Jazyk C# 2

8. seminář

Radek Janoštík

Univerzita Palackého v Olomouci

12. 4. 2024

Reakce na úkoly

- Rozdělení kroků: Načtení - Parsování - Vložení. $\rightarrow$ Lepší hledání chyb.
- Někdy zbytečné joiny (přitom navázané entity dostupné přes `.`)
- (Ne)provázaní entit - degradace vazby 1:N na 1:1
- SaveChanges() po každém příkazu - na konci ”transakce”

Webový vývoj – základy

- Jaké máte zkušenosti s vývojem webových aplikací?

Webový vývoj – základy

- Jaké máte zkušenosti s vývojem webových aplikací?
- Klient-server architektura
- Backend a frontend aplikace
- HTTP požadavky

Webový vývoj – základy

- Jaké máte zkušenosti s vývojem webových aplikací?
- Klient-server architektura
- Backend a frontend aplikace
- HTTP požadavky
- Základní typy webových aplikací
  - Statické webové stránky
  - Dynamické (generované na serveru)
  - Dynamické s AJAX
  - *Single Page Application(SPA)*
  - Kombinace

Webový vývoj – základy

- Jaké máte zkušenosti s vývojem webových aplikací?
- Klient-server architektura
- Backend a frontend aplikace
- HTTP požadavky
- Základní typy webových aplikací
  - Statické webové stránky
  - Dynamické (generované na serveru)
  - Dynamické s AJAX
  - *Single Page Application(SPA)*
  - Kombinace
- Dnes si ukážeme vývoj (klasických) dynamických webových stránek pomocí ASP.NET

ASP.NET

- Rozšíření .NET pro webové prostředí
- Využití standardních knihoven(řetězce, IO, databáze) a C# pro programování webů
- Backend: klasický C# jak jej známe
- ASP.NET přináší:
  - Framework pro zpracování webových requestů
  - Šablonovací systém pro FrontEnd (`Razor`)
  - Knihovny pro standardní webové postupy (MVC, API)
  - Zabezpečení (Auth, HTTPS, . . .)
- Ukážeme si základ MVC Frameworku

MVC architektura

- Obecný programovací model
- (Poměrně) striktní oddělení vrstev aplikace
- Usnadňuje orientaci v kódu, hledání chyb
- Význam zkratky
  - M – Models
  - V – Views
  - C – Controllers

MVC architektura – Models

- Modely – (Datové) objekty implementující datovou logiku aplikace
- Velmi často získávání a ukládání dat do DB
- Pro nás budou ztotožněny s entitami EntityFramworku
- Příklad:
  - Produkt
  - Student
  - Předmět
  - ...

MVC architektura – Views

- Views – komponenty zobrazující UI aplikace
- Pohledy, Výjevy
- UI se vytváří na základě dat z modelu
- Přesný popis toho, co a jak se má uživateli zobrazit
- V ASP.NET - popis, šablona webové stránky

MVC architektura – Controllers

- Kontrolery – objekty zajišťujíící interakci s uživatelem
- Manipulují s modely, zajišťují funkcionalitu
- Vytváří Views pro vyrenderování
- Celková aplikační/procesní logika

ASP.NET MVC Execution Process

- Aplikace neběží samostatně, ale ve webovém serveru, který odstiňuje aplikaci od síťové vrstvy
- Proces práce aplikace (zkráceně):
  - HTTP request na webový server $\rightarrow$ naše aplikace
  - Routování – zjištění, co je potřeba a která část aplikace to má dělat
  - Vytvoření MVC request handleru (přístupný v RequestContext – surovější, servisní data requestu)
  - Vytvoření Controlleru, nastavení RequestContextu
  - Zavolání konkrétní akce controlleru
  - Zobrazení výsledku (ViewResult,Redirect,JsonResult,...)

MVC aplikace – vytvoření

- Nový projekt → C# → Web → ASP.NET Core Web Application (Model-View-Controler)
- Struktura:
  - Adresáře `Controllers`, `Models` a `Views` pro příslušné objekty
  - Adresář `wwwroot` pro statická data (css, js). Výchozí *Bootstrap, jquery*
  - `appsettings.json` – nastavení aplikace
  - `Program.cs` – spouštěcí soubor dříve s metodou `Main`
    - runtime konfigurace projektu (dříve v metodách)
    - více `https://docs.microsoft.com/en-us/aspnet/core/fundamentals/startup`

Jednoduchá evidence studentů

- Pomocí NuGet package manageru doinstalujeme EntityFramework
- Vytvoříme příslušné modely (Student, Address, Subject)
- Vytvoříme předka pro kontrolery, který bude obsahovat Context
- Vytvoříme nový kontroler `StudentController`
- `return View();` vrátí defaultně View pojmenovaný jako akce (dle složky)
- Při spuštění aplikace se nám otevře webový prohlížeč
- Volání akcí: URL/JmenoKontroleru/JmenoAkce
- např.: `http://localhost:61317/Home/Index` Spustí HomeController s akcí Index (default)

Zobrazení všech studentů

- Do `StudentController` přidáme akci `All`
- Ta z databáze vybere všechny studenty a předá je View

```
1 public IActionResult All() {
2           List<Student> students = Ctx.Students.ToList();
3           return View(students);
4 }
```

- Ve složce Views/Student vytvoříme All.cshtml – náš View, nastavíme model

```
1 @model List<WebApplication1.Models.Student>
```

- `ViewBag` Dynamický kontejner pro „další data“

Views – Razor

- Šablonovací jazyk pro renderování dynamických Views – mix HTML a C#
- Důležitý znak: @ – způsobuje vyhodnocení

```
@{ var title = "Titulek stranky";
   var date = DateTime.Now;
   var message = " Today is: " + date; }
```

