6、编写配置类

SecurityConfig.java

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

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
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.WebSecurityConfigurerAdapt
er;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

/**
 * Spring Security 配置类
 *
 * @author zhoubin
 */
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

   @Bean
   public PasswordEncoder passwordEncoder() {
      return new BCryptPasswordEncoder();
   }


   @Override
   public void configure(HttpSecurity http) throws Exception {
      http.csrf()
            .disable()
            .authorizeRequests()
            .antMatchers("/oauth/**", "/login/**", "/logout/**")
            .permitAll()
            .anyRequest()
            .authenticated()
            .and()
            .formLogin()
            .permitAll();
   }
}
```

AuthorizationServerConfig.java

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

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
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;

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

    @Autowired
    private PasswordEncoder passwordEncoder;


    @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");
    }
}
```

ResourceServerConfig.java

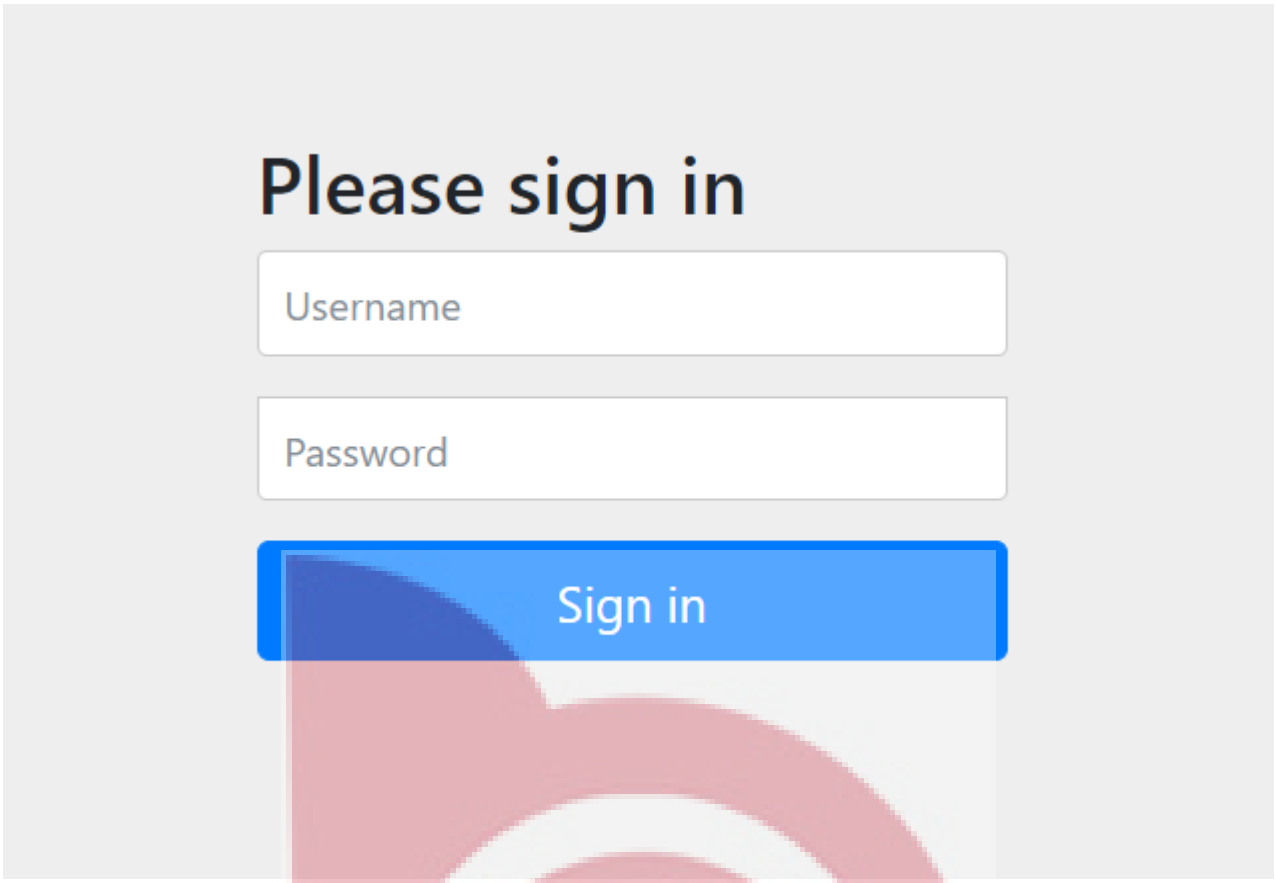
```
package com.xxxx.springsecurityoauth2demo.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
```

```
import
org.springframework.security.oauth2.config.annotation.web.configuration.EnableResourceServe
r;
import
org.springframework.security.oauth2.config.annotation.web.configuration.ResourceServerConfi
gurerAdapter;

/**
 * 资源服务器配置
 *
 * @author zhoubin
 * @since 1.0.0
 */
@Configuration
@EnableResourceServer
public class ResourceServerConfig extends ResourceServerConfigurerAdapter {

    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .anyRequest()
                .authenticated()
                .and()
                .requestMatchers()
                .antMatchers("/user/**");//配置需要保护的资源路径
    }
}
```

