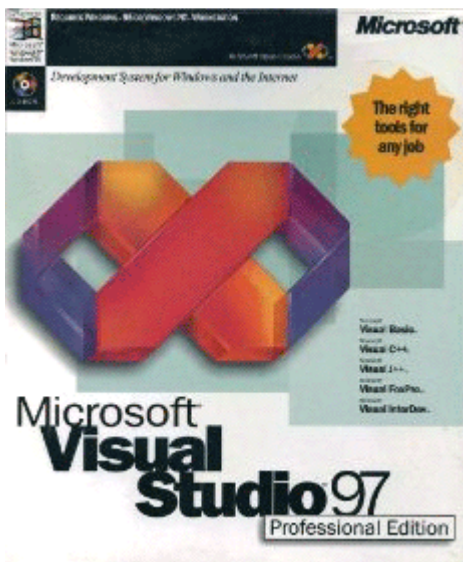


Introduzione e breve storia di Visual Studio

Visual Studio è lo strumento di punta che Microsoft dedica a chi sviluppa su piattaforma Windows. Fin dalla sua prima versione, datata 1997, la sua missione era già quella di fornire **un ambiente di sviluppo grafico ed integrato** che aiutasse lo sviluppatore a gestire i progetti in maniera semplice, ma efficace, aumentandone quindi la produttività.

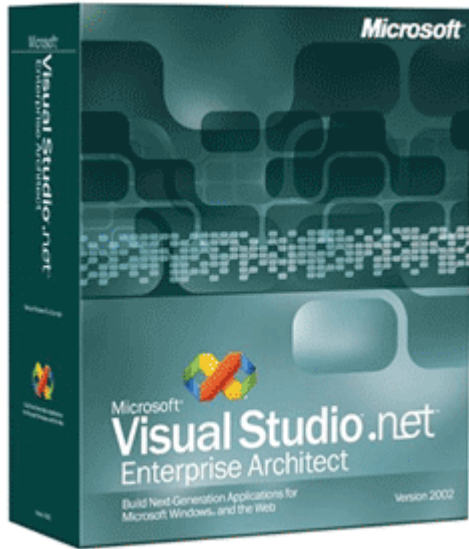
Figura 1. La prima versione di visual studio (1997)



La novità principale fu che, per la prima volta, Microsoft incluse, in un solo prodotto, il supporto a differenti linguaggi (C++, J++). L'intenzione era quella di ridurre la complessità attuale, in cui ogni linguaggio o tecnologia possedeva uno strumento dedicato ed obbligava lo sviluppatore a dover familiarizzare con molti ambienti differenti.

La versione successiva, Visual Studio 6.0, rimase sul mercato per quattro anni e fu una versione di "transizione" perché nel 2002 uscì la prima versione del **Visual Studio .NET**, il cui nome deriva dal framework **.NET** e di cui la versione attuale è diretta evoluzione.

Figura 2. Prima apparizione del .NET (2002)



La rivoluzione stava nell'introduzione di un linguaggio assembly intermedio, chiamato MSIL e supportato da differenti linguaggi, ma soprattutto **viene introdotto il nuovo linguaggio C#**, che di lì a breve ottenne un ottimo successo anche in ambienti Open Source (http://www.mono-project.com/Main_Page).

Finalmente l'interoperatività tra i linguaggi non era più un sogno, era possibile scrivere una routine in Visual Basic.NET ed utilizzarla poi in C# senza alcun problema; inoltre, scrivendo qualche riga di codice, era possibile anche effettuare chiamate a routine C++, il che doveva stabilire anche la fine dell'era di COM (<http://www.microsoft.com/com/default.mspix>).

Dalla prima versione è iniziata quindi una serie di upgrade cadenzati con i quali Microsoft rilascia una nuova versione orientativamente ogni due anni ed un service pack negli anni dispari. Ad ogni nuovo rilascio le novità interessano sia l'ambiente di sviluppo, sia le tecnologie ed i linguaggi oltre che l'introduzione di nuove tecnologie e linguaggi, come la tecnologia LINQ (<http://aspnet.html.it/articoli/leggi/2527/introduzione-a-linq/>), introdotta con il Visual Studio 2008 ed il linguaggio F#, introdotto con il Visual Studio 2010.

Nel corso degli anni Visual Studio diventa sempre di più fulcro di ogni attività legata allo sviluppo, anche di quelle non propriamente legate al codice, come dimostra il supporto fornito dal sviluppo di pacchetti di **Sql Service Integration Services** o **Sql Server Reporting services**.

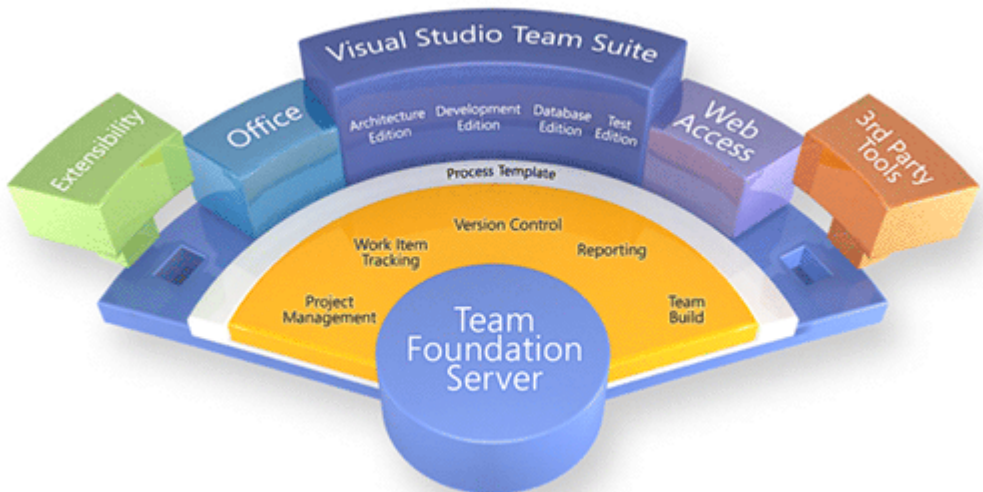
Parallelamente al Visual Studio Microsoft propone una soluzione per la gestione del ciclo di vita dei progetti, il primo tentativo si chiama **Source Safe**, ed è semplicemente un version control system che si appoggia alla condivisione dei file di Windows. Rapidamente il prodotto diventa abbastanza obsoleto, soprattutto con l'avanzare di internet e delle vpn; lavorare in una vpn su internet con Visual Source Safe era infatti decisamente lento anche per team piccoli, era necessaria quindi una nuova soluzione.

