

Spring Boot集成MyBatis

Spring Boot集成MyBatis

1.整合Druid数据源

2.整合MyBatis

运行插件---生成代码---测试

1.整合Druid数据源

1、引入Jar包

```
1
2 <dependencies>
3   <dependency>
4     <groupId>org.springframework.boot</groupId>
5     <artifactId>spring-boot-starter-jdbc</artifactId>
6   </dependency>
7   <dependency>
8     <groupId>org.springframework.boot</groupId>
9     <artifactId>spring-boot-starter-web</artifactId>
10  </dependency>
11  <!--mybatis的场景启动器
12  因为MyBatis它是Spring data jpa
13  <dependency>
14    <groupId>org.mybatis.spring.boot</groupId>
15    <artifactId>mybatis-spring-boot-starter</artifactId>
16    <version>2.1.4</version>
17  </dependency>
18  -->
19  <dependency>
20    <groupId>com.alibaba</groupId>
21    <artifactId>druid</artifactId>
22    <version>1.2.3</version>
23  </dependency>
```

```
24
25 <dependency>
26 <groupId>mysql</groupId>
27 <artifactId>mysql-connector-java</artifactId>
28 <scope>runtime</scope>
29 </dependency>
30 <dependency>
31 <groupId>org.springframework.boot</groupId>
32 <artifactId>spring-boot-starter-test</artifactId>
33 <scope>test</scope>
34 </dependency>
35 </dependencies>
```

2.application.yml配置

```
1 #数据源
2 spring:
3   datasource:
4     username: root
5     password: 123456
6     url: jdbc:mysql://localhost:3306/springboot_mybatis?characterEn
      coding=utf8&useSSL=false&serverTimezone=UTC&
7     driver-class-name: com.mysql.cj.jdbc.Driver
8     type: com.alibaba.druid.pool.DruidDataSource
9
10  # 数据源其他配置
11  initialSize: 5
12  minIdle: 5
13  maxActive: 20
14  maxWait: 60000
15  timeBetweenEvictionRunsMillis: 60000
16  minEvictableIdleTimeMillis: 300000
17  validationQuery: SELECT 1 FROM DUAL
18  testWhileIdle: true
19  testOnBorrow: false
20  testOnReturn: false
```

```

21 poolPreparedStatements: true
22 # 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
23 filters: stat,wall
24 maxPoolPreparedStatementPerConnectionSize: 20
25 useGlobalDataSourceStat: true
26 connectionProperties: druid.stat.mergeSql=true;druid.stat.slow
    SqlMillis=500
27 schema: classpath:sql/mybatis.sql
28 initialization-mode: ALWAYS

```

3.读取配置类DruidConfig

```

1 /**
2  * 数据源配置类
3  */
4 @Configuration
5 public class DruidConfig {
6     // 将所有前缀为spring.datasource下的配置项都加载到DataSource中
7     @ConfigurationProperties(prefix = "spring.datasource")
8     @Bean
9     public DataSource druidDataSource() {
10         return new DruidDataSource();
11     }
12
13     @Bean
14     public ServletRegistrationBean statViewServlet() {
15         ServletRegistrationBean servletRegistrationBean = new ServletR
            egistrationBean(new StatViewServlet(), "/druid/*");
16         // 添加IP白名单
17         servletRegistrationBean.addInitParameter("allow",
            "127.0.0.1");
18         // 添加IP黑名单，当白名单和黑名单重复时，黑名单优先级更高
19         servletRegistrationBean.addInitParameter("deny", "127.0.0.1");
20         // 添加控制台管理用户
21         servletRegistrationBean.addInitParameter("loginUsername", "adm
            in");
22         servletRegistrationBean.addInitParameter("loginPassword", "123
            456");

