UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

FACOLTÀ DI INGEGNERIA - SEDE DI MODENA

Corso di Diploma di Laurea in Ingegneria Informatica

Re-engeeneering di una Rete Civica:

Implementazione mediante la nuova tecnologia ASP.NET

Relatore:

Chiar.mo Prof**.**

Sonia Bergamaschi

Correlatori:

Ing. Lorenzo Canali

Fabio Neri

Tesi di laurea di:

Daniele Bergonzini

Anno Accademico 2001/2002

SOMMARIO:

Elenco figure

Ringraziamenti

Desidero ringraziare tutti coloro che mi hanno aiutato nello svolgere questa tesi.
In modo particolare ringrazio i miei genitori e tutti gli amici, per il sostegno che mi hanno dato standomi vicino e per la pazienza che hanno dimostrato in questo periodo.

1. Introduzione

La Tesi da me realizzata rappresenta un esempio di applicazione Web con accesso a database. Il punto di partenza è un'applicazione Web realizzata con la tecnologia ASP, una rete civica appunto! La rete civica esistente si compone di vari servizi che vengono offerti agli utenti della rete.

Ogni servizio si avvale di letture a Database, contenenti informazioni inserite dagli addetti all'amministrazione della rete e non dai progettisti, il che sgrava notevolmente il lavoro di creazione della rete da parte della ditta! Il personale "amministratore" fa parte del comune cui viene venduta la rete, e non deve essere necessariamente personale con avanzate conoscenze informatiche ma è sufficiente che conosca le basi dell'uso di un computer e di Internet.

Una rete civica graficamente è molo bella, ma è molto complessa da realizzare e le righe di codice sono tante, ciò che io ho dovuto fare è studiare un nuovo ambiente di sviluppo e con esso riscrivere quello che altri in passato hanno realizzato, possibilmente correggendo errori e apportando miglioramenti, ove possibile, il risultato sono state ovviamente in altrettante righe di codice.

ORGANIZZAZIONE DEI CAPITOLI:

Questa tesi è stata strutturata in otto capitoli più un glossario

Nei primi capitoli, il *primo* e il *secondo*, si illustra il progetto sviluppato e gli strumenti utilizzati.

Dopo una introduzione inizia la parte di presentazione, rispettivamente nei capitoli *tre* e *quattro*; vengono presentate le caratteristiche della nuova piattaforma Microsoft, partendo dalla descrizione del Framework .Net fino all'ambiente integrato Visual Studio .Net, passando per i nuovi linguaggi offerti dal .Net.

Nei capitoli centrali, ossia il *quinto* e il *sesto* e il *settimo*, viene illustrato il passaggio da ASP e ADO al nuovo modo di creare pagine dinamiche secondo Microsoft: ASP.NET e ADO.NET.

In fine l'ultimo capitolo, *l'ottavo*, illustra mediante esempi alcuni servizi realizzati con ASP.NET e ADO.NET, che fanno parte del progetto realizzato.

Dato che l'ambiente .NET è molto recente e contiene dei costrutti nuovi si è pensato di inserire un breve glossario per aiutare la lettura di termini nuovi.

2. Considerazioni generali

La fase di Re-engeeneering è stata realizzata con l'evoluzione della tecnologia Microsoft ASP che si chiama ASP.NET! ASP.NET è parte integrante di un nuovo ambiente di sviluppo chiamato .NET, che verrà largamente presentata in seguito, ed è una tecnologia per lo sviluppo nel Web.

Una rete civica può essere realizzata in diversi modi, dipende molto dalle richieste delle singole realtà locali, i comuni. Sicuramente realizzarla in modo che posa adattarsi alle varie richieste è la scelta migliore, comporta però un notevole sforzo iniziale, ma alla fine basteranno poche semplici modifiche per ottenere reti diverse con servizi diversi partendo da una base comune.

Questa è la strada scelta per la riprogettazione della rete civica che si è cominciata a trasformare, gli strumenti che ho avuto a disposizione si sono rivelati molto utili per questo.

Prima di iniziare la riprogettazione sono state esplicitate le specifiche di progetto, indispensabili per un'applicazione di questo tipo, basti pensare che il sito, rete civica, non è composto solo da una parte di rappresentazione dei dati, ma anche da una amministrativa dove un utente può aggiungere, modificare, cancellare le informazioni della rete. Queste azioni sono diverse da utente a utente, una persona avrà alcuni diritti sulla rete mentre un'altra ne avrà altri. Questo è possibile grazie alla presenza di un sistema d'accesso all'area amministratore che discrimina, mediante ID e Password, l'utente che sta accedendo al sistema.

Le specifiche riguardano ad esempio il come realizzare i database; quali caratteristiche dei vari servizi si possono generalizzare, in modo da poter richiamare funzioni esterne, senza aumentare il numero di righe con codice doppio; quali pagine sono da realizzare a se stanti e quali invece possono essere riutilizzate con piccoli cambiamenti usando come discriminante una semplice variabile, ecc..

Il lavoro svolto è una minima parte di quello che si ha in mente di creare, lo scopo è di rendere il tutto più dinamico possibile, le difficoltà incontrate sono state tante e altre se ne incontreranno, ma il risultato finale è sicuramente sorprendente, pochi cambiamenti permetteranno di ottenere siti diversi non solo graficamente.

2.1. Strumenti utilizzati

Per la realizzazione della rete civica sono stati utilizzati i seguenti strumenti:

Hardwere:

- Piattaforma come da requisiti Microsoft per progettare ed eseguire applicazioni Web .NET

Softwere:

- Sistema operativo: Microsoft Windows 2000 Professional (SP2) con IIS 5.0.
  Sono stati eseguiti, come richiesto dalle specifiche Microsoft per eseguire il Framework, tutti gli aggiornamenti disponibili nel sito Microsoft.

- Ambiente di Programmazione Integrato Visual Studio Professional affiancato dai semplici compilatori a riga di comando!

- Il RDBMS (Relational DataBase Management System) utilizzato è SQL Server 7.0

- Il linguaggio di Programmazione scelto per programmare le pagine ASP.NET è C#.

3. Presentazione di .NET

Il Microsoft .NET Framework è una piattaforma per sviluppare, distribuire ed ospitare applicazioni e servizi Web XML.

Il Microsoft® .NET Framework e Microsoft Visual Studio® .NET rientrano, nel senso più ampio del termine, alla categoria dello *sviluppo software*. L'area più avanzata di tale categoria è rappresentata, secondo il parere di clienti ed analisti di mercato, dal segmento dello *sviluppo di applicazioni distribuite*. La categoria dello sviluppo software comprende tutti gli strumenti ed i prodotti di sviluppo, mentre il segmento relativo alle applicazioni distribuite include solo gli strumenti finalizzati alla realizzazione di applicazioni lato server e lato client.

Se si esaminano le necessità degli sviluppatori e delle aziende, è possibile stilare un elenco di criteri che consentono di valutare le funzionalità del .NET Framework e Visual Studio .NET in relazione alla rapidità di sviluppo ed alla semplicità di integrazione della nuova generazione di applicazioni e servizi Web XML. Tra questi vi sono:

Aspetti che influenzano i tempi di rilascio:

- *Possibilità di utilizzare un qualsiasi linguaggio di programmazione*. Consentire agli sviluppatori di adottare un qualunque linguaggio di programmazione. Integrare applicazioni scritte con linguaggi differenti.

- *Accesso a strumenti di sviluppo allo stato dell'arte*. Rilasciare strumenti di sviluppo progettati accuratamente come il debug ed il profiler integrati.

- *Miglioramenti nella struttura del codice*. Mettere a disposizione una struttura altamente componentizzata, che non richiede la scrittura di codice di infrastruttura, e che consente agli sviluppatori di concentrarsi sull'implementazione della logica di business.

Aspetti che facilitano l'integrazione:

- *Fornitura di software come servizio*. Creare e distribuire il software come servizio. Impiegare standard e protocolli basati su XML e sulla famiglia SOAP degli standard di integrazione.

- *Standard riconosciuti per linguaggio ed infrastruttura*. L'utilizzo di standard rappresenta il requisito fondamentale per poter distribuire il software come servizio. Microsoft ha sottoposto le specifiche del linguaggio di programmazione C# ed un sottoinsieme del .NET Framework, la Common Language Infrastructure, all'ECMA per la standardizzazione. Tali specifiche, frutto di una collaborazione tra sei partner ECMA tra i quali Hewlett-Packard ed Intel, dovrebbero ottenere l'approvazione formale dall'assemblea generale dell'ECMA entro l'anno.

- *Facilità di accesso ai dati*. Rendere disponibile un'interfaccia efficiente verso qualsiasi database, che sia progettata appositamente affinché le applicazioni Web possano adottare una metodologia di accesso ai dati debolmente accoppiata. Utilizzare l'XML come formato nativo per i dati.

Aspetti che migliorano l'operatività:

- *Implementare la sicurezza sulla base delle evidenze*. Mettere a disposizione un modello di sicurezza basato sull'evidenza, che consenta un controllo puntuale a livello di singolo metodo che condiziona le funzionalità di un'applicazione sulla base di informazioni quali l'autore del codice, la funzionalità richiesta, il percorso di installazione e l'identità dell'utente.

- *Aumentare l'affidabilità delle applicazioni*. Rendere le applicazioni più affidabili grazie a tecnologie, quali la gestione automatica della memoria ad elevate prestazioni ed il monitoraggio di applicazioni Web basato su eventi, con un controllo fine ed automatico dei riavvii.

- *Aumentare in modo significativo le prestazioni*. In definitiva, migliorare le prestazioni di applicazioni e servizi Web.

- *Un'offerta completa per fornire tutti i servizi necessari alla realizzazione di soluzioni di livello enterprise*. Queste includono servizi applicativi quali object request broker, monitor per la gestione delle transazioni, motori di scripting, server Web ricchi di funzionalità, gestione completa della messaggistica, connettività a database, sicurezza ed un'infrastruttura per il monitoraggio e la gestione.

- *Un modello di programmazione coerente ed unificato per applicazioni intranet/Internet*. Connesso alle architetture debolmente accoppiate, questo criterio va a cercare quelle soluzioni che consentano alle aziende di apprendere, creare e gestire un modello di programmazione semplice per applicazioni intranet o Internet in modo da ridurre i costi di formazione e di gestione.

- *Un modello di programmazione coerente ed unificato per i client Web più disparati, oltre ad interfacce client evolute e nuove periferiche intelligenti*. I clienti possono oggi prendere in considerazione un'ampia gamma di periferiche client, tra le quali PC, browser, PDA e telefoni cellulari.

3.1. Il Framework

Il .NET Framework è il risultato della fusione di due progetti. L'obiettivo del primo progetto consisteva nel migliorare lo sviluppo per Windows, con un occhio di riguardo verso COM, il Microsoft Component Object Model. Il secondo aspirava alla creazione di una piattaforma per la distribuzione del software sotto forma di servizio. Questi due progetti sono arrivati a fondersi più di tre anni fa.

Quando è stato creato, il Web era fondamentalmente un file system di sola lettura con in più il vantaggio di utilizzare standard e protocolli aziendali, il che consentiva di accedere facilmente al contenuto dei file. I pochi siti Web interattivi erano tipicamente estensioni verso l'esterno di applicazioni two-tier esistenti.

I primi prodotti sviluppati per il Web erano generalmente scritti con il linguaggio di programmazione C e la Common Gateway Interface (CGI). Oltre a ciò, la maggior parte delle applicazioni Web si basava su architetture two-tier, fatto che ostacolava la scalabilità e l'integrazione tra le applicazioni. Le applicazioni Web venivano progettate per essere eseguite esclusivamente all'interno delle pagine che le ospitavano; in altre parole, interfaccia utente e logica applicativa coincidevano. Come conseguenza, era difficile collegare più applicazioni Web al fine di ottenere soluzioni più articolate. Grazie ai miglioramenti apportati al Microsoft Component Object Model (COM) ed al rilascio di tecnologie come l'Active Server Pages (ASP) nel 1996, i siti Web sono stati in grado di offrire soluzioni sempre più interattive. ASP ha consentito di invocare in modo semplice la logica di business ed i servizi di piattaforma necessari utilizzando semplici linguaggi di script. Il supporto di COM ha semplificato la realizzazione di applicazioni grazie alla sua capacità di incapsulare la logica di business in unità modulari che possono essere scritte utilizzando un'ampia varietà di linguaggi, quali Microsoft Visual Basic®, C++ o COBOL.

Figura 1: XML Service

I SERVIZI WEB XML

Per vincere d'ora in avanti le sfide legate allo sviluppo per Internet, è necessario scrivere applicazioni utilizzando un qualsiasi linguaggio di programmazione, accedere a qualunque piattaforma e garantire una totale scalabilità. Questa strategia di sviluppo è molto interessante, dal momento che consente alle aziende di utilizzare l'hardware, le applicazioni e gli sviluppatori esistenti senza obbligare questi ultimi ad apprendere un nuovo linguaggio di programmazione.

Un servizio Web XML è un'applicazione che espone le proprie funzionalità programmaticamente su Internet o intranet, utilizzando protocolli Internet standard come HTTP e XML. Un servizio Web XML può essere paragonato a un componente, o una scatola nera, che offre funzionalità specifiche ai client, definiti anche consumer. Come il modello DCOM (Distributed Component Object Model) rappresenta un'estensione del modello COM, un servizio Web XML è un componente con una portata effettivamente globale. Diversamente da DCOM, RMI, IIOP o altri protocolli comunemente utilizzati e specifici del modello di programmazione orientato a oggetti, un consumer ottiene l'accesso a un servizio Web XML tramite un protocollo standard, collaudato e di vasta diffusione come HTTP, e un formato di dati universale basato su XML.

RMI e IIOP

RMI è l'acronimo di Remote Method Invocation. Si tratta del protocollo specifico utilizzato dalla piattaforma Java per l'intercomunicazione tra componenti lava. IIOP è l'acronimo di Internet Inter-ORB Protocol. Anche questo protocollo viene utilizzato nel mondo Java per supportare la comunicazione tra componenti Java e non Java conformi al modello CORBA. CORBA è l'acronimo di Common Object Request Broker Architecture. Si tratta di uno schema definito dal gruppo OMG (Object Management Group) e progettato per permettere la comunicazione tra componenti distribuiti sviluppati con linguaggi diversi.[1]

Per l'implementazione di un servizio Web si possono utilizzare vari linguaggi. Attualmente nell'ambito di Microsoft. NET Framework sono supportati C++, Microsoft JScript, Microsoft Visual C# e Microsoft Visual Basic .NET ed è prevista l'estensione del supporto ad altri linguaggi nel prossimo futuro. Nella piattaforma Microsoft .NET, i servizi Web sono basati su XML. Dal punto di vista del consumer, tuttavia, il linguaggio utilizzato e anche le modalità di funzionamento del servizio Web sono irrilevanti. Per il consumer, un servizio Web XML è semplicemente un'interfaccia che espone un certo numero di metodi ben definiti e pertanto il consumer non deve far altro che richiamare tali metodi utilizzando i protocolli Internet standard, passare i parametri in formato XML e ricevere le risposte in formato XML.

Uno degli aspetti più innovativi e significativi di .NET Framework è sicuramente il concetto di "Web programmabile". Tale concetto si basa sull'idea che in futuro i sistemi verranno costruiti tramite i dati e i servizi offerti dall'interazione di numerosi Web Sevice XML. Nell'ambito di questi sistemi il Web è lo strumento di accesso e gli sviluppatori hanno il compito fondamentale di integrarli in modi significativi.

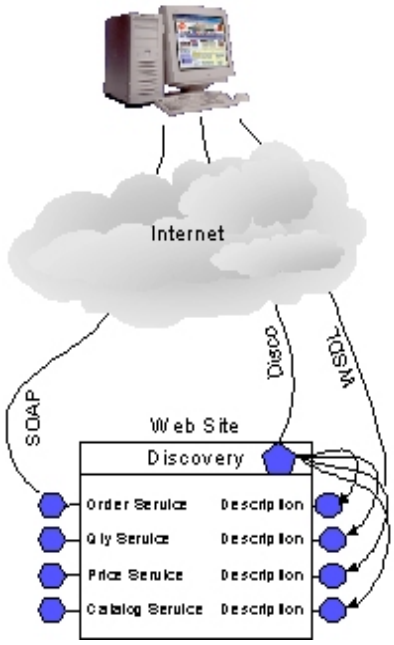


Figura 2: Come accedere ad un Web Service XML

I servizi Web XML possono essere specifici dell'applicazione e pertanto limitati ad ambiti di utilizzo particolari. Esistono comunque numerosi servizi comuni, definiti per questo orizzontali, che possono risultare utili in varie applicazioni. Tra questi Microsoft .NET Passport, che permette di accedere a più siti Web con informazioni di registrazione uniche, e Microsoft .NET Alerts, un servizio di notifica utilizzato dalle aziende per informare i clienti abbonati di offerte ed eventi speciali. Esiste inoltre un mercato affermato per i servizi Web XML verticali di terze parti, che implementano alcune funzionalità comuni a segmenti specifici, per esempio nei settori finanziario o manifatturiero. Benché possa sembrare ovvio, è opportuno sottolineare l'importanza cruciale di questi servizi per lo sviluppo di applicazioni Business-To-Business (B2B) e Business-To-Consumer (B2C). [1]

GLI OBIETTIVI DI PROGETTAZIONE DEL .NET FRAMEWORK

L'obiettivo della piattaforma del .NET Framework è quello di combinare un semplice paradigma di programmazione con i protocolli aperti e scalabili di Internet.

- Integrazione attraverso gli standard pubblici di Internet :
  Per poter comunicare con partner, clienti, divisioni geograficamente separate e persino con applicazioni future, le soluzioni di sviluppo devono offrire sia il supporto per gli standard aperti di Internet sia una profonda e trasparente integrazione con tali protocolli che non obblighi lo sviluppatore a conoscere l'infrastruttura sottostante.

- Scalabilità attraverso un'architettura debolmente accoppiata:
  I sistemi più grandi e scalabili al mondo sono stati realizzati su architetture asincrone basate su messaggi. Il .NET Framework è stato creato per combinare i vantaggi di produttività delle architetture fortemente accoppiate con la scalabilità e l'interoperabilità di quelle debolmente accoppiate.

