扩展JWT中存储的内容

有时候我们需要扩展JWT中存储的内容，这里我们在JWT中扩展一个`key为enhance，value为enhance info`的数据。

继承TokenEnhancer实现一个JWT内容增强器

```
package com.xxxx.springsecurityoauth2demo.config;

import org.springframework.security.oauth2.common.DefaultOAuth2AccessToken;
import org.springframework.security.oauth2.common.OAuth2AccessToken;
import org.springframework.security.oauth2.provider.OAuth2Authentication;
import org.springframework.security.oauth2.provider.token.TokenEnhancer;

import java.util.HashMap;
import java.util.Map;

/**
 * JWT内容增强器
 * @author zhoubin
 * @since 1.0.0
 */
public class JwtTokenEnhancer implements TokenEnhancer {

   @Override
   public OAuth2AccessToken enhance(OAuth2AccessToken accessToken, OAuth2Authentication
authentication) {
      Map<String,Object> info = new HashMap<>();
      info.put("enhance","enhance info");
      ((DefaultOAuth2AccessToken)accessToken).setAdditionalInformation(info);
      return accessToken;
   }
}
```

创建一个JwtTokenEnhancer实例

```
package com.xxxx.springsecurityoauth2demo.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.oauth2.provider.token.TokenStore;
import org.springframework.security.oauth2.provider.token.store.JwtAccessTokenConverter;
import org.springframework.security.oauth2.provider.token.store.JwtTokenStore;

/**
 * 使用Jwt存储token的配置
 * @author zhoubin
 * @since 1.0.0
 */
@Configuration
```

```
public class JwtTokenStoreConfig {

   @Bean
   public TokenStore jwtTokenStore(){
      return new JwtTokenStore(jwtAccessTokenConverter());
   }

   @Bean
   public JwtAccessTokenConverter jwtAccessTokenConverter(){
      JwtAccessTokenConverter accessTokenConverter = new JwtAccessTokenConverter();
      //配置JWT使用的秘钥
      accessTokenConverter.setSigningKey("test_key");
      return accessTokenConverter;
   }

   @Bean
   public JwtTokenEnhancer jwtTokenEnhancer() {
      return new JwtTokenEnhancer();
   }
}
```

在认证服务器配置中配置JWT的内容增强器

```
package com.xxxx.springsecurityoauth2demo.config;

import com.xxxx.springsecurityoauth2demo.component.JwtTokenEnhancer;
import com.xxxx.springsecurityoauth2demo.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.crypto.password.PasswordEncoder;
import
org.springframework.security.oauth2.config.annotation.configurers.ClientDetailsServiceConfi
gurer;
import
org.springframework.security.oauth2.config.annotation.web.configuration.AuthorizationServer
ConfigurerAdapter;
import
org.springframework.security.oauth2.config.annotation.web.configuration.EnableAuthorization
Server;
import
org.springframework.security.oauth2.config.annotation.web.configurers.AuthorizationServerEn
dpointsConfigurer;
import org.springframework.security.oauth2.provider.token.TokenEnhancer;
import org.springframework.security.oauth2.provider.token.TokenEnhancerChain;
import org.springframework.security.oauth2.provider.token.TokenStore;
import org.springframework.security.oauth2.provider.token.store.JwtAccessTokenConverter;

import java.util.ArrayList;
import java.util.List;

/**
```

```
 * 授权服务器配置
 * @author zhoubin
 * @since 1.0.0
 */
@Configuration
@EnableAuthorizationServer
public class AuthorizationServerConfig extends AuthorizationServerConfigurerAdapter {

    @Autowired
    private PasswordEncoder passwordEncoder;

    @Autowired
    private AuthenticationManager authenticationManager;

    @Autowired
    private UserService userService;

    @Autowired
    @Qualifier("jwtTokenStore")
    private TokenStore tokenStore;
    @Autowired
    private JwtAccessTokenConverter jwtAccessTokenConverter;
    @Autowired
    private JwtTokenEnhancer jwtTokenEnhancer;

    /**
     * 使用密码模式需要配置
     */
    @Override
    public void configure(AuthorizationServerEndpointsConfigurer endpoints) {
        TokenEnhancerChain enhancerChain = new TokenEnhancerChain();
        List<TokenEnhancer> delegates = new ArrayList<>();
        //配置JWT的内容增强器
        delegates.add(jwtTokenEnhancer);
        delegates.add(jwtAccessTokenConverter);
        enhancerChain.setTokenEnhancers(delegates);
        endpoints.authenticationManager(authenticationManager)
                .userDetailsService(userService)
                //配置存储令牌策略
                .tokenStore(tokenStore)
                .accessTokenConverter(jwtAccessTokenConverter)
                .tokenEnhancer(enhancerChain);
    }

    @Override
    public void configure(ClientDetailsServiceConfigurer clients) throws Exception {
        clients.inMemory()
                //配置client_id
                .withClient("admin")
                //配置client-secret
                .secret(passwordEncoder.encode("112233"))
                //配置访问token的有效期
                .accessTokenValiditySeconds(3600)
```

```
                //配置刷新token的有效期
                .refreshTokenValiditySeconds(864000)
                //配置redirect_uri，用于授权成功后跳转
                .redirectUris("http://www.baidu.com")
                //配置申请的权限范围
                .scopes("all")
                //配置grant_type，表示授权类型
                .authorizedGrantTypes("authorization_code","password");
    }
}
```

运行项目后使用密码模式来获取令牌，之后对令牌进行解析，发现已经包含扩展的内容。


```
{
  "user_name": "admin",
  "scope": [
    "all"
  ],
  "exp": 1578906530,
  "authorities": [
    "admin"
  ],
  "jti": "8566cc9c-18cc-4bad-a29f-e54edd7fb19f",
  "client_id": "admin",
  "enhance": "enhance info"
}
```