



WYŻSZA SZKOŁA BIZNESU
NATIONAL-LOUIS UNIVERSITY

Złożenie pracy online:

2011-03-27 01:51:07

Kod pracy:

4789

Kod załącznika:

4739

**WYDZIAŁ INFORMATYKI
KIERUNEK: INFORMATYKA
SPECJALNOŚĆ: INŻYNIERIA OPROGRAMOWANIA**

Piotr Pruski

(Nr albumu: 8433*INF/INŻ)

**Aplikacja typu RIA wraz z systemem zarządzania treścią oraz
zaawansowanymi metodami SEO: plushaki.pl**

**RIA type application with content management system and advanced
SEO capabilities: plushaki.pl**

Praca inżynierska

Promotor: **dr Franciszek Białas**

Spis Treści

1 Wstęp.....	5
1.1 Tło komercyjne projektu.....	5
1.2 Opis wykonanej pracy.....	6
1.2.1 Logotyp.....	6
1.2.2 Wizytówki, papier firmowy, metki itp.....	7
1.3 Strona internetowa.....	8
1.4 Użyte aplikacje.....	8
1.4.1 Adobe Photoshop CS4.....	8
1.4.2 Adobe Illustrator CS4.....	8
1.4.3 Adobe InDesign CS4.....	8
1.4.4 Adobe Flash CS3 + FlashDevelop.....	9
1.5 Użyte technologie.....	9
1.5.1 PHP 5 oraz PRADO.....	9
1.5.2 ActionScript 3.0 oraz Gaia.....	10
1.5.3 TweenLite.....	10
1.5.4 MySQL v5.....	12
1.5.5 JavaScript.....	13
2 Prace programistyczne Front-end.....	13
2.1 Struktura aplikacji oraz organizacja plików.....	13
2.1.1 Warstwy aplikacji.....	13
2.1.2 Pliki aplikacji.....	14
2.2 Obsługa języków.....	15
2.3 Nawigacja.....	16
2.3.1 Guziki Menu.....	16
2.3.2 Kontrolowanie wyświetlanych podstron.....	18
2.3.3 Kontrolowanie zaznaczonych elementów menu.....	19
2.4 Moduł kolekcji.....	20
2.4.1 Struktura danych.....	20
2.4.2 Parsowanie danych XML.....	21
2.4.3 Menu kategorii.....	22
2.4.4 Prezentacja Obiektów kategorii.....	24
2.5 Moduł kontaktu.....	28
2.5.1 Formularz kontaktowy.....	28
2.5.2 Wysłanie wiadomości.....	29
2.6 Wykorzystanie swfobjec oraz swfaddress.....	30
2.6.1 SEO oraz deep linking.....	30
2.6.2 Sekwencja działania.....	31
2.7 Mniejsze moduły.....	33
2.7.1 Scroller.....	33
2.7.2 Dynamiczna galeria fade in-out.....	35
3 Prace programistyczne back-end.....	37
3.1 Struktura projektu.....	37
3.2 Struktura bazy danych.....	39
3.3 Autentykacja oraz zabezpieczenia.....	41
3.4 Edytor Kolekcji.....	42
3.4.1 CategoryDetails.....	45
Moduł mailingu.....	50

3.5.1 Szablon wiadomości HTML.....	50
3.5.2 Wysyłanie wiadomości.....	51
3.5.3 Test wiadomości.....	53
4 Podsumowanie.....	55
5 Źródła.....	56
5.1 Bibliografia.....	56
5.2 Spis rysunków.....	56

1 Wstęp

Celem niniejszej pracy jest opisanie procesu tworzenia projektu plushaki.pl, będącego aplikacją internetową wykonaną głównie w technologii Adobe Flash i posiadającą ciekawe rozwiązania pozycjonujące produkt w wyszukiwarkach internetowych. Strona posiada system zarządzania treścią bazujący na technologii PHP oraz MySQL.

Gotowy produkt do testów przez czytelnika dostępny jest pod adresem:

<http://www.mmediasc.com/plushaki/>

Do dostępu do panelu administracyjnego wymagane są następujące dane:

URL: <http://www.mmediasc.com/plushaki/admin>

login: admin

hasło: test

Jako załączniki do niniejszej pracy dostarczone zostały również:

- Źródła strony Front-end
- Źródła panelu administracyjnego
- Skrypt SQL generujący strukturę bazy
- Pliki projektowe grafik wykorzystanych podczas produkcji strony

1.1 Tło komercyjne projektu

Plushaki.pl to dystrybutor pluszowych maskotek mających zastosowanie w promocji i brandingu przedsiębiorstw. Firma przede wszystkim oferuje maskotki posiadające nadruk indywidualnie projektowany dla klientów oraz maskotki 'zwykłe', nie posiadające dodatkowych nadruków. Produkty wytwarzane całkowicie w fabryce w Chinach włącznie z ewentualnym procesem dodatkowego brandingu.

Projekt polegał na stworzeniu kompletnego wizerunku graficznego nowo powstałej marki. Nakreślono wytyczne dla CI (Corporate Identity) definiując pożądane cechy rezultatu końcowego jako:

- Czysty, lekki i przyjazny dla oka
- Mocny akcent zabawek
- Oryginalny

Grupa docelowa została zdefiniowana jako:

- Manager marketingowy firm i korporacji
- Właściciele firm z sektora MŚP
- Manager marketingowy sklepów sieciowych

1.2 Opis wykonanej pracy

1.2.1 Logotyp

Logotyp Plushaki.pl z założenia miał kojarzyć się z zabawkami oraz misiami. Z dostarczonych ok. 10 propozycji klient wybrał poniższą:



Rys. 1

Po więcej informacji odnośnie rozwiązań internetowych dla plushaki.pl proszę się odnieść do wszystkich rozdziałów następujących po rozdziale drugim włącznie.

1.4 Użyte aplikacje

1.4.1 *Adobe Photoshop CS4*

Według wielu, Photoshop to najlepszy program do tworzenia grafiki rastrowej dostępny na rynku. Za jego pomocą zostały wykonane koncepcje graficzne strony – front-end jak i back-end.

1.4.2 *Adobe Illustrator CS4*

W przeciwieństwie do Photoshopa, Illustrator operuje na grafice wektorowej, dzięki czemu był idealnym narzędziem do produkcji Logotypu.

1.4.3 *Adobe InDesign CS4*

InDesign jest aplikacją pozwalającą łatwo aranżować projekty przeznaczone do druku w jedną całość, dlatego też był doskonałym narzędziem do stworzenia koncepcji wizytówek oraz metek produktów.

1.4.4 *Adobe Flash CS3 + FlashDevelop*

Flash to doskonałe narzędzie do produkcji aplikacji internetowych typu RIA (Rich Internet Application) chcących oferować wysoki poziom interaktywności z użytkownikiem. Przede wszystkim oferuje zintegrowaną platformę umożliwiającą poszerzenie granic animacji, interaktywności, rozmieszczenia grafiki czy synchronizowanego wykorzystania dodatkowych mediów, takich jak dźwięk czy film. Projekty tworzone za pomocą Adobe Flash są kompilowane do plików wykonawczych SWF ('Small Web Format' – poprzednio 'ShockWave Flash') i odtwarzane przez Adobe Flash Player i jego warianty będące popularnymi wtyczkami do przeglądarek internetowych.

Pliki wykonawcze SWF są ogólnie uznawane za aplikacje wykonywane po stronie klienta (Client Side Coding) lecz pozostaje wiele możliwości do asynchronicznej komunikacji ze skryptami po stronie serwera (web serwisy SOAP czy JSON, zdalna komunikacja ze skryptami PHP czy ASP, otwieranie socketów itp.) co dodaje jeszcze więcej możliwości.

Mimo, iż ActionScript od wersji 3.0 jest w pełni dojrzałym, obiektowym językiem programowania to środowisko programistyczne oferowane przez aplikacje Adobe Flash pozostawia wiele do życzenia, dlatego postanowiłem wykorzystywać potęgę FlashDevelop.

FlashDevelop to aplikacja rozpowszechniana na licencji MIT, umożliwiająca znacznie wygodniejsze pisanie i organizowanie kodu w oparciu o ActionScript niż jest to oferowane przez Adobe. Jego główne zalety to: automatyczne podkreślanie oraz formatowanie syntaksu, code hinting, linkowanie własnych bibliotek oraz praktyczne zarządzanie plikami projektu.

Adobe Flash w zestawieniu z FlashDevelop stwarzają doskonałe środowisko do pracy nad projektami, których podstawowa funkcjonalność oparta jest na ActionScriptie 3.0 lecz, które wykorzystują również funkcjonalność oferowaną przez graficzne i animacyjne narzędzia Flasha.

1.5 Użyte technologie

1.5.1 PHP 5 oraz PRADO

PHP to skryptowy język programowania zorientowany głównie na tworzenie dynamicznych stron/aplikacji internetowych. Interpreter PHP jest darmowy i z założenia działa po stronie serwera jako 'Serwer Side Skriptyng', na wejściu przyjmując kod PHP a na wyjściu serwując kod HTML. PHP posiada moduły umożliwiające łączenie się z wieloma rodzajami relacyjnych baz danych oraz wykonywanie kwerend i odbieranie danych. Podstawowe koncepcje programowania obiektowego zostały wprowadzane już w wersji 3 i ulepszone w wersji 4 PHP lecz dopiero w wersji 5 zaimplementowano w pełni dojrzałe podejście programowania obiektowego, takiego jakie możemy spotkać w np. C++ czy Javie.

PRADO (PHP Rapid Application Development Object-oriented) to 'Framework', bazujący na PHP5 i rozszerzający jego funkcjonalność. PRADO został stworzony celem ułatwienia i przyśpieszenia budowy aplikacji webowych, jego kluczowe funkcjonalności to:

- Architektura zorientowana na zdarzenia

- Wyższy poziom abstrakcji dla obsługi baz danych – zaimplementowany Active Record Pattern
- Bazuje na wariancie MVC pattern (Model View Controller)

