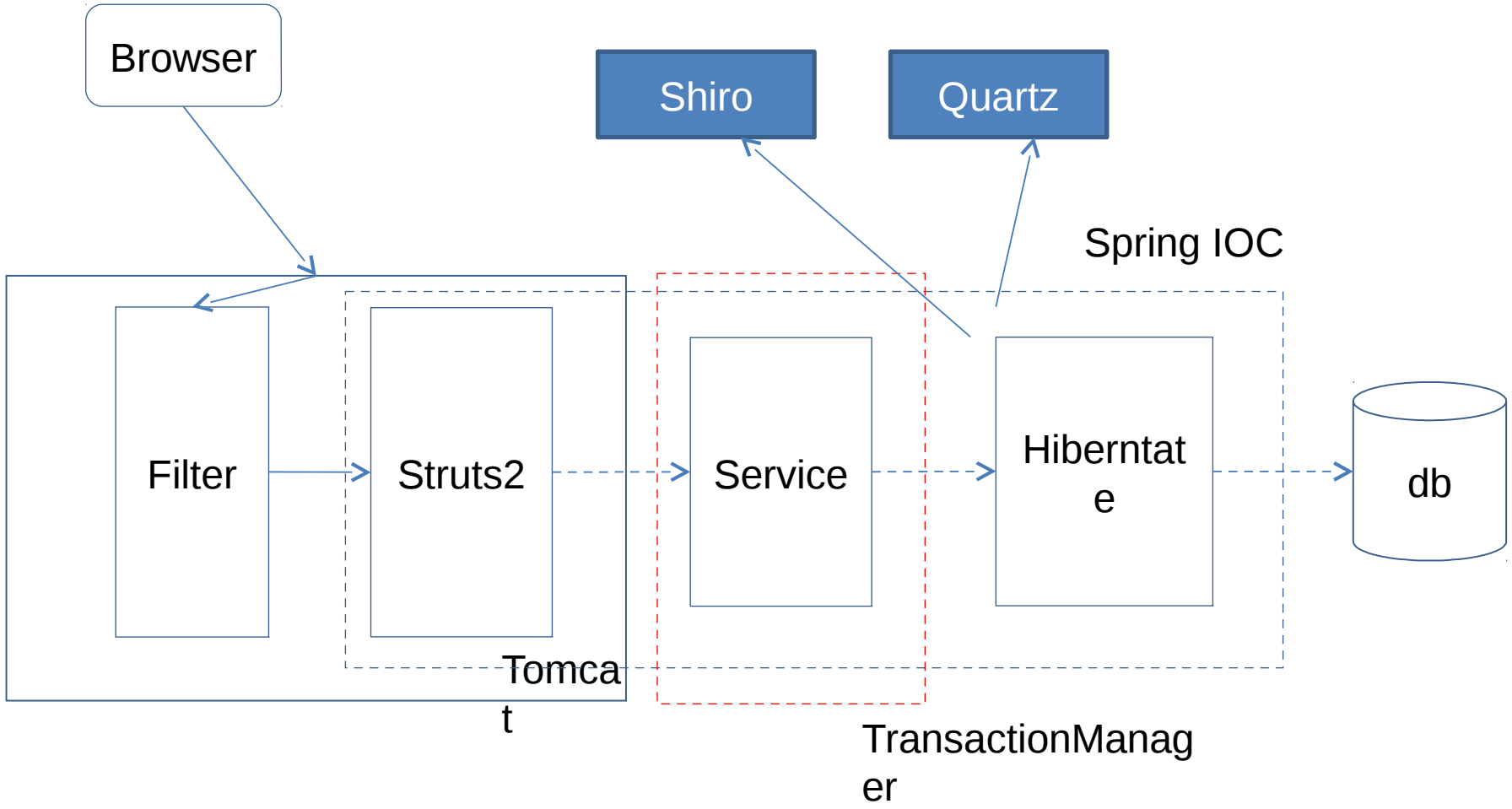
讲师：佟刚

新浪微博：尚硅谷 - 佟刚

B\S


Browser
Shiro
Quartz
Spring IOC
Filter
Struts2
Service
Hiberntat
e
db
Tomca
t
TransactionManag
er

Hello World

讲讲：讲讲

讲讲讲讲：讲讲讲-讲讲

Spring □□□ (1)

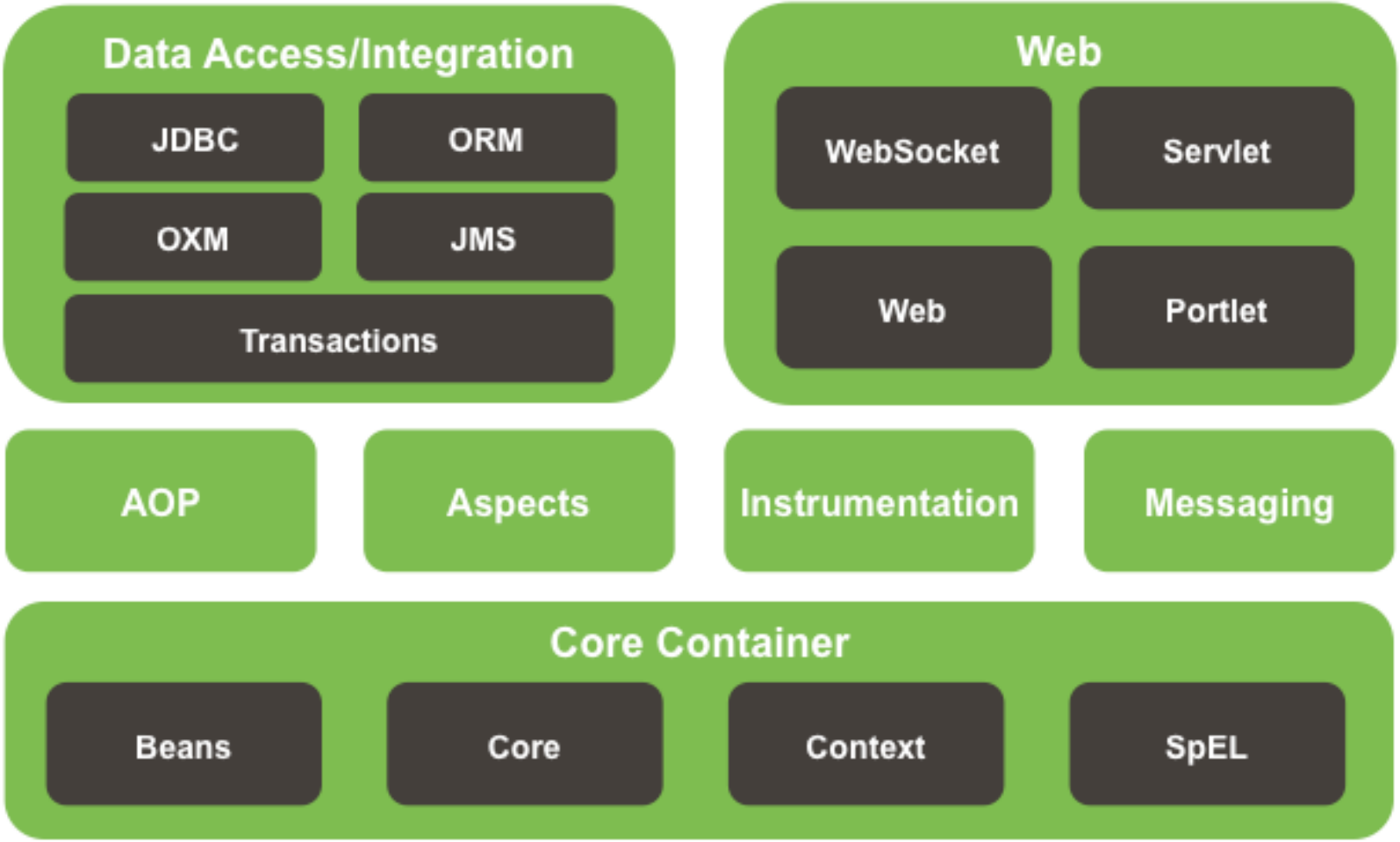
- Spring □□□□□□□ .
- Spring □□□□□□□□□□□□ . □□ Spring □□□□□□ JavaBean □□□ □□□ EJB □□□□□□□ .
- Spring □□□ IOC(DI) □ AOP □□□□ .

J2ee without ejb

Spring □□□ (2)

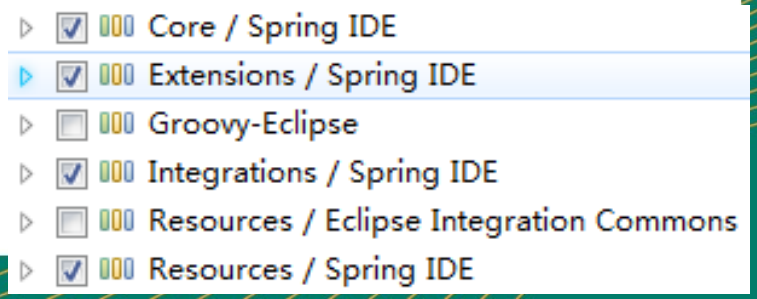
- □□□□ Spring:
  - □□□□ **Spring** □□□□□□ - □□ Spring □□□□□□□□□□□□□□□□ Spring □ API
  - □□□□ (DI --- dependency injection □ IOC)
  - □□□□□□ (AOP --- aspect oriented programming)
  - □□ : Spring □□□□□ , □□□□□□□□□□□□□□□□□□
  - □□ : Spring □□□□□□□□□□□□□□□□□□□□□□□□ . □ Spring □□□□□ XML □ Java □□□ □□□□□
  - □□□□ IOC □ AOP □□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□ Spring □□□□□□□□□□ SpringMVC □ □□□□ Spring JDBC □

Spring 框架



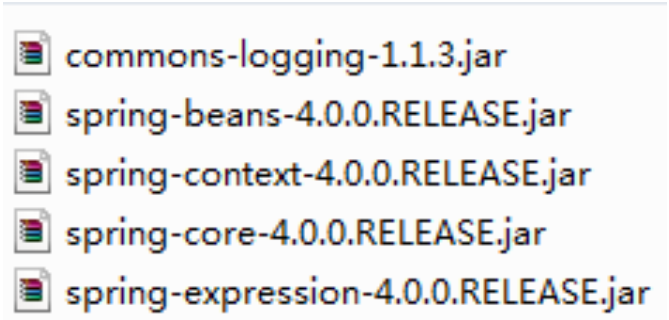
□□ SPRING TOOL SUITE

- SPRING TOOL SUITE □□□ Eclipse □□□□□□□□□□□□□□□ Eclipse □□□□□□□ Spring □□□□
- □□□□□□□ springsource-tool-suite-3.4.0.RELEASE-e4.3.1-updatesite.zip □□
  - **Help** --> **Install New Software...**
  - Click Add...
  - In dialog Add Site dialog, click **Archive...**
  - Navigate to **springsource-tool-suite-3.4.0.RELEASE-e4.3.1-updatesite.zip** and click **Open**
  - Clicking **OK** in the Add Site dialog will bring you back to the dialog 'Install'
  - Select the **xxx/Spring IDE** that has appeared
  - Click **Next** and then **Finish**
  - **Approve the license**
  - Restart eclipse when that is asked



□□ Spring □□□□

- □□□ jar □□□□□□□ classpath □:



- Spring □□□□□: □□□□□ Spring □□□□□□□□□□□ Bean □□□□, □□□□□□□□□□ Spring IOC □□□□□ Bean. Bean □□□□□□□□□ **classpath** □, □□□ □□□□□□□□.

第一个 Spring 程序

```java
package com.atguigu.spring.helloworld;

public class HelloWorld {

    private String userName;

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public void hello(){
        System.out.println("Hello: " + userName);
    }

}
```

HelloWorld.java

```xml
<bean id="helloWorld"
    class="com.atguigu.spring.helloworld.HelloWorld">
    <property name="userName" value="Spring"></property>
</bean>
```

applicationContext.xm

□□ Spring □□

```java
public static void main(String[] args) {

    //1. 创建 Spring 的 IOC 容器
    ApplicationContext ctx =
            new ClassPathXmlApplicationContext("applicationContext.xml");

    //2. 从容器中获取 Bean
    HelloWorld helloWorld = (HelloWorld) ctx.getBean("helloWorld");
    System.out.println(helloWorld);

    //3. 调用方法
    helloWorld.hello();
}
```

Spring 是是 Bean 是是

讲讲：讲讲

讲讲讲讲 ：讲讲讲 - 讲讲

□□□□

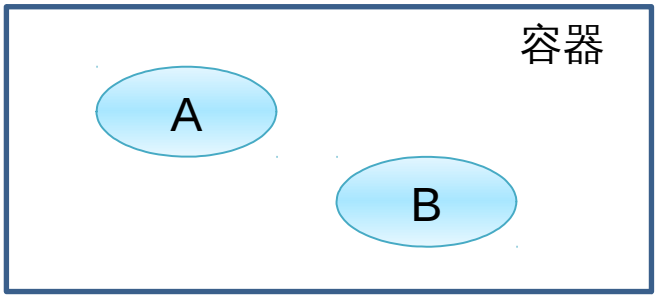
- **IOC & DI □□**
- □□ bean
  - □□□□□□□□ XML □□□□□□□□□□□□□□□□
  - Bean □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ & □□□□□□□□□ FactoryBean
  - IOC □□ BeanFactory & ApplicationContext □□
  - □□□□□□□□□□□□□□□□□□□□□
  - □□□□□□□□
  - □□□□
  - bean □□□□□□□□□□□□□
  - bean □□□□□ singleton □ prototype □ WEB □□□□□
  - □□□□□□□□□
  - spEL
  - IOC □□□ Bean □□□□□
  - Spring 4.x □□□□□□□□□□□

IOC □ DI

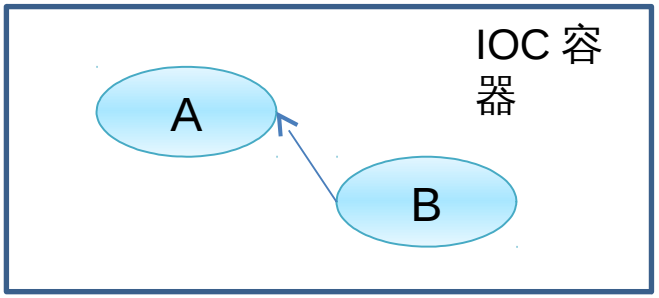
- IOC(Inversion of Control)□□□□□□□□□□□□□□. □□□□□□□□□□□□□□□□□□□□□□□□. □□□□, □□□□□□□□□. □□□□ IOC □□, □□□□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□□□□□. □□□□□□□□□□□□□□□
- DI(Dependency Injection) — IOC □□□□□□□□□□□□□□□□□□□□□□□ **(□□ : setter □□)**□□□□□□□□□□□□□**.** □□□ IOC □□□□□□□□□□

```
class A{}

class B{
        private A a;
        public void setA(A a){
                this.a = a;
        }
}
```

需求：从容器中获取 B 对象，并使 B 对象的 a 属性被赋值为容器中 A 对象的引用



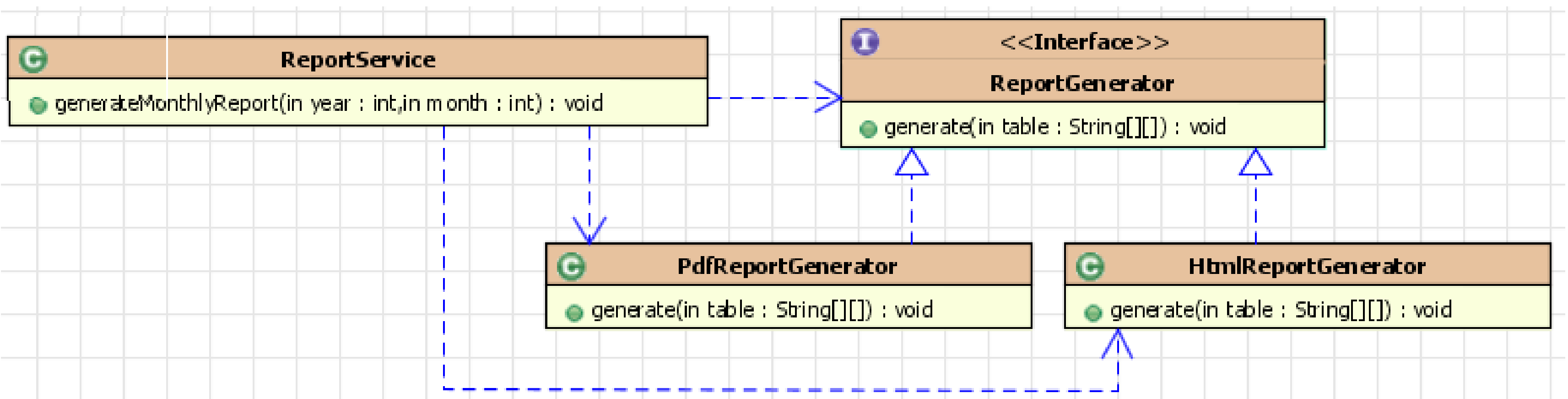

```
A a = getA();
B b = getB();
b.setA(a);
```




```
B b = getB();
```

