自：https://blog.csdn.net/xiaolyuh123/article/details/73506189

**Spring Boot+Mybatis+Pagehelper分页**

2017年06月20日 19:00:12 [xiaolyuh123](https://me.csdn.net/xiaolyuh123) 阅读数：45961 标签： [Spring Boot+Mybatis](http://so.csdn.net/so/search/s.do?q=Spring%20Boot+Mybatis&t=blog)[Mybatis+Pagehelper分页](http://so.csdn.net/so/search/s.do?q=Mybatis+Pagehelper%E5%88%86%E9%A1%B5&t=blog)[Mybatis Pagehelper分页](http://so.csdn.net/so/search/s.do?q=Mybatis%20Pagehelper%E5%88%86%E9%A1%B5&t=blog) 更多

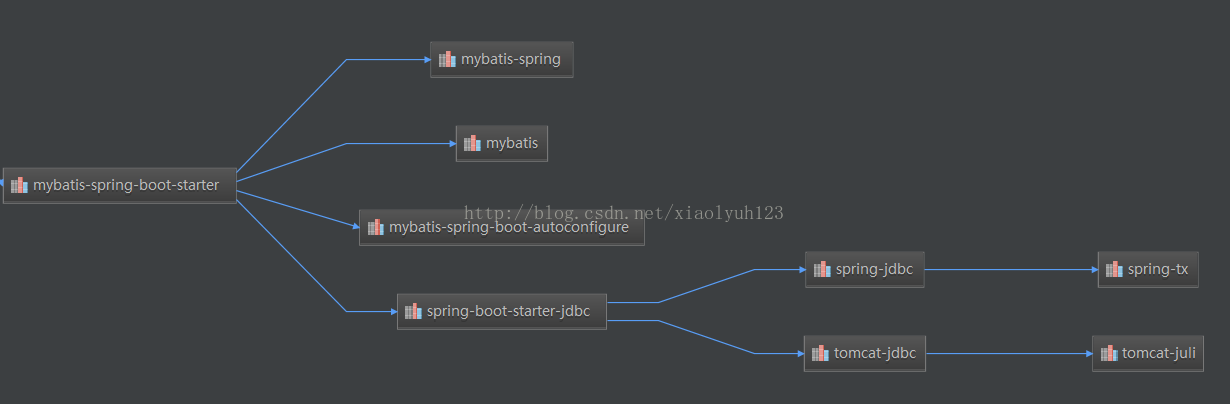
个人分类： [JavaEE开发的颠覆者 Spring Boot实战](https://blog.csdn.net/xiaolyuh123/article/category/1311530)

所属专栏： [springboot微服务](https://blog.csdn.net/column/details/21673.html)

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/xiaolyuh123/article/details/73506189

