The Biml Book

Business Intelligence and Data Warehouse Automation

Andy Leonard
Scott Currie
Jacob Alley
Martin Andersson
Peter Avenant
Bill Fellows
Simon Peck
Reeves Smith
Raymond Sondak
Benjamin Weissman
Cathrine Wilhelmsen

Apress®

The Biml Book

Business Intelligence and Data Warehouse Automation

Andy Leonard
Scott Currie
Jacob Alley
Martin Andersson
Peter Avenant
Bill Fellows
Simon Peck
Reeves Smith
Raymond Sondak
Benjamin Weissman
Cathrine Wilhelmsen

Apress®

The Biml Book

Andy Leonard
Farmville, Virginia, USA

Scott Currie
Greer, South Carolina, USA

Jacob Alley
Simpsonville, South Carolina, USA

Martin Andersson
Billeberga, Sweden

Peter Avenant
Erina, New South Wales, Australia

Bill Fellows
Kansas City, Missouri, USA

Simon Peck
Christchurch, New Zealand

Reeves Smith
HIGHLANDS RANCH, Colorado, USA

Raymond Sondak
Voorburg, The Netherlands

Benjamin Weissman
Nuernberg, Germany

Cathrine Wilhelmsen
Hagan, Norway

ISBN-13 (pbk): 978-1-4842-3134-0 ISBN-13 (electronic): 978-1-4842-3135-7
https://doi.org/10.1007/978-1-4842-3135-7

Library of Congress Control Number: 2017958634

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Jonathan Gennick
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano
Copy Editor: Mary Behr
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484231340. For more detailed information, please visit www.apress.com/source-code.

Printed on acid-free paper

For Christy.

—Andy Leonard

To those who possess "all the virtues of Man without his Vices"
and to one without equal who shared our journey for 16 years. Mac,
you will be missed.

—Scott Currie

Contents at a Glance

About the Authors xvii

Acknowledgments xix

Foreword xxi

Introduction xxvii

■ Part I: Learning Biml 1

■ Chapter 1: Biml Tools 3

■ Chapter 2: Introduction to the Biml Language 27

■ Chapter 3: Basic Staging Operations 57

■ Chapter 4: Importing Metadata 93

■ Chapter 5: Reusing Code, Helper Classes, and Methods 119

■ Part II: Biml Frameworks 139

■ Chapter 6: A Custom Biml Framework 141

■ Chapter 7: Using Biml as an SSIS Design Patterns Engine 199

■ Chapter 8: Integration with a Custom SSIS Execution Framework 221

■ Chapter 9: Metadata Automation 239

■ Chapter 10: Advanced Biml Frameworks and BimlFlex 265

Part III: Biml Topics 275

Chapter 11: Biml and Analysis Services 277

Chapter 12: Biml for T-SQL and Other Little Helpers 313

Chapter 13: Documenting Your Biml Solution 331

Chapter 14: Troubleshooting Metadata 353

Chapter 15: Troubleshooting Biml 373

Part IV: Appendices 389

Appendix A: Source Control 391

Appendix B: Parallel Load Patterns in Biml 413

Appendix C: Metadata Persistence 419

Index 479

Contents

About the Authors xvii

Acknowledgments xix

Foreword xxi

Introduction xxvii

Part I: Learning Biml 1

Chapter 1: Biml Tools 3

BimlExpress 3

BimlExpress Installation 3

BimlOnline 9

BimlStudio 14

Logical vs. Project View 17

Biml Editor 20

Biml Designer 20

Visual Designers 21

Relational 22

Integration Services 22

Analysis Services 22

Metadata 22

Library 23

Documentation 23

Feature Comparison 23

Summary 26

Chapter 2: Introduction to the Biml Language 27

Motivation 28

Biml Language Structure 29

XML Introduction 30

Biml 31

Root Elements 31

Text Blocks 32

Comments 33

A First Package 33

Referencing Objects by Name 35

References with Scoped Name 37

More on the Biml Language 38

Biml Script 39

Code Nuggets 39

Directives 47

Summary 56

Chapter 3: Basic Staging Operations 57

Basic Staging Load Pattern 57

Creating Basic Staging Operations Using BimlStudio 59

Creating Relational Hierarchy Artifacts 59

Creating SSIS Packages 67

Creating Basic Staging Operations Using BimlExpress 82

Creating Relational Hierarchy Artifacts 82

Summary 92

Chapter 4: Importing Metadata 93

SQL Server 94

ImportTableNodes and ImportDB 94

GetDatabaseSchema 103

Which Method Should I Use? 114

Excel 115
Flat Files 116
Basic Biml Elements for Flat Files 116
Summary 118
Chapter 5: Reusing Code, Helper Classes, and Methods 119
Include Files 119
Using Include Files to Control Logging 121
CallBimlScript 123
Using CallBimlScript to Control Logging 125
CallBimlScriptWithOutput 127
Using CallBimlScriptWithOutput 127
Helper Classes and Methods 130
Helper Method: Check If Annotation Exists 130
Extension Methods 134
Using the VB Extension Method 135
Summary 137
Part II: Biml Frameworks 139
Chapter 6: A Custom Biml Framework 141
What Is a Framework? 141
Why Use Metadata? 142
The Biml Relational Hierarchy 142
Storing the Relational Biml Hierarchy Metadata in a Database 145
Creating a Database and Schema for Custom Biml Relational Hierarchy Metadata 146
Creating a Table for Custom Biml Relational Hierarchy Connections Metadata 146
Creating a Table for Custom Biml Relational Hierarchy Database Metadata 147

