

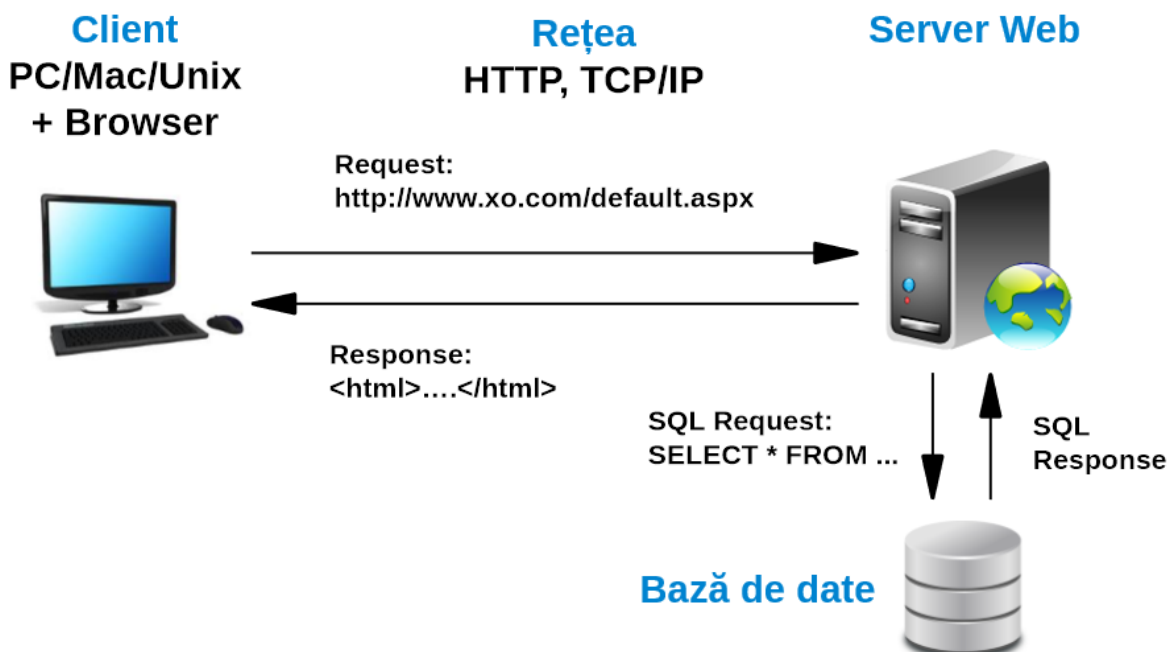
Ce este o aplicatie Web? Arhitectura Web. Avantaje/Dezavantaje ale aplicatiilor Web. Introducere in ASP.NET. Framework .NET. Introducere in C#. Conventii de notare. Ciclul de viata al unei pagini Web. Post Back. Prezentare IDE – Visual Studio

Ce este o aplicatie Web?

O **aplicatie web** este o aplicatie care ruleaza intr-o arhitectura Client-Server bazata pe: **protocolul HTTP, TCP/IP; browser Web; Server Web.**

Aplicatiile web sunt executate intr-un browser web si implementate folosind tehnologii precum: PHP, ASP, PYTHON, HTML, CSS, JAVASCRIPT, etc.

Arhitectura Web



Avantajele oferite de aplicatiile web

- Sunt independente de sistemul de operare
- Nu necesita instalare
- Actualizari foarte usor de facut deoarece modificarile se fac intr-un singur loc – pe server, ele propagandu-se pentru toti utilizatorii (in cazul aplicatiilor client-server clasice, interfata cu utilizatorul este asigurata prin intermediul unui program client instalat pe calculatorul fiecarui utilizator, orice modificare necesitand reinstalarea aplicatiei pentru fiecare utilizator in parte)

Dezavantaje

- Depind de conexiunea la Internet
- Riscuri de securitate

Introducere in ASP.NET

- **ASP.NET** este un framework Web open source conceput si dezvoltat de Microsoft.
- Este utilizat pentru a dezvolta site-uri, aplicatii si servicii web
- Oferă o integrare foarte buna a codului HTML, CSS, JAVASCRIPT
- Este construit pe baza **CLR (Common Language Runtime)** – ruleaza **cod compilat** si permite utilizatorilor sa scrie cod folosind **orice limbaj .NET**

Ce este CLR – Common Language Runtime?

