



# Gekosale

DOKUMENTACJA DLA WEBDEVELOPERÓW

dotyczy wersji Gekosale 1.0.2

# SPIS TREŚCI

---

1 . Informacje podstawowe.....	3
1.2 Wstęp.....	3
1.3 Struktura folderów.....	3
1.4 Sposób ładowania plików CSS w Gekosale .....	3
1.5 Sposób ładowania plików JS w Gekosale.....	5
1.6 Sposób ładowania szablonów globalnych w Gekosale.....	5
1.7 Sposób ładowania szablonów modułów w Gekosale.....	5

# 1 INFORMACJE PODSTAWOWE

---

## 1.2 Wstęp

Niniejszy dokument jest przeznaczony w szczególności dla osób zainteresowanych dostosowaniem wyglądu strony w sposób niestandardowy lub wdrożeniem indywidualnych projektów graficznych. Standardowe modyfikacje wyglądu sklepu można realizować poprzez wbudowany w moduł CMS system szablonów. Dokładny opis działania tego systemu pojawi się wraz z pełną dokumentacją użytkownika.

Aby w pełni zrozumieć zagadnienia opisywane w tym dokumencie wskazana jest przynajmniej dobra znajomość standardów XHTML, CSS 2.0, sposobu obsługi panelu administracyjnego Gekosale. Pożądana jest znajomość systemu szablonów Smarty.

## 1.3 Struktura folderów

Wszystkie elementy dotyczące wyglądu strony takie jak pliki CSS, JS, pliki graficzne znajdziesz w folderze **design**. W niniejszym dokumencie ograniczymy się tylko do elementów przydatnych przy wdrażaniu własnych projektów.

**\_css\_frontend** (folder zawierający wszystkie pliki CSS wykorzystywane przez Gekosale)

**\_images\_frontend** (folder zawierający wszystkie pliki graficzne)

**\_js\_frontend** (folder zawierający wszystkie pliki JS)

**\_tpl** (folder zawierający globalnie wykorzystywane pliki szablonów np. header, footer)

**frontend** (folder zawierający pliki szablonów dla każdego z modułów)

## 1.4 Sposób ładowania plików CSS w Gekosale

W przypadku każdego z folderów wykorzystywana jest koncepcja tzw. namespace. Są to specjalne foldery służące do dynamicznego ładowania plików graficznych oraz samej aplikacji dla danego sklepu.

Standardowy namespace nosi nazwę **core** i w nim domyślnie obsługiwane są wszelkie mechanizmy sklepu. O ile sklep nie ma zdefiniowanego innego namespace, aplikacja będzie łączyć pliki z core. W przypadku arkuszy CSS może to wyglądać następująco:

```
<link rel="stylesheet" href="http://geko.pl/design/_css_frontend/core/static.css" type="text/css"/>
```