Creating a Table for Custom Biml Relational Hierarchy Schema Metadata 148
Creating a Table for Custom Biml Relational Hierarchy Tables Metadata 149
Creating a Table for Custom Biml Relational Hierarchy Columns Metadata 150
Creating a Table for Mappings Metadata 151
Using the Relational Database Metadata to Build the Biml Relational Hierarchy 153
Building Biml Connection Tags from Metadata 155
Building Biml Database Tags from Metadata 156
Building Biml Schema Tags from Metadata 157
Building Biml Table Tags from Metadata 160
Building Biml Column Tags from Metadata 163
Using the Relational Database Metadata to Build Packages 170
Building an SSIS Package 170
Adding the Biml Package Tag 171
Moving the Project to BimlStudio 175
Adding the Tasks and Dataflow Tags 177
Adding the Transformations Tag and Your First Dataflow Component 179
Adding a Lookup Transformation 181
Adding a Derived Column Transformation 184
Adding an Ole DB Destination Adapter 188
Testing the CreatePackages.biml File 191
Load Currency Metadata 193
Just Add Metadata… 194
Summary 198
Chapter 7: Using Biml as an SSIS Design Patterns Engine 199
Identifying the Design Pattern 199
Decoupling the Design Pattern 200
Adding the Design Pattern Name to the SSIS Package Name 204
Adding the Design Pattern Name to the Metadata Database 205

Using the Design Pattern Tables 206
Testing Metadata-Driven Design Pattern Selection 212
Adding the IncrementalLoad Pattern 213
Adding Pattern BimlScript to CreatePackages.biml 216
Updating the Metadata 217
Building the Packages 219
Summary 220

Chapter 8: Integration with a Custom SSIS Execution Framework 221

An SSIS Execution Framework 221
Persisting Applications Metadata 222
Persisting Packages Metadata 222
Persisting Application Packages Metadata 223
Preparing a Demo SSIS Project 224
A Biml File to Generate SSIS Framework Metadata 225
Adding SSIS Framework Application Metadata 228
Adding SSIS Package Metadata 230
Adding Framework Application Package Metadata 233
Testing the SSIS Project and SSIS Framework Application 236
Summary 238

Chapter 9: Metadata Automation 239

What Is Metadata? 239
The Evolution of Metadata 240
Natural Metadata 241
Supplemental or Hybrid Metadata 241
Why Hybrid Approaches Eventually Break Down 244
Synthetic Metadata 245

Types of Metadata 246
Business Metadata 246
Technical Metadata 246
Operational Metadata 246
Why Metadata Models? 247
Biml Metadata Models and Instances 248
Models 248
Instances 249
Why You Need Them Both 251
Manually Creating Metadata in BimlStudio 251
Models 252
Instances 252
Metadata Model Wrapper Objects 253
Accessing a Model Using VB.NET 254
Accessing a Model Using C# 255
Applying Metadata Models to Your Framework 256
Offline Schemas 261
Generating an Offline Schema File 262
Conclusion 263
Chapter 10: Advanced Biml Frameworks and BimlFlex 265
Bundles 265
Overview 265
Bundle Manifest 266
Extension Points 267
Building the Bundle 271
BimlFlex 272
Capabilities 272
Metadata Entry 273
Summary 274

