# INDICE

# Capitolo1

# INTRODUZIONE

La presente tesi completa un progetto precedentemente affrontato da un collega riguardante la creazione di pagine Web dinamiche grafiche per la progettazione concettuale ER di database, la traduzione degli schemi disegnati in RDF, il salvataggio del file RDF e la successiva ricostruzione, dal file salvato dello schema ER.

Per sviluppare l'applicazione si è utilizzato il nuovo framework .NET di Microsoft. Questa scelta deriva dal fatto che la tecnologia ASP.NET sostituisce il modello lineare di elaborare le pagine di ASP con un'emulazione del modello ad eventi. Il framework delle pagine ASP.NET è fornito di un implicito produttore di associazioni tra un evento e il suo ascoltatore e permette di creare semplicemente interfacce utente che reagiscono alle azioni dell'utente.

Come linguaggio di programmazione per la parte di codice per le pagine ASP.NET si è scelto il C#.

Per lo sviluppo dell'applicazione si è scelto di utilizzare l'ambiente integrato Visual Studio .NET all'interno del sistema operativo Windows XP Professional dotato di Internet Information Service 5.0 come Web server per testare le pagine.

La scrittura del vocabolario RDF e delle nuove definizioni di tipi di dato consiste nella scrittura di codice XML e non richiede particolari ambienti di sviluppo (basta un editor di testo).

# Capitolo 2
# IL MODELLO RELAZIONALE

Nell' ambito dello sviluppo di applicazioni basate su database, il progetto concettuale dello schema riveste una importanza particolarmente vitale. Riguardo le tecniche di sviluppo e progettazione di un database, bisogna dire che la loro applicazione deve essere preceduta da una attenta analisi della realtà che si vuole modellare. Il modello relazionale è il modello logico più diffuso. I dati sono rappresentati in forma tabellare, cioè usa come unica struttura le relazioni o tabelle. Poi occorrono degli strumenti efficaci, chiari e sintetici per rappresentare i dati di interesse e le loro associazioni. È necessaria la scelta di un modello concettuale rappresentativo che consenta di modellare la realtà tramite un diagramma, come il modello rappresentativo ER (Entity/Relationships).

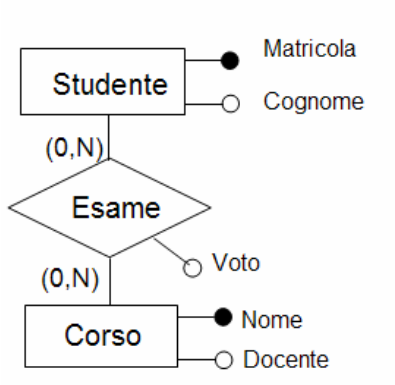
## 2.1 Il Modello ER

Il modello ER fu proposto da Chen 1976. Esso è al contempo molto semplice e consente una rappresentazione espressiva delle realtà in esame. Inoltre, tramite un metodo assolutamente algoritmico è possibile passare con semplici passaggi da questo modello a quello relazionale.
Il modello ER è costituito soltanto da tre elementi: *entità*, *relazioni* ed *attributi*. Questi elementi vengono disposti su un diagramma seguendo questo criterio: le entità corrispondono ai nomi e vengono raffigurate con dei rettangoli, le relazioni corrispondono ai verbi e si raffigurano con dei rombi (o con degli ellissi). I rettangoli ed i rombi vengono collegati con delle linee.
Vediamo ad esempio come vengono tradotte in uno schema ER le seguenti locuzioni:

- Uno studente ha la matricola univoca ed un cognome
- Un corso ha un nome univoco ed un docente

- Uno studente sostiene un esame per un certo corso, riportando un voto

  ⇒l'esame è un'associazione tra uno studente ed un corso
- Uno studente può sostenere più esami, ma non per lo stesso corso




Il metodo sotteso dall' esempio precedente è semplice da capire e funzionale. Passiamo ora in rassegna le caratteristiche essenziali degli elementi del diagramma ER: le entità sono gli elementi del modo reale, come gli studenti o i corsi. D' altra parte ogni entità è costituita da una serie di attributi. Ad esempio gli studenti hanno gli attributi di Cognome e Matricola, mentre i corsi hanno come attributi il Nome e il Docente. Una relazione, invece, è un legame di tipo logico che coinvolge due o più entità. Le relazioni hanno un unico attributo esplicito, la cardinalità. In generale la cardinalità indica il numero di entità dell' insieme A che può essere associato ad ogni elemento di un insieme B.

Per maggiori informazioni sulla gestione dello stato delle pagine ASP.NET vedi bibliografia [2].

# Capitolo 3
# IL METADATO

## 3.1 Metadati: alla ricerca del significato

Il termine metadato insieme a tutta la rete di concetti correlati, si presenta in realtà con sfaccettature semantiche molto numerose. Noto e usato già nell'ambito bibliotecario espande il suo raggio di valenza semantica alle risorse web.
Un metadato è un'informazione a proposito di un dato, più specificatamente un'informazione descrittiva fornita attraverso una codifica a tag all'interno di un documento HTML o XML. In generale si potrebbe quindi dire che i metadati sono informazioni riguardanti qualcos'altro, che di volta in volta è l'oggetto principale del nostro interesse all'interno di un contesto ben definito.
Per questi motivi non si vuole dare qui una definizione ultimativa del termine "metadati" poiché si crede che la forte dipendenza dal contesto porterebbe a perdere di vista quello che si ritiene essere il vero nocciolo della questione: l'informazione e il suo significato. Si sposterà quindi il fuoco dell'attenzione dal semplice termine verso ciò che a livello sostanziale ne costituisce l'origine e la ragion d'essere, cioè *l'informazione* e la sua formalizzazione in modelli (cioè entità, relazioni e procedure dettate da regole di business ben identificate) relativi alla conoscenza dei processi e degli oggetti propri di un dominio. I metadati assumono quindi la veste di forma codificata dell'informazione tipica di un dominio, completando ciò cui si riferiscono, arricchendolo e spesso permettendone una gestione efficiente, e diventando a volte elementi indispensabili per la corretta fruizione.
Ma occorre tener presente che i metadati in sé sarebbero privi di significato se non si considerasse l'intreccio di relazioni che li lega tra loro e alle entità cui si riferiscono e il contesto stesso in cui i metadati sono definiti.

## 3.2 Ricerca di uno Standard per i metadati

Il primo standard di metadati è stato sviluppato da una Task Force di utilizzatori, cioè il Dublin Core Metadata Element Set (DCMES). Ricordiamo che DCMES stabilisce un vocabolario semantico per descrivere informazioni sulle caratteristiche "core" di un oggetto web e categorizzarlo ai fini di una ricerca semplificata da parte dell'utente. Fin dall'inizio però il consenso nella comunità del DC ritenne utile il suo utilizzo nell'ambito sia dell'oggetto digitale che di quello reale.
In questi ultimi tempi infatti l'estensione e lo sviluppo di alcuni metadati ha portato la loro applicabilità non solo alle risorse elettroniche o digitali ma anche alla descrizione di un qualsiasi tipo di risorsa quale ad esempio oggetti di un museo, di una biblioteca, di un archivio, etc. In alcune situazioni specifiche la risorsa può essere una collezione di oggetti, o una parte di essi: fra le tipologie degli elementi vi possono essere immagini, suoni, servizi, eventi, collezioni, testi, etc. Il metadato può descrivere il modello di un processo o di un evento., persone e ruoli, etc.
Il processo di standardizzazione dei metadati è iniziato all'interno di domini specifici. Un modello generale RDF (Resource Description Framework) è stato proposto con una implementazione in linguaggio XML (eXtended Markup Language).

## 3.3 RDF per la rappresentazione della conoscenza

Uno degli obiettivi a lungo termine del World Wide Web Consortium, che definisce regole e strategie di evoluzione del Web, è il Semantic Web. Un’esigenza fondamentale per il raggiungimento di questo obiettivo è la rappresentazione della conoscenza in maniera comprensibile alla macchina. In questo lavoro verranno illustrati il ruolo dei metadati e le caratteristiche fondamentali di RDF.

## 3.4 Il World Wide Web Consortium

Il World Wide Web Consortium (W3C, http://www.w3.org), è un consorzio che sviluppa tecnologie (specifiche, linee guida, software, e strumenti) per portare il Web al massimo del suo potenziale, definendo protocolli comuni che ne favoriscano l’*evoluzione* e

assicurino l' *interoperabilità*. Gli *obiettivi a lungo termine* del W3C sono coerenti con le motivazioni iniziali che hanno portato alla nascita del web. Essi possono essere espressi sinteticamente come:

- *Universal Access*: Rendere il Web accessibile a tutti, promuovendo tecnologie che tengono conto delle notevoli differenze in termini di cultura, formazione, capacità, risorse materiali, e limitazioni fisiche degli utenti in tutti i continenti.
- *Semantic Web*: Sviluppare un ambiente software che consenta ad ogni utente di fare il miglior uso possibile delle risorse disponibili sul Web.
- *Web of Trust*: guidare lo sviluppo del Web tenendo in attenta considerazione gli aspetti innovativi che questa tecnologia solleva in campo legale, commerciale e sociale.

Nel seguito, vedremo come la rappresentazione della conoscenza costituisca un elemento fondamentale per il raggiungimento di questi obiettivi, e descriveremo in particolare il ruolo dei metadati e la loro descrizione mediante RDF.

## 3.5 Utilizzo dei metadati

Nel navigare sul web, si seguono dei link, che portano a quella che formalmente viene detta *risorsa* (*resource*) identificata univocamente da un URI (*Uniform Resource Identifier* è il generico insieme di tutti i nomi/indirizzi che costituiscono le brevi sequenze di caratteri che fanno riferimento ad una risorsa). Nel linguaggio corrente una risorsa viene anche detta "*documento*", per mettere in evidenza il fatto che sia leggibile da un essere umano, o "*oggetto*", per mettere in evidenza che è leggibile da una macchina. Qualunque sia il termine utilizzato, la risorsa non è una entità a sé, ma è accompagnata da informazioni che la descrivono. Le informazioni sulla risorsa vengono generalmente dette *Metadati*. Si può quindi dire che *i metadati sono informazioni, comprensibili dalla macchina, relative a una risorsa web o a qualche altra cosa*. Il punto chiave è costituito appunto dal fatto che i metadati sono comprensibili dalla macchina (*machine understandable*). Di conseguenza, i metadati costituiscono un tipo di informazione che può essere utilizzata dai *software agent*, per fare un uso appropriato delle risorse, rendendo più semplice e veloce il funzionamento del Web, aumentando la nostra fiducia in esso. A titolo di esempio, quando si reperisce un documento (o un oggetto) sul web, utilizzando il protocollo HTTP, è possibile che il server invii alcune informazioni sulla risorsa, quali la sua data di

aggiornamento, la data massima di validità dell' informazione, il suo autore, etc. Quindi il Web, come insieme di risorse e di informazioni sulle risorse (cioè metadati) è già una realtà alla quale siamo abituati. Va tenuto presente che *i metadati sono dati*, e questo fatto ha alcune conseguenze. La prima è che i metadati possono essere memorizzati come dati, in una risorsa, che può quindi contenere informazioni relative a se stessa o ad un' altra risorsa.

Attualmente esistono tre modi per acquisire i metadati:

1. i metadati sono contenuti nella risorsa medesima, come per esempio nella sezione HEAD di un documento HTML;
2. al momento del trasferimento della risorsa, le informazioni vengono trasferite dal server al client (GET) o dal client al server (PUT o POST);
3. i metadati vengono estratti da un' altra risorsa; quindi: *i metadati relativi ad un documento possono essere estratti dalla risorsa stessa, o da un' altra risorsa, o possono essere trasferiti con il documento*.

La seconda conseguenza del fatto che i metadati sono dati, è che *i metadati possono essere descritti da altri metadati*, e così via.

I metadati consistono in *asserzioni* sui dati, le quali vengono quindi rappresentate sotto forma di un *nome di asserzione* e un insieme di *parametri*. L' assioma è che *i metadati sono rappresentati da un insieme di asserzioni indipendenti*. Nel caso vi siano più asserzioni relative alla stessa risorsa, l' asserzione risultante è quella ottenuta mediante AND logico delle due asserzioni.

Le asserzioni relative alle risorse vengono spesso riferite come attributi della risorsa. Per esempio, se per una risorsa vale l'asserzione che essa ha una proprietà specifica come ad esempio autore, il parametro è il nome dell' autore. Analogamente, se il soggetto è un attributo della risorsa, allora il parametro è l'argomento trattato.

Gruppi di asserzioni relative alla stessa risorsa prendono spesso la forma di una lista di coppie (attributo-valore). Il termine asserzione enfatizza il fatto che la coppia (attributo-valore), quando viene trasferita con la risorsa, è uno statement fatto da una terza parte. Questo aspetto assume una particolare rilevanza nel contesto in cui il "Web of trust" diventa un elemento importante, in quanto ci permette di conoscere l'origine e l' affidabilità dei dati e dei metadati.

## 3.6 Il Resource Description Framework

Abbiamo già sottolineato come fosse difficile automatizzare il Web restando ancorati alla sua architettura originaria, in cui tutte le informazioni erano *machine-readable*, ma non *machine-understandable*, e come la soluzione al problema sembri venire dai *metadati*.
L' uso efficace dei metadati, tuttavia, richiede che vengano stabilite delle convenzioni per la *semantica*, la *sintassi* e la *struttura*. Le singole comunità interessate alla descrizione delle loro risorse specifiche definiscono la semantica dei metadati pertinenti alle loro esigenze. La sintassi, cioè l'organizzazione sistematica dei data element per la elaborazione automatica, facilita lo scambio e l' utilizzo dei metadati tra applicazioni diverse. La struttura può essere vista come un vincolo formale sulla sintassi, per una rappresentazione consistente della semantica.
RDF (Resource Description Framework) è lo strumento base per la codifica, lo scambio e il riutilizzo di metadati strutturati e consente l'interoperabilità tra applicazioni che si scambiano sul Web informazioni machine-understandable. I settori nei quali RDF può essere utilizzato e portare vantaggi sono i più disparati, basti citare, a titolo di esempio:

- descrizione del contenuto di un sito Web, o di una pagina, o di una biblioteca digitale;
- implementazione di *intelligent software agent*, per lo scambio di conoscenza e un utilizzo migliore delle risorse Web;
- *classificazione* del contenuto, per applicare criteri di selezione;
- descrizione di un *insieme di pagine*, che rappresentano un singolo documento logico;
- stabilire i criteri di *proprietà intellettuale* delle singole pagine;
- esprimere criteri di *privacy preference* degli utenti e le *privacy policies* di un sito Web;
- con il meccanismo della *digital signature*, contribuire alla creazione del Web of Trust, per le applicazioni nel commercio elettronico, la cooperazione, etc..