- Supporto multi linguaggio:
  Il .Net Framework consente di integrare tra loro applicazioni scritte con linguaggi differenti permettendo di sfruttare il proprio patrimonio di conoscenze, ed agli sviluppatori di programmare nel linguaggio che preferiscono.

- Aumentare la produttività degli sviluppatori:
  Il .NET Framework include funzionalità che consentono di risparmiare tempo, quali transazioni automatiche semplici da utilizzare, la gestione automatica della memoria ed un ricco insieme di controlli che incapsulano molte funzioni di uso comune.

- Proteggere gli investimenti con le opzioni avanzate di sicurezza :
  L'architettura di sicurezza del .NET Framework è stata progettata da zero per consentire la protezione di dati ed applicazioni attraverso un modello di sicurezza a granularità fine e basato sulle evidenze.

- Utilizzare i servizi del sistema operativo:
  Windows mette a disposizione un ricco insieme di servizi quali un accesso ai dati universale, la sicurezza integrata, interfacce utente interattive, un modello ad oggetti maturo e basato su componenti, monitor per la gestione delle transazioni e servizi per la gestione delle code di messaggi. Il .NET Framework sfrutta tutta la ricchezza di servizi di Windows e la espone in un modo semplice da utilizzare.

LE TECNOLOGIE DI BASE DEI SERVIZI WEB XML BASATE SU SOAP

I servizi Web XML sono alla base delle soluzioni che indirizzano tematiche tipiche nell'integrazione e distribuzione del software come servizio. Questi forniscono un modello basato su standard per legare tra loro applicazioni sull'infrastruttura Internet esistente. Le applicazioni Web possono essere facilmente assemblate con servizi nuovi o esistenti, indipendentemente dalla piattaforma, dal linguaggio di sviluppo o dal modello ad oggetti utilizzato per implementare i servizi o le applicazioni costituenti.
La chiave per far sì che i servizi Web XML siano compatibili con l'infrastruttura eterogenea del Web consiste nell'avere applicazioni che condividono un semplice formato di descrizione dei dati, basato su XML. Nel seguito vengono descritti tali formati ed i relativi principi di funzionamento:

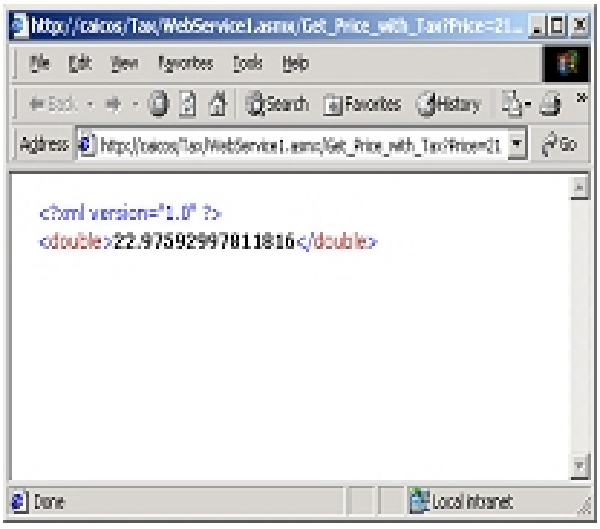


Figura 3: Esempio di codice XML restituito da un Web Service XML

EXTENSIBLE MARKUP LANGUAGE (XML)

L'XML è il principale costituente dei Web Service XML. La sua forza è costituita dall'enorme flessibilità nel trattare qualsiasi tipo di dati indipendentemente dalla piattaforma. Un esempio di codice XML restituito da un Web Service XML è mostrato nella figura.

SIMPLE OBJECT ACCESS PROTOCOL (SOAP)

A basso livello, i sistemi necessitano di parlare lo stesso linguaggio. In particolare, le applicazioni che comunicano devono possedere un insieme di regole per la rappresentazione di tipi di dati differenti (ad esempio interi e matrici) e di comandi (ad esempio come trattare i dati). Inoltre, le applicazioni hanno bisogno di un metodo per estendere il linguaggio a seconda delle necessità. Il Simple Object Access Protocol (SOAP), una grammatica XML, rappresenta un insieme comune di regole per la rappresentazione e l'estensione di tipi di dato e comandi. SOAP è stato accettato dal W3C per la standardizzazione. [1]

SOAP è un protocollo leggero basato su HTTP. Si possono utilizzare altri protocolli per lo scambio di messaggi SOAP, ma attualmente sono stati definiti unicamente i binding HTTP. Tale protocollo definisce la sintassi XML necessaria per specificare i nomi dei metodi che un consumer desidera richiamare su un servizio Web XML, per definire i parametri e i valori restituiti, nonché per descrivere i tipi dei parametri e dei valori restituiti. Quando un client richiama un servizio Web XML dovrà specificare il metodo e i relativi parametri tramite questa sintassi XML. [1]
La funzione del protocollo SOAP è quella di migliorare l'interoperabilità tra piattaforme diverse. Il punto di forza di SOAP è sicuramente la semplicità, oltre al fatto che si basa su altre tecnologie standard come HTTP e XML.

La specifica SOAP definisce vari aspetti, tra i quali i più importanti sono i seguenti:

- Formato di un messaggio SOAP.
- Modalità di codifica dei dati.
- Modalità di invio dei messaggi (chiamate dei metodi).
- Modalità di ricezione delle risposte.

WEB SERVICE DESCRIPTION LANGUAGE (WSDL)

Una volta che le applicazioni possiedono delle regole generali per rappresentare i tipi di dato ed i comandi, hanno bisogno di un modo per descrivere i particolari dati e comandi che accettano. Non è sufficiente che un'applicazione dichiari di accettare interi; talvolta, è necessario che vi sia un modo per dichiarare che, se si passano due interi, li si dovrà moltiplicare. Il Web Services Description Language (WSDL) è una grammatica XML che consente a sviluppatori e strumenti di sviluppo di descrivere le funzionalità di un servizio Web XML.

SOAP DISCOVERY ("DISCO")

Oltre al WSDL, occorre un insieme di regole per individuare la descrizione di un servizio Web XML – dove deve andare a guardare uno strumento per ottenere informazioni sulle funzionalità di un servizio Web XML? Le specifiche del SOAP Discovery forniscono un insieme di regole per reperire automaticamente i file di descrizione WSDL di un particolare sito Web.

UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION (UDDI)

UDDI è una directory, come lo schedario di una biblioteca, che fornisce un metodo per individuare tutte le tipologie di servizi Web. Le specifiche dell'UDDI comprendono:

- le pagine bianche, che forniscono informazioni sulle aziende
- le pagine gialle che organizzano i servizi Web in categorie (ad esempio, "servizi di autorizzazioni per le carte di credito")
- le pagine verdi che forniscono informazioni tecniche dettagliate sui singoli servizi.

I TRE COMPONENTI

Il .NET Framework è costituito da tre parti principali: il Common Language Runtime, un insieme gerarchico di librerie di classe ed una versione componentizzata del Microsoft Active Server Pages denominata Microsoft® ASP .NET.
Il Common Language Runtime è uno strato posto al di sopra dei servizi del sistema operativo. Esso è responsabile dell'esecuzione vera e propria delle applicazioni - assicura che vengano rispettate tutte le dipendenze, gestisce la memoria, la sicurezza, l'integrazione del linguaggio e così via. Il runtime fornisce numerosi servizi che consentono di semplificare la stesura del codice, la distribuzione dell'applicazione e di migliorare l'affidabilità della stessa.
Gli sviluppatori, in realtà, non interagiscono direttamente con il runtime, ma utilizzano un insieme di classi unificate. Tali classi possono essere adoperate da qualsiasi linguaggio di programmazione.

Il .NET Framework include, come parte delle librerie di classe, un modello di programmazione per applicazioni Web denominato ASP .NET che fornisce componenti e servizi di più alto livello creati appositamente per lo sviluppo di applicazioni e servizi Web XML.

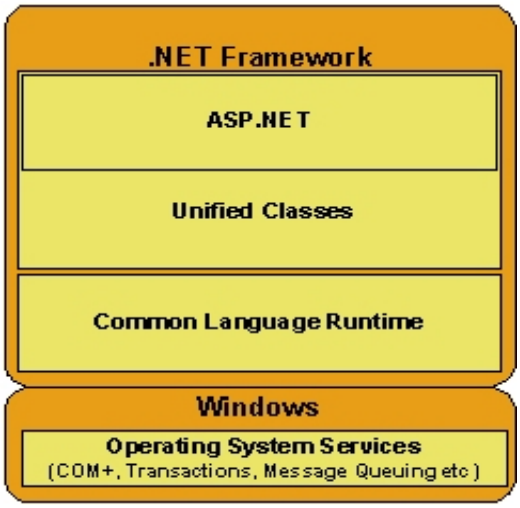


Figura 4: Componenti del Framework

CLASSI DI SISTEMA

Le classi del .NET Framework costituiscono un insieme di librerie di classe (API) unificato, orientato all'oggetto, gerarchico ed estensibile che gli sviluppatori possono utilizzare dai linguaggi che già conoscono.

Il .NET Framework fornisce classi che possono essere invocate da qualsiasi linguaggio di programmazione. Tali classi si conformano ad un insieme di criteri di nomenclatura e di progettazione che riducono ulteriormente la curva di apprendimento degli sviluppatori. Alcune delle librerie di classe principali sono illustrate nella figura a lato.

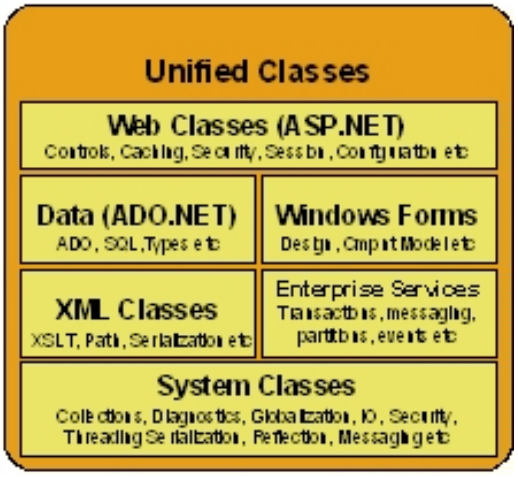


Figura 5: Classi unificate

Il .NET Framework include un insieme di base di librerie di classe che deve essere presente in ogni libreria standard, come le classi collection, le classi input/output, le classi data type e le classi numeriche. Oltre a ciò, sono presenti classi che consentono di accedere a tutti i servizi del sistema operativo, come la grafica, la gestione delle reti, la

gestione dei thread, la localizzazione, la crittografia, l'accesso ai dati, classi che vengono utilizzate dagli strumenti di sviluppo come il debug ed infine classi che forniscono i servizi necessari alla realizzazione di applicazioni su scala aziendale come le transazioni, gli eventi, le partizioni e la messaggistica.

Le classi di programmazione unificate offrono anche pieno supporto per lo sviluppo di applicazioni Windows di tipo tradizionale (naturalmente, tali applicazioni, a loro volta, possono utilizzare i servizi Web XML).

ORIENTAMENTO DELLE CLASSI

La Class Library .NET è un'ampia gerarchia organizzata di classi. Questi oggetti indicano i servizi da usare per sviluppare servizi personalizzati. Includono supporto per Windows form e servizi Web, nonché oggetti per lavorare con XML e dati. Per includere questi servizi nelle applicazioni, si esplora la gerarchia usando i principi tradizionali della programmazione orientata agli oggetti. Esplorare questa gerarchia non è la stessa cosa di esplorare le cartelle sul disco rigido. Per esempio, per fare riferimento a un driver SQL specifico su un sistema, dovrei scrivere "c:\WINNT\System\Data\SQLClient\". Un riferimento simile incluso nel codice sarebbe "System.Data.SQLClient".

L'unica differenza è che i riferimenti nel codice orientato agli oggetti separano ogni livello della gerarchia con un punto. Questi riferimenti espliciti a gruppi di classi nelle librerie di classi del Framework sono definiti spazi dei nomi in .NET. Gli spazi dei nomi sono classi di organizzazione, proprio come le cartelle organizzano i file nel file system.

Uno ***spazio dei nomi*** è un riferimento gerarchico univoco a una classe specifica o gruppo di classi simili. Per esempio. le classi base che offrono i servizi che supportano l'ambiente di runtime si trovano nello spazio dei nomi System. Questo spazio dei nomi include servizi come I/O. sicurezza, dati e operazioni Web. Per accedere a questi spazi dei nomi, il riferimento sarà System.Web o System.Data. Più specifico è lo spazio dei nomi, più specifici saranno i suoi servizi. Per esempio, per connettersi a un database SQL, si può usare lo spazio dei nomi System.Data.SqlClient.

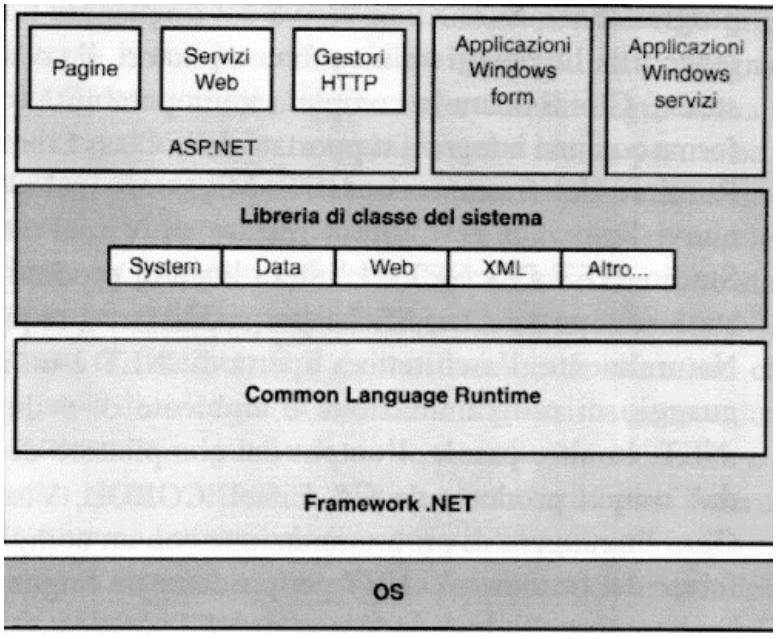


Figura 6: Livelli del Framework .NET

Il riferimento a una classe univoca evita che due oggetti con lo stesso nome entrino in conflitto. Si immagini di avere un'applicazione che consente ai client remoti di chiamare i servizi Web nell'oggetto Customers.Customer. Se il client remoto ha un riferimento locale alla propria versione di un oggetto di nome Customers.Customer si potrebbero verificare problemi. Microsoft consiglia di creare almeno due livelli di spazi dei nomi univoci da usare come riferimenti principali per tutte le creazioni di oggetti. [4]

Il Common Language Runtime

Il Common Language Runtime rappresenta un motore di esecuzione. Il codice cui il runtime si riferisce e la cui esecuzione è gestita dal runtime, viene detto codice gestito. La responsabilità per attività quali la creazione di oggetti, l'esecuzione di chiamate a metodi e così via, è demandata al Common Language Runtime che consente al runtime di fornire servizi aggiuntivi al codice in esecuzione.
A dispetto del suo nome, il Common Language Runtime riveste in realtà un ruolo tanto nella fase di sviluppo di componenti quanto nella fase di esecuzione.

Mentre la componente è in esecuzione, il runtime fornisce servizi quali la gestione della memoria (inclusa la garbage collection), la gestione di processi e thread, il potenziamento della sicurezza e si preoccupa di soddisfare le eventuali dipendenze da altre componenti.

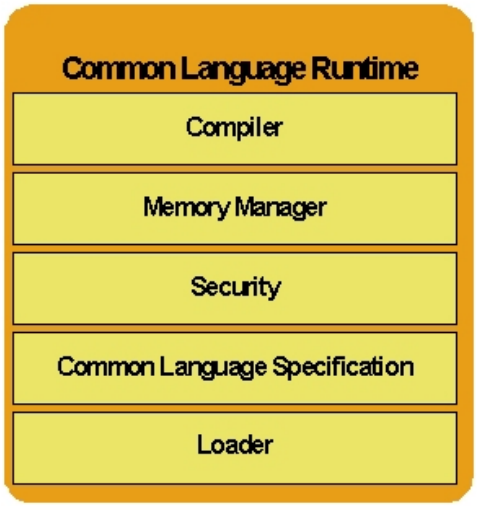


Figura 7: CLR

Durante la fase di sviluppo, il ruolo del runtime cambia leggermente. Grazie agli automatismi introdotti (ad esempio nella gestione della memoria), il runtime semplifica la fase di sviluppo. In particolare, caratteristiche quali la gestione del ciclo di vita, una nomenclatura forte dei tipi, la gestione delle eccezioni tra linguaggi, la gestione degli eventi basati sui delegate, il binding dinamico e la reflection, riducono considerevolmente la quantità di codice che occorre scrivere per trasformare la logica di business in componenti riutilizzabili.

Le caratteristiche fondamentali del runtime includono un sistema comune di tipi (che consente l'integrazione tra linguaggi), componenti autodescrittive, un deployment più semplice ed un miglior controllo dei conflitti di versione ed infine, servizi di sicurezza integrati.

SICUREZZA BASATA SULLE EVIDENZE

Il sistema di sicurezza del .NET Framework fornisce un controllo fine ed a livello di metodo su ciò che le applicazioni possono o meno fare in base alle intenzioni di chi ha scritto il codice, di dove il codice è stato installato e di chi lo esegue.

- Sicurezza per Internet:
  L'infrastruttura per la sicurezza del .NET Framework comprende funzionalità che consentono di gestire sia l'autenticazione degli utenti sia la sicurezza di accesso al codice, che rafforza i permessi sul codice sulla base di criteri di fiducia. Una libreria espandibile di funzioni di crittografia fornisce un facile accesso all'hashing e alla crittografazione del codice gestito, incluse le firme digitali per l'XML.