Spring Boot 集成MyBatis和Pagehelper分页插件

mybatis-spring-boot-starter依赖树如下：



**pom配置**

1. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3. <modelVersion>4.0.0</modelVersion>
4. <groupId>com.xiaolyuh</groupId>
5. <artifactId>spring-boot-student-mybatis</artifactId>
6. <version>0.0.1-SNAPSHOT</version>
7. <packaging>jar</packaging>
8. <name>spring-boot-student-mybatis</name>
10. *<!-- 添加Spring Boot的父类依赖，这样当前项目就是Spring Boot项目了。 spring-boot-starter-parent是一个特殊的starter，他用来*
11. *提供相关的maven默认依赖， 使用它之后，常用的依赖可以省去version标签 -->*
12. <parent>
13. <groupId>org.springframework.boot</groupId>
14. <artifactId>spring-boot-starter-parent</artifactId>
15. <version>1.5.3.RELEASE</version>
16. <relativePath/> *<!-- lookup parent from repository -->*
17. </parent>
19. <properties>
20. <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
21. <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
22. <java.version>1.8</java.version>
23. </properties>
25. <dependencies>
26. <dependency>
27. <groupId>org.springframework.boot</groupId>
28. <artifactId>spring-boot-starter</artifactId>
29. </dependency>
30. <dependency>
31. <groupId>org.springframework.boot</groupId>
32. <artifactId>spring-boot-starter-web</artifactId>
33. </dependency>
34. <dependency>
35. <groupId>mysql</groupId>
36. <artifactId>mysql-connector-java</artifactId>
37. </dependency>
39. <dependency>
40. <groupId>org.mybatis.spring.boot</groupId>
41. <artifactId>mybatis-spring-boot-starter</artifactId>
42. <version>1.3.0</version>
43. </dependency>
44. *<!--pagehelper -->*
45. <dependency>
46. <groupId>com.github.pagehelper</groupId>
47. <artifactId>pagehelper-spring-boot-starter</artifactId>
48. <version>1.1.1</version>
49. </dependency>
51. <dependency>
52. <groupId>com.alibaba</groupId>
53. <artifactId>fastjson</artifactId>
54. <version>1.2.31</version>
55. </dependency>
57. <dependency>
58. <groupId>org.springframework.boot</groupId>
59. <artifactId>spring-boot-starter-test</artifactId>
60. <scope>test</scope>
61. </dependency>
62. </dependencies>
64. <build>
65. <plugins>
66. <plugin>
67. <groupId>org.springframework.boot</groupId>
68. <artifactId>spring-boot-maven-plugin</artifactId>
69. </plugin>
70. </plugins>
71. </build>
73. </project>

**application.properties配置**

1. server.port=80
2. *# 数据源配置*
3. spring.datasource.url=jdbc:mysql://localhost:3306/ssb\_test
4. spring.datasource.driver-class-name=com.mysql.jdbc.Driver
5. spring.datasource.username=root
6. spring.datasource.password=root
7. *#连接池配置*
8. *#spring.datasource.type=org.apache.commons.dbcp2.BasicDataSource*
10. *#mybatis*
11. *#entity扫描的包名*
12. mybatis.type-aliases-package=com.xiaolyuh.domain.model
13. *#Mapper.xml所在的位置*
14. mybatis.mapper-locations=classpath\*:/mybaits/\*Mapper.xml
16. *#pagehelper分页插件配置*
17. pagehelper.helperDialect=mysql
18. pagehelper.reasonable=**true**
19. pagehelper.supportMethodsArguments=**true**
20. pagehelper.params=count=countSql
22. *#日志配置*
23. logging.level.com.xiaolyuh=debug
24. logging.level.org.springframework.web=debug
25. logging.level.org.springframework.transaction=debug
26. logging.level.org.mybatis=debug
28. debug=**false**

除了上面常见的两项配置，还有：

1. **mybatis**.config-location=mybatis-config.xml配置文件的路径
2. **mybatis**.**type**-handlers-package=扫描typeHandlers的包
3. **mybatis**.check-config-location=检查配置文件是否存在
4. **mybatis**.executor-**type**=设置执行模式（**SIMPLE**, **REUSE**, **BATCH**），默认为**SIMPLE**

