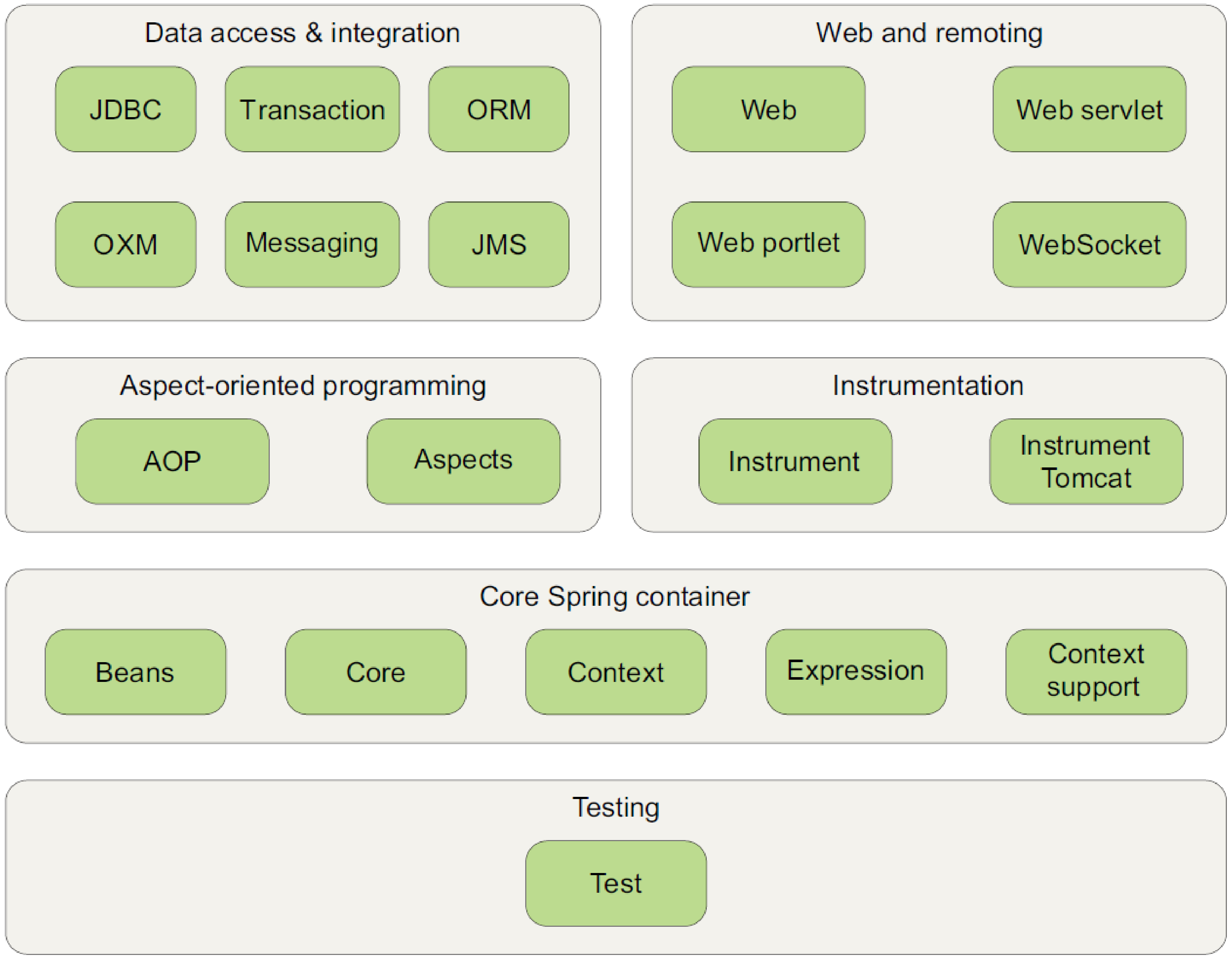
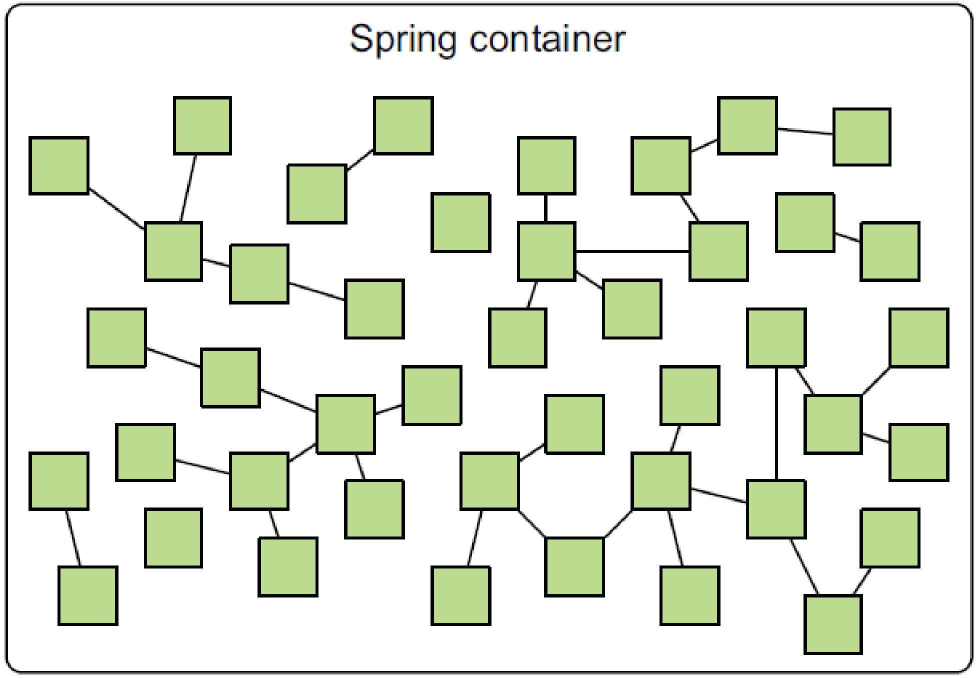
Fundamentals of Spring Framework

