

Implementing Domain Specific Languages With Xtext And Xtend By Bettini Lorenzo 2013 Paperback

A step-by-step guide that enables you to quickly implement a DSL with Xtext and Xtend in a test-driven way with the aid of simplified examples. This book is for programmers who want to learn about Xtext and how to use it to implement a DSL (or a programming language) together with Eclipse IDE tooling. It assumes that the user is familiar with Eclipse and its functionality. Existing basic knowledge of a compiler implementation would be useful, though not strictly required, since the book will explain all the stages of the development of a DSL.

Domain specific languages are small, special purpose languages created to describe computational solutions in a particular problem domain. Domain specific languages have proven themselves useful many times over; however, the cost of defining and implementing a domain specific language can be high. An approach that avoids the overhead of domain specific language definition is to define an embedded language i.e. a collection of definitions in a sufficiently expressive host language. Embedding a domain specific language places high demands on a host language. The host language must be able to express the essence of the domain, while not sacrificing too much in syntax. This report, presents a suite of seven examples of embeddings using the functional programming language Haskell.

The second instance of the international summer school on Generative and Transformational Techniques in Software Engineering (GTTSE 2007) was held in Braga, Portugal, during July 2–7, 2007. This volume contains an augmented selection of the material presented at the school, including full tutorials, short tutorials, and contributions to the participants workshop. The GTTSE summer school series brings together PhD students, lecturers, technology presenters, as well as other researchers and practitioners who are interested in the generation and the transformation of programs, data, models, metamodels, documentation, and entire software systems. This concerns many areas of software engineering: software reverse and re-engineering, model-driven engineering, automated software engineering, generic language technology, to name a few. These areas differ with regard to the specific sorts of metamodels (or grammars, schemas, formats etc.) that underlie the involved artifacts, and with regard to the specific techniques that are employed for the generation and the transformation of the artifacts. The first instance of the school was held in 2005 and its proceedings appeared as volume 4143 in the LNCS series.

A guide to language implementation covers such topics as data readers, model-driven code generators, source-to-source translators, and source analyzers.

"[The authors] are pioneers. . . . Few in our industry have their breadth of knowledge and experience." —From the Foreword by Dave Thomas, Bedarra Labs Domain-Specific Modeling (DSM) is the latest approach to software development, promising to greatly increase the speed and ease of software creation. Early adopters of DSM have been enjoying productivity increases of 500–1000% in production for over a decade. This book introduces DSM and offers examples from various fields to illustrate to experienced developers how DSM can improve software development in their teams. Two authorities in the field explain what DSM is, why it works, and how to successfully create and use a DSM solution to improve productivity and quality. Divided into four parts, the book covers: background and motivation; fundamentals; in-depth examples; and creating DSM solutions. There is an emphasis throughout the book on practical guidelines for implementing DSM, including how to identify the necessary language constructs, how to generate full code from models, and how to provide tool support for a new DSM language. The example cases described in the book are available the book's Website, www.dsmbook.com, along with, an evaluation copy of the MetaEdit+ tool (for Windows, Mac OS X, and Linux), which allows readers to examine and try out the modeling languages and code generators. Domain-Specific Modeling is an essential reference for lead developers, software engineers, architects, methodologists, and technical managers who want to learn how to create a DSM solution and successfully put it into practice.

This supplement covers topics ranging from academic library funding to visual information querying.

The MPS Language Workbench, Volume I (third edition). The first volume of the series is both a simple introduction to the JetBrains MPS language workbench and a complete reference manual. The Meta-Programming System (MPS) is a new kind of tool called a language workbench that simply stated makes it easier and more fun to write programs. With traditional programming, it is common to choose one programming language to solve a problem and being limited by this choice. When working with MPS you can use and combine different languages to solve a problem. You can also create simple languages (e.g., Domain Specific Languages) or extend existing ones when the languages available do not exactly meet the evolving needs of the problem at hand. The languages that you create with MPS will integrate nicely with languages developed by others. MPS is open-source and can be obtained from <http://jetbrains.com/mps> or <http://github.com/JetBrains/MPS>. This book explains the MPS programming paradigm and gradually introduces the reader to the many features of the MPS platform. This book may yet be the simplest way to discover the MPS language workbench and the powerful new approach to programming that this tool offers. The third edition of this book describes MPS 3.3.

