|  |
| --- |
| DB가 안올라온상태에서  WEBAPP를 키고 나중에 DB가 붙어도 처리되도록 해야한다  즉 CODE값을 WEBAPP에서 처음 부팅됐을때 가져오는 로직이 있다면  그부분을 못가져왔을것이며 (DB꺼져있어) 이부분이 다시 DB가 붙으면 다시 CODE값을 가져도록 해야한다 |
| 로그 찍는 부분을 다시 한번 확인해라..  로그를 찍을때는 로그 포맷을 정해서 찍어라  예를들어  StringBuffer logInfo = **new** StringBuffer();  logInfo.append(ex.getClass().getName());  logInfo.append("|").append(request.getRequestURI());//API CODE(URL)  logInfo.append("|").append(rcd);  //BootCodeVO codevo = BootManager.getInstance().getCode(rcd);  BootCodeVO codevo =**null**;  logInfo.append("|").append( (codevo==**null**?"알수없음":codevo.getCd\_nm()) );  logInfo.append("|").append(RequestUtil.*getRemoteAddr*(request)); // 접속IP  logInfo.append("|").append(**null**==ex.getMessage()?"":ex.getMessage().replace("\n","").replace("\r", "")); // 처리 담당자 ID  logInfo.append("|").append(trace);  log.error(logInfo.toString().replace("\n", "").replace("\r",""));   |  |  |  | | --- | --- | --- | | 사용자 로그파일 형식 | | | | 로그 구분자 필드 | | (기호를 기준으로 로그필드 구분 함) | | | 참고 | 로그 생성 시 로그레벨을 고려 합니다. 상용적용 시 INFO 이상만 로그로 남길 예정 임. 상용반영을 고려하여 로그레벨을 지정 해야 함. | | | 구분 | 필수여부 | 용도 | | 로그시간 | O | YYYYMMDDHH24MISS 14byte 형식 | | 로그레벨 | O | FATAL, ERROR, WARN, INFO, DEBUG 5단계만 명시 | | Controller | O | 로그 생성 컨트롤러 명시 | | Function or Method | X | 로그 생성 함수 및 메소드 명시 | | 에러코드 | O | 에러코드 명시 (성공 case도 로그로 명시 함) | | 에러부연설명 | X | 에러부연설명 | | API Code | X | API 처리 관련 로그 일 경우 전문번호 지정 필요 | | IP | X | 접속 IP | | ID | X | 처리 담당자 ID (사용자/관리자 등등 로그인상태 ID) | | Reserve field1 | X | 필요한 정보 추가 명시 | | Reserve field2 | X | 필요한 정보 추가 명시 | | Reserve field3 | X | 필요한 정보 추가 명시 | | 예시 | 20160325125110|DEGUG|APIController||M1001|공통 필수 입력정보 부족|AI001|221.142.80.99|omnifit||| | | |
|  |
| 1. /guest/\* 로그인하지 않은 즉 ANONYMOUS 사용자가 Controller 탈수 있는 URL (시큐어 검사 안함) 2. /anon/\* 로그인하지 않은 즉 ANONYMOUS 사용자가 Controller 탈수 있는 URL (csrf등 시큐어 검사함) 3. /interface/\* 로그인하지 않는 즉 ANONYMOUS 사용자와 연계관련된 Controller 탈수 있는 URL(csrf등 시큐어 검사함) 4. /auth/\* 로그인된 사용자의 Contoller 탈수 있는 URL |
|  |

DB 데이터베이스

|  |
| --- |
| 시퀀스정책  시퀀스 사용   * 유일성, 속도를 보장하긴 하지만 이빨이 빠질수가 있다   시퀀스 관리테이블   * 관리테이블을 사용하게되면 속도,유일성에 문제가 생길수도 있다 테이블 롹걸릴수도있다. |
| 키 정책  복합키   * 키만보아도 데이터 의미를 추론할수 있다 * 조인걸기위하여 많은 맵핑이 필요하다   단일 유일키   * 일반 시퀀스성 단일키를 사용하게된다면 키를 통해 데이터 의미를 추론할수 없다 또한 조인할때 느릴수가 있다 * 키를 하나만 가지고 다닐수 있어서 정확하게 키추론이 가능하다면 조인이 훨신 빨라진다 |

query 쿼리

|  |
| --- |
| 조인걸때 서브 쿼리보단 키 기준으로 걸어라  조건을 걸때 메인이되는 테이블 기준으로 키를 잡아 조인을 걸어라  from절뒤에 테이블 순서는 중요하다 첫번째에 마스터 테이블을 두어라 |

