

Projektowanie aplikacji ASP.NET

Wykład 01/15 - Wprowadzenie

Wiktor Zychła 2019/2020

1 Sprawy organizacyjne

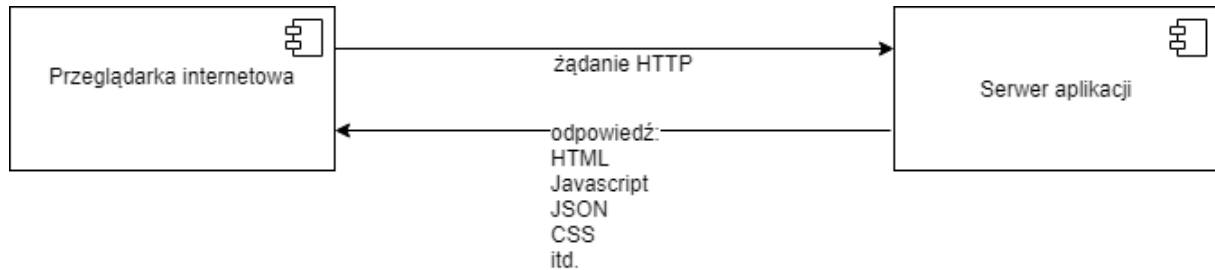
Z przyjemnością witam Państwa na wykładzie Projektowanie aplikacji ASP.NET, który będzie okazją do zapoznania się z bardzo konkretną technologią wytwarzania usług internetowych – z technologią ASP.NET.

W ramach zajęć zostanie zaprezentowany cykl 15 wykładów uzupełnionych spotkaniami w laboratorium, w trakcie którego studenci będą mogli zmierzyć się z szeregiem praktycznych zadań, związanych z materiałem wykładu.

Wykłady będą uzupełnione notatkami, które proszę systematycznie przeglądać i korzystać z gęsto zamieszczonych w nich odnośników, stanowiących zachętę do samodzielnego poszukiwania i poszerzania wiedzy. Listy zadań będą publikowane w formie osobnych dokumentów.

2 Tło technologiczne ASP.NET

Architektura aplikacji internetowej obejmuje dwa główne komponenty i ścisły podział między nimi – to [przeglądarka internetowa](#) oraz [serwer aplikacji](#).



[Technologie dynamicznego WWW](#) – to żargonowe określenie dotyczy technologii, w których serwer aplikacji realizuje coś więcej niż tylko odczyt statycznych plików i wysyłanie ich do przeglądarki. Jeśli serwer realizuje ten sam protokół komunikacyjny (HTTP) ale potrafi wykonać **kod**, który produkuje odpowiedź na żądanie klienta, to znacząco zmieniają się możliwości aplikacji, w szczególności – istnieje możliwość dostarczenia użytkownikowi spersonalizowanej zawartości, w tym np. wymuszanie zalogowania się użytkowników i udostępnianie im ich własnych danych.

Współczesne technologie wytwarzania aplikacji internetowych mocno rozdzielają obszary wytwarzania aplikacji po stronie serwera i po stronie przeglądarki. Technologie serwerowe to prawdziwy [przekrój języków i platform technologicznych](#) natomiast po stronie przeglądarki wszystkie aplikacje, bez względu na wybór sposobu ich wytwarzania, muszą normalizować się do trzech wspólnych elementów:

- [HTML](#)
- [CSS](#)
- [JavaScript](#)

2.1 CGI

Za protoplastę technologii ASP.NET można uznać pierwsze próby dynamicznego WWW czyli technologię [CGI](#) i jej wprowadzenie do serwerowej linii Windows (wtedy jeszcze Windows NT 3.51) w ramach serwera aplikacyjnego, [Internet Information Services](#) (IIS) w 1995 roku.

CGI pozwala na pisanie skryptów w dowolnym języku, w którym istnieje dostęp do strumienia wejścia i wyjścia (na przykład w C++). Serwer aplikacyjny przetwarza żądanie, buduje z niego zbiór argumentów, uruchamia zewnętrzny proces skryptowy (na przykład plik wykonywalny), odbiera od niego ze strumienia wyjściowego odpowiedź i odsyła do przeglądarki.

