Rejestry

Tworzenie rejestru

Aby założyć rejestr w module rejestry musimy rozpocząć od założenia tabeli. Tabelę tworzymy za pomocą komendy *create table*.

Przykład założenia tabeli

```
-- Table: cregisters.creg_r_imi

-- DROP TABLE cregisters.creg_r_imi;

CREATE TABLE cregisters.creg_r_imi
(
-- Odziedziczony from table cregisters.register_entry:  id____ integer NOT NULL DEFAULT nextval('cregisters.register_entry
-- Odziedziczony from table cregisters.register_entry:  cregid integer,
-- Odziedziczony from table cregisters.register_entry:  uid___ text,
-- Odziedziczony from table cregisters.register_entry:  is_del boolean NOT NULL DEFAULT false,
-- Odziedziczony from table cregisters.register_entry:  adduid integer,
-- Odziedziczony from table cregisters.register_entry:  adddat timestamp with time zone DEFAULT now(),
-- Odziedziczony from table cregisters.register_entry:  lm_uid integer,
-- Odziedziczony from table cregisters.register_entry:  lm_dat timestamp with time zone DEFAULT now(),
-- Odziedziczony from table cregisters.register_entry:  doc_id integer,
-- Odziedziczony from table cregisters.register_entry:  prc_id integer,
-- Odziedziczony from table cregisters.register_entry:  cre_id integer,
 vorgnr character varying(25) NOT NULL,
 rwa character varying(20) NOT NULL,
 zrodlo character varying(50),
 rodzaj character varying(50),
 terminod timestamp without time zone,
 termindo timestamp without time zone,
 nazwa character varying(250),
 ulica character varying(80),
 budnr character varying(10),
 kwota double precision,
 archiv boolean NOT NULL DEFAULT false,
 status smallint NOT NULL DEFAULT 0,
 edtuser character varying(50),
 edtdate timestamp without time zone,
 id serial NOT NULL
-- Odziedziczony from table :  tpstid integer,
-- Odziedziczony from table :  stcuid integer,
-- Odziedziczony from table :  stcdat timestamp with time zone
)
INHERITS (cregisters.register_entry)
WITH (
 OIDS=FALSE
);
ALTER TABLE cregisters.creg_r_imi
 OWNER TO edokumenty;
GRANT ALL ON TABLE cregisters.creg_r_imi TO edokumenty;
GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE cregisters.creg_r_imi TO http;
GRANT SELECT, UPDATE ON TABLE cregisters.creg_r_imi TO useraspbip;
```

Uwaga ! Pola używane przez eDokumenty są dziedziczone - nie trzeba ich zakładać. Są to pola:

```
id____, cregid, uid___, is_del, adduid, adddat, lm_uid, lm_dat, doc_id, prc_id, cre_id, tpstid, stcuid, stcdat
```

Uwaga ! Nie zapominamy o nadaniu uprawnień do tabeli dla edokumenty, useraspbip oraz http

Następnie w module rejestry zakładamy nowy rejestr, a w polu nazwa tabeli wprowadzamy nazwę założonej tabeli. System będzie od nas żądać, aby nazwa tabeli rozpoczynała się od "creg_".

Tworzenie raportu

Po utworzeniu tabeli w schema cregisters rozpoczynającej się od ciągu creg_ należy utworzyć raport np:

```
SELECT ('CREGISTER_ENTRY') AS clsnam, cd.id____ AS keyval, cd.*
FROM cregisters.creg_ddm_dokumenty cd
INNER JOIN cregisters.creg_archiv_formularz af ON cd.menuid = af.formularz
WHERE {FILTER_STRING} AND cd.is_del IS NOT true
{ORDER_BY}
{LIMIT}
```

Raport należy podlinkować do rejestru ustawiając w tabeli registers pole rep_id.

Ważne tabele

```
registers (klucz główny: )
registers_entry (klucz głowny: , klucz obcy: )
register_fields (klucz główny: , klucz obcy: )
```

Uwaga! Kluczem obcym w registers_entry referujacym do rejestru jest XXXX

Rejestr jako lista w dokumencie

Definiujemy powiązanie rejestru z typem dokumentu (jeżeli na to być lista a nie formularz to ustawiamy parametr *collection* na *true*):

```
INSERT INTO cregisters.register_links (cregid, keyval, clsnam, params)
VALUES ({cregisters.register.id____}, {types_of_documents.dctpid}, 'DOCUMENT', '{"collection":true}')
```

Podrejestr w rejestrze

Aby zbudować strukturę hierarchiczną rejestru wystarczy zlinkować odpowiednio 2 wcześniej utworzone rejestry. Pierwszy ze wskazanych zacznie się pojawiać jako lista rekordów w formatce rejestru nadrzędnego.

```
INSERT INTO cregisters.register_links (cregid, keyval, clsnam, params)
VALUES ({cregisters.register.id____}, {cregisters.register.id____}, 'CREGISTER', '{"collection":true}')
```

Uwaga! Id podrejestru jest wprowadzany w insercie jako pierwsze, następny jest id rejestru do którego będzie należeć podrejestr.

W raporcie w podrejestrze za filtrowanie rekordów odpowiada makro {FILTER_STRING}, które dokleja do zapytania warunek po atrybucie cre_id (cre_id wskazuje na rekord rejestru nadrzędnego).