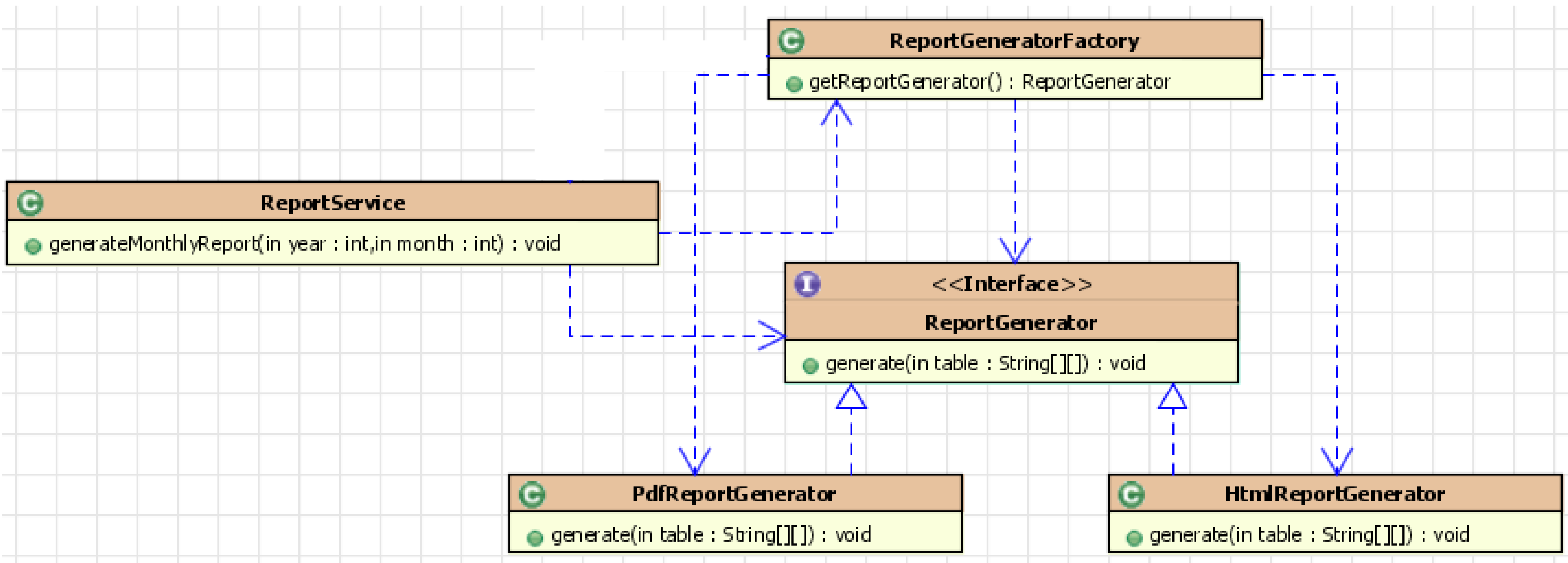
IOC □□ --- □□□□□□□

- □□: □□ HTML □ PDF □□□□□□□□□□.

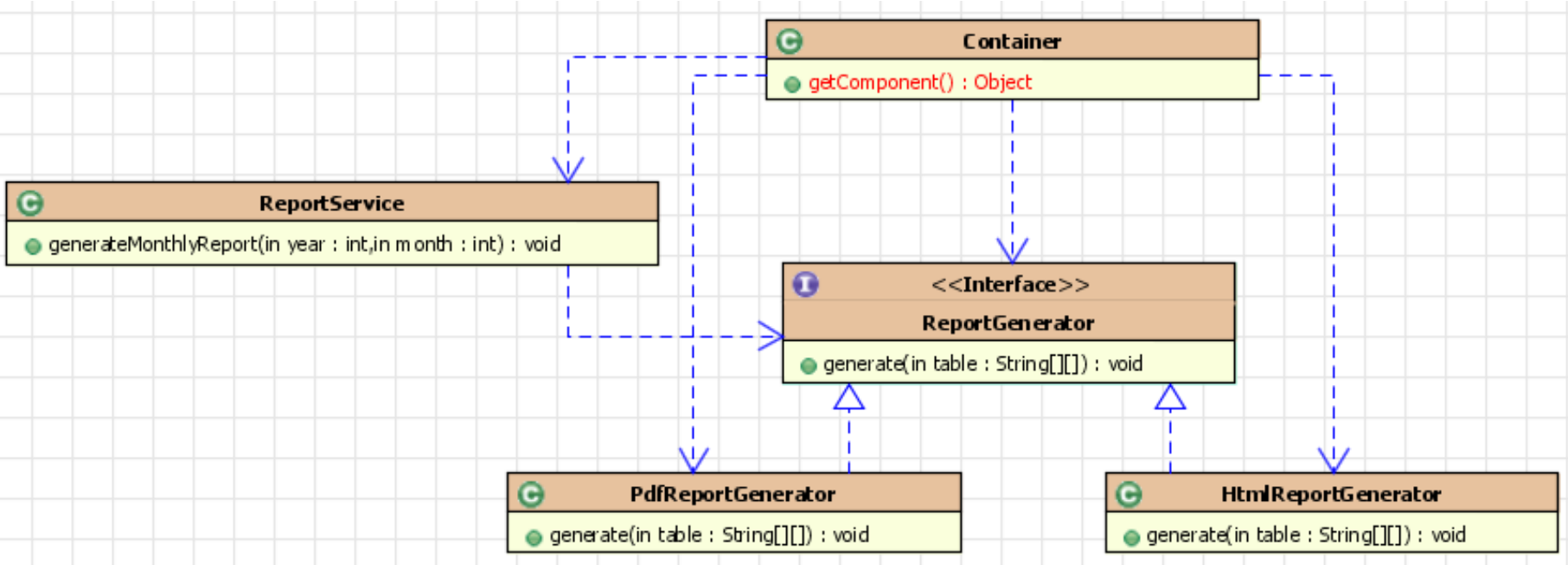


ReportService
generateMonthlyReport(in year : int,in month : int) : void
<<Interface>>
ReportGenerator
generate(in table : String[][]) : void
PdfReportGenerator
generate(in table : String[][]) : void
HtmlReportGenerator
generate(in table : String[][]) : void

IOC □□ --- □□□□□□


ReportGeneratorFactory
getReportGenerator() : ReportGenerator
ReportService
generateMonthlyReport(in year : int,in month : int) : void
<<Interface>>
ReportGenerator
generate(in table : String[][]) : void
PdfReportGenerator
generate(in table : String[][]) : void
HtmlReportGenerator
generate(in table : String[][]) : void

IOC --- □□□□□□


Container
getComponent() : Object
ReportService
generateMonthlyReport(in year : int,in month : int) : void
<<Interface>>
ReportGenerator
generate(in table : String[][]) : void
PdfReportGenerator
generate(in table : String[][]) : void
HtmlReportGenerator
generate(in table : String[][]) : void

□□□□

- IOC & DI □□
- □□ **bean**
  - □□□□□□□□ **XML** □□□□□□□□□□□□□
  - **Bean** □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ & □□□□□□□□□ FactoryBean
  - **IOC □□ BeanFactory & ApplicationContext □□**
  - □□□□□□□□□□□□□□□□□□□□□
  - □□□□□□□□
  - □□□□
  - bean □□□□□□□□□□□□□
  - bean □□□□□ singleton □ prototype □ WEB □□□□□
  - □□□□□□□□□
  - spEL
  - IOC □□□ Bean □□□□□
  - Spring 4.x □□□□□□□□□□□

□ Spring □ IOC □□□□□ Bean

- □ xml □□□□□ bean □□□□□ bean

```
<!-- 通过全类名的方式来配置 bean -->
<bean id="helloWorld"
    class="com.atguigu.spring.helloworld.HelloWorld">
</bean>
```

- id □ Bean □□□□

  – □ **IOC** □□□□□□□□□

  – □ **id** □□□□□ **Spring** □□□□□□□□□□ **Bean** □□□

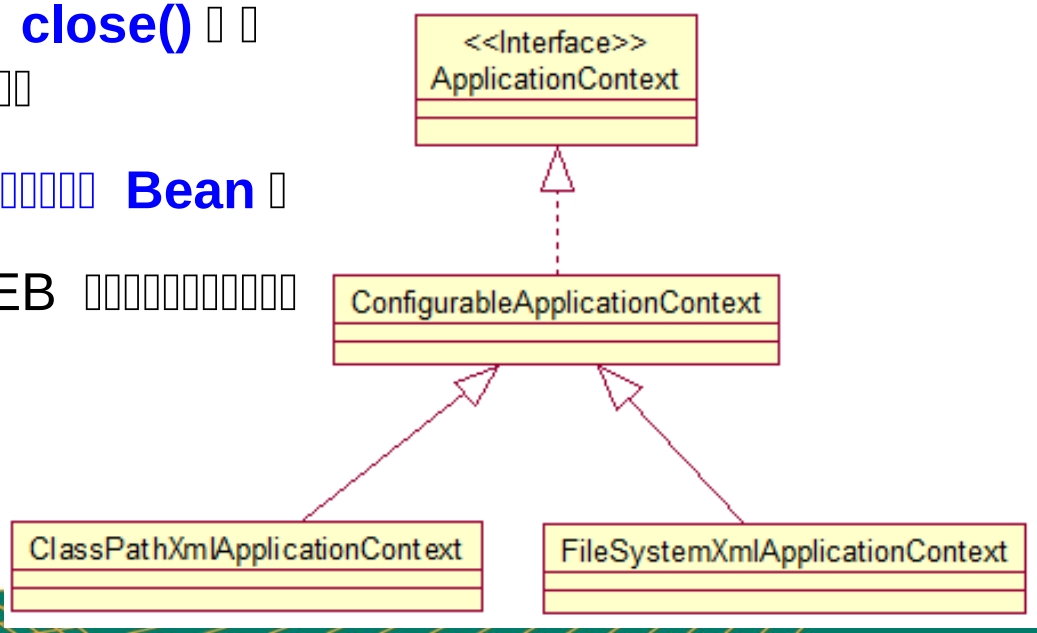
  – id □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Spring □□

- □ **Spring IOC** □□□□ Bean □□□□ Bean □□□□, □□□□□□□□□□. □□□□□□□□□□, □□□□ IOC □□□□□ Bean □□□□□.
- Spring □□□□□□□□□ IOC □□□□.
  - **BeanFactory**: IOC □□□□□□□□.
  - **ApplicationContext**: □□□□□□□□□□□. □ BeanFactory □□□□.
  - BeanFactory □ Spring □□□□□□□□□□□ Spring □□□ ApplicationContext □□□□ Spring □□□□□□□□□□□□□□□□□□□□□□ **ApplicationContext** □□□□□ **BeanFactory**
  - □□□□□□□□□, □□□□□□□□□.

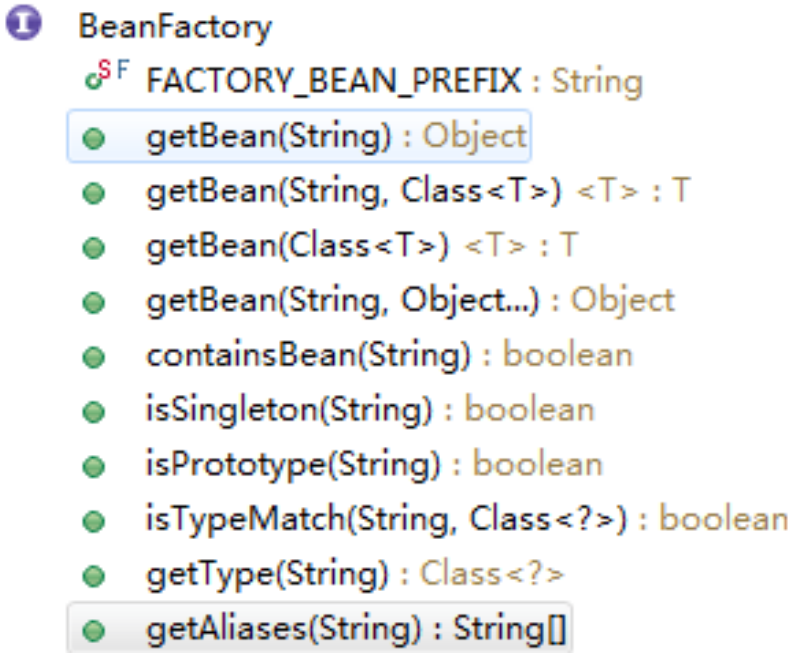
ApplicationContext

- ApplicationContext □□□□□□□□
  - **ClassPathXmlApplicationContext** □□ □□□□□□□□□□□□
  - FileSystemXmlApplicationContext: □□□□□□□□□□□□□□
- ConfigurableApplicationContext □□□ ApplicationContext □□□□□□□□□□□□ refresh() □ **close()** □ □ ApplicationContext □□□□□□□□□□□□□□□□□□
- **ApplicationContext** □□□□□□□□□□□□□□□□□□□ **Bean** □
- WebApplicationContext □□□□ WEB □□□□□□□□□□□□□□□□□ WEB □□□□□□□□□□□□□□□□



IOC Bean

- ApplicationContext getBean()



□□□□□□□

- Spring □□ 3 □□□□□□□□
  - □□□□
  - □□□□□
  - □□□□□□□□□□□□□□□□□

□□□□

- □□□□□□□ **setter** □□□ Bean □□□□□□□□□□
- □□□□□□ <property> □□ , □□ name □□□□ Bean □□□□□□ value □□□ <value> □□□□□□□□
- □□□□□□□□□□□□□□□□□□□□

```
<!-- 通过全类名的方式来配置 bean -->
<bean id="helloWorld"
    class="com.atguigu.spring.helloworld.HelloWorld">
    <property name="userName" value="atguigu"></property>
</bean>
```

□□□□□□

- □□□□□□□□ Bean □□□□□□□□□□□□□□□□ Bean □□□□□□□□□□□□□□□
- □□□□□□ <constructor-arg> □□□□□□□ , **<constructor-arg> □□□ name □□**

是是是是是是

- 是是是是是是是是

```
<bean id="car" class="com.atguigu.spring.helloworld.Car">
    <constructor-arg value="奥迪" index="0"></constructor-arg>
    <constructor-arg value="长春一汽" index="1"></constructor-arg>
    <constructor-arg value="500000" index="2"></constructor-arg>
</bean>
```

- 是

```
<bean id="car" class="com.atguigu.spring.helloworld.Car">
    <constructor-arg value="奥迪" type="java.lang.String"/>
    <constructor-arg value="长春一汽" type="java.lang.String"/>
    <constructor-arg value="500000" type="double"/>
</bean>
```

□□□□

- IOC & DI □□
- □□ **bean**
  - □□□□□□□□ **XML** □□□□□□□□□□□□□
  - Bean □□ & □□□□□□□□□ FactoryBean
  - IOC □□ BeanFactory & ApplicationContext □□
  - □□□□□□□□□□□□□□□□□□□□□
  - □□□□□□□□
  - □□□□
  - bean □□□□□□□□□□□□□
  - bean □□□□□ singleton □ prototype □ WEB □□□□□
  - □□□□□□□□□
  - spEL
  - IOC □□□ Bean □□□□□
  - Spring 4.x □□□□□□□□□□□

□□□

- □□□□□□□□□□□□□□□□□□□□ <value> □□□□□ value □□□□□□□
- □□□□□□□□□□□□ String □□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□ <![CDATA[]]> □□□□□□□□□

□□□□ Bean

- □□□□□□□ Bean □□□□□□□□□□□□□□□□□□□□□□. □**□ Bean □□□□□□**, □□□□ Bean □□□□□□□□ Bean □□□
- □ Bean □□□□□□, □□**□□ <ref> □□□ ref □□**□ Bean □□□□□□□□□□□□□ Bean □□□.
- □□□**□□□□□□□□□□□ Bean □□□**, □□□ Bean □□**□□ Bean**

```
<bean id="service" class="com.atguigu.spring.ioc.ref.Service"></bean>

<bean id="action" class="com.atguigu.spring.ioc.ref.Action">
    <!--
        为 service 属性赋值
        因为 service 属性是一个 bean 类型, 可以使用 ref 指向 ioc 容器中的其他的 bean
    -->
    <property name="service" ref="service"></property>
</bean>
```

□□ Bean

- □ Bean □□□□□□□□□□□□□□□□□□□□□, □□□□□□□□□□ Bean. □□ Bean □□□□□□□□ <property> □ <constructor-arg> □□□, □□□□□□□□ id □ name □□
- □□ Bean □□□□□□□□□□□□

□□□□□□□ null □□□□□□

- □□□□□□□ **<null/>** □□□□□ Bean □□□□□□□□□□□□□□□□ null □
- □ Struts □ Hiberante □□□□□□ **Spring** □□□□□□□□□□

□□□□

- □ Spring □□□□□□□□□□ xml □□ (□□ : <list>, <set> □ <map>) □□□□ □□□ .
- □□ java.util.List □□□□□ , □□□□ **<list>** □□ , □□□□□□□□□□ . □□□□□□□□ **<value>** □□□□□□□□ , □□ **<ref>** □□□□□ Bean □□□ . □□ **<bean>** □□□□ Bean □□ . □□ <null/> □□□□□ . □□□□□□□□□□ .
- □□□□□□ List □□ , □□□ <list>
- □□ java.util.Set □□□□ <set> □□ , □□□□□□□□ List □□ .

□□□□

- Java.util.Map □□ **<map>** □□□□ , <map> □□□□□□□□□ **<entry>** □□□□□ . □□□□□□□□□□□□□ .
- □□□ **<key>** □□□□□□
- □□□□□□□□□□□□ , □□□□□□□□□□□□ **<value>, <ref>, <bean>** □ **<null >** □□ .
- □□□ Map □□□□□□ <entry> □□□□□ : □□□□□□ key □ value □□□ ; Be an □□□□ key-ref □ value-ref □□□□
- □□ **<props>** □□ java.util.Properties, □□□□□□□ **<prop>** □□□□□ . □□ **<prop>** □□□□□□ **key** □□ .

