**General C#**

1. Ce inseamna internal, protected, public, private?

**public** : public, clasă publică sau membru al unei clase. Un astfel de membru al clasei este accesibil de oriunde din cod, precum și din alte programe și întruniri.

**private** : clasa privată sau membru al clasei. Reprezintă opusul exact al modificatorului public. O astfel de clasă privată sau membru al clasei este disponibil numai din cod în aceeași clasă sau context.

**protejat** : un astfel de membru al clasei este accesibil de oriunde în clasa curentă sau în clasele derivate. În acest caz, clasele derivate pot fi localizate în alte ansambluri.

**intern** : membrii de clasă și de clasă cu un modificator similar sunt accesibili de oriunde din cod în același ansamblu, dar nu sunt disponibili pentru alte programe și ansambluri (ca în cazul modificatorului public).

**protejat intern** : combină funcționalitatea a doi modificatori. Clasele și membrii clasei cu acest modificator sunt accesibili din ansamblul actual și din clasele derivate.

**privat protejat** : un astfel de membru este accesibil de oriunde în clasa curentă sau în clase derivate care sunt definite în același ansamblu.

1. Ce inseamna using?

Cuvântul **using** cheie are două utilizări majore:

* Ca directivă:

1. pentru importarea tipurilor

using System.Text;

1. pentru a crea un alias pentru un spațiu de nume sau pentru un tip

using Project = PC.MyCompany.Project;

* Ca o declarație:

1. Oferă posibilitatea distrugerii variabilei in urma finisarii codului care se include in using

using (DbContext db = new ApplicationDbContext())

{

// database logic

}

1. Ce este o Interfata?

O interfață conține doar definitii(declarare) de metode , proprietăți , evenimente sau indexatori.

O clasă sau structura care implementează interfața trebuie să implementeze obligatoriu membrii interfeței care sunt specificați în definiția interfeței.

interface IInterfata

{

void Metoda1();

void Metoda2();

}

1. Ce este un ControllerContext?

Incapsuleaza informații despre o solicitare HTTP pentru un anumit controller care se potrivește cu instanțele specificate ale RouteBase și ControllerBase.

1. Ce este un Attribute?

Un atribut este o etichetă declarativă care este folosită pentru a transmite informații despre comportamentele diferitelor elemente cum ar fi clase, metode, structuri, enumeratori, ansambluri etc. Se poate de adăugat informații declarative unui program utilizând un atribut. O etichetă declarativă este reprezentată de paranteze pătrate ([ ]) plasate deasupra elementului pentru care este folosit.

Atributele sunt utilizate pentru adăugarea de metadate, cum ar fi instrucțiunile compilatorului și alte informații, cum ar fi comentarii, descriere, metode și clase la un program. Cadrul .Net oferă două tipuri de atribute: atributele predefinite și atributele personalizate construite.

1. Procese si metode. Scopul implementarii metodelor ASYNC (asynchrone)

Clasa Process se află în spațiul de nume System.Diagnostics care are metode pentru a rula un fișier .exe pentru a vedea orice document sau pagină web. Clasa Process oferă metode de pornire pentru lansarea unei alte aplicații în programarea C #.

O metodă este un bloc de cod care conține o serie de instrucțiuni. Un program face ca afirmațiile să fie executate apelând metoda și specificând orice argumente de metodă cerute.

Metodele asincrone permit efectuarea unui număr mult mai mare de I/O operațiuni legat „în paralel“, deoarece fiecare apel nu ocupă un întreg fir, astfel încât acest lucru este mai eficient și, de asemenea, se beneficiaza de prezenta la mai multe nuclee pe sistemul

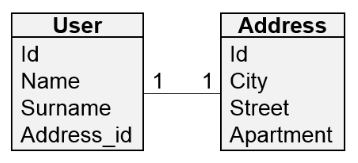
1. Tabele si structuri de date. Relatii intre tabele (EF)

Pentru a descrie structura de date definim 3 cuvinte cheie:

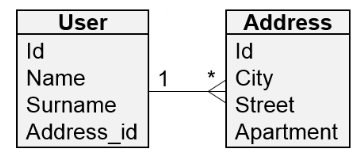
1. Tip de entitate (bloc care descrie structura de date, continutul)
2. Tip de asociere (tipul de relatii intre tabele)
3. Proprietate (contine caracteristici precum bool, int, float etc, sau date structurate)

Relatii intre table:

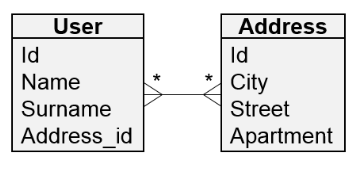
* ***Unu la unu***



* ***Unu la multe***

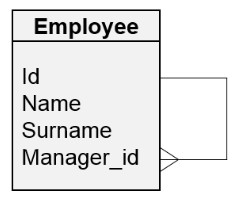


* ***Multe la multe***



*(trebuie sa existe un tabel intermediar)*

* ***Recursie***



**Orientat ASP.NET (MVC)**

1. Ce reprezinta un WebService?

Un WebService este un program software utilizat pentru a face schimb de informații cu alte programe prin intermediul protocoalelor de internet obișnuite, așa ca HTTP.

Într-un sens simplu, serviciile Web reprezintă o modalitate de a interacționa cu obiecte pe Internet.

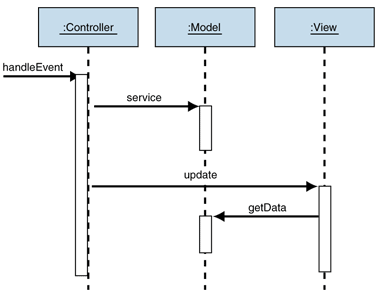
1. Structura unei aplicatii ASP.NET MVC

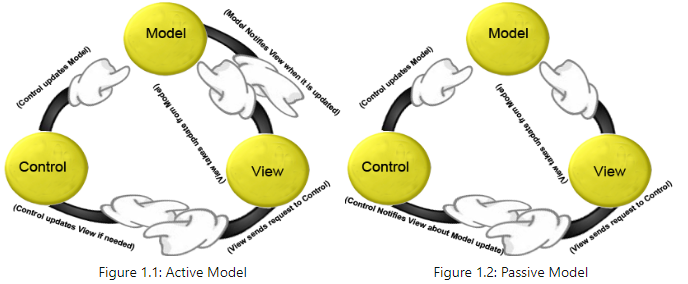
|  |  |
| --- | --- |
| Directoriul | Folderul proiectului |
| /Controllers | Contine clasele pentru Controller ce gestioneaza cererile URL |
| /Models | Contine clasele ce reprezinta datele modelului. |
| /Views | Contine fisiere template UI responsabile pentru afisarea rezultatului. |
| /Scripts | Contine biblioteci JavaScript si scripturi (.js). |
| /App\_Data | Contine fisierele de date Read/Write. DB. |
| /App\_Start | Contine cod de configurare, grupari de fisiere si Web API. |
| /Content | Contine fisierele CSS |
| /Images | Contine imaginile folosite in cadrul aplicatiei |

Aceasta structura este generata (nu in totalitate) de catre VS. Se poate folosi si o structura personalizata in sensul adaugarii a altor noi foldere.

1. Modelul pasiv de de manipulare asupra modelelor MVC.

Modelul pasiv constă în aceea că un controler manipulează exclusiv datele modelului. Controlerul modifică datele modelului și apoi informează View-urile că datele modelului s-au schimbat și ar trebui să fie actualizate. Modelul din acest scenariu este complet independent de Views și Controllers.





1. Actiuni disponibile asupra unui Controller

View, BadRequest, Json, Ok, Redirect, RedirectToAction, RedirectToRoute, Created etc.

1. Ce este un Bundle - cum il setam?

Bundle (Grupare) – este o caracteristică care îmbunătățește combinarea sau legarea a mai multor fișiere într-un singur fișier. Ex: fișiere CSS, JavaScript.

