



Własna szata graficzna

SYSTEM SZABLONÓW **TYPO3**

Dla kogo jest ta prezentacja

- ⦿ Niniejsza prezentacja omawia podstawy wdrażania szablonów w TYP03.
- ⦿ Prezentacja przeznaczona jest dla osób rozpoczynających naukę TYP03, **nie** omawia ona zaawansowanych technik (możliwe jest rozwinięcie użytych przykładów na podstawie oryginalnej dokumentacji [TSref](#)). Jest to niejako plan minimum wymagany do zobaczenia we własnym serwisie czegoś więcej niż napis **HELLO WORLD!**
- ⦿ Zakładam, że słuchacz posiada **podstawowe** umiejętności tworzenia struktury serwisu oraz dodawania elementów treści, dzięki czemu będzie w stanie zweryfikować poprawność swojego kodu szablonu TS.

- ⦿ W przeciwieństwie do wielu prostych i/lub szeroko stosowanych systemów CMS, TYP03 **nie słynie** z dostępności na rynku gotowych szablonów graficznych.
- ⦿ Wynika to przede wszystkim z faktu, że TYP03 nie posiada predefiniowanej struktury początkowej (co zapisujemy in plus) ani narzuconych rozwiązań dla **pozornie** typowych zadań.
- ⦿ Ostatecznie **rzekomy** brak szablonów dla TYP03 to **niska** cena w odniesieniu do skalowalności systemu.

- ◉ Wiele znanych serwisów i firm sprzedających lub udostępniających za darmo gotowe szablony graficzne oferuje produkty predefiniowane dla różnych systemów CMS, blogów, etc., ale próżno szukać tam zakładki **TYP03**.
 - Czy to znaczy, że jestem skazany na drogie studia graficzne albo niewielki wybór firm specjalizujących się w szablonach dla mojego systemu?
- ◉ **TYP03 Profiled**
 - Każdy szablon stworzony zgodnie ze sztuką webmasterską* jesteś w stanie przypiąć w krótkim czasie do swojego wdrożenia. Dotyczy to zarówno szablonów *stricte* HTML'owych, jak również odmian wyłącznie FLASH'owych (także sterowanych via XML) oraz oczywiście **wszelkich możliwych kombinacji** tych i **innych** technologii web.

- Co istotne dostosowanie każdego kolejnego szablonu do danego serwisu staje się **jeszcze łatwiejsze**, gdyż najczęściej oznacza jedynie modyfikację pliku HTML w celu dodania znaczników **marks** oraz **subparts** i ewentualnie drobne zmiany w nazwach klas (oczywiście przy założeniu, że oba szablony zostały skonstruowane za pomocą zbieżnych technik).
- Najważniejsze jest jednak, że każdy nawet początkujący integrator systemu zauważy spore zbieżności pomiędzy różnymi szablonami, co pozwoli mu na utworzenie własnego, modelowego szablonu TS, którego dostosowanie nawet w obrębie różnych wdrożeń TYP03 ograniczy się do drobnych modyfikacji i usprawnień.



Własne szablony w TYP03

SZABLON HTML

I ###ZNACZNIKI###

Czym jest szablon?

- W TYPO3 pojęcie „szablon” odnosi się do kilku typów obiektów, są to między innymi (dla nas najbardziej interesujące):
 - **Szablon HTML**, który jest plikiem HTML, **zwykłą statyczną stroną** zawierającą w swoim kodzie **znaczniki**, które wskazują systemowi w którym miejscu ma nastąpić **dopięcie** dynamicznej treści oraz w jaki sposób ma to być wykonane.
 - **Szablon TS** jest **rekordem** systemu w bazie danych w którym za pomocą języka **TypoScript** konfigurujesz zachowanie systemu a także określasz sposób wykorzystania szablonu HTML i zawartych w nim znaczników.

Istnieje także pojęcie szablon serwisu, w którego skład wchodzi struktura stron, szablon graficzny, preinstalowane rozszerzenia oraz korespondujący TyPoScript, jednak w tej prezentacji nie będziemy tego tematu poruszać.

Znaczniki w plikach HTML

- Znaczniki w pliku szablonu HTML umieszczasz w miejscach gdzie ma zostać wstawiona (marks) lub zastąpiona oryginalna zawartość (subparts)
- Nazwa znacznika to dowolny łańcuch otoczony potrójnym hashem:

```
###MOJ_ZNACZNIK###
```

- Chociaż nie ma znaczenia jakiej użyjesz wielkości liter istotne jest abyś odwołując się do znacznika z poziomu szablonu TS użył dokładnie tego samego zapisu.