Build domain specific languages (DSLs) using Java's most popular functional programming language: Scala. This book introduces the basics of Scala and DSLs using a series of practical examples. In Practical Scala DSLs, you'll learn to create pragmatic and complete code examples that explain the actual use of DSLs with Scala: a web API and microservices; a custom language; a mobile app; a Forex system; a game; and cloud applications. At the end of this unique book, you'll be able to describe the differences between external and internal DSLs; understand when and how to apply DSLs; create DSLs using Scala; and even create a DSL using another programming language. What You'll Learn Build DSLs in Scala Write a web API and microservices Create a custom language Apply DSLs to mobile apps development, a Forex trading system, game development, and more Discover the role of DSLs in cloud development Integrate DSLs as part of a DevOps program or structure Build internal and external DSLs Who This Book Is For Experienced Java coders with at least some prior experience with Scala. You may be new to DSLs.

This book covers several topics related to domain-specific language (DSL) engineering in general and how they can be handled by means of the JetBrains Meta Programming System (MPS), an open source language workbench developed

by JetBrains over the last 15 years. The book begins with an overview of the domain of language workbenches, which provides perspectives and motivations underpinning the creation of MPS. Moreover, technical details of the language underneath MPS together with the definition of the tool's main features are discussed. The remaining ten chapters are then organized in three parts, each dedicated to a specific aspect of the topic. Part I "MPS in Industrial Applications" deals with the challenges and inadequacies of general-purpose languages used in companies, as opposed to the reasons why DSLs are essential, together with their benefits and efficiency, and summarizes lessons learnt by using MPS. Part II about "MPS in Research Projects" covers the benefits of text-based languages, the design and development of gamification applications, and research fields with generally low expertise in language engineering. Eventually, Part III focuses on "Teaching and Learning with MPS" by discussing the organization of both commercial and academic courses on MPS. MPS is used to implement languages for real-world use. Its distinguishing feature is projectional editing, which supports practically unlimited language extension and composition possibilities as well as a flexible mix of a wide range of textual, tabular, mathematical and graphical notations. The number and diversity of the presented use-cases demonstrate the strength and malleability of the DSLs defined using MPS. The selected contributions represent the current state of the art and practice in using JetBrains MPS to implement languages for real-world applications.

Gain an accelerated introduction to domain-specific languages in R, including coverage of regular expressions. This compact, in-depth book shows you how DSLs are programming languages specialized for a particular purpose, as opposed to general purpose programming languages. Along the way, you'll learn to specify tasks you want to do in a precise way and achieve programming goals within a domain-specific context. Domain-Specific Languages in R includes examples of DSLs including large data sets or matrix multiplication; pattern matching DSLs for application in computer vision; and DSLs for continuous time Markov chains and their applications in data science. After reading and using this book, you'll understand how to write DSLs in R and have skills you can extrapolate to other programming languages. What You'll Learn Program with domain-specific languages using R Discover the components of DSLs Carry out large matrix expressions and multiplications Implement metaprogramming with DSLs Parse and manipulate expressions Who This Book Is For Those with prior programming experience. R knowledge is helpful but not required.

Program generation holds the promise of helping to bridge the gap between application-level problem solutions and efficient implementations at the level of today's source programs as written in C or Java. Thus, program generation can substantially contribute to reducing production cost and time-to-market in future software production, while improving the quality and stability of the product. This book is about domain-specific program generation; it is the outcome of a Dagstuhl seminar on the topic held in March 2003. After an introductory preface by the volume editors, the 18 carefully reviewed revised full papers presented are organized into topical sections on - surveys of domain-specific programming technologies - domain-specific programming languages - tool support for program generation - domain-specific techniques for program optimization

Sammanfattning.

