Spring Framework

* Spring is a lightweight Java Framework for making enterprise level applications
  + Created by Rod Johnson
* The Java backend framework
  + EJB Entity Java Beans existed before Spring
  + They were created by Oracle to make enterprise web applications
    - EJBs were awful, they were also proprietary
* Build around this concept of the IoC (Inversion of Control) container
  + ApplicationContext (newer version)
  + BeanFactory (Older version)
* Dependency injection is vital and the key to making your Spring Application work
  + Dependency injection is creating an object by passing in the dependencies it needs to runs
* Spring will manage all the dependency injection for you
  + Abstracts away you explicitly having to inject dependencies
  + Abstracts away managing dependencies
    - For example you can tell spring to make certain classes singletons without having to explicitly write that code
* Spring Modules
  + Spring is modular by design
    - A module is a dependency that contains a lot of functionality for a particular part of your application
  + Main/Popular modules
    - Spring CORE
      * Essential module must be in a Spring
      * Is the IoC container (Application context)
    - Spring DATA
      * Responsible for creating a persistence layer
      * Performing CRUD operations with the database
    - Spring WEB aka Spring MVC
      * Responsible for creating an API layer
      * Creates controllers and handlers with route mappings
    - Spring AOP
      * Responsible for Aspect Oriented Programming in Spring
    - Sprint Actuator
      * Developer tools modules useful for while you develop/monitoring and application
      * Usually not included in final builds that end users would use
* Spring Boot
  + Spring Boot is not Spring Core
  + Spring Boot is an opinionated template for creating backend web application in Java
  + It has an embedded tomcat server to handle http requests
    - Javalin also uses an embedded Jetty server
* Spring Boot works heavily on annotations
  + The line of code in main SpringApplication.run(HelloWorldSpringApplication.**class**, args);
  + Creates an IoC container
  + You will then create classes label them with annotations
  + That line of code will read through all of your classes create beans out of them and set up a web application for you

Spring AOP

* AOP (Aspect Oriented Programming)
  + Paradigm of programming much like OOP or Functional
* Program ‘Aspects’ of your application
* Terminology
  + Cross Cutting Concerns (CCCs)
    - Some feature of your application that cuts across multiple levels of your application
      * Examples, Security and logging
  + Aspect
    - A class that addresses one of the cross-cutting concerns
  + Advice
    - An Advice is a method that executes when a certain part of your application executes
    - These methods ‘advise’ some other part of your code
      * Types of advice
        + @Before advice method executes before
        + @After advice method executes after whatever method you are advising
        + @Around the advice method completely encapsulates the advised method (most powerful type of advice)
        + @AfterReturning advice method only executes when the method returns successfully
        + @AfterThrowing advice method only executes if the advised method throws an uncaught exception
  + Join-Point
    - Any location in your application that can be advised
      * In Spring this is any method
  + Point-Cut Expression
    - Regular expression that selects certain join points to advise
* Pros
  + Separate and decouple your code
    - Code related to logging or security does not have to go into your controller/services
  + Refactoring your aspects is much simpler
    - Ex If the security requirements change the only alterations have to be made in the security aspect class
* Cons
  + AOP is less efficient than coding directly
    - The process of intercepting method calls slows down the application slight
  + Aspects can affect your code without realizing it which can make following code execution and bugs INCREDIBLY HARD to find with a lot of aspects