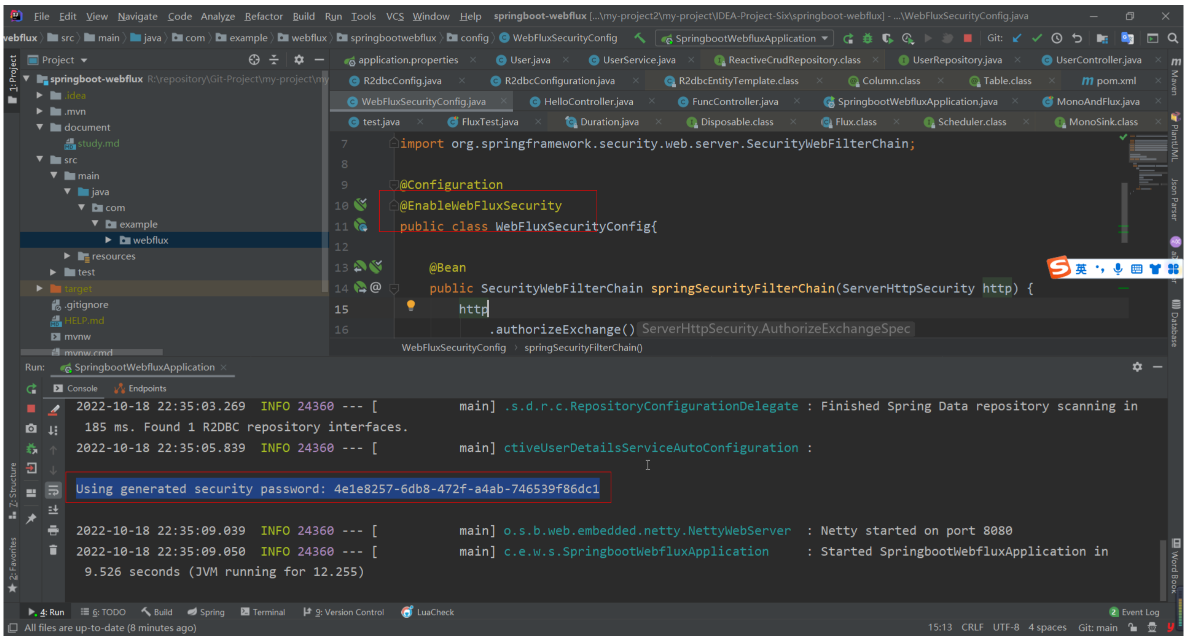
spring webflux security

1.认证的几种方式

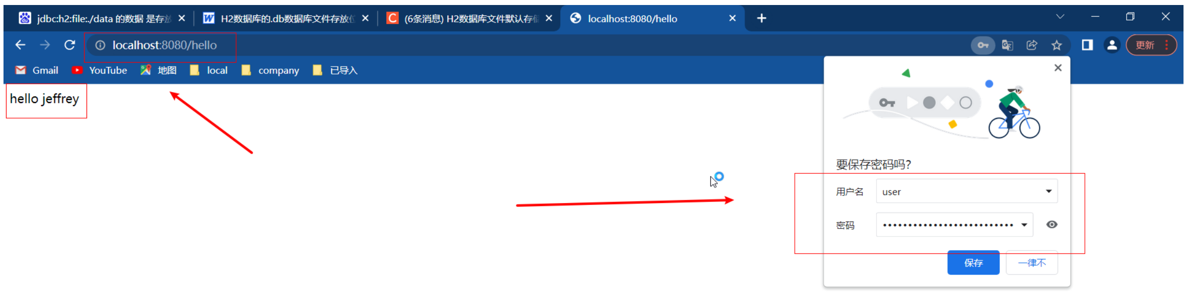
1.自带的

```
只加注解@EnableWebFluxSecurity时，默认的是用户名user,密码是启动时控制台打印出来的
Using generated security password: 4e1e8257-6db8-472f-a4ab-746539f86dc1

com.example.webflux.springbootwebflux.config.WebFluxSecurityConfig
```



```java
import org.springframework.security.web.server.SecurityWebFilterChain;

@Configuration
@EnableWebFluxSecurity
public class WebFluxSecurityConfig{

    @Bean
    public SecurityWebFilterChain springSecurityFilterChain(ServerHttpSecurity http) {
        http
            .authorizeExchange() ServerHttpSecurity.AuthorizeExchangeSpec
```

```
2022-10-18 22:35:03.269  INFO 24360 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in
 185 ms. Found 1 R2DBC repository interfaces.
2022-10-18 22:35:05.839  INFO 24360 --- [           main] ctiveUserDetailsServiceAutoConfiguration :

Using generated security password: 4e1e8257-6db8-472f-a4ab-746539f86dc1

2022-10-18 22:35:09.039  INFO 24360 --- [           main] o.s.b.web.embedded.netty.NettyWebServer  : Netty started on port 8080
2022-10-18 22:35:09.050  INFO 24360 --- [           main] c.e.w.s.SpringbootWebfluxApplication     : Started SpringbootWebfluxApplication in
 9.526 seconds (JVM running for 12.255)
```



localhost:8080/hello

hello jeffrey

要保存密码吗?

