|  |
| --- |
| 1- XMLBasedConfiguration 2- AnnotationBasedConfiguration 3- Bean Scope- Singleton & Prototype 4- Bean Life Cycle: It contain 7 different steps  1- Definition the bean i.e. we will define that these the bean that  we will create.  2- Bean creation or instantiation process: After Define, Spring will  Instantiate those beans.  3- Assigning: Populating the properties related to bean like , id,  scope or default values provided to properties  5- Post Initialization Step: In this step whatever the interface we  Have created will be executed  i.e. custom execution will be performed in this step.  So now Bean is ready to server and bean will be available in the  container  6- Pre- Destruction of Bean: When the bean is destroying then Here we  can write some custom logic for cleanup  7- Destruction of Bean: Here the bean will be destroyed from the JVM  Itself. So the above is all about the Spring basics and dependency injection  5- AOP- Aspect-Oriented-Programming:   1. It is the methodology the create the application just like OOPS 2. It gives us the flexibility to remove the cross cutting concern(like login, authenticating, authorization, request sanitization etc) from the application and to separate it from main business logic. 3. So the help of AOP we can separate the cross-cutting concern from the business because these cross-cutting concern may come with each request so instead handling first then handle business, AOP will separate these from the business so that we can improve the business. These cross cutting concern is called Aspect. And for all cutting concern we can create the different Aspect 4. For example for Logging we will create one Aspect, For authentication and authorization we will create one Aspect and for data sanitization we will create one aspect 5. Here we will use AspectJ library with spring framework 6. Here we have differ point-cut which can be used inside the aspect like @Before(“Mention the method to be invoked after Before annotated method, @After(“Mention Method”), @AfterReturingValue, @AfterThrowingValue and we can call @AroundTheMethod |
|  |
|  |

JPA Query Method Syntax  
<https://docs.spring.io/spring-data/data-jpa/docs/1.4.x/reference/htmlsingle/???#repositories.query-methods>