General C#

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

3.7 Declararea unei clase

Declararea unei clase se face în felul următor:
atribute$_{opt}$ modificatori-de-clasa$_{opt}$ `class` identificator clasa-de-baza$_{opt}$ corp-clasa ;$_{opt}$

Modificatorii de clasă sunt:

public - clasele publice sunt accesibile de oriunde; poate fi folosit atât pentru clase imbricate, cât şi pentru clase care sunt conţinute în spaţii de nume;

internal - se poate folosi atât pentru clase imbricate, cât şi pentru clase care sunt conţinute în spaţii de nume (este modificatorul implicit pentru clase care sunt conţinute în spaţii de nume). Semnifică acces permis doar în clasa sau spaţiul de nume care o cuprinde;

protected - se poate specifica doar pentru clase imbricate; tipurile astfel calificate sunt accesibile în clasa curentă sau în cele derivate (chiar dacă clasa derivată face parte din alt spaţiu de nume);

private - doar pentru clase imbricate; semnifică acces limitat la clasa conţinătoare; este modificatorul implicit;

2. **Ce inseamna using?**

3.5.8 Instrucţiunea using

Determină obţinerea a unei sau mai multor resurse, execută o instrucţiune şi apoi disponibilizează resursa:

```
using ( achizitie de resurse ) instructiune
```

O resursă este o clasă sau o structură care implementează interfaţa *System.IDisposable*, care include o sigură metodă fără parametri *Dispose()*. Achiziţia de resurse se poate face sub formă de variabile locale sau a unor expresii; toate acestea trebuie să fie implicit convertibile la *IDisposable*. Variabilele locale alocate ca resurse sunt read–only. Resursele sunt automat dealocate (prin apelul de *Dispose*) la sfârşitul instrucţiunii (care poate fi bloc de instrucţiuni).

Motivul pentru care există această instrucţiune este unul simplu: uneori se doreşte ca pentru anumite obiecte care deţin resurse importante să se apeleze automat metodă *Dispose()* de dezalocare a lor, cât mai repede cu putinţă.

Exemplu:

3. **Ce este o Interfață?**

Interfață

De la Wikipedia, enciclopedia liberă

O **interfață** este o parte a unui sistem care servește comunicării, facilitând aceasta. În tehnologia calculatoarelor termenul de interfață se referă la un punct (loc) de interacțiune dintre două unități, dispozitive componente etc. ale unui sistem, care (loc) este compatibil din punct de vedere hardware și software spre ambele părți ce comunică prin el uni- sau bidirecțional. Deci, prin analogie, sensul de interfață poate fi uzual interpretat ca o față (suprafață) de margine, de graniță a unui element, care servește comunicației spre și/sau dinspre alte elemente.

Orientat ASP.NET (MVC)

1. **Ce reprezinta un WebService?**

- Un serviciu web este o colecție de protocoale și standarde folosite pentru schimbul de date între aplicații sau sisteme. Aplicațiile software scrise în limbaje de programare diferite și care rulează pe diverse platforme pot folosi serviciile web pentru a face schimb de date pe rețea (Internet), într-o manieră oarecum asemănătoare comunicării între procesele de pe un singur calculator. Interoperabilitatea se datorează folosirii unor standarde publice adecvate.

2. **Structura unei aplicatii ASP.NET MVC**

Structura unei aplicatii web in ASP .NET

Într-o aplicaţie web realizată cu ASP .NET vom avea următoarele tipuri de fişiere şi directoare importante

Fişiere cu extensia .aspx reprezintă locul unde este stocat conţinutul paginii web. Se pot adăuga elemente HMTL, javascript, controale ASP .NET server-side.

Fişierele cu extensia .cs reprezintă codul ce se va executa pe server. Se poate scrie cod C# sau Visual Basic. Se pot defini variabile, funcţii, clase, etc.

Fişierul web.config este un fişier de configurare unde este stocată informaţia referitoare la aplicaţia web. În acest fişier se pot defini mai multe proprietăţi ale aplicaţiei. De exemplu, putem defini string-ul de conectare la baza de date pentru a-l putea folosi in orice pagină, sau putem modifica limita mărimii fişierelor upload-ate.

Fişierul Global.asax este opţional şi se poate folosi pentru a manipula anumite evenimente cum ar fi Application_Start, Application_End, Session_Start, Session_End

Directorul BIN este folosit pentru a stoca diverse fişiere dll folosite ca şi componente sau controale în aplicaţie.

Directorul App_Code conţine cod ce se va compila şi va fi vizibil în orice pagină a aplicaţiei web. Se pot stoca aici de exemplu diverse clase.

Directorul App_Data conţine fişiere mdf specifice bazelor de date SQL sau Access.