7、测试

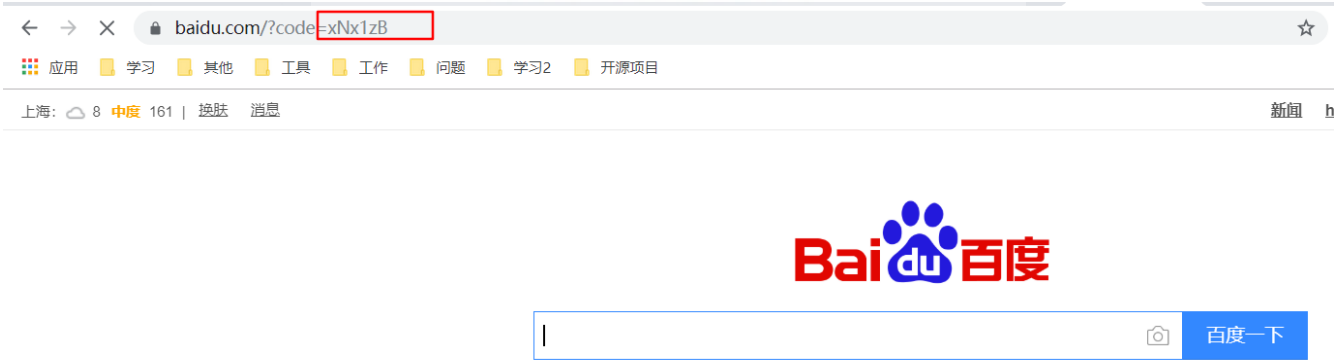
获取授权码

http://localhost:8080/oauth/authorize?response_type=code&client_id=admin&redirect_uri=http://www.baidu.com&scope=all

