티킷 데이터 암호화/복호화v2

작성자:이지훈

Jwt, aes방식으로 2중 암,복호화 적용(서버단 복호화시 데이터를 유에 저장하기 위해 객체 생성해서 사용)

Client 코드 변경 순서

1. Common/Secret.js 를 import 해줍니다.

import { salt, key, f\_decodeData, f\_encodeUserPw, f\_encodeJsonData } from "../common/secret";

1. Server로 보낼 데이터를 json 타입으로 생성후 f\_encodeJsonData를 사용하여 인코딩합니다.

var jsonData = '{}';

jsonData = {

"userId" : this.state.userId,

"userPw" : encodePw

}

const encodeJsonData = f\_encodeJsonData(jsonData);

1. 데이터 타입을 json으로 통일 하기 위해 기존 axios get방식을 post 방식으로 변경합니다.

.post(API\_Login, encodeJsonData)

1. Axios 에서 response 받은 데이터를 f\_decodeData를 사용하여 디코딩 합니다.

.then(response => {

/\*\*ADD jihoon decode responseData \*/

var jsonDataDecode=f\_decodeData(response.data.data);

1. Decode한 Json 타입 데이터를 set해준후 필요한 param을 추출해 사용합니다.

this.setState({ ApiResults: response.data, ApiDatas:jsonDataDecode });

Server 코드 변경 순서

1. Controller 에서 @RequsetMapping (value=”/tikit\_url” , method=RequestMethod.POST ) 로 바꿔줍니다.

@RequestMapping(value="/", method=RequestMethod.***POST***)

1. Controller 에서 requestParam 을 requestBody String bodyParam 형태로 바꿔줍니다.

**private** ResponseEntity<Map<String, Object>> LoginValidation(@RequestBody String bodyParam) **throws** Exception {

1. BodyParam 을 bodyParamConvert 를 사용하여 json타입으로 디코딩 해줍니다.

BodyParamConvert mBodyParamConvert = **new** BodyParamConvert();

JsonElement mJsonElement = mBodyParamConvert.BodyParamConvert(bodyParam);

1. 디코딩된 데이터에서 필요한 파람을 추출해 mapping 합니다.

String userId = mJsonElement.getAsJsonObject().get("userId").getAsString();

String userPw = mJsonElement.getAsJsonObject().get("userPw").getAsString();

Cf) String -> getAsString();

int -> getAsInt();

데이터 암호화 방식

Userpw는 sha-1 로 단방향 암호화.

Original Data를 aes-128-ecb 방식으로 암호화후 jws 토큰으로 전송합니다

Server->client, client->server 동일한 방식.

Client /secret.js

function f\_decodeData(responseData)

response data를 디코딩하여 json타입으로 return 합니다.

ResponseData->Jws decode -> aes-128-ecb decode -> oriData

unction f\_encodeUserPw(oriUserPw)

userPw 를 sha-1 방식으로 암호화 합니다. 사용자의 oriPw는 알수없고 암호화된 데이터로만 DB와 비교합니다.

function f\_encodeJsonData(jsonData)

json 타입의 데이터를 인코딩하여 String 타입으로 return 합니다.

OriData -> aes-128-ecb encode -> jws encode -> encodedStringData

Server apiIF.java

String aesEncodeData = **new** String(Base64.*encodeBase64*(cipher.doFinal(jsonData.getBytes())));

aes-128-ecb방식으로 암호화합니다.

JWSObject jwsObject = **new** JWSObject(**new** JWSHeader(JWSAlgorithm.***HS256***), **new** Payload(aesEncodeData));

aes-128-ecb방식으로 암호화된 데이터를 jws토큰 으로 변환 후 전송합니다.

Client 객체 방식

f\_contentRemoveApiCall() {

if (confirm("삭제하시겠습니까?")) {

/\*\*ADD jihoon create json data \*/

const jsonData = {

contentDeleteKeys: this.state.clickContentNumberList.map(i => {

return {

memb\_id: this.props.userId,

conts\_no: i

};

})

};

console.log("content delete test"+jsonData.toString());

/\*\*ADD jihoon json data encode\*/

const encodeJsonData = f\_encodeJsonData(jsonData);

axios

.post(API\_ContentManagement\_DELETE\_CONTENT, encodeJsonData)

.then(response => {

this.f\_getContentList();

this.setState({ clickContentNumberList: [] });

})

.catch(error => {

console.log(error);

});

}

/\*\*

\* 컨텐츠 삭제하기

\* @param contentKeys 컨텐츠 정보

\* @return Response Success 알림

\* @throws Exception Response Fail 알림

\*/

/\*\*ADD jihoon \*/

@RequestMapping(value = "/delete-contents", method = RequestMethod.POST)

public ResponseEntity<Map<String, Object>> deleteContents(@RequestBody String bodyParam) throws Exception {

ApiIF apiIF = new ApiIF();

BodyParamConvert mBodyParamConvert = new BodyParamConvert();

JsonElement mJsonElement = mBodyParamConvert.BodyParamConvert(bodyParam);

ContentDeleteKeysVO contentKeys = new ContentDeleteKeysVO();

ObjectParser mObjectParser = new ObjectParser();

contentKeys = mObjectParser.ContentDeleteKeysVOParser(mJsonElement.getAsJsonObject());

try {

contentService.deleteContents(contentKeys);

apiIF.setResultCode(ApiIF.RESPONSE\_CODE\_SUCCESS);

apiIF.setResultMsg(ApiIF.RESPONSE\_MSG\_SUCCESS);

} catch (Exception e) {

apiIF.setResultCode(ApiIF.RESPONSE\_CODE\_FAIL);

apiIF.setResultMsg(ApiIF.RESPONSE\_MSG\_FAIL);

e.printStackTrace();

}

return apiIF.toResponseEntiry();

}

}