Przewodnik wdrożeniowca > Praca z szablonami Flexy

Menu

1. Wprowadzenie
2. Tworzenie szablonu Flexy i wykorzystanie go w raportach
   2.1 Tworzenie raportu SQL
   2.2 Tworzenie szablonu Flexy dla raportu
   2.3 Instalacja szablonu Flexy w raporcie
   2.4 Instalowanie raportu w odpowiednim module i dokumencie
3. Funkcje PHP dostępne w szablonach
4. Dołączania skryptów js/css do szablonów

1. Wprowadzenie

Flexy to bardzo szybki i dysponujący wielkimi możliwościami silnik szablonów. Może być używany zarówno w raportach, blokach jak i generowaniu plików zewnętrznych np. do połaczenia z Symfonią.

http://pear.php.net/manual/en/package.html.html-template-flexy.php

Przejdź do Menu

2. Tworzenie szablonu Flexy i wykorzystanie go w raportach

2.1 Tworzenie raportu SQL

Tworzenie raportu z wykorzystaniem Flexy należy rozpocząć od zdefiniowania raportu SQL. W module raporty tworzymy nowy raport. W zakładkę *Ogólne* wprowadzamy nazwę raportu, miejsce jego przechowywania. Przechodzimy do zakładki *Definicja*, gdzie wprowadzamy definicję SQL. W zakładce Wybór kolumn możemy wybrać, jakie kolumny będą wykorzystywane w raporcie.

Przykładowo tworzymy raport dla *Notatek służbowych*. W tym celu tworzymy raport w grupie Dokumenty pt. *Notatka Służbowa*. W zakładce definicji raportu wprowadzamy SQL-a:

```
SELECT
dv.dscrpt,
dv.conten,
dv.srctxt,
dv.trgtxt,
p.symbol||' - '||p.dscrpt as symbol,
COALESCE(c.name_2, c.name_1) AS cname_
FROM documents_view dv
LEFT JOIN processes p USING (prc_id) --ON (dv.prc_id = p.prc_id)
LEFT JOIN doc_link_cont dlc ON (dv.doc_id = dlc.doc_id AND dlc.role__ = 'RELATED')
LEFT JOIN contacts c ON (dlc.contid = c.contid)
WHERE dv.doc_id = {DOC_ID}
```

Dodatkową funkcjonalnością, którą może być wykorzystana w raportach Flexy są kwerendy. jest to ostatnia zakładka okienka definicji raportu. Parametry takiego raportu muszą być identyczne z parametrami głównego raportu. W odniesieniu do naszego przykładu mogą być to np. kometarze:

```
SELECT
dc.commnt,
u.firnam||' '|| u.lasnam as addtxt,
to_char(dc.adddat, 'YYYY-MM-DD') AS adddat
FROM document_comments AS dc
LEFT JOIN users u ON (dc.adduid = u.usr_id)
WHERE dc.doc_id = {DOC_ID} ORDER BY dc.adddat ASC
```

Dodatkowy kwerend można zdefiniować dowolną ilość.

