

SQL매니저



데이터를 조회(select)하거나 변형(insert/update/delete)해야할 경우 SQL을 사용합니다.

'SQL 매니저'를 통해 SQL을 2가지 유형(조회, 등록/수정/삭제)으로 나누어 쿼리 작성과 조건문/파라미터 설정을 보다 편리하게 할 수 있으며, 작성한 쿼리를 실행시켜 오류 확인도 가능합니다.

가이드

접근방법

'SQL 관리' 메뉴에서 오른쪽 상단 추가버튼 클릭 시 SQL을 추가 할 수 있고, 리스트에 있는 ID 클릭 시 SQL을 수정할 수 있습니다.

사용하는 목적

데이터를 조회하거나 변형해야 할 경우 SQL을 사용하며, SQL에는 쿼리 뿐만 아니라 JSON 형태인 조건문/파라미터 설정이 필요합니다.

SQL 매니저를 사용하면 JSON 형태의 설정을 직접 작성하지 않아도 되며, 작성한 쿼리의 오류 여부를 체크 할 수 있습니다.

사용하기 전 사전지식

SQL 매니저를 사용하기 전에는 기본적으로 쿼리 작성에 대한 지식이 필요합니다.

SQL 생성하기

SQL 관리 기본 속성

2가지 유형(조회, 등록/수정/삭제) 모두 기본 속성은 동일합니다.

① ID	<input type="text" value="WSQL00323"/>	② SQL명	<input type="text" value="Sql Name"/>
③ 사용여부	<input checked="" type="radio"/> 예 <input type="radio"/> 아니오	④ DB Name	<input type="text" value="Sql DB Name"/>
<p>❗ SQL을 실행 할 DB를 입력합니다. 입력하지 않을 시 시스템 기본 DB로 연결됩니다.</p>			
⑤ 설명	<input type="text" value="Sql Descr"/>		

① SQL 고유 아이디

② SQL 이름

③ SQL 사용여부

④ JNDI Name - 서버 설정 시 EGENE 솔루션 DB외에 추가로 JNDI설정이 있다면 JNDI Name을 작성하여 해당 DB에서 쿼리를 수행 할 수 있도록 하는 기능입니다. (입력 후 접속 여부 확인 가능)

⑤ SQL 설명

Select SQL 생성하기

EGENE DOCUMENT



데이터를 조회하거나 검색하기 위한 쿼리를 작성할 때 사용합니다.

SQL과 조건문을 설정할 수 있습니다.

속성



1) SQL

데이터 조회, 검색하는 select 문 쿼리를 작성합니다.

조건으로 **#{변수명}**과 같이 입력 할 수 있고, 변수명과 조건문 설정의 ID를 연결하여 조건문 쿼리를 설정 할 수 있습니다.

< SELECT 문 쿼리 샘플 >

select [컬럼명] from [테이블]	select * from ecr_sql
[조건]	#{cond}
	select * from #{tname}
	#{cond}

2) 조건문

조건문 설정에는 크게 조건 ID, WHERE 사용여부, 조건항목이 있습니다.

조건문

조건문 추가

1 조건 ID


cond

2 WHERE 사용여부

☒

3 조건항목

조건항목 추가

항목	유형	* 컬럼명	* 매핑 변수	필수
	직접입력	sql_id = ?	sql_id	<input type="checkbox"/>

- 조건 ID : SQL 쿼리에 작성한 **#{변수명}**의 변수명과 조건 ID가 동일 해야 합니다.
- WHERE 사용여부 : 해당 조건이 where 절을 사용한다면 사용여부에 체크합니다. (체크 : where 사용, 체크 X : and 사용)
- 조건항목 : 유형, 컬럼명, 매핑 변수, 필수 체크를 할 수 있습니다.
 - 유형 : '직접입력' 또는 '연산자' 를 선택 합니다.
 - 컬럼명 : '직접입력' 유형 선택 시 sql 쿼리를 직접 입력합니다. ex) sql_id = ?
'연산자' 유형 선택 시 컬럼명을 선택합니다. ex) sql_id
(컬럼명은 SQL 실행 시 결과 값으로 나오는 컬럼들이 표시됩니다.)
 - 매핑 변수 : 컬럼명과 매핑 되는 파라미터 변수이며 %를 사용 할 수 있고, 콤마(,)로 여러 개를 입력 할 수 있습니다.
 - '직접입력' 유형 - '?' 개수와 동일하게 설정, #{array}의 경우 1개 설정
 - '연산자' 유형 - 1개 설정
 - 필수 체크 : 해당 조건항목이 필수로 쿼리에 포함되어야 할 경우 체크합니다.