When carefully selected and used, Domain-Specific Languages (DSLs) may simplify complex code, promote effective communication with customers, improve productivity, and unclog development bottlenecks. In Domain-Specific Languages, noted software development expert Martin Fowler first provides the information software professionals need to decide if and when to utilize DSLs. Then, where DSLs prove suitable, Fowler presents effective techniques for building them, and guides software engineers in choosing the right approaches for their applications. This book's techniques may be utilized with most modern object-oriented languages; the author provides numerous examples in Java and C#, as well as selected examples in Ruby. Wherever possible, chapters are organized to be self-standing, and most reference topics are presented in a familiar patterns format. Armed with this wide-ranging book, developers will have the knowledge they need to make important decisions about DSLs—and, where appropriate, gain the significant technical and business benefits they offer. The topics covered include: How DSLs compare to frameworks and libraries, and when those alternatives are sufficient Using parsers and parser generators, and parsing external DSLs Understanding, comparing, and choosing DSL language constructs Determining whether to use code generation, and comparing code generation strategies Previewing new language workbench tools for creating DSLs

"This book presents current research on all aspects of domain-specific language for scholars and practitioners in the software engineering fields, providing new results and answers to open problems in DSL research"--

Software engineering is understood as a broad term linking science, traditional engineering, art and management and is additionally conditioned by social and external factors (conditioned to the point that brilliant engineering solutions based on strong science, showing artistic creativity and skillfully managed can still fail for reasons beyond the control of the development team). Modern software engineering needs a paradigm shift commensurate with a change of the computing paradigm from: 1. Algorithms to interactions (and from procedural to object-oriented programming) 2. Systems development to systems integration 3. Products to services Traditional software engineering struggles to address this paradigm shift to interactions, integration, and services. It offers only incomplete and disconnected methods for building information systems with fragmentary ability to dynamically accommodate change and to grow gracefully. The principal objective of contemporary software engineering should therefore be to try to redefine the entire discipline and offer a complete set of methods, tools and techniques to address challenges ahead that will shape the information systems of the future.

This book constitutes the refereed proceedings of the 17th International Conference on Model Driven Engineering Languages and Systems, MODELS 2014, held in Valencia, Spain, in September/October 2014. The 41 full papers presented in this volume were carefully reviewed and selected from a total of 126 submissions. The scope of the conference series is broad, encompassing modeling languages, methods, tools, and applications considered from theoretical and practical angles and in academic and industrial settings. The papers report on the use of modeling in a wide range of cloud, mobile, and web computing, model transformation behavioral modeling, MDE: past, present, future, formal semantics, specification, and verification, models at runtime, feature and variability modeling, composition and adaptation, practices and experience, modeling for analysis, pragmatics, model extraction, manipulation and persistence, querying, and reasoning.

Learn how to implement a DSL with Xtext and Xtend using easy-to-understand examples and best practices About This Book Leverage the latest features of Xtext and Xtend to develop a domain-specific language. Integrate Xtext with popular third party IDEs and get the best out of both worlds. Discover how to test a DSL implementation and how to customize runtime and IDE aspects of the DSL Who This Book Is For This book is targeted at programmers and developers who want to create a domain-specific language with Xtext. They should have a basic familiarity with Eclipse and its functionality. Previous experience with compiler implementation can be helpful but is not necessary since this

book will explain all the development stages of a DSL. What You Will Learn Write Xtext grammar for a DSL; Use Xtend as an alternative to Java to write cleaner, easier-to-read, and more maintainable code; Build your Xtext DSLs easily with Maven/Tycho and Gradle; Write a code generator and an interpreter for a DSL; Explore the Xtext scoping mechanism for symbol resolution; Test most aspects of the DSL implementation with JUnit; Understand best practices in DSL implementations with Xtext and Xtend; Develop your Xtext DSLs using Continuous Integration mechanisms; Use an Xtext editor in a web application In Detail Xtext is an open source Eclipse framework for implementing domain-specific languages together with IDE functionalities. It lets you implement languages really quickly; most of all, it covers all aspects of a complete language infrastructure, including the parser, code generator, interpreter, and more. This book will enable you to implement Domain Specific Languages (DSL) efficiently, together with their IDE tooling, with Xtext and Xtend. Opening with brief coverage of Xtext features involved in DSL implementation, including integration in an IDE, the book will then introduce you to Xtend as this language will be used in all the examples throughout the book. You will then explore the typical programming development workflow with Xtext when we modify the grammar of the DSL. Further, the Xtend programming language (a fully-featured Java-like language tightly integrated with Java) will be introduced. We then explain the main concepts of Xtext, such as validation, code generation, and customizations of runtime and UI aspects. You will have learned how to test a DSL implemented in Xtext with JUnit and will progress to advanced concepts such as type checking and scoping. You will then integrate the typical Continuous Integration systems built in to Xtext DSLs and familiarize yourself with Xbase. By the end of the book, you will manually maintain the EMF model for an Xtext DSL and will see how an Xtext DSL can also be used in IntelliJ. Style and approach A step-by step-tutorial with illustrative examples that will let you master using Xtext and implementing DSLs with its custom language, Xtend.