Il **Resource Description Framework** (RDF), quindi, non descrive la semantica, ma fornisce una base comune per poterla esprimere, permetendo di definire la semantica dei tag XML. RDF è costituito da due componenti:

- **RDF Model and Syntax**: definisce il *data model* RDF e la sua codifica XML;
- **RDF Schema**: permette di definire specifici *vocabolari* per i metadati.

## 3.7 RDF Data Model

RDF fornisce un modello per descrivere le risorse. Come abbiamo visto precedentemente, le risorse hanno delle proprietà (o anche attributi o caratteristiche). RDF definisce una risorsa come un qualsiasi oggetto che sia identificabile univocamente mediante un Uniform Resource Identifier (URI).

Il data model RDF è molto semplice, ed è basato su tre tipi di oggetti:

*Resource:* Qualunque cosa descritta da una espressione RDF viene detta risorsa (*resource*). Una risorsa può essere una pagina Web, o una sua parte, o un elemento XML all'interno del documento sorgente. Una risorsa può anche essere un'intera collezione di pagine web, o anche un oggetto non direttamente accessibile via Web (per es. un libro, un dipinto, etc.). Le risorse sono sempre individuate da un URI, eventualmente con un anchor id.

*Property:* Una *property* (proprietà) è un aspetto specifico, una caratteristica, un attributo, o una relazione utilizzata per descrivere una risorsa. Ogni proprietà ha un significato specifico, definisce i valori ammissibili, i tipi di risorse che può descrivere, e le sue relazioni con altre proprietà. Le proprietà associate alle risorse sono identificate da un *nome*, e assumono dei *valori*.

*Statement:* Una risorsa, con una proprietà distinta da un nome, e un valore della proprietà per la specifica risorsa, costituisce un RDF *statement*. Uno statement è quindi una tupla composta da un *soggetto* (risorsa), un *predicato* (proprietà) e un *oggetto* (valore). L' oggetto di uno statement (cioè il property value) può essere un' espressione (sequenza di caratteri o qualche altro tipo primitivo definito da XML) oppure un' altra risorsa.

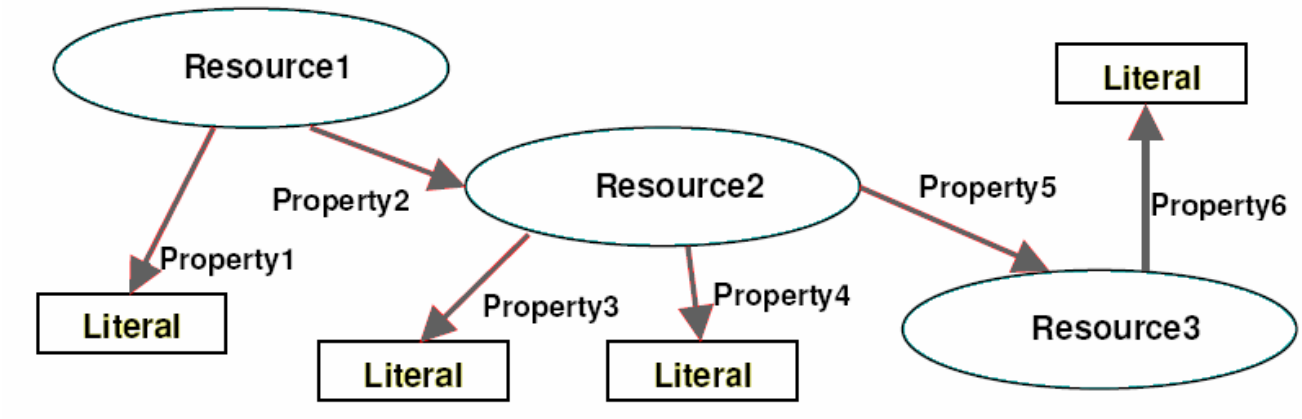



**Fig. 1.** Il modello generico di una descrizione RDF

Graficamente, le relazioni tra Resource, Property e Value vengono rappresentate mediante *grafi etichettati orientati*, in cui le risorse vengono identificate come nodi (graficamente delle ellissi), le proprietà come archi orientati etichettati, e i valori corrispondenti a sequenze di caratteri come rettangoli. Una rappresentazione grafica di una generica descrizione RDF è quella della figura 1.
Un insieme di proprietà che fanno riferimento alla stessa risorsa viene detto descrizione (*description*).

## 3.8 Utilizzo di RDF

L' utilizzo di RDF può essere chiarito con qualche semplice esempio.
Consideriamo queste due espressioni:
1. "*Mario Rossi è l' autore del DocumentoX*"
2. "*L' autore del DocumentoX è Mario Rossi*"
Ovviamente, le due espressioni sono del tutto equivalenti per un essere umano, in quanto veicolano la stessa informazione, mentre verrebbero viste come due espressioni diverse da una macchina. RDF, mediante il suo semplice modello basato su resource, property e value, intende fornire un metodo non ambiguo per esprimere la semantica con una codifica comprensibile dalla macchina.
Dato che RDF fornisce un meccanismo per associare le proprietà alle risorse, per poter dare un valore alle proprietà occorre che sia dichiarata la risorsa. Quindi, per prima cosa va dichiarata una risorsa che rappresenti il DocumentoX. Abbiamo quindi la tripla:

*Resource* http://www.w3c.it/Mario/DocX
*Property* author
*Value* Mario Rossi

Lo statement dell' esempio verrebbe quindi rappresentato come:

*La risorsa http://www.w3c.it/Mario/DocX has Author Mario Rossi*

E verrebbe rappresentata graficamente come in Figura 2.

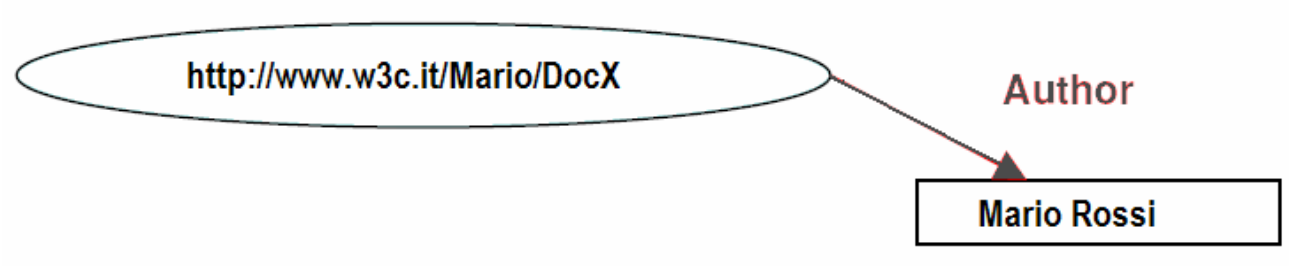



**Fig. 2.** La rappresentazione grafica di uno statement RDF

La direzione dell' arco è importante: essa parte dal soggetto dello statement e punta all'oggetto.

Se invece vogliamo dire qualcosa in più riguardo all' oggetto, per esempio vogliamo dire:

*Mario Rossi*, la cui Email è *Mario@w3.org*, e lavora presso il *C.N.R*,

è l' *autore* del *DocumentoX*

desideriamo esprimere delle informazioni *riguardo a* (*about*) Mario Rossi, e quindi trasformare il valore della proprietà autore in una entità strutturata. In RDF, un' entità di questo tipo viene rappresentata come un' altra risorsa. Poiché nello statement non viene attribuito un nome a questa entità strutturata, il diagramma risultante sarebbe quello di Figura 3.

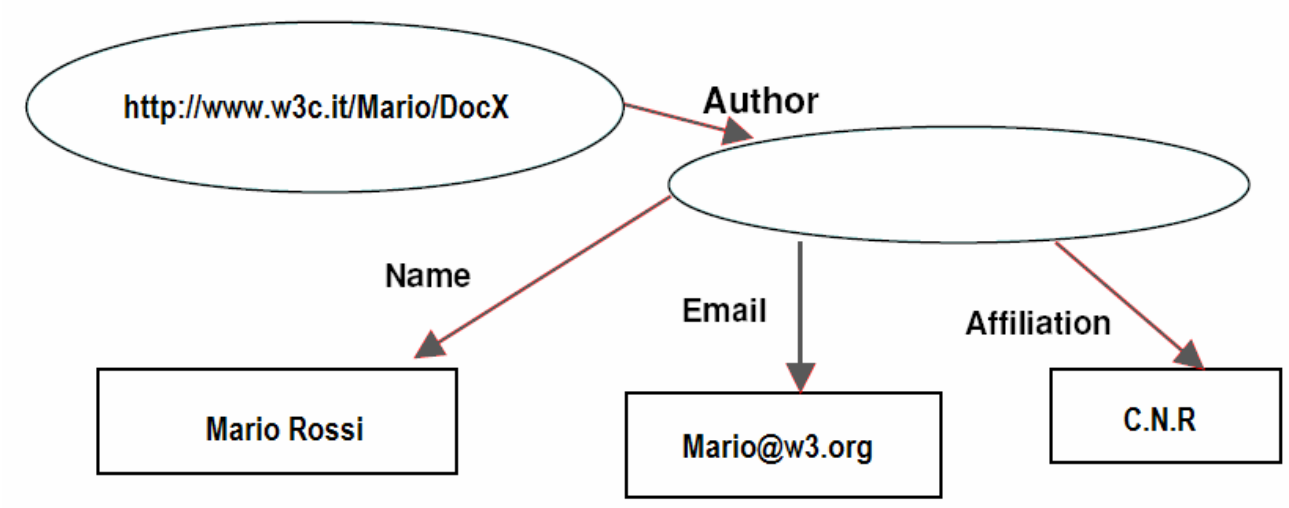



**Fig. 3.** Una proprietà con un valore strutturato

Si noti che questo diagramma potrebbe essere letto come:

*http://www.w3c.it/Mario/DocX has author* qualcuno *e* questo qualcuno

*has Name Mario Rossi*, *Email Mario@w3.org, e Affiliation C.N.R*.

Alla risorsa anonima " qualcuno" potrebbe essere assegnato un identificatore univoco, per esempio il CodiceFiscale, corrispondente a un URI del tipo:

http://www.finanze.it/CF/RSSMRA99A99X111Y.

Il diagramma risultante sarebbe quello di Figura 4.

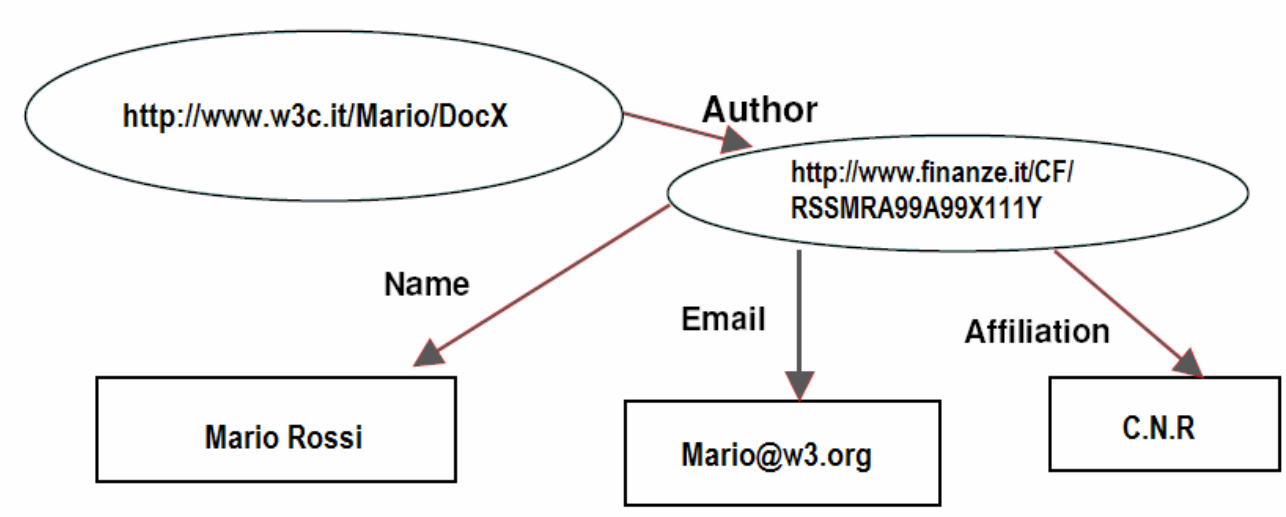



**Fig. 4.** Un valore strutturato con identificatore

Questo diagramma potrebbe essere letto come la concatenazione di due frasi:

*La persona identificata dal Codice Fiscale RSSMRA99A99X111Y has Name Mario Rossi, Email Mario@w3.org, e Affiliation C.N.R..*

*La risorsa http://www.w3c.it/Mario/DocX has author questa persona*

In questo esempio è stata creata una risorsa, identificabile univocamente, per l' autore, ma non per il nome, la Email, l' affiliazione. Il modello RDF consente la creazione di risorse a più livelli. Per esempio, sarebbe stato possibile creare una risorsa per l'Affiliazione, con proprietà come: tipoDiEnte, partita IVA, sedeSociale, etc. Quali siano i limiti pratici e logici per il numero di livelli, dipende essenzialmente dalle caratteristiche e dalle tradizioni delle singole comunità che definiscono la descrizione delle risorse.

