Technologie internetowe
ASP.NET Core

Paweł Rajba
pawel@cs.uni.wroc.pl
http://pawel.ii.uni.wroc.pl/

Plan wykładu

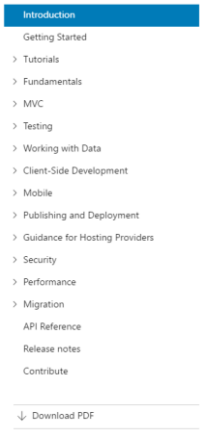
- Wprowadzenie
- Podstawowa aplikacja
- Usługi i middleware
- Obsługa błędów
- Request, Response, ciastka i sesje
- MVC
  - Routing, kontrolery, modele, widoki
  - Nawigacja, Formularze, Razor
- Podsumowanie

Wprowadzenie

- Najnowsze rozwiązanie od Microsoftu do tworzenia aplikacji WWW
- Tworzone na zasadach OpenSource
- Możliwość uruchomienia i tworzenia na Windows, Mac, Linux
  - Główne narzędzia od Microsoft to
    - Visual Studio i
    - Visual Studio Code

Wprowadzenie

- Więcej detali technicznych
  - Prosty i przejrzysty mechanizm obsługi żądań
    - Stosuje mechanizm middleware
  - Wbudowane mechanizmy dependency injection
  - Niezależność od środowiska hostingowego
    - Niemniej, wbudowana integracja z IIS
  - Zależności zarządzane przez NuGet-a
- Sporo materiałów wprowadzających
  - http://docs.asp.net/



Uruchomienie przykładów

- Do pobrania:
  - https://www.microsoft.com/net/download/dotnet-core/2.1
- Polecenia:
  - dotnet build
  - dotnet run

Podstawowa aplikacja

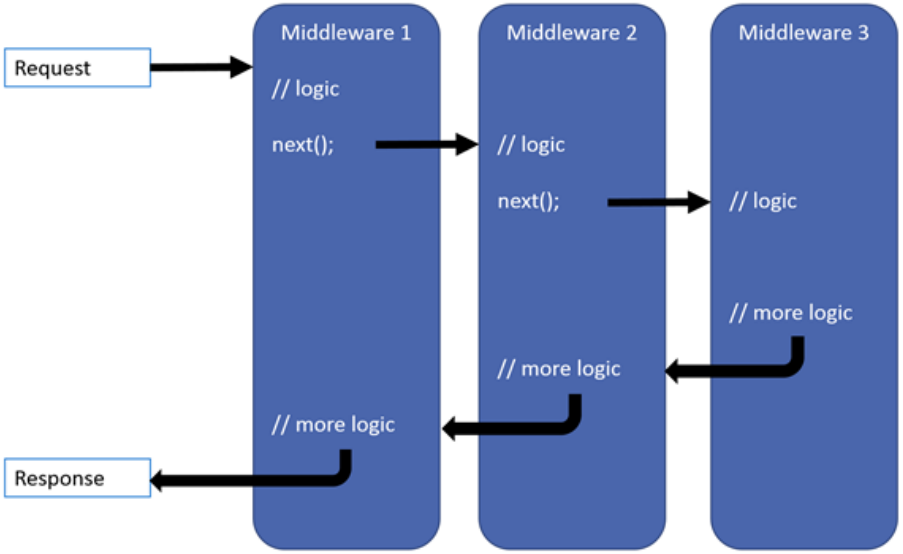
- Główne składowe:
  - Plik Startup.cs
    - ConfigureServices
      - Lista obiektów „services” do wykorzystania w aplikacji
    - Configure
      - Koncept middleware do obsługi żądania
  - Plik project.json
  - Folder wwwroot
- Warto zwrócić uwagę
  - Project w VS jest 1-1 z systemem plików

DEMO

- EmptyApplication

Usługi i middleware

- Sposób działania middleware



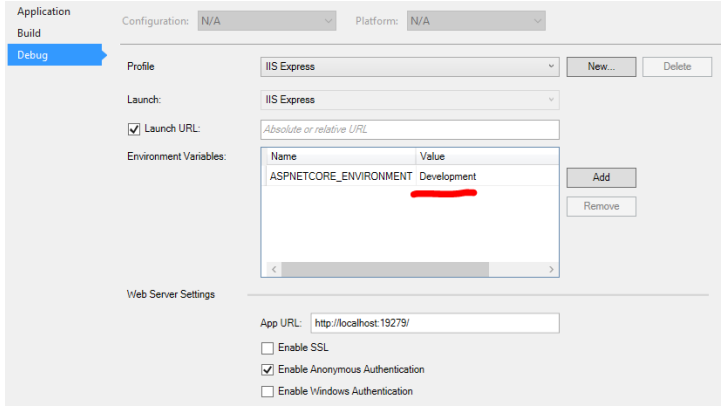
- Sporo wbudowanych
  - Logging, Error handling, Static file server
  - Authentication, MVC, WelcomePage ☺
- Przykładowy potok: Logging, Errors, AuthN, MVC

