**实验五 SpringBoot整合Redis点赞功能的简单实现**

**实验学时：2 实验类型：设计**

**一、目的与任务**

目的：学习SpringBoot整合redis简单点赞功能

任务 通过给一篇文章点赞，点赞成功+1,取消点赞-1，数据都放在reids缓存里

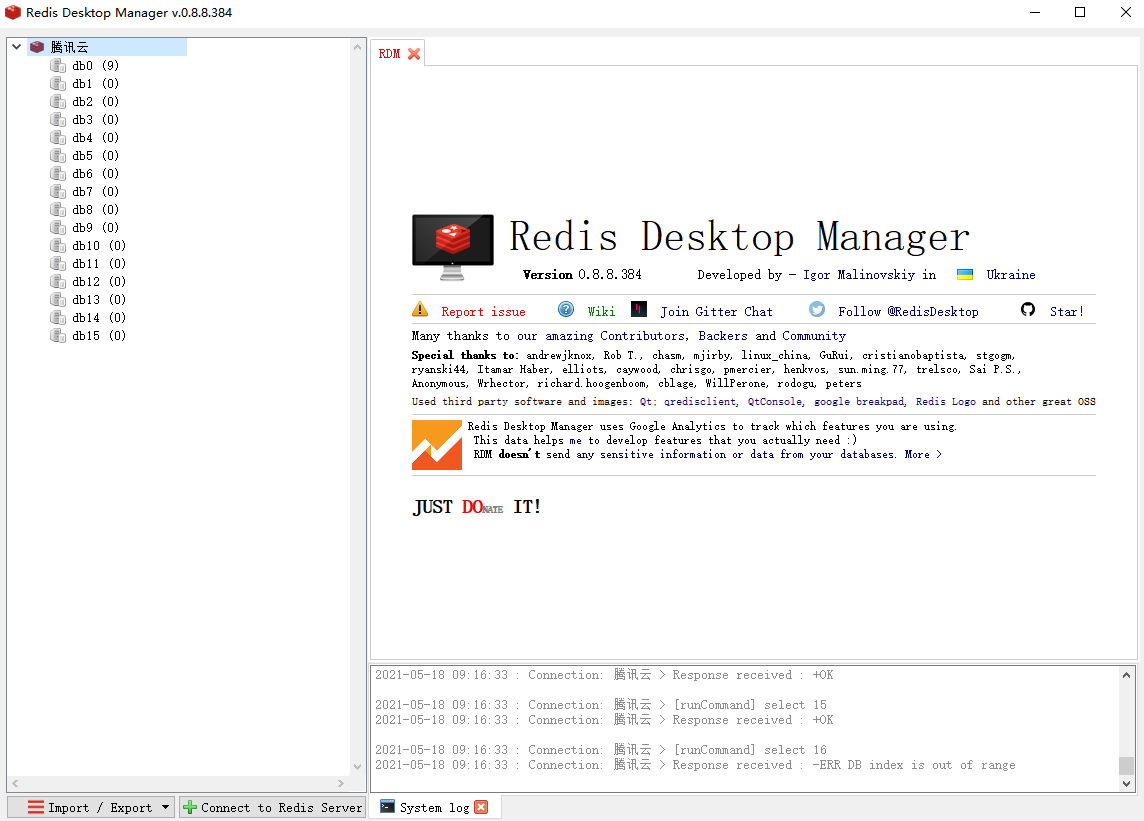
**二、内容、要求与安排方式**

1、实验内容与要求：

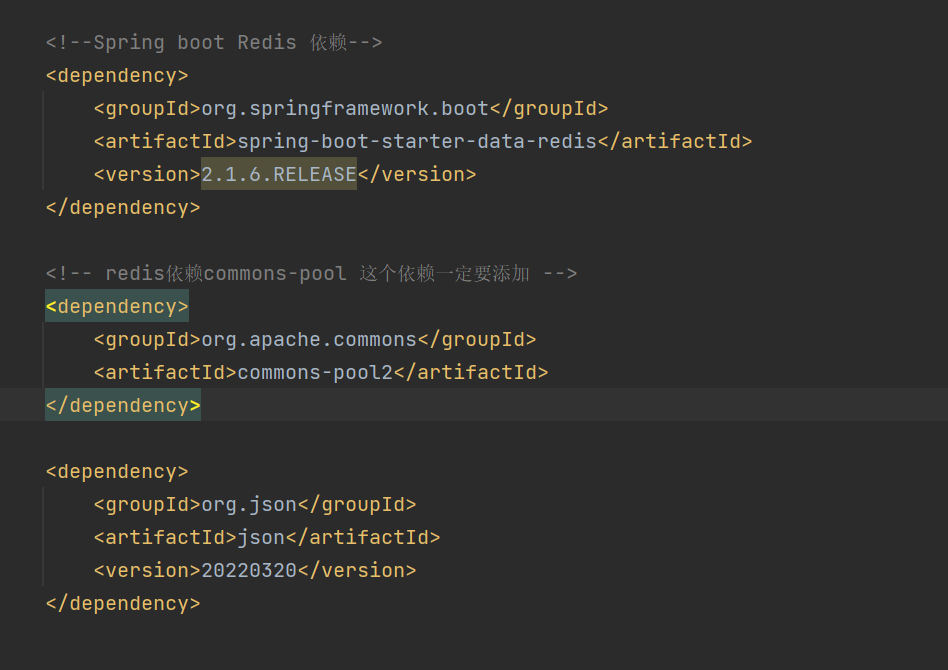
下载redis可以使用windows版也可以是linux版的(如果有虚拟机或者云服务器的同学可以尝试一下)。