Nel 2005 esce quindi la prima versione di **Team Foundation Server**, la parte "server" di Visual Studio che è molto di più di una semplice evoluzione di Visual Source Safe. TFS include naturalmente un Version Control System, ma stavolta basato su database SQL ed accesso tramite web services; il risultato è una maggiore affidabilità e velocità anche su team distribuiti con molti sviluppatori.

L'innovazione maggiore di TFS è però **l'introduzione di un sistema di Work Item Tracking**, con cui gli sviluppatori possono tenere traccia dei requisiti, dei bug, delle feature da implementare, il tutto integrato con SharePoint Services e Sql Server Reporting Services. Una struttura di integrazione continua

(Tfs Build) completa poi la suite; TFS diventa quindi de-facto uno strumento in grado di gestire il ciclo di vita di un'applicazione a tutto tondo.

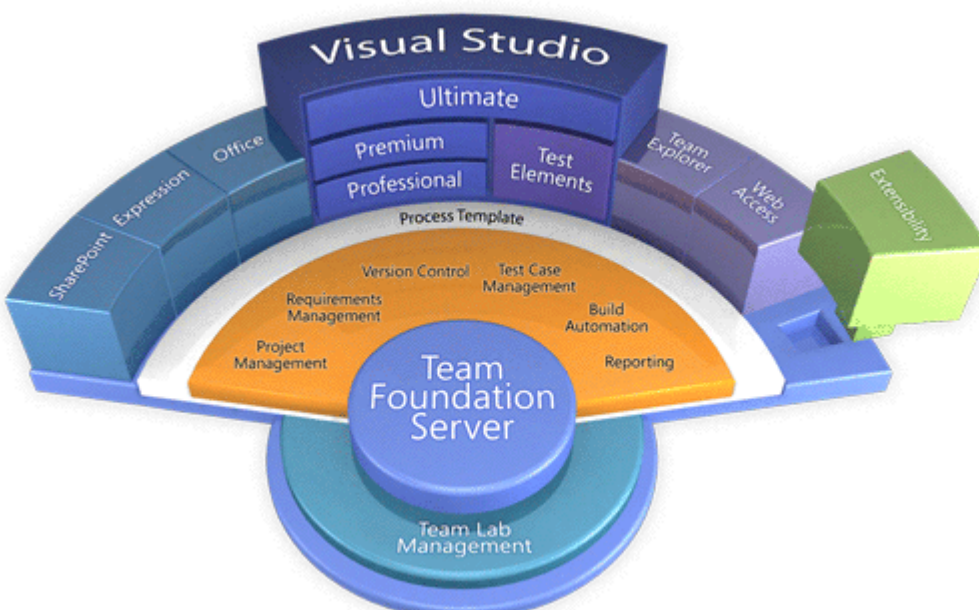
Figura 3. Team Foundation Server e tool di sviluppo correlati



Il vantaggio maggiore nell'uso di TFS è l'elevato livello di integrazione fornito con gli strumenti Microsoft, TFS può infatti essere gestito da Visual Studio, ma è possibile anche gestire i Work Item da Excel o Microsoft Project. Inoltre TFS espone le sue funzionalità tramite API ed è quindi possibile per sviluppatori o aziende di terze parti fornire strumenti che ne estendono le funzionalità. L'esempio migliore di questa interazione è Teamprise (<http://www.teamprise.com/>), recentemente acquisita da Microsoft, che fornisce un plugin per eclipse, che permette di **usare TFS in progetti java**, integrando anche strumenti di build tipici di altre piattaforme, come Maven.

Con il 2010 arriva la nuova versione, il cui nome in codice è *Rosario*, di cui, in questa guida, esploreremo le nuove funzionalità.

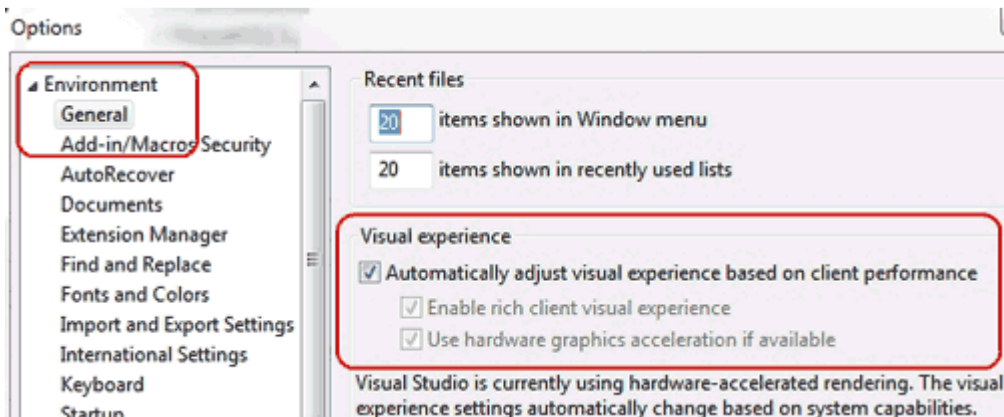
Figura 4. Il panorama attuale degli strumenti di sviluppo Microsoft



Personalizzare l'area di lavoro

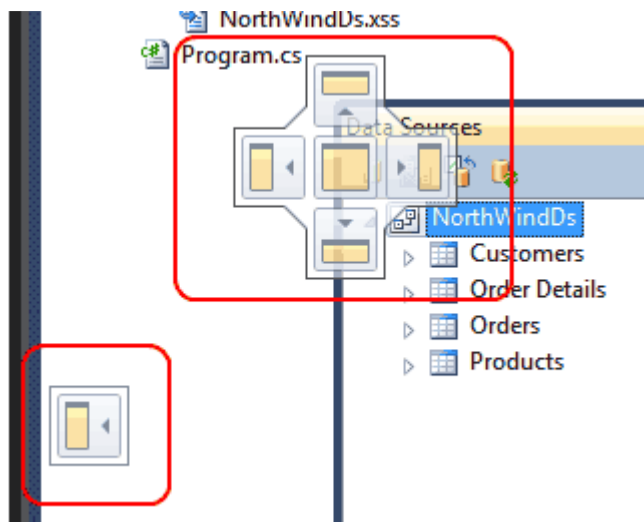
Una delle modifiche sostanziali introdotte con VS2010 è l'interfaccia utente completamente riscritta in WPF. Le implicazioni in questo cambiamento sono decisamente molte, prima fra tutte la possibilità di usare l'accelerazione hardware, opzione che è automaticamente abilitata dal menu `Tools > Options`.

