**Struts2**

1.项目的配置

a.配置tomcat

b.创建项目（选 Target RunTime ApatchTomcat7.0）

c.复制项目文件到WebContent目录下

a.jstl包需要导入

2.struts2环境的搭建

a.导入jar包（13基础jar包）

b.创建Action（详细见UserAction类）

c.编写struts.xml配置文件，并放置在src目录下

d.在web.xml文件中配置struts2核心过滤器（StrutsPrepareAndExecuteFilter）

3.几个小点

404：action的请求路径需要加上项目名称

500：在action标签中配置method方法

4.strust2的加载顺序

a.启动tmocat

b.加载web.xml

c.加载<filter>struts</filter>

d.加载struts-default.xml,struts-plugin.xml,struts.xml

5.struts配置详解见（struts.xml文件）

6.页面中请求路径的写法

a.页面一律改成jsp

b.${pageContext.request.contextPath}获取项目名

c.从新拼接请求路径

7.使用原生态的sevletAPI

1.struts属性封装

2.添加额外的struts-XX.xml文件

1.创建struts-xx.xml文件（自己定义，位置不限,但是格式要与struts.xml一致）

2.在主struts.xml中通过下面的标签来加载额外的struts-xx.xml文件

<include file="cn/oracle/oaec/struts/customer/struts-customer.xml"></include>

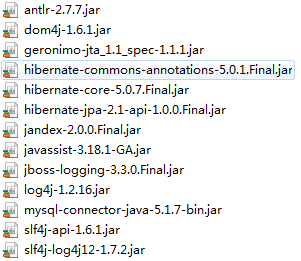
拦截器：

高级 方法继承methodFilterIntercepter

异步同步

如果从数据库查询再把标签写到页面上，如果使用异步无法再加载的时候就在这些标签上绑定事件或进行操作，若要操作则必须进行同步请求 或者

**Hibernate 框架配置**



13个包（3个日志包 一个数据库驱动jar包）

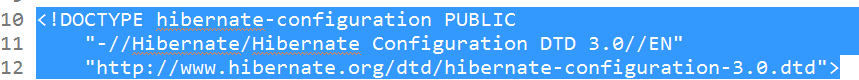
src下创建

hibernate.cfg.xml

C:\Users\OracleOAEC\AppData\Roaming\Tencent\Users\361370668\QQ\WinTemp\RichOle\W[ZJ~3JQ[FH1$UGPUDTVFD3.png

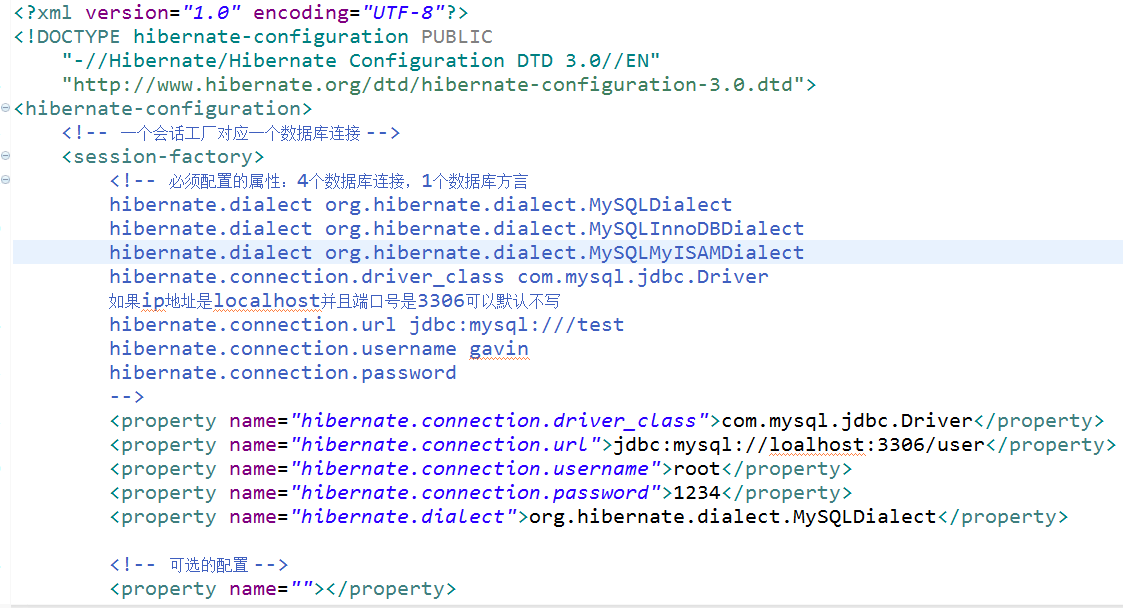
C:\Users\OracleOAEC\AppData\Roaming\Tencent\Users\361370668\QQ\WinTemp\RichOle\W8NAXB4[A6ASUPHSR4J3$MX.png

C:\Users\OracleOAEC\AppData\Roaming\Tencent\Users\361370668\QQ\WinTemp\RichOle\}~17CA`2RACJSEZDOP{)0LU.png



对xml进行约束

写xml：



1.持久化类

1.遵循一些规范

a.无参构造

b.提供get、set方法

c.要有一个唯一标识列 OID

d.数据类型都使用包装类型

2.同时还需要有hibernate的映射文件

主键的生成策略

increment 自动增长（先查询数据库中最大的ID，然后加1，在多线程中会出现问题）

identity 自动增长，要求数据库可以设置自增长（一般用在mysql数据库）

sequence 使用序列，要求数据库能提供序列（一般用在oracle数据库）

★★★★★★★★★★★★★★★★★★★★★★

native 本地化，会自动识别数据库（要求主键必须是数值型）

uuid 会自动生成uuid的字符串，要求主键的类型是String类型

★★★★★★★★★★★★★★★★★★★★★★

assigned 手动配置主键，需要手动去赋值

2.持久化对象

通过我们的持久化类去创建出来的对象，并且该对象的属性值在数据库中有对应的记录存在

瞬时状态：通过new关键字创建的对象

持久状态：在session的管理中，并且在数据库中有对应的OID（理解为存在数据）

脱管（游离）状态：不在session的管理中，但是在数据库中也有对应的OID（理解为存在数据）

session的缓存管理（3个方法）

evict（Object obj）:去除指定的对象obj

clear() :清空session缓存

close() :关闭session

删除状态：不在session的管理中，在数据库中也没有对应的OID（理解为存在数据）

特点：持久化对象如果在持久状态，session会自动去更新他的数据

记住结论：调用delete方法后变成删除状态

调用evict(obj),clear(),close()变成脱管(游离状态)