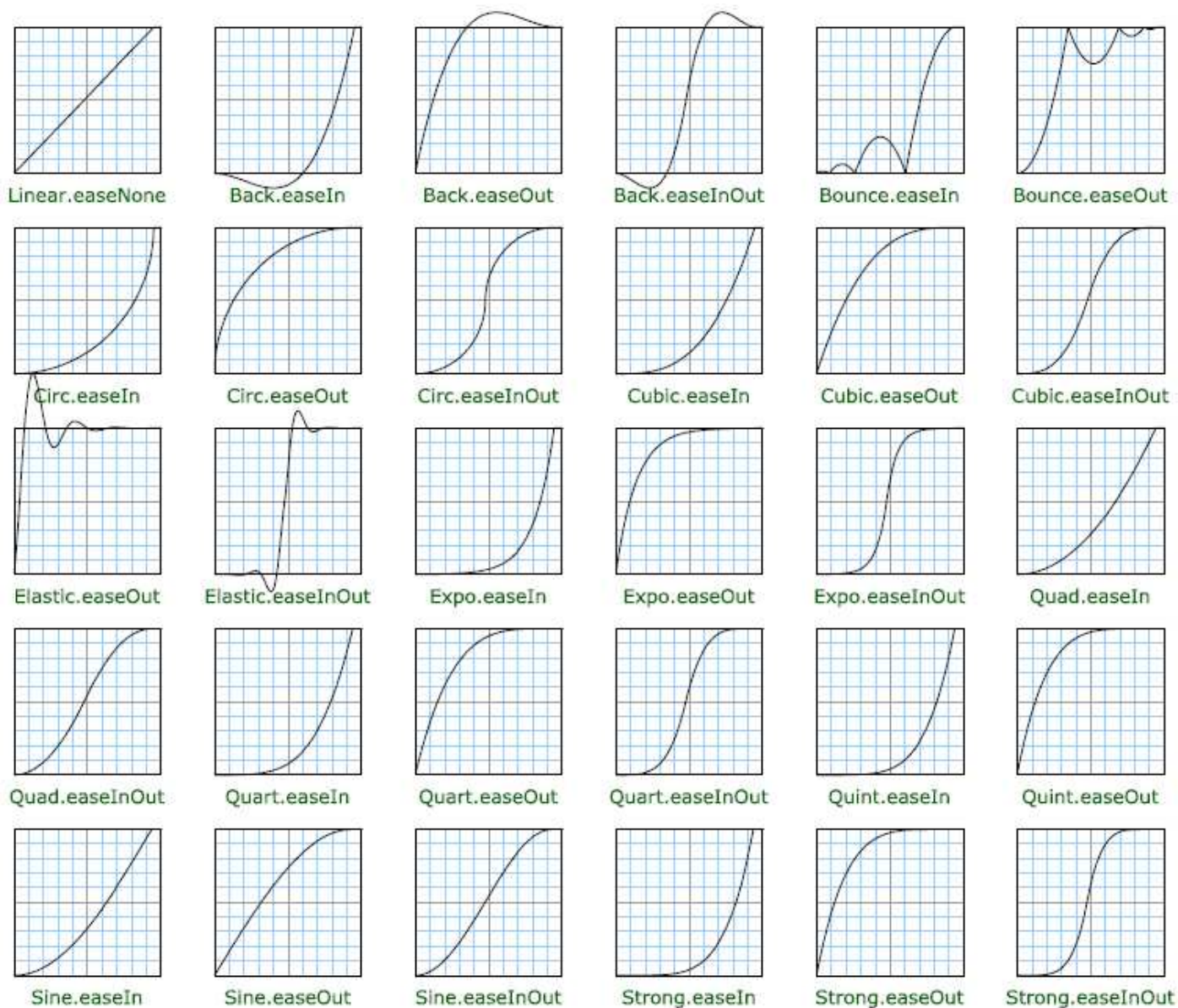
1.5.2 ActionScript 3.0 oraz Gaia

ActionScript 3.0 to w pełni obiektowy język programowania, z przeznaczeniem do tworzenia aplikacji odtwarzanych przez Adobe Flash Player. W przeciwieństwie do AS1 oraz AS2, które swojej 'obiektywnej' naturze pozostawiały wiele do życzenia, trzecia już wersja jest prawdziwie obiektowym językiem programowania, swoją semantyką bardziej przypominając Jave.

Gaia Framework to wtyczka do Adobe Flah oraz biblioteka AS3 dająca dodatkową funkcjonalność przydatną podczas tworzenia stron bazujących w 100% na technologii Flash jako GUI. Podczas używania proponowanej przez Gaia struktury kodu jesteśmy w stanie kontrolować, poprzez nasłuchiwanie odpowiednich zdarzeń, stan aplikacji i serwować odpowiednio dynamicznie ładowaną treść. Śmiało mogę powiedzieć, iż używanie Gaia framework skraca czas produkcji aplikacji o ustalonej strukturze o ok. 40%.

1.5.3 TweenLite

To lekka i szybka biblioteka napisana w AS3 przeznaczona do programistycznego animowania obiektów na scenie. Działa na zasadzie zmiany wyznaczonych numerycznych wartości obiektów w określonej przestrzeni czasowej. Obsługuje również wiele zdarzeń takich jak 'onInit', 'onComplete', 'onUpdate', itp. kontrolowanych poprzez delegacje własnych funkcji. Wykonanie żądanej animacji odbywa się poprzez egzekucje funkcji matematycznych względem zadanego czasu. Jest wiele dostępnych funkcji napisanych przez autorów, lecz istnieje możliwość napisania własnych. Oto nie kompletna lista podstawowych funkcji animacji:



Rys. 2

1.5.4 MySQL v5

MySQL to aktualnie najpopularniejszy serwer relacyjnej bazy danych używany przez właścicieli stron internetowych. Oferuje relatywnie szybkie działanie i wykorzystuje łatwy w użyciu język SQL. Ponadto jest to aplikacją OpenSource o niedrożej licencji, co przyczynia się do tak znacznego zdominowania rynku.

1.5.5 JavaScript

JavaScript to bardzo popularny język skryptowy typu Client-Side Scripting wykonywany głównie przez przeglądarkę internetową po załadowaniu strony na maszynie klienta. Stosowany przede wszystkim do zwiększenia interaktywności stron internetowych i wykonywania czynności bez konieczności przeładowania strony.

2 Prace programistyczne Front-end

2.1 Struktura aplikacji oraz organizacja plików

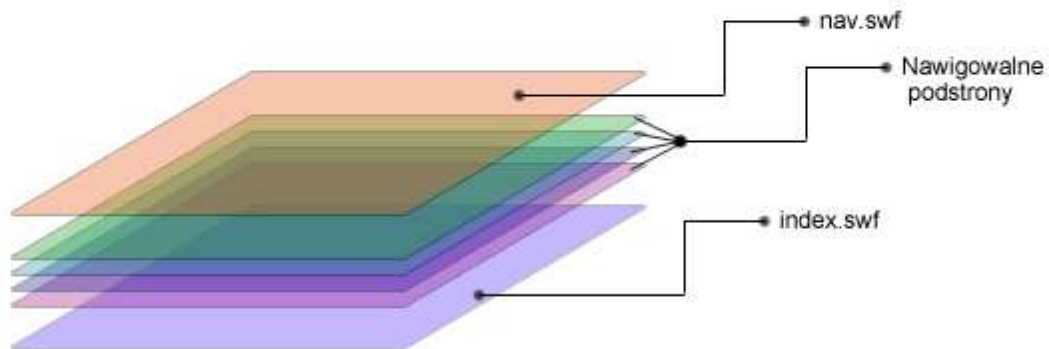
Struktura aplikacji oraz organizacja plików jest ściśle powiązana z założeniami Gaia Framework. W podkatalogu 'src/' znajdują się wszystkie pliki projektowe Flash (rozszerzenie *.fla) natomiast w podkatalogu 'src/classes/pages/' znajdziemy definicje ich klas.

Każdy plik jest kompilowany osobno jako autonomiczny byt, lecz synchronizowany poprzez zdarzenia wysyłane przez Gaia Framework.

Głównym punktem wejścia aplikacji jest plik main.fla (po kompilacji main.swf) oraz jego klasa definiująca Main (znajdująca się w 'src/classes/Main.as'). To ona, w sposób dynamiczny, importuje na scenę pozostałe pliki swf. Informacje jakie pliki importować, pobiera z pliku XML znajdujące się pod ścieżką relatywną '/xml/site.xml'.

2.1.1 Warstwy aplikacji

Filmy swf są importowane na zasadzie warstw i nakładane w odpowiedni sposób na siebie. Poniższy diagram obrazuje sposób ładowania plików aplikacji plushaki.pl oraz ich układanie:



Rys. 3

Przy takim sposobie ładowania plików:

- index.swf staje się 'tłem' i zawiera wszystkie dzielone grafiki i funkcjonalności.
- nav.swf zawiera nawigację, która jest zawsze widoczna na każdej z podstron (oprócz wyboru języka) jako najwyższa warstwa.
- main.swf jest głównym kontenerem aplikacji, zawiera wszystkie globalne funkcje i zmienne dostępne dla jej dzieci.

2.1.2 Pliki aplikacji

Aplikacja posiada bardzo wyraźnie zorganizowaną strukturę plików, gdzie każdy plik jest odpowiedzialny za konkretną funkcjonalność danej części aplikacji. By ułatwić czytelnikowi odnośnienie się do tych konkretnych plików proszę użyć poniższej tabeli:

Page ID	Plik Projektowy	Plik Klasy	Plik SEO	Przestrzeń nazw obiektów (pakiet)	Plik wyjściowy
index	index fla	IndexPage.as	---	----	index.swf
nav	nav fla	NavPage.as	---	pages.nav.*	nav.swf
language	language fla	LanguagePage.as	language.php	pages.languageSelect.*	Language.swf
mtw	mtw fla	MtwPage.as	mtw.php	---	mtw.swf
filozofia	filozofia fla	FilozofiaPage.as	filozofia.php	---	filozofia.swf
kolekcja	kolekcja fla	KolekcjaPage.as	kolekcja.php	pages.kolekcja.*	kolekcja.swf
firmowa	firmowa fla	FirmowaPage.as	firmowa.php	---	firmowa.swf
dystrybutorzy	dystrybutorzy fla	DystrybutorzyPage.as	dystrybutorzy.php	---	dystrybutorzy.swf
kontakt	kontakt fla	KontaktPage.as	kontakt.php	pages.kontakt.*	kontakt.swf
jakosc	jakosc fla	JakoscPage.as	jakosc.php	----	jakosc.swf

- Page ID** – To nazwa modułu do której można odwoływać się w aplikacji.
- Plik Projektowy** – Plik do otworzenia za pomocą Flash CS3 lub wyższy. Pliki znajdują się w katalogu 'src/'.
- Pliki Klasy** – Korespondujące pliki klas zawierające kod AS3. Pliki znajdują się w katalogu. 'src/classes/pages/' i należą do pakietu (przestrzeni nazw) 'pages'.
- Plik SEO** – Zawierają dynamicznie ładowany контент indeksowalny przez wyszukiwarki internetowe. Pliki znajdują się w głównym katalogu rozruchowym ('deploy').
- Przestrzeń nazw obiektów**– Wszystkie dodatkowe obiekty stworzone specyficznie pod użyczek danego modułu.
- Plik wyjściowy** – Skompilowany plik wykonawczy znajdujący się w katalogu 'deploy/'.

2.2 Obsługa języków

Aplikacja została napisana dla dwóch języków. Wybór języka odbywa się na pierwszej stronie (do której można wrócić, w każdej chwili poprzez kliknięcie loga w nagłówku).



Informacja o aktualnie wybranym języku przechowywana jest w publicznej statycznej zmiennej klasy Languages, która wygląda następująco:

```
package pages.languageSelect
{
    public class Languages
    {
        public static const POLISH:String = "pl";
        public static const ENGLISH:String = "en";

        public static var currentLanguage:String;
    }
}
```

Tym samym, w każdym miejscu aplikacji, po uprzednim imporcie pakietu 'pages.languageSelect' możemy sprawdzić aktualnie wybrany język poprzez przyrównanie go do statycznych stałych reprezentujących dany język.

Przykładowo:

```
switch (Languages.currentLanguage)
{
    case Languages.ENGLISH:
        //aktualny język to angielski
        break;
    case Languages.POLISH:
        //aktualny język to polski
        break;
}
```

2.3 Nawigacja

2.3.1 *Guziki Menu*

Celem uniknięcia 'przeskoków' w animacji, które towarzyszą tradycyjnej metodzie tworzenia

animowanych guzików w Adobe Flash zamiast użycia klasy Button napisałem własną klasę do obsługi

guzików menu.

