Artur Żarski

Microsoft
SharePoint 2010

dla programistów

APN Promise
Warszawa 2010

Microsoft SharePoint 2010 dla programistów

Projekt graficzny okładki: Maciej Kiełkucki

APN PROMISE Sp. z o. o. Warszawa 2010

APN PROMISE Sp. z o. o., biuro: ul. Kryniczna 2, 03-934 Warszawa
tel. (022) 355-16-00; fax (022) 355-16-99
e-mail: mspress@promise.pl

ISBN: 978-83-7541-069-3

Redakcja i korekta: Ewa Swędrowska
Skład i łamanie: MAWart Marek Włodarz

Spis treści

Wprowadzenie

Microsoft SharePoint Server to rozwiązanie, które za pomocą wbudowanych funkcji pozwala na tworzenie zaawansowanych rozwiązań portalowych dla intranetu, extranetu oraz Internetu. Microsoft SharePoint Server, w skrócie MOSS, SPS, SPPS* lub po prostu portal, został zaprojektowany, aby przy jego pomocy możliwa była obsługa wszystkich aplikacji w obrębie jednej zintegrowanej platformy (zamiast w wielu oddzielnych systemach). Dodatkowo serwer ten pozwala na współpracę i zarządzanie zawartością zapewniając informatykom i programistom narzędzia potrzebne do administrowania serwerem oraz do zapewnienia współdziałania i rozszerzalności aplikacji.

Standardowa instalacja produktu dostarcza wiele komponentów, które są w stanie sprostać większości wymagań biznesowych użytkowników. Jednocześnie, po dłuższym czasie pracy z portalem, możemy stwierdzić, że liczba funkcjonalności, które są wbudowane w portal, jest stanowczo za mała dla naszych potrzeb. Jak każdy produkt Microsoft również i ten daje nam możliwość rozbudowy. Bardzo często można spotkać się z określeniem, że SharePoint jest jak mieszkanie kupione u developera – ma okna i ściany, ale ostateczny wygląd i funkcjonalność zależy tylko i wyłącznie od nas. Programista jest co prawda związany budową samego serwera oraz ograniczeniami narzuconymi przez API. Nie zmienia to faktu, że możliwości działania są w zasadzie nieograniczone, a programista może zbudować dowolny element i umieścić go jako swoje rozwiązanie w portalu.

Microsoft SharePoint 2010 jest najnowszą wersją portalu i jest dostępny w trzech podstawowych edycjach:

- SharePoint 2010 Server Enterprise
- SharePoint 2010 Server Standard
- Microsoft SharePoint Foundation

Każde z tych wydań zawiera inny zestaw funkcjonalności; edycja Microsoft SharePoint Foundation jest odpowiednikiem Windows SharePoint Services z poprzednich wersji.

---

* SharePoint ma już dość długą historię i wraz z kolejnymi wersjami modyfikacjom ulegała także nazwa. Poprzednia wersja rozwiązania nosiła nazwę „Microsoft Office SharePoint Server", w skrócie MOSS – i ten skrót nadal jest popularny wśród programistów. Skrót SPS pochodzi od nazwy „SharePoint Services", a SPPS – od „SharePoint Portal Server" (wersja jeszcze wcześniejsza). Spotkać też można akronim WSS (Windows SharePoint Services). Dla zapewnienia przejrzystości zdecydowałem się na konsekwentne stosowanie skrótu MOSS jako najbardziej rozpowszechnionego i nie kolidującego z innymi, często spotykanymi akronimami.

Nie tylko sama nazwa produktów rozróżnia starą i nową wersję produktu. Microsoft SharePoint 2010 to przede wszystkim technologia oparta na 64-bitowym systemie operacyjnym. SharePoint 2010 to również możliwość instalacji na systemach klienckich, takich jak Windows Vista czy Windows 7, a także zmiany w sposobie tworzenia rozwiązań i administracji systemem. Aktualnie podstawowym narzędziem do tworzenia rozwiązań jest Visual Studio 2010, które zostało wyposażone w zestaw dodatkowych szablonów umożliwiających tworzenie zaawansowanych rozwiązań.

Książka ta ma na celu pokazanie nowych elementów istotnych z punktu widzenia programisty. Nie zawiera omówienia podstaw programowania dla SharePoint i Office, a raczej pokazuje różnice i właśnie te nowe elementy, które są ważne dla programistów.

Konwencje zastosowane w tej książce

W celu podniesienia czytelności i przejrzystości w książce zastosowano kilka sposobów wyróżniania tekstu. Nazwy poleceń lub elementów interfejsu użytkownika (zarówno w Visual Studio, jak i na stronie administracyjnej portalu) zostały wyróżnione kolorem zielonym. Opcje lub polecenia menu wybierane (klikane) przez użytkownika wyróżniono kolorem czerwonym. Nazwy struktur programistycznych, metod i innych elementów logicznych wydrukowano w kolorze niebieskim. Fragmenty kodu (zarówno jako odrębny blok, jak i włączone w zasadniczy tekst) zostały złożone `czcionką stałopozycyjną`. Wyróżnienia występują też we fragmentach kodu – w tym wypadku zastosowane zostało **`pogrubienie i kolor niebieski`**. Nazwy plików i katalogów zostały złożone *kursywą*.

Szczególnie istotne uwagi kierowane do Czytelnika zostały wyróżnione ramkami. Także na przedstawionych przykładach ekranów istotne elementy zostały umieszczone w kolorowych ramkach.

1 Wstęp do programowania w środowisku SharePoint

Główne zagadnienia:

- Nowe elementy dla programisty w SharePoint 2010
- DeveloperDashboard
- Debugowanie aplikacji SharePoint

Microsoft SharePoint 2010 to zestaw kilkudziesięciu funkcjonalności, które możemy podzielić na sześć podstawowych obszarów tematycznych: witryny, społeczności, wyszukiwanie, treść, dostęp do danych czy raportowanie. Krótka charakterystyka każdego z tych obszarów wygląda następująco:

- **Witryny** tworzenie i zarządzanie witrynami, obszarami roboczymi, współpraca nad dokumentami;
- **Społeczności** mechanizmy obsługujące blogi, strony typu Wiki czy profile użytkowników;
- **Wyszukiwanie** mechanizmy wyszukiwania i indeksowania treści, podpowiadania wyników wyszukiwania czy też łączenia ludzi na portalu;
- **Treść** typowe obszary związane z zarządzaniem treścią;
- **Dostęp do danych** możliwość budowania aplikacji opartych o zewnętrzne źródła danych bez konieczności pisania kodu programu (Business Conectivity Services – BCS);
- **Raportowanie** SharePoint 2010 zawiera zaawansowane narzędzia analizy biznesowej (Business Intelligence), prezentacji danych na wykresach oraz raportowania.

