Overview of Spring Boot Software Patterns

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

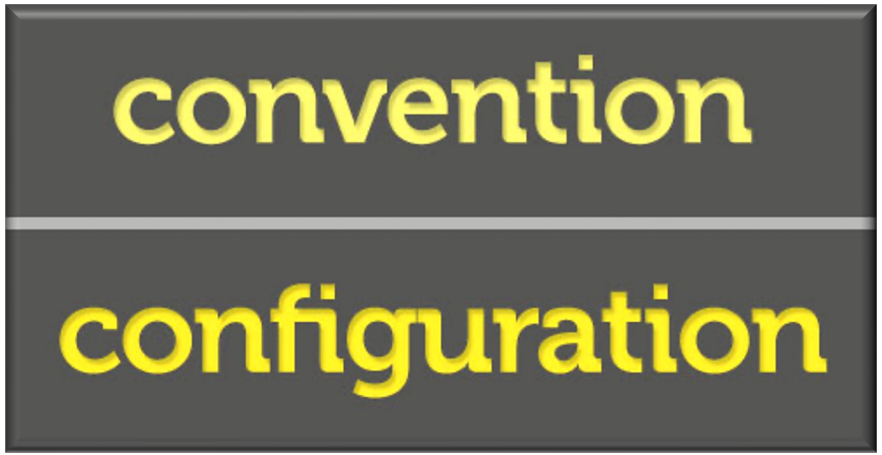
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Lesson

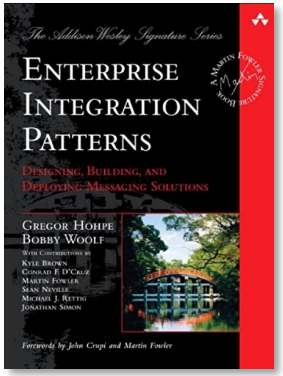
- Recognize Spring Boot's key design approach

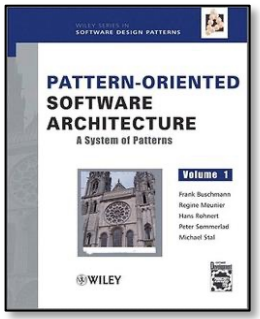


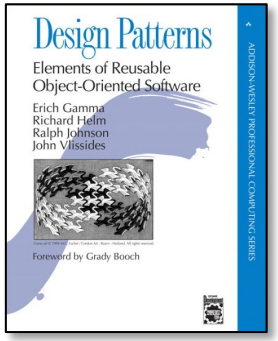
See en.wikipedia.org/wiki/Convention_over_configuration

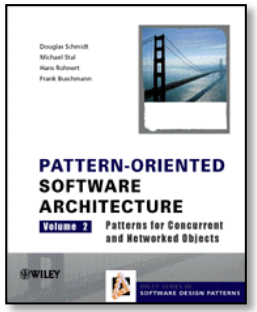
Learning Objectives in this Lesson

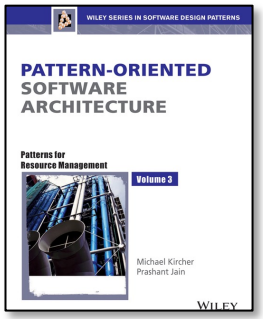
- Recognize Spring Boot's key design pattern
- Be aware of other patterns implemented by Spring Boot

