

## [Przewodnik użytkownika](#) > Dodatkowe moduły i zakładki

Informacje o wprowadzenie funkcjonalności:

Wersja systemu	Wersja modułu/funkcji	Data kompilacji	Zmiany	Opis
5.0	0.4	18.11.2015	Zmiana	Dodano filtr (globalny oraz dla raportu) Projektu oraz obsługę parametru {PROJID}
4.8.100, 4.10.16, 4.11.2	0.3	18.11.2015	Zmiana	Dodano możliwość definicji separatora dla przycisków jako tag <separator/>

### Menu

1. [Dodatkowe zakładki](#)
  - 1.1 [Ograniczanie widoczności zakładek](#)
2. [Dodatkowe moduły](#)
  - 2.1 [Definiowanie dodatkowych modułów](#)
  - 2.2 [Ograniczanie widoczności modułów](#)
  - 2.3 [Filtry](#)
  - 2.4 [Ikona modułu](#)

### Dodatkowe zakładki

System eDokumenty umożliwia dodawanie customowych zakładek do kartoteki

- sprawy
- kontrahenta
- urządzenia
- dokumentu

Zakładki te oparte są na raportach oraz pliku konfiguracyjnym xml znajdującym się w

```
$APP_PATH\var\tpl\tabs
$APP_PATH oznacza /home/edokumenty/public_html/apps/edokumenty
lub
C:\Program files\BetaSoft\eDokumenty\public_html\apps\edokumenty
```

Jeśli katalog var\tpl\tabs jest pusty należy skopiować szablony plików xml z \$APP\_PATH\var\tpl\_default\tabs.

Nazwy obsługiwanych plików xml

- sprawy - process\_tpl.xml
- kontrahenta - contact\_tpl.xml
- urządzenia - device\_tpl.xml
- dokumentu - document\_tpl.xml
- ewidencji - evidence\_tpl.xml
- pracownik - user\_tpl.xml
- produkt - product\_tpl.xml
- profil użytkownika - user\_profile\_tpl.xml

Aby dodać dodatkową zakładkę do jednej z wyżej wymienionych kartotek należy utworzyć odpowiedni raport oraz wyedytować wybrany plik xml dla danej kartoteki.

Definicja pliku xml

```

<?xml version="1.0" encoding="UTF-8"?>
<tabs>
  <tab label="" rep_id="" grp_id="" def_tab="1" alwaysVisible="1" forceOpen="1" prior="0">
    <buttons>
      <button>
        <id>
          new
        </id>
        <label>
          Nowa
        </label>
        <dscript>
          Nowa
        </dscript>
        <onclick>
          App.openDialogByCls({CLSNAM}, {KEYVAL}, ({afterSubmit:'{AFTER_SUBMIT}', mode:'new'}).toJSONString())
        </onclick>
        <icon>
          new.png
        </icon>
      </button>
      <separator/>
      <button>
        <custom_widget>
          3
        </custom_widget>
      </button>
    </buttons>
  </tab>
</tabs>

```

#### Dodatkowe tokeny do użycia w tagu onclick:

Token	Opis
{CLSNAM}	klasa zaznaczonego elementu na liście jeśli zostało podane pole w raporcie
{KEYVAL}	identyfikator zaznaczonego elementu na liście jeśli zostało podane pole w raporcie
{KEYVALS}	identyfikatory zaznaczonych elementów
{AFTER_SUBMIT}	akcja odświeżenia widoku po edycji wpisu lub np. jako dodatkowa akcji na pasku zadań
{MODE}	wartość pobierana z taga <id></id> używać jako new, edit, del, delete

Od wersji 4.1-alfa51 istnieje możliwość zadeklarowania przyciski jako custom widget

(<http://support.edokumenty.eu/trac/wiki/DeployerGuide/Others/CustomWidgets>). Tag button musi zawierać informację w postaci taga custom\_widget o wartości id danego widgeta czyli kolumna cswgid. Tabela custom\_widget powinna zawierać rekord zgodnie z definicją w dokumentacji a wartość w kolumnie c\_path to

```
c_path = custom/cswgid czyli custom/3
```

gdzie 3 to identyfikator rekordu. Id zaznaczonych rekordów są przekazywane pod parametrem keyval.