HTML

```
###MOJ_ZNACZNIK###  
###mojZnacznik###  
###Moj-ZnaCznik###
```

TypeScript

```
marks.MOJ_ZNACZNIK  
marks.mojZnacznik  
marks.Moj-ZnaCznik
```

Dwa typy znaczników

- ⦿ Istnieją dwa typy znaczników: **marks** i **subparts** i w obu przypadkach stosuje się identyczny sposób ich tworzenia czyli **###ZNACZNIK_123###**.
- ⦿ **marks** stosuje się pojedynczo **###KIEDY###** **bez** komentarzy HTML.

TypoScript:

```
temp.mojSzablon.marks.KIEDY = TEXT
temp.mojSzablon.marks.KIEDY.value = (najczęściej)
```

```
<!-- ###OPIS_SUBPARTS### Początek mojego subparta-->
```

subparts stosowane parami obejmują blok do zastąpienia (początek/koniec)
zawsze w komentarzu HTML.

```
<!-- ###OPIS_SUBPARTS### Koniec -->
```

TypoScript:

```
temp.mojSzablon.subparts.OPIS_SUBPARTS = TEXT
temp.mojSzablon.subparts.OPIS_SUBPARTS.value = Pamiętaj: Subparts BEZ komentarzy!
```

Znaczniki typu **marks**

- marks używasz pojedynczo w miejscu gdzie ma zostać **wstawiony** wygenerowany kod. Żadna treść pliku HTML **nie** zostanie usunięta ani zastąpiona:

```
<div id="left_menu">  
    <h3>Ten Tytuł NIE zostanie usunięty... </h3>  
    ###LEFT_MENU_CONTENT###  
</div>
```

- marks **NIE** powinno używać się w komentarzu HTML, gdyż system **nie usunie** tego komentarza, czyniąc wyrenderowaną treść niewidoczną na stronie:

```
<!-- ###AUTOR_ARTYKULU### -->
```

Pojawi się w kodzie HTML jako:

```
<!-- Marcus Biesioroff @ TYPO3 Społeczność Polska -->
```

Znaczniki typu **subparts**

- ⦿ subparts służą do **obejmowania** fragmentów kodu HTML, który ma być **zastąpiony**.
- ⦿ subparts stosuje się parami, a same znaczniki zawsze **obejmuje się** komentarzem HTML wg schematu:

```
<!-- ###nazwa_znacznika### komentarz -->  
    {fragment kodu HTML, który zostanie zastąpiony}  
<!-- ###nazwa_znacznika### komentarz -->
```

- ⦿ Nie ma znaczenia jakiego komentarza użyjesz obok nazwy subpartu, ma on ułatwić Tobie pracę podczas edycji szablonu HTML i finalnie zawsze jest usuwany (nie pojawia się nigdy w wyrenderowanej stronie). Domyślnie opisuje się w komentarzu **początek** i **koniec** sekcji subpart.

Który typ wybrać

- ⦿ Nie ma znaczenia, czy będziesz używał znaczników typu **marks** w pustych warstwach, czy też użyjesz **subparts** do podmiany przykładowego kodu.
- ⦿ Domyślnie za pomocą subparts łatwiej edytować szablon jeśli masz przykładową treść widoczną podczas pracy w edytorze HTML (np. wielopoziomowe menu składające się z hierarchii list wypunktowanych):

```
<div id=„menu-lewe“>
  <!-- ###MenuLewe### Początek subpartu -->
  <ul class=„poziom_1“>
    <li><a href=„#“>Pozycja 1.1</a></li>
    <li><a href=„#“>Pozycja 1.2</a></li>
    <li><ul class=„poziom_2„>
      <li><a href=„#“>Pozycja 2.1</a></li>
    </ul></li>
  </ul>
  <!-- ###MenuLewe### Koniec tego fragmentu-->
</div>
```

Dla porównania, konstrukcja z użyciem znacznika typu **marks**

```
<div id=„menu-lewe“>
  ###MenuLewe###
</div>
```

Który typ wybrać

- ⦿ Z drugiej strony nie ma sensu używania **subparts**, jeśli zamierzasz jedynie wstawić tekst w określonym miejscu:

```
<div id="footer">  
    ###TEKST_STOPKI###  
</div>
```