Se ocupa de executia programelor C#. Atunci cand este compilat un program C# rezultatul compilarii **nu este un cod executabil**. In locul acestuia se produce un fisier care contine un tip de cod apropiat de codul masinii, numit limbaj intermediar sau pe scurt **IL – Intermediate Language**.

Prin intermediul unui compilator denumit **JIT – Just in Time**, CLR transforma codul intermediar in cod executabil.

Framework-ul .NET

- Este compatibil cu peste 20 de limbaje diferite, cele mai populare fiind C#, C++, Visual Basic, F#
- Pune la dispozitie o colectie impresionanta de clase, organizate in biblioteci
- Este construit din doua entitati importante:

1. Common Language Runtime (CLR)

- mediul de executie al programelor fiind cel care se ocupa cu managementul si executia codului scris in limbaje specifice .NET

2. Base Class Library

- Este biblioteca de clase .NET
- Acopera o arie larga a necesitatilor de programare, incluzand **interfata cu utilizatorul, conectarea cu baza de date, accesarea datelor**

Introducere in 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++
- Numele limbajului a fost inspirat de notatia # din muzica (nota muzicala urmata de # este mai inalta cu un semiton)

Limbaj compilat vs limbaj interpretat

Limbaj compilat – codul scris, numit cod sursa, este translatat de catre compiler intr-un cod apropiat de nivelul masinii, numit **cod executabil**. Atunci cand aplicatia trece de compilare fara erori de sintaxa se va produce codul executabil, iar aplicatia va putea fi rulata. (exemplu limbaje compilate: C, C++, Java, C#).

Limbaj interpretat (la rulare) – cu ajutorul unui interpretor specific limbajului, fiecare linie de cod este interpretata chiar in momentul rularii, fiind preschimbata imediat in cod masina si executata (exemplu limbaje interpretate: PHP, Ruby, Python)

Care tip de limbaj (compilat sau interpretat) este mai rapid?

- Cel compilat deoarece nu mai are nevoie de un interpretor

C# si Java

Hello World in Java:

```
public class Hello {  
    public static void main(String args []) {  
        System.out.println("Hello world! This is Java Code!");  
    }  
}
```

Correspondentul in C#:

```
using System; -- in Java se numeste pachet, in c# namespace  
public class Hello {  
    public static void Main(string [] args) {  
        System.Console.WriteLine("Hello world! This is C# code!");  
    }  
}
```

Conventii de notare

- Pentru variabile se utilizeaza **camelCase** (age, firstName, placeOfBirth)
- Pentru clase si metode se utilizeaza **PascalCase** (Age, FirstName, PlaceOfBirth)

Ciclul de viata al unei pagini Web

Paginile ASP.NET ruleaza pe server-ul Web Microsoft IIS (Internet Information Server). In urma prelucrarii pe server rezulta o pagina web HTML, care este trimisa catre browser.

Ciclul de viata al unei pagini Web ASP.NET are urmatoorii pasi:

- **Page request (accesarea paginii)** – acest pas se intampla inaintea ciclului de viata, atunci cand o pagina este ceruta serverului
- **Start** – in acest stadiu se incarca proprietatile paginii, cum ar fi request-ul si raspunsul si se identifica tipul acestora (**GET – cerere resurse, POST – trimiterea de informatii catre server**)
- **Initialization (initializare)** – in acest pas se initializeaza directivele si controalele si se aplica codul din Master Page
- **Load (incarcarea)** – in aceasta faza daca cererea este de tip **postback**, controalele sunt incarcate cu informatii
- **Evenimentele Postback** – daca cererea este de tip postback se executa codul aferent. Dupa executia codului se aplica sistemele de validare
- **Rendering (ex: afisarea paginii)** – in acest pas se construiesc pagina finala pe server, care va fi afisata in browser
- **Unload (eliberarea memoriei)** – dupa ce pagina a fost trimisa utilizatorului, resursele alocate pentru aceasta sunt eliberate

