

웹코드보안 프로젝트 전자봉투 시스템

20191007 이희원



목차

1. 전자봉투 설계 과정
2. 코드 리뷰
3. 프로젝트 실행



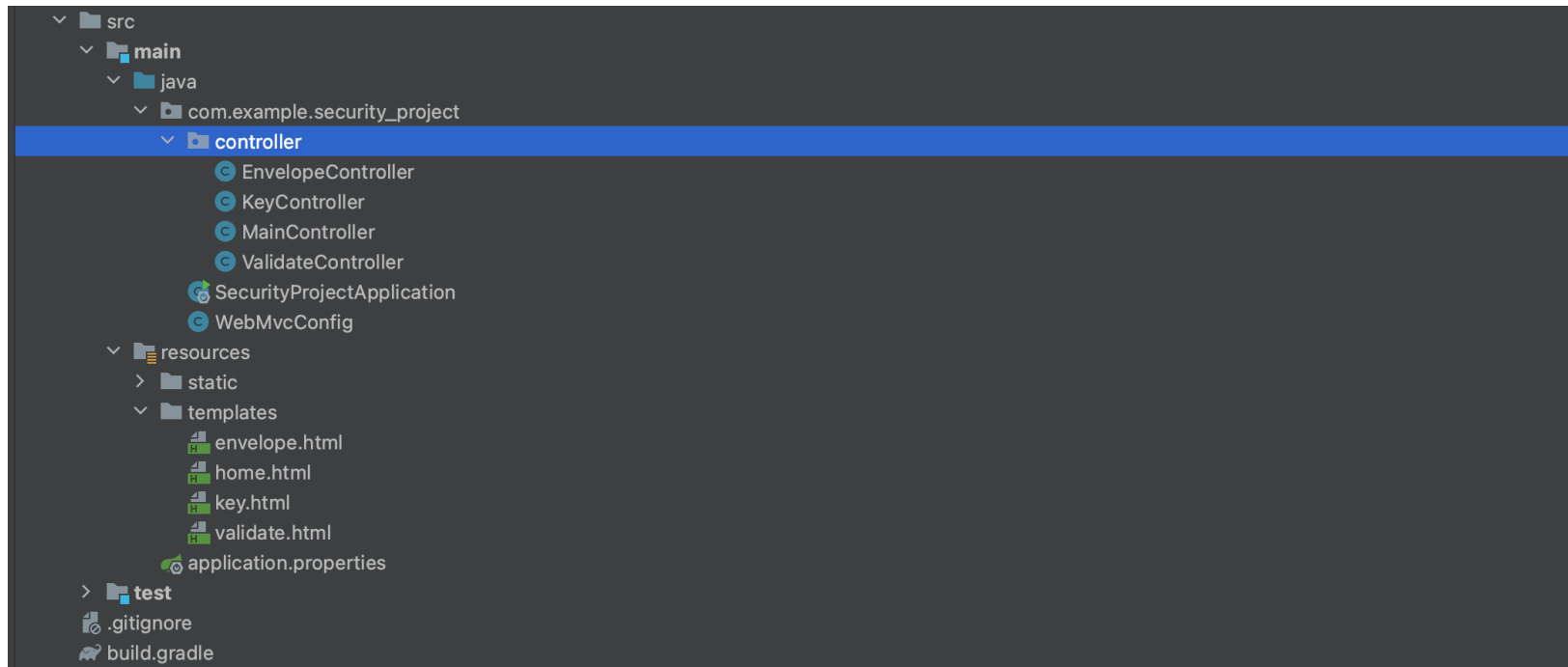
전자봉투 설계

1. 코드 설계
2. 인터페이스 설계



코드 설계

• 클래스 설계



코드 설계

- 클래스 설계(KeyController)

```
no usages 1 Leehuiwon +1
@PostMapping("/key")
public String generateKeys(Model model, @RequestParam("secretKey") String secretKeyFile,
                                   @RequestParam("publicKey") String publicKeyFile,
                                   @RequestParam("privateKey") String privateKeyFile){

    // 대칭키 생성하기
    try {
        KeyGenerator keyGenerator = KeyGenerator.getInstance("DES");
        keyGenerator.init(keysize: 56);
        Key secretKey = keyGenerator.generateKey();
        // 대칭키 저장하기
        saveKeyToFile(secretKey, secretKeyFile);
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }

    // 공개키, 개인키 생성하기
    try {
        // KeyPairGenerator를 사용하여 키 생성
        KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("RSA");
        keyPairGen.initialize(keysize: 1024);
        KeyPair keyPair = keyPairGen.generateKeyPair();

        PublicKey publicKey = keyPair.getPublic();
        PrivateKey privateKey = keyPair.getPrivate();

        // 개인키 저장
        saveKeyToFile(privateKey, privateKeyFile);

        // 공개키 저장
        saveKeyToFile(publicKey, publicKeyFile);
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }

    model.addAttribute(attributeName: "generatedKeys", attributeValue: true);
    model.addAttribute(attributeName: "secretKeyFile", secretKeyFile);
    model.addAttribute(attributeName: "publicKeyFile", publicKeyFile);
    model.addAttribute(attributeName: "privateKeyFile", privateKeyFile);

    return "home";
}
```