3) SQL 미리보기

설정한 조건문을 토대로 'SQL 미리보기'에 동일하게 변수입력 테이블이 생성됩니다.

조건문에 설정 되어있는 경우 해당 조건 ID로 테이블이 생성되며, 조건문에 설정되지 않은 **#{변수명}**은 조건문 외 파라미터 테이블에 추가가 됩니다.

변수 입력 테이블에서 값을 입력 후 'SQL 실행' 버튼을 누르면 해당 결과를 미리 볼 수 있습니다.

INSERT / UPDATE / DELETE SQL 생성하기



데이터에 변형을 가하는 종류의 쿼리를 작성할 때 사용합니다.

SQL과 파라미터를 설정할 수 있습니다.

속성

4 ~ 6

+

1

×

2

SQL 1

1 update ecr_sql set sql_name = 'test'

2 where sql_id = ?

파라미터 2

순번	*변수명	필수 여부
1	<div>변수값</div> sql_id	<input type="checkbox"/>

SQL 미리보기

변수 입력

순번

매핑 변수

값

1	sql_id	wsq00219
---	--------	----------

SQL 실행

결과

Result: [DML_SUCCESS:elapsed=5]
update ecr_sql set sql_name = 'test'
where sql_id = 'wsq00219'

1) SQL

데이터에 변형을 가하는 insert/update/delete 문 쿼리를 작성합니다.

파라미터 값을 넘겨줄 부분을 ?로 입력 해야 하며, ?의 개수에 따라 파라미터 설정을 할 수 있습니다.

#{변수명}은 필요한 경우 입력 가능하며, 파라미터 설정은 따로 하지 않고 소스 부분에서 해당 변수명으로 값을 넘겨 줄 수 있습니다. (SQL 활용하기 – 샘플 참고)

< INSERT / UPDATE / DELETE 문 쿼리 샘플 >

insert into [테이블] ([컬럼명]) values ([값])	insert into #{tname} (att_id, att_src_id, att_tas_id,) values (?,?,?,)
update [테이블] set [컬럼명] = '변경 할 값' where [조건]	update ecr_sql set sql_name = 'test' where sql_id = ?
delete from [테이블] where [조건]	delete from #{tname} where att_id = ?

2) 파라미터

파라미터 설정에서는 변수값과 예약어를 선택할 수 있습니다.

변수값은 파라미터 value를 넘겨받기 위한 key에서 사용하는 변수 명을 입력 해야 하며, 대부분 컬럼명을 사용하고 있습니다. 예약어는 '현재일시', '로그인 사용자', '로그인 사용자 부서', '로그인 사용자 회사', '유니크 아이디' 중 하나를 선택하여 사용할 수 있습니다.

■ 파라미터

순번	* 변수 명	필수 여부
1	<div>변수값</div> <div>sql_id</div>	<input type="checkbox"/>
2	<div>변수값</div> <div>sql_name</div>	<input type="checkbox"/>

예약어에서 '유니크 아이디' 선택 시 시퀀스 설정을 할 수 있는 톱니바퀴가 나오게 되며, 톱니바퀴 클릭 시 변수명과 시퀀스 방식을 선택할 수 있습니다.

시퀀스 방식에는 Ukey 방식과 엔터티 방식이 있습니다. Ukey 방식은 임의로 key를 만들어주는 방식이고, 엔터티 방식은 Sequence 메뉴에서 엔티티 별로 설정한 시퀀스로 key를 만들어주는 방식입니다.

3) SQL 미리보기

설정한 파라미터를 토대로 'SQL 미리보기'에 동일하게 변수입력 테이블이 생성됩니다.

