CSDN 博客 下载 学习 社区 知道 GitCode InsCode 会议 java 搜索 会员中心 消息

ITKaven 关注

原创 ITKaven 已于 2022-01-22 15:01:05 修改 阅读量4.4k 收藏 18 点赞数 3

分类专栏： Spring Security 文章标签： spring java 后端

Spring Security 专栏收录该内容 31 订阅 12 篇文章

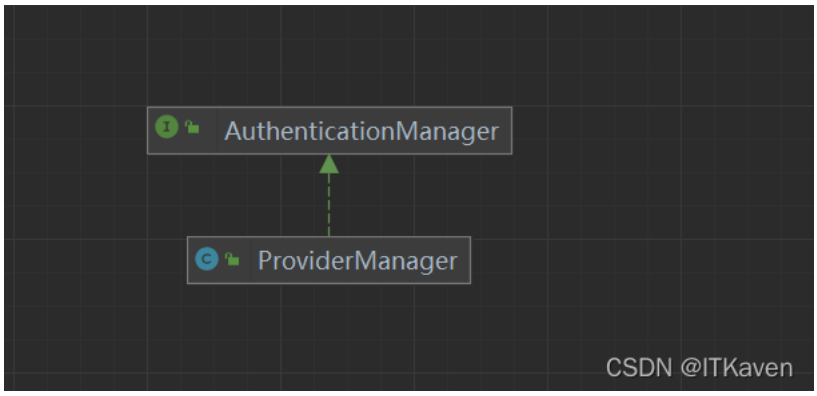
AuthenticationManager

AuthenticationManager 用于处理 Authentication 请求，上一篇博客已经介绍了 Authentication ， Spring Security 在进行身份验证时，会创建身份验
即 Authentication 实例，提供给 AuthenticationManager 接口的实现类进行处理，由实现类中的 AuthenticationProvider 列表进行验证。

- Spring Security：身份验证令牌Authentication介绍与Debug分析

```
public interface AuthenticationManager {
    /**
     * 尝试对传递的Authentication对象进行身份验证
     * 如果验证成功，则返回完全填充的Authentication对象（包括授予的权限）
     * AuthenticationManager必须遵守以下关于异常的约定：
     * 如果帐户被禁用并且AuthenticationManager可以测试此状态，则必须抛出DisabledException
     * 如果帐户被锁定并且AuthenticationManager可以测试帐户锁定，则必须抛出LockedException
     * 如果提供了不正确的凭证（比如密码），则必须抛出BadCredentialsException
     * 参数：authentication - 身份验证请求的封装
     * 返回：一个完全经过身份验证的对象，包括凭证
     */
    Authentication authenticate(Authentication authentication)
            throws AuthenticationException;
}
```

AuthenticationManager 接口只有一个实用的实现类 ProviderManager ，其他的实现类都是内部类，并且不提供身份验证处理的具体实现。



ProviderManager

ProviderManager 通过 AuthenticationProvider 列表迭代验证 Authentication 请求， AuthenticationProvider 列表通常会按顺序尝试，直到提供非空
响应表示 AuthenticationProvider 有权决定身份验证请求，并且不再尝试其他 AuthenticationProvider 。 如果后续 AuthenticationProvider 成功验证
则会忽略之前的验证异常并使用此次成功的验证。如果没有后续 AuthenticationProvider 提供非空响应或新的 AuthenticationException ，则将使用
AuthenticationException 。

如果没有 AuthenticationProvider 返回非空响应，或者没有 AuthenticationProvider 可以处理 Authentication ，则 ProviderManager 将抛出
ProviderNotFoundException 。也可以设置父 AuthenticationManager ，如果配置的 AuthenticationProvider 都不能执行身份验证，也会尝试这样做。

此过程的例外情况是 AuthenticationProvider 抛出 AccountStatusException ，在这种情况下，将不会继续迭代列表中的其他 AuthenticationProvider
后，则将从返回的 Authentication 对象中清除凭证（如果它实现了 CredentialsContainer 接口）。可以通过修改 eraseCredentialsAfterAuthenticat
控制此行为。

