1. Spring Boot Starter Parent
   1. Spring Boot Dependencies is the parent for Spring Boot starter parent
   2. It has default configurations for all important maven configurations
   3. Specifies version of dependencies
   4. By Defaults provides java version as JDK 1.8
2. Spring Boot Starter Web
   1. All dependencies required for running web applications
   2. Spring Dependencies are provided by Spring Boot Starter web
   3. Gives us Embedded Tomcat, Hibernate-Validator, etc.
   4. Brings in by default all the items required for building web applications.
3. Embedded Tomcat
   1. Is received through Spring Boot Starter Web
   2. Tomcat is included in the application hence it is called an embedded tomcat
4. Dev Tools
   1. Keeps monitoring project folders and if any change was spotted it restarts the application
5. Dispatcher Servlet
   1. Dispatcher Servlet searches for request e.g., /login, and then searches for a view which we want it to return if we have not added response body on the URL
   2. Whenever we provide view name to Dispatcher servlet it makes use of View Resolver to resolve the view
   3. How do we inform the view resolver of the path of view?
      1. In the application. Properties
         1. “<viewname>”->path to the view
         2. Spring.mvc.view.prefix = /WEB-INF/jsp/
         3. Spring.mvc.view.suffix = .jsp
   4. To enable JSP support in embedded tomcat server we need to have a tomcat-embed-jasper dependency
6. @Controller
   1. This annotation is used by Spring Boot to detect controller classes
7. @ResponseBody
   1. When dispatcher servlet redirects request to controller class with @Controller Annotation it searches for view return to back, @ResponseBody is used if we do not want request to search for JSP view in return and return the value
8. Model
   1. Model is used to pass data from controller to the view in JSP
   2. In JSP expression language is used to fetch the value from Model
9. Spring MVC Request Flow
   1. Any Request that comes to the web application first goes to dispatcher servlet
   2. Dispatcher servlets acts as a front controller
   3. Dispatcher servlet then searches the mappings available; he would know that there is a method in login controller that can handle these request
   4. Looks at the URL and the data available to identify the right controller for it
   5. Controller puts the data to the model, also the controller says I would like to go the login view or we want to refer
   6. Dispatcher servlet then goes to the view resolver and says that someone is looking for login, view resolver then looks for prefix and suffix and sends to the dispatcher servlet, Dispatcher servlet takes the model makes it available to the view, Once the view is executed, we get the HTML Response back which is then taken by the front controller that is dispatcher servlet to end-user
   7. If we are just using Spring MVC then we will have to create the Dispatcher servlet map the URLs and do all other configurations
   8. Using Spring Boot Starter Dependency provides us auto-configuration, where it automatically provides us with Dispatcher Servlet
10. Request comes to Front Controller -> Front Controller to the actual controller -> controller sends view to Dispatcher Servlet i.e., Front Controller -> Front Controller sends view to View Resolver -> View Resolver sends the HTML Back to Front Controller -> Front Controllers send this HTML to End User
11. Request Mapping
    1. Request Mapping uses all types of requests
    2. To avoid this, we should provide specific mapping
12. Issues when developing applications with the traditional spring approach
    1. Core
       1. Used for dependency injection
    2. MVC
       1. To develop web layer for application
    3. DAO
       1. Where we used Spring JDBC
    4. ORM
    5. XML Based Configurations and annotations were used
    6. These configurations become cumbersome and hard to use with time
    7. At runtime, we have to be sure that versions are compatible and we have to deploy it
13. Microservices
    1. It is a different way of building applications unlike monolithic architecture, the traditional way is one code base or maybe multiple code base but gets deployed as a single entity i.e., monolithically deployed on the server
