

Mini CMS

z wykorzystaniem Flex'a

Cezary Tomczak

Aby rozpocząć tłumaczenie przykładu użytego w tutorialu należy wyjaśnić co to jest CMS. Otóż z angielskiego (Content Management System) - system zarządzania treścią jest to jedna lub zestaw aplikacji internetowych pozwalających na łatwe utworzenie oraz późniejszą aktualizację i rozbudowę serwisu WWW przez redakcyjny personel nietechniczny. Kształtowanie treści i sposobu ich prezentacji w serwisie zarządzanym poprzez CMS odbywa się za pomocą prostych w obsłudze interfejsów użytkownika, zazwyczaj w postaci stron WWW zawierających rozbudowane formularze i moduły.

Stworzyłem mało skomplikowany wzór miniCMS'a opartego na komponencie Flex'a „RichTextEditor”. Wszystko aby wskazać odpowiednią drogę do tworzenia CMS'ów.

Do rozpoczęcia tutoriala będziemy potrzebować:

- Flex Builder albo (SDK2 + FlashDevelop),
- AMFPHP 1.9,
- serwer HTTP wraz z PHP oraz bazą danych MySQL.

Opis elementów

Aby funkcjonalność naszego miniCMS'a mogła zaistnieć potrzebujemy trzech elementów:

1. Zawartości (content) wyświetlanej na stronie www.
2. Panelu miniCMS do modyfikacji zawartości (content).
3. Bazy danych, w której zapisywać będziemy naszą zawartość (content).

ad 1.

Opis:

Utworzmy proste GUI do wyświetlania informacji tekstowej czytającej znaczniki html'owe.

Technologia:

GUI utworzymy za pomocą Flex'a korzystając z standardowych komponentów. Wyświetlana zawartość tekstowa będzie zapisana w bazie danych MySQL.

ad 2.

Opis:

Utworzmy przyjazny UI (możliwość standardowego Rich Text Editor) do edycji informacji tekstowej wraz z możliwością zapisania naniesionych poprawek.

Technologia:

UI utworzymy za pomocą Flex'a korzystając z komponentu „RichTextEditor”. Dostępne będą także dwa przyciski, jeden do zapisywania danych w bazie MySQL, drugi do czyszczenia aktualnie wyświetlonego tekstu. Obecnie, dostęp do panelu miniCMS nie będzie wymagał autoryzacji, jedynie wejścia pod odpowiedni adres internetowy. Proszę pamiętać aby w przyszłości wykonując jakiegokolwiek CMS'y stworzyć system logowania.

ad 3.

Opis:

Baza danych, miejsce w którym przechowywać będziemy całą zawartość tekstową.

Technologia:

Baza danych oparta na MySQL, nazwiemy ją „site_content”. Aby ułatwić pracę proponuję zainstalować panel phpMyAdmin.

Przygotowanie środowiska

Przed wszystkim będziemy potrzebować warunków jakie oferuje nam serwer HTTP. Można przeprowadzić instalację Apache wraz z PHP oraz bazą MySQL, jednak to rozwiązanie wymusza na użytkownika dokonania wielu zmian konfiguracyjnych. Dla początkujących proponuję zainstalować jeden z pakietów oferujących skonfigurowane oraz przygotowane środowisko serwera HTTP, np. WAMP (<http://www.wampserver.com/en/download.php>).

Na tym etapie najistotniejsze jest znać najważniejsze pliki konfiguracyjne:

- [config.inc.php](#) (dla phpMyAdmin - user:pass),
- [httpd.conf](#) (dla Apache - integracja z PHP),
- [php.ini](#) (dla PHP).

Chcąc sprawdzić poprawność zainstalowanego pakietu, proszę wpisać w przeglądarce 'http://localhost'. Powinna wyświetlić się standardowa strona pakietu bądź Apache'a.

Kolejnym krokiem jest instalacja AMFPHP. Ściągnąć go możemy z strony: <http://amfphp.org/>

Przed skopiowaniem jego zawartości proszę przygotować odrębny katalog pośród aplikacji web'owych naszego localhost. Często pojawiającą się nazwą katalogu jest 'FLASHSERVICES'. Tam proszę rozpakować całą zawartość zip'a AMFPHP.

