Projektowanie aplikacji ASP.NET
Wykład 01/15 - Wprowadzenie

Wiktor Zychla 2024/2025

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 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 zamieszczonych w nich odnośników, stanowiących zachętę do samodzielnego poszukiwania i poszerzania wiedzy. Listy zadań będą publikowane w formie osobnych dokumentów.

W trakcie wykładu będziemy bardzo mocno używać języka C# i elementów środowisk .NET.Framework i .NET. Jeżeli ktoś z Państwa **nie wysłuchał wykładu Programowanie pod Windows .NET**, prowadzonego w semestrze letnim, to **proszę rozważyć powrót do ASP.NET w przyszłości**, po Programowaniu pod Windows – gdzie jest okazja do gruntownego zapoznania się z językiem i środowiskiem.

Uzupełnieniem wykładu będą listy zadań. **Termin ważności listy zadań** to data laboratorium na którym należy zadeklarować zadania. Termin ważności wyznacza **tydzień** dla wszystkich dni tygodnia, w których odbywają się laboratoria a nie ścisłą datę. Przykładowo – na pierwszej liście zadań mamy adnotację:

> Zestaw ważny do: 22.10.2024
>
> Oznacza to że
>
> - grupy poniedziałkowe prezentują ten zestaw 21.10
> - grupy wtorkowe 22.10
> - grupy środowe 23.10
> - grupy czwartkowe 24.10
> - grupy piątkowe 25.10

2 Historia

ASP.NET zostało włączone do pierwszej wersji platformy .NET w roku 2002. Przełomowe wersje to wersja 2 z roku 2005 i wersja 4 z roku 2010. Aktualna wersja 4.8 pochodzi z roku 2019.

Równolegle rozwija się projekt .NET (wcześniej NET.Core), próbujący rozwiązać problem wieloplatformowości środowiska .NET, ostatnia wersja 8 pochodzi z 2023 roku.

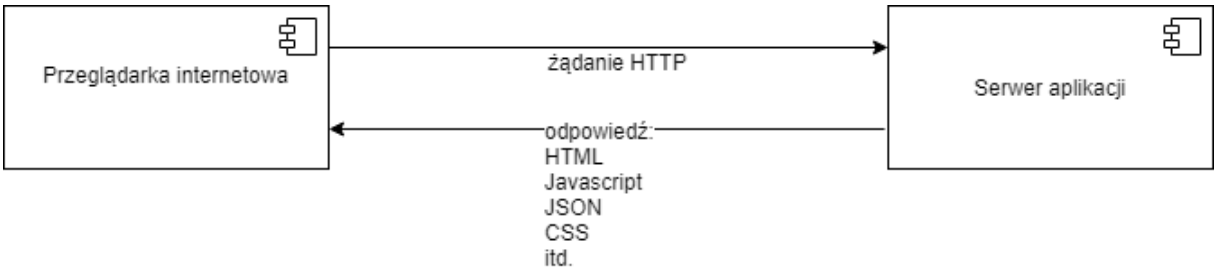
W przyszłości należy spodziewać się długiego okresu wsparcia dla .NET Framework oraz corocznego udostępniania wersji .NET.

Oba środowiska różnią się zasadniczo architekturą przetwarzania żądań na serwerze i o tym będziemy mówić w trakcie wykładu. Architektura ASP.NET.Framework oparta jest o tzw. **moduły i handlery**, architektura .NET oparta jest o tzw. **middleware**.

Mimo tych różnic w sposobie przetwarzania żądań, obie platformy są zasadniczo zbieżne jeśli chodzi o dostępne podsystemy (m.in. MVC, WebAPI, WCF), stąd można myśleć o wymienności i swobodzie wyboru konkretnej wersji platformy do konkretnych zastosowań.

3 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

3.1 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.

3.2 Interner 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 Inernetowego na serwerze sieciowym.

3.3 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 bibilioteki, 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.

3.4 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

Wraz z pojawieniem się wieloplatformowego środowiska .NET, dodano m.in.