Post back

- Post Back este fenomenul prin care la generarea unui eveniment de catre utilizator (client), pagina web este transmisa server-ului, unde se poate executa o secventa de cod care sa trateze evenimentul respectiv.

- Unele controale genereaza automat un post back catre server. De exemplu controlul **Button** genereaza un post back la apasarea butonului. Alte controale nu genereaza un post back automat. De exemplu, controlul **TextBox** are un eveniment numit **TextChanged**. Acesta este generat de fiecare data cand valoarea din caseta de text este modificata. Implicit acest eveniment nu genereaza un post back. Insa, in cazul in care un alt control genereaza un post back (de exemplu un buton), in momentul executarii codului pe server este tratat prima data evenimentul TextChanged si apoi evenimentul Click al butonului.
- Daca proprietatea **AutoPostBack** are valoarea true, atunci controlul respectiv va genera automat un post back.

Un exemplu in care avem nevoie de aceasta proprietate

- Consideram o pagina Web in care avem un control **TextBox** initializat cu valoarea 0. La apasarea unui buton dorim incrementarea valorii din TextBox.

Codul ASP este urmatorul:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>PostBack</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:TextBox ID="TextBox1" runat="server" ReadOnly="True"> </asp:TextBox>
            <asp:Button ID="Button1" runat="server" onclick="Button1_Click1" Text="+"/>
        </div>
    </form>
</body>
</html>
```

Codul C#:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

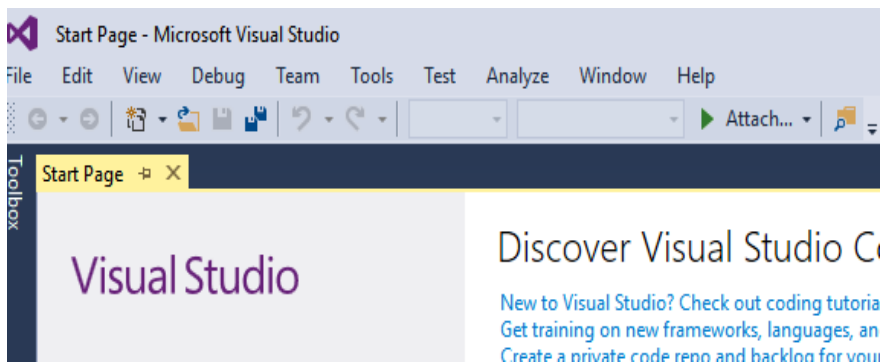
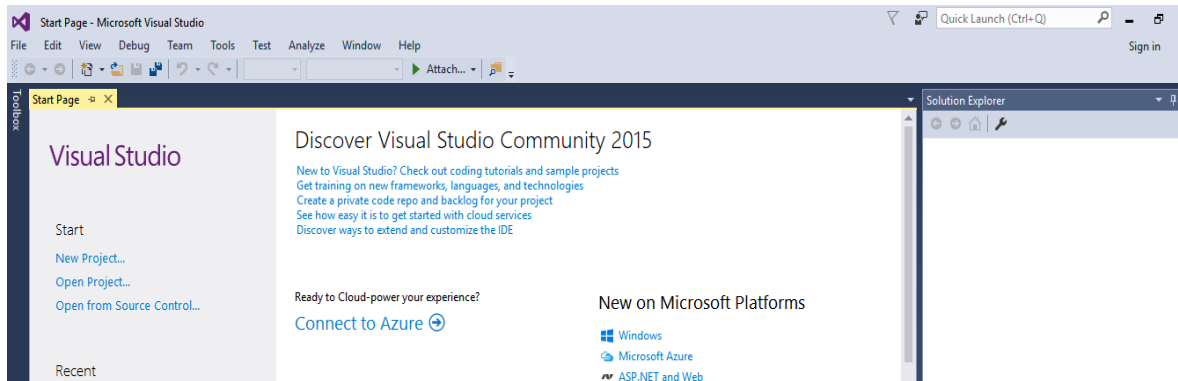
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Page.IsPostBack == false)
        {
            TextBox1.Text = "0";
        }
    }

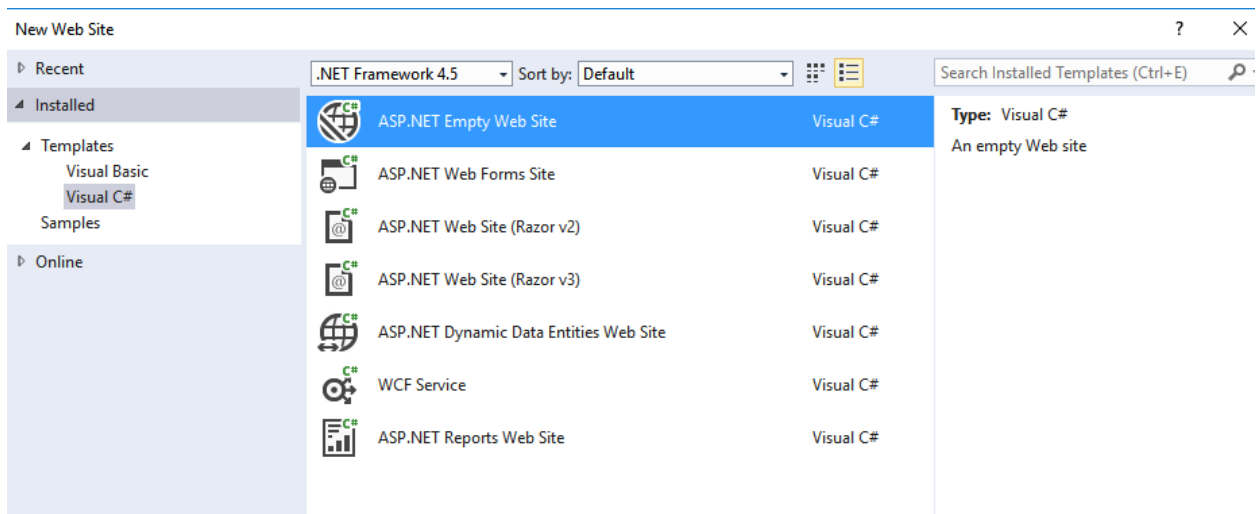
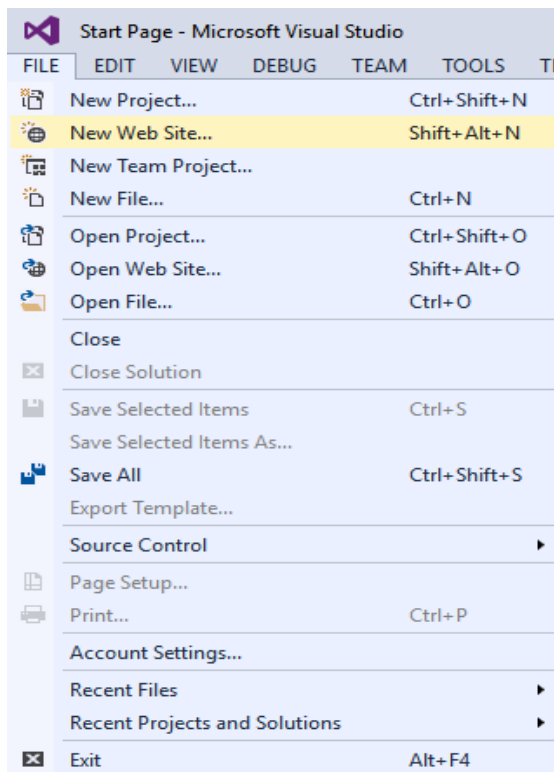
    protected void Button1_Click1(object sender, EventArgs e)
    {
        TextBox1.Text = Convert.ToString(Convert.ToInt32(TextBox1.Text) + 1);
    }
}
```