Przedstawione powyżej punkty są podstawowymi obszarami funkcjonalnymi. Warto również zastanowić się, co jest nowe z punktu widzenia programisty. Wśród nowości wyróżnić możemy:

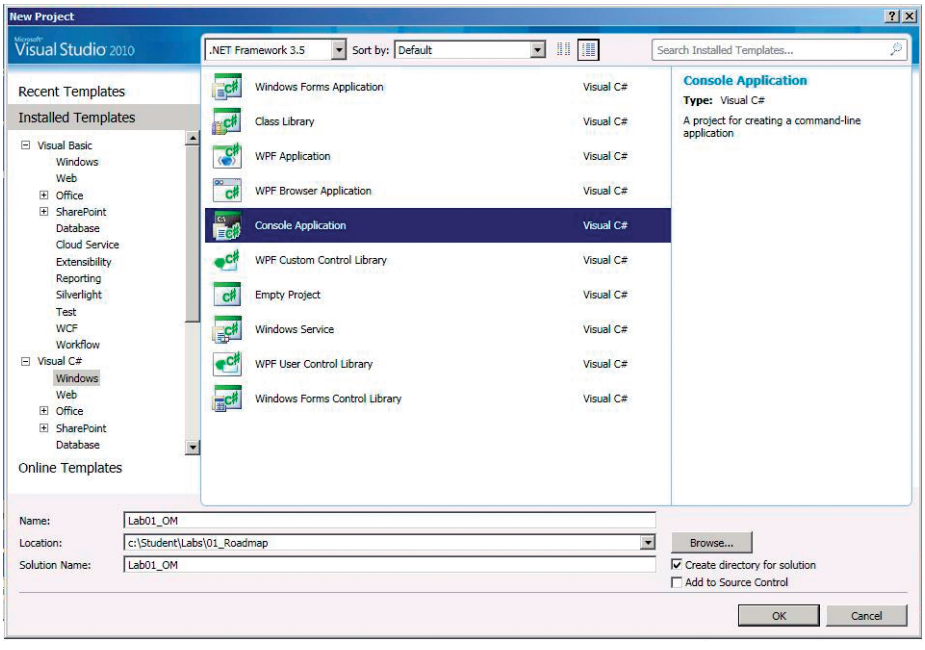
- Zaawansowane mechanizmy interfejsu użytkownika (UI) i możliwości dostępu do ich definicji;
- Wsparcie narzędziowe dla tworzenia rozwiązań w Visual Studio;
- Pełna integracja z PowerShell;
- Wzbogacone mechanizmy obsługi zdarzeń na listach;
- Integracja z LINQ;
- Nowy i lekki kliencki model obiektowy;
- Rozszerzone wsparcie do pracy z procesami biznesowymi;
- Nowa, rozszerzalna architektura aplikacji stanowiących usługi w portalu;
- Współpraca z zewnętrznymi źródłami danych;

- Nowe mechanizmy w architekturze wyszukiwania;
- Tworzenie bezpiecznych rozwiązań w oparciu o mechanizmy Sandbox oraz Partially Trusted Code;
- Bezpieczeństwo oparte o tokeny;
- Możliwość instalacji na komputerach z Windows Vista i Windows 7.

Kolejne rozdziały książki będą poruszać i rozwijać wymienione wcześniej zagadnienia. W tym rozdziale zaprezentowane będą podstawowe kwestie związane z tworzeniem rozwiązań dla MOSS przy wykorzystaniu Visual Studio 2010. Istotne problemy związane z tworzeniem rozwiązań to między innymi:

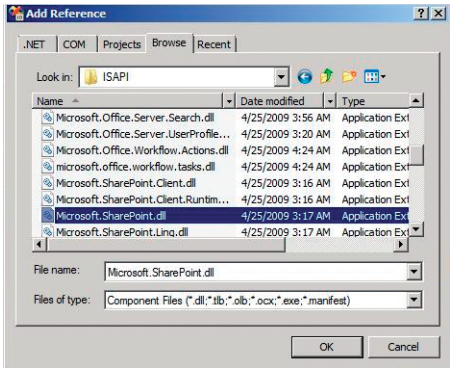
- Pisanie aplikacji wymaga posiadania Visual Studio 2010.
- Tworzone rozwiązania nie mogą być oparte o .NET Framework 4.0 (które jest standardowym środowiskiem w VS 2010); należy wybierać środowisko .NET Framework 3.5, ponieważ na nim oparty jest cały portal.
- MOSS jest rozwiązaniem przeznaczonym dla architektury 64 bitowej, w związku z tym wszystkie rozwiązania muszą być tworzone na platformę x64 lub „All CPU".

Zacznijmy od stworzenia najprostszej aplikacji konsolowej prezentującej wspomniane wcześniej informacje. Z menu Start → All Programs → Microsoft Visual Studio 2010 uruchamiamy Microsoft Visual Studio 2010, następnie tworzymy nowy projekt i wybieramy do utworzenia aplikację konsolową, tak jak na rysunku 1-1.



Rysunek 1-1. *Tworzenie aplikacji konsolowej w Visual Studio 2010*

Na tym etapie wybieramy .NET Framework 3.5 z rozwijanej listy Target Framework. Po utworzeniu projektu dodajemy odpowiednie referencje do bibliotek SharePoint. W oknie Solution Explorer klikamy prawym przyciskiem myszy na gałęzi References, a następnie Add Reference. Wszystkie biblioteki związane z programowaniem portalu znajdują się w folderze: *C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\ISAPI*. Wybieramy Microsoft.SharePoint.dll, tak jak na rysunku 1-2.