This volume contains information about the automatic acquisition of biographic knowledge from encyclopedic texts, Web interaction and the navigation problem in hypertext.

The three-volume set LNCS 12476 - 12478 constitutes the refereed proceedings of the 9th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2020, which was planned to take place during October 20–30, 2020, on Rhodes, Greece. The event itself was postponed to 2021 due to the COVID-19 pandemic. The papers presented were carefully reviewed and selected for inclusion in the proceedings. Each volume focusses on an individual topic with topical section headings within the volume: Part I, Verification Principles: Modularity and (De-)Composition in Verification; X-by-Construction: Correctness meets Probability; 30 Years of Statistical Model Checking; Verification and Validation of Concurrent and Distributed Systems. Part II, Engineering Principles: Automating Software Re-Engineering; Rigorous Engineering of Collective Adaptive Systems. Part III, Applications: Reliable Smart Contracts: State-of-the-art, Applications, Challenges and Future Directions; Automated Verification of Embedded Control Software; Formal methods for DIStributed COmputing in future RAILway systems.

Guardol, a domain-specific language (DSL) developed by Rockwell Collins, was designed to streamline the process of specifying, implementing, and verifying Cross Domain Solution (CDS) security policies. Guardol's syntax and intended computational behavior tightly resembles the core of many functional programming languages, but a number of features have been added to ease the development of high assurance cross domain solutions. A significant portion of the formalization and implementation of Guardol's grammar and type system was performed by the SANtoS group at Kansas State University. This report summarizes the key conceptual components of Guardol's grammar and tool-chain architecture. The focus of the report is a detailed description of Guardol's type system implementation and formalization. A great deal of effort was put into a formalization which provided a high level of assurance that the specification of types and data structures were maintained in the intended implementation.

This book constitutes the proceedings of the 48th International Conference on Objects, Models, Components, Patterns, held in Málaga, Spain, in June/July 2010.

The definitive resource on domain-specific languages: based on years of real-world experience, relying on modern language workbenches and full of examples. Domain-Specific Languages are programming languages specialized for a particular application domain. By incorporating knowledge about that domain, DSLs can lead to more concise and more analyzable programs, better code quality and increased development speed. This book provides a thorough introduction to DSL, relying on today's state of the art language workbenches. The book has four parts: introduction, DSL design, DSL implementation as well as the role of DSLs in various aspects of software engineering. Part I Introduction: This part introduces DSLs in general and discusses their advantages and drawbacks. It also defines important terms and concepts and introduces the case studies used in the most of the remainder of the book. Part II DSL Design: This part discusses the design of DSLs - independent of implementation techniques. It reviews seven design dimensions, explains a number of reusable language paradigms and points out a number of process-related issues. Part III DSL Implementation: This part provides details about the implementation of DSLs with lots of code. It uses three state-of-the-art but quite different language workbenches: JetBrains MPS, Eclipse Xtext and TU Delft's Spoofox. Part IV DSLs and Software Engineering: This part discusses the use of DSLs for requirements, architecture, implementation and product line engineering, as well as their roles as a developer utility and for implementing business logic. The book is available as a printed version (the one your are looking at) and as a PDF. For details see the book's companion website at <http://dslbook.org>

This book contains a selection of the papers presented at the 19th International Workshop on Functional and Constraint Logic Programming, WFLP 2010, held in Madrid, Spain, in January 2010, as part of the ACM-SIGPLAN Principles of Programming Languages event, POPL 2010. From the 15 papers submitted, 12 were accepted for presentation at the workshop. The 8 regular papers presented in this volume were selected following a second round of reviewing, which took place after the event. They are complemented by a full-length invited talk by the workshop's guest speaker, Mariangiola Dezani-Ciancaglini. All current issues in the areas of functional and constraint logic programming are covered including foundational aspects, language design, implementation, transformation and analysis, software engineering, integration of paradigms, and applications.

