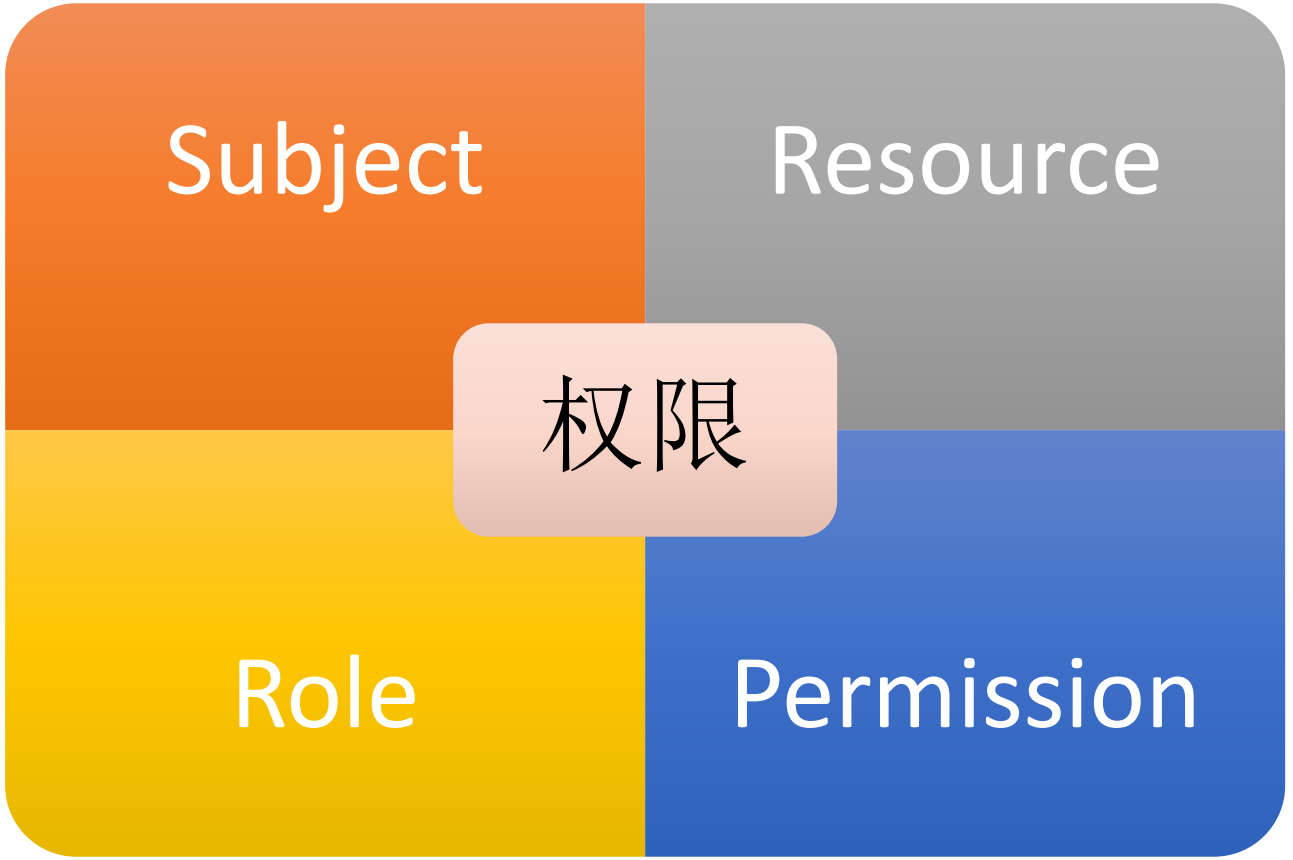
Apache Shiro

凯盛软件

流行的权限框架

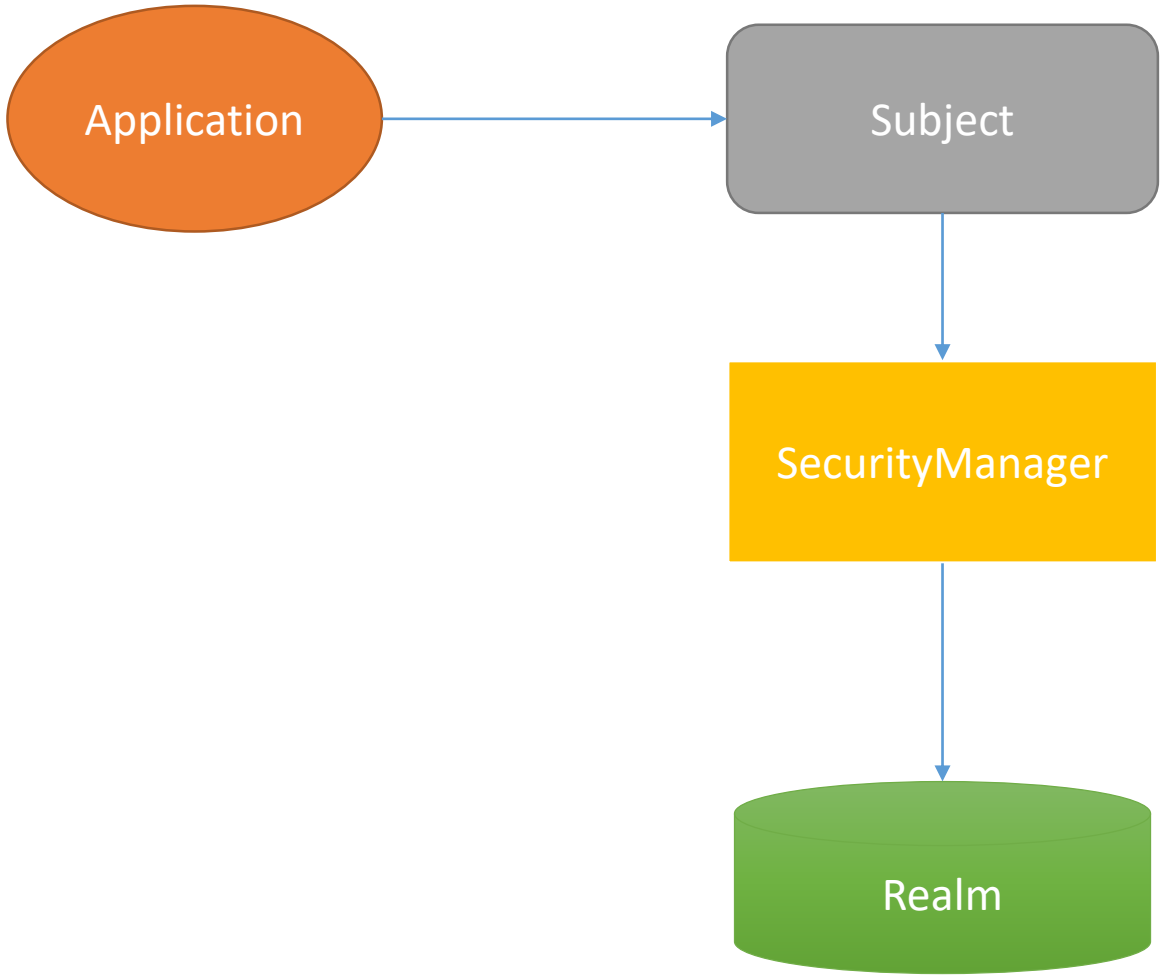
权限的组成部分

凯盛软件



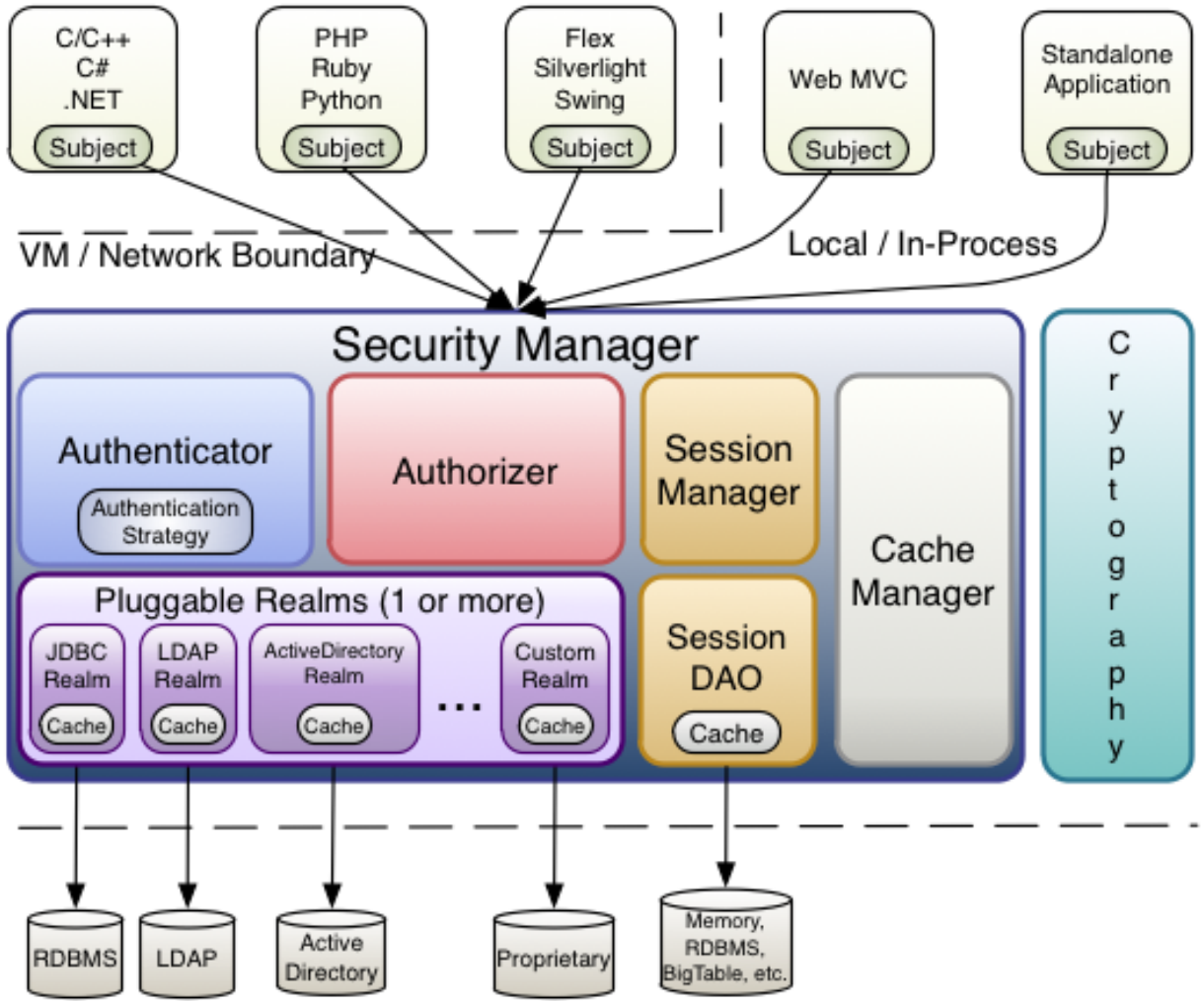
Shiro工作原理

凯盛软件



Shiro架构

凯盛软件



First Shiro App

凯盛软件

- 添加Maven依赖

```
<dependency>
    <groupId>org.apache.shiro</groupId>
    <artifactId>shiro-all</artifactId>
    <version>1.3.2</version>
</dependency>
<dependency>
    <groupId>commons-logging</groupId>
    <artifactId>commons-logging</artifactId>
    <version>1.2</version>
</dependency>
```

