directory structure similar to that shown in Figure 1.2 under your target directory.

The final step to finish the installation procedure consists of setting up your environment to include the location to the JRE and the Project Zero binaries. This will be accomplished by manipulating the `PATH` environment variable appropriately. Placing these commands in your normal login environment ensures that everything is ready to go the next time you log in. This is an optional step, but will save you the hassle of having to specify full paths when running the Zero command line. Examples of these paths for Linux are shown in Listing 1.4.

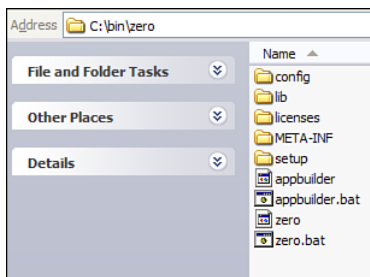


Figure 1.2 Zero directory structure

Listing 1.4 Example Linux Paths

```
export JAVA_HOME=/usr/lib/jvm/java-1.5.0-ibm-1.5.0.8/jre
export ZERO_HOME=~/.development/zero
export PATH=$JAVA_HOME/bin:$ZERO_HOME:$PATH
```

For Windows users, you can set the environment variables by accessing the Control Panel > System > Advanced tab > Environmental Variables. Then, in either the user or system areas, add/edit the following values, as shown in Listing 1.5. Be sure to adjust the directory paths to match your actual environment.

Listing 1.5 Example Windows Paths

```
JAVA_HOME    C:\Program Files\Java\jdk1.6.0_16\jre
ZERO_HOME    C:\development\zero
PATH         %JAVA_HOME%\bin;%ZERO_HOME%\bin;%PATH%
```

This completes the basic installation of IBM WebSphere sMash. The next section discusses setting up proxy support. However, if your connection to external websites doesn't require you to use a proxy, skip the proxy section and continue to the section on testing your installation.

Activating HTTP(S) Proxy Support

The command-line interface (CLI) retrieves modules from remote repositories. For the command-line interface to do this, Java needs access the Internet. If you use a proxy server to access external websites, you need to set up the proxy server for IBM WebSphere sMash. If you don't know if you use a proxy server, skip this section and try testing your WebSphere sMash installation. If you receive an unknown host error message, you probably need to come back to this section and set up your proxy server. The easiest way to do this is to edit the zero script file for Linux or zero.bat file for Windows in the installation directory.

Open the script file or the batch file that is appropriate for your platform in your favorite text editor. At the top of the file, there are comments that direct you to uncomment particular lines and update the values (see Listing 1.6 for Linux and Listing 1.7 for Windows).

Listing 1.6 Linux Zero Script Fragment Demonstrating Proxy Configuration

```
# proxy support -
#   uncomment the following line( and the export ZERO_OPTS) and
#   update proxy values
# ZERO_OPTS="$ZERO_OPTS
#     -Dhttp.proxyHost=myProxyHost
#     -Dhttp.proxyPort=myProxyPort
#     -Dhttps.proxyHost=myProxyHost
#     -Dhttps.proxyPort=myProxyPort"
# export ZERO_OPTS
```

For Linux, uncomment by removing the # sign at the start of the ZERO_OPTS line and the start of the export line. Set the proxy variables to the appropriate values, and your proxy is set up.

Listing 1.7 Windows Zero Batch File Fragment Demonstrating Proxy Configuration

```
rem proxy support -
rem   uncomment the following line and
rem   update values to add proxy settings.
rem set ZERO_OPTS=%ZERO_OPTS%
-Dhttp.proxyHost=myProxyHost
-Dhttp.proxyPort=myProxyPort
-Dhttps.proxyHost=myProxyHost
-Dhttps.proxyPort=myProxyPort
```

For Windows, uncomment by removing the rem at the start of the set ZERO_OPTS. Set the proxy variables to the appropriate values, and your proxy is set up.

Now that your proxy is set, we can continue on and sanity test our IBM WebSphere sMash installation.

Test Your IBM WebSphere sMash Installation

The quickest way to verify that your IBM WebSphere sMash installation is installed correctly is to run a version check. To do this, open a command line and change directories to the zero installation directory. In this directory, there is a script for Linux named zero and a batch file for Windows named zero.bat. On the command line, type zero version. If your installation has been done properly, you should see output similar to Listing 1.8.

Listing 1.8 Output from the “Zero Version” Command on Windows

```
C:\temp\zero>zero version
CWPZT0901I: The following module(s) are not currently in the local
  repository:
      zero:zero.cli.tasks:[1.0.0.0,)
CWPZT0902I: Trying to locate the module(s) using one of the config-
  ured remote repositories
CWPZT0545I: Retrieving zero.cli.tasks-1.1.1.1.30731.zip from host
http://www.projectzero.org/sMash/1.1.x/repo/base
CWPZT0545I: Retrieving zero.kernel-1.1.1.1.30725.zip from host
http://www.projectzero.org/sMash/1.1.x/repo/base
CWPZT0600I: Command resolve was successful
Command-line Version:  1.1.1.1.30719 20090908 2152
Command-line Information:
    Name: zero.cli
    Version: 1.1.1.1.30719
    Location: C:\temp\zero
    Modulegroup: stable

    Dependencies:
        zero:zero.cli.tasks:1.1.1.1.30731 (userhome)
        zero:zero.kernel:1.1.1.1.30725 (userhome)

The java command path is
C:\bin\ibm-java-sdk-60-win-i386\jre\bin\java.exe

java version "1.6.0"
Java(TM) SE Runtime Environment (build pwi3260sr2-20080818_01(SR2))
IBM J9 VM (build 2.4, J2RE 1.6.0 IBM J9 2.4 Windows XP x86-32
jvmwi3260-20080816_22093 (JIT enabled, AOT enabled)
J9VM - 20080816_022093_1HdSMr
JIT - r9_20080721_1330ifx2
GC - 20080724_AA)
JCL - 20080808_02
```

This listing came from a Windows-based machine, so the output will look slightly different than what you might see on a Linux machine. Also, the versions and paths will be appropriate for the version of IBM WebSphere sMash you downloaded, the version of Java you’re using, and the paths you have selected for installation.

You should notice at the top of the output that IBM WebSphere sMash automatically retrieves certain modules from the remote repository. Namely, it retrieves the CLI tasks and the zero kernel modules. These modules are necessary for IBM WebSphere sMash to run and, if successfully downloaded, indicate that you have connectivity to the remote repository.

If you receive errors during the download of these modules, you need to fix those errors before moving on. An easy test to see if you have connectivity to the repository is to take the URL listed in the output in Listing 1.8 and plug it into your web browser. In Listing 1.8, the URL we'd use is the following:

```
http://www.projectzero.org/sMash/1.1.x/repo/base
```

This URL will be different depending on the version of IBM WebSphere sMash you are using. If you have access to the URL from your web browser, but see errors using the zero version check, you probably need to go back to the proxy section and set up a proxy server for IBM WebSphere sMash to use.

When the zero version check works, you can get started using the command-line interface.