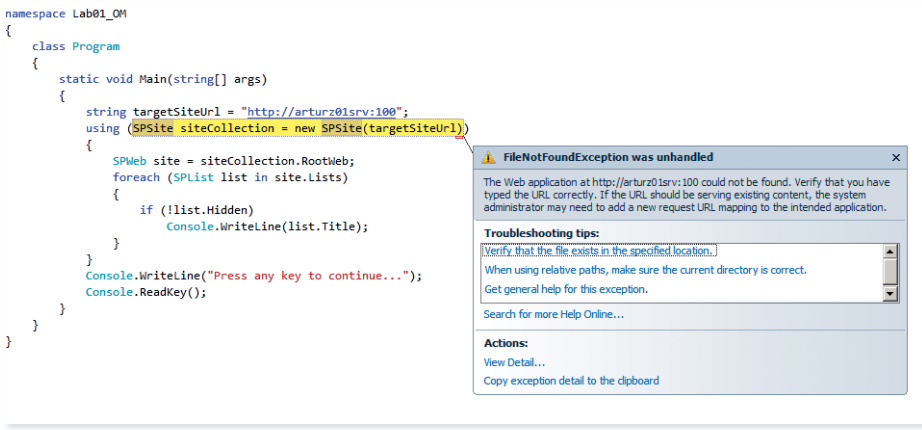
Rysunek 1-2. *Wybór bibliotek SharePoint*

Piszemy krótki program wyświetlający tytuł strony.

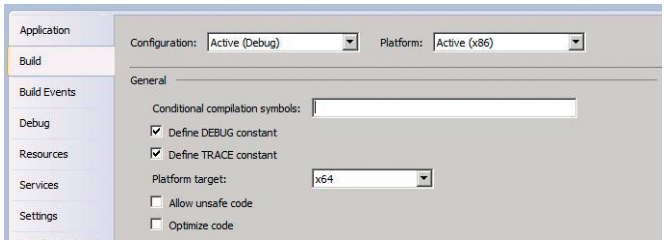
```
using System;
using Microsoft.SharePoint;
using Microsoft.SharePoint.Administration;

namespace Lab01_OM
{
  class Program {
   static void Main()
   {
     string targetSiteUrl = "http://intranet.contoso.com/sites/Lab01A";
     using (SPSite siteCollection = new SPSite(targetSiteUrl))
     {
      SPWeb site = siteCollection.RootWeb;
      Console.WriteLine(site.Title);
      Console.ReadLine();
     }
   }
  }
}
```

Uruchamiamy program. Próba uruchomienia kończy się błędem, takim jak na rysunku 1-3. Jest to spowodowane tym, że została wybrana platforma x86. We właściwościach projektu z listy Platform target wybierzmy x64 lub All CPU (rysunek 1-4).



Rysunek 1-3. *Typowy błąd spowodowany uruchomieniem aplikacji w trybie x86*



Rysunek 1-4. *Właściwości projektu – konfiguracja platformy dla kompilatora*

Ponowne uruchomienie programu nie generuje już błędu i program działa w prawidłowy sposób (rysunek 1-5).

Rysunek 1-5. *Wynik działania aplikacji konsolowej*

Przy wykorzystaniu modelu obiektowego możliwe jest stworzenie dowolnej listy lub umieszczenie na liście właściwego elementu. Oto przykładowy fragment kodu:

```
Guid ListId = site.Lists.Add("tytuł", "opis",
   SPListTemplateType.Tasks);
SPList newList = site.Lists[ListId];
newList.EnableAttachments = false;
newList.EnableFolderCreation = false;
newList.OnQuickLaunch = true;
newList.Update();
```

Kod ten tworzy listę typu zadania, a dopisanie elementów do listy realizowane jest za pomocą poniższego fragmentu:

```
SPUser targetUser = site.CurrentUser;
DateTime today = DateTime.Today;
AddItem(newList, "Projekt architektury", targetUser, "(1) High", today.AddDays(2), 0);
AddItem(newList, "Pozyskanie programistów", targetUser, "(2) Normal", today.AddDays(4), 1);
AddItem(newList, "Edukacja zespołu", targetUser, "(1) High", today.AddDays(5), 2);
```

gdzie metoda AddItem zdefiniowana jest następujaco:

```
private static void AddItem(SPList newList, string Title, SPUser AssignedTo, string
Priority, DateTime DudeDate, int Predecessors) {
     SPListItem newItem = newList.Items.Add();
     newItem["Title"] = Title;
     newItem["AssignedTo"] = AssignedTo;
     newItem["Priority"] = Priority;
     newItem["DueDate"] = DudeDate;
     newItem["PercentComplete"] = 0;
     if (Predecessors > 0) {
      newItem["Predecessors"] = Predecessors;
     }
     newItem.Update();
   }
```

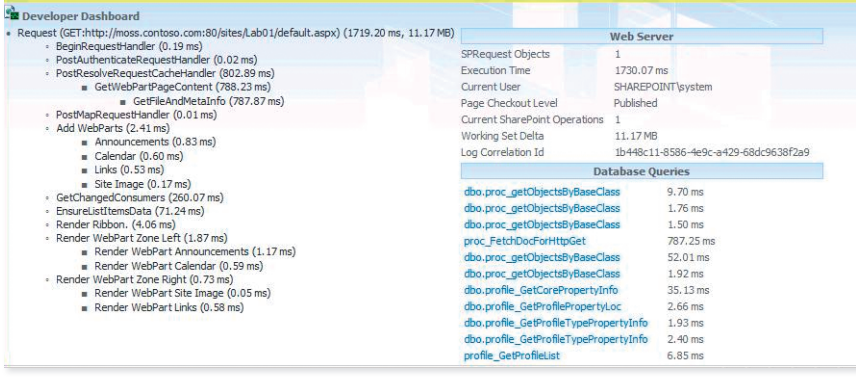
Programista tworząc aplikacje uruchamiane w portalu często napotyka na różnego rodzaju problemy. W przypadku rozwiązywania problemów przydatne są informacje diagnostyczne, które mówią między innymi o tym, jak długo wykonywało się zapytanie, który Event Handler został uruchomiony, czy i w jakiej sekwencji nastąpiła obsługa zdarzeń. Do wyświetlenia tego typu informacji został utworzony mechanizm o nazwie Developer Dashboard. W wersji SharePoint 2010 nie ma standardowo w menu opcji pozwalającej na włączenie lub wyłączenie tej funkcjonalności, ale przy użyciu krótkiego programu możliwe jest jej włączenie lub wyłączenie.

