microservices patterns with examples in java pdf

microservices patterns with examples in java pdf are crucial for building robust, scalable, and maintainable distributed systems. This article delves into the core microservices patterns, explaining their purpose, benefits, and common challenges, with a strong focus on practical implementation using Java examples. We will explore essential architectural concepts like API Gateway, Service Discovery, Circuit Breaker, and various data management strategies, providing clear explanations and illustrative code snippets to solidify your understanding. Whether you're a seasoned developer looking to refine your microservices architecture or a beginner embarking on your journey, this comprehensive guide will equip you with the knowledge to design and implement effective microservices solutions in Java, making the PDF accessible for offline learning.

Understanding Microservices Patterns: The Foundation

Microservices architecture, a departure from monolithic applications, breaks down complex systems into smaller, independent services. This decomposition offers numerous advantages, including increased agility, independent deployment, and technological diversity. However, managing these distributed services introduces new complexities. Microservices patterns are established solutions to recurring problems encountered when designing, developing, and operating microservices. They provide a blueprint for addressing challenges related to communication, data management, fault tolerance, and deployment, ensuring that the benefits of microservices are fully realized.

Why Microservices Patterns are Essential

The distributed nature of microservices inherently introduces challenges that traditional monolithic architectures do not face. Without well-defined patterns, teams can quickly fall into traps that negate the very advantages microservices aim to deliver. Patterns provide a common language and a set of proven best practices, enabling teams to build resilient systems that can withstand failures, scale effectively under load, and evolve independently. They are not merely theoretical constructs but practical tools that guide developers in making informed architectural decisions, especially when dealing with complex Java applications.

Key Microservices Patterns and Their Java Implementations

This section explores some of the most fundamental and widely adopted microservices patterns. Each pattern is explained in terms of its problem, solution, and how it can be implemented using Java, often with considerations for popular frameworks and libraries.

API Gateway Pattern

The API Gateway pattern acts as a single entry point for all client requests to the microservices. It decouples clients from the underlying microservice architecture, providing a unified interface. This pattern can handle concerns like authentication, authorization, rate limiting, request routing, and response aggregation, thereby simplifying client interactions and enhancing security. For Java applications, implementing an API Gateway can be achieved using frameworks like Spring Cloud Gateway or by building a custom gateway.

Benefits of an API Gateway

- Simplified client interaction.
- Centralized cross-cutting concerns.
- Improved security by hiding internal service details.
- Protocol translation between clients and services.
- Reduced chattiness between clients and multiple services.

Java Example Considerations

When building an API Gateway in Java, developers often leverage Spring Boot for rapid development. Libraries like Spring Cloud Gateway provide declarative routing and filtering capabilities, making it straightforward to define how requests are directed to specific microservices. For instance, you might configure routes based on URL paths or headers, applying filters for authentication checks before forwarding the request to the appropriate backend service.

Service Discovery Pattern

In a dynamic microservices environment, services are frequently scaled up or down, and their network locations can change. The Service Discovery pattern addresses this by providing a mechanism for services to register themselves and for clients to discover the network locations of available service instances. This eliminates the need for hardcoding service addresses. Common implementations involve a service registry, such as Eureka or Consul.

Client-Side vs. Server-Side Discovery

- **Client-Side Discovery:** The client is responsible for querying the service registry and selecting a service instance. Popular Java libraries like Netflix Ribbon (though deprecated in favor of Spring Cloud LoadBalancer) were used for this.
- **Server-Side Discovery:** The client makes a request to a load balancer, which queries the service registry and routes the request to an available service instance. This is often the preferred approach for its simplicity at the client level.

Java Implementation with Eureka

Spring Cloud Netflix Eureka provides a robust service registry and discovery solution for Java applications. A microservice built with Spring Boot can be configured to register with a Eureka server upon startup. Other services or the API Gateway can then query Eureka to obtain the network location of instances for a particular service, enabling dynamic and resilient communication.

Circuit Breaker Pattern

The Circuit Breaker pattern is a crucial fault tolerance mechanism. It prevents a system from repeatedly trying to perform an operation that is likely to fail. If a service call fails multiple times, the circuit breaker "opens," and subsequent calls are immediately failed without attempting the actual operation. After a timeout, it enters a "half-open" state, allowing a limited number of requests to test if the downstream service has recovered. This prevents cascading failures in a distributed system.

Common States of a Circuit Breaker

1. **Closed:** All requests are allowed to pass through to the service. If failures occur, the failure count increases.

2. **Open:** All requests are immediately rejected. After a configured timeout, it transitions to half-open.

3. **Half-Open:** A limited number of requests are allowed to pass through. If these requests succeed, the circuit breaker closes; otherwise, it opens again.

Java with Resilience4j or Hystrix

Libraries like Resilience4j (a modern alternative to Netflix Hystrix) offer powerful circuit breaker implementations in Java. You can wrap service calls with a circuit breaker using annotations or programmatic configurations. For example, Resilience4j allows you to define a `CircuitBreakerRegistry` and apply a circuit breaker to a specific function, gracefully handling potential exceptions and preventing system instability.

Database per Service Pattern

In a microservices architecture, each service ideally owns its data and has its own independent database. This pattern ensures that services are truly decoupled and can evolve their data models without impacting other services. It promotes autonomy and allows teams to choose the most appropriate database technology for their specific needs. However, it also introduces challenges in managing data consistency across services.

Challenges and Solutions

- **Data Consistency:** Achieving strong consistency across multiple databases is difficult. Eventual consistency, often managed through asynchronous event-driven mechanisms, is a common approach.

- **Queries Spanning Multiple Services:** When queries require data from multiple services, patterns like API composition or CQRS (Command Query Responsibility Segregation) are employed.

Java Considerations

When implementing "database per service" in Java, each Spring Boot microservice would typically have its own data source configuration pointing to its dedicated database. For inter-service data querying, you might use a combination of REST calls orchestrated by an API Gateway or asynchronous message queues (e.g., Kafka, RabbitMQ) to propagate data changes between services, aiming for eventual consistency.

Event-Driven Architecture Patterns

Event-driven patterns are fundamental for building loosely coupled and asynchronous microservices. Services communicate by producing and consuming events, which represent state changes or significant occurrences. This asynchronous communication model enhances scalability and resilience, as services don't need to be available simultaneously to interact.

Publish-Subscribe Pattern

The publish-subscribe (pub-sub) pattern is a cornerstone of event-driven microservices. Producers publish messages (events) to a topic, and consumers subscribe to topics they are interested in. This decouples producers from consumers, allowing for flexible scaling and addition of new consumers without modifying producers. Messaging systems like Apache Kafka or RabbitMQ are commonly used for implementing pub-sub in Java microservices.

Saga Pattern for Distributed Transactions

Since each microservice has its own database, traditional ACID transactions across services are not feasible. The Saga pattern provides a way to manage data consistency across distributed services. A saga is a sequence of local transactions, where each transaction updates data within a single service. If a transaction fails, compensating transactions are executed to undo the preceding operations, ensuring the system eventually reaches a consistent state.

Java with Messaging Queues

In Java, frameworks like Spring Cloud Stream can be used to abstract the complexities of messaging brokers like Kafka or RabbitMQ. Developers can define message producers and consumers using simple interfaces and annotations. For Sagas, custom orchestration logic or frameworks like Axon Framework can be employed to manage the sequence of local transactions and their compensating actions.

Externalized Configuration Pattern

Managing configuration for numerous microservices can become cumbersome. The Externalized Configuration pattern centralizes configuration management, allowing settings to be updated without redeploying services. This is crucial for environments where configurations vary across different deployments (development, staging, production) or need frequent updates.

Benefits

- Simplified management of application settings.
- Environment-specific configurations easily applied.
- Reduced risk of configuration errors during deployment.
- Enables dynamic updates to application behavior.

Java with Spring Cloud Config

Spring Cloud Config provides a server-client architecture for externalized configuration. A Spring Cloud Config Server holds configuration properties, which can be stored in Git repositories or other backends. Microservices built with Spring Boot act as clients, fetching their configuration from the server upon startup or during runtime. This pattern is highly effective for managing Java microservice configurations at scale.

Consumer-Driven Contracts Pattern

Ensuring compatibility between services in a microservices ecosystem can be challenging, especially as services evolve independently. The Consumer-Driven Contracts (CDC) pattern addresses this by defining contracts between consumers and providers of an API. Consumers specify the interactions they expect from a provider, and these expectations are tested against the provider's implementation. This proactive approach helps prevent integration issues.

How it Works

A consumer writes tests that define the API interactions it requires. These tests generate "contracts." The provider then runs these contracts against its implementation to ensure it meets the consumers' expectations. Tools like Pact are commonly used to facilitate this pattern.

Java and Pact Integration

When working with Java microservices, you can integrate Pact to generate and verify consumer-driven contracts. Consumers write Pact tests in Java that describe their desired API interactions. These contracts are then used by the provider service to verify its endpoints against the specified requirements, fostering better communication and reducing integration friction.

