

Ivan_K

博客园 首页 新随笔 联系 订阅 管理

springboot2动态数据源的绑定

由于springboot2更新了绑定参数的api，部分springboot1用于绑定的工具类如RelaxedPropertyResolver已经无法在新版本中使用。本文实现参考了<https://blog.csdn.net/catoop/article/details/50575038>这篇文章，大致思路是一致的，如果需要详细实现可以参考。都是通过AbstractRoutingDataSource实现动态数据源的切换，以前我用spring配置多数据源的时候就是通过它实现的，有兴趣的可以了解下其原理，这里就不多赘述了。

废话不多说了，先上数据源注册工具类，springboot2与1的主要区别也就在这：

MultiDataSourceRegister.java：

```
package top.ivan.demo.springboot.mapper;

import com.zaxxer.hikari.HikariDataSource;
import org.springframework.beans.MutablePropertyValues;
import org.springframework.beans.factory.support.BeanDefinitionRegistry;
import org.springframework.beans.factory.support.GenericBeanDefinition;
import org.springframework.boot.context.properties.bind.Bindable;
import org.springframework.boot.context.properties.bind.Binder;
import org.springframework.boot.context.properties.source.ConfigurationPropertyName;
import org.springframework.boot.context.properties.source.ConfigurationPropertyNameAliases;
import org.springframework.boot.context.properties.source.ConfigurationPropertySource;
import org.springframework.boot.context.properties.source.MapConfigurationPropertySource;
import org.springframework.context.EnvironmentAware;
import org.springframework.context.annotation.ImportBeanDefinitionRegistrar;
import org.springframework.core.env.Environment;
import org.springframework.core.type.AnnotationMetadata;
import org.springframework.util.StringUtils;

import javax.sql.DataSource;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MultiDataSourceRegister implements EnvironmentAware,
    ImportBeanDefinitionRegistrar {

    private final static ConfigurationPropertyNameAliases aliases = new
    ConfigurationPropertyNameAliases(); //别名

    static {
        //由于部分数据源配置不同，所以在此处添加别名，避免切换数据源出现某些参数无法注入的情况
        aliases.addAliases("url", new String[]{"jdbc-url"});
        aliases.addAliases("username", new String[]{"user"});
    }

    private Environment evn; //配置上下文（也可以理解为配置文件的获取工具）

    private Map<String, DataSource> sourceMap; //数据源列表
```

公告

昵称：Ivan_K
园龄：9个月
粉丝：1
关注：0
+加关注

2019年3月						
日	一	二	三	四	五	六
24	25	26	27	28	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔档案

2018年5月 (2)

