Tehnici Avansate de Programare

ASP.NET Core MVC — Introducere

Petru Rebeja, Marius Apetrii

30 Aprilie 2020

Facultatea de Matematică
Universitatea Alexandru Ioan Cuza, Iași

Introducere

Recapitulare

Recapitulare

- Baza de date este o colecție organizată de date care pot fi manipulate prin intermediul unui Sistem de Gestiune al Bazelor de Date.

Recapitulare

- Baza de date este o colecție organizată de date care pot fi manipulate prin intermediul unui Sistem de Gestiune al Bazelor de Date.

- Schema bazei de date este reprezentarea structurii bazei de date. Visual Studio ne pune la dispoziție un șablon de proiect pentru păstrarea și modificarea schemei.

Recapitulare

- Baza de date este o colecție organizată de date care pot fi manipulate prin intermediul unui Sistem de Gestiune al Bazelor de Date.
- Schema bazei de date este reprezentarea structurii bazei de date. Visual Studio ne pune la dispoziție un șablon de proiect pentru păstrarea și modificarea schemei.
- Un proces de dezvoltare implică:
  - O listă a sarcinilor de lucru,
  - Un flux de lucru,
  - Un minim necesar de documentație și
  - Revizuirea periodică a priorităților.

Agenda

- Șablonul `MVC`
- Introducere în `ASP.NET Core MVC`

Șablonul MVC

MVC

Model-View-Controller este un șablon de proiectare utilizat pentru a decupla interfața grafică (`view`), datele (`model`) și logica aplicației (`controller`)[1].

[1] `https://dotnet.microsoft.com/apps/aspnet/mvc`

Componente

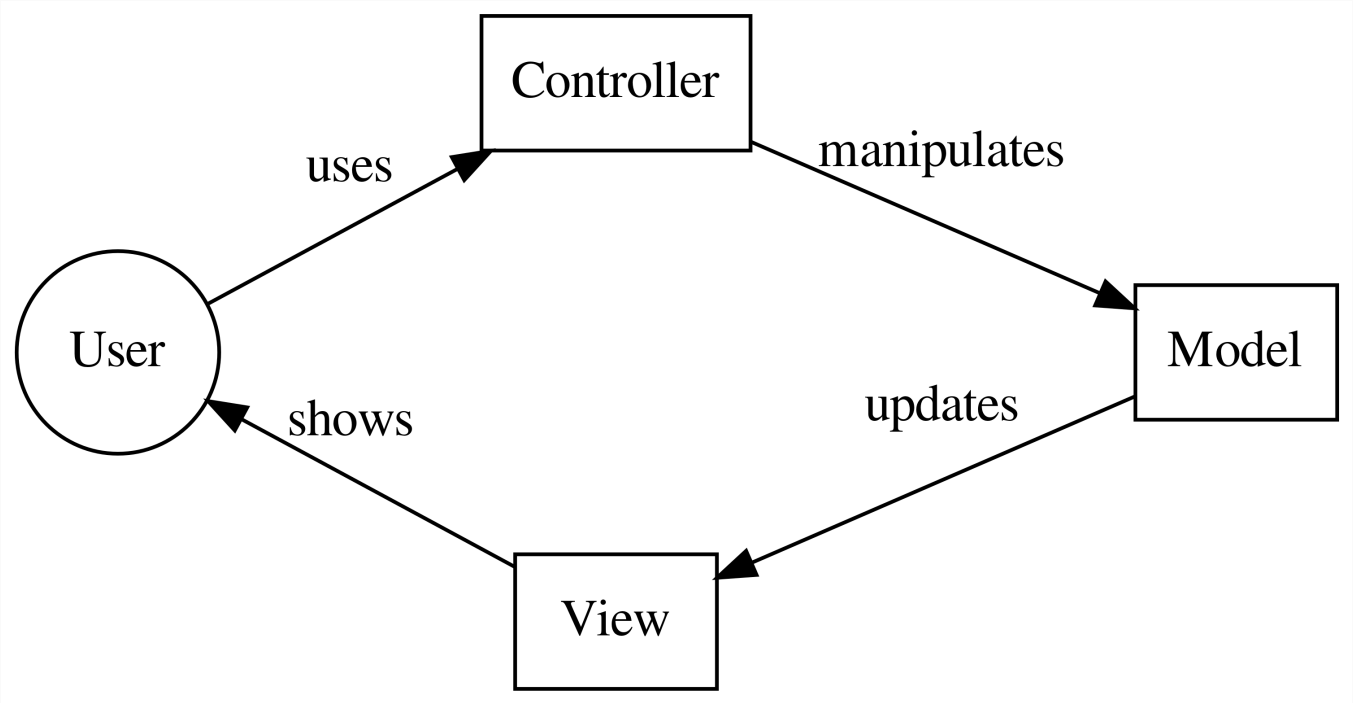
- `Modelul` — încapsulează datele și logica aplicației; este independent de interfața cu utilizatorul[2].
- `Viewul` — prezintă datele din `model` în diverse formate; același `model` poate avea mai multe `viewuri`.
- `Controllerul` — convertește datele de intrare în comenzi pentru `Model` sau `View`[3] și le validează dacă este nevoie.