변수 입력 테이블에서 값을 입력 후 'SQL 실행' 버튼을 누르면 해당 결과를 미리 볼 수 있습니다.

4) SQL 추가 기능

조회 유형과는 달리 왼쪽 상단 플러스 버튼을 통해 SQL을 추가하여 여러 SQL을 작성 할 수 있습니다.

플러스 버튼 오른쪽으로 SQL이 나열되어 있고 순서대로 저장됩니다.

5) SQL 삭제기능

나열되어 있는 순번을 클릭하면 옆에 X 버튼이 보이게 되며, X 버튼을 통해 SQL을 삭제 할 수 있습니다. SQL을 모두 삭제 할 경우 첫번째 순번 SQL이 자동 생성됩니다.

6) SQL 순서 변경 방법

순서 변경을 원하는 순번을 클릭하여 drag&drop으로 원하는 위치로 변경이 가능합니다.

SQL 활용하기

샘플

1) 품

폼에서 'Sql.Select' 아톰과 같이 SQL을 사용하는 아톰이 있습니다.

해당 아톰에 대한 Config 설정은 아래와 같습니다.

```
{
  "sql_id": "SQL_ID",
  "sql_args": [{ "col" : key, "value" : value }]
}
```

- sql_id : SQL ID
- sql_args : SQL의 조건 설정

Ex) 쿼리와 조건문을 아래와 같이 설정 후 아톰 Config 설정 방법

■ SQL

```
1 select emp_id as id
2 ,emp_name as name
3 from ecf_employee
4 #{cond}
5 order by emp_id
```

■ 조건문

* 조건 ID

cond

WHERE 사용여부

☒

조건항목

▶ 조건항목 추가

항목	유형	* 컬럼명	* 매핑 변수	필수
<div><div>조건</div><div>✕</div></div>	<div>직접입력</div> <div>▼</div>	emp_id = ?	emp_id	<input type="checkbox"/>

- value 값을 고정 시키는 경우

EGENE DOCUMENT

```
"sql_id": "SQL_TMPT_005",

"sql_args": [{ "col" : "emp_id", "value" : "admin" }]

}
```

조건문에 설정한 매핑 변수를 key 값으로 작성, value 값으로 넘겨 줄 값 작성

- value 값에 특정 값을 전달하는 경우

```
{

"sql_id": "SQL_TMPT_005",

"sql_args": [{ "col" : "emp_id", "value" : "#{session.user.emp_id}" }]

}
```

#{session.user.emp_id} => 현재 로그인한 유저 ID

* box / row / session의 정보들을 사용할 수 있습니다.

#{box.컬럼명} / #{row.컬럼명}

Ex) **#{box.pms_id} / #{row.pem_mtn_id}**

2) 리스트

리스트 검색조건에서 ‘Sql.Select’ 아톰과 같이 SQL을 사용하는 아톰이 있습니다.

해당 아톰에 대한 아톰 설정은 아래와 같습니다.

```
[

  "SQL_ID",

  [

    {

      "col": key

      "value": value

    }

  ]

]
```

- SQL ID

- SQL의 조건 설정

Ex) 쿼리와 조건문을 아래와 같이 설정 후 아톰 설정 방법

■ SQL

```
1 with recursive t_cmcat as (
2   select cod_id, cod_pid from ecf_code #{cond}
3   union all
4   select t.cod_id, t.cod_pid from ecf_code t, t_cmcat p where t.cod_pid = p.cod_id
5   ), t_cmsys as (
6   select cm_id, cm_name, cm_pid, 1 lev, cm_order from eso_cm, t_cmcat where cm_cat_cd = cod_id and cm_pid is null
7   union all
8   select ct.cm_id, ct.cm_name, ct.cm_pid, pt.lev + 1 lev, ct.cm_order from eso_cm ct, t_cmsys pt
9   where ct.cm_pid = pt.cm_id
10  )
11 SELECT cm_id id, cm_name name, cm_pid pid, lev, cm_order from t_cmsys order by lev, cm_order, cm_id
```

■ 조건문

▶ 조건문 추가

* 조건 ID cond WHERE 사용여부 ☒

조건항목 ▶ 조건항목 추가

항목	유형	* 컬럼명	* 매핑 변수	필수
	직접입력	cod_id=?	cm_cat_cd	<input type="checkbox"/>

```
[

  "COND_SQL_TMPT_009",

  [

    {

      "col": "cm_cat_cd"

      "value": "SYS"

    }

  ]

]
```

조건문에 설정한 매핑 변수를 key 값으로 작성, value 값으로 넘겨 줄 값 작성

3) 릴레이션

릴레이션에서 ‘Sql.Select’ 아톰과 같이 SQL을 사용하는 아톰이 있습니다.