데이터

|  |
| --- |
| 데이터 성격  로그성   * 그때그때 쌓이는 그때 Insert될때 현시점에 정보를 저장해둔다 쌓아둔다   Current성   * 현재 상태를 그때그때 UPDATE한다 단일 ROW |

암호화, 마스킹?

|  |
| --- |
| 개인정보   * 양방향 암복호화 예)AES256   비밀번호   * 단방향 , 개인 Slat키사용하여 만들면된다 예)SHA512   1. Salt 키 생성(개인키) 32byte  2. SHA512(Salt + Password) 단방향 비밀번호 암호화후 저장  1. 개인정보 AES256 암호화 (global 32byte key = “e43547b73fa99e755bbd5bb07ertyfkk”) |
| 개인정보는 화면에 보여줄때 마스킹되어서 처리되어야 한다.   |  |  |  | | --- | --- | --- | | 구분 | 적용가이드 | 비고 | | 통신구간 | API 통신 구간 SSL 적용 | ominfit.co.kr 멀티도메인 SSL 구매 필요 | | 개인정보 | 통신전문 중 개인정보는 AES-256 암화화 적용 | 공통 key는 별도 공유 예정 | | ID/PW 저장 | 자동로그인을 위한 ID/PW 저장 AES-256 암호화 적용 | ID/PW 저장용 개별 key 제공 예정 | | 로그인 | 로그인 성공 시 access\_token 발행. 인증 필요 API 연동 시 access\_token 필수 전달 필요. |  | | access\_token | 로그인 시 refresh\_token, access\_token, access\_token expired\_date 제공. refresh\_token 고정, access\_token은 로그인 시 매번 변경 됨. |  | |  |  |  | | 암호화 대상 | | | | 대상컬럼 | 화면 노출가이드 | 암호화 방식 | | 성명 | 성명 뒤 1글자 (예) 박병\* | AES-256 양방향 암호화 | | 예금주 | 예금주 뒤 1글자 (예) 박병\* | AES-256 양방향 암호화 | | 생년월일 | 생년월일 (미적용) | AES-256 양방향 암호화 | | 전화번호 | 전화번호 국번 뒤2글자, 번호 앞1글자 (예) 0\*\*-\*077-2393 | AES-256 양방향 암호화 | | 주소 | 동 이후, 도로명 이후 모두 적용 (예) 서울특별시 구로구 구로3동 \*\*\*\* | AES-256 양방향 암호화 | | 이메일 | 아이디 뒤 3글자 (예) pppc\*\*\*@omnicns.co.kr | AES-256 양방향 암호화 | | 계좌번호 | 앞 6글자 까지만 표시 (예) 132-01-3\*\*\*\*\* | AES-256 양방향 암호화 | | 사업자등록번호 | 앞 5글자 까지만 표시 (예) 113-86-\*\*\*\*\* | AES-256 양방향 암호화 | | 아이디 | 아이디 뒤 3글자 (예) pppc\*\*\* | AES-256 양방향 암호화 | | 비밀번호 | 노출 불가 | SHA-512 단방향 암호화 (개별 salt 값 적용) | | IP | 첫 3글자, 셋째 3글자만 표시 (예시) 112.\*\*\*.31.\*\* | AES-256 양방향 암호화 | | 직업 | 앞 2글자 까지만 표시 - 여러 건 노출 시 적용 (예) 회사\* | AES-256 양방향 암호화 | | 회사명 | 앞 3글자 까지만 표시 - 여러 건 노출 시 적용 (예) 옴니씨\*\*\* | AES-256 양방향 암호화 | | 키 | 앞 2글자 까지만 표시 - 여러 건 노출 시 적용 (예) 16\*.\*\* | AES-256 양방향 암호화 | | 몸무게 | 앞 1글자 까지만 표시 - 여러 건 노출 시 적용 (예) 6\*.\*\* | AES-256 양방향 암호화 | | 허리둘레 | 앞 1글자 까지만 표시 - 여러 건 노출 시 적용 (예) 3\*.\*\* | AES-256 양방향 암호화 | |

front end

|  |
| --- |
| 싱크 처리일때 사용자가 submit 버튼 중복 실행 못하게  클릭 처리한번 되고나면 해당버튼을 disable 처리한다 |