Getting Started with the Command-Line Interface (CLI)

In the next chapter, we look at various development environments, including the command-line interface (CLI). Let's take a quick look at the CLI now just to get acquainted with it. As we saw, the CLI is a module retrieved from a remote module repository. There are many commands available, but don't let this scare you. The CLI is very helpful. To get a list of available commands, type the following:

```
zero help
```

The results of typing this into the command line can be seen in Listing 1.9.

Listing 1.9 Results of the "Zero Help" Command

```
Usage: zero [main-opts] <task> [task-args]
main-opts:
  -d          enable minimal logging
  -v          enable all logging
  -l=<file>  use the given file to output log to

compile      Compiles all Java source files under the module's
/java        directory.
create       Creates a new module in the current directory.
help         Prints the documentation for the given command.
modulegroup Manage modulegroups.
package      Packages the module as a zip file.
```

publish	Publish the module to your local repository.
repository	Manage repositories.
resolve	Determine the module's dependencies.
rollback	Reverts the effects of the last resolve or update.
search	Finds and prints all matching modules in the repository.
switch	Switch the module group.
update	Resolves a module to its latest dependencies.
version	Displays version information

As you can see, there are a bunch of available commands. Each command may have arguments that it can be given, but again the command line is very helpful. You need only to type the following to get help on a particular command:

```
zero help <command>
```

For example, if we use `zero help search`, we get the results shown in Listing 1.10.

Listing 1.10 Results of “Zero Help Search” in the CLI

Usage:

```
zero search <org:module[:revision]> [-remote] [-json]
```

The command uses the current module group to find in the local repository for a module that matches the given `<org:module:revision>` value and prints its search results. If no module name is given, the command will print information about all available modules. You can also use the `-remote` option to tell the command to search the remote repository instead of the local one.

If the `-json` option is used the report is formatted as a JSON document.

The return codes for this command are:

```
0 - success
1 - command failed
```

The search command enables you to search for modules in the repository. There are commands to create new IBM WebSphere sMash applications, start and stop applications, package applications, and many more. Spend some time familiarizing yourself with the commands, and we'll take another look at the CLI in the next chapter.

Conclusion

In this chapter, we learned how to install IBM WebSphere sMash and how to test that installation. We also started working with the command-line interface (CLI) and became familiar with the available commands. In the next chapter, we're going to take a look at three different development environments that you can use to develop IBM WebSphere sMash applications. The first is a web browser-based environment called AppBuilder that is built into IBM WebSphere sMash. The second is an Eclipse-based development environment. Finally, we'll come back to the CLI and take a look at what else it can do.

Index

Numbers

200:[HTTP_OK], 145
201:[HTTP_CREATED], 145
202:[HTTP_ACCEPTED], 145
2xx: Successful response code class, 145
400:[HTTP_BAD_REQUEST], 145-146
401:[HTTP_UNAUTHORIZED], 146
403:[HTTP_FORBIDDEN], 146
404:[HTTP_NOT_FOUND], 146
405:[HTTP_BAD_METHOD], 146
406:[HTTP_NOT_ACCEPTABLE], 146
4xx: Client-level errors, 145
500:[HTTP_SERVER_ERROR], 146
503:[HTTP_UNAVAILABLE], 146
5xx: Server-related failures, 146

A

Accept header, 147
Accept-Charset, 147
Accept-Encoding, 147

Accept-Language headers, 147

accessing

data

with Java, 195
in PHP, 195-196

global context, 85

Java classes, PHP, 304

static Java class members,
PHP, 304-305

ACF (Active Content Filtering),
259-263

extensions, 331

acf_process, 331

acf_process_stream, 331

acf_validate, 331

acf_validate_stream, 331

activating HTTP proxy support,
8-9

Active Content Filtering. *See*

ACF (Active Content Filtering)

adding

data to ZRM (Zero Resource
Model), 182-183

dependencies

AppBuilder, 19-21

CLI environment, 30

Dojo, 271

in Eclipse, 24-27

to lists, 359

PHP to applications, 301-302

Allow header, 148

Apache Commons, logging into
PHP, 305-307

Apache Derby, 173-175

app directories, 35

app zones, 85

AppBuilder, 15-16

applications

creating, 18

editing, 18-19

sample applications, 16

configuration, 77

page designer, 277-279

AppBuilder, adding

dependencies, 19-21

app/errors subdirectories, 35

app/iwidgets directories, 35

application configuration

custom configuration data,
64-65

Global Context and
zero.config, 63-64

include files, 66

variable substitution, 65-66

application initialization, timers,
237-239

- applications
 - AppBuilder
 - creating, 18
 - editing, 18-19
 - CLI environment
 - creating new, 29
 - deploying, 30
 - sample applications, 28-29
 - directories, 33-34
 - source directories, 34-35
 - supporting, 35-36
 - Eclipse, sample applications, 21-23
 - initializing in Dojo, 288-291
 - PHP, 302-303
 - adding, 301-302
 - running, 44-46
 - secret keys, 213-214
 - testing secure applications, 223-224
 - ZRM (Zero Resource Model), establishing applications, 177
 - app/models directories, 35
 - app/resources directories, 35
 - app/scripts directories, 35
 - app/views directories, 35
 - app/zwidgets directories, 35
 - arguments, PHP to Java variables, 315-316
 - Assemble Flow, 263-267
 - authentication, 207
 - form-based authentication
 - configuration settings, 216-217
 - LDAP (Lightweight Directory Access Protocol), 225
 - authentication types, SSL (Secure Sockets Layer)
 - configuration, 214-217
- B**
- black hats, 259
 - bonding, resources, 157-159
 - bonding files, 67
 - bookmarks, 44-45
 - boolean evaluation, Groovy, 356-357
 - browsing directories, 127-128
 - bytes, 334
- C**
- classes directory, 36
 - .classpath file, 36
 - CLI (command-line interface), 11-12
 - installing, 6-8
 - CLI configuration, 77-80
 - CLI environment, 15, 27-28
 - applications
 - creating new, 29
 - deploying, 30
 - sample applications, 28-29
 - CLI environment, adding
 - dependencies, 30
 - clients, creating, 52
 - with Dojo, 59-60
 - Groovy templates, 52-55
 - in PHP, 56-59
 - closures, Groovy, 357-358
 - code validation, JSLint, 297
 - collect method, 360
 - command-line database management
 - runsql {dbKey} {sqlFile}, 205-206
 - validatedb {dbKey}, 205
 - command-line interface. *See* CLI (command-line interface)
 - communication handlers, enabling SSL, 160-162
 - concatenation
 - lists, 359
 - maps, 362
 - conditions, event zones, 68-69
 - confidentiality, 207
 - config directories, 36
 - config zone, non-persistent zones, 81-82
 - config/ivy.xml, 36
 - config/php.ini, 36
 - configuration, 63
 - App Builder configuration, 77
 - application configuration. *See* application configuration
 - CLI configuration, 77-80
 - communication
 - configuration, 161
 - directory servers, 224-226
 - Eclipse configuration, 77-78
 - environment configuration. *See* environment configuration
 - handler configuration, 66-68
 - JVM configuration, 78
 - OpenID, 228-230
 - overriding configuration parameters, 79-80
 - response configuration, 75-76
 - reverse proxy server
 - configuration, 80
 - runtime configuration, 75
 - SSL (Secure Sockets Layer), 209-212
 - authentication types, 214-217
 - enabling security, 213
 - zero user configuration, 220-221
 - configuration settings, databases, 172-173
 - configuring event handlers, 245-246
 - config/zero.config, 36
 - connection pooling, 194-195
 - connection zones, non-persistent zones, 84
 - connection_get, 330
 - connection_post, 330
 - connections, remote connections, 329-330
 - content negotiation, 154-157
 - Content-Encoding headers, 148
 - Content-Language headers, 148
 - Content-Type headers, 148
 - \$_COOKIE, 320
 - CSRF (Cross Site Request Forgery), 213
 - extensions, 331-332

csrf_protected_form_field, 332
 csrf_protected_uri, 332
 custom configuration data,
 application configuration,
 64-65
 custom Dojo builds, creating for
 performance, 294-295
 custom events, 247-249
 custom renderers, 128