- Sicurezza per l'accesso al codice e sicurezza basata sui ruoli. :
  Il .NET Framework mette a disposizione la sicurezza attraverso due categorie di alto livello: la sicurezza per l'accesso al codice e la sicurezza basata sui ruoli. La sicurezza per l'accesso al codice è il meccanismo tramite il quale si può specificare il livello di accesso che il codice dovrebbe avere su risorse ed operazioni. La sicurezza basata sui ruoli viene utilizzata per controllare i permessi sulla base dell'identità dell'utente.

ENTERMEDIATE LANGUAGE (IL) E METADATI

Intermediate Language (IL) è la versione .NET del codice compilato. Sia che si compili un file DLL ASP.NET scritto in COBOL o un file EXE Windows form scritto in C#, il risultato sarà sempre lo stesso linguaggio intermedio autodescrittivo. IL è una semplice sintassi basata su testo che fa da complemento a un componente autodescrittivo, i metadati. La combinazione di queste due tecnologie offre al runtime gestito di .NET l'abilità di eseguire più operazioni in meno tempo e con meno overhead.

Le analogie tra il modello di runtime gestito di Java e di .NET confrontano spesso l'IL di .NET con il codice a byte di Java. Questa analogia deriva dai vantaggi di un codice compilato comune in grado di essere eseguito su qualsiasi computer che supporti i rispettivi ambienti di runtime. Tuttavia, in termini di forma e funzionamento, i due modelli sono completamente diversi. Il codice compilato di .NET (IL e metadati) offre vantaggi significativi rispetto alle compilazioni di codice a byte basato sugli interi di Java.[4]

Le applicazioni .NET sono compilate Just-In-Time una seconda volta nel codice macchina nativo. Si tratta dello stesso tipo di codice di linguaggio macchina in cui sarebbe compilato un eseguibile C++ a 32 bit. La sintassi basata su testo di un eseguibile consente al runtime di compilare un'applicazione nell'insieme più efficace di istruzioni macchina native per un dato sistema. Il runtime .NET offre allo sviluppatore opzioni di compilazione JustIn-Time (JIT) dell'applicazione in linguaggio macchina, o la possibilità di precompilare ogni cosa nel codice macchina nativo in fase di installazione. Per esempio, se si desidera evitare l'overhead della compilazione JIT, si può indicare al runtime di compilare l'applicazione in codice macchina nativo durante il processo di installazione iniziale.

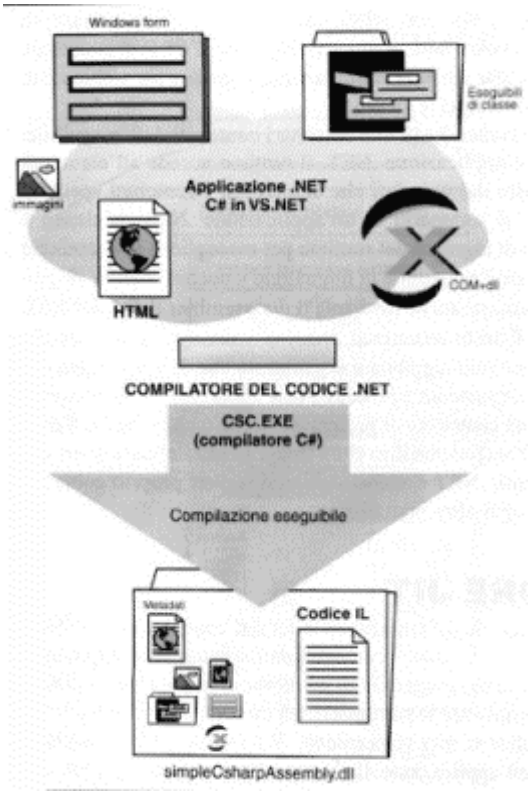


Figura 8: I. Generazione dell'IL

Le applicazioni Java sono compilate in una sintassi basata sugli interi che non è mai compilata in codice macchina nativo. Il runtime Java usa invece un processo che interpreta dinamicamente le compilazioni di codice a byte. Poiché l'interprete è software che emula una CPU, aumenta l'overhead richiesto per supportare questo tipo di struttura.[4]

.NET precompila le applicazioni in linguaggio macchina nativo con l'aiuto dei metadati di un eseguibile. I *metadati* sono XML che descrive la posizione e l'obiettivo di ogni oggetto, proprietà, argomento, delegato e metodo nell'eseguibile di un'applicazione .NET. Eliminano l'overhead associato all'interpretazione e alla gestione dell'ignoto. Il runtime usa i metadati per convalidare l'accuratezza e l'obiettivo di ogni funzione per evitare errori, ottimizzare la gestione della memoria e proteggere l'utente da attacchi malevoli.

La figura mostra il processo di compilazione di un'applicazione in un eseguibile. Pur essendo codificata principalmente in C#. si possono notare riferimenti a file codificati in altri linguaggi.

Il codice compilato di .NET risolve anche il cosiddetto problema del "DLL Hell". In passato, gli sviluppatori dovevano creare più versioni delle loro applicazioni Win32 per produrre eseguibili compatibili per ogni categoria delle principali configurazioni di sistema. Tutti gli sforzi fatti non serviranno a nulla quando un'altra applicazione dovrà aggiornare una DLL di sistema condivisa a una nuova versione. Gli sviluppatori .NET non dovranno più preoccuparsi di questi problemi, ma di altre questioni che non riguardano invece le applicazioni Win32. In altre parole, non sarà possibile eseguire le applicazioni .NET gestite alla stessa velocità di applicazioni Win32 non gestite. Microsoft afferma che in futuro il codice autodescrittivo di IL consentirà al runtime di compilare codice nativo così specifico per il sistema dell'utente finale da surclassare le applicazioni Win32. Naturalmente, ASP.NET è eseguito molto più velocemente del classico ASP.

Il vero punto di forza di ogni applicazione .NET compilata sono i metadati. È una soluzione che mantiene i vantaggi dell'ambiente di codice gestito usando una nuova tecnologia che supera le perdite di prestazioni a esso associate. L'idea è che il codice compilato non debba lasciare spazio a interpretazioni, ambiguità e supposizioni: solo i fatti, prego. [4] Ciò consente al CLR di compilare velocemente (sì, avviene una seconda compilazione) e gestire un ambiente di esecuzione il più possibile veloce e sicuro.

I metadati sono semplice codice XML che descrive i contenuti dell'eseguibile. Quando l'utente avvia un'applicazione .NET, il runtime accede all'elenco dell'eseguibile, che è composto dai metadati che descrivono i contenuti specifici del pacchetto. E un po' il sommario di un'applicazione .NET. L'elenco è anche il principale punto di ingresso del runtime per raccogliere ed esaminare i dati che devono compilare l'eseguibile in linguaggio macchina nativo. Si può accedere a queste informazioni anche mediante il disassembler ILDASM.EXE incluso nel framework. Questo strumento presenta i dettagli di alto livello dell'eseguibile in una gerarchia leggibile e organizzata che descrive i dettagli di ogni oggetto, metodo, argomento e proprietà nell'applicazione. Si potrebbe pensare che questa sia una cantonata di protezione presa da Microsoft. Senza addentrarsi nei dettagli, è sufficiente dire che l'integrità intellettuale di un'applicazione per il framework NET è vulnerabile agli hacker proprio come gli eseguibili compilati per ogni altra piattaforma. [4]

IL COMPILATORE JIT

Un altro componente essenziale del framework .NET è il compilatore JIT (JustIn-Time). Ha il compito di compilare i contenuti dell'eseguibile in linguaggio macchina. Non compila l'intero eseguibile in memoria in una volta sola. Deve esaminare l'elenco dell'eseguibile,contenete i metadati, per identificare le parti essenziali da compilare e caricare per assicurare che l'applicazione si apra velocemente. Il compilatore JIT caricherà solo le parti necessarie dell'applicazione. Il prodotto finale di quest'approccio è un ambiente di runtime con più spazio di memoria disponibile per elaborare più velocemente gli elementi usati. Secondo Microsoft, questo approccio dovrebbe garantire che le applicazioni .NET riusciranno a surclassare gli eseguibili Win32 più vecchi e non gestiti. [4]

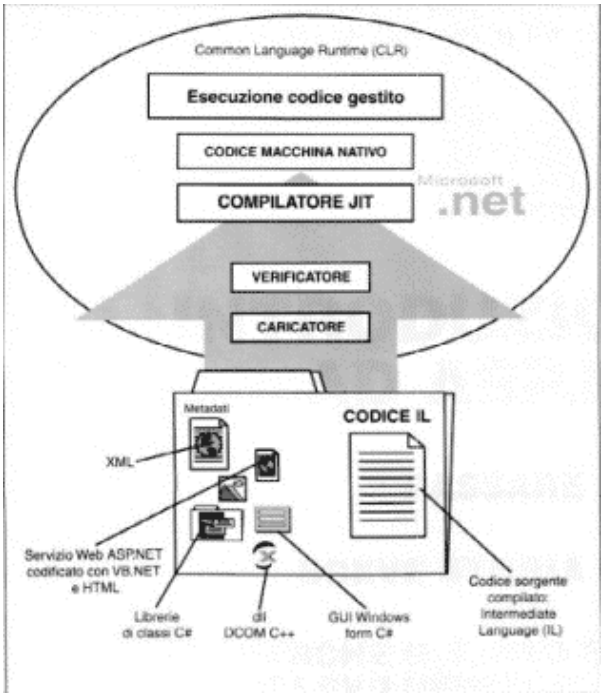


Figura 9: Generazione del Native Code

4. Visual Studio .NET

Introduzione

Visual Studio .NET è costituito da un unico ambiente di sviluppo integrato (IDE) per tutti i linguaggi, inclusi Microsoft Visual Basic® .NET, Microsoft Visual C++® .NET e Microsoft Visual C#™ .NET, gli sviluppatori possono avvalersi di una casella degli strumenti, un debugger e una finestra delle attività comuni, e questo consente di scegliere il linguaggio più adeguato al loro lavoro e alla loro esperienza.

Grazie alla tecnologia Intellisense, che rende disponibili il completamento automatico delle istruzioni e il controllo automatico degli errori di sintassi, Visual Studio .NET avvisa gli sviluppatori quando il codice non è corretto e permette di comprendere rapidamente le gerarchie delle classi e le API.

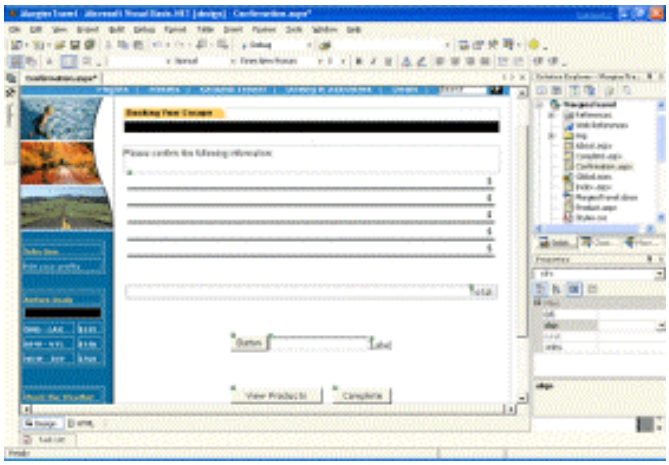


Figura 10: Shoot di Visual Studio.NET

Utilizzando Solution Explorer, gli sviluppatori sono in grado di riutilizzare facilmente il codice in più progetti e persino creare soluzioni in più linguaggi, in base alle necessità aziendali.

Grazie all'ambiente di sviluppo integrato (IDE) completamente estensibile, gli sviluppatori possono sfruttare le innovazioni e soluzioni offerte da una prolifera comunità di produttori, fra cui componenti aggiuntivi e controlli che favoriscono la personalizzazione ed estensione dell'ambiente.

Creazione di applicazioni basate su Windows

Visual Studio .NET offre agli sviluppatori Windows efficienti e facili Windows Forms. I Windows Forms sono compatibili con qualsiasi linguaggio basato su .NET, inclusi Microsoft Visual Basic® .NET e Microsoft Visual C#™ .NET. La possibilità di gestire graficamente l'ereditarietà semplifica notevolmente le attività di sviluppo delle applicazioni per Windows; infatti, è possibile centralizzare in form padre la logica e gli elementi dell'interfaccia utente comuni dell'intera soluzione.

Sviluppo per dispositivi portatili

Visual Studio .NET offre funzionalità Internet per dispositivi portatili consentendo la creazione un'unica interfaccia Web in grado di supportare una vasta gamma di disposiviti mobili, inclusi WML 1.1 per telefoni cellulari WAP, cHTML (HTML compatto) per telefoni i-mode e HTML per Pocket PC, palmari e cercapersone. I controlli mobili sul lato server generano in modo intelligente il rendering e l'impaginazione più appropriati per il dispositivo Web di destinazione.

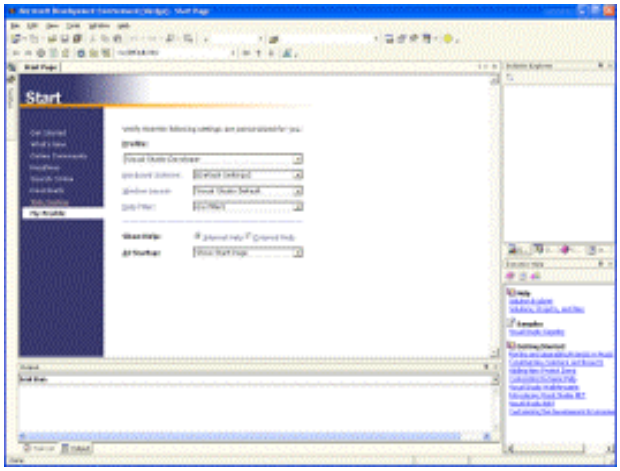


Figura 11: Shoot di Visual Studio.NET

4.1. I Linguaggi

La scelta di un linguaggio di programmazione dipende dalla propria esperienza e dallo scopo dell'applicazione che si vuole realizzare. Le piccole applicazioni di solito vengono create con un solo linguaggio, mentre applicazioni più complesse vengono realizzate con più di un linguaggio.
I linguaggi supportati del .NET sono quelli nell'elenco successivo:

- Visual Basic .NET
- Visual C# .NET
- Visual C++ .NET
- Visual J# .NET
- Managed Extensions for C++
- Transact-SQL
- Scripting Languages:
    - VBScript
    - Jscript
    - JScript .NET
- Extensible Markup Language (XML)
- Alternative Languages:
    - **COBOL** for Microsoft .NET.
    - **Perl** for Microsoft .NET.
    - **Eiffel** for Microsoft .NET.
    - **Python** for Microsoft .NET.
    - **Pascal** for Microsoft .NET.
    - **Mercury** for Microsoft .NET.
    - **Mondrian** for Microsoft .NET.
    - **Oberon** for Microsoft .NET.
    - **Salford FTN95 (Fortran)** for Microsoft .NET.
    - **SmallTalk** for Microsoft .NET.
    - **Standard ML** for Microsoft .NET.
    - **Dyalog APL** for Microsoft .NET.

Di quelli elencati ne descriveremo in modo approfondito solo alcuni, C#.Net,Vb.Net e VC++.Net, sono quelli che contengono le maggiori novità legate al Framework.

4.2. Visual C#

Negli ultimi vent'anni i linguaggi C e C++ sono stati senz'altro i più diffusi e utilizzati per lo sviluppo di prodotti software commerciali e aziendali. Entrambi i linguaggi forniscono ai programmatori una notevole quantità di controlli estremamente specifici. Purtroppo a causa della complessità e dei lunghi cicli di sviluppo associati a questi due linguaggi, molti programmatori che utilizzano C e C++ sono da tempo alla ricerca di un linguaggio che offra un migliore equilibrio fra efficacia e produttività.

Esistono oggi linguaggi che aumentano la produttività sacrificando però quella flessibilità tanto apprezzata da chi programma con C e C++. Soluzioni di questo tipo limitano eccessivamente lo sviluppatore (ad esempio sono privi di meccanismi per il controllo del codice a basso livello) e offrono funzionalità difficili da generalizzare. Non sono in grado di interagire facilmente con i sistemi già esistenti e non sempre si adattano alle tecniche attuali di programmazione per il Web.

C# ("C SHARP")

La soluzione proposta da Microsoft per rispondere a queste esigenze è il linguaggio C#. C# è un linguaggio orientato agli oggetti che consente ai programmatori di creare rapidamente una vasta gamma di applicazioni per la nuova piattaforma Microsoft .NET, che a sua volta offre strumenti e servizi in grado di sfruttare appieno le tecnologie sia di elaborazione che di comunicazione.

Grazie alle potenzialità della progettazione orientata agli oggetti, C# permette la creazione di architetture per una vasta gamma di componenti, dagli oggetti business di alto livello ad applicazioni a livello di sistema. Utilizzando i semplici costrutti del linguaggio C#, è possibile convertire questi componenti in XML Web service, che possono essere richiamati su Internet da qualunque linguaggio in esecuzione su qualsiasi sistema operativo.

Inoltre, C# è progettato per non sacrificare la potenza e il controllo che hanno rappresentato le caratteristiche distintive di C e C++. Grazie a questa eredità, C# presenta un livello di somiglianza molto elevato con C e C++.

Produttività e sicurezza

La nuova economia Web chiede agli sviluppatori di ridurre i tempi del ciclo di sviluppo e di produrre un numero superiore di versioni incrementali dei programmi, piuttosto che un'unica versione monumentale.
C# è stato concepito tenendo presenti queste considerazioni. Il linguaggio è progettato per aiutare gli sviluppatori a trarre il massimo da un numero inferiore di righe di codice, con minori possibilità d'errore.

- Integrazione degli standard di programmazione Web emergenti

  Il nuovo modello di sviluppo delle applicazioni evidenzia che un numero sempre più vasto di applicazioni richiede l'utilizzo di standard Web emergenti, quali i linguaggi HTML (Hypertext Markup Language), XML (Extensible Markup Language) e SOAP. Gli strumenti di sviluppo esistenti sono stati realizzati prima dell'espansione di Internet, e comunque quando il Web era ancora ai primordi. Per tale ragione non sempre si rivelano adatti a operare con le nuove tecnologie Web.
  I programmatori che utilizzano C# possono contare su una struttura completa per la creazione di applicazioni nella piattaforma Microsoft .NET. C# incorpora il supporto per trasformare ogni componente in un XML Web service che possa essere richiamato su Internet da qualsiasi applicazione in esecuzione in qualsiasi piattaforma.