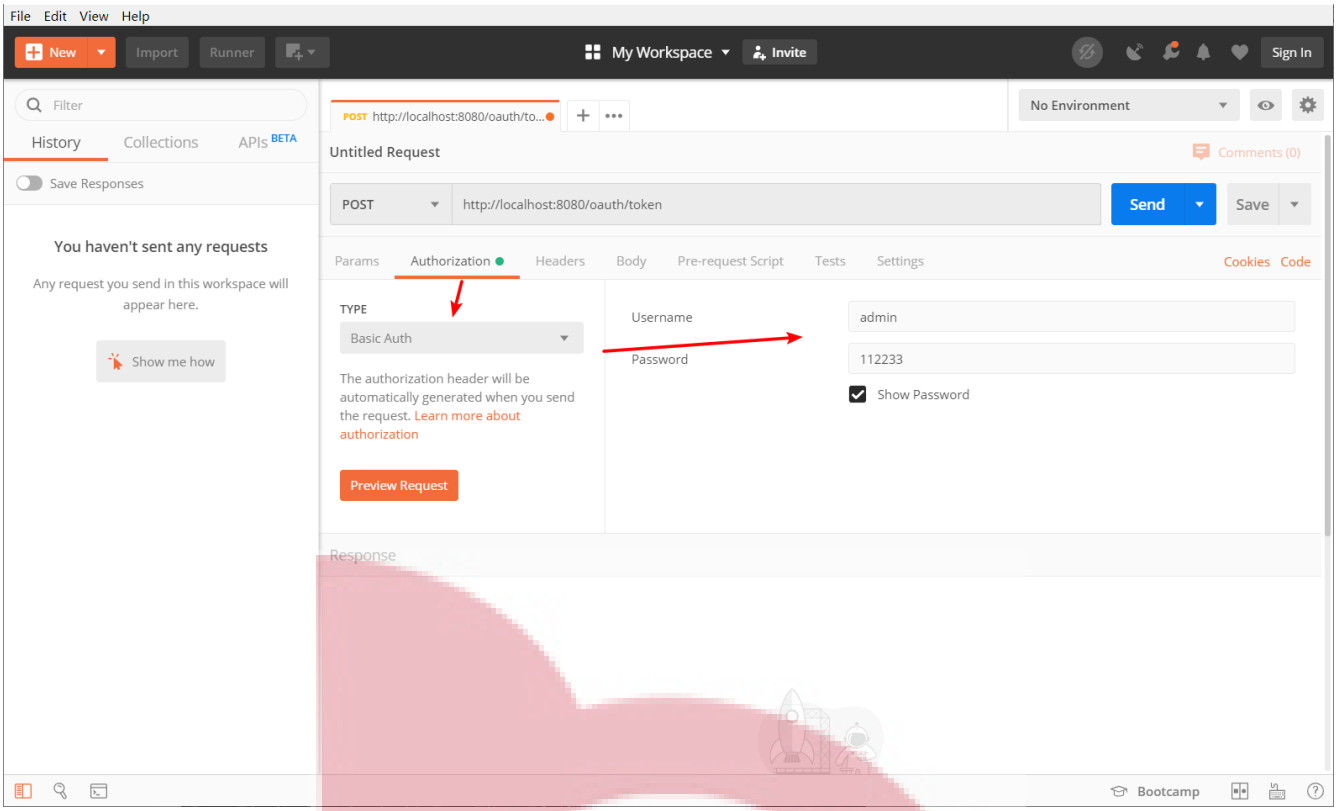


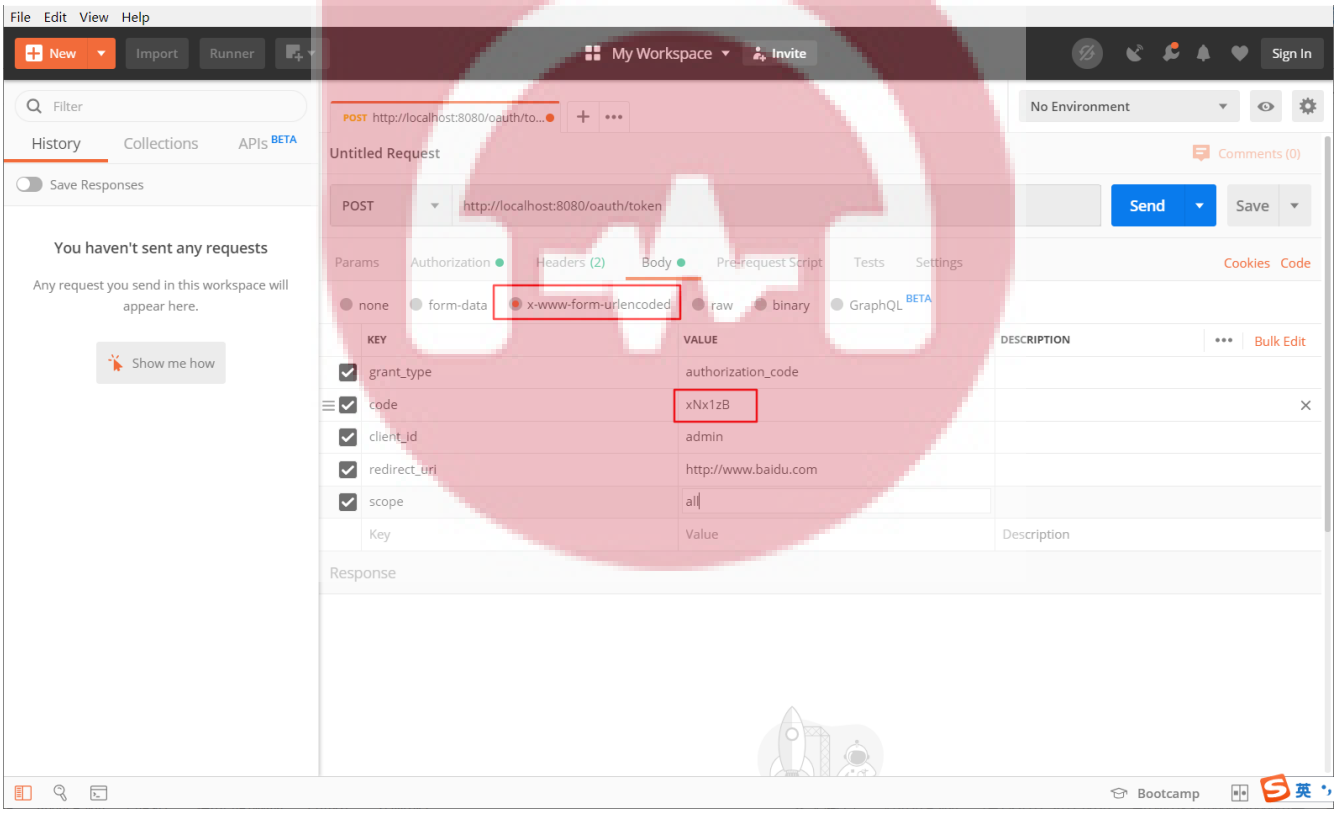
输入账户密码

点击授权获取授权码



根据授权码获取令牌（POST请求）