Klasa 'MenuButton' znajduje się w pakiecie 'pages.nav' i dziedziczy z klasy MovieClip. W konstruktorze przypisane zostają dwie funkcje nasłuchujące zdarzeń związanych z kursorem, 'Mouse_Over' oraz 'Mouse_Out' ponadto przyjmuje dwa stany 'selected=1' i 'selected=0'. Guzik graficznie reaguje na zdarzenia oraz stany w jakich się znajduje. Dla przykładu, tak wygląda obsługa zdarzenia 'Mouse_Out':

```
[...]  
//rejestrowanie nasłuchiwanie w konstruktorze  
this.addEventListener(MouseEvent.CLICK, rollOut, false, 0, true);  
[...]  
private function rollOut(Ev:MouseEvent):void  
{  
    if (!selected)  
    {  
        TweenLite.to(selectBg, 0.3, { alpha:0 } );  
    }  
    else  
    {  
        TweenLite.to(selectBg, 0.3, { alpha:1 } );  
    }  
}
```

O klasie 'MenuButton' warto omówić trzy rzeczy:

1. Wszystkie zdarzenia w aplikacji pochodzące z pakietu 'flash.events.*' zawsze warto rejestrować z ustawieniem flagi 'useWeakReference' na 'true'. Praktyka ta, jest zbyt rzadko stosowana wśród programistów a zabezpiecza przed nie kontrolowanymi wyciekami pamięci wynikającymi z nie zarejestrowanych funkcji nasłuchujących po usunięciu innych referencji do danego obiektu. Przy ustawieniu pozytywnie flagi 'useWeakReference' w sytuacji gdy znikną odniesienia do obiektu, Garbage Collector natychmiast usunie obiekt z pamięci, ignorując wciąż zarejestrowaną funkcję nasłuchującą.
2. Standardowy obiekt klasy 'Button' obsługuje 3 stany 'up', 'over', 'down', które mogą zawierać animacje. Niestety przy przejściu z jednego stanu w drugi, w krótszym czasie niż zdąży się wykonać animacja następuje 'przeskok' do klatki wyjściowej animacji nowego stanu guzika. Powoduje to widoczne przez użytkownika załamania w animacji.

Wykorzystując TweenLite oraz zdarzenia kursora do animacji obiektu znikają w/w problemy 'skaczących' czy 'przerwanych' animacji. Zadając TweenLite nowe wartości, przerywa on wykonywanie poprzedniej animacji, natomiast aktualną wartość właściwości, którą animujemy przyjmuje jako początek nowej animacji. Daje to płynny efekt animowania guzików.

3. Obiekty typu 'MenuButton' nie wykonują, żadnej czynności poza zmianą stanów. By przypisać im funkcjonalność po kliknięciu, należy zarejestrować odpowiednie funkcje obsługujące zdarzenia 'MouseEvent.CLICK'.

2.3.2 Kontrolowanie wyświetlanych podstron

Wszystkie tranzycje podstron są kontrolowane w module nawigacyjnym (klasa 'NavPage'), który wyświetla guziki menu uzależnione od aktualnie wybranego języka.

W konstruktorze klasy 'NavPage' następuje rejestracja funkcji obsługującej kliknięcia guzików przez użytkownika (przy czym przyjmujemy, że różne języki mogą mieć różną ilość elementów menu).

```
[...]  
for (var i:uint = 0 ; i < plMenu.numChildren; i++)  
{  
    if (plMenu.getChildAt(i) is MenuButton)  
    {  
        plMenu.getChildAt(i).addEventListener(MouseEvent.CLICK, menuButtonClicked, false, 0, true);  
    }  
}  
  
for (var j:uint = 0 ; j < enMenu.numChildren; j++)  
{  
    if (enMenu.getChildAt(j) is MenuButton)  
    {  
        enMenu.getChildAt(j).addEventListener(MouseEvent.CLICK, menuButtonClicked, false, 0, true);  
    }  
}  
[...]
```

obiekty MovieClip) będące obiektami typu 'MenuButton' i przypisuje im funkcje reagującą na zdarzenie kliknięcia przez użytkownika. Następnie czynność tą powtarza dla obiektów enMenu.

Ponieważ nazwa każdego guzika odpowiada nazwom podstron (Page ID) można pozwolić sobie na poniższą 'wygodę' podczas obsługi zdarzenia kliknięcia :

```
public function menuButtonClicked(Ev:MouseEvent):void
{
    Gaia.api.goto("index/nav/"+ Ev.currentTarget.name);
}
```

Powyższa metoda analizuje obiekt na rzecz, którego zostało wywołane zdarzenie kliknięcia i wykorzystuje jego publiczną zmienną 'name' by wykonać zmianę wyświetlanej podstrony.

Metoda 'Gaia.api.goto(target:string):void' jest odpowiedzialna za schowanie aktualnie wyświetlanej strony (poprzez wywołanie jej metody 'transitionOut():void') oraz wyświetlenie strony, której ścieżka podana jest w parametrze 'target' (poprzez wywołanie jej metody 'transitionIn():void').

2.3.3 Kontrolowanie zaznaczonych elementów menu

Ponieważ aplikacja wykorzystuje pasek adresu przeglądarki podczas zmiany podstron oraz oferuje indeksowalny przez wyszukiwarki content, może zdarzyć się sytuacja, że użytkownik wejdzie bezpośrednio na konkretną podstronę (np. <http://domainName.com/#/kolekcja>). Tym samym, elementy menu nie mogą być zaznaczane na zdarzenia kliknięcia przez użytkownika lecz muszą nasłuchiwać zdarzeń propagowanych podczas zmiany aktualnie wyświetlanej strony. Rejestracja nasłuchu następuje w konstruktorze klasy 'NavPage' :

```
[...]
Gaia.api.afterGoto(onAfterGoto);
Gaia.api.afterComplete(onAfterGoto, false, true);
[...]
```

Pierwsza linijka kodu rejestruje funkcje nasłuchującą zdarzeń, wywoływanych podczas komendy zmiany aktualnie wyświetlanej podstrony (lecz przed jej faktycznym rozporządzeniem).

Druga linijka rejestruje funkcje nasłuchującą zdarzeń wywoływanych podczas zakończenia procesu przejścia na konkretną podstronę. Przekazywane dodatkowe flagi, informują że zdarzenie ma być zarejestrowane tylko raz a następnie funkcja nasłuchująca ma zostać wyrejestrowana. Sytuacja ta ma miejsce tylko i wyłącznie przy pierwszym wyświetleniu którejkolwiek z podstron, to znaczy gdy przeglądarka zostaje przekierowana spoza aplikacji.

Funkcja obsługująca powyższe zdarzenia ma za zadanie przeanalizować przychodzące informacje i zaznaczyć odpowiedni guzik reprezentujący aktualnie wyświetloną podstronę oraz wybrany język:

```
private function onAfterGoto(Ev:GaiaEvent):void
{
    deselectAllButtons();
    var branchChildName:String = Ev.validBranch.split("/").pop();
    [...]
    currentLangMenu.getChildByName(branchChildName).select();
}
```

2.4 Moduł kolekcji

Moduł kolekcji jest połączony z systemem CMS gdzie administrator może manipulować elementami menu oraz obiektami znajdującymi się w nich.

2.4.1 Struktura danych

Struktura danych reprezentowana jest przez kod XML znajdujący się w pliku 'kolekcja.php' i generowana przez skrypt PHP na podstawie wpisów do bazy danych.

```
[...]
<category pl_name="Polska Nazwa" en_name="Angielska Nazwa" >
    <object imageURL="url Pliku PNG" >
        <title_en>Tytuł angielski</title_en>
        <subtitle_en>Opis angielski</subtitle_en>
        <prop_en text="właściwość 1 EN">wartość</prop_en>
```

```

    <prop_en text="właściwość 2 EN">wartość</prop_en>
    <prop_en text="właściwość 3 EN">wartość</prop_en>
    <title_pl>Tytuł polski</title_pl>
    <subtitle_pl>Opis polski</subtitle_pl>
    <prop_pl text="właściwość 1 PL">wartość</prop_pl>
    <prop_pl text="właściwość 2 PL">wartość</prop_pl>
    <prop_pl text="właściwość 3 PL">wartość</prop_pl>
  </object>
  <object imageURL="objectImages/891f4a5a0c5c2003ecc8eb2e30eaf18e.png" >
</category>
<category pl_name="...." en_name="..." >
  [...]
</category>
[...]
```

2.4.2 Parsowanie danych XML

Surowe dane przechowywane są w prywatnej właściwości klasy KolekcjaPage 'categoryData_arr' będącej tabelą obiektów typu 'CategoryRecord' (pakiet pages.kolekcja). 'CategoryRecord' reprezentuje obiekt kategorii i zawiera następujące właściwości:

```

public var pl_name:String;
public var en_name:String;
public var objects_arr:Array;
```

Publiczny atrybut 'objects_arr' zawiera z kolei obiekty typu 'ObjectRecord' (pakiet pages.kolekcja) reprezentujące dane poszczególnych elementów kategorii. Zapełnianie 'objects_arr' odbywa się albo poprzez podanie wejściowej tablicy w konstruktorze albo wykorzystując publiczną metodę 'addObject(obj:ObjectRecord):void'.

Załadowanie zewnętrznych danych oraz ich przetwarzanie odbywa się po zarejestrowaniu polecenia 'transitionIn' lecz przed wykonaniem czynności mających ujawnić moduł użytkownikowi oraz przed wykonaniem metody 'transitionInComplete'. Czynność parsowania danych XML odbywa się w prywatnej metodzie 'parseXMLgroups(categories:Object):Array', która zwraca tablice gotową do zapisania w/w atrybutowi 'categoryData_arr'.

Sam proces parsowania odbywa się w bardzo standardowy sposób, zalecany przez Adobe z wykorzystaniem publicznej klasy 'XML' i sprowadza się do mapowania struktury wejściowego kodu XML na obiekty typu w/w klas.

2.4.3 *Menu kategorii*

Kontenerem najwyższego poziomu dla menu bocznego kategorii jest obiekt klasy 'CategoryMenu' (pakiet pages.kolekcja.categoryMenu) odpowiada on za rozmieszczenie elementów na scenie oraz propagację zdarzeń związanych z interakcją użytkownika.

Atrybuty jakie znajdziemy w klasie to:

```
private const itemSpacing:uint = 30;
public var category_lbl:TextField;
public var itemsHolder:MovieClip;
public var scroller:Scroller;
```

gdzie:

- itemSpacing** – to wartość w px odpowiadająca przestrzeni między jednym elementem a następnym.
- itemsHolder** – to kontener dla obiektów typu 'MenuItemMC' (pakiet pages.kolekcja.categoryMenu), posiada on dodatkową maskę przystosowaną do użycia Scrollera.
- scroller** – instancja klasy Scroller (pakiet pruskiUtils), czyli dynamicznie przypisywanego 'suwaka' który w animowany sposób przewija obiekty typu DisplayObject i ich potomstwo.

Konstrukcja oraz dodawanie nowych obiektów do menu następuje za pomocą publicznej metody ConstructItem.

```
public function constructItem(categoryRec:CategoryRecord):void
{
    var item = new MenuItemMC();
    switch (Languages.currentLanguage)
    {
```

```

        case Languages.POLISH:
            item.construct_MenuElement(categoryRec.pl_name, categoryRec.objects_arr);
            break;

        case Languages.ENGLISH:
            item.construct_MenuElement(categoryRec.en_name, categoryRec.objects_arr);
            break;
    }
    item.y = itemsHolder.numChildren * itemSpacing;

    MovieClip(item).addEventListener(MouseEvent.CLICK, menuItemClickedHandler, false, 0, true);

    itemsHolder.addChild(item);
}

```

Zostało wspomniane wcześniej, że obiekt typu 'CategoryRecord' przekazywany do powyższej metody zawiera surowe dane synchronizowane z bazą danych. W zależności od aktualnego języka, dane te są używane do konstrukcji poszczególnych elementów menu. Elementy menu otrzymują w konstruktorze nazwę oraz kopie tabeli 'objects_arr' zawierającej dane o elementach kategorii. Następnie otrzymują swoje koordynaty na osi Y oraz zostaje im przypisana metoda obsługująca zdarzenie kliknięcia użytkownika. Na zakończenie zostają one dodane do sceny obiektu itemsHolder (typu MovieClip) i tym samym widoczne dla użytkownika.