W tym momencie posiadamy działający już raport

2.2 Tworzenie szablonu Flexy dla raportu

Skoro posiadamy już raport SQL teraz należy utworzyć szablon. Czym jest szablon? Otóż jest to plik z rozszerzeniem **.html**, którego przykładowa zawartość wygląda następująco:

```
<style>
  TABLE .ReportTable TD {
       font-size: 12px;
  }

  TD.printLabel {
      font-size: 12px;
      vertical-align: top;
      font-style: italic;
      border-bottom: 1px solid grey;
  }

  TD.printData {
      font-size: 12px;
      vertical-align: top;
      border-bottom: 1px solid grey;
  }

  .printHeader {
      font-size: 16px;
      font-weight: bold;
      margin-bottom: 10px;
  }

  .parag {
       text-indent: 20px;
   }
</style>
<div style="margin-left:20px;">

   <p class="printHeader" style="margin-top: 10px;">Temat</p>
   {resultQueries[0][0][dscrpt]}

   <p class="printHeader" style="margin-top: 10px;">Treść</p>
   {resultQueries[0][0][conten]:h}

   <p class="printHeader" style="margin-top: 10px;">Załączniki</p>
   <ul style="list-style-type:none;">
       {foreach:resultQueries[2],k,v}
           <li style="font-size:12px;"> - {v[filenm]}<li>
       {end:}
   </ul>

   <p class="printHeader" style="margin-top: 10px;">Komentarze</p>
   <ul style="list-style-type:none;">
       {foreach:resultQueries[1],k,v}
           <li><i>{v[addtxt]} {v[adddat]}</i><BR>{v[commnt]:h}<li>
       {end:}
   </ul>

   <table class="RaportTable RaportTable2" style="margin-top:20px; background-color: white; width: 100%" border=0 cellpadd
       <tr>
           <td class="printLabel">Dokument od:</td>
           <td class="printData">{resultQueries[0][0][srctxt]}</td>
           <td class="printLabel">Dokument do:</td>
           <td class="printData">{resultQueries[0][0][trgtxt]}</td>
```

```
      </tr>
      <tr>
          <td class="printLabel">Klient:</td>
          <td class="printData">{resultQueries[0][0][cname_]}</td>
          <td class="printLabel">Numer sprawy:</td>
          <td class="printData">{resultQueries[0][0][symbol]}</td>
      </tr>
   </table>

</div>
```

Należy zwrócić uwagę, że nie ma tutaj definicji typowych dla pliku html, jak *<html>, <head>, <body>, <!DOCTYPE...>*. Jest niejako tylko czysta treść, na którą składają się:

- style CSS pomiędzy znacznikami <style></style>
- elementy struktury dokumentu, jak: , <td></td>, , itp.
- znaczniki Flexy.

Znaczniki Flexy

W tym miejscu zatrzymamy się, aby omówić pokrótce konstrukcję znaczników FLexy. Najważniejszym znacznikiem wykorzystywanym w szablonach jest {resultQueries}. Przykładowe zastosowanie (z powyższego przykładu):

```
 {resultQueries[0][0][dscrpt]}
```

Znacznik ten wybiera dane korzystając z definicji SQL raportu, dla którego jest zdefiniowany. I tak `resultQueries` - pobierz dane ze skryptu SQL, `[0]` - pierwszy skrypt z definicji raportu (główna definicja raportu - z zakładki Definicja SQL). Dla kwerend będzie to odpowiednio wg kolejności na liście `[1], [2]` . `[0]` - Pobiera pierwszy wiersz (rekord) raportu. `[dscrpt]` - wybiera dane z kolumny dscrpt pobranego rekordu.

Powyższą konstrukcję stosuje się, jeżeli mamy pewność, iż wynikiem będzie jeden wiersz rekordu. Dla większej ilości rekordów stosuje się pętlę **foreach** o konstrukcji:

```
<ul style="list-style-type:none;">
{foreach:resultQueries[1],k,v}
       <li><i>{v[addtxt]} {v[adddat]}</i><BR>{v[commnt]:h}</li>
{end:}
</ul>
```

Gdzie: `{foreach:resultQueries[1],k,v}` oznaczają początek pętli z kwerendy (zakładka Kwerendy), która zwraca parę wartości: klucz (k) oraz wartość (v). Należy pamiętać o zamknięciu pętli konstrukcją `{end:}` . Odwołanie do wartości każdego raportu w pętli to: `{v[adddat]}` , gdzie oznacza to pobranie wartości (v) kolumny [adddat]. Jeżeli źródłem danych jest pole tekstowe, to należy zastosować konstrukcję `{v[commnt]:h}` , gdzie istotny jest modyfikator **:h**, który wyłącza przetwarzanie funkcji *htmlspecialchars*. Uniemożliwia to wykonanie np. szkodliwych skryptów JavaScript.

Domyślne znaczniki Flexy

Od wersji 4.2-beta82 dodane zostały domyślne znaczniki. Posługujemy się nimi podobnie jak innymi znacznikami czyli {znacznik}.

