**Springboot+Mybatis+Druid 实现多数据源，与事务管理**

[Springboot](https://my.oschina.net/swiftloop?q=Springboot)[多数据源](https://my.oschina.net/swiftloop?q=%E5%A4%9A%E6%95%B0%E6%8D%AE%E6%BA%90)[Druid](https://my.oschina.net/swiftloop?q=Druid)[Mybatis](https://my.oschina.net/swiftloop?q=Mybatis)

1.需求

    一个主库的数据已经不能满足我们生产需求，需要额外的辅助库集成到项目里来获取所需的数据。或是将读写分开为两个库，完成自有的逻辑。

2.实现原理

    如同在Springboot中配置一个数据源一样，我们需要一个主数据源，然后再用同样的方法配置另外一个数据源作为辅，两个数据源有各自独立的DataSourceTransactionManager与SqlSessionFactroy。

3.具体的实现

    1) pom.xml

<?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.datasources</groupId>

<artifactId>data</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>data</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.5.4.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.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mybatis.spring.boot</groupId>

<artifactId>mybatis-spring-boot-starter</artifactId>

<version>1.2.0</version>

</dependency>

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid</artifactId>

<version>1.0.18</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

    2)application.properties

# 主数据源，默认的

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

spring.datasource.url=jdbc:mysql://localhost:3306/test

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.filters=stat,wall,log4j

# 更多数据源

custom.datasource.names=ds1,ds2

custom.datasource.ds1.driver-class-name=com.mysql.jdbc.Driver

custom.datasource.ds1.url=jdbc:mysql://localhost:3306/test1

custom.datasource.ds1.username=root

custom.datasource.ds1.password=root

custom.datasource.ds1.filters=stat,wall,log4j

3)数据库

数据库test ，表test

CREATE TABLE `test` (

`id` int(11) unsigned zerofill NOT NULL AUTO\_INCREMENT,

`name` varchar(36) NOT NULL,

`age` int(2) NOT NULL,

`sex` varchar(2) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=9 DEFAULT CHARSET=utf8;

数据库test1，表test

CREATE TABLE `test` (

`id` int(11) unsigned zerofill NOT NULL AUTO\_INCREMENT,

`name` varchar(36) NOT NULL,

`age` int(2) NOT NULL,

`sex` varchar(2) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=9 DEFAULT CHARSET=utf8;

两个库表没什么区别，只是单纯的分两个库用户测试。

4）配置数据源Master

package com.sorata.custom.datasources;

import com.alibaba.druid.pool.DruidDataSource;

import org.apache.ibatis.session.SqlSessionFactory;

import org.mybatis.spring.SqlSessionFactoryBean;

import org.mybatis.spring.annotation.MapperScan;

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

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Primary;

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

import org.springframework.jdbc.datasource.DataSourceTransactionManager;

import javax.sql.DataSource;

/\*\*

\* Created by : Sorata 2017/6/26 0026 上午 11:02.

\*/

@Configuration

@MapperScan(basePackages = MasterDataSourcesConfig.PACKAGES, sqlSessionFactoryRef = "masterSqlSessionFactory")

public class MasterDataSourcesConfig {

static final String PACKAGES = "com.sorata.dao.master";

private static final String MAPPER\_LOCAL = "classpath:mapper/master/\*.xml";

@ConfigurationProperties("spring.datasource")

@Primary

@Bean(name = "masterDataSource")

public DruidDataSource druidDataSource() {

return new DruidDataSource();

}

@Bean(name = "masterTransactionManager")

@Primary

public DataSourceTransactionManager masterTransactionManager() {

return new DataSourceTransactionManager(druidDataSource());

}

@Bean(name = "masterSqlSessionFactory")

@Primary

public SqlSessionFactory masterSqlSessionFactory(@Qualifier("masterDataSource") DataSource dataSource) throws Exception {

final SqlSessionFactoryBean sessionFactoryBean = new SqlSessionFactoryBean();

sessionFactoryBean.setDataSource(dataSource);

sessionFactoryBean.setMapperLocations(new PathMatchingResourcePatternResolver().getResources(MAPPER\_LOCAL));

return sessionFactoryBean.getObject();