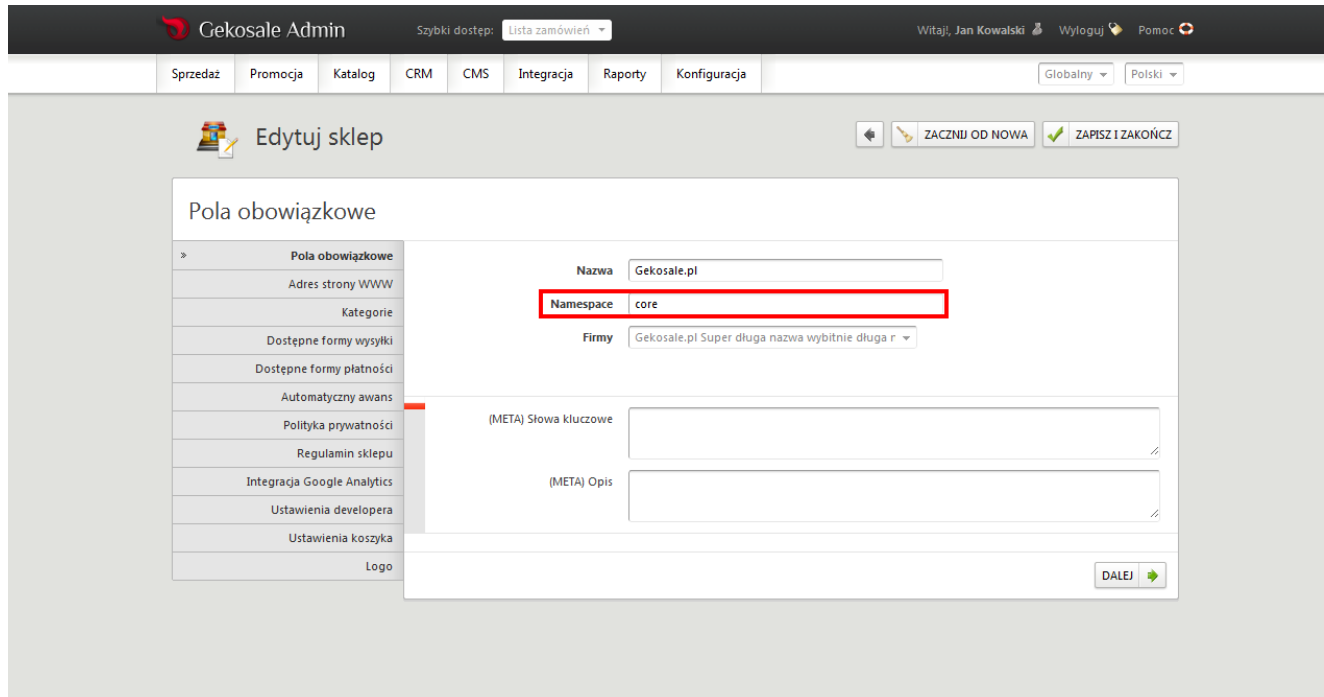
```
<link rel="stylesheet" href="http://geko.pl/design/_css_frontend/core/old.css" type="text/css"/>
```

```
<link rel="stylesheet" href="http://geko.pl/design/_css_frontend/core/scheme.css" type="text/css"/>
```

<link rel="stylesheet" href="http://geko.pl/design/\_css\_frontend/core/scheme-new.css" type="text/css"/>

Jak widać aplikacja wczytuje pliki CSS z folderu o takiej samej nazwie jak namespace.

Bardzo prosto można obejść ten mechanizm dzięki możliwości ustawienia własnego namespace w edycji sklepu. W tym celu należy wejść w menu Konfiguracja->Multistore->Sklepy a następnie w edycję danego sklepu.



W pierwszej zakładce widoczne będzie pole namespace z domyślnie podstawioną wartością core.

Jeżeli zmienimy tą nazwę na cokolwiek innego np. **mynamespace** spowodujemy że aplikacja ładując dane będzie szukać plików o identycznych nazwach w folderach z nazwą nowego namespace.

W przypadku **\_css\_frontend** możemy zatem wykonać prostą operację polegającą na **zduplikowaniu** zawartości folderu do nowego namespace. Należy w tym celu utworzyć nowy folder:

**\_css\_frontend**

**core**

**mynamespace**

i skopiować do niego wszystkie lub wybrane pliki CSS z folderu core.

W praktyce przy jakimkolwiek wdrożeniu wystarczy skopiować plik **scheme-new.css** i w nim wprowadzać wszelkie modyfikacje. Jeżeli taki plik znajdzie się w naszym namespace to w kodzie zostanie wygenerowany poniższy link do arkusza CSS

```
<link rel="stylesheet" href="http://geko.pl/design/_css_frontend/mynamespace/scheme-new.css" type="text/css"/>
```

W tym przypadku plik **scheme-new.css** ładowany jest jako ostatni więc można w nim zastosować koncepcję nadpisywania klas CSS. Jeżeli w którymkolwiek pliku CSS ładowanym wcześniej np. **static.css**, **scheme.css** znajdują się klasy które chcesz zmienić, możesz je skopiować do nowego pliku **scheme-new.css** i tam wykonać dowolne zmiany.

To wszystko co jest potrzebne aby samodzielnie modyfikować pliki CSS w Gekosale.