**Mapping XML文件**

在resources文件夹下创建mybaits/PersonMapper.xml文件

1. **<?xml version="1.0" encoding="UTF-8" ?>**
2. **<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >**
3. <mapper namespace="com.xiaolyuh.domain.mapper.PersonMapper" >
4. <resultMap id="BaseResultMap" type="com.xiaolyuh.domain.model.Person" >
5. *<!--*
6. *WARNING - @mbggenerated*
7. *This element is automatically generated by MyBatis Generator, do not modify.*
8. *-->*
9. <id column="id" property="id" jdbcType="BIGINT" />
10. <result column="name" property="name" jdbcType="VARCHAR" />
11. <result column="age" property="age" jdbcType="INTEGER" />
12. <result column="address" property="address" jdbcType="VARCHAR" />
13. </resultMap>
14. <sql id="Base\_Column\_List" >
15. *<!--*
16. *WARNING - @mbggenerated*
17. *This element is automatically generated by MyBatis Generator, do not modify.*
18. *-->*
19. id, name, age, address
20. </sql>
21. <select id="selectByPrimaryKey" resultMap="BaseResultMap" parameterType="java.lang.Long" >
22. *<!--*
23. *WARNING - @mbggenerated*
24. *This element is automatically generated by MyBatis Generator, do not modify.*
25. *-->*
26. select
27. <include refid="Base\_Column\_List" />
28. from person
29. where id = #{id,jdbcType=BIGINT}
30. </select>
31. <delete id="deleteByPrimaryKey" parameterType="java.lang.Long" >
32. *<!--*
33. *WARNING - @mbggenerated*
34. *This element is automatically generated by MyBatis Generator, do not modify.*
35. *-->*
36. delete from person
37. where id = #{id,jdbcType=BIGINT}
38. </delete>
39. <insert id="insert" parameterType="com.xiaolyuh.domain.model.Person" >
40. *<!--*
41. *WARNING - @mbggenerated*
42. *This element is automatically generated by MyBatis Generator, do not modify.*
43. *-->*
44. <selectKey resultType="java.lang.Long" keyProperty="id" order="AFTER" >
45. SELECT LAST\_INSERT\_ID()
46. </selectKey>
47. insert into person (name, age, address
48. )
49. values (#{name,jdbcType=VARCHAR}, #{age,jdbcType=INTEGER}, #{address,jdbcType=VARCHAR}
50. )
51. </insert>
52. <insert id="insertSelective" parameterType="com.xiaolyuh.domain.model.Person" >
53. *<!--*
54. *WARNING - @mbggenerated*
55. *This element is automatically generated by MyBatis Generator, do not modify.*
56. *-->*
57. <selectKey resultType="java.lang.Long" keyProperty="id" order="AFTER" >
58. SELECT LAST\_INSERT\_ID()
59. </selectKey>
60. insert into person
61. <trim prefix="(" suffix=")" suffixOverrides="," >
62. <if test="name != null" >
63. name,
64. </if>
65. <if test="age != null" >
66. age,
67. </if>
68. <if test="address != null" >
69. address,
70. </if>
71. </trim>
72. <trim prefix="values (" suffix=")" suffixOverrides="," >
73. <if test="name != null" >
74. #{name,jdbcType=VARCHAR},
75. </if>
76. <if test="age != null" >
77. #{age,jdbcType=INTEGER},
78. </if>
79. <if test="address != null" >
80. #{address,jdbcType=VARCHAR},
81. </if>
82. </trim>
83. </insert>
84. <update id="updateByPrimaryKeySelective" parameterType="com.xiaolyuh.domain.model.Person" >
85. *<!--*
86. *WARNING - @mbggenerated*
87. *This element is automatically generated by MyBatis Generator, do not modify.*
88. *-->*
89. update person
90. <set >
91. <if test="name != null" >
92. name = #{name,jdbcType=VARCHAR},
93. </if>
94. <if test="age != null" >
95. age = #{age,jdbcType=INTEGER},
96. </if>
97. <if test="address != null" >
98. address = #{address,jdbcType=VARCHAR},
99. </if>
100. </set>
101. where id = #{id,jdbcType=BIGINT}
102. </update>
103. <update id="updateByPrimaryKey" parameterType="com.xiaolyuh.domain.model.Person" >
104. *<!--*
105. *WARNING - @mbggenerated*
106. *This element is automatically generated by MyBatis Generator, do not modify.*
107. *-->*
108. update person
109. set name = #{name,jdbcType=VARCHAR},
110. age = #{age,jdbcType=INTEGER},
111. address = #{address,jdbcType=VARCHAR}
112. where id = #{id,jdbcType=BIGINT}
113. </update>
115. <select id="findAll" resultMap="BaseResultMap">
116. select
117. <include refid="Base\_Column\_List" />
118. from person
119. </select>
121. <select id="findByPage" resultMap="BaseResultMap">
122. select
123. <include refid="Base\_Column\_List" />
124. from person
125. </select>
126. </mapper>