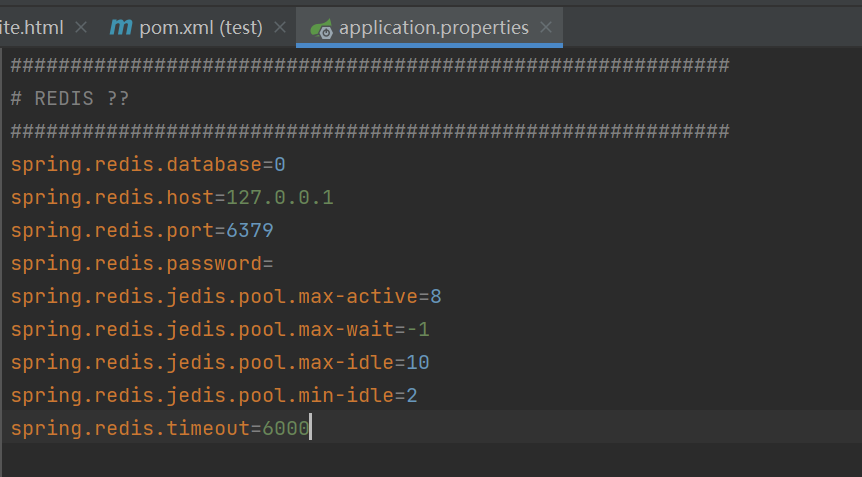
安装redis并掌握redis的基本使用。

可以下载RedisDesktopManager去直接操作redis数据。RedisDesktopManager界面如下：

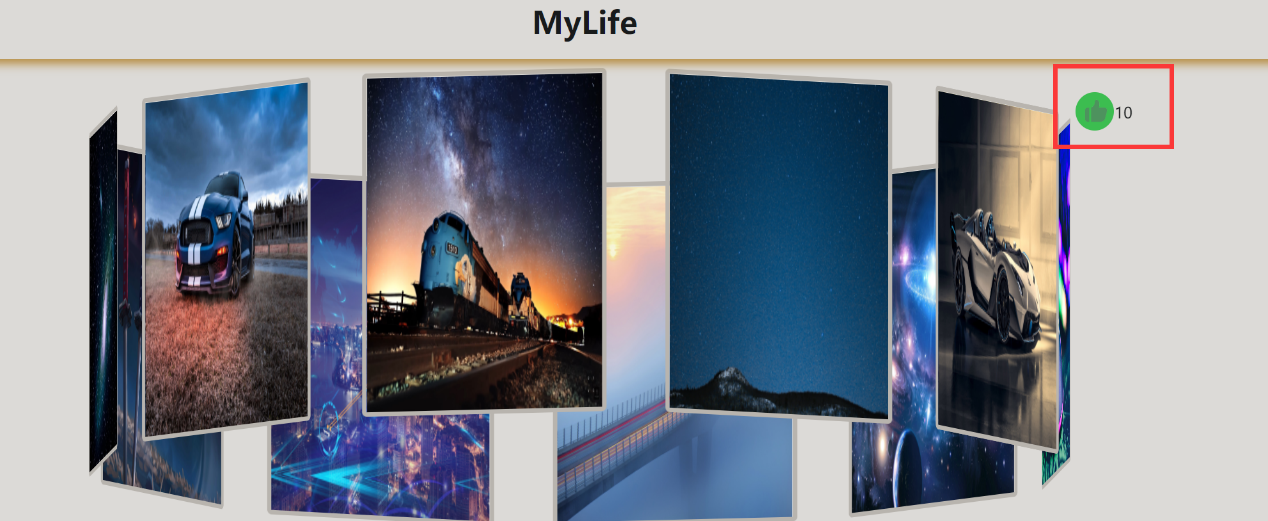


引入Springboot整合redis的依赖，并在配置文件中配置redis相关属性。

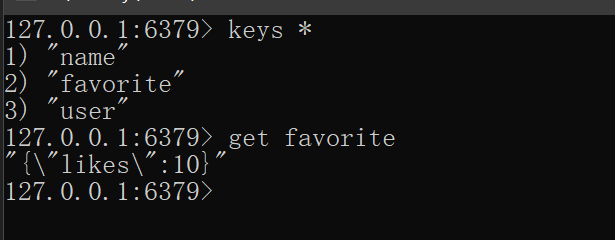




模拟实现文章点赞功能，选用合适的redis数据类型，判断如果当前未点赞，则存入缓存，如果已经存在点赞，那么就删除这条缓存，从而实现前端的点赞功能，根据后端返回1(true)或0(false)去控制。