## 3.9 I Container

Talvolta, è necessario far riferimento a più di una risorsa, per esempio per descrivere il fatto che un libro è stato scritto da più autori, oppure che un documento è composto da una serie di componenti, oppure che una funzione può essere svolta da una delle persone elencate. RDF definisce tre tipi di contenitori (*container*):

*Bag* È una lista non ordinata di risorse o costanti. Viene utilizzato per dichiarare che una proprietà ha valori multipli, senza alcun significato particolare attribuito al loro ordine (per esempio, i componenti di una commissione). Sono ammessi valori duplicati.

*Sequenze* È una lista ordinata di risorse o costanti. Viene utilizzato per dichiarare che una proprietà ha valori multipli, e che il loro ordine è significativo (per esempio, gli autori di un libro, un insieme di nomi di cui si voglia preservare l' ordine alfabetico). Sono ammessi valori duplicati.

*Alternative* È una lista di risorse o costanti che rappresentano una alternativa per il valore (singolo) di una proprietà. Può essere utilizzato, per esempio, per fornire titoli alternativi in varie lingue.

È possibile definire proprietà sia dell' intero container che dei singoli elementi.

## 3.10 Container e proprietà multiple

Una risorsa può essere soggetto in più statement, sempre con lo stesso predicato (per esempio, Calvino è autore di "Se una notte d' inverno un viaggiatore", "Le fiabe italiane", "Il barone rampante"). Si noti che è semanticamente diverso il caso in cui si ha un singolo statement il cui oggetto è un container contenente vari esemplari. Per esempio, lo statement: "La commissione composta da X, Y e Z ha adottato una decisione", non implica che ogni membro della commissione abbia espresso lo stesso
parere, come invece sarebbe nel caso in cui si usasse uno statement multiplo.

## 3.11 Statement about statement

Gli statement riguardano le risorse. In alcuni casi, è utile poter certificare l’affidabilita` di un particolare statement, quindi formulare degli statement relativi ad altri statement. Per esempio:

*La risorsa http://www.w3c.it/Mario/DocX has Author Mario Rossi*

Viene vista da RDF come un fatto. Invece, lo statement:

*Paolo Bianchi dice che la risorsa http://www.w3c.it/Mario/DocX has Author Mario Rossi*

Non afferma un fatto relativo alla risorsa *http://www.w3c.it/Mario/DocX*, ma un fatto relativo all' affermazione di Paolo Bianchi. Per esprimere questo fatto a RDF, dobbiamo modellare lo statement come una risorsa con quattro proprietà. Nell' ambito della comunità che si interessa di Knowledge Representation, questo processo prende il nome di *reificazione* (*reification*), e il modello di statement prende il nome di *reified statement*.

Per modellare gli statement, RDF definisce queste quattro proprietà sono:

*subject* identifica la risorsa che viene descritta dallo statement modellato, quindi il soggetto è la risorsa relativamente alla quale era stato formulato lo statement originale (http://www.w3c.it/Mario/DocX, nell'esempio).

*predicate* identifica la proprietà originale nello statement modellato. Il valore del predicato è una risorsa che rappresenta la specifica proprietà nello statement originale (nel nostro esempio, Author).

*object* identifica il valore della proprietà nello statement modellato. Il valore di object è l' object nello statement originale (nel nostro esempio: "Mario Signore")

*type* descrive il tipo della nuova risorsa

Una nuova risorsa con queste quattro proprietà rappresenta lo statement originale, e può essere utilizzata come object di altri statement e avere ulteriori statement che lo riguardano.

## 3.12 I Namespace

RDF consente alle singole comunità di definire la semantica. Tuttavia, non è possibile affidare la semantica semplicemente al nome, che potrebbe avere significati più o meno ampi a seconda degli interessi specifici delle singole comunità. RDF identifica univocamente le proprietà mediante il meccanismo dei namespace. ([XMLns]). I namespace XML forniscono un metodo per identificare in maniera non ambigua la semantica e le convenzioni che regolano l'utilizzo delle proprietà identificando l'authority che gestisce il vocabolario.

Uno degli esempi più noti è la Dublin Core Initiative ([DC], [DCES], [DCRDF]) che definisce, per esempio, il campo " *Subject and Keywords"* nel seguente modo:

```
Name: Subject and Keywords
Identifier: Subject
Definition: The topic of the content of the resource.
Comment: Typically, a Subject will be expressed as keywords,
key phrases or classification codes that describe a
topic of the resource.
Recommended best practice is to select a value from a
controlled vocabulary or formal classification
scheme.
```

Si può quindi utilizzare un namespace XML per identificare in maniera non ambigua lo schema per il vocabolario Dublin Core puntando alla risorsa Dublin Core che ne definisce la semantica.

Quindi, la descrizione di un sito Web mediante le proprietà definite nel vocabolario Dublin Core e quelle di una personale estensione potrebbe essere:

```
<rdf:RDF
xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
xmlns:dc="http://purl.org/metadata/dublin_core#"
xmlns:mydc="http://www.w3c.it/metadata/DCaddendum#">
<rdf:Description about="http://www.dlib.org">
<dc:Title>
D-Lib Program - Research in Digital Libraries
</dc:Title>
<dc:Description>
The D-Lib program supports the community of people
with research interests in digital libraries and
electronic publishing
.</dc:Description>
<dc:Publisher>
Corporation For National Research Initiatives
</dc:Publisher>
<dc:Date>1995-01-07</dc:Date>
<dc:Subject>
<rdf:Bag>
<rdf:li>Research; statistical methods</rdf:li>
<rdf:li>Education, research, related topics</rdf:li>
<rdf:li>Library use Studies</rdf:li>
</rdf:Bag>
</dc:Subject>
<dc:Type>World Wide Web Home Page</dc:Type>
<dc:Format>text/html</dc:Format>
<dc:Language>en</dc:Language>
<mydc:Rating>
Well known and often referenced site
</mydc:Rating>
<mydc:Originality>High</mydc:Originality>
</rdf:Description>
</rdf:RDF>
```

Si noti in questo esempio, che costituisce una variante di uno di quelli presentati in [RDFMSS], la presenza di tre *namespace*, referenziati dai prefissi **rdf**, **dc** e **mydc** che permettono di utilizzare le proprietà definite nei tre namespace. In particolare, il namespace mydc permette di ampliare il numero di proprietà definite dal namespace che referenzia Dublin Core.

-

## 3.13 RDF Schema

Il data model RDF permette di definire un modello semplice per descrivere le relazioni tra le risorse, in termini di proprietà identificate da un nome e relativi valori. Tuttavia, RDF data model non fornisce nessun meccanismo per dichiarare queste proprietà, né per definire le relazioni tra queste proprietà ed altre risorse. RDF Schema permette definire significato, caratteristiche e relazioni di un insieme di proprietà, compresi eventuali vincoli sul dominio e sui valori delle singole proprietà. Inoltre, implementando il concetto (transitivo) di classe e sottoclasse, consente di definire gerarchie di classi, con il conseguente vantaggio che agenti software intelligenti possono utilizzare queste relazioni per svolgere i loro compiti.

Per maggiori informazioni sulla gestione dello stato delle pagine ASP.NET vedi bibliografia [6].

# Capitolo4

# ASP.NET

Nel presente capitolo viene presentata la tecnologia utilizzata per la realizzazione della tesi. In particolare viene introdotto ASP.NET che è stato utilizzato per scrivere le pagine Web dinamiche del progetto. Per la scrittura del codice delle pagine è stato scelto il linguaggio C#.

.NET è un ambiente di sviluppo software ed esecuzione d'applicazioni che consente di creare, compilare, provare, distribuire ed eseguire software che può essere codificato in diversi linguaggi che si attengono tutti ad un singolo insieme di file Common Language Runtime. Questo significa consentire allo sviluppatore di adottare un qualunque linguaggio di programmazione conforme alle regole ferree del framework .NET e poter far interagire applicazioni scritte in linguaggi differenti.

## 4.1 Il framework .NET

Da un punto di vista di alto livello, il framework .NET può essere descritto come un sistema operativo virtuale eseguito sopra il sistema operativo vero e proprio. Un esame più preciso rivela che il framework è formato da due componenti principali: il CLR (Common Language Runtime) la Class Library .NET.

## 4.2 Orientamento alle classi del framework .NET

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 ad oggetti (ES: System.Data.SQLClient). Questi riferimenti espliciti a gruppi di classi nelle librerie del framework sono definiti spazi dei nomi in .NET. Uno

spazio dei nomi è un riferimento gerarchico univoco ad una classe specificata o ad un gruppo di classi simili.

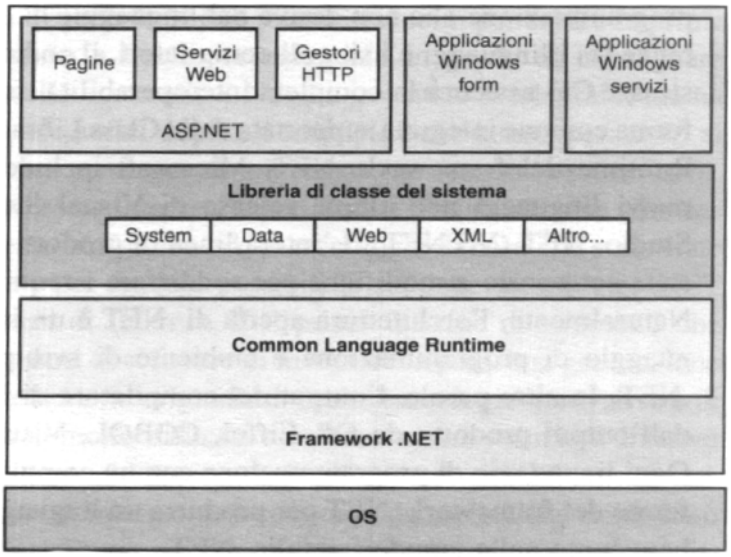



**Figura 1 - ASP.NET – Gerarchia**

## 4.3 Common language runtime

La responsabilità per attività quali la creazione di oggetti, l'esecuzione di chiamate ai metodi e così via, è denominata Common Language Runtime che consente a runtime di fornire servizi aggiuntivi al codice in esecuzione.

Il CLR riveste due ruoli diversi, uno in fase di sviluppo e uno in fase di progettazione.

- In fase di sviluppo, il CLR include un sistema comune di tipi (che consente l'integrazione tra linguaggi diversi), un miglior controllo dei conflitti di versione e servizi di sicurezza oltre la gestione delle eccezioni tra linguaggi, la gestione degli eventi basati sui delegate, il binding dinamico e la reflection che riducono considerevolmente la quantità di codice che occorre scrivere per trasformare la logica di business in componenti riutilizzabili.
- In fase di esecuzione, il CLR fornisce servizi quali la gestione della memoria (incluso il garbage collection), la gestione dei processi e dei thread e il potenziamento della sicurezza.

## 4.4 Intermediate language (IL) e metadati

IL (Intermediate Language) è la versione .NET del codice compilato. Qualsiasi linguaggio di programmazione si scelga per la scrittura di un programma, il risultato di una compilazione sarà sempre nello 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.

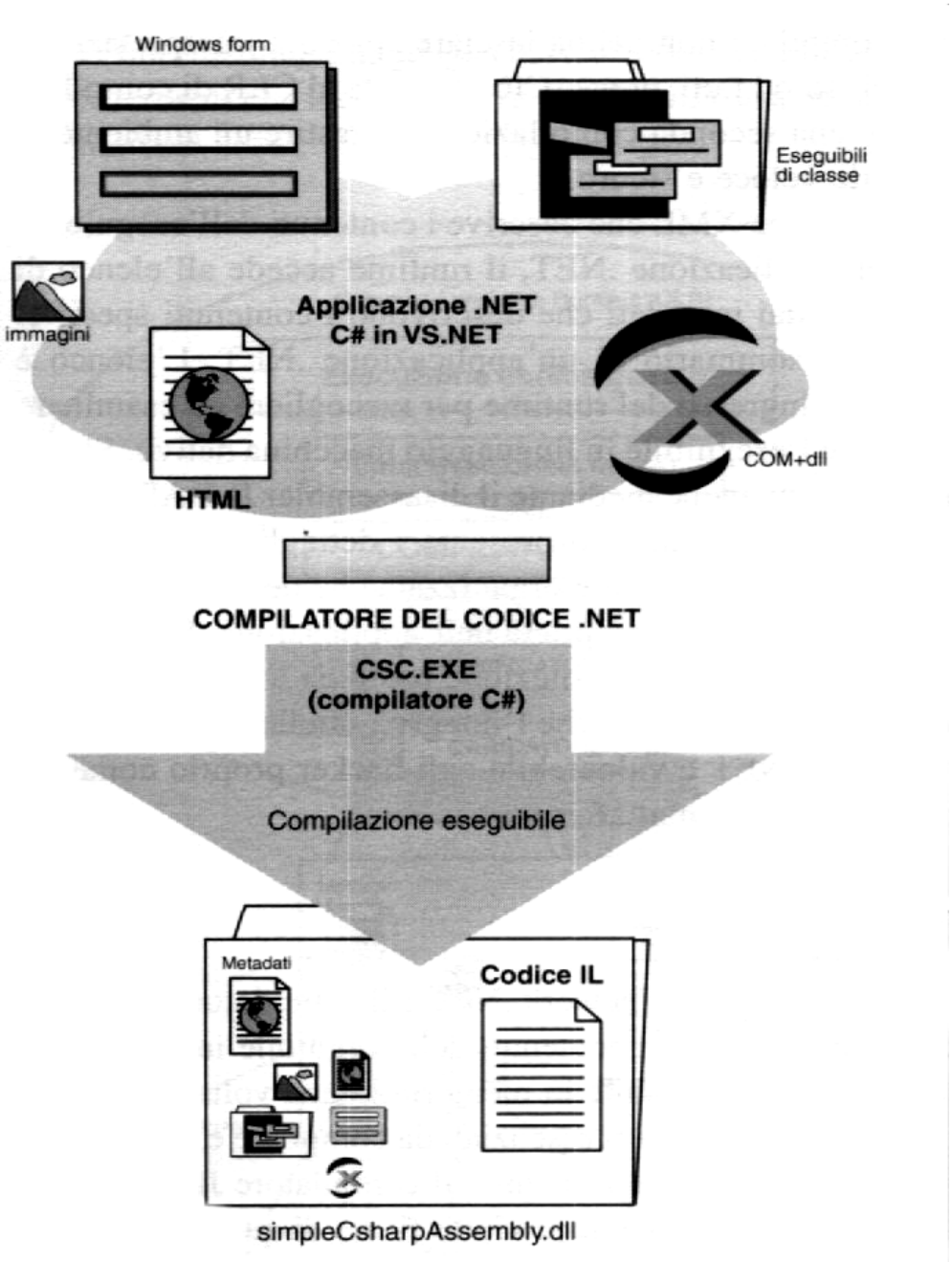



**Figura 2 - ASP.NET - Compilare (1P)**

I metadati sono rappresentati in formato XML e descrivono i contenuti dell'eseguibile. Contengono la posizione e l'obbiettivo di ogni oggetto e le sue proprietà, i suoi argomenti, i suoi delegati e i suoi metodi.