**DAO层Mapper类**

在Mapper接口上需要加上@Mapper注解，@Mapper注解声明成mybatis Dao层的Bean。也可以在配置类上使用@MapperScan("com.xiaolyuh.domain.mapper")注解声明。

1. **package** com.xiaolyuh.domain.mapper;
3. **import** com.github.pagehelper.Page;
4. **import** com.xiaolyuh.domain.model.Person;
5. **import** org.apache.ibatis.annotations.Mapper;
7. **import** java.util.List;
9. @Mapper*//声明成mybatis Dao层的Bean，也可以在配置类上使用@MapperScan("com.xiaolyuh.domain.mapper")注解声明*
10. **public** **interface** **PersonMapper** {
12. **int** **deleteByPrimaryKey**(Long id);
14. **int** **insert**(Person record);
16. **int** **insertSelective**(Person record);
18. Person **selectByPrimaryKey**(Long id);
20. **int** **updateByPrimaryKeySelective**(Person record);
22. **int** **updateByPrimaryKey**(Person record);
24. */\*\**
25. *\* 获取所有数据*
26. *\* @return*
27. \*/
28. List<Person> **findAll**();
30. */\*\**
31. *\* 分页查询数据*
32. *\* @return*
33. \*/
34. Page<Person> **findByPage**();
35. }

**实体类**

1. **package** com.xiaolyuh.domain.model;
3. **public** **class** **Person** {
5. **private** Long id;
7. */\*\**
8. *\* 名称*
9. *\*/*
10. **private** String name;
12. */\*\**
13. *\* 年龄*
14. *\*/*
15. **private** Integer age;
17. */\*\**
18. *\* 地址*
19. *\*/*
20. **private** String address;
22. **public** Long **getId**() {
23. **return** id;
24. }
26. **public** **void** **setId**(Long id) {
27. **this**.id = id;
28. }
30. **public** String **getName**() {
31. **return** name;
32. }
34. **public** **void** **setName**(String name) {
35. **this**.name = name;
36. }
38. **public** Integer **getAge**() {
39. **return** age;
40. }
42. **public** **void** **setAge**(Integer age) {
43. **this**.age = age;
44. }
46. **public** String **getAddress**() {
47. **return** address;
48. }
50. **public** **void** **setAddress**(String address) {
51. **this**.address = address;
52. }
53. }

**Service层**

**接口**

1. **package** com.xiaolyuh.service;
3. **import** com.github.pagehelper.Page;
4. **import** com.xiaolyuh.domain.model.Person;
6. **import** java.util.List;
8. */\*\**
9. *\* Created by yuhao.wang on 2017/6/19.*
10. *\*/*
11. **public** **interface** **PersonService** {
13. List<Person> **findAll**();
15. */\*\**
16. *\* 分页查询*
17. *\* @param pageNo 页号*
18. *\* @param pageSize 每页显示记录数*
19. *\* @return*
20. \*/
21. Page<Person> **findByPage**(**int** pageNo, **int** pageSize);
23. **void** **insert**(Person person);
24. }

**实现类**

1. **package** com.xiaolyuh.service.impl;
3. **import** com.github.pagehelper.Page;
4. **import** com.github.pagehelper.PageHelper;
5. **import** com.xiaolyuh.domain.mapper.PersonMapper;
6. **import** com.xiaolyuh.domain.model.Person;
7. **import** com.xiaolyuh.service.PersonService;
8. **import** org.springframework.beans.factory.annotation.Autowired;
9. **import** org.springframework.stereotype.Service;
10. **import** org.springframework.transaction.annotation.Transactional;
12. **import** java.util.List;
14. */\*\**
15. *\* Created by yuhao.wang on 2017/6/19.*
16. *\*/*
17. @Service
18. @Transactional(readOnly = **true**)
19. **public** **class** **PersonServiceImpl** **implements** **PersonService** {
21. @Autowired
22. **private** PersonMapper personMapper;
24. @Override
25. **public** List<Person> **findAll**() {
26. **return** personMapper.findAll();
27. }
29. @Override
30. **public** Page<Person> **findByPage**(**int** pageNo, **int** pageSize) {
31. PageHelper.startPage(pageNo, pageSize);
32. **return** personMapper.findByPage();
33. }
35. @Override
36. @Transactional
37. **public** **void** **insert**(Person person) {
38. personMapper.insert(person);
39. }
41. }

