* Spring
  + What is Spring
    - Java framework for building enterprise grade software quickly and easily.
    - Spring is made out of many different modules and besides the core four we can mix and match how we choose
    - Not tied into web, but great web modules
    - Reliable, maintainable, loosely coupled, separation of concerns, interfaces
  + What is IoC
    - Inversion of control - we are going to give up control to spring framework and let it control the program.
    - Let Spring control the creation and lifecycle of all of our objects and the linking of our objects together
  + What is Dependency Injection
    - This is how spring can take the objects that it is managing and piece them together based on who needs what
    - 3 types Constructor, Setter, field
    - In java there is also interface DI but spring doesn’t support it
    - Constructor - we use this for dependencies that are absolutely required
    - Setter - we use for dependencies that are optional
    - Field - we don’t use because it makes it very hard to test our code ( uses reflections to directly get the Field object and then set it to the dependency object)
  + How are those two related
    - We can achieve IoC by using Dependency Injection
  + What are benefits, downsides
    - Benefits
      * Separate our concerns
      * Loosely coupled code
      * Code reusability/avoid code duplication
      * Work with Contracts(interfaces) instead of implementation
    - Downsides
      * Introduce Complexity
      * Errors, become much harder to trace
  + IoC container in Spring?
    - BeanFactory
      * Interface
      * it can be built from a couple different configurations including xml and annotations
      * Older, it lazily instantiates beans
    - ApplicationContext
      * Interface
      * It can also be built using xml and annotations
      * It gives default support for I18N ( internationalization)
      * Automatically register Bean Post Processors
      * Eager instantiate Singleton beans
  + What is a spring bean? (relation to java bean?)
    - Object that we have given to Spring for it to manage
    - They have different scopes and a lifecycle that spring puts beans through
  + Modules of spring?
    - Core, beans, Context, Web, ORM, AOP, Transactions
  + How to configure spring
    - XML configurations - beans.xml file that defines all of the beans and their relationships
    - Java configuration - @Configuration Annotation, then in that class we can define beans (@Bean) that are used for configuring spring
    - Annotation Configuration - we annotate classes using things like the stereotype annotations to define them as beans and their purpose
  + Stereotype annotations?
    - @Component - This class is a bean
    - @Controller
    - @RestController - This class is a bean and it will be used with Spring MVC
    - @Service - This is a bean, but its also going to be a service
    - @Repository - This is bean, it interacts with the ORM and Transaction modules
  + Spring Bean Scopes
    - In Spring there are 2 scopes
      * Singleton
        + Only make one instance of the bean ( default scope)
      * Prototype
        + Every time a class needs this bean as a dependency, make a new copy
    - In Spring Web we add 3 more scopes
      * Request
        + Every time we get an http request, make a new copy of the bean
      * Session
        + For each http session (unique connection between client and server) make a new copy of the bean
      * Global Session
        + This has to do with portlets, Don’t worry about it
  + What is wiring? Autowiring
    - Wiring is the process of defining what beans have dependencies on which other beans - connecting beans together
    - AutoWiring - we let spring calculate (using type) how to piece the different beans together
    - @Autowired annotation to do autowiring - tells spring to automagically connect our beans
    - If there are multiple beans of the same type, spring will give you an error
  + Spring bean lifecycle
    - Instantiation - Basically Spring uses the new keyword and calls a constructor
    - Populate Properties - Spring will do any autowiring that it needs to do
    - setBeanNameAware
    - setBeanFactoryAware
    - setApplicationContextAware - They have a method that gets called with the data the represent, so this we can set a field/use that information in the bean
    - BeanPostProcessor - this is a special bean that has 2 different methods
      * Every single bean we make will go through the post processor’s methods
      * beanInitialization()
    - initializingBeans(Interface) - afterPropertiesSet()
    - Custom Init Method - @PostConstruct anno
    - BeanPostProcessor - afterInitialization()
    - Ready to use the bean.
    - Shut Down Container
    - disposableBeans(Interface) destroy()
    - Custom destroy - @PreDestroy annotation
* Spring Boot
  + What is it?
    - Opinionated Configuration of a spring project
    - Many people were doing a lot of the same base configuration for their spring projects and the spring boot team said why don’t we make a project that does that from the get go
    - Convention over configuration ( you want to follow the conventions that people do and not worry about excessive configuration )
    - 1 XML is the worst, we want to use annotations and classpath scans only
    - 2 you want to use Tomcat for your webserver ( Tomcat is a web container and it is made for use with servlets - how to do http req res in java)
      * Embedded Tomcat Server
    - 3 If it is on the classpath you want to use ( do config based on dependencies in the classpath )
    - 4 You always are going to use an ApplicationContext not a beanfactory
    - 5 for any config you do want to do, you need to use YAML (Yet Another Markup Language )
  + How does it relate to spring
    - It is still a spring project iot just handles how we want that project to be configured
  + Is it a module?
    - No It is a spring project
  + What is tomcat?
    - Web container that holds servlets
    - It can process http requests into object and objects into http responses
    - It keeps track of sessions
  + How does it change configuration
    - Annotations instead of xml
    - Classpath
* Spring MVC
  + Is this a module of spring?
    - Not Technically, it is a major part of Spring Web which is a module of Spring
  + What is Java EE?
    - Java Enterprise Edition this is the standard for doing web services in Java before Spring
    - It utilizes the Servlet Framework along side a web container for handling http requests and responses
    - Spring MVC uses Servlets and Tomcat to handle its requests and responses
  + What is a servlet
    - Is a special object that’s purpose it to handle Request and Response objects and relay information to the web container
    - The web container then takes those objects and turns them to actual requests and responses
    - Has methods like doGet and doPost
  + How does mvc manage servlets
    - MVC makes exactly one servlet that handles every single request.
    - That servlet basically has a lookup switch for matching http verbs and paths to another method
    - That other method will then actually process the request
  + Design pattern?
    - This is called the Front Controller pattern
    - Uses a dispatcher Servlet, to dispatch request verb and path to an appropriate method
  + What is a controller? A rest controller?
    - Two different ways of sending back data to spring mvc
    - Rest Controller does not send a view, just a model
  + Spring MVC annotations
    - @Controller
    - @RestController - implies @Controller with @ResponseBody above every single method
    - @RequestMapping
    - @GetMapping
    - @PostMapping
    - …
    - @RequestBody - to get data from the body of a request
    - @ResponseBody - say we are not sending a view, just model data
    - @PathVariable - allow us to get data from the path of a request /users/1
    - @RequestParam - things in the path that have a ? ie: ?name=Gerard
* Spring Data
  + Is this a module of spring?
    - No it is a spring project
  + How does it relate to JPA and Hibernate
    - Spring Data utilices JPA entities under the hood for it to generate its dao implementations
  + How to use it
    - We make an interface and have that interface extends JpaRepository<E, T>
    - E a jpa annotated entity
    - T type of that entity’s primary Key (mostly just Integer)
    - That’s it
    - We now have a completely function DAO just by making an interface
  + Default Methods from JpaRepository
    - getOne(id)
    - findById(id) - special optional object which can be a value or null and has a method to check
    - findAll()
    - save(E obj)// to do updates as well
    - Delete( E obj)
    - flush() - force changes to DB
  + How can I write queries using it
    - We can write methods in the interface and based on the names of those methods, spring data will generate implementation for a new query
    - findByUsernameAndPassword(String username, String password)
  + Can I use HQL? SQL?
    - We can put an @query annotation above a method and spring data will use the string in the annotation instead of the method for generating impl
    - We can use sql if @Query(“sql string”, nativeQuery = true)
  + Annotations?
    - @Query