Spring Boot 集成MyBatis - 偶尔记一下 - 博客频道

在集成MyBatis前，我们先配置一个druid数据源。

Spring Boot 集成druid

`druid`有很多个配置选项，使用spring Boot 的配置文件可以方便的配置`druid`。

在`application.yml`配置文件中写上：

```
spring:
    datasource:
        name: test
        url: jdbc:mysql://192.168.16.137:3306/test
        username: root
        password:
        # 使用druid数据源
        type: com.alibaba.druid.pool.DruidDataSource
        driver-class-name: com.mysql.jdbc.Driver
        filters: stat
        maxActive: 20
        initialSize: 1
        maxWait: 60000
        minIdle: 1
        timeBetweenEvictionRunsMillis: 60000
        minEvictableIdleTimeMillis: 300000
        validationQuery: select 'x'
        testWhileIdle: true
        testOnBorrow: false
        testOnReturn: false
        poolPreparedStatements: true
        maxOpenPreparedStatements: 20
```

```
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

Spring Boot 集成MyBatis有两种方式，一种简单的方式就是使用MyBatis官方提供的：

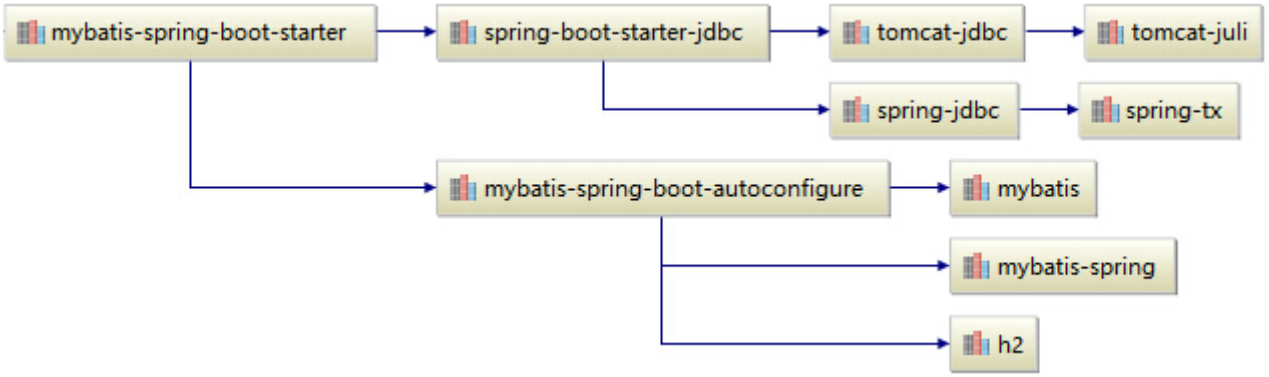
另外一种方式就是仍然用类似`mybatis-spring`的配置方式，这种方式需要自己写一些代码，但是可以很方便的控制MyBatis的各项配置。

一、mybatis-spring-boot-starter方式

在`pom.xml`中添加依赖：

```
<dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
    <version>1.0.0</version>
</dependency>
1
2
3
4
5
```

`mybatis-spring-boot-starter`依赖树如下：



其中`mybatis`使用的3.3.0版本，可以通过：
`<mybatis.version>3.3.0</mybatis.version>`属性修改默认版本。
`mybatis-spring`使用版本1.2.3，可以通过：
`<mybatis-spring.version>1.2.3</mybatis-spring.version>`修改默认版本。

在`application.yml`中增加配置：

```
mybatis:
  mapperLocations: classpath:mapper/*.xml
  typeAliasesPackage: tk.mapper.model
```

除了上面常见的两项配置，还有：

- mybatis.config：mybatis-config.xml配置文件的路径
- mybatis.typeHandlersPackage：扫描typeHandlers的包
- mybatis.checkConfigLocation：检查配置文件是否存在
- mybatis.executorType：设置执行模式（`SIMPLE, REUSE, BATCH`），默认为`SIMPLE`

二、mybatis-spring方式

这种方式和平常的用法比较接近。需要添加`mybatis`依赖和`mybatis-spring`依赖。

然后创建一个`MyBatisConfig`配置类：

```
/**
 * MyBatis基础配置
 *
 * @author liuzh
 * @since 2015-12-19 10:11
 */
```

```
@Configuration
@EnableTransactionManagement
public class MyBatisConfig implements TransactionManagementConfigurer
{

    @Autowired
    DataSource dataSource;

    @Bean(name = "sqlSessionFactory")
    public SqlSessionFactory sqlSessionFactoryBean() {
        SqlSessionFactoryBean bean = new SqlSessionFactoryBean();
        bean.setDataSource(dataSource);
        bean.setTypeAliasesPackage("tk.mybatis.springboot.model");

        //分页插件
        PageHelper pageHelper = new PageHelper();
        Properties properties = new Properties();
        properties.setProperty("reasonable", "true");
        properties.setProperty("supportMethodsArguments", "true");
        properties.setProperty("returnPageInfo", "check");
        properties.setProperty("params", "count=countSql");
        pageHelper.setProperties(properties);

        //添加插件
        bean.setPlugins(new Interceptor[]{pageHelper});