Conclusion

Mastering microservices patterns is fundamental for building successful distributed systems with Java. From managing entry points with API Gateways and ensuring discoverability with Service Discovery, to implementing robust fault tolerance using Circuit Breakers and handling data consistency with patterns like Database per Service and Sagas, each pattern plays a vital role. Externalized Configuration and Consumer-Driven Contracts further enhance manageability and stability. By understanding and applying these patterns, Java developers can architect resilient, scalable, and maintainable microservices that deliver significant business value.

Frequently Asked Questions

What are the core benefits of adopting a microservices architecture?

The primary benefits include improved scalability, independent deployment of services, technology diversity (using the best tool for each job), resilience (failure in one service doesn't bring down the whole system), and faster development cycles due to smaller, focused teams. For example, a large e-commerce platform can scale its product catalog service independently during a holiday sale without affecting the user authentication service. A Java PDF detailing these benefits would often illustrate these points with architectural diagrams and case studies.

Explain the 'Database per Service' pattern and its advantages/disadvantages in a Java microservices context.

This pattern dictates that each microservice should have its own private database. Advantages include loose coupling between services, allowing independent schema evolution and technology choices (e.g., one service uses PostgreSQL, another uses MongoDB). Disadvantages can be increased complexity in data consistency and distributed transactions. A Java PDF might show code examples using Spring Data JPA for different databases per service, highlighting challenges in cross-service queries.

How does the 'API Gateway' pattern address cross-cutting concerns in Java microservices?

An API Gateway acts as a single entry point for all client requests, abstracting away the complexity of multiple microservices. It handles concerns like authentication, authorization, rate limiting, request routing, and response aggregation. For instance, in a Java microservices application,

a Spring Cloud Gateway can centralize these functionalities, simplifying client-side code. A PDF would likely demonstrate configurations and code snippets for such a gateway.

Describe the 'Saga' pattern for managing distributed transactions in Java microservices.

The Saga pattern is used to maintain data consistency across multiple microservices without relying on traditional ACID transactions. It involves a sequence of local transactions, each updating its own database and publishing an event to trigger the next transaction. If a step fails, compensating transactions are executed to undo previous changes. A Java PDF might illustrate this with a Spring Boot application using Kafka for event-driven communication between services, demonstrating rollback logic.

What is the 'Service Discovery' pattern and how is it implemented in Java microservices?

Service Discovery allows services to find and communicate with each other without hardcoding their network locations. Common implementations involve a registry (like Eureka or Consul) where services register themselves upon startup and query for other services. In a Java context, Spring Cloud Netflix Eureka is a popular choice. A PDF could show Java code for service registration and client-side discovery using a `RestTemplate` or `WebClient`.

Discuss the 'Circuit Breaker' pattern and its role in improving fault tolerance in Java microservices.

The Circuit Breaker pattern prevents a microservice from repeatedly trying to execute an operation that's likely to fail. If a service experiences a high rate of failures, the circuit breaker 'opens,' preventing further calls for a configurable period. After the timeout, it enters a 'half-open' state to test if the service has recovered. Hystrix (though now in maintenance mode) or Resilience4j are common Java libraries for implementing this. A PDF would explain the states (closed, open, half-open) and provide Java code examples.

Explain the 'Event-Driven Architecture' pattern and its application in Java microservices.

In an Event-Driven Architecture (EDA), services communicate by producing and consuming events. This promotes loose coupling and asynchronous communication. For example, when an order is placed, an 'OrderPlaced' event is published. Other services (like inventory or shipping) can subscribe to this event and react accordingly. Java examples often involve message brokers like Kafka or RabbitMQ, using frameworks like Spring Kafka or Spring AMQP. A PDF would detail event schemas and listener implementations.

What are the challenges of testing microservices, and what patterns can help?

Testing microservices is complex due to their distributed nature. Challenges include integration testing, end-to-end testing, and mocking dependencies. Patterns like 'Contract Testing' (e.g., Pact) ensure that services communicate according to agreed-upon interfaces. 'Consumer-Driven Contract Testing' is a key approach. A Java PDF might show how to write consumer and provider tests in Java for verifying contracts between services.

How can the 'CQRS' (Command Query Responsibility Segregation) pattern be beneficial in Java microservices?

CQRS separates read operations (queries) from write operations (commands). This allows for optimizing each path independently. For instance, a microservice managing product inventory might have a highly optimized read model for displaying products to customers and a separate, optimized write model for handling stock updates. Java implementations might use different data stores or even different read and write APIs. A PDF could illustrate this with a Spring Boot application demonstrating separate repositories and controllers for commands and queries.

Additional Resources

Here are 9 book titles related to microservices patterns with examples in Java, each with a short description:

1. *Microservices Patterns: With examples in Java*
This foundational book delves into the complexities of building and managing microservice architectures. It meticulously explains common patterns, such as API Gateway, Service Discovery, and Circuit Breaker, providing practical Java code examples to illustrate their implementation. The authors guide readers through designing robust, scalable, and resilient microservices systems from the ground up.

2. *Mastering Microservices with Java: Patterns and Best Practices*
This comprehensive guide focuses on practical application and best practices for Java developers venturing into microservices. It covers essential design patterns for inter-service communication, data management, and resilience, all reinforced with Java code snippets. The book emphasizes building maintainable and production-ready microservices by incorporating industry-standard techniques.

3. *Spring Boot Microservices: Patterns and Practices for Building Scalable Applications*
Leveraging the popular Spring Boot framework, this book explores microservice patterns specifically tailored for Java developers. It showcases how to

implement patterns like asynchronous messaging, distributed tracing, and command query responsibility segregation (CQRS) using Spring Boot. The content is rich with practical examples for creating robust and observable microservices.

4. *Building Microservices with Java: Design Patterns for Distributed Systems*
This resource offers a deep dive into the design principles and patterns crucial for distributed microservices. It provides clear explanations of concepts like event-driven architectures, domain-driven design (DDD) in a microservices context, and strategies for managing eventual consistency, all with Java examples. The book aims to equip readers with the knowledge to build complex, yet manageable, distributed systems.

5. *Hands-On Microservices Patterns in Java: A Practical Approach*
As the title suggests, this book takes a hands-on approach to understanding microservices patterns. Through numerous Java code examples, readers will learn to implement solutions for common challenges in microservice development, including handling failures, deploying services, and securing communications. It's ideal for developers who prefer learning by doing.

6. *Cloud Native Java: Designing Resilient Microservices with Spring Boot, Spring Cloud, and Cloud Foundry*
While broader in scope, this book extensively covers microservice patterns essential for cloud-native development. It details how to leverage Spring Cloud components to implement patterns like configuration management, resilience patterns, and routing for Java applications. The book is invaluable for those aiming to build and deploy microservices on cloud platforms.

7. *Java Microservices: Design Patterns for the Enterprise*
This book targets enterprise-level microservice development using Java. It explores patterns that address the unique challenges of large organizations, such as distributed transaction management, security patterns, and strategies for migrating from monolithic architectures. Readers will find practical Java examples for implementing these advanced patterns.

8. *Effective Microservices in Java: A Pragmatic Guide to Design and Implementation*
Focusing on practicality, this guide provides actionable advice and clear Java examples for implementing microservice patterns. It covers essential patterns for inter-service communication, data persistence, and fault tolerance, emphasizing strategies that lead to maintainable and efficient microservices. The book aims to help developers make informed decisions when designing their microservice systems.

9. *Microservices Architecture with Java: Patterns for Distributed Systems Explained*
This book offers a thorough explanation of microservices patterns within the context of Java development. It systematically breaks down each pattern, providing conceptual understanding and then demonstrating its implementation with clear, concise Java code. The focus is on building scalable, fault-

tolerant, and manageable distributed systems.

Microservices Patterns With Examples In Java Pdf

Microservices Patterns with Examples in Java: A Deep Dive

This ebook provides a comprehensive exploration of microservices architecture, focusing on practical implementation patterns using Java, backed by recent research and best practices to enhance application scalability, maintainability, and resilience. We'll delve into various design patterns, common challenges, and effective solutions, illustrated with concrete Java code examples, making it ideal for developers seeking to master microservices development.

Ebook Title: Mastering Microservices with Java: Patterns, Practices, and Production-Ready Solutions

Outline:

Introduction: Understanding Microservices Architecture and its Benefits
Chapter 1: Key Microservices Design Patterns: Exploring common architectural and communication patterns.
Chapter 2: Implementing Microservices in Java: A hands-on guide to building microservices with Spring Boot.
Chapter 3: Data Management in Microservices: Strategies for handling data consistency and transactions.
Chapter 4: Inter-Service Communication: Examining synchronous and asynchronous communication methods (REST, gRPC, Kafka).
Chapter 5: Microservices Security: Securing your microservices architecture against common threats.
Chapter 6: Monitoring and Logging in Microservices: Implementing robust monitoring and logging for observability.
Chapter 7: Testing Microservices: Strategies for unit, integration, and end-to-end testing.
Chapter 8: Deployment and Orchestration: Utilizing containerization (Docker, Kubernetes) for efficient deployment.
Chapter 9: Advanced Microservices Concepts: Exploring topics like circuit breakers, service meshes, and chaos engineering.
Conclusion: Recap and future trends in microservices architecture.

