

Cqrs The Example

Describes ways to incorporate domain modeling into software development.

The creator of the incredibly popular webcomic xkcd presents his heavily researched answers to his fans' oddest questions, including "What if I took a swim in a spent-nuclear-fuel pool?" and "Could you build a jetpack using downward-firing machine guns?" 100,000 first printing.

In this new book, leading practitioner Greg Young shows how to incorporate effective domain modeling throughout the software development process, designing large and complex systems so they can be built more efficiently, dynamically, and successfully. Young takes the next steps beyond the DDD principles and best practices introduced by Eric Evans in Domain-Driven Design: Tackling Complexity in the Heart of Software. One step at a time, he explains how to use DDD with Command-Query Responsibility Separation (CQRS) to select the right design solutions and make them work in the real world. System designers and architects will learn how CQRS and event sourcing can simplify construction, decentralize decision-making, and make system development more flexible and responsive. Young also shows how DDD and CQRS make it possible to coordinate larger development teams without higher levels of management maturity. To write this book, Young has drawn on his widely-praised 3-day course on CQRS, Domain Events, Event Sourcing, and DDD. He answers many of the questions course participants have raised, shows how to overcome common architectural obstacles to DDD, and guides professionals in solving the #1 problem they've encountered: translating DDD's abstract concepts into concrete solutions.

Patterns, Domain-Driven Design (DDD), and Test-Driven Development (TDD) enable architects and developers to create systems that are powerful, robust, and maintainable. Now, there's a comprehensive, practical guide to leveraging all these techniques primarily in Microsoft .NET environments, but the discussions are just as useful for Java developers. Drawing on seminal work by Martin Fowler (Patterns of Enterprise Application Architecture) and Eric Evans (Domain-Driven Design), Jimmy Nilsson shows how to create real-world architectures for any .NET application. Nilsson illuminates each principle with clear, well-annotated code examples based on C# 1.1 and 2.0. His examples and discussions will be valuable both to C# developers and those working with other .NET languages and any databases—even with other platforms, such as J2EE.

Coverage includes · Quick primers on patterns, TDD, and refactoring · Using architectural techniques to improve software quality · Using domain models to support business rules and validation · Applying enterprise patterns to provide persistence support via NHibernate · Planning effectively for the presentation layer and UI testing · Designing for Dependency Injection, Aspect Orientation, and other new paradigms

Pro ASP.NET Web API shows you how to build flexible, extensible web services that run seamlessly on a range of operating systems and devices, from desktops to tablets to smart phones—even the ones we don't know

today. ASP.NET Web API is a new framework designed to simplify web service architecture. So if you're tired of interoperability issues between inflexible web services and clients tied to specific platforms or programming languages, or if you've ever struggled with WCF, this book is for you. To start with, you'll get up to speed on Web API's modern HTTP programming model, REST and your hosting options. You'll then dive into building a real application over a series of three chapters, so you can get an immediate feel for how this technology works in practice. The second half of the book features dedicated chapters on topics like routing, controllers, validation and tracing, and the authors close with discussions on performance, hosting and an all-important look at unit testing to help you prepare your application for the real world. ASP.NET Web API makes HTTP a first-class citizen of .NET. With Pro ASP.NET Web API, you can build HTTP-based web services for your company or business, expose your data to the world across different formats and devices and gain the best possible global reach for your application.

Ansible for DevOps

Microservices Patterns

Enabling Test-Driven Development, Domain-Driven Design, and Event-Driven Microservices

Domain-driven Design

Tackling Complexity in the Heart of Software

Microsoft .NET - Architecting Applications for the Enterprise

Building Microservices with Go

Microservices architectures offer faster change speeds, better scalability, and cleaner, evolvable system designs. But implementing your first microservices architecture is difficult. How do you make myriad choices, educate your team on all the technical details, and navigate the organization to a successful execution to maximize your chance of success? With this book, authors Ronnie Mitra and Irakli Nadareishvili provide step-by-step guidance for building an effective microservices architecture. Architects and engineers will follow an implementation journey based on techniques and architectures that have proven to work for microservices systems. You'll build an operating model, a microservices design, an infrastructure foundation, and two working microservices, then put those pieces together as a single implementation. For anyone tasked with building microservices or a microservices architecture, this guide is invaluable. Learn an effective and explicit end-to-end microservices system design Define teams, their responsibilities, and guidelines for working together Understand how to slice a big application into a collection of microservices Examine how to isolate and embed data into corresponding microservices Build a simple yet powerful CI/CD pipeline for infrastructure changes Write code for sample microservices Deploy a working microservices application on Amazon Web Services

Domain-Driven Design (DDD) is an approach to software development for complex businesses and other domains. DDD tackles that complexity by focusing the team's attention on knowledge of the domain, picking apart the most tricky, intricate problems with models, and shaping the software around those models. Easier said than done! The techniques of DDD help us approach this systematically. This reference gives a quick and authoritative summary of the key concepts of DDD. It is not meant as a learning

introduction to the subject. Eric Evans' original book and a handful of others explain DDD in depth from different perspectives. On the other hand, we often need to scan a topic quickly or get the gist of a particular pattern. That is the purpose of this reference. It is complementary to the more discursive books. The starting point of this text was a set of excerpts from the original book by Eric Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*, 2004 - in particular, the pattern summaries, which were placed in the Creative Commons by Evans and the publisher, Pearson Education. In this reference, those original summaries have been updated and expanded with new content. The practice and understanding of DDD has not stood still over the past decade, and Evans has taken this chance to document some important refinements. Some of the patterns and definitions have been edited or rewritten by Evans to clarify the original intent. Three patterns have been added, describing concepts whose usefulness and importance has emerged in the intervening years. Also, the sequence and grouping of the topics has been changed significantly to better emphasize the core principles. This is an up-to-date, quick reference to DDD.