Obsługa zdarzenia kliknięcia przez użytkownika odbywa się w prywatnej metodzie 'menuItemClickedHandler', która wykorzystuje publiczną metodę 'selectMenuItem'.

```

private function menuItemClickedHandler(Ev:MouseEvent):void
{
    selectMenuItem(MenuElementMC(Ev.currentTarget));
}

public function selectMenuItem(element:MenuElementMC):void
{
    deselectAll();
    MenuElement(element).select();
    dispatchEvent(new CategoryMenuEvent(CategoryMenuEvent.ITEM_CLICKED, true, false,
MenuElement(element).objectArray));
}

```

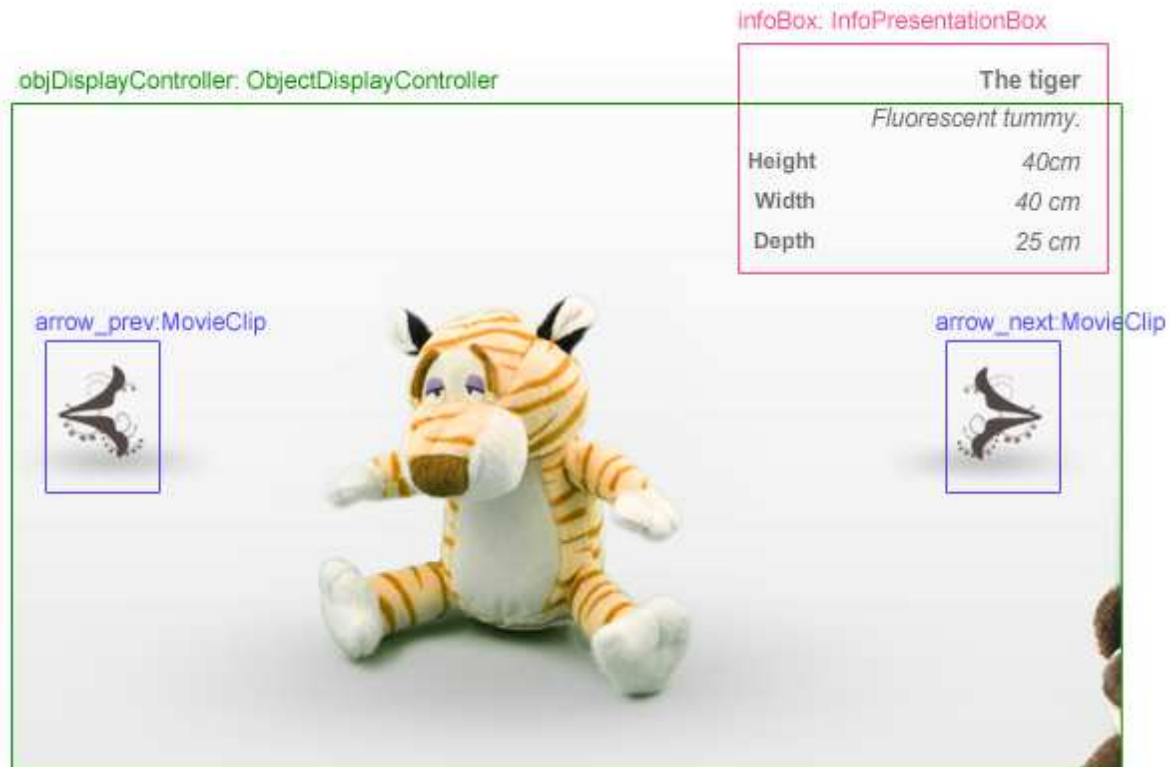
Metoda 'selectMenuItem' ma na celu wizualne zaznaczenie aktualnie wyświetlanej kategorii oraz propagację zdarzenia selekcji owej kategorii. Propagowany obiekt typu 'CategoryMenuEvent' (pakiet pages.kolekcja.categoryMenu) rozszerza klasę Event (pakiet flash.events) dzięki czemu można wykorzystać mechanizm 'dispatchEvent' i poinformować nasłuchujące obiekty o zaistniałym zdarzeniu. Ponadto obiekt 'CategoryMenuEvent' w konstruktorze otrzymuje tablicę danych o elementach danej kategorii, dzięki czemu obiekty odpowiedzialne za wyświetlenie owej kategorii będą mogły korzystać bezpośrednio z danych zawartych w obiekcie zdarzenia.

2.4.4 Prezentacja Obiektów kategorii

Zdarzenie kliknięcia elementu menu obsługiwane jest w metodzie 'categoryMenuItemClickedHandler' obiektu klasy 'KolekcjaPage'.

```
public function categoryMenuItemClickedHandler(Ev:CategoryMenuEvent):void
{
    if(Ev.getObjectsArr().length>0)
        objectPresentator.loadObjects(Ev.getObjectsArr());
}
```

W metodzie tej wykonywane jest polecenie załadowania elementów do wyświetlenia poprzez objectPresentator, wykorzystywana jest tabela zawierająca informacje pochodzące z bazy danych. objectPresentator to instancja klasy 'ObjectPresentation' (pakiet pages.kolekcja.objectPresentation) i jest odpowiedzialny za wyświetlanie galerii obiektów wybranej kategorii. Ponadto nasłuchuje na zdarzenia operacji użytkownika i ustawia stan galerii by pokazywały żądane informacje. Oto elementy znajdujące się w obiekcie typu 'ObjectPresentation':

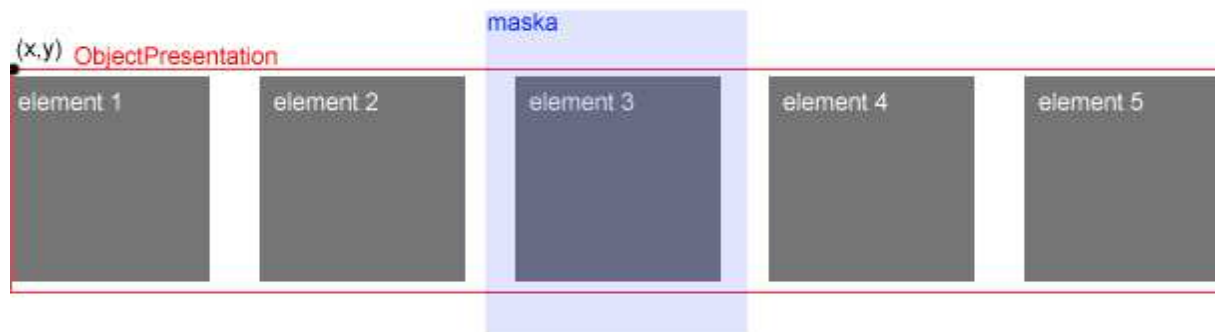


Rys. 4

Przed wszystkim objectPresentator wykonuje następujące funkcje:

1. Inicjuje potomne komponenty danymi z bazy.
2. Nasłuchuje kliknięcia w arrow_prev oraz arrow_next.
3. Informuje objDisplayController w jakiej pozycji się ustawić oraz nakazuje infoBox wyświetlić konkretne dane aktualnie prezentowanego obiektu.
4. W razie dojścia do wartości granicznych zbioru, włącza i wyłącza funkcjonalność oraz widoczność arrow_prev oraz arrow_next.

objDisplayController (instancja klasy pages.kolekcja.objectPresentation) odpowiada za przesunięcia wyświetlanych obiektów. Oto uproszczony schemat logiki działania tego obiektu:



Rys. 5

Elementy do wyświetlenia zostają ładowane poprzez użycie metody 'loadObjects'

```
public function loadObjects(objArr:Array):void
{
    unload();
    for (var i = 0; i < objArr.length; i++)
    {
        displayObjectsArray.push(this.addChild(new ObjectDisplayItem(objArr[i].imageURL)));
    }
    distributeObjects();
}
```

W metodzie tej tworzone są elementy, dodawane do sceny a ich referencja jest umieszczana w prywatnym atrybucie displayObjectsArray. Metoda 'distributeObjects' rozmieszcza dodane elementy na scenie używając parametrów ustawień ('private const objectSpacing:uint').

Wyświetlenie żądanego obiektu sprowadza się do przesunięcia całego kontenera (ObjectPresentation) o odpowiednią ilość piksli na osi X. Dzięki zastosowaniu maski przykrywającej kontener, w danej chwili wyświetlana jest tylko jedna pozycja. Proces wyświetlania zadanego obiektu realizowany jest przez wywołanie publicznej metody 'goToItemNo'

```
public function goToItemNo(itemNo:uint):void
{
    var newXLocation:Number = initialX - (objectSpacing * itemNo);
    Tweener.addTween(this,{x:newXLocation,time:transitionSpeed,transition:transitionType});
}
```

Metoda ta w pierwszej kolejności oblicza nową lokalizację X (`newXLocation`) a następnie używa biblioteki Tweenery do animowanego przemieszczenia kontenera.

2.5 Moduł kontaktu

Zakładka kontakt, wyświetla formularz kontaktowy umożliwiający wysłanie pojedynczej wiadomości email do właściciela serwisu.

2.5.1 Formularz kontaktowy

Formularz to trzy pola tekstowe oraz guzik wysyłający, klasa `MailForm` znajduje się w pakiecie `'pages.kontakt'`. Ciekawym rozwiązaniem jest zaznaczenie pola aktualnej edycji poprzez animowanie tła. Efekt jest osiągnięty poprzez stworzenie 'poniżej' pola tekstowego obiektu typu `'MovieClip'` i animowanie go na zdarzenia `'FOCUS_IN'` oraz `'FOCUS_OUT'`.

W konstruktorze klasy `'MailForm'` następuje rejestracja zdarzeń myszki.

```
public function MailForm():void
{
    sendButton.buttonMode = true;
    sendButton.addEventListener(MouseEvent.CLICK, sendButtonClickedHandler);
    [...]
    wiadomoscText_txt.addEventListener(FocusEvent.FOCUS_IN, focusInHandler, false, 0, true);
    wiadomoscText_txt.addEventListener(FocusEvent.FOCUS_OUT, focusOutHandler, false, 0, true);
    [...]
}
```

Zdarzenia zostają obsługiwane w następujący sposób.

```
private function focusInHandler(Ev:FocusEvent):void
{
    [...]
    if (Ev.currentTarget == wiadomoscText_txt)
```

```

        {
            wiadBG.gotoAndPlay("_show");
        }
    }

private function focusOutHandler(Ev:FocusEvent):void
{
    [...]
    if (Ev.currentTarget == wiadomoscText_txt)
    {
        wiadBG.gotoAndPlay("_hide");
    }
}

```

Najpierw zostaje zidentyfikowany obiekt inicjujący zdarzenie, następnie zostaje odtwarzany odpowiedni obiekt typu 'MovieClip' w klatce '_show' lub '_hide' co powoduje jego płynne ukazanie lub schowanie się.

