

# Guida a mystiqueXML



Tutti i marchi citati all'interno di questa guida appartengono ai loro legittimi proprietari.

## Indice generale

1	Prefazione .....	4
1.1	Scopo del documento .....	4
1.2	Feedback .....	4
1.3	Cosa è mystiqueXML .....	4
1.4	A cosa serve .....	6
2	Installazione e configurazione .....	8
2.1	Dove trovarlo .....	8
2.2	Prerequisiti .....	9
2.3	Permessi su Windows .....	9
2.4	Installazione su Windows .....	12
2.5	Installazione su Linux .....	15
2.6	Configurazione .....	15
2.7	X-Template .....	16
2.8	X-Build .....	17
2.9	X-Splicer .....	20
3	Gli strumenti .....	20
3.1	Il processo .....	21
3.2	XML e dintorni .....	21
3.2.1	XML .....	22
3.2.2	XSD .....	23
3.3	XPath .....	28
3.4	Writer .....	31
4	La visione di insieme .....	31
4.1	I modelli ODT in funzione di XPath .....	31
4.1.1	I segnalibri .....	31
4.1.2	Segnalibri avanzati .....	36
4.1.3	Segnalibro avanzati: i comandi di campo .....	37
4.1.4	Le tabelle .....	43
4.1.5	Tabelle condizionali .....	44
4.1.6	Tabelle multiple .....	46
5	X-Template .....	57
6	X-Splicer .....	61
7	X-Build .....	64
8	APPENDICE A .....	67
9	APPENDICE B: i segnalibri .....	68
10	APPENDICE C: le tabelle .....	68
11	GLOSSARIO .....	73

# 1 Prefazione

## 1.1 Scopo del documento

Lo scopo di questo documento è spiegare cosa è mystiqueXML e le sue potenzialità accompagnando l'utente nell'utilizzo del programma.

Il manuale è strutturato a stadi, si è cercato di simulare l'approccio al programma e di risolvere le domande che potreste porvi man mano che affrontate l'argomento.

Iniziamo spiegando cosa è mystiqueXML per capire se è lo strumento adatto alle vostre esigenze per poi passare all'installazione e all'analisi degli strumenti che verranno utilizzati da e con mystiqueXML.

Per aiutare gli utenti a comprendere i vari argomenti vengono proposti esempi pratici di facile riutilizzo nel lavoro di tutti i giorni.

La guida è rilasciata in copyleft sotto licenza GNU FDL, da intendersi applicata ad ogni parte dei documenti che la compongono.

## 1.2 Feedback

Per ogni commento e suggerimento a riguardo del presente documento:

[sevastian.foglia@yacme.com](mailto:sevastian.foglia@yacme.com)

[maurizio.beriti@yacme.com](mailto:maurizio.beriti@yacme.com)

[patrizia@made4web.it](mailto:patrizia@made4web.it)

## 1.3 Cosa è mystiqueXML

**mystiqueXML** è framework OpenSource per la creazione automatizzata dei documenti basata su OpenOffice.org e su XML.

Un *framework* è una struttura di supporto che ha lo scopo di risparmiare tempo all'utente nello svolgere operazioni ripetitive. Nel nostro caso è uno strumento di supporto per la creazione e lo scambio di documenti.

Alla base di un *framework* ci sono delle parti di [software](#) utilizzabili e corredate da altri strumenti di supporto che consentono di risparmiare al pro-

grammatore la riscrittura di software già elaborato in precedenza con funzionalità simili.

Prendiamo ad esempio un cuoco che usa il suo ricettario.

Il ricettario altro non è che un contenitore di “modelli” di ricette: dalla ricetta di una torta alla ricetta della crema pasticcera per farcire la torta.

Quindi il ricettario contiene tutte le informazioni per la preparazione di un pranzo. Inizierà con l'antipasto sino ad arrivare a preparare il dolce. Ovviamente non tutti i pranzi saranno uguali: ad esempio una volta potrà esserci la torta farcita con la crema al cioccolato e la volta dopo con la crema all'arancia.

Il *framework* utilizzato dal cuoco è composto dal ricettario e dalle ricette.

### Riassumendo:

cuoco	utente
ricettario <sup>1</sup>	parti del modello (tabella, segnalibro, etc)
nuova ricetta	ideazione del modello
ricette	modelli di documento
preparazione della pietanza	creazione di un documento da un modello
replica della pietanza	creazione automatica dei documenti in serie
l'insieme delle pietanze	parco documentale

L'utente crea il proprio modello di documento, ne verifica la correttezza delle impostazioni testandolo con dati realistici e crea tanti modelli quante sono le tipologie di documento di cui necessita.

Tramite il motore di generazione, i modelli vengono trasformati nei documenti finali che andranno a comporre il parco documentale.

Il cuoco deve creare manualmente tutte le pietanze per preparare il suo pranzo; con mystiqueXML i documenti saranno creati automaticamente.

Il risultato è un *framework* composto da:

- X-Template: interfaccia che guida l'utente nella realizzazione del modello di documento;

<sup>1</sup> Inteso come contenitore di ricette di base utilizzate come ingredienti per la preparazione di piatti più elaborati

- X-Splicer: [parser Python](#) integrato in [OpenOffice.org](#) che analizza il modello ed estrae le parti necessarie per la generazione del documento finale;
- X-Build: libreria [Java](#) che trasforma il flusso XML e il modello creando il documento finale.

mystiqueXML è un [software](#) rilasciato con licenza [GPL](#), secondo la quale il [codice sorgente](#) è lasciato alla disponibilità di chiunque voglia contribuire al progetto.

OpenOffice.org e XML sono gli strumenti utilizzati per la generazione dei documenti, ma questi aspetti verranno affrontati più dettagliatamente in seguito.

### 1.4 A cosa serve

mystiqueXML nasce dall'esigenza di organizzare in modo semplice e affidabile la gestione documentale di un ente pubblico, di un'azienda o di un privato.

Nel caso di Pubbliche Amministrazioni, mystiqueXML può essere utilizzato per i seguenti scopi:

- creazione automatica di documenti contabili, amministrativi, referti e certificazioni;
- organizzazione dell'archiviazione digitale dei documenti in entrata e in uscita;
- interazione e dialogo con il sistema di protocollo;
- elaborazione automatica di lettere, avvisi, fax e documenti da ufficio in genere;
- standardizzazione e semplificazione dei processi.

Sfruttando tutti questi aspetti l'ente sarà in grado di:

- garantire innovazione tecnologica ed essere coerente con le direttive europee e nazionali in materia di e-government;
- ridurre sensibilmente il margine di errore;
- controllare e verificare la correttezza dei documenti;
- eliminare il rischio di perdita dei documenti;
- creare report ed analisi aggregati.

Nel caso di aziende private si potrà:

- creare automaticamente fatture, documenti di trasporto e altri documenti contabili/amministrativi;
- realizzare schede prodotto;

- creare cataloghi di prodotto;
- elaborare automaticamente lettere, avvisi, fax e documenti di ufficio in genere;
- organizzare l'archiviazione digitale dei documenti in entrata ed in uscita;
- condividere documenti e rispondere alle esigenze di clienti e fornitori, anche non avendo gli stessi strumenti a disposizione.

Si ottiene così:

- una generazione automatica dei documenti con dati ricavati da database con un *layout* già pronto per l'uso;
- la possibilità di scegliere la modalità di spedizione e/o di archiviazione del documento a seconda delle esigenze interne o esterne;
- la comodità e la rapidità nell'elaborazione dei documenti insieme alla precisione e alla puntualità nella condivisione di questi con clienti e fornitori (anche non in possesso degli stessi software);
- un documento dall'aspetto più accurato.

## 2 Installazione e configurazione

Una volta capito se mystiqueXML è lo strumento adatto alle vostre esigenze è il momento di installarlo.

Il primo passo da fare è reperire la versione più aggiornata del software e verificare la presenza di tutti i componenti necessari e dei requisiti minimi di sistema.

### 2.1 Dove trovarlo

Potete scaricare il *software* direttamente dal sito

<http://mystiquexml.yacme.com/download.htm>

Una volta aperta la pagina dovete scegliere il pacchetto da scaricare in base al vostro *sistema operativo*:

- se avete un sistema Microsoft Windows (come ad esempio Vista, Windows XP, Windows 2000) dovete scaricare il file con estensione “.exe”;
- se avete un sistema operativo [Linux](#) dovete effettuare il download del file con estensione zip.

Nella pagina di download sono presenti anche altri file già contenuti nei pacchetti sopra citati utilizzabili in caso di aggiornamenti segnalati.

Controllate come il vostro [browser](#) gestisce i [pop-up](#).

Nel caso in cui il vostro browser non consenta l'apertura automatica di finestre, collegatevi alla pagina

[http://sourceforge.net/project/downloading.php?groupname=mystiqueXML&filename=mystiqueXML.exe&use\\_mirror=heanet](http://sourceforge.net/project/downloading.php?groupname=mystiqueXML&filename=mystiqueXML.exe&use_mirror=heanet)

cliccate sul testo “*this direct link*” poche righe sotto la scritta “*Downloading...*”.

## 2.2 Prerequisiti

I pacchetti di installazione di mystiqueXML comprendono tutte le librerie necessarie per un corretto funzionamento.

mystiqueXML non ha particolari necessità per quanto riguarda i prerequisiti hardware che vengono “ereditati” dalla versione installata di OpenOffice.org 2 (nell'Appendice A potete trovare i requisiti richiesti).

Nel caso in cui non abbiate ancora installato OpenOffice.org sul vostro PC, potete trovare tutte le informazioni e le versioni aggiornate sul sito:

<http://it.openoffice.org/>

Prima di iniziare il processo di installazione di mystiqueXML bisogna essere certi di avere installati e funzionanti:

- Java Virtual Machine ufficiale [Sun](http://java.sun.com/javase/downloads/index_jdk5.jsp) (la versione consigliata è la JRE 5.0) scaricabile dal sito:

[http://java.sun.com/javase/downloads/index\\_jdk5.jsp](http://java.sun.com/javase/downloads/index_jdk5.jsp)

- OpenOffice.org 2.

## 2.3 Permessi su Windows

Occorre accertarsi che di avere i permessi per poter installare programmi. Se in precedenza siete riusciti ad installare qualche programma (ad esempio: *Acrobat Reader*) significa che avete i permessi necessari. In caso di incertezza, potete verificarlo seguendo le istruzioni di seguito.

Cliccate su “Start”, in basso a sinistra dello schermo (Figura 1)

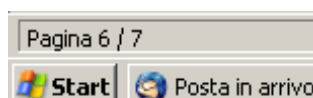


Figura 1: Bottone start

Selezionate la voce “Pannello di controllo” (Figura 2)

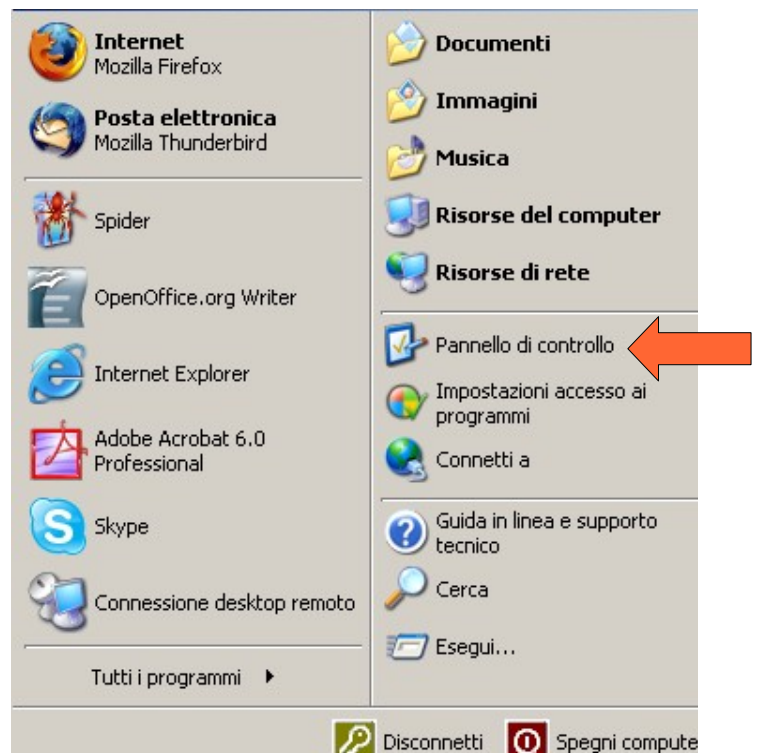


Figura 2: menù d'avvio

Scegliete la categoria “Account Utente”(Figura 3)

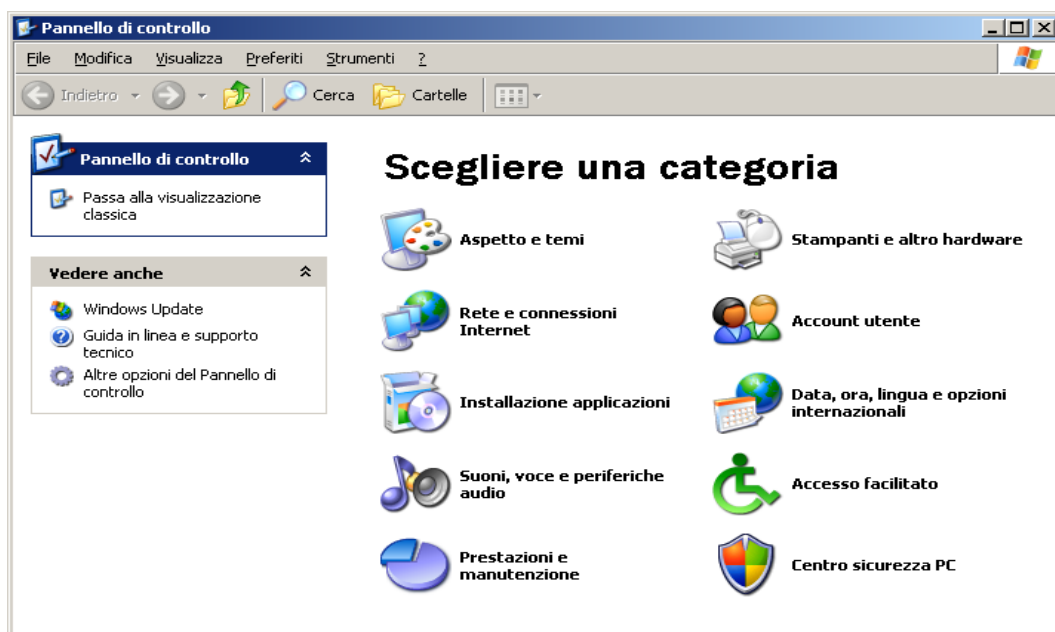


Figura 3: Pannello di controllo

Verificate che sotto il nome del proprio utente compaia la descrizione “Amministratore del computer”(Figura 4).

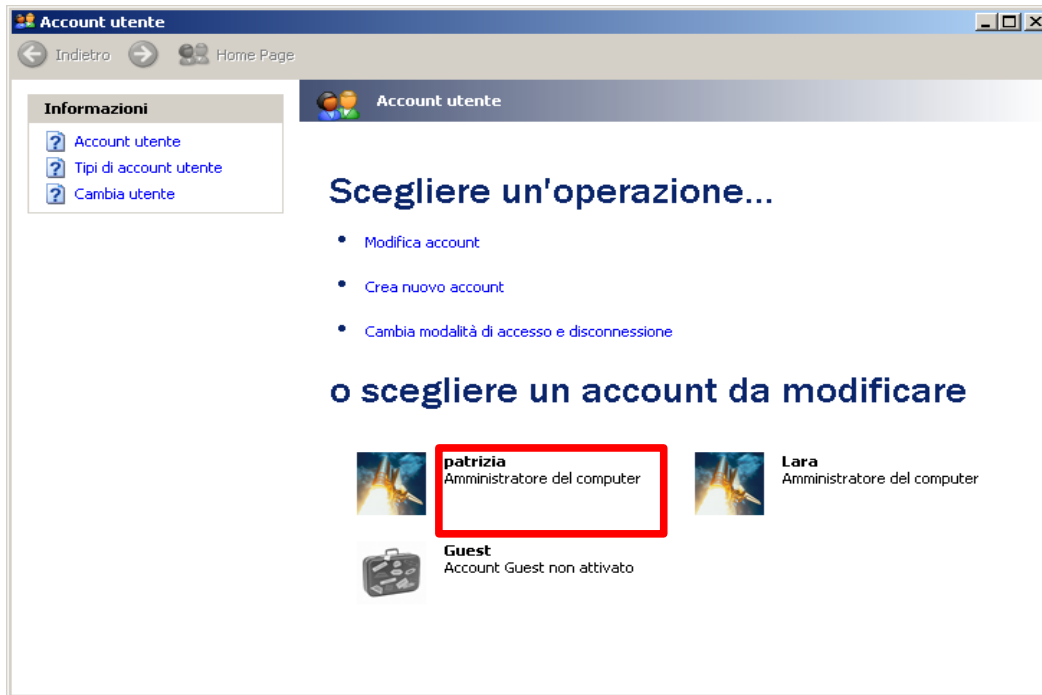


Figura 4: Account utente

A seconda del sistema operativo installato sul vostro PC, pannello potrebbe essere così (Figura 5).



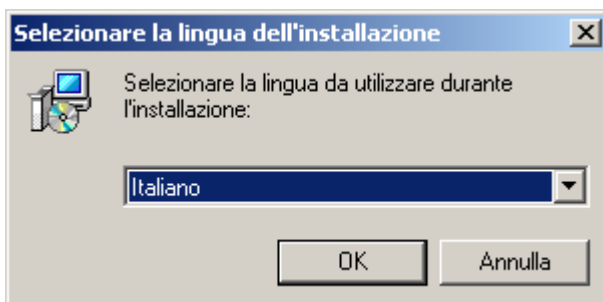
Figura 5: User account

Nel caso in cui il vostro PC faccia parte di una rete aziendale chiedete questa informazione al vostro amministratore di sistema o al vostro tecnico di riferimento.

Se sul vostro sistema sono presenti più utenti, nel corso dell'installazione vi verrà chiesto in che modalità installare il programma: solo per il vostro utente o per tutti.