컨트롤처리

|  |
| --- |
| 특정값 update처리할시  어떠한 값에 대한 값만 처리하도록 즉  A값인 상태일때 A-1 값으로밖에 진행할수 없는데.  B-1값을 업데이트 할수도 있으니  그값을 검증 기존값 검증하는 로직 및 class를 만들어 공동 처리한다  즉 데이터의 연결성을 보장해야한다  //상태에 따른 step logic 처리  //APPROVED, REJECTED, DELETED 로 UPDATE칠때에는 기존 상태값이 WAIT 이여야 한다.  Code inStatus = Code.***CODES\_MSG\_STATUS***.child(msgWait.getStatus());  Code dbStatus = Code.***CODES\_MSG\_STATUS***.child(selectWait.getStatus());  **if**( (inStatus==Code.***CODE\_MSG\_STATUS\_APPROVED*** || inStatus==Code.***CODE\_MSG\_STATUS\_REJECTD*** ||inStatus==Code.***CODE\_MSG\_STATUS\_DELETED***) && dbStatus != Code.***CODE\_MSG\_STATUS\_WAIT*** ){  **throw** **new** Exception("이미 처리되었습니다.");  } |

네트워크

|  |
| --- |
| 일반적인 인터페이스 작업 처리할때  또다른 테스트 베드 처리를 즉 실질적 execute 처리되지 않지만 로직 검증 된후 return처리 보내주는  또다른 패스가 있어야된다. 그래야지 테스트 서버가 없어도 실질적 리얼로 테스트(실질적 처리 안되지만) 확인 가능한지 확인이 가능하다 |

느낌점

|  |
| --- |
| 절때 개발 테스트 서버에서 테스트 해보지 않으면 절때 그 프로젝트는 피해라  아니면 책임자의 확답을 받고 리얼로 반영해라 무조건 회의록을 작성하고 녹취가능하면 녹취를해라 |
| 여러 업체들간의 프로젝트 수행시  어떤 문제가 발생시 같은 책임을 지지 않고 어떤 안을 그럴듯한 안을 제시하고 떠넘기고 떠날려고 하는 그런 KT같은 새끼들이 있으니 무조건 중장기적으로 확인해보아라 우리 회사에 지속적인 로스가 있는지  그들은 달콤한 안을 제시하고 그러면 되죠 라는 식으로 하고 계약을만 채우고 빠질놈들이다.  그러니 그 달콤한 혀바닥에 넘어가지 말어라 |
| 절때 아이티사업은 기술적이지 않은 사업이다.  그저 책임전가 회피 계약 문서만이 남을뿐이다. |
| 명분을 가지고 정확하게 말을해라 |
| 상대방이 못알아들었다고 느껴질때는 두세번이라도 다시 말하여라 |