D

data

 accessing in PHP, 195-196
 accessing with Java, 195
 adding to ZRM (Zero
 Resource Model), 182-183
 loading to ZRM Test page,
 183-184
 rendering, 138
 JSON (JavaScript Object
 Notation), 138-141
 data manipulation statements,
 pureQuery, 191-192
 data validation, JSONLint, 298
 data_begin_transaction, 345
 data_blob_get_bytes, 341-342
 data_blob_length, 341-342
 data_blob_set_bytes, 341
 data_clob_get_string, 341
 data_clob_length, 342
 data_commit_transaction, 345
 data_end_transaction, 345
 data_exec, 335-336
 data_exec_opt, 335-337
 data_insert, 336
 data_is_in_transaction, 345
 data_is_valid, 333
 data_iter_has_next, 342
 data_iter_next, 342
 data_iter_remove, 342
 data_last_error, 333
 data_manager, 333
 data_new_data_source, 334
 data_new_manager, 333
 data_new_result_handler, 343
 data_query, 337
 data_query_array, 338

 data_query_array_by_factory,
 338
 data_query_first, 338
 data_query_first_by_factory, 339
 data_query_interator, 339
 data_query_iterator_by_factory,
 339
 data_query_results, 340
 data_rollback_transaction, 346
 data_rs_absolute, 343
 data_rs_close, 343
 data_rs_get_column_count, 343
 data_rs_get_column_name, 343
 data_rs_get_object, 344
 data_rs_get_row, 344
 data_rs_next, 344
 data_rs_previous, 344
 data_source, 334
 data_string_as_byte_array, 334
 data_transaction, 346
 data_update, 340
 data_update_many, 340
 database access, extensions, 332
 Database Administrator
 Application (DBA), 203-204
 database query functions,
 334-335
 database results functions, 341
 database transaction functions,
 345-346
 databases
 Apache Derby, 173-175
 configuration settings,
 172-173
 IBM DB2, 175
 Microsoft SQL Server, 177
 MySQL, 175-176
 Oracle, 176
 supported in WebSphere
 sMash, 172
 dataBeginTransaction, 345
 dataBlobGetBytes, 341, 342
 dataBlobLength, 341, 342
 dataBlobSetBytes, 341
 dataClobGetString, 341
 dataClobLength, 342
 dataCommitTransaction, 345
 dataEndTransaction, 345

 dataExec, 335, 336
 dataExecOpt, 335, 337
 dataInsert, 336
 dataIsInTransaction, 345
 dataIsValid, 333
 dataIteratorHasNext, 342
 dataIteratorNext, 342
 dataIteratorRemove, 342
 dataLastError, 333
 dataManager, 333
 dataNewDataSource, 334
 dataNewManager, 333
 dataNewResultHandler, 343
 dataQuery, 337
 dataQueryArray, 338
 dataQueryArrayByFactory, 338
 dataQueryFirst, 338
 dataQueryFirstByFactory, 339
 dataQueryIterator, 339
 dataQueryIteratorByFactory, 339
 dataQueryResults, 340
 dataResultSetAbsolute, 343
 dataResultSetClose, 343
 dataResultSetGetColumnCount,
 343
 dataResultSetGetColumnName,
 343
 dataResultSetGetObject, 344
 dataResultSetGetRow, 344
 dataResultSetNext, 344
 dataResultSetPrevious, 344
 dataRollbackTransaction, 346
 dataSource, 334
 DataSources, Dojo, 282
 DataStore, Dojo, 279
 dataStringAsByteArray, 334
 dataTransaction, 346
 dataUpdate, 340
 dataUpdateMany, 340
 DBA
 final application, 294
 layout, 284-285
 RIA (Rich Internet
 Application), 282-283
 debugging
 Dojo, 297
 Firebug, 297

- declaring dependencies, REST, 41-42
 - default files, serving, 126-127
 - dependencies
 - adding
 - in AppBuilder, 19-21
 - in CLI environment, 30
 - Dojo, 271
 - in Eclipse, 24-27
 - declaring in REST, 41-42
 - Dependencies tab, 261
 - dependency management, Ivy. *See* Ivy
 - deploying applications, CLI environment, 30
 - dereferencing Groovy, 355
 - development environments
 - AppBuilder. *See* AppBuilder
 - CLI environment. *See* CLI environment
 - Eclipse. *See* Eclipse
 - digital signing, 208
 - Dijit Property editor, 278
 - directories, 33-34
 - app directories, 35
 - app/iwidgets directories, 35
 - app/models directories, 35
 - app/resources directories, 35
 - app/scripts directories, 35
 - app/views directories, 35
 - app/zwidgets directories, 35
 - browsing, 127-128
 - classes directory, 36
 - config directories, 36
 - java directories, 35
 - lib directories, 36
 - logs directories, 36
 - public directories, 35
 - reports directories, 36
 - source directories, 34-35
 - supporting, 35-36
 - virtual directories, REST, 43
 - WebSphere sMash
 - application directory structure, 34
 - directories, virtual directories (REST), 43
 - directory servers
 - configuration, 224-226
 - user details, 226-228
 - DOCROOT, 324
 - documentation, REST, 162-170
 - documentation annotations, 165
 - Dojo, 270-271
 - adding logic to pages, 274
 - applications, initializing, 288-291
 - best practices, 297
 - creating clients, 59-60
 - creating custom Dojo builds
 - for performance, 294-295
 - DataSources, 282
 - DataStore, 279
 - DBA, 282-283
 - final application, 294
 - debugging, 297
 - dependencies, adding, 271
 - driver details, 291-293
 - enabling, 271-277
 - initial page loading, 286-287
 - layout mockup, 284-286
 - non-supplied versions of, 295-297
 - project creation, 283
 - references, 298-299
 - schema loading, 291-293
 - SQL, running, 293
 - table selection, 293
 - widgets, 270
 - custom widgets, 274
 - ZRM (Zero Resource Model), 279-282
 - dojo.byId() function, 275
 - dojox, 270
 - driver details, Dojo, 291-293
 - dynamic content, serving, 124
 - dynamic SQL, 201-203
 - dynamic typing, Groovy, 350-351
- ## E
- each method, 364-365
 - Eclipse, 15, 21
 - adding, dependencies, 24-27
 - applications, sample applications, 21-23
 - configuration, 77-78
 - creating new projects, 23
 - editing applications, AppBuilder, 18-19
 - embedded quotes, Groovy, 352
 - enter zones, non-persistent zones, 83
 - environment configuration, 74
 - URIs, 74
 - @error, 164
 - error codes, HTTP, 145-146
 - error handling
 - PHP, 197
 - response codes, 159-160
 - errors, managing, 135-138
 - ETags, 257-259
 - event handlers, configuring, 245-246
 - event handling, 46-49
 - Groovy, 44
 - in Java, 52
 - in PHP, 49-51
 - event processing
 - file kicker/receiver, 243-245
 - kickers, 239-240
 - simple kickers, 240-243
 - timers, 235-237
 - application initialization, 237-239
 - event to method mapping, 149-150
 - event zones, conditions, 68-69
 - events, 245-247
 - custom events, 247-249
 - JSON (JavaScript Object Notation), 68
 - @example, 165
 - exception handling, Groovy, 354-355
 - extending PHP, 311-313