最新评论

1. Re:springboot2动态数据源的绑定
@Ivan_K版本没问题，我用的是最新的基本...
--Dorla
2. Re:springboot2动态数据源的绑定
@Dorla额，我的项目把那几个依赖去掉换成spring-boot-starter-aop之后也是正常运行的。假如不行的话只能猜测是不是版本问题了
...
--Ivan_K
3. Re:springboot2动态数据源的绑定
@Ivan_K我一开始也是以为加了自带的就不用加aspectj依赖了，但是就是进不了切面...

```
private Binder binder; //参数绑定工具

/**
 * ImportBeanDefinitionRegistrar接口的实现方法，通过该方法可以按照自己的方式注册bean
 *
 * @param annotationMetadata
 * @param beanDefinitionRegistry
 */
@Override
public void registerBeanDefinitions(AnnotationMetadata annotationMetadata,
BeanDefinitionRegistry beanDefinitionRegistry) {
    Map config, properties, defaultConfig = binder.bind("spring.datasource",
Map.class).get(); //获取所有数据源配置
    sourceMap = new HashMap<>(); //默认配置
    properties = defaultConfig;
    String typeStr = env.getProperty("spring.datasource.type"); //默认数据源类型
    Class<? extends DataSource> clazz = getDataSourceType(typeStr); //获取数据源类型
    DataSource consumerDataSource, defaultDataSource = bind(clazz, properties); //绑定默
认数据源参数

    List<Map> configs = binder.bind("spring.datasource.multi",
Bindable.listOf(Map.class)).get(); //获取其他数据源配置
    for (int i = 0; i < configs.size(); i++) { //遍历生成其他数据源
        config = configs.get(i);
        clazz = getDataSourceType((String) config.get("type"));
        if ((boolean) config.getDefault("extend", Boolean.TRUE)) { //获取extend字段，未
定义或为true则为继承状态
            properties = new HashMap(defaultConfig); //继承默认数据源配置
            properties.putAll(config); //添加数据源参数
        } else {
            properties = config; //不继承默认配置
        }
        consumerDataSource = bind(clazz, properties); //绑定参数
        sourceMap.put(config.get("key").toString(), consumerDataSource); //获取数据源的
key，以便通过该key可以定位到数据源
    }

    GenericBeanDefinition define = new GenericBeanDefinition(); //bean定义类
    define.setBeanClass(MultiDataSource.class); //设置bean的类型，此处MultiDataSource是继
承AbstractRoutingDataSource的实现类
    MutablePropertyValues mpv = define.getPropertyValues(); //需要注入的参数，类似spring配
置文件中的<property/>
    mpv.add("defaultTargetDataSource", defaultDataSource); //添加默认数据源，避免key不存在
的情况没有数据源可用
    mpv.add("targetDataSources", sourceMap); //添加其他数据源
    beanDefinitionRegistry.registerBeanDefinition("datasource", define); //将该bean注册
为datasource，不使用springboot自动生成的datasource
}

/**
 * 通过字符串获取数据源class对象
 *
 * @param typeStr
 * @return
 */
private Class<? extends DataSource> getDataSourceType(String typeStr) {
    Class<? extends DataSource> type;
    try {
        if (StringUtils.hasLength(typeStr)) { //字符串不为空则通过反射获取class对象
            type = (Class<? extends DataSource>) Class.forName(typeStr);
        } else {
            type = HikariDataSource.class; //默认为hikariCP数据源，与springboot默认数据源
```

--Dorla

4. Re:springboot2动态数据源的绑定

@Dorla可以考虑使用springboot自带的aop
注解，应该可以达到同样的效果： org.spring
framework.boot s.....

--Ivan_K

5. Re:springboot2动态数据源的绑定

@Ivan_K找到问题所在了，缺少了aspectj依
赖导致在mapper层不能进入切面...

--Dorla

阅读排行榜

1. springboot2动态数据源的绑定(5093)
2. 谈谈Java中使用DataInputStream与Dat
aOutputStream的雷区(41)

评论排行榜

1. springboot2动态数据源的绑定(12)

保持一致

```

    }

    return type;
} catch (Exception e) {
    throw new IllegalArgumentException("can not resolve class with type: " +
typeStr); //无法通过反射获取class对象的情况则抛出异常, 该情况一般是写错了, 所以此次抛出一个
runtimeexception
}
}

/**
 * 绑定参数, 以下三个方法都是参考DataSourceBuilder的bind方法实现的, 目的是尽量保证我们自己添加的数
据源构造过程与springboot保持一致
 *
 * @param result
 * @param properties
 */
private void bind(DataSource result, Map properties) {
    ConfigurationPropertySource source = new
MapConfigurationPropertySource(properties);
    Binder binder = new Binder(new ConfigurationPropertySource[]
{source.withAliases(aliases)});
    binder.bind(ConfigurationPropertyName.EMPTY, Bindable.ofInstance(result)); //将参数
绑定到对象
}

private <T extends DataSource> T bind(Class<T> clazz, Map properties) {
    ConfigurationPropertySource source = new
MapConfigurationPropertySource(properties);
    Binder binder = new Binder(new ConfigurationPropertySource[]
{source.withAliases(aliases)});
    return binder.bind(ConfigurationPropertyName.EMPTY, Bindable.of(clazz)).get(); //通
过类型绑定参数并获得实例对象
}

/**
 * @param clazz
 * @param sourcePath 参数路径, 对应配置文件中的值, 如: spring.datasource
 * @param <T>
 * @return
 */
private <T extends DataSource> T bind(Class<T> clazz, String sourcePath) {
    Map properties = binder.bind(sourcePath, Map.class).get();
    return bind(clazz, properties);
}

/**
 * EnvironmentAware接口的实现方法, 通过aware的方式注入, 此处是environment对象
 *
 * @param environment
 */
@Override
public void setEnvironment(Environment environment) {
    this.envn = environment;
    binder = Binder.get(envn); //绑定配置器
}
}

