Politechnika Wrocławska

Platformy programistyczne

Omówienie technologii ASP.NET

Dr inż. Radosław Idzikowski

Katedra Automatyki, Mechatroniki i Systemów Sterowania
Wydział Informatyki i Teleinformatyki

19 kwietnia 2024

HR EXCELLENCE IN RESEARCH

ASP.NET

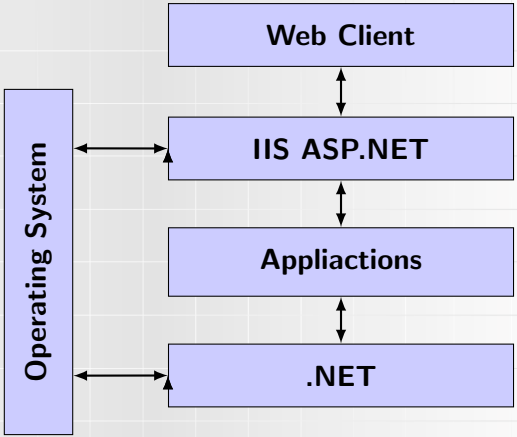
ang. *Active Server Pages Network Enabled Technologies*

- środowisko projektowania dynamicznych aplikacji WWW,
- platforma typu open-source,
- rozdzielenie warty prezentacji i logiki,
- działa po stronie serwera,
- kod jest kompilowany przy pierwszym żądaniu strony.

IIS

Internet Information Services (IIS) to rozszerzalny serwer WWW obsługujący protokoły:

- HTTP,
- HTTP/2,
- HTTPS,
- FTP,
- FTPS,
- SMTP.
- NNTP.



Elementy ASP.NET

- formularze Internetowe (**.aspx*),
- usługi Web (`*.asmx`),
- pliki logiki aplikacji (`*.vb` albo `*.cs`),
- globalnej klasy aplikacji (`.asax`),
- plików konfiguracyjnych (`Web.config`),
- pliki stron (`*.html`),
- style (`*.css`).

Wydania ASP.NET

Platforma	rok wydania	
ASP.NET 1.0	Styczeń 2002	Web Forms
ASP.NET 1.1	Kwiecień 2003	Wsparcie dla mobilności
ASP.NET 2.0	Listopad 2005	Wsparcie dla 64-bitowych systemów
ASP.NET 3.5	Listopad 2007	WCF Integration i AJAX
ASP.NET 4.0	Kwiecień 2010	Dynamic Data
ASP.NET 4.5	Sierpień 2012	One ASP.NET
ASP.NET 4.5.1	Październik 2013	Bootstrap
ASP.NET 4.5.2	Maj 2014	Wsparcie dla TLS 1.2
ASP.NET 4.6	Lipiec 2015	HTTP/2
ASP.NET Core 1.0	Czerwiec 2016	Nowa architektura modułowa
ASP.NET Core 1.1	Listopad 2016	Wsparcie dla .NET Standard 1.6
ASP.NET Core 2.0	Wrzesień 2017	.NET Standard 2.0
ASP.NET Core 2.1	Maj 2018	SignalR
ASP.NET Core 2.2	Grudzień 2018	Health Checks
ASP.NET Core 3.0	Wrzesień 2019	Endpoint Routing
ASP.NET Core 3.1	Grudzień 2019	Blazor Server
ASP.NET Core 5.0	Listopad 2020	Blazor WebAssembly

MVC vs Blazor

W ramach platformy ASP.NET są dostępne różne technologie do tworzenia systemów webowych. Dwa najbardziej ppopularne to:

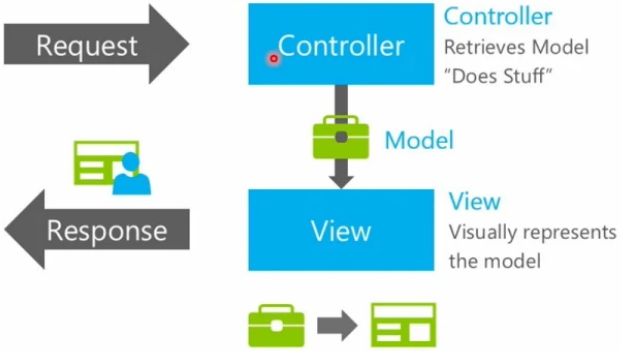
- MVC with Razor Pages,
- Blazor.

MVC

`Model-View-Controller`, to wzorzec architektury oprogramowania, który separuje logikę aplikacji na trzy powiązane ze sobą komponenty: Model, Widok i Kontroler. Ten wzorzec został szeroko zaakceptowany w rozwoju aplikacji internetowych ze względu na czytelną separację zadań oraz łatwość utrzymania. W architekturze MVC, Model reprezentuje dane i logikę biznesową, Widok definiuje interfejs użytkownika, a Kontroler obsługuje wejście użytkownika i aktualizuje Model oraz Widok odpowiednio.