코드 설계

- 클래스 설계(EnvelopeController)

```
@Controller
public class EnvelopeController {

    // @RequestMapping("envelope")
    // public String Envelope(){
    //     return "envelope";
    // }

    no usages  LeeHuiwon
    @PostMapping(Consumes = {"envelope"})
    public String EnvelopeDocument(@RequestParam("data") String data,
                                   @RequestParam("secretKey") String secretKeyFileName,
                                   @RequestParam("privateKey") String privateKeyFileName,
                                   @RequestParam("envelopeFileName") String envelopeFileName,
                                   Model model){

        String keyAlgorithm = "RSA";
        String signAlgorithm = "SHA256withRSA";

        KeyPairGenerator keyPairGen;
        KeyGenerator keyGenerator;

        try {
            keyPairGen = KeyPairGenerator.getInstance(keyAlgorithm);
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }

        keyPairGen.initialize(keySize: 1024);
        KeyPair keyPair = keyPairGen.generateKeyPair();

        PublicKey publicKey = keyPair.getPublic();
        PrivateKey privateKey = keyPair.getPrivate();

        // secretKey 객체 생성
        try {
            keyGenerator = KeyGenerator.getInstance(keyAlgorithm: "DES");
            keyGenerator.init(keySize: 56);
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }

        Key secretKey;

        //원본 데이터 바이트 배열에 저장
        byte[] bufferData = data.getBytes();
    }
}
```



코드 설계

- 클래스 설계(ValidateController)

```
no usages 1 Leehuiwon
@Controller
public class ValidateController {
    // @RequestMapping("validate")
    // public String Validate(){
    //     return "/validate";
    // }
    // }

    no usages 1 Leehuiwon
    @PostMapping("@"/validate")
    public String ValidateDocument(@RequestParam("data") String data,
                                   @RequestParam("secretKey") String secretKeyFileName,
                                   @RequestParam("publicKey") String publicKeyFileName,
                                   @RequestParam("validateFileName") String validateFileName,
                                   Model model) throws NoSuchAlgorithmException, InvalidKeyException, SignatureException, IOException {

        String keyAlgorithm = "RSA";
        String signAlgorithm = "SHA256withRSA";
        String signatureName = "sig.bin";

        KeyPairGenerator keyPairGen;
        KeyGenerator keyGenerator;
        try {
            keyPairGen = KeyPairGenerator.getInstance( algorithm: "RSA");
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
        keyPairGen.initialize( keysize: 1024);
        KeyPair keyPair = keyPairGen.generateKeyPair();

        PublicKey publicKey;
        PrivateKey privateKey = keyPair.getPrivate();

        // secretKey 객체 생성
        try {
            keyGenerator = KeyGenerator.getInstance( algorithm: "DES");
            keyGenerator.init( keysize: 56);

        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
        Key secretKey;




        // bufferData => data의 byte타입으로 변환
        byte[] bufferData = data.getBytes();
        // signature 생성하기
        Signature signature;
```



인터페이스 설계

시스템 메인

전자봉투 서명 시스템



키 생성하기

전자봉투 만들기

전자봉투 검증하기

키 생성

키 생성하기

대칭키 입력하세요

대칭키 생성

비밀키를 입력하세요

비밀키 생성

공개키를 입력하세요

공개키 생성

홈으로

전자봉투 생성

전자봉투 만들기

100byte 내로 입력하세요

텍스트 입력

비밀키 파일을 입력하세요

비밀키 입력

대칭키 파일을 입력하세요

대칭키 입력

생성할 파일이름을 입력하세요

파일 이름

전자봉투 생성

홈으로

전자봉투 검증

전자봉투 검증하기

100byte 내로 입력하세요

텍스트 입력

공개키 파일을 입력하세요

공개키 입력

대칭키 파일을 입력하세요

대칭키 입력

검증할 파일이름을 입력하세요

검증

홈으로



스프링부트 프로젝트의 한계

- validate를 동작 시 검증 결과

The screenshot displays a web application interface for digital signature verification. The page title is "전자봉투 검증" (Digital Envelope Verification). It contains four input fields: "Dongduk" (Text input), "public.key" (Public key input), "secret.key" (Secret key input), and "envelope.bin" (Envelope file input). To the right of each input field is a corresponding button: "텍스트 입력" (Text input), "공개키 입력" (Public key input), "대칭키 입력" (Symmetric key input), and "전자봉투 검증" (Digital envelope verification). A "홈으로" (Home) button is located at the bottom center.