Part III: Biml Topics 275

Chapter 11: Biml and Analysis Services..... 277

Analysis Services Project Recommendations 278

Multidimensional Biml Projects..... 279

Tabular Biml Projects 292

Using Biml to Process Multidimensional Objects in SSIS..... 306

Biml Syntax for SSIS Processing 306

Automated Approach 308

Summary..... 311

Chapter 12: Biml for T-SQL and Other Little Helpers 313

Why Biml? 313

How Does Biml Work For T-SQL?..... 313

Using the Preview Pane..... 314

Generating SSIS Packages 315

Saving T-SQL to Files..... 317

Executing T-SQL Using SqlCommand 320

Samples 322

Truncating and Dropping Tables 322

Building a Clustered Columnstore Index..... 323

Development Reset..... 324

Comparing Source and Target Tables 324

Detecting Stale Data..... 325

Checking for Duplicate Indexes 327

Checking for Broken Views..... 328

Generating Sample Data..... 328

Summary..... 330

Chapter 13: Documenting Your Biml Solution 331

Problems with Traditional Approaches 331

Requirements and Specifications: A Red Herring 331

Building Documentation Is Hard 332

Maintaining Documentation Is Hard 333

Documentation with Biml 334

Where to Store Documentation Metadata 335

Enforcing Documentation Standards 336

Autogenerating Documentation from Biml 338

Getting the Flat XML for a Biml Project 338

XSLT 338

.NET XML APIs 342

BimlScripts 344

GetJson 346

Built-in Documentation Function in BimlStudio 347

Summary 351

Chapter 14: Troubleshooting Metadata 353

General Metadata Availability 353

Collecting Database Metadata Using SQL Server 358

Using INFORMATION_SCHEMA 358

Using dm_exec_describe_first_result_set 360

Collecting Database Metadata Using the .NET Framework 362

Using SqlConnection.GetSchema 362

Using SqlDataReader.GetSchemaTable 363

Collecting Flat File Metadata 365

Using a BCP-Generated XML Format Definition 365

Programmatically Identifying Flat File Structures 371

Summary 371

Chapter 15: Troubleshooting Biml 373

Logging 373

Biml Engine Logging 373

Writing a Custom Logger 379

Diagnostics 380

Creating Validation Items 380

Manipulating the Validation List 383

Logging and Diagnostics Together 384

Troubleshooting 385

Preview Pane 385

Debug Utilities 386

Code Files and Visual Studio 387

Summary 387

Part IV: Appendices 389

Appendix A: Source Control 391

Common Terminology 392

Can I Use Source Control with Biml? 392

Using Source Control for Biml with BimlExpress 392

Which Source Control Clients Are Compatible with BimlStudio? 393

Setting up a Source Control Repository 393

Team Foundation Version Control 393

Subversion 395

Git 397

Tracking a BimlStudio Project in Source Control 398

Git and Subversion 398

Team Foundation Version Control 402

Source Control Features in BimlStudio 405
Project View Status 405
Project View Context Menu 406
File Comparison 407
Sync Tab 409
History Tab 411
Conclusion 412
Appendix B: Parallel Load Patterns in Biml 413
Parallelizing Dataflows 413
Using Balanced Data Distributor (BDD) 414
Using a Row Counter and Conditional Split 416
Conclusion 418
Appendix C: Metadata Persistence 419
Persisting Biml Metadata in a Database 419
Creating Metadata Instance Tables 419
Designing a Custom Schema for Biml Metadata Instances 420
Designing a Generic Schema for Biml Metadata Instances 432
Retrieving Biml Metadata from a Database 455
Retrieving a Biml Metadata Instance from a Custom Schema 455
Retrieving a Biml Metadata Instance from a Generic Schema 464
Conclusion 477
Index 479

About the Authors

Andy Leonard is the founder and Chief Data Engineer at Enterprise Data & Analytics, an SSIS trainer, consultant, and developer; a Biml developer and BimlHero; SQL Server database and data warehouse developer, community mentor, engineer, and farmer. He is a co-author of *SQL Server Integration Services Design Patterns* and author of *Managing Geeks - A Journey of Leading by Doing*.

Scott Currie is the founder and CEO of Varigence, Inc. Scott has led the development at Varigence of Biml and the BimlStudio IDE. Before founding Varigence, Scott worked at Microsoft for seven years on the .NET Framework, Visual Studio, the C++ compiler, various customer connection initiatives, and internal BI/DW projects.

Jacob Alley is a software developer for Varigence, Inc., in Greenville, SC. He is a graduate of Florida State University and is currently enrolled in the Master's Computer Science program at Georgia Tech.

Martin Andersson lives in Landskrona, Sweden, and works as a consultant for SolidQ. He specializes in development and architecture design for the Microsoft SQL Server platform. As a former network technician, Martin took the leap over to databases in 2010 and continues to work with SQL Server to this day. Martin holds certifications for MSCE Business Intelligence and Data Vault Modeling. When he is not digging through ETL flows he can be found in front of a Kanban board planning his next project.

Peter Avenant is the Director of Technology and founder of Varigence Australia. He is a developer, trainer, and technical data architect with over 20 years' experience focusing on data warehousing. He was an early adopter of Biml and has given numerous community talks and training sessions on Biml worldwide. He is a certified data vault data modeler and creator of the data vault accelerator implemented within BimlFlex. He is the technical lead on the BimlFlex project, working closely with the BimlStudio development team to create a flexible, extensible, and completely customizable data warehouse framework.

Bill Fellows is the owner and principal architect at Sterling Data Consulting. He is a SQL Server MVP and has been a database developer for the past 16 years, with a focus on automation and ETL. He is the organizer of Kansas City's seven SQL Saturdays and maintains the SSIS tag on StackOverflow. Bill blogs at `ssis.science` and tweets under `@billinkc`.

Simon Peck is MCSE in SQL Server BI and has been working with SQL Server for 20 years. In March of 2016 he became the first person in the APAC region to become a Varigence-certified Biml expert (BimlHero). He has been fortunate enough to have worked for close to 50 organizations worldwide, including some of the leading Microsoft BI consultancies in Europe. He is also a co-leader of the Christchurch SQL User Group in New Zealand. In 2014, after nearly 20 years away, he returned home to New Zealand with his family and is keen to dust off his skis and motorbikes (and do some BI/DWH stuff).

Reeves Smith is an independent consultant and trainer with over 20 years of experience working with SQL Server on various development and data warehouse projects. He is a Microsoft Certified Master of SQL Server and SQL Server MVP. He holds a BS in Applied Mathematics from the University of Colorado and delivers technical presentations at international, regional, and local conferences and user groups. You can visit his blog at `reevessmith.wordpress.com` or follow him on Twitter at `@SQLReeves`.