Le applicazioni .NET sono compilate Just-In-Time una seconda volta nel codice macchina nativo. Il runtime .NET offre allo sviluppatore la scelta di opzioni di compilazione o la possibilità di specificare che la compilazione venga eseguita al momento dell'istallazione piuttosto che nella fase di lancio dell'applicazione. Questa opzione diventa utile quando si desidera evitare l'overhead della compilazione al lancio dell'applicazione.

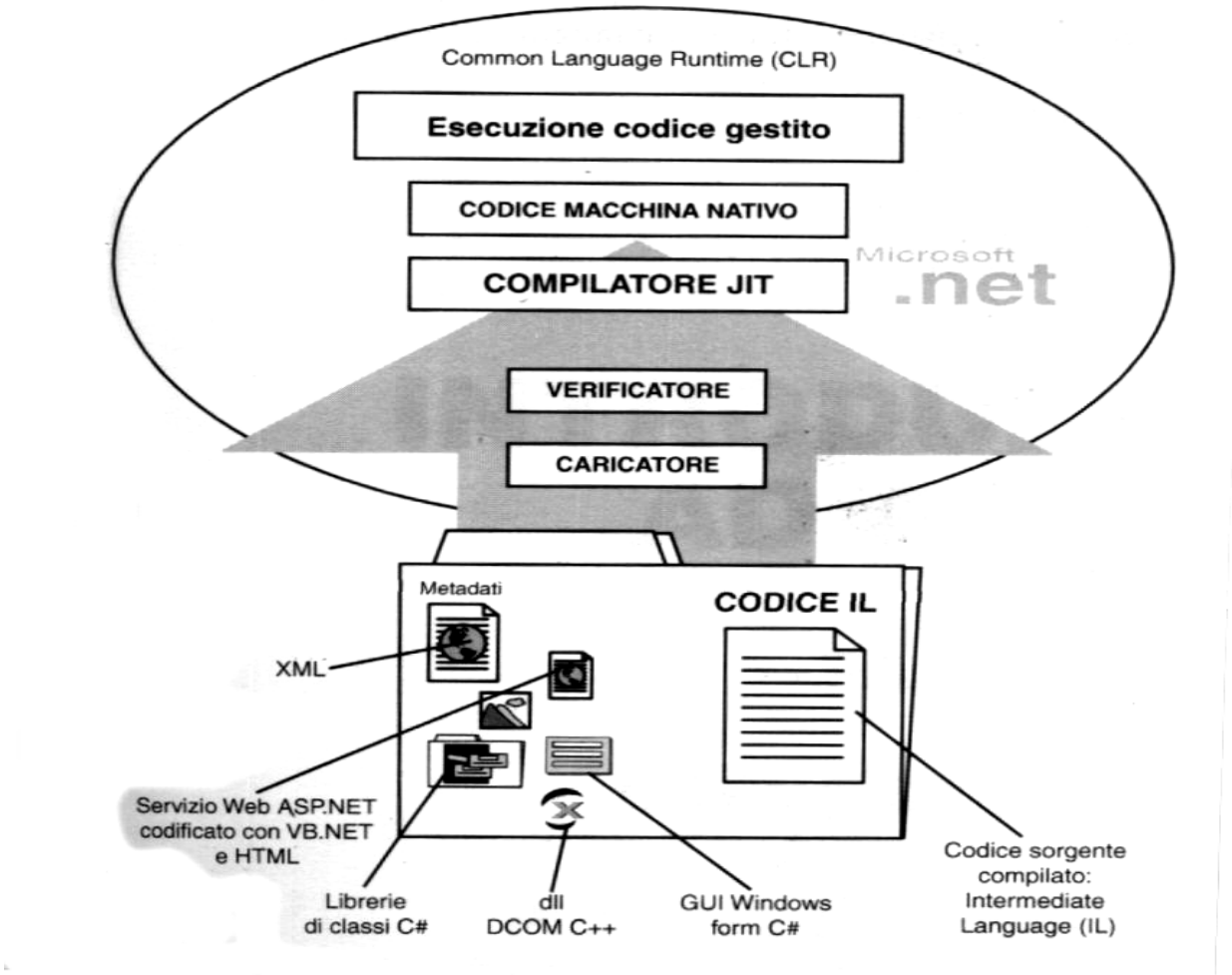



**Figura 3 - ASP.NET - Compilare (2P) e eseguire**

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 erano sufficienti ad evitare problemi qual’ora un’altra applicazione avesse aggiornato una DLL di sistema condivisa a una nuova versione. Gli sviluppatori .NET non dovranno più preoccuparsi di questi problemi.

## 4.5 Il compilatore jit

Un componente fondamentale del framework .NET è il compilatore JIT (Just-In-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 il codice per individuare le sue 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 questo approccio è un ambiente di runtime con più spazio di memoria disponibile per elaborare più velocemente gli elementi usati.

## 4.6 Breve storia di ASP:NET

Quando ASP (Active Server Pages) è stato rilasciato per la prima volta nel novembre 1996, ha introdotto un metodo semplice per creare pagine Web dinamiche. ASP si è subito affermato per quattro motivi:

- Facilità di accesso ai dati: probabilmente se ASP non fosse stato rilasciato con ADO (ActiveX Data Object) come metodo di accesso a database preferito da Microsoft non si sarebbe affermato così facilmente.
- Semplice struttura di pagine
- Interoperatività con COM (Component Object Model): 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 il via a un nuovo mercato di produttori di componenti, che continuano a offrire strumenti prefabbricati potenti e facilmente integrabili nelle applicazioni ASP.

- Facilità di apprendimento per gli sviluppatori di Visual Basic: l'utilizzo di VBScript come linguaggio predefinito ha consentito ai programmatori già esperti di Visual Basic di iniziare subito a usare ASP con il minimo studio.

Ciononostante ASP 1.0 presentava limiti significativi. Il limite principale per chi lavorava con componenti COM era che il Web server doveva essere riavviato a ogni aggiornamento di una DLL (Dynamic Link Library) dove sono memorizzati gli oggetti COM.

Il principale miglioramento da ASP 1.0 a ASP 2.0 è stato MTS (Microsoft Transaction Server). 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. Infine, agiva da intermediario di oggetti, inserendo nella cache le istanze di oggetti e restituendole a ogni richiesta.

Nel febbraio 2000, Microsoft ha rilasciato insieme a IIS 5.0, ASP 3.0. In questa nuova versione MTS venne sostituito dai servizi COM+. COM+ combinava le funzionalità di MTS con i servizi di accoramento dei messaggi.

Mark Anders e Scott Guthtie hanno iniziato a sviluppare ciò che sarebbe diventato ASP.NET nel gennaio 1998. Gli sviluppatori hanno scelto di creare ASP.NET (all'epoca chiamato ASP+) su NGWS (Next Generation Web Servives) Runtime a quell'epoca in fase di sviluppo che sarebbe diventato .NET. NGWS offriva un ricco insieme di librerie di programmazione e presto avrebbe 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 come necessario.
- Scalabilità. Sono stati fatti molti sforzi per creare un modello altamente scalabile, nel rispetto del mantenimento dello stato.
- Disponibilità. AST.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 di Visual Basic.

## 4.7 Benefici di ASP.NET rispetto ad ASP

ASP.NET e il framework .NET offrono diversi vantaggi rispetto ad ASP classico. ASP.NET è più robusto, sicuro e scalabile, offre strumenti migliori, che consentono ai programmatori di essere più produttivi e supporta diversi linguaggi, in modo da usare quello che si preferisce. ASP.NET è anche più facile da gestire e distribuire. Di seguito sono illustrati i vantaggi ASP.NET rispetto a ASP.

- ASP.NET è compilato, non interpretato. I programmi compilati sono eseguiti più velocemente di quelli interpretati. Ogni pagina è compilata alla prima richiesta; il codice compilato viene conservato fino a quando non si modifica la pagina o si riavvia l'applicazione.
- Separazione del codice che implementa la logica di business dal codice per la presentazione delle informazioni. 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.
- Fine del "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+.
- Installazione lato per lato. I nuovi servizi e funzioni possono essere installati ed eseguiti in parallelo con applicazioni ASP classiche esistenti. Condividono infatti la stessa struttura di cartelle; per migrare ogni file, dopo averlo impostato per usare le nuove funzioni di ASP.NET, sarà sufficiente cambiare l'estensione da .asp a .aspx e aggiornare tutti i collegamenti di conseguenza. Sarà quindi possibile migrare le applicazioni una pagina alla volta.
- Debug reale. In ASP.NET, il debug è più semplice che in ASP classico. E' stato aggiunto un comando trace compilato solo nel codice in esecuzione quando si imposta un flag di compilazione. Con Visual Studio .NET si può esaminare passo per passo il codice ASP.NET, i file include, i controlli Web e i componenti .NET anche se ciascuno di essi usa un linguaggio di programmazione diverso.

- Linguaggi di programmazione reali. Anche se ASP supporta diversi linguaggi di scripting, ASP.NET e il framework .NET supportano qualsiasi linguaggio che può essere compilato in formato IL.
- Gestione degli errori reale. ASP.NET offre una migliore gestione degli errori, tra cui l'abilità di inviare gli errori di programmazione a una sola pagina di gestione degli errori, trasferendo anche tutti gli attributi della pagina. Disporre di una locazione centrale per la gestione degli errori evita di controllare gli errori su ogni riga di codice e scrivere un gestore di errori personalizzato per ogni caso. Inoltre Visual Basic .NET supporta ora la struttura try … catch.
- Distribuzione basata sulle directory. Migrare un'applicazione ASP da un server all'altro è un'operazione complicata. Le estensioni di FrontPage, i componenti COM e le impostazioni del Web server sono separati dai file nella directory da spostare. Con ASP.NET, si può distribuire l'applicazione completa di componenti e impostazioni del server, usando XCOPY o FTP. Questo semplifica il beckup dei siti e elimina molti dei problemi relativi all'hosting Web remoto.
- Configurazione di applicazioni basata su file. L'amministrazione dell'applicazione può avvenire interamente mediante file di configurazione XML senza necessità di alcun accesso diretto al server.
- Modello di programmazione basato sugli eventi. La 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 su eventi.
- Modello ad oggetti migliorato ed estensibile. I Web form possono essere creati con di una interfaccia drag-and-drop semplificando la creazione di pagine guidate agli eventi. Inoltre, il framework .NET include un elenco completo di classi disponibili per gli sviluppatori non incluse negli oggetti di ASP 3.0.
- Più funzioni integrate. Non è più necessario accedere esplicitamente alle variabili modulo usando l'oggetto request. Tutto questo è gestito automaticamente da ASP.NET. E' sufficiente aggiungere runat="server" al modulo e a ogni elemento del modulo. Le funzioni di convalida dei moduli sono incorporate in ASP.NET, che viene fornito con le funzioni di convalida più comuni e consente agli sviluppatori di scrivere controlli di convalida personalizzati.

- Servizi Web. ASP.NET include supporto per i servizi Web, che consentono alle applicazioni di funzionare assieme su Internet visualizzando e/o utilizzando metodi e dati di altri siti. I servizi Web usano lo standard SOAP (Simple Object Access Protocol)
- Miglioramenti nelle prestazioni. Le prestazioni di ASP.NET sono migliori di quelle di ASP classico. Il vantaggio principale è che ASP.NET è compilato, non interpretato. ASP.NET supporta il caching a livello di pagina, che può essere configurato di pagina in pagina.
- Strumenti migliori. Non esistono più strumenti separati per ogni linguaggio supportato da Microsoft. Fa tutto Visual Studio .NET
  Per maggiori informazioni su ASP.NET vedi bibliografia [4].

## 4.8 Come vengono elaborate le pagine ASP.NET

In generale, il ciclo di vita di una pagina Web con un Forms è simile a quello che ogni processo Web svolge nel server. Certe caratteristiche dei processi Web (informazioni comunicate con il protocollo HTTP, la natura senza stato delle pagine Web, ecc. ) sono applicabili alle pagine Web con Form.

Comunque, il framework delle pagine ASP.NET fornisce molti servizi per applicazioni Web. Per esempio, il framework delle pagine ASP.NET cattura le informazioni inviate con i Form delle pagine Web, estrae i valori rilevanti, e produce informazioni accessibili tramite proprietà di oggetti.

È importante capire la sequenza degli eventi scatenati quando un From di una pagina Web viene elaborato.

## 4.9 Round trips

Una delle cose più interessanti è la divisione del lavoro nei form nelle pagine Web. Il browser presenta il form all'utente, e l'utente interagisce con il form, causando dei post back al server. Tutti i processi che interagiscono con i componenti server devono essere elaborati nel server. Questo significa che per ogni azione che richiede di essere elaborata, deve essere inviata al server, elaborata e restituita al browser. Questa sequenza di eventi è chiamata round trip.