Do uruchomienia Developer Dashboard posłużymy się następującym kawałkiem kodu dla aplikacji konsolowej:

```
using System;
using Microsoft.SharePoint;
using Microsoft.SharePoint.Administration;
namespace Lab01_OM {
  class Program {
   static void Main() {
     SPWebService contentService = SPWebService.ContentService;
     SPDeveloperDashboardSettings developerDashboard =
       contentService.DeveloperDashboardSettings;
     developerDashboard.DisplayLevel = SPDeveloperDashboardLevel.On;
     developerDashboard.Update();
     Console.WriteLine("Developer Dashboard zaktualizowany.");
   }
  }
}
```

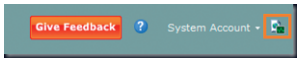
Najważniejszym obiektem jest developerDashboard typu SPDeveloperDashboard-Settings, który pozwala na włączenie lub wyłączenie konsoli programisty. Może on przyjąć trzy wartości:

- `SPDeveloperDashboardLevel.On` – panel włączony
- `SPDeveloperDashboardLevel.Off` – panel wyłączony
- `SPDeveloperDashboardLevel.OnDemand` – panel włączany na żądanie programisty za pomocą dodatkowej ikony. Na rysunku 1-6 przestawiony jest efekt włączenia opcji Developer Dashboard. Na stronie portalu wyświetlane są dodatkowe informacje o tym, jak wyglądał proces uruchamiania danej strony.



Rysunek 1-6. *Włączona opcja DeveloperDashboard*

Rysunek 1-7 ukazuje ikonę dostępną po wybraniu opcji OnDemand. W takiej konfiguracji, obok nazwy użytkownika w lewej górnej części ekranu pojawia się ikona pozwalająca na pokazanie Developer Dashboard.



Rysunek 1-7. *Opcja OnDemand dla Developer Dashboard*

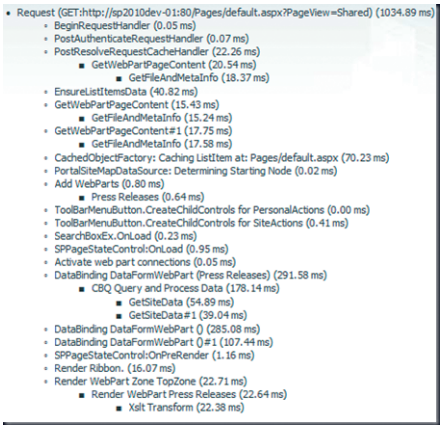
Włączenie tej opcji przy użyciu PowerShell wygląda następująco:

```
$svc=[Microsoft.SharePoint.Administration.SPWebService]::ContentService
$ddsetting=$svc.DeveloperDashboardSettings
$ddsetting.DisplayLevel=[Microsoft.SharePoint.Administration.SPDeveloperDashboardLevel]::On
Demand
$ddsetting.Update()
```

Można ją też włączyć za pomocą polecenia STSADM:

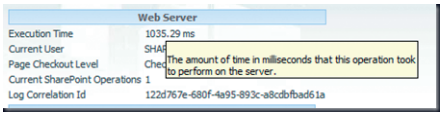
```
stsadm -o setproperty -pn developer-dashboard -pv on
stsadm -o setproperty -pn developer-dashboard -pv off
stsadm -o setproperty -pn developer-dashboard -pv OnDemand
```

Jakie informacje można wyczytać za pomocą tego rozszerzenia? Pierwsza sekcja z prawej strony pokazuje każdą wykonaną operację, całkowity czas potrzebny na jej wykonanie oraz całkowity czas wykonania (rysunek 1-8).



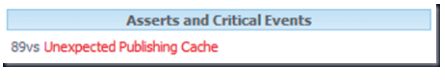
Rysunek 1-8. *Developer Dashboard – wykonane operacje*

Następna sekcja (Web Servers), która znajduje się w lewej górnej części, pokazuje kilka ogólnych informacji, takich jak całkowity czas wykonania, aktualnego użytkownika, tryb zaewidencjonowania strony, liczbę wykonywanych operacji oraz identyfikator korelacji logu, aby łatwiej można było zidentyfikować wpis w dzienniku zdarzeń SharePoint. Dodatkowo po najechaniu kursorem myszy na daną informację dostajemy podpowiedź, czego dokładnie dotyczy dana informacja (rysunek 1-9).



Rysunek 1-9. *Developer Dashboard – informacje w sekcji WebServer*

Kolejna sekcja (Asserts and Critical Errors) dotyczy informacji o krytycznych zdarzeniach podczas ładowania danej strony – np. jeśli dany składnik WebPart długo się wyświetla.

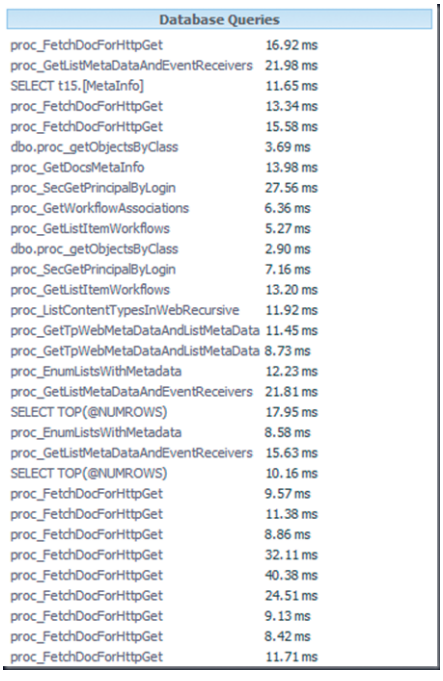


Rysunek 1-10. *Developer Dashboard – Asserts and Critical Events*

Kolejna porcja informacji dotyczy bazy danych (Database Queries) oraz wywołań konkretnych procedur po stronie bazy danych – ich czas wykonania podczas ładowania danej strony.

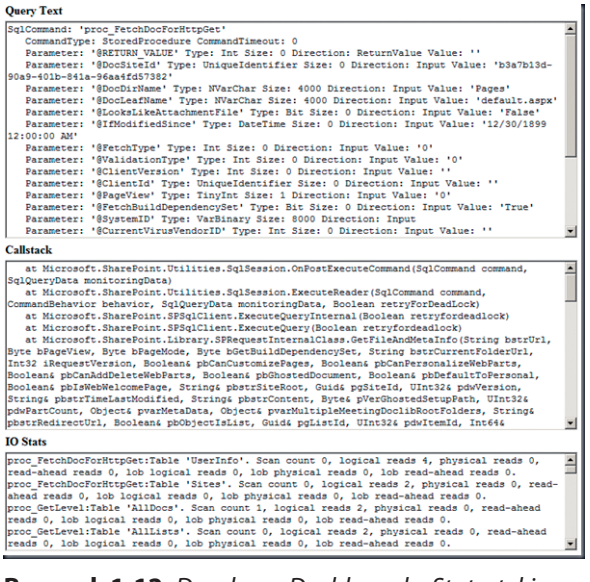
Każdej z wymienionych procedur towarzyszy łącze, które wyświetla okno z dodatkowymi informacjami na temat sposobu wywołania danej procedury, parametrów, statystyk

IO i stos wywołań. Dzięki tym informacjom możemy stwierdzić, że problem z działaniem portalu leży po stronie konfiguracji bazy danych i infrastruktury, a niekoniecznie wynika z winy naszej aplikacji.



