



Spring Boot Introduction

Introduction to Spring Boot

Objectives

After completing this lesson, you should be able to

- Explain what Spring Boot is and why it is opinionated
- Explain each major feature of Spring Boot and its value proposition

Agenda

■ Why Spring Boot?

■ Spring Boot Features

- Dependency mgmt
- Auto-Configuration
- Actuators & Health Metrics
- Packaging and Runtime
- Integration Testing

■ Summary





Why Spring Boot?

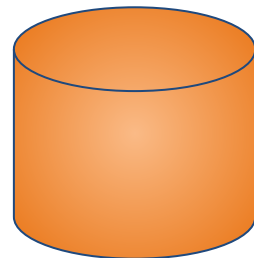
- Enable developers to build applications that are:
 - Easy to develop and test
 - Easy to manage dependencies
 - Easy to configure
 - Easy to manage and monitor in production
 - Cloud ready
- Keep developers focused on building business value, less on non-functional concerns

Spring's DRY principle applied to itself!

Consider a App that Integrates with a Database ...

- Requirement
 - Simple Spring application
 - Talking to a database
 - Running in production
- Assume basic domain logic exists
 - Need plumbing for the domain logic to talk to a Relational Database ...

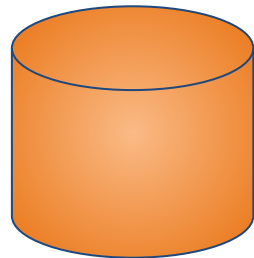
Requirement



Implementing Data Access *without* Spring Boot, You need



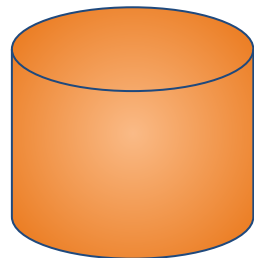
- A database for the app to talk to ...
- A JDBC Driver on the application classpath ...
- Configuration for JDBC driver to connect to the actual database ...
- Loading the JDBC Driver in the application ...
- Code for the data source to use the JDBC driver ...
- Code for the JdbcTemplate to use the DataSource ...
- Dependencies for other data access APIs added to project ...
 - Such as JPA, MyBatis or other mapping tools
- ... *Finally, we can access data in the application!*



Create same Application *with* Spring Boot



- *How can we improve the situation?*
- Provide a database for the app to talk to ...
 - Add appropriate dependency JAR
- Connect to the actual database ...
 - Define configuration properties for Boot to use
- Use the JdbcTemplate logic in your application
 - JdbcTemplate automatically created by Spring Boot



But what about Production Hardening?

- *How do we monitor for, or handle the database connection failures?*
- *What if we want to expose realtime business metrics to a production monitoring solution via our domain logic?*
- Solutions are likely hand-coded, or 3rd party solution integrated by hand.



Production Hardening with Spring Boot



- View status and configuration via JMX or REST
 - Spring Boot “*Actuators*”
- Solutions for Database health checks are included “out of the box”
- Spring Boot provides clean, standards-based API for custom metrics
 - *Project Micrometer*



<http://micrometer.io>

Spring Boot vs. Spring

- For simple Spring applications, we can reduce the number of “plumbing” steps by nearly 70% by using Spring Boot!



Agenda

■ Why Spring Boot?

■ Spring Boot Features

- **Dependency Management**
- Auto-Configuration
- Actuators & Health Metrics
- Packaging and Runtime
- Integration Testing

■ Summary



Spring Initializr - What is it?

- Framework, API, and default implementation to generate initial Spring Boot application projects
- Spring's public web-site: <http://start.spring.io>
- Or build your own: <https://github.com/spring-io/initializr>

Spring Initializr - What is its value?