Da notare che nei form delle pagine Web si possono creare script client, che sono usati per convalidare gli input dell'utente e che possono essere elaborati senza essere inviati al server perché non richiedono l'interazione con i componenti server.

Consideriamo un esempio nel mondo del business. Un utente inserisce un ordine e conferma un numero sufficiente di prodotti per quell'ordine, quindi la sua applicazione invia la pagina a un server appropriato per elaborare l'ordine. Il processo server esamina l'ordine, scorre l'inventario, svolge alcune azioni definite dalla logica di business e restituisce la pagina al browser.

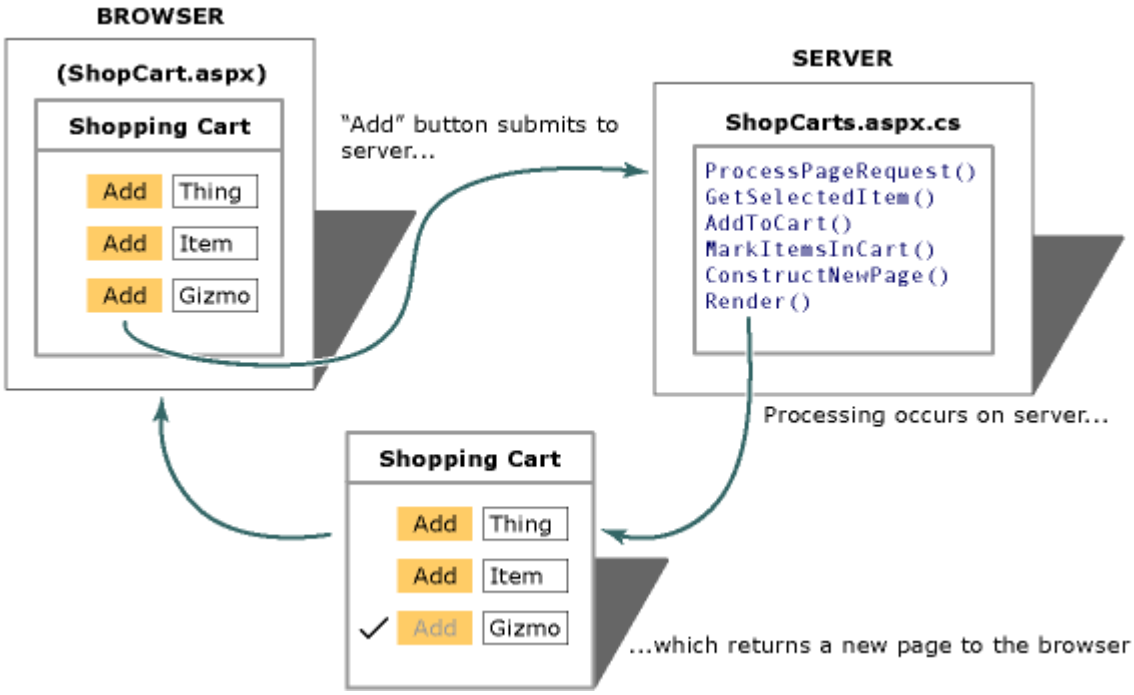



**Figura 4 - ASP.NET Round Trip**

Nei forms Web, molte azioni degli utenti, come cliccare un bottone, scatenano un round trip. Per questa ragione, gli eventi disponibili nei controlli server di ASP.NET sono solitamente limitati ai click. Molti controlli server espongono l'evento click, che richiede però una gestione da parte dell'utente.

Per la stessa ragione, i controlli server non espongono eventi usati di frequente come onmouseover, perché tutte le volte che vengono generati, viene eseguito un round trip, che ha considerevoli effetti sulla risposta del form.

## 4.10 Ricreazione delle pagine (view state e state management)

Nello scenario del Web, le pagine sono ricaricate ad ogni round trip. Non appena il server finisce di elaborare e spedire una pagina al browser, elimina le informazioni sulla

pagina, per liberare le sue risorse dato che deve supportare centinaia o migliaia di utenti contemporanei. La volta successiva che viene inviata la pagina, il server crea un nuovo processo per elaborarla, e per questa ragione la pagina Web viene chiamata senza stato (stateless) e i valori delle variabili e dei controlli della pagina non vengono preservati nel server.

Da notare che il server può essere configurato per memorizzare le informazioni delle pagine in una cache per ottimizzarne l'esecuzione, ma non allo scopo di supporto alla programmazione delle pagine.

Nelle applicazioni Web tradizionali, le sole informazioni che il server ha sul form sono le informazioni che l'utente ha inserito nei controlli del form, perché esse sono inviate al server quando il form viene spedito. Le altre informazioni, come i valori delle variabili e le proprietà settate, sono perse.

ASP.NET lavora all'interno di queste limitazioni con le seguenti modalità:

- Salva le proprietà della pagina e dei controlli ad ogni round trip. Questa tecnica è conosciuta come salvare il view state del controllo.
- Fornisce facilitazioni nella gestione dello stato per cui si possono salvare le variabili e le informazioni relative alla applicazione o alla sessione ad ogni round trip.
- Può rilevare quando un form è richiesto per la prima volta o quando è stato già inviato.

## 4.11 Benefici del modello a eventi rispetto al modello lineare

Un programmatore che ha esperienza nello scrivere pagine usando ASP, saprà che ASP utilizza un modello di elaborazione delle pagine lineare. Una pagina ASP viene elaborata in sequenza dall'alto verso il basso. Ogni linea di codice ASP e di HTML statico viene elaborata in sequenza come appare nel file. Un'azione dell'utente causa l'invio della pagina al server in un round trip. Quando questa azione causa un round trip, il server ricrea la pagina. Dopo che la pagina è stata ricreata, viene elaborata nella stessa sequenza dall'alto verso il basso come in precedenza. Per creare un metodo di elaborazione delle pagine a eventi, è necessario avere a disposizione un metodo per mantenere lo stato della pagina e dei controlli. Questo limite del modello limita la ricchezza dell'interfaccia utente che viene assemblata, e incrementa la complessità del codice necessario per gestirla.

A confronto, il modello a eventi, come quello delle tradizionali applicazioni Visual Basic, contiene degli elementi programmabili che vengono inizializzati e visualizzati nel form. L'utente interagisce con essi, causando degli eventi che vengono ascoltati da un ascoltatore degli eventi. Questo modello supporta un vero modello ad eventi, che estende la ricchezza dell'interfaccia utente che viene assemblata, e riduce la complessità del codice necessario per gestirla.

ASP.NET sostituisce il modello lineare di processare le pagine di ASP con una emulazione del modello ad eventi. Il framework delle pagine ASP.NET è fornito di un implicito produttore di associazioni tra un evento e il suo ascoltatore. Usando il framework si possono creare semplicemente interfacce utente che reagiscono alle azioni dell'utente.

In più, il framework semplifica l'implementazione della gestione dello stato della pagina e dei controlli.

Per esempio, ASP.NET permette di settare un ascoltatore degli eventi nel codice del server per un evento passato dal browser. Assumiamo che l'utente stia interagendo per un Form Web che contiene un controllo server bottone. L'utente clicca nel controllo bottone e viene generato un evento che viene trasmesso con un post HTTP al server dove il framework della pagina ASP.NET interpreta le informazioni inviate e associa l'evento scatenato con l'appropriato ascoltatore di eventi. Questo ascoltatore di eventi può essere l'ascoltatore di eventi di default fornito da ASP.NET o può essere una implementazione personalizzata. Il framework automaticamente chiama l'appropriato ascoltatore degli eventi per il bottone premuto.

## 4.12 Fasi di elaborazione di una pagina ASP.NET

Il framework delle pagine ASP.NET elabora i from Web in diversi passaggi. Durante ogni fase dell'elaborazione dei From Web, sono scatenati degli eventi, e per ogni evento scatenato viene chiamato il rispettivo ascoltatore. Questo metodo fornisce la possibilità di aggiornare il contenuto di un Form di una pagina Web.

La tabella che segue elenca i più comuni fasi di come vengono elaborate le pagine, gli eventi generati, quando vengono generati, e il loro uso tipico. Questi passaggi vengono ripetuti tutte le volte che un form viene richiesto o inviato. La proprietà Page.IsPostBack permette al programmatore di testare quando la pagina viene processata per la prima volta.

| Passaggio | Significato | Uso tipico |
|---|---|---|
| ASP.NET Page Framework Initialization | L'evento della pagina Page_Init viene scatenato e il view state della pagina e dei controlli sono ricaricati. | Durante questi evento, il framework delle pagine ASP.NET, ricarica le proprietà dei controlli e dei dati dei postback. |
| User Code Initialization | L'evento della pagina Page_Load viene scatenato | Leggere e ricaricare i valori salvati precedentemente. |
| Validation | Il metodo Validate di ogni controllo server validator viene invocato per eseguire i controlli specificati. | |
| Event Handling | Se la pagina viene chiamata in risposta a un evento, il corrispondente ascoltatore degli eventi viene chiamato durante questa fase. | Esegue le istruzioni per l'applicazione specifica. |
| Cleanup | L'evento Page_Unload viene chiamato perché la pagina è stata interpretata e le sue informazioni sono pronte per essere eliminate. | Esegue i lavori di pulizia finale come chiudere file, chiudere connessioni a database e eliminare oggetti. |

Per maggiori informazioni su come vengono elaborate le pagine ASP.NET vedi bibliografia [5].

## 4.13 Gestione dello stato delle pagine ASP.NET

Le pagine Web sono ricreate ogni volta che sono spedite al server. Nella programmazione Web tradizionale, questo significa che tutte le informazioni associate alla pagina e ai controlli contenuti nella pagina vengono perse ad ogni round trip. Per esempio, se un utente inserisce informazioni in una textbox, questa informazione viene persa nel round trip tra il browser al server.

Per superare questa limitazione intrinseca della programmazione tradizionale Web, il framework delle pagine ASP.NET include varie opzioni per aiutare il programmatore a preservare lo stato. Il framework delle pagine include una facilitazione, chiamata view state che automaticamente preserva il valore di una proprietà della pagina o di un controllo tra i round trip. Comunque, è probabile avere a che fare con specifiche applicazioni che devono preservare particolari valori. Per fare questo, si può usare una delle modalità per la gestione dello stato.

Alcune di queste opzioni comportano la memorizzazione delle informazioni nel client, ad esempio direttamente nella pagina o in un cookie, altre comportano la memorizzazione nel server delle informazioni tra un round trip e il successivo.

## 4.14 View state

La proprietà Control.ViewState fornisce un dizionario di oggetti per conservare valori tra più richieste della stessa pagina. Questo è il metodo che le pagine utilizzano per preservare i valori delle proprietà della pagina e dei controlli tra i round trip.

Quando la pagina viene elaborata, lo stato corrente della pagina e dei controlli vengono memorizzate in una stringa e salvati come un campo nascosto. Quando la pagina viene rispedita dal server, la pagina scansiona la stringa di view state durante la sua inizializzazione e ripristina le proprietà salvate.

## 4.15 Campi nascosti

ASP.NET permette di usare i campi nascosti di HTML nei form. Un campo nascosto non viene reso visibile nel browser, ma è possibile settarne il valore come per un controllo standard. Quando la pagina viene spedita al server, il contenuto del campo nascosto viene spedito nel form con i valori degli atri controlli.

Per quanto riguarda la sicurezza, c'è da dire che per i malintenzionati è facile vedere e modificare il contenuto di un campo nascosto.

Il campo nascosto memorizza una singola variabile nella sua proprietà value e deve essere aggiunto esplicitamente al contenuto della pagina. Dopo aver inserito il campo nascosto, ASP.NET fornisce il controllo HtmlInputHidden che offre tutte le funzionalità del campo nascosto.

## 4.16 Cookies

Un cookie è una piccola quantità di dati memorizzata all'interno di un file di testo nel file system del client. Esso contiene delle specifiche informazioni della pagina che il server spedisce al client insieme alla pagina. I cookie possono essere temporanei o persistenti.

Si possono usare cookie per memorizzare informazioni riguardanti un particolare client, o una particolare sessione o una particolare applicazione. Il cookie viene salvato sul client e quando il browser richiede la pagina, il client invia le informazioni del cookie insieme con la richiesta. Il server può leggere il cookie e estrarre il suo valore. Un suo uso tipico è quello per memorizzare se un utente si è già autenticato nella applicazione.

## 4.17 Query string

La query string sono le informazioni che vengono appese alla fine dell'URL della pagina. Un tipico esempio potrebbe essere il seguente:

*http://www.contoso.com/listwidgets.aspx?category=basic&price=100*

Nel precedente URL, la query string parte con il simbolo di punto interrogativo (?) e include due coppie attributo valore, una chiamata category e l'altra chiamata price.

La query string fornisce un semplice, ma limitato, modo per mantenere alcune informazioni di stato. Per esempio, è un semplice modo per passare delle informazioni tra una pagina e un'altra. Comunque, molti browser impongono un limite di 255 caratteri alla lunghezza dell'URL.

Per quanto riguarda la sicurezza, le informazioni passate tramite query string possono essere modificate da utenti malintenzionati, quindi non possono essere usate per passare dati importanti.

## 4.18 Application state

ASP.NET permette di salvare valori usando lo stato di applicazione. Lo stato di applicazione è un meccanismo di memorizzazione accessibile da tutte le applicazioni Web ed è molto usato per salvare informazioni che necessitano di essere mantenute tra round trip e tra pagine.

Lo stato di applicazione è un vocabolario creato durante ogni richiesta a uno specifico URL. Si possono aggiungere le informazioni della specifica applicazione alla struttura per memorizzarle per ogni richiesta di pagina.

## 4.19 Session State

ASP.NET permette di salvare valori usando lo stato di sessione. Lo stato di sessione è simile allo stato di applicazione, eccezion fatta che la sua vita è limitata a una sessione corrente del browser. Se utenti diversi usano la stessa applicazione, ognuno di loro avrà uno stato di sessione diverso. In più, se lo stesso utente lascia l'applicazione per poi tornarvi in seguito avrà comunque uno stato di sessione diverso..

