SPRING BOOT QUICK START

Goals of Spring

* Light weight development with java POJOs (plain-old-java-objects).
* Dependency injection to support loose coupling.
* Minimize boilerplate java code.

Core Containers - manages how beans are created. reconfig files for setting properties and dependencies. Spring expression Language used to refer to other beans. spring container that holds the beans in memory.

* Beans
* Core
* SpEL
* Context

Infrastructure – Aspect Oriented Programming. Adds functionality to objects declaratively. Allows for the building of application wide services like Logging, security, transactions, etc.

* AOP
* Aspects
* Instrumentation
* Messaging

Data Access Layer – Communicating with the database, relational or NoSQL. The JDBC helper classes make it much easier to access a database using JDBC which allows the reduction of code by over 50%. ORM is Object to Relational Mapping which basically allows you to hook into hibernate or JPA. Java Messaging Service (JMS) allows you to send messages to a message queue asynchronously which is a core part of Java EE so JMS provides helper classes that allow you to make use of JMS which can reduce your code by over 50%. The Spring Transaction Manager allows make use of transactions on methods, database calls, and more due to its flexibility w/ transactions.

* JDBC
* ORM
* Transactions
* JMS
* OXM

Web Layer – Home of the Spring MVC framework. Used for building web apps using spring core and making use of spring controllers and spring view.

* Servlet
* WebSocket
* Web

Test Layer – Spring has support for test driven development.

* Unit
* Integration
* Mock

Spring Projects – Additional modules built on top of the core spring framework or addons.

* Spring Cloud, Spring data
* Spring batch, Spring security
* Spring web services, Spring LDAP

Maven – Used to assist with build management and dependencies.

* When you generate projects using spring initializr it can generate a Maven project for you. Maven is a project management tool for your app.
* The most popular use for maven is build management and dependencies.
* Maven is like NPM?
* Maven is like Door Dash for Spring Boot. You make a list of what you need, and maven will retrieve it for you.
* Allows programmers to build and run a project with minimal local configuration.

How does Maven work?

* Maven will read your project configuration file aka your shopping list.
* Maven will check a Maven Local Repository on your computer.
* Maven will check remote repo if local repo is not available on your machine.
* Files will be saved in your local repo.
* Maven will use files to build and run your application.
* Based on your config file Maven will handles class/build paths for you.

Maven key concepts

* POM file (pom.xml) – Project Object Model file which is your config file for your project aka your shopping list for dependencies. This file is always located in your root. The POM file consists of:
  + Project meta data – name of project, version numbers being used, output file type etc.
  + Dependencies – List of projects that your application depends on like Spring, Hibernate, JSON, etc.
  + Plug-ins – additional custom tasks to run.
  + Includes information that you entered at Spring Initializr website.
* Project coordinates – Uniquely identify a project.
  + Should contain a:
    - Group ID – name of company, group, or organization
    - Artifact ID – name of project
    - Version – release version
  + We can add more dependencies in the dependencies section, and this should include the projects:
    - Group ID, Artifact ID, and Version (GAV).

Spring Boot Project Files

* Mvnw – a maven wrapper file that allows you to run a maven project. If the incorrect version of maven is not installed it will be installed automatically.
  + Mvnw.cmd for windows
  + Mvnw.sh for linux/mac
* Application Properties
  + By default, sprint boot will load properties from application.properties
  + Read data from application.properties
* Static Content
  + By default, spring boot will load static resources from “/static” directory.
  + Static resources are often html files, CSS, JavaScript, images, etc.

Spring Boot Starters

* The Problem - Building a spring application is complex.
* Developers should not have to search for each dependency.
* The Solution – spring boot starters – a curated list or collection of maven dependencies. Makes it much easier for programmers to get started with provided dependencies based on what the programmer wants to build.
* Available on initializr dependencies. Choose what you need and when you generate the project the dependencies will be automatically added to your pom.xml file.
* Types of spring boot starters:
  + spring-boot-starter-web is for building web apps, includes validation, REST.
  + spring-boot-starter-security is for adding spring security support.
  + spring-boot-starter-data-jpa adds spring database support with JPA and Hibernate
  + and of course, there’s more!

Spring Boot Starter Parent

* A special starter that provides maven defaults.
* Defined in the starter parent.

Spring Boot Dev Tools

* The problem – when running spring boot applications if you make changes to your source code you have to manually restart your application.
* The solution – spring boot dev tools – automatically restart your application when your code is updated sort of like mount or react app etc. but for java.
* Just add the dependency to your POM file. No additional code needed.
* Only needs the Group ID and Artifact ID.

Spring Boot Actuator

* Problem – How can I monitor, manage, and check the status or metrics of my application?
* Solution – spring-boot-starter-actuator – exposes endpoints that allow you to monitor and manage your application.
* DevOps functionality out of the box by adding the dependency to the POM file and the REST end points will be automatically added to your application.
* /health will provide the programmer with application health information by checking the status of the app
* /info will provide the programmer more information about your app
* /auditevents will provide the programmer with audit events for your app
* /beans list all beans registered in the spring app context
* /mappings list of all @RequestMapping events

Spring Boot Actuator Security (Endpoints)

* Add spring-boot-starter-security to application for endpoint security to the pom.xml file.
* Once you add the dependency spring security will prompt you for a login
  + Username: user
  + Password: check the console for the password
* These given username and passwords can be over ridden by editing the application.properties file.
  + spring.security.user.name=jerrynewusername
  + spring.security.user.password=jerrysnewpassword
* We can also exclude endpoints as well like:
  + Management.endpoints.web.exposure.exclude=health,info

Running Spring Boot apps from the command line

* When running from the command line theres no need for an IDE
* The server will be embedded into our JAR file
* No need to have separate server installed/running (Tomcat)
* Spring boot apps are self-contained
* To run the app we can use:
  + java -jar nameofapp.jar
  + mvnw spring-boot:run

Injecting Custom Application Properties

* Problem – your app needs to be configurable w/ no hard coding of values and you need to read app configuration from a properties file.
* Solution – Spring Boot reads information from a standard properties file located @ src/main/resources/application.properties
* The programmer can define ANY custom properties in this file.
* Your spring boot app can access properties using @Value.

Spring Boot Server Properties

* Can be config’d in the application.properties file
* Properties that can be set:
  + Server port
  + Context path
  + Actuator
  + Security
  + And more…
* Properties are roughly grouped into the following categories:
  + Core – Spring Boot Logging configurations
  + Web – Port listening setup, context path (URL customization), and session timeout
  + Security – secure end points
  + Data – database properties for logging into database
  + Actuator – Endpoints customization
  + Integration
  + DevTools
  + Testing