□□ utility scheme □□□□

- □□□□□□□□□□□□□□□□, □□□□□□□□□□□ **Bean** □□, □□□□ **Bean** □□□□□□□□, □□□□□□□ **Bean** □□□□□□□.
- □□□□ util schema □□□□□□□□□□□□□□ Bean. □□□□□□, □□□ <beans> □□□□□□ util schema □□

□□ p □□□□

- □□□□ XML □□□□□□□□□□□ XML □□□□□□□□□□□□□□□□□
- Spring □ 2.5 □□□□□□□□□□□ p □□□□□□□□□ <bean> □□□□□□□□□ Bean □□□□
- □□ p □□□□□□□□ XML □□□□□□□□□□□

□□□□

- IOC & DI □□
- □□ **bean**
  - □□□□□□□□ **XML** □□□□□□□□□□□□□
  - Bean □□ & □□□□□□□□□ FactoryBean
  - IOC □□ BeanFactory & ApplicationContext □□
  - □□□□□□□□□□□□□□□□□□□□□
  - □□□□□□□□
  - □□□□
  - bean □□□□□□□□□□□□□
  - bean □□□□□ singleton □ prototype □ WEB □□□□□
  - □□□□□□□□□
  - spEL
  - IOC □□□ Bean □□□□□
  - Spring 4.x □□□□□□□□□□□

XML □□□□ Bean □□□□

- Spring IOC □□□□□□□□ Bean. □□□□□□□□ **<bean>** □ **autowire** □□ □□□□□□□□□□
- **byType**(□□□□□□□□): □ IOC □□□□□□□□□ Bean □□□□□ Bean. □□□□□□ , Spring □□□□□□□ Bean □□□□□□ , □□□□□□□□□□ .
- **byName**(□□□□□□□□): □□□□□ Bean □□□□□□□□□□□□□□□ .
- constructor(□□□□□□□□□): □ Bean □□□□□□□□□ , □□□□□□□□□□□□□□ . □□□□□

XML [illegible] Bean [illegible]

- [illegible] Bean [illegible] autowire [illegible] Bean [illegible]. [illegible], [illegible], autowire [illegible].
- autowire [illegible], [illegible], [illegible].
- [illegible]

□□□□

- IOC & DI □□
- □□ **bean**
  - □□□□□□□□ **XML** □□□□□□□□□□□□□
  - Bean □□ & □□□□□□□□□ FactoryBean
  - IOC □□ BeanFactory & ApplicationContext □□
  - □□□□□□□□□□□□□□□□□□□□□
  - □□□□□□□□
  - □□□□
  - **bean** □□□□□□□□□□□□□
  - bean □□□□□ singleton □ prototype □ WEB □□□□□
  - □□□□□□□□□
  - spEL
  - IOC □□□ Bean □□□□□
  - Spring 4.x □□□□□□□□□□□

□□ Bean □□

- **Spring □□□□ bean □□□**, □□□□ bean □□□ bean. □□□□□ Bean □ Bean □□□ Bean
- □ **Bean □□ Bean □□□□□, □□ Bean □□□□□**
- □ Bean □□□□□□□ Bean □□□□□□□
- □ Bean □□□□□□□□, □□□□□ Bean □□. □□□□□ **Bean □□□□, □□□□ <bean > □ abstract □□□ true**, □□ Spring □□□□□□□□ Bean
- □□□ **<bean>** □□□□□□□□□□□□□. □□: autowire, abstract □.
- □□□□□□ **Bean □ class □□**, □□ Bean □□□□□□, □□□□□□□□□□. □□□ **abstract □□□□ true**

□□ Bean □□

- **Spring □□□□□□ depends-on □□□□ Bean □□□□□ Bea n** □□□□□□ Bean □□□ Bean □□□□□□□□
- □□□□□□□□□ **Bean** □□□□□□□□□□□□□□□□□ **Bean** □□□

□□□□

- IOC & DI □□
- □□ **bean**
  - □□□□□□□□ **XML** □□□□□□□□□□□□□
  - Bean □□ & □□□□□□□□□ FactoryBean
  - IOC □□ BeanFactory & ApplicationContext □□
  - □□□□□□□□□□□□□□□□□□□□□
  - □□□□□□□□
  - □□□□□
  - bean □□□□□□□□□□□□□
  - **bean** □□□□□□ **singleton** □ **prototype** □ **WEB** □□□□□□
  - □□□□□□□□□□
  - spEL
  - IOC □□□ Bean □□□□□□
  - Spring 4.x □□□□□□□□□□□□

Bean □□□□

- □ Spring □, □□□ <bean> □□□ **scope** □□□□□ Bean □□□□.
- □□□□□, **Spring** □□□□□ **IOC** □□□□□□ **Bean** □□□□□□□□, □□ **IOC** □□□□□□□□□□□□□□ □□□□□□ getBean() □□□ Bean □□□□□□□□□□□□ Bean □□. □□□□□□□□ **singleton**, □□□□ Bean □□□□□□.

类别	说明
singleton	在 SpringIOC 容器中仅存在一个 Bean 实例，Bean 以单实例的方式存在
prototype	每次调用 getBean() 时都会返回一个新的实例
request	每次 HTTP 请求都会创建一个新的 Bean，该作用域仅适用于 WebApplicationContext 环境
session	同一个 HTTP Session 共享一个 Bean，不同的 HTTP Session 使用不同的 Bean。该作用域仅适用于 WebApplicationContext 环境

□□□□

- IOC & DI □□
- □□ **bean**
  - □□□□□□□□ **XML** □□□□□□□□□□□□□
  - Bean □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ & □□□□□□□□ FactoryBean
  - IOC □□ BeanFactory & ApplicationContext □□
  - □□□□□□□□□□□□□□□□□□□□□
  - □□□□□□□□
  - □□□□□
  - bean □□□□□□□□□□□□□
  - bean □□□□□□ singleton □ prototype □ WEB □□□□□□
  - □□□□□□□□□
  - spEL
  - IOC □□□ Bean □□□□□□
  - Spring 4.x □□□□□□□□□□□□

□□□□□□□□

- □□□□□□□□ Bean □, □□□□□ Bean □□□□□□□□□□□□□□ (□□: □□□□, □□□□□□□□□□). □□□□□□□□□□□□□□ Bean □□□□□
- Spring □□□□□ PropertyPlaceholderConfigurer □ **BeanFactory** □□□□□, □□□□□□□□□□ Bean □□□□□□□□□□□□□□□□. □□□ Bean □□□□□□□□□□ **${var}** □□□, PropertyPlaceholderConfigurer □□□□□□□□□□, □□□□□□□□□□□.
- Spring □□□□□□□□□□ ${propName} □□□□□□□□□□□□□□□

是是 PropertyPlaceholderConfigurer

- Spring 2.0:

```
<bean
    class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="location" value="classpath:jdbc.propertiess"></property>
</bean>
```

- **Spring 2.5 轻轻：轻轻轻 <context:property-placeholder> 轻轻轻：**
  - <beans> 是是是 context Schema 是是

```
<context:property-placeholder
    location="classpath:db.properties"/>
```

□□□□

- IOC & DI □□
- □□ **bean**
  - □□□□□□□□ **XML** □□□□□□□□□□□□□
  - Bean □□ & □□□□□□□□□ FactoryBean
  - IOC □□ BeanFactory & ApplicationContext □□
  - □□□□□□□□□□□□□□□□□□□□□
  - □□□□□□□□
  - □□□□
  - bean □□□□□□□□□□□□□
  - bean □□□□□ singleton □ prototype □ WEB □□□□□
  - □□□□□□□□□
  - **SpEL**
  - IOC □□□ Bean □□□□□
  - Spring 4.x □□□□□□□□□□

Spring [illegible] SpEL

- **Spring** [illegible] **SpEL** [illegible]
- [illegible] **EL** [illegible] **SpEL** [illegible] **#{…}** [illegible] **SpEL**
- **SpEL** [illegible] **bean** [illegible]
- [illegible] SpEL [illegible]
  - [illegible] bean [illegible] id [illegible] bean [illegible]
  - [illegible]
  - [illegible]
  - [illegible]

SpEL □□□□

- □□□□□□□□
  - □□□ <property name="count" value="**#{5}**"/>
  - □□□ <property name="frequency" value="**#{89.7}**"/>
  - □□□□□□ <property name="capacity" value="**#{1e4}**"/>
  - **String** □□□□□□□□□□□□□□□□□□□□□□□□□ <property name=“name” value="**#{'Chuck'}**"/> □ <property name='name' value='**#{"Chuck"}**'/>
  - Boolean □ <property name="enabled" value="**#{false}**"/>

SpEL 是是是 Bean 是是是是是是是 1 是

- 轻轻轻轻轻轻轻

```
<!-- 通过 value 属性和 SpEL 配置 Bean 之间的应用关系 -->
<property name="prefix" value="#{prefixGenerator}"></property>
```

-

```
<!-- 通过 value 属性和 SpEL 配置 suffix 属性值为另一个 Bean 的 suffix 属性值 -->
<property name="suffix" value="#{sequenceGenerator2.suffix}"/>
```

```
<!-- 通过 value 属性和 SpEL 配置 suffix 属性值为另一个 Bean 的方法的返回值 -->
```

- `<property name="suffix" value="#{sequenceGenerator2.toString()}"/>`

```
<!-- 方法的连缀 -->
<property name="suffix"
    value="#{sequenceGenerator2.toString().toUpperCase()}"/>
```

SpEL □□□□□□□□ 1 □

- □□□□□□ +, -, *, /, %, ^ □

```
<property name="adjustedAmount" value="#{counter.total + 42}"/>
<property name="adjustedAmount" value="#{counter.total - 20}"/>
<property name="circumference" value="#{2 * T(java.lang.Math).PI * circle.radius}"/>
<property name="average" value="#{counter.total / counter.count}"/>
<property name="remainder" value="#{counter.total % counter.count}"/>
<property name="area" value="#{T(java.lang.Math).PI * circle.radius ^ 2}"/>
```

- □□□□□□□□□□□□□□

```
<constructor-arg
    value="#{performer.firstName + ' ' + performer.lastName}"/>
```

- □□□□□□ <, >, ==, <=, >=, lt, gt, eq, le, ge

```
<property name="equal" value="#{counter.total == 100}"/>
<property name="hasCapacity" value="#{counter.total le 100000}"/>
```

SpEL 是是是是是是是是 2 是

- 轻轻轻轻轻轻轻 **and, or, not, |**

```
<property name="largeCircle" value="#{shape.kind == 'circle' and shape.perimeter gt 10000}"/>
<property name="outOfStock" value="#{!product.available}"/>
<property name="outOfStock" value="#{not product.available}"/>
```

- **if-else □□□□ ?: (ternary), ?: (Elvis)**

```
<constructor-arg
    value="#{songSelector.selectSong()=='Jingle Bells'?piano:' Jingle Bells '}"/>
```

-
```
<constructor-arg
    value="#{kenny.song ?: 'Greensleeves'}"/>
```

```
<constructor-arg
    value="#{admin.email matches '[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4}'}"/>
```

- □□□□□□ **matches**

SpEL □□□ Bean □□□□□□□ 2 □

- □□□□□□□□□□□□□□ **T()** □□□□□□□□□□□□□□□□□□ Class Object □□□□□□□□□□□□□□□□□

```
<property name="initValue"
    value="#{T(java.lang.Math).PI}"></property>
```

□□□□

- IOC & DI □□
- □□ **bean**
  - □□□□□□□□ **XML** □□□□□□□□□□□□□
  - Bean □□ & □□□□□□□□□ FactoryBean
  - IOC □□ BeanFactory & ApplicationContext □□
  - □□□□□□□□□□□□□□□□□□□□□
  - □□□□□□□□
  - □□□□
  - bean □□□□□□□□□□□□□
  - bean □□□□□ singleton □ prototype □ WEB □□□□□
  - □□□□□□□□□
  - spEL
  - **IOC □□□ Bean □□□□□**
  - Spring 4.x □□□□□□□□□□

IOC □□□ Bean □□□□□□□

- **Spring IOC □□□□□□ Bean □□□□□**, Spring □□□ Bean □□□□□□□□□□□□□□□□.
- Spring IOC □□□ Bean □□□□□□□□□□□□□:
  - □□□□□□□□□□□□ Bean □□
  - □ Bean □□□□□□□□□□ Bean □□□
  - □□ **Bean □□□□□□**
  - Bean □□□□□
  - □□□□□□, □□ **Bean □□□□□**
- □ Bean □□□□□□ init-method □ destroy-method □□, □ Bean □□□□□□□□□□.

□□ Bean □□□□□

- **Bean** □□□□□□□□□□□□□□□□□□ **Bean** □□□□□□□.
- **Bean** □□□□□□ **IOC** □□□□□□ **Bean** □□□□□□, □□□□□□. □□□□□□: □□ Bean □□□□□□□□□□□□□□□□ Bean □□□.
- □ Bean □□□□□□□, □□□□ □□. □□□□□□□□□□□, Spring □□□□ Bean □□□□□□□□□□□□□□□□□□□: org.springframework.beans.factory.config **Interface BeanPostProcessor**

Object	**postProcessAfterInitialization**(Object bean, String beanName) Apply this BeanPostProcessor to the given new bean instance *after* any afterPropertiesSet or a custom init-method).
Object	**postProcessBeforeInitialization**(Object bean, String beanName) Apply this BeanPostProcessor to the given new bean instance *before* any afterPropertiesSet or a custom init-method).

□□ Bean □□□□□□ Bean □□□□□

- Spring IOC □□□ Bean □□□□□□□□□□□□□:
  - □□□□□□□□□□□□ Bean □□
  - □ Bean □□□□□□□□□□ Bean □□□
  - **□ Bean □□□□□ Bean □□□□□□ postProcessBeforeInitialization □□**
  - □□ Bean □□□□□□
  - **□ Bean □□□□□ Bean □□□□□□ postProcessAfterInitialization □□**
  - Bean □□□□□
  - □□□□□□, □□ Bean □□□□□

□□□□

- IOC & DI □□
- □□ bean
  - □□□□□□□□ XML □□□□□□□□□□□□□□
  - **Bean** □□□□□□□□□□□□□□□□□□□□□□□□□□□□□ **&** □□□□□□□□□ FactoryBean
  - IOC □□ BeanFactory & ApplicationContext □□
  - □□□□□□□□□□□□□□□□□□□□□□
  - □□□□□□□□□ bean □□□□□□□□□
  - □□□□□
  - bean □□□□□□□□□□□□□□□
  - bean □□□□□□ singleton □ prototype □ WEB □□□□□□
  - □□□□□□□□□
  - spEL
  - IOC □□□ Bean □□□□□□
  - Spring 4.x □□□□□□□□□□□

Bean

- Bean .
- Bean, Bean **class** , **factory-method** . , **<constructor-arg>** .

[illegible] Bean

- [illegible]: [illegible]. [illegible], [illegible] [illegible].
- [illegible] Bean
  - [illegible] bean [illegible] **factory-bean** [illegible] Bean
  - [illegible] **factory-method** [illegible]
  - [illegible] **construtor-arg** [illegible]

□□□□

- IOC & DI □□
- □□ bean
  - □□□□□□□□ XML □□□□□□□□□□□□□□
  - **Bean** □□□□□□□□□□□□□□□□□□□□□□□□□□□□□ & □□□□□□□□□ **FactoryBean**
  - IOC □□ BeanFactory & ApplicationContext □□
  - □□□□□□□□□□□□□□□□□□□□□□
  - □□□□□□□□
  - □□□□□
  - bean □□□□□□□□□□□□□□□
  - bean □□□□□□ singleton □ prototype □ WEB □□□□□□
  - □□□□□□□□□□
  - spEL
  - IOC □□□ Bean □□□□□□
  - Spring 4.x □□□□□□□□□□□□

□□ FactoryBean □□□ Spring IOC □□□□□ Bean

- Spring □□□□□□□ Bean, □□□□□ Bean, □□□□□□ Bean, □ FactoryBean.
- □□ Bean □□□ Bean □□, □□□□□□□□□□□□□□□□□, □□□□□□□□ Bean □ getObject □□□□□□□□

```
public interface FactoryBean {
  //FactoryBean 返回的实例
  Object getObject() throws Exception;

  //FactoryBean 返回的类型
  Class getObjectType();

  //FactoryBean 返回的实例是否为单例
  boolean isSingleton();
}
```

[illegible]

- IOC & DI [illegible]
- [illegible] bean
  - [illegible] XML [illegible] **Bean** [illegible] **Bean** [illegible]
  - Bean [illegible] & [illegible] FactoryBean
  - IOC [illegible] BeanFactory & ApplicationContext [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - bean [illegible]
  - bean [illegible] singleton [illegible] prototype [illegible] WEB [illegible]
  - [illegible]
  - spEL
  - IOC [illegible] Bean [illegible]
  - Spring 4 [illegible]

□ classpath □□□□□

- □□□□ (component scanning): Spring □□□ classpath □□□ □□ , □□□□□□□□□□□□□□□□ .
- □□□□□□ :
  - @Component: □□□□ , □□□□□□ Spring □□□□□
  - @Respository: □□□□□□□
  - @Service: □□□□□ (□□□) □□
  - @Controller: □□□□□□□
- □□□□□□□□ , **Spring** □□□□□□□□ : □□□□□□□ , □□□□□□□ **.** □□□□□□□□□ **valu e** □□□□□□□□□□

□ classpath □□□□□

- □□□□□□□□□□□□□□□□□, □□□□ Spring □□□□□□□□ **<context:component-scan>** □
  - **base-package** □□□□□□□□□□□□□□□ **Spring** □□□□□□□□□□□□□□□□□□□□□□□□□□□ **.**
  - □□□□□□□□□□ **,** □□□□□□□□□ .
  - □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ resource-pattern □□□□□□□□□□□□□□

```
<context:component-scan
    base-package="com.atguigu.spring.beans"
    resource-pattern="autowire/*.class"/>
```

  - **<context:include-filter>** □□□□□□□□□□□□□□
  - **<context:exclude-filter>** □□□□□□□□□□□□□□□□
  - <context:component-scan> □□□□□□□□□ <context:include-filter> □ <context:exclude-filter> □□□

□ classpath □□□□□

- <context:include-filter> □ <context:exclude-filter> □□□□□□□□□□□□□□□□

类别	示例	说明
annotation	com.atguigu.XxxAnnotation	所有标注了 XxxAnnotation 的类。该类型采用目标类是否标注了某个注解进行过滤
assinable	com.atguigu.XxxService	所有继承或扩展 XxxService 的类。该类型采用目标类是否继承或扩展某个特定类进行过滤
aspectj	com.atguigu..*Service+	所有类名以 Service 结束的类及继承或扩展它们的类。该类型采用 AspejctJ 表达式进行过滤
regex	com.\atguigu\.anno\..*	所有 com.atguigu.anno 包下的类。该类型采用正则表达式根据类的类名进行过滤
custom	com.atguigu.XxxTypeFilter	采用 XxxTypeFilter 通过代码的方式定义过滤规则。该类必须实现 org.springframework.core.type.TypeFilter 接口

□□□□

- <context:component-scan> □□□□□□□□ AutowiredAnnotationBeanPostProcessor □□ , □□□□□□□□□□□□ **@Autowired** □ **@Resource** □ **@Inject** □□□□□ .