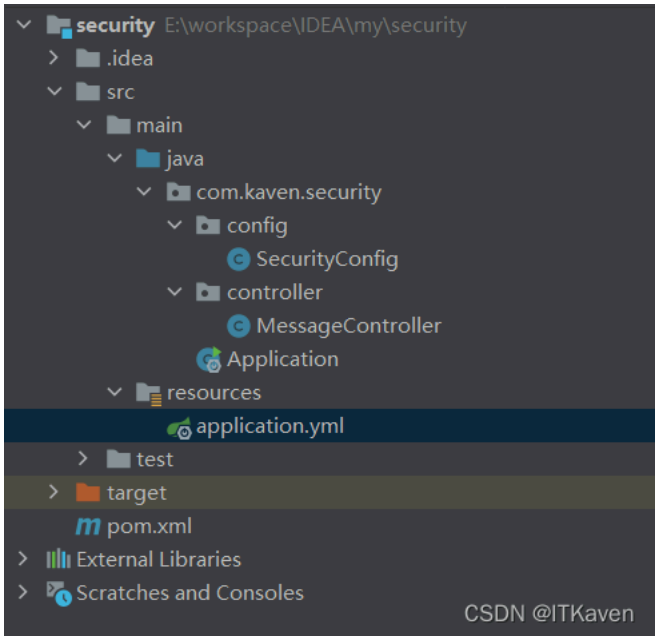
身份验证事件发布被委托给配置的 AuthenticationEventPublisher ，它默认为不发布事件的空实现，如果想接收事件，则必须设置
AuthenticationEventPublisher ，标准实现是 DefaultAuthenticationEventPublisher ，它将常见异常映射到事件（在身份验证失败的情况下），并在
功时发布 AuthenticationSuccessEvent 。

ITKaven 关注

```
    private static final Log logger = LogFactory.getLog(ProviderManager.class);

    // 用于验证事件的发布
    private AuthenticationEventPublisher eventPublisher = new NullEventPublisher();
    // 用于验证Authentication的AuthenticationProvider列表
    private List<AuthenticationProvider> providers = Collections.emptyList();
    // 用于访问来自MessageSource的消息
    protected MessageSourceAccessor messages = SpringSecurityMessageSource.getAccessor();
```

Debug分析

项目结构图：



pom.xml：

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.kaven</groupId>
    <artifactId>security</artifactId>
    <version>1.0-SNAPSHOT</version>

    <parent>
```

application.yml：

```
spring:
  security:
    user:
      name: kaven
      password: itkaven
logging:
  level:
    org:
```

ITKaven 关注

```
springframework:
  security: DEBUG
```

`SecurityConfig` （`Spring Security` 的配置类，不是必须的，因为有默认的配置）：

```
package com.kaven.security.config;

import org.springframework.security.config.Customizer;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
```

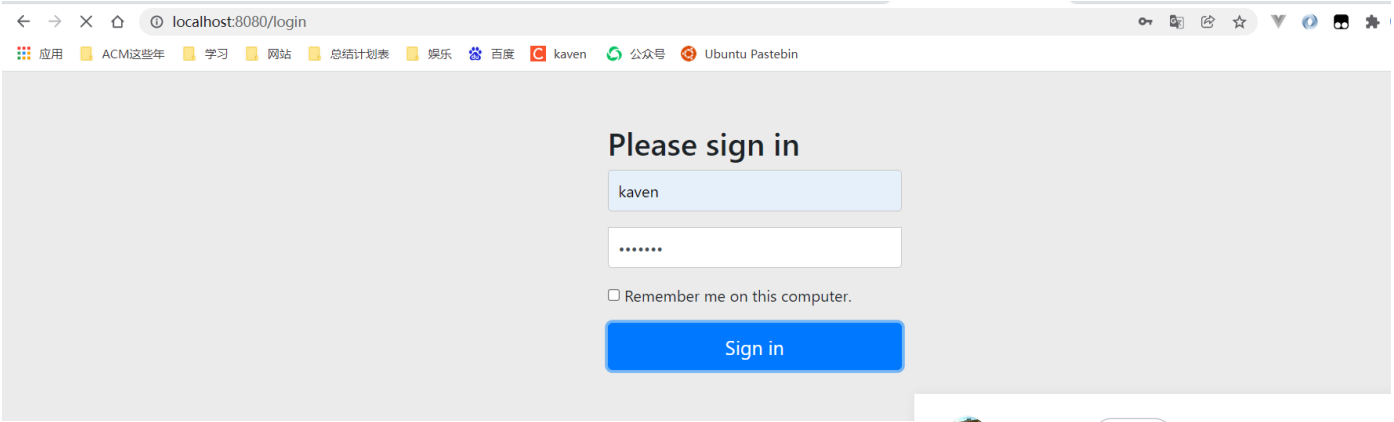
`MessageController` （定义接口）：