EGENE DOCUMENT

해당 아툼에 대한 아툼 설정은 아래와 같습니다.

```
{
  "sql_id" : "SQL_ID",
  "params": {
    key : value
  }
}
```

- sql_id : SQL ID

- params : SQL의 조건 설정

Ex) 쿼리와 조건문을 아래와 같이 설정 후 아툼 설정 방법


■ SQL

```
1 select rownum,
2   ent_id as id,
3   ent_name as name
4   from (select row_number() over() as rownum, ent_id, ent_name
5         from efc_entity) as tmp
6   #{cond}
```

■ 조건문

조건문 ID cond WHERE 사용여부 ☒

조건항목

항목	유형	* 컬럼명	* 매핑 변수	필수
	직접입력	rownum <= 10 and '1' <> ?	key	<input type="checkbox"/>

```
{
  "sql_id" : "RLT.SQL_TMPT_006",
  "params": {
    "key" : "2"
  }
}
```

조건문에 설정한 매핑 변수를 key 값으로 작성, value 값으로 넘겨 줄 값 작성

4) 소스

1. JavaScript

자바 스크립트에서는 비동기 방식인 AJAX을 이용하여 SQL을 사용할 수 있습니다.

SQL을 사용하기 위한 API는 2가지가 있으며, 유형별로 API를 다르게 사용하고 있습니다.

1) 조회 유형 => " /sql/{id:.+} " API 사용

```
$service.ajax({
  url: '/api/egene/sql/SQL_ID',
  data: { key : value }
}, function (result) {
});
```

url => /api/egene/sql/(SQL ID)

data => 파라미터 변수 명 : 데이터 값

Ex) 쿼리와 조건문을 아래와 같이 설정 후 소스에서 SQL을 사용할 때 data 설정 방법


■ SQL

```
1 select * from ecr_sql
2   #{cond}
```

■ 조건문

조건문 ID cond WHERE 사용여부 ☒

조건항목

항목	유형	* 컬럼명	* 매핑 변수	필수
	직접입력	sql_id = ?	sql_id	<input type="checkbox"/>

조건문에 설정한 매핑 변수를 key 값으로 작성, value 값으로 넘겨 줄 값 작성

=> data : { sql_id : 'TestSql' }

2) 등록/수정/삭제 유형 => " /sql/exec/{id:.+} " API 사용

EGENE DOCUMENT

```
ice.sqls().
    url: '/api/egene/sql/exec/SQL_ID',
    type: 'POST',
    data: { key : value }
}, function (result) {
});
```

url => /api/egene/sql/exec/(SQL ID)

type => type은 반드시 POST 유형

data => 파라미터 변수 명 : 데이터 값

Ex) 쿼리와 파라미터를 아래와 같이 설정 후 소스에서 data 설정 방법

■ SQL

1 DELETE FROM ecr_sql WHERE sql_id = ?

■ 파라미터

순번	* 변수 명	필수 여부
1	<div><div>변수값</div><div>sql_id</div></div>	<input type="checkbox"/>

조건문에 설정한 매핑 변수를 key 값으로 작성, value 값으로 넘겨 줄 값 작성

```
=> data : { sql_id : 'TestSql' }
```

2. JAVA

자바에서 유형별로 사용하는 소스는 아래와 같습니다.