}

}

说明：@Primary标记着主库，作用就是在同类型存在多个bean的时候，选择标记的bean使用。PACKAGES描述的是被扫描的dao的包，扫描mybatis的dao接口。MAPPER\_LOCAL描述的是配置的mybatis的XML文件的路径。

5）配置custom数据源

package com.sorata.custom.datasources;

import com.alibaba.druid.pool.DruidDataSource;

import org.apache.ibatis.session.SqlSessionFactory;

import org.mybatis.spring.SqlSessionFactoryBean;

import org.mybatis.spring.annotation.MapperScan;

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

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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

import org.springframework.jdbc.datasource.DataSourceTransactionManager;

/\*\*

\* Created by : Sorata 2017/6/26 0026 上午 11:02.

\*/

@Configuration

@MapperScan(basePackages = CustomDataSourcesConfig.PACKAGES, sqlSessionFactoryRef = "customSqlSessionFactory")

public class CustomDataSourcesConfig {

static final String PACKAGES = "com.sorata.dao.custom";

private static final String MAPPER\_LOCAL = "classpath:mapper/custom/\*.xml";

@ConfigurationProperties("custom.datasource.ds1")

@Bean(name = "customDataSource")

public DruidDataSource druidDataSource() {

return new DruidDataSource();

}

@Bean(name = "customTransactionManager")

public DataSourceTransactionManager customTransactionManager() {

return new DataSourceTransactionManager(druidDataSource());

}

@Bean(name = "customSqlSessionFactory")

public SqlSessionFactory customSqlSessionFactory(@Qualifier("customDataSource") DruidDataSource dataSource) throws Exception {

final SqlSessionFactoryBean sessionFactoryBean = new SqlSessionFactoryBean();

sessionFactoryBean.setDataSource(dataSource);

sessionFactoryBean.setMapperLocations(new PathMatchingResourcePatternResolver().getResources(MAPPER\_LOCAL));

return sessionFactoryBean.getObject();

}

}

与主数据源并无太大差异，只是没有使用@Primary。

6）配置阿里巴巴Druid的监控

package com.sorata.custom.datasources;

import com.alibaba.druid.support.http.StatViewServlet;

import javax.servlet.annotation.WebInitParam;

import javax.servlet.annotation.WebServlet;

/\*\*

\* Created by: guobo 17/1/19 下午5:20

\*

\* alibaba druid 页面监控配置 访问url http://localhost:8080/项目名/druid/index.html

\*

\* 在使用springboot的时候,需要在启动类使用 @ServletComponentScan注解

\*/

@SuppressWarnings("serial")

@WebServlet(urlPatterns = "/druid/\*",initParams = {

//@WebInitParam(name = "allow",value = "127.0.0.1,192.168.1.142"),//白名单

@WebInitParam(name = "deny",value = "126.12.22.1"),//黑名单 (存在共同时，deny优先于allow)

@WebInitParam(name="loginUsername",value="admin"),// 用户名

@WebInitParam(name="loginPassword",value="123456"),// 密码

@WebInitParam(name="resetEnable",value="false")// 禁用HTML页面上的“Reset All”功能

})

public class DruidStateViewServlet extends StatViewServlet {

}

package com.sorata.custom.datasources;

import com.alibaba.druid.support.http.WebStatFilter;

import javax.servlet.annotation.WebFilter;

import javax.servlet.annotation.WebInitParam;

/\*\*

\* Created by: guobo 17/1/19 下午5:27

\*

\* alibaba druid的过滤

\*/

@WebFilter(filterName="druidWebStatFilter",urlPatterns="/\*",

initParams={

@WebInitParam(name="exclusions",value="\*.mp4,\*.js,\*.gif,\*.jpg,\*.bmp,\*.png,\*.css,\*.ico,/druid/\*")// 忽略资源