```
package com.kaven.security.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class MessageController {
    @GetMapping("/message")
    public String getMessage() {
        return "hello spring security";
    }
}
```

启动类：

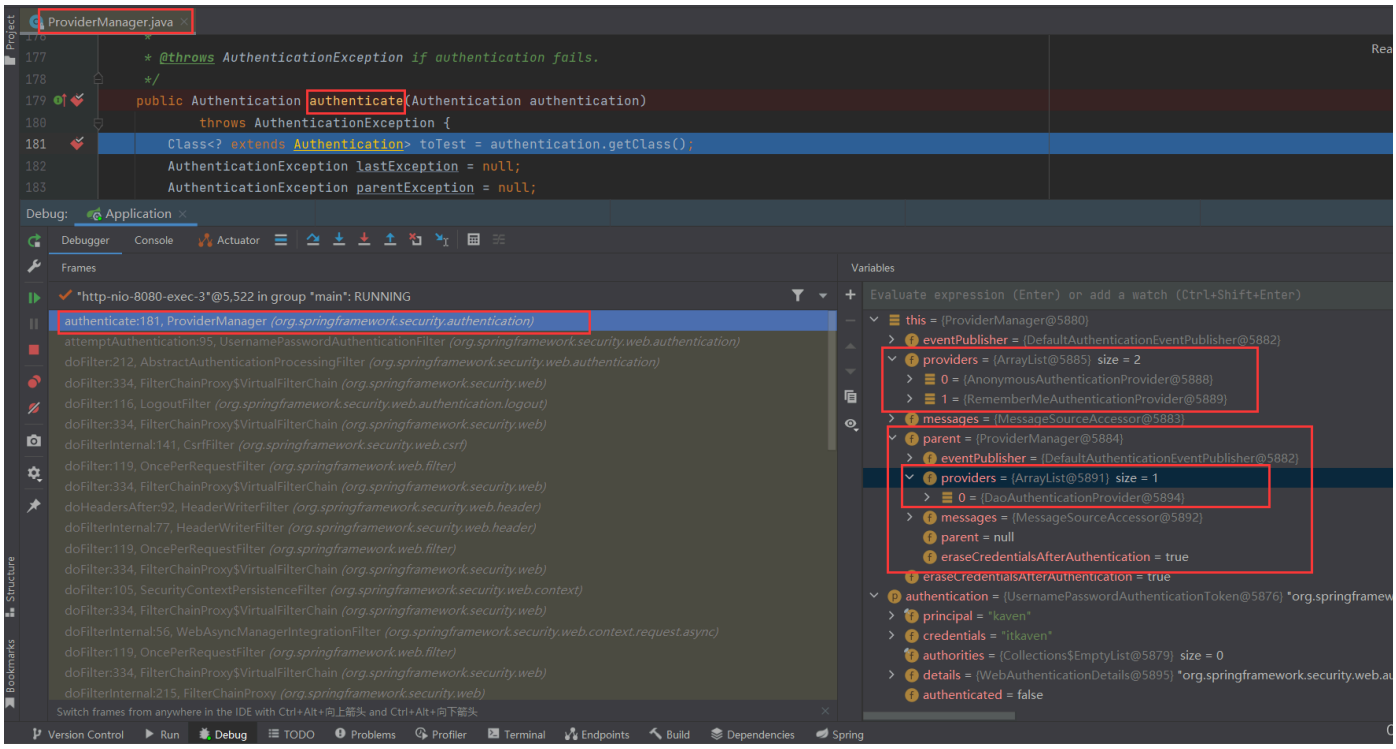
```
package com.kaven.security;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class);
    }
}
```

`Debug` 方式启动应用，访问 `http://localhost:8080/message` ，请求会被重定向到表单登陆页，需要输入用户名和密码。

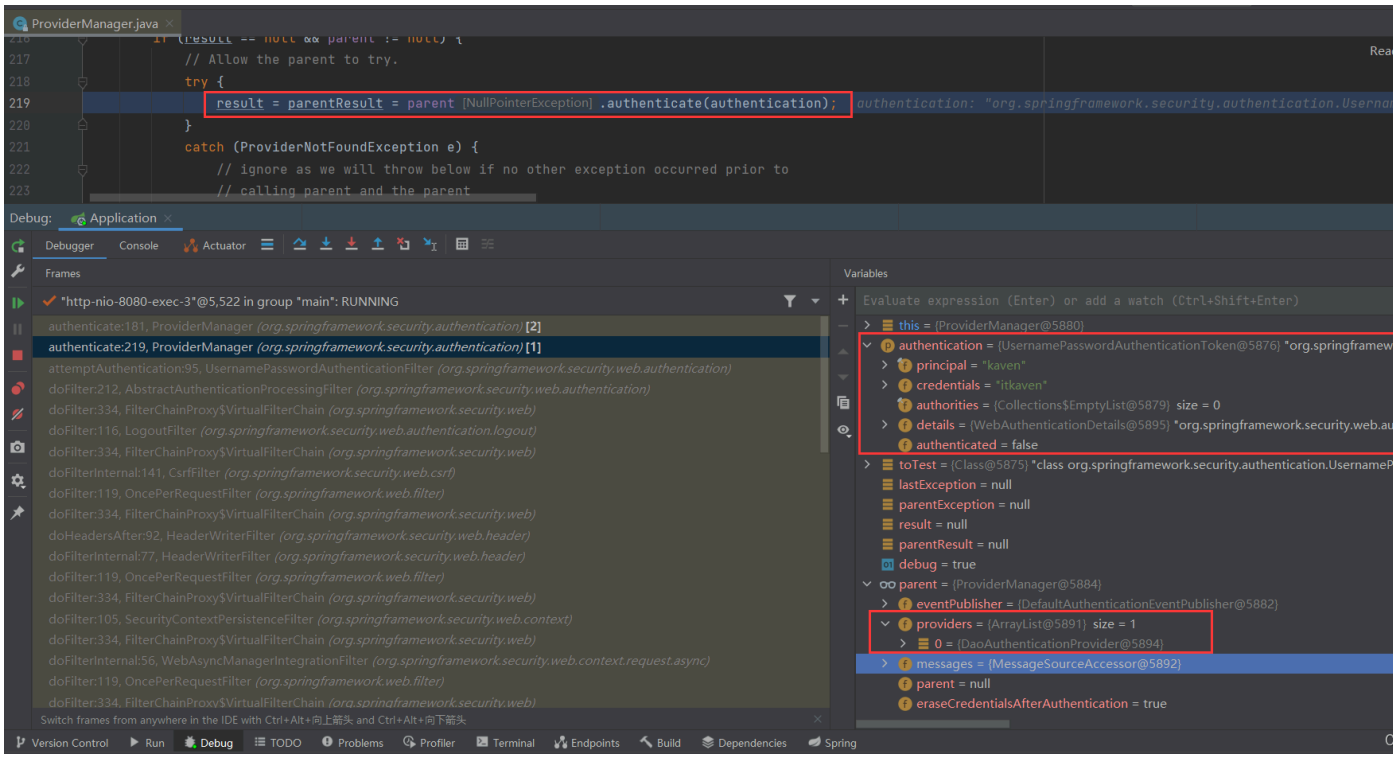


身份验证请求会被 `ProviderManager` 实例处理，如下图所示，该 `ProviderManager` 实例有两个 `Auth`

ITKaven 关注

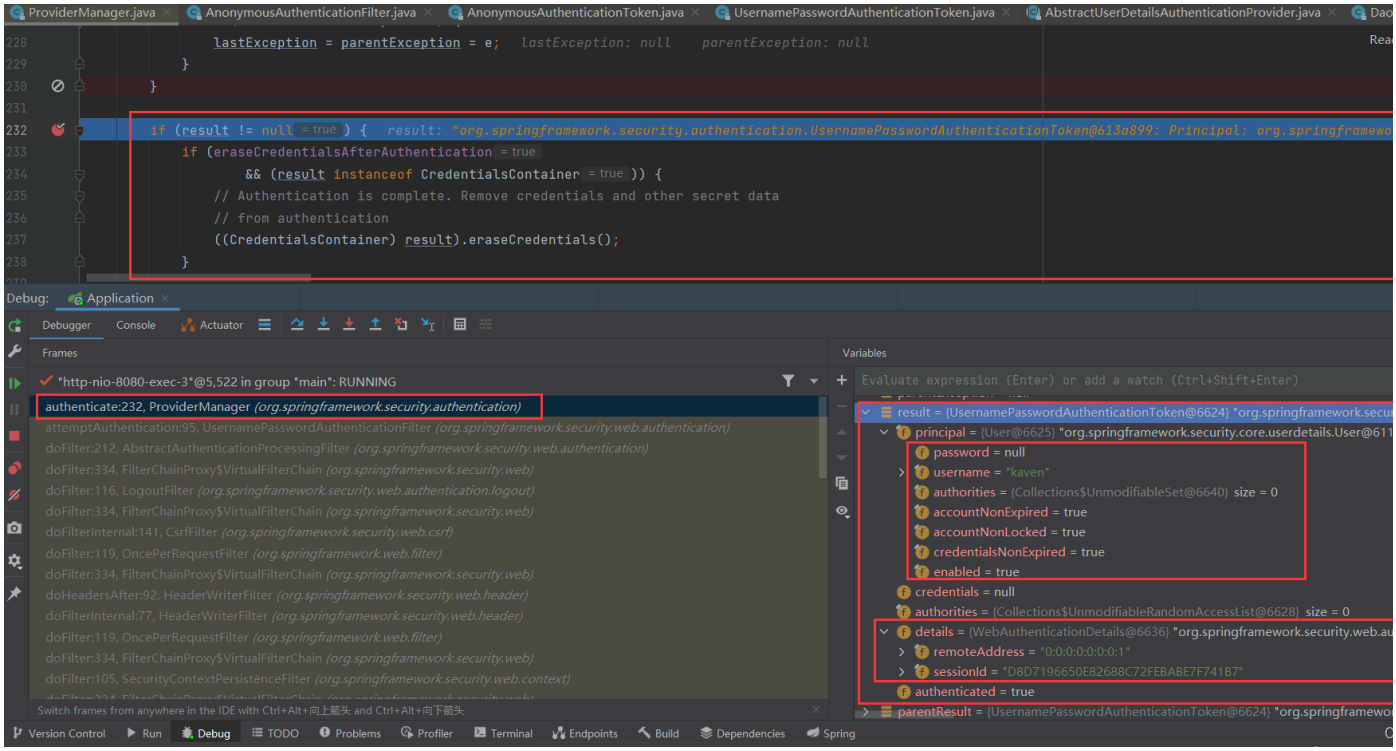


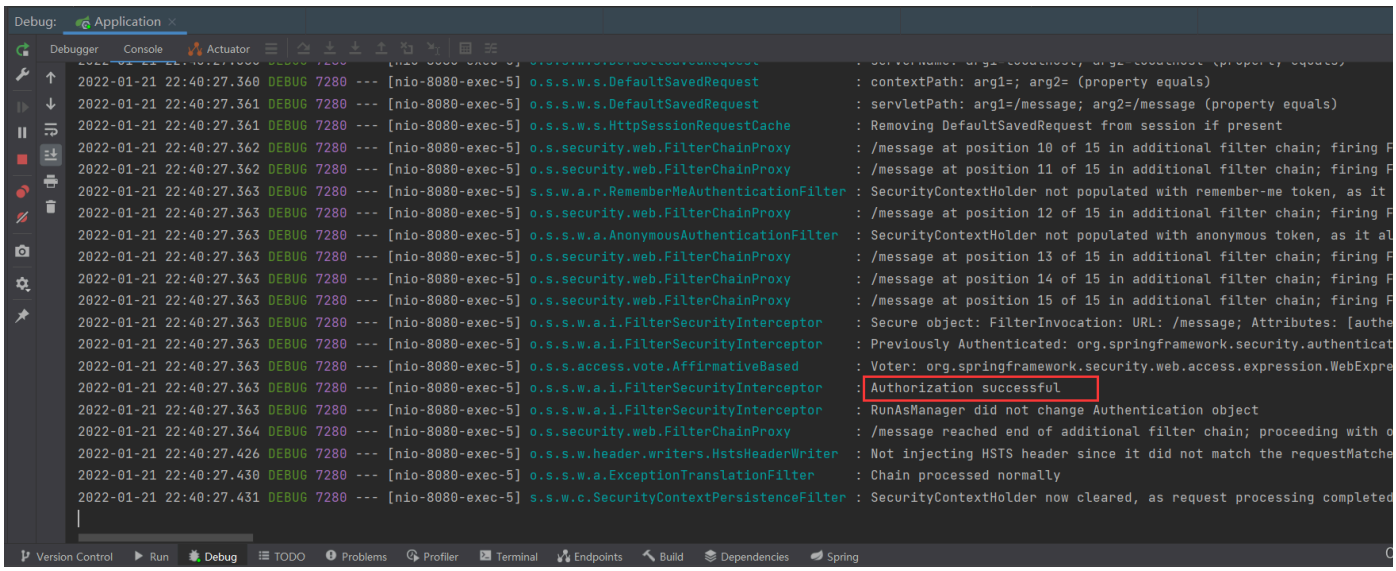
该 ProviderManager 实例的两个 AuthenticationProvider 都不支持该身份验证请求，但它的父 AuthenticationManager 的 AuthenticationProvider 支持请求。



父 AuthenticationManager 的 AuthenticationProvider 会对身份验证请求进行验证，很显然是验证成功了。

ITKaven 关注





身份验证处理 `AuthenticationManager` 介绍与 `Debug` 分析就到这里，如果博主有说错的地方或者大家有不同的见解，欢迎大家评论补充。

文章知识点与官方知识档案匹配，可进一步学习相关知识

Java技能树 首页 概览 149725 人正在系统学习中

如何使用Spring Security手动验证用户的方法示例