Figura 5. Le opzioni relative all'accelerazione hardware dell'interfaccia utente



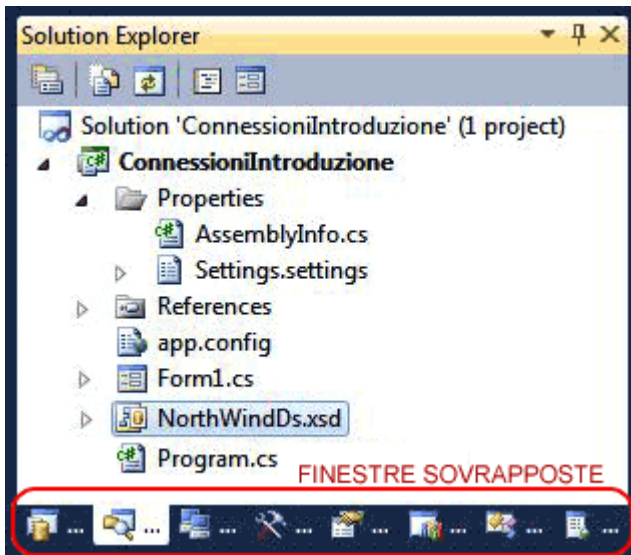
Abbiamo la possibilità di modificare a nostro piacimento la disposizione di finestre e pannelli grazie alla flessibilità offerta dall'IDE e ai **gadget per l'ancoraggio**, che appaiono quando trasciniamo le finestre.

Figura 6. Ancoraggio delle finestre



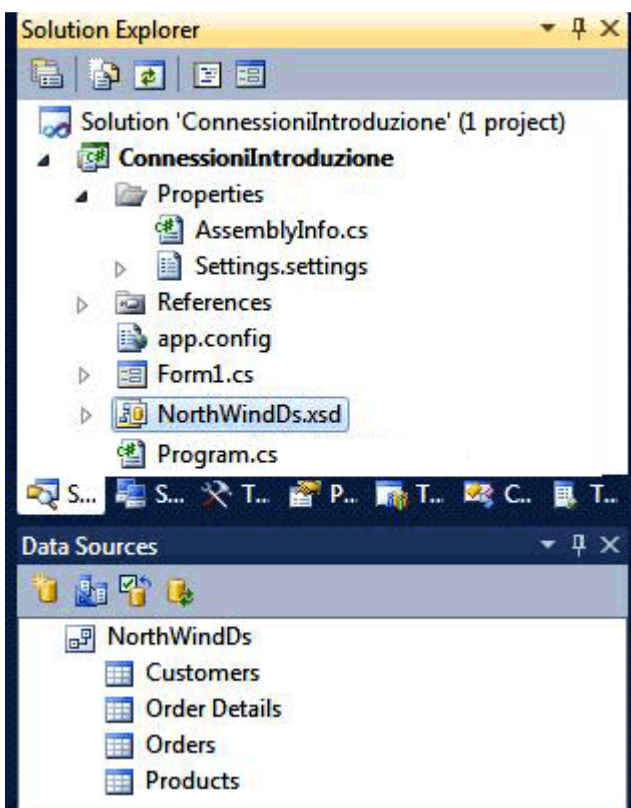
Lo strumento appare come una croce che indica quale posizione avrà la finestra rispetto al pannello sottostante. Se ad esempio rilasciamo il tasto del mouse sul simbolo al centro della croce, sovrapponiamo la finestra trascinata a quella sottostante e creiamo così un pannello con più tab. Per passare da una finestra all'altra ci muoviamo utilizzando le relative linguette.

Figura 7. Accorpamento delle finestre



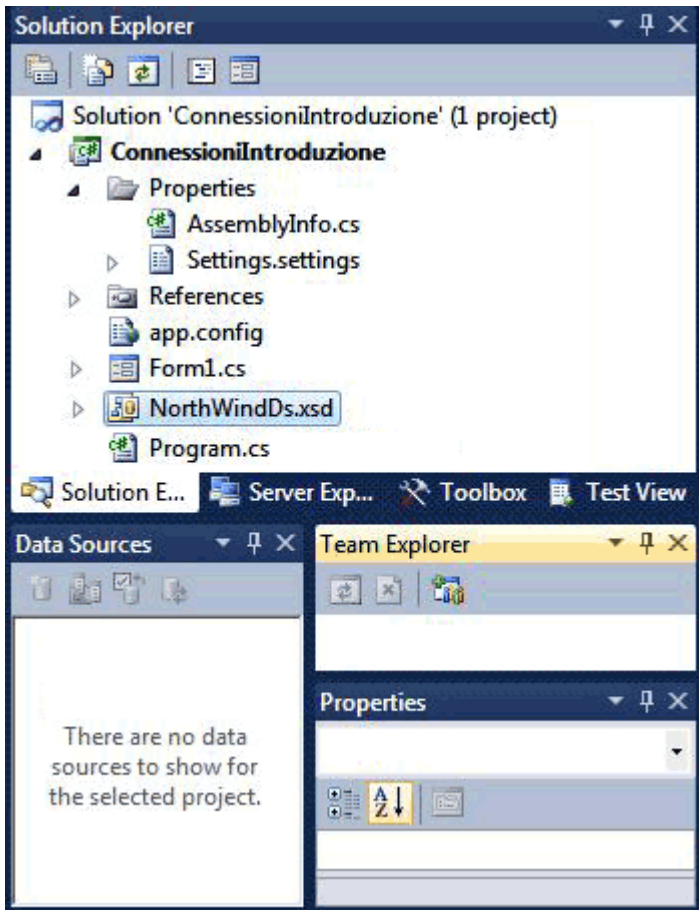
Nel caso in figura molte finestre condividono la stessa area dello schermo, tra cui "Solution Explorer" che è attiva. Vediamo cosa succede invece se rilasciamo il tasto del mouse in una posizione differente, ad esempio in basso:

Figura 8. Ancoraggio in basso della finestra



Le possibilità di ancoraggio non hanno limiti, ogni volta che una finestra viene ancorata, può essere soggetta ad ulteriori ancoraggi, come nella figura seguente, che mostra un layout decisamente "strano", ma è un esempio della flessibilità con cui possiamo posizionare finestre e toolbar.