Od wersji 4.3.23 obsługiwany jest atrybut def\_tab dla taga tab. Oznacza on, że dana zakładka ma być domyślnie otwierana. Działa tylko dla customowych modułów. W przypadku customowych zakładek dla kartotek ten atrybut nie działa.

#### Widoczność zakładki

Dodatkowo od wersji 4.6.4, 4.7.2 dla zakładki możemy zdefiniować czy ma ona się pokazać domyślnie na pasku zakładek czy po kliknięciu "więcej...". Dokonujemy tego poprzez dodanie atrybutu `alwaysVisible` dla taga `tab` podobnie jak `def_tab`.

## Wymuszenie otwarcia jako domyślna zakładka

Zdarza się, że chcemy ustawić dodatkową zakładkę jako domyślnie otwartą w danej kartotece podczas jej otwierania. Standardowo ustawienie, która zakładka ma się otworzyć jest przypisane w kartotece na stałe. Zachowanie można to zmienić i wymusić otwarcie dodatkowej zakładki poprzez atrybut `forceOpen="1"`. Atrybut ten można ustawić tylko dla 1 zakładki.

## Ustawienie na odpowiedniej pozycji

Domyślnie zakładki są dodawane w kolejności wpisania ich w definicji xml. Zachowanie to można zmienić i ustawić zakładki w dowolnej kolejności (nawet pomiędzy standardowymi zakładkami) poprzez atrybut `prior` (od 0 do n gdzie n <= liczbie zakładek). Atrybut ten w połączeniu z `forceOpen` umożliwia nam dodanie nowej domyślnie otwartej zakładki na pierwszej pozycji w kartotece.

Zakładka dla dokumentu dodatkowo przyjmuje parametr `dctpid` (ID typu dokumentu) np.

```
<tab label="zakładka 1" rep_id="1" dctpid="1">
```

lub

```
<tab label="zakładka 1" iframe="http://www.onet.pl"></tab>
```

Gdzie:

- `rep_id` - identyfikator raportu z systemu eDokumenty
- `iframe` - url do strony, którą chcemy wyświetlić w eDokumentach w zakładce

Dodatkowo do taga `onclick` oraz atrybutu `iframe` można dodawać parametry z postaci danych z aktualnie otwartej kartoteki. Dla przykładu niech posłuży kartoteka produktu oraz dodatkowa zakładka w postaci `iframe` wyświetlającego katalog zdjęć:

```
<tab label="Zdjęcia" iframe="http://www.moj.serwer/produkty/?symbol={symbol}"></tab>
```

Token `{symbol}` zostanie zamieniony na `symbol` produktu z aktualnie otwartej kartoteki.

Zakładka pojawi się tylko w dokumentach o `dctpid = 1`. Brak tego parametru spowoduje dodanie zakładki dla wszystkich typów. Można też podać więcej identyfikatorów typu po przecinku (np. `dctpid="1,3,5,6,9"`).

Zakładka dla sprawy dodatkowo przyjmuje parametr `dos_id` (ID teczki z wykazu akt) np.

```
<tab label="zakładka 1" rep_id="1" dos_id="1">
```

Zakładka pojawi się tylko w sprawach o `dos_id = 1`. Brak tego parametru spowoduje dodanie zakładki dla wszystkich spraw. Można też podać więcej identyfikatorów typu po przecinku (np. `dos_id="1,3,5,6,9"`).

Przejdź do [Menu](#)

## Ograniczanie widoczności zakładek

Widoczność Zakładki może również być ograniczona poprzez parametr `grp_id` (np. `grp_id="2,5,10"`) który ograniczy widoczność zakładki wyłącznie do członków wymienionych po przecinku `grup`.