其他的就是持久状态

3.一级缓存session

为什么需要缓存：我们将一些常用的数据存放在内存中，减少数据库的连接，提高访问速度

session缓存的快照

4.session的线程绑定

回顾事务

原子性

一致性

隔离性

持久性

事务的隔离级别

不提交读 1

提交读 2（oracle的默认级别）

可重复读 4（mysql的默认级别）

序列化读 8

为什么要把session绑定到线程中去呢

让我们可以在同一个线程的任何地方访问到的session都是同一个对象

如何操作

1.配置主配置文件hibernate.cfg.xml

<property name="hibernate.current\_session\_context\_class">thread</property>

2.代码中使用sessionFactory.getCurrentSession();

重点：hibernate需要手动提交事务（commit），默认情况下不会自动提交事务（commit）。

2.如果使用的是openSession()方法，需要手动关闭session;

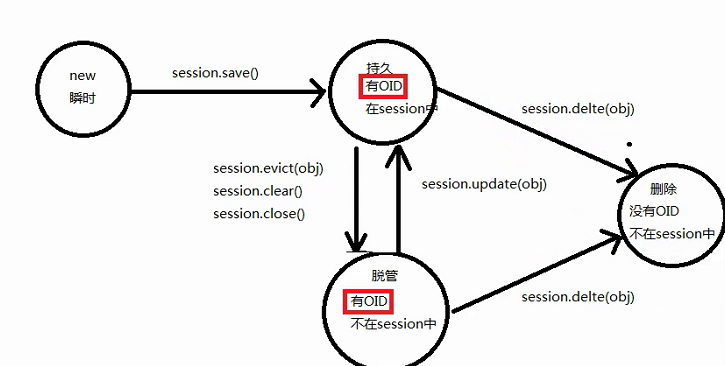
2.如果使用的是getCurrentSession()方法，不需要手动关闭session

在commit或rollback的时候会自动关闭session

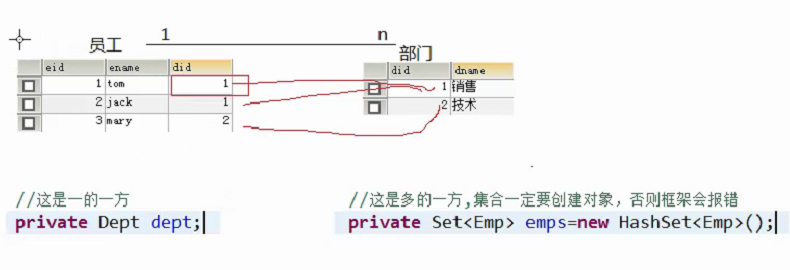
5.查询接口

HQL语句：它是一个查询语句

QBC的好处：它主要用来做条件查询，最方便



一对多





关联映射

1.一对多

2.多对多

cascade 级联操作

none 默认没有

★★★★★★★★★★★★★★★★★★★★★

save-update 保存或更新

delete 删除

all 保存、更新、删除

★★★★★★★★★★★★★★★★★★★★★

delete-orphan 孤儿删除

all-delete-orphan 保存、更新、删除、孤儿删除

inverse 外键维护设置

默认情况都是false，就是维护外键

true就是放弃维护

cascader 做保存或更新或删除时有用

inverse 与外键有关

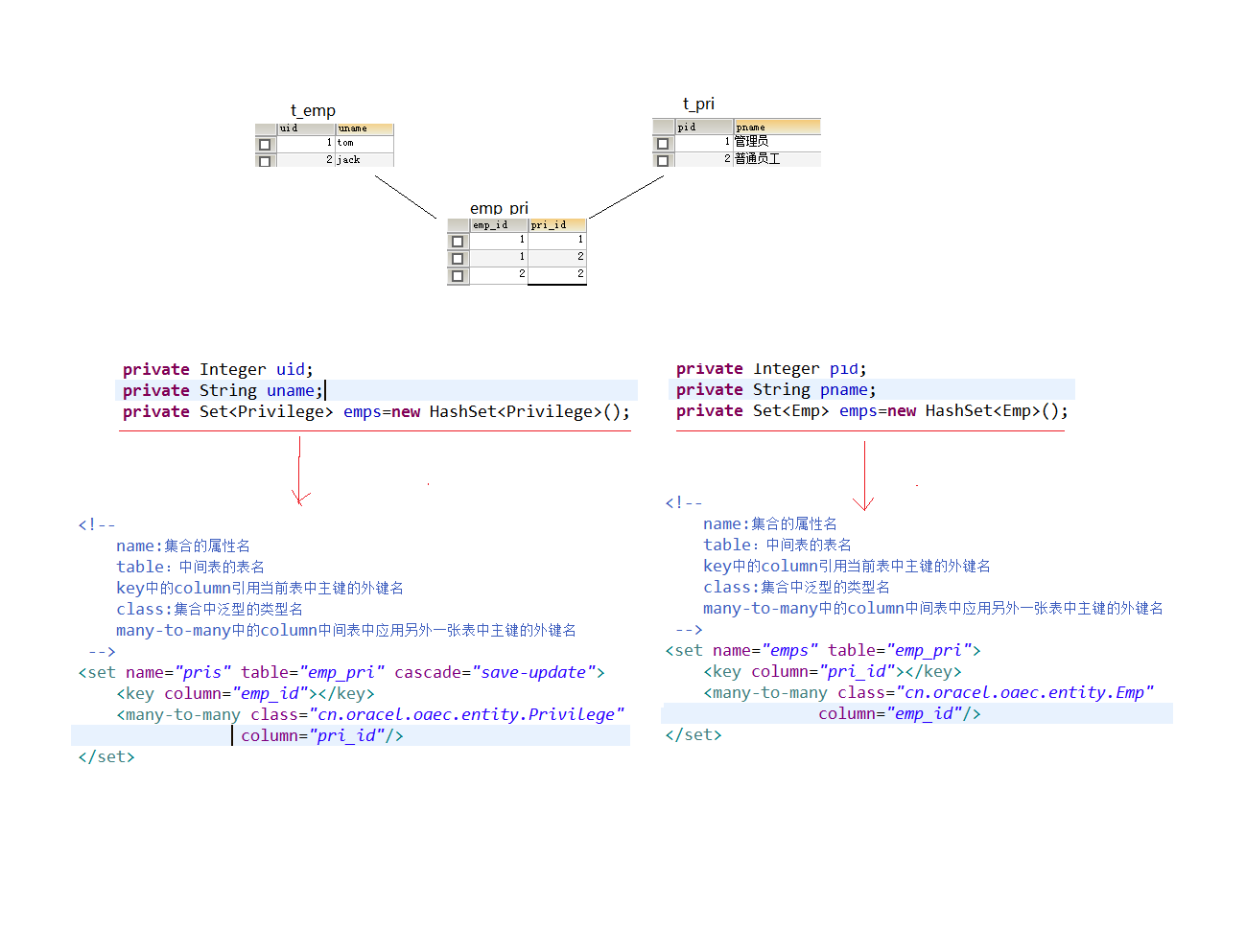
★★★★★★★★★★★★★★★★★★★★★

在<many-to-one>里面配置 cascade属性

在<set>里面配置 inverse="true"

