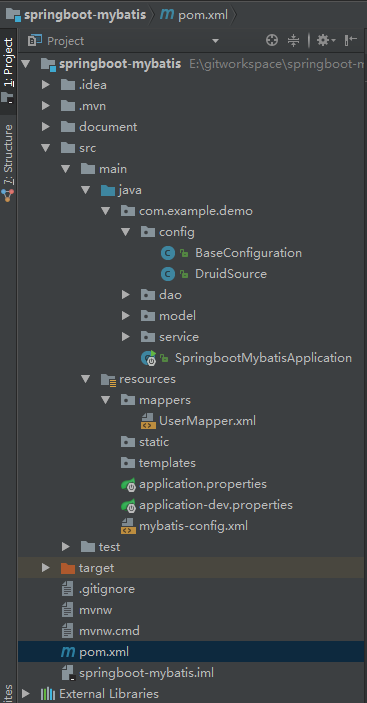
# springboot-mybatis说明

## 项目结构：



## Pom.xml引入的jar包

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.example</groupId>  
 <artifactId>demo</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>springboot-mybatis</name>  
 <description>Demo project for Spring Boot</description>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>1.5.10.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
 <java.version>1.8</java.version>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.mybatis.spring.boot</groupId>  
 <artifactId>mybatis-spring-boot-starter</artifactId>  
 <version>1.3.2</version>  
 </dependency>  
  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid</artifactId>  
 <version>1.0.5</version>  
 </dependency>  
 <dependency>  
 <groupId>tk.mybatis</groupId>  
 <artifactId>mapper-spring-boot-starter</artifactId>  
 <version>1.1.4</version>  
 </dependency>  
  
 <dependency>  
 <groupId>javax.persistence</groupId>  
 <artifactId>persistence-api</artifactId>  
 <version>1.0</version>  
 </dependency>  
  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
  
</project>

## 配置文件application.properties

server.context-path=/test1  
server.port=8080  
  
server.tomcat.uri-encoding=UTF-8  
spring.http.encoding.charset=UTF-8  
spring.http.encoding.enabled=true  
spring.http.encoding.force=true  
spring.messages.encoding=UTF-8  
  
# ===========================================================  
spring.application.name=springboot-mybatis  
spring.application.index=0  
  
spring.devtools.restart.exclude=static/\*\*  
spring.devtools.restart.additional-exclude=static/\*\*,public/\*\*  
  
spring.session.store-type=none  
  
spring.profiles.active=dev

# ============================================================  
spring.datasource.type=com.alibaba.druid.pool.DruidDataSource  
spring.datasource.driver-class-name=com.mysql.jdbc.Driver  
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull&rewriteBatchedStatements=false&autoReconnect=true   
spring.datasource.username: root  
spring.datasource.password: 111111  
  
#连接池的配置信息  
## 初始化大小，最小，最大  
spring.druid.initialSize=5  
spring.druid.minIdle=5  
spring.druid.maxActive=20  
## 配置获取连接等待超时的时间  
spring.druid.maxWait=60000  
# 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒  
spring.druid.timeBetweenEvictionRunsMillis=60000  
# 配置一个连接在池中最小生存的时间，单位是毫秒  
spring.druid.minEvictableIdleTimeMillis=300000  
spring.druid.validationQuery=SELECT 1 FROM DUAL  
spring.druid.testWhileIdle=true  
spring.druid.testOnBorrow=false  
spring.druid.testOnReturn=false  
spring.druid.poolPreparedStatements=true  
spring.druid.maxPoolPreparedStatementPerConnectionSize=20  
# 配置监控统计拦截的filters，去掉后监控界面sql无法统计，'wall'用于防火墙  
spring.druid.filters=stat,wall,log4j  
# 通过connectProperties属性来打开mergeSql功能；慢SQL记录  
spring.druid.connectionProperties=druid.stat.mergeSql=true;druid.stat.slowSqlMillis=5000  
  
mybatis.config-location=classpath:mybatis-config.xml   
mybatis.mapper-locations=classpath\*:mappers/\*Mapper.xml  
mybatis.type-aliases-package=com.example.demo.model