- ✓What is Spring?
- ✓Reflection
- ✓Inversion of control
- ✓Dependency Inversion
- ✓Dependency Injection
- ✓Spring context
- ✓SpEL


Data access & integration
JDBC
Transaction
ORM
OXM
Messaging
JMS
Web and remoting
Web
Web servlet
Web portlet
WebSocket
Aspect-oriented programming
AOP
Aspects
Instrumentation
Instrument
Instrument Tomcat
Core Spring container
Beans
Core
Context
Expression
Context support
Testing
Test


Spring container

AnnotationConfigApplicationContext

Loads a Spring application context
from one or more Java-based configuration classes

ClassPathXmlApplicationContext

Loads a context definition from one or
more XML files located in the classpath,
treating context-definition files as classpath resources

FileSystemXmlApplicationContext

Loads a context definition from one or
more XML files in the filesystem

AnnotationConfigWebApplicationContext

Loads a Spring web application
context from one or more Java-based configuration classes

XmlWebApplicationContext

Loads context definitions from one or more
XML files contained in a web application

- Explicit configuration in XML
- Explicit configuration in Java
- Implicit bean discovery and automatic wiring

Explicit configuration in XML

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context">

  <!-- configuration details go here -->

</beans>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

  <context:component-scan base-package="soundsystem" />

</beans>
```

```
<bean class=“mypack.Cat" />

<bean id=“myCat” class=“mypack.Cat" />
```

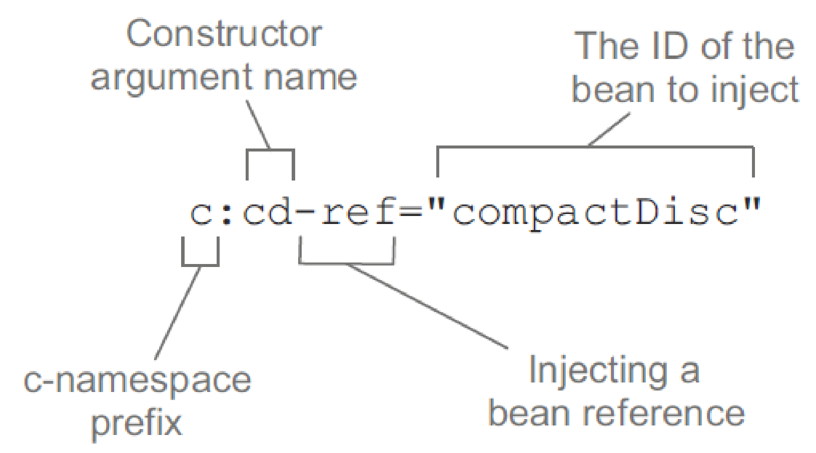
```
<bean id=“circle" class=“mypack.Circle">
        <constructor-arg ref=“myPoint" />
</bean>


<bean id=“myPoint" class=“mypack.Point">
        <constructor-arg value=“point1" />
</bean>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:c="http://www.springframework.org/schema/c"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans.xsd">

 ...