## 2.4 Installazione su Windows

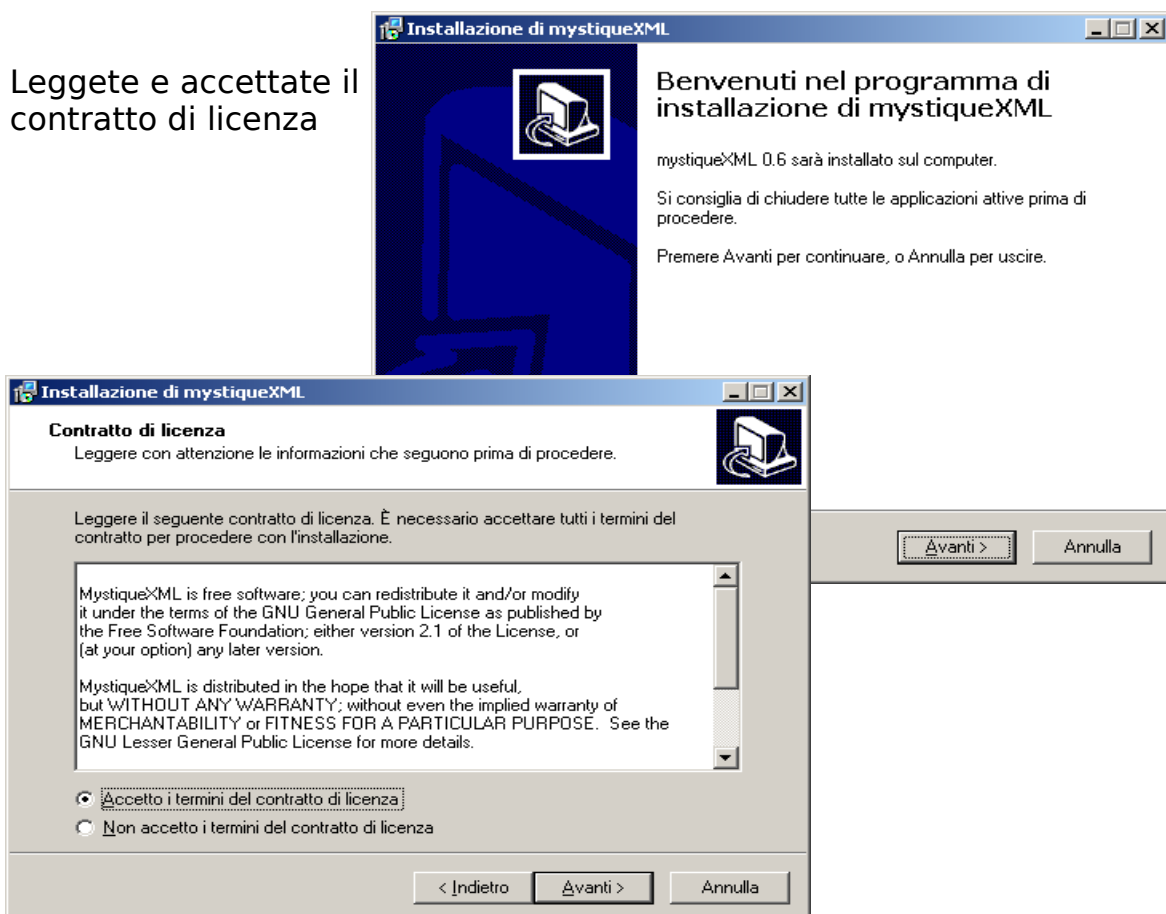
Lanciando il file “.exe” comparirà la seguente schermata:



Selezionate la lingua dall'elenco proposto (Figura 6)

Figura 6: Selezione lingua

Leggete e accettate il contratto di licenza



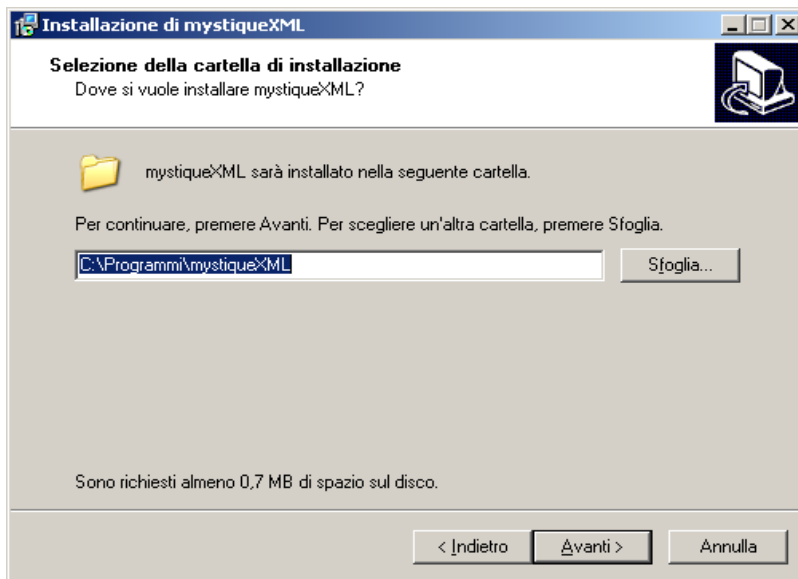


Figura 7: inserimento cartella installazione

Scegliete in quale cartella installarlo (Figura 7).

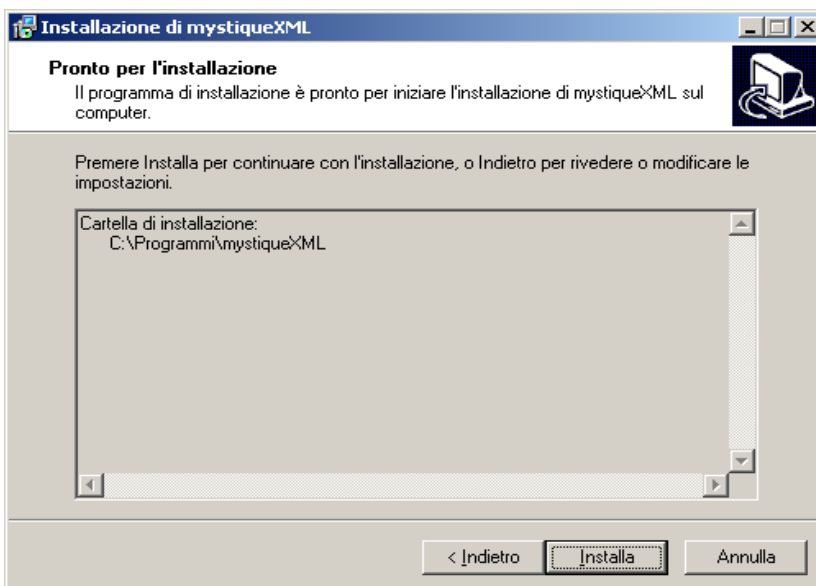


Figura 8: Verifica dati immessi

Se siete sicuri di avere tutti componenti e i percorsi corretti, potete passare all'installazione effettiva del programma (Figura 8).

Verificate che OpenOffice.org non sia aperto (Figura 9).

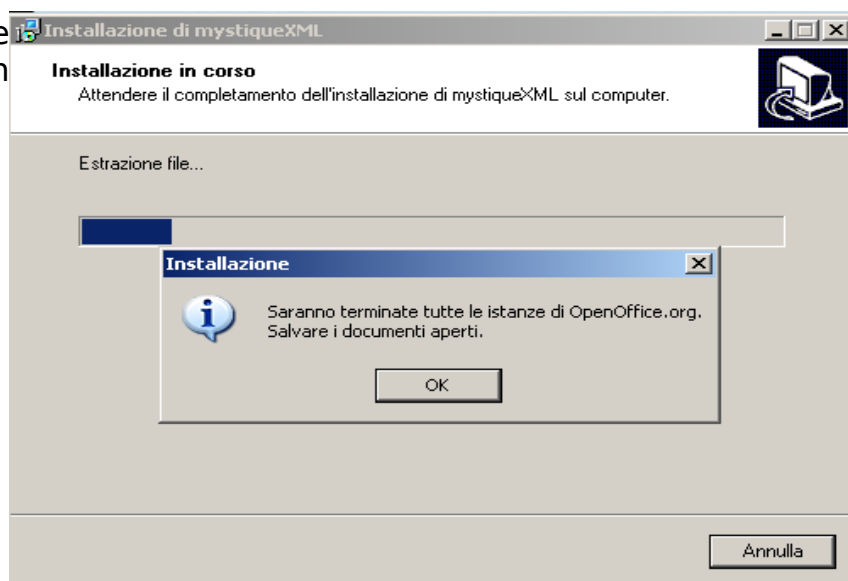


Figura 9: Verifica stato OpenOffice.org

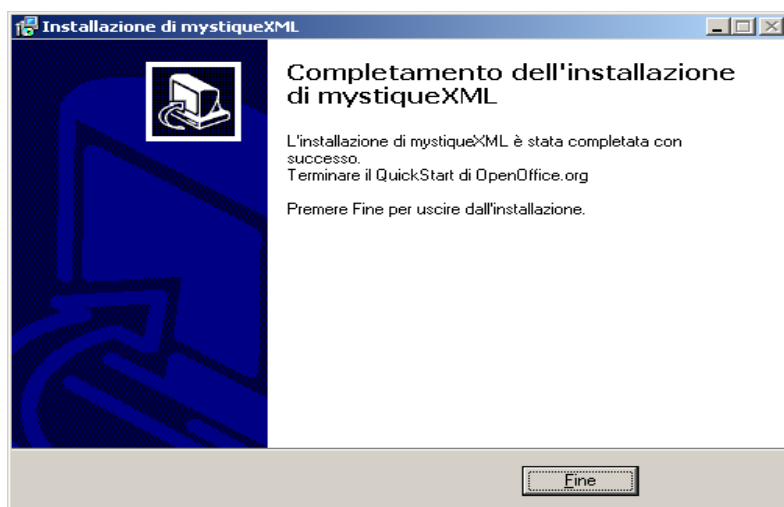


Figura 10: Fine installazione

**INSTALLAZIONE TERMINATA!  
(Figura 10)**

## 2.5 Installazione su Linux

Una volta scaricato il file con estensione .zip, avete il necessario per installare mystiqueXML.

mystiqueXML è stato testato su OpenSuSE con OOo Sun Edition.

Una volta scompattato il file zip in una cartella eseguite lo script *install.sh*

Il programma verifica la presenza di OpenOffice.org, scompatta e installa il framework.

## 2.6 Configurazione

Terminato il processo di installazione sul vostro sistema, se si apre OpenOffice.org, si nota la presenza della voce mystiqueXML nella barra dei menù, che consente l'attivazione delle parti del framework.

Per effettuare la configurazione di ogni modulo occorre attivare le funzionalità dalle voci di menù opportune (Figura 11, Figura 12 Figura 13).

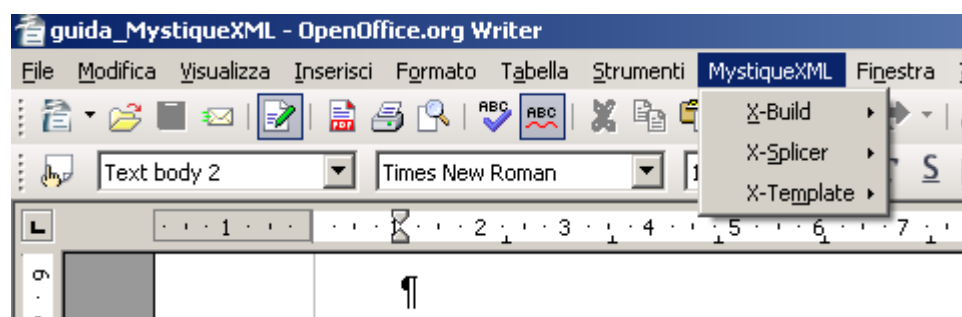


Figura 11: Menù principale

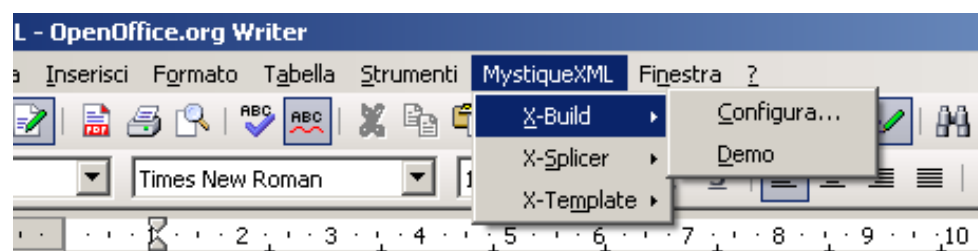


Figura 12: Menù di X-build

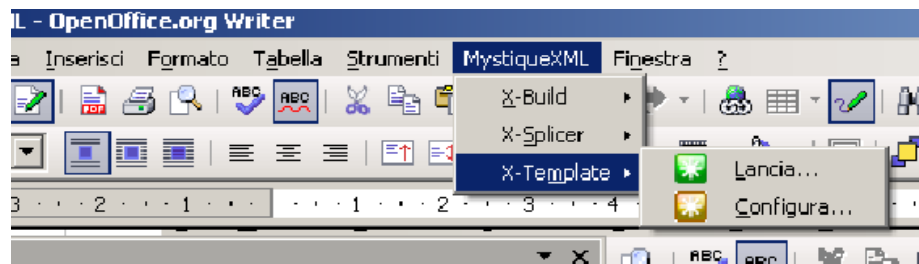


Figura 13: Menù di X-Template

## 2.7 X-Template

Cliccate sul menù *mystiqueXML* -> *X-Template* -> *Configura...*

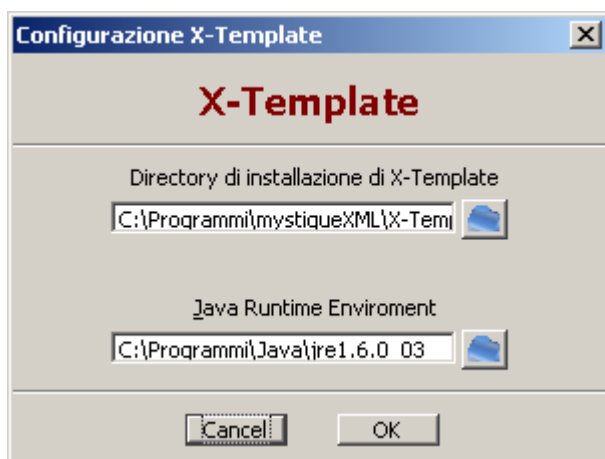


Figura 14: Finestra configurazione di X-Template

Nel primo campo dovete selezionare la cartella di installazione di X-Template; cliccando sul bottone a lato potete esplorare le cartelle del vostro PC (Figura 14).

Se avete un PC win32 e non avete modificato la cartella di installazione, il percorso sarà:

**C:\Programmi\mystiqueXML\X-Template** o **C:\Program Files\mystiqueXML\X-Template**.

Se usate un sistema Linux la directory è quella in cui avete decompresso il file zip.

Nel secondo campo dovete inserire la directory di installazione di Java . Cliccando sul bottone a destra, potete sfogliare le directory del PC e cercare la cartella jre.

In un sistema Windows, le JRE vengono installate sotto la directory **c:\Programmi\Java**.

In un sistema Linux, le installazioni di default sono nella directory

**/usr/lib/jre.**

## 2.8 X-Build

La configurazione di X-Build imposta i percorsi usati durante l'utilizzo dal programma stesso.

Queste informazioni vengono inserite durante la configurazione e non durante il processo di installazione per consentirne la modifica in itinere. Ad esempio, se voleste classificare i documenti in base all'anno di creazione potreste passare dalla cartella 2007 al 2008 cambiando il percorso dei file generati semplicemente lanciando il *wizard* della configurazione di X-Build.



Figura 15: Primo passo

Questo è il percorso della versione di jre installata sul vostro PC (Figura 15).

Se avete già configurato X-Template, il campo contiene il valore corretto.

Questa è la cartella che contiene il file X-Build.jar (Figura 16)

Per trovare il file .jar potete seguire il procedimento descritto per la configurazione di X-Template.

Se i documenti che intendete creare dovranno avere delle immagini (ad esempio il logo



Figura 16: Secondo passo

dell'azienda) questo è il punto in cui specificare la cartella che le contiene (Figura 17).

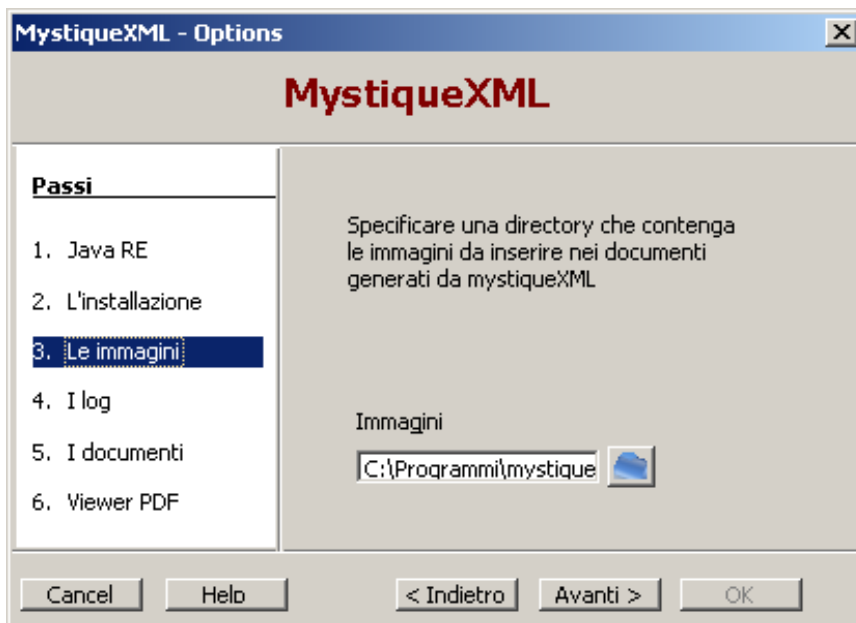


Figura 17: Immagini

La Figura 18 mostra lo step per impostare la cartella dei log. I log sono file in cui vengono registrate le operazioni fatte dal programma, utili in caso di errori per avere informazioni aggiuntive sui problemi avuti.

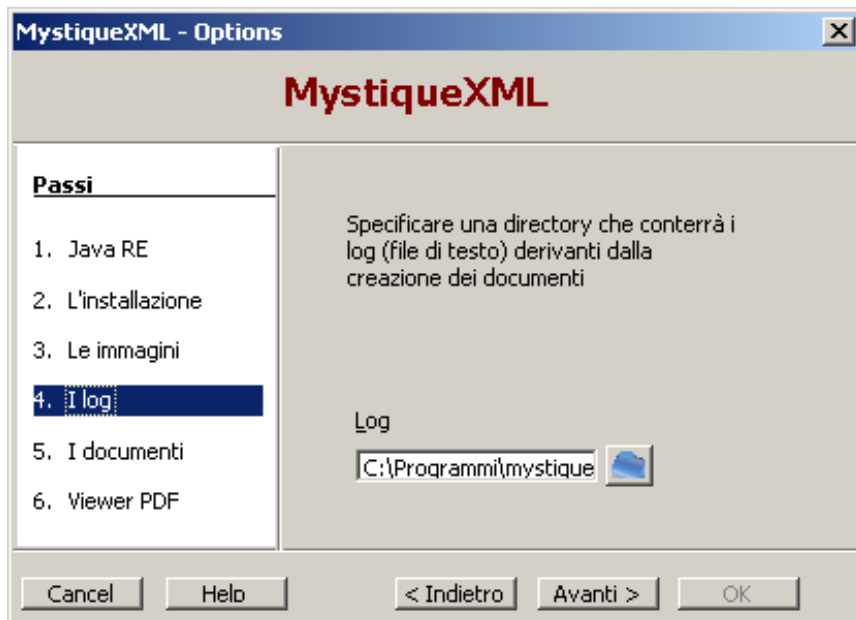


Figura 18: Impostazione della cartella per i log

La Figura 19 mostra la cartella dove salvare i documenti creati.