@Autowired

- @Autowired
  - , (public), @Authwired
  - , @Authwired . Spring Bean , , , @Authwired required false
  - , IOC Bean , . @Qualifier Bean . Spring @Qualifiter Bean
  - @Authwired , Spring Bean .
  - @Authwired , Spring , Bean.
  - @Authwired java.util.Map , Map String, Spring M ap Bean, Bean

@Resource 和 @Inject

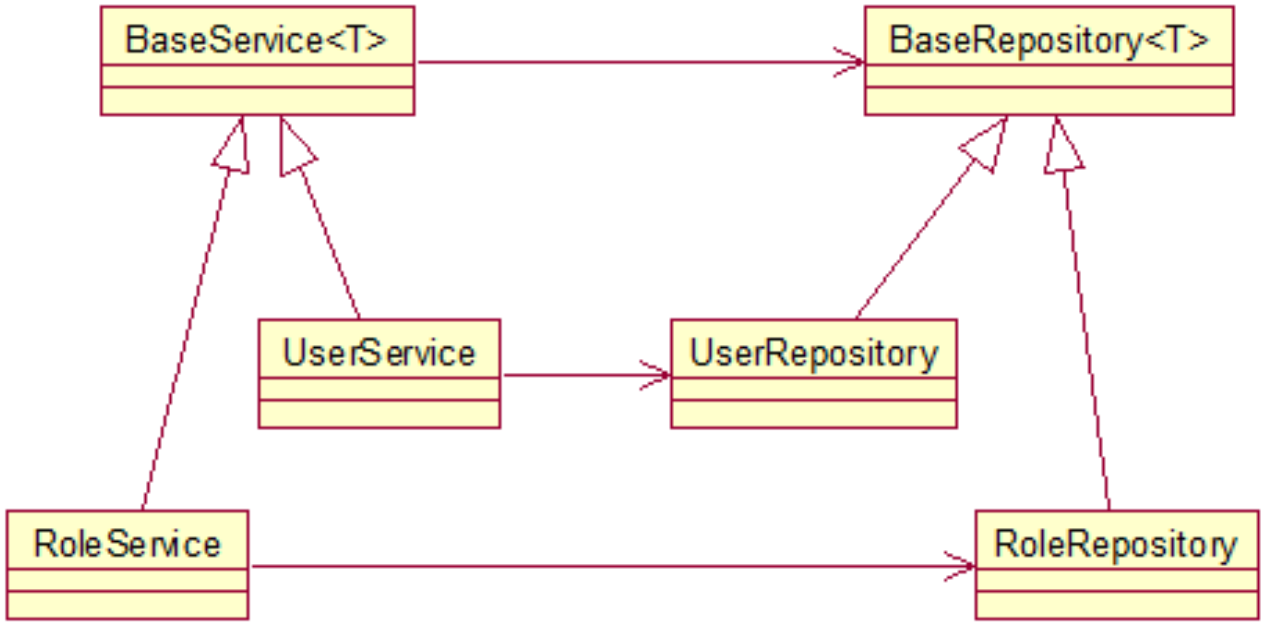
- Spring 还支持 @Resource 和 @Inject 注解，这两个注解和 @Autowired 注解功能类似。
- @Resource 注解要求提供一个 Bean 名称的属性，若属性为空，则自动采用标注处的变量或方法名作为 Bean 的名称
- @Inject 和 @Autowired 注解一样也按类型匹配注入的 Bean，但没有 reqired 属性
- 建议使用 **@Autowired** 注解

[illegible]

- IOC & DI [illegible]
- [illegible] bean
  - [illegible] XML [illegible]
  - Bean [illegible] & [illegible] FactoryBean
  - IOC [illegible] BeanFactory & ApplicationContext [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - bean [illegible]
  - bean [illegible] singleton [illegible] prototype [illegible] WEB [illegible]
  - [illegible]
  - spEL
  - IOC [illegible] Bean [illegible]
  - **Spring 4.x [illegible]**

□□□□□□

- Spring 4.x □□□□□□□□□□□□□□□□□□□□□□□□□□□□


BaseService<T>
BaseRepository<T>
UserService
UserRepository
RoleService
RoleRepository

□□□□□□□□

- Spring □□□□ <import> □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Spring □□□ □□□□□□□□□□□□□□□□□□□□

- import □□□ resource □□□□ Spring □□□□□□□□

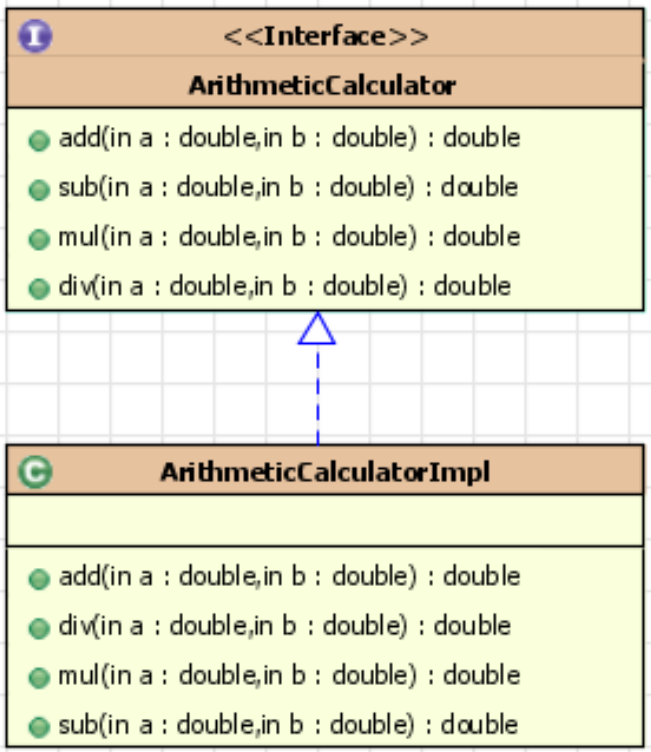
地址前缀	示例	对应资源类型
classpath:	classpath:spring-mvc.xml	从类路径下加载资源，classpath: 和 classpath:/ 是等价的
file:	file:/conf/security/spring-shiro.xml	从文件系统目录中装载资源，可采用绝对或相对路径
http://	http://www.atguigu.com/resource/beans.xml	从 WEB 服务器中加载资源
ftp://	ftp://www.atguigu.com/resource/beans.xml	从 FTP 服务器中加载资源

Spring AOP

讲讲：讲讲
讲讲讲讲：讲讲讲 - 讲讲

AOP □□

- WHY AOP □



□□ 1- □□□□□□□□□□□□□□□□□□□□□□□□

□□ 2- □□□□□□□□□□□□□□□□□□□□

□□□□□□

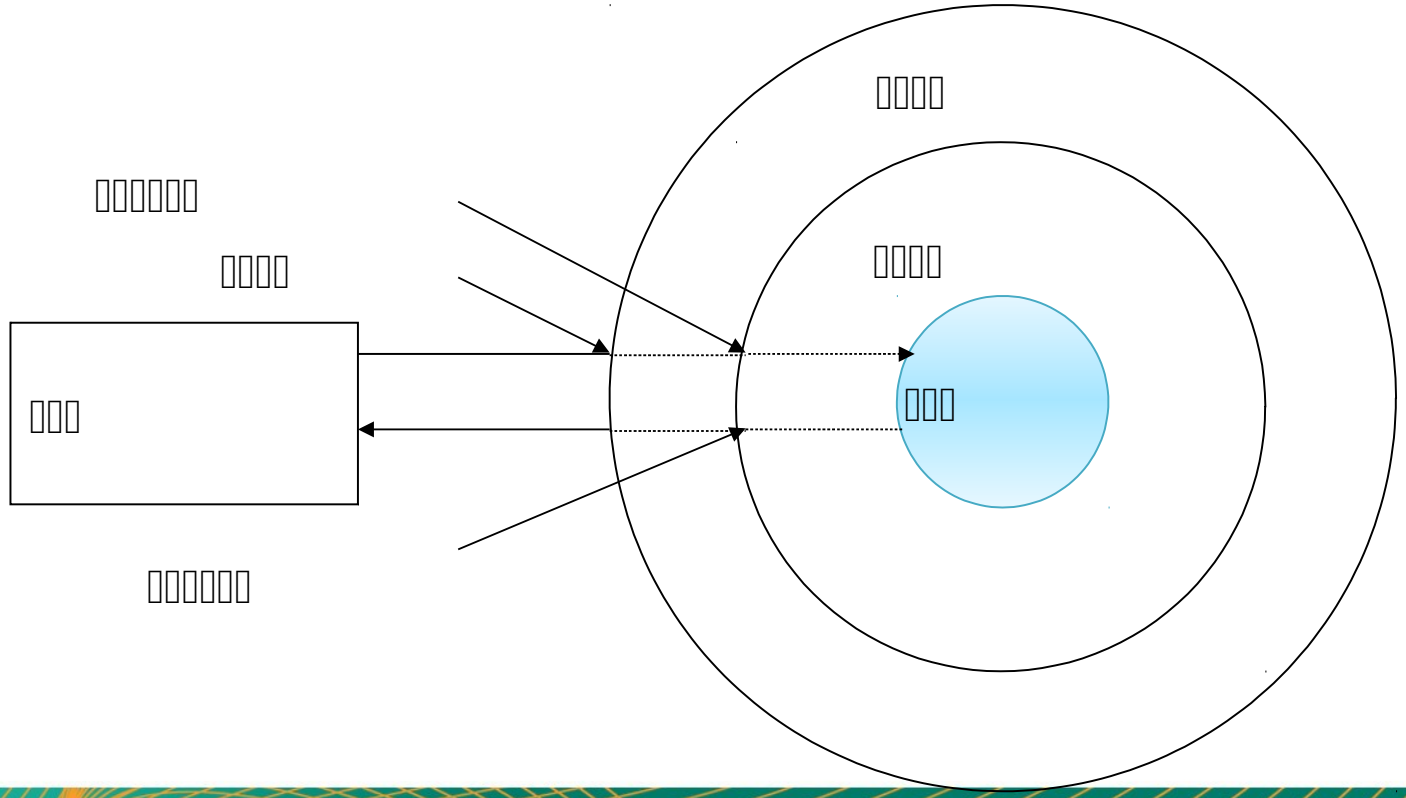
```
public class ArithmeticCalculatorImpl implements ArithmeticCalculator {

    @Override
    public void add(int i, int j) {

        System.out.println("日志:The method add begins with [" +
                i + ", " +  j + "]" );

        int result = i + j;
        System.out.println("result: " + result);

        System.out.println("日志:The method add ends with " + result);
    }

    @Override
    public void sub(int i, int j) {

        System.out.println("日志:The method sub begins with [" +
                i + ", " +  j + "]" );

        int result = i - j;
        System.out.println("result: " + result);

        System.out.println("日志:The method sub ends with " + result);
    }
```

[illegible]

- [illegible] ([illegible]) [illegible], [illegible]. [illegible] [illegible].
- [illegible]: [illegible], [illegible], [illegible]. [illegible], [illegible].

□□□□□□□□□□□□□

- □□□□□□□□□□：□□□□□□□□□□□□□□，□□□□□□□□□□□□□□□□．□□□□□□□□□□□□□□□□□□．□□□．



CalculatorLoggingHandler

```
public class CalculatorLoggingHandler implements InvocationHandler {

    private Log log = LogFactory.getLog(this.getClass());

    private Object target;

    public CalculatorLoggingHandler(Object target) {
        super();
        this.target = target;
    }

    public Object invoke(Object proxy, Method method, Object[] args)
            throws Throwable {
        log.info("The method " + method.getName() + "() begins with " + Arrays.toString(args));
        Object result = method.invoke(target, args);
        log.info("The method " + method.getName() + "() ends with " + result);
        return result;
    }

    public static Object createProxy(Object target){
        return Proxy.newProxyInstance(target.getClass().getClassLoader(),
                target.getClass().getInterfaces(),
                new CalculatorLoggingHandler(target));
    }

}
```

CalculatorValidationHandler

```
public class CalculatorValidationHandler implements InvocationHandler {
    private Object target;

    public CalculatorValidationHandler(Object target) {
        this.target = target;
    }

    public Object invoke(Object proxy, Method method, Object[] args)
            throws Throwable {
        for(Object arg : args){
            validate((Double)arg);
        }
        Object result = method.invoke(target, args);
        return result;
    }

    public static Object createProxy(Object target){
        return Proxy.newProxyInstance(target.getClass().getClassLoader(),
                target.getClass().getInterfaces(),
                new CalculatorValidationHandler(target));
    }

    private void validate(double a){
        if(a < 0)
            throw new IllegalArgumentException("Positive numbers only");
    }

}
```

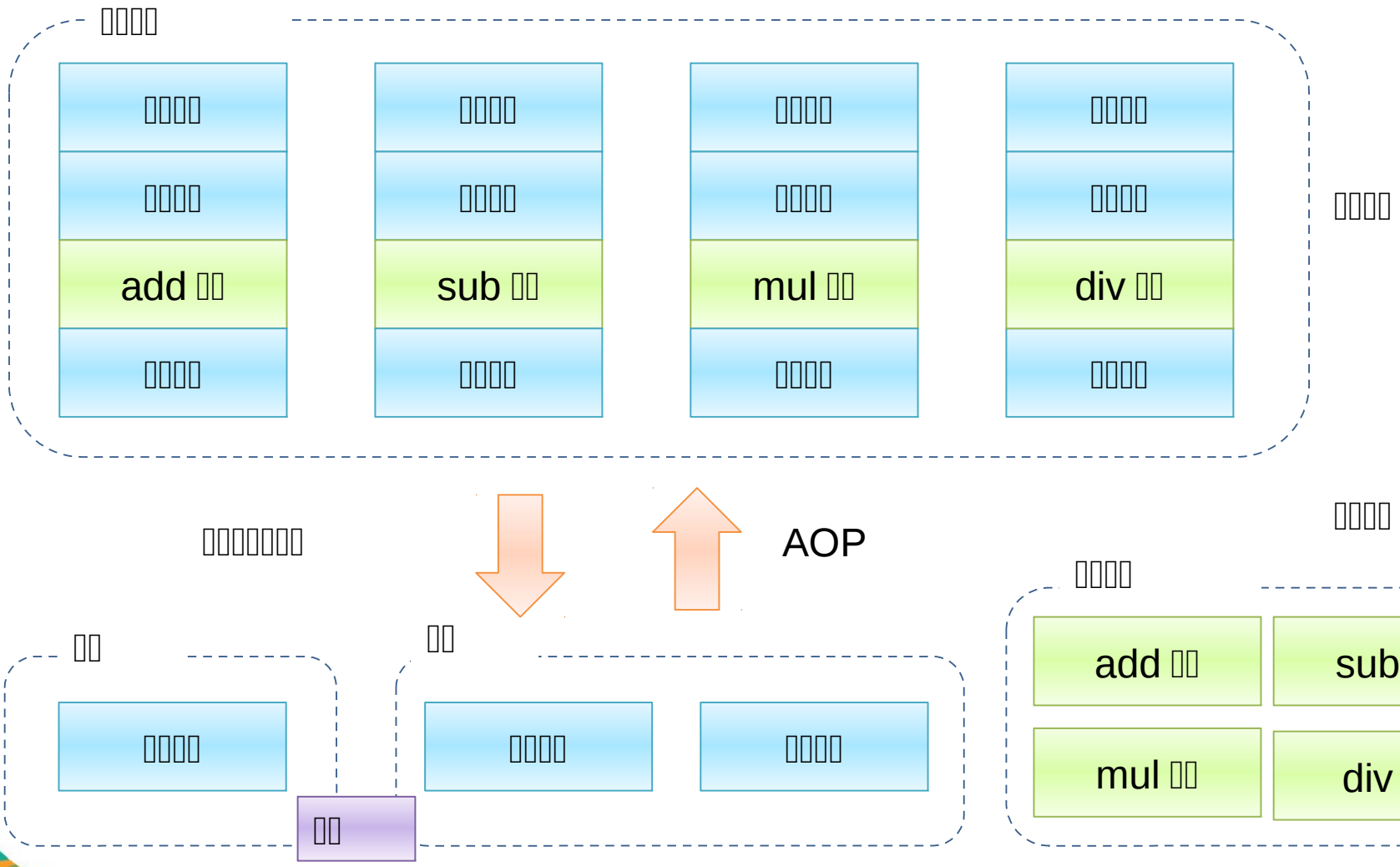
□□□□

```
public class Main {
    public static void main(String[] args) {
        ArithmeticCalculator arithmeticCalculatorImpl =
            new ArithmeticCalculatorImpl();

        ArithmeticCalculator arithmeticCalculator
            = (ArithmeticCalculator) CalculatorValidationHandler
                .createProxy(CalculatorLoggingHandler
                    .createProxy(arithmeticCalculatorImpl));
        System.out.println(arithmeticCalculator.add(-12, 13));
    }
}
```

AOP □□

- AOP(Aspect-Oriented Programming, □□□□□□): □□□□□□□□ , □ □□□ OOP(Object-Oriented Programming, □□□□□□) □□□ .
- AOP □□□□□□□□□□ (aspect), □□□□□□□□□□□ .
- □□□ AOP □□□ , □□□□□□□□□ , □□□□□□□□□□□□□□□□ , □□□□□□□ , □□□□□□□□□□ . □□□□□□□□□□□□□□□□□□□□□□ (□□) □ .
- AOP □□□ :
  - □□□□□□□□□□□□ , □□□□□ , □□□□□□□
  - □□□□□□□ , □□□□□□□□□ .