For both beginning and experienced programmers! From the author of the multi-award-winning *Thinking in C++* and *Thinking in Java* together with a member of the Kotlin language team comes a book that breaks the concepts into small, easy-to-digest "atoms," along with exercises supported by hints and solutions directly inside IntelliJ IDEA! No programming background necessary. Summaries for experienced programmers. Easy steps via very small chapters ("atoms"). Free accompanying exercises/solutions within IntelliJ Idea. Gives you a strong Kotlin foundation. Kotlin is cleaner, more consistent and far more powerful than Java. Increase programming productivity with Kotlin's clear, concise syntax. Produce safer, more reliable programs. Kotlin easily interacts with Java. Effortlessly migrate by adding pieces of Kotlin to an existing Java project. Support for Windows, Mac and Linux. Free version of IntelliJ IDEA includes extensive Kotlin support. Book resources, live seminars, workshops and consulting available at AtomicKotlin.com.

A professional's guide to solving complex problems while designing modern software **Key Features** Learn best practices for designing enterprise-grade software systems from a seasoned CTODeeper your understanding of system reliability, maintainability, and scalabilityElevate your skills to a professional level by learning the most effective software design patterns and architectural conceptsBook Description As businesses are undergoing a digital transformation to keep up with competition, it is now more important than ever for IT professionals to design systems to keep up with the rate of change while maintaining stability. This book takes you through the architectural patterns that power enterprise-grade software systems and the key architectural elements that enable change (such as events, autonomous services, and micro frontends), along with showing you how to implement and operate anti-fragile systems. First, you'll divide up a system and define boundaries so that your teams can work autonomously and accelerate innovation. You'll cover low-level event and data patterns that support the entire architecture, while getting up and running with the different autonomous service design patterns. Next, the book will focus on best practices for security, reliability, testability, observability, and performance. You'll combine all that you've learned and build upon that foundation, exploring the methodologies of continuous experimentation, deployment, and delivery before delving into some final thoughts on how to start making progress. By the end of this book, you'll be able to architect your own event-driven, serverless systems that are ready to adapt and change so that you can deliver value at the pace needed by your business. What you will learnExplore architectural patterns to create anti-fragile systems that thrive with changeFocus on DevOps practices that empower

self-sufficient, full-stack teams
Build enterprise-scale serverless systems
Apply microservices principles to the frontend
Discover how SOLID principles apply to software and database architecture
Create event stream processors that power the event sourcing and CQRS pattern
Deploy a multi-regional system, including regional health checks, latency-based routing, and replication
Explore the Strangler pattern for migrating legacy systems
Who this book is for
This book is for software architects who want to learn more about different software design patterns and best practices. This isn't a beginner's manual – you'll need an intermediate level of programming proficiency and software design to get started. You'll get the most out of this software design book if you already know the basics of the cloud, but it isn't a prerequisite.

Delve deep into the various technical practices, principles, and values of Agile. Key Features
Discover the essence of Agile software development and the key principles of software design
Explore the fundamental practices of Agile working, including test-driven development (TDD), refactoring, pair programming, and continuous integration
Learn and apply the four elements of simple design
Book Description
The number of popular technical practices has grown exponentially in the last few years. Learning the common fundamental software development practices can help you become a better programmer. This book uses the term Agile as a wide umbrella and covers Agile principles and practices, as well as most methodologies associated with it. You'll begin by discovering how driver-navigator, chess clock, and other techniques used in the pair programming approach introduce discipline while writing code. You'll then learn to safely change the design of your code using refactoring. While learning these techniques, you'll also explore various best practices to write efficient tests. The concluding chapters of the book delve deep into the SOLID principles - the five design principles that you can use to make your software more understandable, flexible and maintainable. By the end of the book, you will have discovered new ideas for improving your software design skills, the relationship within your team, and the way your business works. What you will learn
Learn the red, green, refactor cycle of classic TDD and practice the best habits such as the rule of 3, triangulation, object calisthenics, and more
Refactor using parallel change and improve legacy code with characterization tests, approval tests, and Golden Master
Use code smells as feedback to improve your design
Learn the double cycle of ATDD and the outside-in mindset using mocks and stubs correctly in your tests
Understand how Coupling, Cohesion, Connascence, SOLID principles, and code smells are all related
Improve the understanding of your business domain using BDD and other principles for "doing the right thing, not only the thing right"
Who this book is for
This book is designed for software developers looking to improve their technical practices. Software coaches may also find it helpful as a teaching reference manual. This is not a beginner's book on how to program. You must be comfortable with at least one programming language and must be able to write unit tests using any unit testing framework.

Design Patterns for Cloud Native Applications

Build Event-Driven Architectures with Event Sourcing and CQRS

Atomic Kotlin

Domain-Driven Design in PHP

Cqrs (Command Query Responsibility Segregation)

