1. Introduction to Spring Boot
   1. Goals
      1. Enable building production ready application quickly
      2. Provide common non-functional features
         1. Embedded servers
         2. Metrics
         3. Health checks
         4. Externalized configuration
   2. Spring boot is neither an application nor a web server
   3. Features
      1. Quick starter projects with auto configuration
         1. Web
         2. JPA
      2. Embedded Servers – Tomcat, Jetty or Undertow
      3. Production-ready features
         1. Metrics
         2. Health checks
         3. Externalized configuration
2. Using Spring Initialize to create a project
3. Creating a simple REST controller
   1. @RestController
   2. @GetMapping
4. Spring Boot Auto Configuration
   1. @SpringBootApplication annotation at SpringBootApplication.java indicates that this is a Spring context file. It enables something called auto configuration.
   2. It enables something called component scan. Component scan is one of the important features of Spring where it would start automatically scanning these classes in this package. And this sub package for any beans.
   3. We have added in an annotation at bookController. This is one of the annotations which is scanned for it, this would be registered as a component. So, the bookController would be registered as a bean and it would be managed by the Spring framework.
   4. @SpringBootApplication = @SpringBootConfiguration + @EnableAutoConfiguration + @ComponentScan
   5. @SpringBootConfiguration: It indicates that this is a Spring context.
   6. @EnableAutoConfiguration: this enables auto configuration.
   7. @ComponentScan: it enables automatic scan of this specific package.
   8. SpringApplication.run() method is used to run a Spring context. So, we are giving a Spring context as an input to it, and it would be able to run that. The run method also returns application context.
   9. So, Spring Boot looks at
      1. the frameworks which are available on the classpath and
      2. It looks at the existing beans which are configured for the application. And based on that, it provides the configuration needed.
   10. Typically, all Auto Configuration classes look at other classes available in the class path (or dependency). If specific classes are available in the class path (or dependency), then configuration for that functionality is enabled through auto configuration. Annotations like **@ConditionalOnClass**, **@ConditionalOnMissingBean**, **@ConditionalOnBean** and **@ConditionalOnProperty** help in providing these features!



* + 1. @ConditionalOnBean: This bean is configured only if there is same bean configured with the same name.
    2. @ConditionalOnProperty: This bean is configured only if property is enable in application properties file.
  1. Enable log for auto configuration in application properties file
     1. **logging.level.org.springframework**=debug
  2. Spring Boot vs Spring vs Spring MVC
     1. Spring Framework
        1. The core problem Spring framework solves is testability. If you don't define proper dependencies, then your applications are not testable. The most important feature is dependency injection. The core of Spring framework is the IOC - inversion of control.
        2. Spring framework takes control of all the beans and their dependencies.
        3. Spring framework solves the problem of dependency injection. It helps you to build loosely coupled applications. Loosely coupled applications can be easily unit tested.
        4. The other thing with Spring framework solves was providing good integration with other frameworks. So, Spring had good integration with Hibernate ORM. iBatis for object mapping.
     2. Spring MVC
        1. Spring MVC is concerned with developing web applications. Spring MVC provides a simple way of developing web applications.
        2. One of the great things, about Spring MVC is the separation of concerns. Dispatchers servlet is concerned with just the basic front controller part. There is a model and view. There is a view resolver which is concerned just with resolving a view name to the physical view.
     3. Spring Boot
        1. Auto configuration
  3. Spring Boot Starter Projects - Starter Web and Starter JPA
  4. Spring Boot Actuator
     1. Production-ready features
     2. Actuator brings in a lot of monitoring around your application. So, in actuator you'd be able to read a lot of metadata about the application.
     3. We can see what are the beans that are configured, we can see how auto configuration has worked.
     4. We can see how many times a specific service is called; we can see how many times a specific service has failed.
  5. Spring Boot Developer
     1. Spring Boot Dev Tools starter
     2. Redeploying application automatically whenever detect any changes in files.

1. Introduction to JPA
   1. JPA stands for Java Persistence API
   2. Object Relational Impedence Mismatch
      1. What is the problem that Java persistence API is trying to solve?
      2. There is a mismatch between how we design objects and how we design tables.
      3. We would want to establish a mapping between them.
      4. Mismatches are:
         1. Different name in object (birthdate) and different in table columns (birth\_date)
         2. Many to Many relationships between table and how to handle this in object
         3. Inheritance in object and how to handle this in tables
   3. World before JPA – JDBC, Spring JDBC and myBatis
   4. JPA
      1. It solves the problem of object relational mapping.
      2. JPA you’d be able to define mappings between your classes and the tables using things like entities, relationships. You would use something called an entity manager to manage your entities.
      3. JPA also has this feature of criteria API and JPQL which provides additional ways of finding data from your database.
      4. JPQL is similar to SQL However, you would be using entities instead of tables in JPQL and criteria API represents a way you can write JPL using Java APIs.
   5. The annotations in JPA
      1. @Entity
      2. @Table(name = “tableName”)
      3. @Id
      4. @GeneratedValue
      5. @Column(name = “columnName”)
      6. @ManyToMany
      7. @ManyToMany(mappedBy = “fieldName”)
      8. @Inheritance(strategy = InheritanceType.SINGLE\_TABLE)
      9. @DiscriminatorColumn(name = “EMPLOYE\_TYPE”)