- Simplify and curate dependency management
 - Gradle or Maven supported
 - Java, Groovy or Kotlin
 - Builds Spring Boot projects for you
- Enable oversight of application dependencies
- Reduced “Dependency Hell”
- Accessible as a “New Project” wizard in STS/Eclipse, IntelliJ IDEs

Spring Initializr Web Page

<http://start.spring.io>

SPRING INITIALIZR

The screenshot shows the Spring Initializr web page with the following configuration:

- Project:** Maven Project
- Language:** Java
- Spring Boot:** 2.1.5
- Project Metadata:**
 - Group: com.example
 - Artifact: demo
 - Options: (expanded)
- Dependencies:**
 - Search dependencies to add: Actuator
 - Selected dependencies: Spring Web Starter

At the bottom, there is a green button labeled "Generate the project".

Switch to full version: more options, explicit check-list of dependencies

Specify dependencies

Starter - What is it?

- Bill-of-Materials (BOM) of Spring and 3rd party dependencies
- Leverages Maven Dependency Management
- Support for Gradle also
 - Custom Spring Boot plugin needed



Starter - What is its value?

- Simplify and curate dependency management
- Use of standard build dependency mechanism makes adoption and customization easier



Agenda

■ Why Spring Boot?

■ Spring Boot Features

- Dependency Management
- **Auto-Configuration**
- Actuators & Health Metrics
- Packaging and Runtime
- Integration Testing

■ Summary



Auto-Configuration - What is it?

- Mechanism to detect dependencies through classpath, beans, and config properties
- Create Spring Bean configuration on developer's behalf based on reasonable default assumptions during start time.
- Existing auto-configuration rules may be overridden or disabled
- New auto-configuration may be built to support new backing resources or features.



Auto-Configuration - What is its value?

- Reduce burden on developers for common Spring bean configuration tasks



Agenda

■ Why Spring Boot?

■ Spring Boot Features

- Dependency Management
- Auto-Configuration
- **Actuators & Health Metrics**
- Packaging and Runtime
- Integration Testing

■ Summary



Actuator - What is it?

- Framework to enable monitoring and management of Spring Boot applications
- Provides clean API exposing telemetry to common 3rd party monitoring tools
- Provides standard method of HTTP health indicators



Actuator - What is its value?

- Reduce burden on developers for non-functional efforts
- Reduce integration efforts between application and monitoring/management tools



Agenda

- Why Spring Boot?
- **Spring Boot Features**
 - Dependency Management
 - Auto-Configuration
 - Actuators & Health Metrics
 - **Packaging and Runtime**
 - Integration Testing
- Summary



Packaging and Runtime - What is it?

- Spring Boot provides tools for building application deployment artifacts such as jar, war or launch scripts
 - As part of Maven “package” goal or Gradle “assemble” task
- Spring Boot allows embedding of web runtime containers, such as Tomcat, Jetty or Undertow



Packaging and Runtime - What is its value?

- Reduce burden on developers for non-functional efforts
- Reduce need for Middleware



Agenda

■ Why Spring Boot?

■ Spring Boot Features

- Dependency Management
- Auto-Configuration
- Actuators & Health Metrics
- Packaging and Runtime
- **Integration Testing**

■ Summary



Integration Testing Support - What is it?

- Spring Boot test tools that provide Spring application environment bootstrap for JUnit component integration tests



designed by freepik.com

Integration Testing Support - What is its value?