Unit Testing Principles, Practices, and Patterns

Building Microservices

CQRS, the Example

Domain-Driven Design (DDD) software modeling delivers powerful results in practice, not just in theory, which is why developers worldwide are rapidly moving to adopt it. Now, for the first time, there's an accessible guide to the basics of DDD: What it is, what problems it solves, how it works, and how to quickly gain value from it. Concise, readable, and actionable, Domain-Driven Design Distilled never buries you in detail—it focuses on what you need to know to get results. Vaughn Vernon, author of the best-selling Implementing Domain-Driven Design, draws on his twenty years of experience applying DDD principles to real-world situations. He is uniquely well-qualified to demystify its complexities, illuminate its subtleties, and help you solve the problems you might encounter. Vernon guides you through each core DDD technique for building better software. You'll learn how to segregate domain models using the powerful Bounded Contexts pattern, to develop a Ubiquitous Language within an explicitly bounded context, and to help domain experts and developers work together to create that language. Vernon shows how to use Subdomains to handle legacy systems and to integrate multiple Bounded Contexts to define both team relationships and technical mechanisms. Domain-Driven Design Distilled brings DDD to life. Whether you're a developer, architect, analyst, consultant, or customer, Vernon helps you truly understand it so you can benefit from its remarkable power. Coverage includes What DDD can do for you and your organization—and why it's so important The cornerstones of strategic design with DDD: Bounded Contexts and Ubiquitous Language Strategic design with Subdomains Context Mapping: helping teams work together and integrate software more strategically Tactical design with Aggregates and Domain Events Using project acceleration and management tools to establish and maintain team cadence

Solve complex business problems by understanding users better, finding the right problem to solve, and building lean event-driven systems to give your customers what they really want Key FeaturesApply DDD principles using modern tools such as EventStorming, Event Sourcing, and CQRS Learn how DDD applies directly to various architectural styles such as REST, reactive systems, and microservices Empower teams to work flexibly with improved services and decoupled interactions Book Description Developers across the world are rapidly adopting DDD principles to deliver powerful results when writing software that deals with complex business requirements. This book will guide you in involving business stakeholders when choosing the software you are planning to build for them. By figuring out the temporal nature of behavior-driven domain models, you will be able to build leaner, more agile, and modular systems. You'll begin by uncovering domain complexity and learn how to capture the behavioral aspects of the domain language. You will then learn about EventStorming and advance to creating a new project in .NET Core 2.1; you'll also and write some code to transfer your events from sticky notes to C#. The book will show you how to use aggregates to handle commands and produce events. As you progress, you'll get to grips with Bounded Contexts, Context Map, Event Sourcing, and CQRS. After translating domain models into executable C# code, you will create a frontend for your application using Vue.js. In addition to this, you'll learn how to refactor your code and cover event versioning and migration essentials. By the end of this DDD book, you will have gained the confidence to implement the DDD approach in your organization and be able to explore new techniques that complement what you've learned from the book. What you will learn Discover and resolve domain complexity together with business stakeholders Avoid common pitfalls when creating the domain model Study the concept of Bounded Context and aggregate Design and build

temporal models based on behavior and not only data Explore benefits and drawbacks of Event Sourcing Get acquainted with CQRS and to-the-point read models with projections Practice building one-way flow UI with Vue.js Understand how a task-based UI conforms to DDD principles Who this book is for This book is for .NET developers who have an intermediate level understanding of C#, and for those who seek to deliver value, not just write code. Intermediate level of competence in JavaScript will be helpful to follow the UI chapters.

There are a lot of misconceptions around the CQRS pattern, especially when it comes to applying it in real-world software projects. In this book, CQRS, you will learn all about CQRS with event sourcing, database configuration for reads and writes, and more. First, you will learn exactly what CQRS is, the principles behind it, and the benefits it can provide for your project. Next, you will explore the common misconceptions and anti-patterns around CQRS. Finally, you will see a detailed, step-by-step implementation of this pattern in practice. The sample project you'll be working on is close to what you will find in the real world, and you will see each step on the way to CQRS in great detail. By the end of this book, you will have a foundational understanding of the CQRS pattern and how to implement it in your software projects.

Annotation Over the past 10 years, distributed systems have become more fine-grained. From the large multi-million line long monolithic applications, we are now seeing the benefits of smaller self-contained services. Rather than heavy-weight, hard to change Service Oriented Architectures, we are now seeing systems consisting of collaborating microservices. Easier to change, deploy, and if required retire, organizations which are in the right position to take advantage of them are yielding significant benefits. This book takes an holistic view of the things you need to be cognizant of in order to pull this off. It covers just enough understanding of technology, architecture, operations and organization to show you how to move towards finer-grained systems.

*Event-Based Java Microservices with Spring Boot and Spring Cloud
Book of F#
Practical Microservices
With examples in Java
Microservices in Action
Breaking Free with Managed Functional Programming
Pro ASP.NET Web API*

Real examples written in PHP showcasing DDD Architectural Styles, Tactical Design, and Bounded Context Integration

This Book Focuses on practical code rather than theory Full of real-world examples that you can apply to your own projects

Shows how to build PHP apps using DDD principles Who This Book Is For This book is for PHP developers who want to apply a

DDD mindset to their code. You should have a good understanding of PHP and some knowledge of DDD. This book is not

on the theory, but instead gives you the code that you need. What You Will Learn Correctly design all design elements of a

Driven Design with PHP Learn all tactical patterns to achieve a fully worked-out Domain-Driven Design Apply hexagonal architecture

Apply hexagonal architecture

Real examples written in PHP showcasing DDD Architectural Styles, Tactical Design, and Bounded Context Integration
This Book Focuses on practical code rather than theory Full of real-world examples that you can apply to your own projects
Shows how to build PHP apps using DDD principles Who This Book Is For This book is for PHP developers who want to apply a
DDD mindset to their code. You should have a good understanding of PHP and some knowledge of DDD. This book is not
on the theory, but instead gives you the code that you need. What You Will Learn Correctly design all design elements of a
Driven Design with PHP Learn all tactical patterns to achieve a fully worked-out Domain-Driven Design Apply hexagonal architecture

architecture within your application Integrate bounded contexts in your applications Use REST and Messaging approaches Detail Domain-Driven Design (DDD) has arrived in the PHP community, but for all the talk, there is very little real code being in a training session and with no PHP real examples, learning DDD can be challenging. This book changes all that by detailing how to implement tactical DDD patterns and gives full examples of topics such as integrating Bounded Contexts, REST, and DDD messaging strategies. In this book, the authors show you, with tons of details and examples, how to design Entities, Value Objects, Services, Domain Events, Aggregates, Factories, Repositories, Services, and Applications with PHP. They show how to apply Hexagonal Architecture within your application whether you use an open source framework or your own. Style and approach This highly practical book shows developers how to apply domain-driven design patterns in PHP. It is full of solid code examples to work through.

