总结 test

Time：2019-5-8

遇到的问题

```
/*
1，WEB-INF下的jsp界面来访问外部静态文件资源，比如js，CSS等等
    解决方案：
    首先在Spring.XML中配置静态资源的配置，配置如下：
    <!-- 静态资源的处理 -->
    <mvc:default-servlet-handler />
    然后再引用资源的href中使用${pageContext.request.contextPath}/资源路径即可。
2，WEB-INF中jsp界面的跳转
    通过创建一个Controller来解决该问题
*/
```

完成的功能

```
/*
1，初步设计了本项目的开发过程，以及数据库表(User表)的设计
2，完成了SSM框架的搭建
3，登录界面的设计，注册界面的设计
4，完成了用户的注册功能
*/
```

Time：2019-5-9

完成的功能

```
/*
1，在注册界面加入了线程池，来提高并发量，线程池采用的是CachedThreadPool，它是一个大小无界的线程池，适合于执行很多短期异步任务的小程序。MaxeimumPoolSize被设置程Integer.MAX_VALUE，corePool为0.
2，整合了shiro与SSM框架，主要是在WEB.XML中配置shiro拦截器，以及在xml中配置shiro的一些拦截配置，管理器，界面的用户权限。
```

Time:2019-5-10

遇见的问题

```
/*
1，当shiro跳转loginurl的时候，会自动在url后面加上sessionID。
    解决办法：
    原因是ShiroHttpServletResponse中的toEncoded函数中处理url时会添加上sesionID这个信息
    重写ShiroHttpServletResponse中的toEncoded和encodeRedirectURL方法。
```

完成的功能

```
/*
1，实现shiroRealm对用户登录信息表单中数据的获取。
2，完成了shiro对一些界面权限的设置。
```

Time:2019-5-11

记录

```
/*
1，jsp中引入另外一个jsp页面
    <%@ include file="url" %>
```

完成的功能

```
/*
1,采用MD5算法对用户的密码加密*/
String hashAlgorithmName = "MD5";  //加密算法
String credentials = password;     //用户输入的密码
int hashIterations = 1024;         //加密的次数
Object Password = new SimpleHash(hashAlgorithmName, credentials, null,
hashIterations);//进行加密
/*
2，当数据库中进行MD5加密之后，那么shiro在进行比对时，要将凭据的匹配credentialsMatcher方式设置成MD5
方式，并且还要将加密的次数设置成1024(是为了与数据库中MD5加密的密码加密次数一致)*/

<bean id="jdbcRealm" class="Shiro.Realm.ShiroRealm">
        <property name="credentialsMatcher">
            <bean class="org.apache.shiro.authc.credential.HashedCredentialsMatcher">
                <property name="hashAlgorithmName" value="MD5"></property>
                <property name="hashIterations" value="1024"></property>
            </bean>
        </property>
    </bean>
/*
上面的方式当两个用户的密码一样的时候，MD5加密之后的字符串是一样的，所以为了保证每个用户的密码加密之后的
相异性，要用到MD5盐值加密
*/
//在往数据库中对密码加密的时候，设置盐值（我设置的时用户的number，要求时盐值要唯一标识用户）
String hashAlgorithmName = "MD5";
String credentials = password;
int hashIterations = 1024;
Object salt = number;  //设置盐值
Object Password = new SimpleHash(hashAlgorithmName, credentials, salt, hashIterations);
//realm中也要对盐值进行相应的设置
Object principal = number;
//2，密码
```

```
String hashedCredentials = result.getBloguserpassword();
//3,当前Realm的名字
String realmName = getName();
//设置盐值
ByteSource credentialsSalt = ByteSource.Util.bytes(number);
SimpleAuthenticationInfo info = new SimpleAuthenticationInfo(principal,
hashedCredentials, credentialsSalt, realmName);
```

Time：2019-5-12

完成的功能

```
/*
1，实现了多个Realm(多个Realm的原因是因为可能数据库不仅仅有Mysql，有可能还会有其他类型的数据库，那么此时加入其他类型的数据库存储密码的方式不是MD5，而是SHA1等其他加密方式，那么此时一个Realm是不能够满足我们的需求的)
2，多个默认认证方式是AtLeastOneSuccessfulStrategy:只要有一个验证成功即可将返回所有Realm验证成功的结果。另外两种方式：FirstSuccessfulStrategy:只返回第一个验证成功的Realm AllSuccessfulStrategy:所有Realm验证成功才算成功，并返回所有的Realm信息*/
<bean id="authenticator"
    class="org.apache.shiro.authc.pam.ModularRealmAuthenticator">
        <property name="realms">
            <!-- 认证器 -->
        <!-- 决定了多Realm认证的时候所执行的顺序 -->
                    <list>
                        <ref bean="jdbcRealm" />
                        <ref bean="SecondRealm" />
                    </list>
        </property>
//在安全管理器中配置
<bean id="securityManager"
        class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">
        <property name="cacheManager" ref="cacheManager" />

        <property name="sessionMode" value="native" />
        <property name="authenticator" ref="authenticator"></property>
</bean>
```

总结(整合shiro)