在这篇文章中，我们将详细介绍如何使用Spring Security手动验证用户的方法示例，主要包括如何在Spring Security和Spring MVC中设置经过身份验证的用户。 概述 在Sp

spring-security：Spring Boot，JWT，基于角色的身份验证
总的来说，Spring Security与Spring Boot的结合，以及JWT的引入，为Web应用提供了一套高效、安全的身份验证和授权方案。通过合理的配置和设计，我们可以构建出

Spring Security —06—自定义认证数据源
在工厂中默认创建AuthenticationManager// @Autowired //代码从官网复制而来// public void initialize(AuthenticationManagerBuilder builder) throws Exception {// System.c

SpringSecurity--权限管理架构介绍_authenticationmanager
Spring Security 整体架构 认证 AuthenticationManager(认证管理器) Authentication SecurityContextHolder 授权 AccessDecisionManager AccessDecisionVoter ConfigAttr

Spring Security通过AuthenticationManager的逻辑实现多种认证方式.docx
为一个 Servlet Filter 应该存在一个doFilter实现方法，而它却没有，其实它的父类AbstractAuthenticationProcessingFilter提供了具体的实现。稍后我们会根据这个实现引出

ITKaven 关注

Spring Security之AuthenticationManager、ProviderManager、Authenticati...
AuthenticationManager是一个顶级接口,用来处理身份验证请求,并返回一个Authentication对象,如果发生异常将会抛出AuthenticationException; AuthenticationManager的实

Spring Security实现权限认证与授权_spring security 权限认证