- Podsystem Razor Pages
- Podsystem Blazor
- Podsystem gRPC

Każdy z tych podsystemów służy do wytwarzania innego typu aplikacji i każdy należy poznać. Dodatkowo w trakcie wykładu omówimy elementy architektury aplikacji:

- Autentykacja/autoryzacja
- Stos aplikacyjny używający bazy danych
- Technologia ClickOnce

3.5 Pierwszy przykład ASP.NET - .NET Framework

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>
```

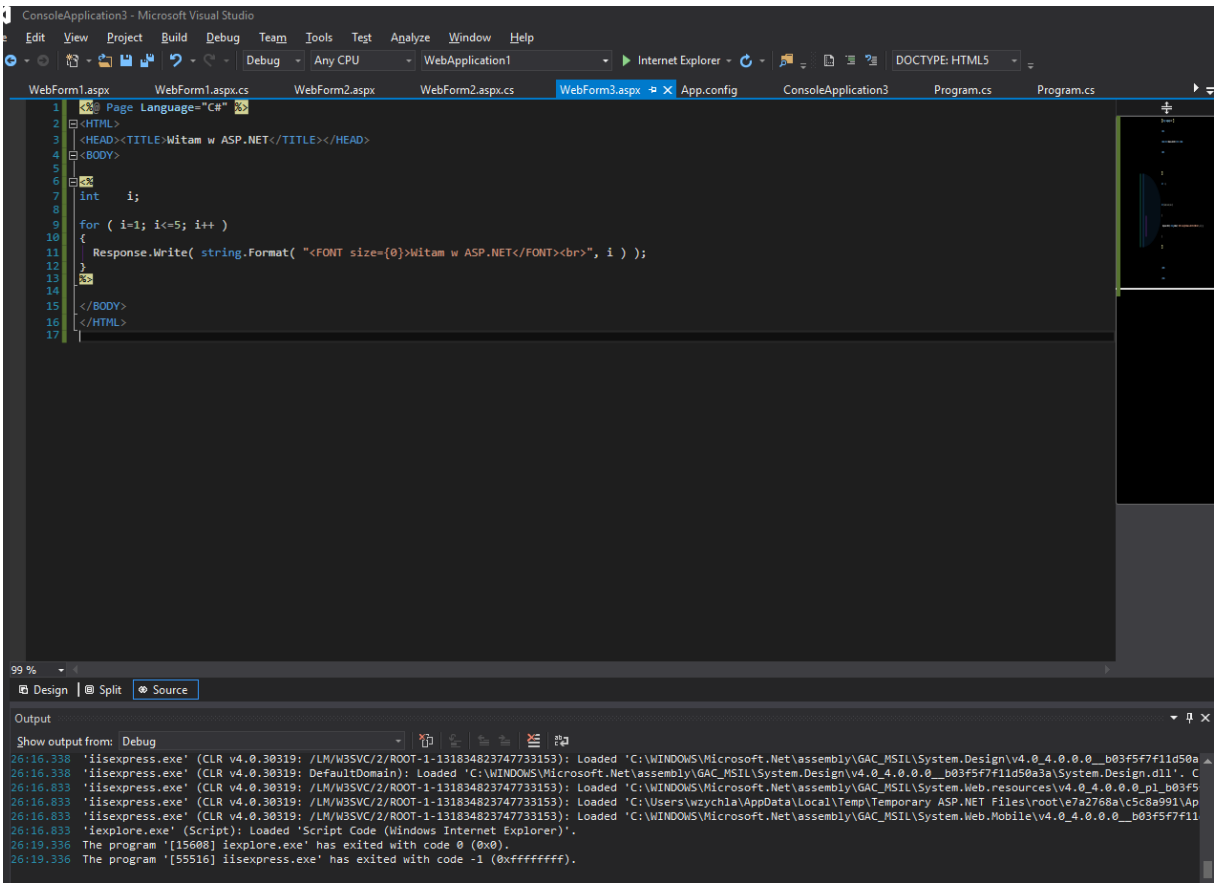
3.6 Drugi przykład - .NET 8

Na przeciwległym biegunie możliwości leży przykład usługi udostępniającej punkt końcowy HTTP, wykonany przy pomocy tzw. Minimal API

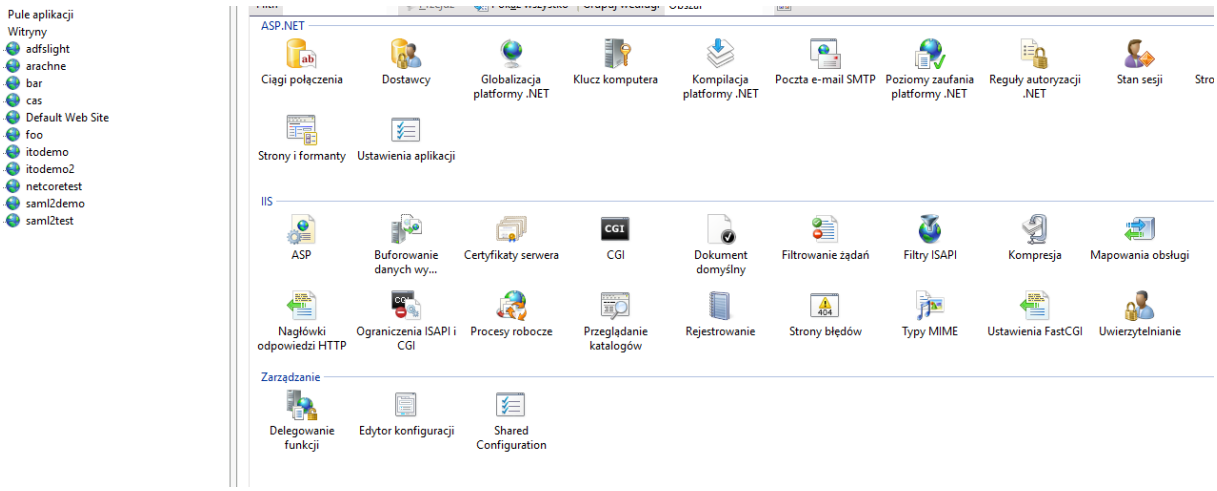
```
namespace WebApplication4
{
    public class Program
    {
        public static void Main( string[] args )
        {
            var builder = WebApplication.CreateBuilder(args);
            var app = builder.Build();