2.5.2 Wysłanie wiadomości

Wiadomość zostaje wysłana po wciśnięciu przez użytkownika guzika 'wyślij'. Zdarzenie to jest przechwycone przez funkcję nasłuchującą 'sendButtonClickedHandler'.

```

private function sendButtonClickedHandler(Ev:MouseEvent):void
{
    var request:URLRequest= new URLRequest( phpMailScriptURL );
    var variables:URLVariables = new URLVariables();
    variables.od = odText_txt.text;
    variables.email = emailText_txt.text;
    variables.wiadomosc = wiadomoscText_txt.text;

    request.data = variables;
    request.method = URLRequestMethod.POST;

    sendToURL(request);
}

```

```
this.gotoAndPlay("success");  
}
```

Prywatna stała 'phpMailScriptURL' zawiera URL skryptu PHP odpowiedzialnego za wysyłanie wiadomości z serwera. Na ten adres zostają wysłane zmienne pozyskane z formularza oraz zawarte w obiekcie variables (obiekt typu URLVariables). Proces wysyłania i odbierania danych jest obsługiwany przez wykorzystanie wbudowanej w Flash funkcjonalności, czyli użycie obiektu typu URLRequest nastawionego na metodę 'POST'.

Po przekazaniu danych wiadomości do skryptu PHP, następuje wysłanie wiadomości.

```
function sendMail($from, $returnAddress, $content)  
{  
    $to = "info@plushaki.pl";  
    $headers.= "Content-Type:text/plain; charset=UTF-8\r\n";  
    $headers.= "From: $returnAddress";  
    $subject= "Wiadomosc ze strony plushaki.pl";  
    $message="\n  
OD: $from\n  
Email: $returnAddress\n  
Wiadomosc:  
-----  
$content  
";  
    return mail($to, $subject, $message, $headers)  
}  
[..]  
//wywołanie funkcji:  
echo sendMail($_POST['od'], $_POST['email'],$_POST['wiadomosc']);
```

Powyższa funkcja formatuje dane oraz wykorzystuje wbudowaną w PHP funkcję mail, która korzystając z ustawień serwera wysyła wiadomość.

2.6 Wykorzystanie swfobjec oraz swfaddress

2.6.1 SEO oraz deep linking



Większość aplikacji webowych odtwarzanych za pomocą Flash Playera posiada problemy z indeksowaniem przez wyszukiwarki internetowe. Stosując bibliotekę SwfObject można wykonać tzw. 'White Hat Cloaking', czyli użycie JavaScript by przesłonić indeksowalny kod HTML aplikacją Flash. W tej sytuacji analizowany jest klient wchodzący na stronę, jeśli nie posiada on zainstalowanej wtyczki Flash Player, lub włączonej obsługi JavaScript zostaje serwowana alternatywna treść, będąca kodem HTML. W tej sytuacji, każdorazowo wchodzący crawler wyszukiwarek 'zobaczy' kod HTML i poprawnie zindeksuje stronę. Natomiast jeśli tym agentem jest nowoczesna przeglądarka internetowa zostanie wyświetlona aplikacja Flash.

Dalszym rozwinięciem użyteczności aplikacji jest wykorzystanie biblioteki SWFAddress by stworzyć głębokie linkowanie ('deep linking'), czyli stworzenie unikalnego adresu URL dla poszczególnych podstron aplikacji. Podczas używania aplikacji, można zaobserwować zmiany na pasku adresu, działający guzik powrotu oraz nowe wpisy wchodzące do historii przeglądarki. SWFAddress to przede wszystkim biblioteka JavaScript ale również, klasy obsługujące zdarzenia po stronie aplikacji Flash. Synchronizacja stanów w SWFAddress odbywa się głównie poprzez wykorzystanie statycznych metod klasy 'ExternalInterface' wbudowanej w Flasha.

2.6.2 Sekwencja działania

GaiaFramework przejmuje całą kontrolę nad SWFAddress oraz SWFObject, poszerzając ją dodatkowo o wezwania URLRequest do adekwatnych podstron struktury, tak by udostępnić wewnątrz aplikacji Flash zawartość alternatywnej treści serwowanej przez SWFObject.

Dla przykładu przeanalizujmy co się dzieje gdy ludzki użytkownik wchodzi na podstronę

['http://mmediasc.com/plushaki/filozofia.php'](http://mmediasc.com/plushaki/filozofia.php)

1. W kodzie embedującym aplikację znajdziemy parametry informujące o stanie jaki ma ona przybrać. Tablica flashvars zawiera informacje o aktualnej gałęzi do wyświetlenia oraz adresie pliku XML zawierające dane o strukturze aplikacji. Tablica ta jest dostępna wewnątrz aplikacji podczas jej działania.

```
<script type="text/javascript">
var params = { [..] };
var flashvars = {
branch: "index/nav/filozofia",
```

```
siteXML: "xml/site.xml"
};
var attributes = {id:"flashcontent"};
swfobject.embedSWF("main.swf", "flashcontent", "100%", "100%", "9.0.124", "expressInstall.swf",
flashvars, params, attributes);
</script>
```

2. GaiaFramework pobiera informacje zawarte w 'site.xml', które zawierają specyficznie formatowane dane o całej strukturze aplikacji. Dane te są wykorzystywane do wyświetlenia konkretnej podstrony, zmiany paska adresu oraz zmiany tytułu strony.
3. Przed wyświetleniem zadanej podstrony (poprzez wykonanie metody 'transitionIn') zostaje rozdysponowana seria zdarzeń, między innymi 'afterGoto'. Dzięki temu zdarzeniu, nasłuchujące menu nawigacyjne może się nastawić na zaznaczenie konkretnej podstrony.
4. Po zdarzeniu 'onAfterPreload' zostaje wykonane żądanie URLRequest do strony, będącej reprezentacją SEO aktualnie załadowanej podstrony. GaiaFramework udostępnia zawartość elementu '<div id="copy"></div>' jako tablice Stringów w prywatnej właściwości 'copy' zdefiniowanej w klasie abstrakcyjnej 'AbstractPage', z której każda podstrona dziedziczy. Jest to doskonały sposób na synchronizację aplikacji z alternatywną treścią.
5. Podczas wykonywania wejściowej tranzycji, tablica 'copy' jest już wypełniona danymi z pliku SEO, tym samym można wykorzystać ją w aplikacji.

```
override public function transitionIn():void
{
    // load language specific HTML content:
    tbxWrapper.loadText(copy[Languages.currentLanguage]);
    [...]
    super.transitionIn();
    [...]
}
```

6. Jeśli zostanie wywołana statyczna metoda 'Gaia.api.goto(branchAddress:String)', spowoduje to wsteczne użycie SWFAddress i zmianę paska adresu oraz zmianę wyświetlanej gałęzi aplikacji. Zostają wykonane kroki 3,4,5 dla nowych wartości.

2.7 Mniejsze moduły

2.7.1 Scroller

Suwak boczny jest instancją klasy Scroller pakietu PruskiUtils. Pakiet ten jest zbiorem autonomicznych, łatwo transferowalnych klas, które można bez trudu używać w różnych projektach. Podczas ich projektowania miałem na uwadze w szczególności łatwość przenoszenia kodu.

Klasa Scroller jest alternatywą dla wbudowanego scrollera tekstu jak i poszerza jego funkcjonalność o możliwość 'scrollowania' wszystkich obiektów na scenie obiektu, ustawionego jako 'target'. Jego dodatkowe zalety to możliwość animowania procesu 'scrollowania'.

Sekwencja użycia scollera jest następująca:

1. Stworzyć obiekt docelowy ('target') poprzez dodanie w designerze lub programistycznie obiektów do jego sceny.
2. Stworzyć maskę na obiekcie 'target'- czyli przestrzeń aktualnie wyświetlaną.
3. Przypisać obiekt docelowy do scrollera oraz określić jego wysokość.

Po wykonaniu powyższej sekwencji, scroller będzie przesuwał obiekty znajdujące się na scenie obiektu 'target' w przestrzeni Y o wektor zbliżony do następującej proporcji:

$$\vec{V} = \frac{-(T_h - S_h) * S_y}{S_h}$$

V – wektor przesunięcia obiektów 'target'

S_y – przesunięcie scrollera od punktu początkowego na osi Y

S_h – wysokość scrollera (maxYdrop)

T_h – wysokość obiektu 'target'

Powyższy wektor reprezentuje sumę wektorów dodawanych w każdej klatce animacji aż do jej zakończenia.

Metoda inicjująca działanie 'scrollera' to 'setTarget', przed jej użyciem obiekt docelowy powinien być całkowicie przygotowany, tzn. jego wysokość powinna być stała (jeśli zmieni się wysokość, można użyć publicznej metody 'updateScroller').

```
public function setTarget(targetIN: MovieClip):void
{
    if (targetIN.height < this.maxYdrop) { this.visible = false; }

    target = targetIN;
    targetDisplayChildren = new Array();
    for (var i:uint = 0; i < target.numChildren; i++)
    {
        var obj:Object = new Object;
        obj.targetDO = target.getChildAt(i);
        obj.initialY = obj.targetDO.y;
        targetDisplayChildren.push(obj);
    }
    this.addEventListener(Event.ENTER_FRAME, scrollTargetToPositin, false, 0, true);
    target.addEventListener(MouseEvent.MOUSE_OVER, activateMouseScrolling, false, 0, true);
    target.addEventListener(MouseEvent.MOUSE_OUT, deactivateMouseScrolling, false, 0, true);
    activateMouseScrolling(null);
}
```

Metoda ta wpieryw sprawdza czy wysokość obiektu docelowego jest większa od maski, więc czy jest w ogóle wymagany scroller.

Następnie zostają kopiowane referencje do obiektów znajdujących się na scenie obiektu 'target' do użytku przy przesunięciach oraz zostają przypisane funkcje nasłuchujące, jak poniżej:

scrollTargetToPositin – funkcja inkrementująca przesunięcie, wywoływana na zdarzenie 'ENTER_FRAME'

activateMouseScrolling – funkcja włączająca reagowanie na scroller myszki, wywoływana na zdarzenie 'MOUSE_OVER'

deactivateMouseScrolling – funkcja dezaktywująca scroller myszki, wywołanie : 'MOUSE_OUT'

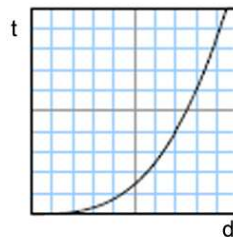
Przesunięcie zostaje wykonane w metodzie 'scrollTargetToPositin'

```

private function scrollTargetToPositin(Ev:Event):void
{
    var scrollerPercPoss:Number = Math.abs( this.y - initialY) / maxYdrop;
    var targetMaxDrop:Number = target.height - maxYdrop;
    var pxT0Scroll:Number = -targetMaxDrop * scrollerPercPoss;
    for (var i:uint = 0; i < targetDisplayChildren.length ; i++)
    {
        if ( Math.abs((pxT0Scroll + targetDisplayChildren[i].initialY) - targetDisplayChildren[i].targetDO.y) > 0.2)
        {
            targetDisplayChildren[i].targetDO.y -= (targetDisplayChildren[i].targetDO.y - (pxT0Scroll + targetDisplayChildren[i].initialY)) * .15;
        }
    }
}

```

Metoda ta w pierw kalkuluje wektor przesunięcia względem aktualnej pozycji i jeśli obiekty nie są jeszcze na pozycji docelowej przesuwa je o 15% tego wektora. Ponieważ metoda jest wykonywana każdorazowo przy wejściu do klatki (ENTER_FRAME) docelowy wektor jest pomniejszany o 15% względem wektora z poprzedniej klatki, daje to efekt animacji zbliżonej do następującego wykresu.