AuthenticationManager就是Spring Security用于执行身份验证的组件,只需要调用它的authenticate()方法即可完成认证。Spring Security默认的认证方式就是在UsernameP

Security AuthenticationManager 和Authentication 最新发布 jinwenjieok的
spring security -AuthenticationManager & Authentication

SpringSecurity框架原理浅谈之AuthenticationManager 黄先生的
AuthenticationManager这个接口方法非常奇特，入参和返回值的类型都是Authentication。该接口的作用是对用户的未授信凭据进行认证，认证通过则返回授信状态的凭据

Spring Security：概念模型 AuthenticationManager 认证管理器 热门推荐 Details Inside Sp
Spring Security中，接口AuthenticationManager用于抽象建模认证管理器，用于处理一个认证请求，也就是Spring Security中的Authentication认证令牌。 AuthenticationM

Spring Security 实战干货：用户是如何进行登录认证的 码农小
1. 前言我们上一篇介绍了UsernamePasswordAuthenticationFilter的工作流程，留下了一个小小的伏笔，作为一个Servlet Filter应该存在一个doF...

SpringBoot_SpringSecurity:前后端分离登陆拦截设计
SpringSecurity提供了认证（Authentication）和授权（Authorization）功能，可以实现用户身份验证、访问控制、CSRF防护等功能。在前后端分离的架构下，通常前端负

Spring Security验证流程剖析及自定义验证方法
总之，Spring Security的验证流程是通过AuthenticationManager和AuthenticationProvider协作完成的。理解这个过程对于定制安全策略和解决安全问题至关重要。通过实现

详解最简单易懂的Spring Security 身份认证流程讲解
