

Dodatkowe moduły i zakładki

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`

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">
    <buttons>
      <button>
        <id>
          new
        </id>
        <label>
          Nowa
        </label>
        <dscrpt>
          Nowa
        </dscrpt>
        <onclick>
          App.openDialogByCls({CLSNAM}, {KEYVAL}, ({afterSubmit:'{AFTER_SUBMIT}', mode:'new'}).toJSONString())
        </onclick>
      </button>
    </buttons>
  </tab>
</tabs>
```

```

        <icon>
            new.gif
        </icon>
    </button>
    <button>
        <custom_widget>
            3
        </custom_widget>
    </button>
</buttons>
</tab>
</tabs>

```

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.

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](#)

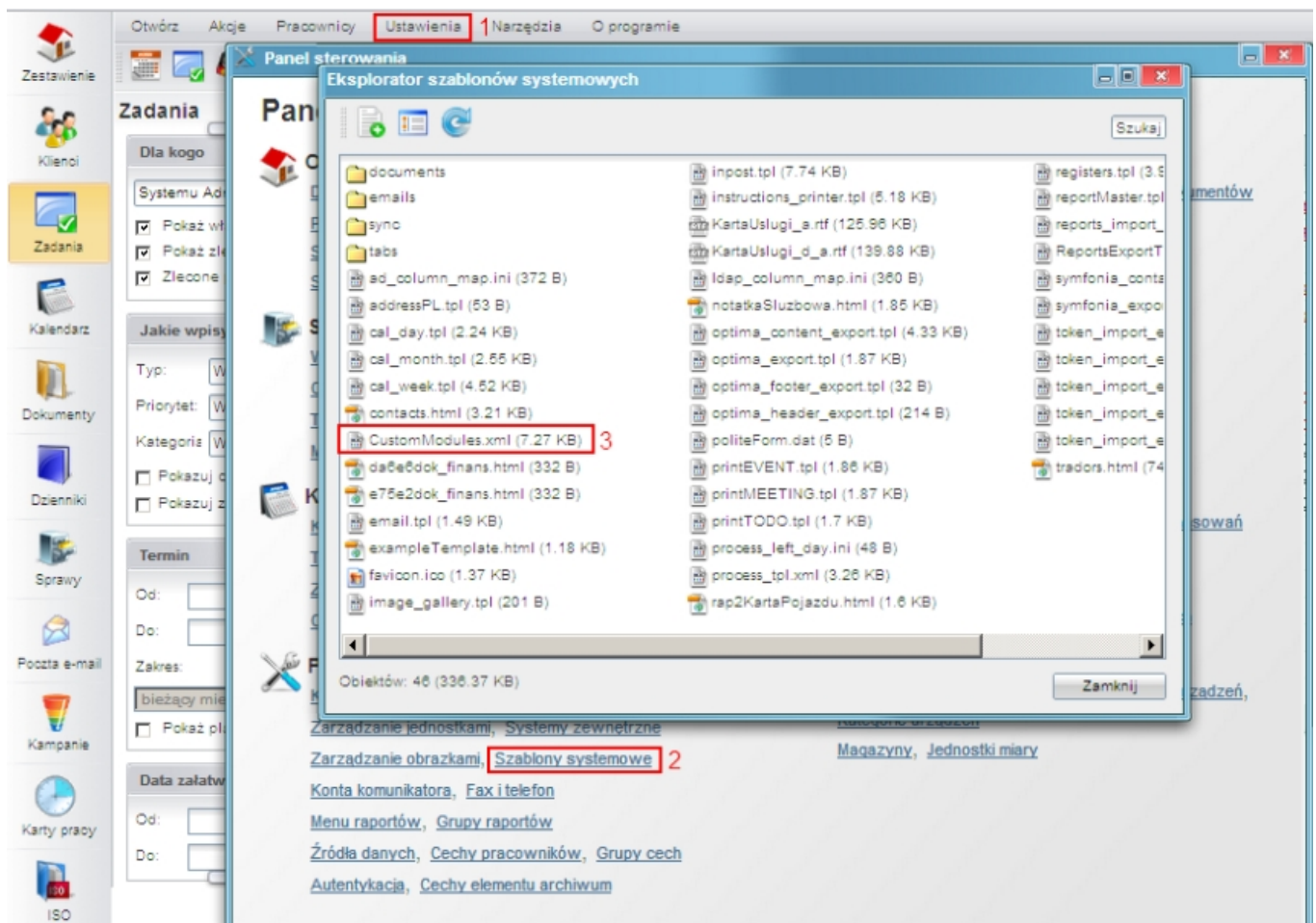
Dodatkowe moduły

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/tp1/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"?>
<modules>
  <module id="cModule_1" name="Delegacje" label="Delegacje"
  icon="processes.gif" right="bswfms.custom_modules.delegations">
    <toolbar>
    </toolbar>
  </module>
</modules>
```

```

<filters>
</filters>
<tabs>
<tab label="Moje delegacje" rep_id="87">
  <buttons>
    <button>
      <id>
        new
      </id>
      <label>
        Nowy
      </label>
      <dscrpt>
        Nowy dokument
      </dscrpt>
      <onclick>
        App.openDialogByCls('DOCUMENT', null,
({afterSubmit: '{AFTER_SUBMIT}', dctpId:23,dctptp:'CustomDocument', mode:'new'}).toJSONString())
      </onclick>
      <icon>
        new.gif
      </icon>
    </button>
    <button>
      <id>
        edit
      </id>
      <label>
        Edycja
      </label>
      <dscrpt>
        Edytuj dokument
      </dscrpt>
      <onclick>
        App.openDialogByCls('DOCUMENT', {KEYVAL},
({afterSubmit: '{AFTER_SUBMIT}', dctpId:23,dctptp:'CustomDocument', mode:'edit'}).toJSONString())
      </onclick>
      <icon>
        edit.gif
      </icon>
    </button>
    <button>
      <id>
        delete
      </id>
      <label>
        Usuń
      </label>
      <dscrpt>
        Usuń
      </dscrpt>
      <onclick>
        App.openDialogByCls('DOCUMENT', {KEYVAL},
({afterSubmit: '{AFTER_SUBMIT}', dctpId:23,dctptp:'CustomDocument', mode:'del'}).toJSONString())
      </onclick>
      <icon>
        delete.gif
      </icon>
    </button>
  </buttons>
</tab>