---

[2] Burbeck, Steve (1992) Applications Programming in Smalltalk-80:How to use Model–View–Controller (MVC)

[3] `https://www.codeproject.com/Articles/25057/Simple-Example-of-MVC-Model-View-Controller-Design`

Model-View-Controller



Interacțiunea dintre `model`, `view` și `controller` în raport cu acțiunile inițiate de utilizator. Imagine adaptată după[4].

[4] https://en.wikipedia.org/wiki/Model-view-controller

Fluxul de lucru

1. Utilizatorul inițiază o acțiune (click pe buton/request http).
2. `Controllerul` procesează acțiunea printr-o metodă dedicată.
3. `Controllerul` notifică `modelul` existent sau crează unul nou.
4. `Modelul` își schimbă starea (dacă este cazul).
5. `Viewul` generează interfața grafică pe baza modelului.
6. Interfața generată așteaptă alte acțiuni din partea utilizatorului.

Avantaje[5]

- Permite dezvoltarea în paralel; ex. câte un programator per componentă.
- Grad ridicat de coeziune — metodele care manipulează același `model` pot fi grupate în același `controller`; la fel și `viewurile`.
- Grad scăzut de acuplare — obținut prin separarea în 3 componente.
- Componente ușor de modificat.
- Componentele pot fi testate independent.

[5] `https://en.wikipedia.org/wiki/Model-view-controller`

Dezavantaje[6]

- Mult cod de umplutură.
- Cod greu de navigat din cauza separării.
- Necesită disciplină și exercițiu — separarea pe componente nu este intuitivă; în cazuri extreme logica este dispersată în toate cele trei componente.

[6] https://en.wikipedia.org/wiki/Model-view-controller

ASP.NET Core MVC

ASP.NET Core MVC[7]

- Platformă open source de dimensiuni mici.
- Suportă cele mai noi standarde web.
- Suportă dezvoltarea în stilul TDD.

[7] https://docs.microsoft.com/en-us/aspnet/core/mvc/overview

Funcționalități suportate

- `Routing`
- `Model binding`
- Validarea datelor
- Filtre
- `Dependency Injection`
- Modularizare (`Areas`)

Routing

- Apelează o anumită metodă din `controller` pe baza unui `URL` asociat acesteia.
- Asocierile se fac:
  - pe baza convențiilor
  - cu atribute dedicate.

Model binding

- Transformă datele dintr-un `request http` în instanțe ale tipurilor definite de utilizator.
- Instanțele rezultate sunt pasate metodelor din controller ca parametri.

Validarea datelor

Validarea se poate face prin:

- Atribute — pe client și server.
- Metode expuse de biblioteci terțe, ex: `FluentValidation`[8] — pe server.

[8] `https://docs.fluentvalidation.net/en/latest/aspnet.html`

Filtre

Filtrele permit invocarea anumitor metode înainte/după executarea unei anumite părți din logica aplicației.

Exemple:

- Restricționarea accesului la o anumită metodă,
- Gestiunea erorilor etc.

Dependency Injection

ASP.NET Core MVC conține și un modul propriu de Dependency Injection dar permite și integrarea de biblioteci terțe specializate (ex. AutoFac[9]) .

[9] https://autofaccn.readthedocs.io/en/latest/integration/aspnetcore.html

Areas

- Oferă posibilitatea de a separa aplicația în mai multe module.
- Fiecare modul (`area`) conține structura convențională a unei aplicații (`modele`, `viewuri`, `controllere`).

Exemple de module:

- `Account` — modificarea datelor utilizatorului; preferințe etc.
- `Accounting` — facturi, plăți, bilanțe contabile etc.
- `Public` — toate acțiunile disponibile utilizatorilor neautentificați.

Încheiere

Recapitulare

- MVC este un șablon de proiectare utilizat pentru a decupla interfața grafică (`view`), datele (`model`) și logica aplicației (`controller`).
- ASP.NET Core MVC este o platformă `open-source` care permite dezvoltarea de aplicații Web pe baza convențiilor asociate șablonului `MVC`.

Vă mulțumesc!

Mulțumesc pentru atenție!