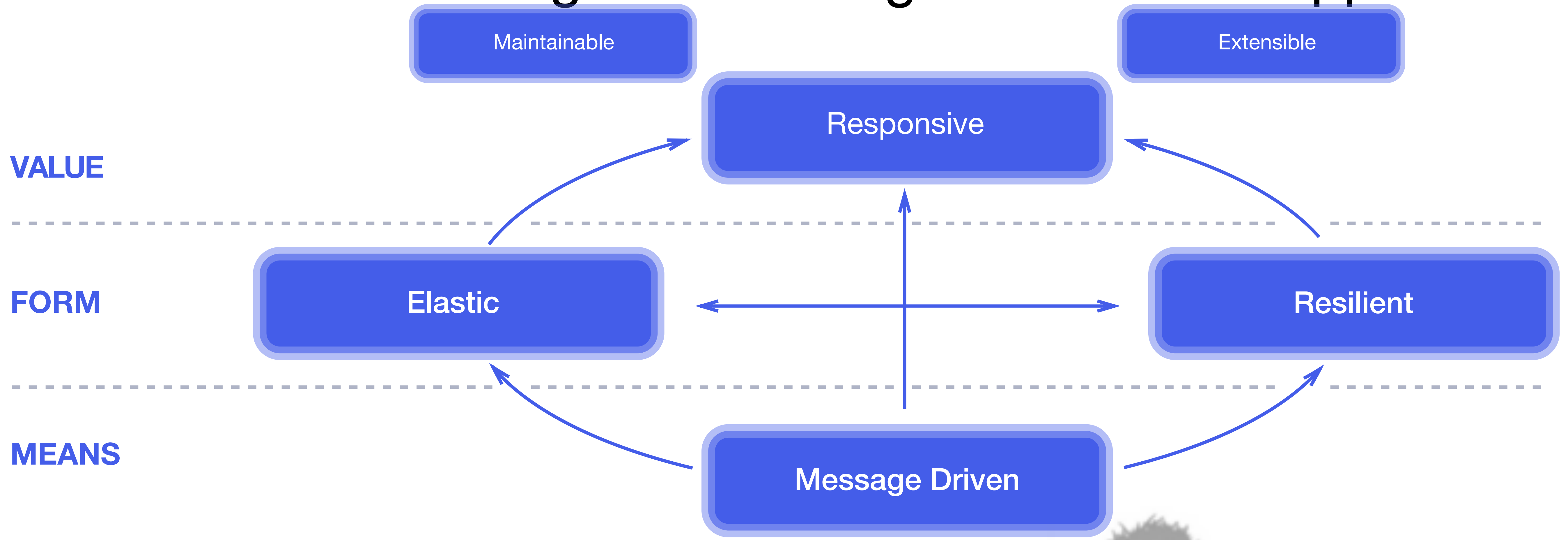
Spring Framework 6

Beginner to Guru

The Reactive Manifesto

Introduction to The Reactive Manifesto

- What is the Reactive Manifesto?
  - A set of principles for building responsive, resilient, elastic, and message-driven systems.
  - Aimed at addressing the challenges of modern application development.



History and Context of the Reactive Manifesto

- **Origins:**
  - Created in 2013 by a group of industry experts.
  - Response to the evolving needs of software systems.
  - Signed by 38,000+ people
- **Key Contributors:**
  - Jonas Bonér, Dave Farley, Roland Kuhn, and Martin Thompson.
- **Purpose:**
  - To establish a common vocabulary and set of design principles.
  - Address challenges in scalability, fault tolerance, and responsiveness.

Core Principles of the Reactive Manifesto

- **Responsive:**
  - Systems should respond in a timely manner.
  - Ensures consistent and quick interactions.
- **Resilient:**
  - Systems should stay responsive in the face of failure.
  - Use techniques like replication and isolation to handle faults gracefully.

Core Principles of the Reactive Manifesto

- **Elastic:**
  - Systems should remain responsive under varying workloads.
  - Ability to scale up or down efficiently based on demand.
- **Message-Driven:**
  - Systems should use asynchronous message-passing.
  - Ensures loose coupling and enables isolation.

Reactive Manifesto in Cloud-Native Applications

- **Cloud-Native:**
  - Applications designed to fully exploit cloud environments.
  - Reactive principles align with the dynamic and distributed nature of cloud infrastructure.
- **Microservices Architecture:**
  - Encourages small, independently deployable services.
  - Reactive principles facilitate communication and coordination between microservices.

Reactive Manifesto in Cloud-Native Applications

- **Scalability and Elasticity:**
  - Ensures applications can efficiently scale in and out.
  - Essential for managing cloud resources and costs effectively.

Benefits of Reactive Cloud-Native Applications

- **Performance:**
  - Improved utilization of system resources.
  - Handles large numbers of concurrent users effectively.
- **Resilience:**
  - Systems handle partial failures gracefully.
  - Enhanced fault tolerance and recovery mechanisms.

Benefits of Reactive Cloud-Native Applications

- **Scalability:**
  - Seamless scaling with demand.
  - Efficient use of cloud resources, reducing costs.
- **User Experience:**
  - Consistent and responsive user interactions.
  - Reduced latency and improved throughput.

Examples and Case Studies

- **Netflix:**
  - Uses reactive principles for its streaming services.
  - Achieved high availability and responsiveness.
- **Twitter:**
  - Migrated to a reactive architecture.
  - Improved system scalability and reliability.

Getting Started with Reactive Principles

- **Understanding Reactive Principles:**
  - Study the Reactive Manifesto and its documentation.
  - Explore the core concepts of responsiveness, resilience, elasticity, and message-driven architecture.
- **Applying the Principles:**
  - Design systems with scalability and fault tolerance in mind.
  - Use asynchronous communication for loose coupling.

SPRING FRAMEWORK

GURU