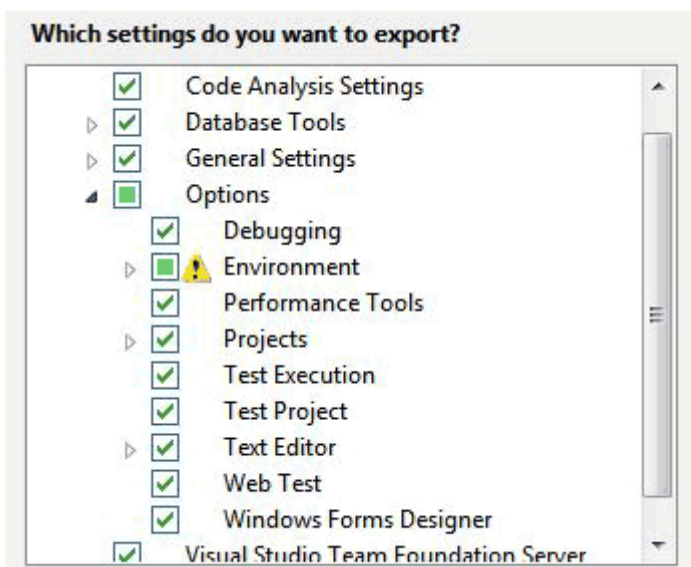
Figura 9. Un layout "estremamente" customizzato



Una volta personalizzato l'ambiente di sviluppo, abbiamo la possibilità di **conservare e riapplicare tutte le nostre preferenze**, grazie alle funzionalità del menu Tools > Import and export settings, che ci permettono anche di effettuare il reset di tutte le impostazioni. Questo tra l'altro ci consente di utilizzare impostazioni diverse a seconda del progetto su cui stiamo lavorando.

Come mostra la prossima immagine, possiamo salvare la impostazioni relative al layout, ma anche quelle relative all'ambiente di sviluppo.

Figura 10. Esportare le impostazioni dell'IDE



Infine, l'ultima potenzialità interessante da sottolineare è il supporto maggiore a chi ha più di un monitor: ora possiamo estrarre dal layout anche le finestre di codice e spostarle tra i diversi schermi. La possibilità di visualizzare contemporaneamente più finestre di codice, designer, etc., permette una visione d'insieme senza precedenti.

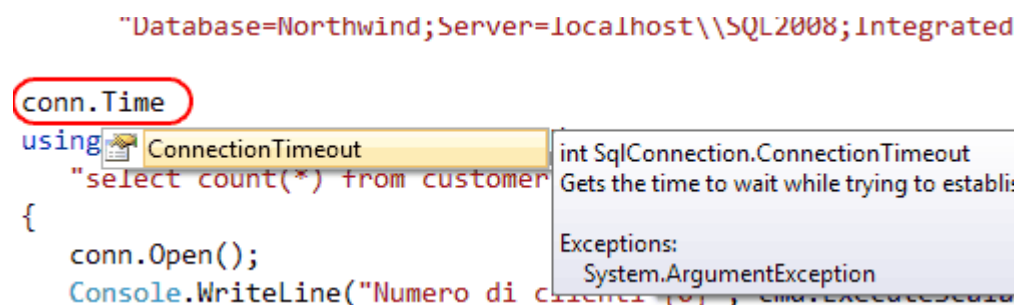
L'IntelliSense

In questa lezione esaminiamo alcune importanti novità introdotte nell'intellisense, funzionalità senza la quale ogni sviluppatore è "perduto".

La logica del "contiene"

La prima importante novità sta nel fatto che **intellisense non ragiona più con la logica "inizia per"** ma con la logica "contiene". Per questa ragione, se stiamo lavorando con un oggetto `SqlConnection` e ci ricordiamo che esiste una proprietà che regola il Timeout, basta iniziare a scrivere `Tim` e subito ci viene presentata la proprietà che stiamo cercando, anche se in realtà si chiama `ConnectionTimeout`.

Figura 11. Il suggerimento mostra i metodi/proprietà che contengono il testo digitato

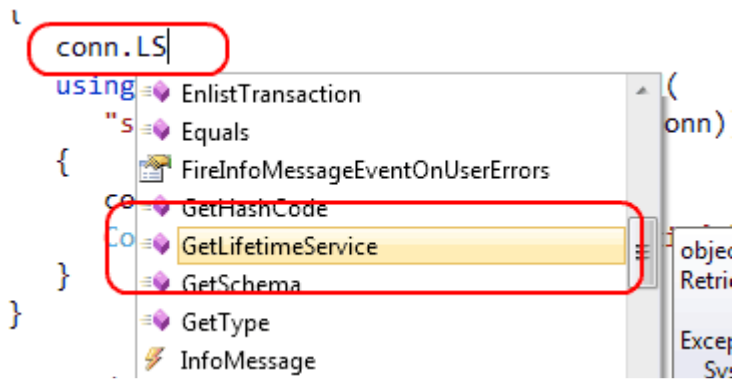


Abbreviazioni e riconoscimento del 'casing'

L'intellisense è ora in grado di interpretare alcune 'sigle' mentre digitiamo, grazie alla convenzione del casing, per il quale le parole composte vengono scritte senza spazi ma con il primo carattere maiuscolo (PascalCase, caso particolare di CamelCase (<http://it.wikipedia.org/wiki/CamelCase>)). L'IDE si serve delle iniziali (lettere maiuscole) delle classi per suggerirci come completare le nostre espressioni.

Ad esempio, sempre per l'oggetto `SqlConnection`, poniamo il caso di ricordare che esiste qualche metodo che permette di interagire con il `LifetimeService`, potremmo digitare in maiuscolo `LS` ed ottenere questo risultato:

Figura 12. IntelliSense avanzato, riconoscimento del casing



L'intellisense, sapendo che le iniziali di ogni parola che compone un membro di classe è maiuscola, quando digitiamo LS cerca un membro composto da due o più parole di cui due iniziano per L ed S.

Consume first: generazione di classi "al volo"

Vediamo cosa accade se tentiamo di utilizzare una classe chiamata MyNewClass che attualmente non esiste nel progetto:

Figura 13. Consume first intellisense

```
MyNewClass mnc = new My
//Database2();
```

The image shows a code editor with a code snippet: 'MyNewClass mnc = new My' followed by a new line with '//Database2();'. A suggestion box is visible below the code, showing 'MyNewClass' with a small icon to its left.

