Spring第二天

如何装配Bean

如何给集合注入值

集合的种类

1. map
2. set
3. list
4. 数组
5. collection

```
Collection col=new ArrayList()是正确的
```

示例

1. 定义一个Collection的包
2. 创建一个Department类
    1. 有一些属性,name.empName;
    2. 设置set和get方法
3. 编写配置文件

```
<bean id="department" class="Department">
<property name="name" value="财务部"/>
    //如果装配数组和list
 <property name="empName">
    <list>
    <value>大明</value>
    <value>小明明</value>
    <value>吓跑，OMG</value>
    </list>
  </property>
</bean>
```

4. 编写测试文件

```
ApplicationContext ac=new ClassPathXmlApplicationContext("xml文件");
Departemnt department=(Department)ac.getBean("department");
for(String empName:department.getEmpName){
System.out.print(empName)
}
```

2. List集合取值是有序的

5. 装配set集合

```
//xml文件
<ref bean="id" />
//遍历
for(String setEmpName:department.getSetEmpName){
System.out.print(setEmpName)
}
```

2. Set集合取值是无序的
3. List集合可以有相同对象,Set集合放相同对象会覆盖,只能设置不同对象

6. 装配map集合

```
<map>
    <entry key="1" value-ref="emp1" />
    <entry key="2" value-ref="emp2" />
</map>
//遍历map集合的方法
    //1.迭代器
    //2.Entry
    for(Entry<String,Employee>
entry1:department.getEmpMaps().entrySet()){
        print(entry1.getKey()+"\t"+entry1.getValue());
    }
```

2. 迭代器遍历
3.

```java
//1.迭代器
Map<String,Employee> empmaps=department.getEmpMaps();
Iterator it=empmaps.keySet().iterator();
while(it.hasNext()){
    String key=(String) it.next();
    Employee emp=empmaps.get(key);
    System.out.println("key="+key+" "+emp.getName());
}
```

装配基本Bean

内部Bean

```
<bean id="" class="">
<property name="" value="">
    <bean id="" class=""></bean>
</property>
</bean>
```

2. 内部bean 表名这个bean当前bean可用,其他人不能重用
3. ref的方式可以引用

装配继承关系的bean

示例

1. 创建student和gradate,前者继承后者,后者有属性学位degree

1.
```java
package com.wb.stu;

/**
 * Create By WeiBin on 2020/3/16 12:32
 */
public class Gradate extends Student {
    private String degree;

    public String getDegree() {
        return degree;
    }

    public void setDegree(String degree) {
        this.degree = degree;
    }

}
```

```java
package com.wb.stu;

/**
 * Create By WeiBin on 2020/3/16 12:31
 */
public class Student {
    private String name;
    private int age;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

2. xml配置bean

1.
```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-2.5.xsd">
    <bean id="student" class="com.wb.stu.Student">
        <property name="name" value="魏斌"/>
        <property name="age" value="23"/>
    </bean>

    <bean id="gradate" parent="student" class="com.wb.stu.Gradate">
        <property name="degree" value="硕士"/>
    </bean>

</beans>
```

3. 编写测试文件测试

1.
```
 * Create By WeiBin on 2020/3/16 12:36
 */
public class StudentTest {
    public static void main(String[] args) {
        ApplicationContext ac=new ClassPathXmlApplicationContext( configLocation: "com/wb/stu/beans.xml");
        Gradate gradate = (Gradate) ac.getBean( name: "gradate");
        System.out.println(gradate.getName()+"\t"+gradate.getAge()+"\t"+gradate.getDegree());
    }
}
```

```
StudentTest › main()
StudentTest
E:\JDK8\bin\java.exe ...
log4j:WARN No appenders could be found for logger (org.springframework.core.env.StandardEnvironment).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
魏斌 23  硕士
```

4. 问题:

    1. 如果自己配置属性name,age则会替换从父对象继承的数据

装配Properties

1. 创建properties属性
2. 配置xml文件

```xml
<bean>
    <props>
    <prop key="pp1">abcd</prop>
    <prop key="pp2">hello</prop>
    </props>
</bean>
```

3. 通过properties取出数据

```
Properties properties=department.getPp();
System.out.println(properties.get("键名").toString())
```

2.
```
}
System.out.println("*****通过Enumeration取出*****");
Enumeration en= pp.keys();
while(en.hasMoreElements()){
//  Entry<Object,Object> elment= (Entry<Object, Object>) en.nextEl
//  System.out.println(elment.getKey()+" "+elment.getValue());
    String key=(String) en.nextElement();

    System.out.println(key+" "+pp.getProperty(key));

}
```

如果用构造函数注入

构造参数索引

1. 构建一个包,constructor
2. 创建一个类 Employee 赋予name,age属性,编写get/set方法和构造方法
3. xml配置对象