extensions

- ACF (Active Content Filtering), 331
 - CSRF (Cross Site Request Forgery), 331-332
 - database access, 332
 - Groovy, 328-329
 - Java, 327-328
 - login, 332
 - remote connections, 329-330
- externalizing SQL statements, 194

F

- fetch function, 292
- field pointers, Groovy, 352-353
- file kicker, 243-245
- \$_FILES, 319-320
- files
 - Ivy, 69-71
 - supporting, 35-36
- findAll method, 360
- fire_event, 324
- Firebug
 - debugging, 297
 - logging, 297
- FirstElementLists
 - Groovy APIs, 104
 - zcontains method, 105-106
 - zdelete method, 105
 - zget method, 104-105
 - zpost method, 104
 - zput method, 104
- Java APIs
 - lists, 93
 - zput method, 93-94
- PHP APIs, 113-114
 - zcontains method, 116-117
 - zdelete method, 116
 - zget method, 115-116
 - zpost method, 114-115
 - zput method, 114
- zcontains method, 96-97
- zdelete method, 96
- zget method, 95
- zpost method, 94

- zputs method, 94-95
- flatten method, 361
- Foo.groovy, 247-248
- form-based authentication
 - configuration settings, 216-217
- @format, 165
- fuzzy search, prepared statements, 193

G

- \$_GET, 318-319
- get_absolute_uri, 326
- get_relative_uri, 327
- get_requested_uri, 327
- getAttribute, 347
- getters, Groovy APIs, 352-353
- getVFile, 324
- global context
 - accessing, 85
 - Groovy APIs, 100
 - FirstElementLists. *See* FirstElementLists
 - lists, 102
 - lists, zcontains method, 103
 - lists, zdelete method, 103
 - lists, zget method, 102-103
 - lists, zpost method, 102
 - lists, zput method, 102
 - maps. *See* maps
 - objects, zcontains method, 101-102
 - objects, zdelete method, 101
 - objects, zget method, 101
 - objects, zput method, 101
- Java APIs
 - FirstElementLists, zcontains method, 96-97
 - FirstElementLists, zdelete method, 96
 - FirstElementLists, zget method, 95
 - FirstElementLists, zpost method, 94
- FirstElementLists, zput method, 93-94
- FirstElementLists, zputs method, 94-95
- lists, FirstElementLists, 93
- lists, zcontains method, 92-93
- lists, zdelete method, 92
- lists, zget method, 91
- lists, zpost method, 90
- lists, zput method, 89-90
- lists, zputs method, 90-91
- maps, 97
- maps, zcontains method, 100
- maps, zdelete method, 99-100
- maps, zget method, 99
- maps, zpost method, 98
- maps, zput method, 97-98
- maps, zputs method, 98-99
- objects, zcontains method, 87-88
- objects, zdelete method, 87
- objects, zdump method, 89
- objects, zget method, 86-87
- objects, zlist method, 88-89
- objects, zput method, 86
- objects, zputs method, 86
- PHP APIs. *See* PHP APIs
- Global Context, zero.config and, 63-64
- Grooving, XML rendering, 142
- Groovy, 124, 349-350
 - boolean evaluation, 356-357
 - closures, 357-358
 - default files, 126
 - default imports, 350
 - dynamic typing, 350-351
 - embedded quotes, 352
 - event handling, 44

- exception handling, 354-355
 - extensions, 328-329
 - field pointers, 352-353
 - getters, 352-353
 - GStrings, 351
 - heredocs, 351
 - lists, 358-361
 - looping, 363-365
 - maps, 361-362
 - method pointers, 353
 - operator overloading, 355-356
 - optional parameters, 365
 - parentheses, 353
 - ranges, 362-363
 - rendering, 125-126
 - resource handlers, creating, 150-152
 - return statements, 354
 - safe dereferencing, 355
 - templates, creating clients, 52-55
 - View files, 129
 - Groovy APIs
 - FirstElementLists, 104
 - zcontains method, 105-106
 - zdelete method, 105
 - zget method, 104-105
 - zpost method, 104
 - zput method, 104
 - global context, 100
 - lists, 102
 - zcontains method, 103
 - zdelete method, 103
 - zget method, 102-103
 - zpost method, 102
 - zput method, 102
 - maps, 106
 - zcontains method, 107-108
 - zdelete method, 107
 - zget method, 106-107
 - zpost method, 106
 - zput method, 106
 - objects
 - zcontains method, 101-102
 - zdelete method, 101
 - zget method, 101
 - zput method, 101
 - URIUtils, 255
 - Groovy bridges, PHP, 308
 - examples, 308-311
 - groovy_create_closure, 328
 - groovy_eval, 329
 - groovy_import, 329
 - GroovyBean, 190
 - Groovy-Ldap, 226
 - GStrings, Groovy, 351
- H**
- handler configuration, 66-68
 - handlers, 44
 - headers
 - last-modified headers, 257-258
 - Request Accept headers, 147
 - Response headers, 148
 - heredocs, Groovy, 351
 - HTTP
 - error codes, 145-146
 - ETags, 257-259
 - HTTP Get request, Assemble Flow, 265
 - HTTP methods, REST, 37-38
 - HTTP proxy support, activating, 8-9
 - \$HTTP_RAW_POST_DATA, 319
- I**
- IBM DB2, 175
 - IBM Reliable Transport Extension for WebSphere sMash, 2
 - IBM WebSphere sMash
 - installation, testing, 9-11
 - offerings, 2
 - PHP, 301
 - PHP applications,
 - running, 303
 - PHP extensions, 323
 - REST, 149-150
 - IBM WebSphere sMash Development Edition, 2
 - ikeman utility, 209-212
 - include files, application configuration, 66
 - initializing applications, Dojo, 288-291
 - installing IBM WebSphere sMash, testing installation, 9-11
 - command-line interface, 6-8
 - instanceData members, kickers, 240
 - integrity, 207
 - internationalizing static files, 122-124
 - iterating over lists, 364
 - iterative Zero Resource Model Design, 184-186
 - Ivy, 69
 - files, 69-71
 - modules, 69
 - resolution and resolvers, 71-72
 - local resolvers, 73-74
 - remote resolvers, 73
 - revision patterns, 70
- J**
- Java
 - accessing data, 195
 - event handling, 52
 - extensions, 327-328
 - PHP arguments to Java variables, 315-316
 - to PHP variable conversion, 317
 - Java Absolute URI API, 253
 - Java APIs
 - FirstElementLists
 - zcontains method, 96-97
 - zdelete method, 96