Detailed Outline Explanation:

Introduction: This section lays the groundwork by defining microservices, contrasting them with monolithic architectures, and highlighting the advantages (scalability, flexibility, independent deployments, technology diversity) and challenges (increased complexity, distributed transactions, operational overhead) associated with adopting a microservices approach. It will set the stage for the subsequent chapters.

Chapter 1: Key Microservices Design Patterns: This chapter explores various architectural patterns like Saga pattern (for handling distributed transactions), CQRS (Command Query Responsibility Segregation), Event Sourcing, and API Gateway patterns. It also examines communication patterns like synchronous (REST) and asynchronous (message queues like RabbitMQ or Kafka). Each pattern will be explained with clear diagrams and concise examples.

Chapter 2: Implementing Microservices in Java: This is a practical chapter showing how to build microservices using the popular Spring Boot framework. We'll cover setting up projects, using Spring Data for data access, creating RESTful APIs, and configuring dependencies. Code examples will be provided for building simple microservices. Specific examples could include creating a user service, a product service, and an order service.

Chapter 3: Data Management in Microservices: This chapter addresses the complexities of data management in a distributed system. Topics include choosing appropriate database technologies (SQL, NoSQL), ensuring data consistency using techniques like eventual consistency, and handling distributed transactions using sagas or two-phase commit protocols. The challenges of data synchronization and schema evolution will also be discussed.

Chapter 4: Inter-Service Communication: This section deep dives into different communication mechanisms. It will thoroughly explain RESTful APIs using Spring REST controllers, gRPC for high-performance communication, and message queues (Kafka, RabbitMQ) for asynchronous communication and event-driven architectures. The trade-offs of each approach will be analyzed.

Chapter 5: Microservices Security: Securing microservices is crucial. This chapter discusses authentication and authorization mechanisms (OAuth 2.0, JWT), securing APIs using HTTPS, input validation, and implementing robust logging and monitoring for security incidents. Best practices for securing sensitive data within and between services will be detailed.

Chapter 6: Monitoring and Logging in Microservices: Effective monitoring and logging are vital for identifying and resolving issues in a distributed system. This chapter covers tools and techniques for centralized logging, metrics collection (using Prometheus, Micrometer), tracing (using Zipkin, Jaeger), and creating dashboards for monitoring system health and performance.

Chapter 7: Testing Microservices: Thorough testing is paramount. This chapter explores different testing strategies, including unit testing individual services, integration testing interactions between services, and end-to-end testing the entire system. It will highlight the use of mocking frameworks and testing frameworks within the Spring ecosystem.

Chapter 8: Deployment and Orchestration: This chapter focuses on deploying and managing microservices using containerization technologies like Docker and Kubernetes. It will guide readers through building Docker images, deploying to Kubernetes clusters, and using Kubernetes features for scaling, load balancing, and rolling updates.

Chapter 9: Advanced Microservices Concepts: This chapter explores more advanced topics such as circuit breakers (Hystrix, Resilience4j) for handling service failures, service meshes (Istio, Linkerd)

for managing service-to-service communication, and chaos engineering for proactively testing the resilience of the system.

Conclusion: This section summarizes the key takeaways from the ebook, reiterates the importance of microservices architecture, and briefly discusses emerging trends and future directions in the field, highlighting areas for continued learning and research.

FAQs

1. What is the difference between microservices and monolithic architecture? Microservices break down an application into small, independent services, while monolithic architecture has all components within a single application.

2. What are the benefits of using Spring Boot for microservices? Spring Boot simplifies development with auto-configuration, dependency injection, and a streamlined development process.

3. How do I handle data consistency in a microservices architecture? Techniques include eventual consistency, sagas, and two-phase commit, each with trade-offs.

4. What are some popular message queues for inter-service communication? Kafka and RabbitMQ are widely used for asynchronous communication.

5. How can I secure my microservices? Implement HTTPS, OAuth 2.0, JWT, input validation, and robust logging for security.

6. What tools are used for monitoring microservices? Prometheus, Micrometer, Zipkin, and Jaeger are commonly used for monitoring and tracing.

7. How do I test microservices effectively? Employ unit, integration, and end-to-end testing strategies.

8. What is the role of Docker and Kubernetes in microservices deployment? Docker provides containerization, while Kubernetes orchestrates the deployment and management of containers.

9. What are some advanced concepts in microservices? Circuit breakers, service meshes, and chaos engineering enhance resilience and observability.

Related Articles:

1. Spring Boot Microservices Tutorial: A step-by-step guide to building your first Spring Boot microservice.

2. Microservices Architecture Design Patterns: A deep dive into various design patterns for

microservices, including Saga, CQRS, and Event Sourcing.

3. Implementing Microservices Security Best Practices: A comprehensive guide to securing your microservices architecture against common threats.

4. Choosing the Right Database for Your Microservices: A comparison of various database technologies suitable for microservices architectures.

5. Asynchronous Communication in Microservices with Kafka: A detailed tutorial on using Apache Kafka for asynchronous communication.

6. Monitoring and Logging Microservices with Prometheus and Grafana: A practical guide to setting up and using Prometheus and Grafana for microservices monitoring.

7. Testing Microservices with JUnit and Mockito: A tutorial on using JUnit and Mockito for unit and integration testing.

8. Deploying Microservices to Kubernetes: A comprehensive guide to deploying and managing your microservices on Kubernetes.

9. Implementing Circuit Breakers in Microservices with Resilience4j: Learn how to use Resilience4j to build fault-tolerant microservices.

microservices patterns with examples in java pdf: *Microservices Patterns* Chris Richardson, 2018-10-27 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

microservices patterns with examples in java pdf: *POJOs in Action* Chris Richardson,

2006-02-02 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.

microservices patterns with examples in java pdf: Practical Microservices Architectural Patterns Binildas Christudas, 2019-06-25 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 transactionsBuild 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.

microservices patterns with examples in java pdf: Learn Microservices with Spring Boot Moises Macero, 2017-12-08 Build a microservices architecture with Spring Boot, by evolving an application from a small monolith to an event-driven architecture composed of several services. This book follows an incremental approach to teach microservice structure, test-driven development, Eureka, Ribbon, Zuul, and end-to-end tests with Cucumber. Author Moises Macero follows a very pragmatic approach to explain the benefits of using this type of software architecture, instead of keeping you distracted with theoretical concepts. He covers some of the state-of-the-art techniques in computer programming, from a practical point of view. You'll focus on what's important, starting with the minimum viable product but keeping the flexibility to evolve it. What You'll Learn Build microservices with Spring Boot Use event-driven architecture and messaging with RabbitMQ Create RESTful services with Spring Master service discovery with Eureka and load balancing with Ribbon Route requests with Zuul as your API gateway Write end-to-end rests for an event-driven architecture using Cucumber Carry out continuous integration and deployment Who This Book Is For Those with at least some prior experience with Java programming. Some prior exposure to Spring Boot recommended but not required.

microservices patterns with examples in java pdf: *Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach* Shahir Daya, Nguyen Van Duy, Kameswara Eati, Carlos M Ferreira, Dejan Glozic, Vasfi Gucer, Manav Gupta, Sunil Joshi, Valerie Lampkin, Marcelo Martins, Shishir Narain, Ramratan Vennam, IBM Redbooks, 2016-04-04 Microservices is an architectural style in which large, complex software applications are composed of one or more smaller services. Each of these microservices focuses on completing one task that represents a small business capability. These microservices can be developed in any programming language. They communicate with each other using language-neutral protocols, such as Representational State Transfer (REST), or messaging applications, such as IBM® MQ Light. This

IBM Redbooks® publication gives a broad understanding of this increasingly popular architectural style, and provides some real-life examples of how you can develop applications using the microservices approach with IBM BluemixTM. The source code for all of these sample scenarios can be found on GitHub (https://github.com/). The book also presents some case studies from IBM products. We explain the architectural decisions made, our experiences, and lessons learned when redesigning these products using the microservices approach. Information technology (IT) professionals interested in learning about microservices and how to develop or redesign an application in Bluemix using microservices can benefit from this book.