- 在classPath中添加shiro.ini文件

```
[users]
tom=123123
jack=00000
```

http://shiro.apache.org/tutorial.html

```
//1. 读取classpath中的shiro.ini配置文件，并创建securityManagerFactory对象
Factory<SecurityManager> securityManagerFactory = new IniSecurityManagerFactory("classpath:shiro.ini");
//2. 获取SecurityManager
SecurityManager securityManager = securityManagerFactory.getInstance();
//3. 设置SecurityManager(仅设置一次)
SecurityUtils.setSecurityManager(securityManager);

//4.获取当前登录的对象
Subject subject = SecurityUtils.getSubject();
```

```
//5.根据账号和密码进行登录
UsernamePasswordToken token = new UsernamePasswordToken("tom","0000");
try {
    //6. 登录
    subject.login(token);
} catch (UnknownAccountException ex) {
    ex.printStackTrace();
    System.out.println("找不到该账号");
} catch (LockedAccountException ex) {
    System.out.println("账号被冻结异常");
} catch (IncorrectCredentialsException ex) {
    System.out.println("账号或密码错误异常");
} catch (AuthenticationException ex) {
    System.out.println("认证异常");
}
//7. 安全退出
subject.logout();
```

自定义Realm

凯盛软件

```
public class MyRealm implements Realm {
    public String getName() {
        return "my-realm";
    }

    public boolean supports(AuthenticationToken authenticationToken) {
        return authenticationToken instanceof UsernamePasswordToken;
    }
```

```
    * 认证的方法
    public AuthenticationInfo getAuthenticationInfo(AuthenticationToken authenticationToken)
                                                                    throws AuthenticationException {
        UsernamePasswordToken token = (UsernamePasswordToken) authenticationToken;
        //获取账号
        String name = token.getUsername();
        //获取密码
        String password = new String(token.getPassword());
        if(!"tom".equals(name)) {
            throw new UnknownAccountException("不是Tom");
        }
        if(!"000000".equals(password)) {
            throw new IncorrectCredentialsException("账号或密码错误");
        }
        return new SimpleAuthenticationInfo(name,password,getName());
    }
}
```

- 在classpath中创建shiro-realm.ini文件

```
myRealm=com.kaishengit.shiro.realm.MyRealm
securityManager.realms=$myRealm
```

- 使用shiro-realm.ini文件

```
//1. 读取classpath中的shiro.ini配置文件，并创建securityManagerFactory对象
Factory<SecurityManager> securityManagerFactory = new IniSecurityManagerFactory("classpath:shiro-realm.ini");
```