Przejdź do [Menu](#)

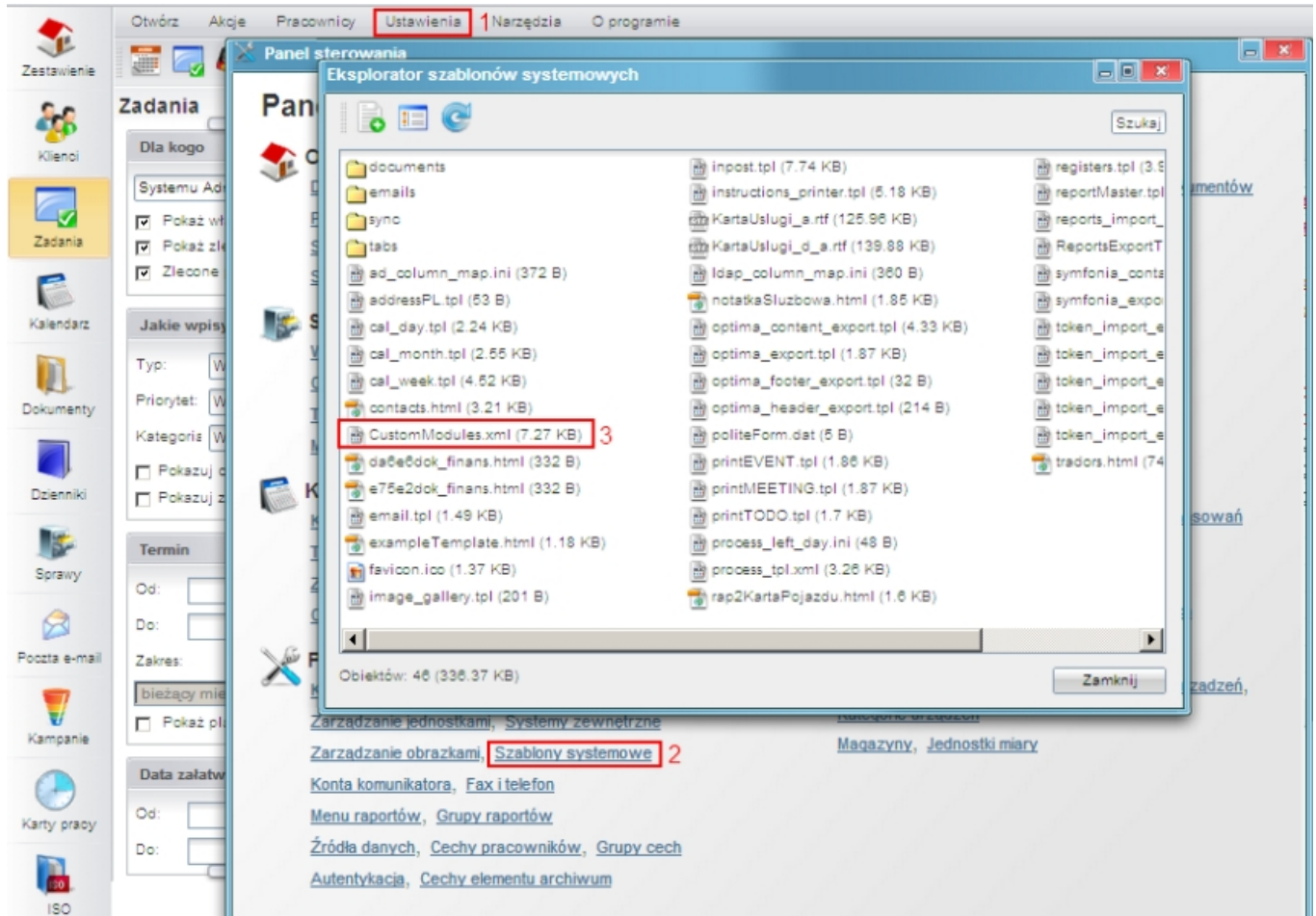
## Dodatkowe moduły

System eDokumenty umożliwia tworzenie własnych modułów w oparciu o podobny mechanizm.

