day06_6_12 spring高级

1. 通过事务切面学习aop

事务管理器对像：MyTransactinManager:是一个管理器工具对象，用来打开，关闭，回滚链接

事务切面：TxAspect：把通知织入带有Tx注解的连接点方法

THreadLocal对象：线程变量对象，在多线程的情况下，确保线程安全的做法

ApplicationContextAware接口：用来获取spring容器引用的接口，需要重写setApplicationContext

方法

- 解决静态类对象中获取容器中的对象的问题
- 在子线程下怎么获取容器中的对象

```
@Component
public class SpringUtil implements
ApplicationContextAware {
    private static ApplicationContext
applicationContext;//静态被所有实例共享

    @Override
    public void
setApplicationContext(ApplicationContext ac) throws
BeansException {
        applicationContext=ac;
    }

    public static <T> T getBean(Class<T> clz){
        return applicationContext.getBean(clz);
    }
}
```

2. 学习spring的AnnotationConfigApplicationContext容器对象

BeanFactory----->DefaultListableBeanFactory

BeanFactory------>ApplicationConqtext-----
>ClassPathXMLApplicationContext

AnnotationConfigApplicationContext

- ClassPathXMLApplicationContext 加载xml配置文件中的.xml配置的方式应用在ssm框架中
- AnnotationConfigApplicationtext加载class类配置文件的，类配置文件的方式应用在springboot框架中
- 类配置的常用的注解

```
@Configuration // 定义配置类
@Bean //注册bean
@PropertySource // 加载properties文件，把配置数据注册进容器中
@ComponentScan  // 开启包扫描
@EnableAspectjAutoProxy // 开启aspectj注解识别
@Import // 引入其他的配置类
@Value  // 获取容器中的properties配置数据
@ImportResource  // 引入其他xml配置文件
-------------------------------------------------------
//基础的配置类
@Configuration//表示当前类是一个配置类,同时该类会被注册容器
@ComponentScan("com.javasm")//<context:component-scan>
@EnableAspectJAutoProxy//开启aop注解识别
@Import(DaoConfig.class)
@ImportResource("classpath:dao.xml")
public class AppConfig {

    //先按照形参名注入,再按照形参类型注入值
    @Bean
    public SqlDaoImpl createUserDao(DataSource dataSource){
```

```
        SqlDaoImpl sysuserDao = new SqlDaoImpl();
        sysuserDao.setDataSource(dataSource);
        return  sysuserDao;
    }
}
-----------------------------------------------------------
//jdbc数据库连接的配置文件
@Configuration
@PropertySource("classpath:jdbc.properties")//context:property-placeholder
public class DaoConfig {
    @Value("${jdbc.url}")
    private String url;
    @Value("${jdbc.driverClassName}")
    private String driverClassname;
    @Value("${jdbc.username}")
    private String username;
    @Value("${jdbc.password}")
    private String password;
    @Value("${jdbc.initialSize}")
    private Integer initSize;

    //把方法的返回值注册容器,id默认是方法名
    @Bean(initMethod = "init",destroyMethod = "close")
    public DataSource createDruidDataSource(){
        DruidDataSource druidDataSource = new DruidDataSource();
        druidDataSource.setUrl(url);

 druidDataSource.setDriverClassName(driverClassname);
        druidDataSource.setUsername(username);
        druidDataSource.setPassword(password);
        druidDataSource.setInitialSize(initSize);
        return druidDataSource;
    }
}
```

3. 写下mybatis的几张表的操作