limeOracle

管理

Spring-Mybatis --- 配置SqlSessionFactoryBean，整合Spring-Mybatis

要利用Mybatis首先是需要导入mybatis-x.x.x.jar，其次，要整合Spring和Mybatis需要导入mybatis-spring-x.x.x.jar。

JAR： mybatis-x.x.x

```
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.2.6</version>
</dependency>
```

JAR： mybatis-spring-x.x.x

```
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.2.2</version>
</dependency>
```

1、Spring整合Mybatis的xml配置

xml： POM

```
<!-- DB -->
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.1.41</version>
</dependency>

<dependency>
    <groupId>org.apache.tomcat</groupId>
    <artifactId>tomcat-servlet-api</artifactId>
    <version>7.0.54</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.tomcat</groupId>
    <artifactId>tomcat-jdbc</artifactId>
    <version>7.0.23</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.18</version>
</dependency>
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.2.2</version>
</dependency>
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.2.6</version>
</dependency>
```


搜索

找找看

谷歌搜索

随笔分类

GoF(10)
ALIME(1)
CSS(2)
DOS Commands(2)
Dubbo(27)
EasyUI(1)
Echarts(3)
Eclipse(5)
Hadoop(5)
Hibernate(2)
HibernateLimeOracle(26)
JAVA(74)
JAVA Annotation(1)
Java Collection(3)
Java IO(3)
JAVA NIO(16)
JAVA RPC(13)
Java Throwable(7)
JAVA WEB(19)
JavaScript(15)
JavaScript BOM(3)
jQuery(4)
Linux(33)
LogBack(5)
Maven(6)
mongodb(8)
MyBatis(13)
MySQL(39)
Netty(7)
Nexus(1)
Nginx(3)
POI(2)
Principles of Computer Composition (2)
Redis(2)
RMI(1)
Spring(24)
Spring @Annotation(1)
SpringLimeOracle(61)
SpringMVC(41)
ThinkingInJava(5)
Tomcat(4)
TWaver HTML5 (2D)(2)
草稿-未总结的小记(23)
数据结构及算法(11)

阅读排行榜

1. Web.xml配置详解(20741)
2. MySQL---循环语句(20549)
3. Spring-Mybatis --- 配置SqlSessionFactoryBean，整合Spring-Mybatis(20216)
4. 设置DIV最小高度以及高度自适应随着内容的变化而变化(6643)

xml： DataSource

```
    <bean id="dataSource" class="org.apache.tomcat.jdbc.pool.DataSource" destroy-method="close">
        <property name="poolProperties">
            <bean class="org.apache.tomcat.jdbc.pool.PoolProperties">
                <property name="driverClassName" value="com.mysql.jdbc.Driver" />
                <property name="url" value="${jdbc.url}" />
                <property name="username" value="${jdbc.user}" />
                <property name="password" value="${jdbc.password}" />
                <!-- Register the pool with JMX. In order for the connection pool object to create the MBean. -->
                <property name="jmxEnabled" value="true" />
                <!-- The indication of whether objects will be validated by the idle object evictor. -->
                <property name="testWhileIdle" value="true" />
                <!-- The indication of whether objects will be validated before being borrowed from the pool. -->
                <property name="testOnBorrow" value="false" />
                <property name="testOnReturn" value="false" />
                <property name="initialSize" value="${jdbc.initialPoolSize}" />
                <property name="maxActive" value="${jdbc.maxActive}" />
                <property name="maxWait" value="${jdbc.maxWait}" />
                <property name="minIdle" value="${jdbc.minIdle}" />
                <property name="maxIdle" value="${jdbc.maxIdle}" />
                <property name="maxAge" value="60000" />
                <!-- The number of milliseconds to sleep between runs of the idle connection validation/cleaner thread. -->
                <property name="timeBetweenEvictionRunsMillis" value="15000" />
                <!-- The minimum amount of time an object may sit idle in the pool before it is eligible for eviction. -->
                <property name="minEvictableIdleTimeMillis" value="60000" />
                <property name="removeAbandoned" value="true" />
                <property name="removeAbandonedTimeout" value="30" />
                <property name="validationQuery" value="SELECT 1" />
                <property name="validationInterval" value="30000" />
            </bean>
        </property>
    </bean>
```

常用配置：

（如果在mybatis-config.xml利用<mappers>进行xml映射文件的配置，就可以不用配置下面的mapperLocation属性了）

```
<!-- mybatis文件配置，扫描所有mapper文件 -->
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean"
      p:dataSource-ref="dataSource"
      p:configLocation="classpath:mybatis-config.xml"
      p:mapperLocations="classpath:com/eliteams/quick4j/web/dao/*.xml"/>

<!-- spring与mybatis整合配置，扫描所有dao，在单数据源的情况下可以不写sqlSessionFactoryBeanName -->
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer"
      p:basePackage="com.eliteams.quick4j.web.dao"
      p:sqlSessionFactoryBeanName="sqlSessionFactory"/>
```