### UWAGA:

Zalecamy stosowanie własnych namespace gdyż wtedy pliki nie będą nadpisywane podczas aktualizacji. W przypadku jeżeli wprowadzisz zmiany w plikach w namespace core podczas najbliższej aktualizacji zostaną one nadpisane przez system. To co wprowadzisz w namespace Gekosale pomija przy aktualizacji.

## 1.5 Sposób ładowania plików JS w Gekosale

Wszystkie pliki JS są wczytywane za pośrednictwem mechanizmu kompresującego Minify. Tym samym nie można bezpośrednio w nie ingerować i w chwili obecnej nie mają tutaj zastosowania namespace. Niemniej jednak jeżeli chcesz dodać własne pliki JS możesz to zrobić standardowo w 2 krokach

1. utworzyć podfolder w **\_js\_frontend** np. **mynamespace**.
2. wgrać do niego wymagane przez szablon pliki JS
3. podpiąć skrypty w standardowy sposób w sekcji **<head>** w pliku **header.tpl** (utworzonym w namespace , ale o tym później).

W przyszłości planowane jest wprowadzenie pełnej obsługi plików JS również w namespace.

## 1.6 Sposób ładowania szablonów globalnych w Gekosale

Skoro już wiesz jak stworzyć namespace i wykorzystać go do plików CSS to wiesz również jak wykorzystać ten mechanizm do nadpisywania 90% szablonów sklepu :).

W folderze **\_tpl/core** umieszczone są globalne pliki szablonów wykorzystywane w Gekosale.

**footer.tpl** – szablon stopki sklepu

**header.tpl** – szablon nagłówka sklepu, w nim ładowane są pliki CSS

**item.tpl** – szablon pojedynczego produktu na liście produktów (aktualnie bez zastosowania)

**items.tpl** – globalny szablon listy produktów, zmieniając go modyfikujesz listy we wszystkich modułach

**layout.tpl** – szablon układu podstrony czyli wszystkiego między header a footer.

**layoutbox.tpl** – szablon każdego boksu w sklepie

**pagination.tpl** – szablon stronicowania wyników w listach produktów

Jeżeli zatem zduplikujesz te pliki do folderu **\_tpl/mynamespace** będziesz mógł wprowadzać zmiany w konstrukcji większości elementów strony ingerując bezpośrednio w pliki .tpl.

## 1.7 Sposób ładowania szablonów modułów w Gekosale

Modyfikacje poszczególnych modułów w Gekosale są bardzo podobne do modyfikacji plików globalnych z tą różnicą, że zawsze musisz stworzyć namespace na pliki szablonu oraz pliki modułu.

Posłużmy się tutaj przykładem. Zakładamy że chcesz całkowicie przebudować wygląd menu kategorii w sklepie. Prócz ustawienia namespace w konfiguracji sklepu musisz wykonać poniższe kroki.

1. w folderze **plugin/frontend** stworzyć podfolder **mynamespace** (równoległy do core)
2. skopiować do niego cały podfolder dotyczący danego modułu (zawiera w sobie najczęściej controller + model lub tylko controller).

3. w folderze **design/frontend** stworzyć podfolder **mynamespace**

4. skopiować do niego pliki szablonu dla danego modułu.

Gekosale posiada usystematyzowaną strukturę plugin'ów oraz szablonów. Oznacza to, że pliki szablonu danego pluginu będą umieszczone w folderze o takiej samej nazwie.

Poniższa tabela to zestawienie wszystkich standardowo dostępnych boksów wraz z odpowiadającymi folderami w **plugin/frontend/core** oraz **design/frontend/core**