# ============================================ 日志 =========  
logging.pattern.console=%d{yyyy/MM/dd-HH:mm:ss} [%thread] %-5level %logger- %msg%n   
logging.pattern.file=%d{yyyy/MM/dd-HH:mm} [%thread] %-5level %logger- %msg%n

logging.level.com.example.demo.dao=DEBUG  
logging.file=E:/testspace/log.log

## 数据库配置mybatis-config.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>  
  
 <properties>  
 <property name="dialect" value="mysql"/>  
 </properties>  
  
 <settings>  
 <!-- 全局映射器启用缓存 -->  
 <setting name="cacheEnabled" value="true" />  
 <!-- 查询时,关闭关联对象及时加载以提高性能 -->  
 <setting name="lazyLoadingEnabled" value="false" />  
 <!-- 设置关联对象加载的形态,此处为按需加载字段(加载字段由SQL指定),不会加载关联表的所有字段,以提高性能 -->  
 <setting name="aggressiveLazyLoading" value="false" />  
 <!-- 对于位置的SQL查询,允许返回不同的结果集以达到通用的效果 -->  
 <setting name="multipleResultSetsEnabled" value="true" />  
 <!-- 允许使用列标签代替列明 -->  
 <setting name="useColumnLabel" value="true" />  
 <!-- 允许使用自定义的主键值(比如由程序生成的UUID 32位编码作为键值), 数据表的pk生成策略将被覆盖 -->  
 <setting name="useGeneratedKeys" value="true" />  
 <!-- 给予被嵌套的resultMap以字段-属性的映射支持 -->  
 <setting name="autoMappingBehavior" value="PARTIAL" />  
 <!-- 对于批量更新操作缓存SQL以提高性能 -->  
 <setting name="defaultExecutorType" value="REUSE" />  
 <!-- 数据库超过25000秒仍未响应则超时 -->  
 <setting name="defaultStatementTimeout" value="25000" />  
 <setting name="safeRowBoundsEnabled" value="false" />  
 <setting name="mapUnderscoreToCamelCase" value="false" />  
 <setting name="localCacheScope" value="SESSION" />  
 <setting name="jdbcTypeForNull" value="OTHER" />  
 <setting name="lazyLoadTriggerMethods" value="equals,clone,hashCode ,toString" />  
 <!-- 打印查询语句 -->  
 <!--<setting name="logImpl" value="STDOUT\_LOGGING" />-->  
 </settings>  
  
 <typeHandlers>  
 <typeHandler handler="org.apache.ibatis.type.InstantTypeHandler" />  
 <typeHandler handler="org.apache.ibatis.type.LocalDateTimeTypeHandler" />  
 <typeHandler handler="org.apache.ibatis.type.LocalDateTypeHandler" />  
 <typeHandler handler="org.apache.ibatis.type.LocalTimeTypeHandler" />  
 <typeHandler handler="org.apache.ibatis.type.OffsetDateTimeTypeHandler" />  
 <typeHandler handler="org.apache.ibatis.type.OffsetTimeTypeHandler" />  
 <typeHandler handler="org.apache.ibatis.type.ZonedDateTimeTypeHandler" />  
 </typeHandlers>  
  
  
</configuration>

## Config：

### 数据源：

package com.example.demo.config;  
  
import com.alibaba.druid.pool.DruidDataSource;  
import org.springframework.boot.context.properties.ConfigurationProperties;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Primary;  
import org.springframework.jdbc.datasource.DataSourceTransactionManager;  
  
import javax.sql.DataSource;  
import java.sql.SQLException;  
  
*/\*\*  
 \* Created by liulanhua on 2018/3/9.  
 \*/*@ConfigurationProperties(prefix = "spring.druid")  
public class DruidSource {  
  
 private String dbUrl;  
  
 private String username;  
  
 private String password;  
  
 private String driverClassName;  
  
 private int initialSize;  
  
 private int minIdle;  
  
 private int maxActive;  
  
 private int maxWait;  
  
 private int timeBetweenEvictionRunsMillis;  
  
 private int minEvictableIdleTimeMillis;  
  
 private String validationQuery;  
  
 private boolean testWhileIdle;  
  
 private boolean testOnBorrow;  
  
 private boolean testOnReturn;  
  
 private boolean poolPreparedStatements;  
  
 private int maxPoolPreparedStatementPerConnectionSize;  
  
 private String filters;  
  
 private String connectionProperties;  
  
 public String getDbUrl() {  
 return dbUrl;  
 }  
  
 public void setDbUrl(String dbUrl) {  
 this.dbUrl = dbUrl;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public String getDriverClassName() {  
 return driverClassName;  
 }  
  
 public void setDriverClassName(String driverClassName) {  
 this.driverClassName = driverClassName;  
 }  
  
 public int getInitialSize() {  
 return initialSize;  
 }  
  
 public void setInitialSize(int initialSize) {  
 this.initialSize = initialSize;  
 }  
  
 public int getMinIdle() {  
 return minIdle;  
 }  
  
 public void setMinIdle(int minIdle) {  
 this.minIdle = minIdle;  
 }  
  
 public int getMaxActive() {  
 return maxActive;  
 }  
  
 public void setMaxActive(int maxActive) {  
 this.maxActive = maxActive;  
 }  
  
 public int getMaxWait() {  
 return maxWait;  
 }  
  
 public void setMaxWait(int maxWait) {  
 this.maxWait = maxWait;  
 }  
  
 public int getTimeBetweenEvictionRunsMillis() {  
 return timeBetweenEvictionRunsMillis;  
 }  
  