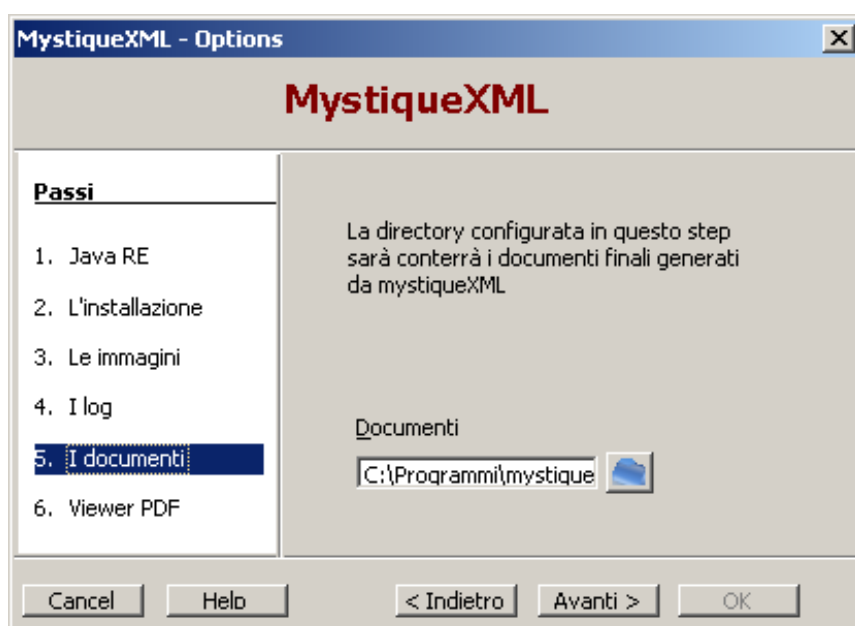


Figura 19: Cartella per i documenti generati

Nell'esempio fatto poteva essere la cartella con il nome dell'anno di creazione del documento.

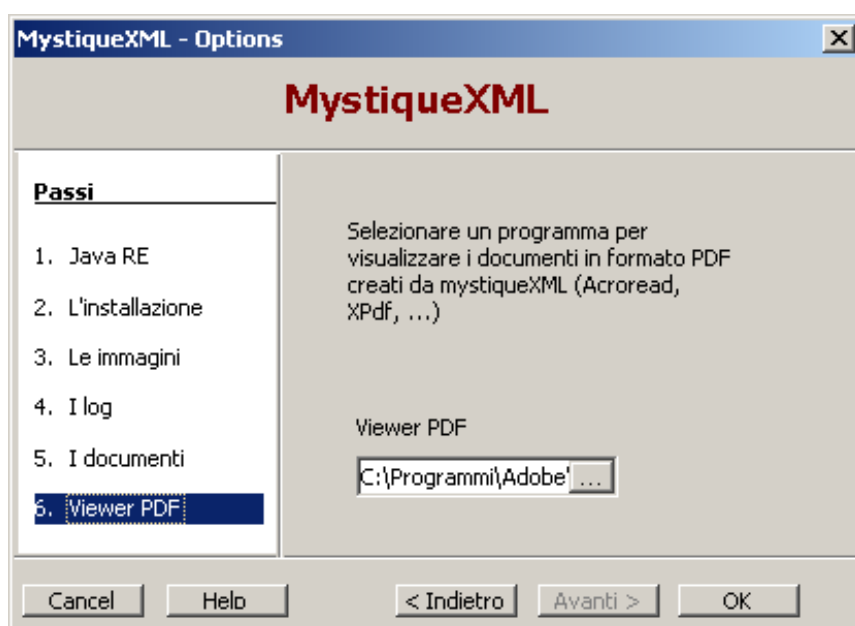


Figura 20: Sesto ed ultimo passo

Come vedremo in seguito, documenti generati non devono essere necessariamente file ODT, ma possono anche essere PDF. Al passo 6 (Figura 20) vi viene chiesto di specificare il percorso del vostro visualizzatore di PDF preferito, per aprire il documento una volta generato.

## 2.9 X-Splicer

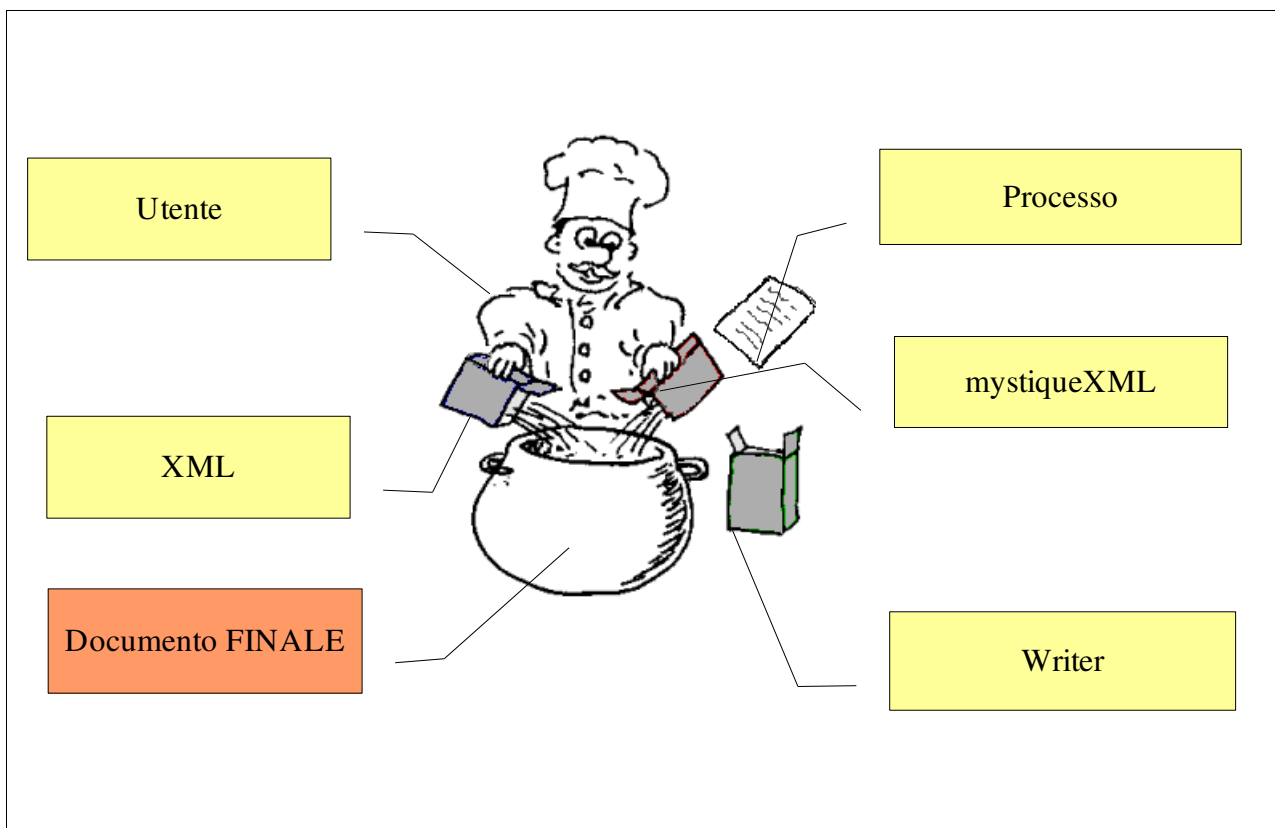
In questo momento non approfondiremo oltre il modulo X-Splicer perché legato al *parsing* (analisi) del modello e all'utilizzo stesso del programma.

## 3 Gli strumenti

Adesso siamo pronti per iniziare a lavorare e a capire come sfruttare al meglio mystiqueXML.

Per poterlo fare è bene approfondire ancora alcuni concetti:

- gli strumenti coinvolti (continuando il parallelo con il cuoco, dovete sapere che ciotole, teglie e fruste utilizzare) ovvero Writer e XML;
- il processo di creazione dei documenti, in modo da sapere a che punto siete man mano che creerete i modelli e li andrete ad usare.



## 3.1 Il processo

Per meglio comprendere come funziona il tutto portiamo avanti il parallelo culinario.

Lo schema sottostante (Figura 21) rappresenta il processo/la preparazione del documento/piatto.

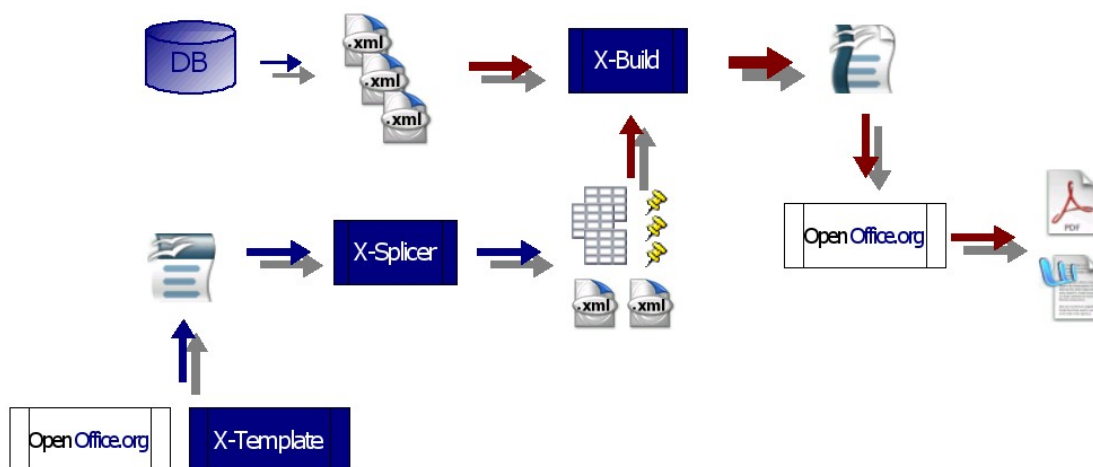


Figura 21: Rappresentazione del processo

Come anticipato, abbiamo gli strumenti che vengono utilizzati per la preparazione del piatto (documento) e gli ingredienti(i dati) da inserire nei modelli.

## 3.2 XML e dintorni

XML si è sviluppato ed evoluto negli ultimi anni grazie alla crescente necessità di scambiare informazioni tra sistemi diversi; così è stato introdotto uno standard che potesse essere utilizzato per sviluppare software e che potesse ricevere e fornire informazioni anche da e per altri sistemi.

Nel nostro caso sono tre gli aspetti che affronteremo perché usati da mystiqueXML: XML, XPath e XSD.

XML, XPath e XSD sono gli strumenti che permettono di far dialogare con la

stessa lingua tutti i componenti coinvolti nell'elaborazione dei documenti.

In questo documento analizzeremo solo i punti principali in funzione dei nostri scopi; qualora si volesse approfondire ulteriormente l'argomento si può fare riferimento ai seguenti link:

- <http://www.w3.org/XML/>
- <http://www.w3.org/TR/XPath>
- <http://www.w3.org/XML/Schema>
- <http://w3schools.org/XML>

### 3.2.1 XML

**XML** (*eXtensible Markup Language*) è un metalinguaggio utilizzato per rappresentare la descrizione di dati strutturati.

Un documento XML è un semplice file di testo, leggibile e scrivibile con un qualsiasi editor di testi, che contiene una serie di *tag*, attributi e testo.

Aperto un file XML possiamo notare alcune particolarità.

Il documento ha una struttura gerarchica composta da *elementi*. Ciascun elemento rappresenta un componente logico e ne può contenere altri (sotto-elementi) oppure può contenere del semplice testo.

Ogni elemento può avere associate altre informazioni che ne descrivono le proprietà. Queste informazioni sono chiamate *attributi*.

L'organizzazione degli elementi segue un ordine gerarchico che inizia con un elemento principale, chiamato **root element** o semplicemente radice. La radice contiene l'insieme degli altri elementi del documento.

Possiamo rappresentare graficamente la struttura di un documento XML tramite un albero, generalmente noto come **document tree**.

Tale struttura è composta da *tag* che servono a descrivere dati secondo regole sintattiche ben definite: sono etichette caratterizzate dai simboli < >.

I *tag*, tranne rari casi, sono composti da un elemento di apertura e uno di chiusura differenziati da /.

Nell'esempio sottostante (Esempio 1) troverete lo stralcio di un *file* XML, dove sono evidenziati i *tag* di apertura/chiusura.

Tutti i file XML iniziano con la seguente intestazione che ne identifica il tipo e l'*encoding*.

```
<?XML version="1.0" encoding="UTF-8" ?>
```

*Esempio 1: Stralcio di file XML*

```
<?xml version="1.0" encoding="UTF-8"?>
<text:list-item>
  <text:p text:style-name="P15">
    <text:span text:style-name="T9">c</text:span>
    se l'XPath non esiste mystiqueXML non
    cancellerÃ la riga della tabella che contiene il
    Segnalibro; naturalmente dovrÃ essere
    preceduto dal flag
    <text:span text:style-name="T9">h</text:span>
    o dovrÃ essere contenuto in una tabella header o
    footer.
  </text:p>
</text:list-item>
```

Un esempio per descrivere con XML una cena, seguendo il nostro “esempio culinario”, è quello mostrato in Esempio 2

*Esempio 2: Le portate di una cena in formato XML*

```
<?xml version="1.0" encoding="UTF-8"?>
<cena>
  <aperitivo>
    <portata>Prosciutto e melone</portata>
    <portata> Tartine burro e salmone</portata>
  </aperitivo>
  <primo>
    <portata> Risotto alla pescatora</portata>
    <portata> Linguine al pesto</portata>
    <portata> Pasta al burro</portata>
  </primo>
</cena>
```

### 3.2.2 XSD

XSD (*XML Schema Definition*) è un metalinguaggio per la definizione della struttura di un documento [XML](#) realizzato dal [W3C](#), ed è inoltre l'unico ad averne ricevuto la validazione ufficiale.

Gli schemi sono anch'essi file XML e come tali hanno una struttura simile a quella vista precedentemente.

Uno schema è formato da un insieme di **dichiarazioni** di elementi e di attributi che ne stabilisce l'interazione (struttura gerarchica) per definire il contenuto e la struttura del documento.

Tramite l'utilizzo degli attributi e degli elementi è possibile:

- definire diverse tipologie di dati;
- stabilire vincoli sul contenuto;
- specificare le occorrenze di un elemento.

L'insieme di relazioni create dagli elementi e dagli attributi nello schema permette la validazione del file XML. Tutti gli elementi inclusi in un documento XML devono essere definiti.

Esistono due categorie di dichiarazioni degli elementi:

- *dichiarazioni di tipo semplice*: utilizzate per elementi che non contengono sotto-elementi e/o attributi;
- *dichiarazioni di tipo complesso*: utilizzate per elementi che contengono sotto-elementi e/o attributi (in questo caso la dichiarazione degli attributi è parte della definizione di tipo complesso).

Un possibile esempio di tipo complesso (Esempio 3):

*Esempio 3: Tipo complesso di dato*

```
<xsd:element name="persona">
  <xsd:complexType>
    <xsd:attribute name="nome" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

Abbiamo definito, in questo caso, un elemento di tipo complesso (*persona*) che prevede un attributo (**nome**) obbligatorio (**required**) di tipo **string**.

Esistono altri comandi per la definizione di ulteriori regole sulla struttura del *file* XML.

I **modelli di contenuto** definiscono cosa l'elemento può contenere:

- *text*: l'elemento può contenere solo testo;
- *empty*: il contenuto è vuoto, l'elemento non può contenere né sotto-elementi né testo (elementi con modello di contenuto *empty* possono però includere attributi);
- *element*: l'elemento può contenere sotto-elementi;
- *mixed*: l'elemento può contenere sotto-elementi e testo (in tal caso definiremo un attributo *mixed* con valore *true* incluso nell'elemento

---

*complexType*); nell'Esempio 5 riportato di seguito lo si riscontra nell'elemento "indirizzoType".

Gli **elementi di composizione** definiscono la struttura di un elemento:

- *sequence*: indica che gli elementi devono comparire in uno specifico ordine all'interno del documento XML;
- *choice*: indica che uno tra gli elementi può essere incluso nel documento XML;
- *all*: indica che qualcuno o tutti gli elementi, in qualsiasi ordine, possono essere inclusi nel documento XML.

E' possibile definire il numero di occorrenze previste utilizzando gli attributi *minOccurs* e *maxOccurs*:

- se *minOccurs* è impostato a 0, l'elemento è opzionale;
- se *maxOccurs* è impostato a *unbounded*, l'elemento può comparire un numero arbitrario di volte;
- in generale, l'elemento può essere ripetuto un numero di volte compreso tra *minOccurs* e *maxOccurs*, estremi inclusi.

Le dichiarazioni di attributi consentono la definizione di alcune caratteristiche come la presenza obbligatoria (**required**) o un valore predefinito (**default**) in combinazione con l'attributo **value**.

Sino ad ora abbiamo affrontato una serie di informazioni e concetti generali. Passiamo ora ad un esempio pratico più comune: la rubrica.

In questo esempio non vengono utilizzati tutti gli elementi descritti precedentemente perché ha lo scopo di sottolineare la relazione tra i *file* XML e i *file* XSD e cosa si intende come struttura gerarchica e dichiarazioni.

Il primo è un *file* XML (Esempio 4) conforme al *file* XSD (Esempio 5)

### Esempio 4: Dati della rubrica in formato XML

```
<?xml version="1.0" encoding="UTF-8"?>
<rubrica>
  <persona>
    <nome>Carlo</nome>
    <cognome>Parodi</cognome>
    <ragionesociale>Parodiamo.S.r.l</ragionesociale>
    <indirizzo tipo="residenza">
      <via>via bianchi 1</via>
      <cap>00000</cap>
      <citta>Genova</citta>
    </indirizzo>
    <partitaiva>001735578330</partitaiva>
    <codicefiscale>PRDCRL55L78CDAL4</codicefiscale>
    <telefono>
      <telefono_fisso>123456</telefono_fisso>
      <cellulare>987656412</cellulare>
    </telefono>
  </persona>
</rubrica>
```

Passando al file XSD, per prima cosa bisogna inserire l'intestazione del documento, dove viene specificato il *namespace*, ovvero il nome di un file dove siano contenute le informazioni di base per i tipi presenti nello schema.

Nel nostro esempio si fa direttamente riferimento alla descrizione di tipi presenti sul sito del W3C attraverso l'inserimento di una riga simile a questa:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

Le specifiche XML Schema prevedono ben 44 differenti tipi di dati e ciascuno di essi ha una lista di "vincoli" che possono essere utilizzati per definire meglio il contenuto. Grazie a questa caratteristica è possibile avere interessanti possibilità di interoperabilità tra documenti XML e database.

**Esempio 5: Schema XML per la creazione di una rubrica**

```
<?xml version="1.0" encoding="utf-16"?>
<xsd:schema version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="rubrica" type="rubricaType" />
  <xsd:complexType name="rubricaType">
    <xsd:sequence>
      <xsd:element name="persona" type="personaType" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="personaType">
    <xsd:sequence>
      <xsd:element name="nome" type="xsd:string" />
      <xsd:element name="cognome" type="xsd:string" />
      <xsd:element name="ragionesociale" type="xsd:string" />
      <xsd:element name="indirizzo" type="indirizzoType" />
      <xsd:element name="partitaiva" type="xsd:int" />
      <xsd:element name="codicefiscale" type="xsd:string" />
      <xsd:element name="telefono" type="telefonoType" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="telefonoType">
    <xsd:sequence>
      <xsd:element name="telefono_fisso" type="xsd:int" />
      <xsd:element name="cellulare" type="xsd:int" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="indirizzoType" mixed="true">
    <xsd:attribute name="tipo" type="xsd:string"/>
    <xsd:sequence>
      <xsd:element name="via" type="xsd:string" />
      <xsd:element name="cap" type="xsd:int" />
      <xsd:element name="citta" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

### 3.3 XPath

XPath si occupa della rappresentazione logica di un documento XML che si può descrivere con la struttura rappresentata in Figura 22.