Raymond Sondak is a technical architect, analytics consultant, and owner of Analyticsaholic. He is also the first BimlHero Certified Expert in the Netherlands. Raymond is an expert in agile end-to-end business intelligence and analytics implementation as well as automation focused on Microsoft technologies. His technical expertise extends from the Microsoft business intelligence and data platform to building cloud solutions with Microsoft Azure. He also delivers training and workshops on various subjects including Biml and is very passionate about sharing his knowledge. Raymond loves to enjoy life. When he is not behind his laptop you'll find him on adventure trips around the world with his wife; they're both foodies at heart.

Ben Weissman is the owner and founder of Solisyon, a consulting firm based in Germany and focused on business intelligence, business analytics, and data warehousing. He is the first German BimlHero and has been working with SQL Server since SQL Server 6.5. In his fight against stupid, repetitive tasks, he has developed numerous automation tools over the years to make his customers' lives easier. If he's not currently working with SQL Server, he is probably travelling to explore the world. Ben is also an MCSE, Charter Member of the Microsoft Professional Program for Big Data and Data Science, and a Certified Data Vault Data Modeler. You can find him online at `http://biml-blog.de/` or @bweissman on Twitter.

Cathrine Wilhelmsen loves teaching and sharing knowledge. She works as a consultant, developer, and trainer, focusing on data warehouse and business intelligence projects. Her core skills are ETL, SSIS, Biml, and T-SQL development, but she enjoys everything from programming to data visualization. Outside of work, she's active in the SQL Server and PASS communities as a Microsoft Data Platform MVP, BimlHero Certified Expert, speaker, blogger, organizer, and chronic volunteer.

Acknowledgments

Andy Leonard: I thank God first for He leads me in right paths for his name's sake (Psalm 23:3). I thank Christy, my lovely bride, and our children, Stevie Ray, Emma, and Riley, for sacrificing some Dad time. Thanks to the awesome team at Enterprise Data & Analytics for their patience and hard work while I wrote: Bill Anton, Kent Bradshaw, Nick Harris, KeShawnda Lawrence, and Penny Trupe. We have an awesome team at Apress: Jill Balzano kept the wheels on the bus going 'round and 'round, and Jonathan Gennick is the best editor in the business. To my co-authors, Scott, Jacob, Martin, Peter, Bill, Simon, Reeves, Raymond, Ben, and Cathrine: thank you for putting up with me during the writing of this book. It has been an honor to write with you. Finally, I thank Scott for inventing Biml, allowing me the honor and privilege of being a BimlHero, patiently coaching on many topics (some technical), and being my friend.

Scott Currie: Thank you to everyone in the Biml community who has shared their insight and expertise with those who are new to data automation. Hopefully this book will ease your burden as Biml usage continues to grow. My eternal gratitude to my wife, Lydia, for her patience and grace this past year and my children, John, James, Sarah, Riley, and Helen, for granting their father the extra time at work to write this book. Dulcius ex asperis.

Jacob Alley: I would like to thank Amanda Tuttle for her unrelenting patience and love, and I would like to thank Scott Currie for affording me the opportunity to be a part of the Biml family.

Martin Andersson: For my best friend Tim, forever comrades in arms. For my father, for showing me that anything is possible. For Kristian, the struggle continues. Love and respect to Julia, Elisabeth, and Rosa for the support. Last but not least, thank you to Varigence for creating Biml.

Bill Fellows: I'd like to thank my co-authors for the opportunity to work on this project, Scott Currie for making development fun again with Biml, and Dr. James Bogan for igniting those sparks of fire so many years ago. Hilary, James, and Max: you are the reasons I get up in the morning and I love you all dearly.

Simon Peck: Marie, Oscar, Lenny, and Grace Peck for their support, love, and encouragement.

Reeves Smith: First and foremost, I would like to thank my wife, Amy, for supporting me through this effort. She took the kids on extra adventures alone to give me time to continue writing. She was selfless in all of this and I truly appreciate all the hard work it takes to raise three girls. Secondly, I want to thank my three girls, Marin, Gracynn, and Brynn, who missed out on the time with Dad. We will have to make it up. Finally, I want to thank Andy and the rest of the folks contributing to the book who gave additional help and motivation. (Yes, Ben, we are calling all of the additional communication "motivation.")

Raymond Sondak: I would like to thank my wife, Stacey, so I can do what I do, and my late sister, Sandra, with whom I share the credit on every goal I achieve. To Scott Currie, you're the man! For Andy's leadership and "The Biml Book" authors whose names are on the cover, you guys and girl rock!

Ben Weissman: Thank you to Scott for creating Biml. Thank you to Andy for including me on this project. Thank you to my awesome colleagues for delivering exceptional quality in our projects every day. And most importantly, thank you to Franzie and my family - for everything.

Cathrine Wilhelmsen: To my co-authors and fellow BimlHeroes, from the bottom of my heart, *thank you*. It has been an honor to write this book with you. Thank you for your hard work, the late nights writing and editing, and all the laughs we shared. A special thank you to Andy, Ben, and Martin; your support meant everything to me. (And to my dearest family, Cathe, Kjell, Jeanette, and Malin, I promise I won't be rambling about Biml during family dinners… for a while:) Glad i dere!)