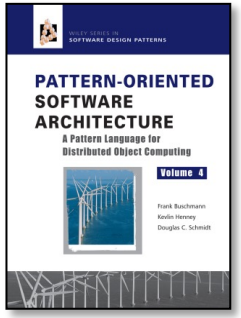










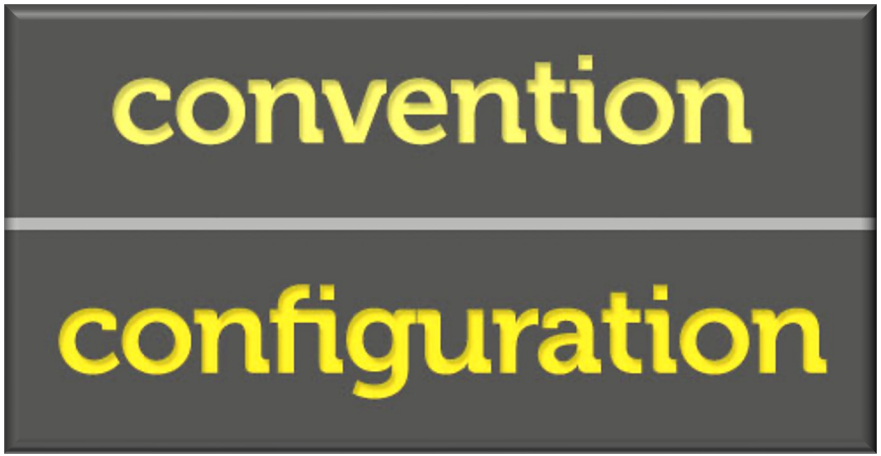


See www.dre.vanderbilt.edu/~schmidt/patterns-frameworks.html

Overview of Spring Boot's Design Approach

Overview of Spring Boot's Design Approach

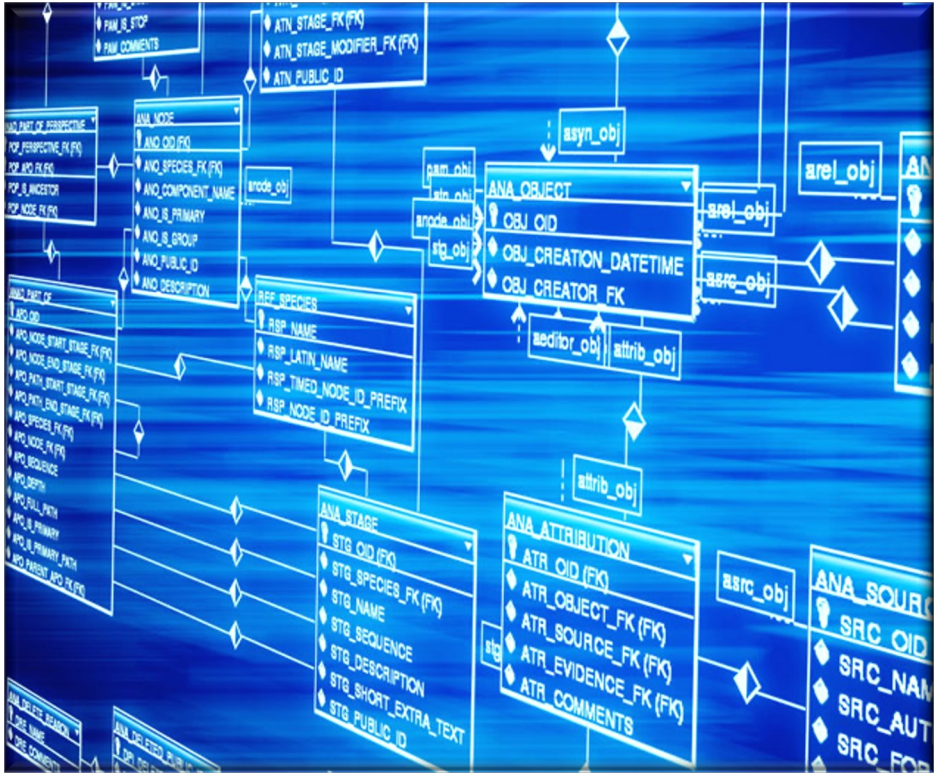
- Spring Boot applies the "Convention-over-configuration" software pattern



See en.wikipedia.org/wiki/Convention_over_configuration

Overview of Spring Boot's Design Approach

- Spring Boot applies the "Convention-over-configuration" software pattern
  - The goal is to create web apps by refining a general reusable "blueprint"



See www.ibm.com/docs/en/wasdtfe?topic=specification-osgi-blueprint-container

Overview of Spring Boot's Design Approach

- Spring Boot applies the "Convention-over-configuration" software pattern
  - The goal is to create web apps by refining a general reusable "blueprint"
  - Software frameworks use this pattern to decrease the number of decisions developers using the framework must make, without sacrificing flexibility

Overview of Spring Boot's Design Approach

- Reasonable defaults
  - e.g., if there is a class Sales in the model, the corresponding table in the database is called "sales" by default

Overview of Spring Boot's Design Approach

- Only specify the unconventional bits
  - e.g., if there's a deviate from conventions, it's necessary to write code regarding these divergent names
    - Such as calling a table "product sales" instead of "sales"