Aby sprawdzić czy program działa prawidłowo, proszę wpisać w przeglądarce adres: 'http://localhost/flashservices/browser'. Jeśli instalacja przebiegła pomyślnie powinien pojawić się panel wyświetlający dostępne usługi AMFPHP.

Przygotowanie bazy danych

Na wstępie potrzebujemy utworzyć bazę danych, w której zapisana będzie nasza zawartość strony www (content).

Bazę tworzymy w MySQL. Proponuję korzystać z programu phpMyAdmin.

Bazę tą nazwiemy 'site_content'.

Tabela o nazwie 'content' zawierająca dwa pola:

- id (liczba porządkowa),
- text (pole przechowujące zawartość, typu LONGTEXT).

Zapytanie SQL - utworzenie bazy:

```
CREATE DATABASE site_content;
```

Zapytanie SQL - utworzenie tabeli:

```
CREATE TABLE `content` (  
    `id` int(11) NOT NULL auto_increment,  
    `text` longtext default NULL,  
    PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Następnie należy wypełnić pola przykładową zawartością - treść wygenerowałem za pomocą Lorem-Ipsum (<http://www.lipsum.com/>).

Zapytanie SQL - wypełnienie pól tabeli:

```
INSERT INTO `site_content`.`content` (  
    `id` ,  
    `text`  
)  
VALUES (  
    '1', 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Mauris et justo non leo fringilla vehicula. Mauris sapien  
libero, aliquam in, ullamcorper ac, pellentesque ut, diam.  
Etiam feugiat nunc in purus. Ut sem ligula, egestas nec,  
tristique a, hendrerit ac, lacus. Cras id lorem. Aliquam non  
ante. Sed tempor, est sit amet scelerisque volutpat, leo eros  
ullamcorper metus, ullamcorper lobortis sem velit tempus ante.  
Vestibulum tincidunt urna sit amet ante. Sed venenatis nisl non  
pede malesuada commodo. Nullam interdum metus sit amet ligula.  
Class aptent taciti sociosqu ad litora torquent per conubia  
nostra, per inceptos hymenaeos. Nam tempus dui nec leo.'  
);
```

*Dla zachowania zasad bezpieczeństwa powinno się nadać prawa konkretnego użytkownika do korzystania z bazy. Ta opcja nie została wykorzystana w tutorialu.

Utworzenie usług AMFPHP

Aby możliwa była komunikacja pomiędzy aplikacjami flash'owymi, a bazą danych potrzebować będziemy tzw. bramy (gateway) oraz konkretnej usługi AMFPHP. W przykładzie miniCMS'a powinna być to usługa zapewniająca odczytanie konkretnego pola z bazy danych oraz przesłanie jego zawartości do aplikacji flash'owej, bądź panelu miniCMS'a. Potrzebujemy także usługi, która pobierze dane z panelu miniCMS'a, a następnie zapisze je w bazie danych.

Usługą odpowiedzialną za odczyt danych z bazy, będzie plik 'ReadContent.php', natomiast za zapis - 'WriteContent.php'.

Obie usługi utworzymy w katalogu usług AMFPHP - '/flashservices/services/AdobersTutorial/'.

// Listing ReadContent.php

```
<?php

/**
 * Return text content from database
 * @author FAKETA
 **/

class ReadContent {

    // Metoda odpowiedzialna za połączenie z bazą danych oraz
    // odczytanie jej zawartości

    function ReadContent () {

        // Ustawienia dla połączenia
        $host = 'localhost'; // nazwa hosta
        $user = 'root';      // nazwa użytkownika dla
        // połączenia z bazą danych
        $pass = '';         // hasło

        // Połączenie z bazą
        $dbh = mysql_connect ( $host, $user, $pass )
                or die('Cannot connect:
        ` . mysql_error());

        // Wybór bazy site_content
        mysql_select_db ( 'site_content' ) or die (
        'Cannot select database' );

        // Zapytanie - wybranie zawartości pola text z
        // tabeli content
        $query = 'SELECT text FROM `content` LIMIT 0, 30
        `;
        $result = mysql_query($query);
```

```
        // Wylistowanie zawartości
        $content = mysql_fetch_row ($result);

        // Zwrot zawartości bazy do aplikacji flash`owej
        return $content;