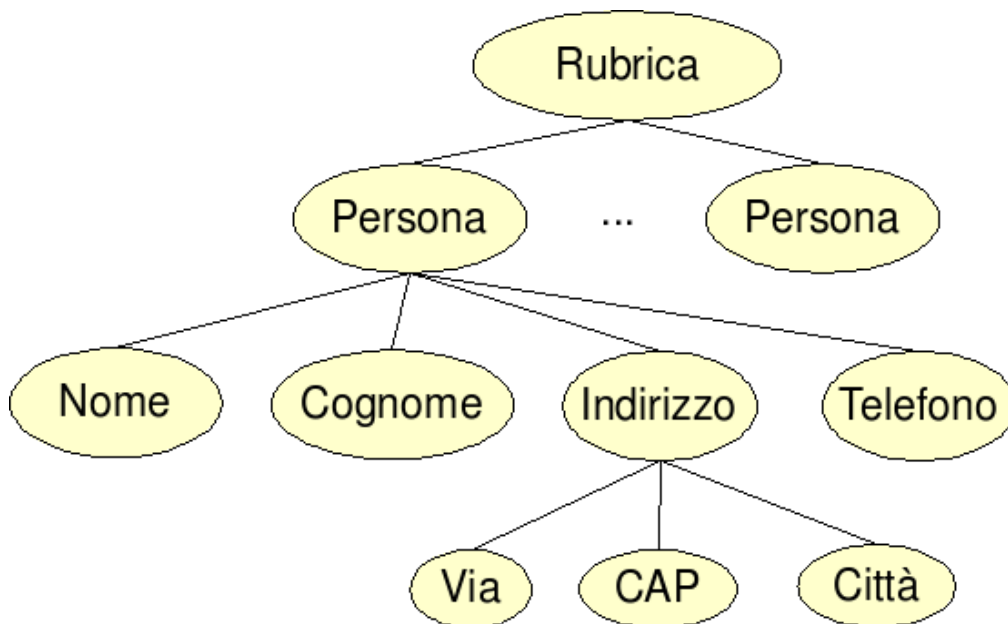


Figura 22: Rappresentazione ad albero della rubrica

Come si può notare la struttura di un documento XML è **ad albero**: questo agevola i meccanismi di analisi (*parsing*) di un documento, ossia consente di riconoscere velocemente i vari elementi del documento stesso tramite un percorso (*location path*).

Nell'esempio rappresentato (Figura 22), l'elemento radice è <Rubrica>, il quale include come elemento (nodo) figlio <Persona>, che a sua volta, è genitore di <Nome>, <Cognome>, <Indirizzo>, eccetera.

Ogni nodo è rappresentato da un percorso; se questo inizia dall'elemento radice viene detto "assoluto". Nel caso in cui si parta dal nodo corrente, il percorso viene definito "relativo". Il percorso assoluto si identifica antepo-  
nendo il carattere "/" prima del nodo radice.

Nel nostro esempio, se sono sul nodo <Persona> e voglio ottenere il conte-  
nuto del nodo <Cognome> posso avere queste alternative:

**Percorso assoluto:** /child::Rubrica/child::Persona/child::Cognome (/Rubrica/Persona/Cognome)<sup>2</sup>

**Percorso relativo:** child::Persona/child::Cognome (Persona/Cognome)

<sup>2</sup> Fra parentesi viene indicata la sintassi abbreviata

Ogni elemento/passaggio è chiamato nodo; il modello XPath i tipi di nodi elencati di seguito:

- nodi radice (*root nodes*);
- nodi elemento (*element nodes*);
- nodi attributo (*attribute nodes*);
- nodi spazio dei nomi (*namespace nodes*);
- nodi istruzione (*processing instruction nodes*);
- nodi commento (*comment nodes*);
- nodi testo (*text nodes*).

Il funzionamento di XPath si basa sull'uso di **espressioni** (*path expressions*). Una *path expression* rappresenta una sequenza di passi per raggiungere un nodo XML.

Ogni elemento ha tre componenti:

axis::node-test[predicate]

- 1) *Axis Specifier*: è il ruolo del nodo e quindi la direzione in cui spostarsi a partire dal nodo di contesto, solitamente *child* (figlio). La componente *axis* esprime la relazione di parentela tra il nodo cercato ed il nodo corrente (come nel nostro esempio il ruolo tra rubrica e nominativo);
- 2) *Node-test*: determina nodo da cercare attraverso il nome;
- 3) *Predicate*: (predicato) può contenere dei filtri (espressi tra parentesi quadre) per specificare delle condizioni più selettive da applicare alla ricerca.

Le relazioni di parentela principali che possono essere contenute in *axis* sono:

- *ancestor*: indica tutti i nodi antenati del nodo corrente, ovvero tutti i nodi che lo precedono nell'albero associato al documento XML;
- *attribute*: indica tutti gli attributi del nodo corrente;
- *child*: indica i nodi figli del nodo corrente;
- *descendant*: indica tutti i discendenti del nodo corrente, ovvero tutti i nodi che seguono il nodo corrente nell'albero XML (figli, nipoti, ...);
- *parent*: indica il nodo genitore del nodo corrente, ovvero quello che lo precede nell'albero;
- *self*: indica il nodo corrente.

Vediamo qualche esempio per capire meglio come utilizzare i *location path* per accedere agli elementi di un documento XML, utilizzando l'esempio della rubrica .

*child::Persona (/Persona)*

Questa espressione seleziona tutti i nodo chiamati 'Persona' che sono figli

del nodo corrente.

`child::* (/*)`

Seleziona tutti i nodi figli del nodo corrente: se fossimo sul nodo "Rubrica" avremmo tutti i nominativi presenti.

`attribute::@tipo`

Seleziona l'attributo di nome 'tipo' del nodo corrente (l'attributo è presente nel tag <indirizzo>).

`descendant::Persona/[Cognome='Rossi'] (/*/Persona[Cognome='Rossi'])`

Seleziona tutti i nodi chiamati 'Cognome' tra i nodi discendenti del nodo corrente, il cui valore è Rossi.

Oltre a mettere in relazione i vari nodi, XPath mette a disposizione funzioni per gestire i nodi, le stringhe, i numeri e i booleani. Queste sono le principali:

- `count(node-set)`: restituisce il numero di nodi contenuti nell'insieme di nodi passato come argomento della funzione;
- `name(nodo)`: restituisce il nome di un nodo;
- `position()`: determina la posizione di un elemento all'interno di un insieme di nodi;
- `last()`: indica la posizione dell'ultimo nodo di un'insieme di nodi;
- `id(valore)`: seleziona gli elementi in funzione del loro identificatore;
- `concat(s1,...,sn)`: restituisce una stringa risultato della concatenazione delle stringhe specificate tra gli argomenti di una funzione;
- `string(valore)`: converte il valore dell'argomento in una stringa;
- `string-length(stringa)`: ritorna la lunghezza della stringa passata come parametro;
- `substring(stringa,inizio,lunghezza)`: restituisce una sotto-stringa della stringa passata come argomento;
- `ceiling(numero)`: arrotonda il numero per eccesso;
- `floor(numero)`: arrotonda il numero per difetto;
- `number(valore)`: converte il valore dell'argomento in un numero;
- `sum(node-set)`: esprime la somma di un insieme di valori numerici contenuti in un insieme di nodi.

Per approfondire l'argomento si consiglia la specifica di XPath, all'indirizzo:

<http://www.w3.org/TR/xpath>

## 3.4 Writer

Writer è uno dei componenti della suite OpenOffice.org e nello specifico ne è il *word processor*.

Gli strumenti di Writer utilizzati in mystiqueXML sono:

- i segnalibri: etichette che vengono collocate nel modello ed associate, attraverso XPath, ad un nodo per inserirne il contenuto nel documento finale;
- le tabelle: per organizzare/formattare il contenuto nel documento. Oltre a compiere le normali funzioni di strutturazione di dati fissi, vengono utilizzate per rappresentare contenuti variabili a seconda di condizioni specificate tramite XPath (in seguito il concetto verrà spiegato meglio).

L'utilizzo di XPath effettua direttamente anche alcune operazioni utilizzando le funzioni. Ad esempio, se il nostro modello fosse quello di una bolla di trasporto, dove i nodi dei dati (rappresentati in XML) sono i singoli colli, tramite la funzione *count* avremmo il totale dei colli automaticamente per realizzare la struttura del documento finale.

## 4 La visione di insieme

Oggetto del manuale è la “versione desktop” che può essere utilizzata per creare, testare (ed eventualmente pubblicare) i modelli di documento. mystiqueXML, ed in particolare X-Build, può essere integrato in un sistema più complesso per la generazione di documenti in serie.

### 4.1 I modelli ODT in funzione di XPath

#### 4.1.1 I segnalibri

I segnalibri rappresentano la componente di OpenOffice.org che consente di inserire nel modello un dato contenuto in un documento XML.

In un modello non possono esistere due segnalibri con lo stesso nome, quindi per inserire una seconda volta un segnalibro già presente è necessario anteporvi un numeratore [01], [02], ... in modo da rendere il nome univoco.

I segnalibri possono essere inseriti come testo normale, nelle celle di tabelle, in righe intestazione o nei piè di pagina.

E' possibile muovere facilmente i segnalibri all'interno del documento con "copia & incolla" oppure per trascinamento. Da notare che è opportuno operare sull'intera etichetta (copiare e incollare l'intera etichetta, trascinare l'intera etichetta).

Affinché X-Build inserisca correttamente il dato in fase di generazione del documento, occorre seguire alcune semplici regole per inserire un segnalibro nel modello:

- creare un testo (etichetta) che termini con il carattere ">" nel punto in cui si vuole che il dato venga immesso;
- inserire, attraverso il menù *Inserisci -> Segnalibro...*, il segnalibro fra il primo e il secondo carattere dell'etichetta;
- il nome del segnalibro deve essere l' XPath di un nodo (o attributo) all'interno del documento XML con le seguenti sostituzioni:
  - ✓ il carattere | al posto di /;
  - ✓ il carattere + al posto di @ per gli eventuali attributi del tag XML;
  - ✓ il carattere ° al posto di .;
  - ✓ il carattere ^ al posto di '';
  - ✓ il carattere \$ al posto di ,;
  - ✓ il carattere % al posto di ::.

L'Esempio 6) mostra come può essere rappresentata l'intestazione di una fattura aziendale.

## Esempio 6: Intestazione fattura

**Yacme S.r.l.**

Via del Mobiliere 9  
40138 Bologna BO  
Italia

Spett. **RagioneSociale>**  
Denominazione>  
codiceNazione> cap>  
localita> codiceProv>  
descrizioneNazione>

Spett. IndirizzoLibero>

Spedire a.. **RagioneSocialeDes**  
**ti>**  
DenominazioneDesti  
>  
CodiceNazioneDesti>  
CapDesti>  
LocalitaDesti>  
DescrNazioneDesti>  
IndirizzoLibero>

L'XPath da inserire come nome del segnalibro **RagioneSociale>** è  
*[01]||FTYA|fattura|testata|informazioni-soggetto|ragione-sociale-soggetto|*  
*ragione-sociale-primaria*  
e l'XML di riferimento è quello mostrato in Esempio 7.

### Esempio 7: XML per la fattura

```
<FTYA>
  <fattura>
    ....
    <testata>
      <informazioni-soggetto>
        .....
        <ragione-sociale-soggetto>
          <ragione-sociale-primaria>GB SRL</ragione-sociale-primaria>
          <ragione-sociale-ricerca>GB SRL</ragione-sociale-ricerca>
        </ragione-sociale-soggetto>
        <indirizzo-soggetto>
          <denominazione-e-numero>VIA VENKMAN, 42</denominazione-e-numero>
          <cap>12060</cap>
          <localita>POLLENZO - BRA</localita>
          <provincia>CN</provincia>
        </indirizzo-soggetto>
        <partita-iva>02800120046</partita-iva>
      </informazioni-soggetto>
    ....
```

L'etichetta, oltre a facilitare l'individuazione del punto d'inserimento del dato, determina il formato (font, colore, dimensione...) con cui il dato stesso sarà inserito.

mystiqueXML supporta le funzionalità degli XPath come, ad esempio, l'individuazione degli attributi, la specifica di condizioni o l'utilizzo delle funzioni.

Nell'esempio appena fatto siamo partiti dal documento affinché si avesse chiara la parte inerente al modello, il modo corretto per organizzare il lavoro è partire dal documento XML.

Prendiamo ad esempio un documento XML (Esempio 8) che rappresenta una scheda prodotto.

*Esempio 8: File XML di una scheda prodotto*

```
<?xml version="1.0" encoding="UTF-8"?>
<YACME>
  <manuale>
    <articolo tipo="DEMO">
      <codice>YACME8842</codice>
      <nome>SuperTux</nome>
      <descrizione>Immagine pinguino TUX</descrizione>
      <immagine>supertux.jpg</immagine>
      <info>
        <prezzo>1820.56</prezzo>
        <valuta>Euro</valuta>
        <colore cod="PV">Rosso</colore>
        <colore cod="RS">Verde</colore>
        <colore cod="ES">Blu</colore>
      </info>
      <nota>XML di test,</nota>
      <nota>volto a capire il funziona-</nota>
      <nota>mento di mystiqueXML.</nota>
      <revisione>
        <tag>1.1</tag>
        <data>2008-02-18</data>
      </revisione>
    </articolo>
  </manuale>
</YACME>
```

Avendo questa struttura potremmo voler inserire nel nostro modello tutti o solo parte dei dati: per questo motivo di seguito vedremo alcuni esempi di segnalibri che potremmo utilizzare nel documento.

### **Segnalibro con attributo**

Per inserire un segnalibro che restituisca il tipo di articolo:

**Nome del segnalibro:** `||YACME|manuale|articolo|+tipo`

**XPath:** `//YACME/manuale/articolo/@tipo`

**Valore restituito:** DEMO

### **Segnalibro con condizione**

Per inserire un segnalibro che ci restituisca il valore del nodo solo se è rispettata una precisa condizione:

**Nome del segnalibro:** `||YACME|manuale|articolo[+tipo='DEMO']|codice`

**XPath:** `//YACME/manuale/articolo[@tipo="DEMO"]/codice`

**Valore restituito:** YACME0042

### **Segnalibro con funzione**

Per utilizzare la funzione *substring* di XPath:

**Nome del segnalibro:** `substring(||YACME|manuale|articolo|codice$8$2)`

**XPath:** `substring(//YACME/manuale/articolo/codice,8,2)`

**Valore restituito:** 42

### **4.1.2 Segnalibri avanzati**

mystiqueXML mette a disposizione dell'utente funzionalità aggiuntive che aumentano la capacità di controllo sul segnalibro e sull'intero documento tramite l'inserimento di codici racchiusi fra parentesi quadre alla fine del nome del segnalibro.

Nel dettaglio:

- **f**

il dato contenuto nel documento XML verrà formattato opportunamente in base al tipo di dato; una data nel formato 2004-05-25 diventerà 25/05/2004, un numero nel formato 1010.1 diventerà 1.010,1.

- **z**

concatena le stringhe mediante uno spazio e le parole divise con il carattere per la sillabazione "-". Seguendo l'esempio di prima (Esempio 8)

**Nome del segnalibro:** `||YACME|manuale|articolo|nota[z]`

**XPath:** `//YACME/manuale/articolo/nota`

**Valore restituito:** XML di test,  
volto a capire il funzionamento di mystiqueXML.

- **d**

con questo tag mystiqueXML visualizzerà solo valori diversi presenti nei nodi individuati da Xpath come se si eseguisse una [query](#) usando il parametro SQL [DISTINCT](#). Se ci fossero due nominativi uguali nella rubrica ne verrebbe visualizzato solo uno.

- **h**

mystiqueXML interpreterà il segnalibro con questo come se fosse contenuto in una tabella *header* o *footer*; cancellerà la riga della tabella che lo contiene se l'XPath corrispondente non esiste nel documento XML.

- **c**

se l'XPath non esiste mystiqueXML non cancellerà la riga della tabella che contiene il segnalibro; dovrà essere preceduto dal [flag h](#) o dovrà es-

sere contenuto in una tabella *header* o *footer*.

- **r**  
rappresenta una condizione per l'eventuale cancellazione di una riga. Se l'XPath individua un valore nell'XML viene cancellata l'etichetta, altrimenti la riga che contiene questo segnalibro viene cancellata (se è contenuto in una tabella).
- **imWWXHH**  
consente l'inserimento di un'immagine nel documento; sostituendo i valori espressi in millimetri di larghezza e altezza dell'immagine ad WW ed HH rispettivamente, è possibile scalarla. Nel nostro esempio  

```
||YACME|manuale|articolo|immagine[im80X80]
```
- **indice**  
consente a mystiqueXML di selezionare fra i nodi trovati mediante l'XPath quello identificato dal numero specificato fra le parentesi quadre.

**Nome del segnalibro:** ||YACME|manuale|articolo|info|colore[1]

**Valore restituito:** Rosso

I codici vengono utilizzati da X-Build; è lui che si occupa della generazione del documento finale una volta selezionato il modello e il file XML. Inoltre, possono essere combinati più codici in modo che X-Build effettui più operazioni contemporaneamente sullo stesso segnalibro.

### 4.1.3 Segnalibro avanzati: i comandi di campo

I comandi di campo sono una funzionalità di Writer che permette l'inserimento di parametri nella pagina che vengono automaticamente compilati. I più conosciuti sono: Data, Orario, Pagine, eccetera.

Non è molto conosciuta la possibilità di poter definire proprie variabili: mystiqueXML sfrutta questa opportunità per implementare dei *dizionari*<sup>3</sup>.

La prima cosa da fare è l'inserimento di un comando di campo in punto qualsiasi del documento, attraverso i seguenti passi (Figura 23):

- seguire il menù **Inserisci -> Comando di campo -> Altro...**;
- selezionare la scheda **Variabili**;
- selezionare come **Tipo di campo** *Campo utente*;
- selezionare il formato **Testo**;
- inserire nella casella **Nome** il nome della variabile;
- inserire nella casella **Valore** il valore secondo la regola *codice::valore[;;codice::valore...]*.