Adresy klienta	ClientAddressBox
Advertisement	GraphicsBox
Akcesoria do produktów	ProductsUpSellBox
Ankieta	PollBox
Bestsellery w sklepie	ProductBestsellersBox
CMS - podgląd treści	CmsBox
Formularz rejestracji klienta	RegistrationCartBox
Formularz logowania	ClientLoginBox
Infolinia	GraphicsBox
Karta produktu	ProductBox
Kontakt	ContactBox
Kupiono również	ProductBuyAlsoBox
Lista dostępnych form płatności	PaymentBox
Lista kategorii	CategoriesBox
Lista newsów	NewsListBox
Lista nowości w sklepie	ProductNewsBox
Lista otagowanych produktów	ProductTagsListBox
Lista produktów w kategorii	ProductsInCategoryBox
Lista promocji w sklepie	ProductPromotionsBox
Lista tagów	TagsBox
Lista tagów klienta	ClientTagsBox
Lista życzeń klienta	WishlistBox
Newsletter	NewsletterBox
Nowości w sklepie	ProductNewsBox
Podgląd zawartości koszyka	CartPreviewBox
Poleć znajomemu	RecommendFriendBox
Polityka prywatności	PrivacyPolicyBox
Produkty podobne	ProductsSimilarBox
Promocje w sklepie	ProductPromotionsBox
Przypomnij hasło	ForgotPasswordBox
Regulamin sklepu	TermsAndConditionsBox
Showcase	ShowcaseBox
Sitemap	SitemapBox
Sprzedaż krzyżowa	ProductsCrossSellBox
Ustawienia klienta	ClientSettingsBox
Wiadomość	NewsBox
Wyszukiwarka	ProductSearchListBox
Wyszukiwarka box	SearchBox
Zamówienia klienta	ClientOrderBox
Zawartość koszyka	CartBox

Wracając do przykładu, chcąc stworzyć pełne modyfikacje szablonu menu kategorii musimy wykonać tylko poniższe kroki.

1. Zduplikować folder `plugin/frontend/core/categoriesbox/` do namespace aby ścieżka była taka : `plugin/frontend/mynamespace/categoriesbox/`
2. Zduplikować folder `design/frontend/core/categoriesbox/` do namespace aby ścieżka była taka : `design/frontend/mynamespace/categoriesbox/`

Od tego momentu możemy wprowadzać ręczne zmiany w pliku `index/index.tpl` umieszczonym w `categoriesbox`. Zmiany te nie zostaną nadpisane podczas aktualizacji.

W analogiczny sposób można postępować z każdym modulem widocznym na liście.

## 1.8 Znaczniki szablonów dostępne globalnie

W Gekosale istnieje krótka lista zmiennych dostępnych globalnie w każdym szablonie. Najważniejsze z nich:

`{ $URL }` – adres sklepu

`{ $CURRENT_URL }` – aktualny adres strony

`{ $DESIGNPATH }` – dynamicznie ustalana ścieżka do folderu design

`{ $NAMESPACE }` – nazwa aktualnie wykorzystywanego namespace

`{ $CURRENT_CONTROLLER }` – aktualny kontroler czyli podstrona na jakiej jesteśmy

`{ $SHOP_NAME }` – nazwa sklepu

`{ $SHOP_LOGO }` – nazwa pliku z logo sklepu

`{ $client }` – tablica zawierająca wszystkie dane aktualnie zalogowanego klienta

`{ $clientdata }` – jak wyżej, może być stosowana zamiennie

`{ $appversion }` – aktualna wersja oprogramowania

`{ $language }` – ID aktualnie wybranego języka

`{ $languageCode }` – symbol aktualnie wybranego języka np. `pl_PL`

`{ $languageFlag }` – tablica języków wraz z ich nazwami i flagami

`{ $currencies }` – tablica walut dostępnych w sklepie

`{ $breadcrumb }` – tablica zawierająca ścieżkę Tu jesteś



`{ $contentcategory }` – tablica wielowymiarowa zawierająca drzewko kategorii statycznych CMS

`{ $gacode }` – kod śledzący Google Analytics

`{ $trackstock }` – wartość 0/1 informująca o tym czy włączone jest śledzenie stanów magazynowych

`{ $enablerating }` – wartość 0/1 informująca o tym czy w sklepie włączone są opinie

`{ $cartredirect }` – adres URL na jaki ma nastąpić przekierowanie do koszyka po dodaniu produktu

`{ $categories }` – tablica wielowymiarowa zawierająca drzewko kategorii w sklepie