Your one-stop guide to the common patterns and practices, showing you how to apply these using the Go programming language About This Book This short, concise, and practical guide is packed with real-world examples of building microservices that is easy to read and will benefit smaller teams who want to extend the functionality of their existing systems Using this approach will save your money in terms of maintaining a monolithic architecture and demonstrate capabilities in existing systems This Book Is For You should have a working knowledge of programming in Go, including writing and compiling basic applications. However, no knowledge of RESTful architecture, microservices, or web services is expected. If you are looking to apply techniques to your own projects, taking your first steps into microservice architecture, this book is for you. Learn Plan a microservice architecture and design a microservice Write a microservice with a RESTful API and a database Understand the common idioms and common patterns in microservices architecture Leverage tools and automation to make microservices become horizontally scalable Get a grounding in containerization with Docker and Docker-Compose, which will greatly accelerate your development lifecycle Manage and secure Microservices at scale with monitoring, logging, service discovery, and automation Test microservices and integrate API tests in Go In Detail Microservice architecture is spreading all over the world as the de facto pattern to build web-based applications. Golang is a language particularly well suited to building a strong community, encouragement of idiomatic style, and statically-linked binary artifacts make integrating it with existing technologies and managing microservices at scale consistent and intuitive. This book will teach you the common patterns and practices, showing you how to apply these using the Go programming language. It will teach you the fundamental architectural design and RESTful communication, and show you patterns that provide manageable code that is supported by development and at scale in production. We will provide you with examples on how to put these concepts and patterns into practice with Go. Whether you are planning a new application or working in an existing monolith, this book will explain and illustrate with practical examples how teams of all sizes can start solving problems with microservices. It will help

Docker and Docker-Compose and how it can be used to isolate microservice dependencies and build environments. by showing you various techniques to monitor, test, and secure your microservices. By the end, you will know the system resilience of a microservice and the advantages of Go stack. Style and approach The step-by-step tutorial building microservices. Each chapter expands upon the previous one, teaching you the main skills and techniques of a successful microservice practitioner.

Vaughn Vernon presents concrete and realistic domain-driven design (DDD) techniques through examples from familiar domains, such as a Scrum-based project management application that integrates with a collaboration suite and social media. Each principle is backed up by realistic Java examples, and all content is tied together by a single case study of a company charged with delivering a set of advanced software systems with DDD.

A software architect's digest of core practices, pragmatically applied Designing effective architecture is your best way to managing project complexity—and improving your results. But the principles and practices of software architecting, as the authors call the “science of hard decisions”—have been evolving for cloud, mobile, and other shifts. Now fully revised and updated, this book shares the knowledge and real-world perspectives that enable you to design for success—and deliver successful solutions. In this fully updated Second Edition, you will: Learn how only a deep understanding of domain-driven design can inform appropriate architecture Examine domain-driven design in both theory and implementation Shift your approach to design to a model later—including multilayer architecture Capture the benefits of prioritizing software maintainability See how testability, and extensibility lead to code quality Take a user experience (UX) first approach, rather than designing for a Review patterns for organizing business logic Use event sourcing and CQRS together to model complex business domains effectively Delve inside the persistence layer, including patterns and implementation.

In 2009 I have had the pleasure of spending a 2 day course and many geek beers with Greg Young talking about Domain-Driven Design specifically focussed on Command Query Responsibility Segregation (CQRS).The example project I created based on these discussions was very well received by the community and regarded a good reference project to explain and demonstrate what that make up CQRS. I decided to add the different blog posts I wrote about the example into a single book so it is easy to read.The code can still be found at: <http://github.com/MarkNijhof/Fohjin>

Functional and Reactive Domain Modeling

Microservice Patterns and Best Practices

Applying Domain-Driven Design and Patterns

Architecture Patterns with Python

Serious Scientific Answers to Absurd Hypothetical Questions

Architecting for innovation with events, autonomous services, and micro frontends

HTTP Web Services in ASP.NET

You want increased customer satisfaction, faster development cycles, and less wasted work. Domain-driven design (DDD) combined with functional programming is the innovative combo that will get you there. In this pragmatic, down-to-earth guide, you'll see how applying the core principles of functional programming can result in software designs that model real-world requirements both elegantly and concisely - often more so than an object-oriented approach. Practical examples in the open-source F# functional language, and examples from familiar business domains, show you how to apply these techniques to build software that is business-focused, flexible, and high quality. Domain-driven design is a well-established approach to designing software that ensures that domain experts and developers work together effectively to create high-quality software. This book is the first to combine DDD with techniques from statically typed functional programming. This book is perfect for newcomers to DDD or functional programming - all the techniques you need will be introduced and explained. Model a complex domain accurately using the F# type system, creating compilable code that is also readable documentation---ensuring that the code and design never get out of sync. Encode business rules in the design so that you have "compile-time unit tests," and eliminate many potential bugs by making illegal states unrepresentable. Assemble a series of small, testable functions into a complete use case, and compose these individual scenarios into a large-scale design. Discover why the combination of functional programming and DDD leads naturally to service-oriented and hexagonal architectures. Finally, create a functional domain model that works with traditional databases, NoSQL, and event stores, and safely expose your domain via a website or API. Solve real problems by focusing on real-world requirements for your software. What You Need: The code in this book is designed to be run interactively on Windows, Mac and Linux. You will need a recent version of F# (4.0 or greater), and the appropriate .NET runtime for your platform. Full installation instructions for all platforms at fsharp.org. Radically improve your testing practice and software quality with new testing styles, good patterns, and reliable automation. Key Features A practical and results-driven approach to unit testing Refine your existing unit tests by implementing modern best practices Learn the four pillars of a good unit test Safely automate your testing process to save time and money Spot which tests need refactoring, and which need to be deleted entirely Purchase of the print book includes a free eBook in PDF, Kindle, and

