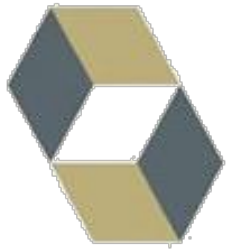


HIBERNATE

Trainer: Nilesh Ghule

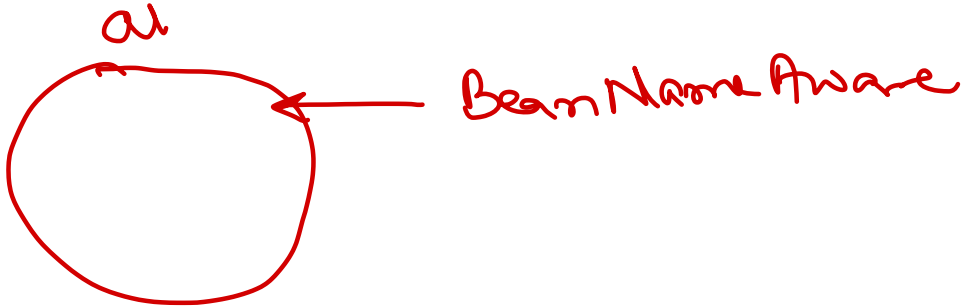
Wake up from Hibernate, Spring up!!!

Agenda

- Bean life cycle
- Bean factory vs Application context
- Auto-wiring
  - Using XML
  - Using annotation
- Mixed config
- SPEL
- Bean scopes
- Stereo type annotations
- Spring AOP

Spring bean life cycle

- Spring container creates (singleton) spring bean objects when container is started.
- Spring container controls creation and destruction of bean objects.
- There are four options to control bean life cycle.
  - InitializingBean and DisposableBean callback interfaces → bean specific logic
  - Aware interfaces for specific behaviours → associate info with specific bean.
    - BeanNameAware, BeanFactoryAware, ApplicationContextAware, etc.
  - BeanPostProcessor callback interface → common logic for all beans.
  - Custom init() / @PostConstruct and destroy() / @PreDestroy methods → bean specific.



Spring bean life cycle

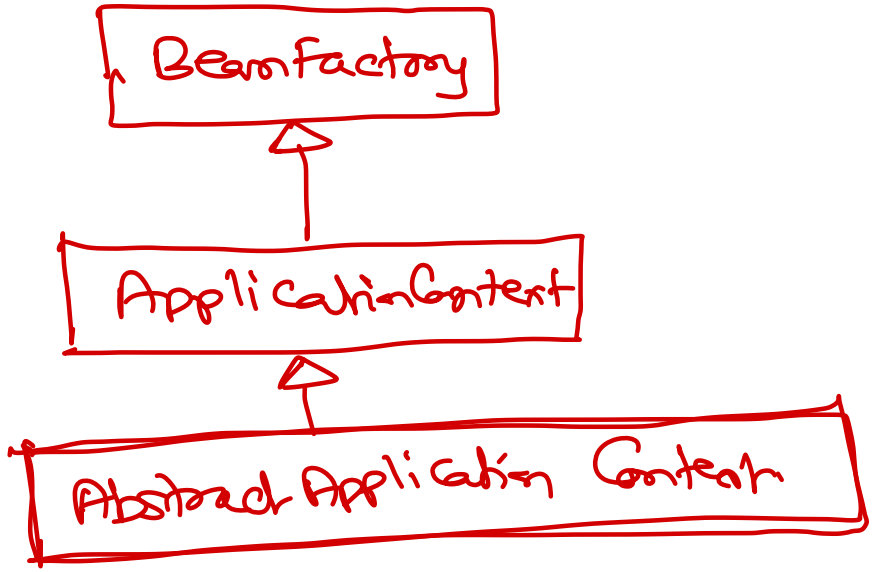
- Bean creation
  1. Instantiation
  2. Set properties
  3. BeanNameAware.setBeanName()
  4. BeanFactoryAware.setBeanFactory()
  5. BeanPostProcessor.postProcessBeforeInitialization()
  6. InitializingBean.afterPropertiesSet()
  7. Custom init()
  8. BeanPostProcessor.postProcessAfterInitialization()
- Bean destruction
  1. DisposableBean.destroy()
  2. Custom destroy()

XML Bean Factory

- Deprecated
- Minimal Dependency injection features
  - Beans loading
  - Auto-wiring
- Create a singleton bean when it is accessed first time in the application.
- No bean life cycle management

Application Context

- Features
  - Dependency injection - Beans loading and Auto-wiring
  - Automatic BeanPostProcessor registration i.e. Bean life cycle management → init / destroy
  - Convenient MessageSource access (Internationalization)
  - ApplicationEvent publication
- Create all singleton beans when application context is loaded.