```



此处放出我的配置文件application.yml :



```
spring:
  datasource:
    password: 123456
    url: jdbc:mysql://127.0.0.1:3306/graduation_project?
useUnicode=true&characterEncoding=UTF-8
    driver-class-name: com.mysql.jdbc.Driver
    username: ivan
    openMulti: true
    type: com.zaxxer.hikari.HikariDataSource
    idle-timeout: 30000
    multi:
      - key: default1
        password: 123456
        url: jdbc:mysql://127.0.0.1:3306/graduation_project?
useUnicode=true&characterEncoding=UTF-8
        idle-timeout: 20000
        driver-class-name: com.mysql.jdbc.Driver
        username: ivan
        type: com.alibaba.druid.pool.DruidDataSource
      - key: gd
        password: 123456
        url: jdbc:mysql://gd.badtheway.xin:****/graduation_project?
useUnicode=true&characterEncoding=UTF-8
        driver-class-name: com.mysql.jdbc.Driver
        username: ivan
mybatis:
  config-location: classpath:mapper/configure.xml
  mapper-locations: classpath:mapper/*Mapper.xml
```



这边说明一下，spring.datasource路径下的配置即默认数据源的配置，我是为了个人美感以及方便，所以在配置多数据源时使用spring.datasource.multi这个路径，假如需要更改的话修改MultiDataSourceRegister.java里面相应的值就可以了。

最后别忘了在@SpringBootApplication加上@Import(MultiDataSourceRegister.class)

下面是我自己使用的一些切面配置，通过@MultiDataSource\$DataSource注解标记需要切换数据源的类，可以通过方法体参数->方法注解->类注解实现切换数据源。供大家参考：