## Definiowanie dodatkowych modułów

W systemie można również skonfigurować w oparciu o ten sam mechanizm własny moduł. Na wersji demonstracyjnej moduły dostępne przez użytkownika *jmamon* "Delegacje" oraz "Urlopy" dla użytkownika *serwis* są utworzone poprzez utworzenie następującego pliku w katalogu `$APP_PATH/var/tpl/CustomModules.xml`

Edycja tego pliku nie jest konieczna z poziomu konsoli. Dostęp do tego pliku jest możliwy z poziomu edokumentów. W Panelu sterowania szukamy odnośnika Szablony Systemowe. Uruchomi się okienko z plikami *Szablonów Systemowych*



(Szablony systemowe)

W okienku tym wyszukujemy plik "CustomModules.xml", klikamy na niego, a następnie zapisujemy na dysku. Edytujemy zmiany, po czym usuwamy istniejącą na serwerze plik i wgrujemy nową wersję.

```
<?xml version="1.0" encoding="UTF-8"?>
<module id="cModule_U" name="Moduł" label="Mój moduł" icon="fixdefect-48.png" color="#c0392b" right="bswfms.processes">
  <toolbar>
  </toolbar>
  <filters>
  </filters>
  <tabs>
    <tab label="AKTYWNE" rep_id="1">
      <buttons>
      </buttons>
    </tab>
    <tab label="WSZYSTKIE" rep_id="2">
      <buttons>
        <button>
          <id>refresh26</id>
          <label>Odśwież</label>
          <dscript>Odśwież</dscript>
        </button>
      </buttons>
    </tab>
  </tabs>
</module>
```

```

        <onclick>
        {AFTER_SUBMIT}
        </onclick>
        <icon>sync-24.png</icon>
        </button>
    </buttons>
</tab>
</tabs>
</module>

```

```

<?xml version="1.0" encoding="UTF-8"?>
<modules>
  <module id="Urlopy" name="Urlopy" label="Urlopy"
  icon="travel.png" right="bswfms.custom_modules.vacations">
    <toolbar></toolbar>
    <filters></filters>
    <tabs>
      <tab label="Moje wnioski" rep_id="95">
        <buttons>
          <button>
            <id>new</id>
            <label>Nowy</label>
            <dscrpt>Nowy wniosek urlopowy</dscrpt>
            <onclick>
              App.openDialogByCls('DOCUMENT', null,
({afterSubmit: '{AFTER_SUBMIT}', dctpid:21, dctptp: 'CustomDocument', mode: 'new'}).toJSONString())
            </onclick>
            <icon>new.png</icon>
          </button>
          <button>
            <id>edit</id>
            <label>Edycja</label>
            <dscrpt>Edytuj wniosek</dscrpt>
            <onclick>
              App.openDialogByCls('DOCUMENT', {KEYVAL},
({afterSubmit: '{AFTER_SUBMIT}', dctpid:21, dctptp: 'CustomDocument', mode: 'edit'}).toJSONString())
            </onclick>
            <icon>edit.png</icon>
          </button>
          <button>
            <id>delete</id>
            <label>Usuń</label>
            <dscrpt>Usuń</dscrpt>
            <onclick>
              App.openDialogByCls('DOCUMENT', {KEYVAL},
({afterSubmit: '{AFTER_SUBMIT}', dctpid:21, dctptp: 'CustomDocument', mode: 'del'}).toJSONString())
            </onclick>
            <icon>delete.png</icon>
          </button>
        <separator/>
        <button>
          <id>report</id>
          <label/>
          <dscrpt>Raport</dscrpt>
          <onclick>window.open('litescript.php/demo/?script=GenerateReport&ent_id=2&ent_name=Demo S.A.&rep_id
          <icon> reports.png</icon>
        </button>
        <button>
          <id>xls</id>