L'intellisense ce la propone come se fosse in realtà già stata dichiarata. Al termine della digitazione, nel menu contestuale si può scegliere di generare la nuova classe, d'altra parte se questo è quello che il programmatore ha digitato, probabilmente è quello che vuole.

Figura 14. Creare al volo la classe

```
MyNewClass mnc = new MyNewClass();
//Database2();
```

The image shows a code editor with a code snippet: 'MyNewClass mnc = new MyNewClass();' followed by a new line with '//Database2();'. A context menu is visible below the code, showing 'Generate class for 'MyNewClass'' and 'Generate new type...'. The 'Generate class for 'MyNewClass'' option is highlighted.

Ora che la nuova classe è stata creata, se scriviamo "mnc.", l'intellisense ci propone gli unici metodi attualmente disponibili per la classe, quelli ereditati da System.Object. Premendo CTRL+ALT+SPACE, però, l'intellisense presenta una interfaccia leggermente differente, in cui la prima opzione lista quello che si sta digitando.

Figura 15. Consume-first intellisense

```
MyNewClass mnc = new MyNewCl
mnc.MyNewMethod
//Da
MyNewMethod
Equals
GetHashCode
GetType
ToString
private
using (Northwind northwind
```

Ora l'intellisense è in **Consume-first mode**, per cui presuppone che tutto quello che digitiamo sia corretto, anche se non esiste ancora. Questa feature è di aiuto ad esempio nel Test Driven Development. Possiamo infatti digitare una chiamata ad un metodo inesistente e poi, grazie allo stesso menu contestuale, far generare a Visual Studio uno stub valido nella classe.

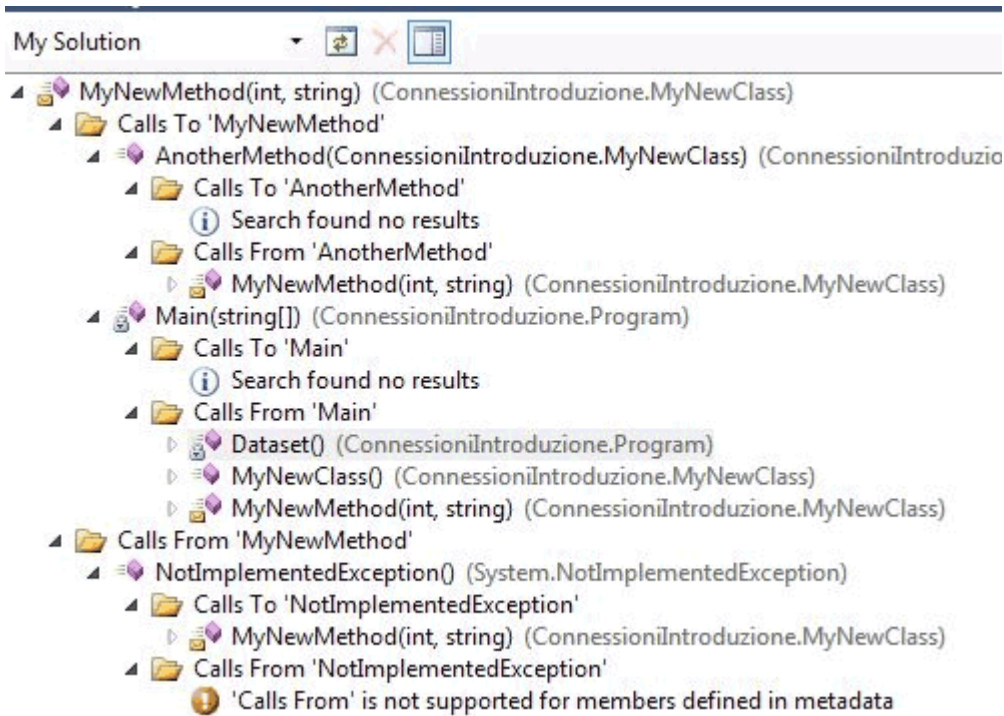
```
mnc.MyNewMethod(12, "pippo");
...
internal void MyNewMethod(int p, string p_2)
{
    throw new NotImplementedException();
}
```

Tutti i metodi autogenerati lanciano una eccezione perché debbono poi essere completati.

Navigazione e visualizzazione del codice

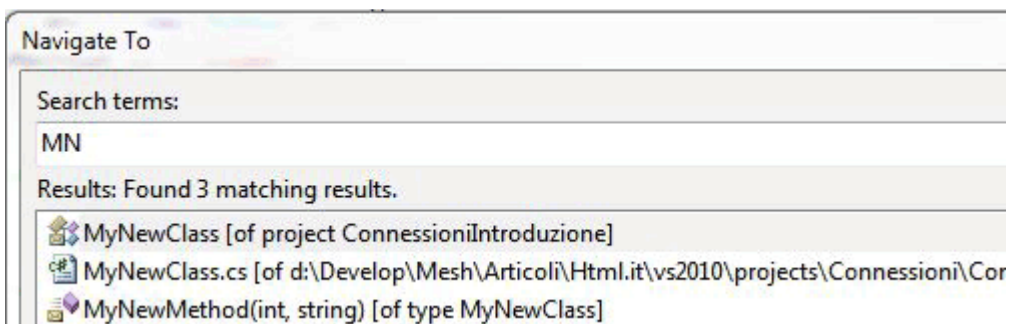
Tra le altre funzionalità interessanti offerte da VS2010 vi sono numerose possibilità per "navigare" nel codice, premendo la combinazione CTRL+K,T oppure effettuando click con il tasto destro e selezionando View Call Hierarchy il VS mostrerà tutti i punti nel vostro codice dove quel particolare oggetto viene chiamato.

Figura 16. Visualizzazione della Call Hierarchy



Grazie a questa finestra si può effettuare una navigazione molto profonda nel codice e comprendere meglio il pattern di utilizzo di una particolare funzione o proprietà. Premendo invece la combinazione di tasti CTRL+, (control + virgola) si apre **la finestra 'Navigate To'**, nella quale possiamo digitare parte del nome di una classe, metodo o proprietà e lasciare che Visual Studio ci mostri tutte le corrispondenze nella solution aperta, anche in questo caso è attiva la funzionalità di "casing", per cui se digito MN maiuscolo ecco il risultato che si ottiene:

Figura 17. Le funzionalità della finestra "navigate to"



Infine ogni volta che si posiziona il cursore su di una variabile o oggetto, VS evidenzia nel codice tutte le volte che quella variabile viene utilizzata, per avere un immediato colpo d'occhio sull'impatto della variabile nello scope su cui si sta lavorando.