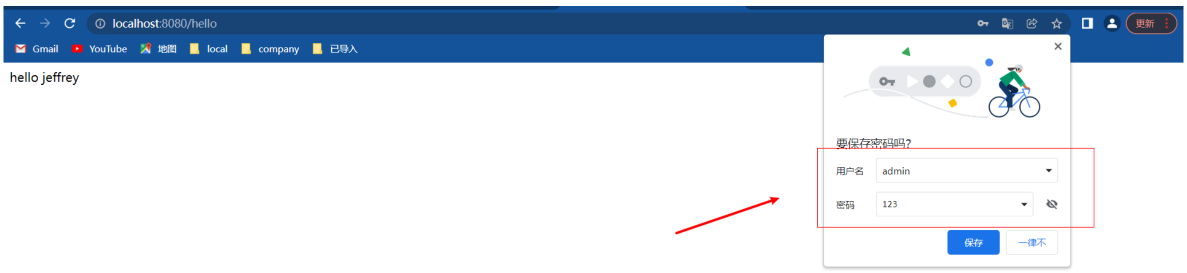
用户名: user

密码: ••••••••••••••••••••••••••••

2.内存密码

```
适合内存型的，不过实际开发中不会用这种

com.example.webflux.springbootwebflux.config.WebFluxSecurityConfigMem

```

```java
//@Configuration
//@EnableWebFluxSecurity
public class WebFluxSecurityConfigMem {

    @Bean
    public SecurityWebFilterChain springSecurityFilterChain(ServerHttpSecurity http) {
        http
            .authorizeExchange()
            //.pathMatchers("/loginPage").permitAll()  //无需进行权限过滤的请求路径
            .anyExchange().authenticated()
            .and()
            .httpBasic().and()
            .formLogin()
            //.loginPage("/loginPage")  //自定义的登陆页面
            ;
        return http.build();
    }

    @Bean
    public ReactiveUserDetailsService userDetailsService() {
        UserDetails build = User.withUsername("admin").password(passwordEncoder().encode("123")).authorities("admin").build();
        return new MapReactiveUserDetailsService(build);
    }

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


```



hello jeffrey

要保存密码吗?

用户名 admin

密码 123

3.数据库

```
实际用的是这个
com.example.webflux.springbootwebflux.config.db.WebFluxSecurityConfigDB
com.example.webflux.springbootwebflux.config.db.MyUserDetailsService
```

```java
@Service
public class MyUserDetailsService implements ReactiveUserDetailsService {
    @Resource
    private UserRepository userRepository;
    @Override
    public Mono<UserDetails> findByUsername(String username) {
        List<GrantedAuthority> auths = 
AuthorityUtils.commaSeparatedStringToAuthorityList("ROLE_sale1,admin");
        org.springframework.security.core.userdetails.User user = new 
org.springframework.security.core.userdetails.User(username, new 
BCryptPasswordEncoder().encode("123456"), auths);
        return Mono.just(user);
    }
}
```

```java
@Configuration
@EnableWebFluxSecurity
public class WebFluxSecurityConfigDB {
    @Autowired
    private MyUserDetailsService myUserDetailsService;

    @Bean
    public SecurityWebFilterChain 
springSecurityFilterChain(ServerHttpSecurity http) {
        http
            .authorizeExchange()
            //.pathMatchers("/loginPage").permitAll()  //无需进行权限过滤的请求
路径
            .anyExchange().authenticated()
            .and()
            .httpBasic().and()
            .formLogin()
            //.loginPage("/loginPage")  //自定义的登陆页面
            ;
        return http.build();
    }

    @Bean
    public ReactiveAuthenticationManager reactiveAuthenticationManager() {
        UserDetailsRepositoryReactiveAuthenticationManager 
authenticationManager = new 
UserDetailsRepositoryReactiveAuthenticationManager(myUserDetailsService);
        authenticationManager.setPasswordEncoder(passwordEncoder());
        return authenticationManager;
    }

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