- Inline vyhodnocení

```
<p>Name: @Model.Name</p> <a href=".../@Model.Id">DELETE</a>
```

- Výrazy:

```
 @if (ViewBag.Neco) {
   <a href="odkaz vedouci nekam">Odkaz</a>
 } else {
 <p>Nic!</p>
 }
```

- Tedy vhodnou kombinací C# a HTML se dají vytvářet stránky reagující na obsah

Razor – Automatická kompilace Views

- Ve starém MVC se rekompilace děla automaticky bez nutnosti restartu aplikace
- Nyní potřeba doinstalovat `Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation`
- V `Program.cs` změnit `builder.Services.AddControllersWithViews();` na:

```
1  builder.Services.AddControllersWithViews()
2  .AddRazorRuntimeCompilation();
```

- Views by se měly dynamicky rekompilovat po změně bez restartu aplikace

Razor – cykly

- `for`, `while`, `do ... while` jak je známe

```
@for(var i = 0; i < Model.Kolekce.Length; i++)
{
    <p>Item @i: @Model.Kolekce[i]</p>
}
```

- Procházení kolekcí pomocí `foreach`

```
@foreach (Student s in Model.Students)
{
<tr><td>@s.Name</td><td>@s.Surname</td></tr>
}
```

- Občas trochu zmatek, kdy použít `@` a kdy ne

Views – Razor – Helpery

- Helpery – pomocné funkce v `Razor` pro práci s HTML, URL, ...
- Dokumentace: `https://learn.microsoft.com/en-us/aspnet/core/mvc/views/tag-helpers`
- Mohou vás (téměř) zcela odstínit od HTML
- Hůře se kombinují s Frontend frameworky (JQuery, KnockoutJS, ...)
- Např.: `asp-action`, `asp-route-id`

```
1 <td><a asp-action="Edit" asp-route-id="@s.Id">Editovat</a></td>
```

Editace studenta (1/2)

- Views/Student → Add View
- Pojmenujeme `Edit`, jako Template zvolíme `Edit` a zvolíme třídu `Student`
- Visual studio vygeneruje `Razor` šablonu dle vlastností třídy
- Přídáme akci `Edit` kontroleru `Student` s parametrem `int id` a anotací `HttpGet`
- Najdeme studenta a vrátíme jej jako model do View

```
1 [HttpGet]
2 public ActionResult Edit(int id) {
3 Student s = Ctx.Students.FirstOrDefault(p => p.Id == id);
4 if (s!=null) {
5   return View(s);
6 } else {
7   return NotFound("Student not found");
8 }
9 }
```

Editace studenta (2/2)

- Vznikl formulář, který se namapoval na hodnoty vlastností třídy `Student`
- Cílem formuláře je opět akce `Edit` – vytvoříme další s anotací `HttpPost`

```
1  [HttpPost]
2  public ActionResult Edit(Student studentEdited) {
3  Student s = Ctx.Students.FirstOrDefault(p => p.Id ==
       studentEdited.Id);
4  if (s != null) {
5     s.Name = studentEdited.Name;
6     s.Surname = studentEdited.Surname;
7     s.Rocnik = studentEdited.Rocnik;
8     Ctx.SaveChanges();
9     return RedirectToAction("All");
10 } else {
11    return NotFound("Student not found");
12 }
13 }
```

Disclaimer

- Předchozí slidy nemají být rozsáhlým průvodcem do ASP.NET MVC
- Spíše slouží jako úvod do problematiky a „co vlastně existuje“
- Byly vynechány nějaké základní principy, jako:
  - Odchytávání výjimek
  - DTO
  - Bezpečnost
  - Anotace u modelů (restrikce pro generování View)
  - Deploy v IIS
  - Autentizace
  - ...
- Téma je rozsáhlé, nejde stihnout vše – je lepší zažít

Úkol 1/2

- Webová aplikace pro útulky pro psy:
  - Úvodní strana budou různé útulky
  - Každý útulek po rozkliknutí zobrazí svůj detail - seznam psů
  - Každý útulek na detailové stránce bude mít možnost přidat nového psa (tlačítkem zobrazí formulář k přidání) - po přidání se zobrazí znovu seznam i s novým psem
  - U psů bude možné nahrát/editovat jeho fotografii
  - U každého psa se mohou psát komentáře
  - V Layoutu bude okénko hledání psa podle jména
  - Pes po rozkliknutí bude mít tlačítko ”adoptovat”, které smaže záznam z databáze
  - V databázi budou persistentně ukládány záznamy o útulcích a o psech. Útulek musí vědět, jaké má psy. Ve výsledcích vyhledávání zobrazte i v jakém útulku pes je.
  - Zkuste se zamyslet, jaké informace chcete uchovávat o daných entitách (Název, popis, umístění, kapacita, ...)(Jméno, věk, rasa, popis, ...)

Úkol 2/2

- MVC architektura tedy bude mít:
  - Pro každou entitu umožněte vkládání/editaci/mazání
  - (Doporučení) Alespoň 3 controllery: HomeController pro úvodní fce, ShelterController, DogController
- Úkol za 6 bodů
  - Snažte se i rozumný vzhled a použitelnost
- Bonusové 2 body: Zkuste implementovat přihlášení a role uživatelů (útulek, zájemce o adopci) s příslušnými oprávněními
- Na úkol máte dva týdny – odevzdání tedy 24. 4.
- Příští seminář – nepovinné konzultace zde v učebně v čase semináře