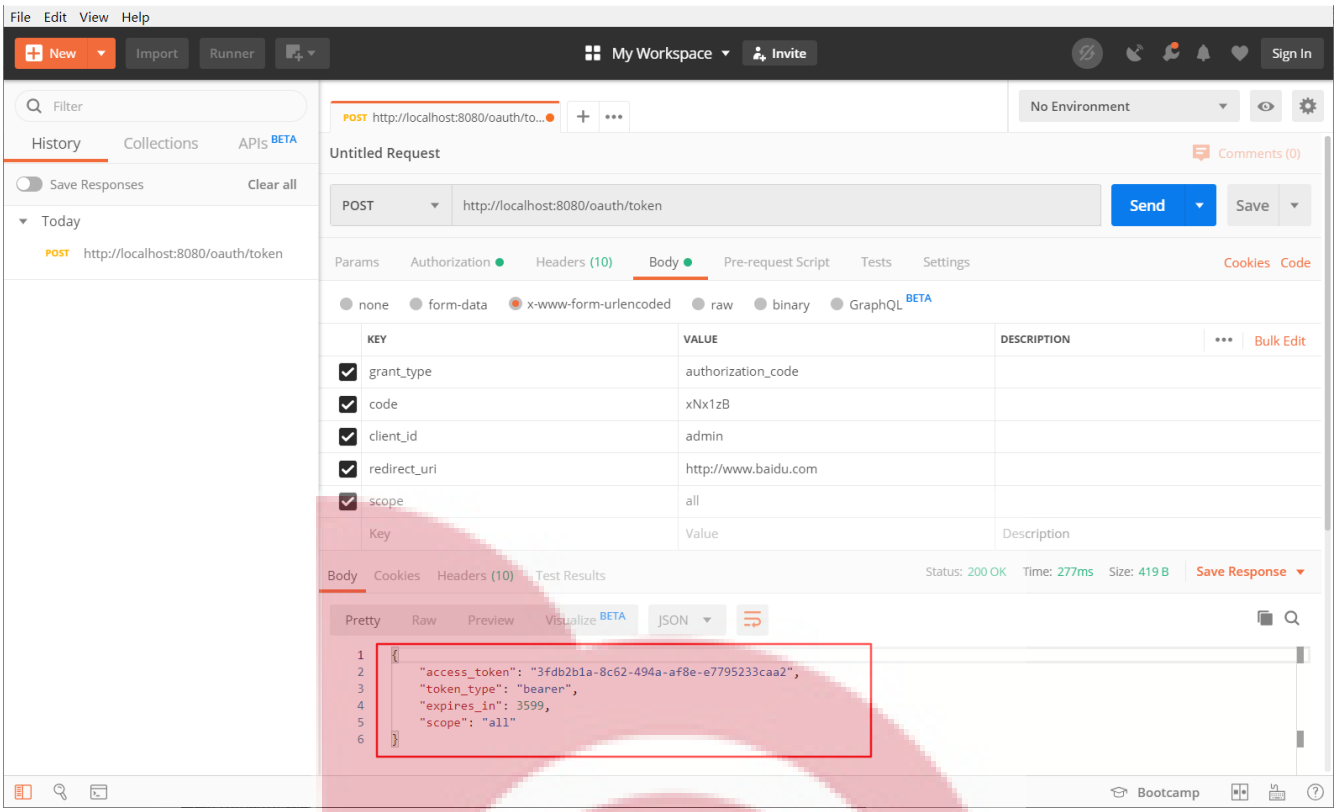




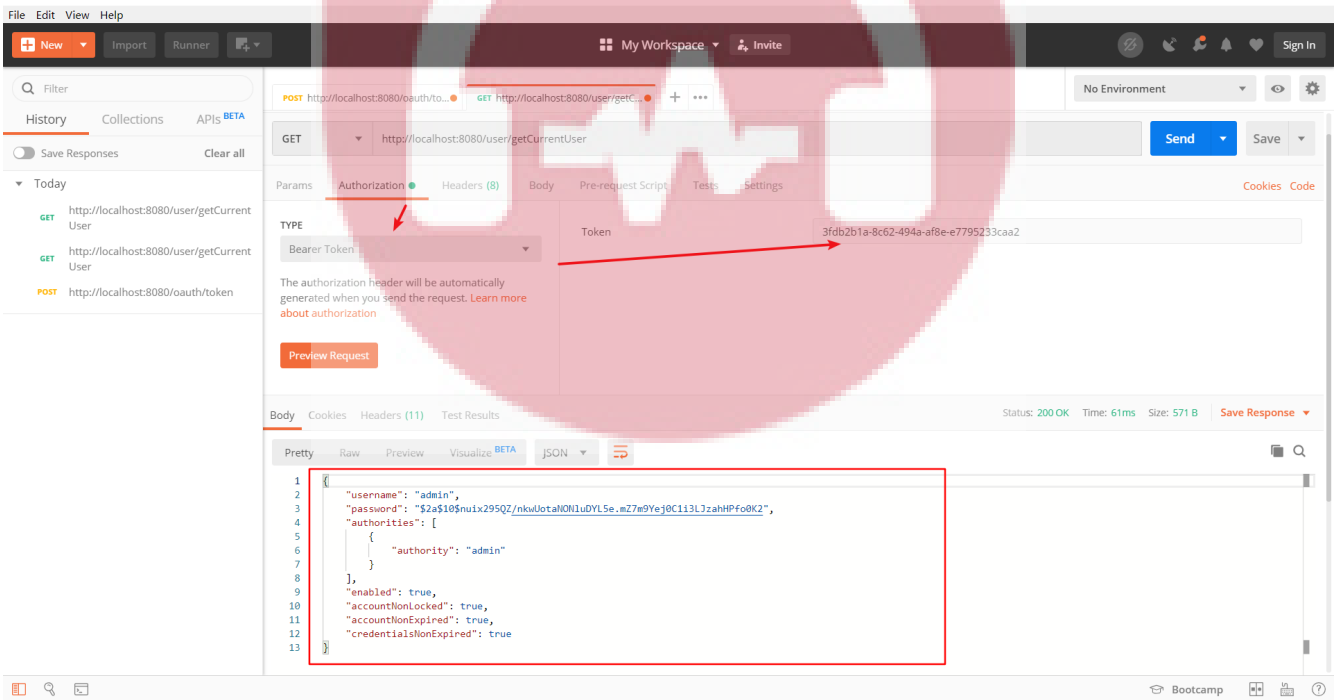
- `grant_type`：授权类型，填写authorization_code，表示授权码模式
- `code`：授权码，就是刚刚获取的授权码，注意：授权码只使用一次就无效了，需要重新申请。
- `client_id`:客户端标识
- `redirect_uri`：申请授权码时的跳转url，一定和申请授权码时用的redirect_uri一致。
- `scope`:授权范围。

