Atitit spring5 集成 mybatis 注解班

目录

[1.1. 0. 创建配置文件——application.properties 1](#_Toc18317)

[1.2. 创建一个数据层接口——这是一个Mapper类 1](#_Toc24611)

[1.3. MyBatisConfiguartion 类 配置扫描mappers 2](#_Toc10511)

[2. Spring调用 4](#_Toc1328)

[3. Spring config 5](#_Toc16212)

码文件内容

Spring MVC之最简Mybatis配置(全注解) - 幻世 - CSDN博客.html

## 0. 创建配置文件——application.properties

写入一下内容：

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/test?useSSL=false&useUnicode=true&characterEncoding=UTF-8

spring.datasource.username=root

spring.datasource.password=123456

---------------------

## 创建一个数据层接口——这是一个Mapper类

package springMybatis;

import java.util.List;

import java.util.Map;

import org.apache.ibatis.annotations.Delete;

import org.apache.ibatis.annotations.Insert;

import org.apache.ibatis.annotations.Mapper;

import org.apache.ibatis.annotations.Param;

import org.apache.ibatis.annotations.Select;

import org.apache.ibatis.annotations.Update;

import org.mybatis.spring.SqlSessionTemplate;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.jdbc.core.JdbcTemplate;

@Mapper

//iocSpring5demo.MybatisMapperCls

public interface MybatisMapperCls {

//@Autowired

//public SqlSessionTemplate sqlSessionTemplate1;

/\*

\* 这是基于注解的映射方式，实现对数据的增删改查，将sql语句直接写在注解的括号中

\* 这是一个接口，其不需要类去实现它

\* 下边分别是插入，删除，修改，查询一个记录，查询所有的记录

\* \*/

@Select("${sql\_intag}")

public List<Map> query(@Param("sql\_intag") String sql);

}

## MyBatisConfiguartion 类 配置扫描mappers

package springMybatis;

import javax.sql.DataSource;

import org.apache.ibatis.session.SqlSessionFactory;

import org.mybatis.spring.SqlSessionFactoryBean;

import org.mybatis.spring.SqlSessionTemplate;

import org.mybatis.spring.annotation.MapperScan;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.core.io.support.PathMatchingResourcePatternResolver;

import org.springframework.core.io.support.ResourcePatternResolver;

import org.springframework.jdbc.datasource.DataSourceTransactionManager;

/\*\*

\* @author tony ittimeline@163.com

\* @date 2018-02-24-上午11:13

\* @website wwww.ittimeline.net

\* @see

\* @since JDK8u162

\*/

@Configuration

//@//Import(DruidDataSourceConfiguraiton.class)

@MapperScan(basePackages ="springMybatis")

public class MyBatisConfiguartion {

@Bean

public DataSource dataSource() {

AtiDateSource instance = new AtiDateSource();

return instance;

// configure and return the necessary JDBC DataSource

}

// @Bean

// public DataSource dataSource() {

// BasicDataSource dataSource = new BasicDataSource();

//

// dataSource.setDriverClassName("com.mysql.jdbc.Driver");

// dataSource.setUrl("jdbc:mysql://localhost:3306/mb?serverTimezone=UTC&useUnicode=true&characterEncoding=utf-8");

// dataSource.setUsername(""); // 这里填写数据库用户名

// dataSource.setPassword(""); // 这里填写数据库密码

//

// return dataSource;

// }

//

@Bean

public DataSourceTransactionManager transactionManager() {

return new DataSourceTransactionManager(dataSource());

}

@Bean

public SqlSessionFactoryBean sqlSessionFactoryBean() throws Exception {

SqlSessionFactoryBean sessionFactory = new SqlSessionFactoryBean();

sessionFactory.setDataSource(dataSource());

// sessionFactory.setTypeAliasesPackage("me.firstsnow.model");

return sessionFactory;

}

/\*

@Bean(name = "sqlSessionFactory")

public SqlSessionFactory sqlSessionFactoryBean(DataSource dataSource){

\*/

/\*\*

\*

\* @param sqlSessionFactory

\* @return

\*/

@Bean

public SqlSessionTemplate sqlSessionTemplate(SqlSessionFactory sqlSessionFactory){

return new SqlSessionTemplate(sqlSessionFactory);

}

}

# Spring调用

package springMybatis;

import java.util.List;

import java.util.Map;

import org.mybatis.spring.annotation.MapperScan;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import com.alibaba.fastjson.JSON;

@MapperScan(basePackages= {"iocSpring5demo"})

public class SpringMybayisStartDemoAnno {

public static void main(String[] args) {

// 创建spring 基于注解配置的容器

AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

// 获取通过注解注入容器的UserService

MybatisMapperCls MybatisMapperCls1 = context.getBean(MybatisMapperCls.class);

// 调用userService的方法执行

List<Map> message = MybatisMapperCls1.query("select \* from user\_tab");

// userService.addUser("user2");

// 输出结果

System.out.println( JSON.toJSONString(message));

// 关闭容器,释放JVM资源

context.close();

}

}

# Spring config

package springMybatis;

import javax.sql.DataSource;

import org.mybatis.spring.annotation.MapperScan;

import org.springframework.beans.factory.config.PropertyPlaceholderConfigurer;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.EnableAspectJAutoProxy;

import org.springframework.core.io.ClassPathResource;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.datasource.DataSourceTransactionManager;

import org.springframework.transaction.PlatformTransactionManager;

import org.springframework.transaction.annotation.EnableTransactionManagement;

import org.springframework.transaction.annotation.TransactionManagementConfigurer;

import org.springframework.transaction.support.TransactionTemplate;

@EnableTransactionManagement

@EnableAspectJAutoProxy

@Configuration /\*\* 该注解表示这个类是一个Spring的配置类 \*\*/

@ComponentScan(basePackages = {

"springMybatis" }) /\*\*\*

\* 该注解表示启用spring的组件扫描功能，并且配置了扫描包net.xqlee.project.

\* demo下的所有类

\*\*/

//@//MapperScan(basePackages= {"springMybatis"})

public class AppConfig implements TransactionManagementConfigurer {

@Bean

public DataSource dataSource() {

AtiDateSource instance = new AtiDateSource();

return instance;

// configure and return the necessary JDBC DataSource

}

@Bean

@Override

public PlatformTransactionManager annotationDrivenTransactionManager() {

// return txManager();

return new DataSourceTransactionManager(dataSource());

}

}