일반적인 프로그램 짤때

통신

|  |
| --- |
| http |
| multipart 로 보낼때는 서버측에서 http 100 Continue 를 보낸다  curl -v -F "userid=1" -F "filecomment=This is an image file" -F "image=@/c/ojdbc6-11.2.0.3.jar" http://127.0.0.1:9991/lgt\_submit.php  \* timeout on name lookup is not supported  \* Trying 127.0.0.1...  % Total % Received % Xferd Average Speed Time Time Time Current  Dload Upload Total Spent Left Speed  0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0\* Connected to 127.0.0.1 (127.0.0.1) port 9991 (#0)  > POST /lgt\_submit.php HTTP/1.1  > Host: 127.0.0.1:9991  > User-Agent: curl/7.49.1  > Accept: \*/\*  > Content-Length: 2714618  > Expect: 100-continue  > Content-Type: multipart/form-data; boundary=------------------------ac0a5dc5d5e88fea  >  < HTTP/1.1 100 Continue  } [381 bytes data]  91 2650k 0 0 91 2416k 0 1516k 0:00:01 0:00:01 --:--:-- 1531k  < HTTP/1.1 200 OK  < Date: Fri, 09 Sep 2016 14:08:22 GMT  < Server: Apache  < X-Powered-By: PHP/7.0.9  < CBPP\_RESULT: FAILURE  < CBPP\_DESCRIPTION: area\_code+%BF%C0%B7%F9  < CBPP\_ID:  < Content-Length: 73  < Connection: close  < Content-Type: text/html; charset=UTF-8  <  { [73 bytes data]  100 2651k 100 73 100 2650k 29 1060k 0:00:02 0:00:02 --:--:-- 1066kCBPP\_RESULT: FAILURE<br>CBPP\_DESCRIPTION: area\_code ▒▒▒▒<br>CBPP\_ID: <br>  \* Closing connection 0  그렇기 때문에 받는쪽에서는 상태값을 100을 필터링 하던지 해서 정상 http 리턴값을 가지고 처리하여라 |
| 일반  curl -v http://127.0.0.1:9991/lgt\_submit.php  \* timeout on name lookup is not supported  \* Trying 127.0.0.1...  % Total % Received % Xferd Average Speed Time Time Time Current  Dload Upload Total Spent Left Speed  0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0\* Connected to 127.0.0.1 (127.0.0.1) port 9991 (#0)  > GET /lgt\_submit.php HTTP/1.1  > Host: 127.0.0.1:9991  > User-Agent: curl/7.49.1  > Accept: \*/\*  >  < HTTP/1.1 200 OK  < Date: Fri, 09 Sep 2016 14:10:16 GMT  < Server: Apache  < X-Powered-By: PHP/7.0.9  < CBPP\_RESULT: FAILURE  < CBPP\_DESCRIPTION: area\_code+%BF%C0%B7%F9  < CBPP\_ID:  < Content-Length: 73  < Connection: close  < Content-Type: text/html; charset=UTF-8  <  { [73 bytes data]  100 73 100 73 0 0 1553 0 --:--:-- --:--:-- --:--:-- 2281CBPP\_RESULT: FAILURE<br>CBPP\_DESCRIPTION: area\_code ▒▒▒▒<br>CBPP\_ID: <br>  \* Closing connection 0 |
| |  |  | | --- | --- | | **정상 case** | **비정상 case** | | cid:image002.jpg@01D20ABC.CB2FE9D0 | cid:image003.jpg@01D20ABC.CB2FE9D0 | |
| **연동할때**  **타 시스템과 연계할때 꼭 규약서를 잘치키고**  **받아주는거에는 퍼블릭하게 보낼때는 닫혀서 보내라**  **또한 나중에 문제가 발생할때 무조건 책임소재를 따지기때문에**  **FULL로 로그를 찍어서 보내기전에 로그, 보내고 난뒤 로그, 받은뒤 로그, 처리후 로그**  **메타성 헤더라던지 다 찍어놔라** |

config property

|  |
| --- |
| config는  WEBAPP 마다 각자의 Config를 빼라  Tomcat실행할때 -D옵션으로 config파일 위치를 넣어서 처리해라  또한 scheduler를 이용해서 특정 초마다 reload 하는기능을 넣어라 |

프로그램 빌드에대한...타입

|  |
| --- |
| 각자의 프로젝트단위 각자의 롤을가진 구간이있으며  그것들이 연결된 멀티 프로젝트 단위가 있다.  또한 IDE에서 서로 링크시켜 컴파일하는 것이 있으며  IDE에서 MAKE하여 타 프로그램으로 포팅시키는것또한 있다  또한 마지막 make하여 상용쪼으로 가는 build가 있다 (예 그래들) |

리턴에 대한생각

|  |
| --- |
| 어떠한것들은 출력값이 아무것도 없이 그냥 출력되야되는것들있다  즉 여러고객들의 정보를 list로 리턴받는경우 고객이 아무도없을경우에도 빈 list로 보내줘야한다.  null 이나 Exception을 내선 안된다 다시말해 optional한 성격을 가진 업무는 Exception또는 null을 반환하면 안된다  또다른 반대상황에서는 키를 가지고 조회하는 즉 조건이 있는 조회일시 그조건이 만족하지 않거나 상황이 안된다면 Exception 이나 null값을 반환하는것이 맞다. |