Figura 18. Evidenziazione dell'uso di una variabile nel codice

```

using (NorthWindDs northwind = new NorthWindDs())
{
    using (NorthWindDsTableAdapters.CustomersTableAdapter ta =
        new NorthWindDsTableAdapters.CustomersTableAdapter())
    {
        ta.FillByCustomerIdSearch(northwind.Customers, "A");
        foreach (NorthWindDs.CustomersRow row in northwind.Custor
        {
            while (row.CompanyName.EndsWith("ADDED"))
            {
                row.CompanyName = row.CompanyName.Remove(row.Compar
            }
        }
        northwind.Customers.Rows[0].Delete();
    }
}

```

Infine qualche piccola "chicca". È possibile **effettuare lo zoom del codice** tenendo premuto il tasto CTRL ed usando la rotellina del mouse, questa funzione è assolutamente fondamentale per chi fa presentazioni in cui si usa VS, ma anche per aumentare la porzione di codice visualizzata. Se si è in presenza di un metodo molto lungo, si può ridurre il livello di zoom temporaneamente per tenere sotto occhio una parte di codice più ampia.

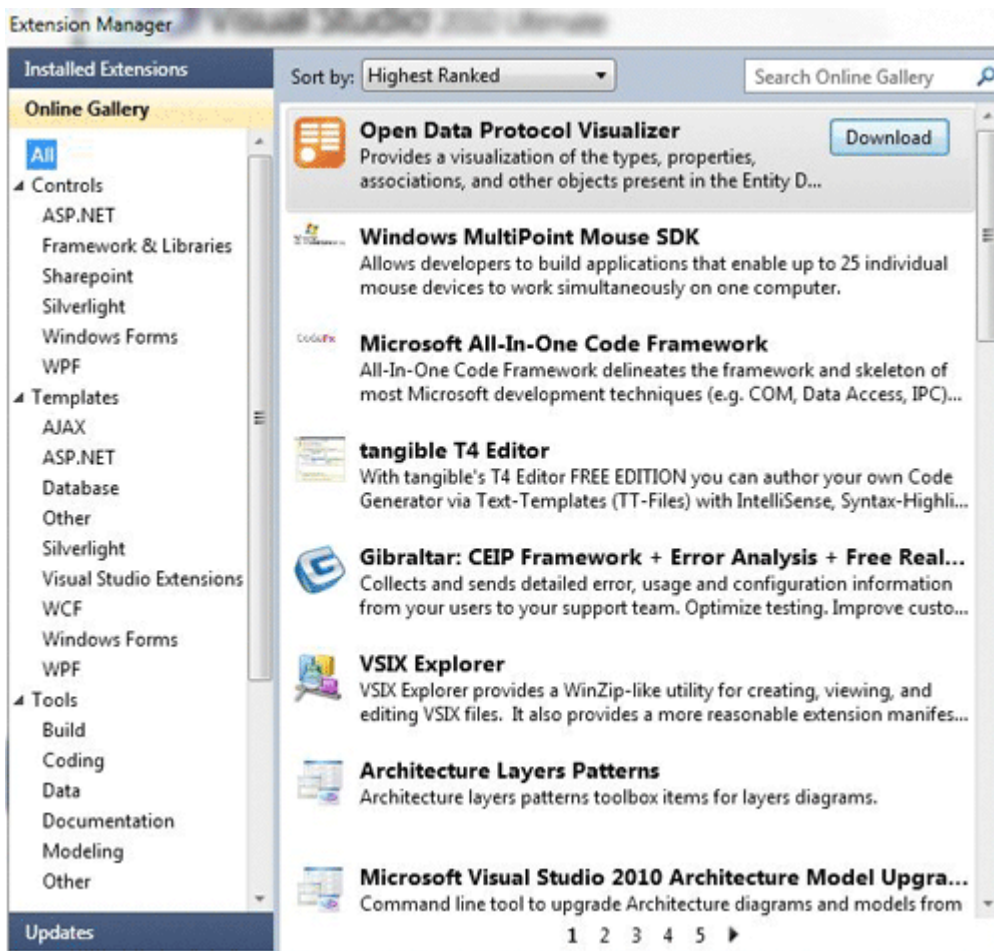
È inoltre possibile **selezionare una parte di testo qualsiasi** tenendo premuto il tasto ALT, in questo caso è possibile selezionare qualsiasi porzione rettangolare di testo o codice, questa selezione può essere copiata ed incollata in altre selezioni rettangolari.

Extension Manager

Per concludere questa introduzione non si può fare a meno di parlare dell'extension manager, (menu Tools > Extension Manager) che permette di **gestire tutti gli addin e le estensioni di Visual Studio**.

L'aspetto più interessante è che ora tutte le estensioni possono essere inserite nella **Microsoft Gallery** e quindi essere recuperate direttamente dal VS come mostrato in figura.

Figura 15. Estensioni online visualizzate dall'Extension Manager



Come si può notare è sufficiente cliccare su Online Gallery e gli addin disponibili vengono mostrati direttamente nella finestra di gestione. Le estensioni sono categorizzate in tre macrocategorie: controlli, template e strumenti ognuna delle quali è poi suddivisa ulteriormente per categoria. **Ogni estensione può essere votata dagli utenti** ed è quindi immediato trovare i prodotti che hanno maggior successo.

Installare un addin è un'operazione veramente banale, è sufficiente selezionarlo, premere il bottone Download, accettare gli eventuali termini di licenza ed il gioco è fatto, l'estensione è immediatamente scaricata ed installata.

Per tutti gli addin che richiedono un restart, dopo l'installazione viene immediatamente proposto di riavviare il Visual Studio affinché il prodotto installato sia disponibile. Come ultima caratteristica, l'extension manager permette di controllare gli eventuali aggiornamenti di tutti i prodotti installati nel tab updates.

La semplicità con cui Visual Studio può essere esteso pone le basi per un proliferare di nuova addin che ci permetterà una personalizzazione senza precedenti.