Models, Views, and Controllers

What does MVC look like?



MVC

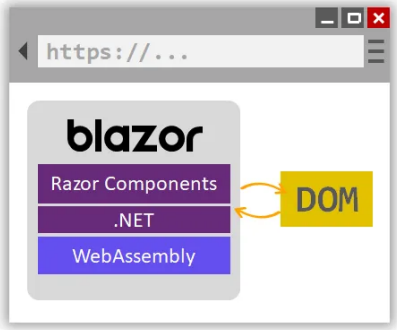
Cykl żądanie-odpowiedź w MVC zapewnia, że wejście użytkownika wywołuje odpowiednie akcje, dane są przetwarzane i aktualizowane, a zaktualizowane dane są prezentowane użytkownikowi. Taka separacja obowiązków ułatwia modularność kodu, jego testowanie i utrzymanie.

Blazor

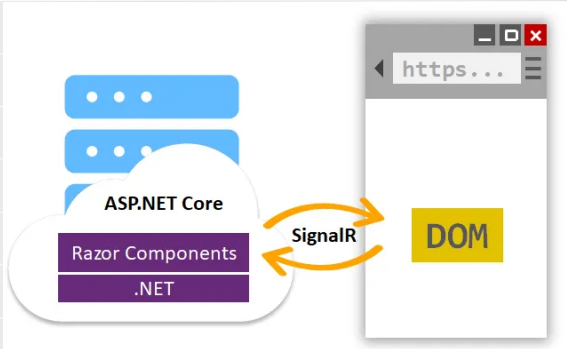
Opracowany przez firmę Microsoft, to darmowy i otwarto źródłowy framework internetowy, który umożliwia programistom budowanie interaktywnych aplikacji internetowych przy użyciu języka C# zamiast polegania głównie na JavaScript. Blazor wykorzystuje WebAssembly, format instrukcji binarnych, który umożliwia uruchamianie kodu napisanego w różnych językach w przeglądarkach internetowych.
Blazor oferuje dwa modele hostowania: Blazor WebAssembly i Blazor Server. W modelu WebAssembly, cała aplikacja jest wykonywana po stronie klienta w przeglądarce, podczas gdy w modelu Server, logika aplikacji działa po stronie serwera, a interfejs użytkownika jest renderowany i aktualizowany w przeglądarce przy użyciu SignalR.

Blazor

Blazor WebAssembly



Blazor Server



Blazor

Zarówno `Blazor WebAssembly`, jak i `Blazor Server` umożliwiają programistom pisanie kodu *C#* zarówno po stronie klienta, jak i po stronie serwera. Oferują funkcje takie jak renderowanie komponentów, wiązanie danych oraz komunikację z interfejsami `API`, co pozwala na rozwijanie bogatych, interaktywnych aplikacji internetowych przy użyciu potęgi języka `C#`.

MVC

Zalety

- Dobrze ugruntowany wzorzec: MVC dostarcza sprawdzony wzorzec architektury do strukturyzowania aplikacji, ułatwiając utrzymanie kodu i testowanie.
- Elastyczność i możliwość dostosowania: MVC oferuje drobiazgową kontrolę nad zachowaniem i wyglądem aplikacji, umożliwiając tworzenie wysoko dostosowanych aplikacji internetowych.
- Silne wsparcie społeczności: MVC posiada dużą i aktywną społeczność, zapewniającą obfite zasoby, dokumentację oraz wsparcie udzielane przez społeczność.
- Przyjazne dla SEO adresy URL: Aplikacje MVC generują czyste i przyjazne dla wyszukiwarek internetowych adresy URL, co ułatwia działania optymalizacji dla wyszukiwarek.
- Integracja z systemami dziedzictwa: MVC doskonale nadaje się do integracji z istniejącymi systemami dziedzictwa lub bazami danych, zapewniając elastyczność w pracy z różnymi źródłami danych.

MVC

Wady

- Zwiększona złożoność przy większych aplikacjach: Aplikacje `MVC` mogą stać się złożone w miarę wzrostu ich rozmiaru i złożoności, wymagając starannego zarządzania zależnościami.
- Dłuższy czas rozwoju: Dostosowanie i drobiazgowa kontrola w `MVC` może prowadzić do dłuższych cykli rozwoju w porównaniu z prostszymi frameworkami.
- Ograniczone aktualizacje w czasie rzeczywistym: Osiągnięcie aktualizacji w czasie rzeczywistym i dynamicznych zmian w interfejsie użytkownika może wymagać dodatkowej złożoności i użycia bibliotek takich jak `SignalR` w `MVC`.