```

```

23 // 是否能够重置数据
24 servletRegistrationBean.addInitParameter("resetEnable", "false");
25 return servletRegistrationBean;
26 }
27
28 /**
29  * 配置服务过滤器
30  *
31  * @return 返回过滤器配置对象
32  */
33 @Bean
34 public FilterRegistrationBean statFilter() {
35     FilterRegistrationBean filterRegistrationBean = new FilterRegistrationBean(new WebStatFilter());
36     // 添加过滤规则
37     filterRegistrationBean.addUrlPatterns("/");
38     // 忽略过滤格式
39     filterRegistrationBean.addInitParameter("exclusions", "*.js,*.gif,*.jpg,*.png,*.css,*.ico,/druid/*,");
40     return filterRegistrationBean;
41 }
42 }

```

其实没有必要一个个手动去配置， druid 启动starter

```

1 <dependency>
2 <groupId>com.alibaba</groupId>
3 <artifactId>druid-spring-boot-starter</artifactId>
4 <version>1.2.3</version>
5 </dependency>

```

druid 自动配置类

```

1
2 @Configuration
3 @ConditionalOnClass(DruidDataSource.class)
4 @AutoConfigureBefore(DataSourceAutoConfiguration.class)

```

```

5 @EnableConfigurationProperties({DruidStatProperties.class, DataS
   ourceProperties.class})
6 @Import({DruidSpringAopConfiguration.class,
7   DruidStatViewServletConfiguration.class,
8   DruidWebStatFilterConfiguration.class,
9   DruidFilterConfiguration.class})
10 public class DruidDataSourceAutoConfigure {
11
12   private static final Logger LOGGER = LoggerFactory.getLogger(D
   ruidDataSourceAutoConfigure.class);
13
14   @Bean(initMethod = "init")
15   @ConditionalOnMissingBean
16   public DataSource dataSource() {
17     LOGGER.info("Init DruidDataSource");
18     return new DruidDataSourceWrapper();
19   }
20 }

```

2.整合MyBatis

2.1生成MyBatis代码:

pom.xml

```

1  <!-- Mybatis-Generator插件，自动生成代码 -->
2  <plugin>
3    <groupId>org.mybatis.generator</groupId>
4    <artifactId>mybatis-generator-maven-plugin</artifactId>
5    <version>1.3.5</version>
6    <configuration>
7      <configurationFile>${project.basedir}/src/main/resources/genera
   torConfig.xml</configurationFile>
8      <verbose>true</verbose>
9      <overwrite>true</overwrite>
10     </configuration>
11     <dependencies>
12     <!-- 必须要引入数据库驱动 -->

```

```

13 <dependency>
14 <groupId>mysql</groupId>
15 <artifactId>mysql-connector-java</artifactId>
16 <!--必须制定版本-->
17 <version>8.0.22</version>
18 </dependency>
19 </dependencies>
20 </plugin>
21 </plugins>
22 </build>

```

generatorConfig.xml

```

1
2
3 <!DOCTYPE generatorConfiguration PUBLIC
4   "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
5   "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">
6 <generatorConfiguration>
7
8
9 <!--如果需要使用 command的方式生成需要配置数据库驱动的jar包路径
10  <classPathEntry location="指定数据驱动的磁盘路径"/>-->
11
12 <!--context 生成上下文 配置生成规则
13  id 随意写
14  targetRuntime 生成策略
15  MyBatis3DynamicSql 默认的，会生成 动态生成sql的方式（没有xml）
16  MyBatis3 生成通用的查询，可以指定动态where条件
17  MyBatis3Simple 只生成简单的CRUD
18  -->
19 <context id="simple" targetRuntime="MyBatis3Simple">
20
21
22 <commentGenerator>