        //dodac rozłączenie z baza
    }

}
?>
```

// Listing WriteContent.php

```
<?php

/**
 * Write text content to database
 * @author FAKETA
 **/

class WriteContent {

    // Metoda odpowiedzialna za połączenie z bazą danych oraz
    // nadpisanie jej zawartości
    // @param changes - zmienna podawana z aplikacji
    // flashowej

    function WriteContent ( $changes ) {
        // Ustawienia dla połączenia
        $host = 'localhost';
        $user = 'root';
        $pass = '';

        // Połączenie z baza
        $dbh = mysql_connect ( $host, $user, $pass )
                or die('Cannot connect:
        ` . mysql_error());

        // Wybór bazy site_content
        mysql_select_db ( 'site_content' ) or die (
        'Cannot select database' );

        // Zapytanie - nadpisanie zawartości pola text
        // zmienną $changes
        $st = addslashes($changes);
        $query = sprintf(„UPDATE content SET text = (%s')
        WHERE id=(%s)”, $st);

        $result = mysql_query($query);
    }

}
?>
```

Utworzenie strony www

Przed przystąpieniem do przygotowania strony należy poruszyć aspekt remotingu we Flex'ie. Aby możliwe było wykorzystanie AMFPHP, Flex potrzebuje wskazania konkretnej konfiguracji usług. Standardowo odpowiedzialny jest za to plik 'services-config.xml'.

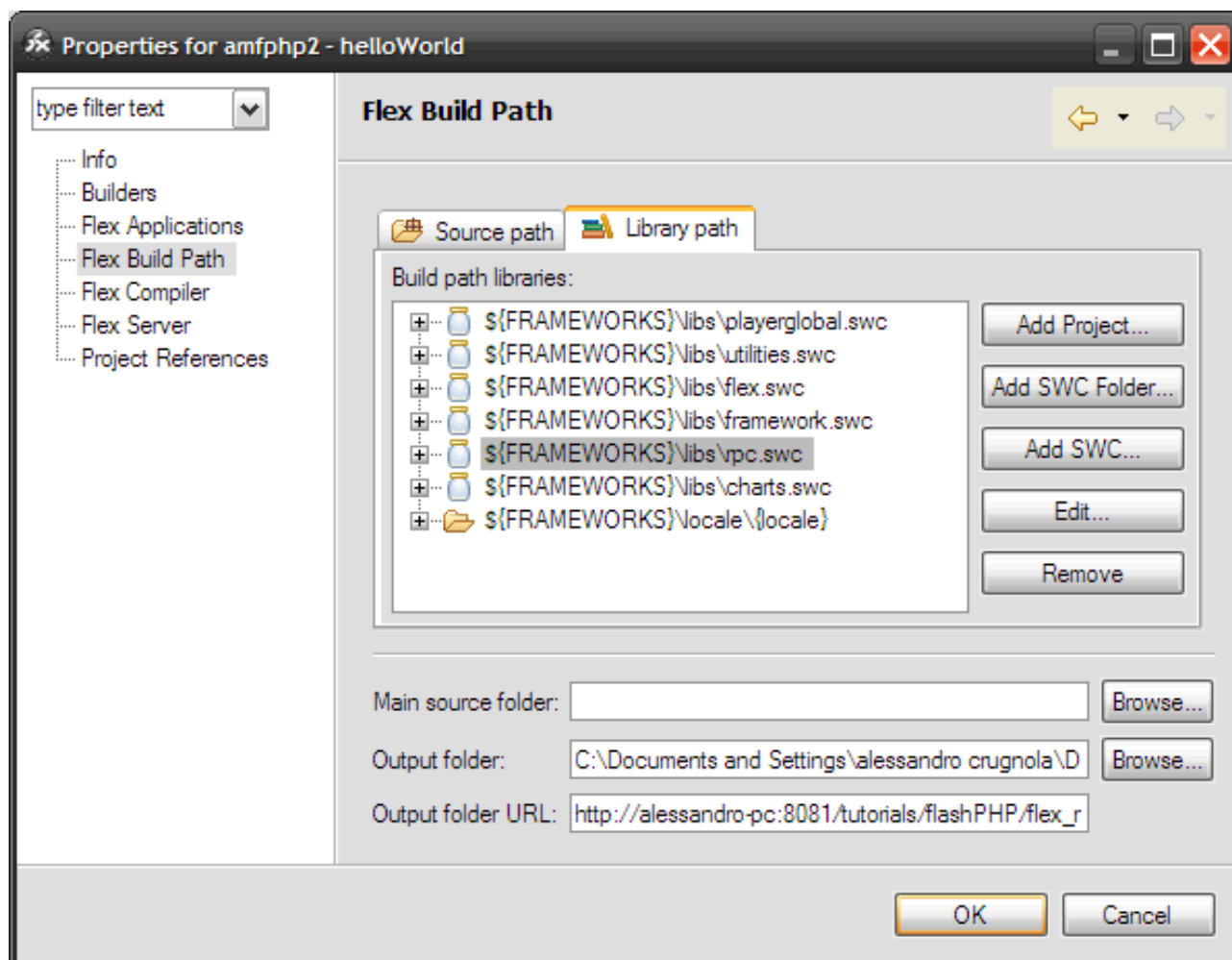
Oto przykład służący do testowania aplikacji na localhost (proszę utworzyć plik xml i zapisać go w katalogu projektu):