 public void setTimeBetweenEvictionRunsMillis(int timeBetweenEvictionRunsMillis) {  
 this.timeBetweenEvictionRunsMillis = timeBetweenEvictionRunsMillis;  
 }  
  
 public int getMinEvictableIdleTimeMillis() {  
 return minEvictableIdleTimeMillis;  
 }  
  
 public void setMinEvictableIdleTimeMillis(int minEvictableIdleTimeMillis) {  
 this.minEvictableIdleTimeMillis = minEvictableIdleTimeMillis;  
 }  
  
 public String getValidationQuery() {  
 return validationQuery;  
 }  
  
 public void setValidationQuery(String validationQuery) {  
 this.validationQuery = validationQuery;  
 }  
  
 public boolean isTestWhileIdle() {  
 return testWhileIdle;  
 }  
  
 public void setTestWhileIdle(boolean testWhileIdle) {  
 this.testWhileIdle = testWhileIdle;  
 }  
  
 public boolean isTestOnBorrow() {  
 return testOnBorrow;  
 }  
  
 public void setTestOnBorrow(boolean testOnBorrow) {  
 this.testOnBorrow = testOnBorrow;  
 }  
  
 public boolean isTestOnReturn() {  
 return testOnReturn;  
 }  
  
 public void setTestOnReturn(boolean testOnReturn) {  
 this.testOnReturn = testOnReturn;  
 }  
  
 public boolean isPoolPreparedStatements() {  
 return poolPreparedStatements;  
 }  
  
 public void setPoolPreparedStatements(boolean poolPreparedStatements) {  
 this.poolPreparedStatements = poolPreparedStatements;  
 }  
  
 public int getMaxPoolPreparedStatementPerConnectionSize() {  
 return maxPoolPreparedStatementPerConnectionSize;  
 }  
  
 public void setMaxPoolPreparedStatementPerConnectionSize(int maxPoolPreparedStatementPerConnectionSize) {  
 this.maxPoolPreparedStatementPerConnectionSize = maxPoolPreparedStatementPerConnectionSize;  
 }  
  
 public String getFilters() {  
 return filters;  
 }  
  
 public void setFilters(String filters) {  
 this.filters = filters;  
 }  
  
 public String getConnectionProperties() {  
 return connectionProperties;  
 }  
  
 public void setConnectionProperties(String connectionProperties) {  
 this.connectionProperties = connectionProperties;  
 }  
  
 @Bean //声明其为Bean实例  
 @Primary //在同样的DataSource中，首先使用被标注的DataSource  
 public DataSource dataSource() {  
 DruidDataSource datasource = new DruidDataSource();  
 datasource.setUrl(this.dbUrl);  
 datasource.setUsername(username);  
 datasource.setPassword(password);  
 datasource.setDriverClassName(driverClassName);  
  
 //configuration  
 datasource.setInitialSize(initialSize);  
 datasource.setMinIdle(minIdle);  
 datasource.setMaxActive(maxActive);  
 datasource.setMaxWait(maxWait);  
 datasource.setTimeBetweenEvictionRunsMillis(timeBetweenEvictionRunsMillis);  
 datasource.setMinEvictableIdleTimeMillis(minEvictableIdleTimeMillis);  
 datasource.setValidationQuery(validationQuery);  
 datasource.setTestWhileIdle(testWhileIdle);  
 datasource.setTestOnBorrow(testOnBorrow);  
 datasource.setTestOnReturn(testOnReturn);  
 datasource.setPoolPreparedStatements(poolPreparedStatements);  
 datasource.setMaxPoolPreparedStatementPerConnectionSize(maxPoolPreparedStatementPerConnectionSize);  
  
 try {  
 datasource.setFilters(filters);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return datasource;  
 }  
  
 @Bean  
 @Primary  
 //配置事物管理  
 public DataSourceTransactionManager masterTransactionManager() throws SQLException {  
 return new DataSourceTransactionManager(dataSource());  
 }  
  
  
  
}

### BaseConfiguration

package com.example.demo.config;  
  
import org.mybatis.spring.annotation.MapperScan;  
import org.springframework.context.annotation.ComponentScan;  
  
*/\*\*  
 \* Created by liulanhua on 2018/3/19.  
 \*/*@ComponentScan({"com.example.demo.service","com.example.demo.controller"})  
@MapperScan(basePackages = "com.example.demo.dao")//mapper接口的路径  
public class BaseConfiguration {  
  
  
}

## 启动入口

package com.example.demo;  
  
import com.example.demo.config.BaseConfiguration;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.annotation.Import;  
  
@SpringBootApplication  
@Import({BaseConfiguration.class})  
public class SpringbootMybatisApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SpringbootMybatisApplication.class, args);  
 }  
}

## 其他

Dao, service,model等照普通项目写

## 单元测试结果展示