```

```

        <label/>
        <dscrpt>XLS</dscrpt>
        <onclick> window.open('litescript.php/demo/?script=GenerateReport&ent_id=2&ent_name=Demo S.A.&rep_i
        <icon>xls.png</icon>
    </button>
    <button>
        <id>pdf</id>
        <label/>
        <dscrpt>PDF</dscrpt>
        <onclick> window.open('litescript.php/demo/?script=GenerateReport&ent_id=2&ent_name=Demo S.A.&rep_i
        <icon> pdf.png</icon>
    </button>
</buttons>
</tab>
</tabs>
</module>

</modules>

```

Funkcję `openDialogByCls()` można wywoływać z innymi parametrami zamiast 'DOCUMENT', co spowoduje otwarcie innych typów okien. Poniżej lista najbardziej przydatnych:

- CONTACT - Okienko edycji kontaktu (wpisujemy CONTACT\_EDIT)
- PROCESS - Okienko edycji sprawy
- RCP - Karta pracy
- MEETING - Spotkanie
- EVENT - Termin
- TODO - Zadanie
- PHONECALL - Rozmowa telefoniczna
- DEVICE - Urządzenie
- PRODUCT - Produkt

Przykład zakładania nowego kontaktu (osoba fizyczna):

```
App.openDialogByCls('CONTACT_EDIT', null,({afterSubmit:'{AFTER_SUBMIT}', phyper:'true' }).toJSONString())
```

Przykład zabezpieczenia w przypadku gdy raport zwróci zero wyników a na zakładce w module będzie dostępny przycisk Edytuj:

```
var k = {KEYVAL}; if (k) {App.openDialogByCls('DOCUMENT', {KEYVAL},({afterSubmit:'{AFTER_SUBMIT}', dctpId:24,dctptp:'Custom
```

Przykład tworzenia nowej umowy z uzupełnionym klientem na podstawie zestawienia (keyval):

```

<button>
    <id>umowa1</id>
    <label>Nowa umowa</label>
    <dscrpt>Nowa umowa</dscrpt>
    <onclick>App.openDialogByCls('DOCUMENT', null,
        ({afterSubmit:'{AFTER_SUBMIT}', dctpId:3,dctptp:'Contract',"contid":{KEYVAL} , mode:'new'}).toJSONString(
    </onclick>
    <icon>add.png</icon>
</button>

```

### Otwieranie modułu w nowym oknie

Przykład w javascript:

```
window.open('litescript.php/aps/?script=NewWindowModule&ent_id=2&modnam=CustomModule&submodule=cExactoris', '_blank', 'to
```

## Przycisk tworzący sprawę

Przyciski mogą też do listy parametrów obsługiwać klucze z bean-ów, dla przykładu:

Przycisk tworzący sprawę - pełna formatka formatka tworząca sprawę

```
App.openDialogByCls('PROCESS_EDIT', null, ({afterSubmit:'{AFTER_SUBMIT}', clsnam:'DOSS', keyval:39, mode:'new' }).toJSONS
```

- keyval przechowuje IDteczki

Przycisk tworzący sprawę - prosta formatka tworząca sprawę

```
App.createDialog('createProcessForm', 'SimpleProcessCreatingForm', './modules
/AProcesses/forms/SimpleProcessCreatingForm.inc', 'Zakadanie', '513',
({clsnam:'DOSS', strpid:351, devcid:'{devcid}', contid:'{contid}'}).toJSONString(), null, 'fast')
```

Opis parametrów przycisku tworzącego sprawę

- createProcessForm - nazwa w kodzie HTML okna
- SimpleProcessCreatingForm - nazwa klasy php'owej
- ./modules/AProcesses/form... - ścieżka do tej klasy
- Sprawa - opis na górnej belce formatki (najczęściej jest nadpisywany przez kod klasy)
- 513 - rodzaj formatki 513 to pełna z przyciskami na górnej belce inne rodzaje (kody) raczej są przeznaczone do wew. wywołań
- ({clsnam:'DOSS', strpid:1253, keyval:776, mode:'new'}).toJSONString() - parametry do formatki w formacie JSON

Ostatnie dwa parametry zawsze są takie same.

