

spring

1 Spring Spring

Spring Java Java

Spring

Spring _____ Spring

Spring

Spring 20 /

, Web AOP

2 Spring

Spring

Dependency Injection(DI)

JavaBean properties

EJB

IoC

IoC

CPU

Spring Spring

uartz JDK Timer

ORM logging J2EE Q

Spring

_____ Spring _____

JavaBean

POJO

Spring Web

Web MVC

web

Struts

web

Spring

DB

JTA

DB

3 (IOC)

Java

1.

2. Setter

3.

4 Spring IoC
 Spring org.springframework.beans org.springframework.context
 Spring IoC
 BeanFactory
 ApplicationContext BeanFactory BeanFactory
 Spring AOP message resource
 Web
 WebApplicationContext
 org.springframework.beans.factory.BeanFactory Spring IoC
 bean BeanFactory Spring IoC
 IOC: spring

5 BeanFactory ApplicationContext

BeanFactory bean BeanFactory bean
 bean
 BeanFactory bean bean
 BeanFactory
 initialization methods bean destruction methods
 application context bean factory bean bean
 bean applicationContext

1.

2.

3. bean

ApplicationContext

1 ClassPathXmlApplicationContext classpath XML

```
ApplicationContext context = new  
ClassPathXmlApplicationContext("bean.xml");
```

2 FileSystemXmlApplicationContext XML

```
ApplicationContext context = new  
FileSystemXmlApplicationContext("bean.xml");
```

3 XmlWebApplicationContext Web XML

4.AnnotationConfigApplicationContext(Java)

1. Spring XML
 - 2.
 3. Java
- 7 XML Spring XML
- Spring bean
- <beans>
- SpringXML Spring xml Java Class
- Spring XML Spring XML
- Spring context beans jdbc tx aop mvc aso

```
<beans>
  <!-- JSON Support -->
  <bean name="viewResolver"
class="org.springframework.web.servlet.view.BeanNameViewResolver"/>
  <bean name="jsonTemplate"
class="org.springframework.web.servlet.view.json.MappingJackson2JsonV
iew"/>
  <bean id="restTemplate"
class="org.springframework.web.client.RestTemplate"/>
</beans>
```

web.xml DispatcherServlet

```
<web-app>
  <display-name>Archetype Created Web Application</display-name>
  <servlet>
    <servlet-name>spring</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
    <load-on-startup>1</load-on-startup>
  </servlet>
```

```

    <servlet-mapping>
        <servlet-name>spring</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>

```

8 Java Spring

Spring Java @Configuration @Bean @Bean

 Spring IoC

@Bean <bean/> @Configuration

 bean @Configuration

 @bean bean

 @Configuration

```

@Configuration
public class AppConfig{
    @Bean
    public MyService myService() {
        return new MyServiceImpl();
    }
}

```

@Beans XML

```

<beans>
    <bean id="myService" class="com.somnus.services.MyServiceImpl"/>
</beans>

```

AnnotationConfigApplicationContext

```

public static void main(String[] args) {
    ApplicationContext ctx = new
    AnnotationConfigApplicationContext(AppConfig.class);
    MyService myService = ctx.getBean(MyService.class);
    myService.doStuff();
}

```

```
}
```

@Configuration

```
@Configuration
@ComponentScan(basePackages = "com.somnus")
public class AppConfig {
    ...
}
```

com.acme
Spring bean

@Component

web

AnnotationConfigWebApplicationContext Spring
Servlet ContextLoaderListener Spring MVC DispatcherServlet

```
<web-app>
  <!-- Configure ContextLoaderListener to use
AnnotationConfigWebApplicationContext
instead of the default XmlWebApplicationContext -->
  <context-param>
    <param-name>contextClass</param-name>
    <param-value>
org.springframework.web.context.support.AnnotationConfigWebApplicati
onContext
    </param-value>
  </context-param>

  <!-- Configuration locations must consist of one or more comma- or
space-delimited
fully-qualified @Configuration classes. Fully-qualified
packages may also be
specified for component-scanning -->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>com.howtodoinjava.AppConfig</param-value>
  </context-param>

  <!-- Bootstrap the root application context as usual using
ContextLoaderListener -->
```

```

    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class>
    </listener>

    <!-- Declare a Spring MVC DispatcherServlet as usual -->
    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
        <!-- Configure DispatcherServlet to use
AnnotationConfigWebApplicationContext
        instead of the default XmlWebApplicationContext -->
        <init-param>
            <param-name>contextClass</param-name>
            <param-value>

org.springframework.web.context.support.AnnotationConfigWebApplicati
onContext

            </param-value>
        </init-param>
        <!-- Again, config locations must consist of one or more comma-
or space-delimited
        and fully-qualified @Configuration classes -->
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>com.howtodoinjava.web.MvcConfig</param-
value>
        </init-param>
    </servlet>

    <!-- map all requests for /app/* to the dispatcher servlet -->
    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>/app/*</url-pattern>
    </servlet-mapping>
</web-app>

```

9 Spring XML
 Spring 2.5 bean bean XML

Spring Spring

```
<beans>
  <context:annotation-config/>
  <!-- bean definitions go here -->
</beans>
```

<context:annotation-config/> Spring

1. @Required
2. @Autowired
3. @Qualifier @Autowired bean
4. JSR-250 Annotations Spring JSR-250 @Resource
 @PostConstruct @PreDestroy

10 Spring Bean
 Spring Bean bean bean
 bean bean

Spring bean factory call back spring bean Bean

- 1.
- 2.