定义角色和权限

凯盛软件

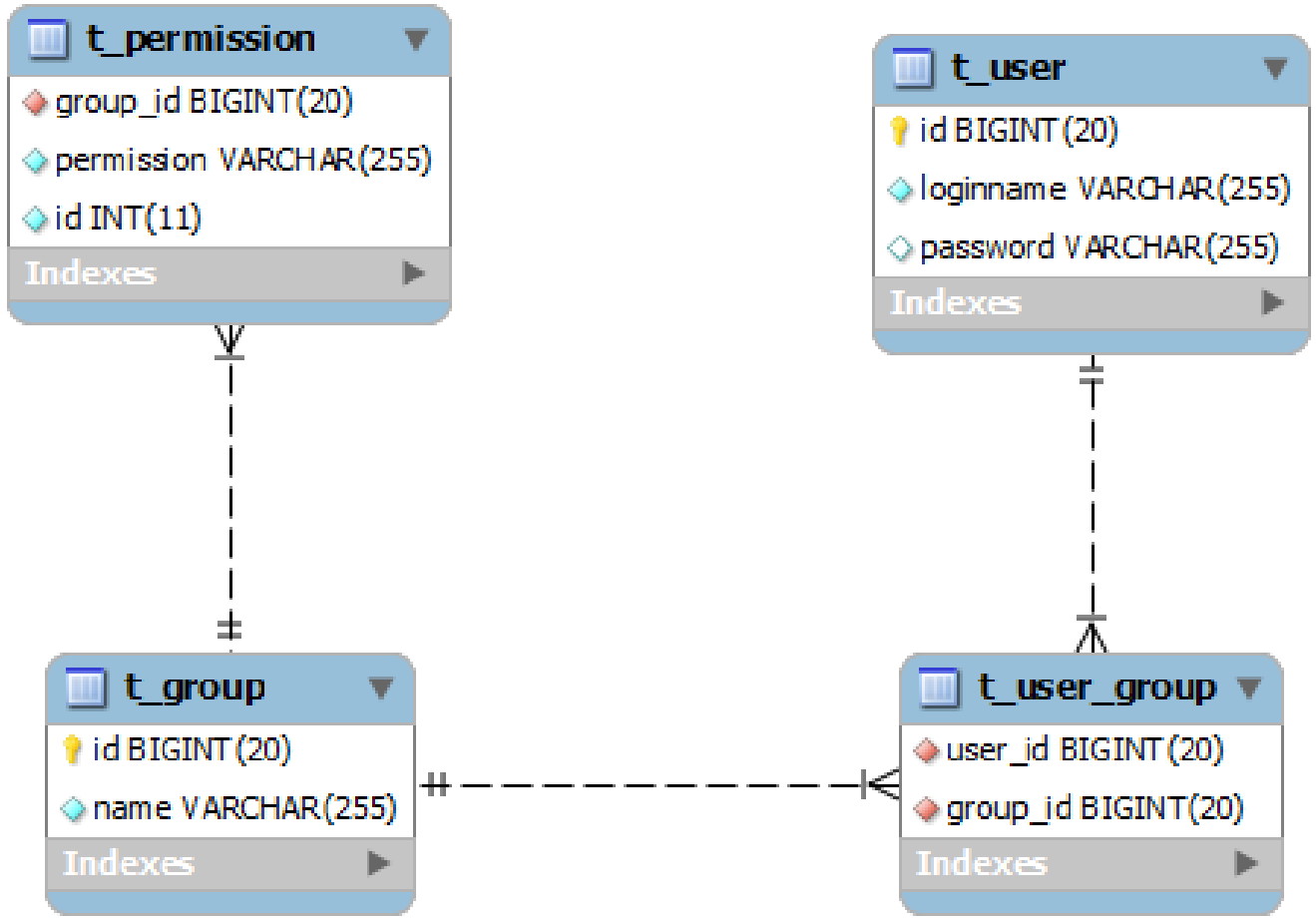
- 在classpath中定义shiro-roles.ini

```
#username=password,roleName...
[users]
tom=123123,cto,admin
jack=000000,admin
#roleName=permissionName...
[roles]
admin=user:add,user:update,user:delete
cto=user:query,user:add
```

- 判断用户是否有某个角色
    - subject.hasRole()
    - subject.hasAllRole()
    - subject.hasRoles()
    - subject.checkRole() 如果没有，则抛异常
- 判断用户是否有某个权限
    - subject.isPermitted()
    - subject.checkPermission() 如果没有，则抛异常

数据库设计

凯盛软件


t_permission
group_id BIGINT(20)
permission VARCHAR(255)
id INT(11)
Indexes
t_user
id BIGINT(20)
loginname VARCHAR(255)
password VARCHAR(255)
Indexes
t_group
id BIGINT(20)
name VARCHAR(255)
Indexes
t_user_group
user_id BIGINT(20)
group_id BIGINT(20)
Indexes

使用步骤

1. 导入jar
2. 配置web.xml
3. 建立dbRelm
4. 在Spring中配置

web.xml

凯盛软件

```
<!--shiro-->
<filter>
    <filter-name>shiro</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
    <init-param>
        <param-name>targetFilterLifecycle</param-name>
        <param-value>true</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>shiro</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

ShiroDbRealm

凯盛软件

```
@Named
public class ShiroDbRealm extends AuthorizingRealm{

        @Inject
        private UserService userService;
```

```
/**
 * 权限认证方法
 */
@Override
protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection pCollection) {

        String loginName = (String) pCollection.fromRealm(getName()).iterator().next();
        User user = userService.findByName(loginName);
        if(user != null) {
                SimpleAuthorizationInfo info = new SimpleAuthorizationInfo();
                info.setRoles(user.getRoleNamesSet());
                for(Role r : user.getRoleList()) {
                        info.addStringPermissions(r.getPermissionNameList());
                }
                return info;
        }

        return null;
}
```

```
    /**
     * 登录认证方法
     */
    @Override
    protected AuthenticationInfo doGetAuthenticationInfo(
                        AuthenticationToken authenticationToken) throws AuthenticationException {
        UsernamePasswordToken token = (UsernamePasswordToken) authenticationToken;
        User user = userService.findByName(token.getUsername());
        if(user != null) {
            return new SimpleAuthenticationInfo(user.getUsername(),user.getPassword(),getName());
        }
        return null;
    }
}
```

applicationContext-Shiro.xml

凯盛软件