A modal dialog box is open, displaying the message: "localhost:8005 내용: 전자봉투 검증 결과: undefined". A blue "확인" (Confirm) button is visible.

The Chrome DevTools console is open, showing the source code for the "validate" script. The code includes a jQuery library reference and a function named "validateEnvelope()". The function retrieves the values of the "data", "publicKey", "secretKey", and "validateFileName" inputs and sends a POST request to the "/validate" endpoint.

```
75 </div>
76 </div>
77 <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
78 <script>
79   function redirectToHome() {
80     window.location.href = "home"; // MainController의 h
81   }
82
83   function validateEnvelope() {
84     const data = document.getElementsByName("data")[0].val();
85     const publicKey = document.getElementsByName("publicKey")[0].val();
86     const secretKey = document.getElementsByName("secretKey")[0].val();
87     const validateFileName = document.getElementsByName("validateFileName")[0].val();
88     $(document).ready(function() {
89       $(".button").click(function() {
90         const data = $("input[name='data']").val();
91         const publicKey = $("input[name='publicKey']").val();
92         const secretKey = $("input[name='secretKey']").val();
93         const validateFileName = $("input[name='validateFileName']").val();
94
95         $.ajax({
96           type: "POST",
97           url: "/validate",
98           data: {
```



코드 리뷰

1. Code Analysis



33. Prefer user-defined exceptions over more general exception types

Before

```
// ..... (코드리뷰-함부로 예외처리 어찌구)
// ..... // 2. 파일이 이미 존재하는 경우 예외 처리
// ..... if (file.exists()) {
// .....     System.out.println("파일이 이미 존재합니다.");
// .....     return;
// ..... }
// ..... // 3. 파일에 대한 쓰기 권한이 없는 경우 예외 처리
// ..... if (!file.getParentFile().canWrite()) {
// .....     System.out.println("파일에 대한 쓰기 권한이 없습니다.");
// .....     return;
// ..... }
```

After

```
// ..... 코드리뷰 : throws 대신 catch로 잡기
// ..... try (FileOutputStream fileOutputStream = new FileOutputStream(fileName);
// .....     ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream)) {
// .....     objectOutputStream.writeObject(key);
// ..... } catch (IOException e) {
// .....     e.printStackTrace();
// ..... }
// ..... }
```



39. Use meaningful symbolic constants to represent literal values in program logic

Before

```
// ...private static String main = "/home";  
// ...private static String key = "/key";  
// ...private static String envelope = "/envelope";  
// ...private static String validate = "/validate";
```

After

```
1 usage  
...private static final String HOME_FORM = "/home";  
1 usage  
...private static final String KEY_FORM = "/key";  
1 usage  
...private static final String ENVELOPE_FORM = "/envelope";  
1 usage  
...private static final String VALIDATE_FORM = "/validate";
```