```
<?xml version="1.0" encoding="UTF-8"?>
<services-config>
  <services>
    <service id="amfphp-flashremoting-service"
      class="flex.messaging.services.RemotingService"
      messageTypes="flex.messaging.messages.
      RemotingMessage">
      <destination id="amfphp">
        <channels>
          <channel ref="my-amfphp"/>
        </channels>
        <properties>
          <source>*</source>
        </properties>
      </destination>
    </service>
  </services>
  <channels>
    <channel-definition id="my-amfphp" class="mx.
    messaging.channels.AMFChannel">
      <endpoint uri="http://localhost/
      flashservices/gateway.php" class="flex.
      messaging.endpoints.AMFEndpoint"/>
    </channel-definition>
  </channels>
</services-config>
```

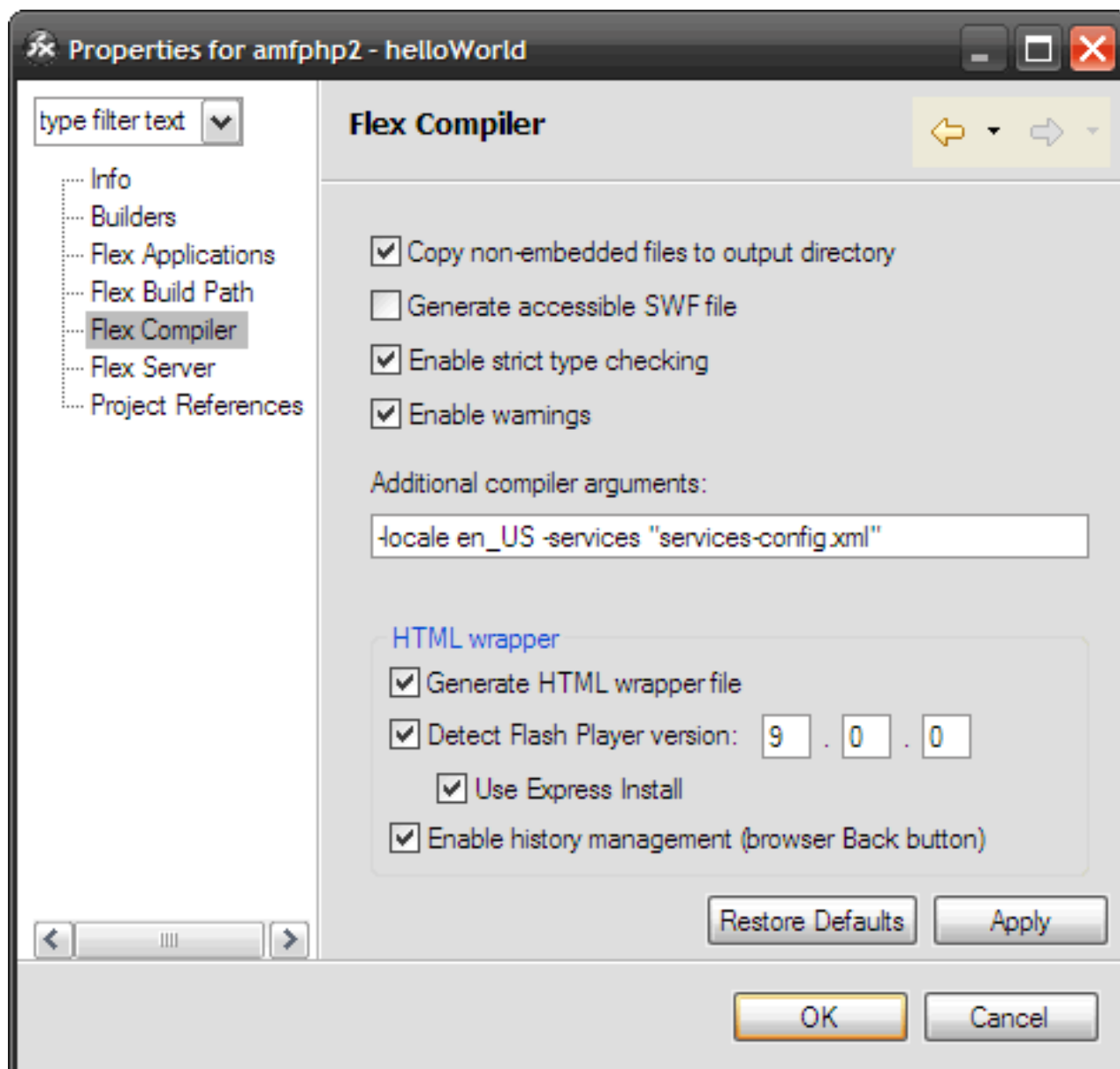
Dodatkowo, aby Flex odpowiednio skompilował projekt na swf należy dopisać argument wskazujący ten plik. Podaję rozwiązanie zarówno dla 'FD + SDK' oraz 'Flex Builder'a':

Flex Builder:

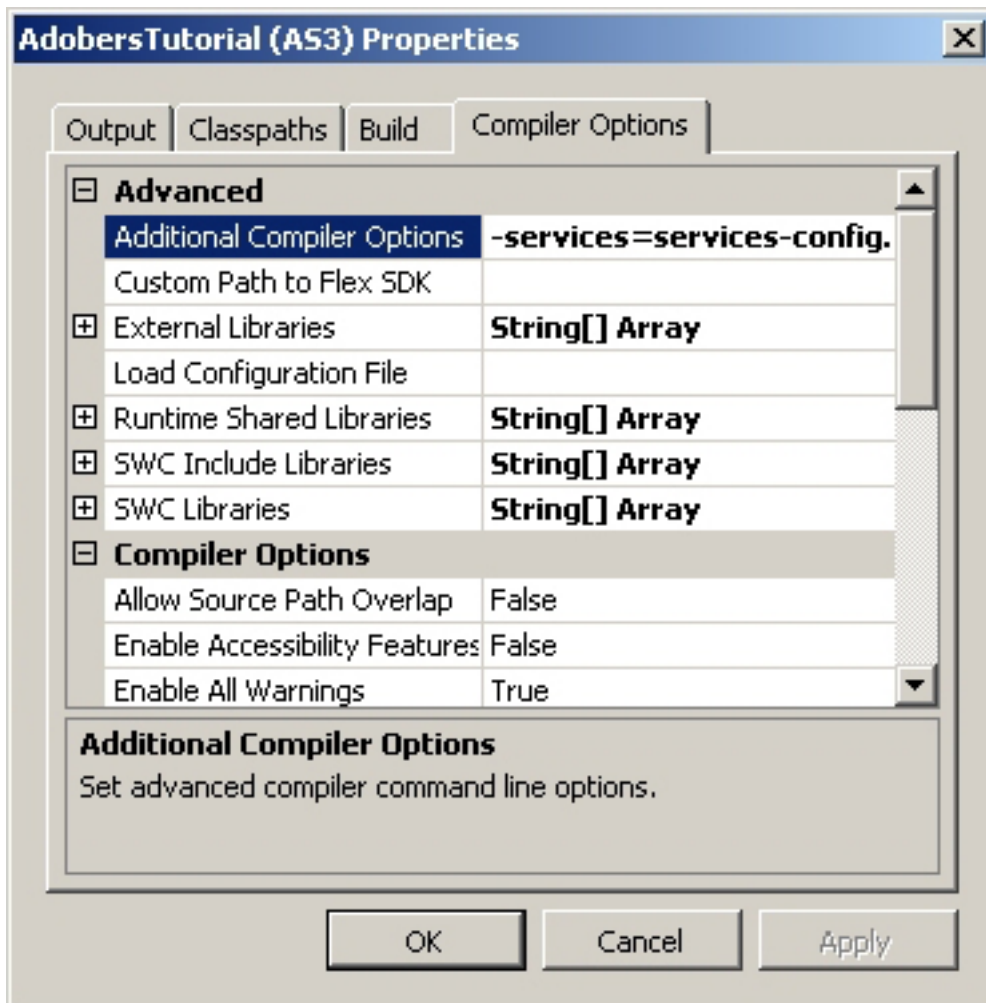
W właściwościach projektu 'Project properties', kategoria 'Flex Build path', zakładka 'Library path' proszę upewnić się, że jest zaznaczona opcja rpc.swc. (Obrazek poniżej):



Następnie wskazujemy na plik services-config.xml. W właściwościach projektu 'Project properties', kategoria 'Flex Compiler' proszę dodać: '-services „services-config.xml”'. (Obrazek poniżej):

**FD + SDK:**

W tym przypadku wystarczy dopisać w zakładce 'Project'/'Properties'/'Compiler Options'/'Additional Compiler Options' linijkę '-services=services-config.xml'. (Obrazek poniżej):



Kolejnym etapem będzie stworzenie strony www wyświetlającej zawartość (content). Dla prostego zobrazowania sytuacji dałem przykład standardowego tekstu wygenerowanego z Lorem-Ipsum.

Na początku nastąpi wywołanie metody ReadContent z naszej usługi, która po połączeniu z bazą zwróci zawartość pola „text” tabeli „content”, a następnie wyświetli ją na stronie.

Oto listing pliku [coreSite.mxml](#) naszej strony:

```
<?xml version="1.0" encoding="utf-8"?>

<!-- Simple site content -->
<!-- @author FAKETA -->

<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
backgroundColor="#1e2d3b">

    // Ustawienie usługi ReadContent.php
    <mx:RemoteObject id="serviceReadContent"
```

```
fault="faultHandler(event)" showBusyCursor="true"
source="AdobersTutorial.ReadContent"
destination="amfphp">
<mx:method name="ReadContent"
result="resultHandler(event)" />
</mx:RemoteObject>

<mx:Script>
<![CDATA[

import mx.managers.CursorManager;
import mx.rpc.events.ResultEvent;
import mx.rpc.events.FaultEvent;

private function faultHandler (fault:FaultEvent) :
void {

CursorManager.removeBusyCursor();
content.htmlText = „code:\n” + fault.fault.
faultCode + „\n\nMessage:\n” + fault.fault.
faultString + „\n\nDetail:\n” + fault.fault.
faultDetail;

}

private function resultHandler (evt:ResultEvent) :
void {

content.htmlText = evt.message.body.
toString();

}

]]>
</mx:Script>

// Panel strony www
<mx:Panel title="Content Site Example" height="240"
width="360" paddingTop="10" paddingLeft="10"
paddingRight="10">

// Pole wyświetlające zawartość pobraną z bazy danych
wykorzystując usługę
<mx:Text width="100%" id="content"
initialize="serviceReadContent.getOperation('ReadContent').
send();" />

</mx:Panel>

</mx:Application>
```

Content Site Example

- Lorem ipsum dolor sit amet, consectetur adipiscing
- Cras non tellus eu erat faucibus pretium.
- Mauris vitae tellus eget ligula condimentum pretium.
- Donec eleifend tincidunt enim.
- Aenean lacinia nunc ut risus.

Lorem ipsum

Nunc iaculis erat in elit. Mauris sit amet sem vitae enim hendrerit vulputate. Nulla gravida malesuada nisl. Cras ipsum magna, auctor vel, vehicula et, pretium eu, ante.

Utworzenie panelu miniCMS

Głównym elementem naszego panelu miniCMS jest RichTextEditor pozwalający na edycję tekstu będącym standardem wykorzystywanym w wielu programach tj. Lotus Notes, MS Office, Open Office, itd.

Zaraz po uruchomieniu panelu (a najczęściej będzie to autoryzacja użytkownika) nastąpi zczytanie aktualnej zawartości pola 'text' z tabeli 'content' naszej bazy 'site_content' po czym, wyświetlenie jej w polu edytowalnym RichTextEditor. Podobnie jak w przypadku strony www, odpowiedzialną za to usługą będzie [ReadContent.php](#).

W tym momencie mamy możliwość edycji zawartości, a następnie zapisanie zmian w bazie danych za pomocą przycisku 'Save'. Programowo, wykorzystujemy do tego usługę [WriteContent.php](#).

Oto listing pliku [panel.mxml](#) naszego panelu miniCMS (plik umieszczamy w katalogu /panel naszego projektu):

