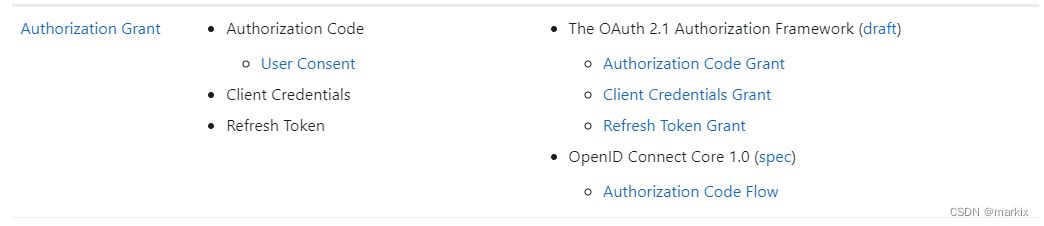
10 令牌颁发 OAuth2TokenEndpointFilter

前面我们已经讲完 客户端认证 的各种方式，那是令牌颁发的前提。这篇开始，我们就来研究下 令牌颁发 个中奥秘。

令牌颁发

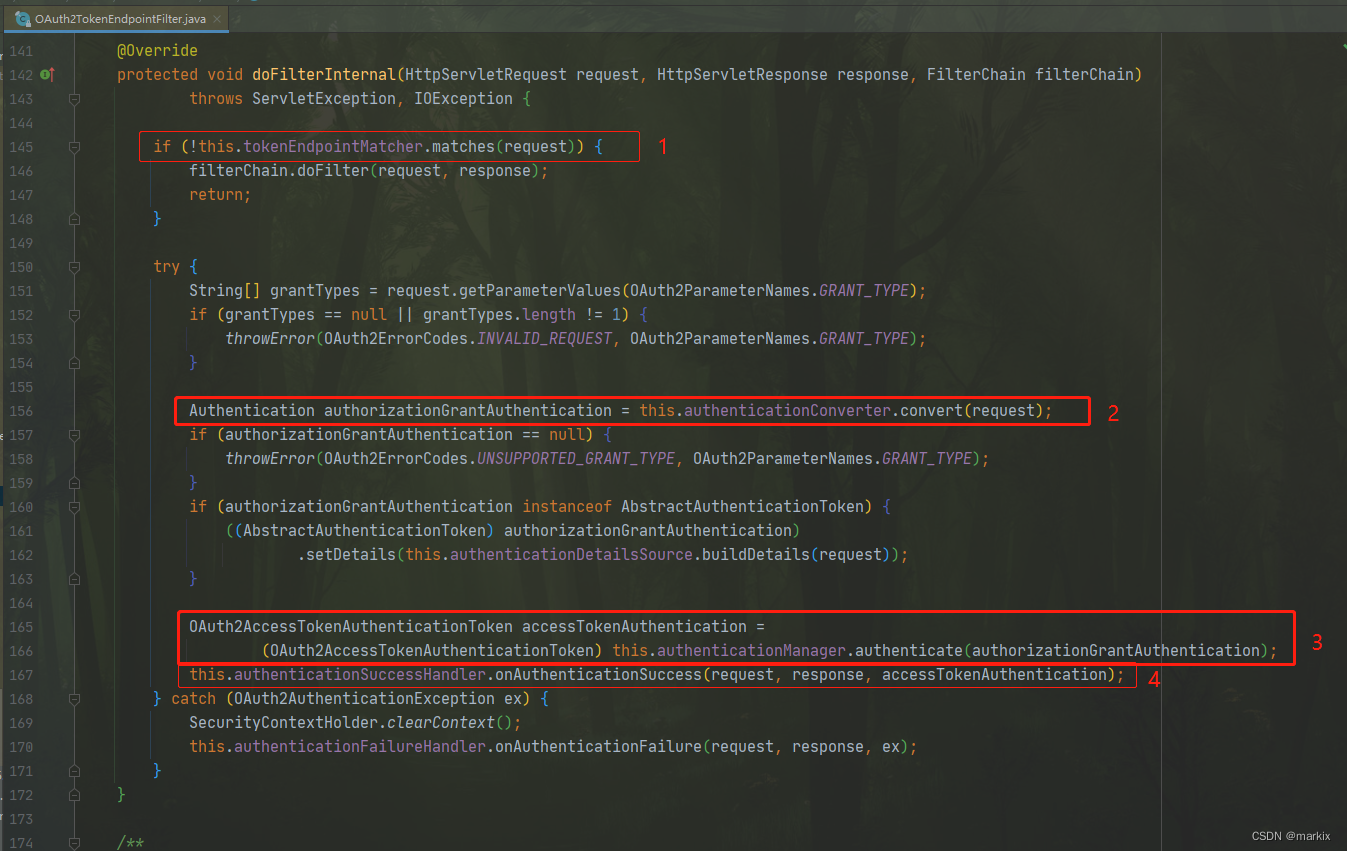
授权服务器提供令牌颁发接口（/oauth2/token），由客户端发起请求，授权服务器生成访问令牌（access\_token）返回，客户端使用此令牌才能去调用资源服务器的接口。