ePub formats from Manning Publications. About The Book Great testing practices maximize your project quality and delivery speed by identifying bad code early in the development process. Wrong tests will break your code, multiply bugs, and increase time and costs. You owe it to yourself—and your projects—to learn how to do excellent unit testing. Unit Testing Principles, Patterns and Practices teaches you to design and write tests that target key areas of your code including the domain model. In this clearly written guide, you learn to develop professional-quality tests and test suites and integrate testing throughout the application life cycle. As you adopt a testing mindset, you'll be amazed at how better tests cause you to write better code. What You Will Learn Universal guidelines to assess any unit test Testing to identify and avoid anti-patterns Refactoring tests along with the production code Using integration tests to verify the whole system This Book Is Written For For readers who know the basics of unit testing. Examples are written in C# and can easily be applied to any language. About the Author Vladimir Khorikov is an author, blogger, and Microsoft MVP. He has mentored numerous teams on the ins and outs of unit testing. Table of Contents: PART 1 THE BIGGER PICTURE 1 | The goal of unit testing 2 | What is a unit test? 3 | The anatomy of a unit test PART 2 MAKING YOUR TESTS WORK FOR YOU 4 | The four pillars of a good unit test 5 | Mocks and test fragility 6 | Styles of unit testing 7 | Refactoring toward valuable unit tests PART 3 INTEGRATION TESTING 8 | Why integration testing? 9 | Mocking best practices 10 | Testing the database PART 4 UNIT TESTING ANTI-PATTERNS 11 | Unit testing anti-patterns "A comprehensive overview of the challenges teams face when moving to microservices, with industry-tested solutions to these problems." - Tim Moore, Lightbend 44 reusable patterns to develop and deploy reliable production-quality microservices-based applications, with worked examples in Java Key Features 44 design patterns for building and deploying microservices applications Drawing on decades of unique experience from author and microservice architecture pioneer Chris Richardson A pragmatic approach to the benefits and the drawbacks of microservices architecture Solve service decomposition, transaction management, and inter-service communication Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About The Book Microservices Patterns teaches you 44 reusable patterns to reliably develop and deploy production-quality microservices-based applications. This invaluable set of design patterns builds on decades of distributed system experience, adding new patterns for composing services into systems that scale and perform under real-world conditions. More than just a patterns catalog, this practical guide with worked examples offers industry-

tested advice to help you design, implement, test, and deploy your microservices-based application. What You Will Learn How (and why!) to use microservices architecture Service decomposition strategies Transaction management and querying patterns Effective testing strategies Deployment patterns This Book Is Written For Written for enterprise developers familiar with standard enterprise application architecture. Examples are in Java. About The Author Chris Richardson is a Java Champion, a JavaOne rock star, author of Manning's POJOs in Action, and creator of the original CloudFoundry.com. Table of Contents Escaping monolithic hell Decomposition strategies Interprocess communication in a microservice architecture Managing transactions with sagas Designing business logic in a microservice architecture Developing business logic with event sourcing Implementing queries in a microservice architecture External API patterns Testing microservices: part 1 Testing microservices: part 2 Developing production-ready services Deploying microservices Refactoring to microservices How do you detangle a monolithic system and migrate it to a microservice architecture? How do you do it while maintaining business-as-usual? As a companion to Sam Newman's extremely popular Building Microservices, this new book details a proven method for transitioning an existing monolithic system to a microservice architecture. With many illustrative examples, insightful migration patterns, and a bevy of practical advice to transition your monolith enterprise into a microservice operation, this practical guide covers multiple scenarios and strategies for a successful migration, from initial planning all the way through application and database decomposition. You'll learn several tried and tested patterns and techniques that you can use as you migrate your existing architecture. Ideal for organizations looking to transition to microservices, rather than rebuild Helps companies determine whether to migrate, when to migrate, and where to begin Addresses communication, integration, and the migration of legacy systems Discusses multiple migration patterns and where they apply Provides database migration examples, along with synchronization strategies Explores application decomposition, including several architectural refactoring patterns Delves into details of database decomposition, including the impact of breaking referential and transactional integrity, new failure modes, and more At a time when nearly every vertical, regardless of domain, seems to need software running in the cloud to make money, microservices provide the agility and drastically reduced time to market you require. This hands-on guide shows you how to create, test, compile, and deploy microservices, using the ASP.NET Core free and open-source framework. Along the way, you'll pick up good, practical habits for

building powerful and robust services. Building microservices isn't about learning a specific framework or programming language; it's about building applications that thrive in elastically scaling environments that don't have host affinity, and that can start and stop at a moment's notice. This practical book guides you through the process. Learn test-driven and API-first development concepts Communicate with other services by creating and consuming backing services such as databases and queues Build a microservice that depends on an external data source Learn about event sourcing, the event-centric approach to persistence Use ASP.NET Core to build web applications designed to thrive in the cloud Build a service that consumes, or is consumed by, other services Create services and applications that accept external configuration Explore ways to secure ASP.NET Core microservices and applications

Server and Configuration Management for Humans

Practical FP in Scala: a Hands-On Approach (2nd Edition)

Clean Architecture

Tackle Software Complexity with Domain-Driven Design and F#

Microservices: Up and Running

Practical Microservices Architectural Patterns

Building Microservices with ASP.NET Core

The standard platform for enterprise application development has been EJB but the difficulties of working with it caused it to become unpopular. They also gave rise to lightweight technologies such as Hibernate, Spring, JDO, iBATIS and others, all of which allow the developer to work directly with the simpler POJOs. Now EJB version 3 solves the problems that gave EJB 2 a black eye-it too works with POJOs. POJOs in Action describes the new, easier ways to develop enterprise Java applications. It describes how to make key design decisions when developing business logic using POJOs, including how to organize and encapsulate the business logic, access the database, manage transactions, and handle database concurrency. This book is a new-generation Java applications guide: it enables readers to successfully build lightweight applications that are easier to develop, test, and maintain.