Extend and enhance your Java applications with domain-specific scripting in Groovy About This Book Build domain-specific mini languages in Groovy that integrate seamlessly with your Java apps with this hands-on guide Increase stakeholder participation in the development process with domain-specific scripting in Groovy Get up to speed with the newest features in Groovy using this second edition and integrate Groovy-based DSLs into your existing Java applications. Who This Book Is For This book is for Java software developers who have an interest in building domain scripting into their Java applications. No knowledge of Groovy is required, although it will be helpful. This book does not teach Groovy, but quickly introduces the basic ideas of Groovy. An experienced Java developer should have no problems with these and move quickly on to the more involved aspects of creating DSLs with Groovy. No experience of creating a DSL is required. What You Will Learn Familiarize yourself with Groovy scripting and work with Groovy closures Use the meta-programming features in Groovy to build mini languages Employ Groovy mark-up and builders to simplify application development Familiarize yourself with Groovy mark-up and build your own Groovy builders Build effective DSLs with operator overloading, command chains, builders, and a host of other Groovy language features Integrate Groovy with your Java and JVM based applications In Detail The times when developing on the JVM meant you were a Java programmer have long passed. The JVM is now firmly established as a polyglot development environment with many projects opting for alternative development languages to Java such as Groovy, Scala, Clojure, and JRuby. In this pantheon of development languages, Groovy stands out for its excellent DSL enabling features which allows it to be manipulated to produce mini languages that are tailored to a project's

needs. A comprehensive tutorial on designing and developing mini Groovy based Domain Specific Languages, this book will guide you through the development of several mini DSLs that will help you gain all the skills needed to develop your own Groovy based DSLs with confidence and ease. Starting with the bare basics, this book will focus on how Groovy can be used to construct domain specific mini languages, and will go through the more complex meta-programming features of Groovy, including using the Abstract Syntax Tree (AST). Practical examples are used throughout this book to de-mystify these seemingly complex language features and to show how they can be used to create simple and elegant DSLs. Packed with examples, including several fully worked DSLs, this book will serve as a springboard for developing your own DSLs. Style and approach This book is a hands-on guide that will walk you through examples for building DSLs with Groovy rather than just talking about "metaprogramming with Groovy". The examples in this book have been designed to help you gain a good working knowledge of the techniques involved and apply these to producing your own Groovy based DSLs.

Your success—and sanity—are closer at hand when you work at a higher level of abstraction, allowing your attention to be on the business problem rather than the details of the programming platform. Domain Specific Languages—"little languages" implemented on top of conventional programming languages—give you a way to do this because they model the domain of your business problem. DSLs in Action introduces the concepts and definitions a developer needs to build high-quality domain specific languages. It provides a solid foundation to the usage as well as implementation aspects of a DSL, focusing on the necessity of applications speaking the language of the domain. After reading this book, a programmer will be able to design APIs that make better domain models. For experienced developers, the book addresses the intricacies of domain language design without the pain of writing parsers by hand. The book discusses DSL usage and implementations in the real world based on a suite of JVM languages like Java, Ruby, Scala, and Groovy. It contains code snippets that implement real world DSL designs and discusses the pros and cons of each implementation. Purchase of the print book comes with an offer of a free PDF, ePub, and Kindle eBook from Manning. Also available is all code from the book. What's Inside Tested, real-world examples How to find the right level of abstraction Using language features to build internal DSLs Designing parser/combinator-based little languages This book details the conceptual foundations, design and implementation of the domain-specific language (DSL) development system DjDSL. DjDSL facilitates design-decision-making on and implementation of reusable DSL and DSL-product lines, and represents the state-of-the-art in language-based and composition-based DSL development. As such, it unites elements at the crossroads between software-language engineering, model-driven software engineering, and feature-oriented software engineering. The book is divided into six chapters. Chapter 1 ("DSL as Variable Software") explains the notion of DSL as variable software in greater detail and introduces readers to the idea of software-product line engineering for DSL-based software systems. Chapter 2 ("Variability Support in DSL Development") sheds light on a number of interrelated dimensions of DSL variability: variable development processes, variable design-decisions, and variability-implementation techniques for DSL. The three subsequent chapters are devoted to the key conceptual and technical contributions of DjDSL: Chapter 3 ("Variable Language Models") explains how to design and implement the abstract syntax of a DSL in a variable manner. Chapter 4 ("Variable Context Conditions") then provides the means to refine an abstract syntax (language model) by using composable context conditions (invariants). Next, Chapter 5 ("Variable Textual Syntaxes") details solutions to implementing variable textual syntaxes for different types of DSL. In closing, Chapter 6 ("A Story of a DSL Family") shows how to develop a mixed DSL in a step-by-step manner, demonstrating how the previously introduced techniques can be employed in an advanced example of developing a DSL family. The book is intended for readers interested in language-oriented as well as model-driven software development, including software-engineering researchers and advanced software developers alike. An understanding of software-engineering basics (architecture, design, implementation, testing) and software patterns is essential. Readers should especially be familiar with the basics of object-oriented modelling (UML, MOF, Ecore) and programming (e.g., Java).