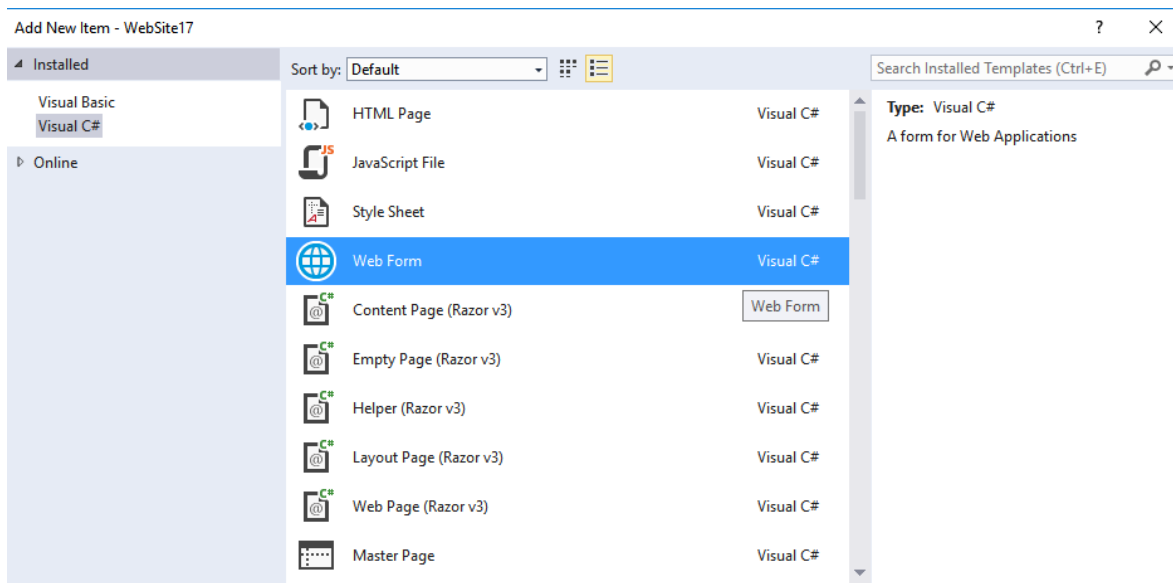
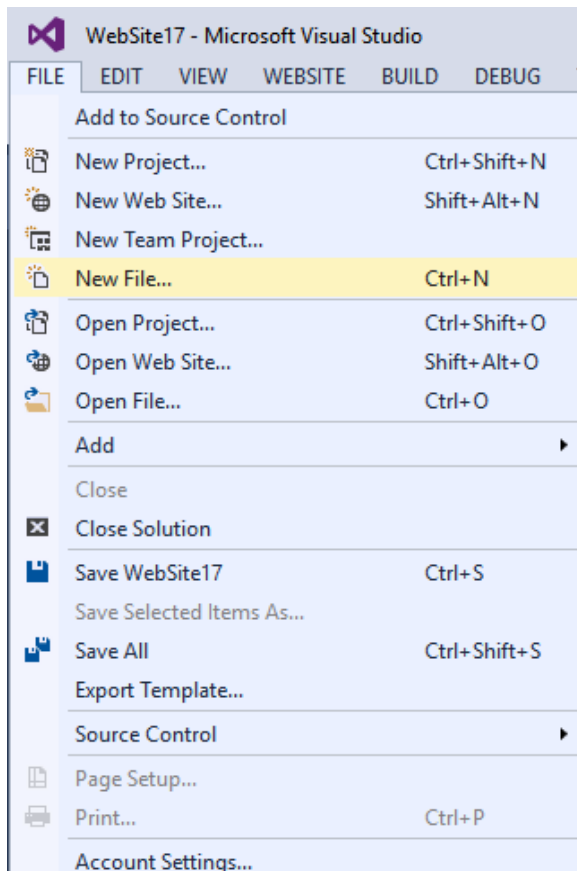
Page este o clasa care reprezinta o pagina ASP. Prin intermediul acesteia se pot accesa proprietati ale paginilor incarcate de pe server (informatii referitoare la request, la raspuns, la clientul care acceseaza site-ul, evenimente care se intampla in ciclul de viata al unei pagini, etc.). Toate fisierele **.aspx** au nevoie de un fisier **.cs** in care se implementeaza o clasa ce mosteneste clasa Page.