AOP



AOP [illegible]

- [illegible] (Aspect): [illegible] ([illegible]) [illegible]
- [illegible] (Advice): [illegible]
- [illegible] (Target): [illegible]
- [illegible] (Proxy): [illegible]
- [illegible] Joinpoint [illegible] ArithmethicCalculator#add() [illegible] ArithmethicCalculator#add() [illegible]
- [illegible] pointcut [illegible] ArithmethicCalculator [illegible] **AOP** [illegible] org.springframework.aop.Pointcut [illegible]

Spring AOP

- **AspectJ** □ Java □□□□□□□□□□ AOP □□ .
- □ Spring2.0 □□□□□ , □□□□□□ AspectJ □□□□□ XML □□□ AOP

□ Spring □□□ AspectJ □□□□

- □□ Spring □□□□□ AspectJ □□ , □□□ **classpath □□□ Aspect J □□** : aopalliance.jar □ aspectj.weaver.jar □ spring-aspects.jar
- □ **aop Schema □□□ <beans> □□□□** .
- □□ Spring IOC □□□□□ AspectJ □□□□ , □□□ **Bean □□□□□□□□□□□□ XML □□ <aop:aspectj-autoproxy>**
- □ Spring IOC □□□□□ Bean □□□□□□ <aop:aspectj-autoproxy> □□□ , □□□□□ AspectJ □□□□□ Bean □□□□ .

□ AspectJ □□□□□□

- □□ **Spring** □□□ **AspectJ** □□ , □□□□ **IOC** □□□□□□□□□ **Bean** □□ . □□ Spring IOC □□□□□□ AspectJ □□□□ , Spring IOC □□□□□□□ □ AspectJ □□□□□□ Bean □□□□ .
- □ **AspectJ** □□□ , □□□□□□□□ **@Aspect** □□□ **Java** □ .
- □□□□□□□□□□□□□□□ **Java** □□ .
- AspectJ □□ 5 □□□□□□□□ :
  - **@Before:** □□□□ , □□□□□□□□□
  - **@After:** □□□□ , □□□□□□□□□
  - **@AfterRunning:** □□□□ , □□□□□□□□□□□
  - **@AfterThrowing:** □□□□ , □□□□□□□□□
  - **@Around:** □□□□ , □□□□□□□

□□□□

- □□□□ : □□□□□□□□□□□□□□
- □□□□□□□ @Before □□ , □□□□□□□□□□□□□□□□□ .

□□□□□□□□□□□

```
@Aspect
public class CalculatorLoggingAspect {
    private Log log = LogFactory.getLog(this.getClass());

    @Before("execution(* ArithmeticCalculator.add(..))")
    public void logBefore(){
        log.info("The method add() begins");
    }
}
```

□□□□□□□□□□□□□ , □□□□□□□□□□ ArithmeticCalculator

□□□ add() □□ . * □□□□□□□□□□□□□□□□□ , □□□□□□□ ..

□□□□□□□□□□

□□□□□□□□ AspectJ □□□□□□

- □□□□□□□□□□□□□□□□□□□□ :
  - execution ***** com.atguigu.spring.ArithmeticCalculator.*****(..): □□ ArithmeticCalculator □□□□□□□□, □□□ ***** □□□□□□□□□□□□□. □□□ ***** □□□□□□. **..** □□□□□□□□□. □□□□□□□□□□□□□□□□□, □□□□□□.
  - execution **public** * ArithmeticCalculator.*(..): □□ ArithmeticCalculator □□□□□□□□□.
  - execution **public** double ArithmeticCalculator.*(..): □□ ArithmeticCalculator □□□ **double** □□□□□□□
  - execution **public** double ArithmeticCalculator.*(**double**, ..): □□□□□□□□ double □□□□□, .. □□□□□□□□□□□□□
  - execution **public** double ArithmeticCalculator.*(**double**, **double**): □□□□□□□ double, double □□□□□.

□□□□□□□□

- □ AspectJ □, □□□□□□□□□□□□□□ &&, ||, ! □□□□.

```
@Pointcut("execution(* *.add(int, ..)) || execution(* *.sub(int, ..))")
private void loggingOperation(){}

@Before("loggingOperation()")
public void logBefore(JoinPoint joinPoint){
    log.info("The method " + joinPoint.getSignature().getName()
            + "() begins with " + Arrays.toString(joinPoint.getArgs()));
}
```

□□□□□□□□□□□□□□□

- □□□□□□□□□□□□□□□□ JoinPoint □□□ . □□□□□□□□□□ . □□□□□□□□□ .

```
@Aspect
public class CalculatorLoggingAspect {
    private Log log = LogFactory.getLog(this.getClass());

    @Before("execution(* *.*(..))")
    public void logBefore(JoinPoint joinPoint){
        log.info("The method " + joinPoint.getSignature().getName()
                + "() begins with " + Arrays.toString(joinPoint.getArgs()));
    }
}
```

□□□□□□□□□□□□□□ , □□□□□□□□□□□□□□□□□□□□□□ . □

□□ * □□□□□□□□□□□□□□□□□ , □□□ * □□□□□□□□□ ,

□□□ * □□□□□□□ , □□□□□□□ .. □□□□□□□□□□

□□□□

- □□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□.
- □□□□□□□□□□□□□□□□.

```
@Aspect
public class CalculatorLoggingAspect {
    private Log log = LogFactory.getLog(this.getClass());

    @Before("execution(* *.*(..))")
    public void logBefore(JoinPoint joinPoint){
        log.info("The method " + joinPoint.getSignature().getName()
                + "() begins with " + Arrays.toString(joinPoint.getArgs()));
    }

    @After("execution(* *.*(..))")
    public void logAfter(JoinPoint joinPoint){
        log.info("The method " + joinPoint.getSignature().getName()
                + "() ends");
    }
}
```

[illegible]

- [illegible], [illegible]. [illegible], [illegible].

```java
@Aspect
public class CalculatorLoggingAspect {
    private Log log = LogFactory.getLog(this.getClass());

    @Before("execution(* *.*(..))")
    public void logBefore(JoinPoint joinPoint){
        log.info("The method " + joinPoint.getSignature().getName()
                + "() begins with " + Arrays.toString(joinPoint.getArgs()));
    }

    @AfterReturning("execution(* *.*(..))")
    public void logAfterReturning(JoinPoint joinPoint){
        log.info("The method " + joinPoint.getSignature().getName()
                + "() ends");
    }
}
```

□□□□□□□□□□□□□□□□□

- □□□□□□, □□□ **returning** □□□□□ @AfterReturning □□□, □□□□□□ □□□□□□. □□□□□□□□□□□□□□□□□□□□.
- □□□□□□□□□□□□□□□□□□□□. □□□□, Spring AOP □□□□□□□□□□□□.
- □□□□□□□□□□□□□□□ **pointcut** □□□

```
@AfterReturning(pointcut="execution(* *.*(..))", returning="result")
public void logAfterReturning(JoinPoint joinPoint, Object result){
    log.info("The method " + joinPoint.getSignature().getName()
            + "() ends with " + result);
}
```

□□□□

- □□□□□□□□□□□□□□□□□□

- □ **throwing** □□□□□ **@AfterThrowing** □□□, □□□□□□□□□□□□□□. Th rowable □□□□□□□□□□□□. □□□□□□□□□□□□□□□□□□□□□□□□.

- □□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□. □□□□□□□□□□□□□□□□□□□□□□□□□□□□.

```
@AfterThrowing(pointcut="execution(* *.*(..))", throwing="e")
public void logAfterThrowing(JoinPoint joinPoint, ArithmeticException e){
    log.info("An exception " + e + " has been throwing in "
            + joinPoint.getSignature().getName() + "()");
}
```

□□□□

- □□□□□□□□□□□□□□□□□□□□□□, □□□□□□□□□□□. □□□□□□□□□□□□□□□.
- □□□□□□□□□, □□□□□□□□□□□□ **ProceedingJoinPoint**. □□ JoinPoint □□□□, □□□□□□□□□, □□□□□□□□.
- □□□□□□□□□□□□□ **ProceedingJoinPoint** □ **proceed()** □□□□□□□□□□□□ **.** □□□□□□□□□□□□□□□□□□□□□, □□□□□□□□□□□.
- □□: □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□, □□□ **joinPoint.proceed();** □□□□, □□□□□□□□□□

□□□□□□□□

```
@Around("execution(* *.*(..))")
public void logAround(ProceedingJoinPoint joinPoint) throws Throwable{
    log.info("The method " + joinPoint.getSignature().getName()
            + "() begins with " + Arrays.toString(joinPoint.getArgs()));

    try {
        joinPoint.proceed();
        log.info("The method " + joinPoint.getSignature().getName()
                + "() ends");
    } catch (Throwable e) {
        log.info("An exception " + e + " has been throwing in "
                + joinPoint.getSignature().getName() + "()");
        throw e;
    }
}
```

[illegible]

- [illegible], [illegible], [illegible].
- [illegible] Ordered [illegible] @Order [illegible].
- [illegible] Ordered [illegible], getOrder() [illegible], [illegible].
- [illegible] @Order [illegible], [illegible]

```
@Aspect
@Order(0)
public class CalculaotorValidationAspect {
```

```
@Aspect
@Order(1)
public class CalculatorLoggingAspect {
```

[illegible]

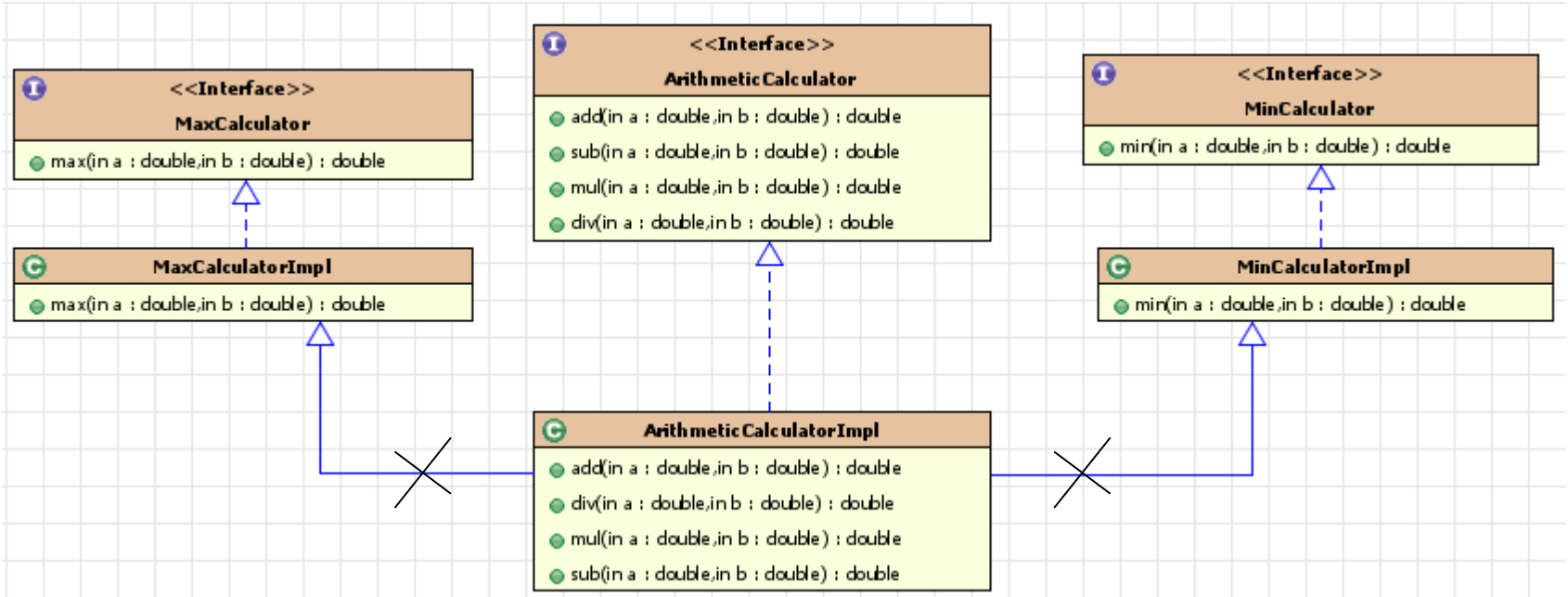
- [illegible] AspectJ [illegible], [illegible]. [illegible].
- [illegible] AspectJ [illegible], [illegible] **@Pointcut** [illegible]. [illegible], [illegible].
- [illegible]. [illegible], [illegible]. [illegible], [illegible] public. [illegible], [illegible]. [illegible], [illegible].
- [illegible].

□□□□□□□□□□□

```
@Pointcut("execution(* *.*(..))")
private void loggingOperation(){}

@Before("loggingOperation()")
public void logBefore(JoinPoint joinPoint){
    log.info("The method " + joinPoint.getSignature().getName()
            + "() begins with " + Arrays.toString(joinPoint.getArgs()));
}

@AfterReturning(pointcut="loggingOperation()", returning="result")
public void logAfterReturning(JoinPoint joinPoint, Object result){
    log.info("The method " + joinPoint.getSignature().getName()
            + "() ends with " + result);
}

@AfterThrowing(pointcut="loggingOperation()", throwing="e")
public void logAfterThrowing(JoinPoint joinPoint, ArithmeticException e){
    log.info("An exception " + e + " has been throwing in "
            + joinPoint.getSignature().getName() + "()");
}
```

□□□□

- □□□□□□□□□□□□□□□□. □□□□□□□□□□□□, □□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□□□□□□□.


<<Interface>>
MaxCalculator
max(in a : double,in b : double) : double
<<Interface>>
ArithmeticCalculator
add(in a : double,in b : double) : double
sub(in a : double,in b : double) : double
mul(in a : double,in b : double) : double
div(in a : double,in b : double) : double
<<Interface>>
MinCalculator
min(in a : double,in b : double) : double
MaxCalculatorImpl
max(in a : double,in b : double) : double
MinCalculatorImpl
min(in a : double,in b : double) : double
ArithmeticCalculatorImpl
add(in a : double,in b : double) : double
div(in a : double,in b : double) : double
mul(in a : double,in b : double) : double
sub(in a : double,in b : double) : double

□□□□

- □□□□□□□□□□□□□ MaxCalculatorImpl □ MinCalculatorImpl, □ ArithmeticCalculatorImpl □□□□□ MaxCalculator □ MinCalculator □□. □□□□ MaxCalculatorImpl □ MinCalculatorImpl □□□□□□□□□□□. □□□□□□□ ArithmeticCalculatorImpl □□□□
- □□□□□□□□□□□□□
- □□□□, □□□□□□□□ **@DeclareParents** □□□□□□□.
- □□□□□ **value** □□□□□□□□□□□□□□□□. value □□□□□□□□ AspectJ □□□□□□, □□□□□□□□□□□□□. **defaultImpl** □□□□□□□□□□□□□□□

□□□□□□□□

```
@Aspect
public class CalculatorLoggingAspect implements Ordered{
    private Log log = LogFactory.getLog(this.getClass());

    @DeclareParents(value="* *.Arithmetic*", defaultImpl=MaxCalculatorImpl.class)
    private MaxCalculator maxCalculator;

    @DeclareParents(value="* *.Arithmetic*", defaultImpl=MinCalculatorImpl.class)
    private MinCalculator minCalculator;
```

```
MinCalculator minCalculator = (MinCalculator)
    ctx.getBean("airthmeticCalculator");
minCalculator.min(1, 2);
```

□□□ XML □□□□□□□

- □□□□ AspectJ □□□□□□, Spring □□□□ Bean □□□□□□□□□. □□□□□□□ aop schema □□ XML □□□□□.
- □□□□□, **□□□□□□□□□□□□□ XML □□□.** □□ AspectJ □□, □□□□□ AspectJ □□, □□□ XML □□□□□ Spring □□□. □□ AspectJ □□□□□□□ AOP □□□□, □□□□□□□□□□□□□□□□□□□□□□□□□.