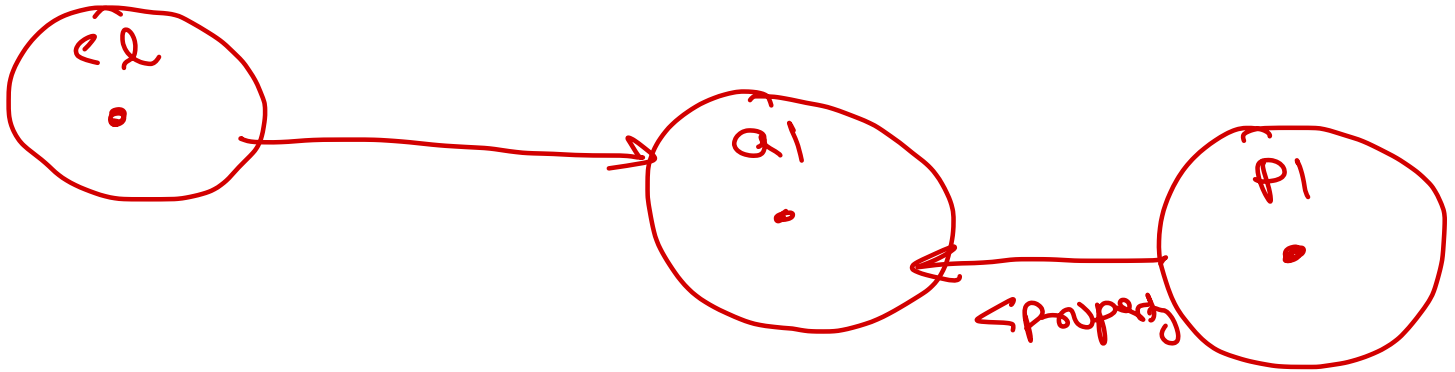
- ApplicationContext 'interface
  - ClassPathXmlApplicationContext
  - FileSystemXmlApplicationContext
  - AnnotationConfigApplicationContext
  - WebApplicationContext
    - XmlWebApplicationContext
    - AnnotationConfigWebApplicationContext

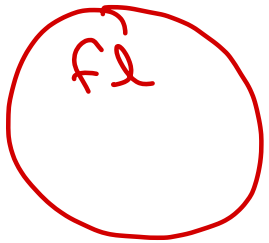
Java SE

D:/ - / beans.xml

Auto-wiring – Automated DI

- Automatically injecting appropriate dependency beans into the dependent beans.