<sup>3</sup> Consente di creare una coppia (codice, valore) per meglio rappresentare i dati

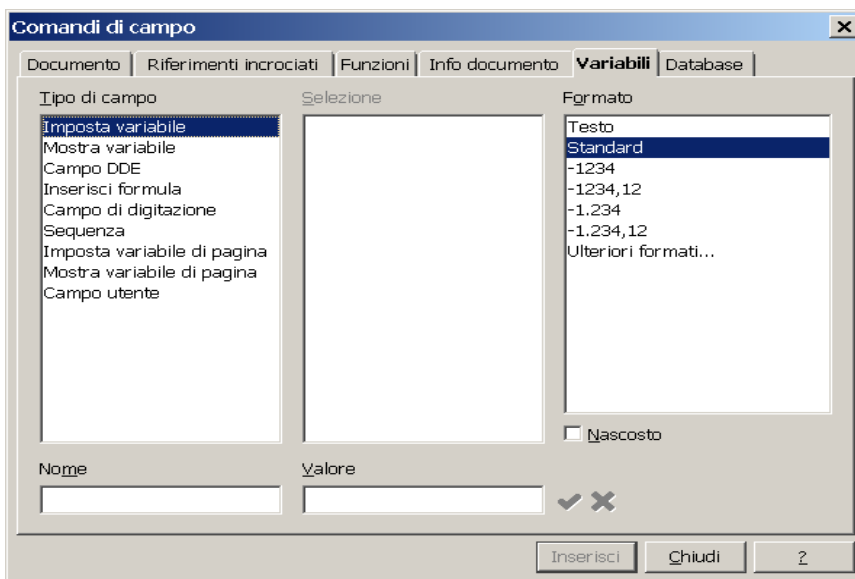


Figura 23: Inserimento comando di campo

A questo punto è possibile specificare un comando di campo insieme ad un segnalibro semplicemente posponendo al nome del segnalibro il nome della variabile definita racchiuso da parentesi tonde .

In tal caso, l'etichetta del documento verrà sostituita con il *valore* della variabile e non con il dato contenuto nell'XML che funge appunto da *codice*.

mystiqueXML supporta due *codici* "fissi" **DEFAULT** e **NULL**. Se non sono stati specificati *valori* per un determinato codice, l'etichetta nel documento verrà sostituita con la stringa vuota usando il *codice* NULL. Il codice DEFAULT viene utilizzato quando si vuole ottenere un *valore* per *codici* non specificati.

Gli esempi che seguono mostrano come il valore del nodo individuato mediante l'XPath viene usato come *codice* per cercare la corrispondente codifica nel dizionario (l'insieme delle variabili definite nel modello).

Il secondo passo è l'inserimento del segnalibro con il comando di campo nel documento che deve avere la forma<sup>4</sup>:

XPath[codice](variabile)

*Comandi di campo – Esempio 1: DEFAULT.*

Il valore di DEFAULT è un valore che viene assegnato automaticamente in caso non vengano date direttive diverse.

Prendiamo il mese di scadenza per il pagamento di una fattura. Se la data di emissione è compresa fra gennaio, febbraio o marzo allora il mese di scadenza sarà quello di emissione, altrimenti il mese sarà dicembre.

<sup>4</sup> Il nome della variabile racchiuso fra parentesi tonde dovrà essere l'ultima parte del nome del segnalibro

La variabile risulterà così:

```
01::gennaio;;02::febbraio;;03::marzo;;DEFAULT::dicembre
```

*Comandi di campo – Esempio 2: NULL.*

Il valore NULL indica un campo vuoto senza caratteri.

Nel caso si voglia inserire la stringa “Omaggio” solo se il nodo individuato da XPath contiene il valore “true”, la variabile sarà:

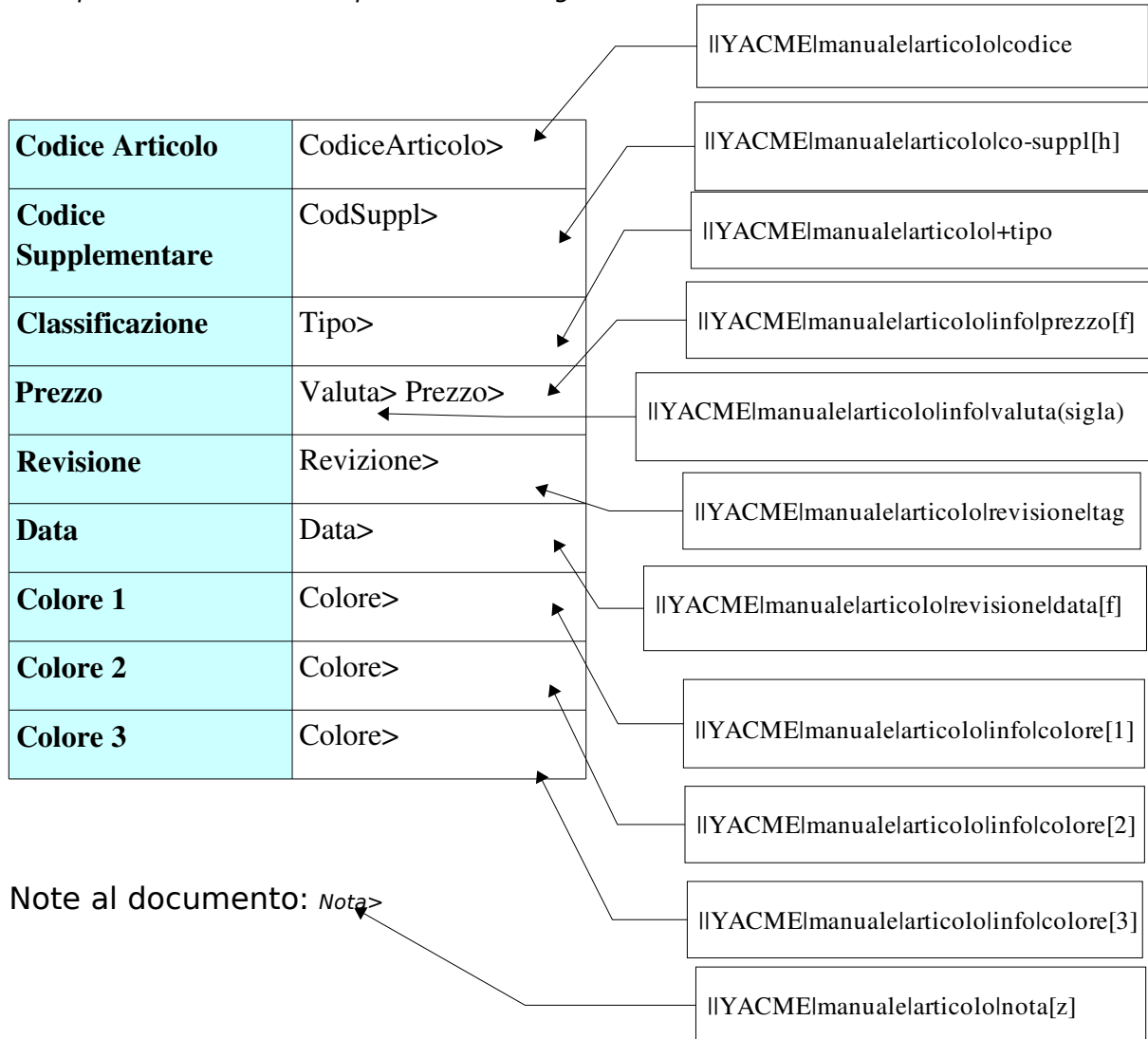
```
true::Omaggio
```

Se il nodo non esiste o contiene un altro valore verrà restituita la stringa vuota, ossia la codifica del codice NULL.

Immaginiamo di avere nel nostro modello una tabella che descrive la scheda di un prodotto a cui “associamo” il *file* XML presentato all'inizio (Esempio 8).

L'Esempio 9 potrebbe essere una delle parti del nostro modello. A destra avete la tabella e a sinistra il nome dei segnalibri in una legenda per meglio chiarire il comportamento del programma.

Esempio 9: Tabella scheda prodotto con segnalibri associati



Il segnalibro con etichetta “Valuta>” contiene la variabile *sigla* (la si individua dal fatto che è compresa tra parentesi tonde) che variabile è stata precedentemente inserita tra i campi utente(Figura 24) con il valore Dollaro::\$;;Sterlina::£;;DEFAULT::€.

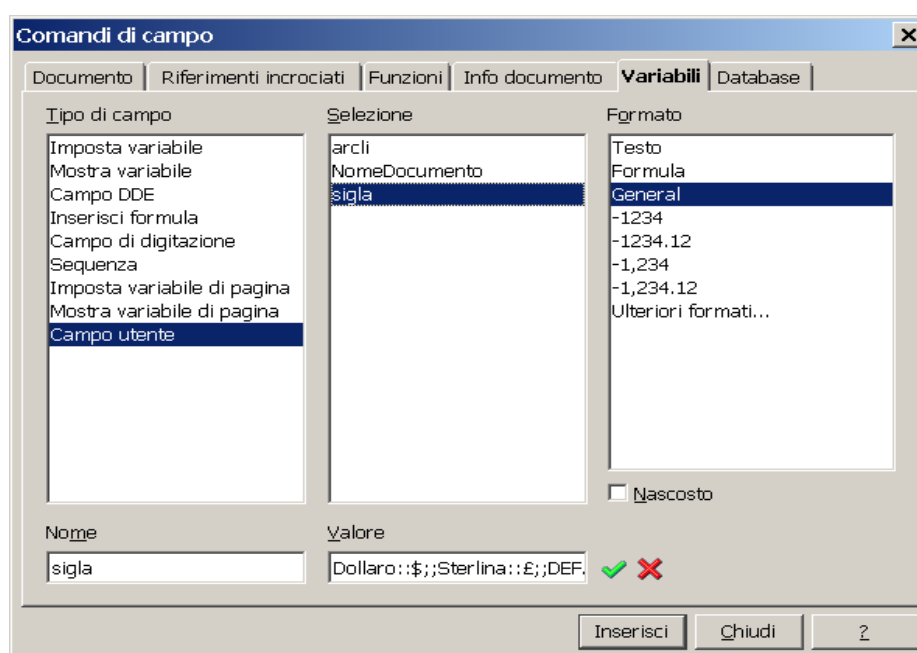


Figura 24: Inserimento comando di campo

Se non vengono specificati i valori della valuta “Dollaro” o “Sterlina” verrà impostato come predefinito il valore €.

Se utilizzassimo X-Build per associare il nostro modello al file XML, otterremmo il documento con la tabella inerente alla scheda prodotto come quella riportata in Esempio 10.

Esempio 10: Risultato

<b>Codice Articolo</b>	YACME0042
<b>Classificazione</b>	DEMO
<b>Prezzo</b>	€ 1.020,56
<b>Revisione</b>	1.1
<b>Data</b>	26/05/04
<b>Colore 1</b>	Rosso
<b>Colore 2</b>	Verde
<b>Colore 3</b>	Blu

Note al documento: XML di test per mystiqueXML, volto a verificare il funzionamento del software.

Osservando il documento creato possiamo notare l'effetto dell'utilizzo dei codici:

- la riga della tabella che conteneva **Codice Supplementare** è stata cancellata perché non esisteva il corrispondente XPath nel documento XML e il segnalibro aveva il codice **[h]**;
- tutti i segnalibri con codice **[f]** sono stati formattati opportunamente;
- il segnalibro "Valuta>" è stato sostituito con la sigla della valuta corrispondente;
- il segnalibro relativo alla **Classificazione** è stato sostituito con il valore dell'attributo tipo dell'elemento articolo;
- le note sono state inserite rispettando la formattazione del documento XML.

#### 4.1.4 Le tabelle

L'inserimento di una tabella in un modello avviene tramite il menù di Writer: *Inserisci->Tabella*.

Affinché mystiqueXML possa mettere in relazione la tabella con un XPath, il nome della tabella deve essere formato da due parti:

- la prima identifica il tipo di tabella che si vuole inserire: è un "codice" che verrà interpretato da X-Build in fase di generazione del documento;
- la seconda deve corrispondere all'XPath di un nodo all'interno del documento XML.

Per approfondire l'uso delle tabelle in Writer si può far riferimento all'APPENDICE B.

L'immagine sottostante (Figura 25) mostra la finestra di Writer dove viene inserito il nome della tabella.

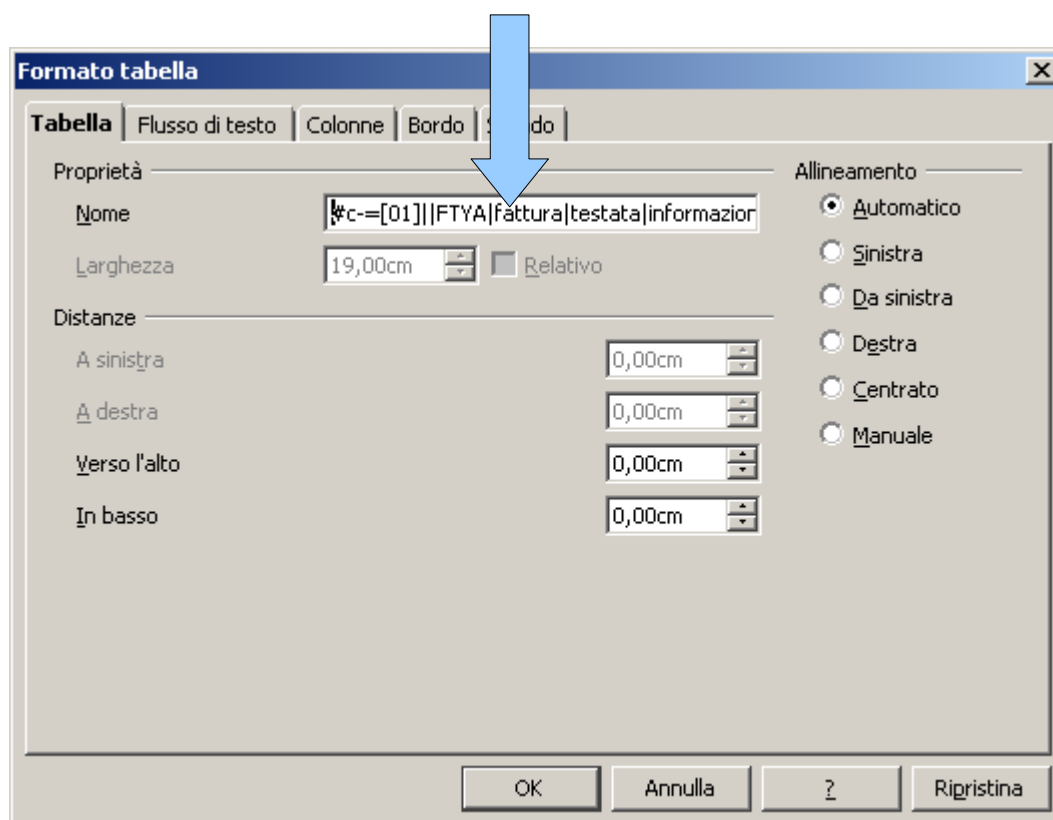


Figura 25: Visualizzazione del nome della tabella

### 4.1.5 Tabelle condizionali

Per definire una **tabella condizionale**, ossia una tabella la cui presenza (all'interno del documento finale) dipende dal verificarsi di una condizione specificata con il nome della tabella stessa, occorre utilizzare il codice **#c** che può essere di due tipi:

- **#c-!**
- **#c-=**

mystiqueXML interpreta il nome della tabella e verifica la condizione (XPath) rappresentata dalla seconda parte del nome stesso:

- se il nome inizia con **#c-!** e la condizione è vera, allora la tabella viene cancellata, altrimenti inserita;
- al contrario, se il nome inizia con **#c-=**, la tabella viene inserita se la condizione è vera, cancellata altrimenti.

L'uso di questo tipo di tabelle risulta utile quando nell'albero XML sono presenti più XPath uguali ma nel documento si vuole che i dati relativi siano inseriti una sola volta.

Può essere altresì utilizzata quando in un modello si vogliono riportare delle caratteristiche generali. Ad esempio, nell'unico modello di un certificato di qualità di una linea di bevande sono inseriti tutti gli elementi chimici (una tabella per ogni elemento) contenuti. Quando si creerà il documento, verranno inserite solo le tabelle corrispondenti agli elementi chimici presenti.

Partiamo dal *file* XML come dell'Esempio 11.

*Esempio 11: Esempio tabella condizionale - file XML*

```
<?xml version="1.0" encoding="UTF-8"?>
<YACME>
  <manuale>
    <tabella-condizionale>
      <condizione> //YACME/manuale/tabella-condizionale </condizione>
      <nome> #c-=|YACME|manuale|tabella-condizionale </nome>
      <tipo>#c-=</tipo>
    </tabella-condizionale>
    <tabella-condizionale>
      <condizione> //YACME/manuale/tabella-condizionale-del </condizione>
      <nome> #c-!|YACME|manuale|tabella-condizionale-del </nome>
      <tipo>#c-!</tipo>
    </tabella-condizionale>
  </manuale>
</YACME>
```

In un caso come questo, vorremmo che venisse visualizzato solo uno dei due elementi `<tabella-condizionale>`.

Gli esempi 12 e 14 mostrano la struttura delle tabelle da inserire nel mo-

dello nel caso si voglia avere nel documento finale una delle due tabelle condizionali. Gli esempi 13 e 15 danno il risultato dell'elaborazione con X-Build.

*Esempio 12: Esempio di tabella condizionale con codice #c-=*

<b>Tabella</b>	<b>TipoTabella</b>	<b>Condizione</b>
Nometabella>	Tipo>	Condizione>

`#c-=||YACME|manuale|tabella-condizionale`

*Esempio 13: Risultato elaborazione mystiqueXML*

<b>Tabella</b>	<b>TipoTabella</b>	<b>Condizione</b>
#c-=  YACME manuale tabella-condizionale	#c-=	//YACME/manuale/tabella-condizionale

*Esempio 14: Esempio di tabella condizionale con codice #c!=*

<b>Tabella</b>	<b>TipoTabella</b>	<b>Condizione</b>
Nometabella>	Tipo>	Condizione>

`#c-!||YACME|manuale|tabella-condizionale-del`

*Esempio 15: Risultato*

Tabella	TipoTabella	Condizione
#c-!  YACME manuale tabella-condizionale-del	#c-!	//YACME/manuale/tabella-condizionale-del

### 4.1.6 Tabelle multiple

#### Monolivello.

Sono tabelle che vengono replicate in base al numero di occorrenze di un nodo all'interno dell'albero XML.

Analogamente alle tabelle condizionali, le tabelle multiple vengono definite da un codice da anteporre al nome della tabella, il quale a sua volta è l'X-Path dell'elemento. Questo tipo di funzionalità è utile, ad esempio nel costruire il modello di una fattura dove l'intestazione è fissa e quello che cambia è il numero di righe degli elementi.

Affinché X-Build identifichi questo tipo di tabella, la prima parte del nome dovrà essere **#n-**.

Quando si vuole ottenere una struttura che si replichi regolarmente occorre usare questo tipo di tabella.

In Esempio 16 è mostrato il file XML utilizzato per spiegare questa funzionalità.

