Spring Security Oauth2 整合JWT

整合JWT

我们拿之前Spring Security Oauth2的完整代码进行修改

添加配置文件JwtTokenStoreConfig.java

```
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 jwtAccessTokenConverter();
    }
}
```

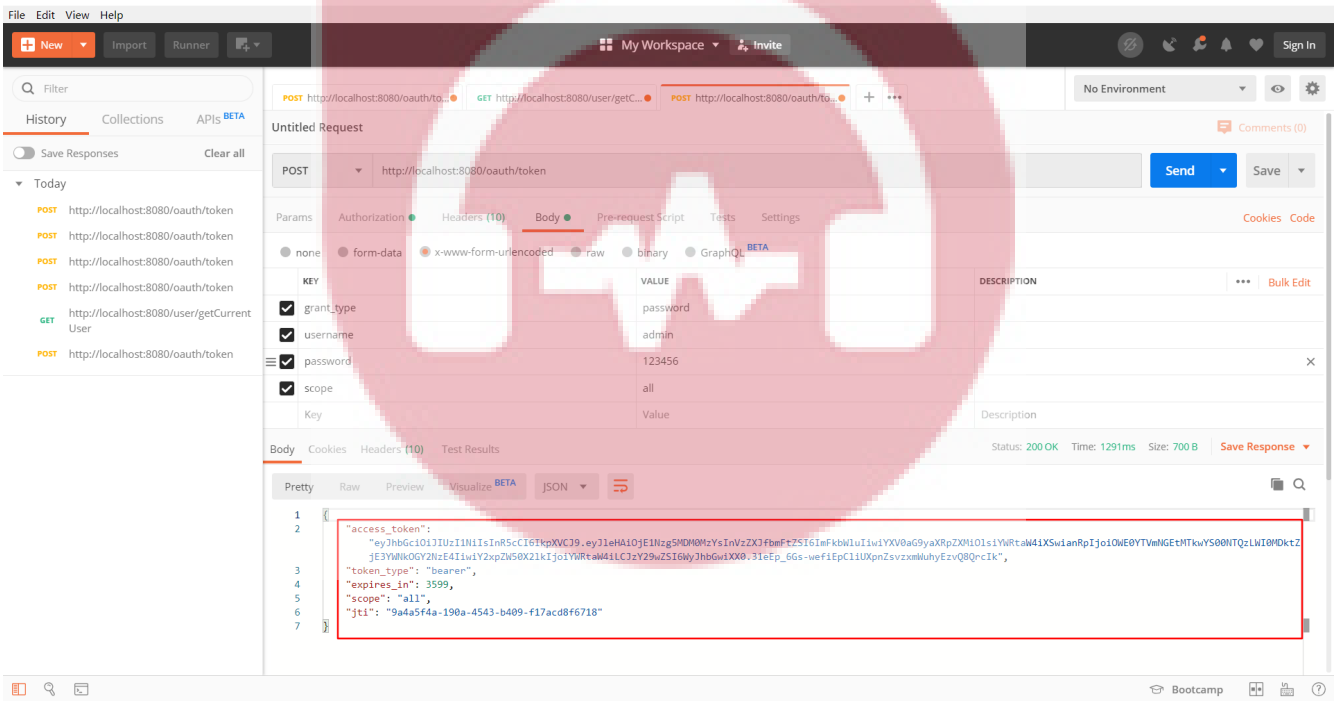
在认证服务器配置中指定令牌的存储策略为JWT

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

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.TokenStore;
import org.springframework.security.oauth2.provider.token.store.JwtAccessTokenConverter;

/**
 * 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;


    /**
     * 使用密码模式需要配置
     */
    @Override
    public void configure(AuthorizationServerEndpointsConfigurer endpoints) {
        endpoints.authenticationManager(authenticationManager)
                .userDetailsService(userService)
                //配置存储令牌策略
                .tokenStore(tokenStore)
                .accessTokenConverter(jwtAccessTokenConverter);
    }
```

```
    @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");
    }
}
```

用密码模式测试：



发现获取到的令牌已经变成了JWT令牌，将access_token拿到https://jwt.io/ 网站上去解析下可以获得其中内容。

```
{
  "exp": 1578903436,
  "user_name": "admin",
  "authorities": [
    "admin"
  ],
  "jti": "9a4a5f4a-190a-4543-b409-f17acd8f6718",
  "client_id": "admin",
  "scope": [
    "all"
  ]
}
```