- Eliminazione di costosi errori di programmazione

  Anche i programmatori in C++ più esperti possono incorrere in semplici dimenticanze, ad esempio possono dimenticare di inizializzare una variabile. Spesso questi errori così banali hanno come conseguenza problemi imprevedibili, che rischiano di passare inosservati anche per lunghi periodi di tempo. Tuttavia, quando un programma è già in uso in un ambiente di produzione, la correzione degli errori più semplici può rivelarsi molto onerosa.
  Il design moderno di C# elimina la maggior parte degli errori di programmazione più comuni in C++. Ad esempio:
  Le attività di garbage collection sollevano il programmatore dal compito di dover gestire la memoria manualmente.
  In C# le variabili vengono inizializzate automaticamente dall'ambiente.
  Le variabili sono indipendenti dai tipi.

- Riduzione dei costi di sviluppo, grazie al supporto integrato per la gestione delle versioni

  L'aggiornamento dei componenti software è un'attività soggetta a errori. Le revisioni apportate al codice possono inavvertitamente modificare la semantica di un programma esistente. Per risolvere questo problema, C# integra direttamente nel linguaggio il supporto per la gestione delle versioni. Ad esempio, l'override dei metodi deve essere esplicito e non può avvenire inavvertitamente come in C++ o in Java. In questo modo si prevengono gli errori di codifica e si conserva la flessibilità nella gestione delle versioni. Una funzionalità correlata è il supporto nativo per le interfacce e l'ereditarietà delle interfacce.
  Insieme, queste funzionalità rendono più affidabile il processo di sviluppo delle versioni successive di un progetto e pertanto riducono i costi globali di sviluppo delle versioni.

Potenza, espressività e flessibilità

- Migliore corrispondenza tra processo di business e implementazione

  Considerata l'importanza che la pianificazione aziendale riveste, è fondamentale che vi sia una stretta correlazione tra il processo di business in astratto e l'effettiva implementazione del prodotto software. Nella maggior parte dei linguaggi non è disponibile una modalità semplice per collegare la logica business al codice. Ad

esempio, gli sviluppatori inseriscono commenti nel codice per identificare le classi che costituiscono uno specifico oggetto business astratto.
Il linguaggio C# consente di applicare a qualsiasi oggetto metadati con definizione del tipo ed estensibili. L'architetto di un progetto può definire attributi specifici di un dominio, quindi applicarli a qualunque classe di elementi del linguaggio, interfacce e così via. Lo sviluppatore può quindi esaminare mediante programmazione gli attributi di ogni elemento. In questo modo risulta più semplice, ad esempio, scrivere uno strumento automatizzato che garantisca che ogni classe o interfaccia venga correttamente identificata come parte di uno specifico oggetto business astratto, oppure creare report basati sugli attributi di un oggetto specifici del dominio. La forte correlazione tra i metadati personalizzati e il codice del programma aiuta a rendere più forte la connessione tra il funzionamento ipotizzato del programma e l'effettiva implementazione.

- Estensione dell'interoperabilità

  L'ambiente gestito e indipendente dai tipi si adatta bene alla maggior parte delle applicazioni di livello enterprise. L'esperienza del mondo reale insegna però che per alcune applicazioni continua a essere necessario il codice nativo, per ragioni di prestazioni o per garantire l'interoperabiltà con le interfacce API già esistenti. Scenari di questo tipo possono imporre agli sviluppatori l'utilizzo del linguaggio C++ anche se preferirebbero impiegare un ambiente di sviluppo più produttivo.

  C# risolve questi problemi perché:

  - Include il supporto nativo per le API COM (Component Object Model) e basate su Windows®.
  - Consente un utilizzo limitato dei puntatori nativi.

  Nel linguaggio C#, ogni oggetto è automaticamente un oggetto COM. Pertanto, gli sviluppatori non devono più implementare esplicitamente le interfacce IUnknown e altre interfacce COM. Queste funzionalità sono infatti integrate. Allo stesso modo, i programmi sviluppati con C# possono utilizzare a livello nativo gli oggetti COM esistenti, indipendentemente dal linguaggio con cui sono stati creati.
  C# include inoltre una funzionalità speciale, che consente a un programma di richiamare qualsiasi API nativa. All'interno di un blocco di codice contrassegnato in modo speciale, gli sviluppatori possono utilizzare i puntatori e funzionalità tradizionali di C e C++, quali la gestione manuale della memoria e l'aritmetica dei puntatori. Ciò consente ai programmatori che si avvalgono di C# di riutilizzare il codice scritto mediante C e C++.

  In entrambi i casi, ovvero con il supporto COM e con l'accesso alle API native, l'obiettivo è quello di rendere disponibili allo sviluppatore la potenza e il controllo a loro necessari, senza dover uscire dall'ambiente C#.

4.3. Visual Basic .NET

Introduzione

Per creare con rapidità applicazioni Web di livello enterprise, gli sviluppatori devono potersi avvalere di una logica business che sia scalabile, affidabile e riutilizzabile. Nel corso degli anni passati, la programmazione orientata agli oggetti è stata senz'altro la metodologia principale impiegata nella creazione di sistemi che potessero soddisfare questi requisiti. L'utilizzo di linguaggi di programmazione orientati agli oggetti rende anche i sistemi su larga scala più facili da comprendere, offre una maggiore facilità di esecuzione del debug e aumenta la velocità di aggiornamento.

Visual Basic .NET offre un linguaggio di programmazione orientato agli oggetti, con nuove funzionalità quali l'ereditarietà dell'implementazione, l'overload e i costruttori con parametri. Inoltre, gli sviluppatori hanno l'opportunità di creare codice altamente scalabile con il modello di threading Free esplicito e codice ampiamente gestibile con l'aggiunta di costrutti di linguaggio modernizzati, ad esempio la gestione delle eccezioni strutturata.

Breve storia delle innovazioni del linguaggio

Il linguaggio Visual Basic ha alle spalle una lunga storia di aggiornamenti che corrispondono grosso modo alle principali modifiche apportate alla piattaforma Windows®. Ad esempio, le significative modifiche apportate a QuickBasic® per consentire al prodotto di supportare lo sviluppo della GUI di Windows 3.0 sono poi confluite nella prima versione di Visual Basic. In Visual Basic 4.0, il passaggio alla programmazione basata su COM ha portato ai costrutti di linguaggio su cui si basa la creazione delle DLL. In Visual Basic 5.0 il linguaggio si è evoluto per supportare la creazione dei controlli COM.

A ogni successiva revisione, la popolarità di Visual Basic è aumentata. Le potenzialità che le nuove funzionalità del linguaggio orientato agli oggetti rendono disponibili agli sviluppatori di applicazioni Web di livello enterprise proseguono in questa direzione.

Programmazione orientata agli oggetti

La tradizionale programmazione strutturata, nella quale i dati sono archiviati separatamente dal codice procedurale, presenta diversi punti deboli. Qualsiasi porzione di codice scritta come codice strutturato non è modulare. Poiché gli elementi di informazione sono accessibili da qualsiasi codice, è possibile che i dati vengano modificati all'insaputa dello sviluppatore. Ciò può provocare errori in fase di esecuzione, estremamente difficili da individuare tempestivamente e correggere. Inoltre, le attività di manutenzione possono diventare onerose. Cercare di capire quale sia l'impatto globale conseguente la modifica di una singola riga di codice mediante la programmazione strutturata può rivelarsi un'operazione estremamente difficile. Infine, confidare nel programmatore per la gestione sia del codice sia delle informazioni può comportare una bassa percentuale di riutilizzo.

La programmazione orientata agli oggetti risolve questi problemi. Crea infatti un package di dati e metodi che agiscono su tali dati, ottenendo una singola unità chiamata oggetto. È possibile nascondere i dati dell'oggetto per impedire modifiche non autorizzate. Inoltre, l'oggetto presenta un insieme di metodi pubblici che agiscono su questi dati. Questo concetto viene definito incapsulamento. Poiché i dettagli dell'implementazione sono separati dall'interfaccia, è possibile modificare la logica di programmazione sottostante anche in momenti successivi, senza timore di danneggiare il codice che richiama l'oggetto.

La programmazione orientata agli oggetti consente inoltre agli sviluppatori di riutilizzare codice e dati insieme, mediante l'ereditarietà. Grazie all'ereditarietà da oggetti predefiniti, gli sviluppatori sono in grado di realizzare in maniera rapida applicazioni complesse. Va sottolineato, infatti, che la scrittura di nuovo codice implica sempre la possibilità di commettere errori, mentre il riutilizzo di codici ormai testati riduce al minimo questa eventualità.

Nuove funzionalità della programmazione orientata agli oggetti:

- **Ereditarietà**

  La funzionalità più richiesta per Visual Basic è il supporto per l'ereditarietà dell'implementazione. Nell'era di Internet, lo sviluppo richiede capacità di assemblaggio rapido e possibilità di continuo riutilizzo. Visual Basic include la funzionalità per la completa ereditarietà dell'implementazione, inclusa l'ereditarietà visuale dei form.

  Gli sviluppatori possono utilizzare la nuova parola chiave **Inherits** per creare una derivazione da una classe esistente.

  L'istruzione Inherits supporta tutte le proprietà solitamente associate all'ereditarietà. Le istanze della classe derivata supportano tutti i metodi e le interfacce supportate dalla classe base. E, naturalmente, la classe derivata è in grado di estendere l'insieme di metodi e di interfacce supportate dalla classe base.

  La classe derivata può ignorare i metodi definiti nella classe base utilizzando la parola chiave ***Overrides***. Al fine di ridurre gli errori di programmazione, Visual Basic impedisce che una funzione venga involontariamente ignorata. Pertanto nelle classi derivate è possibile ignorare solo le funzioni contrassegnate come "Overridable".

- **Overload**

  Visual Basic consente l'overload delle funzioni, offrendo agli sviluppatori la possibilità di creare diverse versioni di Sub o di Function con lo stesso nome, ma con diversi tipi di argomento.

  Senza l'overload, è necessario creare nomi distinti per ogni routine oppure utilizzare un parametro Variant. L'overload rappresenta una modalità più esplicita ed efficiente per gestire più tipi di dati.

- **Costruttori con parametri**

  I costruttori con parametri, o semplicemente costruttori, consentono di creare una nuova istanza di una classe e allo stesso tempo di passare argomenti alla nuova istanza. I costruttori sono essenziali nella programmazione orientata agli oggetti, poiché consentono di passare i parametri del creatore dell'istanza al codice di costruzione definito dall'utente.

Ulteriori funzionalità del linguaggio:

- **Modello di threading Free**

  Quando gli sviluppatori creano applicazioni mediante Visual Basic, scrivono codice di tipo sincrono. Ciò significa che ogni riga di codice deve essere eseguita prima della riga successiva. Poiché la scalabilità rappresenta l'elemento chiave dello sviluppo delle applicazioni Web, gli sviluppatori hanno bisogno di strumenti che consentano un tipo di elaborazione simultanea.

  Con l'inclusione del modello di threading Free, gli sviluppatori possono generare un thread in grado di eseguire alcune attività a esecuzione prolungata, eseguire una query complessa o elaborare un calcolo suddiviso in più parti, mentre il resto dell'applicazione prosegue autonomamente, realizzando così l'elaborazione asincrona.

```
Sub CreateMyThread()
    Dim b As BackGroundWork
    Dim t As Thread
    Set b = New BackGroundWork()
    Set t = New Thread(New ThreadStart(AddressOf b.Doit))
    t.Start
End Sub
Class BackGroundWork
    Sub DoIt()
    .
    End Sub
End Class
```

- **Gestione delle eccezioni strutturata**

  Lo sviluppo di applicazioni di livello enterprise richiede la costruzione di componenti riutilizzabili e sui quali sia semplice eseguire attività di manutenzione. Nelle versioni precedenti di Visual Basic, un aspetto complesso del linguaggio Basic era rappresentato dal supporto per la gestione degli errori. La gestione degli errori mediante l'istruzione esistente **On Error GoTo** rallenta a volte le attività di sviluppo e la manutenzione di applicazioni di grandi dimensioni. La gestione di vari errori mediante la combinazione di istruzioni **Resume** e **Next** genera codice illeggibile e comporta errori frequenti quando i percorsi di esecuzione non vengono considerati con attenzione.

  Con l'istruzione **Try...Catch...Finally**, questi problemi vengono risolti e gli sviluppatori possono nidificare la gestione delle eccezioni. Esiste inoltre una

struttura di controllo per la scrittura del codice di pulitura, eseguibile sia in condizioni normali che in condizioni di eccezione.

```
Sub SEH()
   Try
            Open "TESTFILE" For Output As #1
            Write #1, CustomerInformation
   Catch
            Kill "TESTFILE"
   Finally
            Close #1
   End try
End Sub
```

- **Controllo accurato dei tipi**

  L'attuale linguaggio Visual Basic è poco rigoroso rispetto alle coercizioni di tipo implicito che genera. Per l'assegnazione e il passaggio dei parametri con modalità diverse da quelle che hanno luogo mediante il riferimento, il compilatore Visual Basic consente praticamente a qualsiasi tipo di dati di essere convertito in un qualsiasi altro tipo, mediante la coercizione in fase di esecuzione. L'operazione di coercizione in fase di esecuzione fallisce se non è possibile convertire il valore senza perdere parte dei dati. Mediante l'aggiunta di una nuova opzione di compilazione, Visual Basic può generare errori in fase di compilazione per ogni conversione che potrebbe provocare un errore in fase di esecuzione.

- **Membri condivisi**

  Si intendono con membri condivisi quei dati e quelle funzioni che fanno parte di classi e che vengono condivisi da tutte le istanze della classe. La condivisione di una singola istanza di un membro dati o di una funzione da parte di tutte le istanze di una classe è obbligatoria in un'applicazione Visual Basic con ereditarietà. Un membro dati esiste indipendentemente da qualsiasi istanza particolare della classe. Un metodo condiviso è un metodo al quale, a differenza dei metodi normali, non viene implicitamente passata un'istanza della classe. Per questa ragione in un metodo condiviso non è consentito alcun riferimento non qualificato a membri dati non condivisi. È possibile accedere ai membri condivisi pubblici in modalità remota. Tali membri possono inoltre essere associati tardivamente a partire da un'istanza della classe.

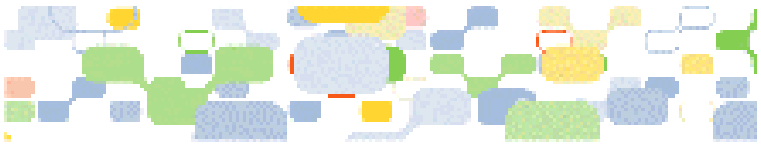
- **Inizializzatori**

  Visual Basic .NET supporta l'inizializzazione delle variabili nella riga in cui vengono dichiarate. È possibile utilizzare gli inizializzatori ovunque, anche all'interno di una struttura di controllo. La semantica della dichiarazione a livello di routine, che include un inizializzatore, è uguale a un'istruzione per la dichiarazione immediatamente seguita da una dichiarazione per l'assegnazione. In altre parole, l'istruzione seguente:

  ```
  Dim X As Integer = 1
  ```

  equivale a queste istruzioni:

  ```
  Dim X As Integer
  X = 1
  ```



4.4. Visual C++ .NET

Visual C++ .NET contiene numerose funzionalità che permettono di creare ed utilizzare applicazioni Web e servizi Web XML. L'obiettivo di Visual C++ .NET è lo sviluppo di programmi più compatti, veloci ed efficienti.
Tipicamente chi scelglie C++ come linguaggio di sviluppo lo fa in quanto mette a disposizione maggiore potenza e flessibilità rispetto ad altri ambienti di sviluppo. Visual C++ .NET si propone per soddisfare questa esigenza.

Potenza e flessibilità

Visual C++ rappresenta un caso unico tra i linguaggi .NET in quanto supporta sia il modello di codice gestito fornito dal .NET Framework sia quello nativo di Windows. In questo modo, Visual C++ .NET preserva e migliora gli investimenti esistenti e rappresenta la miglior scelta possibile per gli sviluppatori.
Visual C++ .NET include un'estensione molto potente all'Active Template Library (ATL), denominata ATL Server, che aiuta gli sviluppatori a creare applicazioni e servizi Web compatti ed a elevate prestazioni. L'ATL Server incapsula il meglio dello sviluppo di applicazioni Web ad alte prestazioni in una classe ATL semplice e flessibile da riutilizzare.
Con le estensioni gestite per Visual C++, gli sviluppatori possono scrivere programmi C++ per il .NET Framework. Le estensioni gestite possono agire anche da ponte tra il codice Visual C++ e qualsiasi linguaggio gestito, tra cui Visual C# .NET e Visual Basic .NET.
Visual C++ .NET presenta, inoltre, numerose funzionalità per migliorare le esistenti applicazioni Windows e semplificarne la gestione. Gli aggiornamenti a Microsoft Foundation Class (MFC) consentono di accedere alle ultime novità introdotte nella piattaforma Windows. Visual C++ continua a mantenere la propria leadership nella qualità e nelle prestazioni del codice generato. Oltre a ciò, le nuove ottimizzazioni al compilatore offrono un codice più compatto e veloce rispetto alle versioni precedenti.

ATL Server: servizi Web XML in Visual C++ .NET

Quando si sceglie di scrivere parti delle proprie applicazioni Web in C++, lo si fa generalmente per questioni di prestazioni o controllo. Le soluzioni Web scritte in ASP.NET hanno buoni livelli di prestazioni e scalabilità ed in generale offrono più potenza di quanto effettivamente necessaria alla maggior parte delle applicazioni Web. Vi sono situazioni, tuttavia, in cui si ha bisogno di più prestazioni di quante possa offrirne un'implementazione C++, o in cui vogliono il controllo totale su ogni bit di codice di una componente critica. ATL Server è progettato proprio per queste situazioni.
ATL Server incapsula le tecniche migliori dello sviluppo di applicazioni Web ad alte prestazioni in un insieme di classi ATL semplici, estensibili e riutilizzabili. ATL Server include funzionalità che rendono le applicazioni Web ed i servizi Web XML scalabili su macchine che appartengono a farm di server Web.