```

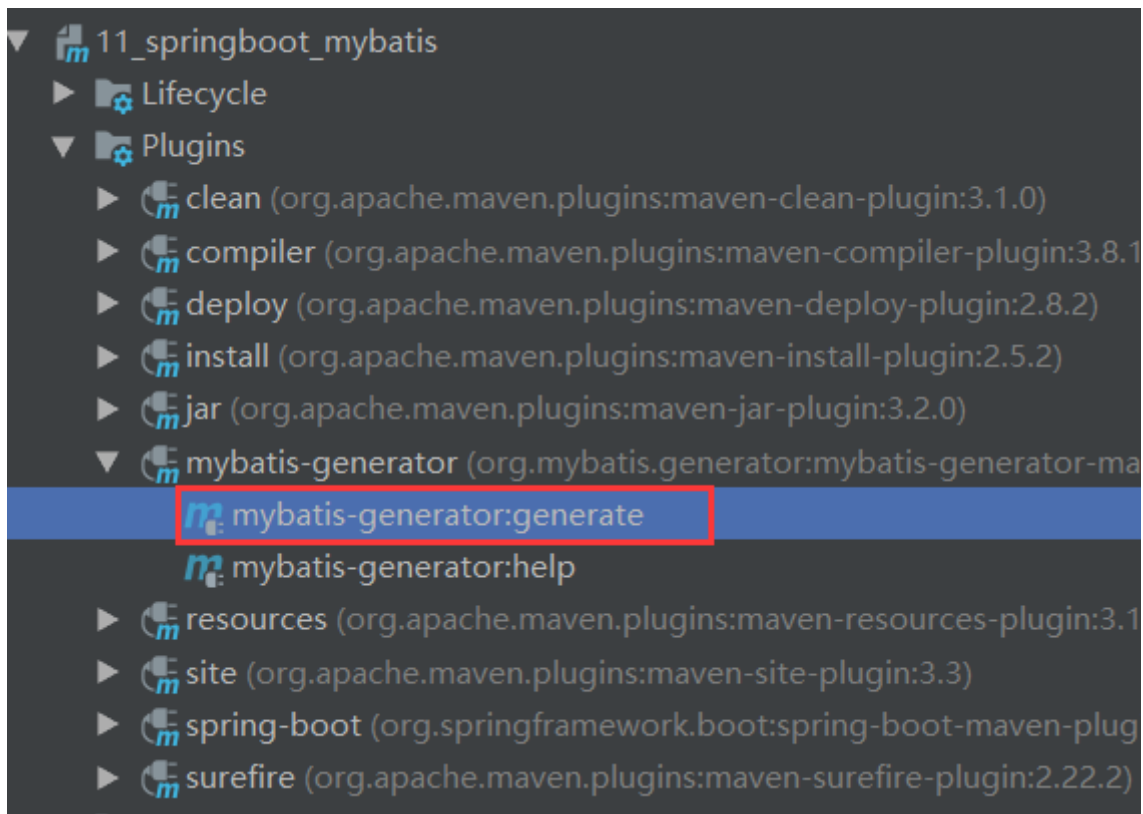
```
23  <!--设置是否生成注释 true 不生成 注意： 如果不生成注释，下次生成代码  
    就不会进行合并-->  
24  <property name="suppressAllComments" value="true"/>  
25  </commentGenerator>  
26  <!--数据源 -->  
27  <jdbcConnection driverClass="com.mysql.jdbc.Driver"  
28  connectionURL="jdbc:mysql://localhost:3306/mybatis"  
29  userId="root"  
30  password="123456"/>  
31  
32  <!--pojo  
33  javaModelGenerator java实体生成规则(POJO)  
34  targetPackage 生成到哪个包下  
35  targetProject 生成到当前文件的哪个相对路径下  
36  -->  
37  <javaModelGenerator targetPackage="cn.tulingxueyuan.pojo" targ  
    etProject="src/main/java"/>  
38  <!--mapper xml映射文件  
39  sqlMapGenerator mapper xml映射文件生成规则  
40  targetPackage 生成到哪个包下  
41  targetProject 生成到当前文件的哪个相对路径下  
42  -->  
43  <sqlMapGenerator targetPackage="cn.tulingxueyuan.mapper" targe  
    tProject="src/main/resources"></sqlMapGenerator>  
44  <!--mapper接口  
45  javaClientGenerator mapper mapper接口生成规则  
46  type 指定生成的方式  
47  1.使用注解的方式生成  
48  2.使用接口绑定的方式生成（要配置sqlMapGenerator）  
49  targetPackage 生成到哪个包下  
50  targetProject 生成到当前文件的哪个相对路径下-->  
51  <javaClientGenerator type="XMLMAPPER" targetPackage="cn.tuling  
    xueyuan.mapper" targetProject="src/main/java"/>  
52  
53  
54  <!--配置哪些表需要进行代码生成
```

```

55  tableName 表名
56  domainObjectName pojo类名
57  mapperName 对应mapper接口的类名 和 mapper xml文件名
58  -->
59  <table tableName="emp" domainObjectName="Emp" mapperName="EmpM
  apper" />
60  <table tableName="dept" domainObjectName="Dept" mapperName="De
  ptMapper" />
61  </context>
62  </generatorConfiguration>