Najprostszy skrypt CGI, napisany w języku C mógłby wyglądać tak:

```
#include <stdio.h>

int main( int argc, char** argv )
{
    printf( "HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n" );
    printf( "<HTML>\r\n<HEAD>" );
    printf( "<TITLE>Witam w CGI</TITLE></HEAD>\r\n" );
    printf( "<BODY>Pierwszy skrypt w CGI</BODY>\r\n" );
    printf( "</HTML>" );

    return 0;
}
```

Taki sposób budowania aplikacji internetowej ma swoją cenę – uruchamianie zewnętrznych procesów jest kosztowne. Ponadto uruchamiane procesy nie współdzielą pamięci, CGI skaluje się więc słabo.

2.2 Internet Server Application Programming Interface (ISAPI)

Aby pokonać problemy związane z wydajnością CGI, Microsoft zaprojektował alternatywną technologię dynamicznego WWW, nazwaną [Internet Server Application Programming Interface](#) (ISAPI). Główny pomysł polegał na tym, że skrypty ISAPI są bibliotekami (DLL) a nie modułami wykonywalnymi, dzięki czemu kod skryptu ładowany jest do pamięci tylko raz.

Istnieją dwa rodzaje bibliotek ISAPI: **rozszerzenia ISAPI**, które spełniają identyczną funkcję jak skrypty CGI oraz **filtry ISAPI**, które reagują na pewne zdarzenia związane z obsługą stron przez serwer.

Mimo, że technologia ISAPI jest zdecydowanie wydajniejsza od CGI, nie jest pozbawiona wad. Po pierwsze, napisanie poprawnej biblioteki ISAPI wymaga zdecydowanie więcej wiedzy niż napisanie skryptu CGI. Po drugie, jeśli biblioteka ISAPI trafi już na serwer Internetowy, to nie ma łatwego sposobu na zastąpienie jej nowszą wersją, ponieważ system operacyjny zabroni dostępu do biblioteki, która wedle jego rozeznania będzie cały czas używana. Wymiana biblioteki wymaga więc zatrzymania usługi serwera Internetowego na serwerze sieciowym.

2.3 Active Server Pages (ASP)

Następcą ISAPI jest technologia [Active Server Pages](#), która, o dziwo, jest zaimplementowana jako **rozszerzenie ISAPI**. W przypadku ASP nie tworzy się jednak żadnej biblioteki, tylko zwykłą stronę HTML, zaś wewnątrz jej kodu umieszcza się dowolne instrukcje języka skryptowego, VBScript. ASP sam dba o interpretowanie kodu VBScript i odsyła do klienta wyniki tej operacji.

Oto przykład bardzo prostej strony ASP:

```
<% Option Explicit %>
<HTML>
<HEAD><TITLE>Witam w ASP</TITLE></HEAD>
<BODY>
<%
Dim n
For n = 1 to 5
    Response.Write( "<FONT size=" & n )
    Response.Write( ">Witam w ASP</FONT><br>" & vbCrLf )
Next
%>
</BODY>
</HTML>
```

Aby strona internetowa była interpretowana jako strona ASP, wystarczy nadać jej rozszerzenie **asp**.

Projektując strony ASP można korzystać z całej siły VBScript. Ale to właśnie siła VBScript ta okazuje się być największą słabością ASP - VBScript, jak przystało na język skryptowy, jest bardzo słabo otypowany. Co więcej – kod jest interpretowany dynamicznie. Oba te fakty oznaczają, że bardzo łatwo popełniać błędy w skryptach, które jeśli się pojawią, to wykrywane są dopiero wtedy, kiedy natrafi na nie pierwszy użytkownik.

2.4 Czym jest ASP.NET

Technologia [ASP.NET](#) jest naturalnym rozszerzeniem ASP, które integruje technologię ASP z platformą .NET. Dzięki ASP.NET możliwe jest używanie praktycznie dowolnego języka platformy .NET do tworzenia dynamicznej zawartości stron WWW.