MultiDataSource.java：



```
package top.ivan.demo.springboot.mapper;

import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;
import org.aspectj.lang.reflect.MethodSignature;
import org.springframework.core.annotation.Order;
import org.springframework.jdbc.datasource.lookup.AbstractRoutingDataSource;
import org.springframework.stereotype.Component;
import org.springframework.util.StringUtils;

import java.lang.annotation.*;
import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.lang.reflect.Parameter;
import java.util.Map;
import java.util.Set;
```

```
public class MultiDataSource extends AbstractRoutingDataSource {

    private final static ThreadLocal<String> DATA_SOURCE_KEY = new ThreadLocal<>(); //保存当前线程的数据源对应的key

    private Set<Object> keySet; //所有数据源的key集合

    private static void switchSource(String key) {
        DATA_SOURCE_KEY.set(key); //切换当前线程的key
    }

    private static void clear() {
        DATA_SOURCE_KEY.remove(); //移除key值
    }

    public static Object execute(String ds, Run run) throws Throwable {
        switchSource(ds);
        try {
            return run.run();
        } finally {
            clear();
        }
    }

    //AbstractRoutingDataSource抽象类实现方法, 即获取当前线程数据源的key
    @Override
    protected Object determineCurrentLookupKey() {
        String key = DATA_SOURCE_KEY.get();
        if (!keySet.contains(key)) {
            logger.info(String.format("can not found datasource by key: '%s', this session may use default datasource", key));
        }
        return key;
    }

    /**
     * 在获取key的集合, 目的只是为了添加一些告警日志
     */
    @Override
    public void afterPropertiesSet() {
        super.afterPropertiesSet();
        try {
            Field sourceMapField =
                AbstractRoutingDataSource.class.getDeclaredField("resolvedDataSources");
            sourceMapField.setAccessible(true);
            Map<Object, javax.sql.DataSource> sourceMap = (Map<Object,
                javax.sql.DataSource>) sourceMapField.get(this);
            this.keySet = sourceMap.keySet();
            sourceMapField.setAccessible(false);
        } catch (NoSuchFieldException | IllegalAccessException e) {
            e.printStackTrace();
        }
    }

    public interface Run {
        Object run() throws Throwable;
    }

    /**
```

```

    * 用于获取AOP切点及数据源key的注解
    */
    @Target({ElementType.METHOD, ElementType.TYPE, ElementType.PARAMETER})
    @Retention(RetentionPolicy.RUNTIME)
    @Documented
    public @interface DataSource {
        String value() default ""; //该值即key值
    }

    /**
     * 声明切面
     */
    @Component
    @Aspect
    @Order(-10) //使该切面在事务之前执行
    public static class DataSourceSwitchInterceptor {

        /**
         * 扫描所有含有@MultiDataSource$DataSource注解的类
         */
        @Pointcut("@within(top.ivan.demo.springboot.mapper.MultiDataSource.DataSource)")
        public void switchDataSource() {
        }

        /**
         * 使用around方式监控
         * @param point
         * @return
         * @throws Throwable
         */
        @Around("switchDataSource()")
        public Object switchByMethod(ProceedingJoinPoint point) throws Throwable {
            Method method = getMethodByPoint(point); //获取执行方法
            Parameter[] params = method.getParameters(); //获取执行参数
            Parameter parameter;
            String source = null;
            boolean isDynamic = false;
            for (int i = params.length - 1; i >= 0; i--) { //扫描是否有参数带有@DataSource注解
                parameter = params[i];
                if (parameter.getAnnotation(DataSource.class) != null && point.getArgs()[i]
instanceof String) {
                    source = (String) point.getArgs()[i]; //key值即该参数的值，要求该参数必须为
String类型

                    isDynamic = true;
                    break;
                }
            }
            if (!isDynamic) { //不存在参数带有DataSource注解
                DataSource dataSource = method.getAnnotation(DataSource.class); //获取方法的
@DataSource注解
                if (null == dataSource || !StringUtils.hasLength(dataSource.value())) { //
方法不含有注解

                    dataSource =
method.getDeclaringClass().getAnnotation(DataSource.class); //获取类级别的@DataSource注解
                }
                if (null != dataSource) {
                    source = dataSource.value(); //设置key值
                }
            }
            return persistBySource(source, point); //继续执行该方法
        }
    }

```

```
private Object persistBySource(String source, ProceedingJoinPoint point) throws
Throwable {
    try {
        switchSource(source); //切换数据源
        return point.proceed(); //执行
    } finally {
        clear(); //清空key值
    }
}

private Method getMethodByPoint(ProceedingJoinPoint point) {
    MethodSignature methodSignature = (MethodSignature) point.getSignature();
    return methodSignature.getMethod();
}
}
```

示例:

```
package top.ivan.demo.springboot.mapper;

import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Param;
import top.ivan.demo.springboot.pojo.ProductPreview;

import java.util.List;

@Mapper
@MultiDataSource.DataSource("ds1")
public interface PreviewMapper {

    //使用ds的值作为key
    List<ProductPreview> getList(@Param("start") int start, @Param("count") int count,
@MultiDataSource.DataSource String ds);

    //使用"ds2"作为key
    @MultiDataSource.DataSource("ds2")
    List<ProductPreview> getList2(@Param("start") int start, @Param("count") int count);

    //使用"ds1"作为key
    List<ProductPreview> getList3(@Param("start") int start, @Param("count") int count);
}
```

这几天刚接触springboot，还处于小白的状态，假如有什么问题的话欢迎大家指教


附上源码文件: <https://files.cnblogs.com/files/badtheway/springboot.zip>

好文要顶

关注我

收藏该文





Ivan_K

关注 - 0

粉丝 - 1

+加关注

« 上一篇: [谈谈Java中使用DataInputStream与DataOutputStream的雷区](#)

posted @ 2018-05-19 17:39 Ivan_K 阅读(5093) 评论(12) 编辑 收藏

评论列表

#1楼 2018-05-29 11:17 Dorla

我按照博主的方式进行动态切换数据源的时候提示找不到数据库，不知道博主能否把项目源代码分享一下看看哪里写的不对

支持(0) 反对(0)