Rys. 6

2.7.2 *Dynamiczna galeria fade in-out*

Galeria ta to kolejny komponent z biblioteki 'PruskiUtils'. Główne założenia podczas jej tworzenia to:

1. Łatwo transferowalny kod
2. Sprawne zarządzanie pamięcią
3. Łatwość użycia

Przykład użycia, można znaleźć w klasie 'FirmowaPage'

```
[...]  
    gallery= new FadelnGallery(5000, 100, 412, 308);  
    gallery.loadFromXMLFile("firmowaGalleryItems.xml");  
    galleryContainer.addChild(gallery);  
[...]
```

W konstruktorze zostają zdefiniowane odpowiednio: czas wyświetlania zdjęcia [ms], czas tranzycji [ms], szerokość, wysokość. Następnie zostaje podany plik XML zawierający ścieżki do obrazków do wyświetlenia a ostatecznie obiekt galerii zostaje dodany do sceny kontenera.

Struktura XML pliku dla galerii powinna wyglądać następująco:

```
<imageList>  
    <image>jakosclmgs/ist2_13857730-tailor-with-scissors-close-up-hands.jpg</image>  
    <image>jakosclmgs/ist2_4171770-textile-mills.jpg</image>  
</imageList>
```

Galeria jest 'inteligentna' i najpierw ładuje i wyświetla zdjęcie będące pierwsze w pliku XML a następnie zaczyna ładować resztę. Przyspiesza to czas pojawienia się pierwszego zdjęcia poprzez dedykowanie mu całego transferu.

Tranzycja zdjęć zadana jest w prywatnej metodzie 'changePicture', która wykonywana jest w odstępach czasu zadanych w konstruktorze.

```
private function changePicture()  
{  
if (imageRefs_arr.length>0)  
{  
    //at least one picture is loaded  
    var hidePictureNum:Number=currentPic;  
    var showPictureNum:Number=(currentPic+1)%imageRefs_arr.length;  
    transformIntervallID= setInterval(showHidePicture,transSpeed,hidePictureNum,showPictureNum);  
    currentPic=showPictureNum;  
}  
if (changePicintervallID==0)  
    ChangePicintervallID=setInterval(changePicture, picDisplayLength);
```

```
}  
}
```

Jeżeli jest więcej niż jedna grafika załadowana to nastąpi rotacja. Wszystkie rotacje wykonywane są za pomocą manipulowania funkcją 'setInterval' i polegają na stosownym zwiększaniu lub zmniejszaniu właściwości 'alpha' wyświetlanych bitmap.

Po zakończeniu używania galerii, powinna zostać wywołana metoda wymagana przez interfejs 'iDestroyable' zwalnająca pamięć (tzn. uwalniająca obiekty by GC mógł je zniszczyć)

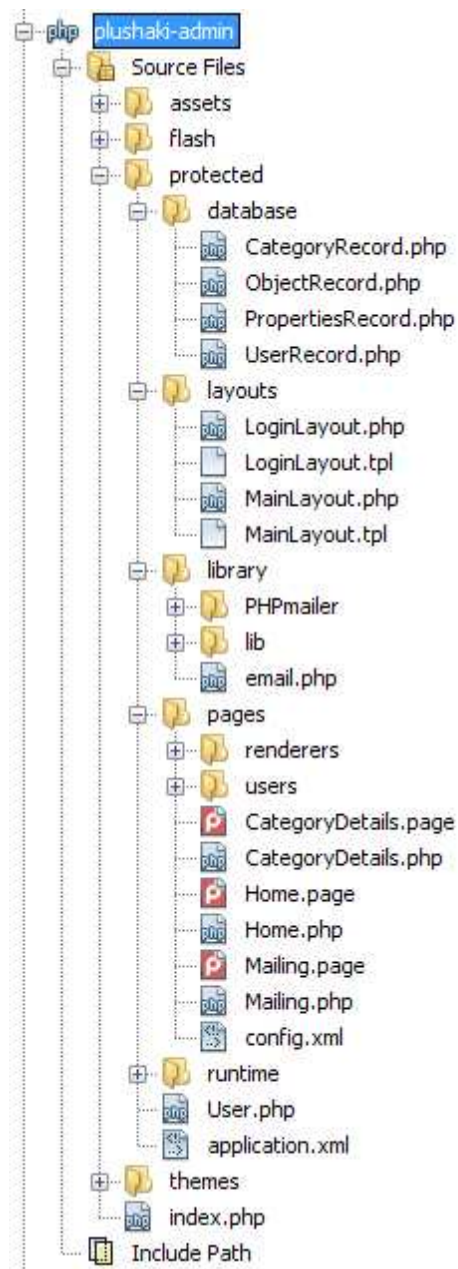
```
public function destroy():void  
{  
    clearInterval(changePicintervalID);  
    clearInterval(transformIntervalID);  
    imagePathList_arr=null;  
    imageRefs_arr=null;  
    (parent as MovieClip).removeChild(this);  
}
```

3 Prace programistyczne back-end

3.1 Struktura projektu

Struktura projektu bazuje na modelu MVC (Model – View – Controller). Gdzie:

- Segmenty mapowania bazy danych (Model) znajdują się w katalogu 'protected/database/'
- Szablony widoków (View) to pliki *.tpl. 'protected/layouts/' dla szablonów głównych oraz 'protected/pages/' dla widoków podstron
- Kod wykonawczy (Controllers) to pliki *.php których nazwy są takie same jak zawartych w nich klas oraz odpowiadających im widoków.



Rys. 7

Ponadto w katalogu 'protected/pages/renderers/' znajdują się pliki pomocnicze do wykonywania konkretnych zadań.

PradoFramework wykorzystuje plik konfiguracyjny 'application.xml' by inicjować swoje moduły.

```
<?xml version="1.0" encoding="utf-8"?>
<application id="plushaki" mode="Debug">
  <services>
    <service id="page" class="TPageService" DefaultPage="Home">
```

```

    <pages MasterClass="Application.layouts.MainLayout" Theme="mMediaCukierek"/>
  </service>
</services>
<paths>
  <using namespace="Application.database.*" />
  <using namespace="Application.library.*" />
  <using namespace="System.I18N.*" />
</paths>
<modules>
<module id="db" class="System.Data.TDataSourceConfig">
  <database ConnectionString="mysql:host=remote-mysql4.servage.net; dbname=plushaki-remote"
    Username="plushaki-remote" Password="Plushaki1234" />
</module>
<module class="System.Data.ActiveRecord.TActiveRecordConfig" ConnectionID="db" />
[...]
```

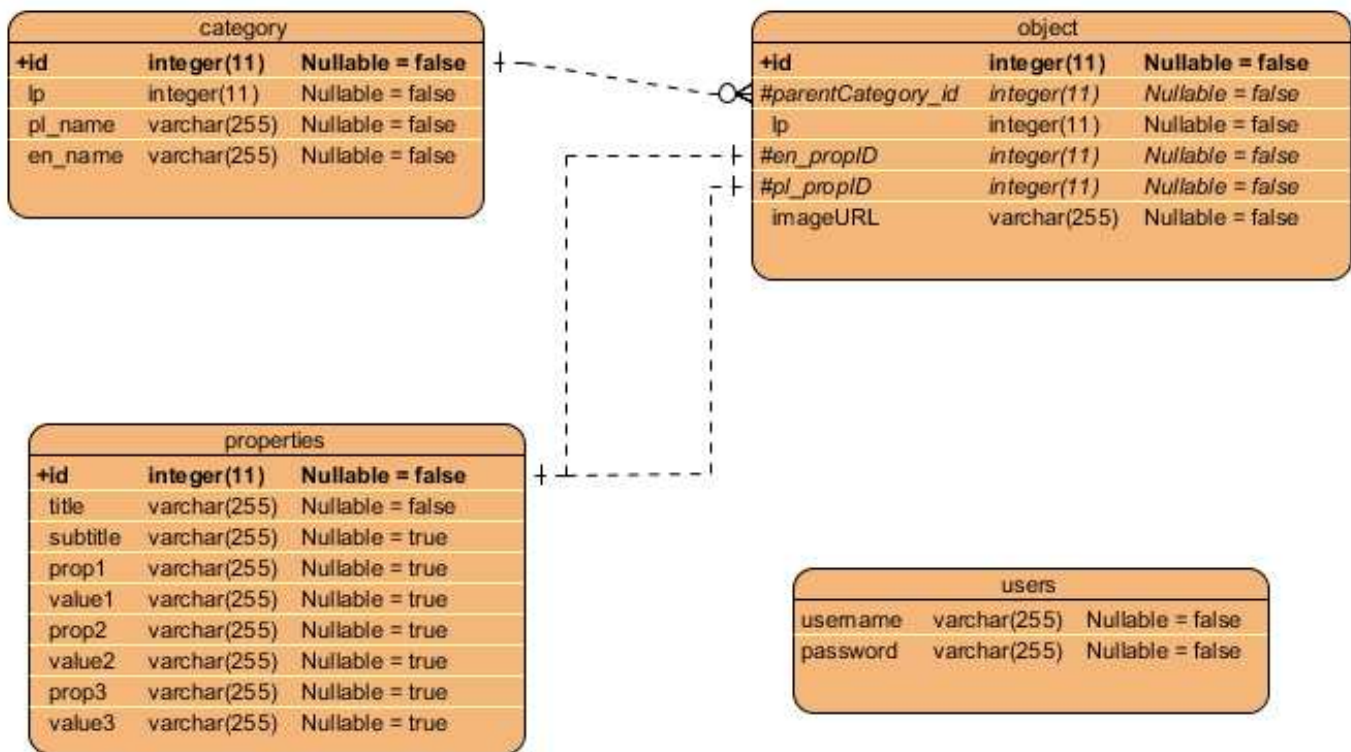
```

<parameters>
  <parameter id="galleryDir" value="objectImages/" />
  <parameter id="galleryObjWidth" value="412" />
  <parameter id="galleryObjHeight" value="308" />
</parameters>
</application>

```

Zawiera on informacje o podstawowych ustawieniach aplikacji, konfiguracje używanych modułów, dane dostępu do bazy danych oraz globalne parametry, które są dostępne podczas rozruchu aplikacji.

3.2 Struktura bazy danych



Rys. 8

Baza danych posiada bardzo prostą strukturę umożliwiającą kontrolę nad podstroną 'kolekcja' oraz trzyma dane autoryzacji dostępu do panelu CMS.

Opis elementów bazy:

- Tabela 'users' zawiera parę 'username' oraz 'password' (hashowany przez md5), które są konfrontowane z danymi podawanymi przez użytkownika podczas procesu logowania do systemu.
- Tabela 'category' reprezentuje kategorie kolekcji. Może być nieskończenie wiele kategorii.
- Tabela 'object' reprezentuje obiekty konkretnej kategorii. Kategoria, może mieć nieskończenie wiele obiektów a połączenie z kategorią będącą właścicielem obiektu trzymane jest w polu 'parentCategory_id'. Obiekt posiada również właściwości, reprezentowane przez wpis do tabeli 'properties', po jednym wierszu dla każdej z wersji językowych ('en_propId' oraz 'pl_propId').