Database Queries	
proc_FetchDocForHttpGet	16.92 ms
proc_GetListMetaDataAndEventReceivers	21.98 ms
SELECT t15.[MetaInfo]	11.65 ms
proc_FetchDocForHttpGet	13.34 ms
proc_FetchDocForHttpGet	15.58 ms
dbo.proc_getObjectsByClass	3.69 ms
proc_GetDocsMetaInfo	13.98 ms
proc_SecGetPrincipalByLogin	27.56 ms
proc_GetWorkflowAssociations	6.36 ms
proc_GetListItemWorkflows	5.27 ms
dbo.proc_getObjectsByClass	2.90 ms
proc_SecGetPrincipalByLogin	7.16 ms
proc_GetListItemWorkflows	13.20 ms
proc_ListContentTypesInWebRecursive	11.92 ms
proc_GetTpWebMetaDataAndListMetaData	11.45 ms
proc_GetTpWebMetaDataAndListMetaData	8.73 ms
proc_EnumListsWithMetadata	12.23 ms
proc_GetListMetaDataAndEventReceivers	21.81 ms
SELECT TOP(@NUMROWS)	17.95 ms
proc_EnumListsWithMetadata	8.58 ms
proc_GetListMetaDataAndEventReceivers	15.63 ms
SELECT TOP(@NUMROWS)	10.16 ms
proc_FetchDocForHttpGet	9.57 ms
proc_FetchDocForHttpGet	11.38 ms
proc_FetchDocForHttpGet	8.86 ms
proc_FetchDocForHttpGet	32.11 ms
proc_FetchDocForHttpGet	40.38 ms
proc_FetchDocForHttpGet	24.51 ms
proc_FetchDocForHttpGet	9.13 ms
proc_FetchDocForHttpGet	8.42 ms
proc_FetchDocForHttpGet	11.71 ms

Rysunek 1-11. *Developer Dashboard – Database Queries*



Query Text

```
SqlCommand: 'proc_FetchDocForHttpGet'
   CommandType: StoredProcedure CommandTimeout: 0
   Parameter: '@RETURN_VALUE' Type: Int Size: 0 Direction: ReturnValue Value: ''
   Parameter: '@DocSiteId' Type: UniqueIdentifier Size: 0 Direction: Input Value: 'b3a7b13d-
90a9-401b-841a-96aa4fd57382'
   Parameter: '@DocDirName' Type: NVarChar Size: 4000 Direction: Input Value: 'Pages'
   Parameter: '@DocLeafName' Type: NVarChar Size: 4000 Direction: Input Value: 'default.aspx'
   Parameter: '@LooksLikeAttachmentFile' Type: Bit Size: 0 Direction: Input Value: 'False'
   Parameter: '@IfModifiedSince' Type: DateTime Size: 0 Direction: Input Value: '12/30/1899
12:00:00 AM'
   Parameter: '@FetchType' Type: Int Size: 0 Direction: Input Value: '0'
   Parameter: '@ValidationType' Type: Int Size: 0 Direction: Input Value: '0'
   Parameter: '@ClientVersion' Type: Int Size: 0 Direction: Input Value: ''
   Parameter: '@ClientId' Type: UniqueIdentifier Size: 0 Direction: Input Value: ''
   Parameter: '@PageView' Type: TinyInt Size: 1 Direction: Input Value: '0'
   Parameter: '@FetchBuildDependencySet' Type: Bit Size: 0 Direction: Input Value: 'True'
   Parameter: '@SystemID' Type: VarBinary Size: 8000 Direction: Input
   Parameter: '@CurrentVirusVendorID' Type: Int Size: 0 Direction: Input Value: ''
```

Callstack

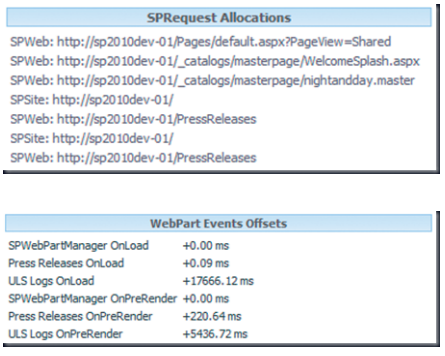
```
   at Microsoft.SharePoint.Utilities.SqlSession.OnPostExecuteCommand(SqlCommand command,
SqlQueryData monitoringData)
   at Microsoft.SharePoint.Utilities.SqlSession.ExecuteReader(SqlCommand command,
CommandBehavior behavior, SqlQueryData monitoringData, Boolean retryForDeadLock)
   at Microsoft.SharePoint.SPSqlClient.ExecuteQueryInternal(Boolean retryfordeadlock)
   at Microsoft.SharePoint.SPSqlClient.ExecuteQuery(Boolean retryfordeadlock)
   at Microsoft.SharePoint.Library.SPRequestInternalClass.GetFileAndMetaInfo(String bstrUrl,
Byte bPageView, Byte bPageMode, Byte bGetBuildDependencySet, String bstrCurrentFolderUrl,
Int32 iRequestVersion, Boolean& pbCanCustomizePages, Boolean& pbCanPersonalizeWebParts,
Boolean& pbCanAddDeleteWebParts, Boolean& pbGhostedDocument, Boolean& pbDefaultToPersonal,
Boolean& pbIsWebWelcomePage, String& pbstrSiteRoot, Guid& pgSiteId, UInt32& pdwVersion,
String& pbstrTimeLastModified, String& pbstrContent, Byte& pVerGhostedSetupPath, UInt32&
pdwPartCount, Object& pvarMetaData, Object& pvarMultipleMeetingDoclibRootFolders, String&
pbstrRedirectUrl, Boolean& pbObjectIsList, Guid& pgListId, UInt32& pdwItemId, Int64&
```