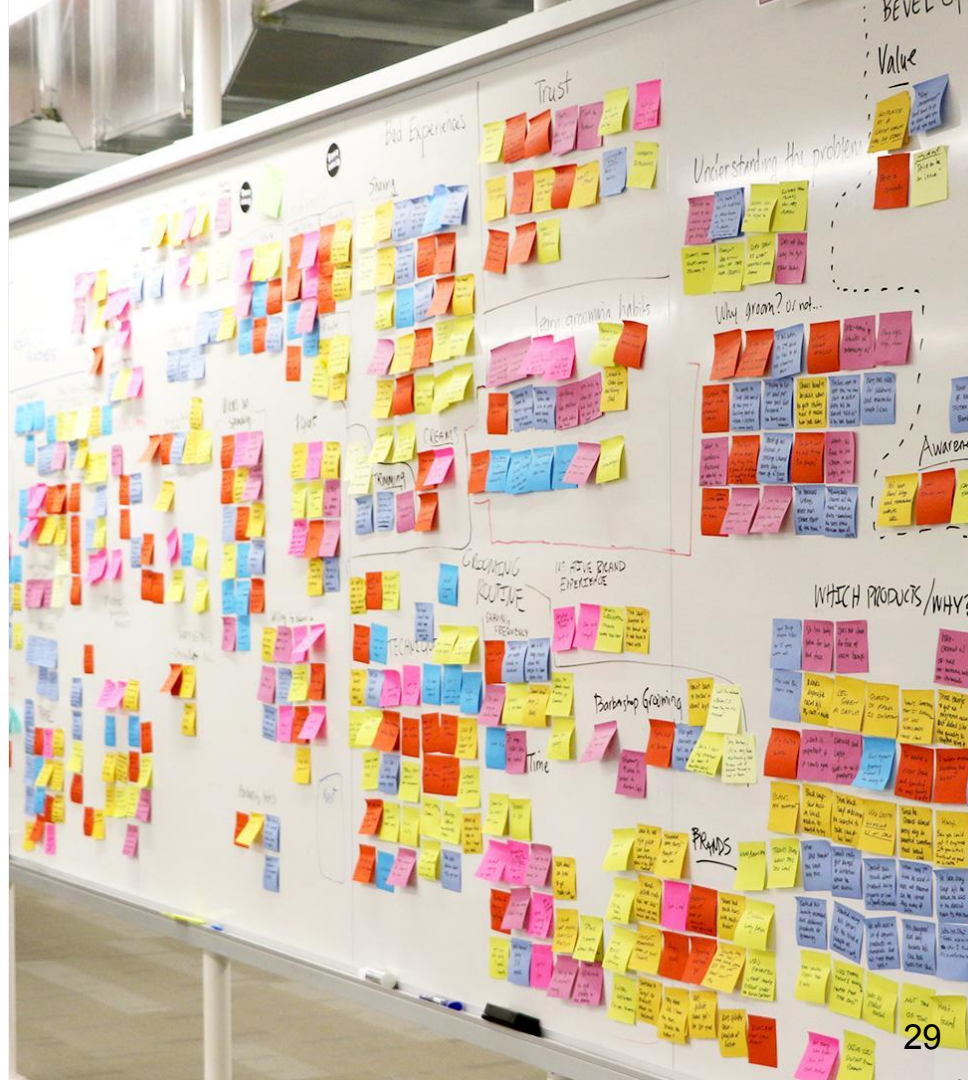
- Reduce burden on developers for writing test plumbing code



designed by Freepik.com

What we Covered

- Why Spring Boot?
- Spring Boot Features
 - Dependency Management
 - Auto-Configuration
 - Actuators & Health Metrics
 - Packaging and Runtime
 - Integration Testing
- Summary



Spring Boot in Summary

- An opinionated runtime for Spring Projects
- Supports different project types like Web and Batch
- Handles most low-level, predictable setup for you
- *It is NOT*
 - A code generator
 - An IDE plug-in



See: [Spring Boot Reference](http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle)

<http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle>

Opinionated Runtime

- Uses sensible defaults, “*opinions*”, mostly based on the classpath contents
 - Sets up a JPA Entity Manager Factory if a JPA implementation is on the classpath
 - Creates a `JdbcTemplate` if `spring-jdbc.jar` is on the classpath
- Everything can be overridden easily but often not necessary

Tying Things Together...

- Spring Boot provides an application holistic plumbing support for backing resources and features:
 - Databases
 - Caching
 - Integration Middleware
 - Security
 - Fault Tolerance features
 - Running on Cloud/Cloud Native infrastructures
 - Template engines
 - Web containers
 - External configuration

A man with a beard and a woman are sitting at a desk, looking at a computer monitor. The man is pointing at the screen while the woman looks on. The image is overlaid with a dark blue tint and two horizontal teal lines.

Lab: Spring Boot Intro

Lab project:
No starting lab provided

Anticipated Lab time:
30 Minutes