Spring bean
 InitializingBean DisposableBean
 Aware
 Bean Custom init() destroy()

@PostConstruct @PreDestroy

customInit() customDestroy() bean

```
<beans>
  <bean id="demoBean" class="com.somnus.task.DemoBean" init-
method="customInit" destroy-method="customDestroy"></bean>
</beans>
```

11 Spring Bean

Spring bean 5

1. singleton bean
bean bean factory
2. prototype bean
3. request bean
4. Session session bean session bean
5. global-session global-session Portlet
portlet portlet
global-session
Servlet session

12 Spring inner beans

Spring bean bean bean setter $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$

Person Customer Person
Person Customer

```
public class Customer{
  private Person person;
  //Setters and Getters
```

```

}

public class Person{
    private String name;
    private String address;
    private int age;
    //Setters and Getters
}

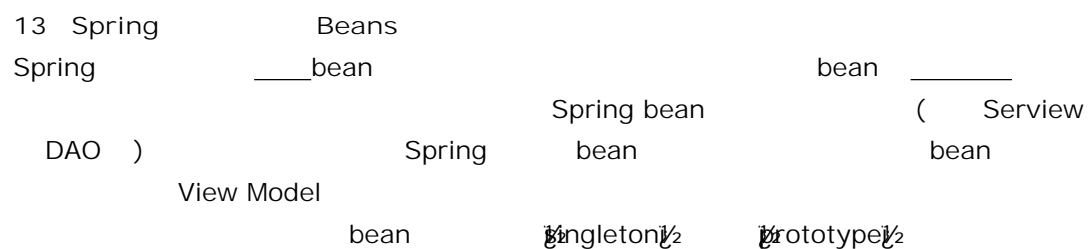
```

bean

```

<bean id="CustomerBean" class="com.somnus.common.Customer">
    <property name="person">
        <!-- This is inner bean -->
        <bean class="com.howtodoijava.common.Person">
            <property name="name" value="Iokesh" />
            <property name="address" value="India" />
            <property name="age" value="34" />
        </bean>
    </property>
</bean>

```



Spring

<list> : list

<set> : set

<map>:

<props> :

```
<beans>
  <!-- Definition for javaCollection -->
  <bean id="javaCollection" class="com.howtodoinjava.JavaCollection">
    <!-- java.util.List -->
    <property name="customList">
      <list>
        <value>INDIA</value>
        <value>Paki stan</value>
        <value>USA</value>
        <value>UK</value>
      </list>
    </property>

    <!-- java.util.Set -->
    <property name="customSet">
      <set>
        <value>INDIA</value>
        <value>Paki stan</value>
        <value>USA</value>
        <value>UK</value>
      </set>
    </property>

    <!-- java.util.Map -->
    <property name="customMap">
      <map>
        <entry key="1" value="INDIA"/>
        <entry key="2" value="Paki stan"/>
        <entry key="3" value="USA"/>
        <entry key="4" value="UK"/>
      </map>
    </property>
  </bean>
</beans>
```

```

    </property>

    <!-- java.util.Properties -->
    <property name="customProperties">
        <props>
            <prop key="admin">admin@nospam.com</prop>
            <prop key="support">support@nospam.com</prop>
        </props>
    </property>

</bean>
</beans>

```

15 Spring Bean Java.util.Properties

<props>

```

<bean id="adminUser" class="com.somnus.common.Customer">

    <!-- java.util.Properties -->
    <property name="emails">
        <props>
            <prop key="admin">admin@nospam.com</prop>
            <prop key="support">support@nospam.com</prop>
        </props>
    </property>

</bean>

```

util:½
bean

properties

propertiesbean

setter

16 Spring Bean

Spring bean Spring Spring
 bean Spring Bean Factory
 bean bean bean

 XML bean

```
<bean id="employeeDAO" class="com.howtodoinjava.EmployeeDAOImpl"
autowire="byName" />
```

bean @Autowired
 bean @Autowired Spring

```
<context:annotation-config />
```

AutowiredAnnotationBeanPostProcessor

```
<bean class
="org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor"/>
```

```

@Autowired
@Autowired
public EmployeeDAOImpl ( EmployeeManager manager ) {
    this.manager = manager;
}

```

17

- Spring 5
1. no Spring bean
 2. byName bean bean bean
bean bean
 3. byType bean bean bean
bean bean
 4. constructor byType bean
bean
 5. autodetect byType bean
byTpe

18

@Autowired AutowiredAnnotationBeanPostProcessor

1 <bean> <context:annotation-config>

```
<beans>
  <context:annotation-config />
</beans>
```