microservices patterns with examples in java pdf: Microservices for the Enterprise Kasun Indrasiri, Prabath Siriwardena, 2018-11-14 Understand the key challenges and solutions around building microservices in the enterprise application environment. This book provides a comprehensive understanding of microservices architectural principles and how to use microservices in real-world scenarios. Architectural challenges using microservices with service integration and API management are presented and you learn how to eliminate the use of centralized integration products such as the enterprise service bus (ESB) through the use of composite/integration microservices. Concepts in the book are supported with use cases, and emphasis is put on the reality that most of you are implementing in a “brownfield” environment in which you must implement microservices alongside legacy applications with minimal disruption to your business. Microservices for the Enterprise covers state-of-the-art techniques around microservices messaging, service development and description, service discovery, governance, and data management technologies and guides you through the microservices design process. Also included is the importance of organizing services as core versus atomic, composite versus integration, and API versus edge, and how such organization helps to eliminate the use of a central ESB and expose services through an API gateway. What You'll LearnDesign and develop microservices architectures with confidence Put into practice the most modern techniques around messaging technologies Apply the Service Mesh pattern to overcome inter-service communication challenges Apply battle-tested microservices security patterns to address real-world scenarios Handle API management, decentralized data management, and observability Who This Book Is For Developers and DevOps engineers responsible for implementing applications around a microservices architecture, and architects and analysts who are designing such systems

microservices patterns with examples in java pdf: Microservice Patterns and Best Practices Vinicius Feitosa Pacheco, 2018-01-31 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.

microservices patterns with examples in java pdf: Mastering Microservices with Java 9 Sourabh Sharma, 2017-12-07 Master the art of implementing scalable microservices in your production environment with ease About This Book Use domain-driven design to build microservices Use Spring Cloud to use Service Discovery and Registeration Use Kafka, Avro and Spring Streams for implementing event based microservices Who This Book Is For This book is for Java developers who are familiar with the microservices architecture and now wants to take a deeper dive into effectively implementing microservices at an enterprise level. A reasonable knowledge level and understanding of core microservice elements and applications is expected. What You Will Learn Use domain-driven design to design and implement microservices Secure microservices using Spring Security Learn to develop REST service development Deploy and test microservices Troubleshoot and debug the issues faced during development Learning best practices and common principals about microservices In Detail Microservices are the next big thing in designing scalable, easy-to-maintain applications. It not only makes app development easier, but also offers great flexibility to utilize various resources optimally. If you want to build an enterprise-ready implementation of the microservices architecture, then this is the book for you! Starting off by understanding the core concepts and framework, you will then focus on the high-level design of large software projects. You will gradually move on to setting up the development environment and configuring it before implementing continuous integration to deploy your microservice architecture. Using Spring security, you will secure microservices and test them effectively using REST Java clients and other tools like RxJava 2.0. We'll show you the best patterns, practices and common principals of microservice design and you'll learn to troubleshoot and debug the issues faced during development. We'll show you how to design and implement reactive microservices. Finally, we'll show you how to migrate a monolithic application to microservices based application. By the end of the book, you will know how to build smaller, lighter, and faster services that can be implemented easily in a production environment. Style and approach This book starts from the basics, including environment setup and provides easy-to-follow steps to implement the sample project using microservices.

microservices patterns with examples in java pdf: *Enterprise Java Microservices* Kenneth Finnigan, 2018-09-27 Summary Enterprise Java Microservices is an example-rich tutorial that shows how to design and manage large-scale Java applications as a collection of microservices. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Large applications are easier to develop and maintain when you build them from small, simple components. Java developers now enjoy a wide range of tools that support microservices application development, including right-sized app servers, open source frameworks, and well-defined patterns. Best of all, you can build microservices applications using your existing Java skills. About the Book Enterprise Java Microservices teaches you to design and build JVM-based microservices applications. You'll start by learning how microservices designs compare to traditional Java EE applications. Always practical, author Ken Finnigan introduces big-picture concepts along with the tools and techniques you'll need to implement them. You'll discover ecosystem components like Netflix Hystrix for fault tolerance and master the Just enough Application Server (JeAS) approach. To ensure smooth operations, you'll also examine monitoring, security, testing, and deploying to the cloud. What's inside The microservices mental model Cloud-native development Strategies for fault tolerance and monitoring Securing your finished applications About the Reader This book is for Java developers familiar with Java EE. About the Author Ken Finnigan leads the Thorntail project at Red Hat, which seeks to make developing microservices for the cloud with Java

and Java EE as easy as possible. Table of Contents PART 1 MICROSERVICES BASICS Enterprise Java microservices Developing a simple RESTful microservice Just enough Application Server for microservices Microservices testing Cloud native development PART 2 - IMPLEMENTING ENTERPRISE JAVA MICROSERVICES Consuming microservices Discovering microservices for consumption Strategies for fault tolerance and monitoring Securing a microservice Architecting a microservice hybrid Data streaming with Apache Kafka

microservices patterns with examples in java pdf: Spring Microservices in Action John Carnell, Kalpit Patel, 2017-06-11 Summary Spring Microservices in Action teaches you how to build microservice-based applications using Java and the Spring platform. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the technology Microservices break up your code into small, distributed, and independent services that require careful forethought and design. Fortunately, Spring Boot and Spring Cloud simplify your microservice applications, just as the Spring Framework simplifies enterprise Java development. Spring Boot removes the boilerplate code involved with writing a REST-based service. Spring Cloud provides a suite of tools for the discovery, routing, and deployment of microservices to the enterprise and the cloud. About the Book Spring Microservices in Action teaches you how to build microservice-based applications using Java and the Spring platform. You'll learn to do microservice design as you build and deploy your first Spring Cloud application. Throughout the book, carefully selected real-life examples expose microservice-based patterns for configuring, routing, scaling, and deploying your services. You'll see how Spring's intuitive tooling can help augment and refactor existing applications with micro services. What's Inside Core microservice design principles Managing configuration with Spring Cloud Config Client-side resiliency with Spring, Hystrix, and Ribbon Intelligent routing using Netflix Zuul Deploying Spring Cloud applications About the Reader This book is written for developers with Java and Spring experience. About the Author John Carnell is a senior cloud engineer with twenty years of experience in Java. Table of contents Welcome to the cloud, Spring Building microservices with Spring Boot Controlling your configuration with Spring Cloud configuration server On service discovery When bad things happen: client resiliency patterns with Spring Cloud and Netflix Hystrix Service routing with Spring Cloud and Zuul Securing your microservices Event-driven architecture with Spring Cloud Stream Distributed tracing with Spring Cloud Sleuth and Zipkin Deploying your microservices

microservices patterns with examples in java pdf: Building Event-Driven Microservices Adam Bellemare, 2020-07-02 Organizations today often struggle to balance business requirements with ever-increasing volumes of data. Additionally, the demand for leveraging large-scale, real-time data is growing rapidly among the most competitive digital industries. Conventional system architectures may not be up to the task. With this practical guide, you'll learn how to leverage large-scale data usage across the business units in your organization using the principles of event-driven microservices. Author Adam Bellemare takes you through the process of building an event-driven microservice-powered organization. You'll reconsider how data is produced, accessed, and propagated across your organization. Learn powerful yet simple patterns for unlocking the value of this data. Incorporate event-driven design and architectural principles into your own systems. And completely rethink how your organization delivers value by unlocking near-real-time access to data at scale. You'll learn: How to leverage event-driven architectures to deliver exceptional business value The role of microservices in supporting event-driven designs Architectural patterns to ensure success both within and between teams in your organization Application patterns for developing powerful event-driven microservices Components and tooling required to get your microservice ecosystem off the ground

microservices patterns with examples in java pdf: The Art of Scalability Martin L. Abbott, Michael T. Fisher, 2015-05-23 The Comprehensive, Proven Approach to IT Scalability–Updated with New Strategies, Technologies, and Case Studies In The Art of Scalability, Second Edition, leading scalability consultants Martin L. Abbott and Michael T. Fisher cover everything you need to know to smoothly scale products and services for any requirement. This extensively revised edition reflects

new technologies, strategies, and lessons, as well as new case studies from the authors' pioneering consulting practice, AKF Partners. Writing for technical and nontechnical decision-makers, Abbott and Fisher cover everything that impacts scalability, including architecture, process, people, organization, and technology. Their insights and recommendations reflect more than thirty years of experience at companies ranging from eBay to Visa, and Salesforce.com to Apple. You'll find updated strategies for structuring organizations to maximize agility and scalability, as well as new insights into the cloud (IaaS/PaaS) transition, NoSQL, DevOps, business metrics, and more. Using this guide's tools and advice, you can systematically clear away obstacles to scalability–and achieve unprecedented IT and business performance. Coverage includes • Why scalability problems start with organizations and people, not technology, and what to do about it • Actionable lessons from real successes and failures • Staffing, structuring, and leading the agile, scalable organization • Scaling processes for hyper-growth environments • Architecting scalability: proprietary models for clarifying needs and making choices–including 15 key success principles • Emerging technologies and challenges: data cost, datacenter planning, cloud evolution, and customer-aligned monitoring • Measuring availability, capacity, load, and performance

