链滴

Spring 整合 Mybatis

作者：Wuhon
原文链接：https://ld246.com/article/1608356804563
来源网站：链滴

Spring整合Mybatis

需要在pom.xml里加入以下依赖

```
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>5.2.9.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
        <version>2.0.3</version>
    </dependency>
```

1. 编写数据源配置

```
<!-- DataSource:  使用spring的数据源替换mybatis的配置
     我们这里使用Spring提供的jdbc org.springframework.jdbc.datasource.DriverManagerDataSource
  -->
  <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
      <property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
      <property name="url" value="jdbc:mysql://localhost:3306/mybatis?useUnicoding=true&amp;characterEncoding=UTF-8&amp;useSSL=true"/>
      <property name="username" value="root"/>
      <property name="password" value="111"/>
  </bean>
```

2. 在配置文件里注入sqlSessionFactory

```
<!-- sqlSessionFactory  -->
  <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
      <property name="dataSource" ref="dataSource" />
      <!-- 绑定mybatis配置文件 -->
      <property name="configLocation" value="classpath:mybatis-config.xml"/>
      <property name="mapperLocations" value="classpath:wuhobin/dao/*.xml"/>
  </bean>
```

3. 在配置文件里注入SqlSessionTemplate

```
<!-- SqlSessionTemplate 就是我们要使用的sqlSession -->
  <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
      <!-- 只能使用构造器注入sqlSessionFactory 没有set方法  -->
      <constructor-arg index="0" ref="sqlSessionFactory"/>
  </bean>
```

4. 需要给接口加实现类

```
public class UserMapperImpl implements UserMapper{
  // 我们所有操作都使用sqlSession来执行原来 现在使用SqlSessionTemplate
  private SqlSessionTemplate sqlSession;
```

```
    // 添加set方法 方便在bean中注入
    public void setSqlSession(SqlSessionTemplate sqlSession) {
        this.sqlSession = sqlSession;
    }

    public User getUserById(int id) {
        UserMapper mapper = sqlSession.getMapper(UserMapper.class);
        return mapper.getUserById(id);
    }
}
```

5. 配置声明式事务

```
<!-- 配置声明式事务  -->
    <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceT
ansactionManager">
        <property name="dataSource" ref="dataSource"/>
    </bean>

    <!--  结合aop实现事务的织入  -->
    <!--  配置事务的类  -->
    <tx:advice id="txAdvice" transaction-manager="transactionManager">
        <!--    给哪些方法配置事务    -->
        <!--    配置事务的传播特性 propagation 默认使用REQUIRED  -->
        <tx:attributes>
          <tx:method name="*" propagation="REQUIRED" />
        </tx:attributes>
    </tx:advice>

    <!--  配置事务切入  -->
     <aop:config>
         <aop:pointcut id="txPointCut" expression="execution(* wuhobin.dao.*.*(..))"/>
         <aop:advisor advice-ref="txAdvice" pointcut-ref="txPointCut"/>
     </aop:config>
```

6. 将自己写的实现类注入到spring中

```
<bean id="userMapper" class="wuhobin.dao.UserMapperImpl">
        <property name="sqlSession" ref="sqlSession"/>
 </bean>
```

7. 测试

```
public class MyTest {
    @Test
    public void demo1(){
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.
ml");
        UserMapperImpl userMapper = (UserMapperImpl) context.getBean("userMapper");
        System.out.println(userMapper.getUserById(1));
    }
}
```