#2楼[楼主] 2018-05-29 14:19 Ivan_K

@ Dorla
你看一下是不是别名的问题，为了兼容HikariDataSource，springboot默认数据源的配置中url和username设置了"jdbc-url"跟"user"两个别名，我这里为了统一也加了这两个别名

支持(0) 反对(0)

#3楼 2018-05-29 17:43 Dorla

@ Ivan_K
我解决了，不是别名的问题，注解不能用在mapper接口类里面，只能用在Service层

支持(0) 反对(0)

#4楼[楼主] 2018-05-30 09:18 Ivan_K

@ Dorla
找不到数据库应该跟注解的问题无关，应该是默认数据源的配置有问题，因为无法进入切面或者key不存在的情况会自动使用默认的数据源。
还有一点，我的注解就是注解在mapper接口上的，运行是正常的。你说不行的原因我猜测可能是该方法没有被aop切入或者mybatis版本问题。当然，在service层注解没有问题，这么设计就是为了适应需求能够更灵活的配置数据源。

支持(0) 反对(0)

#5楼 2018-05-30 09:21 Dorla

@ Ivan_K
找不到数据库就是切换数据源不成功导致的，嗯，博主mapper接口能注入可能是在aop切入的时候把mapper也加进去环绕增强了？我的mybatis版本是1.3.2的，能否借博主的源代码来参考一下？

支持(0) 反对(0)

#6楼[楼主] 2018-05-30 09:38 Ivan_K

@ Dorla
我的mybatis是3.4.x版本的。之前没有放出源码是我的疏忽，现已补上，在文章末尾部分。

支持(0) 反对(0)

#7楼 2018-05-30 09:41 Dorla

好的，谢谢博主，有问题再探讨

支持(0) 反对(0)

#8楼 2018-05-30 10:52 Dorla

@ Ivan_K
找到问题所在了，缺少了aspectj依赖导致在mapper层不能进入切面

支持(1) 反对(0)

#9楼[楼主] 2018-05-30 11:44 Ivan_K

@ Dorla
可以考虑使用springboot自带的aop注解，应该可以达到同样的效果：
<dependency>
<groupId>org.springframework.boot</groupId>


```
<artifactId>spring-boot-starter-aop</artifactId>
</dependency>
```

支持(0) 反对(0)

10楼 2018-05-30 11:45 Dorla

@ Ivan_K
我一开始也是以为加了自带的就不用加aspectj依赖了，但是就是进不了切面

支持(0) 反对(0)

11楼[楼主] 2018-05-30 11:53 Ivan_K

@ Dorla
额，我的项目把那三个依赖去掉换成spring-boot-starter-aop之后也是正常运行的。假如不行的话只能猜测是不是版本问题了😂

支持(0) 反对(0)

12楼 2018-05-30 12:32 Dorla

@ Ivan_K
版本没问题，我用的是最新的基本

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](#)。

- 【课程】IT热门课程全场6折起，报名还额外赠送大礼
- 【推荐】超50万C++/C#源码：大型实时仿真HMI组态CAD\GIS图形源码！
- 【推荐】百度云猪你开年行大运，红包疯狂拿
- 【推荐】专业便捷的企业级代码托管服务 - Gitee 码云
- 【活动】2019开源技术盛宴(6.24~26上海世博中心)
- 【推荐】免费领取Java中高级工程师名企面试题

相关博文：

- [TreeView动态绑定数据源](#)
- [数据源绑定](#)
- [数据源绑定GridView](#)
- [数据源绑定DataGridViewComboBox](#)
- [扩展GeoServer数据源](#)

最新新闻：

- [Android Beta官方社区转移到Reddit](#)
- [供应链消息称今年会有新的iPad，但它会继续保留Touch ID和耳机孔](#)
- [清华姚班首届毕业生、17科满分传奇，现斩获“诺贝尔风向标”斯隆奖](#)
- [微软的“Andromeda”和“Polaris” 要被Santorini取代](#)
- [微软发布了三款适用于Windows 10的全新春季主题免费壁纸包](#)
- » [更多新闻...](#)