Foreword

You have a long journey ahead of you to learn how to use Biml in your data integration projects. Before getting started, let's take a look at the upcoming chapters and what you will learn in each.

Who Is This Book For?

Anyone who is interested in Business Intelligence Markup Language (Biml), whether entirely new to the technology or a seasoned veteran, will likely find value and insight within these pages. There are, however, two audiences that will particularly benefit from this book.

First, those who are entirely new to Biml will learn everything from the very basics of writing Biml code to creating frameworks to more advanced Biml usage scenarios.

The second audience is those who have been coding with Biml for a while and are looking to take their skills to the next level through formal framework development, metadata development, or other more advanced topics. These readers might choose to skip or just skim some of the earlier chapters, but there is still much to learn from the later chapters, even for someone who has done a substantial amount of work with Biml already.

For both audiences, it is strongly recommended that you have a solid understanding of SQL Server and any SQL Server services, such as SSIS or SSAS, that you intend to use with Biml.

You should also be sure to install BIDS or SSDT for SQL Server 2005 or later.

How Is This Book Structured?

In the following sections, we will explore the chapter layout of this book. At a high level, there are four parts that guide you through the process of learning Biml, using Biml to build a framework, enhancing your Biml projects with a variety of additional capabilities, and providing a reference for additional topics of interest.

Part I: Learning Biml

In the first part of this book, we will focus on getting you up to speed on writing Biml solutions, assuming very little previous background with Biml or .NET coding. While these chapters are targeted primarily at those who are new to Biml, it is likely that even a

seasoned Biml developer will pick up a few tips, tricks, and insights along the way. Note that we do not endeavor to create the most comprehensive primer for every aspect of Biml and BimlScript (which is the way to automate Biml). We will cover all of the basics, but if you feel that you need more details on XML syntax, C# syntax, VB syntax, or any other aspect of what we cover, there are excellent resources available online, both from the `http://bimlscript.com` community site, the `http://varigence.com` support and documentation site, and from third parties.

Chapter 1: Biml Tools

Before you can get started writing code, you must have your development environment set up with tools that enable you to author and build Biml code. There are a few great options–free and paid–to choose from. In this chapter, we'll give you a brief overview of each tool and provide some guidelines on how to choose among them.

Chapter 2: Introduction to the Biml Language

Once you have your tools set up, it's time to start writing some code. In the Introduction, you'll see some Biml code and we hope it gets you excited to get started writing your own code. To get you up to speed, here we're going to cover how all of the most critical language elements–the Biml language, BimlScript code nuggets, directives, and supplemental files–work together to build data solutions.

Chapter 3: Basic Staging Operations

Once you have a firm grasp on syntax, you're going to build your first Biml solution to stage data from a source system into a target system. This first real solution will take you step-by-step through the coding, building, and execution workflows. To ensure you focus on the mechanics of creating your first solution from scratch, it will not use any BimlScript code nuggets or .NET coding.

Chapter 4: Importing Metadata

Writing flat Biml by hand can get repetitive, so in this chapter you enhance your staging solution from Chapter 3 to use BimlScript code nuggets to automatically generate the staging targets and packages for all of the tables in your source system. As part of this process, we introduce some of the most important Biml utility methods for retrieving database schema metadata from relational databases.

Chapter 5: Reusing Code, Helper Classes, and Methods

At this point, you'll have written a decent amount of .NET code, and you're going to write even more in the next part of the book as you explore Biml frameworks. Consequently, in this chapter, you'll take a look at the various methods Biml offers to organize your code to enable easier authoring and maintenance.

Part II: Biml Frameworks

In the second part of this book, you will use everything you learned in the previous chapters to build a Biml framework. The idea behind a framework is to build a metadata store that holds all of your configuration information in addition to BimlScripts that will consume that metadata to create your solution. Once you have completed your framework, you will spend most of your data development time modeling, mapping, and writing business logic, because all of the plumbing that once consumed your workday is now handled automatically by the framework.

Chapter 6: A Custom Biml Framework

In this chapter, you will build a simple metadata database and custom BimlScripts to autogenerate a customizable data load solution. Once this sample is complete, you will be able to create entirely new data load solutions for arbitrarily large databases simply by modifying your metadata and rebuilding the framework code.

Chapter 7: Using Biml as an SSIS Design Patterns Engine

In the previous chapter, the custom Biml framework used a hardcoded data load pattern for the sake of simplicity. In this chapter, you will refactor the framework code to use an architecture where you can dynamically change the data load pattern on a table-by-table basis. This will enable you to develop a library of as many patterns as you need to support business requirements while preserving the maintainability of the code base.

Chapter 8: Integration with a Custom SSIS Execution Framework

Chapters 6 and 7 focused on the creation of build-time frameworks. These frameworks are responsible for building consistent data load packages. When you want to execute those packages, you also need an execution framework to manage the order in which packages execute and a variety of other runtime behaviors. In this chapter, we will present such an execution framework and show how to integrate it into the solution.