2 bean AutowiredAnnotationBeanPostProcessor

```
<beans>
  <bean
class="org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor"/>
</beans>
```

```

19         @Required
           IoC           bean bean bean

<bean>    dependency-check½
           bean
           bean dependency-
check½    @Required
           bean

```

```

public class EmployeeFactoryBean extends AbstractFactoryBean<Object>{
    private String designation;
    public String getDesignation() {
        return designation;
    }
    @Required
    public void setDesignation(String designation) {
        this.designation = designation;
    }
    //more code here
}

```

```

RequiredAnnotationBeanPostProcessor Spring
@Required bean
RequiredAnnotationBeanPostProcessor bean IoC

```

```

<bean
class="org.springframework.beans.factory.annotation.RequiredAnnotationBeanPostProcessor" />

```

@Required
BeanInitializationException

20 @Autowired

@Autowired @Required bean @Autowired bean
Spring @Autowired setter @Autowired <property>
bean @Autowired @Autowired byType
<constructor-arg>

```
public class TextEditor {  
    private SpellChecker spellChecker;  
    @Autowired  
    public TextEditor(SpellChecker spellChecker){  
        System.out.println("Inside TextEditor constructor. " );  
        this.spellChecker = spellChecker;  
    }  
    public void spellCheck(){  
        spellChecker.checkSpelling();  
    }  
}
```

<beans>

<context:annotation-config/>

<!-- Definition for textEditor bean without constructor-arg -->
<bean id="textEditor" class="com.howtodoinjava.TextEditor"/>

<!-- Definition for spellChecker bean -->
<bean id="spellChecker" class="com.howtodoinjava.SpellChecker"/>

21 @Qualifier

```
@Qualifier      bean      Qualifier
      Spring      bean
      Customer  person      person
```

```
public class Customer{
    @Autowired
    private Person person;
}
```

Person

```
<bean id="customer" class="com.somnus.common.Customer" />

<bean id="personA" class="com.somnus.common.Person" >
    <property name="name" value="Iokesh" />
</bean>

<bean id="personB" class="com.somnus.common.Person" >
    <property name="name" value="alex" />
</bean>
```

Spring

person bean

Caused by:

org.springframework.beans.factory.NoSuchBeanDefinitionException:

No unique bean of type [com.howtodoinjava.common.Person] is defined:
expected single matching bean but found 2: [personA, personB]

@Qualifier

Spring

bean

```
public class Customer{  
    @Autowired  
    @Qualifier("personA")  
    private Person person;  
}
```

22

1. int string long

2.

3.

4. A B A Spring B A
sObjectCurrentlyInCreationException
Spring

23 Spring

Spring ApplicationContext
 bean ApplicationContext ApplicationEvent
 ApplicationContext bean
 ApplicationListener ApplicationEvent bean

```
public class AllApplicationEventListener implements ApplicationListener
< ApplicationEvent >{
    @Override
    public void onApplicationEvent(ApplicationEvent applicationEvent)
    {
        //process event
    }
}
```

- Spring 5
- ContextRefreshedEvent ApplicationContext
 ConfigurableApplicationContext refresh()
 - ContextStartedEvent ConfigurableApplicationContext
 Start() /
 - ContextStoppedEvent ConfigurableApplicationContext
 Stop()
 - ContextClosedEvent ApplicationContext
 Bean
 - RequestHandledEvent Web http request

ApplicationEvent

```
public class CustomApplicationEvent extends ApplicationEvent{
    public CustomApplicationEvent ( Object source, final String msg ){
        super(source);
        System.out.println("Created a Custom event");
    }
}
```

```

public class CustomEventListener implements ApplicationListener <
CustomApplicationEvent >{
    @Override
    public void onApplicationEvent(CustomApplicationEvent
applicationEvent) {
        //handle event
    }
}

```

```

        applicationContext.publishEvent()
CustomApplicationEvent customEvent = new
CustomApplicationEvent(applicationContext, "Test message");
applicationContext.publishEvent(customEvent);

```

24 FileSystemResource ClassPathResource

```

    FileSystemResource spring-config.xml
        ClassPathResource spring ClassPath
ClassPathResource ClassPath
    spring-config.xml src
src
        ClassPathResource
        FileSystemResource

```

25 Spring

Spring

- o ½ AOP remoting
- o ½ spring bean

- $\frac{1}{2}$ RestTemplate, JmsTemplate, JpaTemplate
- $\frac{1}{2}$ Spring DispatcherServlet
- (View Helper) $\frac{1}{2}$ Spring JSP
- $\frac{1}{2}$ BeanFactory / ApplicationContext
- $\frac{1}{2}$ BeanFactory

1. Spring ?

. IOC

. AOP

. .

2. AOP IOC AOP:

Aspect Oriented Program, () ;Filter() AOP. AOP
 , OOP(Object-Oriented Programming,) . AOP
 (aspect), .

IOC: Invert Of Control, . DI() .

IOC , , .

3. Spring Bean ?

Bean : &
 FactoryBean

4. IOC Bean :

. Bean

. Bean Bean

. Bean Bean postProcessBeforeInitialization

. Bean (init-method)

```
. Bean Bean postProcessAfterInitialization
. Bean
. , Bean (destroy-method)
```