Dodatkowo od wersji 4.0 obsługiwany jest parametr {KEYVALS}, który odpowiada ze przekazanie do dialoga wszystkich zaznaczonych elementów z listy z nie tylko pierwszego jak to ma miejsce w przypadku parametru {KEYVAL}.

Parametr strpid jest dostępny w **Ustawienia -> Panel sterowania -> Sprawy -> Wyciąg z wykazu akt -> Kolumna Miejsce (strpid)** W przypadku jeśli kolumna ta nie jest włączona należy kliknąć na ikonę "Widoczne kolumny" (na dole listy) i zaznaczyć ptaszka.

Sprawa automatycznie otrzyma atrybuty id urządzenia oraz id kontrahenta urządzenia. Na razie zaimplementowano w urządzeniu.

## Przekazywanie parametrów

Od wersji 4.4.40, 4.6 oraz 4.5.24 w przypadku przycisku customowego w konfigurowanych zakładkach na kartotekach np. sprawy istnieje możliwość przekazania parametrów z beana sprawy podobnie jak do akcji onClick.

Wymagane jest aby w definicji przycisku w polu Parametry podać odpowiedni klucz - params. Przykład poniżej:

```
{
  "script": "Test.inc",
  "image": "24x24\deadline.png",
  "params": "prc_id: '{prc_id}', dsexid: '{dsexid}'"
}
```

W momencie generowania przycisku na zakładce klucze w nawiasach klamrowych zostaną zastąpione na właściwe atrybuty sprawy. Następnie cały ciąg z klucza params zostanie dołączony do wywołania widgeta. Po stronie widgeta w tablicy \$args będą dostępne klucze jak podano w params - w naszym przypadku będzie to prc\_id oraz dsexid.

Przejdź do [Menu](#)

## Przyciski operujące na rejestrze

W custom module można również operować na dowolnym rejestrze. Aby edytować rekord raport skojarzony z modulem musi posiadać klucze clsnam, keyval. Wówczas działać będzie przekazywanie wartości {KEYVAL}

```

<buttons>
<button>
<id>new</id>
<label>Nowy</label>
<dscrpt>Nowy wpis</dscrpt>
<onclick>
    App.openDialogByCls('REGISTER_ENTRY', null,
        ({afterSubmit:'{AFTER_SUBMIT}', mode:'new',cregid:1}).toJSONString())
</onclick>
<icon>new.png</icon>
</button>
<button>
<id>edit</id>
<label>Edycja</label>
<dscrpt>Edytuj wpis</dscrpt>
<onclick>
    App.openDialogByCls('REGISTER_ENTRY', {KEYVAL},
        ({afterSubmit:'{AFTER_SUBMIT}', mode:'edit',cregid:1}).toJSONString())
</onclick>
<icon>edit.png</icon>
</button>
<button>
<id>delete</id>
<label>Usuń</label>
<dscrpt>Usuń</dscrpt>
<onclick>
    App.openDialogByCls('REGISTER_ENTRY', {KEYVAL},
        ({afterSubmit:'{AFTER_SUBMIT}', mode:'del',cregid:1}).toJSONString())
</onclick>
<icon>delete.png</icon>
</button>
</buttons>

```

### Ograniczanie widoczności modułów

Prawa dostępu do modułów (inaczej widoczność modułów) należy zdefiniować w Panelu sterowania > Definicje uprawnień. Zaleca się utworzenie prawa bswfms.custom\_modules (Moduły dodatkowe). A następnie dodanie prawa podrzędnego do custom\_modules.

Następnym krokiem w procesie jest ustawienie praw dla modułów w pliku *CustomModules.xml* w definicji właściwości *rights* modułu

```

...
<module id="cModule_1" name="Delegacje" label="Delegacje"
icon="processes.png" right="bswfms.custom_modules.delegations">
...