- ⦿ Ostatecznie po zakończeniu prac nad wyglądem szablonu zawsze możesz pozmieniać sekcje **subparts** na **marks** i **odwrotnie**

WYMAGANY subpart

- TYPO3 automatycznie tworzy szkielet strony HTML z prawidłową deklaracją typu dokumentu oraz sekcjami `<html>`, `<head>` i pustą `<body>`. Z własnego szablonu chcemy więc wykorzystać TYLKO zawartość **naszej** sekcji `<body>`.
- Aby to osiągnąć obejmij wewnątrz sekcji body własnym (głównym) subpartsem a potem wykorzystaj jego nazwę w zmiennej `workOnSubpart` obiektu typu `TEMPLATE` w swoim szablonie TS.

```
<html>
<head>
  <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
<title>Plik statyczny</title>
</head>
<body>
<!-- ###MOJ_GLOWNY_SUBPART### Początek -->
  {...Pozostały kod HTML szablonu.
  subpart główny go nie zastępuje a jedynie określa jego granice!}
<!-- ###MOJ_GLOWNY_SUBPART### Koniec -->
</body>
</html>
```

`temp.mojSzablon.workOnSubpart = MOJ_GLOWNY_SUBPART`





Własne szablony w TYPO3

SZABLON TYPOSCRIPT (TS)

Co to jest szablon TS?

- ◎ Szablon TS, jest rekordem systemu w bazie danych w którym konfigurujesz podstawowe właściwości danej części lub całości wdrożenia. (formularz ogólny właściwości szablonu TS)
- ◎ Za pomocą kodu **TypoScript** określasz sposób renderowania strony frontowej a także sposób wykorzystania szablonu HTML oraz zawartych w nim znaczników.
- ◎ Rekord szablonu TS może również posłużyć do spięcia innych szablonów TS (np. dedykowanych konkretnym rozszerzeniom) za pomocą opcji dołączania szablonów.

- ⦿ Skoro omówiliśmy szablon HTML czas wskazać jak go użyć w TYP03.
- ⦿ Użyj narzędzia **Web > Szablon** i wybierz stronę na której chcesz utworzyć nowy szablon (zakładam, że serwis lub przynajmniej jego część nie zawiera jeszcze szablonu a wskazana strona jest początkiem danej części)
- ⦿ **NO TEMPLATE**, no problem. Kliknij **Create template for new site**.
(Nie wybieraj żadnych gotowców, zależy nam na pustym szablonie)
- ⦿ Po pojawieniu się tabeli właściwości szablonu kliknij link:
Click here to edit whole template record

Szablon TS

Właściwości

- ⦿ Na zakładce **Ogólne** wpisz nazwę szablonu oraz **tytuł serwisu** (zostanie użyty jako początek tytułu strony w przeglądarce)
- ⦿ Na zakładce **Opcje** włącz czyszczenie **Constants** i **Setup**, żeby pozbyć się naleciałości z poprzednich części. Włącz też **RootLevel** to oznacza, że wszystko jest niejako wyzerowane od tego miejsca w dół drzewa.
- ⦿ Na zakładce **Dołączaj** dołącz statyczny TS z rozszerzenia: **CSS Styled Content** – dostarcza wielu predefiniowanych **cObject'ów** do renderowania treści na stronie frontowej.

Szablon TS

SETUP, fragment **config**

- ◉ Wróć na zakładkę **Ogólne** i wykorzystaj pole **Setup** do konfiguracji systemu. Poniżej dość typowy kod, możesz go po prostu użyć i/lub zmodyfikować zgodnie z dokumentacją **TSref**:

```
config{
    simulateStaticDocuments = 0
    baseURL = http://twoja-domena.loc/
    admPanel = 0
    metaCharset = utf-8
    doctype = xhtml_trans
    xmlprologue = none
    disablePrefixComment = 1
    htmlTag_langKey = pl
    language = pl
    locale_all = pl_PL
}
```

- ⦿ Dzięki dołączeniu **CSS Styled Content** bez dodatkowych zabiegów mamy utworzone cObject'y reprezentujące 4 **domyślne kolumny**:

Nazwa:

`styles.content.get`

`styles.content.getLeft`

`styles.content.getRight`

`styles.content.getBorder`

Zawartość:

kolumna normalna

kolumna lewa

kolumna prawa

kolumna boczna

- Wykorzystaj dokumentację **TSref** do rozbudowania tego przykładu menu:

```
temp.mojeMenuLewe = HMENU
temp.mojeMenuLewe {
  entryLevel = 1
  1 = TMENU
  1 {
    wrap = <ul class="lvl-1">|</ul>
    NO{
      wrapItemAndSub = <li>|</li>
    }
    ACT < .NO
    ACT.ATagParams = class="act"
    ACT = 1
  }

  2 < .1
  2.wrap = <ul class="lvl-2">|</ul>

  3 < .1
  3.wrap = <ul class="lvl-3">|</ul>
}
```

Zwróć uwagę na to, że poziomy menu dziedziczą z poziomu 1 a potem zmieniają tylko przypisaną do **** klasę CSS.