        //添加XML目录
        ResourcePatternResolver resolver = new
PathMatchingResourcePatternResolver();
        try {

bean.setMapperLocations(resolver.getResources("classpath:mapper/*.xml
"));
            return bean.getObject();
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }
    }
```

```
    @Bean
    public SqlSessionTemplate sqlSessionTemplate(SqlSessionFactory
sqlSessionFactory) {
        return new SqlSessionTemplate(sqlSessionFactory);
    }

    @Bean
    @Override
    public PlatformTransactionManager
annotationDrivenTransactionManager() {
        return new DataSourceTransactionManager(dataSource);
    }
}
```

- 28
- 29
- 30
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- 48
- 49
- 50
- 51
- 52
- 53

- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45

- 46
- 47
- 48
- 49
- 50
- 51
- 52
- 53

上面代码创建了一个`SqlSessionFactory`和一个`SqlSessionTemplate`，为了支持注解事务，增加了`@EnableTransactionManagement`注解，并且反回了一个`PlatformTransactionManager`Bean。

另外应该注意到这个配置中没有`MapperScannerConfigurer`，如果我们想要扫描MyBatis的Mapper接口，我们就需要配置这个类，这个配置我们需要单独放到一个类中。

```
/**
 * MyBatis扫描接口
 *
 * @author liuzh
 * @since 2015-12-19 14:46
 */
@Configuration
//TODO 注意，由于MapperScannerConfigurer执行的比较早，所以必须有下面的注解
@AutoConfigureAfter(MyBatisConfig.class)
public class MyBatisMapperScannerConfig {

    @Bean
    public MapperScannerConfigurer mapperScannerConfigurer() {
        MapperScannerConfigurer mapperScannerConfigurer = new
MapperScannerConfigurer();

mapperScannerConfigurer.setSqlSessionFactoryBeanName("sqlSessionFacto
ry");

mapperScannerConfigurer.setBasePackage("tk.mybatis.springboot.mapper"
);
        return mapperScannerConfigurer;
    }
```

```
}
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17

```
18
19
20
```

这个配置一定要注意`@AutoConfigureAfter(MyBatisConfig.class)`，必须有这个配置，否则会有异常。原因就是这个类执行的比较早，由于`sqlSessionFactory`还不存在，后续执行出错。

做好上面配置以后就可以使用MyBatis了。

关于分页插件和通用Mapper集成
分页插件作为插件的例子在上面代码中有。

通用Mapper配置实际就是配置`MapperScannerConfigurer`的时候使用`tk.mybatis.spring.mapper.MapperScannerConfigurer`即可，配置属性使用`Properties`。

Spring Boot集成MyBatis的基础项目

我上传到github一个采用第二种方式的集成项目，并且集成了分页插件和通用Mapper，项目包含了简单的配置和操作，仅作为参考。

项目地址：https://github.com/abel533/MyBatis-Spring-Boot

分页插件和通用Mapper的相关信息可以通过上面地址找到。

Spring Boot 系列
由于我博客Spring Boot 系列文章还不够多，所以暂时不打算创建专栏，如果再多几篇我就建专栏。

@AutoConfigureAfter(MyBatisConfig.class
这句话拯救了我！！！！内牛满面！！

@AutoConfigureAfter(MyBatisConfig.class
这句话拯救了我！！！！感谢楼主，内牛满面！！！

mybatis-spring-boot-starter
添加 typeAliasesPackage: tk.mapper.model 还是无法执行哈
楼主能出个 spring-boot-starter 的 继承 项目吗

mybatis-spring-boot-starter

添加 typeAliasesPackage: tk.mapper.model 还是无法执行哈
楼主能出个 spring-boot-starter 的 继承 项目吗

楼主请问一下啊，MyBatisMapperScannerConfig 这个类里面的一些配置信息我抽出来，然后用配置类注入的形式进行获取，但是获取的都是空的！但是MyBatisConfig这个类的配置信息我抽取出来后，用配置类注入的时候，却可以拿到值！！！不解，求或啊~~~配置类是参考你的另外一篇博文的；

```
@ConfigurationProperties(prefix="my")
public class Config {
private String name;
private Integer port;
private List<String> servers = new ArrayList<String>();

public String geName(){
return this.name;
}

public Integer gePort(){
return this.port;
}
public List<String> getServers() {
return this.servers;
}
}
```

Cannot enhance @Configuration bean definition 'myBatisMapperScannerConfig' since its singleton
instance has been created too early. The typical cause is a non-static @Bean method with a BeanDefinitionRegistryPostProcessor return type: Consider declaring such methods as 'static'.
我使用的pagehelper 和通用的mapper 请问在项目启动的时候提示这个，就卡在这里，请问我怎么能把这个去掉，

楼主，你好！请问啥时候能发布springboot+mybatis配置多数据源的版本了

版本号3.3.6
通过主键更新的时候，where后面跟的条件是把所有的entity条件都加上，而不是

```
</set>
where id = #{id,jdbcType=BIGINT}
</update>
```

而是，where后面把实体类中的所有不为空的值全部加入按到条件中来了。这样就导致查询不出来这个信息。