Spring

了解spring

- 为什么需要使用spring　应对企业应用开发的复杂性
- 特点
  - 方便解耦
  - AOP编程支持
  - 方便程序的测试
  - 与其他程序之间的整合
  - 方便进行事务操作
  - 降低了API的使用难度

IOC控制反转

- 入门案例：使用spring来代替创建对象
  - 1.导入spring四大基本jar包
    - bean
    - core
    - aop
    - exception
  - 2.创建实体类
  - 3.创建xml文件
  - 4.创建一个bean对象
  - 5.通过ApplicationContext来装入加载的spring配置文件
  - 6.获取spring配置文件
- spring的基本根源是xml文件管理
- IOC控制反转，指的其实就是把创建对象的过程交给spring
- IOC bean管理中的几种创建对象的方式
  - 1.通过set方法注入属性
    - 需要set方法
    - 关键字：property
  - 2.通过有参构造来注入属性
    - 需要有参构造方法
    - 关键字：constructor-arg
    - 有两种注入的方式
      - name：value
      - index：value
  - 3.通过p名称空间来注入属性（不常用）　需要修改名称空间，将beans改成p
- IOC bean管理
  - xml注入其他类型的属性
    - 注入属性：字面量
      - null值　<value>用<null>代替
      - 属性值的特殊符号

```
<property name="bauthor">
    <value>
        <![CDATA[<<ljr>>]]>
    </value>
</property>
```

    - 注入属性：外部bean　需要在bean内引用另一个bean，需要ref属性，属性值为被引用的bean的id
    - 注入属性：内部bean和级联赋值
      - 内部bean，直接在bean内再写一个bean即可
      - 级联赋值的两种方式
        - 方式一：使用ref引用另一个bean 的值
        - 方式二：使用类似包名的方式查找属性
          如： name=“dept.name” value=“财务部”
    - 注入集合属性CollectionType
  - Factorybean，IOC核心：工厂模式
    - 什么是工厂模式?
      工厂模式就是让spring来完成对象的创建和属性的注入，（完成生产任务）
      通过识别配置文件的内容：来自动的生成相对应的类和对象
      但是这样也有着一定的缺点：容易生成太多的工厂类，拖慢速度
    - 为什么要使用IOC来代替工厂模式?
      工厂模式中创建对象一般都是线性的，在对象中创建需要的其他对象
      这样就不可避免的导致代码之间的耦合度变高
      而使用IOC就将对象创建的需求交给了子类，需要的时候再创建相对应的类和对象
      从而省去了程序员不断的对类对象的操作，创建和销毁，另外为什么叫控制反转，
      是因为子类来主动创建类，父类被动等待，然后等待子类的注入
  - bean 的生命周期
    - 1.通过构造器来创建bean实例　无参构造
    - 2.为bean的属性设置值和对其他bean进行引用　通过set方法
    - 3.调用bean 的初始化方法　需要进行配置初始化方法
    - 4.可以对bean进行使用了　获取相应的对象
    - 5.当容器关闭时，需要对bean销毁的方法　需要进行配置销毁的方法
  - xml的自动装配　autowire属性
    - ByName
    - ByType
    - No
    - Constructor
  - 通过注解方式来实现
    - Spring的Bean管理的中常用的注解　@Component 组件（常用于class）
      - 三个衍生注解：@Controller　一般用于controller层
      - @Service　一般用于service层
      - @Repository　持久层
    - 属性注入常用的注解
      - @Value　用于注入普通类型.
      - @Autowired
        - 默认按类型进行装配
        - 按名称注入
      - @Qualifier　强制使用名称注入
      - @Resource　相当于:@Autowired和@Qualifier一起使用

AOP面向切面

- 代理模式
  - 静态代理
  - 动态代理

jdbcTemplate操作数据库

事务操作