Chapter 9: Metadata Automation

So far, our approach for metadata management was to store it in database tables and retrieve it via direct SQL queries. This is a fine solution, but it experiences some management issues as the metadata store grows to include additional configuration. In this chapter, we will present a solution for modeling your metadata in a more reusable way, one that offers a variety of value-added services on top of your metadata store.

Chapter 10: Advanced Biml Frameworks and BimlFlex

Until now, in this part of the book, you built a functional Biml framework for your data solutions. Packaging that framework for deployment, providing extensibility points for third-party users, and protecting your code from unauthorized users still presents challenges. In this chapter, we show you how to solve these problems using the Biml bundles feature.

Additionally, we provide a brief overview of the BimlFlex bundle built by Varigence as a commercial product. This will give you a very clear picture of just how extensive your Biml framework capabilities can be with sufficient development investment. It will also give you a real-life example of how to build an out-of-the-box product using Biml.

Part III: Biml Topics

In the third part of the book, you explore a variety of Biml topics of general use to you throughout your Biml development activities, whether you are building a framework or developing an ad hoc solution.

Chapter 11: Biml and Analysis Services

Most of the code you've seen in this book and likely through other sources online focuses on using Biml to develop SQL scripts and SSIS packages. In this chapter, we show how to use Biml to automate SQL Server Analysis Services (SSAS) multidimensional cube and tabular model development. Additionally, we demonstrate how Biml can be used to reduce the overhead of SSAS multidimensional cube processing management by creating automated SSIS packages.

Chapter 12: Biml for T-SQL and Other Little Helpers

Many people assume Biml automation is only useful for building the core data transformation processes that you run on a recurring basis. While Biml can do this, it is also a great option for autogenerating single-use SQL scripts that would be tedious or difficult to code by hand. In this chapter, we show you various options for creating such scripts and demonstrate via several real-world examples.

Chapter 13: Documenting Your Biml Solution

An important, though often overlooked, part of any data solution is user-friendly documentation. In this chapter, we present several strategies to automatically generate high-quality documentation from your Biml code.

Chapter 14: Troubleshooting Metadata

Throughout the book, we have shown examples of using built-in Biml utility and library methods to solve a wide variety of common data automation tasks. In some cases, you will find that one of the provided utility methods does not quite meet your needs. The

wonderful thing about Biml and BimlScript is that you have the option to return to first principles and build exactly what you need for the specific task at hand. In this chapter, we show an example of this principle in action. We directly read information schema tables in source databases to address project-specific needs around database schema discovery that are not supported by the built-in methods.

Chapter 15: Troubleshooting Biml

The samples shown in books are developed with the benefits of expert authors, plenty of time, and a technical review process to ensure the highest quality. When you first begin to write your own Biml code from scratch, you likely won't enjoy any of those benefits. When something goes wrong with your code, how should you go about finding and fixing the issue? In this chapter, we will show you several options for logging, diagnostics, and troubleshooting that should significantly improve your ability to hunt down and squash bugs in your code.

Part IV: Appendices

Even after reading this entire book, you're still just scratching the surface of what Biml can do to automate your data solutions and the flexibility Biml provides to author new approaches to common data integration problems. In the following appendices, we examine a few topics that expand on some ideas we developed earlier in the book. Our goal is to kickstart the next phase of your journey to mastering Biml.

Appendix A: Source Control

When using a human readable and writable coding language such as Biml, source control becomes a powerful tool for managing your project across multiple release cycles. In this chapter, you learn about some of the benefits of using source control and how to configure source control for BimlStudio.

Appendix B: Parallel Load Patterns in Biml

In Chapter 7, you learned how to create a modular system for choosing design patterns on a table-by-table basis. Since our focus was primarily on the modular pattern architecture, we only implemented two data load patterns. In this appendix, we give you samples of additional patterns to parallelize data flow and leverage the Change Data Capture (CDC) feature in SQL Server and SSIS.

Appendix C: Metadata Persistence

In Chapter 9, you learned how to build a Biml metadata model for your framework and how to interact with that metadata model. In this appendix, we expand that solution to include a completely generic mechanism for storing and retrieving any Biml metadata

model to and from a database. This is a solution that will be useful for those who extensively use metadata models or develop multiple complementary Biml frameworks.

Conclusion

As you can see, you have a long journey filled with learning and new insights ahead of you. Let's get started!

Introduction

You will be more productive by using Biml for stupid data transfers… and more. This book will give you an idea of what Biml can do and how it is structured. Don't worry if some of the code samples look confusing or you don't understand it all at once. Instead of having you read multiple chapters just to get your first glimpse at Biml code, you will dive right in. But after that, you will take a step back and be guided through the process from the very basics to building the most advanced frameworks!

For those of you who continuously work with the SQL Server Integration Services (SSIS) to get data from A to B (no matter if it is to build a data warehouse or to achieve some other data integration goals), you will probably have figured that, despite the magic you do for your internal and external customers every day, a lot of what you do is just repeating yourself: create a staging table in your staging database based on the source table's structure and data types, and then build a simple package to load the data using SSIS. Many tables do not actually need your magic; they are simply data copy tasks. No more, no less.