            app.MapGet( "/", () => "Hello World!" );

            app.Run();
        }
    }
}
```

4 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 2022).



Omówimy architekturę serwera IIS, nauczymy się poruszać po jego konfiguracji



Opowiemy o

- Uprawnieniach wymaganych przez serwer aplikacji do zasobów (plików aplikacji)
- 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

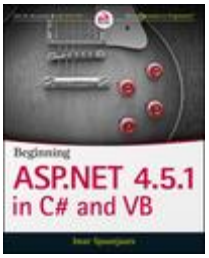
Zobaczymy też podstawowy scenariusz deploymentu aplikacji, czyli o tym jak przenosić aplikacje ze środowiska deweloperskiego na docelowy serwer fizyczny, na którym pracuje serwer aplikacyjny.

Scenariusz będzie dotyczył aplikacji ASP.NET wytworzonej dla .NET Framework, o deploymencie aplikacji .NET >=5 opowiemy na jednym z kolejnych wykładów.

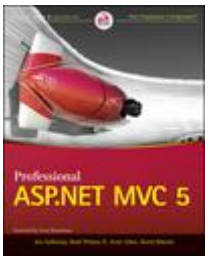
5 Wybrana literatura

ASP.NET Framework

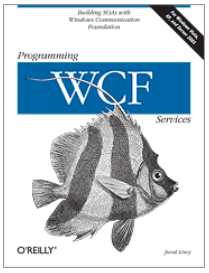
Spaanjaars, Beginning ASP.NET 4.5.1 in C# and VB



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



Lowy, Programming WCF



Esposito – Programming ASP.NET Core (Developer Reference)