Zamiast tego możesz w pełni opisać każdy poziom (tak jak poziom 1) wymuszając również inne zmiany

Skorzystaj z **TSref** żeby zapoznać się z pełnymi możliwościami konfiguracji menu.

Szablon TS

cObject – **TEMPLATE**

- Skoro mamy już stworzone cObjecty dla potrzebnych elementów zawartości czas stworzyć cObject typu TEMPLATE, w którym zaimportujemy nasz plik HTML oraz przypiszemy dynamiczną zawartość do jego znaczników (**marks** i **subparts**)

```
temp.mainTemplate = TEMPLATE
temp.mainTemplate{
    template = FILE
    template.file = fileadmin/katalog/plik.html
    workOnSubpart = MOJ_GLOWNY_SUBPART
```

```
marks {
    KOLUMNA_NORMALNA < styles.content.get
    KOLUMNA_LEWA < styles.content.getLeft
    KOLUMNA_PRAWA < styles.content.getRight
}
```

```
subparts {
    MENU_LEWE < temp.mojeMenuLewe
}
}
```

Szablon TS

cObject – **PAGE**

- Ostatnią rzeczą do umieszczenia w polu Setup jest deklaracja cObjectu typu **PAGE**, zaimportowanie w nim naszego pliku **CSS** (**page.includeCSS**) i przypisanie wcześniej utworzonego obiektu **TEMPLATE**

```
page = PAGE
page {
    //teoretycznie poniższa wartość nie jest wymagana, ale jej brak przypomni o sobie, gdy dodasz inne cObjecty typu PAGE
    typeNum = 0

    includeCSS{
        file1 = fileadmin/katalog/styl.css
        file1.title = Styl główny
        file1.media = screen
    }

    10 < temp.mainTemplate
}
```



Własne szablony w TYPO3

PODSUMOWANIE

Schemat blokowy procedury

1. Utwórz statyczny szablon HTML (HTML+CSS)
 1. Umieść w nim swoje znaczniki **marks** / **subparts**
2. Utwórz szablon TS
 1. Ustaw właściwości szablonu TS (formularz właściwości)
 1. Zakładka **Ogólne**: Nazwa, tytuł
 2. Zakładka **Opcje**: zaznacz czyszczenie **Setup** i **Constants** oraz **RootLevel**
 3. Zakładka **Dołączaj**: Dołącz statyczny z rozszerzenia **CSS Styled Content**
3. Stwórz kod TypeScript (w polu Setup)
 1. Dla sekcji **config** (skopiuj i popraw gotowca)
 2. Dla obiektu **TEMPLATE**
 1. Określ typ: **template = FILE**
 2. Określ lokalizację pliku HTML
 3. Określ nazwę głównego znacznika dla całego `<body>`
 4. Przypisz odpowiednią zawartość do znaczników `marks` i `subparts`
 3. Dla obiektu **PAGE**
 1. `typeNum = 0`
 2. `includeCSS { }`
 3. Przypisz obiekt **template** do obiektu **page** (np. `page.10`)

- ⦿ Jeśli po (teoretycznie) poprawnym napisaniu kodu TypoScript na stronie głównej nic się nie pojawia (jest cała pusta) może to wskazywać, na:
 - Brak obiektu **TEMPLATE**
 - Błąd w ścieżce do pliku HTML w obiekcie **TEMPLATE**
 - Brak przypisania obiektu **TEMPLATE** do obiektu **PAGE**
- ⦿ Możesz się również natknąć na komunikat błędu:
Error! The page is not configured! [type= 0][[]
 - To oznacza, że nie utworzyłeś obiektu PAGE o numerze **typeNum = 0**. Jest to typ domyślny i wymagany do wyświetlania zwykłej strony (bez parametru w linku **&type=X**)



Marek Krawczyk

TYPO3 Społeczność Polska

DZIĘKUJĘ ZA UWAGĘ