"Spring Security 身份认证流程讲解" Spring Security 是一个功能强大且复杂的框架，对于 JavaEE 工程师来说是一个必备的知识点。本文将从身份认证的角度讲解 Spring

AuthenticationManager weixin_45802793的
1.了解 1.1ProviderManager ProviderManager管理了一个AuthenticationProvider列表，每个AuthenticationProvider都是一个认证器 ProviderManager 相当于代理了多个 A

Spring Security 之 AuthenticationManager 源码解析 weixin_38982591的
在 UsernamePasswordAuthenticationFilter 源码分析 中，最后在类UsernamePasswordAuthenticationFilter 的验证方法 attemptAuthentication() 会将用户表单提交过来的用

Spring Security API： AuthenticationManager dengdeying的
文章目录AuthenticationManagerDeclaredClass JDocMethod authenticateMethod JDOC AuthenticationManager Declared package org.springframework.security.authenti

Spring Security配置AuthenticationManager qq_44113347的
Override总结一旦通过 configure 方法自定义 AuthenticationManager 实现 就回将工厂中自动配置 AuthenticationManager 进行覆盖一旦通过 configure 方法自定义 Auther

AuthenticationManager 认证原理分析 （Spring Security） qq_44815020的
这里写目录标题说明概述代码执行流程认证函数执行流程图总结 说明 AuthenticationManager 是 Spring Security提供的一套认证服务 类运行流程图： 概述 上述是Authent

从源码角度拆解SpringSecurity之C位的AuthenticationManager 会飞的