```
<bean id="employee" class="">
<!-- 通过构造函数来注入 --!>
<constructor-arg index="0" type="java.lang.String" value="大明"/>
<constructor-arg index="1" type="int" value="23"/>
</bean>
```

4. **set注入的缺点是无法清晰表达哪些属性是必须的,哪些是可选的,构造注入的优势是通过构造强制依赖关系,不可能实例化不完全的或无法使用的bean**

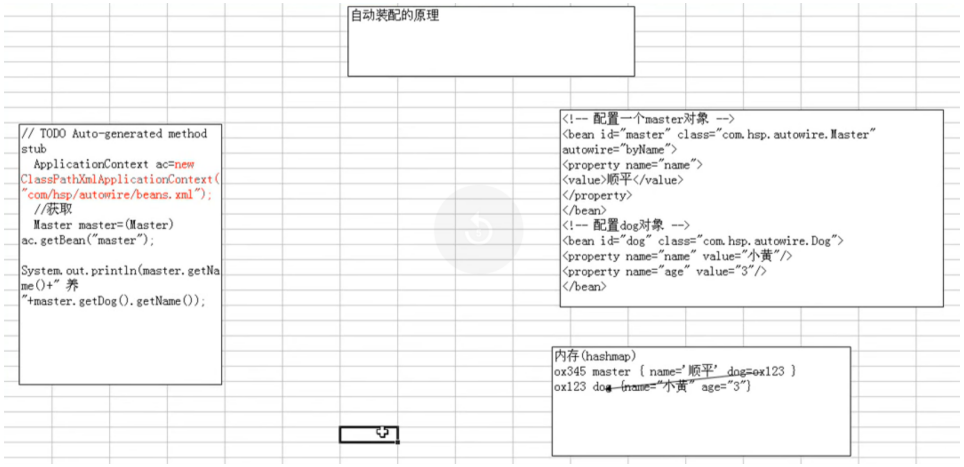
自动装配bean的属性值

示例

1. 创建dog和master 主人有名字和一只狗
2. 配置bean.xml

```
//配置一个master对象
<bean id="master" class="Master">
<property name="name" value="魏斌"/>
    //传统
    <property ref="dog"/>
    //自动装配
</bean>
//配置一个dog对象
<bean id="dog" class="Dog">
    <property name="name" value="小黄"/>
    <property name="age" value="15"/>
</bean>
```

3. 测试文件
    1. 获取上下文容器调用getBean方法
    2. 如果不自动装配,会出现自动装配
    3. autowire="byName" 是根据容器中是否有相同名字的id的对象
    4. autowire="byType" 根据类型找对象,但是只能有唯一的对象,多个对象会抛异常
    5. autowire="constructor" 根据构造函数来找对象,查找和bean的构造参数一致的一个或多规格bean，若找不到或找到多个,抛异常。按照参数的类型装配
    6. autowire="autodetect" 在5和4之间选一个方式,不确定性的处理与5和4一致
    7. autowire="default" 在beans里面配置 default-autowire="xxx",不写的话是no,如果bean中不写autowire代表的是autowire="default"
    8. autowire="no" 不自动装配 ,所谓自动装配是在没有set属性的时候,去自动装配
4. 自动装配细节

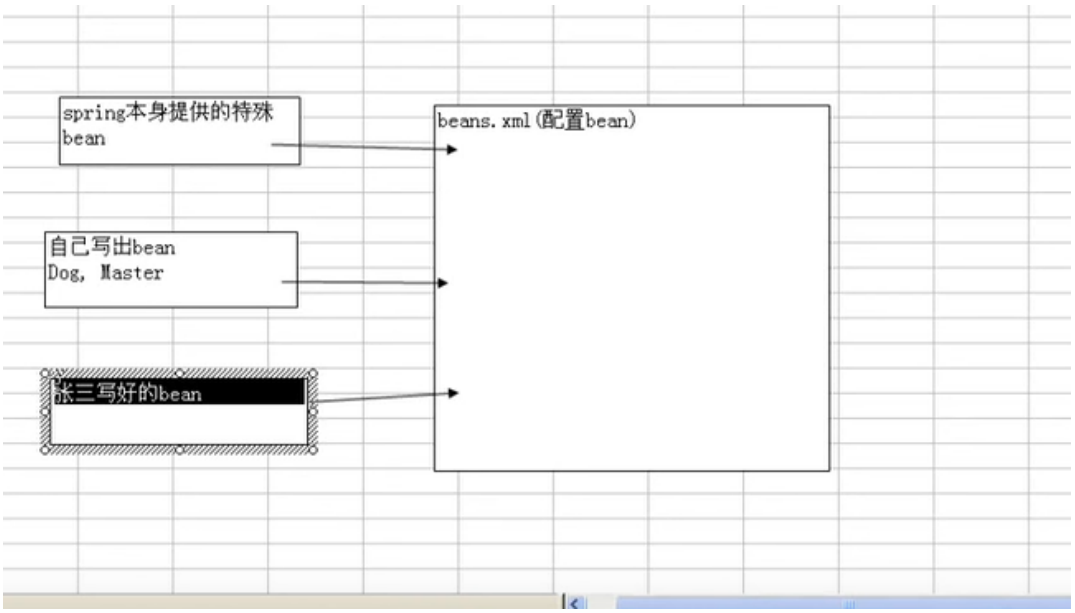
1. 

2.

context:annotation-config

1. 代表启用注解,激活在类中探测到的各种注解

使用spring的特殊bean

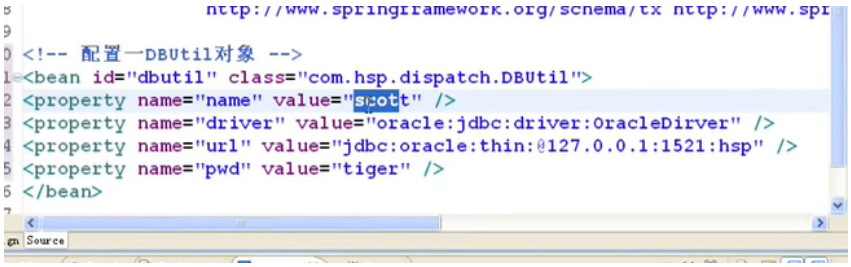
1. 

2. 实现了接口的后置器处理Bean
3. bean工厂
4. 分散配置
    1. 打包
    2. 编写一个工具类DBUtil有属性 driverName,url,username,password
    3. 配置xml DBUtil

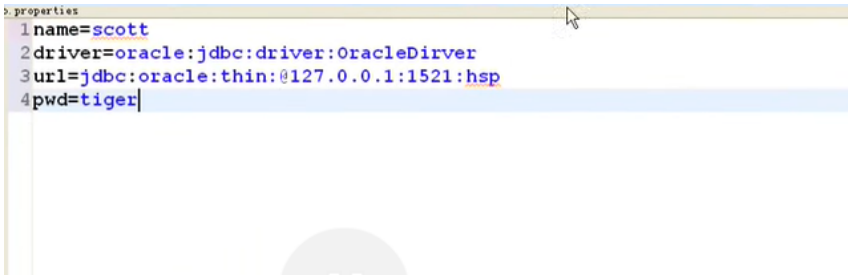
```
<bean id="dbutil" class="DBUtil">
//四种属性

