

Table of Contents

Preface	1
Chapter 1: Acquiring PEAR: What is it and How do I Get It?	5
A Democratic Innovation for PHP: PEAR Channels	8
What is PEAR? A Code Repository or an Installer?	9
PEAR Package Repository and PEAR Channel	9
PEAR Installer	10
Installing the PEAR Installer	12
PEAR Bundled with PHP	12
Installation for PHP Versions Older than 5.1.0	15
Other Unofficial Sources	17
Synchronizing to a Server with no Shell Access Using PEAR_RemoteInstaller	18
Summary	22
Chapter 2: Mastering PHP Software Management with the PEAR Installer	23
Distributing Libraries and Applications	25
Differences between Libraries and Applications from the Installer's Point of View	26
Using Versioning and Dependencies to Help Track and Eliminate Bugs	27
Versioning	27
PEAR Packaging and Strict Version Validation	29
Enterprise-Level Dependency Management	32
Distribution and Upgrades for the End User	34
An Overview of package.xml Structure	36
Tags Shared between package.xml 1.0 and 2.0	44
Package Metadata	44
Package Name/Channel	45
Maintainers (Authors)	46
Package Description and Summary	47

Basic Release Metadata	48
Package Version	48
Package Stability	49
External Dependencies	51
Release Notes	58
Release License	58
Changelog	58
File List, or Contents of the Package	59
New Tags in package.xml	60
File/Directory Attributes: name, role, and baseinstalldir	63
Summary	66
Chapter 3: Leveraging Full Application Support with the PEAR Installer	69
package.xml Version 2.0: Your Sexy New Friend	69
PEAR Channels: A Revolution in PHP Installation	70
Application Support	72
Introduction to Custom File Roles	73
Creating PEAR_Installer_Role_Chiaramdb2schema Custom Role	76
Full Range of Possible Custom File Roles	81
Introduction to Custom File Tasks	83
Creating the PEAR_Task_Chiara_Managedb Custom Task	87
The Full Range of Possible Custom File Tasks	96
Post-Installation Scripts for Ultimate Customization	99
Bundling Several Packages into a Single Archive	115
Backwards Compatibility: Using package.xml 1.0 and 2.0	116
Why Support Old and Crusty package.xml 1.0?	117
Case Study: The PEAR Package	118
PEAR_PackageFileManager	119
Obtaining PEAR_PackageFileManager	119
PEAR_PackageFileManager Script and the package.xml Files it Generates	119
How PEAR_PackageFileManager Makes a Hard Life Easy	129
Globbing Files for package.xml	129
Managing Changelog	130
Synchronizing package.xml Version 1.0 and package.xml Version 2.0	131
Creating a Package for Installation with the PEAR Installer	132
Summary	133
Chapter 4: Clever Website Coordination Using the PEAR Installer	135
Overview of the Problem	135
Understanding the Problem	136
Managing Code Breakage and Reverting to Previous Versions	137
Managing Missing or Extraneous Files	138

Coordinating Development with a Team of Developers	139
Backing Up Code: Redundancy as a Necessary Precaution	140
The Solution, Part I: All-Important Source Control	140
Providing Redundancy and Revision History	141
Installing CVS or Subversion	141
Concurrent Versions System	142
Subversion	145
Intelligent Source Control	148
Maintaining Branches for Complex Versioning Support	148
Using Tags to Mark Point Releases	149
The Solution, Part II: Using the PEAR Installer to Update the Website	150
Generating package.xml from the Source Control Checkout	154
Packaging: Coordinating Release Versions with Tags and Branches	160
Testing the Release before Uploading	161
Upgrading the Live Server	164
Using the pear upgrade Command	166
The Real Beauty of Using Pear to Fix Problems	167
Summary	168
Chapter 5: Releasing to the World: PEAR Channels	169
Distributing a package.xml-Based Package	170
Distributing Packages through a Channel Server	173
The channel.xml File	174
channel.xml Tag Summary	176
Obtaining Chiara_PEAR_Server	182
Configuring the Server; Obtaining a Front End for End Users	183
Adding a Package and Releasing Packages	185
Installing a Public Channel Front End	188
Distributing Pay-For-Use PHP Applications through a Channel	191
Distributing Packages through Static tarballs for Single-Client Installations	197
Who Needs this Feature?	197
Differences in package.xml and Dependencies	197
Releasing Equals Uploading	199
Security Issues Inherent in Remote Installation	200
How do PEAR Installer and Chiara_PEAR_Server Provide Security?	201
Extra Security beyond what PEAR Provides	201
Specific Security Principles Applied in Designing the PEAR Installer and Chiara_PEAR_Server	202
Summary	204

Chapter 6: Embedding the PEAR Installer: Designing a Custom Plug-In System	205
Why Embed PEAR?	207
Simplify User Choices for Installation	208
Eliminate Chances for Error	208
Other Plug-In Systems	208
Bundling Plug-Ins Directly in the Source Code	209
Subpackages – PEAR Dependencies	209
Case Study: MDB2	210
Custom Plug-In Systems: Remote Server	214
Case Study: Serendipity Blog's Spartacus Plug-In Manager	215
Case Study: Seagull Framework's Embedded PEAR Installer	219
Designing a Custom PEAR Channel-Based Plug-In System	224
Reusing Existing Functionality	225
PEAR Installer Infrastructure: REST and PEAR Installer Classes	226
Extending REST with Custom Information	238
Designing a Lightweight Installer Plug-In: The Code At Last	239
MyBlog_Template_IConfig and MyBlog_Template_Config	240
MyBlog_Template_REST	241
MyBlog_Template_Lister	248
MyBlog_Template_Fetcher	253
The MyBlog Post-Install Script	258
The Rest of the Fake MyBlog Package	264
Improvements for the Ambitious	272
Summary	273
Index	275