```

运行插件---生成代码



2.2 整合Mybatis

1.引入jar包

```

1
2 1、引入Jar包
3 <dependency>
4   <groupId>org.mybatis.spring.boot</groupId>
5   <artifactId>mybatis-spring-boot-starter</artifactId>
6   <version>1.3.2</version>

```



```
7 </dependency>
8
9
```

application.yml

```
1  mybatis:
2  # 映射文件所在路径
3  mapper-locations: classpath:mappers/*.xml
4  # pojo类所在包路径
5  type-aliases-package: com.cx.user.model
```

MyBatis自动配置原理

```
1 @Bean
2 @ConditionalOnMissingBean
3 public SqlSessionFactory sqlSessionFactory(DataSource
4     dataSource) throws Exception {
5     SqlSessionFactoryBean factory = new SqlSessionFactoryBean();
6     factory.setDataSource(dataSource);
7     factory.setVfs(SpringBootVFS.class);
8     // 设置Mybaitis的全局配置文件
9     if (StringUtils.hasText(this.properties.getConfigLocation())) {
10         factory.setConfigLocation(this.resourceLoader.getResource(this.pro
11             perties.getConfigLocation()));
12     }
13     // ? 有另一种定制方式的体现
14     applyConfiguration(factory);
15     // 相当于mybatis全局配置文件中
16     /*<properties>
17     <property name="" value=""/>
18     </properties>*/
19     if (this.properties.getConfigurationProperties() != null) {
20         factory.setConfigurationProperties(this.properties.getConfigur
21             ationProperties());
22     }
23     // 就是配置插件-拦截器 只需要配置一个实现了Interceptor的接口为Bean
```

```
22  if (!ObjectUtils.isEmpty(this.interceptors)) {
23  factory.setPlugins(this.interceptors);
24  }
25  // 设置数据库厂商id
26  if (this.databaseIdProvider != null) {
27  factory.setDatabaseIdProvider(this.databaseIdProvider);
28  }
29  // 设置别名: 去application.yml中获取mybatis.typeAliasesPackage
30  if (StringUtils.hasLength(this.properties.getTypeAliasesPackage())) {
31  factory.setTypeAliasesPackage(this.properties.getTypeAliasesPackage());
32  }
33  // 可以通过父类过滤哪些类需要使用别名
34  比如: pojo.user extends basePojo
35  pojo.user2
36  去application.yml中设置mybatis.typeAliasesSuperType: com.tulingxueyuan.pojo.basePojo
37  if (this.properties.getTypeAliasesSuperType() != null) {
38  factory.setTypeAliasesSuperType(this.properties.getTypeAliasesSuperType());
39  }
40  // 设置类型处理器
41  <typeHandlers>
42  <package name=""/>
43  </typeHandlers>
44  if (StringUtils.hasLength(this.properties.getTypeHandlersPackage())) {
45  factory.setTypeHandlersPackage(this.properties.getTypeHandlersPackage());
46  }
47  // 设置类型处理器
48  <typeHandlers>
49  <typeHandler handler=""
50  </typeHandlers>
51  if (!ObjectUtils.isEmpty(this.typeHandlers)) {
52  factory.setTypeHandlers(this.typeHandlers);
```

```

53     }
54     // 设置mapper.xml映射文件: mapper-locations: classpath:com/tulin
    gxueyuan/mapper/*Mapper.xml
55     if (!ObjectUtils.isEmpty(this.properties.resolveMapperLocations())) {
56         factory.setMapperLocations(this.properties.resolveMapperLocations());
57     }
58     Set<String> factoryPropertyNames = Stream
59         .of(new BeanWrapperImpl(SqlSessionFactoryBean.class).getPropertyDescriptors()).map(PropertyDescriptor::getName)
60         .collect(Collectors.toSet());
61     Class<? extends LanguageDriver> defaultLanguageDriver = this.properties.getDefaultScriptingLanguageDriver();
62     if (factoryPropertyNames.contains("scriptingLanguageDrivers")
        && !ObjectUtils.isEmpty(this.languageDrivers)) {
63         // Need to mybatis-spring 2.0.2+
64         factory.setScriptingLanguageDrivers(this.languageDrivers);
65         if (defaultLanguageDriver == null && this.languageDrivers.length == 1) {
66             defaultLanguageDriver = this.languageDrivers[0].getClass();
67         }
68     }
69     if (factoryPropertyNames.contains("defaultScriptingLanguageDriver")) {
70         // Need to mybatis-spring 2.0.2+
71         factory.setDefaultScriptingLanguageDriver(defaultLanguageDriver);
72     }
73
74     return factory.getObject();
75 }