认证失败服务端返回 401 Unauthorized

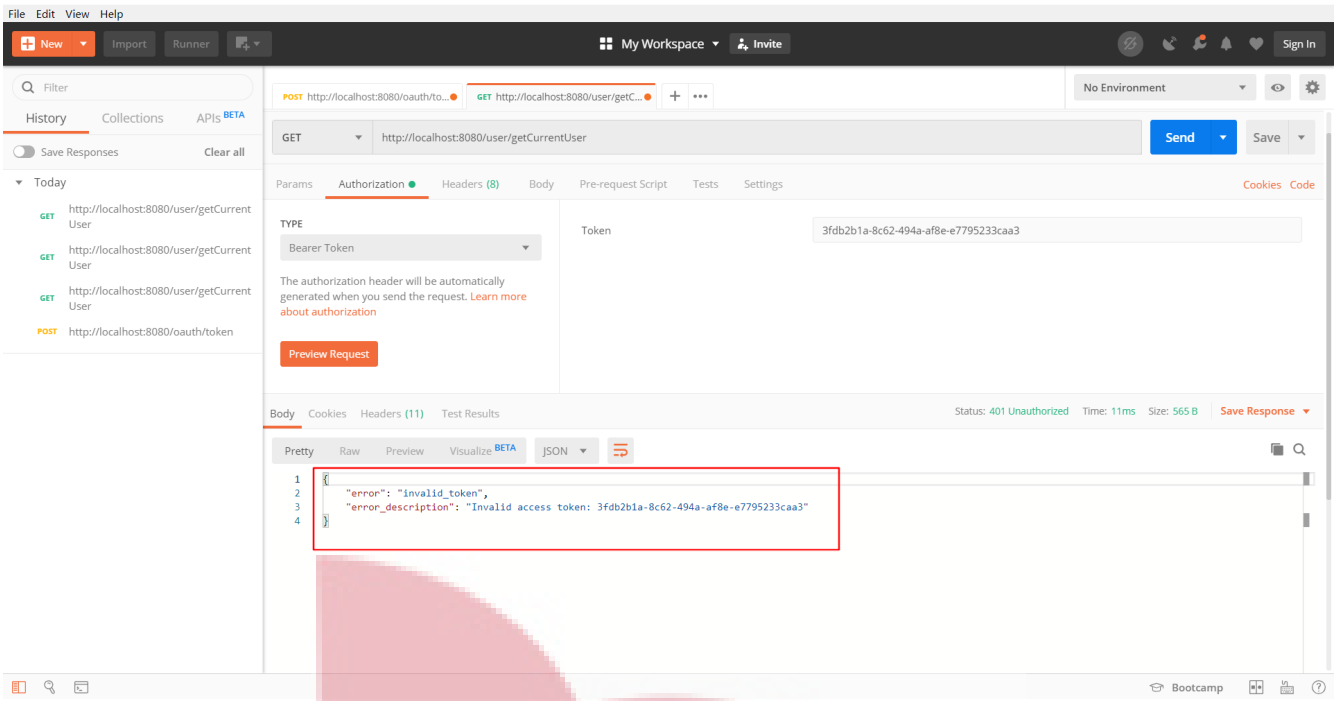
注意：此时无法请求到令牌，访问服务器会报错



根据token去资源服务器拿资源



如果修改token就会报错



Spring Security Oauth2 密码模式

在上面的代码中进行适当的修改即可

SecurityConfig.java

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

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
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.WebSecurityConfigurerAdapt
er;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