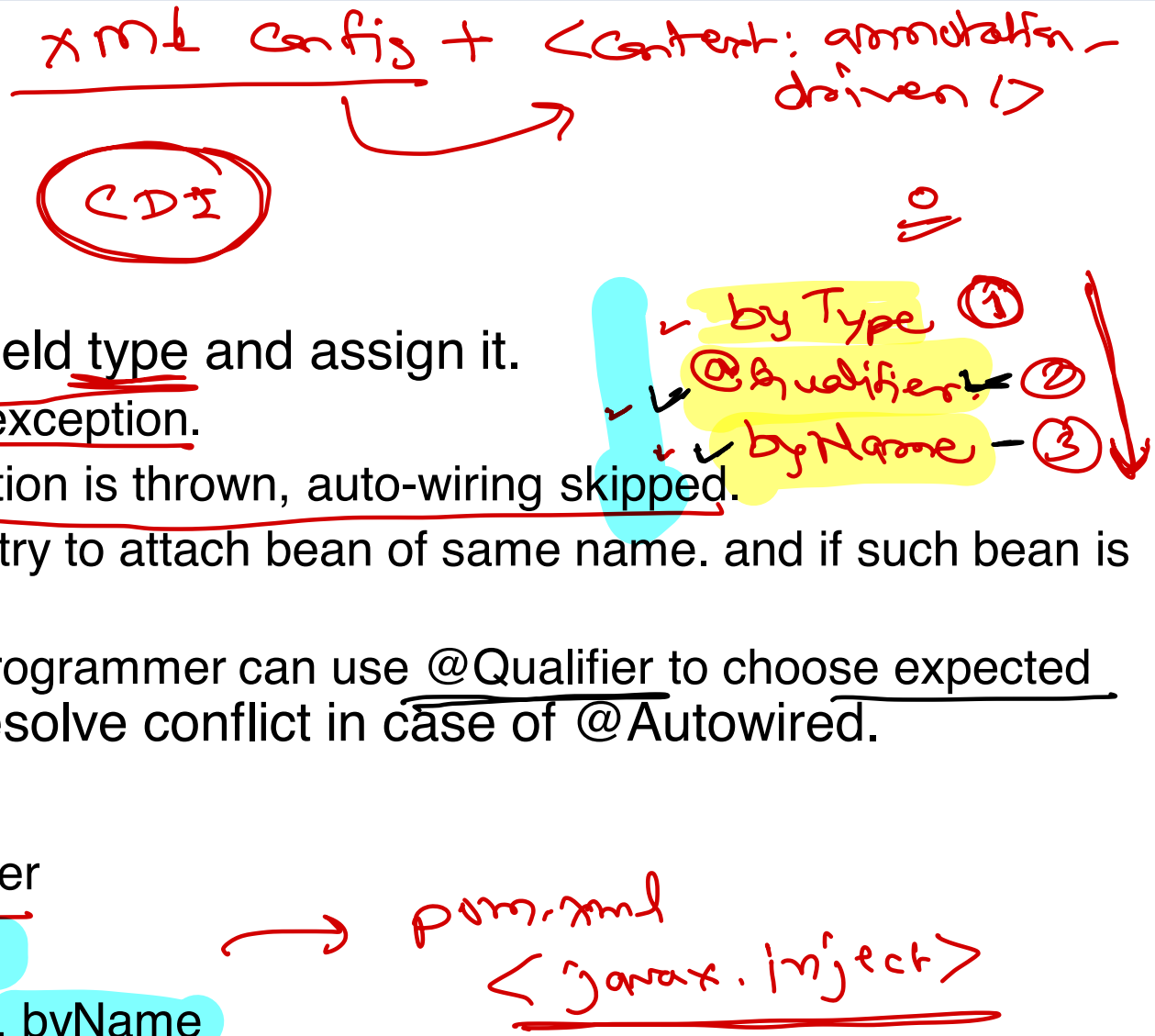


Auto-wiring (XML config)

- <bean id="..." class="..." autowire="default|no|byType|byName|constructor" .../>
  - default / no: Auto-wiring is disabled.
  - byType: Dependency bean of property type will be assigned (via setter).
    - If multiple beans of required type are available, then exception is thrown.
    - If no bean of required type is available, auto-wiring is not done.
  - byName: Dependency bean of property name will be assigned (via setter).
    - If no bean of required name is available, auto-wiring is not done.
  - constructor: Dependency bean of property type will be assigned via single argument constructor (of bean type).
- <bean id="..." class="..." autowire-candidate="true|false" .../>
  - false -- do not consider this dependency bean for auto-wiring.

Auto-wiring (Annotation config)

- @Autowired
  - Setter level: setter based DI
  - Constructor level: constructor based DI
  - Field level: field based DI
- @Autowired: Find bean of corresponding field type and assign it.
  - If no bean is found of given type, it throws exception.
    - @Autowired(required=false): no exception is thrown, auto-wiring skipped.
  - If multiple beans are found of given type, it try to attach bean of same name. and if such bean is not found, then throw exception.
  - If multiple beans are found of given type, programmer can use @Qualifier to choose expected bean. @Qualifier can only be used to resolve conflict in case of @Autowired.
- @Resource (JSR 250)
  - @Resource: DI byName, byType, byQualifier
- @Inject (JSR 330) – same as @Autowired
  - @Inject: DI byType, byQualifier (@Named), byName



Mixed configuration

- XML config + Annotations
  - XML config file is loaded using ClassPathXmlApplicationContext.
  - <context:annotation-config/> activate annotation processing.
    - @PostConstruct
    - @PreDestroy
    - @Autowired
    - @Qualifier

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>