Od premiery w 2001 roku, technologia ASP.NET zyskiwała kolejne podsystemy, które omówimy w trakcie naszego wykładu

- Podsystem WebForms
- Podsystem [ASP.NET MVC](#)
- Podsystem [WCF](#)
- Podsystem WebAPI

W 2016 roku miała miejsce premiera alternatywnej implementacji środowiska ASP.NET, czyli [ASP.NET Core](#), uruchamianego w ramach platformy [.NET Core](#).

2.5 Pierwszy przykład ASP.NET

Najprostszy przykład dynamicznej strony ASP.NET ukazuje jednocześnie, że ASP.NET umożliwia użycie C# jako języka skryptowego. Przy próbie uruchomienia kod strony będzie prekompilowany, a błędy będą statycznie raportowane osobie testującej.

```
<%@ Page Language="C#" %>
<HTML>
<HEAD><TITLE>Witam w ASP.NET</TITLE></HEAD>
<BODY>

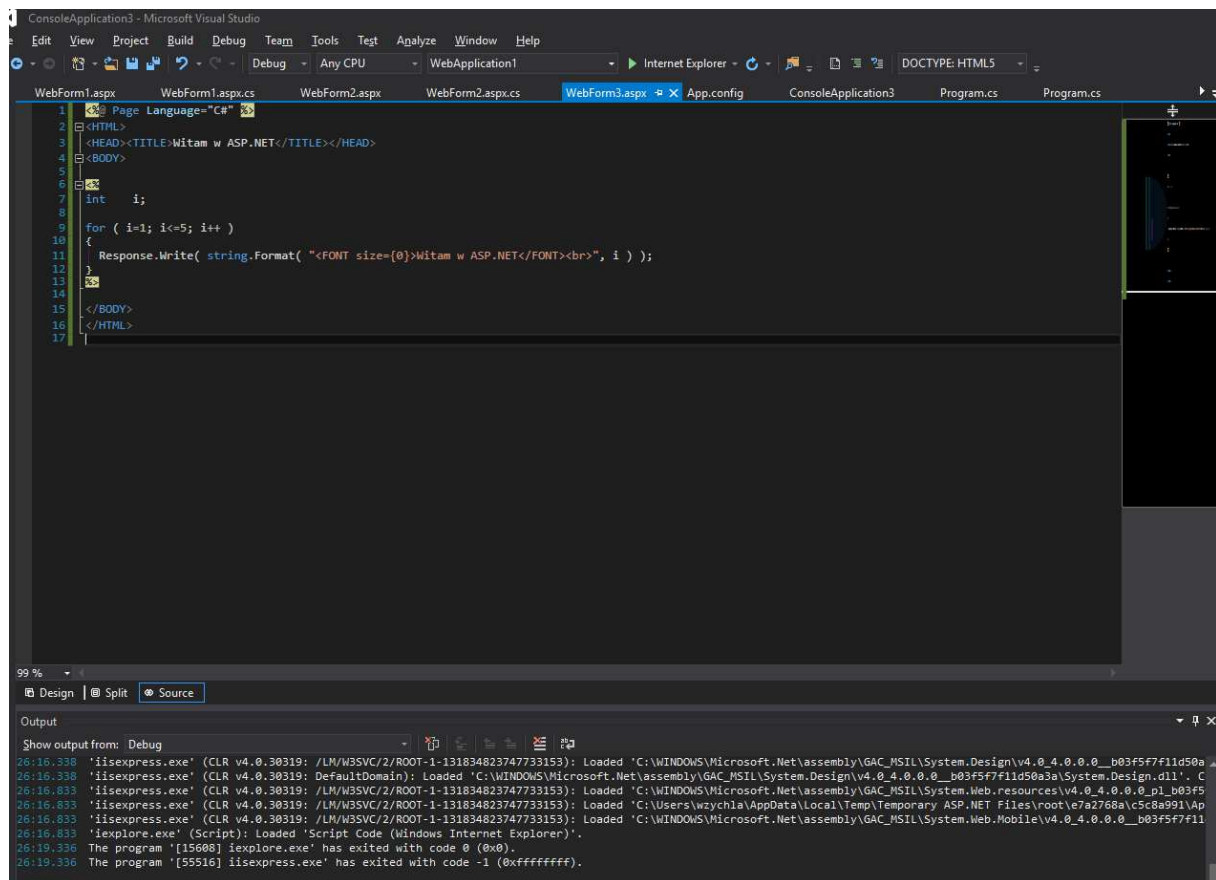
<%
int    i;

for ( i=1; i<=5; i++ )
{
    Response.Write( string.Format( "<FONT size={0}>Witam w ASP.NET</FONT><br>", i ) );
}
%>

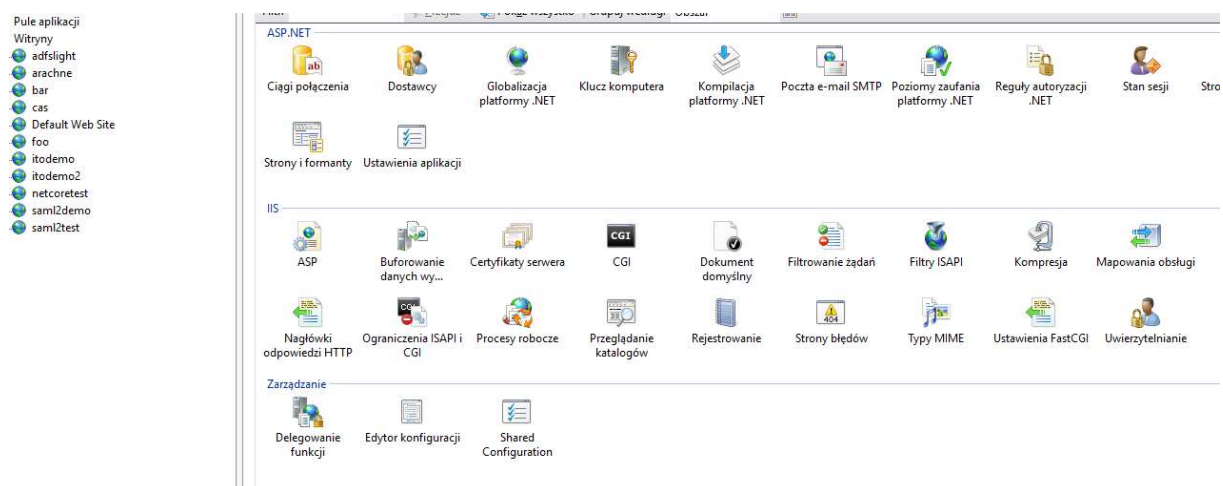
</BODY>
</HTML>
```