IO Stats

```
proc_FetchDocForHttpGet:Table 'UserInfo'. Scan count 0, logical reads 4, physical reads 0,
read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
proc_FetchDocForHttpGet:Table 'Sites'. Scan count 0, logical reads 2, physical reads 0, read-
ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
proc_GetLevel:Table 'AllDocs'. Scan count 1, logical reads 2, physical reads 0, read-ahead
reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
proc_GetLevel:Table 'AllLists'. Scan count 0, logical reads 2, physical reads 0, read-ahead
reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
```

Rysunek 1-12. *Developer Dashboard – Statystyki wywołań procedur*

Kolejne dwie części dotyczą wywołań serwisowych – SPRequest Allocations oraz WebPart Events Offsets. Pierwsza z nich wyświetla informację na temat alokacji obiektów *SPSite* oraz *SPWeb*. Druga ze statystyk pokazuje informacje o czasie wywołania zdarzeń wewnątrz danego składnika WebPart, np. metoda OnLoad dla obiektu SPWebPartManager. Na rysunku 1-13 przedstawiony jest widok, gdzie ładowany jest więcej niż jeden WebPart. Dla każdego WebPart pokazane są zdarzenia OnLoad, OnPreRender itd., które wywoływane są z SPWebPartManager. Na rysunku widać, że jeden z obiektów ładuje się bardzo długo – 17666,12 ms.



SPRequest Allocations

SPWeb: http://sp2010dev-01/Pages/default.aspx?PageView=Shared
SPWeb: http://sp2010dev-01/_catalogs/masterpage/WelcomeSplash.aspx
SPWeb: http://sp2010dev-01/_catalogs/masterpage/nightandday.master
SPSite: http://sp2010dev-01/
SPWeb: http://sp2010dev-01/PressReleases
SPSite: http://sp2010dev-01/
SPWeb: http://sp2010dev-01/PressReleases

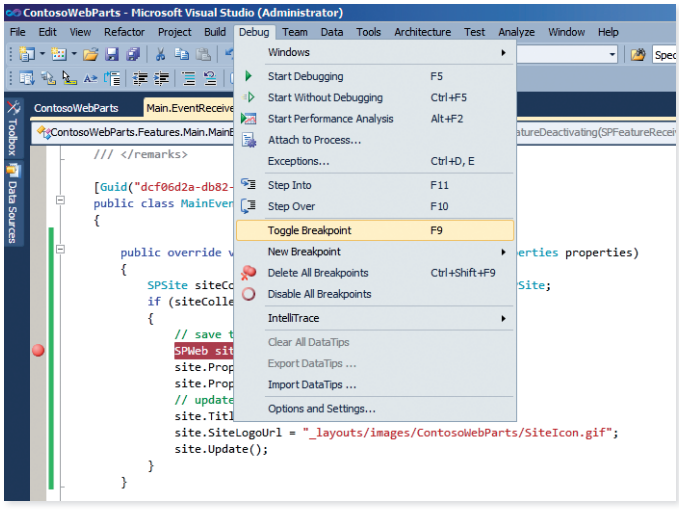
WebPart Events Offsets

SPWebPartManager OnLoad	+0.00 ms
Press Releases OnLoad	+0.09 ms
ULS Logs OnLoad	+17666.12 ms
SPWebPartManager OnPreRender	+0.00 ms
Press Releases OnPreRender	+220.64 ms
ULS Logs OnPreRender	+5436.72 ms

Rysunek 1-13. *Developer Dashboard – SPRequest Allocation oraz WebPart Events Offsets*

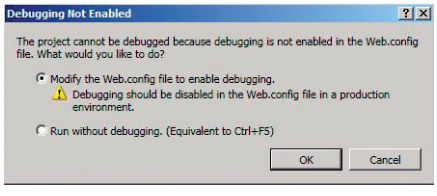
Jednym z ważniejszych procesów podczas tworzenia rozwiązań jest sprawdzenie kodu przy wykorzystaniu mechanizmów debuggera. Podczas pisania aplikacji dla SharePoint 2007 w Visual Studio 2005 i Visual Studio 2008 proces debugowania wymagał od programisty dodatkowej wiedzy na temat procesów obsługujących SharePoint, a w szczególności, który *Worker Process* w IIS obsługiwał działanie naszego portalu. Aby można było wykonać debugowanie aplikacji w poprzednich wersjach, należało ustawić w kodzie programu pułapkę, a następnie z poziomu VS podpiąć się pod odpowiedni proces IIS (*w3wp.exe*).

W Visual Studio 2010 i MOSS 2010 możliwe jest debugowanie z poziomu VS bez konieczności podpinania się pod proces IIS – wykonywane jest to automatycznie za programistę. Aby możliwe było debugowanie rozwiązań, należy uruchomić Visual Studio z prawami administratora, aby mieć odpowiednie uprawnienia do podpięcia się pod odpowiednie procesy. Na rysunku 1-14 pokazane jest ustawienie pułapki w kodzie naszego WebPart.



Rysunek 1-14. *Wstawienie pułapki w kodzie*

Przy pierwszym uruchomieniu rozwiązania, w którym są wstawione pułapki, system zapyta nas, czy zmodyfikować odpowiednio plik *web.config*, aby włączyć obsługę debugowania.



Rysunek 1-15. *Włączenie opcji debugowania*

Teraz możliwe jest standardowe sprawdzanie kodu aplikacji, a system zatrzymuje działanie aplikacji w odpowiednich miejscach (rysunek 1-16).

Rysunek 1-16. *Debugowanie aplikacji*

Z punktu widzenia programisty przydatną nową opcją w Visual Studio jest możliwość przejrzenia zawartości serwera MOSS bezpośrednio w Visual Studio. Server Explorer został rozszerzony o funkcje dostępu do wybranego portalu SharePoint. Rysunek 1-16 przedstawia widok, w którym Server Explorer wyświetla zawartość wskazanego portalu.

SharePoint Explorer jest dodatkiem do Server Explorer, a co za tym idzie, istnieje możliwość jego modyfikacji przez programistę oraz dopisania nowych funkcjonalności.