上一篇我们简单拆解了运行时的大致流程，知道了拦截动作都来自于FilterChainProxy，以及内部的Filter链怎么来的，顺序的重要性，以及如何分发的，还针对所学习内容

Spring Security配置全局 AuthenticationManager Leon_Jinhai_Sun的
Topical Guide | Spring Security Architecture 默认的全局 AuthenticationManager @Configuration public class WebSecurityConfigurer extends WebSecurityConfigurerAda

Spring Security学习（七）——父子AuthenticationManager（ProviderManager） sadoshi的
Spring Security学习（六）——配置多个Provider》有个很奇怪的现象，如果我们不添加DaoAuthenticationProvider到HttpSecurity中，似乎也能够达到类似的效果。那我们

关于我们 招贤纳士 商务合作 寻求报道 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00
公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心
家长监护 网络110报警服务 中国互联网举报中心 Chrome商店下载 账号管理规范 版权与免责声明 版权申诉 出版物许可证 营业执照
©1999-2024北京创新乐知网络技术有限公司

ITKaven
码龄7年 暂无认证

542	2万+	105万+	227万+	
原创	周排名	总排名	访问	等级

2万+	1368	1992	937	3844
积分	粉丝	获赞	评论	收藏

ITKaven 关注

私信 关注

搜博主文章

热门文章

分类专栏

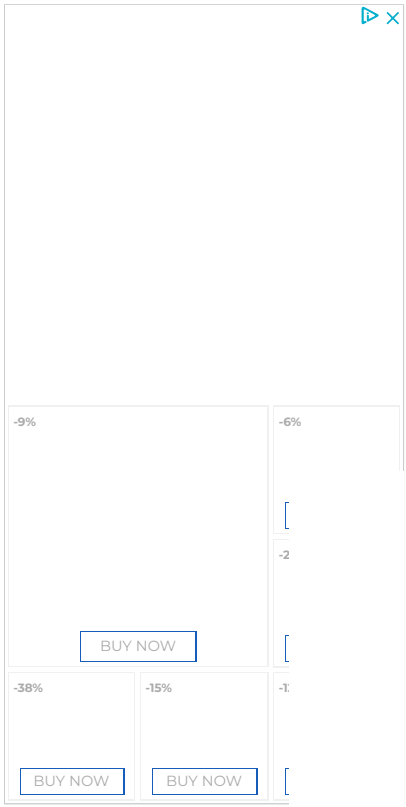
最新评论

最新文章

ITKaven 关注

VirtualBox：系统安装重启后又要重新安装

2022年 80篇 2021年 55篇
2020年 206篇 2019年 10篇
2018年 220篇 2017年 31篇



目录

ITKaven 关注