This book predominately covers Microservices architecture with real-world example which can help professionals with ease of adoption of this technology. Following the trend of modularity in real world, the idea behind Microservice by Examples is to allow developers to build their applications from various independent components which can be easily changed, removed or upgraded. Also, it is relevant now because of enterprises are moving towards DevOps/ Modernization, this book will emphasize on containers and Dockers as well.

Read Free Cqrs The Example

This guide is focused on building highly scalable, highly available, and maintainable applications with the Command & Query Responsibility Segregation and the Event Sourcing architectural patterns. It presents a learning journey, not definitive guidance. It describes the experiences of a development team with no prior CQRS proficiency in building, deploying (to Windows Azure), and maintaining a sample real-world, complex, enterprise system to showcase various CQRS and ES concepts, challenges, and techniques. The development team did not work in isolation; we actively sought input from industry experts and from a wide group of advisors to ensure that the guidance is both detailed and practical. The CQRS pattern and event sourcing are not mere simplistic solutions to the problems associated with large-scale, distributed systems. By providing you with both a working application and written guidance, we expect you'll be well prepared to embark on your own CQRS journey.

F# brings the power of functional-first programming to the .NET Framework, a platform for developing software in the Microsoft Windows ecosystem. If you're a traditional .NET developer used to C# and Visual Basic, discovering F# will be a revelation that will change how you code, and how you think about coding. In *The Book of F#*, Microsoft MVP Dave Fancher shares his expertise and teaches you how to wield the power of F# to write succinct, reliable, and predictable code. As you learn to take advantage of features like default immutability, pipelining, type inference, and pattern matching, you'll be amazed at how efficient and elegant your code can be. You'll also learn how to:

- * Exploit F#'s functional nature using currying, partial application, and delegation
- * Streamline type creation and safety with record types and discriminated unions
- * Use collection types and modules to handle data sets more effectively
- * Use pattern matching to decompose complex types and branch your code within a single expression
- * Make your software more responsive with parallel programming and asynchronous workflows
- * Harness object orientation to develop rich frameworks and interact with code written in other .NET languages
- * Use query expressions and type providers to access and manipulate data sets from disparate sources

Break free of that old school of programming. *The Book of F#* will show you how to unleash the expressiveness of F# to create smarter, leaner code.

Ansible is a simple, but powerful, server and configuration management tool. Learn to use Ansible effectively, whether you manage one server--or thousands.

Domain-Driven Design Reference
With Examples in C# and .NET

Microservice by examples using .NET Core
Event Centric

Implementing Domain-driven Design

A Craftsman's Guide to Software Structure and Design

Read Free Cqrs The Example

With the immense cost savings and scalability the cloud provides, the rationale for building cloud native applications is no longer in question. The real issue is how. With this practical guide, developers will learn about the most commonly used design patterns for building cloud native applications using APIs, data, events, and streams in both greenfield and brownfield development. You'll learn how to incrementally design, develop, and deploy large and effective cloud native applications that you can manage and maintain at scale with minimal cost, time, and effort. Authors Kasun Indrasiri and Sriskandarajah Suhothayan highlight use cases that effectively demonstrate the challenges you might encounter at each step. Learn the fundamentals of cloud native applications Explore key cloud native communication, connectivity, and composition patterns Learn decentralized data management techniques Use event-driven architecture to build distributed and scalable cloud native applications Explore the most commonly used patterns for API management and consumption Examine some of the tools and technologies you'll need for building cloud native systems

A book for intermediate to advanced Scala developers. Aimed at those who understand functional effects, referential transparency and the benefits of functional programming to some extent but who are missing some pieces to put all these concepts together to build a large application in a time-constrained manner. Throughout the chapters we will design, architect and develop a complete stateful application serving an API via HTTP, accessing a database and dealing with cached data, using the best practices and best functional libraries available in the Cats ecosystem such as Cats Effect, Fs2, Http4s, Skunk, Refined and others. You will also learn about common design patterns such as managing state, error handling and anti-patterns, all accompanied by clear examples. Furthermore, in the Bonus Chapter, we will dive into some advanced concepts such as MTL and Optics, and will explore Fs2 streams with a few interesting examples. A digital version is also available on LeanPub.