microservices patterns with examples in java pdf: *Monolith to Microservices* Sam Newman, 2019-11-14 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

microservices patterns with examples in java pdf: Pro Spring Boot 2 Felipe Gutierrez, 2018-12-12 Quickly and productively develop complex Spring applications and microservices out of the box, with minimal concern over things like configurations. This revised book will show you how to fully leverage the Spring Boot 2 technology and how to apply it to create enterprise ready applications that just work. It will also cover what's been added to the new Spring Boot 2 release, including Spring Framework 5 features like WebFlux, Security, Actuator and the new way to expose Metrics through Micrometer framework, and more. This book is your authoritative hands-on practical guide for increasing your enterprise Java and cloud application productivity while decreasing development time. It's a no nonsense guide with case studies of increasing complexity throughout the book. The author, a senior solutions architect and Principal Technical instructor with Pivotal, the company behind the Spring Framework, shares his experience, insights and first-hand knowledge about how Spring Boot technology works and best practices. Pro Spring Boot 2 is an essential book for your Spring learning and reference library. What You Will Learn Configure and use Spring Boot Use non-functional requirements with Spring Boot Actuator Carry out web development with Spring Boot Persistence with JDBC, JPA and NoSQL Databases Messaging with JMS, RabbitMQ and WebSockets Test and deploy with Spring Boot A quick look at the Spring Cloud projects Microservices and deployment to the Cloud Extend Spring Boot by creating your own Spring Boot Starter and @Enable feature Who This Book Is For Experienced Spring and Java developers seeking increased productivity gains and decreased complexity and development time in their applications and software services.

microservices patterns with examples in java pdf: *Production-Ready Microservices* Susan J. Fowler, 2016-11-30 One of the biggest challenges for organizations that have adopted microservice architecture is the lack of architectural, operational, and organizational standardization. After splitting a monolithic application or building a microservice ecosystem from scratch, many engineers are left wondering what's next. In this practical book, author Susan Fowler presents a set of microservice standards in depth, drawing from her experience standardizing over a thousand microservices at Uber. You'll learn how to design microservices that are stable, reliable, scalable, fault tolerant, performant, monitored, documented, and prepared for any catastrophe. Explore production-readiness standards, including: Stability and Reliability: develop, deploy, introduce, and deprecate microservices; protect against dependency failures Scalability and Performance: learn essential components for achieving greater microservice efficiency Fault Tolerance and Catastrophe Preparedness: ensure availability by actively pushing microservices to fail in real time Monitoring: learn how to monitor, log, and display key metrics; establish alerting and on-call procedures Documentation and Understanding: mitigate tradeoffs that come with microservice adoption, including organizational sprawl and technical debt

microservices patterns with examples in java pdf: Present and Ulterior Software Engineering Manuel Mazzara, Bertrand Meyer, 2017-11-01 This book provides an effective overview of the state-of-the art in software engineering, with a projection of the future of the discipline. It includes 13 papers, written by leading researchers in the respective fields, on important topics like model-driven software development, programming language design, microservices, software reliability, model checking and simulation. The papers are edited and extended versions of the presentations at the PAUSE symposium, which marked the completion of 14 years of work at the Chair of Software Engineering at ETH Zurich. In this inspiring context, some of the greatest minds in the field extensively discussed the past, present and future of software engineering. It guides readers on a voyage of discovery through the discipline of software engineering today, offering unique food for thought for researchers and professionals, and inspiring future research and development.

microservices patterns with examples in java pdf: *Java EE 8 Design Patterns and Best Practices* Rhuan Rocha, João Purificação, 2018-08-10 Get the deep insights you need to master efficient architectural design considerations and solve common design problems in your enterprise applications. Key Features The benefits and applicability of using different design patterns in JAVA EE Learn best practices to solve common design and architectural challenges Choose the right patterns to improve the efficiency of your programs Book Description Patterns are essential design tools for Java developers. Java EE Design Patterns and Best Practices helps developers attain better code quality and progress to higher levels of architectural creativity by examining the purpose of each available pattern and demonstrating its implementation with various code examples. This book will take you through a number of patterns and their Java EE-specific implementations. In the beginning, you will learn the foundation for, and importance of, design patterns in Java EE, and then will move on to implement various patterns on the presentation tier, business tier, and integration tier. Further, you will explore the patterns involved in Aspect-Oriented Programming (AOP) and take a closer look at reactive patterns. Moving on, you will be introduced to modern architectural patterns involved in composing microservices and cloud-native applications. You will get acquainted with security patterns and operational patterns involved in scaling and monitoring, along with some patterns involved in deployment. By the end of the book, you will be able to efficiently address common problems faced when developing applications and will be comfortable working on scalable and maintainable projects of any size. What you will learn Implement presentation layers, such as the front controller pattern Understand the business tier and implement the business delegate pattern Master the implementation of AOP Get involved with asynchronous EJB methods and REST services Involve key patterns in the adoption of microservices architecture Manage performance and scalability for enterprise-level applications Who this book is for Java developers who are comfortable with programming in Java and now want to learn how to implement design patterns to create robust,

reusable and easily maintainable apps.

microservices patterns with examples in java pdf: *Building Microservices with Go* Nic Jackson, 2017-07-27 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 with Go It is easy to read and will benefit smaller teams who want to extend the functionality of their existing systems Using this practical approach will save your money in terms of maintaining a monolithic architecture and demonstrate capabilities in ease of use Who 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. What You Will 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 that helps 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 sweeping the world as the de facto pattern to build web-based applications. Golang is a language particularly well suited to building them. Its strong community, encouragement of idiomatic style, and statically-linked binary artifacts make integrating it with other 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 concepts of architectural design and RESTful communication, and show you patterns that provide manageable code that is supportable in 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 you understand Docker and Docker-Compose and how it can be used to isolate microservice dependencies and build environments. We finish off by showing you various techniques to monitor, test, and secure your microservices. By the end, you will know the benefits of system resilience of a microservice and the advantages of Go stack. Style and approach The step-by-step tutorial focuses on building microservices. Each chapter expands upon the previous one, teaching you the main skills and techniques required to be a successful microservice practitioner.

microservices patterns with examples in java pdf: *Microservices Best Practices for Java* Michael Hofmann, Erin Schnabel, Katherine Stanley, IBM Redbooks, 2017-03-13 Microservices is an architectural style in which large, complex software applications are composed of one or more smaller services. Each of these microservices focuses on completing one task that represents a small business capability. These microservices can be developed in any programming language. This IBM® Redbooks® publication covers Microservices best practices for Java. It focuses on creating cloud native applications using the latest version of IBM WebSphere® Application Server Liberty, IBM Bluemix® and other Open Source Frameworks in the Microservices ecosystem to highlight Microservices best practices for Java.

microservices patterns with examples in java pdf: Microservices in Action Morgan Bruce, Paulo A Pereira, 2018-10-03 The one [and only] book on implementing microservices with a real-world, cover-to-cover example you can relate to. - Christian Bach, Swiss Re 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 Designing and running microservices Microservices at SimpleBank Architecture of a microservice application Designing new features Transactions and queries in microservices Designing reliable services Building a reusable microservice framework Deploying microservices Deployment with containers and schedulers Building a delivery pipeline for microservices Building a monitoring system Using logs and traces to understand behavior Building microservice teams PART 1 - The lay of the land PART 2 - Design PART 3 - Deployment PART 4 - Observability and ownership

microservices patterns with examples in java pdf: *Java Design Patterns* Vaskaran Sarcar, 2018-12-06 Get hands-on experience implementing 26 of the most common design patterns using Java and Eclipse. In addition to Gang of Four (GoF) design patterns, you will also learn about alternative design patterns, and understand the criticisms of design patterns with an overview of anti-patterns. For each pattern you will see at least one real-world scenario, a computer-world example, and a complete implementation including output. This book has three parts. The first part covers 23 Gang of Four (GoF) design patterns. The second part includes three alternative design patterns. The third part presents criticisms of design patterns with an overview of anti-patterns. You will work through easy-to-follow examples to understand the concepts in depth and you will have a collection of programs to port over to your own projects. A Q&A session is included in each chapter and covers the pros and cons of each pattern. The last chapter presents FAQs about the design patterns. The step-by-step approach of the book helps you apply your skills to learn other patterns on your own, and to be familiar with the latest version of Java and Eclipse. What You'll Learn Work with each of the design patterns Implement design patterns in real-world applications Choose from alternative design patterns by comparing their pros and cons Use the Eclipse IDE to write code and generate output Read the in-depth Q&A session in each chapter with pros and cons for each design pattern Who This Book Is For Software developers, architects, and programmers