**分页的包装类PageInfo**

需要把Page包装成PageInfo对象才能序列化。该插件也默认实现了一个PageInfo

1. **package** com.xiaolyuh.page;
3. **import** java.io.Serializable;
4. **import** java.util.Collection;
5. **import** java.util.List;
7. **import** com.github.pagehelper.Page;
9. */\*\**
10. *\* 对Page<E>结果进行包装*
11. *\* <p/>*
12. *\* 新增分页的多项属性，主要参考:http://bbs.csdn.net/topics/360010907*
13. *\**
14. *\* @author liuzh/abel533/isea533*
15. *\* @version 3.3.0*
16. *\* @since 3.2.2*
17. *\* 项目地址 : http://git.oschina.net/free/Mybatis\_PageHelper*
18. *\*/*
19. @SuppressWarnings({"rawtypes", "unchecked"})
20. **public** **class** **PageInfo**<**T**> **implements** **Serializable** {
21. **private** **static** **final** **long** serialVersionUID = 1L;
22. *//当前页*
23. **private** **int** pageNum;
24. *//每页的数量*
25. **private** **int** pageSize;
26. *//总记录数*
27. **private** **long** total;
28. *//总页数*
29. **private** **int** pages;
30. *//结果集*
31. **private** List<T> list;
32. *//是否为第一页*
33. **private** **boolean** isFirstPage = **false**;
34. *//是否为最后一页*
35. **private** **boolean** isLastPage = **false**;

38. **public** **PageInfo**() {
39. }
41. */\*\**
42. *\* 包装Page对象*
43. *\**
44. *\* @param list*
45. *\*/*
46. **public** **PageInfo**(List<T> list) {
47. **if** (list **instanceof** Page) {
48. Page page = (Page) list;
49. **this**.pageNum = page.getPageNum();
50. **this**.pageSize = page.getPageSize();
52. **this**.pages = page.getPages();
53. **this**.list = page;
54. **this**.total = page.getTotal();
55. } **else** **if** (list **instanceof** Collection) {
56. **this**.pageNum = 1;
57. **this**.pageSize = list.size();
59. **this**.pages = 1;
60. **this**.list = list;
61. **this**.total = list.size();
62. }
63. **if** (list **instanceof** Collection) {
64. *//判断页面边界*
65. judgePageBoudary();
66. }
67. }
69. */\*\**
70. *\* 判定页面边界*
71. *\*/*
72. **private** **void** **judgePageBoudary**() {
73. isFirstPage = pageNum == 1;
74. isLastPage = pageNum == pages;
75. }
77. **public** **int** **getPageNum**() {
78. **return** pageNum;
79. }
81. **public** **void** **setPageNum**(**int** pageNum) {
82. **this**.pageNum = pageNum;
83. }
85. **public** **int** **getPageSize**() {
86. **return** pageSize;
87. }
89. **public** **void** **setPageSize**(**int** pageSize) {
90. **this**.pageSize = pageSize;
91. }
93. **public** **long** **getTotal**() {
94. **return** total;
95. }
97. **public** **void** **setTotal**(**long** total) {
98. **this**.total = total;
99. }
101. **public** **int** **getPages**() {
102. **return** pages;
103. }
105. **public** **void** **setPages**(**int** pages) {
106. **this**.pages = pages;
107. }
109. **public** List<T> **getList**() {
110. **return** list;
111. }
113. **public** **void** **setList**(List<T> list) {
114. **this**.list = list;
115. }
117. **public** **boolean** **isIsFirstPage**() {
118. **return** isFirstPage;
119. }
121. **public** **void** **setIsFirstPage**(**boolean** isFirstPage) {
122. **this**.isFirstPage = isFirstPage;
123. }
125. **public** **boolean** **isIsLastPage**() {
126. **return** isLastPage;
127. }
129. **public** **void** **setIsLastPage**(**boolean** isLastPage) {
130. **this**.isLastPage = isLastPage;
131. }
133. @Override
134. **public** String **toString**() {
135. **final** StringBuffer sb = **new** StringBuffer("PageInfo{");
136. sb.append("pageNum=").append(pageNum);
137. sb.append(", pageSize=").append(pageSize);
138. sb.append(", total=").append(total);
139. sb.append(", pages=").append(pages);
140. sb.append(", list=").append(list);
141. sb.append(", isFirstPage=").append(isFirstPage);
142. sb.append(", isLastPage=").append(isLastPage);
143. sb.append(", navigatepageNums=");
144. sb.append('}');
145. **return** sb.toString();
146. }
147. }