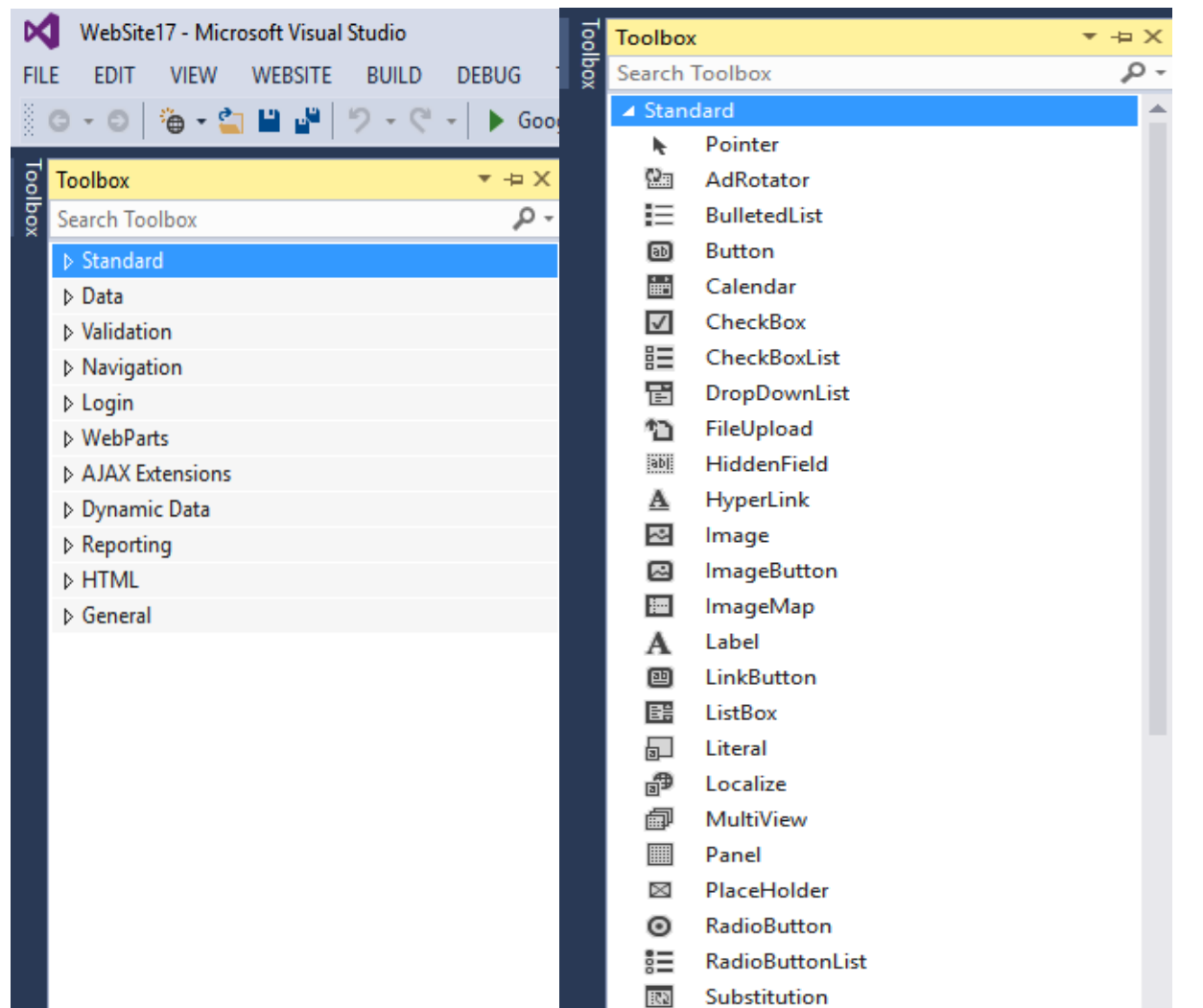
Page_Load – metoda care se executa in ciclul de viata al unei pagini, imediat dupa initializarea unei pagini Web. Aceasta este folosita fie pentru a initializa valorile default in pagina, fie pentru a face procesari inainte de afisarea paginii.