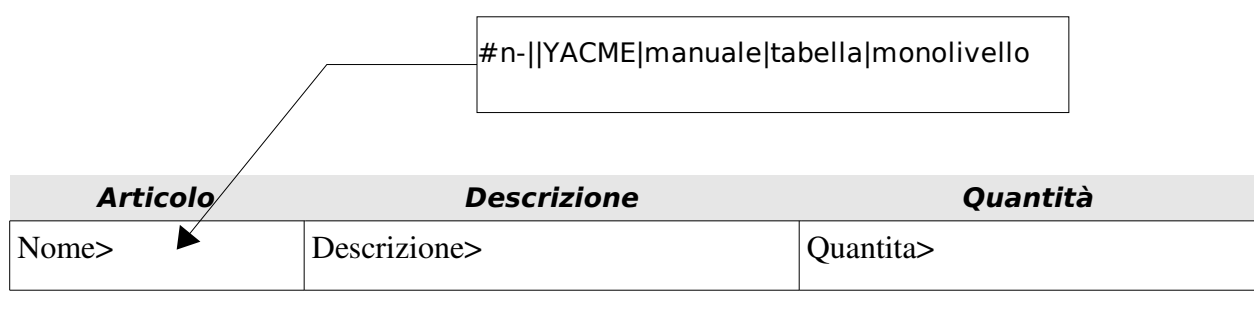
*Esempio 16: Esempio tabelle multilivello – file XML*

```

<?xml version="1.0" encoding="UTF-8"?>
<YACME>
  <manuale>
    <tabella>
      <monolivello>
        <articolo>YarenderHS</articolo>
        <descrizione>Yarender high speed</descrizione>
        <quantità >1</quantità >
      </monolivello>
      <monolivello>
        <articolo>YarenderLS</articolo>
        <descrizione>Yarender low speed</descrizione>
        <quantita>1</quantita>
      </monolivello>
    </tabella>
  </manuale>
</YACME>

```

Una parte del nostro modello di fattura è visualizzato nell'Esempio 17 (la legenda contiene il nome della tabella).

*Esempio 17: esempio di tabella multi livello in un modello*

Il risultato ottenuto con X-Build è quello di Esempio 18.

*Esempio 18: Risultato elaborazione X-Build*

<b>Articolo</b>	<b>Descrizione</b>	<b>Quantità</b>
mystiqueXMLHS	mystiqueXML high speed	1
mystiqueXMLLS	mystiqueXML low speed	1

### **Multilivello.**

Con l'utilizzo di questo tipo di tabelle è possibile creare una struttura a più livelli/sezioni (*header*, *body*, *footer*).

Ogni livello è rappresentato da una tabella che viene raggruppata, ordinata e replicata in base al numero di occorrenze e, eventualmente, ad una condizione di un nodo all'interno dell'albero XML.

Come sempre l' XPath relativo al nodo è ricavato dalla seconda parte del nome della tabella e il codice da anteporre può essere:

- **#noh-** , tabella *header*;
- **#nob-** , tabella *body*;
- **#nof-** , tabella *footer*.

Nella seconda parte del nome (XPath) di tabelle *header* o *footer* è possibile indicare la condizione che X-Build controllerà per sapere quante volte quella particolare tabella dovrà essere replicata. Nel caso di tabelle *body*, invece, il nome dovrà contenere un'ulteriore condizione che il software controllerà per raggruppare e ordinare tutta la struttura.

La struttura "*multilivello*" non necessita di tutte e tre le parti: ovvero è possibile creare un blocco formato solo da *header* e *body* o solo da *body* e *footer*.

L'utilità di questa struttura la si nota se si pensa ad un documento di trasporto in cui gli articoli (contenuti nel *body*) prima di essere inseriti dovranno essere raggruppati per destinazione (*header*).

Per approfondire vedremo di seguito tre casi che rispecchiano queste esigenze: un modello di ordine, di bolla e di estratto conto.

Viene presentato prima il file XML, di seguito una parte del modello contenente le tabelle ed infine il risultato dell'elaborazione da parte di X-Build.

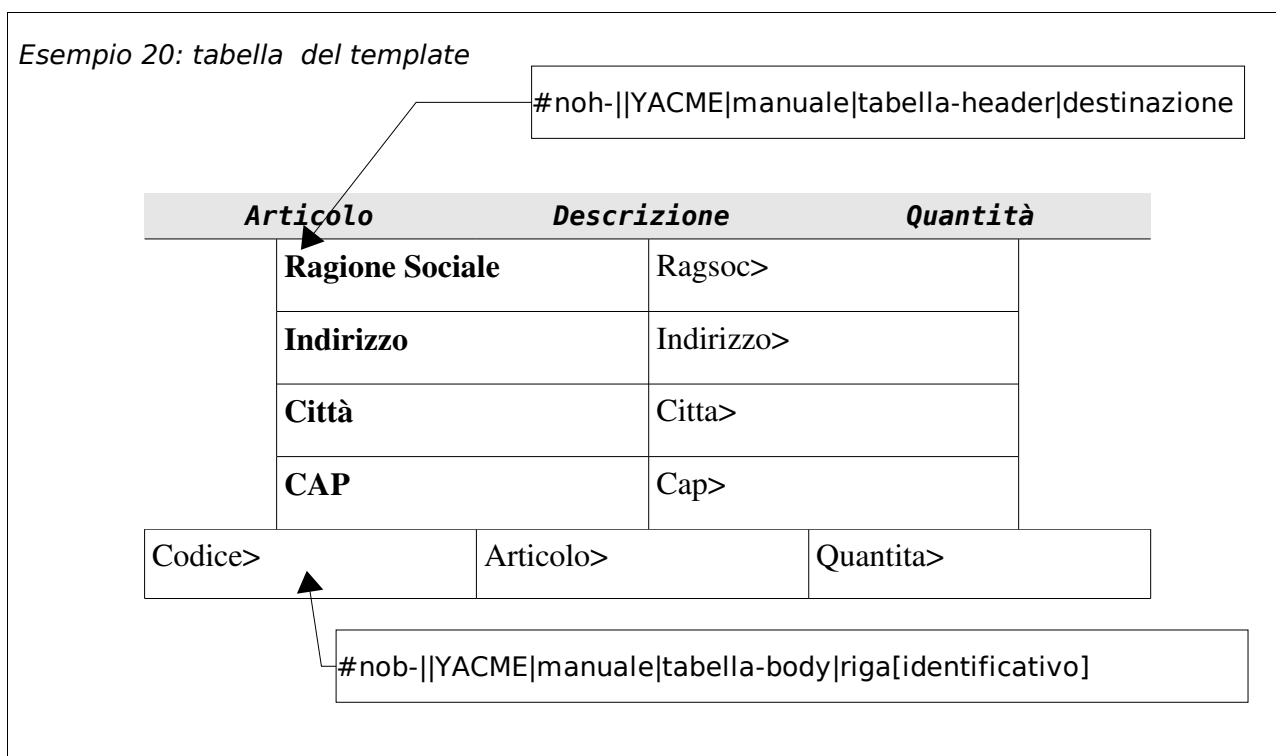
## ORDINE

Il file XML (Esempio 19).

*Esempio 19: File XML per esempio di ordine (prima parte)*

```
<?xml version="1.0" encoding="UTF-8"?>
<YACME>
  <manuale>
    <tabella-header>
      <destinazione>
        <numero>1</numero>
        <ragione-sociale>Nulla s.p.a.</ragione-sociale>
        <indirizzo>Via Garibaldi, 1</indirizzo>
        <citta>Bologna</citta>
        <cap>40100</cap>
      </destinazione>
      <destinazione>
        <numero>2</numero>
        <ragione-sociale>Niente s.r.l.</ragione-sociale>
        <indirizzo>Via Mazzini, 15</indirizzo>
        <citta>Bologna</citta>
        <cap>40100</cap>
      </destinazione>
    </tabella-header>
    <tabella-body>
      <riga>
        <identificativo>1</identificativo>
        <codice>NU001</codice>
        <articolo>>nulla 001</articolo>
        <quantita>10</quantita>
      </riga>
      <riga>
        <identificativo>2</identificativo>
        <codice>NIE001</codice>
        <articolo>>nulla 001</articolo>
        <quantita>15</quantita>
      </riga>
      <riga>
        <identificativo>2</identificativo>
        <codice>NU012</codice>
        <articolo>>nulla 012</articolo>
        <quantita>4</quantita>
      </riga>
    </tabella-body>
  </manuale>
</YACME>
```

Lo stralcio di modello contenente le tabelle (Esempio 20).



Risultato dell'elaborazione (Esempio 21).

Esempio 21: Risultato elaborazione da parte di X-Build

Articolo	Descrizione	Quantità
	<b>Ragione Sociale</b>	NIENTE s.r.l.
	<b>Indirizzo</b>	Via Mazzini, 1
	<b>Città</b>	Bologna
	<b>CAP</b>	40100
NIE001	niente 001	15
	<b>Ragione Sociale</b>	NULLA s.p.a.
	<b>Indirizzo</b>	Via Garibaldi, 1000
	<b>Città</b>	Bologna
	<b>CAP</b>	40100
NU001	nulla 001	10
NU012	nulla 012	4

Confrontiamo il file XML, il modello e il risultato finale.

In particolare, osserviamo:

- nel modello, l'inserimento dei codici **noh** e **nob** nei nomi delle tabelle
- nel documento finale
  - x la tabella *header* con le informazioni dei due destinatari;
  - x la tabella *body* con i dati dei singoli articoli.

X-Build, grazie all'impostazione dell'elemento identificativo come condizione nel nome della tabella di tipo body (`#nob-||YACME|manuale|tabella-body|riga[identificativo]`), raggruppa automaticamente gli articoli suddivisi per destinatario identificando la relazione in base agli elementi individuati dall'espressione

`||YACME|manuale|tabella-body|riga|identificativo` e `|YACME|manuale|tabella-header|destinazione|numero`.

I dati contenuti nella tabella *header* compaiono una sola volta mentre nelle tabelle *body* raggruppate sono ripetute le righe inerenti i singoli articoli.

## **BOLLA** (o documento di trasporto)

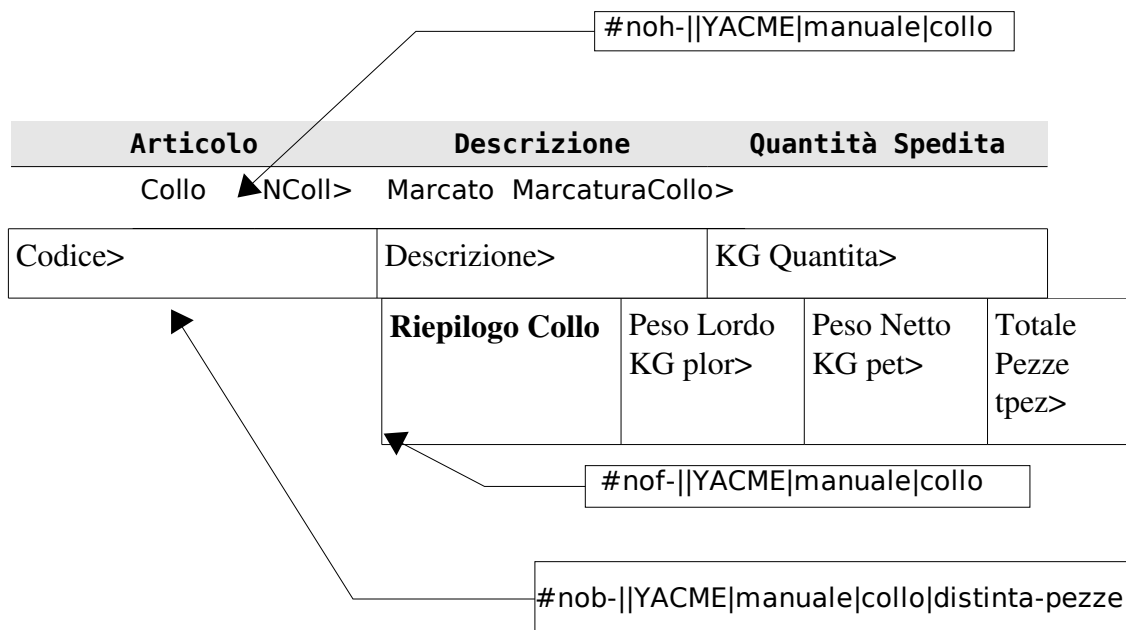
Il documento XML associato in Esempio 22.

*Esempio 22: File XML per esempio di bolla (prima parte)*

```
<?xml version="1.0" encoding="UTF-8"?>
<YACME>
  <manuale>
    <collo>
      <header>
        <numero>1</numero>
        <marcatatura>Ex001</marcatatura>
      </header>
      <footer>
        <peso-lordo>22</peso-lordo>
        <peso-netto>20</peso-netto>
        <totale-pezzo>2</totale-pezzo>
      </footer>
      <distinta-pezzo>
        <codice>NU001</codice>
        <descrizione>>nulla 001</descrizione>
        <quantita>10</quantita>
      </distinta-pezzo>
      <distinta-pezzo>
        <codice>NU010</codice>
        <descrizione>>nulla 010</descrizione>
        <quantita>12</quantita>
      </distinta-pezzo>
    </collo>
    <collo>
      <header>
        <numero>2</numero>
        <marcatatura>Ex002</marcatatura>
      </header>
      <footer>
        <peso-lordo>15</peso-lordo>
        <peso-netto>12</peso-netto>
        <totale-pezzo>1</totale-pezzo>
      </footer>
      <distinta-pezzo>
        <codice>NU012</codice>
        <descrizione>>nulla 012</descrizione>
        <quantita>1</quantita>
      </distinta-pezzo>
    </collo>
  </manuale>
</YACME>
```

La parte del modello utilizzato (Esempio 23).

Esempio 23: Esempio di tabella nel modello



Il risultato dell'elaborazione in Esempio 24.

Esempio 24: Risultato elaborazione da parte di X-Build

Articolo	Descrizione	Quantità Spedita		
Collo 1	Marcato EX001			
NU001	nulla 001	KG 10		
NIE010	niente 010	KG 12		
	<b>Riepilogo Collo</b>	Peso Lordo KG 22	Peso Netto KG 20	Totale Pezze 2
Collo 2	Marcato EX002			
NU012	nulla 012	KG 15		
	<b>Riepilogo Collo</b>	Peso Lordo KG 15	Peso Netto KG 12	Totale Pezze 1

In questo caso X-Build individua i tre elementi *header*, *body* e *footer* e:

- raggruppa gli articoli in base ai colli di appartenenza
- crea la struttura intestazione, righe degli articoli e riepilogo;
- ripete le righe della tabella *body* in base al numero di nodi `<distintamente>`;
- inserisce i valori corrispondenti ai segnalibri nelle tabelle opportune.

## ESTRATTO CONTO

Il file XML in Esempio 25.

*Esempio 25: esempio file XML per estratto conto*

```
<?xml version="1.0" encoding="UTF-8"?>
<YACME>
  <manuale>
    <estratto-conto>
      <partita-contabile>
        <movimento-partita>
          <causale>Nostra Fattura </causale>
          <numero-doc>999</numero-doc>
          <creazione-doc> 2003-12-05 </creazione-doc>
          <scadenza-doc>2004-03-31</scadenza-doc>
          <importo>1010.10</importo>
          <segno value="D"/>
        </movimento-partita>
        <movimento-partita>
          <causale>Ricevuta Bancaria</causale>
          <numero-doc>000</numero-doc>
          <creazione-doc>2004-02-18 </creazione-doc>
          <importo>1010.10</importo>
          <segno value="A"/>
        </movimento-partita>
        <footer>
          <numero-movimenti>2</numero-movimenti>
          <importo>1010.10</importo>
          <numero-partita>999</numero-partita>
          <anno-partita>2003</anno-partita>
          <segno value="A"/>
        </footer>
      </partita-contabile>
    </estratto-conto>
  </manuale>
</YACME>
```

La parte di modello interessata(Esempio 26).

*Esempio 26: Esempio di tabella nel modello*

Operazione	Data	Num.	Scadenza	DARE	AVERE
DescrizioneCausPartita>	DataCreazioneDoc>	NumDocumento>	ScadenzaDocumento>	ImportoPartitaMovimentoD>	ImportoPartitaMovimentoA>
			Saldo Partita	SaldoPartita>	SaldoPartita>
		NumDocumento>/AnnoPartita>			

Diagram showing relationships between table elements and XML paths:

- Arrow from `#nob-||YACME|manuale|estratto-conto|partita` to `DescrizioneCausPartita>`
- Arrow from `#nob-||YACME|manuale|estratto-conto|partita|movimento-partita` to `NumDocumento>/AnnoPartita>`

Il risultato dell'elaborazione di X-Build in Esempio 27.

*Esempio 27: documento finale generato da X-Build*

<b>Operazione</b>	<b>Data</b>	<b>Num.</b>	<b>Scadenza</b>	<b>DARE</b>	<b>AVERE</b>
Nostra Fattura	05/12/2003	999	31/03/2004	1010,1	
Ricevuta Bancaria	18/02/2004	10			1010,1
			Saldo Partita 999/2003	1010,1	

Come si può osservare in questo caso non abbiamo tutte le tipologie della struttura multilivello, ma solo tabelle *body* e *footer*. Analogamente al caso precedente, abbiamo una ripetizione delle righe della tabella *body* (per ogni elemento <movimento-partita>) relative ai movimenti contabili ed una sola per gli elementi della tabella *footer* con il riepilogo dell'estratto conto.

## 5 X-Template

X-Template è l'interfaccia che guida l'utente nella realizzazione del modello di documento, visualizzando da una parte il file XML e dall'altra un riepilogo dei segnalibri e delle tabelle inserite nel modello.

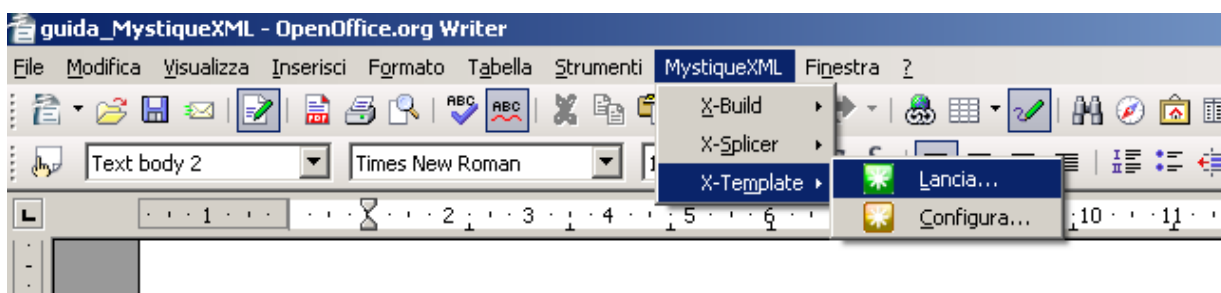


Figura 26: menù principale mystiqueXML

Per accedere a X-Template selezionare mystiqueXML -> X-Template -> Lancia (Figura 26).

Il sistema mostrerà finestra in Figura 25.