```
<bean id="securityManager" class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">
        <property name="realm" ref="shiroDbRealm"/>
        <property name="cacheManager" ref="cacheManager"></property>
</bean>
```

```
<bean id="shiro" class="org.apache.shiro.spring.web.ShiroFilterFactoryBean">
    <property name="securityManager" ref="securityManager"/>
    <property name="loginUrl" value="/"/>
    <property name="successUrl" value="/home"/>
    <property name="unauthorizedUrl" value="/403"/>
    <property name="filters">
        <map>
            <entry key="authc" value-ref="formAuthenticationFilter"></entry>
        </map>
    </property>
    <property name="filterChainDefinitions">
        <value>
            /static/** = anon
            /** = authc
        </value>
    </property>
</bean>
```

```
<bean id="cacheManager"
            class="org.apache.shiro.cache.MemoryConstrainedCacheManager" />

<bean id="lifecycleBeanPostProcessor"
            class="org.apache.shiro.spring.LifecycleBeanPostProcessor" />


<!--form认证过滤器-->
<bean id="formAuthenticationFilter" class="org.apache.shiro.web.filter.authc.FormAuthenticationFilter">
    <!--value值为表单中账号的name属性值-->
    <property name="usernameParam" value="userName"/>
    <!--value值为表单中密码的name属性值-->
    <property name="passwordParam" value="password"/>
    <!--登录的URL-->
    <property name="loginUrl" value="/"/>
</bean>
```

常用的Filter

凯盛软件

Filter Name	Class
anon	org.apache.shiro.web.filter.authc.AnonymousFilter
authc	org.apache.shiro.web.filter.authc.FormAuthenticationFilter
authcBasic	org.apache.shiro.web.filter.authc.BasicHttpAuthenticationFilter
logout	org.apache.shiro.web.filter.authc.LogoutFilter
perms	org.apache.shiro.web.filter.authz.PermissionsAuthorizationFilter
port	org.apache.shiro.web.filter.authz.PortFilter
rest	org.apache.shiro.web.filter.authz.HttpMethodPermissionFilter
roles	org.apache.shiro.web.filter.authz.RolesAuthorizationFilter
ssl	org.apache.shiro.web.filter.authz.SslFilter
user	org.apache.shiro.web.filter.authc.UserFilter

用户登录

凯盛软件

```
@RequestMapping(value="/login",method=RequestMethod.POST)
public String login(User user,RedirectAttributes redirectAttributes) {
        try {
                SecurityUtils.getSubject().login(new UsernamePasswordToken(user.getUsername(),
user.getPassword()));
        } catch (AuthenticationException e) {
                e.printStackTrace();
                redirectAttributes.addFlashAttribute("message","用户名或密码错误");
                return "redirect:/login";
        }
        return "redirect:/home";
}
```

用户退出

凯盛软件

```
@RequestMapping(value="/logout",method=RequestMethod.GET)
public String logout(RedirectAttributes redirectAttributes) {
        redirectAttributes.addFlashAttribute("message", "你已安全退出");
        SecurityUtils.getSubject().logout();
        return "redirect:/login";
}
```

将当前登录的用户放入Session

凯盛软件

- 在登录方法中

```
//获取当前登录的对象
User user = (User) subject.getPrincipal();
//将当前登录的用户放入session
Session session = subject.getSession();
session.setAttribute("curr_user",user);
```

- 在JSP中获取当前登录对象

```
<%@ taglib prefix="shiro" uri="http://shiro.apache.org/tags" %>
Hello, <shiro:principal property="name"/>
```

rememberMe

凯盛软件

在UserNamePasswordToken传入指定的参数，Shiro可以将登录信息以密文的形式写到Cookie中