Podsumowanie

Microsoft SharePoint 2010 to kolejna wersja znanego już produktu. Jego architektura, sposób działania, a także sposób tworzenia rozwiązań również uległy zmianom w porównaniu do poprzednich wersji. W nowym środowisku znalazło się bardzo wiele udogodnień dla każdego rodzaju użytkownika – od zwykłego czytelnika informacji zawartych na portalu do programisty tworzącego rozwiązania oparte o tę platformę.

2 Interfejs użytkownika

Główne zagadnienia:

- Podstawowe informacje o zmianach w interfejsie użytkownika
- Wstążka w SharePoint 2010
- Status Bar oraz Notification Area
- Dialog Framework

Jednym z najważniejszych elementów dowolnego systemu informatycznego, czy to strony WWW, czy aplikacji typu desktop, jest interfejs użytkownika. To za jego pomocą następuje interakcja pomiędzy użytkownikiem a programem. Interfejs SharePoint ewoluuje podobnie jak interfejsy wszystkich produktów Microsoft. W MOSS 2010 pojawia się wstążka, która znana jest wszystkim użytkownikom Office 2007. Jednocześnie pojawia się cały nowy system obsługi zmienionego interfejsu i jego elementów: pasków statusu, wyskakujących okien czy też elementów funkcjonalnych wstążki. Jest to ogromny krok naprzód w porównaniu z poprzednią wersją, w której użytkownik otoczony był ogromną liczbą rozwijanych menu, komend i, co najgorsze, odświeżaniem strony po każdej zmianie. Interfejs MOSS 2007 pozwalał na wiele, ale podczas długiej pracy można było w pewnym momencie zgubić się i stracić kontekst, w którym pracowaliśmy.

Przykładowe scenariusze użycia portalu pokazują różnicę w pracy obu wersji interfejsów. Scenariusz pierwszy to dopisanie nowego elementu na dowolnej liście albo umieszczenie dokumentu w bibliotece dokumentów. W SharePoint 2007 należało wejść do danej biblioteki (pierwsze przeładowanie strony), następnie wybrać opcję dodania nowego elementu (kolejne przeładowanie i wyświetlenie nowej strony), wprowadzić dane i zapisać dane (kolejne przeładowanie strony i wyświetlenie uzupełnionej listy). Jedna czynność, a trzykrotne przeładowanie strony. To jest najprostsza sytuacja, w przypadku bardziej złożonych akcji, takich jak dodanie kolumny lub utworzenie nowego widoku, takich przeładowań jest znacznie więcej.

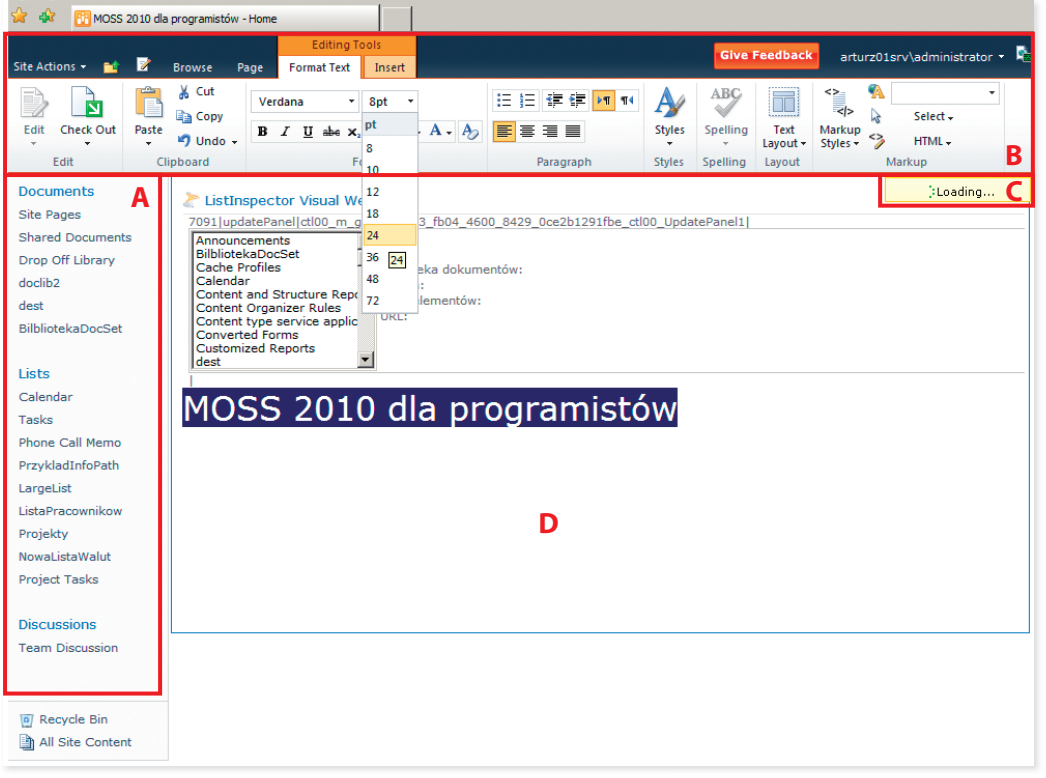
Kolejnym scenariuszem jest typowa praca redakcyjna polegająca na uzupełnieniu treści wraz z umieszczeniem obrazu w tekście. Cały proces przebiega w sposób następujący:

1. Stworzenie biblioteki do przechowywania stron i utworzenie kolejnej strony WebPart.
2. Umieszczenie strony WebPart typu Content Editor, do którego, aby wprowadzić treść, należy przejść przez tryb edycji właściwości danej strony WebPart.
3. Umieszczenie obiektów typu obraz lub video wymaga utworzenia odpowiedniej biblioteki, wcześniejsze załadowanie do niej plików i zapamiętanie URL do obrazów.

Wszystko to sprawia, że użytkownik bardzo szybko może zniechęcić się do pracy, a jego praca staje się mniej wydajna.

SharePoint 2010 to zmiana podejścia użytkownika do pracy z portalem. Praca kontekstowa jest szybka i czytelna, bez względu na miejsce w portalu, w którym się znajdujemy. Co znaczy kontekstowa? Po prostu w zależności od tego, z jakim obiektem na stronie pracujemy, menu automatycznie dostosowuje się udostępniając odpowiednie zestawy funkcji, z których możemy dowolnie korzystać.

Przykład takiego interfejsu użytkownika został przedstawiony na rysunku 2-1.