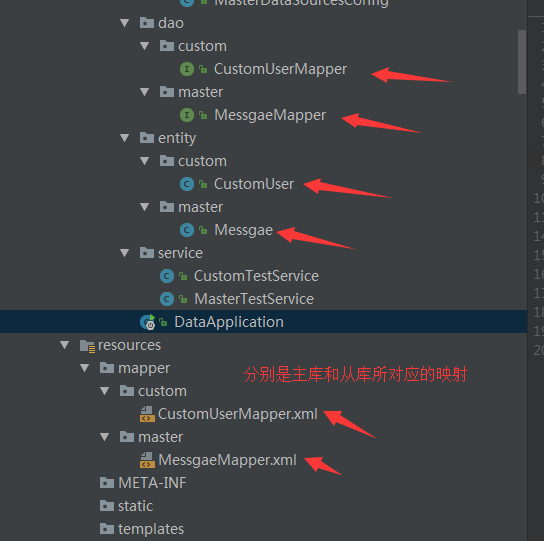
})

public class DruidStatFilter extends WebStatFilter {

}

配置完成了，一定要在启动类上加上@ServletComponentScan,同时需要注意的是，启动类最好放在所有包的最下面，其中缘由请自行百度。

7）接下来，我们使用Mybatis的Generator工具，自动生成实体类，dao，与XML。创建的时候，对象所对应的包一定不能错，test库中test表所对应的包是在master数据源中配置的PACKAGE包路径，而test1则是在custom中配置的。完成之后，就可以编写测试类。



8)编写service

1.MasterTestService

package com.sorata.service;

import com.sorata.dao.master.MessgaeMapper;

import com.sorata.entity.master.Messgae;

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

import org.springframework.stereotype.Service;

/\*\*

\* Created by : Sorata 2017/6/26 0026 下午 1:59.

\*/

@Service

public class MasterTestService {

@Autowired

private MessgaeMapper messgaeMapper;

public Messgae getM(Integer id){

return messgaeMapper.selectByPrimaryKey(id);

}

public int instet(){

Messgae messgae = new Messgae();

messgae.setAge(35);

messgae.setName("王五");

messgae.setSex("男");

return messgaeMapper.insertSelective(messgae);

}

}

2.CustomTestService

package com.sorata.service;

import com.sorata.dao.custom.CustomUserMapper;

import com.sorata.entity.custom.CustomUser;

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

import org.springframework.stereotype.Service;

/\*\*

\* Created by : Sorata 2017/6/26 0026 下午 1:59.

\*/

@Service

public class CustomTestService {

@Autowired

private CustomUserMapper customUserMapper;

public CustomUser getM(Integer id){

return customUserMapper.selectByPrimaryKey(id);

}

public int instet(){

CustomUser customUser = new CustomUser();

customUser.setAge(36);

customUser.setName("赵六");

customUser.setSex("男");

return customUserMapper.insertSelective(customUser);

}

}

9）编写Controller

package com.sorata.api;

import com.sorata.service.CustomTestService;

import com.sorata.service.MasterTestService;

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

import org.springframework.transaction.annotation.Transactional;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

/\*\*

\* Created by : Sorata 2017/6/26 0026 下午 2:00.

\*/

@RestController

public class TestApi {

@Autowired

private MasterTestService masterTestService;

@Autowired

private CustomTestService customTestService;

/\*\*

\* 测试主库查询

\*

\* @param id

\* @return

\*/

@RequestMapping("/master/select")

public Object getaster(@RequestParam Integer id) {

return masterTestService.getM(id);

}

/\*\*

\* 测试主库插入

\*

\* @return

\*/

@RequestMapping("/master/insert")

public Object insertMaster() {

return masterTestService.instet();

}

/\*\*

\* 测试主库事务

\*

\* @return

\*/

@RequestMapping("/master/ex")

@Transactional

public Object exMaster() {

int m = masterTestService.instet();

int s = 1000 / 0;

return m;

}

/\*\*

\* 测试从库查询

\*

\* @param id

\* @return

\*/

@RequestMapping("/custom/select")

public Object getcustom(@RequestParam Integer id) {

return customTestService.getM(id);

}

/\*\*

\* 测试从库插入

\*

\* @return

\*/

@RequestMapping("/custom/insert")

public Object insertCustom() {

return customTestService.instet();