----------阿弥陀佛----佛祖保佑----永无BUG----------

2、Spring和Mybatis整合的三种方式

—

- SqlSessionFactoryBean来替代SqlSessionFactoryBuilder来创建SqlSession
- 利用mybatis映射文件**.xml来配置

SqlSessionFactoryBean有一个必须属性dataSource，另外其还有一个通用属性configLocation（用来指定mybatis的xml配置文件路径）。

5. mysql timestamp类型字段的CURRENT_TIMESTAMP与ON UPDATE CURRENT_TIMESTAMP属性(6000)
6. Java -- 获取指定接口的所有实现类或获取指定类的所有继承类(5741)
7. spring定时任务详解（@Scheduled注解）多线程讲解(5480)
8. 如何查看连接mysql的ip地址(5337)
9. @ControllerAdvice(5280)
10. 一个tomcat设置多个端口，多个端口对应多个应用(5130)
11. mybatis if判断中的特殊符号(4629)
12. Spring @Lazy(3589)
13. easyui中对于dialog页面传值的接收(3068)
14. Eclipse Maven 配置setting.xml 的镜像远程仓库(2750)
15. PostMan 发送list<Object>(2458)
16. 8 -- 深入使用Spring -- 5...3 使用@CacheEvict清除缓存(2229)
17. HttpServletRequest -- 获取请求主机真实的IP地址(2114)
18. JAVA WEB ------ 文件下载及导出数据到office Execl表格(2079)
19. ECharts 3 -- gauge表盘的配置项(1712)
20. MySQL --- 计算指定日期为当月的第几周(1679)
21. Ubuntu 16.04.1下修改MySQL默认编码(1323)
22. CMD 切换管理员权限(1269)
23. 8 -- 深入使用Spring -- 4...6 AOP代理：基于注解的XML配置文件的管理方式(1182)
24. 日期操作类--GregorianCalendar类(1165)
25. TWaver HTML5 (2D)----数据元素(1122)
26. js将html5日期格式转为long型(1091)
27. Dubbo -- 系统学习 笔记 -- 示例 -- 泛化引用(925)
28. 使用mybatis操作mysql数据库SUM方法返回NULL解决(864)
29. MySQL -- 行转列 -- GROUP_CONCAT -- MAX(CASE WHEN THEN)(831)
30. com.alibaba.fastjson.JSONObject(816)
31. 多线程 TCP 连接(790)
32. ubuntu MySQL采用apt-get install安装目录情况(752)
33. java.lang.Class<T> -- 反射机制(745)
34. Ubuntu 16.04 更新源(724)
35. js将long日期格式转换为标准日期格式(709)
36. DispatcherServlet默认配置(673)
37. Ubuntu 16.04服务器安装及软件配置(643)
38. Dubbo -- 系统学习 笔记 -- 示例 -- 分组聚合(634)
39. 7 -- Spring的基本用法 -- 10... 获取其他Bean的属性值；获取Field值；获取任意方法的返回值(620)
40. Servlet入门总结及第一个Servlet程序(598)

Spring的xml配置：

```
    <!-- 创建SqlSessionFactory，同时指定数据源-->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <!-- 指定sqlMapConfig总配置文件，订制的environment在spring容器中不在生效-->
        <property  name="configLocation"  value="classpath:sqlMapConfig.xml"/>
    </bean>
```

mybatis总配置文件sqlMapConfig.xml：

```
<configuration>
  <typeAliases>
     <typeAlias type="com.xxt.ibatis.dbcp.domain.User" alias="User" />
  </typeAliases>
  <mappers>
     <mapper resource="com/xxt/ibatis/dbcp/domain/userMapper.xml" />
  </mappers>
</configuration>
```

userMapper.xml：

```
<mapper namespace="com.xxt.ibatis.dbcp.dao.UserDao">
    <resultMap type="User" id="userMap">
       <id property="id" column="id" />
       <result property="name" column="name" />
       <result property="password" column="password" />
       <result property="createTime" column="createtime" />
    </resultMap>
    <select id="getUserById" parameterType="int" resultMap="userMap">
      select * from user where id = #{id}
    </select>
<mapper/>
```

DAO层接口类UserDao.java：注意此处定义的接口方法需要和UserMapper.xml映射的<select>标签的id对应

```
public interface UserDao {
    public User getUserById(int id);
}
```