Znacznik	Opis
{LOGGED_USR_ID}	ID zalogowanego pracownika

Obecnie mamy dwa elementy takiego systemu. Jak to teraz połączyć, aby ze sobą współpracowały?

2.3 Instalacja szablonu Flexy w raporcie

Posiadając w pliku szablon html oraz zdefiniowany raport w systemie należy razem te elementy ze sobą scalić, tak aby ze sobą współpracowały. Pierwszym krokiem w tym procesie jest import szablonu do systemu eDokumenty. Przechodzimy do *Panelu Sterowania* > *Szablony Systemowe*. Importujemy plik szablonu. Następnie przechodzimy do definicji raportu, gdzie w zakładce Ogólne w polu domyślny szablon wpisujemy nazwę szablonu flexy (z rozszerzeniem np. notatkaSluzbowa.html). Po zapisie mamy scalony szablon z raportem.

Ale to jeszcze nie koniec. Teraz należy podpiąć odpowiednio raport pod dokument.

2.4 Instalowanie raportu w odpowiednim module i dokumencie

Posiadany raport należy zainstalować w odpowiednim module np. Module Dokumenty oraz odpowiednim dokumencie np. dokumencie typu Notatka służbowa. Wtedy uruchomienie akcji Drukuj > Notakta Służbowa na dokumencie uruchomi raport Flexy.

Przejdź do Menu

3. Funkcje PHP dostępne w szablonach:

Parametry w funkcjach muszą być oddzielone przecinkiem (bez spacji).

1. str_replace

```
Przykład użycia:

{str_replace(#-#,##,documents.dscrpt):h}

Powyższe wywołanie usunie wszystkie wystąpienia znaku "-" z tekstu znajdującego się pod zmienną "documents.dscrpt"
```

1. trim

```
{trim(documents.dscrpt):h}

Powyższe wywołanie usunie wszystkie białe znaki z początku i końca tekstu znajdującego się pod zmienną "documents.dscrpt
```

1. preg_replace
1. substr

```
{substr(documents.dscrpt,2,0):h}

Powyższe wywołanie zwróci tekst ze zmiennej "documents.dscrpt" począwszy od drugiego znaku.
```

1. strpos / stripos

```
Opis dokumentu {if:strpos(documents.dscrpt,#123#)}ZAWIERA{else:}NIE ZAWIERA{end:} ciąg: "123"
```

1. funkcje porównujące (eq, gt, gte, lt, lte)

```
!eq(1,2) negacja równości !(1==2)
eq(1,2) jednoznaczne z 1 == 2
gt(1,2) jednoznaczne z 1 > 2
gte(1,2) jednoznaczne z 1 >= 2
lt(1,2) jednoznaczne z 1 < 2
lte(1,2) jednoznaczne z 1 <= 2
```

przykład:

```
{if:gt(vatnote.netto_,0)}
kwota netto jest większa od zera
{end:}
```

1. funkcja wykonująca zapytanie SQL

```
{sql_query(#SELECT street FROM addresses WHERE addrid = (SELECT mainad FROM contacts WHERE contid = {%1})#,event[contid

{sql_query(#SELECT ext_id FROM bs_connect_contacts_coherences where int_id = {%1}#,contact.contid)}
```

Ostatni parametr do którego nie ma odwołania w zapytaniu będzie brany jako nazwa zmiennej. I w takim przypadku, funkcja nie zwróci bezpośrednio wyniku tylko zapisze go do zmiennej o podanej nazwie. Uwaga! w szablonach w których dane pobierane są z obiektów w notacji {obiekt.pole} - umieszczając nazwę w poleceniu należy pominąć nawiasy francuskie.

```
{sql_query(#select 1#,#aqq#)}
{if:aqq}
SUKCES
{end:}
```

W przypadku zapisania wyniku do zmienne pobranie go odbywa się poprzez odwołanie się do nazwy tej zmiennej jak poniżej:

```
{aqq[0]}
```

1. funkcja wykonująca zdefiniowany raport.
   clsnam - dowolny token obiektu obsługiwany przez raporty (np. SELECT * FROM documents WHERE doc_id = {DOC_ID}).