14. What Spring Boot is?
    1. Spring -> It is the Spring Framework
    2. Boot is Bootstrap
    3. Spring Boot lets you bootstrap spring application
    4. Spring Boot makes it easy to create stand-alone, production-grade spring-based applications that you can just run
15. What is Spring?
    1. Application Framework
    2. Let’s you build enterprise Java applications
    3. Programming and configuration model
    4. Infrastructure Support
    5. Problems with Spring
       1. Spring is a huge framework
          1. Over the year Spring has solved most of the problems like huge problems because of which it has turned out to be huge Framework
    6. Multiple Setup Steps
       1. Spring can connect to multiple applications like Mongo DB, SQL, and many other applications so it also has multiple steps for Setting the application connections
    7. Multiple Configuration Steps
    8. Multiple build and Deploy Steps
16. Spring Boot
    1. It is opinionated
    2. It features convention over configuration
    3. Standalone
       1. You can use Spring Boot to generate Spring Application
       2. Ready to use
    4. Production Ready
17. When we build spring application
    1. We need dependencies
    2. These dependencies are provided in pom.xml
    3. When we run the maven command maven goes to the URL to download the list of dependencies required and automatically adds it to the classpath rather than explicitly adding jars on the classpath
18. Spring Boot Starter Parent
    1. When we are adding this dependency, we are simply specifying that our project is a child of this parent
19. Json conversions happen automatically while returning
20. Dependencies specify what to download
21. And Spring Boot Parent specifies what version to download which is called Bill of Materials
22. Embedded Tomcat Server
    1. Convenience
       1. When we want to use Tomcat, we don’t need to download it and deploy it to use it
    2. Servlet Configurations steps not required, if we want one, we can specify in the application Config
    3. Stand-Alone Application can be built easily with an embedded tomcat server
    4. Useful for microservices architecture
       1. No multiple deployments required for building a microservices architecture
23. Spring MVC Controller
    1. Spring MVC makes u build a controller which allows you to map URL to functionality
       1. Controller is simple Java Class
       2. In Controller Functionality is mapped with 2
          1. URL
          2. Request Method
    2. Spring MVC has built is libraries to convert Java class to JSON
24. Resources of Rest API
    1. Once we have identified the resources, we can make use of that to be identified by HTTP Method
25. Business Service
    1. Business Services are singletons
    2. When we start a spring boot application spring boot creates an instance of these services and we can have other classes which depend on these services
    3. How do we specify the Spring Framework that this class in Business service?
       1. We will add @Service annotation over those Classes
    4. When applications start it detects the service classes adds them to the classpath
    5. As this is a singleton it will not always create new instance wherever these services are used
    6. How do we ask spring boot for instance of these services?
       1. We declare the class as private where we want to use when spring boot application starts it also looks for member variables required by class and injects those dependencies, we declare dependency by another annotation @Autowired
       2. How it works is Spring when detects the member variable required from the class it looks into registry if there is an instance of required member variable present and injects it inside the class where it is required
26. How to Start Spring Boot API?
    1. Spring Initializer
    2. Spring Boot CLI
    3. STS IDE
27. Customizing Spring Boot
    1. application.properties
       * 1. It lets u configure 20% of the application
         2. We can find fields that can be configured in the application.properties file on spring boot official website searching for the common application.properties
28. JPA
    1. Java Persistence API
    2. Specification that lets u do ORM – Object Relational Mapping
    3. ORM lets u map entity classes to SQL Tables
    4. We do not need to map ourselves
    5. JPA is a way to map our Data
    6. Spring Data JPA makes it easy for us to work with SQL tables and lets the framework do all the configurations
    7. We need to specify the entity classes which will be tables
       1. Each member variable inside entity classes will act as a column for the tables
       2. Each instance of the entity class will go as a row in the table
       3. We need to annotate entity classes with @Entity Annotation
       4. With the above annotation JPA knows that we need to create a table
       5. We need to specify the primary key of the class with @Id annotation
    8. Repository
       1. JPA Provides a common repository that can be used to perform crud operations on the entity classes
       2. Name of the common repository is CrudRespository, JPARespository
       3. We need to specify a few generic types to use the above repository which is: Entity classes and Primary Key
       4. We can autowire specific repository in-service classes and make use of it to perform curd operations
    9. We can make use of embedded Apache Derby database to create dummy application
    10. We can perform get, update, delete, save operations as required using repositories
    11. Foreign Key can be specified in Entity using below annotations
        1. ManyToOne to OneToMany accordingly
29. How to get application deployment-ready?
    1. Require a jar file which bundles are the configurations
       1. Add Packaging as jar in pom.xml
       2. Go to project location
       3. Perform mvn clean install
       4. Command: java -jar name
       5. To generate war
          1. Add packaging as war in pom.xml
          2. Perform maven clean install
          3. Same process will be done as above just war will be present instead of jar
30. How can we monitor our spring Boot application deployed?
    1. Actuator
       1. Production ready features to monitor your application
       2. We can perform get request to URL: /health
          1. Above is the new endpoint which got added to show us status of our application
          2. If we want our health check URLs to run on separate port on local we can specify it in application.properties with management.port property