Dezvoltarea Aplicațiilor Web utilizând ASP.NET Core MVC
Curs 1

Cuprins

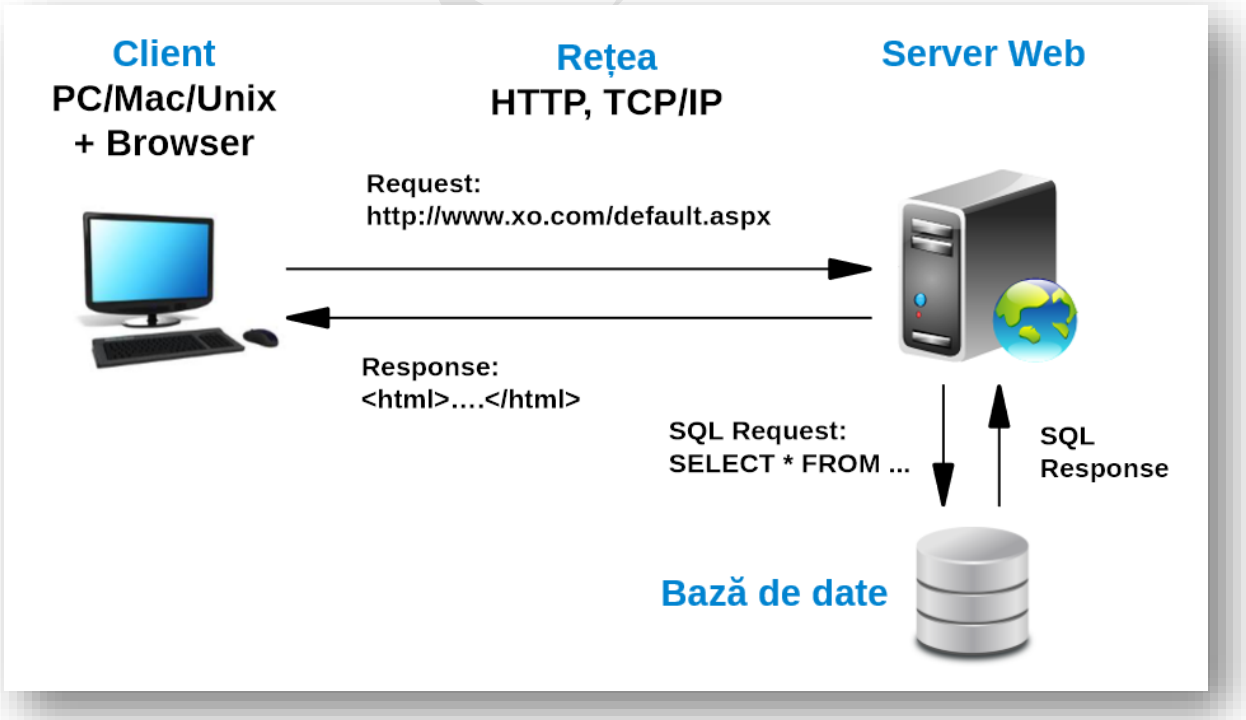
Ce este o aplicație Web

O aplicație web este o aplicație care rulează într-o arhitectură **Client–Server**, bazată pe: protocolul **HTTP** (HyperText Transfer Protocol), **TCP/IP** (Transmission Control Protocol/Internet Protocol), un **browser web** și un **server web**.

Aplicațiile web sunt executate într-un browser și sunt implementate folosind tehnologii precum PHP, ASP.NET, Python, HTML, CSS, JavaScript etc.

Aplicațiile web permit interacțiunea directă cu utilizatorul printr-o interfață grafică accesibilă de pe orice dispozitiv conectat la internet. Ele sunt frecvent utilizate pentru platforme de e-commerce, sisteme educaționale online, aplicații de management și rețele sociale, oferind o experiență dinamică și actualizări în timp real.

Arhitectura Web



Avantajele aplicațiilor Web

- **Sunt independente de sistemul de operare**, ceea ce înseamnă că pot fi accesate de pe Windows, macOS, Linux sau chiar dispozitive mobile, fără diferențe majore de funcționalitate;

- **Nu necesită instalare**, deoarece aplicația rulează direct în browser, eliminând problemele legate de compatibilitate și spațiu de stocare;

- **Actualizările sunt foarte ușor de făcut**, deoarece modificările se realizează într-un singur loc, pe server, iar acestea se propagă automat pentru toți utilizatorii;
  În cazul aplicațiilor client–server clasice, interfața cu utilizatorul este asigurată printr-un program client instalat pe calculatorul fiecărui utilizator, iar orice modificare necesită reinstalarea aplicației pentru fiecare utilizator în parte.