Creare un progetto ASP.NET Web Form

Quando parliamo di sviluppo Web in VS2010, possiamo pensare principalmente a due tipi di applicazione: le applicazioni ASP.NET Web Form "classiche" e quelle realizzate con il nuovo framework ASP.NET MVC2, evoluzione di asp.net MVC1 già disponibile come addin per Visual Studio 2008.

Queste due tipologie di progetto sono molto differenti, sebbene entrambe siano basate sul motore comune di ASP.NET.

| Tipo di applicazione | Descrizione |
|----------------------|---|
| Web Form | adottano il classico paradigma delle applicazioni Windows, ovvero pagine Web con controlli ed una gestione tipo eventi |
| MVC2 | è una tecnologia orientata verso chi conosce bene HTML e vuole poter creare degli unit test per le logiche di interfaccia |

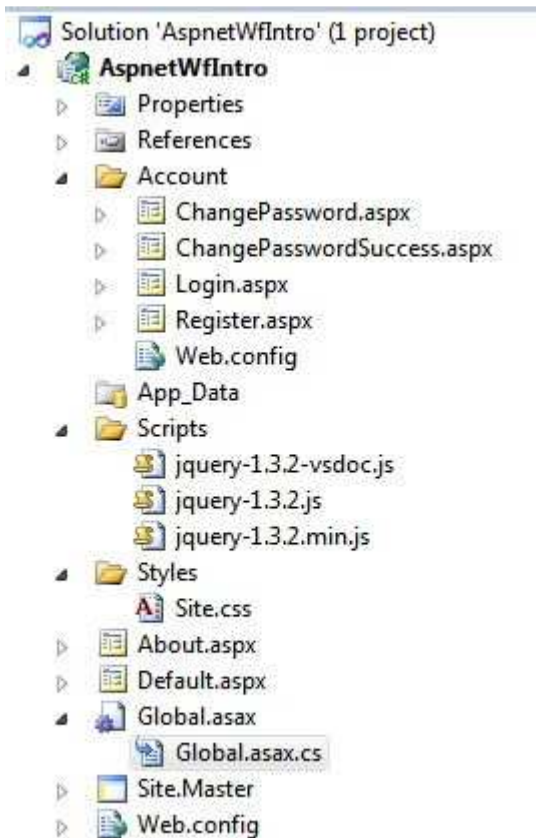
La scelta del progetto non è del tutto irreversibile: la tecnologia di base è sempre ASP.NET ed è comunque possibile includere in uno stesso progetto sia pagine web form sia pagine in mvc2, perché le due tecnologie non sono mutuamente esclusive.

Per **sviluppare una applicazione Web basata sul Web Forms** è sufficiente creare un progetto di tipo "ASP.NET Web Application", disponibile per Visual Basic e per C#.

Visual Studio 2010 fornisce dei template di siti più funzionali rispetto alla versione precedente: creando il nuovo progetto otteniamo già uno scheletro di applicazione con alcune funzionalità funzionanti, come la gestione utenti, su questa base si potranno poi sviluppare le proprie pagine.

Per i più esperti di progetti Web è possibile richiedere la creazione di un progetto completamente vuoto, senza nessuna pagina preesistente, per creare tutto da zero.

Figura 1. I file creati in un nuovo progetto asp.net



L'anatomia di un progetto ASP.NET è sicuramente complessa e non è certamente possibile coprire tutti gli argomenti in questa guida, ma è interessante comunque avere una visione globale delle varie parti che compongono un sito.

Tra le novità, si può notare come nella cartella Scripts venga inclusa automaticamente **la libreria jQuery**, chiaro segno che in questa nuova versione le funzionalità Ajax hanno un ruolo di primo piano.

Tra i file presenti nella radice del progetto **il web.config** riveste un ruolo unico, perché contiene tutte le impostazioni del sito e dell'ambiente, come le stringhe di connessione, le opzioni di ASP.NET e le configurazioni di IIS (solo per IIS 7.0 o 7.5).

Altro file molto importante è **il global.asax** che permette di interagire con gli eventi dell'applicazione ASP.NET, come ad esempio intercettare gli errori non gestiti, gli eventi di inizializzazione e chiusura applicazione etc.

Se apriamo la cartella `Account` notiamo la presenza di un ulteriore `web.config`; ASP.NET supporta infatti il concetto di **configurazione annidata** ed è quindi possibile cambiare alcuni parametri per cartelle specifiche.

Aprendo **il web.config della cartella 'account'** troviamo:

```
<?xml version="1.0"?>
<configuration>
  <location path="Register.aspx">
    <system.web>
      <authorization>
        <allow users="*" />
      </authorization>
    </system.web>
  </location>

  <system.web>
    <authorization>
      <deny users="?" />
    </authorization>
  </system.web>
</configuration>
```

Il template suggerisce di usare il `web.config` secondario per **modificare i permessi di accesso alla cartella Accounts**. In questo esempio si è impostato per la pagina `Register.aspx` un livello di accesso `<allow users="*" />` ovvero accesso permesso a chiunque, anche agli utenti non registrati; questo requisito è necessario dato che la pagina serve proprio per creare un nuovo account e deve quindi essere accessibile da tutti.

Nel nodo `system.web` si imposta invece il comportamento di default, che vale pertanto per tutte le pagine la cui autorizzazione non è stata impostata esplicitamente; il livello di accesso in questo caso è `<deny users="?" />` ovvero l'accesso viene negato a tutti gli utenti che non hanno effettuato un login corretto.

Aperto il `web.config` principale si nota infine una stringa di connessione che utilizza `SQLEXPRESS` per effettuare l'attach di un database chiamato `aspnet.mdf` (situato nella cartella dati) già impostato per supportare la gestione degli utenti.

L'editor di Visual Studio 2010

I progetti ASP.NET più che siti sono delle vere e proprie applicazioni e nel progetto sono infatti compresi una serie di file che vengono "compilati" in un "qualcosa" che può essere eseguito da IIS. Come abbiamo visto nella lezione precedente, il progetto Web appare proprio come un normale progetto di tipo Windows e viene quindi gestito interamente da Visual Studio.

I file che definiscono le pagine hanno estensione `.aspx` e sono composti logicamente da più parti: la principale è il file con estensione `.aspx` che contiene il codice HTML ed i controlli ASP.NET, abbiamo poi un file con estensione `.designer.cs` (`designer.vb` nei progetti VB.NET) in cui il designer di Visual Studio inserisce il codice autogenerato (questo file non va mai modificato) ed infine il file con estensione `.aspx.cs` che contiene il proprio codice.