/**
 * Spring Security 配置类
 *
 * @author zhoubin
 */
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
```

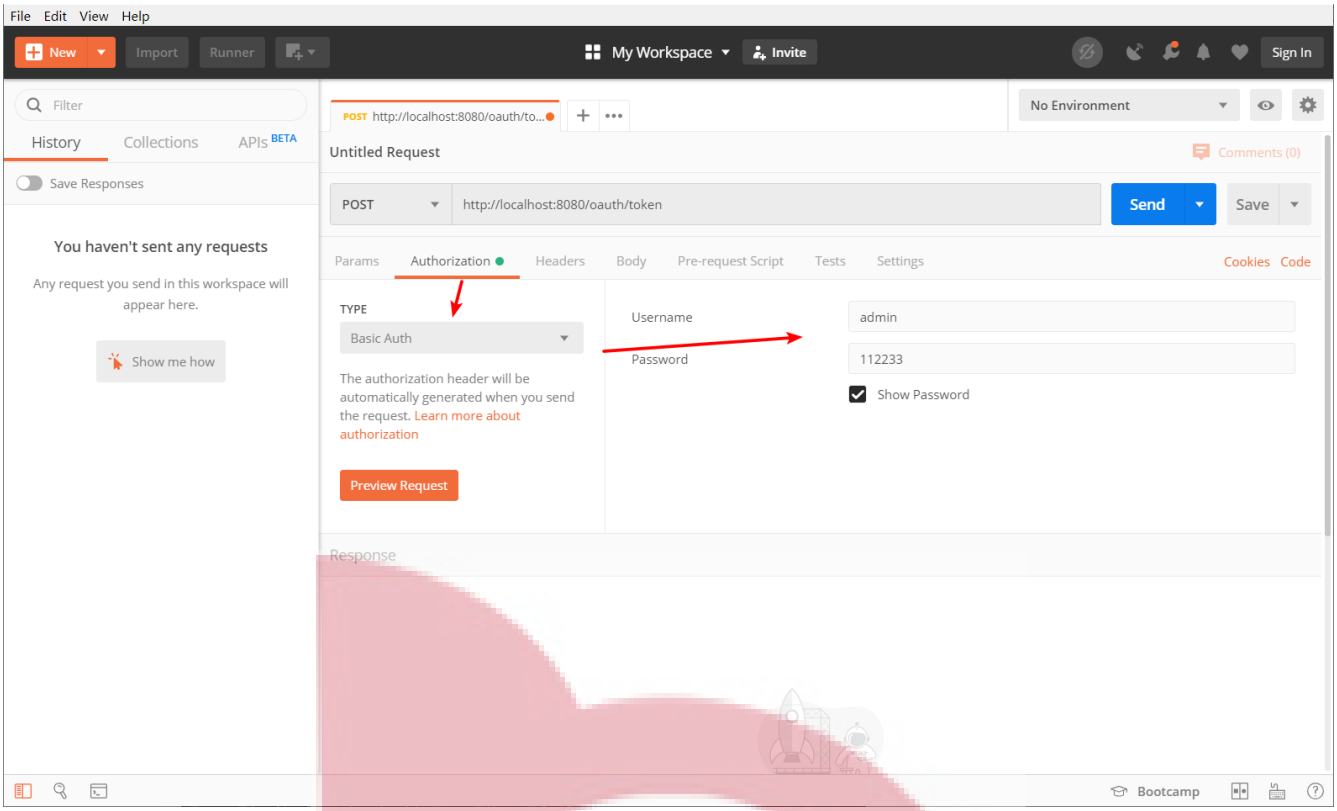
```
    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.csrf()
                .disable()
                .authorizeRequests()
                .antMatchers("/oauth/**", "/login/**", "/logout/**")
                .permitAll()
                .anyRequest()
                .authenticated()
                .and()
                .formLogin()
                .permitAll();
    }
}
```

AuthorizationServerConfig.java

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

import com.xxxx.springsecurityoauth2demo.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
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;

/**
 * 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;

    /**
     * 使用密码模式需要配置
     */
    @Override
    public void configure(AuthorizationServerEndpointsConfigurer endpoints) {
        endpoints.authenticationManager(authenticationManager)
                .userDetailsService(userService);
    }

    @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");
    }
}
```

测试：


File Edit View Help
New
Import
Runner
My Workspace
Invite
Sign In
Filter
History
Collections
APIs BETA
Save Responses
You haven't sent any requests
Any request you send in this workspace will appear here.
Show me how
POST http://localhost:8080/oauth/to...
No Environment
Untitled Request
Comments (0)
POST
http://localhost:8080/oauth/token
Send
Save
Params
Authorization
Headers
Body
Pre-request Script
Tests
Settings
Cookies
Code
TYPE
Basic Auth
The authorization header will be automatically generated when you send the request. Learn more about authorization
Preview Request
Username
admin
Password
112233
Show Password
Response
Bootcamp