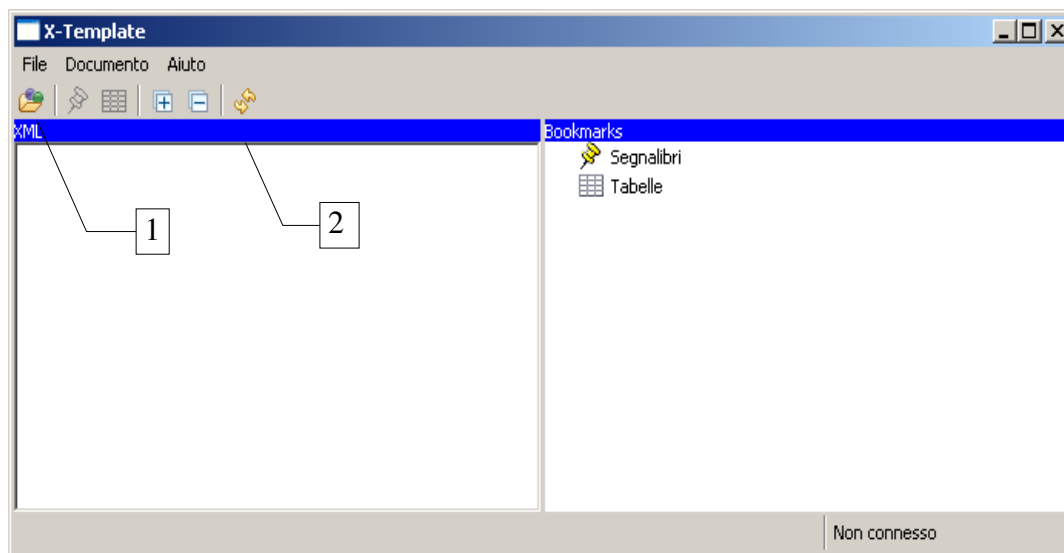


Figura 27: La finestra principale di X-Template

Questa è la finestra di partenza, a destra avremo la visualizzazione del file XML e dall'altra i segnalibri e le tabelle contenute nel documento attivo.

**1** Tramite questo pulsante si seleziona il *file* XML da associare al modello ODT attivo; dopo l'apertura, nella parte sinistra dell'immagine avremo la rappresentazione ad albero del *file* XML scelto.

**2** tramite questo bottone è possibile “connettersi” al modello aperto. X-Template mostrerà nella parte destra della pagina le tabelle e i segnalibri presenti nel documento.

Durante l'operazione di connessione verranno chiesti i dati di connessione (quelli preimpostati sono corretti se non si è modificata la configurazione di mystiqueXML) (Figura 28).

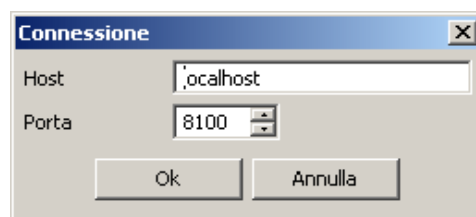


Figura 28: Finestra connessione OpenOffice.org

Analizziamo il menù messo a disposizione e quindi le operazioni che possiamo effet-

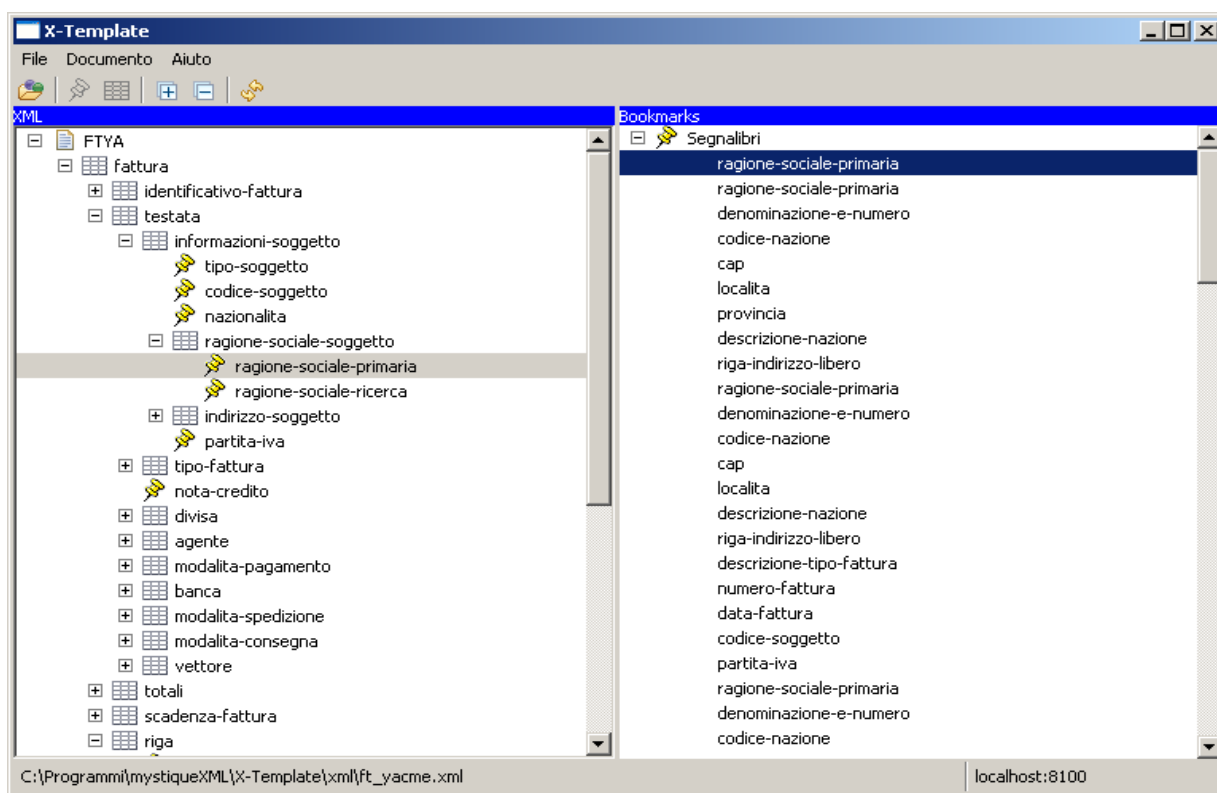


Figura 29: Finestra principale di X-Template

tuare con riferimento alle figure 29 e 30.

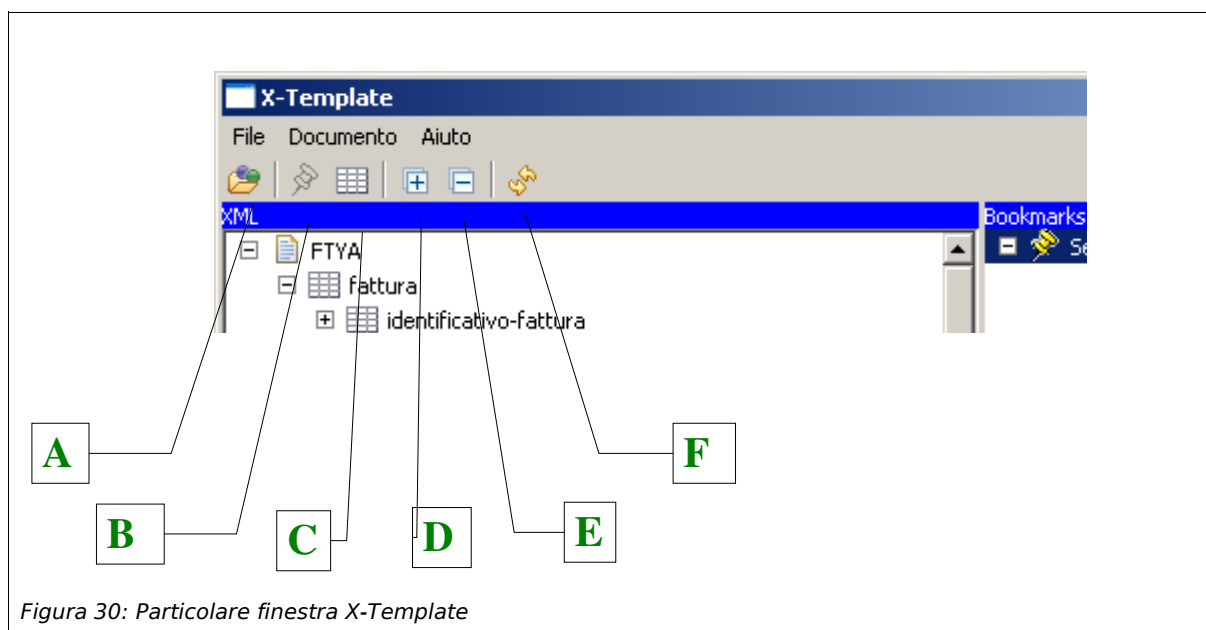


Figura 30: Particolare finestra X-Template

## A Apri file XML

Tramite questa icona possiamo selezionare il file XML da associare al modello attualmente aperto.

## B Inserisci segnalibro

Una volta selezionato l'elemento tra i nodi dell'albero XML (a sinistra della finestra), consente l'inserimento di un segnalibro nel documento.

La Figura 31, a lato, mostra la finestra che apparirà cliccando sull'icona e che permette di impostare le proprietà del nuovo segnalibro, per cui potete aggiungere i codici della sezione 4.1.2 (Segnalibri avanzati) in modo più intuitivo che non l'utilizzo delle lettere dei codici.

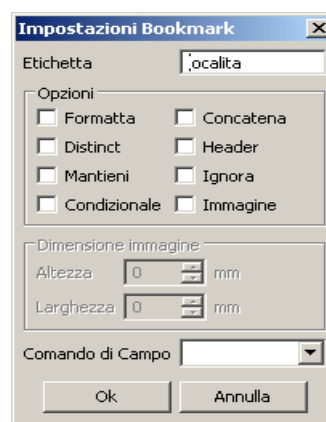


Figura 31: Opzioni dei segnalibri

## C Inserisci tabella

Cliccando su questa icona è possibile aggiungere tabelle all'interno del documento attivo. La finestra che apparirà (Figura 32) consente di specificare, oltre al numero di colonne e di righe, il codice per definire il tipo di tabella da inserire.

## D Espandi tutti

Espande tutti i nodi dell'albero XML e le sezioni del navigatore a destra.

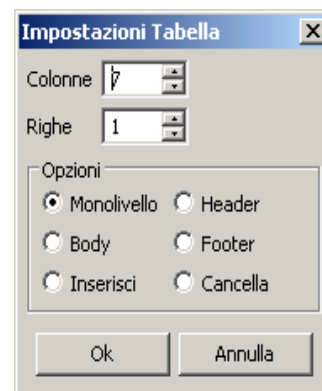


Figura 32: Opzioni tabella

## **E Collassa tutti**

Chiude i nodi dell'albero XML e le sezioni del navigatore a destra.

## **F Aggiorna navigatore**

Aggiorna le strutture dopo le modifiche permettendo così la visualizzazione delle operazioni effettuate.

## 6 X-Splicer

X-Splicer è un **Parser**, scritto in Python e integrato in OpenOffice.org che analizza il modello ed estrae le parti necessarie per la generazione del documento finale.

Come mostra l'immagine in Figura 33 è possibile lanciare X-Splicer sia sul documento attivo che su più file simultaneamente. Questa è la prima operazione da effettuare perché consente di configurare le directory che contengono i modelli.

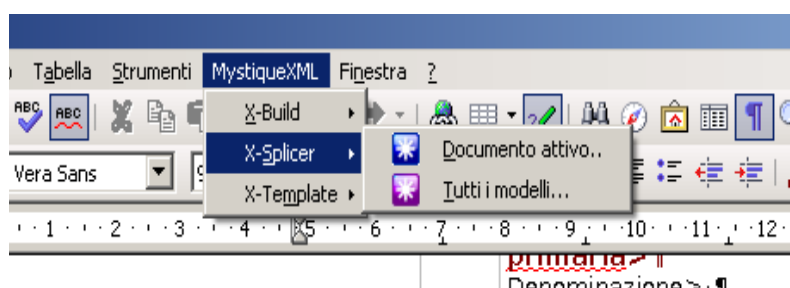


Figura 33: Menù X-Splicer

Selezionando il menù *mystiqueXML* -> *X-Splicer* -> *Documento attivo...* il programma mostrerà una finestra di avviso (Figura 34) per informare l'utente della prossima operazione.

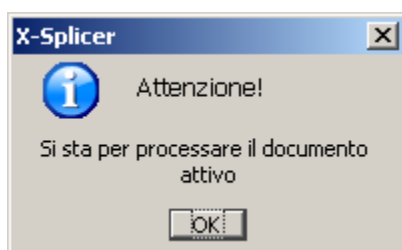


Figura 34: Avviso di processing

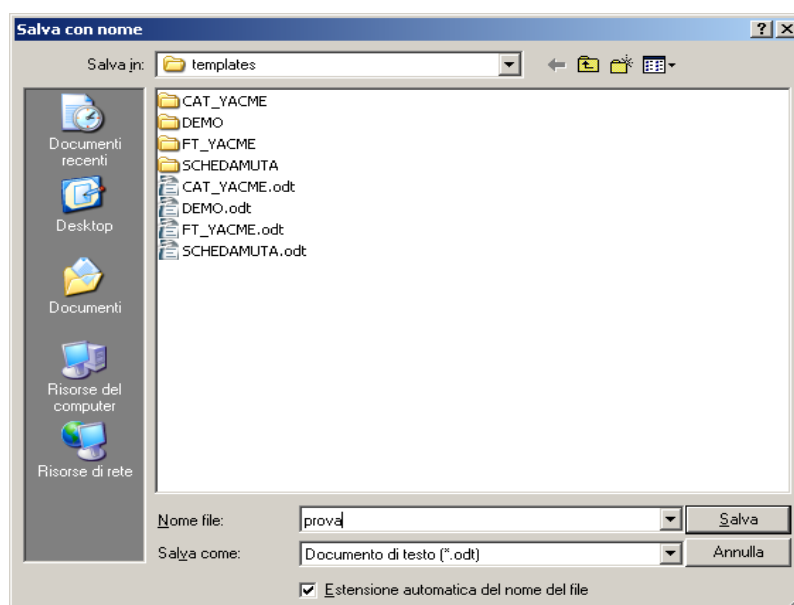


Figura 35: Finestra di selezione del modello

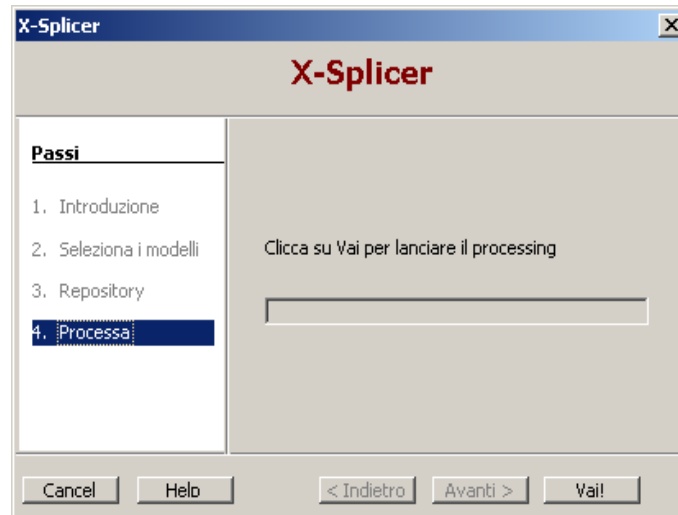


Figura 36: Lancia il parsing dei modelli

Successivamente, verrà mostrata la finestra (Figura 35) che consente di salvare il documento nella apposita cartella.

X-Splicer mostrerà poi la finestra in Figura 36 e cliccando sul tasto "Vai!" si avvierà il processo di parsing.

Nel caso in cui sceglieste di eseguire X-Splicer su più file (*mystiqueXML -> X-Splicer -> Tutti i modelli...*) sarà visualizzato il wizard che vi guiderà nel processo.

La Figura 37 mostra il primo passo che consente di impostare la directory che contiene i modelli. Il bottone **1** permette di sfogliare le cartelle del proprio PC.

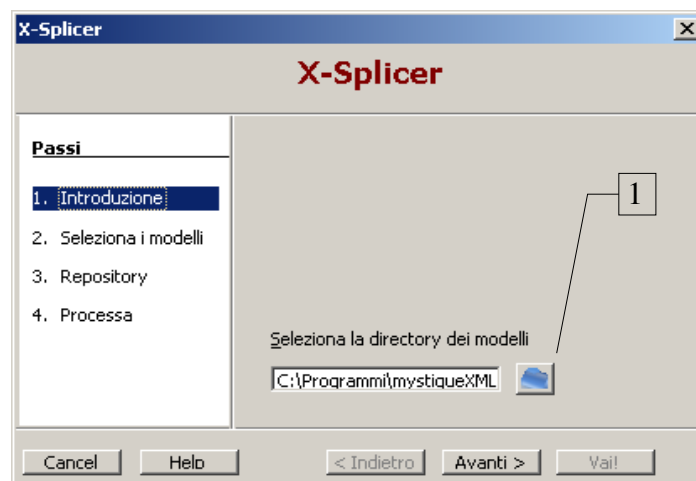


Figura 37: Scelta della cartella dei modelli

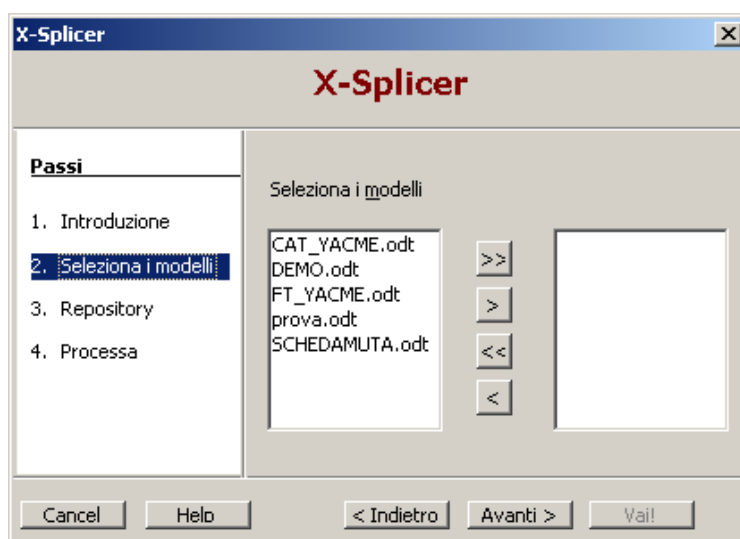


Figura 38: Selezione dei modelli

Il passo successivo (Figura 38) consente di scegliere i documenti su cui lanciare il parser fra quelli presenti nella cartella scelta. Nella lista a sinistra ho quelli a disposizione, in quella di destra i modelli scelti. I bottoni:

- >> consente di scegliere tutti i modelli (dalla lista di sinistra);
- > porta nella lista di destra i modelli selezionati;
- << elimina dalla lista (di destra) tutti i modelli;
- < elimina dalla lista (di destra) solo i modelli selezionati.