```

gdzie wstawiamy wartość pola *define* danego prawa.

Po zdefiniowaniu praw w tabeli oraz w pliku *CustomModules.xml* należy przejść do menu *Pracownicy*, a następnie wybrać *Grupy*. Jeżeli danej grupy nie ma, to tworzymy ją. Po zapisaniu grupy pojawiają się nowe zakładki. Przechodzimy do zakładki *Pracownicy* gdzie wprowadzamy wybranych przez nas użytkowników, a następnie klikamy w zakładkę *Prawa do systemu*, gdzie ze struktury drzewiastej praw wybieramy odpowiednie prawo i przypisujemy dostęp do prawa.

Przejdź do [Menu](#)

### Filtry

Na chwilę obecną dostępne są następujące filtry, które można zdefiniować bezpośrednio z poziomu xml:

- MonthSelectorTreeFilter - wybór dat z drzewka (parametry {DATE\_FROM} i {DATE\_TO}); zakres dat dla drzewka wyznaczany jest jako przedział od min(documents.sysdat) do max(documents.adddat) dla modułów customowych oraz zakres min(documents.adddat) do max(documents.sysdat) dla modułu Ewidencja. W celu wygenerowania drzewka na większy okres należy dodać do systemu np. notatkę służbową z datą dodania np. 2016-12-12.



- ClearingUnitsFilter - wybór jednostki rozliczeniowej (parametr {ACORID})
- ProjectsFilter - wybór projektu (parametr {PROJID})

```
<filters>
  <filter name="Daty" type="MonthSelectorTreeFilter" height="auto" opened="0">
  </filter>
  <filter name="Jednostka rozliczeniowa" type="ClearingUnitsFilter" height="30px" opened="0">
  </filter>
</filters>
```

```
<filters>
  <filter name="Data delegacji" type="MonthSelectorTreeFilter" height="auto" opened="0" startDate="SELECT min(start_) FROM
</filter>
</filters>
```

Od wersji 4.5.8 następuje zmiana i dochodzą dwa nowe parametry

- {DATE\_FROM\_RANGE}
- {DATE\_TO\_RANGE}

Odpowiedni przechowują dane do swych odpowiedników bez końcówki RANGE. Modyfikacja ta ma na celu umożliwienie korzystanie z panelu filtrów oraz drzewka gdzie oba parametry były wypełniana przez 2 różne komponenty. Teraz jeśli zdefiniujemy parametr {DATE\_FROM} lub {DATE\_TO} to zostanie od pokazany na panelu filtrów i dodatkowo jeśli dodamy do tpl filter wtedy będą dodatkowe tokeny.

### Filtry statyczne

Filtry statyczne działają zarówno dla modułu jak i pojedynczej zakładki. Definicja takiego filtra jest taka sama jak powyżej z tą różnicą, że nazwa komponentu wskazuje na StaticFilter

```
<filters>
  <filter name="Daty" type="StaticFilter" token="{ANNA}">
    1=2
  </filter>
</filters>
```

Ważne w takim filtrze jest określenie tokena (osadzamy go w raporcie) w tym przypadku token="{ANNA}". Token może mieć dowolną nazwę. Wartość takiego filtra to stała wartość filtrująca. W tym przypadku jest to 1=2 czyli nigdy nic nie pokaże. Jednak można tu wstawić np.

```
d.name__ = 'Pompa'
```

gdzie d to alias na tabelę z raportu a name\_\_ to kolumna, która występuje w tabeli, która jest ukryta pod d.