Dijkstra once wrote that computer science is no more about computers than astronomy is about telescopes. Despite the many incredible advances in computer science from times that predate practical mechanical computing, there is still a myriad of fundamental questions in understanding the interface between computers and the rest of the world. Why is it still hard to mechanize many tasks that seem to be fundamentally routine, even as we see ever-increasing capacity for raw mechanical computing? The disciplined study of domain-specific languages (DSLs) is an emerging area in computer science, and is one which has the potential to revolutionize the field, and bring us closer to answering this question. DSLs are formalisms that have four general characteristics. – They relate to a well-defined domain of discourse, be it controlling traffic lights or space ships. – They have well-defined notation, such as the ones that exist for prescribing music, dance routines, or strategy in a football game. – The informal or intuitive meaning of the notation is clear. This can easily be overlooked, especially since intuitive meaning can be expressed by many different notations that may be received very differently by users. – The formal meaning is clear and mechanizable, as is, hopefully, the case for the instructions we give to our bank or to a merchant online.

Using Domain Specific Languages (DSLs), you can solve a wide spectrum of problems more rapidly and effectively than ever before. However, early adopters have struggled with the absence of sophisticated IDE tools for their new languages. Eclipse Xtext solves this problem, offering complete infrastructure for your DSLs -- from parsers, linkers, compilers, and interpreters to developer productivity features like validation and code completion. Now, for the first time, there's a comprehensive guide to developing industry-strength DSL projects with Xtext. Creating Domain-Specific Languages with Eclipse Xtext covers all you need to know to succeed with the widely-anticipated new Xtext 2 release. You'll learn: How Xtext works and how it relates to other Eclipse frameworks How to create your own DSLs with Xtext How to implement custom DSL editors that maximize user productivity How to write Xtext grammars for existing (legacy) DSLs How to integrate Xtext DSLs with other EMF models and Eclipse-based technologies How to solve "industrial-strength" problems, including scalability of large grammars and models; advanced customization; and proper build and continuous integration set-up Xtext 2.0 isn't just another DSL creation tool: it's a breakthrough enabler that will bring DSLs into far wider usage. This guide will help you get results with Xtext 2.0 fast - whether you're just started working with DSLs or you've been pioneering them for years.

The book begins with an overview of the domain of language workbenches, which provides perspectives and motivations underpinning the creation of MPS. Moreover, technical details of the language underneath MPS together with the definition of the tool's main features are discussed. The remaining ten chapters are then organized in three parts, each dedicated to a specific aspect of the topic. Part I "MPS in Industrial Applications" deals with the challenges and inadequacies of general-purpose languages used in companies, as opposed to the reasons why DSLs are essential, together with their benefits and efficiency, and summarizes lessons learnt by using MPS. Part II about "MPS in Research Projects" covers the benefits of text-based languages, the design and development of gamification applications, and research fields with generally low expertise in language engineering. Eventually, Part III focuses on "Teaching and Learning with MPS" by discussing the organization of both commercial and academic courses on MPS.

[Copyright: 3aeceeff178105afdecccb027d74923](https://www.manning.com/books/creating-domain-specific-languages-with-eclipse-xtext)