3.3 Autentykacja oraz zabezpieczenia

Zabezpieczenia obsługiwane są przez PradoFramework poprzez wykorzystanie dwóch blisko współpracujących modułów 'TAuthManager' oraz 'TDbUserManager'. Ich dane inicjujące znajdują się w pliku 'application.xml':

```
<modules>
[.]
  <module id="auth" class="System.Security.TAuthManager" UserManager="users" LoginPage="users.LoginUser" />
  <module id="users" class="System.Security.TDbUserManager" UserClass="Application.User" />
[.]
</modules>
```

Pierwsza linijka inicjuje TAuthManager, informując go, że ma sprawdzać tożsamość osób odwiedzających stronę i wskazuje miejsce logowania. Namespace 'users.LoginUser' wskazuje na klasę 'LoginUser' zawartą w plikach 'LoginUser.page'(widok) oraz 'LoginUser.php'(kontroler) znajdujące się w katalogu 'protected/pages/users/'. Parametr 'UserManager' wskazuje na moduł użytkowników do obsługi autentykacji, który jest zdefiniowany w drugiej linijce.

By przeprowadzić autentykację z bazą danych używając klasy wbudowanej w Prado 'TDbUserManager' wymagane jest zdefiniowanie klasy użytkownika dziedziczącej z klasy 'TDbUser'. Służy temu właściwość 'UserClass' i w tym przypadku wskazuje ona na klasę 'User' znajdującą się w głównym katalogu 'protected'.

Gdy wszystko jest ustawione, można przeprowadzić proces walidacyjny. W szablonie (widoku) 'LoginUser.page' zostaje rejestrowany 'custom validator'

```
[...]
<com:TCustomValidator ControlToValidate="Password" ErrorMessage="Wprowadzono błędne hasło."
  Display="Dynamic"
  OnServerValidate="validateUser" />
[...]
```

którego kod wykonawczy znajduje się w metodzie 'validateUser' klasy LoginUser.

```
public function validateUser($sender, $param)
```

```

    $authManager=$this->Application->getModule('auth');
    if(!$authManager->login($this->Username->Text, md5($this->Password->Text)) )
    {
        $param->IsValid= false;
    }
}

```

W metodzie tej użyty jest moduł autentykacji oraz wcześniej zdefiniowana metoda walidacji znajdującą się w klasie 'User'

```

public function validateUser($username,$password)
{
    return (UserRecord::finder()->findBy_username_AND_password($username,$password) !== null);
}

```

W metodzie tej z kolei użyta jest klasa UserRecord, która mapuje tabele użytkowników (users) z bazy danych i dostarcza wpisy pasujące do zapytania. Jeśli w tym miejscu nie zostanie odnaleziony wiersz zawierający podaną parę 'username' oraz 'password' zwracana jest wartość negatywna a proces autentykacji kończy się niepowodzeniem.

3.4 Edytor Kolekcji

System pozwala na dodawanie, edytowanie, usuwanie oraz zmianę kolejności wyświetlania kategorii jak i obiektów zawartych w kategoriach. Istnieje oczywiście, możliwość usuwania tychże obiektów, przy czym usuwając kategorie następuje rekursywnie i bezpowrotne usunięcie wszystkich sub obiektów (czyli obiektów kategorii, plików graficznych, oraz właściwości).

Będąc w głównym widoku kolekcji można przejść do widoku sub obiektów poprzez kliknięcie w nazwę wybranej kategorii .



Katalog Mailing Wyloguj **Górne Menu**

Dodawanie Kategorii

PL: EN:

Kategorie:

LP	Nazwa PL	Nazwa EN	Opcje
1	Zwykłe	Regular	<input type="button" value="Usuń"/> <input type="button" value="Edytuj"/>
2	Reklamowe	Commercial	<input type="button" value="Usuń"/> <input type="button" value="Edytuj"/>
3	Funkcjonalne	Functional	<input type="button" value="Usuń"/> <input type="button" value="Edytuj"/>

CMS Created by Piotr Pruski

Opcje elementu

Rys. 9

3.4.1 CategoryDetails

CategoryDetails to klasa odpowiedzialna za operacje na elementach konkretnej kategorii. Umożliwia listowanie, edytowanie, dodawanie oraz usuwanie elementów. Definicja klasy (Controller) znajduje się w 'protected/pages/CategoryDetails.php' natomiast jej szablon (View) w 'protected/pages/CategoryDetails.page'.

Listowanie elementów odbywa się poprzez wykorzystanie komponentu TRepeater oraz klasy pomocniczej ('ObjectElement') wraz z jej szablonem, które wspólnie definiują wygląd i zachowanie poszczególnych elementów.

Zdefiniowanie obiektu TRepeater odbywa się w szablonie klasy CategoryDetails (w 'CategoryDetails.page')

```
<com:TRepeater ID="ObjectsRepeater" ItemRenderer="Application.pages.renderers.ObjectElement">  
  <prop:HeaderTemplate>  
    <ul class="objectList">
```

```

    </prop:HeaderTemplate>
    <prop:FooterTemplate>
      </ul>
    </prop:FooterTemplate>
  </com:TRepeater>

```

Jak widać powyżej, wszystkie elementy są zamykane w znacznikach 'ul', czyli nienumerowanej listy.

W szablonie klasy 'ObjectElement' można zauważyć, iż każdy element składa się z dwóch paneli.

```

[.]
<com:TPanel ID="View" Visible='true'>
    [.] // tu znajdują się renderowane elementy dla stanu zwykłego
</com:TPanel>
<com:TPanel ID="Edit" Visible='false'>
    [.] // tu znajdują się renderowane elementy dla stanu zwykłego
</com:TPanel>
[.]

```

W zależności od stanu elementu ('view' lub 'edit') wyświetlany jest odpowiedni panel oferujący adekwatną funkcjonalność.

Ponieważ aplikacja front-end wymaga bardzo specyficznego formatu załadowanego obrazka, każdy plik powiązany z obiektem jest ściśle monitorowany. Jeżeli nie spełnia wymagań, wyraźna informacja o zaistniałym problemie jest wyświetlana użytkownikowi. Metoda analizująca plik to 'getImageErrorsArr'

```

private function getImageErrorsArr($imgUrl)
{
    $info = getimagesize($imgUrl);

    $errorArr = array();
    if(image_type_to_mime_type($info[2]) != "image/png")
    {
        $errorArr[] = "Załadowany plik nie jest poprawnym formatem PNG";
    }
    //check if the file has a transparency channel

```

```

else if( ((ord ( file_get_contents ($imgUrl, false, null, 25, 1) ) & 6) & 4) != 4)
{
    $errorArr[] = "Wydaje się, że załadowany plik PNG nie posiada informacji kanału Alpha.";
}

$errorMargin=2;
$targetWidth=$this->Application->Parameters['galleryObjWidth'];
$targetHeight= $this->Application->Parameters['galleryObjHeight'];
if(
    ($info[0]< ($targetWidth-$errorMargin)) || ($info[0]> ($targetWidth+$errorMargin))
    ||
    ($info[1]< ($targetHeight-$errorMargin)) || ($info[1]> ($targetHeight+$errorMargin))
)
{
    $errorArr[] = "Zalecane wymiary obrazka to: ".$targetWidth."x".$targetHeight.". Natomiast
załadowany obrazek ma wymiary: ".$info[0]."x".$info[1].".";
}

return $errorArr;
}

```

Powyższa metoda zwraca tablice błędów, jeśli tablica jest pusta znaczy to, że plik jest w poprawnym formacie. Badane są następujące rzeczy:

1. Czy jest to plik PNG
2. Czy plik PNG posiada informacje o kanale alpha (czy jest to PNG-24 lub PNG-32)
3. Czy wymiary pliku w px. są odpowiednie.

Dodawanie nowych obiektów do kategorii odbywa się poprzez użycie formularza znajdującego się poniżej aktualnej listy elementów. Formularz posiada zaimplementowane walidatory na polach nazwy oraz zdjęcia- co sprawia, że pola te są obowiązkowe.

Dodaj Nowy Obiekt

	Polski	English
Nazwa:	<input type="text"/>	<input type="text"/>
Opis:	<input type="text"/>	<input type="text"/>
Wartosc 1:	<input type="text"/>	<input type="text"/>
Wartosc 2:	<input type="text"/>	<input type="text"/>
Wartosc 3:	<input type="text"/>	<input type="text"/>
Załaduj Plik:	<input type="text"/> <input type="button" value="Browse_"/>	<input type="text"/>

*dopuszczalne są tylko pliki .PNG 24 lub 32-bit

Rys. 10

Tworzenie nowego obiektu odbywa się w publicznej metodzie 'AddObject' klasy 'CategoryDetails' i sprowadza się do walidacji formularza, załadowania pliku graficznego oraz przypisania obiektom mapującym bazy danych odpowiednich wartości.

```
public function AddObject($sender, $param)
{
[.]
$object= new ObjectRecord();
$object->lp = (ObjectRecord::finder()->count('parentCategory_id='.$this->Request['selectedCategory'])
+1 );
$object->parentCategory_id = $this->Request['selectedCategory'];
$object->pl_propID = $pl_properties->id;
$object->en_propID = $en_properties->id;
$object->imageUrl = $this->uploadNewImage(false);

$object->save();
$this->Response->reload();
}
```

```
}
```

Załadowanie pliku graficznego następuje w metodzie 'uploadNewImage' wywoływanej z metody 'addObject'

```
public function uploadNewImage($resize=true)
{
    if(!$this->NewImage->HasFile)
        return;

    $dir = "../".$this->Application->Parameters['galleryDir'];

    $fileName = md5(time()).".png";

    if($resize)
    {
        $this->NewImage->saveAs($dir.$fileName."_TMP", true);
        $width= $this->Application->Parameters['galleryObjWidth'];
        $height = $this->Application->Parameters['galleryObjHeight'];
        $this->savelImage($dir.$fileName."_TMP", $dir.$fileName, $width, $height);
    }
    else
    {
        $this->NewImage->saveAs($dir.$fileName, true);
    }
    return $this->Application->Parameters['galleryDir'].$fileName;
}
```

Celem uniknięcia nadpisowań plików, ich nazwy zmieniane są na wynik funkcji haszującej md5 od aktualnego czasu. Pliki zapisywane są w katalogu wyznaczonym przez parametry aplikacji (znajdujące się w pliku 'application.xml') przechowywane w tabeli parameters (\$this->Application->Parameters['galleryDir'])

Ponadto, celem uzyskania najwyższej jakości zdjęć, zalecane jest wywoływanie powyższej metody z ustawieniem flagi 'resize'=false, ustawienie rozmiaru pliku graficznego powinno odbyć się przy użyciu specjalistycznego oprogramowania (np. Adobe Photoshop).

3.5 Moduł mailingu

Po zalogowaniu do panelu administracyjnego zostaje udostępniony moduł pozwalający na wykonywanie masowego mailingu wiadomości HTML. Uprawniony użytkownik, powinien wypełnić następujące pola:

- Temat wiadomości
- Lista odbiorców – każdy email oddzielony średnikiem
- Treść wiadomości HTML – Szablon jest gotowy do użytku i zawiera logo oraz inne grafiki, które zostaną dołączone do wiadomości. Do dyspozycji użytkownika oddany jest również edytor typu WYSIWYG, który pozwala formatować pisany tekst za pomocą JavaScript.
- Treść alternatywna – Niestety nie wszystkie aplikacje obsługujące pocztę są w stanie wyświetlić treść HTML, więc zalecane jest by użytkownik podał wiadomość alternatywną, która zostanie wyświetlona jako nie formatowany tekst.

Moduł wykorzystuje bibliotekę PHPmailer (znajdącą się w katalogu 'admin/protected/library/PHPmailer') do obsługi procesu wysyłania oraz używa zewnętrznego serwera poczty- w udostępnionej wersji testowej jest to 'gmail.com' .