3. Modelul pasiv de de manipulare asupra modelelor MVC

Modelul pasiv este folosit cand un controller manipuleaza model-ul exclusiv. Controller-ul modifica model-ul si apoi informeaza view-ul ca model-ul a fost schimbat si ca trebuie reimprospatat. In acest scenario model-ul este complet independent de view si controller, ceea ce inseamna ca model-ul nu are cum sa raporteze schimbarile sale de stare. Protocolul HTTP este un exemplu eleocvent al acestui model pasiv. Browserul nu are cum sa primeasca update-uri asincrone de la server. Browserul afiseaza view-ul si raspunde la input utilizator dar nu detecteaza schimbari in datele de pe server. Doar atunci cand utilizatorul cere in mod explicit o reimprospatare serverul este interogat pentru eventualele schimbari.

4. Actiuni disponibile asupra unui Controller

Controllerul este componenta principală a arhitecturii MVC, ce conţine logica de execuţie a aplicaţiei. El reprezintă atât punctul de intrare în aplicaţie, cât şi cel de ieşire şi se foloseşte de celelalte componente (model şi view) pentru a-şi îndeplini sarcina dată. Controllerul este conceptualizat ca o clasă ce poate

îndeplini mai multe funcţii, numite acţiuni. Fiecare funcţie primeşte un set de date de intrare, ce reprezintă parametri sau informaţii date de către utilizator (datele problemei ce se doreşte a fi soluţionată).

5. Ce este un Bundle - cum il setam?

Bundle îmbunătățește timpul de încărcare prin reducerea numărului de cereri către server.

```
namespace eUseControl.Web
{
    0 references
    public class Global : HttpApplication
    {
        0 references | 0 exceptions
        void Application_Start(object sender, EventArgs e)
        {
            // Code that runs on application startup
            AreaRegistration.RegisterAllAreas();
            RouteConfig.RegisterRoutes(RouteTable.Routes);

            BundleConfig.RegisterBundles(BundleTable.Bundles);
        }
    }
}
```

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

Layout: Este modul în care elementele grafice, reclamele și textele sunt distribuite și aranjate. Poate sa contina portiuni ale interfetei utilizator care sunt la fel in mai multe view-uri. De exemplu, un layout ar putea contine portiuni header si footer si sa includa la mijloc continutul view-ului

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

Proprietățile automate este un membru care oferă un mecanism flexibil pentru citirea, scrierea sau calcularea valorii unui câmp privat.

public int SomeProperty { get; set; }

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

Metodele Html.BeginForm() Ajax.BeginForm() = metode de extensie care scrie răspunsul la o etichetă de deschidere "<form>".

Diferențe: metoda AjaxHelper transmite formularul asincron folosind JavaScript.

9. Tipurile de actiuni cu redirectionare. 2 exemple de implementare.

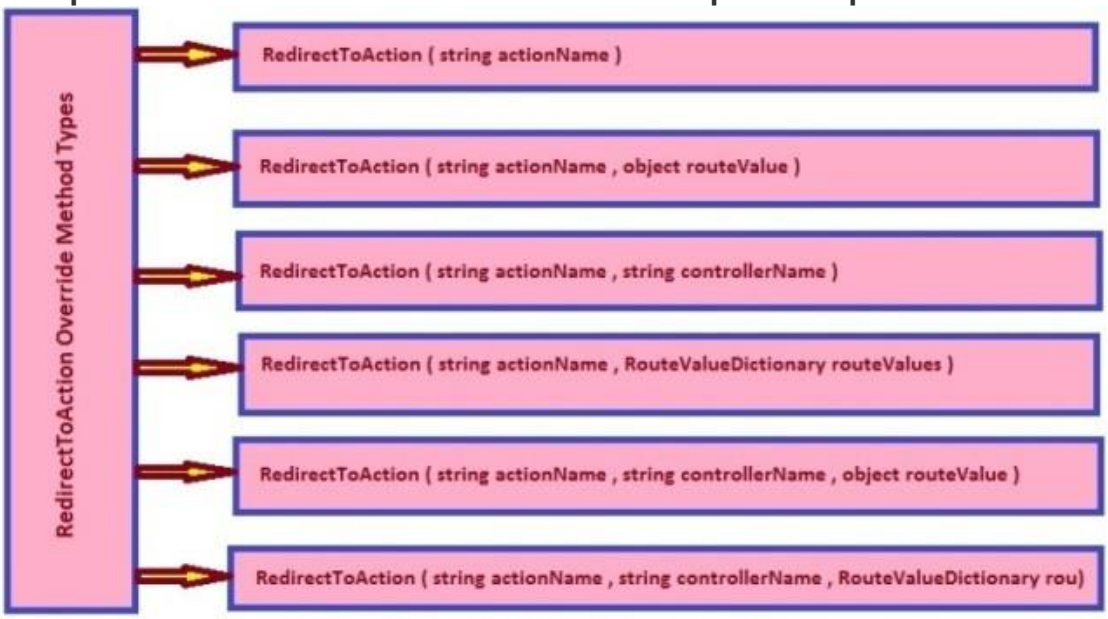

Figure : ASP.NET MVC RedirectToAction Methods

10. Nivelul prezentare date. BusinessLogic.

- Nivelul de prezentare (engl. Presentation Layer) care se ocupă cu afișarea interfeței aplicației prin folosirea serviciilor RESTfull;
- Nivelul de logică (engl. Business Logic Layer) care conține partea de logică a aplicației;