As Python continues to grow in popularity, projects are becoming larger and more complex. Many Python developers are now taking an interest in high-level software design patterns such as hexagonal/clean architecture, event-driven architecture, and the strategic patterns prescribed by domain-driven design (DDD). But translating those patterns into Python isn't always straightforward. With this hands-on guide, Harry Percival and Bob Gregory from MADE.com introduce proven architectural design patterns to help Python developers manage application complexity—and get the most value out of their test suites. Each pattern is illustrated with concrete examples in beautiful, idiomatic Python, avoiding some of the verbosity of Java and C# syntax. Patterns include: Dependency inversion and its links to ports and adapters (hexagonal/clean architecture) Domain-driven design's distinction between entities, value objects, and aggregates Repository and Unit of Work patterns for persistent storage Events, commands, and the message bus Command-query responsibility segregation (CQRS) Event-driven architecture and reactive microservices MVC and CRUD make software easier to write, but harder to change. Microservice-based architectures can help even the smallest of projects remain agile in the long term, but most tutorials meander in theory or completely miss the point of

what it means to be microservice-based. Roll up your sleeves with real projects and learn the most important concepts of evented architectures. You'll have your own deployable, testable project and a direction for where to go next. Much ink has been spilled on the topic of microservices, but all of this writing fails to accurately identify what makes a system a monolith, define what microservices are, or give complete, practical examples, so you're probably left thinking they have nothing to offer you. You don't have to be at Google or Facebook scale to benefit from a microservice-based architecture. Microservices will keep even small and medium teams productive by keeping the pieces of your system focused and decoupled. Discover the basics of message-based architectures, render the same state in different shapes to fit the task at hand, and learn what it is that makes something a monolith (it has nothing to do with how many machines you deploy to). Conserve resources by performing background jobs with microservices. Deploy specialized microservices for registration, authentication, payment processing, e-mail, and more. Tune your services by defining appropriate service boundaries. Deploy your services effectively for continuous integration. Master debugging techniques that work across different services. You'll finish with a deployable system and skills you can apply to your current project. Add the responsiveness and flexibility of microservices to your project, no matter what the size or complexity. What You Need: While the principles of this book transcend programming language, the code examples are in Node.js because JavaScript, for better or worse, is widely read. You'll use PostgreSQL for data storage, so familiarity with it is a plus. The book does provide Docker images to make working with PostgreSQL a bit easier, but extensive Docker knowledge is not required.

Explore the concepts and tools you need to discover the world of microservices with various design patterns Key Features Get to grips with the microservice architecture and build enterprise-ready microservice applications Learn design patterns and the best practices while building a microservice application Obtain hands-on techniques and tools to create high-performing microservices resilient to possible fails Book Description Microservices are a hot trend in the development world right now. Many enterprises have adopted this approach to achieve agility and the continuous delivery of applications to gain a competitive advantage. This book will take you through different design patterns at different stages of the microservice application development along with their best practices. Microservice Patterns and Best Practices starts with the learning of microservices key concepts and showing how to make the right choices while designing microservices. You will then move onto internal microservices application patterns, such as caching strategy, asynchronism, CQRS and event sourcing, circuit breaker, and bulkheads. As you progress, you'll learn the design patterns of microservices. The book will guide you on where to use the perfect design pattern at the application development stage and how to break monolithic application into microservices. You will also be taken through the best practices and patterns involved while testing, securing, and deploying your microservice application. At the end of the book, you will easily be able to create interoperable microservices, which are testable and prepared for optimum

performance. What you will learn How to break monolithic application into microservices Implement caching strategies, CQRS and event sourcing, and circuit breaker patterns Incorporate different microservice design patterns, such as shared data, aggregator, proxy, and chained Utilize consolidate testing patterns such as integration, signature, and monkey tests Secure microservices with JWT, API gateway, and single sign on Deploy microservices with continuous integration or delivery, Blue-Green deployment Who this book is for This book is for architects and senior developers who would like implement microservice design patterns in their enterprise application development. The book assumes some prior programming knowledge.

Explore patterns like CQRS and event sourcing to create scalable, maintainable, and testable microservices

Evolutionary Patterns to Transform Your Monolith

Developing Enterprise Applications with Lightweight Frameworks

Exploring CQRS and Event Sourcing

Patterns, Principles, and Practices of Domain-Driven Design

Develop, Test, and Deploy Cross-Platform Services in the Cloud

What If?

Summary Microservices in Action is a practical book about building and deploying microservice-based applications. Written for developers and architects with a solid grasp of service-oriented development, it tackles the challenge of putting microservices into production. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Invest your time in designing great applications, improving infrastructure, and making the most out of your dev teams. Microservices are easier to write, scale, and maintain than traditional enterprise applications because they're built as a system of independent components. Master a few important new patterns and processes, and you'll be ready to develop, deploy, and run production-quality microservices. About the Book Microservices in Action teaches you how to write and maintain microservice-based applications. Created with day-to-day development in mind, this informative guide immerses you in real-world use cases from design to deployment. You'll discover how microservices enable an efficient continuous delivery pipeline, and explore examples using Kubernetes, Docker, and Google Container Engine. What's inside An overview of microservice architecture Building a delivery pipeline Best practices for designing multi-service transactions and queries Deploying with containers Monitoring your microservices About the Reader Written for intermediate developers familiar with enterprise architecture and cloud platforms like AWS and GCP. About the Author Morgan Bruce and Paulo A. Pereira are experienced engineering leaders. They work daily with microservices in a production environment, using the techniques detailed in this book. Table of Contents PART 1 - The lay of the land Designing and running microservices Microservices at SimpleBank PART 2 - Design Architecture of a microservice application Designing new features Transactions and queries in microservices Designing reliable services Building a reusable microservice framework PART 3 - Deployment Deploying microservices Deployment with

containers and schedulers Building a delivery pipeline for microservices PART 4 - Observability and ownership Building a monitoring system Using logs and traces to understand behavior Building microservice teams

Building software is harder than ever. As a developer, you not only have to chase ever-changing technological trends but also need to understand the business domains behind the software. This practical book provides you with a set of core patterns, principles, and practices for analyzing business domains, understanding business strategy, and, most importantly, aligning software design with its business needs. Author Vlad Khononov shows you how these practices lead to robust implementation of business logic and help to future-proof software design and architecture. You'll examine the relationship between domain-driven design (DDD) and other methodologies to ensure you make architectural decisions that meet business requirements. You'll also explore the real-life story of implementing DDD in a startup company. With this book, you'll learn how to: Analyze a company's business domain to learn how the system you're building fits its competitive strategy Use DDD's strategic and tactical tools to architect effective software solutions that address business needs Build a shared understanding of the business domains you encounter Decompose a system into bounded contexts Coordinate the work of multiple teams Gradually introduce DDD to brownfield projects