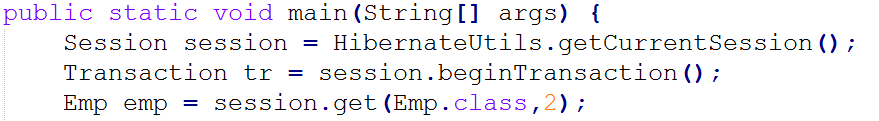
★★★★★★★★★★★★★★★★★★★★★

多对多



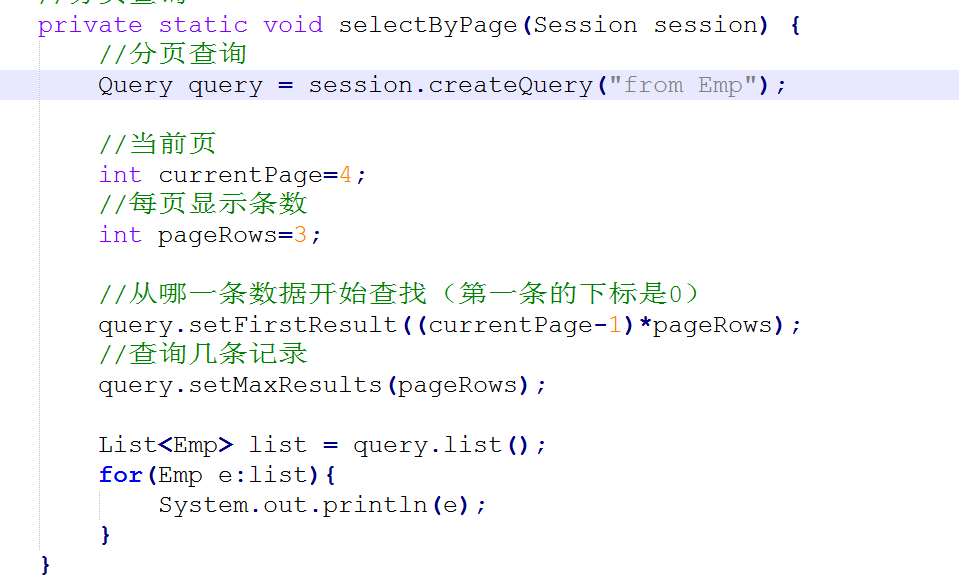
Hibernate查询

1. OID查询

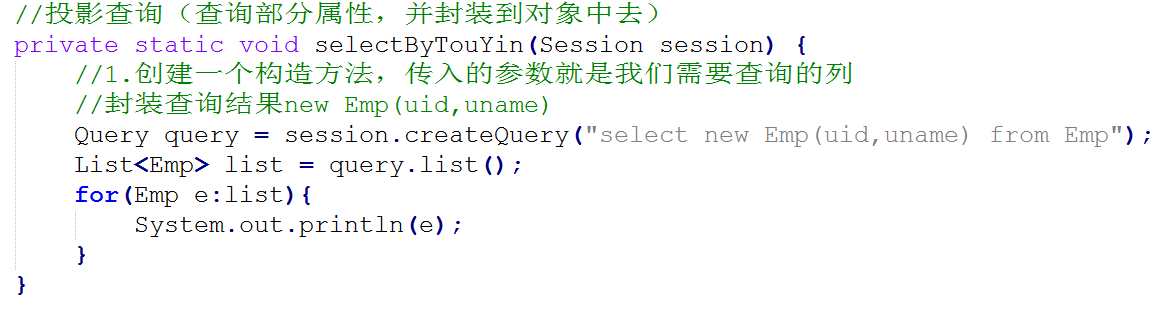


1. HQL查询

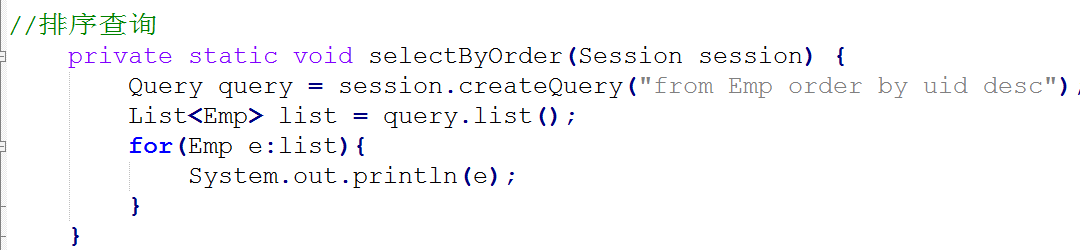
1.分页查询



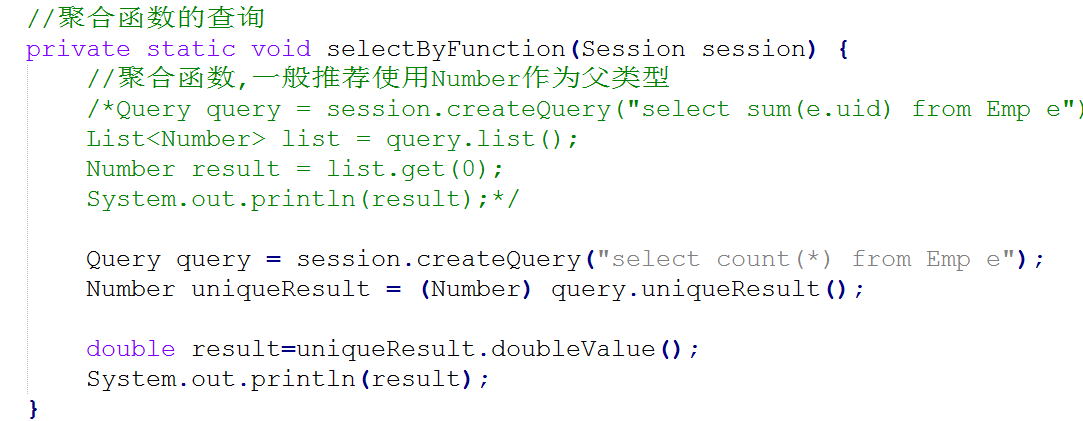
2.投影查询



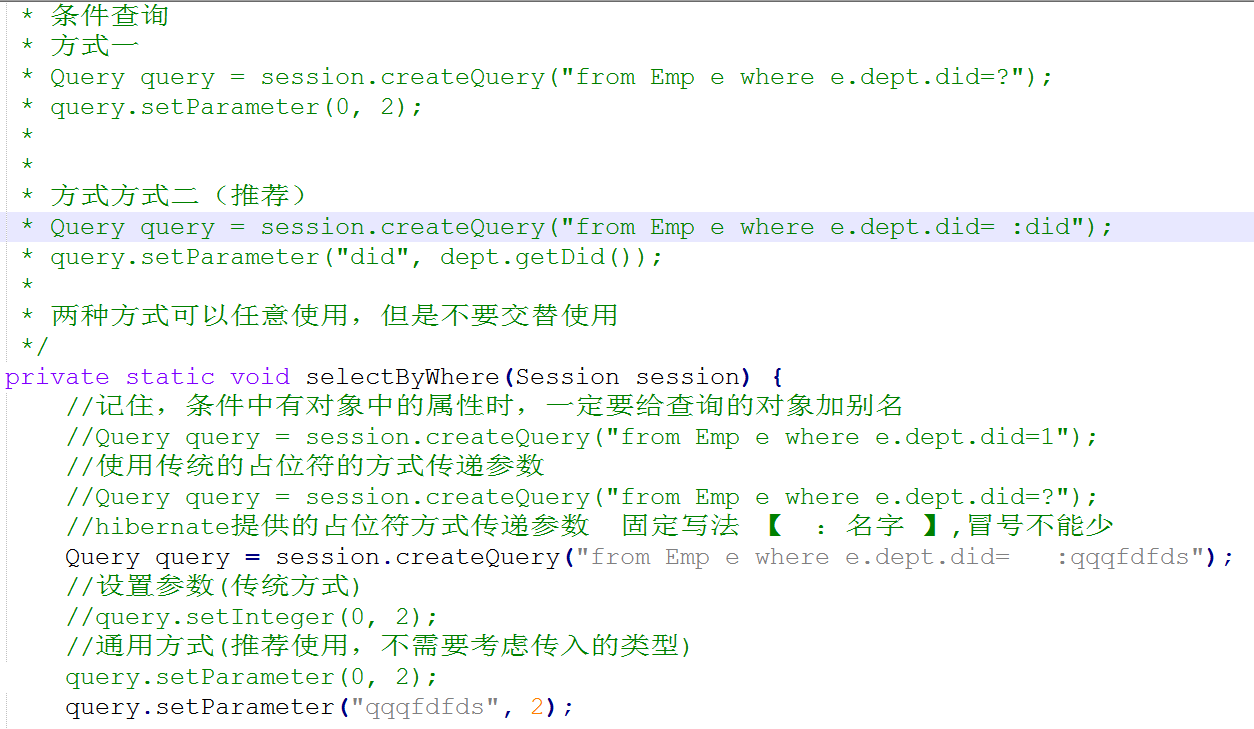
3．排序查询



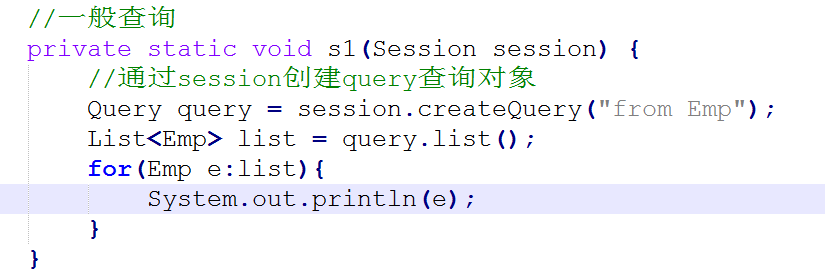
1. 聚合函数的查询



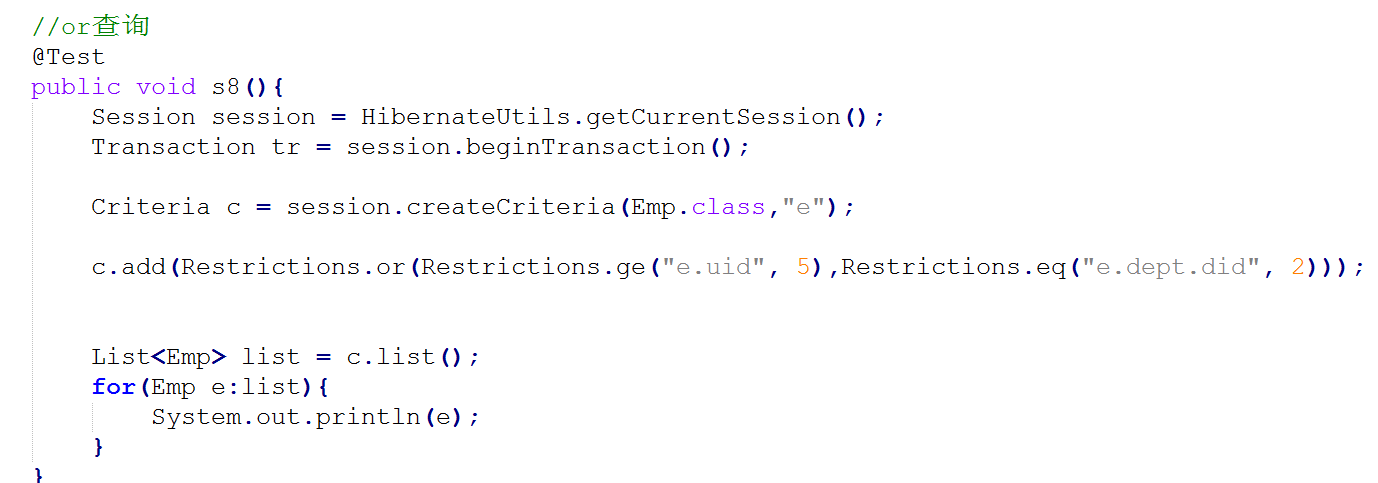
4.条件查询



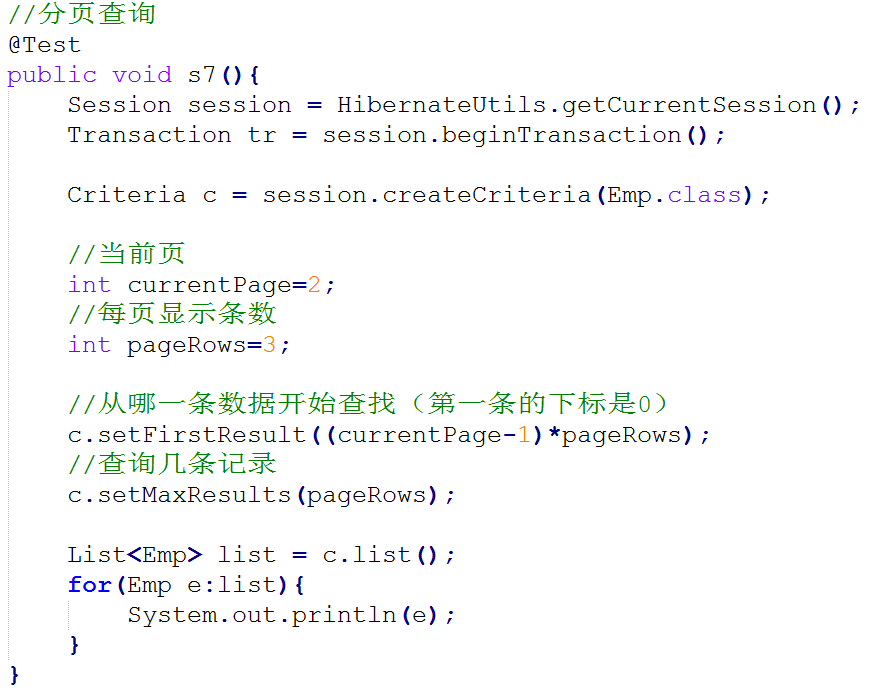
5.一般查询



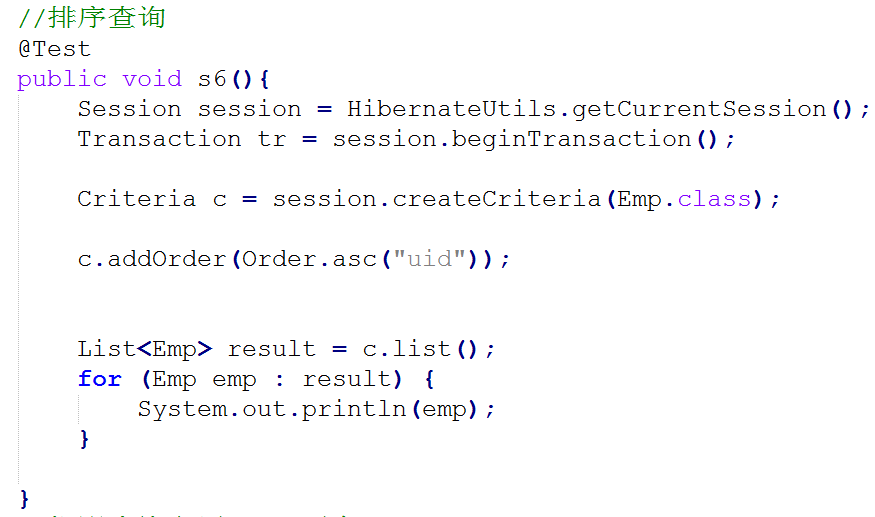
1. QBC查询
2. or查询



2.分页查询

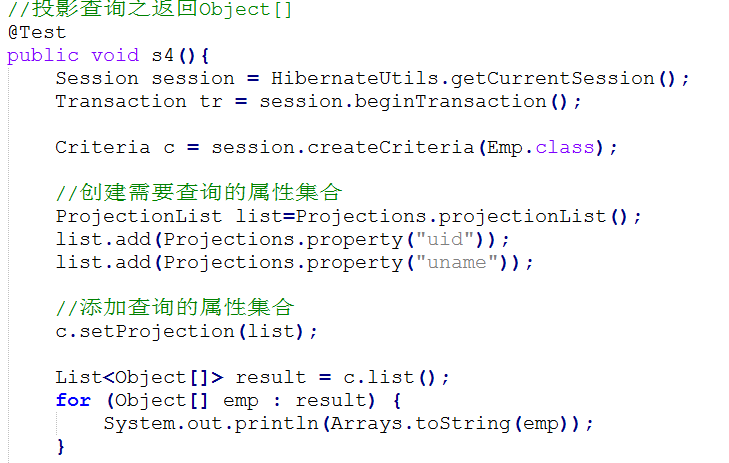


3.排序查询

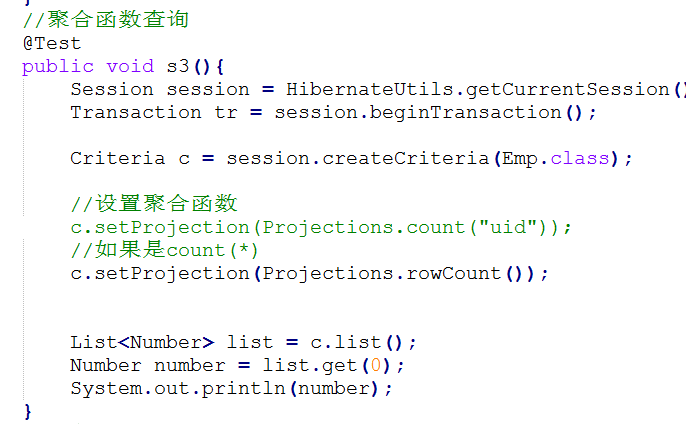


4.投影查询

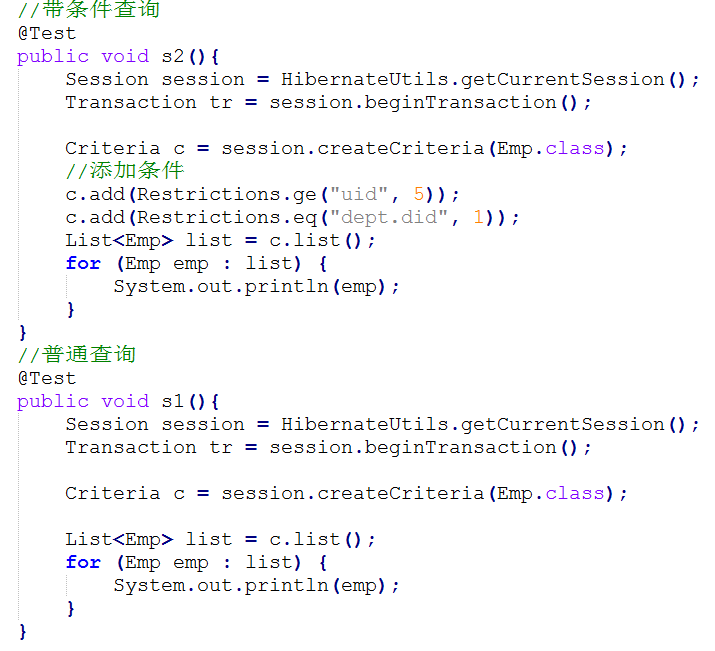




5.聚合函数查询



6.带条件查询



**Spring框架**

1.ioc

控制反转

引入包

层次结构

