오르빗 프로젝트 데이터 암호화/복호화 v.1

작성자 : 이지훈

Jwt 를 사용한 1차 인,디코딩

Client common/secret.js

function f\_encodeUserPw(oriUserPw){

var encodePw = bcrypt.hashSync(oriUserPw, salt);

return encodePw;

사용자가 입력한 pw bcrypt형식으로 암호화. Salt는 salt키값으로 client에 저장합니다.

export function f\_encodeJsonData(jsonData){

const encodejsonData = jws.sign({

header: { alg: 'HS256' },

payload: jsonData,

secret: key,

});

return encodejsonData;

client에서 server로 넘기는json 데이타 jws형식으로 암호화후 return 해줍니다..

export function f\_decodeData(responseData){

var oriData=jws.decode(responseData);

var jsonData= JSON.parse(oriData.payload);

return jsonData;

server에서 response 받은 데이터 jws.decode로 복호화후 데이타 return.

Client LoginContainer.js

var jsonData = '{}';

jsonData = {

"userId" : this.state.userId,

"userPw" : encodePw

}

axios

.post(API\_Login, encodeJsonData)/\*\*post 형식으로 변경 \*/

client에서 로그인시 server로 넘겨주는 파라미터를 json 타입으로 변경

Server cp.orbit.common/BodyParamConvert

**public** JsonElement BodyParamConvert(String bodyparam) {

String convertJson = "";

**try** {

**final** JWSObject jwsObject = JWSObject.*parse*(bodyparam);

convertJson = jwsObject.getPayload().toJSONObject().toJSONString();

} **catch** (Exception e) {

e.printStackTrace();

}

JsonParser parser = **new** JsonParser();

JsonElement mJsomelement = parser.parse(convertJson);

**return** mJsomelement;

}

Client 에서 받아온 암호화된 json데이터를 복호화후 json타입으로 리턴

리턴된 json데이타에서 업무에 필요한 파라미터 추출후 사용

Server cp.orbit.api/ApiIF.java

Gson gson = **new** Gson();

String jsonData = gson.toJson(**this**.data);

Gson을 사용하여 object타입 이였던 기존의 데이타를 json타입으로 바꿈

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

**try** {

jwsObject.sign(**new** MACSigner(key));

String encodeData = jwsObject.serialize();

bodys.put("data", encodeData);

json타입 데이터를 jws형식으로 암호화 합니다.

key는 key.properties에서 jwskey 가져옵니다.

암호화된 데이터를 body에 put합니다.

Client -> Server

1. 기존의 get방식으로 파라미터를 보내던 것을 post방식으로 json타입의 데이터를 보내기 때문에 json타입의 데이터를 만들어 줘야 합니다. Json 타입데이타 생성후 f\_encodeJsonData 함수를 사용하여 암호화후 server로 데이터를 보내줍니다.

2. Server로 보낼 데이터중 userpw가 포함된 경우 f\_encodeUserPw 함수를 사용하여 oriUserPw를 암호화후 json타입에 넣어줍니다.

3. Server에서는 암호화된 스트링을 bodyParamConvert 를 사용해 복호화후 json타입의 데이터로 변환시켜주고 업무에 필요한 데이터를 추출해 사용합니다.

Server->Client

1. Server에서는 데이터를 api형식으로 client로 넘겨줄 때 기존의 hashmap타입의 데이터를 Gson라이브러리를 사용하여 json타입으로 바꿔주고 jws형식으로 암호화 후 client로 보내줍니다. 모든 server데이터는 apiIF를 통해 client로 전달되기 때문에 apiIF.java파일만 코드 수정했습니다.

2.Client에서 reponse받은 암호화된 데이터를 f\_decodeData를 사용하여 json타입으로 복호화후 업무에 맞는 데이터를 추출해 사용합니다.