Copying a table to staging is easy. The actual setup probably only takes 5 minutes, but doing so doesn't scale very well (from a human interaction point of view). To get the job done for two tables takes twice as long. For three tables, it takes three times as long, and so on.

Moreover, if the source schema changes or someone decides to convert the whole source database from non-Unicode to Unicode (and if that hasn't happened to you yet, it probably will at some point), you start touching and modifying all those single tasks one by one. It might only take 5 minutes per table to update your packages for these minor changes, but again, it does not scale. Even more important, it's tedious, boring, and the result is the same as before. Who wants to spend all that time on something that no one will care about (as long as it works)?

Having experienced that lack of appreciation and boredom, I was looking for a way to complete these repetitive and non-challenging tasks more quickly and easily, without cloning myself.

Let's take a minute to think about SSIS packages. In the end, they are eXtensible Markup Language (XML) files, which might contain a lot of XML as the packages become bigger and bigger. A .dtsx file that simply truncates a target table and then performs a dataflow task ends up around 400 lines of XML. Writing my own XML was no option; that became clear to me relatively quick. So effectively, I was looking for a tool that would write those `.dtsx` files for me.

Biml offered the answer to that question. In the end, Biml is also an XML format, so to avoid writing `.dtsx` (SSIS package files) XML, we are just writing a different dialect of XML. This doesn't seem to make too much sense at first sight, but if you look under the covers, it does! Why? Because Biml code is much easier to maintain due to almost no overhead. Biml is a markup language, designed by Varigence with the intention to automate development of Microsoft business intelligence (BI) objects like SSIS packages. Always keep that in mind: it is not an out-of-the box solution but rather a highly flexible language to automate. If you don't know how to solve a problem manually, Biml will not help you. But if you have a design pattern that you need to build multiple times, Biml will do just that!

Biml metadata for SSIS follows pretty much exactly the same patterns you already know from manual SSIS packages. A simple illustration of the types of database and SSIS assets you can create with Biml is shown in Figure F-1.

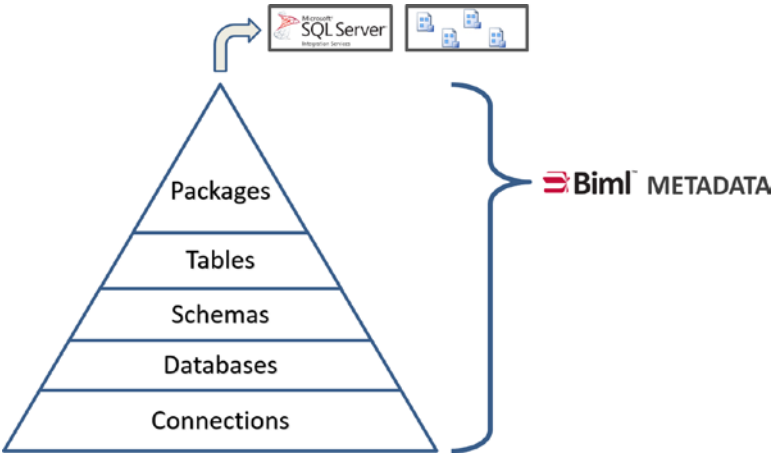

Figure F-1. *The Biml metadata structure*

A Biml file to generate that 400-line `.dtsx` file has less than 30 lines of code. It is easily readable on the screen, and it is easily manageable with your favorite text editor. So even basic functions like copying and pasting the code for those many dataflow tasks can result in saving a notable amount of time. However, that is just a small fraction of what you can achieve using Biml.

As coders, we want to avoid repeating ourselves, so we can make use of scripts (*code nuggets* containing variables, parameters, and loops) instead of having to copy and paste code repeatedly. This obviously also allows us to maintain one central codebase for annotations, transformations, etc.

In addition, one especially annoying, as in non-value adding, step is to actually create and maintain your destination tables based on the *metadata* in the underlying sources. Biml will take care of that for you, too!

Technically speaking, that means you build your Biml code, run it through the Biml compiler, and receive regular Microsoft BI objects or files, as shown in Figure F-2.

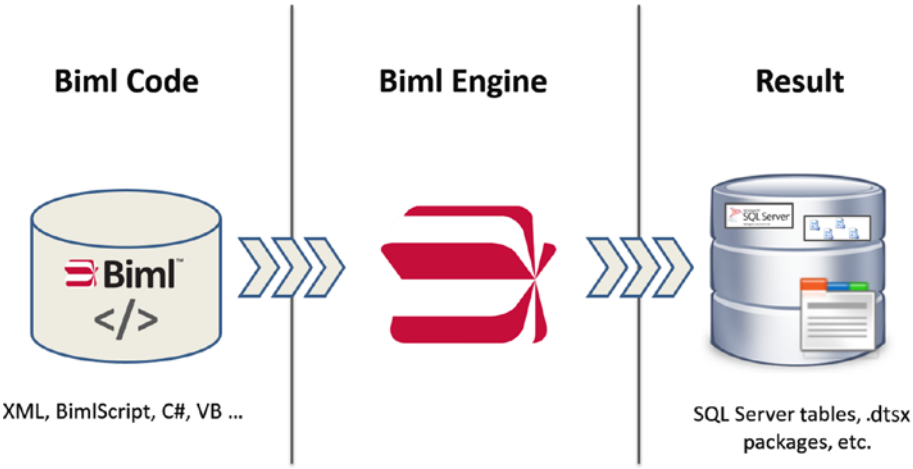

Figure F-2. *The Biml compiler process*

Given the huge amount of resources available, including the community of very active Biml users as well as *BimlHero certified experts*[1] (many of whom were involved in this book), Biml is very easy to adopt, assuming you already consider yourself a good Microsoft BI and solid .NET developer.