spring-framework-4.2.4.RELEASE-dist

spring-framework-4.2.4.RELEASE

libs

1.beans

2.context

3.core

4.expression

层次结构

spring-framework-3.0.2.RELEASE-dependencies

org.apache.commons

com.springsource.org.apache.commons.logging\1.1.1

5.com.springsource.org.apache.commons.logging-1.1.1.jar

spring-framework-3.0.2.RELEASE-dependencies

org.apache.log4j

com.springsource.org.apache.log4j\1.2.15

6.com.springsource.org.apache.log4j-1.2.15.jar

applicationContext.xml的dtd约束配置

spring-framework-4.2.4.RELEASE-dist

spring-framework-4.2.4.RELEASE

docs

spring-framework-reference

html

xsd-configuration.html

拉到最下面

复制如下代码

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="

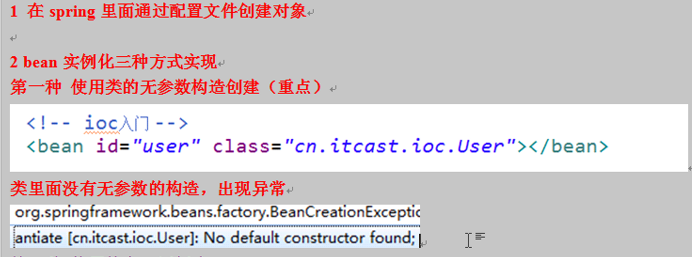