Supporto per le applicazioni server

ATL Server presenta numerose altre funzionalità progettate specificamente per le applicazioni server. È presente un compatto ed efficiente client HTTP creato per applicazioni server-to-server. Questo client consente agli sviluppatori di invocare altri servizi Web XML dal proprio codice lato server senza il sovraccarico di implementazioni HTTP più pesanti e lato client.

Le estensioni gestite per Visual C++

Le estensioni gestite per Visual C++ costituiscono un insieme di estensioni del linguaggio che aiuta a riscrivere applicazioni Visual C++ per il .NET Framework.

Il codice C++ gestito e non può essere mescolato liberamente all'interno della stessa applicazione. Le nuove applicazioni scritte con le estensioni gestite possono sfruttare il meglio di entrambi i mondi.

Le estensioni gestite sono un insieme di nuovi attributi e parole chiave per il sistema di sviluppo Visual C++. Esse consentono di decidere quali classi e quali funzioni compilare come codice gestito o non gestito. Queste parti possono quindi interoperare facilmente tra loro con le librerie esterne.

Scrivere codice robusto

Visual C++ .NET include una nuova funzionalità per individuare se si scrive inavvertitamente i dati all'esterno del buffer – un errore conosciuto come "buffer overrun". Questi sono alcuni degli errori più comuni che possono provocare il crash del codice ed anche una delle modalità più utilizzate per attaccare i server Web su Internet.

Visual C++ .NET può essere configurato per verificare l'integrità dei dati scritti in un buffer allocato dinamicamente. Gli sviluppatori possono scrivere un gestore personalizzato che verrà invocato se si verifica un errore di buffer overrun. Ciò consente di effettuare correttamente lo shut down dell'applicazione piuttosto che rischiare comportamenti inaspettati.

5. Applicazioni Web

Un'applicazione Web moderna può essere realizzata con diverse tecnologie ad esempio: JSP, CGI, PHP, ASP. È appunto l'evoluzione di quest'ultima che è stata impiegata nel progetto realizzato. L'evoluzione si chiama ASP.NET, vediamo ora le sue principali caratteristiche partendo dall'evoluzione del modo di concepire l'applicazione web, il modo di progettarla.

5.1. Progettazione a Tre livelli

Un grosso problema che si è riscontrato analizzando le attuali applicazioni Web è che esse "soffrono", quando sono sottoposte ad un carico di richieste molto elevato, diventano così lente e costringono chi le utilizza a lunghi tempi d'attesa.

Il problema attuale dello sviluppo d'applicazioni Web è di dover gestire transazioni di molteplici utenti, così da poter ridurre i costi e i tempi di risposta, permettendo un accesso semplice ai diversi tipi di funzionalità offerti dall'applicazione. Generalmente le applicazioni che permettono di fornire servizi di questo tipo devono integrare degli EIS (Enterprise Information Systems), già esistenti, con nuove applicazioni che permettono di fornire diverse funzionalità che devono poter essere messe a disposizione di un numero elevato d'utenti.
I servizi messi a disposizione dall'applicazione devono per prima cosa essere:

- *Versatili:* per poter soddisfare le diverse esigenze degli utenti che fanno uso di questi servizi
- *Sicure:* per poter proteggere la privacy degli utenti e l'integrità dei dati trasmessi;
- *Riutilizzabili e scalabili:* per poter realizzare applicazioni che possano essere recuperate e riutilizzate da diversi utenti.

Questo modello per la realizzazione di applicazioni fornisce i benefici legati Write Once, Run Anywhere che equivale a dire che le applicazioni multi-livello sono portabili e scalabili.

Questi servizi vengono forniti per mezzo di applicazioni distribuite costituite da diversi livelli, caratterizzati all'estremità superiore da un numero elevato e variegato di utenti, alla estremità inferiore da diversi tipi di risorse-dati e tutto quello che riguarda il lavoro di sviluppo della applicazione.

Il livello intermedio (definito Middle-Tier) contiene tutti quei servizi che sono in grado di integrare gli EIS preesistenti con nuove applicazioni e risorse-dati.

L'introduzione di questo livello intermedio protegge l'utente dalla complessità connessa con lo sviluppo dell'applicazione, permettendogli un accesso facile e rapido al servizio.

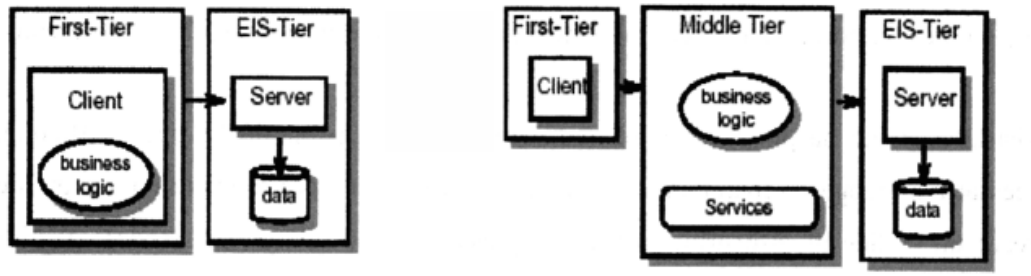


Figura 12: Passaggio da una struttura a due livelli ad una a tre livelli

Solitamente il Middle-Tier risiede su un software dedicato ed ha accesso a tutte le risorse messe a disposizione del sistema, in modo tale da poter usufruire di tutto quello che è necessario per un corretto funzionamento del programma.

Al contrario tutti i dati critici e le funzioni che li gestiscono risiedono nell'EIS-Tier e rappresentano il nucleo più interno del sistema.

Passaggio da una piattaforma Two-Tier ad una Multi-Tier:

Inizialmente dovendo gestire delle applicazioni client-server questo tipo di strutturazione permetteva funzionalità e scalabilità precedentemente non disponibili.

Questa struttura però ha diversi limiti che sono rappresentati da:

- dover inviare a ciascun utente i servizi EIS.
- installare e mantenere efficiente, mediante aggiornamenti, la parte di applicazione logica sulle diverse macchine utilizzate da diversi utilizzatori.

Per risolvere questo problema si è pensato di passare ad una struttura a più livelli, che oltre ad ovviare agli inconvenienti precedenti i permette di incrementare l'accessibilità che viene richiesta dai vari elementi che costituiscono l'applicazione.

L'attenzione si è quindi spostata sullo sviluppo del software residente nel Middle-Tier.

Questa evoluzione non ha portato solo benefici ma anche alcune difficoltà. Ora in fatti è necessario sviluppare separatamente le funzioni di sviluppo, per soddisfare i bisogni degli utilizzatori e creare un codice di infrastruttura più complesso in modo da poter accedere ai diversi database e alle diverse risorse di sistema.

.NET definisce un architettura che permette di fornire servizi che vengono realizzati per mezzo di applicazioni Multi-Tier in grado di sopperire a queste difficoltà e di raggiungere la scalabilità e la maneggevolezza richiesta.

Il modello implementativo della .NET richiede di partizionare le applicazioni strutturate su più livelli in due parti:

- ***Business logic,*** parte dell'applicazione che riguarda tutte le funzionalità richieste dall'utente

- ***Presentation logic,*** rappresenta il modo attraverso il quale l'utente può richiedere i diversi servizi e può osservare i risultati ottenuti.

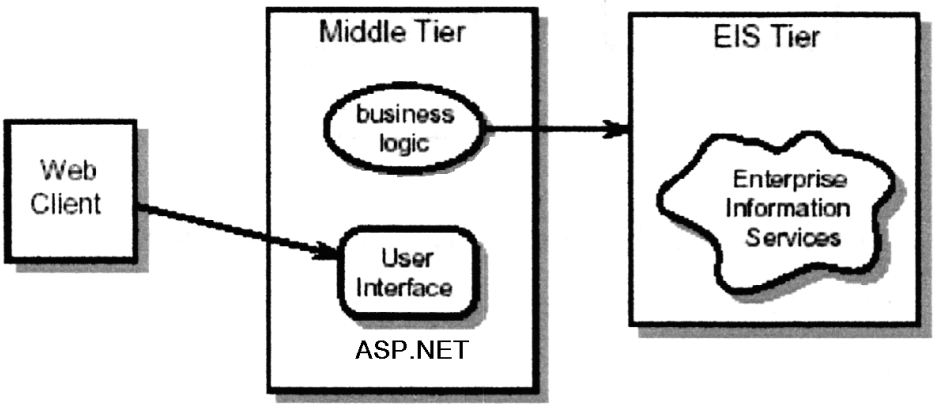


Figura 13: Applicazione a tre livelli

Le diverse funzionalità offerte all'utente sono rese disponibili, per mezzo dell'utilizzo della tecnologia che si basa su ASP.NET, secondo servizi in forma Internet Style facilmente accessibili a qualsiasi utente.

La tecnologia delle ASP.NET (Active Server Pages .NET) permette di rendere molto semplice, per uno sviluppatore di interfacce utente, realizzare pagine dinamicamente generate per qualunque tipo di browser.
Il codice non è misto a tag HTML come succedeva in ASP, ma separato, ciò aiuta molto la realizzazione e la comprensione della pagina da parte di sviluppatori diversi.

I linguaggi .NET consentono ai programmatori la possibilità di realizzare presentazioni dinamiche, essendo veri e propri linguaggi sono compilabili e ciò aumenta le prestazioni.

5.2. ASP.NET

Breve storia di ASP

Quando Active Server Pages è stato rilasciato per la prima volta nel novembre 1996, ha introdotto un metodo semplice per creare pagine Web dinamiche. Anche se Common Gateway Interface (CGI) e Perl erano molto usati all'epoca, ASP si è subito affermato per quattro motivi: facilità di accesso ai dati, semplice struttura di pagine, interoperabilità con COM (Component Object Model) e facilità di apprendimento per gli sviluppatori esperti di Visual Basic. Innanzi tutto, se ASP non fosse stato rilasciato con ActiveX Data Objects (ADO), probabilmente non si sarebbe affermato così facilmente. ADO ha sostituito Remote Data Objects (RDO) e in seguito Data Access Objects (DAO) come metodo di accesso a database preferito da Microsoft e ha fornito un semplice modello a oggetti.

Secondo, con la release di ASP 1.0 è nata anche la prima suite di Microsoft Visual Studio, che includeva Visual lnterDev 1.0. Questo strumento offriva molte funzioni e utilizzava Microsoft FrontPage Server Extensions (FPSE) per semplificare la manutenzione di siti Web senza la necessità di un client FTP separato. Intellisense per COM, oggetti ASP incorporati e controllo di accesso ai dati lo hanno trasformato in un editor importante nonostante gli svantaggi. come il suo editor GUI (Graphical User Interface) che si basa su una versione precedente di FrontPage.

Probabilmente, se non fosse stato per ASP, il mercato dei componenti di terze parti non sarebbe l'industria da miliardi di dollari annui che è oggi. Di certo non è stato l'unico fattore, poiché i controlli di terze parti per Visual Basic precedono la rivoluzione COM, ma ASP e COM hanno fatto il successo di molte imprese che rivendevano pacchetti di componenti da usare su server Web. Prima di ASP 1.0. l'abilità di acquistare componenti prefabbricati e installarli in un sito Web era disponibile solo per i programmatori esperti. Portando questa capacità alle masse, ASP ha dato vita a un nuovo mercato di produttori di componenti. che continuano a offrire strumenti prefabbricati potenti facilmente integrabili nelle applicazioni ASP.

La quarta caratteristica di ASP che ne ha permesso il successo era l'utilizzo di VBScript come linguaggio predefinito, consentendo ai programmatori già esperti di Visual Basic di iniziare subito a usare ASP con il minimo studio. Per i programmatori esperti di JavaScript o C, è stato offerto JScript. Infatti, Microsoft ha fornito la possibilità di usare anche altri linguaggi di terze parti e, dopo poco tempo, è stato possibile scrivere codice ASP in Perl.

Ciononostante, ASP 1.0 presentava limiti significativi. Il limite principale per chi lavorava con componenti COM era che il server Web doveva essere riavviato a ogni aggiornamento di una DLL (Dynamic Link Library). Gli oggetti COM sono memorizzati infatti come file .DLL. Vennero affrontati diversi problemi di prestazioni e di sicurezza, ma il principale miglioramento da ASP 1.0 a 2.0 è stato Microsoft Transaction Server (MTS). ASP 2.0 veniva fornito come parte di Internet Information Server (IIS) 4.0, mentre MTS 1.0 era fornito come parte del gratuito Windows NT 4 Option Pack.

Con IIS 4.0, Microsoft ha introdotto Microsoft Management Console (MMC), che veniva usato per amministrare IIS 4.0 e MTS. Con MTS, la vita di chi sviluppava o usava componenti COM divenne più semplice. Gestiva l'installazione e la disinstallazione dei componenti, rimuoveva la necessità di riavviare il servizio Web e spesso anche il server e persino buona parte della gestione delle transazioni da parte dello sviluppatore. Infine, agiva da intermediario di oggetti, inserendo nella cache le istanze di oggetti e restituendole a ogni richiesta. Questa tecnica ha condotto al concetto di componenti "privi di stato", caratteristica essenziale al riutilizzo degli oggetti. Inoltre, le nuove versioni di ADO hanno migliorato ulteriormente l'abilità degli sviluppatori di lavorare con dati remoti, usando nuove tecniche come i flussi XML.

Nel febbraio 2000, Microsoft ha rilasciato IIS 5.0 con Windows 2000. Con IIS 5.0, ASP era alla versione 3.0 e MTS venne sostituito dai servizi COM+. COM+ combinava la funzionalità di MTS con i servizi di accodamento dei messaggi, fattore che aggiungeva nuove funzionalità ad ASP, tra cui alcuni metodi e proprietà intrinseche degli oggetti. Le differenze principali tra la programmazione in ASP 2.0 e ASP 3.0 si basano sui servizi supportati, come COM+, più che sul linguaggio stesso. Chiunque fosse stato in grado di scrivere VBscript in ASP 1.0 con Visual lnterDev 1.0 si sarebbe trovato a suo agio nell'usare IIS 4.0 o IIS 5.0. Non è questo il caso di ASP.NET.

Mark Anders e Scott Guthrie di Microsoft hanno iniziato a sviluppare ciò che sarebbe diventato ASP.NET nel gennaio 1998. A quell'epoca, ASP aveva solo un anno, ma alcuni dei suoi limiti erano già evidenti. Per esempio, la restrizione a linguaggi di scripting e l'assenza di un modello di componenti impedivano lo sviluppo di validi strumenti per ASP. Lo sparpagliamento del codice con output HyperText Markup Language (HTML) causava spesso problemi quando designer e sviluppatori lavoravano sullo stesso progetto. ASP.NET è stato progettato dall'inizio per risolvere i limiti di ASP.
Mark e Scott hanno scelto di creare ASP.NET - allora ASP+ - sul Next Generation Web Services (NGWS) Runtime a quell'epoca in fase di sviluppo.

NGWS, che sarebbe diventato .NET, offriva un ricco insieme di librerie di programmazione e avrebbe presto incluso il nuovo linguaggio C#, in cui è scritto ASP.NET. ASP.NET è stato in fase di sviluppo per più di tre anni e Microsoft ha deciso di basare questo prodotto sulle priorità seguenti:

- **Struttura fattorizzata.** ASP.NET è scritto come insieme di componenti modulari che possono essere sostituiti o estesi se necessario.

- **Scalabilità.** È stato creato un modello altamente scalabile, nel rispetto del mantenimento dello stato.

- **Disponibilità.** ASP.NET è stato strutturato per rilevare crash, memory leak, deadlock ed effettuare il ripristino dopo questi eventi.

- **Prestazioni.** ASP.NET trae vantaggio dai linguaggi compilati e dall'early binding per migliorare le prestazioni e offre supporto di caching estensivo.

- **Integrazione di strumenti.** L'obbiettivo di Microsoft è rendere la creazione di un sito Web semplice come la creazione di un form in Visual Basic. Visual Studio .NET è il primo strumento a offrire questa funzionalità, ma presto anche altri produttori

offriranno soluzioni simili.[4]

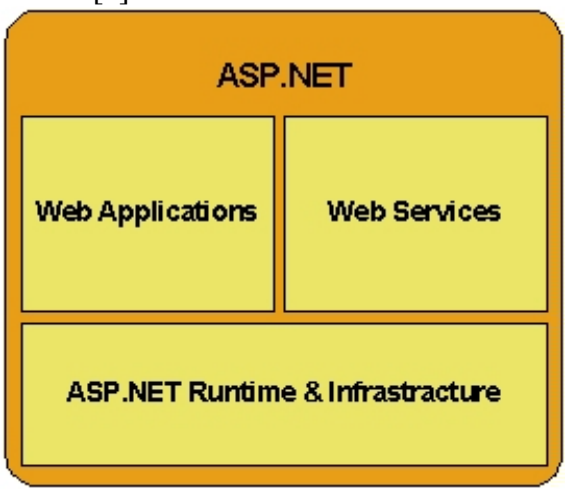


Figura 14: Livelli del ASP.NET

ASP.NET

ASP.NET fornisce un modello applicativo sotto forma di controlli ed infrastruttura che semplifica la creazione di applicazioni Web.
Una caratteristica fondamentale di questi controlli è che possono essere scritti per adattarsi a funzionalità lato client; i controlli delle form Web possono determinare le caratteristiche del client che richiede una pagina e garantire un'adeguata esperienza all'utente - WML per i telefonini, HTML 3.2 per i browser a di vecchia generazione ed il Dynamic HTML per Internet Explorer 5.5.

ASP.NET mette, inoltre, a disposizione funzionalità quali la gestione dello stato in architetture cluster e la possibilità di riciclare i processi, cosa che riduce ulteriormente la quantità di codice da scrivere ed aumenta l'affidabilità dell'applicazione.
Utilizzando le funzionalità di ASP.NET per i servizi Web XML, è sufficiente che gli sviluppatori scrivano la propria logica di business e sarà poi l'infrastruttura di ASP.NET ad occuparsi dell'invio di tale servizio tramite SOAP ed altri protocolli pubblici.

