Wikiprint Book

Title: Import rejestrów

Subject: eDokumenty - elektroniczny system obiegu dokumentów, workflow i CRM - UserGuide/AdvancedConfiguration/CustomRegisters/Import

Version: 3

Date: 04/25/25 00:32:34

Table of Contents

Import rejestrów

tworzymy ręcznie tabelę (rejestr) w bazie w schemacie *cregisters* z dziedziczeniem po *cregisters.register_entry*. **Nazwa tabeli musi posiadać przedrostek "creg_"**
przykład:

```
CREATE TABLE cregisters.creg_przekaz (
 id INT NOT NULL ,
 nazwisko VARCHAR(50) NULL ,
 imie VARCHAR(50) NULL ,
 ulica VARCHAR(50) NULL ,
 nr_domu VARCHAR(50) NULL ,
 nr_mieszkania VARCHAR(50) NULL ,
 uwagi TEXT
)
INHERITS (cregisters.register_entry)
WITH (OIDS=FALSE);
```

1. Dane (rekordy) możemy załadować w dowolnym momencie.
1. Funkcja zakładająca definicję rejestru i pól na podstawie struktury tabel (zamiast tworzenia definicji ręcznie). Zakłada tylko dla rejestrów które nie mają jeszcze definicji.

```
create or replace function my_exec1(text) returns void as $body$
begin
execute $1;
end;
$body$ language plpgsql;


CREATE OR REPLACE FUNCTION creg_get_field_type(text) RETURNS text AS $$
SELECT CASE $1
       WHEN 'bool' THEN 'bool'
       WHEN 'timestamp' THEN 'datetime'
       WHEN 'numeric' THEN 'text'
       WHEN 'int2' THEN 'integer'
       WHEN 'int4' THEN 'integer'
       WHEN 'int8' THEN 'integer'
       WHEN 'float' THEN 'float'
       WHEN 'float8' THEN 'float'
       WHEN 'varchar' THEN 'string'
       WHEN 'timestamp' THEN 'datetime'
       WHEN 'timestamptz' THEN 'datetime'
       ELSE $1
       END;
$$ LANGUAGE 'sql' STRICT IMMUTABLE;


SELECT my_exec1('INSERT INTO cregisters.register (name__,tabnam,label_) VALUES ('''||replace(tablename,'creg_','')||''','''
FROM pg_tables
WHERE schemaname = 'cregisters' AND NOT EXISTS (SELECT 1 FROM cregisters.register WHERE tabnam = tablename) AND tablename


SELECT my_exec1('INSERT INTO cregisters.register_fields (cregid,name__,label1,label2,type__) VALUES ('||cregid||','''||nam
FROM (
       SELECT reg.id____ as cregid, lower(a.attname::text) as name__, creg_get_field_type(typ.typname) as type__
       FROM pg_attribute a
       LEFT JOIN pg_type typ ON typ.oid = a.atttypid
       LEFT JOIN pg_index p ON p.indrelid = a.attrelid AND a.attnum = ANY(p.indkey)
       LEFT JOIN pg_description d ON d.objoid  = a.attrelid AND d.objsubid = a.attnum
       LEFT JOIN pg_attrdef f ON f.adrelid = a.attrelid  AND f.adnum = a.attnum
```

```
       LEFT JOIN pg_inherits inh ON (inh.inhrelid = a.attrelid)
       LEFT JOIN cregisters.register reg ON (('cregisters.' || reg.tabnam) = (inh.inhrelid::regclass)::text)
       WHERE reg.id____ IS NOT NULL AND a.attnum > 0 AND NOT a.attisdropped AND inh.inhparent = 'cregisters.register_entry
               AND NOT EXISTS (SELECT 1 FROM cregisters.register_fields rf WHERE rf.cregid = reg.id____)
       ORDER BY reg.id____, a.attnum
) bb
;

SELECT my_exec1('GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE ' || (inhrelid::regclass)::text || ' TO http;')
FROM pg_inherits
WHERE inhparent = 'cregisters.register_entry'::regclass;


drop function my_exec1(text);
drop function creg_get_field_type(text);
```