3 Wprowadzenie do narzędzi

W trakcie pierwszego wykładu omówimy i zaprezentujemy warsztat narzędziowy, niezbędny do rozwijania aplikacji ASP.NET, w szczególności – [Visual Studio](#) (w wersji co najmniej 2015).



Omówimy architekturę serwera IIS, nauczymy się poruszać po jego konfiguracji



Opowiemy o

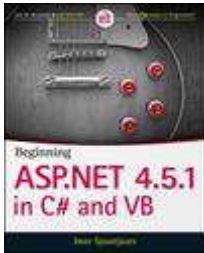
- Witrynach
- Pulach aplikacji
- Konfiguracji wiązania witryn – w tym o tym jak przeglądarka decyduje o tym gdzie wysłać żądanie i jak to możliwe że serwer obsługuje wiele aplikacji na tym samym **porcie** HTTP

- Konfiguracji uprawnień pul aplikacji

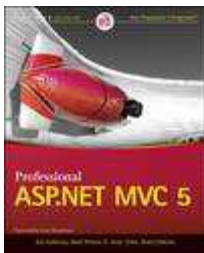
Zobaczmy też scenariusze deploymentu aplikacji, czyli o tym jak przenosić aplikacje ze środowiska deweloperskiego na docelowy serwer fizyczny, na którym pracuje serwer aplikacyjny.

4 Wybrana literatura

Spaanjaars, Beginning ASP.NET 4.5.1 in C# and VB



Galloway, Wilson, Allen, Matson - Professional ASP.NET MVC 5



Lowy, Programming WCF