需要操作数据时调用的类UserService.java：

```
public class UserService {
    //此处省略sqlSession的获取方法
    private SqlSession sqlSession;
    private UserDao userDao;
    public User getUserById(int id) {
        return userDao.getUserById(id);
    }
}
```

二

- SqlSessionFactoryBean来替代SqlSessionFactoryBuilder来创建SqlSession
- 采用数据映射器（MapperFactoryBean）的方式
- 不用写mybatis映射文件
- 采用注解方式提供相应的sql语句和输入参数。

Spring的xml配置：

```
    <!-- 引入jdbc配置文件 -->
    <context:property-placeholder location="jdbc.properties"/>
```

```
    <!--创建jdbc数据源 -->
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-
method="close">
        <property name="driverClassName" value="${driver}"/>
        <property name="url" value="${url}"/>
        <property name="username" value="${username}"/>
        <property name="password" value="${password}"/>
        <property name="initialSize" value="${initialSize}"/>
        <property name="maxActive" value="${maxActive}"/>
        <property name="maxIdle" value="${maxIdle}"/>
        <property name="minIdle" value="${minIdle}"/>
    </bean>

    <!-- 创建SqlSessionFactory，同时指定数据源-->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
    </bean>

    <!--创建数据映射器，数据映射器必须为接口-->
    <bean id="userMapper" class="org.mybatis.spring.mapper.MapperFactoryBean">
        <property name="mapperInterface" value="com.xxt.ibatis.dbcp.dao.UserMapper" />
        <property name="sqlSessionFactory" ref="sqlSessionFactory" />
    </bean>

    <bean id="userDaoImpl" class="com.xxt.ibatis.dbcp.dao.impl.UserDaoImpl">
        <property name="userMapper" ref="userMapper"/>
    </bean>
```

数据映射器UserMapper.java：

```
public interface UserMapper {
        @Select("SELECT * FROM user WHERE id = #{userId}")
        User getUser(@Param("userId") long id);
}
```

DAO接口类UserDao.java：

```
public interface UserDao {
       public User getUserById(User user);
}
```

DAO接口实现类UserDaoImpl.java：

```
public class UserDaoImpl implements UserDao {
       private UserMapper userMapper;

       public void setUserMapper(UserMapper userMapper) {
           this.userMapper = userMapper;
       }

       public User getUserById(User user) {
          return userMapper.getUser(user.getId());
       }
}
```

三

- SqlSessionFactoryBean来替代SqlSessionFactoryBuilder创建SqlSession
- 不采用采用数据映射器（MapperFactoryBean）的方式，改为MapperScannerConfigurer 进行扫描
- 不用写mybatis映射文件
- 采用注解方式提供相应的sql语句和输入参数。
- 采用注解方式省去定义mapper的Bean

MapperFactoryBean 创建的代理类实现了 UserMapper 接口,并且注入到应用程序中。 因为代理创建在运行时环境中(Runtime,译者注) ,那么指定的映射器必须是一个接口,而 不是一个具体的实现类。

上面的MapperFactoryBean配置有一个很大的缺点，就是系统有很多的配置文件时 全部需要手动编写，所以上述的方式已经不用了。

没有必要在 Spring 的 XML 配置文件中注册所有的映射器。相反,你可以使用一个 MapperScannerConfigurer，它将会查找类路径下的映射器并自动将它们创建成MapperFactoryBean。

Spring的xml配置：

```
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
  <property name="basePackage" value="org.mybatis.spring.sample.mapper" />
</bean>
```

basePackage 属性是让你为映射器接口文件设置基本的包路径。 你可以使用分号或逗号 作为分隔符设置多于一个的包路径。每个映射器将会在指定的包路径中递归地被搜索到。

注 意，没有必要去指定SqlSessionFactory 或 SqlSessionTemplate，因为 MapperScannerConfigurer 将会创建MapperFactoryBean，之后自动装配。但是,如果你使 用了一个 以上的 DataSource，那 么自动装配可能会失效 。这种情况下， 你可以使用 sqlSessionFactoryBeanName 或 sqlSessionTemplateBeanName 属性来设置正确的 bean 名称来使用。

啦啦啦

啦啦啦

啦啦啦

分类：Spring

limeOracle
关注 - 1
粉丝 - 13
+加关注

1 0

« 上一篇：ServletContextListener使用详解
» 下一篇：8 -- 深入使用Spring -- 4...4 Spring 的 AOP 支持

posted @ 2017-02-21 18:23 limeOracle 阅读(20217) 评论(2) 编辑 收藏

评论

#1楼 2017-10-16 19:59 | 小妖，快跑我断后

感觉博主写博客的时候挺开心的，心态很好啊

支持(0) 反对(0)

#2楼 2017-11-08 15:44 | xuzhen97

文章讲的比较好，感谢楼主分享！

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。