</beans>
```


Constructor argument name
The ID of the bean to inject
c:cd-ref="compactDisc"
c-namespace prefix
Injecting a bean reference

```
<bean id="myCircle"
      class="mypack.Circle"
      c:point-ref="myPoint" />
```

```
<bean id="myCircle"
      class="mypack.Circle"
      c:_0-ref="myPoint" />
```

```
<bean id="myPoint"
        class="mypack.Point"
        c:_x="10" c:_y ="20" />


<bean id="myPoint"
        class="mypack.Point"
        c:_0="10" c:_1 ="20" />
```

```
public class Book{
        private List<String> authors;

        public Book(List<String> authors){
                this.authors = authors;
        }
}
```

```
<bean id=“b1" class=“mypack.Book">
        <constructor-arg><null/></constructor-arg>
</bean>


<bean id=“b1" class=“mypack.Book">
        <constructor-arg>
                <list>
                        <value>First Author</value>
                        <value>Second Author</value>
                </list>
        </constructor-arg>
</bean>
```

What if author was an object?

```
<bean id=“b1" class=“mypack.Book">
        <constructor-arg>
                <list>
                        <ref bean=“a1” />
                        <ref bean=“a2” />
                </list>
        </constructor-arg>
</bean>
```

What if we had a set and not a list?

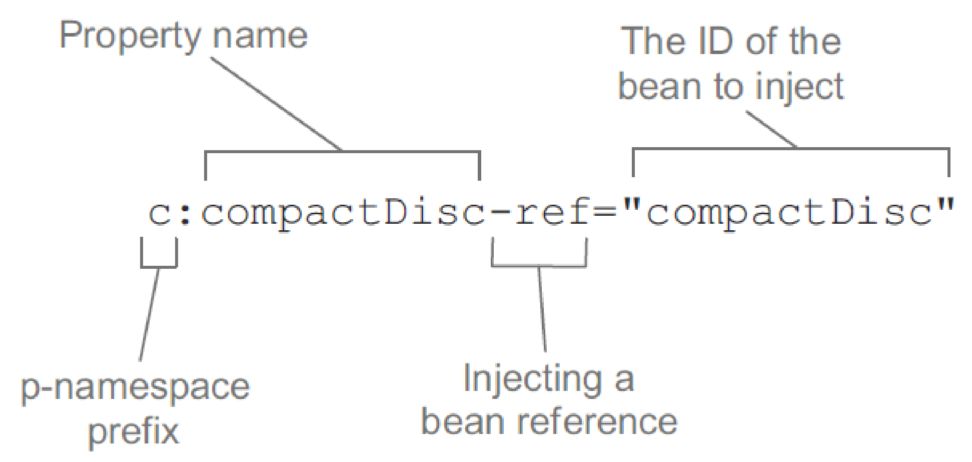
```
<bean id="b1" class="mypack.Book">
        <constructor-arg>
                <set>
                        <value>First Author</value>
                        <value>Second Author</value>
                </set>
        </constructor-arg>
</bean>
```

But how do we inject without constructors?

```
<bean id=“circle1“ class=“mypack.Circle">
        <property name=“point" ref=“myPoint" />
</bean>


<bean id=“myPoint“ class=“mypack.Point">
        <property name=“x" value=“10" />
        <property name=“y" value=“20" />
</bean>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:p="http://www.springframework.org/schema/p"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">
  ...
</bean>
```


Property name
The ID of the bean to inject
c:compactDisc-ref="compactDisc"
p-namespace prefix
Injecting a bean reference

```
<bean id="myCircle"
        class="mypack.Circle"
        p:point-ref="myPoint" />

<bean id="myCircle"
        class="mypack.Circle"
        p:_0-ref="myPoint" />
```

```
<bean id="“b1" class="“mypack.Book">
        <property name="“authors”"><null/></ property >
</bean>


<bean id="“b1" class="“mypack.Book">
         <property name="“authors”">
                 <list>
                         <value>First Author</value>
                         <value>Second Author</value>
                 </list>
        </property>
</bean>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:p="http://www.springframework.org/schema/p"
  xmlns:util="http://www.springframework.org/schema/util"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/util
    http://www.springframework.org/schema/util/spring-util.xsd">

    ...