Prezentare IDE – Visual Studio

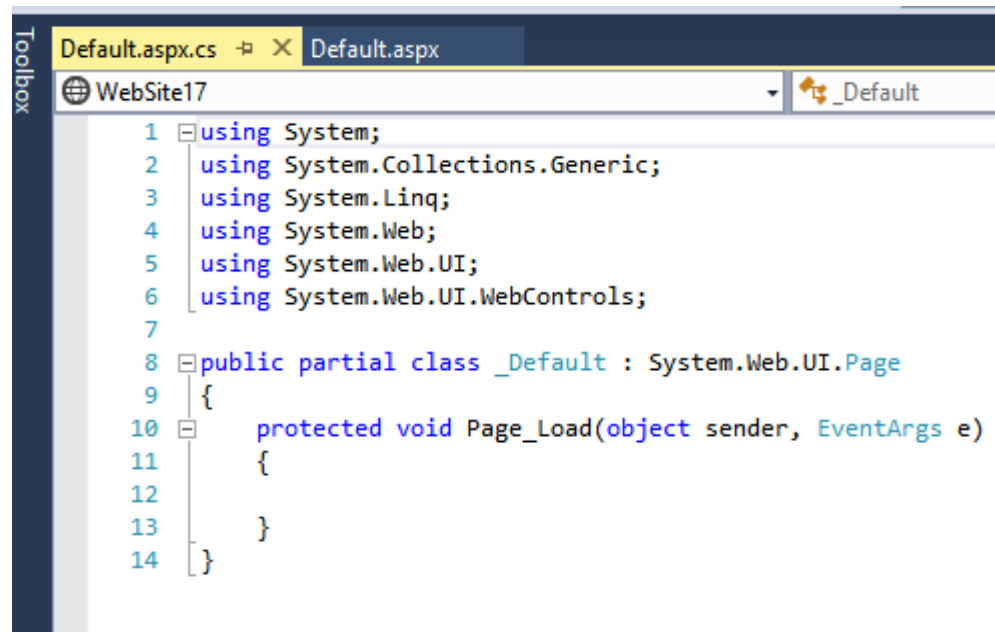








```
Default.aspx
1 <%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
2
3 <!DOCTYPE html>
4
5 <html xmlns="http://www.w3.org/1999/xhtml">
6 <head runat="server">
7 <title></title>
8 </head>
9 <body>
10 <form id="form1" runat="server">
11 <div>
12
13 </div>
14 </form>
15 </body>
16 </html>
17
```



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.UI;
6 using System.Web.UI.WebControls;
7
8 public partial class _Default : System.Web.UI.Page
9 {
10     protected void Page_Load(object sender, EventArgs e)
11     {
12
13     }
14 }
```