□□ XML ---- □□□□

- □□□ XML □□□□□, □□□ <beans> □□□□□□ aop Schema
- □ Bean □□□□□, □□□ Spring AOP □□□□□□□□ **<aop:config>** □□□□. □□□□□□□□, □□□□□□ **<aop:aspect>** □□□□□□□□□□□□□□□□ Bean □□.
- □□ Bean □□□□□□□□, □ <aop:aspect> □□□□

□□□□□□□□□

```
<bean id="calculatorLoggingAspect"
    class="org.simpleit.CalculatorLoggingAspect"></bean>

<bean id="calculatorValidationAspect"
    class="org.simpleit.CalculaotorValidationAspect"></bean>

<aop:config>
    <aop:aspect id="loggingAspect"
        ref="calculatorLoggingAspect"></aop:aspect>

    <aop:aspect id="validationAspect"
        ref="calculatorValidationAspect"></aop:aspect>
</aop:config>
```

□□ XML ---- □□□□□

- □□□□□ **<aop:pointcut>** □□□□
- □□□□□□□□□ <aop:aspect> □□□ , □□□□□□□□ <aop:config> □□□ .
  - □□□ <aop:aspect> □□□ : □□□□□□□□□
  - □□□ <aop:config> □□□ : □□□□□□□□□
- □□ XML □ AOP □□□□□□□□□□□□□□□□□□□□□□□□□□□□□ .

是是是是是是是是是是

```
<aop:config>
    <aop:pointcut id="testOperation"
        expression="execution(* org.simpleit.bean.Arithmetic*.*(..))"/>

    <aop:aspect id="loggingAspect"
        ref="calculatorLoggingAspect">
    </aop:aspect>

    <aop:aspect id="validationAspect"
        ref="calculatorValidationAspect">
    </aop:aspect>
</aop:config>
```

□□ XML ---- □□□□

- □ aop Schema □, □□□□□□□□□□□□□□□ XML □□.
- □□□□□□□□ <pointcut-ref> □□□□□□, □□ <pointcut> □□□□□□□□□□. method □□□□□□□□□□□□□□□.

□□□□□□□□

```
<aop:config>
    <aop:pointcut id="testOperation"
        expression="execution(* org.simpleit.bean.Arithmetic*.*(..))"/>

    <aop:aspect id="loggingAspect"
        ref="calculatorLoggingAspect">
        <aop:after method="logBefore"
            pointcut-ref="testOperation"/>
    </aop:aspect>

    <aop:aspect id="validationAspect"
        ref="calculatorValidationAspect">
        <aop:before method="validateBefore"
            pointcut-ref="testOperation"/>
    </aop:aspect>
</aop:config>
```

是是是是

- 是是是是 <aop:declare-parents> 是是是是是是是是是是是是是是

```
<aop:aspect id="loggingAspect"
    ref="calculatorLoggingAspect">
    <aop:after method="logBefore"
        pointcut-ref="testOperation"/>

    <aop:declare-parents
        types-matching="org.simpleit.bean.Arithmetic*"
        implement-interface="org.simpleit.bean.MinCalculator"
        default-impl="org.simpleit.bean.MinCalculatorImpl"/>

</aop:aspect>
```

Spring 与 JDBC

JdbcTemplate [illegible]

- [illegible] JDBC [illegible], Spring [illegible] JDBC API [illegible], [illegible] JDBC [illegible].
- [illegible] Spring JDBC [illegible], **JDBC** [illegible] JDBC [illegible]. [illegible], [illegible]. [illegible], [illegible], [illegible].

JdbcTemplate

- sql :

update

```
public int update(String sql,
                  Object... args)
           throws DataAccessException
```

- **batchUpdate**

```
public int[] batchUpdate(String sql,
                         List<Object[]> batchArgs)
```

是是 JdbcTemplate 是是是是是

- 是是是是：

queryForObject

```
public <T> T queryForObject(String sql,
                            ParameterizedRowMapper<T> rm,
                            Object... args)
                     throws DataAccessException
```

- org.springframework.jdbc.core.simple

Class ParameterizedBeanPropertyRowMapper<T>

```
java.lang.Object
  └ org.springframework.jdbc.core.BeanPropertyRowMapper
      └ org.springframework.jdbc.core.simple.ParameterizedBeanPropertyRowMapper<T>
```

是是 JdbcTemplate 是是是是是

- 是是是是：

query

```
public <T> List<T> query(String sql,
                         ParameterizedRowMapper<T> rm,
                         Object... args)
                  throws DataAccessException
```

- 是[**queryForObject**

```
public <T> T queryForObject(String sql,
                            Class<T> requiredType,
                            Object... args)
                     throws DataAccessException
```

□□ JDBC □□□□

- □□□□□□□□□ JdbcTemplate □□□□ , □□□□□□□□□ .
- **JdbcTemplate** □□□□□□□□□□□□ , □□□□□ IOC □□□□□□□□□□□ , □□□□□□□□□□ □□ DAO □□□ .
- JdbcTemplate □□□□ Java 1.5 □□□ (□□□□ , □□ , □□□□□) □□□□□
- Spring JDBC □□□□□□□□ JdbcDaoSupport □□□□ DAO □□ . □□□□ □ jdbcTemplate □□ , □□□□ IOC □□□□□ , □□□□□□□□□□□ .

□□ JDBC □□□□□□

```
<bean id="jdbcTemplate"
    class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="dataSource"/>
</bean>

<bean id="personDAO"
    class="org.simpleit.jdbc.PersonDAO">
    <property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
```

□□ JdbcDaoSupport □□□□

```
public class PersonDAO extends JdbcDaoSupport
```

```
<bean id="personDAO"
    class="org.simpleit.jdbc.PersonDAO">
    <property name="dataSource" ref="dataSource" />
</bean>
```

□ JDBC □□□□□□□□□

- □□□□ JDBC □□□ , SQL □□□□□□□ ? □□ , □□□□□□□□□ . □□□□□□□□□ , □□□ □□□□□□□□ , □□□□□□□□□ .
- □ Spring JDBC □□□ , □□ SQL □□□□□□□□□□□□□□□ (named parameter).
- □□□□ : SQL □□□ (□□□□□) □□□□□□□□□□ . □□□□□□□□□ , □□□□□□□ . □□□□□□□□ □□□□□□□□□□
- □□□□□□ NamedParameterJdbcTemplate □□□□□

□ JDBC □□□□□□□□□□

- □ SQL □□□□□□□□□□ , □□□□□ Map □□□□□□ , □□□□□
- □□□□□ SqlParameterSource □□
- □□□□□□□□□ Map □ SqlParameterSource □□□

update

```
public int update(String sql,
                  Map args)
           throws DataAccessException
```

update

```
public int update(String sql,
                  SqlParameterSource args)
           throws DataAccessException
```

batchUpdate

```
public int[] batchUpdate(String sql,
                         Map[] batchValues)
```

batchUpdate

```
public int[] batchUpdate(String sql,
                         SqlParameterSource[] batchArgs)
```

Spring-ing □□□□□□

[illegible]

- [illegible], [illegible].
- [illegible], [illegible]. [illegible], [illegible]
- [illegible] (**ACID**)
  - [illegible] (atomicity): [illegible], [illegible]. [illegible].
  - [illegible] (consistency): [illegible], [illegible]. [illegible].
  - [illegible] (isolation): [illegible], [illegible], [illegible].
  - [illegible] (durability): [illegible], [illegible], [illegible]. [illegible], [illegible].

□□□

- □□：
  - □□□□□□□□□□□□□□□□□□
  - □□□□□□□□ JDBC □，□□□□□□□□□□□□，□□□□□□□□□□□□

```
public void purchase(String isbn, String username){
    Connection conn = null;

    try {
        conn = dataSource.getConnection();
        conn.setAutoCommit(false);

        //...

        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        if(conn != null){
            try {
                conn.rollback();
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
        }
        throw new RuntimeException(e);
    } finally{
        if(conn != null){
            try {
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

}
```

Spring □□□□□□

- □□□□□□□□□□□□, **Spring** □□□□□□□□ **API** □□□□□□□□□□. □□□□□□□□□□□□□□□□□□□□□□□□ API, □□□□□ Spring □□□□□□□.
- Spring □□□□□□□□□□, □□□□□□□□□□□□.
- □□□□□□□**:** □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□. □□□□□□□□□, □□□□□□□□□□□□□□□□□□□□□□□□.
- □□□□□□□: □□□□□□□□□□□□□□□□□□. □□□□□□□□□□□□□□□□□□**,** □□□□□□□□□□□□□□□. □□□□□□□□ □□□□□□, □□□□ AOP □□□□□. **Spring** □□ **Spring AOP** □□□□□□□□□□□□.

Spring □□□□□□□□

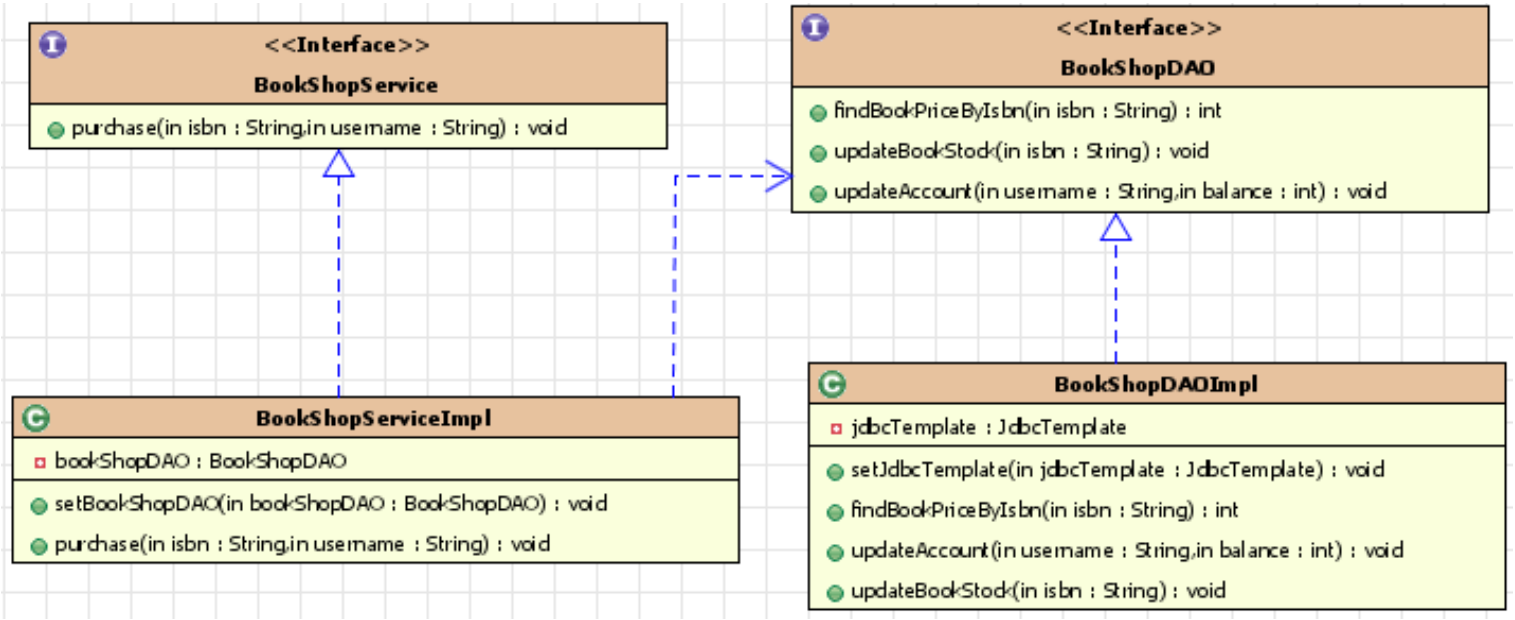
- Spring □□□□□□□□ API □□□□□□□□□□□□□. □□□□□□□□□□□□□□ API, □□□□□□□□□□. □□□□□□□□□, □□□□□□□□□□□□□□□□□□□□□□□□.
- Spring □□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□. □□□□ Spring □□□□□□□□□□ (□□□□□□□□), □□□□□□□□□□

org.springframework.transaction

Interface PlatformTransactionManager

Spring □□□□□□□□□□□□

- org.springframework.jdbc.datasource
  Class DataSourceTransactionManager ：□□□□□□□□□□□□□□□□ , □□□□ JDBC □□
- org.springframework.transaction.jta
  Class JtaTransactionManager ： □ JavaEE □□□□□□□ JTA(Java Transaction API) □□□□□□
- org.springframework.orm.hibernate3
  Class HibernateTransactionManager □□ Hibernate □□□□□□□
- ……
- □□□□□□□□□ Bean □□□□□ Spring IOC □□□


<<Interface>>
BookShopService
purchase(in isbn : String,in username : String) : void
<<Interface>>
BookShopDAO
findBookPriceByIsbn(in isbn : String) : int
updateBookStock(in isbn : String) : void
updateAccount(in username : String,in balance : int) : void
BookShopServiceImpl
bookShopDAO : BookShopDAO
setBookShopDAO(in bookShopDAO : BookShopDAO) : void
purchase(in isbn : String,in username : String) : void
BookShopDAOImpl
jdbcTemplate : JdbcTemplate
setJdbcTemplate(in jdbcTemplate : JdbcTemplate) : void
findBookPriceByIsbn(in isbn : String) : int
updateAccount(in username : String,in balance : int) : void
updateBookStock(in isbn : String) : void

是是是是是是是

Account 是

username	balance
Tom	30

Book 是

isbn	book_name	price
0001	Java	50

Book_STOCK 是

isbn	stock
0001	10

□□□□□□□□□□□□□□□

- □□□□□□□□□□□□□□
- □□□ Spring 2.x □□□□□□□□□□□, □□□□ tx Schema □□□□ **<tx:advice>** □□□ □□□□□, □□□□□□□□□□ Schema □□□□□ <beans> □□□□□.
- □□□□□□□□□, □□□□□□□□□□□□□□□. □□□□□□□□ <aop:config> □□□□□□□□, □□□□□□□□□□□□□□□□□□. □□□□□ **<aop:config>** □□□□□□□□□□□□□□□□□□□□□□□□.
- □□ Spring AOP □□□□□□□□□, □□□□□□□□□□□. □□, □□□□□□□□□□□ **Spring AOP** □□□□□□□.

□□□□□□□□□□□□□□□□□□□□

```
<bean id="bookShopService"
    class="org.simpleit.transaction.BookShopServiceImpl">
    <property name="bookShopDAO" ref="bookDAO"/>
</bean>

<bean id="transactionManager"
    class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"></property>
</bean>

<tx:advice id="bookShopTxAdvice"
    transaction-manager="transactionManager">
</tx:advice>

<aop:config>
    <aop:pointcut expression="execution(* *.BookShopService.*(..))"
        id="bookShopOperation"/>
    <aop:advisor advice-ref="bookShopTxAdvice"
        pointcut-ref="bookShopOperation"/>
</aop:config>
```

□□□□□□□□

□□□□□□□

□□ □□□□□□□□□□□ (□□□□□□□□□□□□□)

@Transactional

- , Bean , Spring @Transactional .
- , **@Transactional** . Spring AOP , .
- @Transactional . , .
- Bean **<tx:annotation-driven>** , .
- **transactionManager**, <tx:annotation-driven> transaction-manager . .

@Transactional

```
<bean id="jdbcTemplate"
    class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="dataSource"/>
</bean>

<bean id="transactionManager"
    class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"></property>
</bean>

<context:component-scan base-package="org.simpleit.transaction_1"/>

<tx:annotation-driven/>
```

[illegible]

- [illegible], [illegible]. [illegible]: [illegible], [illegible], [illegible].
- [illegible]. Spring [illegible] 7 [illegible].