redis的数据变化



2、实验安排方式：上机编码。

**三、实验代码**

container 层的favoritecontroller.java

1. package test.test.com.exp4.Controller;
3. import org.json.JSONObject;
4. import org.springframework.boot.SpringApplication;
5. import org.springframework.boot.autoconfigure.SpringBootApplication;
6. import org.springframework.data.redis.core.RedisTemplate;
7. import org.springframework.web.bind.annotation.GetMapping;
8. import org.springframework.web.bind.annotation.RequestParam;
9. import org.springframework.web.bind.annotation.RestController;
11. public Object likeStatus(String infoId, String likeUserId) {
12. if (redisTemplate.opsForHash().hasKey(RedisKeyUtils.MAP\_KEY\_USER\_LIKED, RedisKeyUtils.getLikedKey(infoId, likeUserId))) {
13. String o = redisTemplate.opsForHash().get(RedisKeyUtils.MAP\_KEY\_USER\_LIKED, RedisKeyUtils.getLikedKey(infoId, likeUserId)).toString();
14. if ("1".equals(o)) {
15. unLikes(infoId, likeUserId);
16. return LikedStatusEum.UNLIKE;
17. }
18. if ("0".equals(o)) {
19. likes(infoId, likeUserId);
20. return LikedStatusEum.LIKE;
21. }
22. }
23. UserLikes userLikes = userLikesDao.selectOne(new QueryWrapper**<UserLikes>**().eq("info\_id", infoId).eq("like\_user\_id", likeUserId));
24. if (userLikes == null) {
25. UserLikes userLikes1 = new UserLikes();
26. userLikes1.setInfoId(infoId);
27. userLikes1.setLikeUserId(likeUserId);
28. userLikesDao.insert(userLikes1);
29. likes(infoId, likeUserId);
30. return LikedStatusEum.LIKE;
31. }
32. if (userLikes.getStatus() == 1) {
33. unLikes(infoId, likeUserId);
34. return LikedStatusEum.UNLIKE;
35. }
37. if (userLikes.getStatus() == 0) {
38. likes(infoId, likeUserId);
39. return LikedStatusEum.LIKE;
40. }
41. return "";
42. }
43. public void unLikes(String infoId, String likeUserId) {
44. String likedKey = RedisKeyUtils.getLikedKey(infoId, likeUserId);
45. redisTemplate.opsForHash().increment(RedisKeyUtils.MAP\_KEY\_USER\_LIKED\_COUNT, infoId, -1);
46. redisTemplate.opsForHash().delete(RedisKeyUtils.MAP\_KEY\_USER\_LIKED, likedKey);
47. }
48. public List**<UserLikCountDTO>** getLikedCountFromRedis() {
49. Cursor**<Map.Entry<Object**, Object**>>** cursor = redisTemplate.opsForHash().scan(RedisKeyUtils.MAP\_KEY\_USER\_LIKED\_COUNT, ScanOptions.NONE);
50. List**<UserLikCountDTO>** list = new ArrayList**<>**();
51. while (cursor.hasNext()) {
52. Map.Entry**<Object**, Object**>** map = cursor.next();
53. String key = (String) map.getKey();
54. Integer value = (Integer) map.getValue();
55. UserLikCountDTO userLikCountDTO = new UserLikCountDTO(key, value);
56. list.add(userLikCountDTO);
57. redisTemplate.opsForHash().delete(RedisKeyUtils.MAP\_KEY\_USER\_LIKED\_COUNT, key);
58. }
59. return list;
60. }

Application.java

1. package test.test.com.exp4.Controller;
3. import org.json.JSONObject;
4. import org.springframework.boot.SpringApplication;
5. import org.springframework.boot.autoconfigure.SpringBootApplication;
6. import org.springframework.data.redis.core.RedisTemplate;
7. import org.springframework.web.bind.annotation.GetMapping;
8. import org.springframework.web.bind.annotation.RequestParam;
9. import org.springframework.web.bind.annotation.RestController;