microservices patterns with examples in java pdf: Microservices Eberhard Wolff, 2016-10-03 The Most Complete, Practical, and Actionable Guide to Microservices Going beyond mere theory and marketing hype, Eberhard Wolff presents all the knowledge you need to capture the full benefits of this emerging paradigm. He illuminates microservice concepts, architectures, and scenarios from a technology-neutral standpoint, and demonstrates how to implement them with today’s leading technologies such as Docker, Java, Spring Boot, the Netflix stack, and Spring Cloud. The author fully explains the benefits and tradeoffs associated with microservices, and guides you through the entire project lifecycle: development, testing, deployment, operations, and more. You’ll find best practices for architecting microservice-based systems, individual microservices, and nanoservices, each illuminated with pragmatic examples. The author supplements opinions based on his experience with concise essays from other experts, enriching your understanding and illuminating areas where experts disagree. Readers are challenged to experiment on their own the concepts explained in the book to gain hands-on experience. Discover what microservices are, and how they differ from other forms of modularization Modernize legacy applications and efficiently build new systems Drive more value from continuous delivery with microservices Learn how

microservices differ from SOA Optimize the microservices project lifecycle Plan, visualize, manage, and evolve architecture Integrate and communicate among microservices Apply advanced architectural techniques, including CQRS and Event Sourcing Maximize resilience and stability Operate and monitor microservices in production Build a full implementation with Docker, Java, Spring Boot, the Netflix stack, and Spring Cloud Explore nanoservices with Amazon Lambda, OSGi, Java EE, Vert.x, Erlang, and Seneca Understand microservices' impact on teams, technical leaders, product owners, and stakeholders Managers will discover better ways to support microservices, and learn how adopting the method affects the entire organization. Developers will master the technical skills and concepts they need to be effective. Architects will gain a deep understanding of key issues in creating or migrating toward microservices, and exactly what it will take to transform their plans into reality.

microservices patterns with examples in java pdf: SRE with Java Microservices Jonathan Schneider, 2020-08-27 In a microservices architecture, the whole is indeed greater than the sum of its parts. But in practice, individual microservices can inadvertently impact others and alter the end user experience. Effective microservices architectures require standardization on an organizational level with the help of a platform engineering team. This practical book provides a series of progressive steps that platform engineers can apply technically and organizationally to achieve highly resilient Java applications. Author Jonathan Schneider covers many effective SRE practices from companies leading the way in microservices adoption. You'll examine several patterns discovered through much trial and error in recent years, complete with Java code examples. Chapters are organized according to specific patterns, including: Application metrics: Monitoring for availability with Micrometer Debugging with observability: Logging and distributed tracing; failure injection testing Charting and alerting: Building effective charts; KPIs for Java microservices Safe multicloud delivery: Spinnaker, deployment strategies, and automated canary analysis Source code observability: Dependency management, API utilization, and end-to-end asset inventory Traffic management: Concurrency of systems; platform, gateway, and client-side load balancing

microservices patterns with examples in java pdf: *Java Program Design* Edward Sciore, 2018-12-08 Get a grounding in polymorphism and other fundamental aspects of object-oriented program design and implementation, and learn a subset of design patterns that any practicing Java professional simply must know in today's job climate. Java Program Design presents program design principles to help practicing programmers up their game and remain relevant in the face of changing trends and an evolving language. The book enhances the traditional design patterns with Java's new functional programming features, such as functional interfaces and lambda expressions. The result is a fresh treatment of design patterns that expands their power and applicability, and reflects current best practice. The book examines some well-designed classes from the Java class library, using them to illustrate the various object-oriented principles and patterns under discussion. Not only does this approach provide good, practical examples, but you will learn useful library classes you might not otherwise know about. The design of a simplified banking program is introduced in chapter 1 in a non-object-oriented incarnation and the example is carried through all chapters. You can see the object orientation develop as various design principles are progressively applied throughout the book to produce a refined, fully object-oriented version of the program in the final chapter. What You'll Learn Create well-designed programs, and identify and improve poorly-designed ones Build a professional-level understanding of polymorphism and its use in Java interfaces and class hierarchies Apply classic design patterns to Java programming problems while respecting the modern features of the Java language Take advantage of classes from the Java library to facilitate the implementation of design patterns in your programs Who This Book Is For Java programmers who are comfortable writing non-object-oriented code and want a guided immersion into the world of object-oriented Java, and intermediate programmers interested in strengthening their foundational knowledge and taking their object-oriented skills to the next level. Even advanced programmers will discover interesting examples and insights in each chapter.

microservices patterns with examples in java pdf: Modernizing Enterprise Java Markus

Eisele, Natale Vinto, 2021-10-21 While containers, microservices, and distributed systems dominate discussions in the tech world, the majority of applications in use today still run monolithic architectures that follow traditional development processes. This practical book helps developers examine long-established Java-based models and demonstrates how to bring these monolithic applications successfully into the future. Relying on their years of experience modernizing applications, authors Markus Eisele and Natale Vinto walk you through the steps necessary to update your organization's Java applications. You'll discover how to dismantle your monolithic application and move to an up-to-date software stack that works across cloud and on-premises installations. Learn cloud native application basics to understand what parts of your organization's Java-based applications and platforms need to migrate and modernize Understand how enterprise Java specifications can help you transition projects and teams Build a cloud native platform that supports effective development without falling into buzzword traps Find a starting point for your migration projects by identifying candidates and staging them through modernization steps Discover how to complement a traditional enterprise Java application with components on top of containers and Kubernetes

microservices patterns with examples in java pdf: Testing Java Microservices Jason Porter, Alex Soto, Andrew Gumbrecht, 2018-08-03 Summary Testing Java Microservices teaches you to implement unit and integration tests for microservice systems running on the JVM. You'll work with a microservice environment built using Java EE, WildFly Swarm, and Docker. You'll learn how to increase your test coverage and productivity, and gain confidence that your system will work as you expect. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Microservice applications present special testing challenges. Even simple services need to handle unpredictable loads, and distributed message-based designs pose unique security and performance concerns. These challenges increase when you throw in asynchronous communication and containers. About the Book Testing Java Microservices teaches you to implement unit and integration tests for microservice systems running on the JVM. You'll work with a microservice environment built using Java EE, WildFly Swarm, and Docker. You'll advance from writing simple unit tests for individual services to more-advanced practices like chaos or integration tests. As you move towards a continuous-delivery pipeline, you'll also master live system testing using technologies like the Arquillian, Wiremock, and Mockito frameworks, along with techniques like contract testing and over-the-wire service virtualization. Master these microservice-specific practices and tools and you'll greatly increase your test coverage and productivity, and gain confidence that your system will work as you expect. What's Inside Test automation Integration testing microservice systems Testing container-centric systems Service virtualization About the Reader Written for Java developers familiar with Java EE, EE4J, Spring, or Spring Boot. About the Authors Alex Soto Bueno and Jason Porter are Arquillian team members. Andy Gumbrecht is an Apache TomEE developer and PMC. They all have extensive enterprise-testing experience. Table of Contents An introduction to microservices Application under test Unit-testing microservices Component-testing microservices Integration-testing microservices Contract tests End-to-end testing Docker and testing Service virtualization Continuous delivery in microservices

microservices patterns with examples in java pdf: Mastering Microservices with Java Sourabh Sharma, 2019-02-26 Master the art of implementing scalable and reactive microservices in your production environment with Java 11 Key FeaturesUse domain-driven designs to build microservicesExplore various microservices design patterns such as service discovery, registration, and API GatewayUse Kafka, Avro, and Spring Streams to implement event-based microservicesBook Description Microservices are key to designing scalable, easy-to-maintain applications. This latest edition of Mastering Microservices with Java, works on Java 11. It covers a wide range of exciting new developments in the world of microservices, including microservices patterns, interprocess communication with gRPC, and service orchestration. This book will help you understand how to implement microservice-based systems from scratch. You'll start off by understanding the core

concepts and framework, before focusing on the high-level design of large software projects. You'll then use Spring Security to secure microservices and test them effectively using REST Java clients and other tools. You will also gain experience of using the Netflix OSS suite, comprising the API Gateway, service discovery and registration, and Circuit Breaker. Additionally, you'll be introduced to the best patterns, practices, and common principles of microservice design that will help you to understand how to troubleshoot and debug the issues faced during development. By the end of this book, you'll have learned how to build smaller, lighter, and faster services that can be implemented easily in a production environment. What you will learnUse domain-driven designs to develop and implement microservicesUnderstand how to implement microservices using Spring BootExplore service orchestration and distributed transactions using the SagasDiscover interprocess communication using REpresentational State Transfer (REST) and eventsGain knowledge of how to implement and design reactive microservicesDeploy and test various microservicesWho this book is for This book is designed for Java developers who are familiar with microservices architecture and now want to effectively implement microservices at an enterprise level. Basic knowledge and understanding of core microservice elements and applications is necessary.