1) 조회 유형

- 조건문 설정만 있는 경우

```
Sqls sqls = ice.sqls();

Data d = new Data();
d.put( key, value );

Result r = sqls.getResult( SQL_ID, d );
RecordSet rs = r.getRecordSet();
```

- 조건문 설정 외 #{변수명}이 있는 경우

```
Sqls sqls = ice.sqls();

HashMap vars = new HashMap();
vars.put( "변수명", value );

Data d = new Data();
d.put( key, value );

Result r = sqls.getResult( SQL_ID, vars, d );
RecordSet rs = r.getRecordSet();
```

Ex) 쿼리와 조건문을 아래와 같이 설정 후 소스에서 SQL 사용 방법



EGENE DOCUMENT

Sqls sqls = ice.sqls();

```
HashMap vars = new HashMap();
vars.put( "tname", ent.tab_master );
```

```
Data d = new Data();
d.put( "sql_id", sql_id );
```

```
Result r = sqls.getResult( SQL_ID, vars, d );
RecordSet rs = r.getRecordSet();
```

조건문 설정 제외한 쿼리에 있는 #{변수명}에 해당하는 변수명을 key 값으로 설정, value 값 설정

조건문에서 설정한 변수명을 key 값으로 설정, value 값 설정

2) 등록/수정/삭제 유형

- 쿼리에 #{변수명}이 없고 ? 만 있을 경우

```
Sqls sqls = ice.sqls();
String sql_id = "SQL_ID";

Data d = new Data();
d.put( key, value );

TxResult tr = sqls.executeArray(con, sql_id, d);
```

- 쿼리에 #{변수명}과 ?가 함께 있을 경우

```
Sqls sqls = ice.sqls();
String sql_id = "SQL_ID";

HashMap vars = new HashMap();
vars.put( "변수명", value );

Data d = new Data();
d.put( key, value );

TxResult tr = sqls.executeArray(con, sql_id, vars, d);
```

Ex) 쿼리와 파라미터를 아래와 같이 설정 후 소스에서 SQL 사용 방법

■ SQL

```
1 update #{tname} t
2 set #{prefix}_tas_id = ?
3 where #{prefix}_id = ?
```

■ 파라미터

순번	* 변수 명	필수 여부
1	<div>변수값</div> tas_id	<input type="checkbox"/>
2	<div>변수값</div> src_id	<input type="checkbox"/>

EGENE DOCUMENT

```
String sql_id = "SQL_ID";

HashMap vars = new HashMap();
vars.put( "tname", ent.tab_master );
vars.put( "prefix", ent.prefix );

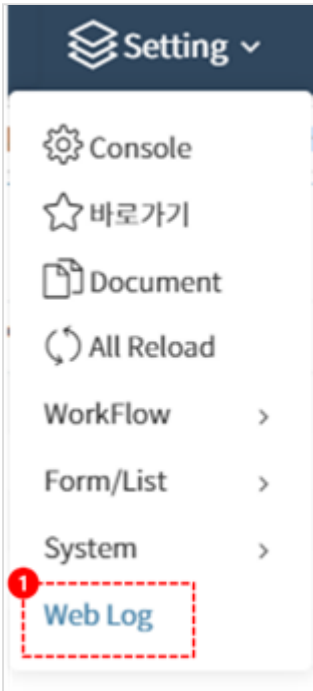
Data d = new Data();
d.put( "src_id" , src_id );
d.put( "tas_id" , tas_id );

TrxResult tr = sqls.execute(con, sql_id, d);
```

쿼리에 있는 #{변수명}에 해당하는 변수명을 key 값으로 설정, value 값 설정
파라미터에서 설정한 변수명을 key 값으로 설정, value 값 설정

SQL 로그 확인 하는 방법

- 1) SQL 로그 보는 이유 및 보는 케이스
SQL 로그는 SQL에 설정한 파라미터 값으로 넘겨 받아 오류 없이 정상적으로 동작 했는지 확인 할 수 있습니다.
- 2) 웹 로그 보는 방법



- ① 오른쪽 상단에 있는 Setting – Web Log 클릭
- ② ‘SQL Log’ 클릭 후 SQL 사용하는 작업 실행 -> 로그 결과 확인
- ③ 로그를 멈추고 싶을 때 ‘중지’ 버튼 클릭
- ④ 로그를 다시 실행하려면 ‘현재 위치부터 시작’ 버튼 클릭

EGENE Maria DB Function

EGENE 솔루션 내부에서 쿼리를 사용하여 작성하는 구문에서는 아래의 내장 Function을 통하여 손쉽게 치환 및 변환이 가능합니다.