3.5.1 Szablon wiadomości HTML

Edytor WYSIWYG użyty w projekcie to TinyMCE, zostaje on załadowany przez PradoFramework w szablonie podstrony 'Mailing' a jego właściwość ID to 'bodyHTML' . Natomiast sam szablon wiadomości HTML zostaje aplikowany w prywatnej metodzie 'initTemplate()' klasy 'Mailing' ('protected/pages/Mailing.php'), która zostaje wywołana w metodzie obsługującej zdarzenie 'onLoad'.

```
private function initTemplate()  
{  
    if(strlen($this->bodyHTML->text)==0)
```

```

        $body="
<table width='600px' align='center'>
<tr>
    <td align='center'>
        <img src='http://www.plushaki.pl/maillingImgs/logo.jpg' border='0' width='118' height='72' />
    </td>
</tr>
[... ]
</table>
";
        $this->bodyHTML->text= $body;
    }
}

```

Powyższa metoda w pierw sprawdza, czy element 'bodyHTML' zawiera już jakieś znaki – w przypadku gdy zawiera, znaczy to, że mamy do czynienia z sytuacją, w której formularz albo nie przeszedł pomyślnie walidacji albo wiadomość już została wysłana – szablon nie zostanie zaaplikowany.

Jeżeli zostanie stwierdzone, że wymagana jest aplikacja szablonu, odpowiednio formatowany kod HTML zostaje przypisany do właściwości 'text' elementu 'bodyHTML'. Ważne elementy konstrukcji kodu HTML dla wiadomości:

- Warto zrezygnować z użycia stylów CSS, nawet w trybie 'inline' – nie wszystkie programy klientów poczty są w stanie je poprawnie wyświetlić.
- Warto trzymać się najprostszych konstrukcji HTML- rozmieszczając treść przy użyciu Tabel oraz manipulować tekstem przy użyciu znaczników .
- Serwer przechowujący grafiki musi mieć włączony tryb 'hot-linking' celem poprawnego linkowania plików.

3.5.2 Wysłanie wiadomości

Po wypełnieniu przez użytkownika formularza wiadomości i wciśnięciu guzika 'wyślij' aplikacja wchodzi w metodę 'sendEmailClickedHandler' klasy 'Mailing' ('protected/pages/Mailing.php'). Gdzie w pierw wykonywana jest walidacja poprawności wypełnienia formularza i w pozytywnym przypadku

zostają pobrane oraz przygotowane dane potrzebne do wykonania procesu wysyłki:

```
public function sendEmailClickedHandler($sender, $param)
{
[...]
```

```
    $title= $this->subject->text;
    $messageHtml=$this->bodyHTML->text;
    $messageText= $this->bodyAlternative->text;

    $receptients= explode(";", $this->receptients->Text);
    foreach ($receptients as $rec)
    {
        $rec = trim($rec);
    }
    $this->sedMassMail($title,$messageHtml,$messageText, $receptients);
[...]
```

```
}
```

Prywatna metoda 'sendMassMail' odpowiedzialna jest za wykonanie dalszych czynności przygotowania danych oraz wysyłki poprzez połączenie się z serwerem SMTP zewnętrznego klienta- w omawianym przypadku jest to 'gmail.com'.

```
private function sedMassMail($title, $messageHtml, $altMessage, $receptients)
{
    $mail= new PHPMailer();
    $mail->Mailer = "smtp";
    $mail->Host = "ssl://smtp.gmail.com";
    $mail->Port = 465;
    $mail->SMTPAuth = true;
    $mail->Username = "biuro.plushaki@gmail.com";
    $mail->Password = "HasloPlushaki";

    $body      = $messageHtml;
    $body      = eregi_replace("[\]", "", $body);

    $mail->CharSet='UTF-8';

    $mail->AddReplyTo("biuro.plushaki@gmail.com", "Biuro Plushaki.pl");
```

```

$mail->SetFrom("biuro.plushaki@gmail.com","Biuro Plushaki.pl");

foreach($reipients as $rec)
{
    $mail->AddBCC($rec);
}

$mail->Subject = $title;
$mail->AltBody = $altMessage;
$mail->MsgHTML($body);

if(!$mail->Send())
    { $this->errorMesg->text= "Błąd w wysłaniu ".$mail->ErrorInfo;}
else
    {$this->errorMesg->text= "Wiadomosc wysłana!";}
}

```

3.5.3 *Test wiadomości*

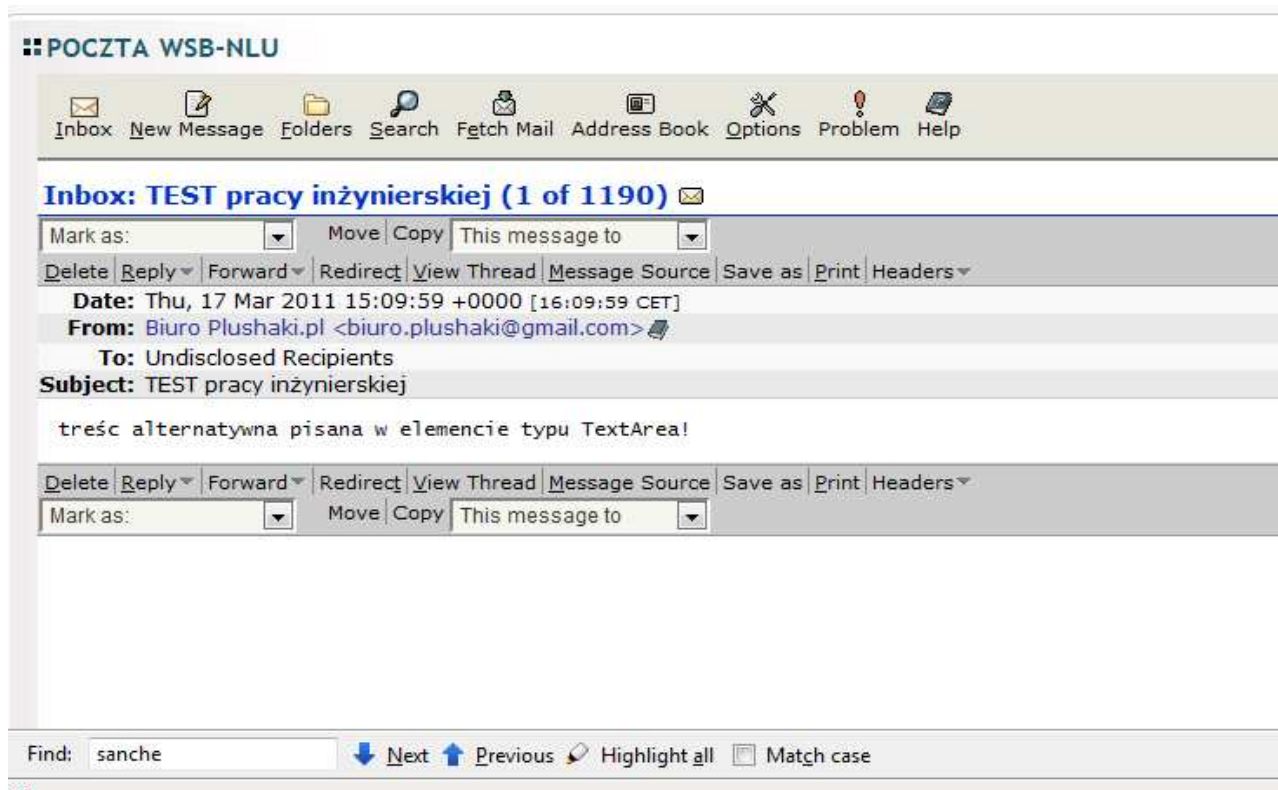
Poniżej znajdują się zrzuty ekranu dla wiadomości wysłanej do dwóch odbiorców, z których jeden obsługuje wiadomości HTML (gmail.com) a drugi jedynie tekstowe(wsb-nlu.edu.pl).

Rezultat dla wiadomości HTML:



Rys. 11

Rezultat dla wiadomości alternatywnej:



Rys. 12

4 Podsumowanie

Jestem przekonany, iż czytelnik jest już w stanie docenić projekt Plushaki.pl jako znacznie więcej niż tylko 'strona internetowa' lecz jako aplikacja webowa, której funkcjonalność i estetyka wynoszą mojego klienta ponad poczynania konkurencji.

Przed wszystkim, warto skupić się na nie często spotykanym użyciu SWFObject jako podstawy połączenia z systemem CMS oraz systemu pozycjonującego w wyszukiwarkach internetowych. Za kolejne rozwiązania warte uwagi, uważam specyficzną galerię produktów płynnie połączoną z systemem zarządzania, autorską bibliotekę 'PruskiUtils' oraz system wysyłania wiadomości mailowych bazujących na HTMLu.

Jestem szczególnie zadowolony z przejrzystej struktury plików projektu. Stosujących ogólnie znane 'design patterns' oraz pisząc kod w sposób łatwy do przeniesienia i ponownego użycia, jestem w stanie wyciągać poszczególne moduły pod kątem użycia w innych projektach.

5 Źródła

5.1 Bibliografia

1. Dokumentacja PRADO framework - <http://www.pradosoft.com/docs/classdoc/>
2. Dokumentacja GAIA framework - <http://www.gaiaflashframework.com/wiki/>
3. Dokumentacja PHP - <http://php.net/docs.php>
4. Dokumentacja ActionScript 3.0 : <http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/>
5. „Design Patterns: Elements of Reusable Object-Oriented Software” Erich Gamma, Richard Helm, Ralph Johanson, John Vlissides ISBN 978-02-016-3361-0
6. „Flash CS3 Professional PL. Techniki zaawansowane. Klatka po klatce” Russel Chun ISBN: 978-83-246-1342-7
7. „ActionScript 3.0. Biblia” Roger Braunstein, Mims H. Wright, Joshua J. Noble ISBN: 978-83-246-1939-9
8. „PHP. Zaawansowane programowanie. Vademecum profesjonalisty” George Schlossnagle ISBN: 83-7361-589-X

5.2 Spis rysunków

Rys. 1 – Logotyp Plushaki.pl. Źródło: opracowanie własne

Rys. 2 – Wykresy algorytmów animacji biblioteki TweenLite. Źródło:
<http://janitor61.com/easingReference/gs.easing.reference.pdf>

Rys. 3 – Warstwy aplikacji plushaki.pl . Źródło: opracowanie własne

Rys. 4 – Elementy obiektów klasy 'ObjectPresentation'. Źródło: opracowanie własne

Rys. 5 – Schemat logiki działania obiektów klasy 'ObjectPresentation'. Źródło: opracowanie własne

Rys. 6 – Wykres animacji przesunięcia potomstwa obiektów zarejestrowanych w obiektach klasy 'Scroller' . Źródło: opracowanie własne

Rys. 7 – Struktura plików aplikacji zarządzania treścią plushaki.pl. Źródło: opracowanie własne

Rys. 8 – Diagram ERD bazy danych. Źródło: opracowanie własne

Rys. 9 – Opis GUI modułu zarządzania kategoriami. Źródło: opracowanie własne

Rys. 10 – Panel dodawania nowych obiektów do galerii. Źródło: opracowanie własne

Rys. 11 – Widok odebranej wiadomości HTML wysłanej za pomocą modułu mailingu. Źródło: opracowanie własne

Rys. 12 – Widok odebranej wiadomości alternatywnej wysłanej za pomocą modułu mailingu..
Źródło: opracowanie własne