Usługi i middleware

- Dodanie obsługi plików statycznych
  - Znaczenie folderu wwwroot
  - W konfiguracji startup.cs możemy dodać:
    - app.UseStaticFiles();
      - Dostępne stają się pliki z wwwroot
    - app.UseDefaultFiles();
      - Wpisanie URL / powoduje szukanie pliku index.html, …
    - app.UseDirectoryBrowser();
      - Pozwala na przeglądanie pliku
    - app.UseFileServer();
      - Udostępnia wszystkie powyższe

Obsługa błędów

- Określenie środowiska
  - Development, Production, …
- Obsługa błędów
  - Pełna informacja (development)
  - Ograniczona obsługa
    - Przekierowanie
      - Np. do /error
    - Własna funkcja obsługi
      - Np. logging a potem przekierowanie



Request, response, ciastka i sesje

- Obiekt context (HttpContext) ma obiekty
  - Request
  - Response
  - Session
- Request i Response mają kolekcje Cookies

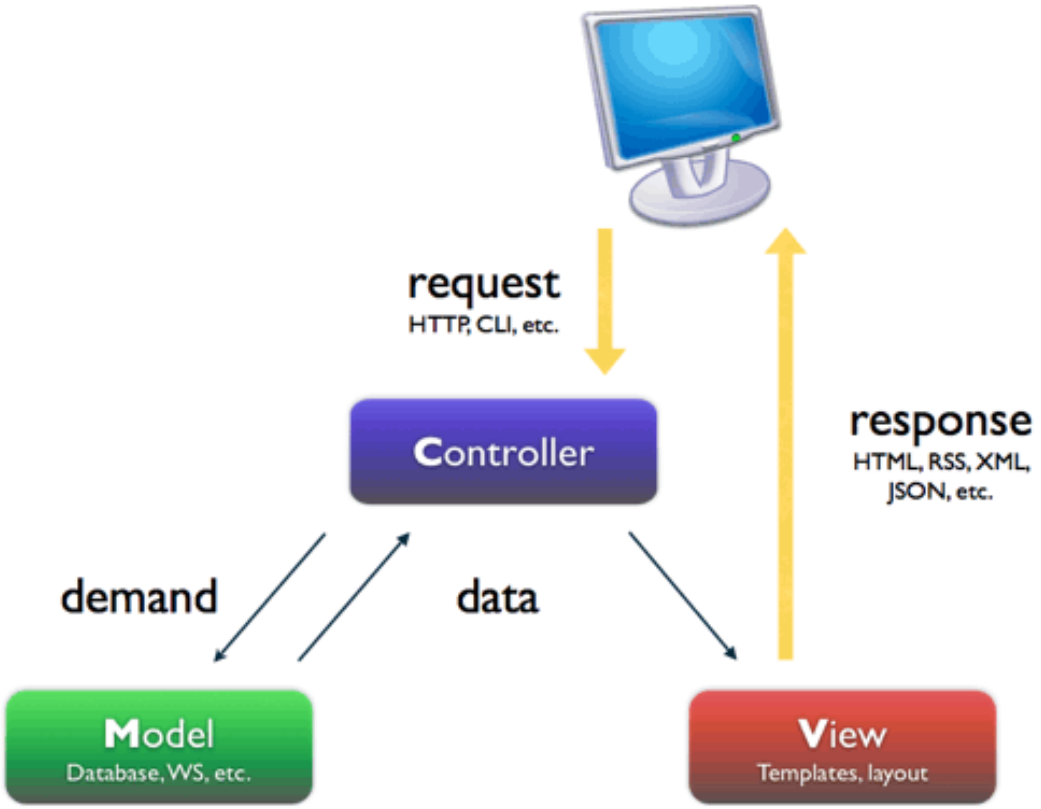
DEMO

- FirstExample

MVC

- Popularny wzorzec architektoniczny
- Składowe
  - Model danych (model)
  - Warstwa prezentacji (widok)
  - Logika sterowania (kontroler)
- Cel: oddzielenie warstw
- Uwaga: MVC to tylko interfejs użytkownika
  - Nie powinno być w nim logiki biznesowej

MVC



Źródło: http://www.indiaosl.com/blog/wp-content/uploads/2010/01/mvc.png

MVC

- Dodajemy do konfiguracji
  - app.UseMvcWithDefaultRoute()
- Dodajemy service
  - services.AddMvc();
- Dodajemy odpowiedni kontroler

DEMO

- FirstRouting

MVC: Więcej o routingu

- Metody
  - app.UseMvc(configureRoutes) i
  - configureRoutes.MapRoute(…)
- Określenie
  - Różnych ścieżek
  - Parametrów domyślnych i opcjonalnych
  - Walidacja parametrów
- Definicja routing kontrolerze przez atrybut [Route(…)]

MVC: Więcej o kontrolerze

- Dodanie dziedziczenia z klasy Controller daje dużo więcej możliwości
- Dostęp do HttpContext
- Typ IActionResult i implementujące go
  - ContentResult, FileResult, UnauthorizedResult
  - JsonResult, RedirectResult, ViewResult
- … oraz zwracające powyższe metody
  - Content, File, Unauthorized, Json, Redirect, View

Przekazywanie danych do widoku

- Kolekcje ViewData i ViewBag
  - ViewData
    - Kontroler: ViewData["Message"]="Hello World!";
    - Widok: @ViewData["Message"]
  - ViewBag
    - Kontroler: ViewBag.Message= "Hello World!";
    - Widok: @ViewBag.Message
- Model
  - Kontroler: View(model)
  - Widok
    - Dyrektywa: @model MvcExample.Models.Person
    - Zmienna @Model

Modele

- Rodzaje modeli
  - Business Object
  - MvcObject (DTO)
  - PresentationModel (ViewModel)
- Co jest w MVC i kiedy co stosować?