```

```

    </tabs>
</module>

<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, dctp: 'CustomDocument', mode: 'new'}).toJSONString())
      </onclick>
      <icon>
        new.gif
      </icon>
    </button>
    <button>
      <id>
        edit
      </id>
      <label>
        Edycja
      </label>
      <dscrpt>
        Edytuj wniosek
      </dscrpt>
      <onclick>
        App.openDialogByCls('DOCUMENT', {KEYVAL},
({afterSubmit: '{AFTER_SUBMIT}', dctpId:21, dctp: 'CustomDocument', mode: 'edit'}).toJSONString())
      </onclick>
      <icon>
        edit.gif
      </icon>
    </button>
    <button>
      <id>
        delete
      </id>
      <label>
        Usuł,,
      </label>
      <dscrpt>
        Usuł,,
      </dscrpt>
      <onclick>
        App.openDialogByCls('DOCUMENT', {KEYVAL},
({afterSubmit: '{AFTER_SUBMIT}', dctpId:21, dctp: 'CustomDocument', mode: 'del'}).toJSONString())

```

```

        </onclick>
        <icon>
            delete.gif
        </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
- PROCESS - Okienko edycji sprawy
- RCP - Karta pracy
- MEETING - Spotkanie
- EVENT - Termin
- TODO - Zadanie
- PHONECALL - Rozmowa telefoniczna
- DEVICE - Urządzenie
- PRODUCT - Produkt

Przycisk tworzący sprawę

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

```

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

```

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.gif", "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 doklejony 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 `{KEVAL}`

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

Ograniczanie widoczności modułów

Prawa dostępu do modułów (inaczej widoczność modułów) są rozwiązane w bardziej skomplikowany sposób. Do tworzonego modułu dodatkowego należy dodać do tabeli **right_def** definicje prawa (UWAGA !!! jeżeli tego prawa nie ma w tej tabeli) `bswfms.custom_modules`, a potem szczegółowe prawo do zakładki np. do zakładki *Delegacje* może to być: `bswfms.custom_modules.delegations`. W w/w tabeli należy wypełnić następujące pola

- `prn_id` - (dla `bswfms.custom_modules` będzie to prawdopodobnie 1, a do `bswfms.custom_modules.delegations` wartość pola `def_id` rekordu `bswfms.custom_modules` - Musi być zapewniona struktura drzewiasta!)
- `define` - nazwa prawa np. `bswfms.custom_modules.delegations`
- `group_` - ustawić należy SYSTEM
- `commen` - nazwa prawa wyświetlana w systemie. np.: *Delegations: delegacje*.

Przykładowe inserty przedstawiono poniżej:

```
INSERT INTO right_def(prn_id,define,group_,commen) VALUES ((SELECT def_id FROM right_def WHERE define = 'bswfms' ), 'bswfms
INSERT INTO right_def(prn_id,define,group_,commen) VALUES ((SELECT def_id FROM right_def WHERE define = 'bswfms.custom_mod
```

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.gif" 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 jest 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})

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

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

[Filtry dla raportów](#)

Przejdź do [Menu](#)

Ikona modułu

Ikony w formacie .gif bądź .png o maksymalnym rozmiarze 36px x 36px wrzucamy do katalogu `public_html/framework/img/PageToolBar`.

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>
  <dscprt>Otwiera tabelę HTML</dscprt>
  <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>
```

Przejdź do [Menu](#)