Spring □□□□□□□□□

传播属性	描述
REQUIRED	如果有事务在运行，当前的方法就在这个事务内运行，否则，就启动一个新的事务，并在自己的事务内运行
REQUIRED_NEW	当前的方法必须启动新事务，并在它自己的事务内运行．如果有事务正在运行，应该将它挂起
SUPPORTS	如果有事务在运行，当前的方法就在这个事务内运行．否则它可以不运行在事务中．
NOT_SUPPORTE	当前的方法不应该运行在事务中．如果有运行的事务，将它挂起
MANDATORY	当前的方法必须运行在事务内部，如果没有正在运行的事务，就抛出异常
NEVER	当前的方法不应该运行在事务中．如果有运行的事务，就抛出异常
NESTED	如果有事务在运行，当前的方法就应该在这个事务的嵌套事务内运行．否则，就启动一个新的事务，并在它自己的事务内运行．

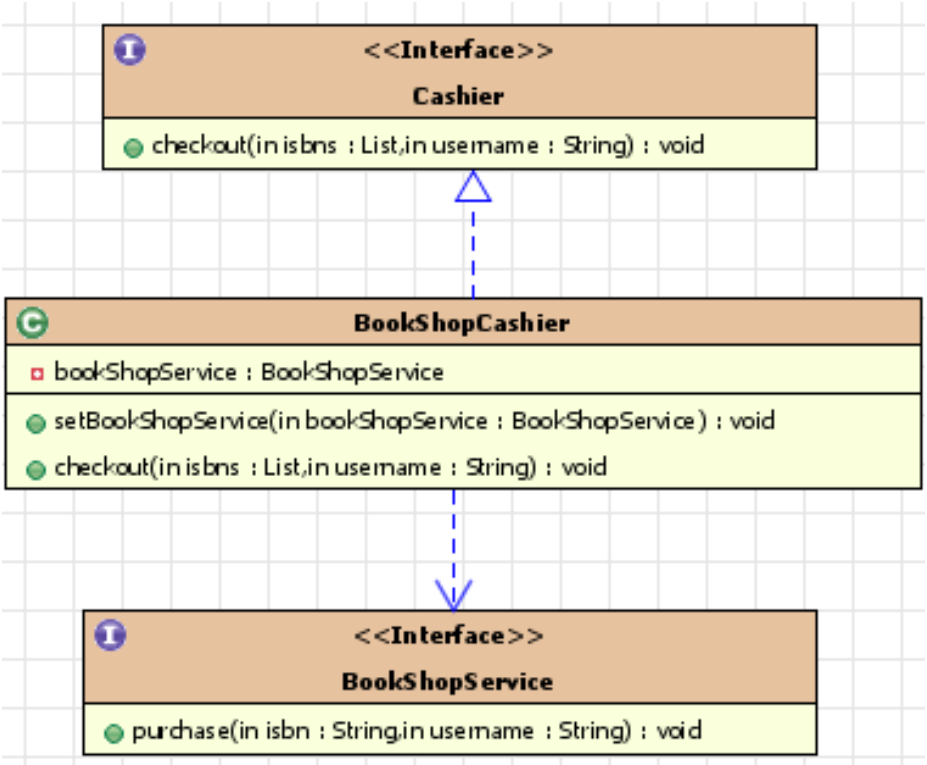
□□

- □□□ Cashier □□ : □□□□□□□□□□
- □□□□□□□□□□ , □□□□□ Tom □□□□ , □□□□□□□□□□ , □□□□□□□□□□

username	balance
Tom	60

isbn	book_name	price
0002	Oracle	80
0001	Java	50

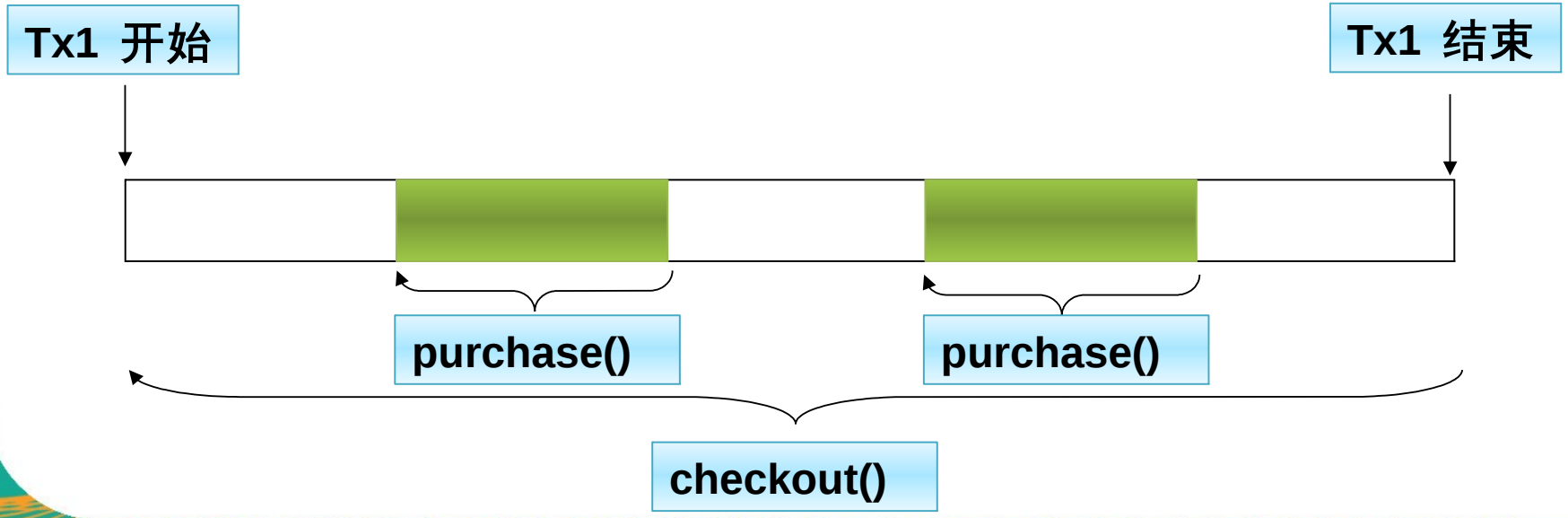
isbn	stock
0001	10
0002	10



REQUIRED □□□□

- □ bookService □ purchase() □□□□□□□□□□ checkout() □□□, □□□□□□□□□□□□□□. □ □□□□□□□□□□ REQUIRED. □□□ checkout() □□□□□□□□□□□□□□□□□□□□. □□□□□□ checkout () □□□□□□□□□□□, □□□□□□□□□□□□

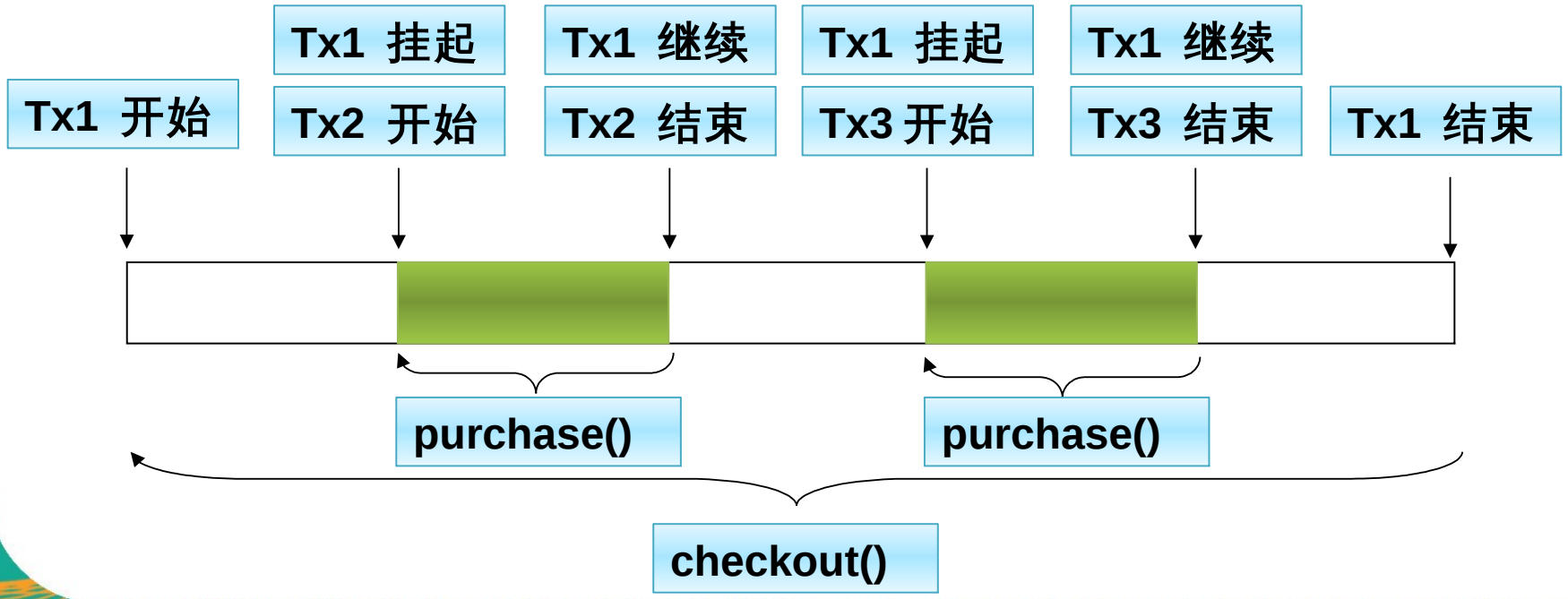
- □□□□□□□□□ @Transactional □□□ propagation □□□□□



REQUIRES_NEW □□□□

- □□□□□□□□□□□ REQUIRES_NEW. □□□□□□□□□□□□□□□□, □□□□□□□□□□□. □□□□□□□□, □□□□□□□□.

```
@Transactional(propagation=Propagation.REQUIRES_NEW)
public void purchase(String isbn, String username) {
```



□ Spring 2.x □□□□□□□□□□□□

- □ Spring 2.x □□□□□ , □□□□□□□□ <tx:method> □□□□□□□□□□□

```
<tx:advice id="bookShopTxAdvice"
    transaction-manager="transactionManager">
    <tx:attributes>
        <tx:method name="purchase" propagation="REQUIRES_NEW"/>
    </tx:attributes>
</tx:advice>
```

[illegible]

- [illegible], [illegible]
- [illegible]:
  - [illegible]: [illegible] T1, T2, T1 [illegible] T2 [illegible]. [illegible], [illegible] T2 [illegible], T1 [illegible].
  - [illegible]: [illegible] T1, T2, T1 [illegible], [illegible] T2 [illegible]. [illegible], T1 [illegible], [illegible].
  - [illegible]: [illegible] T1, T2, T1 [illegible], [illegible] T2 [illegible]. [illegible], [illegible] T1 [illegible], [illegible].

□□□□□□□□

- □□□□□□□, □□□□□□□□□□□□, □□□□□□□□□□□□□□□□. □□, □□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□.
- □□□□□□□, □□□□□□□, □□□□□□□□□□□□□□□□.
- □□□□□□□□□□□□□□□□□□□□□□□□

Spring □□□□□□□□□

隔离级别	描述
DEFAULT	使用底层数据库的默认隔离级别. 对于大多数数据库来说, 默认隔离级别都是 READ_COMMITED
READ_UNCOMMITTED	允许事务读取未被其他事物提交的变更. 脏读, 不可重复读和幻读的问题都会出现
READ_COMMITED	只允许事务读取已经被其它事务提交的变更. 可以避免脏读, 但不可重复读和幻读问题仍然可能出现
REPEATABLE_READ	确保事务可以多次从一个字段中读取相同的值. 在这个事务持续期间, 禁止其他事物对这个字段进行更新. 可以避免脏读和不可重复读, 但幻读的问题仍然存在.
SERIALIZABLE	确保事务可以从一个表中读取相同的行. 在这个事务持续期间, 禁止其他事务对该表执行插入, 更新和删除操作. 所有并发问题都可以避免, 但性能十分低下.

- □□□□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□.
- Oracle □□□ 2 □□□□□□□□ READ_COMMITED , SERIALIZABLE
- Mysql □□ 4 □□□□□□□.

□□□□□□□□

- □ @Transactional □□□□□□□□□□□□□□ @Transactional □ isolation □□□□□□□□□ .

```
@Transactional(propagation=Propagation.REQUIRES_NEW,
        isolation=Isolation.READ_COMMITTED)
public void purchase(String isbn, String username) {
```

- □ Spring 2.x □□□□□ , □□□ <tx:method> □□□□□□□□□

```
<tx:advice id="bookShopTxAdvice"
    transaction-manager="transactionManager">
    <tx:attributes>
        <tx:method name="purchase"
            propagation="REQUIRES_NEW"
            isolation="READ_COMMITTED"/>
    </tx:attributes>
</tx:advice>
```

□□□□□□□□

- □□□□□□□□□□□□ (RuntimeException □ Error □□□□□) □□□□□□□ . □□ □□□□□□ .
- □□□□□□□□□□□ @Transactional □□□ rollbackFor □ noRollbackFor □□□□□ . □□□□□□□□□ Class[] □□□ , □□□□□□□□□□□□□□□□□ .
  - rollbackFor: □□□□□□□□□
  - noRollbackFor: □□□□□□□□□□□□□□□

```
@Transactional(propagation=Propagation.REQUIRES_NEW,
        isolation=Isolation.READ_COMMITTED,
        rollbackFor={IOException.class, SQLException.class},
        noRollbackFor=ArithmeticException.class)
public void purchase(String isbn, String username) {
```

□□□□□□□□

- □ Spring 2.x □□□□□, □□□ <tx:method> □□□□□□□□□. □□□□□□□□□, □□□□□.

```
<tx:advice id="bookShopTxAdvice"
    transaction-manager="transactionManager">
    <tx:attributes>
        <tx:method name="purchase"
            propagation="REQUIRES_NEW"
            isolation="READ_COMMITTED"
            rollback-for="java.io.IOException, java.sql.SQLException"
            no-rollback-for="java.lang.ArithmeticException"/>
    </tx:attributes>
</tx:advice>
```

□□□□□□□

- □□□□□□□□□□□□□□□, □□□□□□□□□□□, □□□□□□□□□□□.
- □□□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□□.
- □□□□□□: □□□□□□□□□□□□□□□□□. □□□□□□□□□□□□□□□□□□□.
- □□□□□□: □□□□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□.

□□□□□□□□□□□□

- □□□□□□□□□□ @Transactional □□□□□ . □□□□□□□□□□□□ .

```
@Transactional(propagation=Propagation.REQUIRES_NEW,
        isolation=Isolation.READ_COMMITTED,
        rollbackFor={IOException.class, SQLException.class},
        noRollbackFor=ArithmeticException.class,
        readOnly=true,
        timeout=30)
public void purchase(String isbn, String username) {
```

- □ Spring 2.x □□□□□ , □□□□□□□□□ <tx:method> □□□□□□□ .

```
<tx:advice id="bookShopTxAdvice"
    transaction-manager="transactionManager">
    <tx:attributes>
        <tx:method name="purchase"
            propagation="REQUIRES_NEW"
            isolation="READ_COMMITTED"
            rollback-for="java.io.IOException, java.sql.SQLException"
            no-rollback-for="java.lang.ArithmeticException"
            timeout="30"
            read-only="true"/>
    </tx:attributes>
</tx:advice>
```

Spring 与 Hibernate

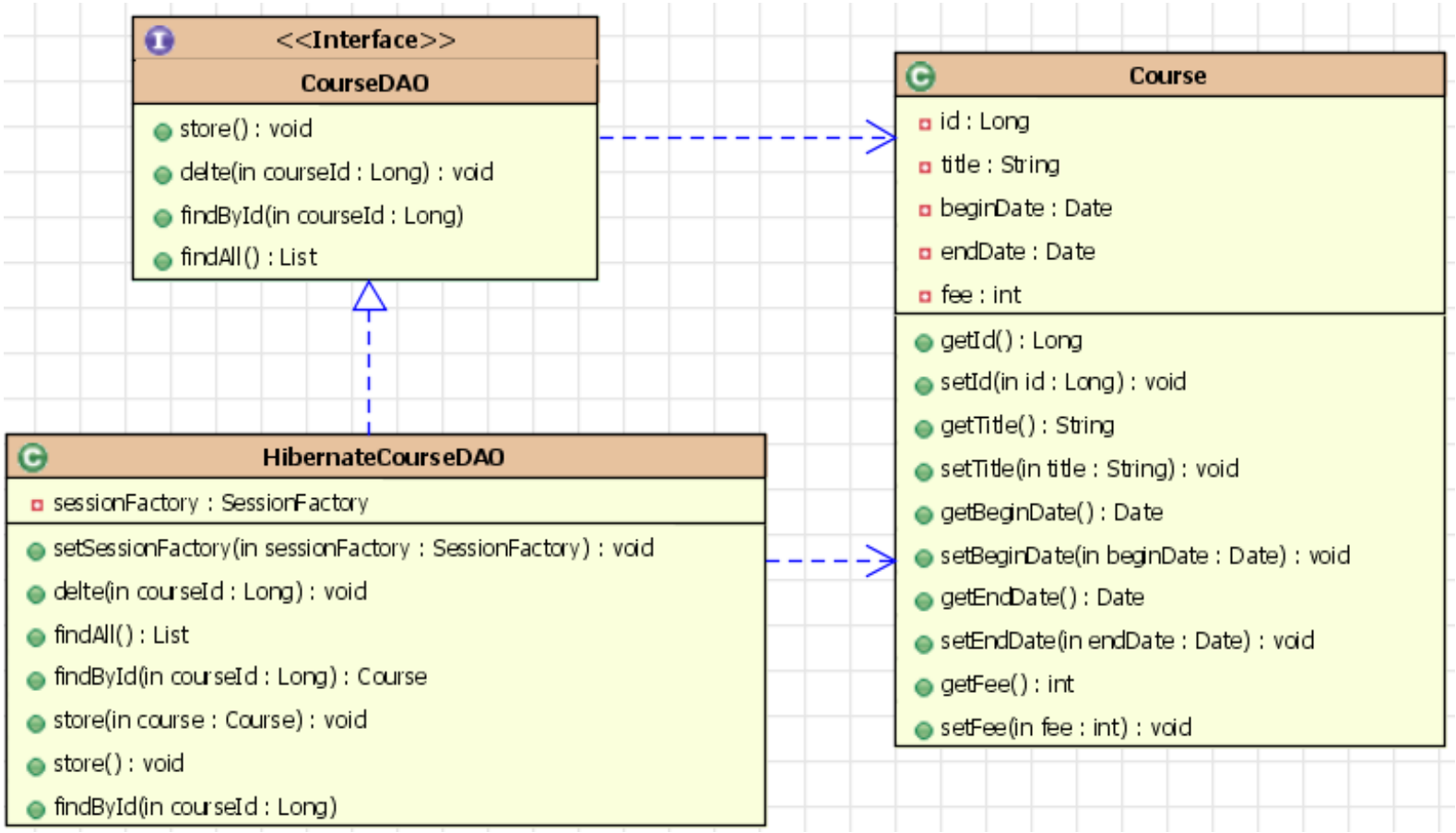
Spring □□ Hibernate

- Spring □□□□□□□□ ORM □□, □□ Hibernate JDO, TopLink, Ibatis □ JPA □
- **Spring □□□ ORM □□□□□□□□□**, □□□□□□ Hibernate □□□□□□□□□ ORM □□□.
- Spring 2.0 □□□□ Hibernate 2.x □ 3.x. □ Spring 2.5 □□□ Hibernate 3.1 □□□□□

□ Spring □□□ SessionFactory

- □□ Hibernate □□, □□□□□□ Hibernate API □□□ SessionFactory. □□, □□□□□□□□□ Spring □□□□□□□□□ (□□: Spring □□□□□□□)
- Spring □□□□□□□□ Bean, □□□□□□□□□□ IOC □□□□□ SessionFactory □□.