Możliwe jest również użycie zmiennej "params" (uwaga! użyta zmienna w wywołaniu funkcji nie ma znaczników). W niej dostępne są wszystkie filtry raportu.

```
{output_report(12,#DOC_ID#,documents.doc_id):h}
{output_report(12,#PRC_ID#,7):h}
{output_report(81,##,##,params):h}
```

wszystkie dostępne funkcje:

```
    round($val, $precision = NULL);

   in_array($val, $array);

   array_key_exists($key, $array);

   count($array);

   implode($delimiter, $array_var, $result_var);

   explode($delimiter, $string, $result_var);

   wordwrap( string $str [, int $width = 75 [, string $break = "\n" [, bool $cut = false ]]] )

   ext_wordwrap( string $str [, int $width = 75 [, string $break = "\n" [, bool $cut = false [, int $parts = NULL [, strin

   strpos($haystack, $needle);

   stripos($haystack, $needle);

   substr($string, $start, $length);

   str_replace($from, $to, $source);

   trim($string);

   preg_quote($str);

   preg_replace($from, $to, $source);

   date($format, $time = NULL);

   number_format( float $number , int $decimals , string $dec_point , string $thousands_sep )

   str_pad( string $input , int $pad_length [, string $pad_string= " " [, int $pad_type= STR_PAD_RIGHT ]] )
```

```
json_decode($target, $json, $param = TRUE)

sql_query($query[,$var1,$var2,...])

output_report($rep_id, $clsnam, $keyval[, $params])

getFeatureValueAsString($featid, $tblnam, $tbl_id)

moneyToText($n, $intUnits = 'zł', $floatUnits = 'gr')

registerDialog($path) - przykład <div onclick="App.openDialogEx('{registerDialog(#./modules/Dictionaries/Projects/forms

formateMailContent($doc_id):h - funkcja formatująca treść zarchiwizowanego maila podobnie jak wydruk domyślny dla maila

printMainTableHtml() - Drukuje tabelę danego raportu

newObject($objectName, $filepath, $className, ...)
callFunc($objectName, $method, ...)

attachmentsList($clsnam, $keyval) - lista załączników jako komponent

array_merge(params,#{"style":{"chart_height":"200px"}}#) - umożliwia złączenie jednej tablicy z inną. Drugi argument na

getValueFromArray($array, $key) - umożliwia wyciągniecie wartości spod klucza, który jest przechowywany w innej zmienne

extractValueFromArray($array, $key, $return_var) - wyciąga wartość tak jak getValueFromArray tylko że wynik zapisuje do

formatColumn($phpfun, $value, $params = null) - formatuje nam wartość tak jak kolumny w raportach
```

Przykłady

Operacje arytmetyczne na zmiennych:

```
{setval(#value#,1)}       - inicjalizajca zmiennej value (value=1)
{setval(#value#,3,#+#)}   - dodawanie (value=4)
{setval(#value#,2,#-#)}   - odejmowanie (value=2)
{setval(#value#,5,#*#)}   - mnożenie (value=10)
{setval(#value#,2,#/#)}   - dzielenie (value=5)

<h2>{value}</h2>          - <h2>5</h2>

{setval(#value#,abc,#.#)} - dopisanie ciągu (value=5abc)


{setval(#parametry[filter_string]#,#doc_id = 10#,#[]#)}   - tworzy/dopisuje wartość do tablicy. parametry Array(filter_str
{setval(#params[style[height]]#,#200px#,#[]#)}            - możliwość rekurencyjnego przypisywania wartości do tablicy.
```

Przekazany parametr w szablonie do URL + obsługa sprawdzenia jaka strona jest aktywna.

```
-- wygenerowany link
<a href="./engine/instal/CustomModule/cModule_1?rep_id=25&{{_(STDKID)_}}={v[stdkid]}">{v[stdknm]}</a>
-- sprawdzenie która sekcja jest aktywna
{if:strpos(resultQueries[2][0][result],#NULL#)}
       {output_report(30,##,##):h}
{end:}
--
zapytanie do bazy (resultQueries[2])
SELECT '{STDKID}' AS result
```