Methods for managing complex software construction following the practices, principles and patterns of Domain-Driven Design with code examples in C# This book presents the philosophy of Domain-Driven Design (DDD) in a down-to-earth and practical manner for experienced developers building applications for complex domains. A focus is placed on the principles and practices of decomposing a complex problem space as well as the implementation patterns and best practices for shaping a maintainable solution space. You will learn how to build effective domain models through the use of tactical patterns and how to retain their integrity by applying the strategic patterns of DDD. Full end-to-end coding examples demonstrate techniques for integrating a decomposed and distributed solution space while coding best practices and patterns advise you on how to architect applications for maintenance and scale. Offers a thorough introduction to the philosophy of DDD for professional developers Includes masses of code and examples of concept in action that other books have only covered theoretically Covers the patterns of CQRS, Messaging, REST, Event Sourcing and Event-Driven Architectures Also ideal for Java developers who want to better understand the implementation of DDD

Summary Functional and Reactive Domain Modeling teaches you how to think of the domain model in terms of pure functions and how to compose them to build larger abstractions. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Traditional distributed applications won't cut it in the reactive world of microservices, fast data, and sensor networks. To capture their dynamic relationships and dependencies, these systems require a different approach to domain modeling. A domain model composed of pure functions is a more natural way of representing a process in a reactive system, and it maps directly onto technologies and patterns like Akka, CQRS, and event sourcing. About the Book Functional and Reactive Domain Modeling teaches you consistent, repeatable techniques for building domain models in

reactive systems. This book reviews the relevant concepts of FP and reactive architectures and then methodically introduces this new approach to domain modeling. As you read, you'll learn where and how to apply it, even if your systems aren't purely reactive or functional. An expert blend of theory and practice, this book presents strong examples you'll return to again and again as you apply these principles to your own projects. What's Inside Real-world libraries and frameworks Establish meaningful reliability guarantees Isolate domain logic from side effects Introduction to reactive design patterns About the Reader Readers should be comfortable with functional programming and traditional domain modeling. Examples use the Scala language. About the Author Software architect Debasish Ghosh was an early adopter of reactive design using Scala and Akka. He's the author of DSLs in Action, published by Manning in 2010. Table of Contents Functional domain modeling: an introduction Scala for functional domain models Designing functional domain models Functional patterns for domain models Modularization of domain models Being reactive Modeling with reactive streams Reactive persistence and event sourcing Testing your domain model Summary - core thoughts and principles

Practical Software Architecture Solutions from the Legendary Robert C. Martin ("Uncle Bob") By applying universal rules of software architecture, you can dramatically improve developer productivity throughout the life of any software system. Now, building upon the success of his best-selling books Clean Code and The Clean Coder, legendary software craftsman Robert C. Martin ("Uncle Bob") reveals those rules and helps you apply them. Martin's Clean Architecture doesn't merely present options. Drawing on over a half-century of experience in software environments of every imaginable type, Martin tells you what choices to make and why they are critical to your success. As you've come to expect from Uncle Bob, this book is packed with direct, no-nonsense solutions for the real challenges you'll face—the ones that will make or break your projects. Learn what software architects need to achieve—and core disciplines and practices for achieving it Master essential software design principles for addressing function, component separation, and data management See how programming paradigms impose discipline by restricting what developers can do Understand what's critically important and what's merely a "detail" Implement optimal, high-level structures for web, database, thick-client, console, and embedded applications Define appropriate boundaries and layers, and organize components and services See why designs and architectures go wrong, and how to prevent (or fix) these failures Clean Architecture is essential reading for every current or aspiring software architect, systems analyst, system designer, and software manager—and for every programmer who must execute someone else's designs. Register your product for convenient access to downloads, updates, and/or corrections as they become available.

Monolith to Microservices

POJOs in Action

Finding Simplicity in Complex Systems

Agile Technical Practices Distilled

CQRS, the Example

Hands-On Domain-Driven Design with .NET Core
Definitions and Pattern Summaries

Take your distributed applications to the next level and see what the reference architectures associated with microservices can do for you. This book begins by showing you the distributed computing architecture landscape and provides an in-depth view of microservices architecture. Following this, you will work with CQRS, an essential pattern for microservices, and get a view of how distributed messaging works. Moving on, you will take a deep dive into Spring Boot and Spring Cloud. Coming back to CQRS, you will learn how event-driven microservices work with this pattern, using the Axon 2 framework. This takes you on to how transactions work with microservices followed by advanced architectures to address non-functional aspects such as high availability and scalability. In the concluding part of the book you develop your own enterprise-grade microservices application using the Axon framework and true BASE transactions, while making it as secure as possible. What You Will Learn Shift from monolith architecture to microservices Work with distributed and ACID transactions Build solid architectures without two-phase commit transactions Discover the high availability principles in microservices Who This Book Is For Java developers with basic knowledge of distributed and multi-threaded application architecture, and no knowledge of Spring Boot or Spring Cloud. Knowledge of CQRS and event-driven architecture is not mandatory as this book will cover these in depth.

Tackling complexity in the heart of software by putting DDD principles into practice

Software Architecture Patterns for Serverless Systems

Domain Modeling Made Functional

A Journey Into High Scalability, Availability and Maintainability with Windows Azure

Domain-Driven Design Distilled

Designing Fine-Grained Systems

Learning Domain-Driven Design