ASP.NET può essere utilizzato con qualsiasi linguaggio e strumento di sviluppo (inclusi Visual Basic, C++, C# e JScript).

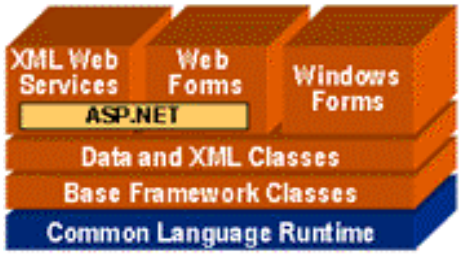


Figura 15: ASP.NET e livelli del .NET

ASP.NET visto da vicino

Alla base di ASP.NET c'è il suo runtime HTTP (diverso dal Common Language Runtime), un motore di esecuzione ad elevate prestazioni per l'elaborazione di comandi HTTP. Il runtime HTTP è responsabile dell'elaborazione di tutte le richieste HTTP in arrivo, della risoluzione dell'URL di ogni richiesta ad un'applicazione e dell'invio della

richiesta all'applicazione per ulteriori elaborazioni. Il runtime HTTP è multi-threaded ed elabora le richieste in modalità asincrona, ciò significa che l'elaborazione di nuove richieste non può essere bloccata da errori presenti nel codice applicativo. Inoltre, il runtime HTTP, grazie ad un'architettura robusta, è in grado di riprendersi automaticamente in seguito a violazioni d'accesso, memory leak, deadlock e quant'altro.

Aggiornamento delle applicazioni.

ASP.NET utilizza le tecnologie di distribuzione del Microsoft .NET Framework. Un altro importante vantaggio di ASP.NET è il supporto per l'aggiornamento a caldo delle applicazioni. Un amministratore non deve più eseguire lo shut down dell'applicazione o addirittura del server Web per aggiornare i file di quest'ultima: i file non vengono mai bloccati e pertanto possono essere sovrascritti anche quando l'applicazione è in esecuzione. Una volta aggiornati i file, il sistema passa automaticamente alla nuova versione.

Espandibile.

All'interno di un'applicazione ASP.NET, le richieste HTTP vengono inviate attraverso una serie di moduli HTTP fino ad arrivare ad un gestore di richieste. I moduli HTTP ed i gestori di richieste sono semplicemente delle classi .NET gestite che implementano specifiche interfacce definite da ASP.NET. Questo tipo di architettura modulare semplifica considerevolmente l'inserimento di servizi all'interno delle applicazioni: basta fornire un modulo HTTP.

Gestione dello stato.

Il Web rappresenta fondamentalmente un modello privo di stato senza alcuna correlazione tra le richieste HTTP. Ciò può rendere la creazione di applicazioni Web piuttosto difficoltosa, dal momento che di solito le applicazioni hanno bisogno di mantenere lo stato tra richieste successive. ASP.NET migliora i servizi per la gestione dello stato introdotti da ASP e fornisce tre tipi di stato per le applicazioni Web: application, session ed user. Lo stato sessiondi ASP.NET, ad esempio, viene memorizzato in un processo separato e può persino essere configurato per essere memorizzato su un computer distinto o reso persistente all'interno di un database SQL Server. Ciò rende lo stato session scalabile anche quando l'applicazione viene distribuita su farm Web di grandi dimensioni.

Memorizzare nella cache.

Il modello di programmazione di ASP.NET fornisce un'API che consente agli sviluppatori di attivare i servizi per la memorizzazione nella cache (nel software aziendale) al fine di aumentare le prestazioni. La output cache salva l'intera pagina una volta renderizzata (generata dopo l'interpretazione del codice), la fragment cache memorizza porzioni di pagine. ASP.NET mette a disposizione delle classi in modo che applicazioni, moduli HTTP e gestori delle richieste possano memorizzare oggetti arbitrari nella cache a seconda delle necessità.

ASP.NET è compilato, non interpretato

I programmi compilati sono eseguiti più velocemente di quelli interpretati. Quindi, ASP.NET, che è compilato, è eseguito più velocemente del ASP classico, che è interpretato. Ogni pagina è compilata alla prima richiesta; il codice compilato viene conservato fino a quando si modifica la pagina o si riavvia l'applicazione. Occasionalmente, i file possono essere precompilati in fase di distribuzione, per ridurre i

tempi di attesa.

Separazione del codice dal contenuto

ASP.NET consente la vera separazione del codice dalla presentazione, che consente ai designer e ai programmatori di collaborare con meno frustrazioni e tempo passato a unire aspetto e funzionalità delle pagine. Questo avviene grazie all'utilizzo di pagine “code behind", che sono referenziate usando una direttiva di pagina nell'intestazione della pagina con il codice di presentazione.

Fine del "DLL HELL"

Utenti e sviluppatori di componenti COM chiamano i problemi relativi alla distribuzione COM “DLL HELL”. Ciò significa che l'installazione o lo spostamento di componenti COM rompe le applicazioni dipendenti senza avvisare. Applicazioni altrimenti stabili vengono rotte quando una nuova applicazione aggiorna un componente esistente. Il “DLL HELL” esiste perché il protocollo COM richiede che i componenti non modifichino l'interfaccia; quindi, ogni volta che cambia interfaccia il componente ottiene un nuovo identificatore che crea una nuova versione del componente. I programmi che si basano su una versione di un componente non funzionano più quando cercano di comunicare con una nuova versione del componente. Chi ha installato un nuovo programma sul computer e ha scoperto che alcuni degli altri programmi non funzionano più dopo l'installazione, sa cos'è il “DLL HELL”. Con ASP.NET, i componenti non devono essere condivisi sul server, ma possono essere posizionati con le singole applicazioni. Inoltre. i componenti sono memorizzati con l'applicazione, che può essere spostata ricorrendo alla copia di file. Non è necessario apportare modifiche al Registro di sistema o preoccuparsi di MTS/COM+. Diventa molto facile mantenere applicazioni ASP.NET in modo remoto, per esempio mediante un provider di hosting Web. I componenti possono, comunque, essere condivisi, ma la decisione spetterà allo sviluppatore o all'amministratore.

Modello di programmazione basato sugli eventi

Le pagine ASP sono semplici script che iniziano l'esecuzione all’inizio del file e continuano riga per riga fino a raggiungere la fine dello script. Al contrario, le pagine ASP.NET seguono un modello di programmazione basato sugli eventi. L’esecuzione delle pagine può essere vista come una serie di eventi e gestori di eventi, come il caricamento di una pagina o un pulsante da premere. Questo elimina buona parte del “codice a spaghetti" associato alle pagine ASP, semplificandone la manutenzione e la modifica. Questo modello consente persino la scrittura di strumenti potenti per aiutare gli sviluppatori.

Accedere ai dati dal Web

Praticamente tutte le applicazioni hanno la necessità di interrogare o aggiornare dati persistenti memorizzati in file piatti, database relazionali o altri tipi di supporto di memorizzazione. Per ovviare a tale necessità, il .NET Framework include ADO.NET, un sottosistema per l'accesso ai dati ottimizzato per ambienti N-tier ed interoperabile con l'XML ed i documenti XML. ADO.NET è stato progettato per gli ambienti debolmente accoppiati. Come il nome stesso implica, ADO.NET rappresenta l'evoluzione di ADO (ActiveX® Data Objects).
ADO.NET è stato pensato per fornire servizi per l'accesso ai dati ad applicazioni scalabili e servizi basati sul Web. ADO.NET mette a disposizione API per modelli di dati sia connessi sia disconnessi particolarmente adatti alla restituzione di dati alle applicazioni Web.

Architettura dati disconnessa

(Vedi anche capitolo su ADO.NET)

Nelle tradizionali applicazioni two-tier, le componenti stabiliscono una connessione ad un database e la mantengono aperta per l'intera durata dell'esecuzione. Un tale tipo di approccio risulta decisamente poco pratico in numerose applicazioni:

- Aprire connessioni a database consuma risorse preziose sul server.
  Il sovraccarico necessario al mantenimento di tali connessioni influisce negativamente sulle prestazioni dell'applicazione.

- Le applicazioni che richiedono una connessione aperta a database risultano estremamente difficili da scalare.
  Non è detto che un'applicazione che funziona in modo accettabile con quattro utenti faccia altrettanto se gli utenti raggiungono il centinaio. Le applicazioni Web, in particolare, hanno bisogno di essere facilmente scalabili, dal momento che il traffico su un sito Web può aumentare considerevolmente in pochissimo tempo.

- Nelle applicazioni Web, le componenti sono di per sé stesse disconnesse le une dalle altre.
  Il browser richiede una pagina al server; una volta che questo ha terminato l'elaborazione ed ha inviato la pagina, non risulta più connesso al browser fino alla successiva richiesta. In queste circostanze, non ha senso mantenere aperta la connessione al database, dal momento che non c'è modo di sapere se il client chiederà o meno di accedere nuovamente ai dati.

- Un modello basato su dati connessi può complicare la condivisione dei dati tra componenti specialmente se queste appartengono ad applicazioni differenti.
  Se due componenti devono condividere gli stessi dati, è necessario che siano connesse, oppure occorre individuare un metodo che consenta di passare i dati dall'una all'altra.

Per tutti questi motivi, l'accesso ai dati in ADO.NET è stato progettato sulla base di un'architettura disconnessa. Le applicazioni rimangono connesse al database solamente per il tempo necessario ad estrarre o aggiornare i dati. Poiché il database non risulta legato

a connessioni potenzialmente inattive, può essere utilizzato da moltissimi utenti.

Memorizzazione dei dati in dataset

(Vedi anche capitolo su ADO.NET)

In un modello di dati disconnesso, non è pratico accedere al database ogni volta che l'applicazione deve elaborare il record successivo. (Così facendo si vanifica il vantaggio di lavorare con i dati disconnessi). La soluzione, pertanto, consiste nel memorizzare temporaneamente i record estratti dal database e lavorare su questo insieme temporaneo. Un dataset non è altro che un insieme di record estratti da un database e memorizzati in una cache local, che e si comporta come store di dati virtuale - include una o più tabelle che si basano sulle quelle di uno o più database - e può includere informazioni sulle relazioni tra le tabelle ed i vincoli sui dati in esse contenuti.

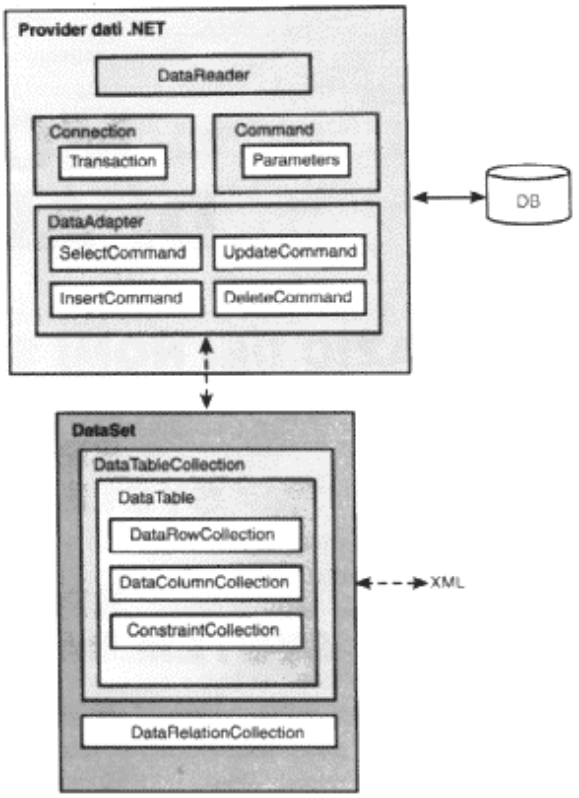


Figura 16: Accesso al DataBase e DataSet

I dati nel dataset rappresentano generalmente una versione molto ridotta di ciò che è contenuto nel database. Tuttavia, è possibile utilizzarli come se fossero dati reali. Durante questa fase, si rimane disconnessi dal database, che a sua volta è libero di effettuare altre operazioni.

Naturalmente, è necessario aggiornare spesso i dati nel database (sebbene non con la stessa frequenza con cui è necessario estrarli). A questo proposito è possibile effettuare le operazioni di aggiornamento sul dataset, e queste verranno propagate al database sottostante.

Un punto importante è che il dataset è un contenitore di dati passivo. Per estrarre effettivamente i dati da un database e (eventualmente) scriverli, occorre utilizzare un data adapter. Un data adapter contiene le istruzioni che consentono di popolare una singola tabella del dataset ed aggiornare la corrispondente tabella del database. Le istruzioni sono metodi che incapsulano codice SQL, come il riferimento ad una stored procedure. Pertanto, il metodo Fill può invocare un'istruzione SQL che viene eseguita ogni volta che

il metodo viene chiamato.

6. DIFFERENZE DI BASE TRA ASP E ASP.NET

Prima di creare la prima pagina ASP.NET, si dovrebbero esplorare le differenze di base tra ASP e ASP.NET, relative al file system e all'organizzazione, e fare alcune considerazioni sull' architettura.

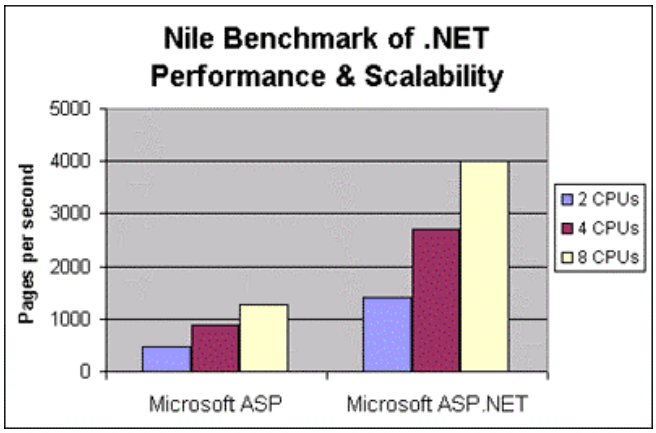


Figura 17: Benchmarck: ASP vs ASP.NET

FILE

La prima cosa da notare è la diversa estensione di file in ASP.NET. Invece di usare .asp, si usa .aspx. Allo stesso modo, se in ASP si usava il file global.asa per gestire alcuni eventi, nelle applicazìcni ASP.NET si userà il file global.asax. Oltre al file global.asax, le applicazioni ASP.NET hanno anche il file web.config, usato per definire molte impostazioni di configurazione specifiche. A differenza di global.asax i file web.config sono ereditati dalle applicazioni in sottocartelle. Infatti, tutte le applicazioni .NET su un dato server ereditano da un file config base, machine.config, situato nella directory del sistema operativo.

Tipi di file ASP.NET importanti

Estensione	Sostituisce il file ASP	Descrizione file
.asax	.asa	Contiene il gestore di eventi per gli eventi di applicazione, sessione e richiesta di file
.ascx	Nessuno	Controlli utente
.asmx	Nessuno	Servizi Web di ASP.NET
.aspx	.asp	Estensione predefinita delle pagine di ASP.NET
.config	Nessuno	File di Configurazione
.cs	Nessuno	File sorgente C#.NET
.vb	Nessuno	File sorgente Visual Basic.NET
.js	Nessuno	File sorgente Java Sctipt.NET

Preservare lo stato tra ASP e ASP.NET

Quando si migra da ASP ad ASP.NET si deve tenere presente che lo stato di sessione non è condiviso tra le due architetture. Ovvero, pur essendo possibile migrare ad ASP.NET una pagina alla volta. le pagine .aspx e .asp non condìvidono lo stesso stato di sessione. Se l'applicazione si basa strettamente sulle variabili di sessione, è opportuno valutare attentamente il problema, per il quale esistono fortunatamente diverse soluzioni.

Per preservare lo stato tra i file .asp e .aspx, è necessario usare un'alternativa all'oggetto session predefinito di entrambe le architetture. Esistono diverse possibilità, ciascuna con i propri vantaggi e svantaggi, e poiché le sessioni hanno diversi problemi di scalabilità in ASP classico, è probabile che non siano molto utilizzate dai siti e che, quindi, non sia molto difficile migrare ad ASP.NET. Invece di usare le variabili di sessione, si possono usare cookie, stringhe di query o campi nascosti. Ciascuna di queste tecniche consente di preservare le informazioni utente di pagina in pagina. Per dati molto piccoli e non sensibili, è sufficiente passare i dati. Tuttavia, per informazioni più importanti, si dovrebbe passare un identificatore univoco legato a un database che contiene i dati veri e propri. Usando uno di questi metodi, si riuscirà a preservare lo stato dell'applicazione tra le pagine .asp e .aspx.[4]

7. ADO E ADO.NET

STORIA DI MICROSOFT DATA ACCESS

Microsoft ha cambiato ancora una volta il modello a oggetti di accesso ai dati. Prima di capire il "come" di ADO.NET, è opportuno esaminare il "perché". Molti anni fa, esistevano diversi modelli a oggetti per l'accesso ai dati. In Access si doveva usare Data Access Objects (DAO). Grazie a DAO, si poteva trarre vantaggio dalle diverse funzioni specifiche di Access. Per esempio, si poteva creare, riparare e compattare i database. Si potevano collegare tabelle e creare QueryDefs. Access non era però l'unico database in circolazione. Come fare con Oracle o SQL Server?

La soluzione era Remote Data Objects (RDO). RDO andava bene per tutto tranne Access. RDO consentiva di comunicare con sorgenti dati Open Database Connectivity (ODBC) e, quindi, di accedere a SQL, Oracle, Informix, Sybase e altri tipi di database. Si potevano eseguire stored procedure e persino creare una cosa di nome "recordset disconnesso".

