**SPRING CORE**

1. Built upon 3 core principles
   1. Dependency Injection
   2. Aspect Oriented Programming
   3. Enterprise Abstraction layers
2. Steps to create a spring maven project
   1. Choose Maven Project
   2. Select maven-archtype-quickstart under org.apache.maven
   3. Provide the group id : <package name>
   4. Provide the ArtifactId: <project name>
   5. Add spring-context dependency in pom,xml and update the same for java and spring version under <properties> tag
3. Should follow a proper package structure for Spring to load the beans and inject them
4. Annotations
   1. @Component :Class-level to inform spring to load the class and instantiate
   2. @Service/ @Repository : Specific annotations for specific functionalities provides by the class
   3. @Configuration : Add on the class that provides with spring related configuration
   4. @ComponentScan: Tells spring to scan the root and sub-packages for the classes with spring specific annotations, load them and instantiate
   5. @Autowired : To inject the dependency for has-a relationship
      1. Constructor injection
      2. Setter injection
      3. Field injection
   6. If there are more than 1 class of the same type, to provide which bean qualifies for injection
      1. @Primary : annotated on the class
      2. @Qualifier : along with @Autowired
   7. @Bean : To create an object and to be managed by. Spring context. It is method level annation only
   8. @Value : to provide values for primitive types
5. Scope: Default all beans have singleton scope. Means only 1 instance per application is created.
   1. @Scope : modify the scope of beans

**SPRING JDBC**

1. Steps to integrate database
   1. Add spring-jdbc and mysql jar in pom.xml
   2. Create a DBConfig class
      1. Provide connection parameters using DriverManagerDataSource
      2. Inject DataSource to JdbcTemplate
   3. Autowire the JdbcTemplate in the respective class to communicate with the database
   4. Implement the CRUD operations

**SPRING BOOT**

1. It is an opinionated framework
2. It provides application context based on the build path dependencies
3. To create a spring boot application
   1. Start.pring.io
   2. STS => Spring starter project
4. To create a simple spring project with database,add below dependencies
   1. Data-jpa
   2. Database driver
5. To create a spring MVC or spring REST
   1. Add web , devtools as dependency
6. For database related connection parameters, add below in application.properties file

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:8889/ecomm

spring.datasource.username=root

spring.datasource.password=root

1. Default controller/ servlet that is responsible for all the HTTP request is DispatcherServlet
2. Spring web follows front controller design pattern

**SPRING BOOT MVC**

1. To create a spring mvc application, follow the below steps
   1. Add web, devtools and respective dependencies
   2. Configure application.properites as follows :  
      spring.datasource.url=jdbc:h2:mem:ecommerce [ H2 ]

spring.mvc.view.prefix=/

spring.mvc.view.suffix=.jsp

spring.jpa.show-sql=true

* 1. Create a class/ controller as follows
     1. Annotate with @Controller
     2. Add methods annotating with respective annotations
        1. @GetMapping
        2. @PostMapping
     3. @PathVariable
     4. @RequestParam
     5. Controller returns a string that is the name of the view as a response
     6. To redirect return “redirect:<url>”

**SPRING BOOT REST**

1. REST -> Representational State Transfer
2. Leverages HTTP
   1. Methods
   2. Status codes
   3. Headers
3. Create REST API as URI
4. To create a spring rest application
   1. Add web as a dependency
   2. Create a class as follows:
      1. Annotate with @RestController [ @Controller + @ResponseBody ]
      2. Add methods annotated with
         1. @GetMapping
         2. @PostMapping
         3. @PutMapping
         4. @DeleteMapping
      3. @PathVariable
      4. @RequestParam
      5. @RequestBody
   3. Return ResponseEntity as it allows to bundle the content, headers, status etc
   4. Exception Handling mechanisms
      1. Class level use @ExceptionHandler
      2. Global level exceptions, annotate class with @ControllerAdvice
   5. Swagger-UI
      1. Used to document the REST API