|  |  |  |
| --- | --- | --- |
| **How data stored in SQL when we pass the data from web application or postman?** | 1. **API Request** - From postman we send the data through API request. 2. **Spring boot Controller** - It receives the request and extract data from request. 3. **Spring Data JPA -** Controller interact with JPA and extract data. JPA provides methods to interact with db. 4. **JPA Entity Manager -** It creates Entity manager and interact with underlying JDBC driver. 5. **JDBC Driver** - It converts JPA query to SQL query and executes them to SQL DB server. 6. **SQL DB -** it processes the received SQL statement and added to specific table. 7. **Response -** Boot app. Received response from SQL db. | |
|  | Line 3 & 4   * It specifies the db. * These lines are used to connect with the specific DB along with the port, DB\_name and URL   Line 9 & 10   * 1. Hibernate uses this to generate SQL statements which are compatible with specific database.   2. Every DB has its own syntax so, dialect helps hibernate to generate query for DB. | |
| **What is Bean?** | 1. Bean is a java object, which manage by spring IoC container. 2. **IoC Container -** contains all the beans which get created and also managed them**.** | |
| **How is Bean created?** | 1. @Bean 2. @Component, @Service 3. @Controller, @RestControlle | |
| **========================================================================================** | If we want the object of the class (@Autowire ) then we need to denote the @Service or @Component annotation over that class.  It will create the bean object of the class and it will be maintained by the IOC container.  If we don’t apply this annotation, then we can’t use the @Autowire annotation.  **Criteria to user @Service Annotation**   1. If we apply the @Service, then that class should have the default constructure (No Arg Const.). (It is by default made by java if u don’t make the parameterized constructure. 2. If you make the parameterized constructure in that class, then java will not make the default constructor. But then we must make the configuration because java doesn’t make the bean of the parameterized constructure.      1. After making the configuration we can remove the @Service annotation from that class. But still if we want to use the @Service then also spring makes the bean object of that class. But preference will goes to our bean and not on the default bean which spring creates.   We can create the multiple beans as per we want. (Increase in parameters or decrease in para. We can do that). | |
| **After Creation How spring boot find the bean?** | 1. Through @ComponentScan spring boot application finds the bean in particular package.   @ComponentSacn (“Pass the package full name) -> Spring will find bean in that package. (Suppose we configure our bean outside the main package then with the help of @ComponentScan spring can able to find it).  @SpringBootApplication already ha the @ComponentScan if we want to give then by using @ComponentScan we can give.     1. Spring also search for the @ComponentScan and then create object of all the beans. | |
| **At what time beans get created?** | 1. **Eagerness**    * Some beans get created when we start up an application.    * For ex: - Beans with **Singleton** are eagerly initialized. 2. **Lazy**    * Some beans are get created lazy when they needed.    * For ex: - Beans with **Prototype etc…** are lazily initialized **or** bean with **@Lazy** annotation. | |
| **Life Cycle Of Bean** | | |
| **IOC Container Started** | | Here the IoC Container Inatalizes.  IOC make use of **@Configuration and @ComponentSacn** to look up the classes for which beans need to be creaed. |
| **Injected Dependency Into Constructed Beans** | | Injecting Dependency Means :    UserController class has the dependency of UserService so IOC make the object of the UserService and inject it to the already constructed UserController bean. So, it will use the method and properties of it. |
| **@PostConstructed** | | This annotation works after the bean is inatalized or create.  We can perfoer any operation on that bean after creation or inatalizing. |
| **@PreDestory** | | Before destry we can perform operation on the bean. |
| **How to apply the validation on the Entity fields?** | | Step 1: Add the **“Spring Boot Started Validation”** dependency.  Step 2: Add the **@NotNull, @NotEmpty…etc** annotatitons on the fields. (Also add the msg also to display the error msg)  Step 3: Make a exception of **MethodArgumentNotValidException** in global exception handler.    Step 4: Add the **@Valid** annotation where we use **@RequestBody** annotation (Because at the time of getting the body that time only we need to verify means when ever we are creating updating or taking the Requestbody at that time we need to add @Valid)    Step 5: Done |
|  | |  |