Spring Review

* Spring is a backend framework for building enterprise Java Applications.
* Spring makes HEAVY use of the reflection API.
  + A java library that can be used to write Java code that creates Java code and can read Java code. (very meta)
  + Your code/classes doesn’t really call each other.
  + The annotations are read by the Spring IoC to generate beans that get stored in the container.
* Spring is Modular by design.
  + Spring CORE (Mandatory essential part of spring)
    - IoC container
  + Spring WEB
    - API Layer Controllers
  + Spring DATA
    - Persistence layer Repos
  + Spring AOP
    - AOP Aspects
  + Spring ACUATOR
    - Helpful endpoints for checking on your application.
* **SPRING BOOT IS NOT A MODULE OF SPRING**

Spring Core

* The essential main module in any Spring Application.
* Dependency Injection
  + Process by which a dependency in a class is fulfilled by an “injector”, an outside class choosing that dependency.
  + Your dependencies are injected by a Spring Injector
    - DO NOT DO BookRepo repo = new BookRepoImpl()
  + @Autowired
    - Hey please inject a dependency here
  + Types of DI
    - Constructor Injection
    - Setter Injection
    - Field Injection
      * Uses Java Reflection to inject the dependency.
* Spring beans
  + **Object** that will be **managed by Spring**.
  + Created and destroyed by Spring.
  + Spring will automatically inject beans into other beans.
  + Beans have scopes.
    - Singleton
      * Only one instance of the bean
    - Prototype
      * 0 to many instances
    - Request
      * 1 per http request
    - Portlet
      * I do not know how to use.
* IoC container
  + Inversion of Control container.
  + All beans live inside this container.
  + This is how Spring is able to keep track of beans.
    - Knows how to inject beans.
    - How to create beans.
    - How many instances there are.
    - When to destroy beans
  + In newer version of spring the IoC container is called the Application Context. (BeanFactory Old)
    - ApplicationContextFailedToInitialize exception.
      * Spring tried to set up the container and configure all the beans but failed.
        + Autowired a bean that does not exist for example

No Dependency of type x

* + Controllers, repos, services they all become beans in the container.
    - Singleton beans specifically.
* Components
  + @Component
  + Class annotation used to tell Spring
    - “Hey look at this class and use it as a blueprint for a bean”
  + Stereotypes
    - @Service
    - @Repostiory
    - @Controller, @RestController
    - @Config
      * Create methods with @Bean to create beans for the IoC that are not necessarily a class.

Framework vs Library

* Library
  + Your code calls some other code written elsewhere.
* Framework
  + The framework reads/incorporates your code into a specific that framework application.
    - Spring reads your components and creates a Spring IoC container based on those components.
    - Angular reads your components and services and ultimately stiches them together into a finished Angular application.
* If you have to conform to a specific way of approaching a problem you are in a framework.

Spring Data

* Persistence module
* Annotations and what they do
  + @Repostiory
  + CrudRepository
  + Abstract query methods
    - findByName(String name)

Spring AOP

* Aspect Oriented Programming.
  + Paradigm of programming like Function or OOP
* Cross cutting concerns.
  + We build applications in stacks where each layer addresses a particular concern.
    - Persistence layer : Database interaction
    - Service layer : Business Logic
    - API layer: How to interact with the software
  + Some things cannot fit neatly into a single layer.
  + Cross cutting concerns
    - Security
    - Logging
* Aspects
  + Classes that are designed to address a single CCC.
* Advice (Advice Methods)
  + Methods in an aspect that ‘advise’ how another method is executed.
  + Types of advice
    - @Before
    - @After
    - @AfterReturning
    - @AfterThrowing
    - @Around
      * Most powerful type. You can edit the input parameters and the output.
* JoinPoint
  + Any place in an application where an advice method can be used.
    - In Spring all method calls are JoinPoints.
* Pointcut
  + Regular expression which tells an advice what joinpoints it should advise.

Spring Web

* API module of Spring
* Annotations
  + @Controller
  + @RestController
  + @ResponseBody
  + @PathVariable
  + @RequestBody
  + @GetMapping
  + @QueryParam
* Front controller design under the hood.
  + All requests are sent to a Dispatcher Servlet bean.
  + That servlet looks and the Handler Mapping bean and matches to URI to a controller.
  + Forward that request to the appropriate controller.

END OF MODULES

Spring Boot (NOT A MODULE)

* Spring when it first came out was very helpful however there was a lot of repetitive information that was the same in every single project.
  + A lot of configuration beans you would define in your code
* Spring Boot will create a vast majority of those tedious configuration beans.
  + Appliccation.properties that contains all the actual configuration information.
  + Spring boot will read this file to set up some of those beans.
* Spring Boot has an embedded TomcatServer.
* Spring Boot is a pre-configured, highly opinionated quick start up version of Spring.