Definicje pól dla rejestru

Ustawianie wartości domyślnych

Jeżeli chcemy aby pole było listą wyboru, to definiujemy w parametrach (register_fields.params) domyślną wartość (defaultValue):

```
-- Id tworzącego dokument
{"defaultValue":"{SQL::SELECT adduid FROM documents WHERE doc_id = {doc_id}}"}

-- domyślne dane zalogowanego użytkownika
{"defaultValue":"{SQL::SELECT o.firnam || ' ' || o.lasnam || ' (' || COALESCE(o.orunsm, '') || ' - ' || o.ndenam || ')' AS
```

Możliwe jest też ustawienie wartości wyliczanej za każdym razem gdy dokonujemy zapisu rejestru (dla pól ukrytych):

```
-- Imię i nazwisko dokonującego zmian w rejestrze
{"value":"{SQL::select firnam || ' ' || lasnam from users where usr_id={LOGGED_USR_ID}}"}
```

1. **defaultValue** jest parsowane tylko dla formularza nowego wpisu w rejestrze (na akcji Open oraz Save).

   value jest parsowane zawsze na akcji Save niezależnie od trybu (edycja, nowy) wyłącznie dla pól:

   1. ukrytych poprzez definicję pola (register_fields.hidden = TRUE)
   2. ukrytych poprzez parametr visible (register_fields.params = {"visible":false})
   3. nieaktywnych (register_fields.params = {"enabled":false})

Pole jako lista wyboru

Jeżeli chcemy aby pole było listą wyboru, to definiujemy w parametrach (register_fields.params) zapytanie zwracające rekordy typu (klucz,wartość), dodatkowo ustawiamy domyślną wartość (defaultValue):

```
{"sql":"SELECT usr_id,usrnam FROM users WHERE is_del IS NOT TRUE", "defaultValue":"{SQL::SELECT adduid FROM documents WHER
```

Parametry: sql, defaultValue, są objęte standardowym mechanizmem parsowania parametrów (tak jak np. w przypisaniach w workflow).

Pole tekstowe typu HTML

```
{"type":"html"}
```

Pole tekstowe typu ComboBox

```
{"type":"combobox","autoSearch":2,"sql":"SELECT usr_id,usrnam FROM users WHERE is_del IS NOT TRUE AND (firnam ~* E'^{SEARC
```

Znacznik {SEARCH_TEXT} zostanie zastąpiony wpisanym w pole tekstem

1. autoSearch - ilosc znaków po których wpisaniu zostanie uruchomione wyszukiwanie / podpowiadanie (wartość -1 spowoduje wyłączenie automatycznego wyszukiwania i pokazanie ikony lupki)

Pole jako status

W definicji pola, w polu Alias wpisujemy "tpstid"

Disablowanie pola

Jeśli pole ma być tylko do odczytu to należy dla niego określić atrybut enabled:

```
{"enabled":false}
```

ToolBar

```
{"type":"toolbutton","icon":"new.gif","visible":1,"doRefresh":true,"onclick":["moj_skrypt.inc","MojaKlasa1","mojaFunkcja",
```

1. icon: plik ikony bez ścieżki która wskazuje domyślnie na ./img/toolbarIcons/24x24/

Skrypt "app/edokumenty/scripts/moj_skrypt.inc"

1. doRefresh: wartość true spowoduje przeładowanie formularza wpisu w rejestrze

```
<?php
class MojaKlasa1 {

   public function __construct() {
   }

   public function mojaFunkcja($params) {
       $params = json_decode($params,TRUE);
```

```
      jscript::alert(json_encode($params));
   }
}
?>
```

Wywołanie / otwarcie formularza poprzez clsnam i keyval (np. otwarcie tego samego wpisu w nowym oknie czyli edycja):

```
{"type":"toolbutton","icon":"edit.gif","enabled":1,"onclick":["","Application","openDialogByCls","","CREGISTER_ENTRY","SQL
```

Usuń wpis z rejestru:

```
{"type":"toolbutton","icon":"del.gif","enabled":1,"onclick":["","Application","openDialogByCls",{"mode":"del"},"CREGISTER_
```

Filtrowanie listy

Dla rejestru można ustawić stały filtr w parametrach (cregisters.register.params)

```
{"FILTER_STRING":"is_del IS TRUE"}
```

Modyfikacje JSON bezpośrednio w bazie danych

Sposób na zmianę wartości jednego pola w obiekcie typu JSON (**dla PostgreSQL v9.3+**):

```
CREATE OR REPLACE FUNCTION "json_set_value"(
 "json"          json,
 "key_to_set"    TEXT,
 "value_to_set"  anyelement
)
 RETURNS json
 LANGUAGE sql
 IMMUTABLE
 STRICT
AS $function$
SELECT COALESCE(
 (SELECT ('{' || string_agg(to_json("key") || ':' || "value", ',') || '}')
    FROM (SELECT *
            FROM json_each("json")
           WHERE "key" <> "key_to_set"
           UNION ALL
          SELECT "key_to_set", to_json("value_to_set")) AS "fields"),
 '{}'
)::json
$function$;

UPDATE cregisters.register_field SET params = json_set_value(params, 'doRefresh', true) WHERE id____ = 1;

UPDATE cregisters.register_field SET params = json_set_value(params, 'value', 'SQL::SELECT ''tekst "kolo"''') WHERE id____
```

Migracja rejestrów z innej bazy

Import rejestrów

Przydatne konstrukcje i zapytania

```
-- użycie w parametrach do przycisków i pól wartości {DOC_ID} powoduje błąd po wejściu na rekord rejestru jeśli jest pusty
select pprosm from documents where doc_id = COALESCE(NULLIF('{DOC_ID}',''),'0')::int
```