Overview of Spring Boot's Design Approach

- Eliminates distractions

Reasonable Defaults

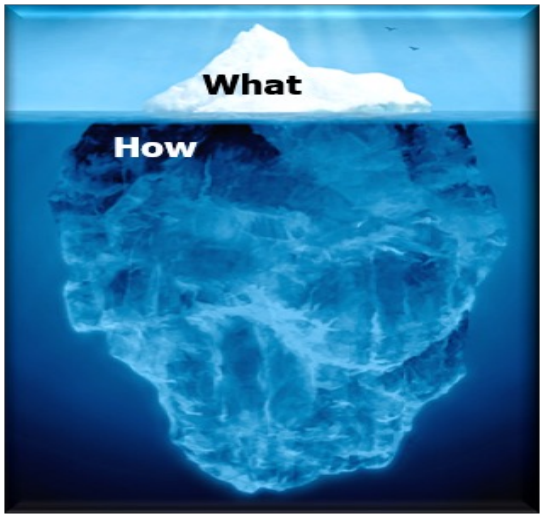
Only Specify the Unconventional Bits

Eliminates Distractions

Reduces the Number of Decisions You Have to Make

Overview of Spring Boot's Design Approach

- Eliminates distractions, e.g.,
  - There's no need to program low-level network details directly
    - Instead leverage declarative configuration mechanisms



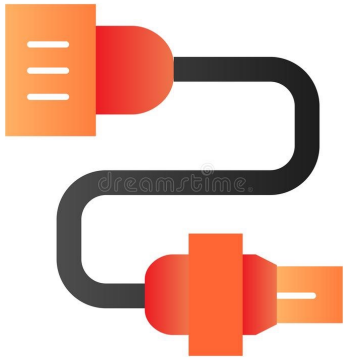
Overview of Spring Boot's Design Approach

- Eliminates distractions, e.g.,
  - There's no need to program low-level network details directly
  - Have the infrastructure manage the event loop(s) via IoC

See en.wikipedia.org/wiki/Inversion_of_control

Overview of Spring Boot's Design Approach

- Reduces the # of decisions you have to make
  - e.g., the auto-wiring of fields to their implementations is handled automatically

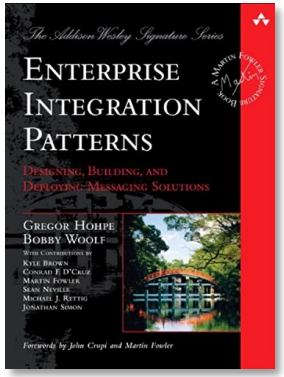


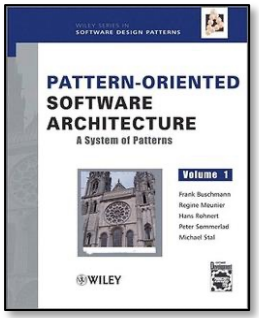
See www.baeldung.com/spring-autowire

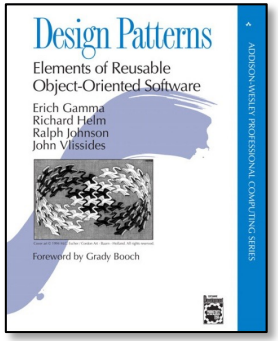
Overview of Spring Boot's Other Patterns

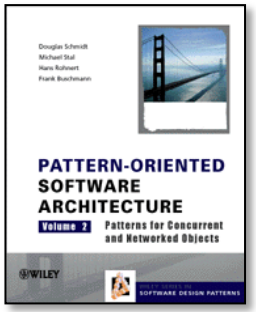
Overview of Spring Boot's Other Patterns

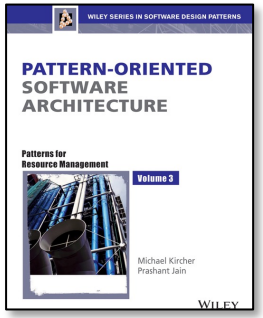
- Spring Boot also implements many other software patterns documented in the literature
  - e.g., Broker, Proxy, Factory Method, Resource Pool, Component Configurator, etc.

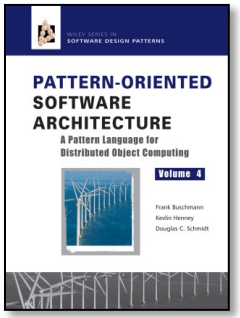












See www.dre.vanderbilt.edu/~schmidt/patterns-frameworks.html

End of Overview of Spring Boot Software Patterns