12. @SpringBootApplication
13. @RestController
14. **public** **class** Application {
16. **public** **static** **void** main(String[] args) {
17. SpringApplication.run(Application.**class**, args);
18. }
19. @GetMapping("/like")
20. **public** String like(@RequestParam(value = "pageName") String pageName) {
21. RedisTemplate redisTemplate = test.test.com.exp4.Controller.RedisBean.redis;
22. Object jsonstrobj = redisTemplate.opsForValue().get(pageName);
23. JSONObject node = **new** JSONObject(jsonstrobj.toString());
24. node.put("likes",node.getInt("likes")+1);
25. redisTemplate.opsForValue().set(pageName,node.toString());
26. **return** "Done";
27. }
28. @GetMapping("/unlike")
29. **public** String unlike(@RequestParam(value = "pageName") String pageName) {
30. RedisTemplate redisTemplate = test.test.com.exp4.Controller.RedisBean.redis;
31. Object jsonstrobj = redisTemplate.opsForValue().get(pageName);
32. JSONObject node = **new** JSONObject(jsonstrobj.toString());
33. node.put("likes",node.getInt("likes")-1);
34. redisTemplate.opsForValue().set(pageName,node.toString());
35. **return** "Done";
36. }
37. @GetMapping("/SeeLike")
38. **public** **int** SeeLike(@RequestParam(value = "pageName") String pageName) {
39. RedisTemplate redisTemplate = test.test.com.exp4.Controller.RedisBean.redis;
40. Object jsonstrobj = redisTemplate.opsForValue().get(pageName);
41. **if**(jsonstrobj == null){
42. JSONObject newJson = **new** JSONObject();
43. newJson.put("likes",0);
44. redisTemplate.opsForValue().set(pageName,newJson.toString());
45. **return** 0;
46. }
47. JSONObject node = **new** JSONObject(jsonstrobj.toString());
48. System.out.println(node.get("likes"));
49. **return** node.getInt("likes");
50. }
51. }

点赞页面 favorite.html

1. <!DOCTYPE html**>**
2. **<html** lang="en"**>**
3. **<head>**
4. **<meta** charset="UTF-8"**>**
5. **<link** rel="stylesheet" href="/css/favorite.css"**>**
6. **<title>**mylife**</title>**
7. **</head>**
8. **<body>**
9. **<a** href="http://localhost:8080/index"**>**返回我的博客**</a>**
10. **<div** id="mylife\_header"**>**
11. **<div** class="mylife\_h\_center"**>**
12. **<div** class="mylife\_h\_left"**>**
13. **<h3** style="margin-top: 50px; text-align: center; font-size: xx-large;"**>**MyLife**</h3>**
14. **</div>**
15. **</div>**
16. **</div>**
18. **<div** id="div2"**>**
19. **<div** id="photo\_box"**>**
20. **<img** src="/images/001.jpg"**>**
21. **<img** src="/images/002.jpg"**>**
22. **<img** src="/images/003.jpg"**>**
23. **<img** src="/images/004.jpg"**>**
24. **<img** src="/images/005.jpg"**>**
25. **<img** src="/images/006.jpg"**>**
26. **<img** src="/images/007.jpg"**>**
27. **<img** src="/images/008.jpg"**>**
28. **<img** src="/images/009.jpg"**>**
29. **<img** src="/images/010.jpg"**>**
30. **<img** src="/images/011.jpg"**>**
31. **</div>**
32. **<div** style=" position:absolute;width:100px;height:50px;left:80%;top:20%;"**>**
33. **<div** id="likeDiv" class="like unliked"**></div>**
34. **<img** id="likeIcon" onclick="bt\_like()" class="unlikedIcon"
35. src="/images/favoriteicon.png"
36. style="margin-left: 4px;width:32px;height:32px;vertical-align: middle;"**>**
37. **<p** id="likeCount" style="display:inline-block;vertical-align: middle;"**>**---**</p>**
38. **</div>**
39. **</div>**
40. **</body>**
41. **<script** src="/js/LikeButton.js"**></script>**
42. **<script>**
43. var currentPage = "favorite"
45. function bt\_like() {
46. onClickLike(currentPage);
47. }
48. window.onload=function(){
49. var collection = document.getElementsByClassName("box");
50. var i;
51. for(i=0;i**<collection.length**;i++){
52. collection[i].classList.add("box"+(i+1));
53. }
54. LoadLikeCount(currentPage);
55. }

58. **</script>**
59. **</html>**

redis配置 application.properties

1. ############################################################
2. # REDIS ??
3. ############################################################
4. spring.redis.database=0
5. spring.redis.host=127.0.0.1
6. spring.redis.port=6379
7. springspring.redis.password=
8. spring.redis.jedis.pool.max-active=8
9. spring.redis.jedis.pool.max-wait=-1
10. spring.redis.jedis.pool.max-idle=10
11. spring.redis.jedis.pool.min-idle=2
12. spring.redis.timeout=6000