```
<?xml version="1.0" ?>

<!-- Simple cms panel -->
<!-- @author FAKETA -->

<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
backgroundColor="#1e2d3b">

    // Ustawienie usługi WriteContent.php
    <mx:RemoteObject id="serviceWriteContent"
fault="faultHandler(event)" showBusyCursor="true"
source="AdobersTutorial.WriteContent"
destination="amfphp">
    <mx:method name="WriteContent"
result="writeResultHandler(event)" />
    </mx:RemoteObject>

    // Ustawienie usługi ReadContent.php
    <mx:RemoteObject id="serviceReadContent"
fault="faultHandler(event)" showBusyCursor="true"
source="AdobersTutorial.ReadContent"
destination="amfphp">
    <mx:method name="ReadContent"
result="readResultHandler(event)" />
    </mx:RemoteObject>

    <mx:Script>
    <![CDATA[

        import mx.managers.CursorManager;
        import mx.rpc.events.ResultEvent;
        import mx.rpc.events.FaultEvent;
        import mx.controls.Alert;
```

```
import mx.events.CloseEvent;

// wyświetlenie zapytania „Czy zapisać”
private function clickHandler (event:Event) : void
{

    Alert.show(„Do you want to save your
changes?”, „Save Changes”, 3, this,
alertClickHandler);

}

// sprawdzenie zaznaczonej odpowiedzi zapytania
„Czy zapisać”
// jeśli odp: YES (1) - wywołanie metody
WriteContent usługi WriteContent.php
private function alertClickHandler (event:
CloseEvent) : void {

    if (event.detail==Alert.YES) {

        serviceWriteContent.getOperation
('WriteContent').send(rte.htmlText);

    }

}

// zdarzenie błędu podczas połączenia z AMFPHP
private function faultHandler (fault:FaultEvent) :
void {

    CursorManager.removeBusyCursor ();
    rte.htmlText = „code:\n” + fault.fault.
faultCode + „\n\nMessage:\n” + fault.fault.
faultString + „\n\nDetail:\n” + fault.fault.
faultDetail;

}

// zdarzenie poprawnego zapisania danych w bazie
private function writeResultHandler (evt:
ResultEvent) : void {

    Alert.show ('Data saved!', 'miniCMS info');

}

private function readResultHandler (evt:
ResultEvent) : void {

    rte.htmlText = evt.message.body.
toString();