microservices patterns with examples in java pdf: Design Patterns and Best Practices in Java Kamalmeet Singh, Adrian Ianculescu, Lucian-Paul Torje, 2018-06-27 Create various design patterns to master the art of solving problems using Java Key Features This book demonstrates the shift from OOP to functional programming and covers reactive and functional patterns in a clear and step-by-step manner All the design patterns come with a practical use case as part of the explanation, which will improve your productivity Tackle all kinds of performance-related issues and streamline your development Book Description Having a knowledge of design patterns enables you, as a developer, to improve your code base, promote code reuse, and make the architecture more robust. As languages evolve, new features take time to fully understand before they are adopted en masse. The mission of this book is to ease the adoption of the latest trends and provide good practices for programmers. We focus on showing you the practical aspects of smarter coding in Java. We'll start off by going over object-oriented (OOP) and functional programming (FP) paradigms, moving on to describe the most frequently used design patterns in their classical format and explain how Java’s functional programming features are changing them. You will learn to enhance implementations by mixing OOP and FP, and finally get to know about the reactive programming model, where FP and OOP are used in conjunction with a view to writing better code. Gradually, the book will show you the latest trends in architecture, moving from MVC to microservices and serverless architecture. We will finish off by highlighting the new Java features and best practices. By the end of the book, you will be able to efficiently address common problems faced while developing applications and be comfortable working on scalable and maintainable projects of any size. What you will learn Understand the OOP and FP paradigms Explore the traditional Java design patterns Get to know the new functional features of Java See how design patterns are changed and affected by the new features Discover what reactive programming is and why is it the natural augmentation of FP Work with reactive design patterns and find the best ways to solve common problems using them See the latest trends in architecture and the shift from MVC to serverless applications Use best practices when working with the new features Who this book is for This book is for those who are familiar with Java development and want to be in the driver’s seat when it comes to modern development techniques. Basic OOP Java programming experience and elementary familiarity with Java is expected.

microservices patterns with examples in java pdf: Kubernetes Native Microservices with Quarkus and MicroProfile John Clingan, Ken Finnigan, 2022-03-01 Build fast, efficient Kubernetes-based Java applications using the Quarkus framework, MicroProfile, and Java standards. In Kubernetes Native Microservices with Quarkus and MicroProfile you’ll learn how to: Deploy enterprise Java applications on Kubernetes Develop applications using the Quarkus runtime Compile natively using GraalVM for blazing speed Create efficient microservices applications Take advantage of MicroProfile specifications Popular Java frameworks like Spring were designed long before

Kubernetes and the microservices revolution. Kubernetes Native Microservices with Quarkus and MicroProfile introduces next generation tools that have been cloud-native and Kubernetes-aware right from the beginning. Written by veteran Java developers John Clingan and Ken Finnigan, this book shares expert insight into Quarkus and MicroProfile directly from contributors at Red Hat. You'll learn how to utilize these modern tools to create efficient enterprise Java applications that are easy to deploy, maintain, and expand. About the technology Build microservices efficiently with modern Kubernetes-first tools! Quarkus works naturally with containers and Kubernetes, radically simplifying the development and deployment of microservices. This powerful framework minimizes startup time and memory use, accelerating performance and reducing hosting cost. And because it's Java from the ground up, it integrates seamlessly with your existing JVM codebase. About the book Kubernetes Native Microservices with Quarkus and MicroProfile teaches you to build microservices using containers, Kubernetes, and the Quarkus framework. You'll immediately start developing a deployable application using Quarkus and the MicroProfile APIs. Then, you'll explore the startup and runtime gains Quarkus delivers out of the box and also learn how to supercharge performance by compiling natively using GraalVM. Along the way, you'll see how to integrate a Quarkus application with Spring and pick up pro tips for monitoring and managing your microservices. What's inside Deploy enterprise Java applications on Kubernetes Develop applications using the Quarkus runtime framework Compile natively using GraalVM for blazing speed Take advantage of MicroProfile specifications About the reader For intermediate Java developers comfortable with Java EE, Jakarta EE, or Spring. Some experience with Docker and Kubernetes required. About the author John Clingan is a senior principal product manager at Red Hat, where he works on enterprise Java standards and Quarkus. Ken Finnigan is a senior principal software engineer at Workday, previously at Red Hat working on Quarkus. Table of Contents PART 1 INTRODUCTION 1 Introduction to Quarkus, MicroProfile, and Kubernetes 2 Your first Quarkus application PART 2 DEVELOPING MICROSERVICES 3 Configuring microservices 4 Database access with Panache 5 Clients for consuming other microservices 6 Application health 7 Resilience strategies 8 Reactive in an imperative world 9 Developing Spring microservices with Quarkus PART 3 OBSERVABILITY, API DEFINITION, AND SECURITY OF MICROSERVICES 10 Capturing metrics 11 Tracing microservices 12 API visualization 13 Securing a microservice

microservices patterns with examples in java pdf: Building Microservices Sam Newman, 2015-02-02 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.

microservices patterns with examples in java pdf: *The Tao of Microservices* Richard Rodger, 2017-12-11 Summary The Tao of Microservices guides you on the path to understanding how to apply microservice architectures to your own real-world projects. This high-level book offers a conceptual view of microservice design, along with core concepts and their application. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology An application, even a complex one, can be designed as a system of independent components, each of which handles a single responsibility. Individual microservices are easy for small teams without extensive knowledge of the entire system design to build and maintain. Microservice applications rely on modern patterns like asynchronous, message-based communication, and they can be optimized to work well in cloud and container-centric environments. About the Book The Tao of Microservices guides you on the path to understanding and building microservices. Based on the invaluable experience of microservices guru Richard Rodger, this book exposes the thinking behind microservice designs. You'll master individual

concepts like asynchronous messaging, service APIs, and encapsulation as you learn to apply microservices architecture to real-world projects. Along the way, you'll dig deep into detailed case studies with source code and documentation and explore best practices for team development, planning for change, and tool choice. What's Inside Principles of the microservice architecture Breaking down real-world case studies Implementing large-scale systems When not to use microservices About the Reader This book is for developers and architects. Examples use JavaScript and Node.js. About the Author Richard Rodger, CEO of voxgig, a social network for the events industry, has many years of experience building microservice-based systems for major global companies. Table of Contents PART 1 - BUILDING MICROSERVICES Brave new world Services Messages Data Deployment PART 2 - RUNNING MICROSERVICES Measurement Migration People Case study: Nodezoo.com

microservices patterns with examples in java pdf: Docker and Kubernetes for Java Developers Jaroslaw Krochmalski, 2017-08-30 Leverage the lethal combination of Docker and Kubernetes to automate deployment and management of Java applications About This Book Master using Docker and Kubernetes to build, deploy and manage Java applications in a jiff Learn how to create your own Docker image and customize your own cluster using Kubernetes Empower the journey from development to production using this practical guide. Who This Book Is For The book is aimed at Java developers who are eager to build, deploy, and manage applications very quickly using container technology. They need have no knowledge of Docker and Kubernetes. What You Will Learn Package Java applications into Docker images Understand the running of containers locally Explore development and deployment options with Docker Integrate Docker into Maven builds Manage and monitor Java applications running on Kubernetes clusters Create Continuous Delivery pipelines for Java applications deployed to Kubernetes In Detail Imagine creating and testing Java EE applications on Apache Tomcat Server or Wildfly Application server in minutes along with deploying and managing Java applications swiftly. Sounds too good to be true? But you have a reason to cheer as such scenarios are only possible by leveraging Docker and Kubernetes. This book will start by introducing Docker and delve deep into its networking and persistent storage concepts. You will then proceed to learn how to refactor monolith application into separate services by building an application and then packaging it into Docker containers. Next, you will create an image containing Java Enterprise Application and later run it using Docker. Moving on, the book will focus on Kubernetes and its features and you will learn to deploy a Java application to Kubernetes using Maven and monitor a Java application in production. By the end of the book, you will get hands-on with some more advanced topics to further extend your knowledge about Docker and Kubernetes. Style and approach An easy-to-follow, practical guide that will help Java developers develop, deploy, and manage Java applications efficiently.

microservices patterns with examples in java pdf: Service Design Patterns Robert Daigneau, 2012 Forewords by Martin Fowler and Ian Robinson--From front cover.