Pașii de setare:

1. creăm o clasă class BundleConfig
2. la aceasta clasă adaugam metoda RegisterBundles(BundlesCollection bundles);
3. apoi în **Global.asax**, în **Application\_Start()**, adăugam BundleConfig.RegisterBundles(BundleTable.Bundles);
4. în **RegisterBundles** ințializăm bundles:

*exemplu*

bundles.Add(new ScriptBundle("~/bundles/jquery/js").Include(

"~/Scripts/jquery-3.3.1.min.js"));

bundles.Add(new StyleBundle("~/bundles/bootstrap/css").Include(

"~/Style /bootstrap.min.css"));

1. Notiunea de Layout. Ce rol are intr-un proiect?

Layout este un view, care conține părți comune ale UI, astfel încât nu trebuie să scriem același cod în fiecare pagină unde este folosit acest view.

Componentele puse în Layout sunt statice și dinamice. Rolul Layot-ului este de a nu încărca acele părți statice pentru fiecare pagină, dar să încarce doar la prima accesare, și să încarce doar părțile dinamice, ex : **RenderBody** și **RenderSection**.

1. Definiti notiunea de AUTO PROPERTY. Da un exemplu.

Auto-Property este un mod similar mai simplu de a declare o propritate la care îi vor fi generate accesurile de **getter** și **setter**.

public string Name { get; set; }

**similar :**

private string \_name;

public string Name

{

get { return \_name; }

set { \_name = value; }

}

1. Metodele Html.BeginForm() Ajax.BeginForm(). Diferente/Asemanari.

Sunt două modalități de a crea un formular în pagină.

Ajax

1. **Nu va redirecționa** formularul chiar dacă faceți a RedirectAction().
2. Va efectua **asincron** salvarea, actualizarea și orice **operațiuni de modificare**.
3. Aceasta creează o formă care transmite valorile sale folosind o solicitare **asincronă ajax**. Acest lucru permite ca **o porțiune a paginii să fie actualizată** fără a fi necesară actualizarea întregii pagini.

html

1. Va redirecționa formularul.
2. Va efectua **sincron** operațiuni de modificare.
3. Va folosi întotdeauna RouteTable pentru a detrmina valoarea atributului de acțiune.
4. Va crea un formular pe pagina care transmite valorile sale serverului ca o cerere HTTP sincronă, actualizând întreaga pagină în proces.
5. Tipurile de actiuni cu redirectionare. 2 exemple de implementare.

**Redirect(***param***)** – redirecționare spre adresă specificată.

**RedirectToAction()** – redirecționare spre o metodă de acțiune, dintr-un controller cu parametru anumit (dacă e specificat).

**RedirectToRoute()** – redirecționare spre o ruta predefinita in RouteConfig.cs

**public** RedirectResult Index()

{

**return** Redirect("Home/Contact");

}

**public** ActionResult Index()

{

return RedirectToAction("Index", "Home", new { id = 2});

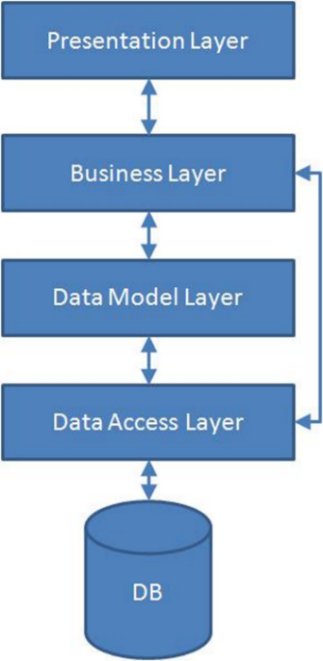
}

**public** ActionResult Index()

{

return RedirectToRoute("MyRoute");

}

1. Nivelul prezentare date. BusinessLogic.

**Nivelul “Presentation” –** este utilizat pentru a afișa datele utilizatorilor. adică interfața grafică, și cu ea trebuie să se comporte. (.html files).

**Nivelul “BusinessLogic” –** este stratul intermediar care comunică cu straturile de prezentare și acces la date. Aici se determină modul în care datele pot fi create, stocate și modificate

1. Metodele protocolului HTTP, GET() si POST().

**HTTP** (Hypertext Transfer Protocol) este un protocol de comunicație responsabil cu transferul de hipertext (text structurat ce conține legături) dintre un client (de regulă, un navigator) și un server web.