```

- 如果依然放不下mybatis全局配置文件，springboot 还是支持的：
 - 配置application.yml

```

1 mybatis:
2   config-location: classpath:mybatis-config.xml

```

- mybatis-config.xml

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE configuration
3   PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
4   "http://mybatis.org/dtd/mybatis-3-config.dtd">
5 <!--就是DOCTYPE后面对应的根节点-->
6 <configuration>
7
8 <!--mybatis的设置选项 可以改变mybatis运行时行为-->
9 <settings>
10  <setting name="mapUnderscoreToCamelCase" value="true"/>
11 </settings>
12
13 <!--类型别名可为 Java 类型设置一个缩写名字。 它仅用于 XML 配置，意在降低冗余的全限定类名书写-->
14 <typeAliases>
15  <package name="com.tulingxueyuan.pojo"/>
16 </typeAliases>
17
18 </configuration>

```

- 如果要设置mybatis的settings怎么设置呢？
 - 1.可以通过mybatis全局配置文件设置
 - 2. 也可以通过在application.yml中配置configuration
 - configuration 它封装mybatis所有信息

```

1 configuration:
2   mapUnderscoreToCamelCase: true

```

- configuration 什么情况=null呢？
 - 没有在application.yml中配置configuration 就会为null
- 如果没有在application.yml中配置config-location 就会new new Configuration();
- 要定制mybatis

- 1. 使用mybatis全局配置文件
- 2. 可以使用application.yml中配置configuration + ConfigurationCustomizer
- 要么使用mybatis的东西，要么使用springboot的，只能用1种

```
1 private void applyConfiguration(SqlSessionFactoryBean factory) {
2     Configuration configuration =
3     this.properties.getConfiguration();
4     if (configuration == null && !StringUtils.hasText(this.properties
5     .getConfigLocation())) {
6         configuration = new Configuration();
7     }
8     if (configuration != null && !CollectionUtils.isEmpty(this.conf
9     igationCustomizers)) {
10        for (ConfigurationCustomizer customizer : this.configurationCus
11        tomizers) {
12            customizer.customize(configuration);
13        }
14    }
15    factory.setConfiguration(configuration);
16 }
```