<http://www.springframework.org/schema/beans> http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="foo" class="x.y.Foo">

<meta key="cacheName" value="foo"/>

<property name="name" value="Rick"/>

</bean>

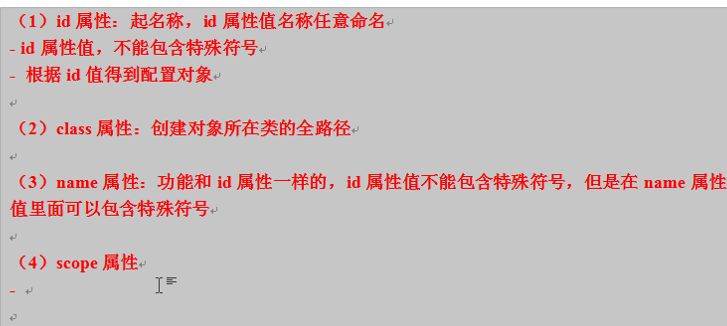


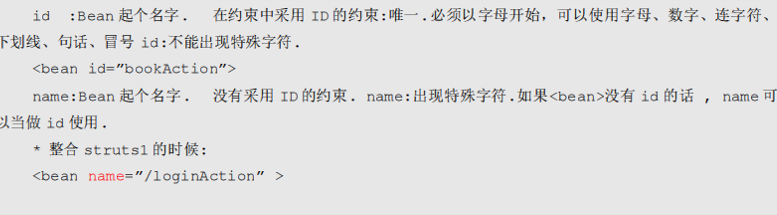
C:\Users\OracleOAEC\AppData\Roaming\Tencent\Users\361370668\QQ\WinTemp\RichOle\%9VVFZ[NKGX~WST7)D8]8CQ.png





**Bean**标签属性







1. DI依赖注入

spirng 的文件管理，通过import标签

<import resource="cn/oracle/oaec/demo1/entity.xml"/>

1.详细讲解<bean>标签

见配置文件entity.xml

2.di

**A.属性注入**

**普通属性注入三种方法**

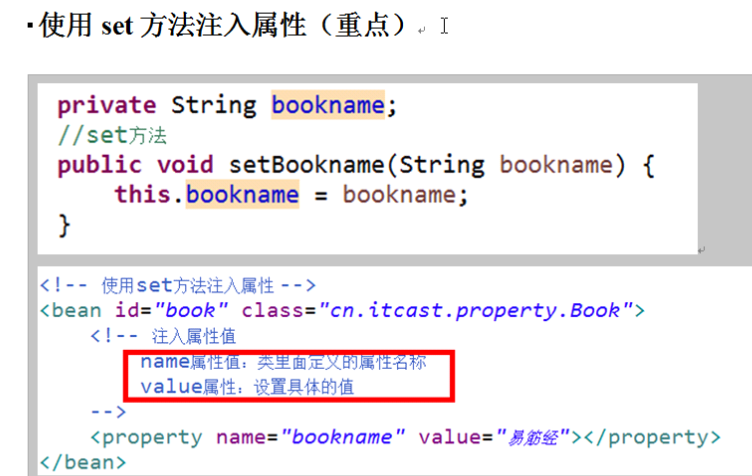
**1使用set方法注入**

**2有参构造方法注入**

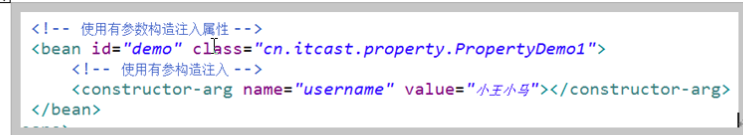
**3使用接口注入**

**Spring属性注入的方法**

**1 Set方法注入（开发重点使用）**



**2有参构造方法注入**



**源码**

<bean id="aaa" class="cn.oracle.oaec.demo1.Dept">

<property name="name" value="部门1"></property>

</bean>

<bean id="p" class="cn.oracle.oaec.demo1.Person" init-method="init" destroy-method="destory">

<!--

注入属性

name:代表类中的属性名,必须跟属性名一致

value：给基本数据类型包括Stirng赋值,注意，一定要用双引号把数据包起来

ref：给引用类型赋值

-->

<property name="name" value="张三"></property>

<property name="age" value="12"></property>

<property name="score" value="100"></property>

<property name="dept" ref="aaa"></property>

</bean>

B.构造方法注入

<bean id="aaa" class="cn.oracle.oaec.demo1.Dept">

<!--

构造注入

name：参数列表中的名字，必须一致

value：给基本数据类型包括Stirng赋值,注意，一定要用双引号把数据包起来

ref：给引用类型赋值

index:就是参数列表的顺序

-->

<constructor-arg name="name" value="部门1"/>

</bean>

<bean id="p" class="cn.oracle.oaec.demo1.Person" init-method="init" destroy-method="destory">