}

/\*\*

\* 测试从库事务 从库这里需要注意的是，在@Transactional注解上需要显示的声明是哪个事务管理

\*

\* @return

\*/

@RequestMapping("/custom/ex")

@Transactional("customTransactionManager")

public Object exCustom() {

int m = customTestService.instet();

int s = 1000 / 0;

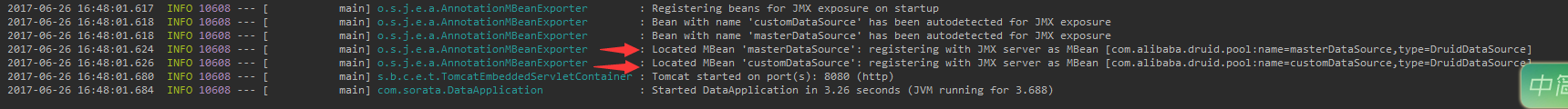
return m;

}

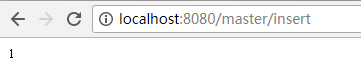
}

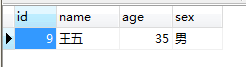
4.测试

    1）启动

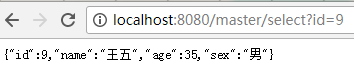


2）主库插入

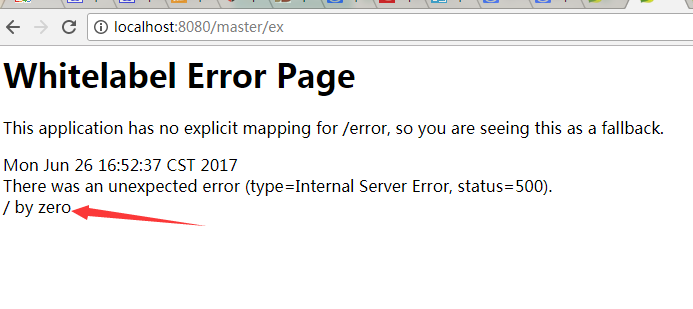




3）主库查询

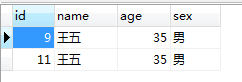


4）主库事务



https://static.oschina.net/uploads/space/2017/0626/165330_OQ5i_2661827.png

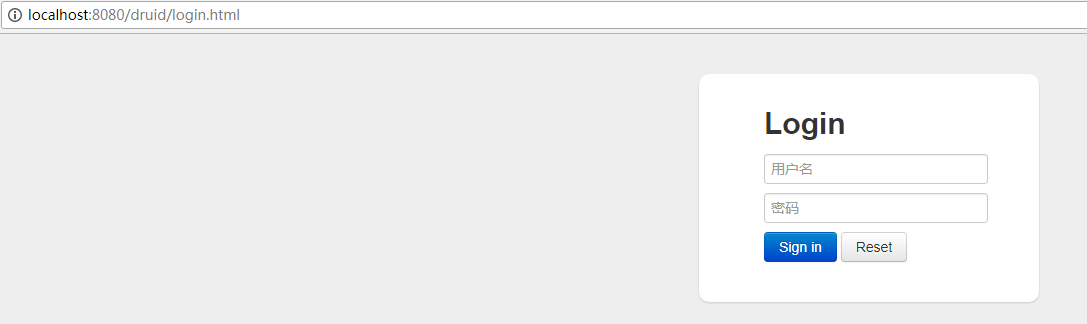
此时数据库仍是一条数据，当我们再次插入一条数据，通过自增的主键可以明显看到事务的处理



5）从库的测试就不在赘述。

5.Druid的监控

    登录网址，



输入密码登录。可以看到Druid的各种监控功能，当然，Druid还有其他的一些参数，在此没有具体的配置，可自行配置，优化。

6.总结

    配置使用中，唯一需要特别注意的就是从库的事务管理的时候需要显示的声明使用的是哪个事务管理器，否则直接使用注解将不能达到事务的功能。在与数据库插入的时候，出现中文变成？的情况，可以在Mysql的my.ini文件中配置，在[mysqld]标签下配置character-set-server=utf8

附：<http://git.oschina.net/lencer93/datasource>

可参考另一篇集成：<https://my.oschina.net/swiftloop/blog/1036943>