function 명	매개변수	반환값	설명
FM_LDT	VARCHAR(50) : 날짜 포맷 변환 컬럼	VARCHAR (50)	8자리의 날짜문자를 포맷화 할 수 있습니다.
FM_LDTTM	VARCHAR(50) : 날짜 포맷 변환 컬럼	VARCHAR (50)	14자리의 날짜문자를 포맷화 할 수 있습니다.
GETCODE_FULLPAT H_NAME	VARCHAR(20) : 코드명 변환 컬럼	VARCHAR (200)	코드테이블의 데이터 기반으로 계층형일 경우 하나의 코드명으로 표 기 할 수 있습니다.
getIncPerform	VARCHAR(20) : 서비스등급 추출을 위한 장애 티켓 ID	VARCHAR (200)	장애(ICM) 테이블 데이터 기반으로 서비스 등급을 추출할 수 있습 니다.

FUNCTION	DESCRIPTION	RETURN TYPE	REMARKS
GET_CODE	VARCHAR(100) : 코드명 변환 컬럼	VARCHAR(100)	코드테이블의 데이터 기반으로 코드명을 표기 할 수 있습니다.
GET_DEADLINE	VARCHAR(50) : 시작일 컬럼 ,VARCHAR(50) : 종료일 컬럼, INT : 시작일로 부터 더해질 일수	VARCHAR(50)	마감일 초과여부를 추출 할 수 있습니다.
GET_DPTNAME	VARCHAR(20) : 부서명 변환 컬럼	VARCHAR(200)	부서테이블의 데이터 기반으로 부서명을 표기할 수 있습니다.
GET_DUR	VARCHAR(20) : 비교시작일시 ,VARCHAR(20) : 비교종료일시	INT(11)	비교 날짜 기준으로 분값을 결과로 표기합니다.
GET_EMPNAME	VARCHAR(20) : 직원명 변환 컬럼	VARCHAR(200)	직원 테이블 데이터 기반으로 직원명을 추출할 수 있습니다.
GET_LASTDAY	VARCHAR(20) : 월의 마지막 날짜 변환 컬럼	VARCHAR(8)	입력된 월의 마지막 날짜를 추출 할 수 있습니다.
GET_ORGNAME	VARCHAR(20) : 회사명 변환 컬럼	VARCHAR(200)	회사테이블의 데이터 기반으로 회사명을 표기할 수 있습니다.
GET_PCAT_ID	VARCHAR(20) : 최상위 코드 추출 컬럼	VARCHAR(20)	코드테이블의 데이터 기반으로 해당 코드의 최상위 코드 ID를 추출 할 수 있습니다.
GET_PROCTIME	VARCHAR(20) : 비교시작일 컬럼 ,VARCHAR(20) : 비교종료일 컬럼	INT(11)	휴일관리 테이블 데이터 기반으로 워킹데이 일수기반으로 분 값을 추출 할 수 있습니다.
GET_SYSDATE		VARCHAR(14)	시스템 날짜를 추출 할 수 있습니다.
GET_SYSNAME	VARCHAR(20) : 시스템명 변환 컬럼	VARCHAR(400)	구성(CM) 테이블 데이터 기반으로 계층형일 경우 하나의 시스템명으로 표기 할 수 있습니다.
GET_TASNAME	VARCHAR(20) : 단계명 변환 컬럼	VARCHAR(200)	단계(041) 테이블 데이터 기반으로 단계명을 추출 할 수 있습니다.
get_tastype	VARCHAR(20) : 단계유형 변환 컬럼	VARCHAR(20)	단계(041) 테이블 데이터 기반으로 단계유형 ID를 추출 할 수 있습니다.
GET_TEXT	VARCHAR(10) : 언어 선택('kr','en') 컬럼 , VARCHAR(400) : 언어에 맞게 치환될 데이터	VARCHAR(400)	언어(054) 테이블 데이터 기반으로 언어를 치환 할 수 있습니다.
GET_WOGNAME	VARCHAR(20) : 작업그룹명으로 변환 컬럼	VARCHAR(400)	작업그룹(055) 테이블 데이터 기반으로 작업그룹명을 치환 할 수 있습니다.