<!-- 方式一 （推荐方式） -->

<!-- <constructor-arg name="score" value="100" index=""/>

<constructor-arg name="dept" ref="aaa"/>

<constructor-arg name="age" value="12"/>

<constructor-arg name="name" value="张三"/> -->

<!-- 方式二 -->

<constructor-arg index="2" value="100"/>

<constructor-arg index="3" ref="aaa"/>

<constructor-arg index="1" value="12"/>

<constructor-arg index="0" value="张三"/>

</bean>

C. p名称空间注入



格式

p:属性名="值"

p:属性名-ref="值"

dtd约束

<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 id="p" class="cn.oracle.oaec.demo1.Person" init-method="init" destroy-method="destory"

p:score="100" p:age="12" p:name="tom" p:dept-ref="aaa">

</beans>

d.spEL表达式注入

它有一个好处是可以调用对象中的方法，并将方法的返回值作为属性的值

{#\*\*\*}

注入集合

1.数组

<property name="arrs">

<!-- <list>

<value>tom1</value>

<value>tom2</value>

<value>tom3</value>

</list> -->

<array>

<value>tom1</value>

<value>tom2</value>

<value>tom3</value>

</array>

</property>

2.list集合（注意：不能用array标签，只能用list标签）

<property name="al">

<list>

<value>tom1</value>

<value>tom2</value>

<value>tom3</value>

</list>

</property>

3.map集合

<property name="mp">

<map>

<entry key="1" value="t1"/>

<entry key="2" value="t2"/>

<entry key="3" value="t3"/>

</map>

</property>

4.Properties

<property name="pr">

<props>

<prop key="url">url</prop>

<prop key="driver">driver</prop>

<prop key="username">username</prop>

<prop key="passwrod">passwrod</prop>

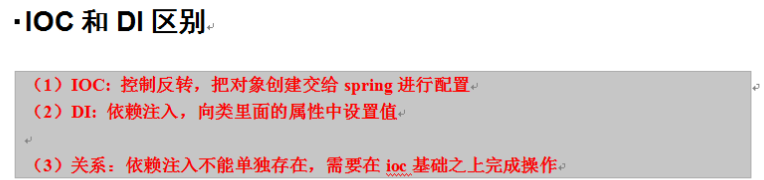
</props>

</property>

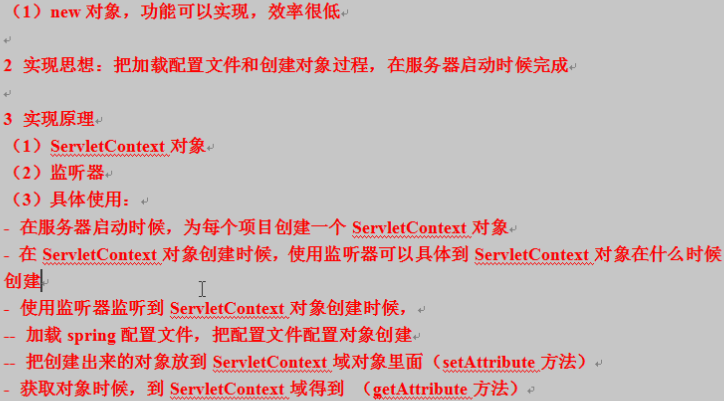
注意点：

spring容器中的每一个bean默认情况下是单例的，就是说只要我们通过同一个id来获取对象，那么都是同一个

但是bean与bean之间不存在单例和多例之分（即使他们的类型一致，属性的值一直，也不存在单例或多例的关系）



Web项目中的应用实现原理



spring的注解

**1.导入jar包**

spring-framework-4.2.4.RELEASE-dist

spring-framework-4.2.4.RELEASE

libs

1.spring-aop-4.2.4.RELEASE.jar aop jar包

2.spring-beans-4.2.4.RELEASE.jar

3.spring-context-4.2.4.RELEASE.jar

4.spring-core-4.2.4.RELEASE.jar

5.spring-expression-4.2.4.RELEASE.jar

spring-framework-3.0.2.RELEASE-dependencies

org.apache.commons

com.springsource.org.apache.commons.logging

1.1.1

6.com.springsource.org.apache.commons.logging-1.1.1.jar

spring-framework-3.0.2.RELEASE-dependencies

org.apache.log4j

com.springsource.org.apache.log4j

1.2.15

7.com.springsource.org.apache.log4j-1.2.15.jar

**2.dtd约束**

<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"> <!-- bean definitions here -->

<!-- 组件扫描 -->

<context:component-scan base-package="cn.oracle.oaec"></context:component-scan>

</beans>



3.IOC注解

@Component(value="d")

如果添加了value值，那么就等同于bean的id就是value值

如果没有添加value值，默认id的值是类名并且首字母小写

\* @Component(value="d") (通用)

\* @Controller (控制层)

\* @Service (业务层)

\* @Repository (持久层)

4.DI注解

如果属性是引用类型

@Autowired(有bug,不推荐)

解决方案

@Qualifier("office")

自动装配，他是根据类型来自动装配的（去找当前类和他的子类的对象）

但是他有一个缺点,就是如果该属性的子类有多个，程序就会报错

@Resource(name="userDaoImp")(推荐使用)

它会根据name中的值去匹配，这个name值其实就是bean中的id

如果属性是基本类型或者是String类型

@Value(value="tom")

@Scope(value="prototype")

@PostConstruct -- 相当于init-method

@PreDestroy -- 相当于destroy-method

@PostConstruct

public void init(){

System.out.println("我是初始方法");

}

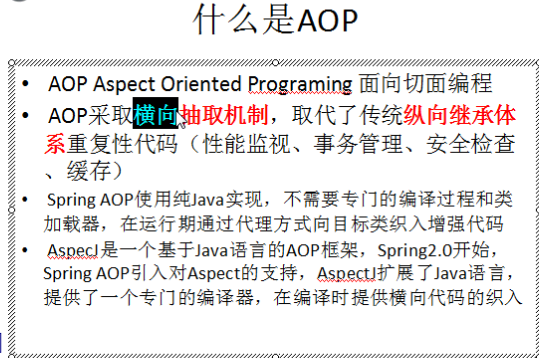
@PreDestroy

public void destory(){

System.out.println("我是销毁方法");