</beans>
```

Element	Description
`<util:constant>`	References a `public static` field on a type and exposes it as a bean
`<util:list>`	Creates a bean that is a `java.util.List` of values or references
`<util:map>`	Creates a bean that is a `java.util.Map` of values or references
`<util:properties>`	Creates a bean that is a `java.util.Properties`
`<util:property-path>`	References a bean property (or nested property) and exposes it as a bean
`<util:set>`	Creates a bean that is a `java.util.Set` of values or references

```
<util:list id="authors">
        <value>First Author</value>
        <value>Second Author</value>
</util:list>


<util:map id="phoneNumbers">
        <entry key="john" value="07222111222" />
        <entry key="harry" value="07233322231" />
</util:map>
```

Implicit bean discovery and automatic wiring

@Component

@Repository

@Service

@Controller

@Component vs. @Named

```
@Component(“myCat")
public class Cat
```

```
@Named(“myCat")
public class Cat
```

Defining a configuration

@Configuration

defines a configuration class

@ComponentScan

specifies the package/packages/class/classes defining components in your project

```
@Configuration
@ComponentScan(“mypackage")
public class ProjectConfig{}
```

```
@Configuration
@ComponentScan(basePackages={“pack1", “pack2"})
public class ProjectConfig{}
```

```
@Configuration
@ComponentScan(basePackageClasses={Cat.class, Dog.class})
public class ProjectConfig{}
```

@Autowired

```
@Autowired
private Point point;

@Autowired
public Circle(Point point){}

@Autowired
public void setPoint(Point point){}

@Autowired(required = false)
public void setPoint(Point point){}
```

@Autowired vs. @Inject

```
@Inject
private Point point;

@Inject
public Circle(Point point){}

@Inject
public void setPoint(Point point){}
```

Explicit configuration in Java

```
@Bean
public Point myPoint() {
        return new Point();
}


@Bean(“nameOfMyPoint”)
public Point myPoint() {
        return new Point();
}
```

```
@Bean(“nameOfMyPoint”)
@Autowired
public Circle myCircle(Point point) {
        return new Circle();
}
```

Mixing configurations

```
@Configuration
@Import(AnotherProjectConfig.class)
@ComponentScan(basePackages={“pack1", “pack2"})
public class ProjectConfig{}
```

```
@Configuration
@Import({Config1.class, Config2.class})
@ComponentScan(basePackages={“pack1", “pack2"})
public class ProjectConfig{}
```

```
@Configuration
@ImportResource(“classpath: config-file.xml”)
@ComponentScan(basePackages={“pack1", “pack2"})
public class ProjectConfig{}
```

```
@Configuration
@Import({Config1.class, Config2.class})
@ImportResource(“classpath: config-file.xml”)
@ComponentScan(basePackages={“pack1", “pack2"})
public class ProjectConfig{}
```

```
<import resource="“second-config.xml" />
```

Ambiguity

If auto wiring is done on a abstract type, as an interface defined reference, the container will try to find a matching concrete bean.

If two or more are available, an ambiguity exception will occur.

SOLUTION:

@Primary

```
<bean id=“cat1“  class=“mypack.Cat“ primary="true" />

<bean id=“cat2“  class=“mypack.Cat“/>
```

@Qualifier

What can @Qualifier be used for?

- Naming a component bean – used above the class definition
- Specify that auto wiring is done with a bean named other way then the variable identifier
- Naming the bean definition in configuration class