**测试类**

1. **package** com.xiaolyuh;
3. **import** com.github.pagehelper.Page;
4. **import** com.xiaolyuh.domain.model.Person;
5. **import** com.xiaolyuh.page.PageInfo;
6. **import** com.xiaolyuh.service.PersonService;
7. **import** org.junit.Assert;
8. **import** org.junit.Before;
9. **import** org.junit.Test;
10. **import** org.junit.runner.RunWith;
11. **import** com.alibaba.fastjson.JSON;
12. **import** org.slf4j.Logger;
13. **import** org.slf4j.LoggerFactory;
14. **import** org.springframework.beans.factory.annotation.Autowired;
15. **import** org.springframework.boot.test.context.SpringBootTest;
16. **import** org.springframework.test.context.junit4.SpringRunner;
18. **import** java.util.List;
20. @RunWith(SpringRunner.class)
21. @SpringBootTest
22. **public** **class** **PersonMapperTests** {
24. **private** Logger logger = LoggerFactory.getLogger(PersonMapperTests.class);
26. @Autowired
27. **private** PersonService personService;
29. @Before
30. **public** **void** **testInsert**() {
31. Person person = **new** Person();
32. person.setName("测试");
33. person.setAddress("address");
34. person.setAge(10);
35. personService.insert(person);
37. Assert.assertNotNull(person.getId());
38. logger.debug(JSON.toJSONString(person));
39. }
41. @Test
42. **public** **void** **testFindAll**() {
43. List<Person> persons = personService.findAll();
45. Assert.assertNotNull(persons);
46. logger.debug(JSON.toJSONString(persons));
47. }
49. @Test
50. **public** **void** **testFindByPage**() {
51. Page<Person> persons = personService.findByPage(1, 2);
52. *// 需要把Page包装成PageInfo对象才能序列化。该插件也默认实现了一个PageInfo*
53. PageInfo<Person> pageInfo = **new** PageInfo<>(persons);
54. Assert.assertNotNull(persons);
55. logger.debug(pageInfo.toString());
56. logger.debug(JSON.toJSONString(pageInfo));
57. }
59. @Test
60. **public** **void** **testCacheByPage**() {
61. **long** begin = System.currentTimeMillis();
62. List<Person> persons = personService.findAll();
63. **long** ing = System.currentTimeMillis();
64. personService.findAll();
65. **long** end = System.currentTimeMillis();
66. logger.debug("第一次请求时间：" + (ing - begin) + "ms");
67. logger.debug("第二次请求时间:" + (end - ing) + "ms");
69. Assert.assertNotNull(persons);
70. logger.debug(persons.toString());
71. logger.debug(JSON.toJSONString(persons));
72. }
74. }

**源码**

<https://github.com/wyh-spring-ecosystem-student/spring-boot-student/tree/releases>

spring-boot-student-mybatis工程