- zget method, 95
 - zpost method, 94
 - zput method, 93-94
 - zputs method, 94-95
 - lists
 - FirstElementLists, 93
 - zcontains method, 92-93
 - zdelete method, 92
 - zget method, 91
 - zputs method, 90-91
 - lists, zput method
 - zpost method, 90
 - zput method, 89-90
 - maps, 97
 - zcontains method, 100
 - zdelete method, 99-100
 - zget method, 99
 - zpost method, 98
 - zput method, 97-98
 - zputs method, 98-99
 - objects
 - zcontains method, 87-88
 - zdelete method, 87
 - zdump method, 89
 - zget method, 86-87
 - zlist method, 88-89
 - zput method, 86
 - zputs method, 86
 - URIUtils, 251-255
 - Java bridges, PHP, 303-304
 - Java classes, accessing from PHP, 304
 - java directories, 35
 - Java Relative URI API, 252
 - Java requested URI API, 254
 - Java SE Development Kit. *See* JDK
 - java_import, 328
 - JDBC, pureQuery, 197-198
 - columns for tables, 200-201
 - Database Administrator Application, 203-204
 - locating all database definitions and details, 198
 - obtaining database schema details, 198-199
 - obtaining tables from schema entries, 199
 - processing dynamic SQL, 201-203
 - JDK (Java SE Development Kit), 5-6
 - join method, 360
 - JSLint, code validation, 297
 - JSON (JavaScript Object Notation), 38
 - data rendering, 138-141
 - events, 68
 - utilities, 330-331
 - JSON (JavaScript Object Notation), rendering with PHP, 140-141
 - json_decode, 330
 - json_encode, 331
 - JSONLint, data validation, 298
 - JVM configuration, 78
- K-L**
- kickers, 239-240
 - simple kickers, 240-243
 - last-modified headers, 257-258
 - layout mockup, Dojo, 284-286
 - LDAP (Lightweight Directory Access Protocol), 224
 - authentication, 225
 - lib directories, 36
 - listFiles, 324
 - listings
 - Adding a Secret Key to the Application's config File, 213
 - Adding Elements in a Map, 362
 - Adding i18n Support in zero.config, 122
 - Adding Some Logic to the Page, 274
 - Adding to a List, 359
 - Another Implicit Return Statement, 354
 - appbuilder Usage, 16
 - /app/config/timers/heartbeat.groovy, 236
 - /app/errors/error403.gt, 219
 - /app/errors/error.gt—Errors Is as Errors Do, 137
 - Application Code to Call a Secured Service, 232
 - /app/resources/bookmark.groovy, 181
 - /app/resources/bookmark.php, 181
 - /app/resources/car.bnd—Bonding File for Car Resources, 158
 - /app/resources/car.groovy—Car Resource Handler, 151-152
 - /app/resources/car.groovy—Dynamic Content Negotiation, 156
 - /app/resources/car.groovy—Filtered Car Resources Selection, 158-159
 - /app/resources/car.groovy—RESTDoc Annotations, 165-166
 - /app/resources/car.php—Alternate Car Resource Handler in PHP, 153-154
 - /app/resources/car.php—Car Resource Handler in PHP, 153
 - /app/scripts/kickers/mySimpleKicker.groovy, 241
 - /app/scripts/quiz.groovy—Business Logic Controller, 132-133
 - /app/scripts/rest/headers.groovy—Process Request Headers, 155
 - /app/scripts/rest/send.groovy—Response Data Helper Functions, 156-157
 - /app/scripts/timers/init.groovy, 238-239
 - /app/views/quiz/quiz.gt—Quiz View Page, 134

- `/app/views/quiz/theme/main.gt(fragment)`—Include Our Content Page, 133
- `/app/views/theme/nav.gt`—Navigation View Block, 130-131
- `args.groovy`—Simple Groovy Script Example, 126
- `args.gt`—Simple Groovy Templates Example, 126
- Assigning a Field Pointer to a Variable, 353
- The BAD Way to Use One-Off Functions, 275-276
- Bookmark GET Response, 45
- Bookmarks Fixture, 40
- Bookmarks Model, 39
- Bookmarks Response, 45
- Calling the GET Method, 53
- Changing Default Variable Access, 352
- Common Template for PHP Extension Classes, 312
- Conditions Expression, 68
- Configure Event Handlers for the `requestBegin`, `log`, and `requestEnd` Events, 245-246
- `/config/zero.config`—SSL-Only Configuration Sample, 162
- Create Form, 54
- Create the New Bookmark, 54-55
- Creating a Date Range, 363
- Creating a
 - `firstElementArrayList`, 93
- Creating a Map in Groovy, 361
- Creating an ETag Validator, 257
- Creating the Last-Modified Header, 258
- Custom Query for Dojo URLs as Strings, 189
- Database Access Using PHP, 195
- Database Manipulation Scripts, 206
- DB2 Database
 - Configuration, 175
- DBA Page Initialization, 287
- DBA Script Initialization, 288-290
- Decode Results and Display, 53
- Default Imports, 350
- Default “it” Variable, 357
- Defining a DataGrid, 60
- Defining a DataStore, 59
- Defining Outbound Connections in Configuration File, 231
- Delete and Redirect, 53
- Dojo-Enabled `index.html` Page, 272
- Dynamic Variable Types, 350
- Embedded Derby Database Configuration, 174
- Embedded Expressions, 351
- Enable Pretty Print for JSON Data, 139
- Enable Security Configuration Setting, 213
- Enabling the Debug Console on the `index.html`, 273
- Enhanced User and Group Details Page, 227-228
- Escaping Quotes, 352
- Evaluate to false, 357
- Evaluate to true, 357
- Example Assignments, 65
- Example Event Handler Declaration,
 - Example Linux Paths, 8
 - Example Windows Paths, 8
- Explicit Event Handling in PHP, 50-51
- External SQL Statements, 194
- Field Pointer, 353
- File Kicker
 - Configuration, 243
- File Receiver
 - Configuration, 244
- File Update Event Handler
 - Configuration, 244
 - File Update Event Handler Output, 245
- Foo Event Configuration, 248
- Foo Event Handler Code, 248-249
- Foo Event Output, 249
- Foo.groovy HTTP GET Event Handler, 247-248
- Form-Based Authentication Configuration, 217
- Generic Database Connection Configuration, 173
- GET Method in PHP, 57
- Groovy Heredocs, 351
- Groovy View Include Syntax, 129
- Implicit Return Statement, 354
- Including a Custom Dojo Widget to the Page, 274
- `index.gt`—Quiz Application Home Page, 129-130
- Initialization Timer Configuration, 238
- Instantiating and Using a Java Class in PHP, 304
- Iterating over a List, 364
- Iterating Using the `each` Method, 364
- Ivy Configuration File, 41
- Java Absolute URI API, 253
- Java Event Handler, 52
- Java for Loop, 363
- Java Relative URI API, 252
- JSON Rendering with Groovy, 139
- JSON Rendering with PHP, 140-141
- Layout Markup, 284-285
- LDAP User Details Function, 226-227
- LDAP-Specific Configuration Settings, 225-226
- `legacy1.gt`—Get Some Data from “That” System, 135-136
- Link Class: `/app/scripts/Link.groovy`, 190

