

# license

2config

## FilterConfig.java

,

```
package cn.vppark.whdev.license_test.config;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * Filter
 *
 * @author Mark sunlightcs@gmail.com
 */
@Configuration
public class FilterConfig {

    @Value("${app.license:}")
    private String license;

    @Bean
    public FilterRegistrationBean licenseFilterRegistration() {
        FilterRegistrationBean registration = new FilterRegistrationBean();
        registration.setFilter(new LicenseFilter(license));
        registration.addUrlPatterns("/");
        registration.setName("licenseFilter");
        registration.setOrder(Integer.MAX_VALUE);
        return registration;
    }
}
```

# LicenseFilter.java

```
package cn.vppark.whdev.license_test.config;

import com.alibaba.fastjson.JSONObject;
import org.apache.commons.io.IOUtils;
import org.springframework.beans.factory.annotation.Value;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.servlet.*;
import javax.servlet.http.HttpServletResponse;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.security.InvalidKeyException;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;
import java.util.Date;

/**
 * @Description: 许可证过滤器
 * @date: 2024-04-09 12:39:48
 * @Copyright:
 */
public class LicenseFilter implements Filter {

    private String license;
```

```

/**
 * []
 */
private final String publicKey = "your public key";

public LicenseFilter(String license) {
    this.license = license;
}

@Override
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
throws IOException, ServletException {
    //[]
    HttpServletResponse httpServletResponse = (HttpServletResponse) response;
    httpServletResponse.setStatus(HttpServletResponse.SC_BAD_REQUEST);
    httpServletResponse.setCharacterEncoding("UTF-8");
    httpServletResponse.setContentType("text/html; charset=UTF-8");

    try {
        if (this.license == null || license.isEmpty()) {
            // []target[]
            this.license = IOUtils.toString(new FileInputStream(getLicensePath()),
StandardCharsets.UTF_8);
        }
        //[]
        String licenseContent = verifyLicense(this.license);
        JSONObject json = JSONObject.parseObject(licenseContent);
        //[]
        Date expire = json.getDate("expire");
        httpServletResponse.setHeader("license-expire", expire.toString());
        //[]
        if (expire.getTime() < System.currentTimeMillis()) {
            httpServletResponse.getWriter().write(" []");
            return;
        }
        httpServletResponse.setStatus(HttpServletResponse.SC_OK);
    } catch (FileNotFoundException e) {
        httpServletResponse.getWriter().write(" []");
        e.printStackTrace();
    }
}

```

```

        return;
    } catch (Exception e) {
        httpServletResponse.getWriter().write(" []");
        e.printStackTrace();
        return;
    }
    chain.doFilter(request, response);
}

/**
 * @throws
 * @Description: []
 * @param: licenseContent []
 */
public String verifyLicense(String licenseContent) throws NoSuchAlgorithmException,
InvalidKeySpecException, InvalidKeyException, NoSuchPaddingException,
IllegalBlockSizeException, BadPaddingException {
    Cipher cipher = Cipher.getInstance("RSA");
    cipher.init(Cipher.DECRYPT_MODE, KeyFactory.getInstance("RSA").generatePublic(new
X509EncodedKeySpec(Base64.getDecoder().decode(publicKey))));
    byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(licenseContent));
    String decryptedString = new String(decryptedBytes);
    // []
    System.out.println("licenseContent: " + decryptedString);
    return decryptedString;
}

/**
 * @throws
 * @Description: []license[]
 */
private String getLicensePath() throws URISyntaxException {

    URL classUrl = this.getClass().getProtectionDomain().getCodeSource().getLocation();
    URI uri = classUrl.toURI();
    Path path;
    if (uri.getScheme().equals("jar")) {
        // []JAR URL[]JAR[]
        String jarPath = uri.getSchemeSpecificPart();
        jarPath = jarPath.substring(0, jarPath.indexOf('!'));
        path = Paths.get(new URI(jarPath)).getParent();
    }
}

```

```
} else {  
    // 获取JAR URL  
    path = Paths.get(uri);  
}  
String licensePath = path.resolve("license/license").toString();  
// 打印  
System.out.println("licensePath: " + licensePath);  
return licensePath;  
}  
  
}
```

---

##2

11 2024 04:26:52

11 2024 04:30:32