Blazor

Zalety

- Jednolity język programowania w *C#*: Aplikacja `Blazor` umożliwia programistom korzystanie z języka *C#* zarówno po stronie klienta, jak i serwera, promując współdzielenie kodu i jego ponowne wykorzystanie.
- Bogate i interaktywne interfejsy użytkownika: Aplikacje `Blazor` umożliwiają tworzenie dynamicznych i angażujących interfejsów użytkownika przy użyciu języka *C#* oraz składni `Razor`.
- Pełna integracja z ekosystemem `.NET`: Programiści mogą wykorzystać obszerną kolekcję bibliotek kodu `.NET`, frameworków i narzędzi dla zwiększenia produktywności.
- Poprawiona wydajność (`Blazor WebAssembly`): Aplikacje `Blazor WebAssembly` działają bezpośrednio w przeglądarce, co skutkuje szybszymi czasami ładowania i zmniejszeniem żądań do serwera.
- Możliwości rozwoju na wielu platformach: `Blazor WebAssembly` wspiera wdrożenie na różnych platformach, poszerzając zasięg aplikacji.

Blazor

Wady

- Krzywa nauki: Jako stosunkowo nowy `framework`, `Blazor` może wymagać od programistów inwestowania czasu w naukę jego koncepcji, składni i najlepszych praktyk.
- Ograniczone wsparcie przeglądarek (`Blazor WebAssembly`): Starsze przeglądarki bez wsparcia dla `WebAssembly` mogą nie być w stanie uruchomić aplikacji `Blazor WebAssembly`, co może wpłynąć na zasięg publiczności.
- Większe rozmiary plików i dłuższe czasy ładowania (`Blazor WebAssembly`): Aplikacje `Blazor WebAssembly` wymagają pobrania plików środowiska uruchomieniowego i aplikacji, co skutkuje większymi rozmiarami plików i dłuższymi czasami ładowania podczas inicjalizacji.

Porównanie

Architektura

`MVC` stosuje dobrze ugruntowany wzorzec architektoniczny, dzieląc logikę aplikacji na trzy komponenty: Model, Widok i Kontroler. Promuje on separację obowiązków i zapewnia strukturalne podejście do rozwoju.

`Blazor` wprowadza architekturę opartą na komponentach, w której komponenty interfejsu użytkownika są tworzone przy użyciu języka `C#` i składni `Razor`. Łączy on korzyści obu podejść: po stronie klienta i po stronie serwera.

Porównanie

Język i narzędzia

`MVC` głównie wykorzystuje język `C#` do logiki po stronie serwera oraz `HTML`, `CSS` i `JavaScript` do części front-endowej. Posiada on szeroki zestaw narzędzi oraz dojrzały ekosystem do tworzenia aplikacji internetowych.

`Blazor` pozwala programistom pisać zarówno logikę po stronie klienta, jak i po stronie serwera, używając języka `C#`. Zapewnia on zjednoczony model programowania do rozwoju zarówno front-endu, jak i back-endu, co redukuje potrzebę przełączania się między różnymi językami.

Porównanie

Wydajność

Aplikacje `MVC` zazwyczaj polegają na renderowaniu po stronie serwera, gdzie serwer generuje `HTML` i wysyła go do klienta. Ten sposób może prowadzić do dłuższych czasów ładowania aplikacji i zwiększonej liczby żądań do serwera dla dynamicznych treści.

`Blazor` oferuje dwa tryby - `Blazor WebAssembly` i `Blazor Server`. `Blazor WebAssembly` działa po stronie klienta w przeglądarce, umożliwiając szybsze czasy ładowania i zmniejszenie liczby żądań do serwera. `Blazor Server` opiera się na komunikacji w czasie rzeczywistym z serwerem, zapewniając responsywną interakcję z użytkownikiem.

Porównanie

Proces tworzenia

`MVC` zapewnia dojrzały wzorzec rozwoju, szerokie narzędzia i dużą społeczność. Programiści mogą wykorzystać istniejące biblioteki i frameworki, przyspieszając rozwój i rozwiązywanie problemów.

Architektura oparta na komponentach `Blazora` promuje ponowne wykorzystanie kodu i modularność, ułatwiając tworzenie złożonych elementów interfejsu użytkownika. Integracja z ekosystemem `.NET` pozwala programistom korzystać z istniejących bibliotek i narzędzi.

Porównanie

Wsparcie przeglądarek

Aplikacje `MVC` mają szeroką kompatybilność przeglądarek, ponieważ polegają na standardowym `HTML`, `CSS` i `JavaScript`.

`Blazor WebAssembly` wymaga wsparcia dla `WebAssembly` w nowoczesnych przeglądarkach. Starsze przeglądarki mogą być niekompatybilne, co ogranicza zasięg publiczności dla aplikacji `Blazor WebAssembly`.

Podsumowanie

Ostateczny wybór między `MVC` a `Blazorem` zależy od czynników takich jak wymagania projektu, wiedza zespołu i rozważania dotyczące wydajności. `MVC` jest solidnym wyborem dla tradycyjnego renderowania po stronie serwera i ustalonych praktyk rozwoju. `Blazor` z kolei oferuje nowoczesne i zintegrowane doświadczenie rozwoju z mocą języka *C#* zarówno po stronie klienta, jak i serwera.