1，本项目用到了shiro的认证，授权没有用到，因为本项目是一个博客项目，大部分界面的访问权限都是user，自己觉得这个授权的特性可以不用。拦截器主要使用authc，anon。

2，本项目还使用了shiro的盐值加密，采用MD5算法，加密次数为1024次，盐值采用的是用户的number。

上面的内容是该项目的登录与注册功能。

++

Time：2019-5-19

完成的功能

```
/*
1，用户主页的设计
2，用户个人信息在个人主页的渲染
3，个人信息的修改功能
```

Time:2019-5-20

问题

```
/*
1，引入Ueditor时controller.jsp报错，解决办法，将utf8-jsp中的lib中的jar包添加到build path中即可。
2,引入ueditor时，后端配置错误，无法完成上传图片的功能。解决办法：
    第一步：
    config.json
    "imageUrlPrefix": "http://localhost:8082/Blog-SSM/",  图片访问路径前缀

    第二步
    将utf8-jsp中的lib中的jar包添加到WEB-INF的目录下的lib中。
```

完成的功能

```
/*
1，完成了博客表的创建，设置表中UserNumber为外键。
2，完成了博客的内容的创建
```

Time:2019-5-22

完成的功能

```
/*
1,用户主页博客界面的设计(参考了博客园的主页)
2,完成了粉丝表的创建
3,引入了mybatis的分页插件PageHelper。实现了对用户博客主页界面的分页功能。
4，引入PageHelper注意事项：配置SqlMapConfig.xml中配置PageHelper，然后还要引入jsqlparser-1.2jar包。
分页配置如下：
/*创建查询条件*/
criteria.andBlogusernumberEqualTo(UserNumber);
//分页处理
PageHelper.startPage(page, 10);
List<TbBlogartical> list = mapper.selectByExample(example);
```

```
PageInfo<TbBlogartical> pageinfo = new PageInfo<TbBlogartical>(list);
return pageinfo;
```

Time:2019-5-23

完成的功能

```
/*
1，博客的编辑功能
2，博客的删除功能
3，查看博客功能
4，博客阅读量的统计：采用了ConsurrentHasHmap，key值为blogid，value值为blogid对应的阅读数
5，采用定时功能，将博客阅读量写入到数据库中：
    遇到的问题：就是自己在写mybatis中的sqlmap中的xml时，需要往xml中传入两个参数，具体操作如下：
    在TbBlogarticalMapper中添加接口函数
    void MyUpdate(@Param("sum") Integer blogtraffic,@Param("id")Integer blogid);
    在xml中添加xml语句，在接口中使用@Param来声明传入xml中的参数名称
    <update id="MyUpdate">
        UPDATE tb_blogartical SET BlogTraffic=#{sum} WHERE BlogId=#{id}
    </update>
```

Time:2019-5-24

完成的功能

```
/*
1，创建了评论表
2，完成了用户对博客的评论功能
3，完成了博主对评论者的回复功能
```

++

整合solr

遇到的问题

```
/*
1，solr的安装：https://blog.csdn.net/qq_18252605/article/details/80336429
2，在连接solr服务器端的时候，要指明core的名称：http://localhost:8983/solr/mycore(mycore是自己创建的core)
```

完成的任务

```
/*
1，完成了对博客文章的搜索功能，并且还对搜索关键字加色处理。
```

++

整合Redis

Redis的用途：

- 缓存home.jsp中浏览量较多的博客。
- 可能感兴趣的人(通过redis的set数据类型来进行处理)

Time：2019-5-26

遇到的问题

```
/*
1，有这样一个业务，就是当用户写了一篇博客之后，那么此时需要去更新solr索引，但是在进行博客插入的时候，博客的id是递增的，不需要程序员去设置，那么此时就要从mybatis中获取到插入博客信息之后，要获取到博客的id。
    解决办法：修改mybatis的逆向工程生成的代码：
    第一种：在insert标签中添加
    useGeneratedKeys="true"  keyProperty="blogid"
    注意keyProperty是xml中输入对象的的属性，即pojo的属性，而不是table中的属性。
    第二种：
    在insert标签中添加
    <selectKey keyProperty="对应的pojo的属性" order="AFTER"
resultType="jaca.lang.Integer">
        SELECT LAST_INSERT_ID()
    </selectKey>
    我采用的是第一种方法。
```

完成的任务

```
/*
1,配置redis，并完成了redis对热门博客的缓存。也就是redis用途中第一点已经完成。
2，完成了粉丝表，关注表的查询.
```

Time:2019-5-27

遇到的问题

```
/*
1，在取redis中set对象交集的时候，遇到了(error) CROSSSLOT Keys in request don't hash to the same slot异常，原因是，在对redis集群中set对象进行取交集或者并集的时候，集合对象必须要在同一个槽中。
    解决办法：
    加上{}这个前缀即可，比如原来的set的名字叫2，那么此时就要变成{usernumber}:2
2,在jsp界面调用el表达式去调用对象的属性，一直报类的属性未找到。
    解决办法：就是将类中的属性首字母要小写
```

完成的功能

```
/*
1，实现了对用户的关注，取消关注的功能。
2,实现了对用户推荐博主的功能。采用redis的set集合。
```

++

Jemeter测试

```
/*
```