* **GET() -** descărcarea resursei specificate de pe serverul web pe client; majoritatea cererilor către un server web sunt de acest tip.
* **POST() -** creează o resursă, trimitând uzual entități (date, acțiuni) spre server e.g., datele introduse de utilizator în câmpurile unui formular Web
* **PUT() -** PUT actualizează o reprezentare de resursă sau eventual creează o resursă la nivel de server Web
* **DELETE() -** ștergerea resursei specificate de pe serverul web.

1. Cimpurile protocolului HTTP, se va cere de definit 7 cimpuri (minim).

|  |  |  |
| --- | --- | --- |
| Accept | Tipul de media care este acceptat | Accept: text/html |
| Accept-Charset | Tipul caracterelor acceptate | Accept-Charset: utf-8 |
| Accept-Encoding | Lista codarilor acceptabile | Accept-Encoding: gzip, deflate |
| Accept-Language | Lista limbajelor acceptate | Accept-Language: ro-RO |
| Accept-Datetime | Versiunea timpului acceptata | Accept-Datetime: Fri, 31 May 2019 11:30:00 GMT |
| Authorization | Credentiale pentru autentificarea HTTP | Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ== |
| Content-Length | Lungimea corpului de solicitare în octeți | Content-Length: 348 |
| Content-Type | Tipul de Media al corpului cererii (utilizat cu cererile POST și PUT). | Content-Type: application/x-www-form-urlencoded |
| Cookie | Un cookie HTTP trimis anterior de server cu Set-Cookie (mai jos) | Cookie: $Version=1; Skin=new; |
| Date | Data și ora la care a apărut mesajul | Date: Tue, 1 Jan 2019 00:00:00 GMT |
| From | Adresa de e-mail a utilizatorului care face cererea. | From: user@example.com |
| Host | Numele de domeniu al serverului si portul TCP(optional) | Host: google.com |
| User-Agent | Informatii despre agentul de utilizare al utilizatorului | User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:12.0) Gecko/20100101 Firefox/12.0 |

1. Action Filters. La ce sunt utilizate? Dati un exemplu de implementare.

*Filtrele de acțiune* sunt utilizate pentru implementarea logicii care se execută înainte sau după executarea unei acțiuni de către controler.

Filtrele de acțiune pot fi aplicate fie acțiunii controlerului, fie controlerului propriu-zis, cu ajutorul filtrului de acțiune putem schimba modul în care se execută acțiunea sau controlerul.

*Exemplu:*

Acest filtru de acțiune vă permite să restricționați accesul la un anumit utilizator sau rol.

[Authorize]

public ActionResult Index()

{

return View();

}

1. Evenimentele disponibile asupra unui Action Filters. Oferiti un exemplu de implementare.

**OnActionExecuting** - Această metodă este apelată înainte ca o acțiune a controlerului să fie executată

**OnActionExecuted** - Această metodă este apelată dupa ce o acțiune a controlerului este executată

**OnResultExecuting** - Această metodă este apelată înainte de executarea unui rezultat al acțiunii controlerului.

**OnResultExecuted** - Această metodă este apelată dupa executarea unui rezultat al acțiunii controlerului.

*Exemplu:*

public class CustomActionFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext filterContext)

*sau*

public override void OnActionExecuted(ActionExecutedContext filterContext)

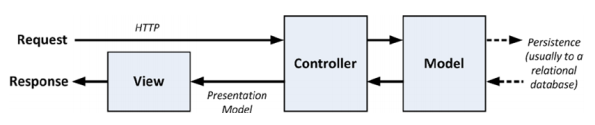
{

filterContext.Controller.ViewBag.Custom = "Custom text";

}

}

1. Ciclul de viata (life cicle) a unei CERERI < - > RASPUNS

* Un utilizator accesează adresa URL a unui site web
* Aceasta creează o cerere care este direcționată către un server web prin intermediul internetului prin HTTP
* Serverul web primește cererea HTTP și răspunde utilizatorului cu pagina web (sau conținutul) care a fost solicitată(in caz ca solicitarea este valida)