- **Accesibilitate de oriunde**, cât timp există o conexiune la internet, utilizatorii pot accesa aplicația de pe orice dispozitiv;

- **Costuri reduse de mentenanță** și **distribuție**, deoarece nu este nevoie de actualizări manuale sau suport tehnic pentru fiecare client;

- **Scalabilitate ridicată**, fiind ușor de adaptat pentru un număr mare de utilizatori simultan;

- **Integrare facilă cu alte servicii online**, cum ar fi baze de date externe, servicii de plată, aplicații mobile sau API-uri;

- **Securitate centralizată**, deoarece toate datele și actualizările sunt gestionate de pe server, reducând riscul de atacuri locale asupra dispozitivelor utilizatorilor;

Introducere în ASP.NET

- **ASP.NET** este un framework Web, open source, conceput și dezvoltat de Microsoft, care face parte din framework-ul .NET;
- Este utilizat pentru a dezvolta aplicații și servicii web;
- Oferă o integrare foarte bună a codului HTML, CSS, JavaScript;
- Oferă posibilitatea creării paginilor dinamice, prin intermediul sintaxei Razor, utilizând C#, HTML, CSS și JavaScript;
- ASP.NET furnizează librării specifice dezvoltării aplicațiilor web, cum sunt cele pentru sistemul de autentificare (autentificare în mai mulți pași, autentificare utilizând componente 3-rd party, etc), cele pentru crearea și prelucrarea bazei de date, cele pentru lucrul cu fișiere, etc;
- Este construit pe baza **CLR (Common Language Runtime) – rulează cod compilat** și permite utilizatorilor să scrie cod folosind orice limbaj acceptat de framework-ul .NET;

Ce este CLR – Common Language Runtime?

Se ocupă de **execuția programelor C#**. Atunci când este compilat un program C#, rezultatul compilării nu este un cod executabil direct. În locul acestuia **se generează un fișier** care conține un tip de cod apropiat de codul mașinii, numit **limbaj intermediar** sau, pe scurt, **IL** (**Intermediate Language**).

Acest proces de compilare este realizat de un **compilator C#** (de exemplu, compilatorul integrat în Visual Studio). Rezultatul procesului este un fișier cu extensia **.exe** sau .**dll**, care conține codul intermediar împreună cu metadatele programului, adică acele informații despre clase, metode, variabile și referințe.

Pentru ca aplicația să poată fi executată, **CLR** (**Common Language Runtime**) intervine și, prin intermediul unui compilator denumit **JIT** (**Just in Time**), transformă codul intermediar (IL) în **cod nativ executabil**, adică în instrucțiuni specifice sistemului de operare și procesorului pe care rulează programul.

Procesul JIT are loc **în momentul rulării aplicației**, motiv pentru care se numește "Just in Time", compilarea având loc exact atunci când este necesar.

Pe lângă executarea codului, **CLR** oferă o serie de servicii esențiale care contribuie la performanța și siguranța aplicațiilor .NET:

- **Gestionarea memoriei prin Garbage Collector**, care eliberează automat memoria neutilizată;
- **Tratarea excepțiilor și erorilor** în mod unitar;
- **Securitatea codului**, asigurând că doar operațiile permise sunt executate;

Astfel, CLR este responsabil nu doar cu transformarea codului în instrucțiuni executabile, ci și cu gestionarea eficientă, sigură și performantă a rulării aplicațiilor.

Framework-ul .NET

- Este compatibil cu peste 20 de limbaje diferite, cele mai populare fiind C#, C++, Visual Basic, F#. În prezent, limbajul cel mai des folosit rămâne C# deoarece restul limbajelor de programare au dezvoltat librării similare cu .NET framework;
- Pune la dispoziție o colecție impresionantă de clase, organizate în biblioteci;
- Este construit din două entități importante:

1. **Common Language Runtime (CLR)**

   - mediul de execuție al programelor, fiind cel care se ocupă cu managementul și execuția codului scris în limbaje specifice .NET;

2. **Base Class Library**

   - Este biblioteca de clase .NET;
   - Acoperă o arie largă a necesităților de programare, incluzând **interfața cu utilizatorul**, **protocoale de conectare cu baza de date**, **accesarea datelor**;

ASP.NET Core

- Este noul framework creat de Microsoft, conceput pentru a permite dezvoltatorilor să construiască aplicații moderne, performante și scalabile.
- A fost proiectat să funcționeze independent de sistemul de operare (Windows, macOS, Linux), fiind astfel mult mai flexibil și rapid.
- ASP.NET Core este un framework complet nou, nu o continuare a versiunii anterioare ASP.NET 4.6, ci o platformă rescrisă de la zero, cu un design modular și open-source.
- Aduce îmbunătățiri semnificative în ceea ce privește securitatea, performanța și costurile de întreținere, oferind o arhitectură mai curată și mai ușor de testat.

- Permite dezvoltarea unei game variate de aplicații: aplicații web, API-uri REST, aplicații mobile, microservicii, aplicații desktop, machine learning și chiar jocuri (prin integrarea cu Unity sau alte framework-uri).