So what's the catch? Well, there really is none besides the fact that, as always, there are multiple ways of achieving the same thing. You will have to figure out which patterns, practices, and approaches are right for solving your data development problems. Other than that, you will probably figure that some very complex, unique, or business-logic-heavy patterns are not actually a time saver in Biml, because Biml's strength is really the automation of repeatable tasks. It simply scales very well and therefore obviously adds most value in scenarios that require such scaling, compared to one-time tasks.

Getting back to the staging database scenario, let's identify, automate, and manage a handful of components:

One or more source databases: This is probably out of your control so let's will treat it as a fixed variable.

Staging database: You want Biml to manage and create the tables in your staging target automatically.

Metadata layer: Depending on personal taste, this can be as simple as extended properties in your source database, an Excel file, a master data services solution (talking about oversizing things) or depending on the complexity, it might even be integrated in BimlStudio making use of Biml's metadata features. See more on this in Chapter 9.

[1]Become a Biml Hero Certified Expert!, https://varigence.com/Blog/Post/65

SSIS solution to populate the staging database: Again, you want Biml to take care of that for you. The goal is that there is no human interaction required, except for maintaining the control tables and generating the SSIS solution.

Effectively, this means iterating source databases, tables, and columns to extract their metadata first, as shown in Figures F-3 and F-4, and then looping over the metadata to generate the Biml to create the staging environment as well as the packages to populate it, as shown in Figures F-5 and F-6.

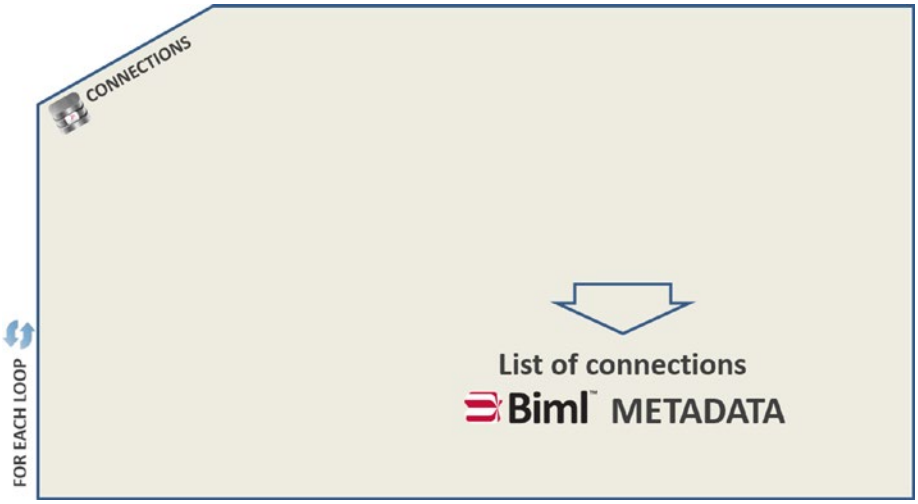

Figure F-3. *Step 1: Build your connections into Biml*

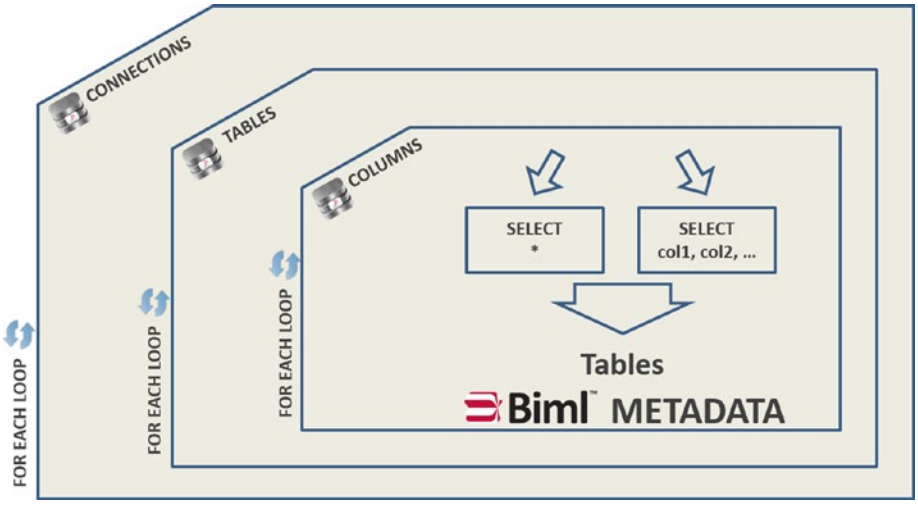

Figure F-4. *Step 2: Build your table metadata*