Spring Authorization Server 目前支持如下三种令牌颁发策略：Authorization Code、Client Credentials、Refresh Token，分别对应 授权码模式、客户端凭证模式、刷新令牌模式。



OAuth2TokenEndpointFilter

实现令牌颁发功能的拦截器就是 OAuth2TokenEndpointFilter。其核心代码如下：



判断此次请求是否是 “令牌颁发” 请求，若是，则继续下面逻辑，否则跳过

解析请求中的参数，构建成一个 Authentication

对 Authentication 进行处理

到这一步说明access\_token生成好了， 将access\_token和相关信息响应给请求方。

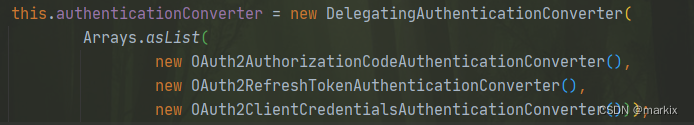
有没有感觉这个代码实现逻辑很眼熟？！！是不是在哪见过？

没错，专栏第五篇 客户端认证 OAuth2ClientAuthenticationFilter 中也正是用的这种实现套路。将不同实现策略抽象为 AuthenticationConverter 和 AuthenticationProvider 接口。每种策略实际上就是一个 AuthenticationConverter 实现类 加上一个 AuthenticationProvider实现类。

相关细节可以回看之前文章，这里不再赘述。 这里只说明一下他们不同的实现类：

authenticationConverter（DelegatingAuthenticationConverter）

不同的AuthenticationConverter实现类



authenticationManager（ProviderManager）

不同的AuthenticationProvider实现类

OAuth2TokenEndpointConfigurer#createDefaultAuthenticationProviders

小结

每种 令牌颁发策略 实际就是一个 AuthenticationConverter 搭配一个 AuthenticationProvider 组合而成，一个负责解析请求参数，一个负责处理请求参数，执行相应的逻辑！

令牌颁发策略

注意：下文中传参不会特意强调 客户端认证 的参数，并不代表不需要，相反的，是必要的！ 只是在前文已经详细说明，不同的认证方式有不同参数，读者可以随便选用一种，常用就是 client\_secret\_basic （请求头）方式。

Authorization Code（授权码模式）

客户端访问授权链接，用户授权，客户端获得授权码code，通过code获取令牌

传参

grant\_type：固定值 authorization\_code

code：授权码

核心类

OAuth2AuthorizationCodeAuthenticationConverter

OAuth2AuthorizationCodeAuthenticationProvider

Client Credentials（客户端凭证模式）

传参

grant\_type：固定值 client\_credentials

核心类

OAuth2ClientCredentialsAuthenticationConverter

OAuth2ClientCredentialsAuthenticationProvider

Refresh Token（刷新令牌模式）

当客户端支持刷新令牌时，授权服务器颁发访问令牌(access\_token)时会同时颁发刷新令牌(refresh\_token)，客户端可以使用刷新令牌重新获取访问令牌。（由于访问令牌时效比较短，刷新令牌时效比较长，通过刷新令牌获取访问令牌，避免多次授权）

传参

grant\_type：固定值 refresh\_token

refresh\_token：刷新令牌

核心类

OAuth2RefreshTokenAuthenticationConverter

OAuth2RefreshTokenAuthenticationProvider

小结

后续通过示例来详细说明，敬请期待。

end