spring-offcn

学习目标：学完之后我希望自己遇到spring问题可以秒级排错

1.什么是spring，IOC是什么

spring是整合应用开发的一款框架，IOC中文叫控制反转，控制什么，反转什么，把以前从手动new对象的操作交割spring这个容器来进行创建，我们直接从容器中获取对象即可，容器是如何创建对象的呢：说生命周期。

2.sql语句执行顺序

from-->on/join --->where----group by/having----->select -->DISTINCT---->order by------>limit

参考连接：https://blog.csdn.net/ai_shuyingzhixia/article/details/80559155

3.spring整合junit4

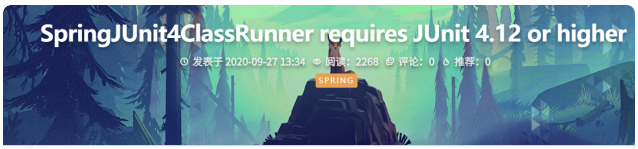
依赖：导入依赖的时候需要junit 和spring-junit连个包

```java
// Spring整合Junit4运行环境
@RunWith(SpringJUnit4ClassRunner.class)
// 初始化容器 new 容器
@ContextConfiguration("classpath:beans.xml")
public class MyTest {

    @Autowired
    private PersonDao personDao;

    @Test
    public void testQuery(){
        System.out.println(personDao.queryById(1));
    }

}
```



报错:需要Junit 4.12 或者更高的版本

原因:查看pom.xml文件引入的Junit版本为4.11

解决办法:修改Junit版本为4.12即可

spring这和test 导入的junit4.12or higher

4.spring整合mybatis注意点

依赖：mybatis 和 mybatis-spring

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
        PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <!--  MyBatis中的配置托管给了Spring容器  -->
</configuration>
```

setting设置一般在这，分页插件，缓存，延迟加载等

```xml
<!--
    spring与Mybatis整合配置
        1。整合环境信息+核心配置
        2。将接口对象注入到容器中
        Mybatis支持多环境开发：开发环境、工作环境、测试环境。。。
-->
<bean id="sqlSessionFactoryBean" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource"></property>
    <property name="typeAliasesPackage" value="com.jeff.pojo"></property>
    <property name="mapperLocations" value="classpath:mappers/*.xml"></property>
    <property name="configLocation" value="classpath:mybatis-config.xml"></property>
</bean>

<bean id="mapperScannerConfigurer" class="org.mybatis.spring.mapper.MapperScannerConfigurer">

    <property name="sqlSessionFactoryBeanName" value="sqlSessionFactoryBean"></property>
    <property name="basePackage" value="com.offcn.mapper"></property>
</bean>
```

5.spring整合数据源

- c3p0------ComboPooledDataSource
- druid------DruidDataSource

- spring-jdbc--------DriverManagerDataSource
- DBCP----------BasicDataSource

1. 注意整合数据源就是注入响应的bean --------可以配合任意的jdbc工具使用 mybatis dbuits jdbc-Template
2. 整合mybatis
3. 整合dbutis commons-dbutils 注入bean即可 QueryRunner(ds)
4. 整合jdbc-Template 注入JdbcTemplate即可 数据源要配置

整合说明一点：在xml整合的时候如果不知道那些类需要那些属性，可以点击类到源码查看

6.学到此处要知道的面试题

1. spring是什么，什么是IOC
2. IOC如何创建对象放到容器中,也就是生命周期
3. 依赖注入的形式有那些
4. 你用过那些spring注解
5. bean的作用域有那些，分别什么意思
6. beanFactory 和ApplicationContext和FactoryBean有什么区别
7. 如何用xml的形式进行静态工厂和实例工厂初始化
8. spring事务的传播机制 7种
9. spring事务的隔离级别,分别防止了什么
10. 什么事务，事务有那些特性，数据库如何控制事务
11. spring中配置事务有几种方式,spring事务的原理是什么实现的aop
12. aop的增强器有几种，分别是什么
13. 什么是aop ，spring如何实现aop的
14. 动态代理有那些
15. spring中用了那些设计模式

7.spring中的事务配置

1. 事务对象：DataSourceTransactionManager
2. 配置方式

注解配置：@bean @EnableTransactionManagement 事务注解

xml配置：

```xml
    <bean class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
          id="transactionManager">
        <property name="dataSource" ref="dataSource"></property>
    </bean>
<tx:annotation-driven transaction-manager="transactionManager">
</tx:annotation-driven>
```

aop配置

```xml
    <bean class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
          id="transactionManager">
        <property name="dataSource" ref="dataSource"></property>
    </bean>
    <tx:advice id="txAdvice" transaction-manager="transactionManager">
        <tx:attributes>
            <tx:method name="moneyCount" read-only="false"/>
        </tx:attributes>
    </tx:advice>
    <!--    配置置入-->
    <aop:config>
        <aop:advisor advice-ref="txAdvice" pointcut="execution(* com..service.CountServiceImpl.*(..))"></aop:advisor>
    </aop:config>
```

8.通知表达式

- 格式：返回值 类全限定名.方法名（参数类型）
- 简化含义