</bean>
```

    2.



4. 用属性文件来注入值,分散配置

    1. 

    2. 配置spring的特殊的bean

```
context:property-placeholder
location="classpath:你的属性文件路径"
```

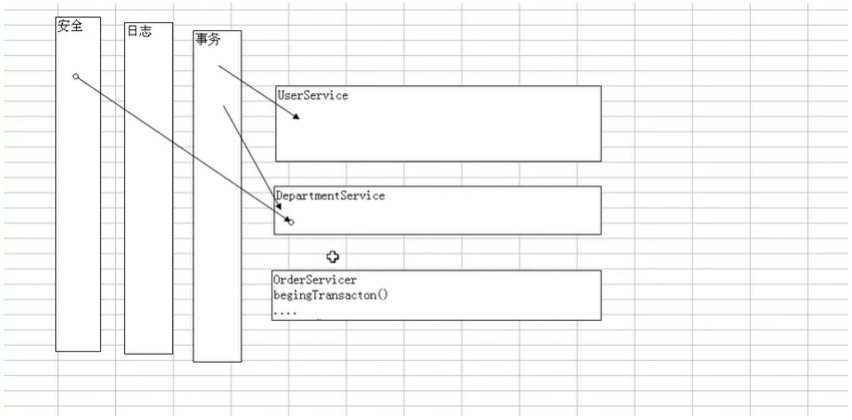
        2. 引入了我们的db.properties文件
        3. 多个要用逗号分隔

SpringAop

AOP编程

AOP概念

1. aop(aspect oriented programming) 面向切面（方面）编程，是对所有对象或者是一类对象编程
2. 核心：（不，还）
    1. 在不增加代码的基础上
    2. 还增加新的功能
3. 发展
    1. 汇编(伪机器指令 mov jump) 面向机器
    2. c语言(面向过程) 语句1:语句2;....---》系统软件(操作系统,数据库,语言本身,杀毒软件,防火墙,驱动)
    3. c++ 结构体 过程>对象 的过度
    4. java语言(面向对象-->类,对象) 属性：变量；行为-->函数
    5. 面向切面 spring(-->aop)
        1. 初步原理图

2. 

AOP特别提醒

1. aop编程，实际上在开发框架本身用的较多，在实际项目中，用的不是很多
2. **但是讲来回越来越多，这是趋势**

AOP原理+案例

1.