Dodatkowe filtry są realizowane za pomocą definiowanych filtrów dla raportów zgodnie z dokumentacją:

[Filtry dla raportów](#)

Przejdź do [Menu](#)

### Ikona modułu / kolor

Ikony w formacie .png,.svg o maksymalnym rozmiarze 48px x 48px wrzucamy do katalogu public\_html/framework/img/PageToolBar. Kolor kodowy modułu, który jest też kolorem tła dla ikony, określamy w atrybucie color tagu module. np.:

```
<module id="cModule_U" name="Moduł" label="Mój moduł" icon="icon.png" color="#c0392b" right="bswfms.processes">
  ...
</module>
```

### Wywołanie raportu

Aby otworzyć raport w module customowym i przekazać zaznaczone elementy z listy jako parametry do raportu należy dodać wywołanie

```
App.openDialogByCls('REPORT_VIEW', {rep_id}, ({keyval:{KEYVALS}}).toJSONString())
```

gdzie {rep\_id} to id raportu.

W raporcie będzie dostępny parametr {KEYVAL}, który będzie przechowywał zaznaczone elementy dlatego dla wielu zaleca się użycie IN ({KEYVAL}).

Przykład wywołania raportu o rep\_id=23 z parametrem FILTER\_STRING do którego zostaną przekazane po przecinku zaznaczone elementy na liście. W raporcie tym należy umieścić w WHERE parametr {FILTER\_STRING}.

```
<tab label="Przydzielone" rep_id="121">
  <buttons>
    <button>
      <custom_widget>1</custom_widget>
    </button>
    <button>
      <id>Do maila</id>
      <label>Tabela</label>
      <dscrpt>Otwiera tabelę HTML</dscrpt>
      <onclick>
        var keyval = {KEYVALS};
        App.openDialogByCls('REPORT_VIEW', 23, ({filter_string:'fkclid IN (' + keyval + ')}).toJSONString());
      </onclick>
      <icon>table24.png</icon>
    </button>
    <button>
      <custom_widget>7</custom_widget>
    </button>
  </buttons>
</tab>
```

## Wykonywanie etapów workflow z przycisków

```
<tab label="WSZYSTKIE" rep_id="1">
  <buttons>
    <button>
      <id>selectFR</id>
      <label>Potwierdź</label>
      <dscrpt>Potwierdza etap workflow</dscrpt>
      <onclick>
        App.openDialogByCls('P_WORKFLOW', null, ({afterSubmit:'{AFTER_SUBMIT}', clsnam: 'PROCESS', keyval:{KEYVAL}, next_ptstid:336,
      </onclick>
      <icon>pass-24.png</icon>
    </button>
  </buttons>
</tab>
```

## Moduł operujący na istniejących obiektach

W poniższym przykładzie ewidencjonować można sprawy. Należy zmienić: dsexid - to id z wyciągu z wykazu akt, rep\_id - id raportu

```
<module id="cModule_U2" name="NEW" label="New" icon="fixdefect-48.png" right="bswfms.processes">
  <toolbar>
  </toolbar>
  <filters>
  </filters>
  <tabs>
    <tab label="AKTYWNE" rep_id="1">
      <buttons>
        <button>
```

```

<id>newbug</id>
<label>Zgłoszenie</label>
<dscrpt>Nowe zgłoszenie. Sprawa zostanie założona w teczce</dscrpt>
<onclick>
App.createDialog('createProcessForm', 'SimpleProcessCreatingForm', './modules/AProcesses/forms/SimpleProcessCreatingForm
</onclick>
<icon>fix-defect-24.png</icon>
</button>
<button>
  <id>refresh20</id>
  <label>Odśwież</label>
  <dscrpt>Odśwież</dscrpt>
  <onclick>
  {AFTER_SUBMIT}
  </onclick>
  <icon>sync-24.png</icon>
</button>
</buttons>
</tab>
<tab label="WSZYSTKIE" rep_id="2">
<buttons>
  <button>
    <id>refresh26</id>
    <label>Odśwież</label>
    <dscrpt>Odśwież</dscrpt>
    <onclick>
    {AFTER_SUBMIT}
    </onclick>
    <icon>sync-24.png</icon>
  </button>
</buttons>
</tab>
</tabs>
</module>

```

Przejdź do [Menu](#)