* Inversion of Control (IoC)
  + outsource to an object property
* Spring IoC methods
  + Using XML Configuration file (legacy)
  + Using beans (modern)
  + Using Java Source Code (modern)
* Spring development process
  + Configuring your spring beans
  + Create a Spring Container
  + Retrieve Beans from Spring Container
* Spring Container is generally known as ApplicationContext
* A 'Spring Bean' is simply a Java Object
* Spring Injection Types
  + Constructor Injection
  + Setter Injection
  + Autowired
* Bean scopes
  + By default bean scope is singleton
  + prototype scopt to create different instance for each getBean call
* Bean Lifecycle Methods

Diagram

Description automatically generated

* + Init and destroy cannot have any args, we can have return type but we can’t capture the return value
  + For prototype scoped beans, spring does not call the destroy method
* Spring Configuration with Annotations
  + XML Configuration can be verbose for big projects. We can configure spring beans using annotations which minimizes the XML configuration
  + Annotations development process
    - Enable component scanning in Spring config file
    - Add the @component annotations to your java classes
    - Retrieve bean from spring container
  + Default bean id: the class name, make the first letter lower-case
* Spring configuration with Java annotations – dependency injection
  + Autowiring injection types
    - Constructor Injection
    - Setter Injection
    - Field Injection
  + Diagram

    Description automatically generated
  + Field injection is accomplished by using Java Reflection
  + Multiple implementations problem is solved by Qualifiers
  + for the special case of when BOTH the first and second characters of the class name are upper case, then the name is NOT converted.
* Bean scopes with annotations
  + Singleton
    - Only one object for all requests
  + Prototype
    - New object for each request
* Bean lifecycle method with annotations
  + Methods – PostConstruct, PreDestroy
  + PostConstruct will run after the execution of constructor and injection of dependencies
  + PreDestroy will run before the bean is destroyed
  + Methods cannot accept any arguments
  + We can have any return type but can’t access the return values
  + For prototype scoped beans, Spring does not call the PreDestroy method
* Spring configuration with java code (no xml)
  + Ways of Configuring spring container
    - Full XML config
    - XML component scan
    - Java configuration class
  + Defining spring beans with java code (no xml)
    - Define method to expose bean
    - Inject bean dependencies
    - Read Spring Java Configuration class
    - Retrieve bean from Spring container
  + Here is a real-time use case of using @Bean: You can use @Bean to make an existing third-party class available to your Spring framework application context