In entrambi i casi, si desiderava solo comunicare con il database, ma per ciò che era logicamente la stessa operazione si dovevano usare due API diverse. Passare da DAO e Access a RDO e SQL era un'operazione complicata. Ma DAO e RDO non erano sufficienti per tutto. Esistevano database non tradizionali con API personalizzate. Per esempio, Index Server ed Exchange usano API personalizzate per accedere ai dati.[4]

Subito dopo, vennero ADO e OLEDB. Nel l998 ADO iniziava ad affermarsi grazie alla commercializzazione di "Universal Data Access" di Microsoft. Esisteva finalmente una sola API per comunicare con qualsiasi sorgente dati. Almeno, questo era ciò che prometteva. ADO doveva combinare il meglio di DAO e RDO. Ma non fu così. Con ADO si perdeva l'abilità di compattare il database Access e, in molti casi non era veloce come RDO. ADO offriva però alcune funzioni utili. Consentiva la creazione di recordset gerarchici e l'accesso a più tipi di dati. Non ci si doveva rivolgere a ODBC perché esistevano "provider nativi" per la maggior parte dei database.

Ma ADO aveva un grosso svantaggio. Era basato sul concetto di connettività. Presumeva che l'utente volesse connettersi alla sorgente dati e rimanere connesso mentre recuperava i dati ed eseguiva operazioni. Si poteva creare un recordset disconnesso, ma non era l'impostazione predefinita. Il mondo dello sviluppo si muoveva intanto in una direzione diversa. Con le architetture a n livelli, aumentava la domanda di dati disconnessi. XML stava andando incontro ad un cambiamento tecnologico.

Microsoft.NET aveva una visione diversa del mondo, in cui ADO non rientrava. Nacque così ADO.NET. ADO.NET si basa sull'accesso disconnesso. In generale, ci si connette al database solo nel momento in cui si recupera o si aggiorna. Negli altri casi, la connessione è chiusa. Questa è l'impostazione predefinita. I dati recuperati sono memorizzati in un “DataSet". I DataSet sono recordset molto più potenti. Innanzi tutto, un singolo DataSet può memorizzare i risultati di molte query SQL. Per esempio, si possono recuperare tutti gli autori e memorizzare i risultati in un DataSet. Quindi. si possono recuperare tutti gli editori e memorizzare i risultati nello stesso DataSet. Con un'altra riga di codice, si può dire al DataSet che gli autori e gli editori sono collegati da una colonna. In altre parole, il DataSet può agire da database limitato in memoria, con una comprensione completa delle relazioni tra le tabelle.Questo database in memoria è

progettato per non richiedere una connessione attiva al database vero e proprio.

Esiste un'altra differenza fondamentale tra ADO e ADO.NET. Con ADO, il Recordset poteva fare tutto. Non solo consentiva l'accesso ai dati, ma offriva funzioni per filtrare e ordinare i dati. Il Recordset controllava i dati e la loro visualizzazione. ADO.NET divide questa funzionalità tra due oggetti. Il DataSet si occupa solo della memorizzazione dei dati. Per la visualizzazione dei dati, che include il filtraggio e l'ordinamento, si usa un oggetto Dataview.

Come già accennato, il DataSet tiene una copia in memoria dei dati restituiti da un'istruzione select. Spesso. questo è proprio ciò che si desidera, ma non è molto pratico per DataSet di grandi dimensioni. Per esempio, come fare per eseguire sul database operazioni massicce di importazione/esportazione? Non si vorrà certo caricare in memoria l'intero database. Per fortuna, esiste un oggetto che fa al caso nostro: DataReader. Il DataReader ha la stessa funzione di un recordset forward-only, read-only di ADO. Consente di iterare un insieme di righe, una alla volta. Inoltre, in molte situazioni, iterare l'insieme dei risultati una sola volta è tutta la funzionalità richiesta.

MODELLO A OGGETTI DI ADO E ADO.NET

Le tabelle confrontano gli oggetti in ADO con i nuovi oggetti in ADO.NET.

Oggetti ADO	
Oggetto	**Descrizione**
Connection	Consente al codice di connettersi a una sorgente dati. Mediante la connessione si possono eseguire comandi che aggiornano il database o restituiscono record.
Command	Usato per inviare comandi al database. Questo oggetto è utile per chiamare store procedure perché contiene un insieme "`Parameters`"da usare per accedere ai parametri di input e output di una stored procedure.
Recordset	Memorizza i risultati di un insieme "`select`". Il recordset è di sola lettura e usa un cursore sul lato server. Consente di creare cursori sul lato server o client e di specificare diverse opzioni di blocco.

ADO è stato definito un *modello a oggetti piatto* perché erano disponibili molte funzionalità duplicate mediante oggetti diversi. L'oggetto connection consentiva di connettersi al database, ma anche di eseguire comandi. Il recordset era usato per memorizzare i risultati, ma anche per mantenere una propria connessione interna ed eseguire comandi. Pur sembrando utile questo era fonte di confusione e complessità. In ADO.NET i singoli oggetti non fanno di tutto. Un oggetto connection è solo una connessione e non consente di eseguire comandi come in ADO. Un oggetto command consente solo di eseguire comandi. Un DataSet memorizza solo dati. Con un recordset ADO, sono disponibili diversi comportamenti incoerenti, in base al database sottostante. Con il DataSet, è disponibile lo stesso comportamento su tutti i database perché non conosce il database sottostante.

Oggetti ADO.NET	
Oggetto	**Descrizione**
DBConnection, SQLConnection, ADOConnection	Usati per stabilire una connessione ad un database.
DBCommand, SQLCommand, ADOCommand	Usati per inviare istruzioni INSERT, UPDATE, SELECT, DELETE al database.
DataSet	Memorizza i risultati di una o più "`select`".
DataView	Usato per filtrare e ordinare un DataSet
DBDataReader, SQLDataReader, ADODataReader	Usato per iterare un insieme di record

8. Applicazione Realizzata

Esempi di applicazione ASP.NET

DATADRID: TABELLA realizzabile in modo automatico o non automatico.

PARTE IN ASP.NET:

```
<ASP:DataGrid id="MyDataGrid" runat="server"
      Width="800"
      BackColor="#ccccff"
      BorderColor="black"
      ShowFooter="false"
      CellPadding=3
      CellSpacing="0"
      Font-Name="Verdana"
      Font-Size="8pt"
      HeaderStyle-BackColor="#aaaadd"
      DataKeyField="St_Codice"
      OnEditCommand="MyDataGrid_Edit"
      OnCancelCommand="MyDataGrid_Cancel"
      OnUpdateCommand="MyDataGrid_Update"
      AutoGenerateColumns="false"  // generato in modo manuale
    >

<Columns>

 <asp:EditCommandColumn EditText="Modifica" ButtonType="PushButton"
CancelText="Annulla" UpdateText="Aggiorna" ItemStyle-Wrap="false"/>
<asp:BoundColumn HeaderText="St_Codice" SortExpression="St_Codice"
ReadOnly="True" DataField="St_Codice" ItemStyle-Wrap="false"/>
<asp:BoundColumn HeaderText="St_Descrizione"
SortExpression="St_Descrizione" ReadOnly="False"
DataField="St_Descrizione"/>
<asp:BoundColumn
HeaderText="St_Descrizione_estesa"SortExpression="St_Descrizione_estesa"
ReadOnly="False" DataField="St_Descrizione_estesa"/>
<asp:BoundColumn HeaderText="St_Booleano" SortExpression="St_Booleano"
ReadOnly="False" DataField="St_Booleano"/>
<asp:BoundColumn HeaderText="St_Numerointero"
SortExpression="St_Numerointero" ReadOnly="False"
DataField="St_Numerointero"/>
<asp:BoundColumn HeaderText="St_Numerodecimale"
SortExpression="St_Numerodecimale" ReadOnly="False"
DataField="St_Numerodecimale"/>
<asp:BoundColumn HeaderText="St_Data" SortExpression="St_Data"
ReadOnly="False" DataField="St_Data"/>
<asp:BoundColumn HeaderText="St_Dataannullamento"
SortExpression="St_Dataannullamento" ReadOnly="True"
DataField="St_Dataannullamento"/>

</Columns>


</ASP:DataGrid>
```

PARTE IN C#:

```
//---------------------------------------------------------------------
//--FUNZIONE CHE SI ATTIVA ALLA PRESSIONE DEL PULSANTE MODIFICA DEL
DATAGRID--
//---------------------------------------------------------------------
```

```
public void MyDataGrid_Edit(Object sender, DataGridCommandEventArgs e)
{
      MyDataGrid.EditItemIndex = (int)e.Item.ItemIndex;
      BindGrid();
}

//---------------------------------------------------------------------
//--FUNZIONE CHE SI ATTIVA ALLA PRESSIONE DEL PULSANTE ANNULLA DEL
DATAGRID--
//---------------------------------------------------------------------

public void MyDataGrid_Cancel(Object sender, DataGridCommandEventArgs e)
{
      MyDataGrid.EditItemIndex = -1;
      BindGrid();
}

//-------------------------------------------------------------
//-------- FUNZIONE ORDINA LE COLONNE -------------------------
//---------solo se la proprietà AllowSorting è impostata a TRUE-
//-------------------------------------------------------------

public void ordina_Colonna(Object sender_ordina,
DataGridSortCommandEventArgs e_Ordina)
{

      DataView vista = new DataView();

      vista = set_connetti.rsMenu.Tables[tabella].DefaultView;

      vista.Sort= e_Ordina.SortExpression.ToString();
      myDataGrid.DataSource = vista;
      myDataGrid.DataBind();
}

//---------------------------------------------------------------------
//--FUNZIONE CHE SI ATTIVA ALLA PRESSIONE DEL PULSANTE AGGIORNA DEL
DATAGRID--
//---------------------------------------------------------------------

public void MyDataGrid_Update(Object sender, DataGridCommandEventArgs e)
{
//--------STRINGA---CONNESSIONE--------------
SqlConnection myConnection = new SqlConnection("user id=daniele;
password=daniele; server=local;database=Standard;");

//---------STRINGA---UPDATE--------------
String updateCmd = "UPDATE Standard SET
St_Descrizione_estesa=@St_Descrizione_estesa,
St_Descrizione=@St_Descrizione, St_Booleano=@St_Booleano,
St_Numerointero=@St_Numerointero , St_Numerodecimale=@St_Numerodecimale,
St_Data=@St_Data where St_Codice=@St_Codice";

//---------COMANDO CONTENENTE LA CONNESSIONE E LA STRINGAB DI UPDATE ---
SqlCommand myCommand = new SqlCommand(updateCmd, myConnection);

//---------PARAMETRI DEL UPDATE -------
myCommand.Parameters.Add(new SqlParameter("@St_Codice", SqlDbType.NChar,
20));
myCommand.Parameters.Add(new SqlParameter("@St_Descrizione",
SqlDbType.NChar, 50));
myCommand.Parameters.Add(new SqlParameter("@St_Descrizione_estesa",
SqlDbType.NVarChar, 256));
myCommand.Parameters.Add(new SqlParameter("@St_Booleano", SqlDbType.Int,
1));
myCommand.Parameters.Add(new SqlParameter("@St_Numerointero",
SqlDbType.Int, 4));
myCommand.Parameters.Add(new SqlParameter("@St_Numerodecimale",
```

```
SqlDbType.Decimal, 9));
myCommand.Parameters.Add(new SqlParameter("@St_Data",
SqlDbType.DateTime, 4));

//---------LEGO I PARARAMETRI CON I VALORI -------
myCommand.Parameters["@St_Codice"].Value =
MyDataGrid.DataKeys[(int)e.Item.ItemIndex];
myCommand.Parameters["@St_Descrizione"].Value =
System.Convert.ToString(((TextBox)e.Item.Cells[2].Controls[0]).Text);
myCommand.Parameters["@St_Descrizione_estesa"].Value =
System.Convert.ToString(((TextBox)e.Item.Cells[3].Controls[0]).Text);
myCommand.Parameters["@St_Numerointero"].Value =
System.Convert.ToInt32(((TextBox)e.Item.Cells[5].Controls[0]).Text);
myCommand.Parameters["@St_Numerodecimale"].Value =
System.Convert.ToDecimal(((TextBox)e.Item.Cells[6].Controls[0]).Text);
myCommand.Parameters["@St_Data"].Value =
System.Convert.ToDateTime(((TextBox)e.Item.Cells[7].Controls[0]).Text);

String[] cols = {"@St_Codice", "@St_Descrizione",
"@St_Descrizione_estesa", "@St_Booleano", "@St_Numerointero",
"@St_Numerodecimale", "@St_Data", "@St_Dataannullamento"};

int numCols = e.Item.Cells.Count;

//-------------------------------------------------
//-----------Controllo se  ho dei campi vuoti------
//-------------------------------------------------

for (int i=2; i<numCols-1; i++)
{
String colvalue =((TextBox)e.Item.Cells[i].Controls[0]).Text;

// check for null values in required fields

if (i<6 && colvalue == "")
{
Message.InnerHtml = "ERRORE. Valori Null non consentiti per Codice,
Descrizione, Data o Data annullamento";
Message.Style["color"] = "red";
return;
}

}

//conversione dei valori true/false in 0/1

if ( String.Compare ( ((TextBox)e.Item.Cells[4].Controls[0]).Text,
"true", true)==0)
      myCommand.Parameters["@St_Booleano"].Value =  "1";
else
      myCommand.Parameters["@St_Booleano"].Value =  "0";

myCommand.Connection.Open();

//-------------------------------------------------
//---------SEQUENZA DI AGGIORNAMENTO ----------
//-------------------------------------------------

try
{
   myCommand.ExecuteNonQuery();
    Message.InnerHtml = "<b>Record aggiornato</b><br>" + updateCmd;
    MyDataGrid.EditItemIndex = -1;
   ALL();
}
catch (SqlException exc)
{
    if (exc.Number == 2627)
    Message.InnerHtml = "ERRORE. Esiste già un record con la stessa PK";
```

```
   else
       Message.InnerHtml = "ERRORE. Impossibile aggiornare il record.
Verificare che le informazioni immesse nei campi siano corrette";

       Message.Style["color"] = "yellow";
       BindGrid();
}

myCommand.Connection.Close();


}

//----------------------------------------
//--FUNZIONE CHE SELEZIONA DAL DATABASE --
//----------------------------------------

public void BindGrid()
{
SqlConnection myConnection = new SqlConnection("user id=daniele;
password=daniele; server=local; database=Standard;");

SqlDataAdapter myCommand = new SqlDataAdapter("SELECT * from Standard",
myConnection);

DataSet dati = new DataSet();
myCommand.Fill(dati, "Standard");

MyDataGrid.DataSource=dati.Tables["Standard"].DefaultView;
MyDataGrid.DataBind();
}
```

DROPDOWNLIST: MENU A TENDINA

PARTE IN ASP.NET:

```
<asp:Dropdownlist id="myDropDownList"
                        width="150"
                        runat="server"
                />
```

PARTE IN C#:

```
//-------------------------------------------------------------------
//--FUNZIONE CHE SELEZIONA DAL DATABASE LGI ELEMENTI DELLA LISTA----
//-------------------------------------------------------------------

public void creaDropDownList()
{
       myConnectionDropDown = new SqlConnection("user
id=webuser;password=;server=local;database= database ");

      myCommandDropDown = new SqlDataAdapter("SELECT ID,categoria FROM
Table", myConnectionDropDown);
      myCommandDropDown.Fill(rsList,"Table"); // riempio l'oggetto dei
dati


      //.................................................
      //............SETTAGGIO PARAMETRI DROPDOWNLIST.......
      //.................................................

      myDropDownList.DataTextField = "categoria";
      myDropDownList.DataValueField = "ID";
```

```
        myDropDownList.DataSource=rsList;
        myDropDownList.DataBind();

        //.....................................................
        //.........AGGIUNGO L'ELEMENTO: TUTTE LE CATEGORIE.......
        //.....................................................

        myDropDownList.Items.Insert(0,"Tutte le Categorie");
        myDropDownList.SelectedItem.Value = "Tutte le Categorie";

}
```

CHIAMATA DI UNA FUNZIONE PRESENTE IN UNA DLL SEPARATA:

Config e connessione sono oggetti creati mediante una classe presente in due DLL esterne:

```
public void elimina_Elemento(Object sender_elimina,
DataGridCommandEventArgs e_Elimina)
{
        //.........................................
        //......Controllo se esiste il file su disco..
        //.........................................

if (File.Exists(config.URL +/"+config.PERCORSO_UPLOAD
+"/"+e_Elimina.Item.Cells[5].Text) )
       {
        //.........................................
        //......CHIAMATA DELLA FUNZIONE.............
        //.........................................

connessione.aggiorna("moduli", "elimina", e_Elimina.Item.Cells[2].Text,
null, null, null, null, null);

             Message.InnerHtml = connessione.Message;
             Message.Style["color"] = "white";

        //.............................................................
        //......Se la cancellazione dal database è andata a ........
        //........buon fine elimino il file da disco................
        //.............................................................

             if (set_connetti.flag == false)
             {
                  File.Delete(config.URL +/"+config.PERCORSO_UPLOAD
+"/"+e_Elimina.Item.Cells[5].Text);

             }

       }
       else
       {
             Message.InnerHtml = "FILE UPLODATO NON TROVATO. <BR>
IMPOSSIBILE ELIMINARE IL RECORD!";
       }
         myDataGrid.EditItemIndex = -1;

         //.....................................................
         //......Resetto e ripopolo il dataste data => il datagrid....
         //.....................................................

       connessione.data.Clear();

       connessione.visualizza("ALL", campi, null);

       myDataGrid.DataSource = connessione.data;
       myDataGrid.DataBind();
```

```
}
```

La funzione connessione.aggiorna richiama a sua volta una funzione in base al tipo di servizio che l'ha invocata, quest'ultima funzione legata al servizio chiama un'altra funzione in base all'operazione da fare; in questo caso l'eliminazione di un record:

```
//-----------------------------------------------------------------------
//-------FUNZIONE ELIMINA RECORD MODULI---------------------------------
//-----------------------------------------------------------------------

public void moduli_elimina(String ID)
{
SQLConnection myConnection = new SqlConnection("user id=daniele;
password=daniele; server=local;database=Standard;");

String deleteCmd = "DELETE FROM "+config.Table+" WHERE ID = @ID";
myCommand_aggiorna = new SqlCommand(deleteCmd, myConnection);
myCommand_aggiorna.Parameters.Add(new SqlParameter("@ID", SqlDbType.Int,
4));
myCommand_aggiorna.Parameters["@ID"].Value = ID;
myCommand_aggiorna.Connection.Open();

try
{
      myCommand_aggiorna.ExecuteNonQuery();
      Message = "<b>RECORD ELIMINATO</b>";
//.....................................................................
//.Il falg serve per confermare all'esterno della DLL se l'aggioenamento
è //.................avvenuto con successo..............

flag = false;
}
catch
{
Message = "ERRORE:<BR> Impossibile eliminare il record";
flag = true;
}
myCommand_aggiorna.Connection.Close();

}
```