}

}
```

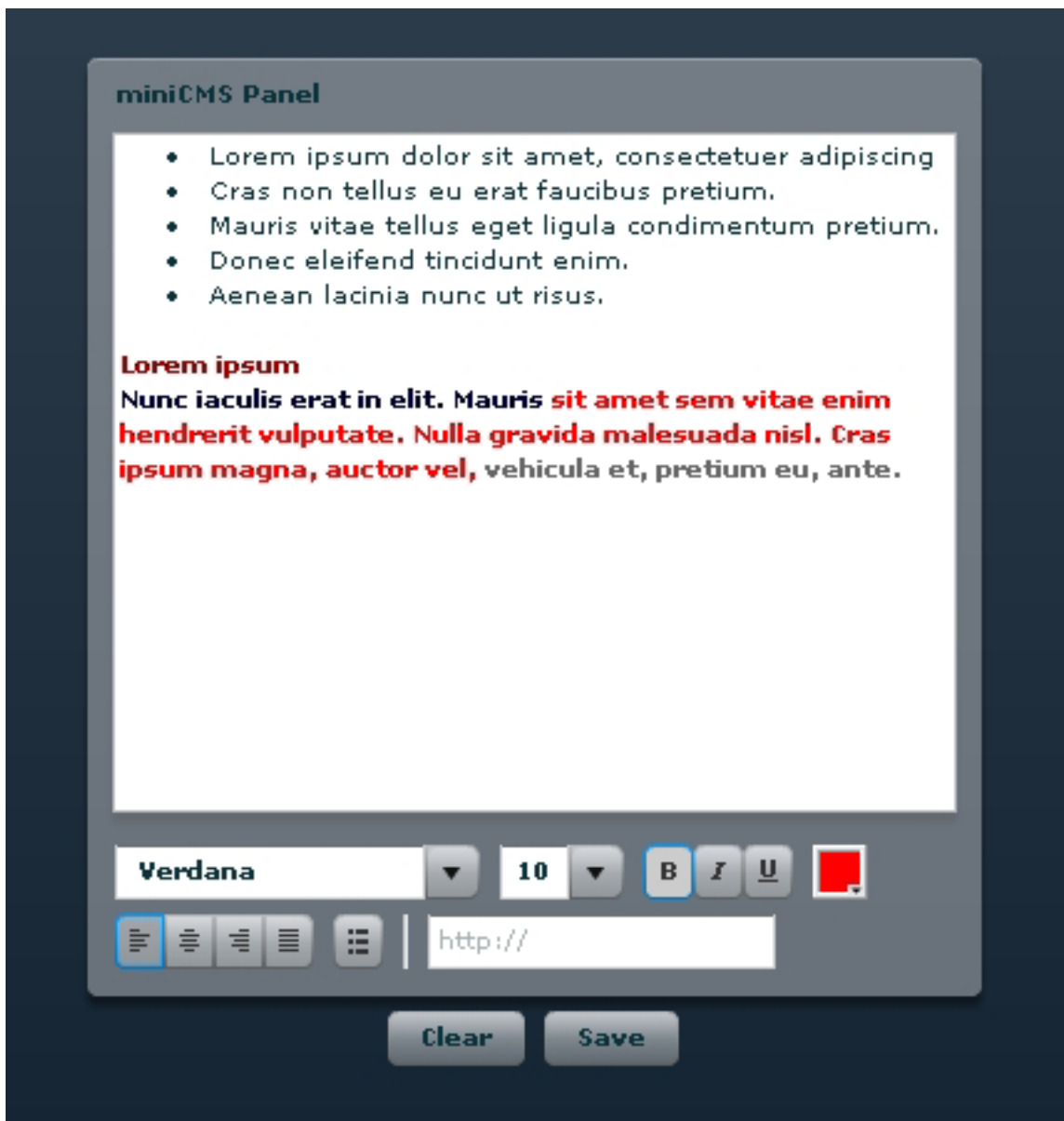
]]>

```
</mx:Script>

<mx:RichTextEditor id="rte" title="miniCMS Panel"
width="360" height="50%" initialize="serviceReadContent.
getOperation('ReadContent').send();" />

<mx:HBox>
<mx:Button label="Clear" click="rte.htmlText = ''" />
<mx:Button label="Save" click="clickHandler(event);" />
</mx:HBox>

</mx:Application>
```



Testowanie funkcjonalności miniCMS

Jak to w każdym projekcie bywa, przychodzi czas na testy. Tak będzie i tutaj.

Zaczynamy od testów lokalnych.

Po efektywnych kompilacjach projektu oraz utworzeniu konkretnych usług, umieszczamy nasz projekt w specjalnym katalogu utworzonym na root localhost'a, np. 'minicms'.

Tam też należy skierować naszą przeglądarkę - przykładowo, aby otworzyć stronę www - '<http://localhost/minicms/coreSite.swf>'. Dla panelu miniCMS będzie to ścieżka '<http://localhost/minicms/panel/panel.swf>'.

Zasada działania jest prosta. Dokonujemy zmian w panelu miniCMS, zapisujemy, a następnie odświeżamy stronę www.

Migracja na serwer

Aby projekt działał na serwerze należy pozmienić ustawienia użytkowników, haseł, a także ścieżek/lokalizacji.

W skrócie, podaje naistotniejsze elementy konfiguracji:

Pliki - [ReadContent.php](#), [WriteContent.php](#) linijki:

// Ustawienia dla połączenia

```
$host = '';  
$user = '';  
$pass = '';
```

Plik - [services-config.xml](#) linijka:

```
<endpoint uri="http://localhost/flashservices/gateway.php"  
class="flex.messaging.endpoints.AMFEndpoint"/>
```

Jeśli wszystko działa prawidłowo to znaczy, że jesteś na dobrej drodze aby rozpocząć projektowanie bardziej rozbudowanych CMS'ów.