- Oferă hostare rapidă și ușoară în cloud, fiind optimizat pentru platforme precum Azure, AWS și Google Cloud.

- Este complet open-source și disponibil pe GitHub, ceea ce permite o dezvoltare colaborativă și actualizări frecvente.

- Folosește un pipeline middleware configurabil, care oferă control total asupra modului în care sunt procesate cererile HTTP.

- Suportă modelul MVC (Model–View–Controller), Razor Pages, Blazor (pentru aplicații interactive în browser, folosind C# în loc de JavaScript) și Minimal APIs, pentru aplicații ușoare și rapide.

- Dispune de un sistem de Dependency Injection integrat, ceea ce facilitează o arhitectură curată și testabilă.

Pe scurt, ASP.NET Core este un framework modern, modular și cross-platform, potrivit atât pentru aplicații enterprise complexe, cât și pentru proiecte mici, cu timp de dezvoltare scurt și performanță ridicată.

Introducere în C#

- Este un limbaj compilat;

- Este un limbaj orientat pe obiecte;

- Permite dezvoltarea de aplicatii industriale, durabile;

- A fost conceput ca un concurent pentru limbajul Java;

- Este derivat al limbajului C++;

Limbaj compilat vs limbaj interpretat

Limbaj compilat → codul scris, numit **cod sursă**, este translatat de către compilator într-un cod apropiat de nivelul mașinii, numit **cod executabil**. Atunci când aplicația trece de compilare fără erori de sintaxă, se va produce codul executabil, iar aplicația va putea fi rulată. (**Exemple** de limbaje compilate: C, C++, Rust, Go etc.)

Limbaj interpretat (la rulare) → cu ajutorul unui **interpretor** specific limbajului, fiecare linie de cod este interpretată chiar în momentul rulării, fiind transformată imediat în cod mașină și executată. (**Exemple** de limbaje interpretate: PHP, Ruby, Python.)

Limbajele compilate oferă **performanță mai mare și optimizare**, dar necesită un pas suplimentar de compilare.

Limbajele interpretate sunt **mai ușor de utilizat** pentru dezvoltarea rapidă și oferă feedback imediat asupra erorilor, însă sunt mai lente la execuție.

C# este considerat un **limbaj compilat**, deoarece codul sursă este transformat într-un **cod intermediar** printr-un proces de compilare. Totuși, C# implică și un **proces de interpretare și compilare JIT** (Just-In-Time), unde codul intermediar este executat și transformat în cod mașină de către **CLR** în momentul execuției.

În acest mod, **C# este considerat un limbaj hibrid**, ceea ce duce la creșterea portabilității și a performanței, deoarece codul intermediar poate rula pe orice sistem care are instalată platforma .NET.

În același timp, compilatorul JIT optimizează execuția în funcție de arhitectura hardware a mașinii pe care rulează aplicația.

Performanța este îmbunătățită deoarece:

- Compilatorul JIT optimizează codul în momentul rulării, ținând cont de arhitectura procesorului și resursele disponibile;
- Astfel, aplicația rulează mai eficient pe fiecare sistem, folosind la maximum capacitățile hardware;
- De asemenea, codul intermediar este verificat și optimizat pentru siguranță, reducând erorile și blocajele;

Ciclul de viață al unei pagini Web

Paginile ASP.NET rulează pe **serverul web Microsoft IIS (Internet Information Server)**. În urma prelucrării pe server, rezultă o pagină web **HTML**, care este trimisă către browserul utilizatorului.

Ciclul de viață al unei pagini Web ASP.NET are următorii pași:

- **Page request** (accesarea paginii) – acest pas are loc înaintea ciclului propriu-zis de viață, atunci când o pagină este cerută serverului de către client;
- **Start** – în acest stadiu se încarcă proprietățile paginii, cum ar fi obiectele Request și Response, după care se identifică tipul cererii (GET – pentru solicitarea de resurse, POST – pentru trimiterea de informații către server).
- **Initialization** (inițializare) – în acest pas se inițializează directivele și controalele, se stabilesc valorile implicite și se aplică codul din Master Page (dacă există).

- **Load** (încărcare) – în această fază, dacă cererea este de tip postback, controalele sunt încărcate cu valorile trimise anterior de utilizator (de exemplu, date introduse într-un formular).

- **Evenimentele Postback** – dacă cererea este de tip postback, se execută codul aferent evenimentelor (de exemplu, apăsarea unui buton). După executarea codului, se aplică mecanismele de validare a datelor.

- **Rendering** (afișarea paginii) – în acest pas se construiește pagina finală în format HTML pe server, care este apoi trimisă și afișată în browserul utilizatorului.

- **Unload** (eliberarea memoriei) – după ce pagina a fost transmisă utilizatorului, resursele alocate pentru procesarea acesteia sunt eliberate de către server, pentru a optimiza performanța.