Lo stato di sessione è un vocabolario per memorizzare informazioni della sessione specifica che è necessario mantenere tra un round trip e il successivo e tra richieste di pagine differenti.

Lo stato di sessione permette di:

- Un identificatore unico della richiesta del browser e di mappare essa in una sessione individuale nel server.
- Memorizzare dati relativi alla specifica sessione nel server.

## 4.20 Database support

Mantenere lo stato usando un database è pratica comune quando si memorizzano informazioni specifiche per ogni utente e queste informazioni sono molte. La memorizzazione in database è particolarmente usata per mantenere informazioni a lungo termine e che devono essere mantenute anche se il server viene riavviato.

Per maggiori informazioni sulla gestione dello stato delle pagine ASP.NET vedi bibliografia [6].

# Capitolo 5

# CREAZIONE DI SERVIZI E APPLICAZIONI WEB

Un aspetto importante di Visual Studio .NET consiste nella possibilità di creare applicazioni distribuite basate sul Web. Visual Studio .NET consente di creare l'interfaccia utente dell'applicazione utilizzando pagine Web Form e di creare componenti utilizzando i servizi Web XML.

Web Form è la tecnologia ASP.NET che consente di creare un'interfaccia utente per applicazioni basate sul Web, indipendentemente dal fatto che l'accesso all'applicazione avvenga tramite un browser tradizionale o tramite un dispositivo mobile. Utilizzando le pagine Web Form è possibile creare un'interfaccia utente indipendente dal browser le cui elaborazioni vengono eseguite sul server Web. In questo modo si elimina la necessità di creare versioni dell'interfaccia utente specifiche per il browser o per il dispositivo.

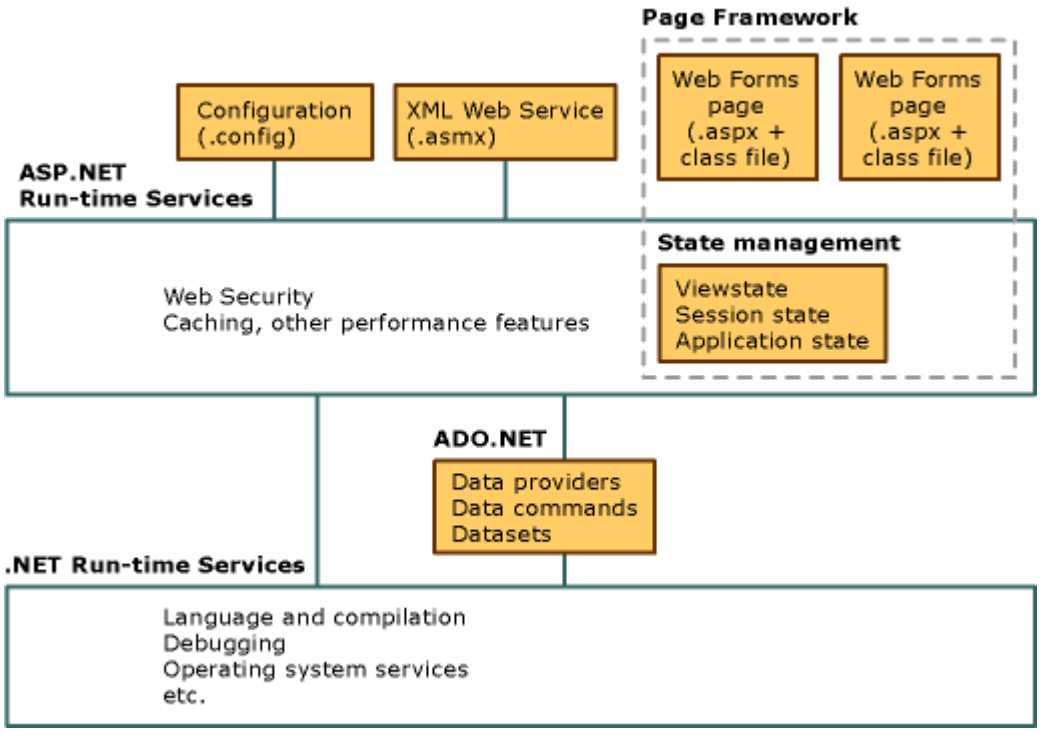
## 5.1 Elementi delle applicazioni Web ASP.NET

La creazione di applicazioni Web ASP.NET implica l'utilizzo di numerosi elementi identici a quelli utilizzati nelle applicazioni desktop o client-server compresi:

- **Funzionalità di gestione dei progetti**. Quando si creano applicazioni Web ASP.NET è opportuno tenere traccia dei file necessari, di quelli da compilare e di quelli da distribuire.
- **Interfaccia utente**. L'applicazione presenta in genere informazioni agli utenti. Nelle applicazioni Web ASP.NET l'interfaccia utente viene presentata sotto forma di pagine Web Form che inviano l'output a un browser. È inoltre possibile creare output personalizzati per periferiche mobili o per altri dispositivi Web.
- **Componenti**. Molte applicazioni includono elementi riutilizzabili che contengono il codice per l'esecuzione di attività specifiche. Nelle applicazioni Web è possibile creare tali componenti come servizi Web XML in modo da poterli chiamare sul Web da un'applicazione Web, un altro servizio Web XML o un Windows Form.

- **Dati**. La maggior parte delle applicazioni richiede una qualche forma di accesso ai dati. Nelle applicazioni Web ASP.NET è possibile utilizzare ADO.NET, i servizi dati che fanno parte di .NET Framework.
- **Protezione, prestazioni e altre funzionalità dell'infrastruttura**. Come per qualsiasi altra applicazione, è necessario implementare le funzionalità di protezione adeguate per impedire l'utilizzo non autorizzato, verificare e sottoporre a debug le applicazioni, ottimizzare le prestazioni ed eseguire altre attività non direttamente correlate alla funzione primaria dell'applicazione.

Nel diagramma illustrato di seguito viene fornita una panoramica dell'integrazione degli elementi delle applicazioni Web ASP.NET e le possibilità di utilizzo all'interno del più vasto ambiente .NET Framework.




Nella sezione riportata di seguito vengono illustrati i diversi elementi di un'applicazione Web ASP.NET e vengono forniti collegamenti a ulteriori informazioni relative all'utilizzo di ogni elemento in Visual Studio.

ASP.NET non rappresenta solamente la nuova versione di ASP (Active Server Pages), bensì fornisce un modello di sviluppo Web unificato che include i servizi necessari per la

generazione di applicazioni Web aziendali. Oltre ad essere ampiamente compatibile con ASP a livello di sintassi, ASP.NET fornisce anche un nuovo modello di programmazione e una nuova infrastruttura per la creazione di applicazioni più scalabili e affidabili che consentono di fornire una maggiore protezione. È possibile aumentare le applicazioni ASP esistenti aggiungendovi funzionalità ASP.NET in modo incrementale.

ASP.NET è un ambiente compilato basato su .NET. È possibile creare applicazioni in un qualsiasi linguaggio compatibile .NET, inclusi Visual Basic .NET, C# e JScript .NET. L'intera piattaforma .NET Framework è inoltre disponibile per qualsiasi applicazione ASP.NET. Gli sviluppatori possono avvalersi facilmente dei vantaggi offerti da queste tecnologie, che includono un ambiente Common Language Runtime gestito, indipendenza dai tipi, ereditarietà e così via.

Per la creazione di un'applicazione ASP.NET, gli sviluppatori possono utilizzare Web Form, servizi Web XML o una combinazione di entrambi, secondo le necessità specifiche. Ogni funzionalità è supportata dalla stessa infrastruttura che consente di utilizzare schemi di autenticazione, memorizzare nella cache i dati utilizzati di frequente, personalizzare la configurazione dell'applicazione e molte altre funzioni.

- I Web Form consentono di generare efficaci pagine Web basate su form. Per la generazione di queste pagine, è possibile utilizzare i controlli server ASP.NET per creare elementi comuni dell'interfaccia utente e programmarli per l'esecuzione di attività comuni. Questi controlli consentono di generare rapidamente un Web Form utilizzando componenti incorporati o personalizzati riutilizzabili, semplificando in tal modo il codice delle pagine..
- I servizi Web XML forniscono i mezzi per accedere in remoto alle funzionalità del server. Mediante i servizi Web XML, le aziende possono esporre interfacce a livello di programmazione per i propri dati o per la propria logica business. Queste informazioni possono essere a loro volta recuperate e modificate da applicazioni server e client. I servizi Web XML consentono di scambiare dati in scenari client-server o server-server mediante l'utilizzo di standard, quali i sistemi di messaggistica XML e HTTP per il trasferimento di dati attraverso i firewall. I servizi Web XML non dipendono da una tecnologia specifica o da una convenzione di chiamata degli oggetti. È quindi possibile accedere a tali servizi con programmi scritti in qualsiasi linguaggio, che utilizzano qualsiasi modello di componente ed eseguiti su qualsiasi sistema operativo.

Questi modelli possono sfruttare appieno tutte le funzionalità offerte da ASP.NET, nonché tutta la potenza di .NET Framework e del relativo Common Language Runtime..

Per maggiori informazioni sulla gestione dello stato delle pagine ASP.NET vedi bibliografia [8].

## 5.2 Introduzione alle pagine Web Form

È possibile utilizzare le pagine Web Form per creare pagine Web programmabili da utilizzare come interfaccia utente per le applicazioni Web. Una pagina Web Form consente di fornire informazioni all'utente in qualsiasi browser o periferica client e di implementare la logica dell'applicazione utilizzando il codice lato server. L'output delle pagine Web Form può contenere quasi tutti i linguaggi che supportano l'HTTP, compresi HTML, XML, WML ed ECMAScript (JScript, JavaScript).

Le pagine Web Form sono:

- Basate su una tecnologia Microsoft ASP.NET in cui il codice eseguito sul server genera in modo dinamico l'output di pagine Web nel browser o nella periferica client.
- Compatibili con qualsiasi browser o dispositivo mobile, nonché in grado di eseguire automaticamente il rendering dell'HTML compatibile con il browser corretto per funzionalità quali gli stili, il layout e così via. In alternativa, è possibile progettare le pagine Web Form in modo che vengano eseguite su un browser specifico, quale Microsoft Internet Explorer 5, e si avvalgano delle funzionalità di un browser client avanzato.
- Compatibili con qualsiasi linguaggio supportato da .NET Common Language Runtime, tra cui Microsoft Visual Basic, Microsoft Visual C# e Microsoft JScript .NET.
- Sviluppate con Microsoft .NET Framework e quindi dotate di tutti i vantaggi del framework, inclusi un ambiente gestito, codice type-safe ed ereditarietà.
- Supportate in Visual Studio con gli efficaci strumenti di sviluppo rapido di applicazioni (RAD) per la progettazione e la programmazione dei form.

- Estendibili, in quanto utilizzano controlli che forniscono la funzionalità RAD per lo sviluppo Web, consentendo di creare in modo rapido interfacce utente avanzate.
- Flessibili, in quanto è possibile aggiungervi controlli di terze parti e creati dagli utenti.

## 5.3 Componenti di Web Form

Nelle pagine Web Form la programmazione dell'interfaccia utente viene suddivisa in due elementi distinti: il componente visivo e la logica.

L'elemento visivo viene definito come *pagina* Web Form. Questa pagina è composta da un file contenente HTML statico o controlli server ASP.NET o entrambi contemporaneamente.

La pagina Web Form funge da contenitore per il testo e i controlli statici che si desidera visualizzare. Se si utilizza Progettazione Web Form di Visual Studio insieme ai controlli server ASP.NET, sarà possibile progettare il form così come in qualsiasi applicazione Visual Studio.

La logica della pagina Web Form è composta dal codice che viene creato per interagire con il form. La logica della programmazione risiede in un file distinto dal file dell'interfaccia utente. Questo file viene definito file di "codice sottostante" e presenta un'estensione "ASPX.VB" o "ASPX.CS". La logica scritta nel file di codice sottostante può essere scritta in Visual Basic o Visual C#.

## 5.4 Struttura dei file Web Form

I file di classe di codice sottostante per tutte le pagine Web Form di un progetto vengono compilati nel progetto file di libreria a collegamento dinamico (DLL). Anche il file della pagina ASPX viene compilato, ma in modo differente. La prima volta che un utente visualizza una pagina ASPX, in ASP.NET viene automaticamente generato un file di classe .NET che rappresenta la pagina e che viene quindi compilato in un secondo file DLL. La classe generata per la pagina ASPX eredita dalla classe di codice sottostante compilata nel file DLL del progetto. Quando un utente richiede l'URL della pagina Web, i

file DLL vengono eseguiti sul server e viene automaticamente generato l'output HTML per la pagina..

## 5.5 Operazioni eseguibili utilizzando le pagine Web Form

La programmazione delle applicazioni Web presenta particolari difficoltà che in genere non si verificano durante la programmazione delle applicazioni client tradizionali. Di seguito sono illustrati alcuni di questi problemi.

- **Implementazione di un'interfaccia utente Web avanzata.** La progettazione e l'implementazione di un'interfaccia utente con un layout complesso, una grande quantità di contenuto dinamico e oggetti completi e interattivi mediante le funzionalità HTML di base possono risultare operazioni difficili e impegnative. Risulta particolarmente difficoltoso creare un'interfaccia utente avanzata per applicazioni da eseguire su piattaforme di periferiche client e browser differenti.
- **Separazione di client e server.** In un'applicazione Web il client (browser) e il server sono programmi differenti, in genere eseguiti su computer differenti e persino su sistemi operativi differenti. Di conseguenza, le due parti dell'applicazione condividono una quantità minima di informazioni; possono comunicare, ma in genere si scambiano solo piccoli blocchi di dati semplici.
- **Esecuzione senza informazioni sullo stato.** Quando un server Web riceve la richiesta di una pagina, la pagina viene trovata, elaborata, inviata al browser e vengono infine eliminate tutte le informazioni relative alla pagina. Se l'utente richiede di nuovo la stessa pagina, il server ripeterà l'intera sequenza di operazioni, rielaborando la pagina da zero. In altre parole, un server non ha memoria delle pagine che ha elaborato. Pertanto, se un'applicazione deve gestire le informazioni su una pagina, è necessario risolvere questo problema nel codice dell'applicazione.
- **Funzionalità del client sconosciute.** In molti casi, la maggior parte degli utenti può accedere alle applicazioni Web utilizzando browser differenti. I browser sono dotati di funzionalità differenti, il che rende difficile la creazione di un'applicazione che possa essere eseguita in modo corretto su tutti i diversi browser.