- Linux Java Version Check, 6
- Linux Zero Script Fragment
 - Demonstrating Proxy Configuration, 9
- List Concatenation, 359
- List Creation, 358
- List Element Access, 359
- Locations of Supported JDKs, 5
- Log Output of
 - /app/resources/kickReceiver.groovy, 242-243
- Log Output of Single Link Request, 189
- LogEvent.groovy Handler Output, 247
- LogEvent.groovy Handler to Handle Several Events, 246
- Logger PHP Extension Fragment, 313-314
- Logging PHP to Java Sample Program, 306
- Logging_extension.php—PHP Script Using Logger Extension, 314-315
- Login Form Page (/login.gt), 217-218
- Logout Page (public/logout.gt), 222
- The for Loop Using a Range, 363
- Looping Over a Code Block with the Times Method, 364
- Looping Using the step Method, 364
- Main Landing Page (public/index.gt), 212
- Manipulating the Type of a List, 358
- Map Concatenation, 362
- Method to Obtain Database Schema, 199
- Method to Obtain Fields for a Table, 200-201
- Method to Obtain Tables for Schema, 199
- Method to Return All Defined Databases, 198
- Method to Run Dynamic SQL, 201-203
- More Embedded Expressions, 351
- More on the zpost Method, 90
- MS SQL Database Configuration, 177
- MySQL Bookmark Connection Configuration, 187
- MySQL Database Configuration, 176
- Naive Parameter Handling, 259
- Named Parameters, 358
- Named Variables with each, 365
- Networked Derby Database Configuration, 174
- New Search Widgets, 280-281
- No-arg Method, 353
- OpenID Security Configuration, 229
- Optional Parameters with Default Values, 365
- Oracle Database Configuration, 176
- Output from the “Zero Version” Command on Windows, 10
- Person Groovy Class, 310-311
- PHP Accessing Groovy Class, 309
- PHP Create Form, 58
- PHP Create New Bookmark, 58-59
- PHP Decode Result and Display, 57
- PHP Delete and Redirect, 57-58
- PHP URIUtils API, 256
- PHP View Include Syntax, 129
- PHP ZRM Resource Handler, 50
- Placing Groovy Variables into the Global Context, 101
- Prepared Statement Using Fuzzy Search, 193
- Prepared Statement Using Named Variable, 193
- Prepared Statement Using Positional List Parameters, 193
- Prepared Statement Using Single List Argument, 193
- Prepared Statements Using Request Parameter, 192
- Protecting from
 - NullPointerExceptions in Java, 355
 - /public/index.gt, 239-238
 - /public/legacy2.gt—Nothing But Application Code, 137
 - /public/quiz/index.gt—Simple Redirect to Business Logic, 132
- Query Log Output, 188
- Quotes in Heredocs, 352
- Require Statements for Added Functionality, 280
- Resource Handler for Explicit Event Handling, 47-49
- /resource/kickReceiver.groovy, 242
- REST Resource to Obtain All Data Sources, 204
- Resulting JSON Object, 139
- Results of Dojo URLs Query, 189
- Results of Iterating the Bean List, 190
- Results of the “Zero Help” Command, 11-12
- Results of Using the Java Requested URI API, 254
- Results of “Zero Help Search” in the CLI, 12
- Retrieving a
 - FirstElementList, 104
- Retrieving a Map from the Global Context, 107

- Retrieving an Element from a FirstElementList, 105
- Retrieving an Element from a List, 103
- Retrieving an Element from a Map, 107
- Retrieving Keys and Value from a Map, 362
- Retrieving Lists from the Global Context with Groovy, 102
- Retrieving the First Element from a FirstElementList, 105
- Retrieving Variables from the Global Context with Groovy, 101
- Running Database Scripts, 206
- Sample Bookmark Data, 183
- Sample Data Manipulation Calls, 191-192
- Sample Error Handling Example, 160
- Sample Ivy File, 69-70
- Sample PHP Data Manipulation Calls, 196
- Sample PHP Query with Error Handling, 197
- Sample zero.config File, 77
- Secured Landing Page (/public/my/index.gt), 221-222
- Selecting Data into a List of Beans, 190, 195
- Setting a Method in a Variable, 353
- Simple Java Class, 349
- Simple Kicker Configuration, 241
- Single Link Request, 188
- Sorting and Reversing a List, 360
- SQL Using External SQL Statement, 194
- SSL Configuration Settings, 211
- Testing if the Global Context Contains a FirstElementList, 105
- Testing if the Global Context Contains a Map, 107
- Testing if the Global Context Contains a Particular Element from a List, 103
- Testing if the Global Context Contains a Particular List, 103
- Testing if the Global Context Contains a Particular Variable, 102
- Testing if the Global Context Contains a Value from a Map, 107
- Testing if the Global Context Contains an Element of a FirstElementList, 106
- Testing if the Global Context Contains the First Element of a FirstElementList, 105
- Timer Configuration, 236
- Two Runs of validatedb Command: One Good and One Bad, 205
- Typical sMash Query Flow, 187-188
- Using Curry, 358
- Using each WithIndex, 365
- Using SimpleXML to Process RSS Feed, 321
- Using the collect Method, 360
- Using the findAll method, 360
- Using the flatten Method, 361
- Using the Groovy Relative URI API, 255
- Using the Java Relative URI API, 252-253
- Using the Java Requested URI API, 254
- Using the join Method, 360
- Using the min, max, and sum Methods, 360
- Using the PHP URIUtils API, 256
- Using the pop Method, 359
- Using the Spread Operator, 361
- Using the Spread-Dot Operator, 361
- Using the upto Method for Looping, 363
- Using zpost to Append a FirstElementList, 94
- Using zpost to Append to a FirstElementList, 94
- Using zputs to Put Multiple Lists into the Global Context, 94-95
- Very Simple Closure, 357
- Windows Java Version Check, 6
- Windows Zero Batch File Fragment Demonstrating Proxy Configuration, 9
- XML Rendering with Groovy, 142
- XML Rendering with PHP, 142
- zcontain to Test if List Exists in the Global Center, 113
- zcontain to Text if an Element in a List Exists in the Global Context, 113
- zcontains a Map, 100
- zcontains a Map Value, 100
- The zcontains Method, 87
- The zcontains Method for a List, 92
- The zcontains Method for a List Element, 93
- zcontains of a FirstElementList, 96
- zcontains of an Element from a FirstElementList, 97
- zcontains of the First Element of a FirstElementList, 97
- zcontains Test if an Object Is in the Global Context, 109