Se apriamo le proprietà del progetto, possiamo visualizzare, divise per tab, le diverse opzioni supportate da un progetto di tipo Web. Nel primo tab, chiamato "application", possiamo scegliere il framework da utilizzare, che di default viene impostato a `.NET Framework 4.0`. Cambiando questo valore è possibile gestire siti che si appoggiano alle versioni precedenti di ASP.NET, pur restando nell'editor di VS2010; questa capacità si chiama **multitargeting** e permette di usufruire delle nuove funzionalità dell'IDE senza dovere forzatamente utilizzare la versione di framework più recente.

Nel **tab di opzioni Web**, è invece possibile specificare altre opzioni, come si vede in questa figura:

Figura 2. Le opzioni Web per un progetto ASP.NET

Start Action

Current Page

Specific Page

Start external program

Command line arguments

Working directory

Start URL

Don't open a page. Wait for a request from an external application.

Servers

Apply server settings to all users (store in project file)

Use Visual Studio Development Server

Auto-assign Port

Specific port

Virtual path:

NTLM Authentication

Enable Edit and Continue

Use Local IIS Web server

Project Url:

Override application root URL

Use Custom Web Server

Server Url:

Debuggers

ASP.NET Native Code SQL Server Silverlight

L'opzione **Start Action** viene applicata solamente in fase di sviluppo: in un progetto Web non esiste il concetto di "pagina di partenza", poiché l'utente può navigare direttamente su qualsiasi pagina del sito. Lo scopo di questa opzione è definire la pagina visualizzata quando si fa partire l'applicazione da Visual Studio, premendo ad esempio il tasto F5. Il valore di default è `current page` che apre il browser direttamente sulla pagina su cui si sta lavorando. Altre opzioni valide sono: specificare una pagina del sito, un URL, oppure eseguire un programma esterno.

Immediatamente a seguire si trovano le **opzioni relative al server**, in questo caso l'opzione di default è `use Visual Studio Development Server` che esegue l'applicazione in un **server integrato** fornito con Visual Studio, in questo modo anche chi non ha installato IIS nel proprio sistema operativo può visualizzare le pagine ed effettuare il debug del codice. Tra le altre opzioni disponibili vi è chiaramente quella di usare un IIS locale oppure un web server custom. Le altre opzioni riguardano opzioni più avanzate e possono essere per ora ignorate.

Master page, creare un layout con Visual Studio

L'utilizzo di un layout comune a tutte le pagine di un sito o comunque distinto per aree è una prassi abbastanza consolidata. Per questo è necessario definire il layout in una pagina base per poter poi cambiare il solo contenuto dinamico in tutte le altre pagine; questo concetto in ASP.NET è supportato dalle **pagine master** (master page). Nel progetto base è presente una pagina chiamata `site.master` che contiene il layout su cui è basato tutto il sito; la parte più importante è la seguente:

```
<div class="main">
  <asp:ContentPlaceHolder ID="MainContent" runat="server"/>
</div>
```

Il tag `asp:ContentPlaceHolder` identifica un controllo ASP.NET, ovvero un componente capace di generare una parte di HTML nella pagina; in questo specifico caso il controllo non genera nulla, ma stabilisce il punto dove le pagine figlie andranno ad inserire il loro contenuto. Grazie alle master page è possibile quindi creare un layout base specificando dove le pagine figlie andranno ad inserire il loro contenuto.

Nella `site.master` sono presenti due placeholder, il primo è quello del contenuto ed è piazzato nel `div` principale mentre il secondo è posizionato nel `taghead` e garantisce la possibilità per le pagine figlie di inserire del contenuto nell'`header` della pagina.

Il restante codice della pagina master serve a renderizzare le parti comuni, come ad esempio la sezione in alto a destra, preposta alla gestione del login.

```
<div class="loginDisplay">
  <asp:LoginView ID="HeadLoginView" runat="server" EnableViewState="false">
    <AnonymousTemplate>
      [ <a href="~/Account/Login.aspx" ID="HeadLoginStatus" runat="server">Log
In</a> ]
    </AnonymousTemplate>
    <LoggedInTemplate>
      Welcome <span class="bold"><asp:LoginName ID="HeadLoginName "
runat="server" /></span>!
      [ <asp:LoginStatus ID="HeadLoginStatus" runat="server"
LogoutAction="Redirect" LogoutText="Log Out" LogoutPageUrl="~/"/> ]
    </LoggedInTemplate>
  </asp:LoginView>
</div>
```

In questo caso si è utilizzato il controllo ASP.NET chiamato `LoginView`, che permette di visualizzare porzioni di codice HTML differenti sulla base dello stato del login dell'utente; nell'esempio viene infatti mostrato un link alla pagina di login per gli utenti non loggati ed un messaggio di benvenuto per chi invece ha effettuato un login corretto. Assieme al messaggio di benvenuto è presente un controllo `LoginStatus` che permette di effettuare il `Logout`. L'unico altro controllo ASP.NET presente è un controllo chiamato `Menu` che si occupa di generare il menu di navigazione.

La home page del sito, chiamata `default.aspx`, è così definita:

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master"
AutoEventWireup="true"
    CodeBehind="Default.aspx.cs" Inherits="AspnetWfIntro._Default" %>

<asp:Content ID="HeaderContent" runat="server"
ContentPlaceHolderID="HeadContent">
<!-- qui mettiamo le definizioni come il titolo della pagina o le librerie JS da
caricare -->
</asp:Content>

<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2>Benvenuto in ASP.NET!</h2>

    <p>Per saperne di più su ASP.NET visita <a href="http://aspnet.html.it"
title="ASPNet.HTML.it">ASPNet.HTML.it</a>.</p>

</asp:Content>
```

L'anatomia di una pagina ASP.NET comprende sempre come primo tag la direttiva `<%@ page >` che contiene le impostazioni specifiche per la pagina corrente.

In questo caso l'impostazione `MasterPageFile` permette di specificare la master page che si vuole utilizzare, di seguito troviamo altre opzioni che indicano al motore di ASP.NET in che file si trova il codice della pagina, etc.

Tutto il contenuto specifico della pagina è quindi inserito in un controllo di tipo `asp:Content`, la cui proprietà `ContentPlaceHolderID` permette di scegliere il placeholder dove il contenuto verrà inserito.