microservices patterns with examples in java pdf: Patterns, Principles, and Practices of Domain-Driven Design Scott Millett, Nick Tune, 2015-04-20 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

microservices patterns with examples in java pdf: *Jakarta EE Cookbook* Elder Moraes, 2020-05-29 An enterprise Java developer's guide to learning JAX-RS, context and dependency injection, JavaServer Faces (JSF), and microservices with Eclipse MicroProfile using the latest features of Jakarta EE Key FeaturesExplore Jakarta EE's latest features and API specifications and discover their benefitsBuild and deploy microservices using Jakarta EE 8 and Eclipse MicroProfileBuild robust RESTful web services for various enterprise scenarios using the JAX-RS, JSON-P, and JSON-B APIsBook Description Jakarta EE is widely used around the world for developing enterprise applications for a variety of domains. With this book, Java professionals will be able to enhance their skills to deliver powerful enterprise solutions using practical recipes. This second edition of the Jakarta EE Cookbook takes you through the improvements introduced in its latest version and helps you get hands-on with its significant APIs and features used for server-side development. You'll use Jakarta EE for creating RESTful web services and web applications with the JAX-RS, JSON-P, and JSON-B APIs and learn how you can improve the security of your enterprise solutions. Not only will you learn how to use the most important servers on the market, but you'll also learn to make the best of what they have to offer for your project. From an architectural point of view, this Jakarta book covers microservices, cloud computing, and containers. It allows you to explore all the tools for building reactive applications using Jakarta EE and core Java features such as lambdas. Finally, you'll discover how professionals can improve their projects by engaging with and contributing to the community. By the end of this book, you'll have become proficient in developing and deploying enterprise applications using Jakarta EE. What you will learnWork with Jakarta EE's most commonly used APIs and features for server-side developmentEnable fast and secure communication in web applications with the help of HTTP2Build enterprise applications with reusable componentsBreak down monoliths into microservices using Jakarta EE and Eclipse MicroProfileImprove your enterprise applications with multithreading and concurrencyRun applications in the cloud with the help of containersGet to grips with continuous delivery and deployment for shipping your applications effectivelyWho this book is for This book is for Java EE developers who want to build enterprise applications or update their legacy apps with Jakarta EE's latest features and specifications. Some experience of working with Java EE and knowledge of web and cloud computing will assist with understanding the concepts covered in this book.

microservices patterns with examples in java pdf: Building Microservices with .NET Core Gaurav Kumar Aroraa, Lalit Kale, Kanwar Manish, 2017-06-14 Architect your .NET applications by breaking them into really small pieces—microservices—using this practical, example-based guide About This Book Start your microservices journey and understand a broader perspective of microservices development Build, deploy, and test microservices using ASP.Net MVC, Web API, and Microsoft Azure Cloud Get started with reactive microservices and understand the fundamentals behind it Who This Book Is For This book is for .NET Core developers who want to learn and understand microservices architecture and implement it in their .NET Core applications. It's ideal for developers who are completely new to microservices or have just a theoretical understanding of this architectural approach and want to gain a practical perspective in order to better manage application complexity. What You Will Learn Compare microservices with monolithic applications and SOA Identify the appropriate service boundaries by mapping them to the relevant bounded contexts Define the service interface and implement the APIs using ASP.NET Web API Integrate the services via synchronous and asynchronous mechanisms Implement microservices security using Azure Active Directory, OpenID Connect, and OAuth 2.0 Understand the operations and scaling of microservices in .NET Core Understand the testing pyramid and implement consumer-driven contract using pact net core Understand what the key features of reactive microservices are and implement them using reactive extension In Detail Microservices is an architectural style that promotes the development of complex applications as a suite of small services based on business capabilities. This book will help you identify the appropriate service boundaries within the business. We'll start by looking at what microservices are, and what the main

characteristics are. Moving forward, you will be introduced to real-life application scenarios, and after assessing the current issues, we will begin the journey of transforming this application by splitting it into a suite of microservices. You will identify the service boundaries, split the application into multiple microservices, and define the service contracts. You will find out how to configure, deploy, and monitor microservices, and configure scaling to allow the application to quickly adapt to increased demand in the future. With an introduction to the reactive microservices, you strategically gain further value to keep your code base simple, focusing on what is more important rather than the messy asynchronous calls. Style and approach This guide serves as a stepping stone that helps .NET Core developers in their microservices architecture. This book provides just enough theory to understand the concepts and apply the examples.

microservices patterns with examples in java pdf: Kubernetes Patterns Bilgin Ibryam, Roland Huß, 2019-04-09 The way developers design, build, and run software has changed significantly with the evolution of microservices and containers. These modern architectures use new primitives that require a different set of practices than most developers, tech leads, and architects are accustomed to. With this focused guide, Bilgin Ibryam and Roland Huß from Red Hat provide common reusable elements, patterns, principles, and practices for designing and implementing cloud-native applications on Kubernetes. Each pattern includes a description of the problem and a proposed solution with Kubernetes specifics. Many patterns are also backed by concrete code examples. This book is ideal for developers already familiar with basic Kubernetes concepts who want to learn common cloud native patterns. You'll learn about the following pattern categories: Foundational patterns cover the core principles and practices for building container-based cloud-native applications. Behavioral patterns explore finer-grained concepts for managing various types of container and platform interactions. Structural patterns help you organize containers within a pod, the atom of the Kubernetes platform. Configuration patterns provide insight into how application configurations can be handled in Kubernetes. Advanced patterns covers more advanced topics such as extending the platform with operators.

microservices patterns with examples in java pdf: Microservices and Containers Parminder Singh Kocher, 2018-03-16 Transition to Microservices and DevOps to Transform Your Software Development Effectiveness Thanks to the tech sector's latest game-changing innovations—the Internet of Things (IoT), software-enabled networking, and software as a service (SaaS), to name a few—there is now a seemingly insatiable demand for platforms and architectures that can improve the process of application development and deployment. In Microservices and Containers, longtime systems architect and engineering team leader Parminder Kocher analyzes two of the hottest new technology trends: microservices and containers. Together, as Kocher demonstrates, microservices and Docker containers can bring unprecedented agility and scalability to application development and deployment, especially in large, complex projects where speed is crucial but small errors can be disastrous. Learn how to leverage microservices and Docker to drive modular architectural design, on-demand scalability, application performance and reliability, time-to-market, code reuse, and exponential improvements in DevOps effectiveness. Kocher offers detailed guidance and a complete roadmap for transitioning from monolithic architectures, as well as an in-depth case study that walks the reader through the migration of an enterprise-class SOA system. Understand how microservices enable you to organize applications into standalone components that are easier to manage, update, and scale Decide whether microservices and containers are worth your investment, and manage the organizational learning curve associated with them Apply best practices for interprocess communication among microservices Migrate monolithic systems in an orderly fashion Understand Docker containers, installation, and interfaces Network, orchestrate, and manage Docker containers effectively Use Docker to maximize scalability in microservices-based applications Apply your learning with an in-depth, hands-on case study Whether you are a software architect/developer or systems professional looking to move on from older approaches or a manager trying to maximize the business value of these technologies, Microservices and Containers will be an invaluable addition to your library. Register your product at

informit.com/register for convenient access to downloads, updates, and/or corrections as they become available.

microservices patterns with examples in java pdf: *Evolve the Monolith to Microservices with Java and Node* Sandro De Santis, Luis Florez, Duy V Nguyen, Eduardo Rosa, IBM Redbooks, 2016-12-05 Microservices is an architectural style in which large, complex software applications are composed of one or more smaller services. Each of these microservices focuses on completing one task that represents a small business capability. These microservices can be developed in any programming language. This IBM® Redbooks® publication shows how to break out a traditional Java EE application into separate microservices and provides a set of code projects that illustrate the various steps along the way. These code projects use the IBM WebSphere® Application Server Liberty, IBM API ConnectTM, IBM Bluemix®, and other Open Source Frameworks in the microservices ecosystem. The sample projects highlight the evolution of monoliths to microservices with Java and Node.

microservices patterns with examples in java pdf: *Patterns of Enterprise Application Architecture* Martin Fowler, 2012-03-09 The practice of enterprise application development has benefited from the emergence of many new enabling technologies. Multi-tiered object-oriented platforms, such as Java and .NET, have become commonplace. These new tools and technologies are capable of building powerful applications, but they are not easily implemented. Common failures in enterprise applications often occur because their developers do not understand the architectural lessons that experienced object developers have learned. Patterns of Enterprise Application Architecture is written in direct response to the stiff challenges that face enterprise application developers. The author, noted object-oriented designer Martin Fowler, noticed that despite changes in technology--from Smalltalk to CORBA to Java to .NET--the same basic design ideas can be adapted and applied to solve common problems. With the help of an expert group of contributors, Martin distills over forty recurring solutions into patterns. The result is an indispensable handbook of solutions that are applicable to any enterprise application platform. This book is actually two books in one. The first section is a short tutorial on developing enterprise applications, which you can read from start to finish to understand the scope of the book's lessons. The next section, the bulk of the book, is a detailed reference to the patterns themselves. Each pattern provides usage and implementation information, as well as detailed code examples in Java or C#. The entire book is also richly illustrated with UML diagrams to further explain the concepts. Armed with this book, you will have the knowledge necessary to make important architectural decisions about building an enterprise application and the proven patterns for use when building them. The topics covered include · Dividing an enterprise application into layers · The major approaches to organizing business logic · An in-depth treatment of mapping between objects and relational databases · Using Model-View-Controller to organize a Web presentation · Handling concurrency for data that spans multiple transactions · Designing distributed object interfaces

Back to Home: https://new.teachat.com