```
@PostMapping("/")
public String login(String mobile, String password,boolean rememberMe, RedirectAttributes redirectAttributes) {


    Subject subject = SecurityUtils.getSubject();
    UsernamePasswordToken token = new UsernamePasswordToken(mobile,
            new Md5Hash(password).toString(),rememberMe);
```

设置cookie相关信息

```
<!--rememberMeCookie-->
<bean id="rememberMeCookie" class="org.apache.shiro.web.servlet.SimpleCookie">
    <property name="maxAge" value="604800"/>
    <property name="httpOnly" value="true"/>
    <property name="name" value="rememberMe"/>
</bean>
```

```
<!--rememberMeManager-->
<bean id="rememberMeManager" class="org.apache.shiro.web.mgt.CookieRememberMeManager">
    <property name="cookie" ref="rememberMeCookie"/>
</bean>

<!--securityManager-->
<bean id="securityManager" class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">
    <property name="realm" ref="myRealm"/>
    <property name="cacheManager" ref="cacheManager"/>
    <property name="rememberMeManager" ref="rememberMeManager"/>
</bean>
```

- 登录对象的状态
  - 被认证
  - 被记住

```
Subject subject = ShiroUtil.getSubject();
System.out.println("isAuthenticated? " + subject.isAuthenticated());
System.out.println("isRemembered? " + subject.isRemembered());
```

- 在filter中配置url可以被认证用户和被记住用户访问

```
<!--定义过滤规则-->
<property name="filterChainDefinitions">
    <value>
        /static/** = anon
        /info/** = roles[基本信息]
        /** = user
    </value>
</property>
```

跳转到登录前的请求页面

凯盛软件

```
@PostMapping("/")
public String login(String mobile, String password,
                    RedirectAttributes redirectAttributes, HttpServletRequest request) {
    try {
        …

        SavedRequest savedRequest = WebUtils.getSavedRequest(request);
        String url = "/home";
        if(savedRequest != null) {
            url = savedRequest.getRequestUrl();
        }

        return "redirect:"+url;
    } catch (AuthenticationException ex) {
        redirectAttributes.addFlashAttribute("message","账号或密码错误");
        return "redirect:/";
    }
}
```

filterChainDefinitions支持多Roles

凯盛软件

- 自定义Roles过滤器

```
public class MyRolesFilter extends RolesAuthorizationFilter {

    @Override
    public boolean isAccessAllowed(ServletRequest request, ServletResponse response, Object mappedValue)
                                                                                    throws IOException {
        //获取当前登录对象
        Subject subject = getSubject(request,response);
        //获取配置文件中传入的角色列表
        String[] roles = (String[]) mappedValue;
        //如果角色列表为null或为空则可以访问
        if(roles == null || roles.length == 0) {
            return true;
        }
        for (String role : roles) {
            //当前登录对象有任意一个角色就可以访问
           if(subject.hasRole(role)) {
                return true;
            }
        }
        return false;
    }
}
```

- 配置自定义过滤器

```
<property name="filters">
    <map>
        <entry key="authc" value-ref="formAuthenticationFilter"/>
        <entry key="roles">
            <bean class="com.kaishengit.crm.auth.MyRolesFilter"/>
        </entry>
    </map>
</property>
<!--访问路径和角色的配置关系-->
<property name="filterChainDefinitions">
    <value>
        /static/** = anon
        /favicon.ico = anon
        /home = roles["admin,hr,CTO"]
        /** = authc
    </value>
</property>
```

JSP Tag

凯盛软件

```
<%@ taglib prefix="shiro" uri="http://shiro.apache.org/tags" %>

Hello, <shiro:principal/>, how are you today?

<shiro:hasRole name="administrator">
        <a href="admin.jsp">Administer the system</a>
</shiro:hasRole>

<shiro:hasAnyRoles name="developer, manager , administrator">
        You are either a developer, manager, or administrator.
</shiro :hasAnyRoles >

<shiro:hasPermission name="user:create">
        <a href="createUser.jsp">Create a new User</a>
</shiro:hasPermission>
```