EGENE Oracle DB Function

API 가이드

내장함수

ajax 통신을 통해 특정 sql을 실행하여 DB 데이터의 CRUD 후 json으로 결과값을 반환하도록 정의되어있습니다.

getBizData(sql_id, params)

select 문을 조회하여 json object 로 반환

```
getBizData("UllItem.Person.Search", "emp_id=testadmin");
```

샘플 EGENE DOCUMENT

```
getBizData("UllItem_Person_Search", "emp_id=testadmin");

// 실행결과

{

  "emp_title_name": "관리자",

  "emp_name": "테스트관리자",

  "dpt_name": "STEG",

  "row_cnt": 1,

  "emp_no": "testadmin",

  "dpt_id": "100000",

  "emp_id": "testadmin"

}
```

getBizArray(sql_id, params)

select 문을 조회하여 json array 로 반환

```
getBizData("UllItem.Person.Search", "emp_nm=테스트");
```

샘플

```
getBizArray("UllItem_Person_Search", "emp_nm=테스트");

// 실행결과

{

  "rows": [

    {

      "emp_title_name": "",

      "emp_name": "01. 현업-요청자",

      "dpt_name": "STEG",

      "emp_no": "test010",

      "dpt_id": "100000",

      "emp_id": "test010"

    },

    {

      "emp_title_name": "팀원",

      "emp_name": "t_01. 현업-요청자_1",

      "dpt_name": "",

      "emp_no": "t_user_test010_1",

      "dpt_id": "",

      "emp_id": "t_user_test010_1"

    }

  ],

  "row_cnt": 2

}
```

setBizData(sql_id, params)

insert/update/delete 문을 단건 수행한 결과 count를 json object로 반환

```
setBizData('TEST.test1', 'key=SR2208-00204');
```

샘플

```
setBizData('TEST_test1', 'key=SR2208-00204');

// 실행결과

{cnt: 1}
```

EGENE DOCUMENT

setBizArray(sql_id, params)

insert/update/delete 문을 다건 수행한 결과 count를 json object로 반환

```
setBizArray('TEST.test4', 'tmp_id=TMP2207-00300&reg_dttm=20220803080510&req_title=테스트입니다.');
```

샘플

```
setBizArray('TEST_test4', 'tmp_id=TMP2207-00300&reg_dttm=20220803080510&req_title=테스트입니다.');// 실행결과{cnt: 2}
```

E-GENE API SQL

ajax 통신을 통해 특정 sql을 실행하여 DB 데이터의 CRUD 후 json으로 결과값을 반환하도록 정의되어있습니다.

** \$service.ajax(reqData, handler, disableMsg) , \$service.ajaxHidden(reqData, handler, disableMsg) 는 E-GENE 에 정의된 내부 함수이며, ajax 호출하도록 구성되어있다. 둘의 차이는 로딩바 표시 여부 이며, ajax 는 표시 ajaHidden 은 미표시 한다.*

/api/egene/sql/

select 문을 조회하여 json array 로 반환

샘플

```
$service.ajax({url: '/api/egene/sql/UllItem_Person_Search',type: 'POST',data: {emp_id: 'testadmin'}}, function (result) {console.log(result);});// 실행결과[ {  "emp_title_name": "관리자",  "emp_name": "테스트관리자",  "dpt_name": "STEG",  "emp_no": "testadmin",  "dpt_id": "100000",  "emp_id": "testadmin"}]
```

/api/egene/sql/exec/

insert/update/delete 문을 다건 수행한 결과 상태를 json object로 반환

샘플

EGENE DOCUMENT

```
service ajax({
  url: '/api/egene/sql/exec/TEST_test4',
  type: 'POST',
  data: {
    tmp_id: 'TMP2207-00300',
    reg_dttm: '20220803080510',
    req_title: '테스트입니다'
  }
}, function (result) {
  console.log(result);
});

// 실행결과
{
  "status": "000",
  "error_code": "LIT000",
  "message": "",
  "data": null,
  "control": null,
  "nextScript": null,
  "mobileNextScript": null
}
```