Nawigacja

- Html.ActionLink
- Html.RouteLink
- Url.Action
- Url.RouteUrl

DEMO

- MvcExample

Formularze

- Główna konstrukcja
  - @using (Html.BeginForm())
    { … }
- Helpery
  - @Html.LabelFor(e => e.FirstName)
  - @Html.EditorFor(e => e.FirstName)
  - @Html.ValidationMessageFor(e => e.FirstName)
- Atrybut [HttpPost]
- Właściwość ModelState
- Przestrzeń nazw
  - System.ComponentModel.DataAnnotations

DEMO

- FormSimple
- FormModel
- FormValidation

Razor

- Razor, czyli język szablonów
  - Co to jest i po co to?
- Podstawowe elementy
  - Views/Shared
    - Współdzielone szablony, w tym szablony główne
  - Plik Views/_ViewStart.cshtml
    - Plik wykonywany przed przetworzeniem każdego widoku
    - Domyślne ustawienia, w tym główny szablon
    - Szablon można też wskazać w widoku

```
@{
    Layout = "~/Views/Shared/_Layout2.cshtml";
}
```

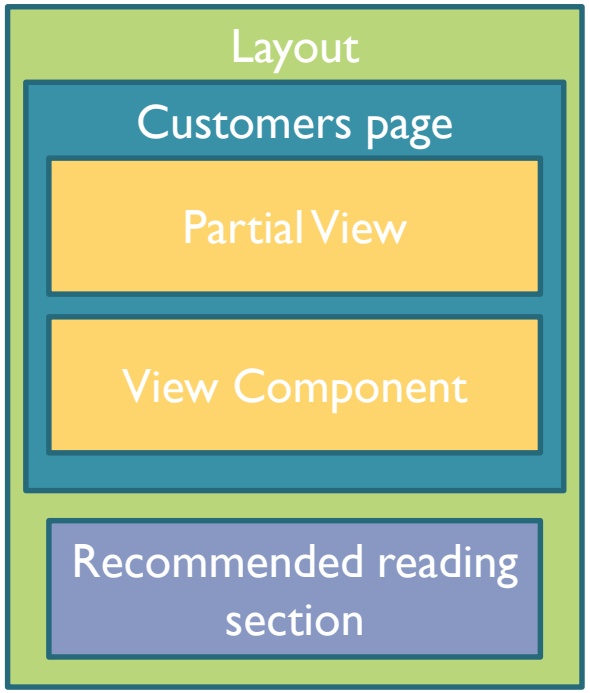
  - Plik Views/_ViewImports.cshtml
    - Import przestrzeni nazw
  - Szablon główny
    - Zwykle Views/Shared/_Layout.cshtml

Razor

- Helpery
  - Html.*
    - @Html.Action, @Html.Route
    - @Html.CheckBox, @Html.Editor, @Html.Label
    - I wiele innych…
  - TagHelpers
    - Jak powyższe, ale składnia typu HTML
    - „Stare” i nowe elementy, atrybuty typu asp-*
    - Przykładowo:
      - <a asp-action="Edytuj" asp-route-id="@itemId">Edytuj
      - <environment names="Development">…</environment>
      - <label>, <input>, , <cache …>
      - I wiele innych

Razor

- Podstawowe kontrukcje
  - RenderBody
  - RenderSection
- Zagnieżdżanie widoków
  - Partial views
    - Html.Partial
  - View components
    - Component.InvokeAsync



Razor

- Kilka wybranych konstrukcji
  - Blok kodu

```
@{
    string s = "this is string";
}
```

  - Zmienna kodowana i niekodowana

```
@Model.Message

@Html.Raw(Model.Message)
```

  - Pętla i IF

```
@foreach(var item in items) {
  <span>@item.Prop</span>
}

@if (foo) {
  @:Plain Text is @bar
}
```

DEMO

- Razor Templates

Podsumowanie

- Na przykładzie ASP.NET Core zobaczyliśmy, że można na różne sposoby tworzyć dynamiczne treści
  - Generowanie treści bezpośrednio w kodzie
  - Prezentacja plików
  - MVC
    - Zwracanie samych danych lub obiektów
    - Zwracanie całych widoków HTML

Podsumowanie

- Większość aplikacji składa się z podobnych elementów:
  - Kontrolki, prezentacja danych
  - Formularze
  - Dostęp do bazy danych
  - Sterowanie dostępem
  - Logging
  - Integracja
- Istotne jest, aby do powyższego wykorzystać biblioteki i gotowe rozwiązania