Rysunek 2-1. *Elementy interfejsu SharePoint 2010*

Składa się on z kilku podstawowych elementów:

A – podręczne menu z zawartością portalu
B – wstążka ze wszystkimi funkcjonalnościami
C – status bar oraz powiadomienia
D – strona z edycją treści w danym oknie

Jak widać na rysunku, wprowadzanie treści następuje w danym oknie. W zależności od elementu, nad którym pracujemy (tekst, tabela, obraz, video, etc), pojawia się inne

menu. W każdym z przypadków jest również automatyczny podgląd, jak będzie wyglądać dany element, zanim dokonamy zamiany (dokładnie tak jak w przypadku Office 2007 i Office 2010).

Każdy ze wspomnianych elementów, wraz z możliwością jego modyfikacji lub rozszerzenia, jest dostępny również dla programisty. Dodatkowo SharePoint 2010 został zbudowany w oparciu o AJAX, dzięki czemu unikamy zbędnych przeładowań stron. Z punktu widzenia programisty ważne jest również to, że SharePoint 2010 jest zgodny ze standardem XHTML 1.0 (poprzednie wersje z HTML 4). Dzięki wsparciu standardu XHTML możliwa jest rownież praca z przeglądarkami innymi niż Internet Explorer. Nowa wersja portalu wspiera Internet Explorer 7 i 8 oraz Firefox 3 na systemach Windows oraz Mac/Linux (z pewnymi ograniczeniami – tabela w Dodatku B), Safari na systemach MacOS (z pewnymi ograniczeniami – Dodatek B) oraz przeglądarki IE7 i IE8 w wersji 64 bit (z ograniczeniami – Dodatek B).

Twórcy stron powinni wziąć również pod uwagę fakt, że nie będzie wsparcia dla Internet Explorer 6.0.

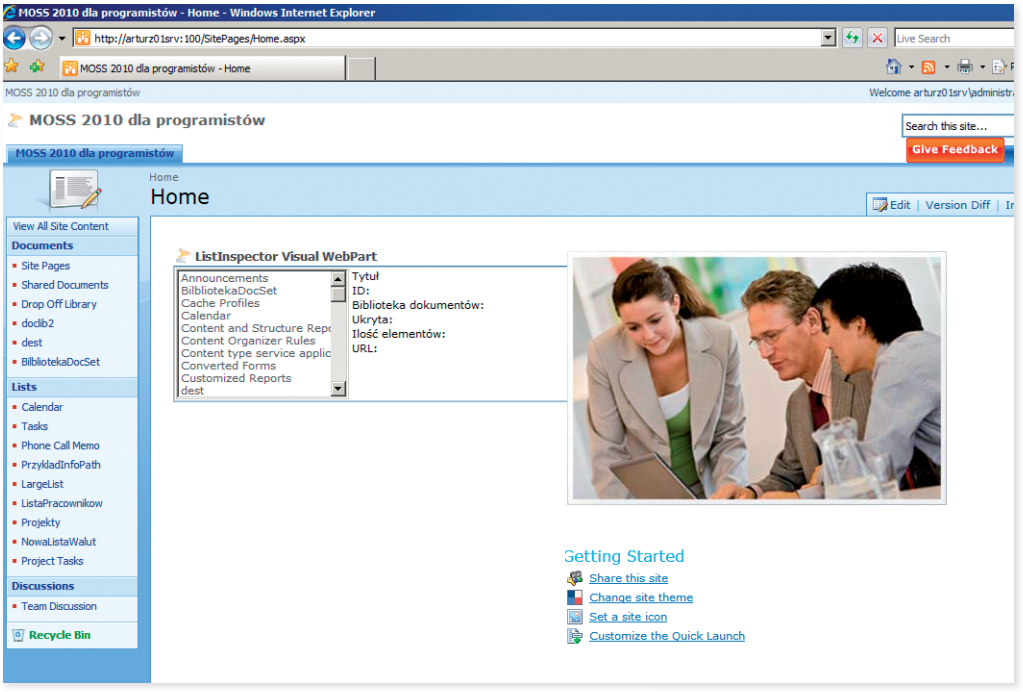
Jeszcze jedną cechą, która może być przydatna dla programistów lub osób odpowiedzialnych za stworzenie odpowiedniego wyglądu naszego portalu, jest możliwość przełączenia wyglądu portalu na poprzednią wersję (SharePoint 2007). Do tego celu należy posłużyć się PowerShell (tylko tutaj dostępna jest ta opcja). Skrypt przedstawiony poniżej przełącza i wyświetla aktualną wersję:

```
PS C:\Users\Administrator> $siteCollection = Get-SPSite "http://arturz01srv:100"
PS C:\Users\Administrator> $site = $ siteCollection.RootWeb
PS C:\Users\Administrator> $site.UIVersion
```

W wyniku otrzymamy 4 – czyli wersję wyglądu SharePoint 2010.
Następujący skrypt

```
PS C:\Users\Administrator> $siteCollection = Get-SPSite "http://arturz01srv:100"
PS C:\Users\Administrator> $site = $ siteCollection.RootWeb
PS C:\Users\Administrator> $ site.UIVersion = 3
PS C:\Users\Administrator> $ site.Update()
```

spowoduje, że nasz portal będzie wyglądał tak, jak na rysunku 2-2.



Rysunek 2-2. *MOSS 2010 z interfejsem w wersji 3*

Wstążka (Ribbon)

Wszystkie funkcjonalności w SharePoint 2007 zostały skonsolidowane do jednego elementu – wstążki. Dzięki temu użytkownik zyskał jednolity wygląd tego elementu, kontrolek w nim zawartych oraz płynność, która jest widoczna między innymi podczas automatycznego dostosowania wielkości wstążki do rozmiarów okna.

Z punktu widzenia programisty najistotniejsze są dwa elementy: JavaScript oraz XML jako platforma służąca rozszerzeniu funkcjonalności wstążki.

Wstążka zawiera wiele różnego rodzaju kontrolek, wśród których można znaleźć między innymi:

- Button – realizujący funkcjonalność przycisku
- Rozwijane oraz pływające menu – rozwijana lista
- Checkbox – pole zaznaczenia (tak/nie)
- Toggle Button – przycisk realizujący obsługę stanów – wciśnięty i nie wciśnięty
- Label – etykieta wyświetlająca tekst
- Textbox – pole do wprowadzenia tekstu
- Combo box – rozwijana lista z możliwością zaznaczenia elementu
- Table control – tabela
- Color picker – wybór koloru