链滴

RESTful token 登陆 (基于 Spring、Redis)

作者：danbai225

原文链接：https://ld246.com/article/1574326748752

来源网站：链滴

为什么使用token

网站服务端在存储用户登陆信息使用session+cookie.在进行其他客户端通信中使用cookie的话操作繁琐,所以选择token这种简易自定义的标识符来进行身份确认.

通信流程

1.客服端通过post请求服务端带上用户名密码进行认证,服务端认证通过返回token,并将token存在redi数据库中key为用户名.

2.客服端在需要权限的请求中带上token,服务端通过token即可判断是那个用户.

3.用户登陆、退出登陆、长时间未操作都会使token改变或失效

代码实战

Token实体类

```
package com.danbai.ys.entity;

import java.io.Serializable;

/**
 * @author danbai
 * @date 2019-11-21 14:21
 */
public class Token implements Serializable {
    private String username;
    private String token;
    public static final String TOKEN="token_";
```

```
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getToken() {
        return token;
    }

    public void setToken(String token) {
        this.token = token;
    }

    public Token(String username, String token) {
        this.username = username;
        this.token = token;
    }

    @Override
    public String toString() {
        return "Token{" +
                "username='" + username + '\'' +
                ", token='" + token + '\'' +
                '}';
    }

    public Token() {
    }
}
```

service接口和实现类

```
    /**
     * 关联用户和token
     * @param username 用户名
     * @return
     */
    Token createToken(String username);

    /**
     * 检测token有效性
     * @param token
     * @return
     */
    boolean checkToken(Token token);

    /**
     * 删除token
     * @param username
```

```
     */
    void deleteToken (String username);
    /**
     * token登陆
     * @param user
     * @return
     */
    Token login(User user);
----------------------------------------------------实现-----------------------------------
 @Override
    public Token createToken(String username) {
        String tokenid= UUID.randomUUID().toString().replace ("-", "");
        Token token=new Token(username,tokenid);
        redisTemplate.opsForValue().set(Token.TOKEN+username,token,7,TimeUnit.DAYS);
        return token;
    }

    @Override
    public boolean checkToken(Token token) {
        if (token == null|token.getUsername()==null|token.getToken()==null|token.getUsernam
().length()<1|token.getToken().length()<1) {
            return false;
        }
        Token rtoken = (Token) redisTemplate.opsForValue().get(Token.TOKEN+token.getUsern
me());
        if(rtoken.getToken().equals(token.getToken())){
            // 如果验证成功，说明此用户进行了一次有效操作，延长 token 的过期时间
            redisTemplate.expire(Token.TOKEN+token.getUsername (),7, TimeUnit.DAYS);
            return true;
        }
        return false;
    }

    @Override
    public void deleteToken(String username) {
        redisTemplate.delete(Token.TOKEN+username);
    }

    @Override
    public Token login(User user) {
        if (user != null) {
            User user1 = new User();
            user1.setUsername(user.getUsername());
            User user2 = getUser(user1);
            if (user2 != null) {
                if (DigestUtils.md5DigestAsHex(user.getPassword().getBytes()).equals(user2.getPass
ord())) {
                    return createToken(user.getUsername());                }
            }
        }
        return null;
    }
```