- **Problemi con l'accesso ai dati.** La lettura e la scrittura da e in un'origine dati nelle applicazioni Web tradizionali può risultare complicata e richiedere un utilizzo intensivo di risorse.
- **Problemi con la scalabilità.** In molti casi le applicazioni Web progettate con metodi esistenti non sono in grado di raggiungere specifici obiettivi di scalabilità a causa della mancanza di compatibilità tra i vari componenti dell'applicazione. Questa caratteristica rappresenta in genere il vero e unico fattore problematico nelle applicazioni con un ciclo di crescita elevato.

Risolvere queste difficoltà per le applicazioni Web può richiedere sforzi e tempi considerevoli. Le pagine Web Form e il framework di pagine ASP.NET affrontano tali problematiche utilizzando le funzionalità riportate di seguito.

- **Modello a oggetti intuitivo e coerente.** Il framework di pagine ASP.NET presenta un modello a oggetti che consente di considerare il form come un'unità, anziché come parti client e server distinte. In questo modello è possibile programmare il form in modo più intuitivo rispetto alle applicazioni Web tradizionali, in quanto è possibile impostare le proprietà per gli elementi dei form e rispondere agli eventi. I controlli server ASP.NET rappresentano inoltre un'astrazione del contenuto fisico di una pagina HTML e costituiscono l'interazione diretta tra browser e server. In generale, è possibile utilizzare i controlli server nello stesso modo in cui si utilizzano i controlli in un'applicazione client senza che sia necessario occuparsi della creazione dell'HTML per presentare ed elaborare i controlli e il relativo contenuto.
- **Modello di programmazione basato su eventi.** Le pagine Web Form consentono di utilizzare per le applicazioni Web il noto modello di scrittura dei gestori eventi per gli eventi generati sul client o sul server. Il framework di pagine ASP.NET astrae questo modello in modo che il meccanismo sottostante per l'acquisizione di un evento sul client, la trasmissione di tale evento al server e la chiamata del metodo appropriato siano automatici e invisibili all'implementatore. Ne risulta una struttura di codice scritta in modo semplice e chiaro che supporta lo sviluppo basato su eventi.
- **Gestione intuitiva dello stato.** Il framework di pagine ASP.NET gestisce automaticamente le attività di gestione dello stato del form e dei relativi controlli e consente di gestire in modo esplicito lo stato delle informazioni specifiche delle applicazioni. Tale gestione può essere eseguita senza un ingente utilizzo di risorse server e può essere implementata con o senza l'invio di cookie al browser.

- **Applicazioni indipendenti dai browser.** Il framework di pagine ASP.NET consente la creazione di tutta la logica delle applicazioni sul server, eliminando la necessità di definire nel codice in modo esplicito le differenze dei browser. Consente comunque di avvalersi automaticamente delle funzionalità specifiche dei browser scrivendo il codice lato client al fine di fornire un livello più elevato di prestazioni e applicazioni client più ricche e complesse.
- **Supporto per il Common Language Runtime di .NET Framework.** Il framework di pagina ASP.NET è una tecnologia ASP.NET. Poiché ASP.NET si basa su .NET Framework, l'intero framework è disponibile a qualsiasi applicazione ASP.NET. È possibile creare le applicazioni in qualsiasi linguaggio compatibile con il runtime, compresi Microsoft Visual Basic, Visual C# e JScript .NET. L'accesso ai dati viene inoltre semplificato mediante l'infrastruttura di accesso ai dati fornita da .NET Framework, incluso ADO.NET.
- **Scalabilità delle prestazioni dei server di .NET Framework.** Il framework di pagine ASP.NET consente di rendere scalabile l'applicazione Web da un computer con un singolo processore a una "Web farm" multicomputer in modo efficace e diretto e senza apportare modifiche complesse alla logica dell'applicazione.

Per maggiori informazioni sulla gestione dello stato delle pagine ASP.NET vedi bibliografia [9].

# Capitolo6

# APPLICAZIONE REALIZZATA

In questo capitolo vengono descritte le modalità con cui è stata realizzata l'applicazione e quali sono state le difficoltà riscontrate. Il problema affrontato consiste nel completare un'applicazione precedentemente realizzata che crea pagine Web dinamiche per la progettazione concettuale ER di database e la traduzione degli stessi in RDF.

Il completamento del progetto prevedeva la realizzazione di funzioni che consentissero il salvataggio dell' RDF ottenuto e la possibilità di ricostruire lo schema utilizzando come risorsa il file contenente l'RDF salvato al fine di ottenere un'applicazione Web Service.

## 6.1 Salvataggio del file RDF

Per il salvataggio dell'RDF è stato inserito un tasto a piè di pagina che estrapola le righe del file dalla casella di testo dove erano state caricate e le salva in un file di testo sul server Web.

La Classe File della libreria di classi System.IO fornisce i metodi statici per creare, copiare, eliminare, spostare e aprire i file. È possibile utilizzare la classe File anche per ottenere e impostare gli attributi dei file o le informazioni di DateTime relative alla creazione, all'accesso e alla scrittura di un file.

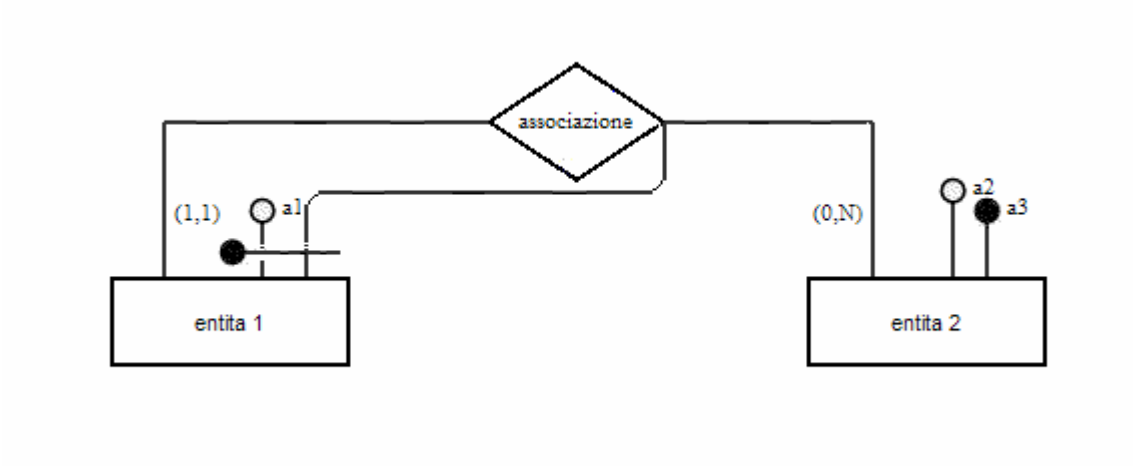
Nella creazione del file è molto importante che sia corretto il percorso. Se un percorso, ad esempio, è completo ma inizia con uno spazio, lo spazio non verrà eliminato nei metodi della classe. Il formato risulterà pertanto scorretto e verrà generata un'eccezione. Analogamente, un percorso o una combinazione di percorsi non possono essere specificati completamente due volte. Anche "c:\temp c:\windows", ad esempio, nella maggior parte dei casi genera un'eccezione.

Tutti i seguenti percorsi, ad esempio, sono accettabili:

- "c:\\MyDir\\MyFile.txt".
- "c:\\MyDir".
- "MyDir\\MySubdir".
- "\\\\MyServer\\MyShare".

## 6.2 Caricamento di uno schema ER dall'RDF salvato

Per il caricamento dello schema ER partendo dall'RDF si è utilizzato una funzione di filtro.
Il filtro è una procedura che ricerca il file lo apre in lettura e lo copia in un array di stringhe suddividendolo riga per riga così che possa essere manipolato e se ne possano ricavare le informazioni relazionali e di posizionamento dei vari componenti, senza andare a modificare la struttura del file originale.
Come esempio riportiamo uno schema ER nel quale sono stati utilizzati diversi tipi di identificatore.




Il testo dell'RFD che traduce lo schema in figura, compresi gli attributi di disegno, si presenta come segue:

```
<?xml version="1.0"?>
<! DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:er="http://localhost/TESI/RDF/Vocabulary/"
xmlns:dt="http://localhost/TESI/RDF/Datatype/" xml:base="http://localhost/TESI/RDF">
        <rdf:description rdf:about="#entita 1">
                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#entita"/>
                <er:posizioneX>141</er:PosizioneX>
                <er:posizioneY>373</er:PosizioneY>
                <er:haAttributi>
                        <rdf:description rdf:about"#entita 1:a1">
                                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Attributo"/>
                                <er:posizioneX>219</er:PosizioneX>
```

```
                        <er:posizioneY>373</er:PosizioneY>
                        <er:verso>N</er:Verso>
                </description>
        </er:haAttributi>
        <er:haIdentificatori>
                <rdf:description rdf:about="#Id1:entita 1">
                        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Identificatore"/>
                        <er:posTestaX>197</er:PosTestaX>
                        <er:posTestaY>361</er:PosTestaY>
                        <er:posCodaX>262</er:PosCodaX>
                        <er:posCodaY>361</er:PosCodaY>
                        <er:versoIdentificatore>O</er:VersoIdentificatore>
                        <er:versoIdentificatoreRispettoEntita>N</er:VersoIdentificatoreRispettoEntita>
                        <er:parteInterna>
                                <er:Attributo rdf:resource="#entita 1:a1">
                        </er:parteInterna>
                        <er:parteEsterna>
                                <er:IdentificazioneEsterna>
                                        <er:tramite rdf:resource="#associazione"/>
                                        <er:entitaEsterna rdf:resource="#entita 2"/>
                                        <er:posXarrivo>240</er:PosXarrivo>
                                        <er:posYarrivo>373</er:PosYarrivo>
                                </er:IdentificazioneEsterna>
                                <er:IdentificazioneEsterna>
                                        <er:tramite rdf:resource="#"/>
                                        <er:entitaEsterna rdf:resource="#"/>
                                        <er:posXarrivo>0</er:PosXarrivo>
                                        <er:posYarrivo>0</er:PosYarrivo>
                                </er:IdentificazioneEsterna>
                        </er:parteEsterna>
        </er:haIdentificatori>
</rdf:description>
<rdf:description rdf:about="#entita 2">
        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#entita"/>
        <er:posizioneX>541</er:PosizioneX>
        <er:posizioneY>373</er:PosizioneY>
        <er:haAttributi>
                <rdf:description rdf:about"#entita 2:a2">
                        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Attributo"/>
                        <er:posizioneX>612</er:PosizioneX>
                        <er:posizioneY>373</er:PosizioneY>
                        <er:verso>N</er:Verso>
                </description>
                <rdf:description rdf:about"#entita 2:a3">
                        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Attributo"/>
                        <er:posizioneX>625</er:PosizioneX>
                        <er:posizioneY>373</er:PosizioneY>
                        <er:verso>N</er:Verso>
                </description>
        </er:haAttributi>
        <er:haIdentificatori>
                <er:Identificatore rdf:resource="entita 2:a3"/>
        </er:haIdentificatori>
</rdf:description>
<rdf:description rdf:about="#associazione">
        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#AssociazioneBinaria"/>
        <er:posizioneX>400</er:PosizioneX>
        <er:posizioneY>295</er:PosizioneY>
        <er:haAttributi>
        </er:haAttributi>
        <er:primaPartecipazione>
                <rdf:description rdf:about"#associazione:entita 1:">
```

```
                                        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Partecipazione"/>
                                        <er:posXarrivoEntita>167</er:PosXarrivoEntita>
                                        <er:posYarrivoEntita>373</er:PosYarrivoEntita>
                                        <er:versoArrivoEntita>N</er:VersoArrivoEntita>
                                        <er:posXpartenzaAssociazione>356</er:PosXpartenzaAssociazione>
                                        <er:posYpartenzaAssociazione>295</er:PosYpartenzaAssociazione>
                                        <er:versoPartenzaAssociazione>O</er:VersoPartenzaAssociazione>
                                        <er:dellaEntita rdf:resource="#entita 1"/>
                                        <er:etichetta rdf:datatype="&xsd;string"></er:Etichetta>
                                        <er:minimaCardinalita
rdf:datatype="http://localhost/TESI/RDF/Datatype/#minCard">1</er:MinimaCardinalita>
                                        <er:massimaCardinalita
rdf:datatype="http://localhost/TESI/RDF/Datatype/#maxCard">1</er:MassimaCardinalita>
                                </rdf:description>
                        </er:primaPartecipazione>
                        <er:secondaPartecipazione>
                                <rdf:description rdf:about"#associazione:entita 2:">
                                        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Partecipazione"/>
                                        <er:posXarrivoEntita>575</er:PosXarrivoEntita>
                                        <er:posYarrivoEntita>373</er:PosYarrivoEntita>
                                        <er:versoArrivoEntita>N</er:VersoArrivoEntita>
                                        <er:posXpartenzaAssociazione>444</er:PosXpartenzaAssociazione>
                                        <er:posYpartenzaAssociazione>295</er:PosYpartenzaAssociazione>
                                        <er:versoPartenzaAssociazione>E</er:VersoPartenzaAssociazione>
                                        <er:dellaEntita rdf:resource="#entita 2"/>
                                        <er:etichetta rdf:datatype="&xsd;string"></er:Etichetta>
                                        <er:minimaCardinalita
rdf:datatype="http://localhost/TESI/RDF/Datatype/#minCard">0</er:MinimaCardinalita>
                                        <er:massimaCardinalita
rdf:datatype="http://localhost/TESI/RDF/Datatype/#maxCard">N</er:MassimaCardinalita>
                                </rdf:description>
                        </er:secondaPartecipazione>
                </rdf:description>
</rdf:RDF>
```

Nel riquadro sottostante è riportata la parte di codice che apre il file e lo suddivide in righe distinte:

```
//lettura del file
FileStream LeggiFile = new
FileStream ("percorso file\\nomefile.txt",FileMode.Open,FileAccess.Read);
StreamReader sRead = new StreamReader("percorso file\\nomefile.txt ");
string TestoFile = sRead.ReadToEnd();
string []Line = TestoFile.Split('\n');
```

Una volta suddiviso il file e copiato nell'array di stringhe ha inizio un ciclo che scorre riga per riga tutto il testo e non appena identifica un oggetto ne ricerca il nome, il tipo di oggetto di cui si tratta, le varie informazioni di posizionamento e lo crea nella sessione corrente.