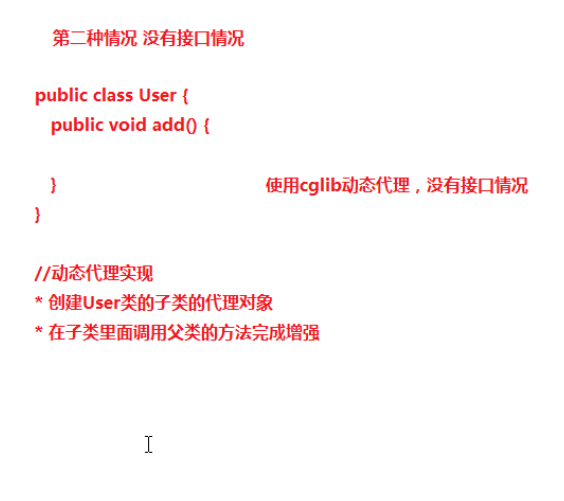
}

Aop面向切口编程



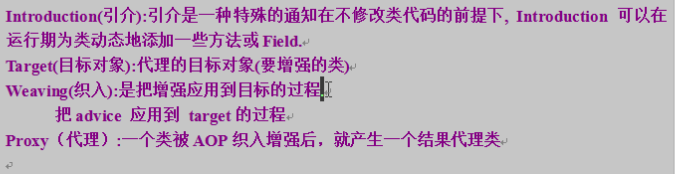
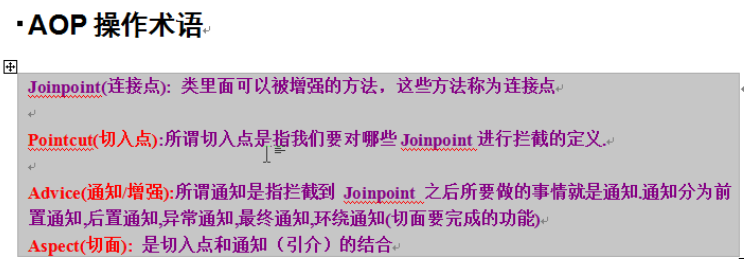
Aop原理

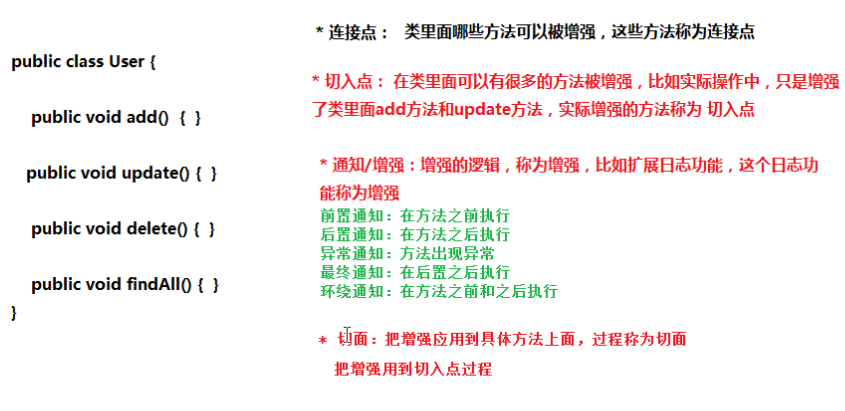




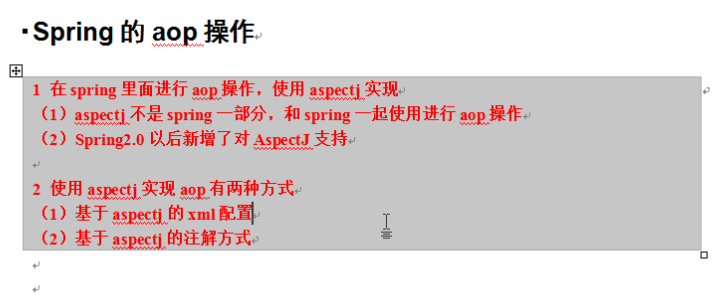


AOP操作术语











1.导入jar包

2.编写applicationContext.xml文件

3.编写具体类

4.测试

spring支持junit测试

1.导入支持的jar包

spring-framework-4.2.4.RELEASE-dist

spring-framework-4.2.4.RELEASE

libs

spring-test-4.2.4.RELEASE.jar

2.添加注解

@RunWith(SpringJUnit4ClassRunner.class)

@ContextConfiguration("classpath:applicationContext.xml")

详细讲解表达式

execution(public void cn.oracle.oaec.service.impl.UserServiceImpl.save( ))

访问权限 public 可以省略不写

方法的返回类型 void 如果想更通用一些 写 \*

包名 cn.oracle.oaec.service.impl 包名可以用通配符 \* 代替

\*.\*.\*.\*.impl 但是不能只写一个 \*，匹配不了

如果想通用 \*..\*

类名 UserServiceImpl 可以使用通配符 \*

User\*

\*ServiceImpl

注意：类名后面一定要加上 点

方法名 save 可以使用通配符 \*

例子:saveUser()

例子:saveCustomer()

save\*()

如果想通用 \*

参数 如果有一个参数 \*

如果两个参数 \*,\*

如果想通用 ..

通用的标准写法

execution( \* \*..\*.\*ServiceImpl.\*(..))

前提是命名要规范

增强/通知

before 在方法之前

after 在方法之后，但是如果程序出项异常，那么增强方法也会执行

after-returnning 在方法之后 ,但是如果程序出现异常，不会执行增强方法

after-throwing 方法出现异常之后执行增强方法

around 方法的前后去执行

public void transaction(ProceedingJoinPoint joinPoint){

try {

System.out.println("开启事务...");

/\*

\* Session s=hiernate.getCurrentSession();

\* Transaction tr=s.begingTransaction();

\*/

//就等同于调用cn.oracle.oaec.service.impl.UserServiceImpl.save();

joinPoint.proceed();

System.out.println("提交事务...");

/\*

\* tr.commit();

\*/

} catch (Throwable e) {

// TODO Auto-generated catch block

e.printStackTrace();

/\*

\* 日志

\* tr.rollback();

\*/

}

}

aop的xml配置

<aop:config>

<aop:aspect ref="myTransaction">

<aop:around method="transaction" pointcut="execution( \* \*..\*.\*ServiceImpl.\*(..))"/>

</aop:aspect>

</aop:config>

aop的注解

注意：切面只能切一次，不能重复