프로젝트 실행

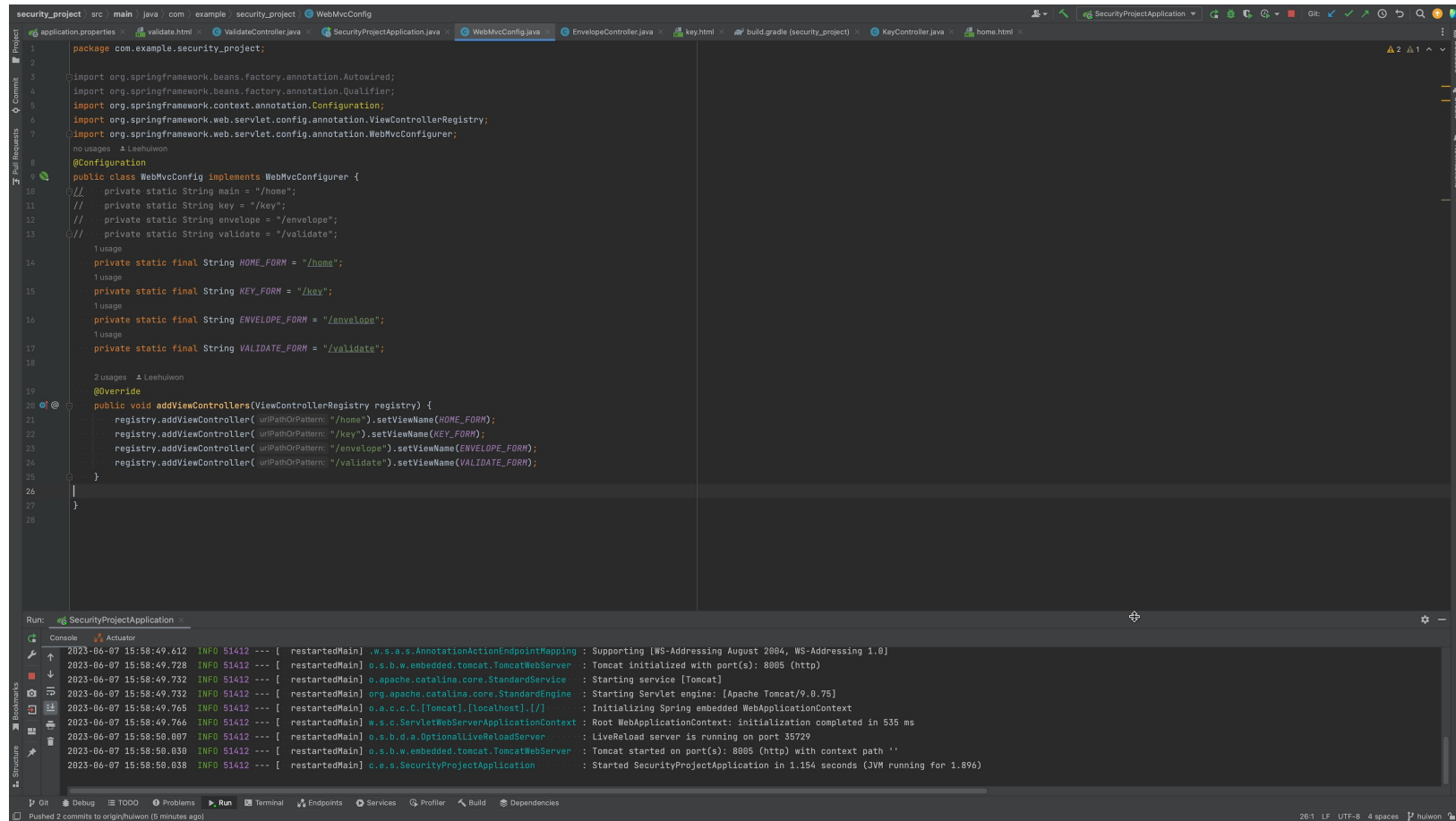


프로젝트 실행 순서

1. 키 생성(가정)
2. 전자봉투 생성
3. 전자봉투 검증



프로젝트 실행



```
security_project src main java com example security_project WebMvcConfig
1 package com.example.security_project;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.beans.factory.annotation.Qualifier;
5 import org.springframework.context.annotation.Configuration;
6 import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
7 import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
8
9 no usages 1 Leehuiwon
10 @Configuration
11 public class WebMvcConfig implements WebMvcConfigurer {
12     // private static String main = "/home";
13     // private static String key = "/key";
14     // private static String envelope = "/envelope";
15     // private static String validate = "/validate";
16     1 usage
17     private static final String HOME_FORM = "/home";
18     1 usage
19     private static final String KEY_FORM = "/key";
20     1 usage
21     private static final String ENVELOPE_FORM = "/envelope";
22     1 usage
23     private static final String VALIDATE_FORM = "/validate";
24     2 usages 1 Leehuiwon
25
26     @Override
27     public void addViewControllers(ViewControllerRegistry registry) {
28         registry.addViewController( @RequestMapping: "/home" ).setViewName(HOME_FORM);
29         registry.addViewController( @RequestMapping: "/key" ).setViewName(KEY_FORM);
30         registry.addViewController( @RequestMapping: "/envelope" ).setViewName(ENVELOPE_FORM);
31         registry.addViewController( @RequestMapping: "/validate" ).setViewName(VALIDATE_FORM);
32     }
33
34 }
35
36 }
```

```
Run: SecurityProjectApplication
Console
2023-06-07 15:58:49.612 INFO 51412 --- [ restartedMain] .w.s.a.AnnotationActionEndpointMapping : Supporting [WS-Addressing August 2004, WS-Addressing 1.0]
2023-06-07 15:58:49.728 INFO 51412 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-06-07 15:58:49.732 INFO 51412 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-06-07 15:58:49.732 INFO 51412 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.75]
2023-06-07 15:58:49.765 INFO 51412 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-06-07 15:58:49.766 INFO 51412 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 535 ms
2023-06-07 15:58:50.007 INFO 51412 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2023-06-07 15:58:50.030 INFO 51412 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2023-06-07 15:58:50.038 INFO 51412 --- [ restartedMain] c.e.s.SecurityProjectApplication : Started SecurityProjectApplication in 1.154 seconds (JVM running for 1.896)
```



감사합니다