类图


<<Interface>>
CourseDAO
store() : void
delte(in courseId : Long) : void
findById(in courseId : Long)
findAll() : List
Course
id : Long
title : String
beginDate : Date
endDate : Date
fee : int
getId() : Long
setId(in id : Long) : void
getTitle() : String
setTitle(in title : String) : void
getBeginDate() : Date
setBeginDate(in beginDate : Date) : void
getEndDate() : Date
setEndDate(in endDate : Date) : void
getFee() : int
setFee(in fee : int) : void
HibernateCourseDAO
sessionFactory : SessionFactory
setSessionFactory(in sessionFactory : SessionFactory) : void
delte(in courseId : Long) : void
findAll() : List
findById(in courseId : Long) : Course
store(in course : Course) : void
store() : void
findById(in courseId : Long)

□ Spring □□□ SessionFactory(1)

- □□□□ **LocalSessionFactoryBean** □□ Bean, □□□□□□ XML □□□□□ SessionFactory □□.
- □□□□□□ Bean □□ configLocation □□□□□ Hibernate □□□□.

```
<bean id="sessionFactory"
    class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
    <property name="configLocation" value="hibernate.cfg.xml"/>
</bean>
```

```
<bean id="courseDAO"
    class="org.simpleit.HibernateCourseDAO">
    <property name="sessionFactory" ref="sessionFactory"/>
</bean>
```

□ Spring □□□ SessionFactory(2)

- □□□ Spring IOC □□□□□□□□. □□□□□□□□□□ LocalSession FactoryBean □ dataSource □□□. □□□□□□□□□□□□□□□ Hibernate □□□□□□□□□□□.

Spring SessionFactory(2)

```
<context:property-placeholder location="C3P0_config.properties"/>

<bean id="dataSource"
    class="com.mchange.v2.c3p0.ComboPooledDataSource">
    <property name="user" value="${user}"/>
    <property name="password" value="${password}"/>
    <property name="jdbcUrl" value="${jdbcUrl}"/>
    <property name="driverClass" value="${driverClass}"/>

    <property name="checkoutTimeout" value="${checkoutTimeout}"/>
    <property name="idleConnectionTestPeriod" value="${idleConnectionTestPeriod}"/>
    <property name="initialPoolSize" value="${initialPoolSize}"/>
    <property name="maxIdleTime" value="${maxIdleTime}"/>

    <property name="maxPoolSize" value="${checkoutTimeout}"></property>
    <property name="minPoolSize" value="${minPoolSize}"></property>
    <property name="maxStatements" value="${maxStatements}"></property>
</bean>

<bean id="sessionFactory"
    class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
    <property name="configLocation" value="hibernate.cfg.xml"/>
    <property name="dataSource" ref="dataSource"></property>
</bean>
```

□ Spring □□□ SessionFactory(3)

- □□□□□□□□□□ LocalSessionFactoryBean □, □□□□ Hibernate □□ □□.
- □□□ LocalSessionFactoryBean □ **mappingResources** □ □□□□ XML □□□□□□□. □□□□ String[] □□. □□□□□□□□□□□□.
- □ hibernateProperties □□□□□□□□□□.

□ Spring □□□ SessionFactory(3)

```
<bean id="sessionFactory"
    class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
    <property name="dataSource" ref="dataSource"></property>

    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
            <prop key="hibernate.show_sql">true</prop>
            <prop key="hibernate.hbm2ddl.auto">update</prop>
        </props>
    </property>

    <property name="mappingResources">
        <list>
            <value>org/simpleit/Course.hbm.xml</value>
        </list>
    </property>
</bean>
```

□ Spring □ ORM □□□□□□□

- □□□□□ ORM □□□, □□□□□ DAO □□□□□□□□□□. □□: □□□□ Session □□; □□, □□, □□□□□.
- □ JDBC □□, Spring □□□□□□□□ ------ □□□□□ **DAO** □□□□□ ORM □□□□□. □□ Spring □□□□□□□□ API □□□□□□□□□□□□. □□□□□ ORM □□, □□□□□□□□□□□□□□□□.

Spring □□□□□□□□□□□□□□

支持类	JDBC	Hibernate
模板类	JdbcTemplate	HibernateTemplate
DAO 支持类	JdbcDaoSupport	HibernateDaoSupport
事务管理类	DataSourceTransactionManager	HibernateTransactionManager

- HibernateTemplate □□□ Hibernate □□□□□□□□□□□□ .
- HibernateTemplate □□□□□□ Hibernate □□□□□ Spring □□□ □□□□□□ . □□□□ Spring □□□□□□□□□□ .

□□ Hibernate □□

- HibernateTemplate □□□□□□□□□□□□□□. □□□□□□□□□□ DAO □□□□□□ Hibernate □□, □□□□□□□□□□□□□□□□□□□□□□□□□□□□. □□□□□□□□□□□□□□□□□ Hibernate API □□□.
- □□□ DAO □□□□ @Transactional □□□□□□□□□□□□□□.
- HibernateTemplate □□□□□□□, □□□□□ Bean □□□□□□□□□□□□, □□□□□□□□ □□□ Hibernate DAO □.

□□ Hibernate □□□□□□

```
private HibernateTemplate hibernateTemplate;

public void setHibernateTemplate(HibernateTemplate hibernateTemplate) {
    this.hibernateTemplate = hibernateTemplate;
}

@Override
@Transactional
public void delte(Long courseId) {
    Course course =
        (Course) hibernateTemplate.get(Course.class, courseId);
    hibernateTemplate.delete(course);
}

@Override
@Transactional(readOnly=true)
public List<Course> findAll() {
    return hibernateTemplate.find("FROM Count");
}

@Override
@Transactional(readOnly=true)
public Course findById(Long courseId) {
    return (Course) hibernateTemplate.get(Course.class, courseId);
}

@Override
@Transactional
public void store(Course course) {
    hibernateTemplate.saveOrUpdate(course);
}
```

□□ Hibernate □□□□□□

```
<bean id="transactionManager"
    class="org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory"></property>
</bean>

<tx:annotation-driven transaction-manager="transactionManager"/>

<bean id="hibernateTemplate"
    class="org.springframework.orm.hibernate3.HibernateTemplate">
    <property name="sessionFactory" ref="sessionFactory"></property>
</bean>

<bean id="courseDAO_"
    class="org.simpleit.HibernateCourseDAO_">
    <property name="hibernateTemplate" ref="hibernateTemplate"/>
</bean>
```

HibernateTemplate Hibernate Session

```
hibernateTemplate.execute(new HibernateCallback() {
    @Override
    public Object doInHibernate(Session arg0) throws HibernateException,
            SQLException {
        //...
        return null;
    }
});
```

□□ Hibernate □ DAO □□□

- Hibernate DAO □□□□□□ HibernateDaoSupport □□□ setSessionFactory() □ setHibernateTemplate() □□. □□, □□□ DAO □□□□□ getHibernateTemplate() □□□□□□□□□□□□□.
- □□□ HibernateDaoSupport □□□□□□ SessionFactory □□, □□□□□ □□□□ HibernateTemplate □□□, □□ HibernateDaoSupport □□□ □□□ SessionFactory □□□□□□□□ HibernateTemplate □□□, □□□ hibernateTemplate □□

□ Hibernate □□□□ Session □□□□□

- Spring □ HibernateTemplate □□□□□□□□□, □□ DAO □□. □□□ HibernateTemplate □□□ DAO □□□□□ Spring □ API
- □□ HibernateTemplate □□□□□□□□□ Hibernate □□□□ Session □□.
- **Hibernate □□□ Session □□□ Spring □□□□□□□□□□, □□□□□□□□□ DAO □□□□□□□**
- □□□□□□□ beans.xml □□□□□, □□ Spring □□□□□□□□, □□□□□ Thread Local □□□□□□ Session □□

```
<prop key="hibernate.current_session_context_class">thread</prop>
```

□ Hibernate □□□□ Session □□□□□

- □ Hibernate □□□□□□□□□□□, □□□□□□□□□□□ HibernateExceptio n.
- □□□□□□□□□□□□□, □□ Hibernate □□□□□ Spring □ DataAcces sException □□, □□□□□□□□□□□□ DAO □□□ @Respository □□.
- □□□□□□□ org.springframework.dao.annotation Class PersistenceExceptionTranslationPostProcessor □□, □□□□ Hibernate □□□□□ Spring □ DataAccessException □□□□ □□□□□□□□. □□ Bean □□□□□□□□□ @Respository □□□ Bean □□□□.

Hibernate □□□□□□ Session(1)

- □ Hibernate 3 □□, SessionFactory □□□□ getCurrentSessi on() □□, □□□□□□□□“□□□“□□□ Session.
- Hibernate □□ CurrentSessionContext □□□□□□□ □□□□ hiberna te.current_session_context_class □□ “□□□”
  - JTASessionContext: □□ JTA □□□□□□ Session □□.
  - ThreadLocalSessionContext: □□□□□□□□□□□□□□□□□ Session □□
  - ManagedSessionContext: □□□□□□□□□□□□□□□ Session □□. □□□□□□□□□□□□□□ □□□□ Sessio □□, □□□□, flush □□□□ Session □□.

Hibernate □□□□□□ Session(2)

- □□□□ ThreadLocalSessionContext □□, Hibernate □ Session □□□ getCurrentSession() □□□□□□, □□□□□□□□□□.
- □□□□□□□□ JTA □□□□□□, □□□□□□□□□; □□□□□□ Hibernate □□□□□□□□□.
- □□:
  - □□ JTA □□□□□□ Session □□:

```
<property name="hibernate.current_session_context_class">thread</property>
```

  - □□□□□□□□□□□□□□□□□ Session □□□

```
<property name="hibernate.current_session_context_class">jta</property>
```

□□ Struts2

□□□□ web □□□□□ Spring

- □□□□ Servlet □□□ ContextLoaderListener, Web □□□□□□□□ Spring □ ApplicationContext □□. □□□□□□□□□□ ApplicationContext □□□□□ Web □□□□□ ServletContext □. □□, Servlet □□□□□ ServletContext □□□□□□□□□□□□□□□□□□ Spring □□□□□□□□□.

□□□□ web □□□□□ Spring □□□□

- □ web.xml □□□□□ Spring □□□ Servlet □□□

  org.springframework.web.context

  Class ContextLoaderListener □□□□ Spring □ ApplicationContext □□□ ServletContext □□□ .

- org.springframework.web.context

  Class ContextLoaderListener □□□□□□□ web □□□□□□□ contextConfigLocation □□□ Bean □□□□□□□ . □□□□□ Bean □□□□ , □□□□□□□ □□□□□□ . contextConfigLocation □□□□□ /WEB-INF/applicationContext.xml. □□□□□□□□□□□□□□□□□□□□ web □□□□□□□□

web.xml □□□□□

```xml
<context-param>
  <param-name>contextConfiguration</param-name>
  <param-value>/WEB-INF/applicationContext.xml</param-value>
</context-param>

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

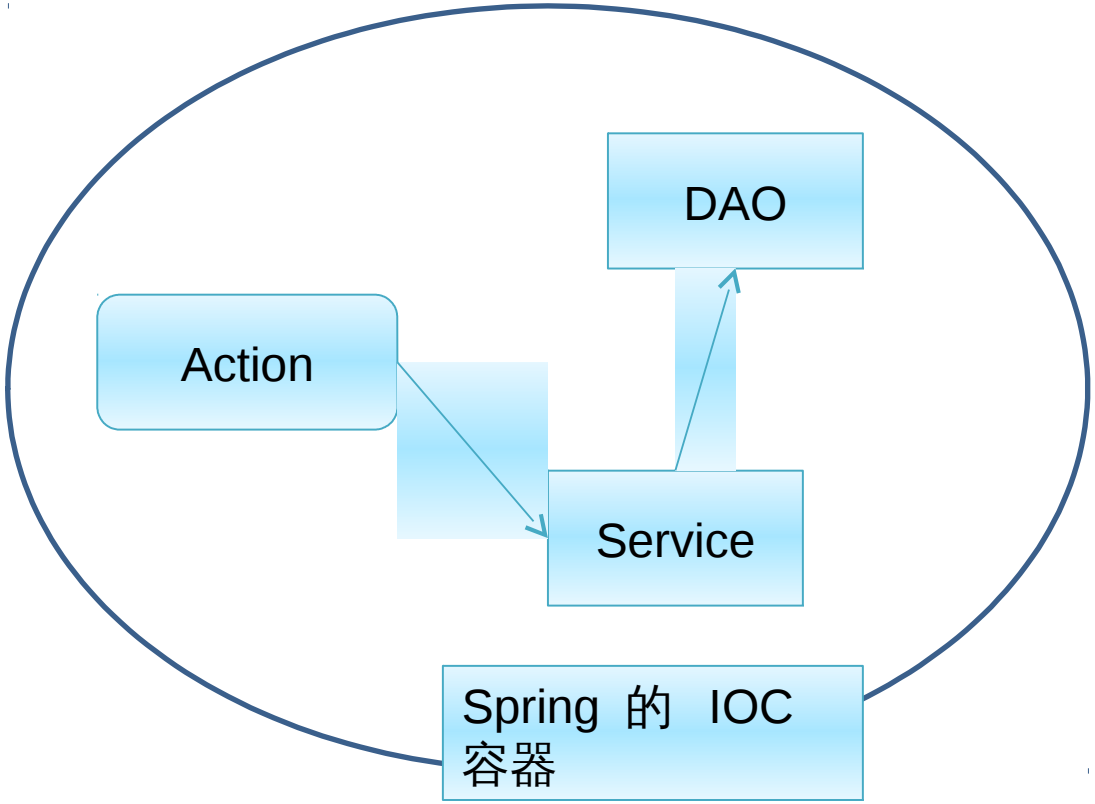
web Spring ApplicationContext

- org.springframework.web.context.support
  Class WebApplicationContextUtils

```
public static WebApplicationContext getRequiredWebApplicationContext(ServletContext sc)
                                                                throws IllegalStateException
```

Spring ApplicationContext

```
WebApplicationContext applicationContext =
    WebApplicationContextUtils.getRequiredWebApplicationContext(this.getServletContext());
Test test = (Test) applicationContext.getBean("test");
test.hello();
```


DAO
Action
Service
Spring 的 IOC 容器

□□ Struts2

- Struts2 □□□□□□□ Spring □□□.
- Struts2 □□□□□□ Spring □□□□□□□:
  - □ Action □□□□ Spring □□□□□□□, □□, □□□□□□, □□□□□□ Spring □□□ IOC □□, □□□□□□□
  - □□ Spring □□□□□□□□□, □ Spring □□□□ Action □□□, □□□ Spring □□□□□□□□□□□□□□□ Action □□.

□ Spring □□□□□

- □ Action □□□□ Spring □□□□□□□ , □□ , □□□□□□ , □□□□□□ Spring □ □□ IOC □□ , □□□□□□□
- □□□□ :
  - □□ Spring □□ : □ struts2-spring-plugin-2.2.1.jar □□□□□ WEB □□□ WEB-INF/lib □□□
  - □ Spring □□□□□□□ Struts2 □ Action □□
  - □ Struts □□□□□□□ action, □□ class □□□□□□□ Action □□□□ , □□□□ Spring □□□ Action □□□ ID

□□□□

- □□ Spring □□□□□□□□□□, □ Spring □□□□ Action □□□, □□□ Spring □□□□□□□□□□□□□□□□ Action □□.
- □□□□□□□□□: Spring □□□□□□□□□□□□ struts.objectFactory.spring.autoWire □□□□, □□□□□□□□□□□:
  - name: □□□□□□□□□□.
  - type: □□□□□□□□□. □□□□ type □□□ Bean, □□□□□□□□□□; □□□□□□□ Bean, □□□□□□□□□, □□□□□□□□
  - auto: Spring □□□□□□□□□□□□□□□□□□□□□□□□□□
  - constructor: □ type □□, □□□ constructor □□□□□□□□□□□□□□□□□
- □□□□□:
  - □□ Spring □□
  - □□□□□ struts □□□□
  - □□ spring □□□□□, □□□□□□□□□□□□□□ Action □□