```
@Component
@Qualifier("mitzy")
public class Cat{}


@Autowired
@Qualitfier("mitzy")
public void setCat(Cat cat){
}

@Bean
@Qualifier("mitzy")
public Cat myCat(){
        return new Cat();
}
```

Defining your own @Qualifier

@Target({ElementType.CONSTRUCTOR, ElementType.FIELD, ElementType.METHOD, ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Qualifier
public @interface **BadCat**{ }

@Component
@BadCat
public class Cat{}

Bean scope singleton vs prototype

- Default in Spring a bean is defined of scope singleton
- A singleton is an unique instance per identifier
- A prototype is always a new instance per identifier

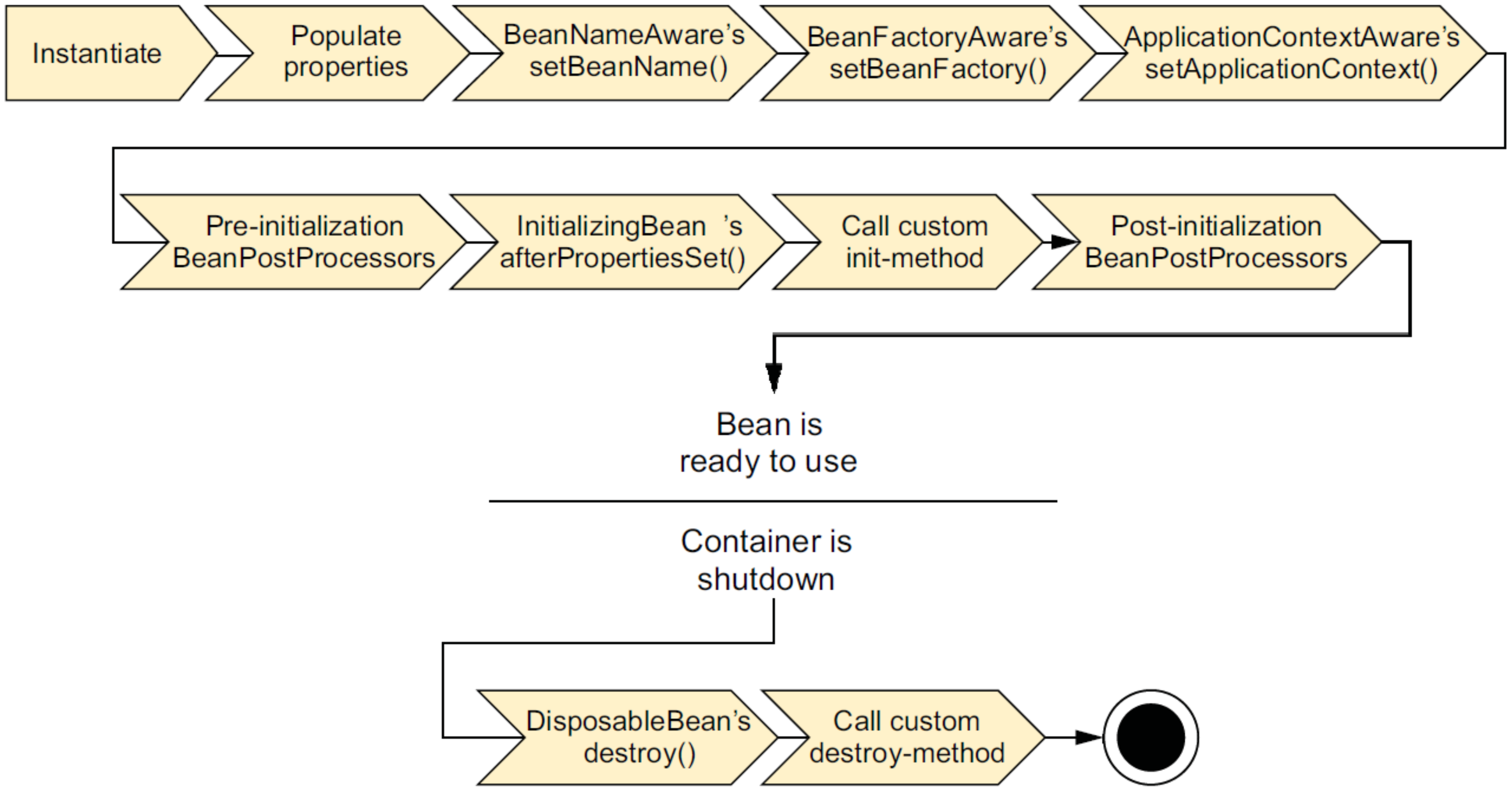
```
<bean id="cat1" class="cat.Cat" scope="prototype" />
```

```
@Component
@Scope(ConfigurableBeanFactory.SCOPE_PROTOTYPE)
public class Cat{ ... }
```

```
@Bean
@Scope(ConfigurableBeanFactory.SCOPE_PROTOTYPE)
public Cat myCat() {
        return new Cat();
}
```

Lazy vs Eager

- Eager means instantiation at context startup
- Lazy means instantiation at request
- The bean definition in Spring is default Eager
- To define a bean with lazy instantiation in Spring use **@Lazy** annotation or lazy="" attribute in XML


Instantiate
Populate properties
BeanNameAware's setBeanName()
BeanFactoryAware's setBeanFactory()
ApplicationContextAware's setApplicationContext()
Pre-initialization BeanPostProcessors
InitializingBean 's afterPropertiesSet()
Call custom init-method
Post-initialization BeanPostProcessors
Bean is ready to use
Container is shutdown
DisposableBean's destroy()
Call custom destroy-method

- BeanNameAware
- BeanFactoryAware
- ApplicationContextAware
- BeanPostProcessor
- InitializingBean
- DisposableBean

@PostConstruct

- annotation used to mark a method of a bean to be called immediately after bean construction

@PostConstruct
public void init(){}

@PreDestroy

- annotation used to mark a method of a bean to be called immediately before eliminating bean from context

@PreDestroy
public void close(){}

SpEL – Spring Expression Language

Way of wiring values into a bean’s properties or constructor arguments using expressions that are evaluated at runtime

- The ability to reference beans by their IDs
- Invoking methods and accessing properties on objects
- Mathematical, relational and logical operations on values
- Regular expression matching Collection manipulation

Syntax:

```
#{ expression }
```

Examples:

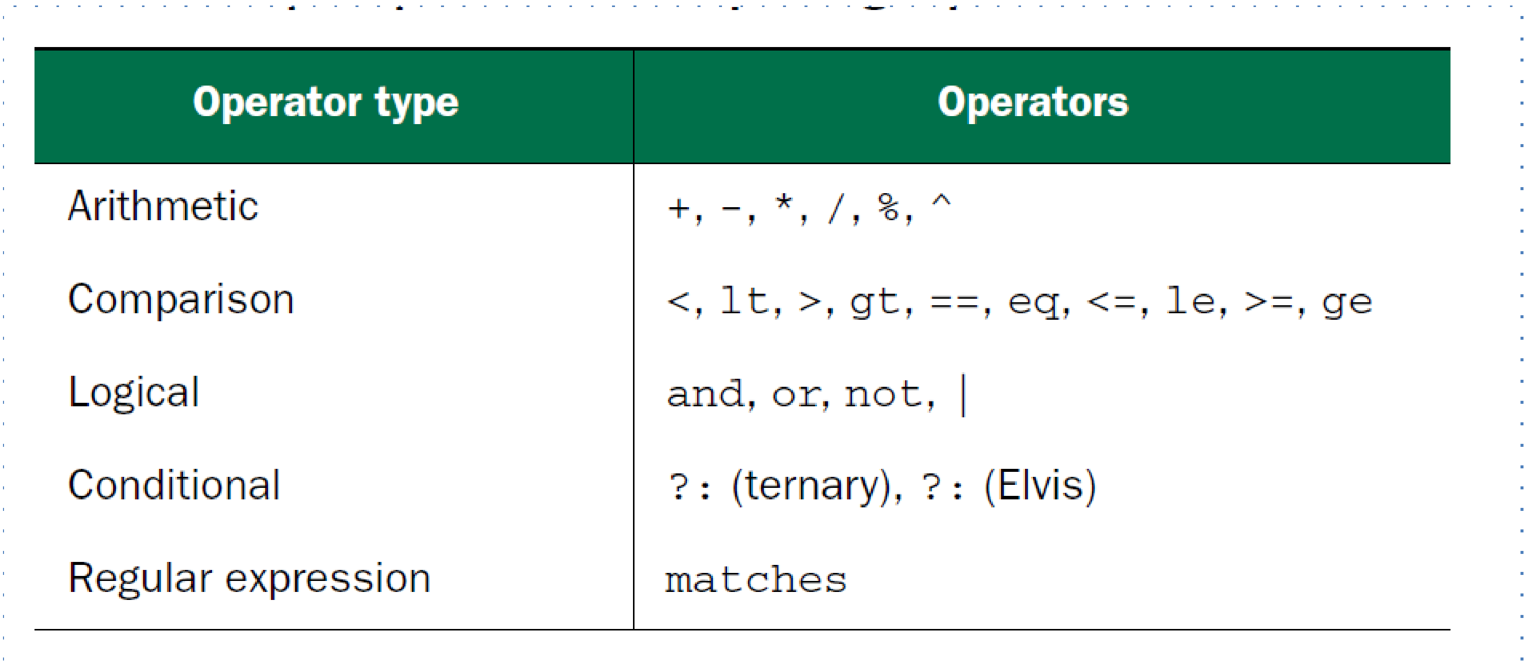
```
#{1}

#{ a == 5 }

#{cat.color}

#{T(System).currentTimeMillis()}

#{systemProperties[‘propertyname']}
```



Operator type	Operators
Arithmetic	+, -, *, /, %, ^
Comparison	<, lt, >, gt, ==, eq, <=, le, >=, ge
Logical	and, or, not, |
Conditional	?: (ternary), ?: (Elvis)
Regular expression	matches