- zcontains to Test if a Map Exists in the Global Context, 120
- zcontains to Test if an element in a Map Exists, 120
- zcontains to Test if the Global Context Contains a FirstElementList, 116
- zcontains to Test if the Global Context Contains an Element of a FirstElementList, 117
- zcontains to Test if the Global Context Contains the First Element of a FirstElementList, 117
- zdelete a List, 92
- zdelete a List Element, 92
- zdelete a Map, 99
- zdelete a Map Value, 100
- The zdelete Method, 87
- zdelete of a FirstElementList, 96
- zdelete of an Element of a FirstElementList, 96
- zdelete to Delete a FirstElementList, 116
- zdelete to Delete a List from the Global Context, 112
- zdelete to Delete an Object from the Global Context, 109
- zdelete to Delete the First Element from a FirstElementList, 116
- zdelete to Remove a Map from the Global Context, 119
- zdelete to Remove an Element from a Map in the Global Context, 119
- zdelete to Retrieve an Element from a List in the Global Context, 113
- The zdump Method, 89
- zdump to Dump All the Objects Stored in a Particular Prefix to a String, 110
- zero.config—Enable Directory Browsing, 128
- zget a Map, 99
- zget a Map Value, 99
- The zget Element, 91
- The zget Lists, 91
- The zget Method, 87
- zget of a FirstElementList, 95
- zget of an Element from a FirstElementList, 95
- zget of the First Element of a FirstElementList, 95
- zget to Get an Object from the Global Context, 108
- zget to Retrieve a FirstElementList in the Global Context, 115
- zget to Retrieve a List from the Global Context, 112
- zget to Retrieve a Map in the Global Context, 118
- zget to Retrieve an Element from a FirstElementList in the Global Context, 115
- zget to Retrieve an Element from a List in the Global Context, 112
- zget to Retrieve an Element from a Map in the Global Context, 119
- zget to Retrieve the First Element from a FirstElementList in the Global Context, 115
- The zlist Method, 88
- zlist to Get a Dump of All the Variables in the App Zone, 109
- zlist_all to Get a Deep Dump of all the Variables in the App Zone, 110
- The zlistAll Method, 88
- zpost an Append to a Map, 98
- The zpost Method, 90
- zpost to Append a List to a FirstElementList in the Global Context, 115
- zpost to Append a List to a List Stored in the Global Context, 111
- zpost to Append to a FirstElementList in the Global Context, 114
- zpost to Append to a List, 111
- zpost to Append to a Map in the Global Context, 118
- zpt to Put a List into the Global Context, 111
- The zput Method, 86, 89
- zput of a Map, 97
- zput to Create a FirstElementList in the Global Context, 114
- zput to Create a Map in the Global Context, 117
- zput to Put an Object into the Global Context, 108
- The zputs Method, 86, 91
- zputs Several Maps into the Global Context, 98
- lists
 - adding to, 359
 - concatenation, 359
 - Groovy, 358-361
 - Groovy APIs, 102
 - zcontains method, 103
 - zdelete method, 103
 - zget method, 102-103
 - zpost method, 102
 - zput method, 102
 - iterating over, 364
 - PHP APIs, 110
 - zcontains method, 113
 - zdelete method, 112-113
 - zget method, 112
 - zpost method, 111-112
 - zput method, 111
 - reversing, 360
 - sorting, 360

loadDataSources() function, 290
 loading
 data to ZRM Test page, 183-184
 initial pages, Dojo, 286-287
 schemas, Dojo, 291-293
 local resolvers, 73-74
 logger extension, PHP sample, 313-315
 logging
 with Firebug, 297
 into PHP with Apache Common, 305-307
 login, extensions, 332
 login form, 217-219
 files for applications, 221-222
 knowing users, 219
 login page, 223
 LoginServiceExtension, 332
 logout page, 224
 logs directories, 36
 looping, Groovy, 363-365

M

malicious users, 260
 managing errors, 135-138
 maps
 concatenation, 362
 Groovy, 361-362
 Groovy APIs, 106
 zcontains method, 107-108
 zdelete method, 107
 zget method, 106-107
 zpost method, 106
 zput method, 106
 Java APIs, 97
 zcontains method, 100
 zdelete method, 99-100
 zget method, 99
 zpost method, 98
 zput method, 97-98
 zputs method, 98-99
 PHP APIs, 117
 zcontains method, 119-120
 zdelete method, 119

 zget method, 118-119
 zpost method, 118
 zput method, 117-118
 max method, 360
 method pointers, Groovy, 353
 Microsoft SQL Server, 177
 min method, 360
 modules, Ivy, 69
 myfunction, 312
 MySQL, 175-176

N—O

non-persistent zones, 81
 config zone, 81-82
 connection zones, 84
 enter zones, 83
 request zones, 82-83
 tmp zones, 84
 non-repudiation, 207
 non-supplied versions of Dojo, 295-297
 objects
 Groovy APIs
 zcontains method, 101-102
 zdelete method, 101
 zget method, 101
 zput method, 101
 PHP APIs, 108
 zcontains method, 109
 zdelete method, 108-109
 zdump method, 110
 zget method, 108
 zlist method, 109-110
 zlist_all method, 110
 zput method, 108
 onError function, 292
 onLog, 246
 onRequestBegin, 246
 onRequestEnd, 246
 OpenID
 configuration, 228-230
 profile fields, 230
 OpenID Simple Registration Extension, 230

operator overloading, Groovy, 355-356
 optional parameters, Groovy, 365
 Oracle, 176
 outbound connections, securing, 230-232
 overriding configuration parameters, 79-80

P

parameters
 configuration parameters, overriding, 79-80
 Groovy, 365
 parentheses, Groovy, 353
 performance, creating custom Dojo builds, 294-295
 persistent zones, 84
 app zones, 85
 storage zones, 85
 user zones, 85
 PHP, 124, 301
 accessing
 Java classes, 304
 static Java class members, 304-305
 accessing data, 195-196
 adding, to applications, 301-302
 applications, 302-303
 arguments to Java variables, 315-316
 creating clients, 56-59
 error handling, 197
 event handling, 49-51
 extending, 311-313
 extensions, IBM WebSphere sMash, 323
 Groovy bridges, 308
 examples, 308-311
 Java bridges, 303-304
 Java to PHP variable conversion, 317
 logger extension sample, 313-315
 logging in with Apache Common, 305-307