Nello scorrimento del file vengono individuate per prime tutte le entità, ognuna delle quali coi relativi attributi e identificatori, poi le associazioni coi relativi attributi e le partecipazioni, e infine le gerarchie ed i subset.

Per la creazione dei componenti sono state utilizzate i metodi delle classi già precedentemente utilizzate con particolare attenzione al passaggio dei parametri che non creassero conflitti all'interno della sessione stessa.

Ad esempio per quanto riguarda gli identificatori si sono presentate tre situazioni differenti.

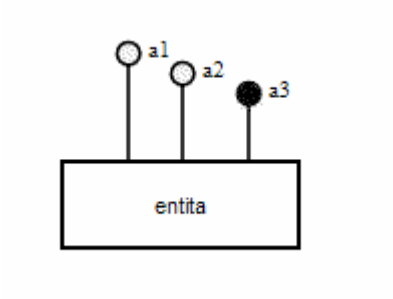



**Fig. 1** Identificatore semplice

Nel caso si trattasse di un identificatore semplice, come in Figura 1, il fatto che l'attributo a3 fosse un identificatore andava specificato al momento della creazione degli attributi dell'entità stessa.

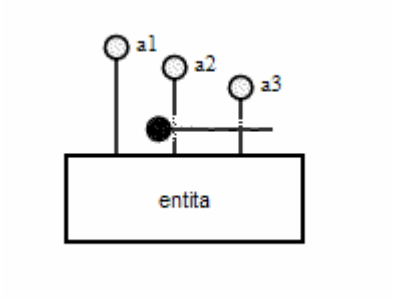



**Fig. 2** Identificatore multiplo

Nel caso dell'identificatore multiplo, quello raffigurato in Figura 2, venivano prima creata l'entità con tutti gli attributi relativi e solo in un secondo tempo veniva creato l'identificatore che coinvolge gli attributi a2 ed a3.

Nel caso invece di un identificatore esterno, Figura 3, il problema è che gli attributi e l'identificatore dovevano essere creati al momento della creazione dell'entità, mentre l'identificatore esterno poteva

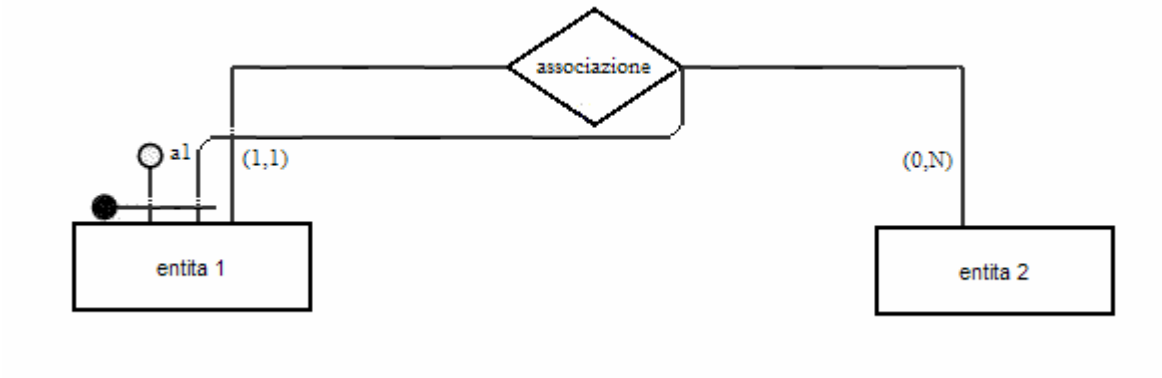



**Fig. 3** Identificatore esterno

essere creato dopo che erano già terminate entrambe le entità, l'associazione e le due partecipazioni. Per ovviare al problema, legato soprattutto alla mancanza di alcune

variabili che si ricavano dalla disposizione dello schema stesso e non sono presenti nell'RDF, vengono memorizzate tutte le informazioni necessarie incontrate nelle varie parti del file e il componente viene creato non appena sono disponibili tutte le informazioni necessarie.

Un esempio analogo riguarda le partecipazioni di un associazione, in fatti il caso in cui un'associazione ha una o due partecipazioni è tradotto in RDF con una codifica diversa da quella del caso in cui un'associazione ha tre o più partecipazioni.

Riportiamo ora di seguito alcune parti del codice della funzione di filtro riguardanti alcuni degli esempi sopra citati.

La prima parte è quella che all'interno di un'entità ricerca eventuali identificatori e verifica di che tipo sono per poi crearli se possibile o memorizzarne in un ArrayList le caratteristiche per crearli poi in seguito.

```
// cerco eventuali identificatori e ne tengo nota
string ident = "";
string []nomeid = null;
string []posTestaX = null;
string []posTestaY = null;
string []posCodaX = null;
string []posCodaY = null;
string []versoIdent = null;
string []versoIdEntita = null;
string []attributoID = null;
string []associazione = null;
string []entEsterna = null;
string []posArrivoX = null;
string []posArrivoY = null;

string []identEsterno = new string[17];

 for (  int t=infoOggetto[1]+4;t<infoOggetto[2];t++)
     {if ( Line[t].StartsWith("<er:haIdentificatori>") )
     {if ( Line[t+1].StartsWith("<er:identificatore") )
{
 string []identificatore = Line[t+1].Split(':');
 identificatore = identificatore[3].Split('"');
 ident = identificatore[0];
 break; //fine for t
}

if ( Line[t+1].StartsWith("<rdf:description") )//ricerca identificatori multiplo
o esterno
{
string []nomeident = Line[t+1].Split('#');
nomeident = nomeident[1].Split('"');
nomeid = nomeident[1].Split(':'); // nomeid[0]= nome identif, nomeid[1]= nome
entita
string []posTX = Line[t+3].Split('>');
posTestaX = posTX[1].Split('<');
```

```
string []posTY = Line[t+4].Split('>');
posTestaY = posTY[1].Split('<');
string []posCX = Line[t+5].Split('>');
posCodaX = posCX[1].Split('<');
string []posCY = Line[t+6].Split('>');
posCodaY = posCY[1].Split('<');
string []versoID = Line[t+7].Split('>');
versoIdent = versoID[1].Split('<');
string []versoIDE = Line[t+8].Split('>');
versoIdEntita = versoIDE[1].Split('<');
t = t+8 ;
}

if ( Line[t].StartsWith("<er:parteInterna>") )
{
 if ( Line[t+2].StartsWith("</er:parteInterna>") ) //identificatore esterno
{
string []att = Line[t+1].Split('#');
attributoID = att[1].Split(':');
attributoID = attributoID[1].Split('"');
associazione = Line[t+5].Split('#');
associazione = associazione[1].Split('"');
entEsterna = Line[t+6].Split('#');
entEsterna = entEsterna[1].Split('"');
string []posAX = Line[t+7].Split('>');
posArrivoX = posAX[1].Split('<');
string []posAY = Line[t+8].Split('>');
posArrivoY = posAY[1].Split('<');
//salvo nell' array identEsterno tutte le informazioni relative
all'identificatore esterno
identEsterno[0] = nomeid[0];//nome identif
identEsterno[1] = nomeid[1];//nome entita
identEsterno[2] = posTestaX[0];
identEsterno[3] = posTestaY[0];
identEsterno[4] = posCodaX[0];
identEsterno[5] = posCodaY [0];
identEsterno[6] = versoIdent[0];
identEsterno[7] = versoIdEntita[0];
identEsterno[8] = attributoID[0];
identEsterno[9] = associazione[0];
identEsterno[10] = entEsterna[0];
identEsterno[11] = posArrivoX[0];
identEsterno[12] = posArrivoY[0];
identEsterno[13] = "";
identEsterno[14] = "";
identEsterno[15] = "";
identEsterno[16] = "";
ListaIdent.Add (identEsterno);
}

else //identificatore multiplo
{
ArrayList IdentMultiplo = new ArrayList();
for (int a=t+1; a<Line.Length; a++)
{if ( Line[a].StartsWith("</er:parteInterna>") )
 {
  //creo identificatore
LibreriaDiClassi.Identificatore I = new LibreriaDiClassi.Identificatore();
I.Imposta(nomeid[1],Convert.ToInt32(posTestaX[0]),Convert.ToInt32(posTestaY[0]))
;
I.FineCoda(Convert.ToInt32(posCodaX[0]),Convert.ToInt32(posCodaY[0]),Convert.ToC
har(versoIdEntita[0]),Convert.ToChar(versoIdent[0]));
```

```
int numattributi = IdentMultiplo.Count;
for (int num=0;num<numattributi;num++)
{
I.AggiungiAttributo((string)IdentMultiplo[num]);
}
I.Ok();
Panel1.Controls.Add(I);
break; //interrompe ciclo for a
}

else
{string []att = Line[a].Split('#');
string []attributoIDM = att[1].Split(':');
attributoIDM = attributoIDM[1].Split('"');
IdentMultiplo.Add (attributoIDM[0]);
}
}// fine for a
}// fine else identificatore multiplo
}//fine  if ( Line[t].StartsWith("<er:parteInterna>") )
} // fine if ( Line[t].StartsWith("<er:haIdentificatori>") )
if ( Line[t].StartsWith("</er:haIdentificatori>"))
{
break; // interrompe ciclo for t
}
}//fine for t
```

Riportiamo infine la parte di codice che dopo la creazione di tutte le entità ed associazioni si occupa degli identificatori esterni dei quali erano state annotate le informazioni.

```
//creo gli identificatori esterni che avevo annotato nell'arraylist
int nident = ListaIdent.Count;
for (int i=0;i<nident;i++)
{
string []id = (string[])ListaIdent[i];
LibreriaDiClassi.IdentificazioneEsterna IE = new
LibreriaDiClassi.IdentificazioneEsterna();
IE.Imposta(id[1],id[9],Convert.ToInt32(id[11]),Convert.ToInt32(id[12]),Convert.T
oChar(id[7][0]),Convert.ToInt32(id[15]),Convert.ToInt32(id[16]),Convert.ToChar(i
d[13][0]),Convert.ToChar(id[14][0]),id[10]);
Panel1.Controls.Add(IE);

}
```

# Capitolo7

# CONCLUSIONI

Il lavoro svolto ha portato ad un ampliamento di una applicazione Web che permette di progettare database tramite il disegno di schemi ER all'interno di una pagina dinamica scritta con al tecnologia ASP.NET, consentendo il caricamento nella pagina stessa di uno schema ER del quale fossero state salvate le informazioni in formato RDF comprensive degli attributi di posizionamento. Inoltre è stata creata una funzione che permette di salvare il file RDF una volta tradotto.

Inoltre è possibile una volta caricato uno schema ER modificarlo aggiungendo altri componenti ma non è possibile eliminare parti dello schema una volta posizionate.

Un possibile sviluppo dell'applicazione è quello della cancellazione e della modifica dei componenti. Queste due operazioni potranno essere introdotte in modo semplice in quanto per effettuarle basta modificare o cancellare gli oggetti contenuti nella variabile sessione della pagina.

Sono lasciati per sviluppi futuri la scrittura di controlli più avanzati da effettuare al momento della traduzione dello schema come, ad esempio, la verifica della connessione dello schema.

# BIBLIOGRAFIA

## Modello ER

**[1]** http://www.artestudio53.it/Database/Sistemi%20di%20database%20relazionali.htm
Sistemi di database relazionali.

## Metadato

**[2]** http://www.crit.rai.it/eletel/2003-1/31-4.pdf
Metadati e Modellazione. Evoluzione della gestione dell'informazione nel mondo dei broadcaster. Ing.. Laurent Boch e ing. Alberto Messina Rai Centro Ricerche e Innovazione Tecnologica Torino

## RDF (Resource description framework)

Tutto il materiale riguardante RDF è stato tratto dal sito della W3C (http://www.w3.org), in particolare le pagine che sono state particolarmente utili sono:

**[3]** http://www.w3c.it/papers/RDF.pdf
RDF per la rappresentazione della conoscenza Oreste Signore
Ufficio Italiano W3C1. presso il C.N.R. - Istituto CNUCE
Area della Ricerca di Pisa San Cataldo - Via G. Moruzzi, 1 - 56124 Pisa

## ASP.NET

**[4]** Per la parte di ASP.NET è stato utilizzato il seguente testo:

"Practical ASP.NET"
Steven Smith
JACKSON

**[5]** http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vbconwebformspageprocessingpage.asp

**[6]** http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vbconintroductiontowebformsstatemanagement.asp

## Servizi e applicazioni Web

Tutto il materiale riguardante servizi e applicazioni Web è stato tratto dal sito della Microsoft (http://msdn.microsoft.com), in particolare le pagine che sono state particolarmente utili sono:

**[7]** http://msdn.microsoft.com/library/ita/default.asp?url=/library/ITA/vbcon/html/vboricreatingwebdistributedappsvb.asp

**[8]** http://msdn.microsoft.com/library/ita/default.asp?url=/library/ITA/vbcon/html/vbconintroductiontowebapplicationsinvisualstudio.asp

**[9]** http://msdn.microsoft.com/library/ita/default.asp?url=/library/ITA/vbcon/html/vbconIntroductionToWebForms.asp