Nella DLL esterna la funzione aggiorna mediante uno "switch case", esegue la funzione di elimina specifica:

```
public void moduli_elimina(String ID)
{
      String deleteCmd = "DELETE FROM "+ config.TABELLA +" WHERE ID =
@ID";

        myCommand_aggiorna = new SqlCommand(deleteCmd, myConnection);

        myCommand_aggiorna.Parameters.Add(new SqlParameter("@ID",
SqlDbType.Int, 4));
        myCommand_aggiorna.Parameters["@ID"].Value = ID;

        myCommand_aggiorna.Connection.Open();

        try
        {
            myCommand_aggiorna.ExecuteNonQuery();
            Message = "<b>RECORD ELIMINATO</b>";
          flag = false;
        }
        catch
```

```
        {
            Message = "ERRORE:<BR> Impossibile eliminare il record";
          flag = true;
        }

        myCommand_aggiorna.Connection.Close();

}
```

UPLOAD DI FILE:

La funziona realizzata è legata ad un pulsante. File, campo_testo, data e vecchia_data_completa sono Web Form presenti nella pagina di inserimento. Il flag serve per controllare se esiste un errore. Utilizzo il metodo che chiama di funzioni presenti in da DLL esterne, connetti e config:

```
public void SubmitBtn_Inserimento_Click(object Source, EventArgs e)
{

      //.....................................................
      //.........ESTRAPOLO IL NOME DEL FILE DAL CONTROLLO FILE ....
      //.....................................................

      String linkdocumento = “”;

      file.PostedFile.FileName.Trim();

      for (int i=0; i <file.PostedFile.FileName.Length; i++)
      {
            if (file.PostedFile.FileName[i] != '\\' )
            {
                  linkdocumento += file.PostedFile.FileName[i];
            }
            else
            {
                  linkdocumento = "";
            }

      }

      //.....................................
      //.........CONTROLLO DEL TIPO DI FILE ...
      //.....................................


      String estensione = "";

      for (int i=0; i<linkdocumento.Length; i++)
      {
            if (linkdocumento[i] != '.')
            {
                  estensione += linkdocumento[i]; // estrapolo
l’estensione
            }
            else
            {
                  estensione = "";
            }
      }



      //.....................................
      //.........CONTROLLO TIPO FILE ....
```

```
      //.....................................

if  ( (file.PostedFile.ContentType  != "text/plain")
       && (file.PostedFile.ContentType  != "text/richtext")
       && (file.PostedFile.ContentType  != "text/msword")
       && (file.PostedFile.ContentType  != "image/pjpeg")
       && (file.PostedFile.ContentType  != "image/gif") )
{

//.................................................
//.........  TIPO DI FILE ERRATO    ............
//.................................................

      Message.InnerHtml = "<b>ERRORE:<BR> SELEZIONA UN FILE VALIDO: DOC,
TXT, RTF, GIF, JPG</b><br>";
      MessageFile.InnerHtml = "File scelto:
<BR>"+file.PostedFile.FileName;
      connetti.flag = true;
}


//.................................................
//.......   CATEGORIA NON SELEZIONATA  ............
//.................................................

if (myDropDownList.SelectedItem.Value == "Tutte le Categorie")
{
      Message.InnerHtml = "<b>ERRORE:<BR> SELEZIONA UNA
CATEGORIA</b><br>";
      MessageFile.InnerHtml = "File scelto:
<BR>"+file.PostedFile.FileName;
      connetti.flag = true;
}

//.................................................
//.............   OGGETTO VUOTO  ...........
//.................................................

if (campo_testo.Value == "")
{
      Message.InnerHtml = "<b>ERRORE:<BR> RIEMPI IL CAPO
OGGETTO</b><br>";
      MessageFile.InnerHtml = "File scelto:
<BR>"+file.PostedFile.FileName;
      connetti.flag = true;
}

      //.....................................
      //.........FASE DI UPLOAD ....
      //.....................................

if ( (set_connetti.flag == false) && (file.PostedFile != null) )
{
            //.....................................
            //.........CONTROLLO ESISTENZA DEL FILE ....
            //.....................................

if (File.Exists(config.URL +"/"+ config.PERCORSO_UPLOAD
+"/"+linkdocumento) )
{
      if (sovrascrivi.Checked == false)
      {
      Message.InnerHtml = "<b>ATTENZIONE:<BR> FILE GIA' ESISTENTE<br>";
      MessageFile.InnerHtml = "File scelto:
<BR>"+file.PostedFile.FileName;
      sovrascrivi.Visible = true; // abilito il chekbox di scelta
      return;
      }
```

```
      else
      {
      sovrascrivi.Visible = false;


      //.....................................................
      //.........Chiamo la funzione di aggiornamento/Inserimento ...
      //.....................................................

          connetti.aggiorna( "moduli", tipo_pagina.Value, null,
data_completa, linkdocumento, scadenza_completa,
myDropDownList.SelectedItem.Value, testo.Value);
          Message.InnerHtml = set_connetti.Message;

            //....................................
            //.......Se tutto Ok copio il file ......
            //....................................

         if (connetti.flag == false )
           {
                  file.PostedFile.SaveAs(config.URL +"/"+
config.PERCORSO_UPLOAD +"/"+linkdocumento);
            }
      }

      }
else
{
      //.....................................................
      //.........Chiamo la funzione di aggiornamento/Inserimento ...
      //.....................................................

    connetti.aggiorna( "moduli", tipo_pagina.Value, null, data_completa,
linkdocumento, scadenza_completa, myDropDownList.SelectedItem.Value,
testo.Value);
    Message.InnerHtml = connetti.Message;

            //....................................
            //.......Se tutto Ok copio il file ......
            //....................................

   if ( set_connetti.flag == false )
     {
            file.PostedFile.SaveAs(config.URL +"/"+
config.PERCORSO_UPLOAD +"/"+linkdocumento);
      }
}

}

}
```

Breve Glossario

Termine	Definizione
.NET Framework	Il .NET Framework è una piattaforma per la realizzazione della nuova generazione di applicazioni e servizi Web XML distribuiti. Espone un modello di programmazione coerente ed indipendente dal linguaggio e condiviso da tutti gli strati di un'applicazione e consente di interoperare in modo trasparente e di migrare facilmente a partire dalle tecnologie esistenti. Il .NET Framework è costituto da tre componenti fondamentali: il Common Language Runtime, le classi unificate ed ASP.NET
ADO.NET	La tecnologia del .NET Framework per l'accesso ai dati.
API Web	API che consentono l'integrazione di un servizio Web XML all'interno del .NET Framework
Architettura debolmente accoppiata	Un'architettura distribuita per la quale è possibile modificare l'implementazione di uno strato senza influenzare gli altri. Contrapposta ad architettura fortemente accoppiata
Architettura fortemente accoppiata	Un'architettura distribuita in cui una modifica ad uno strato qualsiasi influenza alcuni strati, se non tutti. Contrapposta ad Architettura debolmente accoppiata
ASP.NET	La tecnologia Active Server Pages per il .NET Framework
Assembly	L'unità per il deployment ed il controllo dei conflitti di versione del .NET Framework. Definisce gli spazi dei nomi per soddisfare le richieste e determina quali risorse esporre esternamente e quali rendere, invece, accessibili esclusivamente dall'interno dell'assembly. Un assembly include un manifesto che ne descrive i contenuti
C#	Il primo linguaggio orientato alle componenti della famiglia C/C++. È stato sottoposto all'ECMA per la standardizzazione
CGI	Common Gateway Interface – il primo protocollo Internet utilizzato per generare contenuti interattivi sul Web.
Classi unificate	L'insieme di librerie di classi (API) unificato, orientato all'oggetto, gerarchico ed estensibile del .NET Framework che gli sviluppatori possono utilizzare con il linguaggio con il quale hanno maggior familiarità
Codice gestito	Il codice gestito fornisce i metadati necessari al CLR per garantire servizi quali la gestione della memoria, l'integrazione cross-language, la sicurezza per l'accesso al codice ed il controllo automatico del ciclo di vita degli oggetti. Tutto il codice che si basa sull'IL viene eseguito come codice gestito
Codice nativo	Codice compilato in codice macchina specifico per il processore
Codice non gestito	Codice creato senza la conoscenza delle convenzioni e dei requiaiti del .NET Framework. Il codice non gestito viene eseguito all'interno dell'ambiente del .NET Framework solo con i servizi minimi (ad esempio, nessuna garbage collection, debug limitato, nessuna sicurezza dichiarativa). Il codice non gestito non possiede metadati che lo descrivono
Common Language Runtime (CLR)	I sistemi dei tipi dei metadata e di esecuzione forniti dal .NET Framework, che forniscono codice gestito e dati con servizi quali l'integrazione cross-language, la sicurezza per l'accesso al codice, la gestione del tempo di vita degli oggetti ed il supporto per il debugging ed il profiling. Utilizzando il CLR, i compilatori e gli altri strumenti consentono agli sviluppatori di sfruttare tutti questi servizi
Common Language Specification (CLS)	Un sottoinsieme delle funzionalità del .NET Framework che viene supportato da un ampio insieme di linguaggi e strumenti compatibili. I linguaggi e gli strumenti compatibili con il CLS interoperano tra loro. Ad esempio, il tipo Int32 è compatibile con il CLS ed i linguaggi e gli strumenti possono aspettarsi che altri linguaggi e strumenti CLS sappiano come utilizzarlo correttamente
SOAP Disco	SOAP Discovery (Disco) - basato sulle specifiche di SOAP Discovery, fornisce un insieme di regole per individuare la descrizione e le funzionalità dei servizi Web

ECMA	Un ente europeo per gli standard nato nel 1961. Accreditato a livello internazionale, l'ECMA è l'organismo che ha approvato numerosi standard pubblici di successo come ECMAScript.
Form Web	Le form Web sono una tecnologia ASP.NET utilizzabile per realizzare pagine Web programmabili. Le form Web possono visualizzare informazioni all'utente, utilizzando un qualsiasi linguaggio di marcatura, in qualsiasi browser ed utilizzare il codice lato server per implementare la logica applicativa
Form Windows	La struttura form Windows incapsula le API Win32 native ed espone classi gestite sicure per la creazione di applicazioni Win32 lato client. La libreria di classi delle form Windows fornisce numerosi controlli quali pulsanti, caselle di controllo, caselle di riepilogo, caselle combinate, griglie ed altro ancora che incapsulano l'interfaccia utente ad altre funzionalità lato client
Garbage collection	Il processo che permette di tener traccia transitivamente di tutti i puntatori agli oggetti attualmente in uso per individuare tutti quelli referenziabili e quindi riarrangiarli per riutilizzare tutte le porzioni della memoria del heap non individuate in questa fase. Il garbage collector del CLR, inoltre, opera per compattare la memoria in uso per ridurre l'area di lavoro necessaria al heap
HTTP	Hyper Text Transfer Protocol, protocollo Internet standard per trasferire informazioni tra client e server, server e server e così via
IDL	Interface Definition Language – un linguaggio utilizzato dall'applicazione per specificare le interfacce che intende mettere esporre ad altre applicazioni
Intermediate Language (IL)	L'Intermediate Language (IL) è un linguaggio utilizzato come output per numerosi compilatori e come input per un compilatore JIT. L'IL definisce un'architettura di esecuzione astratta e basata sullo stack. Il CLR può includere diversi compilatori JIT per convertire l'IL in codice nativo
JIT	Just-In-Time – una frase che descrive un'azione che viene eseguita solo quando necessario, come la compilazione Just-In-Time o l'attivazione dell'oggetto Just-In-Time. Per convenzione, il termine JIT viene utilizzato per riferirsi ad un compilatore JIT
Manifesto	Metadati che descrivono i moduli ed i file di risorse che fanno parte di un particolare assembly, quali sono i tipi esportati e quali altri assembly sono referenziati. Il manifesto specifica, inoltre, i permessi di sicurezza necessari, quelli opzionali e quelli che non vengono accettati dall'assembly
Metadati	Informazioni sui dati. Nel CLR, i metadati vengono utilizzati per descrivere assembly e tipi. I metadati vengono memorizzati assieme a questi all'interno dei file eseguibili ed utilizzati da compilatori, strumenti e dal runtime per fornire un ricco insieme di servizi. I metadati sono essenziali per le informazioni sul tipo di runtime e per l'invocazione di metodi dinamici. Sono molti i sistemi che utilizzano i metadati, ad esempio le type library di COM
N-tier	Architettura di sistema che separa gli strati di interfaccia, logica di business, accesso ai dati e database (o altri meccanismi di persistenza)
Reflection	Tecnologia del .NET Framework che consente di esaminare i metadati che descrivono i tipi ed i relativi membri
Servizio Web XML	Un servizio Web XML è un'applicazione che espone le proprie funzionalità da codice su Internet o una intranet utilizzando protocolli standard per Internet e standard come HTTP e XML
Sicurezza basata sull'evidenza	Il .NET Framework introduce il concetto di sicurezza basata sull'evidenza, che si riferisce ad input per i criteri di sicurezza del codice quali informazioni circa il sito, la zona di sicurezza o l'URL di riferimento di un assembly, la relativa nomenclatura forte e l'eventuale disponibilità per una firma digitale e relativo autore. Sulla base di queste ed altre informazioni, è possibile applicare appropriati criteri di sicurezza ed impostare i permessi corretti sull'assembly. Le risposte possono derivare da più risorse, tra cui uno o più CLR, il browser, ASP.NET e la shell a seconda della sorgente del codice

SOAP	Simple Object Access Protocol, standard WC3 – un protocollo leggero per lo scambio di informazioni in un ambiente decentralizzato e distribuito. È un protocollo basato su XML costituito da tre parti: uno strato esterno che definisce una struttura per la descrizione e l'elaborazione del contenuto di un messaggio, un insieme di regole di codifica per esprimere istanze di tipi di dato definiti dall'applicazione, ed una convenzione per la rappresentazione di invocazioni a procedure remote e relative risposte
UDDI	Le specifiche Universal Description, Discovery and Integration [UDDI]– un'iniziativa che crea una struttura aperta, globale ed indipendente dalla piattaforma che consente ai servizi aziendali di (1) individuarsi gli uni rispetto agli altri, (2) definire le modalità di interazione su Internet, e (3) condividere informazioni in un registry globale
WebMethod	Parola chiave del .NET Framework che consente di accedere agli oggetti da Internet
WSDL	Web Service Description Language – una grammatica XML utilizzabile da sviluppatori e strumenti di sviluppo per rappresentare le funzionalità di un servizio Web XML
XML	Extensible Markup Language – uno standard WC3 per la formattazione di documenti e dati strutturati sul Web.

Conclusioni

Il progetto che dovevo realizzare è stato portato a termine completamente, ovviamente rientra in un insieme molto più grande, chi avrà il compito di continuare troverà una buona base.

Il lavoro svolto per riprogettare la rete civica non è concluso, ancora molto c'è da fare, ma le basi sono state poste. La maggior parte delle funzioni base dei servizi è stata realizzata, non resta che personalizzare ciò che è stato fatto per ogni singolo servizio.

L'esperienza di stage ha permesso di conoscere in modo approfondito una nuova piattaforma di sviluppo che sarà uno degli standard per lo sviluppo del Web nel prossimo futuro.
Ciò si aggiunge al bagaglio di conoscenze che il corso di studi mi ha dato. Ha ampliato le conoscenze delle fasi dello sviluppo di un software per il Web, portato alla conoscenza di un nuovo linguaggio di programmazione(C#) per ambiente Web e Windows ® e di un evoluzione di un linguaggio(ASP.NET) per la creazione di pagine dinamiche per internet.

Concludendo il progetto svolto è stato molto interessante e particolarmente stimolate in quanto ha offerto la possibilità di lavorare con qualcosa di nuovo, di affrontare tante problematiche, anche vecchie, in modo diverso, di poter valutare e scegliere gli strumenti, quelli che più si addicevano la proprio modo di lavorare e alle reali necessità della realizzazione del progetto.

INFO:

Contemporaneamente alla fine della stesura di questa tesi la Microsoft ha rilasciato gratuitamente un Software per la realizzazione di pagine ASP.NET: ***ASP.NET Web Matrix Project*** unito a ***MSDE***

Occupa molto poco spazio su disco rispetto a Visual Studio, è gratuito, integra un piccolo WebServer per caricare le pagine ASP.NET senza IIS, richiesto per la pubblicazione finale delle pagine.

Non ho informazioni a sufficienza per valutarne l'efficienza, ma è sicuramente un aiuto per chi realizzerà in futuro solo ASP.NET, non dovrà appoggiarsi alla pesante struttura di Visual Studio .NET

Bibliografia

Link Internet

.NET Home page:
www.microsoft.com/net
Visual C# .NET:
http://msdn.microsoft.com/vcsharp/
Visual Basic .NET:
http://msdn.microsoft.com/vbasic/
Tutto su .NET (esempi, help, articoli, ecc..):
www.gotdotnet.com
World Wide Web Consortium:
www.w3c.org

Libri

[1] Titolo: Microsoft Visual C#.NET "Passo per Passo"
Aurore/i: John Sharp, Jon Jagger
Editore: Mondadori Informatica

[2] Titolo: Microsoft Visual C#.NET "Il Linguaggio"
Aurore/i: Microsoft Corporation
Editore: Mondadori Informatica

[3] Titolo: Microsoft ASP.NET "Passo per Passo"
Aurore/i: Duthie G.Andrew
Editore: Mondadori Informatica

[4] Titolo: ASP.NET Practical
Aurore/i: Steven Smith
Editore: Jackson Libri