- rendering, 124-125
 - with JSON, 140-141
- resource handlers, creating, 152-154
- running apps in WebSphere
 - sMash, 303
- sMash and, 301
- SuperGlobals, 317
 - \$_COOKIE, 320
 - \$_FILES, 319-320
 - \$_GET, 318-319
 - \$_POST, 318-319
 - \$_REQUEST, 320
 - \$_SERVER, 318
 - \$HTTP_RAW_POST_DATA, 319
- XML processing, 320-322
- PHP APIs, 108
 - FirstElementLists, 113-114
 - zcontains method, 116-117
 - zdelete method, 116
 - zget method, 115-116
 - zpost method, 114-115
 - zput method, 114
 - lists, 110
 - zcontains method, 113
 - zdelete method, 112-113
 - zget method, 112
 - zpost method, 111-112
 - zput method, 111
 - maps, 117
 - zcontains method, 119-120
 - zdelete method, 119
 - zget method, 118-119
 - zpost method, 118
 - zput method, 117-118
 - objects, 108
 - zcontains method, 109
 - zdelete method, 108-109
 - zdump method, 110
 - zget method, 108
 - zlist method, 109-110
 - zlist_all method, 110
 - zput method, 108
- URIUtils, 256
- XML rendering, 142
- positional list parameters,
 - prepared statements, 193
- \$_POST, 318-319
- Poster, last-modified
 - headers, 259
- prepared statements, 192-194
 - fuzzy search, -193
 - pureQuery, 192-194
 - single list arguments, 193
- profile fields, OpenID, 230
- project creation, Dojo, 283
- .project file, 36
- Project Zero, 2
- projects, creating in Eclipse, 23
- proxy servers, 8-9
- public directories, 35
- pureQuery, 186-188
 - connection pooling, 194-195
 - data manipulation statements, 191-192
 - prepared statements, 192-194
 - simple query methods, 188-190
 - standard JDBC database access, 197-198
 - columns for tables, 200-201
 - Database Administrator Application, 203-204
 - locating all database definitions and details, 198
 - obtaining database schema details, 198-199
 - obtaining tables from schema entries, 199
 - processing dynamic SQL, 201-203
- Reliable Transport Extension.
 - See* RTE (Reliable Transport Extension)
- remote connections, 329-330
- remote repositories, 71
- remote resolvers, 73
- Remote Zero repositories, 71-72
- render_view, 324
- renderers, custom renderers, 128
- rendering
 - data, 138
 - JSON (JavaScript Object Notation), 138-141
 - Groovy, 125-126
 - PHP, 124-125
 - views for, 128-135
 - XML, 141
 - reports directories, 36
 - \$_REQUEST, 320
 - Request Accept headers, 147
 - request zones, 82-83
 - resolution, Ivy, 71-72
 - resolvers, Ivy, 71-72
 - local resolvers, 73-74
 - remote resolvers, 73
 - resource handlers
 - Groovy, creating, 150-152
 - PHP, creating, 152-154
 - resources, bonding, 157-159
 - response codes, 145-146
 - error handling, 159-160
 - response configuration, 75-76
 - Response headers, 148
 - responses, 121
 - REST, 36-37, 143-145
 - calls, 282
 - declaring dependencies, 41-42
 - HTTP methods, 37-38
 - IBM WebSphere sMash, 149-150
 - methods, 144
 - resources for database access, 204
 - testing and documentation, 162-170

Q–R

- query fields, 197
- quotes, Groovy, 352
- ranges, Groovy, 362-363
- receivers, 243-245
- references, Dojo, 298-299

- validatedb {dbKey}, 205
 - validation
 - code validation, JSLint, 297
 - data validation, JSONLint, 298
 - validators, 257-259
 - variable substitution, application configuration, 65-66
 - variables
 - PHP arguments to Java variables, 315-316
 - prepared statements, 193
 - views for rendering, 128-135
 - virtual directories, REST, 43
- W**
- WebSphere sMash Export Wizard, 27
 - widgets, Dojo, 270, 280-281
 - custom widgets, 274
 - wizards
 - WebSphere sMash Export Wizard, 27
 - ZRM Wizard, 178
 - workspace resolver, 73
- X-Y-Z**
- XML
 - rendering, 141
 - utilities, 346-347
 - XML processing, PHP, 320-322
 - xml_decode, 347
 - xml_encode, 347
 - zcontains, 324
 - zcontains method
 - FirstElementLists, 96-97
 - Groovy APIs
 - FirstElementLists, 105-106
 - lists, 103
 - maps, 107-108
 - objects, 101-102
 - Java APIs, 87-88
 - lists, 92-93
 - maps, 100
 - PHP APIs
 - FirstElementLists, 116-117
 - lists, 113
 - maps, 119-120
 - objects, 109
 - zdelete, 325
 - FirstElementLists, 96
 - Groovy APIs
 - FirstElementLists, 105
 - lists, 103
 - maps, 107
 - objects, 101
 - Java APIs, 87
 - lists, 92
 - maps, 99-100
 - PHP APIs
 - FirstElementLists, 116
 - lists, 112-113
 - maps, 119
 - objects, 108-109
 - zdump method, 325
 - Java APIs, 89
 - PHP APIs, 110
 - .zro directory, 36
 - ZRM (Zero Resource Model), 171, 177
 - adding data to, 182-183
 - creating, 178-181
 - Dojo, 279-282
 - loading data using ZRM Test Page, 183-184
 - making data available as a service, 181-182
 - synchronizing, 43
 - zero user configuration, 220-221
 - zero_login, 332
 - zero_logout, 332
 - zero.config, 75
 - Global Context and zero.config, 63-64
 - zero.php.QueryExtension, 332
 - zero.php.URIUtilsExtension, 326
 - zero.php.XMLExtension, 346
 - zero.php.ZeroExtension, 323
 - zget method, 325
 - FirstElementLists, 95
 - Groovy APIs
 - FirstElementLists, 104-105
 - lists, 102-103
 - maps, 106-107
 - objects, 101
 - Java APIs, 86-87
 - lists, 91
 - maps, 99
 - PHP APIs
 - FirstElementLists, 115-116
 - lists, 112
 - maps, 118-119
 - objects, 108
 - zlist method, 325
 - Java APIs, 88-89
 - PHP APIs, 109-110
 - zlist_all method, 325
 - PHP APIs, objects, 110
 - zlistAll method, 88-89
 - zones, 81
 - non-persistent zones, 81
 - config zone, 81-82
 - connection zones, 84
 - enter zones, 83
 - request zones, 82-83
 - tmp zones, 84
 - persistent zones, 84
 - app zones, 85
 - storage zones, 85
 - user zones, 85
 - zpost method, 326
 - FirstElementLists, 94
 - Groovy APIs
 - FirstElementLists, 104
 - lists, 102
 - maps, 106
 - Java APIs, 90
 - maps, 98
 - PHP APIs
 - FirstElementLists, 114-115
 - lists, 111-112
 - maps, 118

- zput method, 326
 - Groovy APIs
 - FirstElementLists, 104
 - lists, 102
 - maps, 106
 - objects, 101
 - Java APIs, 86
 - lists, 89-90
 - maps, 97-98
 - PHP APIs
 - FirstElementLists, 114
 - lists, 111
 - maps, 117-118
 - objects, 108
- zputs method
 - FirstElementLists, 94-95
 - Java APIs, 86
 - lists, 90-91
 - maps, 98-99
- ZRM (Zero Resource Model),
171, 177
 - adding data to, 182-183
 - creating, 178-181
 - Dojo,
 - establishing applications, 177
 - iterative zero resource model
 - design, 184-186
 - loading data using (ZRM Test Page), 183-184
 - making data available as a service, 181-182
 - REST, 38-40
 - synchronizing, 43
- ZRM Test Page, loading data,
183-184
- ZRM Wizard, 178
- ZSO, timers, 239