权限控制组件设计及应用概要

[权限控制组件设计及应用概要 1](#_Toc468365617)

[1 总体说明 1](#_Toc468365618)

[2 权限控制器 1](#_Toc468365619)

[2.1匿名用户登录控制器 2](#_Toc468365620)

[2.2基于token的登录控制器 2](#_Toc468365621)

[2.3基于ROLE的权限控制器 2](#_Toc468365622)

[2.4基于PERMISSION的权限控制器 2](#_Toc468365623)

[3如何在应用程序中引入并应用该组件 3](#_Toc468365624)

[3.1在POM中添加依赖 3](#_Toc468365625)

[3.2在web.xml中引入Shiro的拦截器 3](#_Toc468365626)

[3.3提供AuthCenter的路径配置文件 4](#_Toc468365627)

[3.4提供权限配置 4](#_Toc468365628)

[3.5在代码接口层面控制权限 4](#_Toc468365629)

[4 如何在业务代码中获取缓存的用户及权限信息 4](#_Toc468365630)

# 1 总体说明

用户和权限数据提供以及认证主要由AuthCenter负责， 但权限控制由support-auth-context组件负责。

主要引入并封装了开源组件Apache Shiro来进行权限控制。Apache Shiro 是一个用 Java 语言实现的框架，通过一个简单易用的 API 提供身份验证和授权。使用 Apache Shiro，就能够为应用程序提供安全性而又无需从头编写所有代码。

support-auth-context组件已经封装了权限控制和Apache Shiro的细节，其它工程和应用程序在使用该组件时，无需过多关注这些底层细节。并且，根据业务需求，可以很容易进行适配和扩展。

# 2 权限控制器

根据当前的业务需求，现已实现并默认配置了以下几种权限控制器：匿名用户登录控制器，基于token的登录控制器，基于ROLE的权限控制器，基于PERMISSION的权限控制器。根据业务需求，可以定制其它控制器。

## 2.1匿名用户登录控制器

配置该类控制器的请求，无需提供token就可以正常消费后端应用程序的服务和资源。配置示例如下：

/api/\*/non-login-urls/\*\*=anon

备注：/api/后面的\*是服务版本号的通配符。

## 2.2基于token的登录控制器

对于需要登录的请求，在请求头部需要提供token信息。没有token或格式不符的请求，会被登录控制器拦截并拒绝，返回401 Unauthorized错误。只配置该控制器的请求，只会检查是否认证通过，而不会对ROLE和PERMISSION权限进行检查。

配置示例：

/api/\*\*=authcToken

请求头部示例：

Authorization: token 7f8ClT6b-zzIzAFhrbVyr3t5IqgJkSxOdHNj80\_xx5wHFrS\_zgXokA

## 2.3基于ROLE的权限控制器

配置该类控制器的请求，组件会对当前用户是否具备该ROLE进行检查。该类控制器的请求，必须要先登录才能正常使用，也就是说必须经过登录控制器拦截过的请求才能进行权限检查。权限数据来源于AuthCenter。权限不具备的请求，会被拦截并拒绝，返回403 Forbidden错误。

配置示例：

/api/\*/nurses/\*\*=authcToken,roles[NURSE,HEAD\_NURSE]

示例中，表示这类接口和服务，必须是具有护士或护士长角色的用户才有权限调用。

## 2.4基于PERMISSION的权限控制器

类似于基于ROLE的权限控制器，可以一并使用。

配置示例：

/api/\*/menus/\*\*=authcToken,perms[SYSTEM:LIST\_MENUS]

# 3如何在应用程序中引入并应用该组件

## 3.1在POM中添加依赖

<dependency>

<groupId>com.lachesis.support</groupId>

<artifactId>support-auth-context</artifactId>

<version>${support-auth-context.version}</version>

</dependency>

当前support-auth-context版本为：1.0.0.0-SNAPSHOT

## 3.2在web.xml中引入Shiro的拦截器

1. 在web.xml将public-support-auth-context.xml添加到application context，如：

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

**classpath\*:spring/public-support-auth-context.xml,**

classpath\*:spring/support-auth-demo-application.xml

</param-value>

</context-param>

1. 在web.xml配置拦截器

<filter>

<filter-name>shiroFilter</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

<async-supported>true</async-supported>

<init-param>

<param-name>targetFilterLifecycle</param-name>

<param-value>true</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>shiroFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

## 3.3提供AuthCenter的路径配置文件

在src\main\resources目录下或在${user.home}/project-support-auth-config目录下提供support-auth-context-comm.properties配置文件，配置项如下：

support.auth.context.authc.authorize.url=http://{auth-center-host}:{9090}/authc/api/{v1}/authorizations

## 3.4提供权限配置

在src\main\resources目录下提供shiro.properties配置文件。

该配置文件示例：

/api/\*/non-login-urls/\*\*=anon

/api/\*/public-resources/\*\*=anon

/api/\*/nurses/\*\*=authcToken,roles[NURSE,HEAD\_NURSE],perms[SYSTEM:LOGIN]

/ api /\*\*=authcToken,perms[SYSTEM:LOGIN]

## 3.5在代码接口层面控制权限

支持通过注解直接在方法粒度控制访问。如：

@RequiresPermissions("DEPT:LIST")

public List<SimpleUserVo> listAllOfDepartment(String deptId) {

return nurses.get(deptId);

}

# 4 如何在业务代码中获取缓存的用户及权限信息

在业务处理中，有很多场景需要获取当前用户的信息以及权限信息来进行后续的业务处理，组件已经将用户数据及权限信息放入到请求对象中，可以直接获取。SecurityContext为不可变对象，从实现上已经控制了其它业务代码不能对该数据进行修改。

SecurityContext ctx = (SecurityContext) request.getAttribute(AuthContextConstants.REQUEST\_ATTR\_SECURITY\_CONTEXT);