```
"execution(void com.offcn.service.AccountServiceImpl.transfer(String,String,double))
```

```
切点表达式：spring进行事务管理时可以根据切点表达式去查找方法，对方法进行事务管理
void com.offcn.service.AccountServiceImpl.transfer(String,String,double)
void com.offcn.service.AccountServiceImpl.transfer(..)
```

```
    transfer
    切点表达式：spring进行事务管理时可以根据切点表达式去查找方法，对方法进行事务管理
    void com.offcn.service.AccountServiceImpl.transfer(String,String,double)
    void com.offcn.service.AccountServiceImpl.transfer(..)
    * com.offcn.service.AccountServiceImpl.transfer(..)
-->
```

```
    切点表达式：spring进行事务管理时可以根据切点表达式去查找方法，对方法进行事务管理
    void com.offcn.service.AccountServiceImpl.transfer(String,String,double)
    void com.offcn.service.AccountServiceImpl.transfer(..)
    * com.offcn.service.AccountServiceImpl.*(..)
-->
```

- 中间目录 * com..service..*(..)中间的两个点代表某一级目录不要下的所有service包

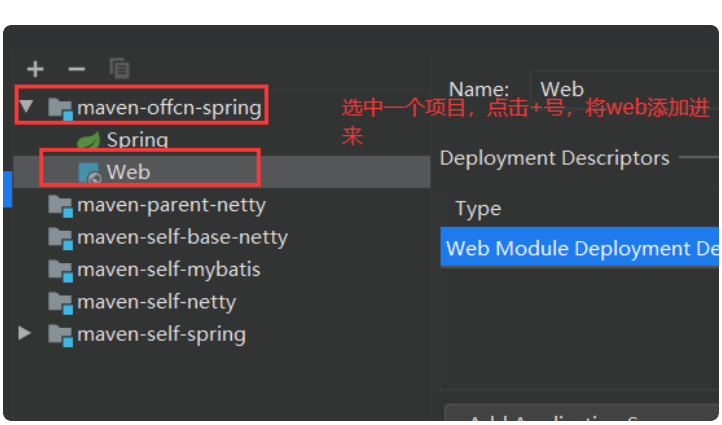
9.aop的xml配置实现

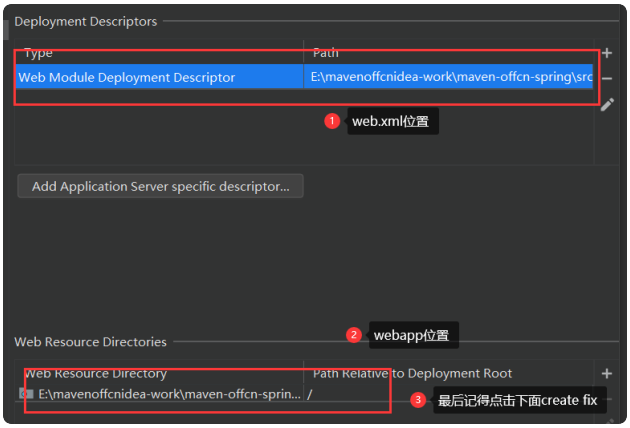
aop的xml配置 XML 复制代码

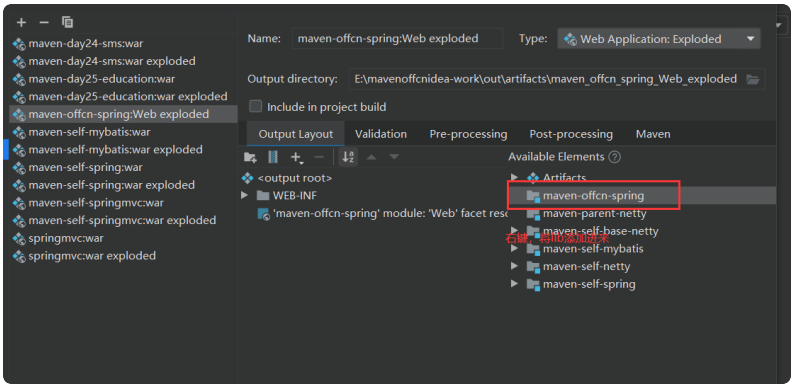
```xml
    <!--    注入切面类-->
    <bean id="txManagerAspect" class="com.edu.aop.TxManagerAspect">
</bean>
    <!--    注入业务类-->
    <bean id="aopService" class="com.edu.service.AopServiceImpl"></bean>

    <aop:config>
        <!--定义切点-->
        <aop:pointcut id="pc" expression="execution(void
com.edu.service.AopServiceImpl.aopAspect(..))"/>
        <!--配置切面-->
        <aop:aspect ref="txManagerAspect">
            <aop:before method="begin" pointcut-ref="pc"></aop:before>
            <aop:after-returning method="commit" pointcut-ref="pc">
</aop:after-returning>
            <aop:after-throwing method="rollback" pointcut-ref="pc">
</aop:after-throwing>
            <aop:after method="close" pointcut-ref="pc"></aop:after>
<!--            <aop:around method="around" pointcut-ref="pc">
</aop:around>-->
        </aop:aspect>
    </aop:config>
```

10.自定义webapp项目







put into out root