Przykład newObject i callFunc

- newObject - funkcja umożliwiająca utworzenie obiektu w PHP gdzie parametry to
  - objectName - nazwa naszego obiektu pod którą będzie dostępny w obrębie danego szablonu flexy
  - filepath - ścieżka do pliku (klasy) w którym istnieje definicja naszego obiektu. Ważne jest to, że plik musi znajdować się w lokalizacji ./scripts. Ścieżka może wskazywać na dowolną lokalizację w apps/edokumenty/scripts czyli może to być np. apps/edokumenty/scripts/moje_flexy/inne/Klasa.php
  - className - nazwa klasy, której definicja jest w filepath. Nazwa klasa nie jest odczytywana z nazwy pliku także należy ją jawnie podać.
  - ... (3 kropki) - dowolne argumenty, które zostaną przekazane jako tablica do obiektu naszej klasy

Przykład wywołania:

```
{newObject(#mojObiekt#,#MojaKlasa.inc#,#Klasa#,#1#,#2#,#3#)}
```

- callFunc - funkcja umożliwiająca wywołanie metody na wcześniej zadeklarowanym obiekcie
  - objectName - nazwa wcześniej zadeklarowanego obiektu (#mojObiekt#)
  - method - nazwa metody dostępnej w klasie #Klasa#
  - ... (3 kropki) - dowolne argumenty, które zostaną przekazane do metody method

Oznacza to, że zostanie utworzony obiekt $mojObiekt = new Klasa([1,2,3]);

Przykład wywołania metody dodajLiczby (musi być dostępna w klasie #Klasa#) dla wcześniej zadeklarowanego obiektu mojObiekt:

```
{callFunc(#mojObiekt#,#dodajLiczby#,#4#,#5#,#6#):h}
```

Co da nam wywołanie metody dodajLiczby na obiekcie mojObiekt klasy Klasa. Przykład klasy w załączniku.

Przykład pobrania wartości KEYVAL z pierwszego znaczonego rekordu

```
<button onclick="var keyval = App.{LIST_NAME}.getSelected()[0]; if (keyval === undefined) keyval = false; alert(keyval);re
{printMainTableHtml()}
```

===Przykłady do formatowania wartości za pomocą formatColumn=== Funkcja ta jest wykorzystywana do formatowania kolumn w raportach.

```
formatColumn(#processState#,resultQueries[1][0][tpstid]) -- formatuje na status
```

- dateTime > Formatuj datę i godzinę
- date > Formatuj datę
- hourWithMinutes > Formatuj godzinę z minutami
- money > Formatuj kwotę
- moneyInt > Formatuj kwotę do liczby całkowitej
- percent > Formatuj procent
- number > Formatuj liczbę
- time > Formatuj czas
- formatMinutes > Formatuj minuty
- externalContid > Formatuj link do kontaktu
- externalopenDialogByCls > Formatuj link do ... w formacie: CLSNAM|KEYVAL|Opis do wyświetlenia|eic-ikona np. DOCUMENT|123|Otwórz dokument|eic-document
- processState > Formatuj status
- checkmark > Formatuj typ logiczny

4. Dołączania skryptów js/css do szablonów

W szablonach używanych w customowych modułach istnieje czasami konieczność dołączenia zew. biblioteki javascript lub stylu css. Aby tego dokonać i aby działało poprawnie wymagane jest po pierwsze umieszczenie tych plików na serwerze (zalecane) a następnie wywołania w szablonie funkcji javascript:

```
<script>
   loadjscssfile('../../framework/lib/contrib/vis/vis.js', 'js');
   loadjscssfile('../../framework/lib/contrib/vis/vis.css', 'css');
</script>
```

Lista załączników jako komponent http://support.edokumenty.eu/trac/wiki/Documentation/Index/FlexyFilesList

Przejdź do Menu