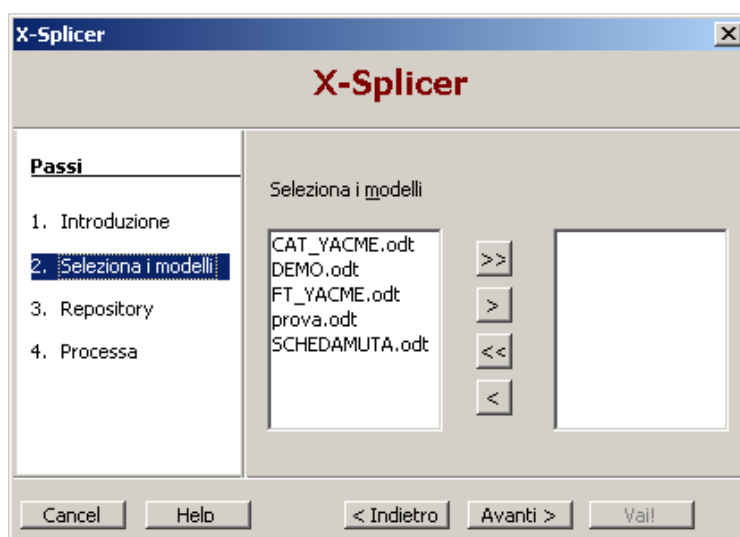


Figura 39: Secondo passo: selezione del modello

Il passo successivo () imposta il *repository*, ovvero l'area dati dove salvare le parti del modello estratte da X-Splicer. E' possibile scegliere una cartella

sul proprio PC<sup>5</sup> o una database collegato ad OpenOffice.org<sup>6</sup>.

L'ultimo passo è quello già visto in Figura 36 e consente di avviare il processo di parsing.

## 7 X-Build

X-Build è la libreria Java che trasforma il flusso XML e il modello creando il documento finale.

All'interno dell'addon per OpenOffice.org è stato realizzato un wizard che consente di lanciare la generazione di un documento scegliendo un file XML (o una ricerca salvata nelle sorgenti dati registrate in OpenOffice.org) e un modello di documento precedentemente *processato* da X-Splicer.

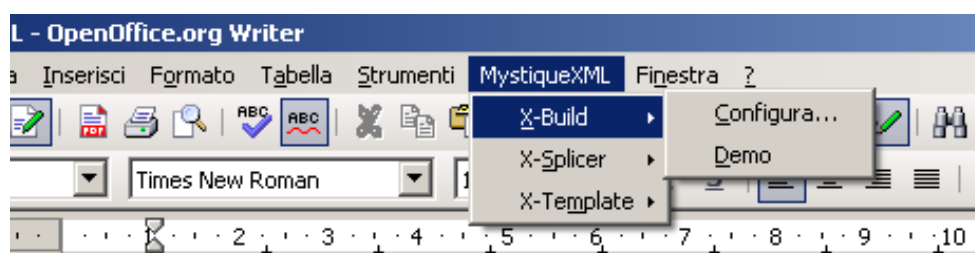


Figura 40: Menù di X-build

La generazione del documento si avvia utilizzando il menù *mystiqueXML* - > *X-Build* - *Demo*.

---

5 Se opportunamente configurata, è possibile utilizzare un'area esportata da un server remoto con un protocollo Samba o NFS.

6 Il supporto per memorizzare i modelli in un database sta per essere terminato. E' possibile già salvare le parti estratte in apposite tabelle, ma manca un modulo per X-Build che le recuperi in fase di generazione del documento.

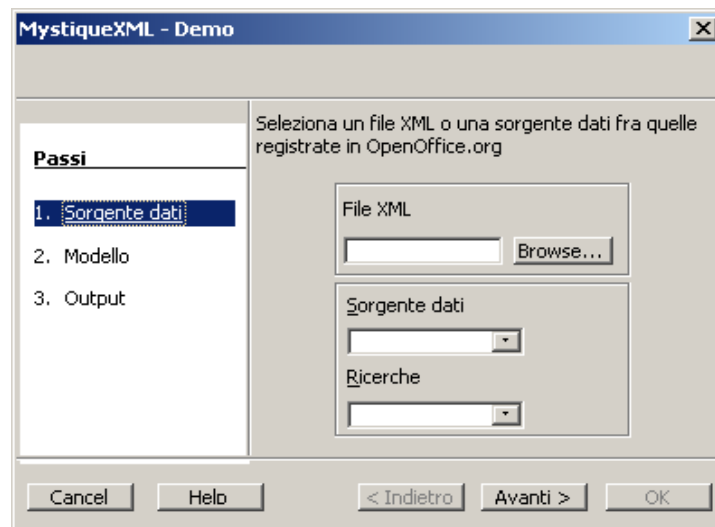


Figura 41: Primo passo: selezione della sorgente dati

Il primo passo consente di scegliere il file XML per comporre il documento come da Figura 41. Selezionando invece una sorgente dati e una ricerca associata è possibile creare un file XML a *runtime* per generare il documento. Questa operazione utilizza uno schema XML predefinito e il risultato è un flusso come quello mostrato in Esempio 28.

*Esempio 28: XML per l'estrazione dei dati da un DB*

```
<?xml version="1.0" encoding="UTF-8"?>
<nome_database>
  <nome_ricerca>
    <riga>
      <campo1>valore</campo1>
      <campo2>valore</campo2>
      <campo3>valore</campo3>
      <campo4>valore</campo4>
      <campo5>valore</campo5>
      ...
    </riga>
    <riga>
      ...
    </riga>
  </nome_ricerca>
</nome_database>
```

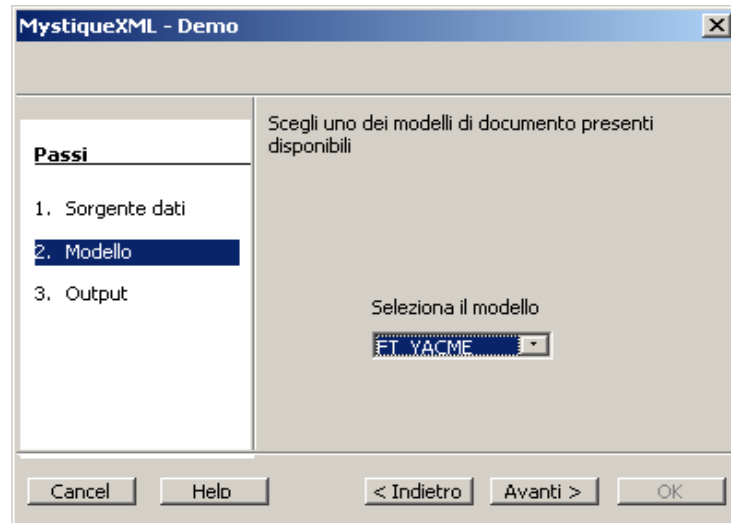


Figura 42: Selezione del modello

Il passo successivo (Figura 42) consente di scegliere il modello da utilizzare per il documento.

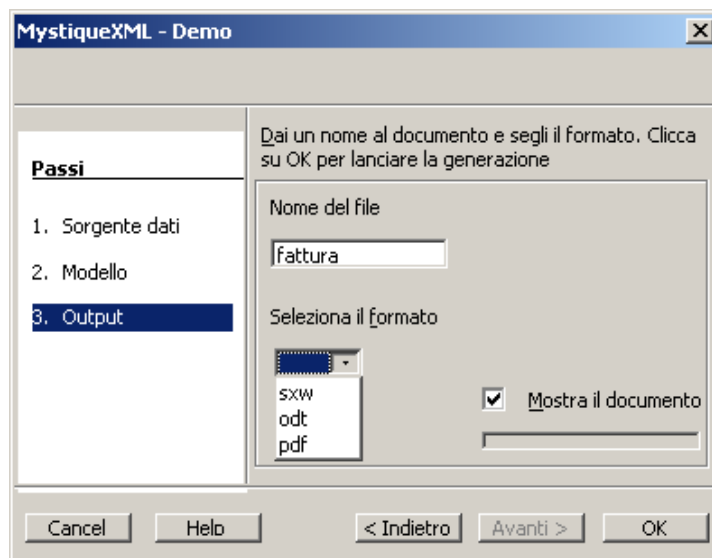


Figura 43: Terzo passo: definizione documento finale

L'ultimo step consente di impostare il nome del documento finale e il suo formato (Figura 43). Il flag “Mostra documento” consente l'apertura automatica del file dopo la sua generazione (se si sceglie uno dei formati di OpenOffice.org, il documento verrà aperto in sola lettura con Writer, altrimenti utilizzando il viewer PDF impostato nella configurazione).

Il bottone “OK” avvia la generazione del documento.

## 8 APPENDICE A

### Requisiti di Sistema per OpenOffice.org 2.0

#### **Microsoft Windows**

- Windows 98, Windows ME, Windows 2000 (Service Pack 2 o superiore), Windows XP, Windows 2003
- 128 Mbytes RAM
- 200 Mbytes di spazio disponibile su disco
- 800 x 600 o risoluzione superiore con almeno 256 colori

#### **Solaris: SPARC platform edition**

- Solaris 8 OS o più recente
- 128 Mbytes RAM
- 250 Mbytes di spazio disponibile su disco
- X-Server con 800 x 600 o risoluzione superiore con almeno 256 colori

#### **Solaris: x86 platform edition**

- Solaris 8 OS o più recente
- 128 Mbytes RAM
- 250 Mbytes di spazio disponibile su disco
- X-Server with 800 x 600 o superiore con almeno 256 colori

#### **Linux:**


- Linux kernel versione 2.2.13 or superiore, glibc2 versione 2.2.0 o superiore
- 128 Mbytes RAM
- 200 Mbytes di spazio disponibile su disco
- X-Server con 800 x 600 o risoluzione superiore con almeno 256 colori

## 9 APPENDICE B: i segnalibri


I **Segnalibri** sono contrassegni interni e non visibili che, inseriti nel testo, permettono di ritrovare velocemente i punti di particolare interesse.

La funzione dei **Segnalibri** è collegata alla gestione del testo a video, sia per la possibilità di individuare subito il segnalibro voluto (con lo strumento **Navigatore**) sia per la possibilità di creare dei **Link ipertestuali** che portano al segnalibro.

Per inserire un segnalibro al punto in cui è posizionato il cursore:

- Menù **Inserisci** -> **Segnalibro**, assegnare un nome al segnalibro e chiudere.
- cliccare sul pulsante  della **Barra dei simboli/Inserisci** (per visualizzare questa barra, Menù **Visualizza** -> **Barre dei simboli** -> **Inserisci**)

Per eliminare un segnalibro:

- Menù **Inserisci** -> **Segnalibro** indipendentemente dal punto in cui si trova il cursore; evidenziare il nome del segnalibro da eliminare; premere il pulsante **Elimina** e poi il pulsante **Annulla** per evitare di inserire un segnalibro al punto di posizionamento del cursore.
- cliccare sul pulsante  e seguire la procedura descritta sopra.

## 10 APPENDICE C: le tabelle

Una tabella è una struttura organizzata a righe e colonne il cui elemento minimo, la cella, può contenere al suo interno diversi tipi di dati: testo, immagini, formule.

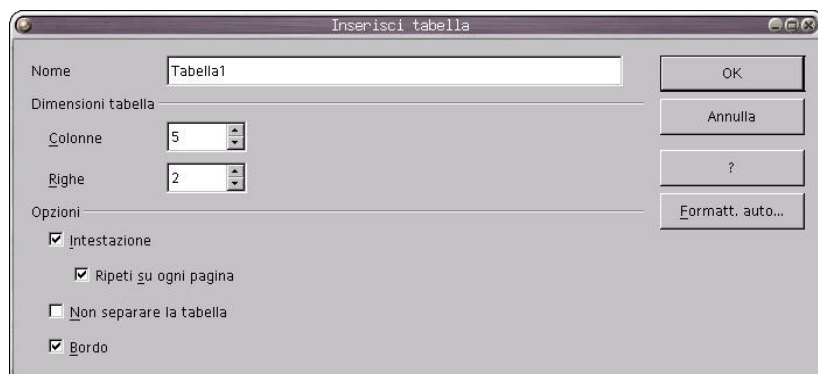
### Inserimento

Per decidere dove collocare una tabella è necessario spostare il cursore nella posizione desiderata.

Per inserire una tabella esistono diversi modi di procedere:

- Menù **Tabella** -> **Inserisci** -> **Tabella**

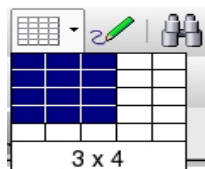
- CTRL+F12
- pulsante **Tabella**  della **Barra dei simboli/Standard**



Si apre la finestra di dialogo **Inserisci Tabella**:

- **Nome**: è il nome della tabella a cui si può fare riferimento attraverso il **Navigatore** (vedi oltre);
- **Dimensioni Tabella**: sono il numero di righe e colonne della tabella;
- **Intestazione**: consente di impostare una riga di intestazione, cioè un nome per i campi delle colonne;
  - **Ripeti su ogni pagina**: ripete la riga di intestazione in tutte le pagine nel caso in cui la tabella si sviluppa su più pagine;
- **Non separare la tabella**: evita che la tabella venga separata da un'interruzione di pagina;
  - **Bordo**: abilita la visibilità del bordo, deselezionare questa voce per avere celle senza bordo.

Un altro modo per inserire una tabella è usare il menù a comparsa che appare tenendo premuto a lungo il pulsante **Tabella** o cliccando nella piccola freccia accanto al pulsante.



Per effettuare l'inserimento occorre trascinare il mouse sulla griglia, tenendo premuto il tasto sinistro, per selezionare il numero di **righe/colonne** desiderato (spostare il mouse verso il basso per aumentare il numero delle righe, verso destra per aumentare il numero delle colonne).

**NOTA:** Nella figura sopra sono state selezionate 3 colonne e 4 righe.

Dopo l'inserimento, viene visualizzata una griglia (grigia se si è scelto di

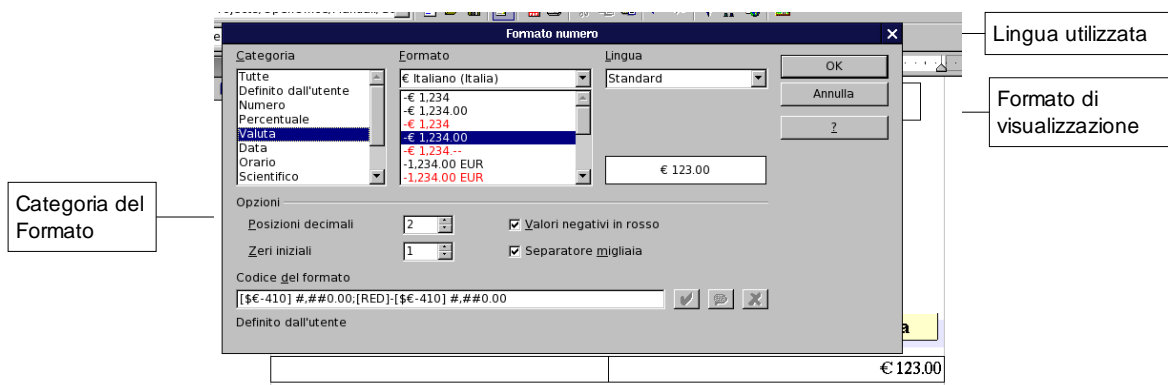
non inserire il bordo) con i margini della tabella e delle celle in cui è possibile effettuare gli inserimenti voluti.

### Inserimento di elementi testuali e grafici in una tabella

Per inserire un testo o un elemento grafico all'interno di una tabella è sufficiente cliccare nella cella selezionata e quindi inserire l'oggetto. L'altezza della cella si adegua automaticamente agli elementi inseriti. **È possibile aggiungere, eliminare e formattare il testo inserito in una cella utilizzando le normali procedure.**

Per formattare una o più celle occorre

- selezionare le celle interessate
- aprire la finestra di dialogo **Formato numero** o attraverso il menù contestuale (tasto destro del mouse, **Formato numero**) oppure da Menù **Tabella -> Formato Numero**)



Per muoversi da una cella ad un'altra si clicca sulla cella oppure si preme **TAB** per passare alla cella successiva o **SHIFT+TAB** per passare alla cella precedente.

Per inserire un carattere di tabulazione premere **CTRL+TAB**.

### Numeri e Calcoli

All'interno delle tabelle è possibile inserire numeri e Calcoli.

Per inserire un valore numerico è sufficiente cliccare sulla cella dove si vuole inserire il dato e digitare il numero.

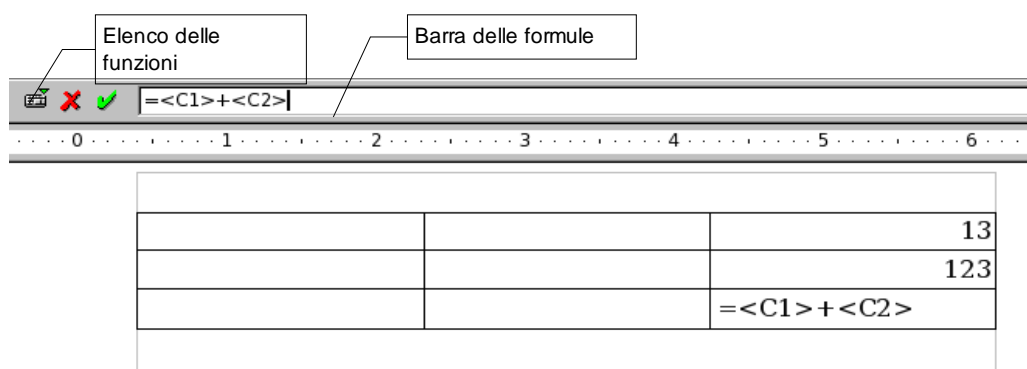
**NOTA:** Qualora la cella contenga un valore testuale, il valore numerico verrà trattato come una stringa e quindi non sarà possibile effettuare Calcoli

Per inserire una formula all'interno di una tabella occorre:

- cliccare sulla cella che deve contenere la formula;

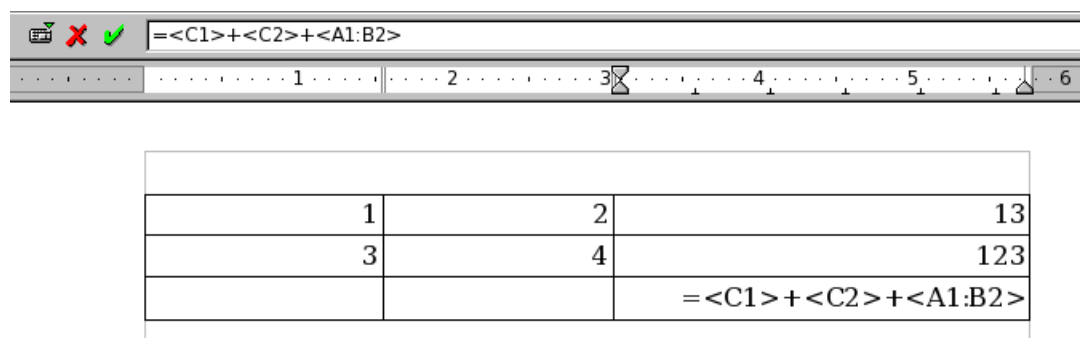
- digitare =, compare la barra delle formule ed un = all'interno della cella interessata;
- selezionare l'operando interessato (la cella) con un clic;
- digitare l'operatore (es. +) e quindi cliccare sull'altro operando.

Nella figura che segue è riportato l'esempio di una somma fra due celle.




Le celle sono identificate da una coppia di coordinate LetteraNumero racchiusa fra i simboli < e >: la lettera corrisponde alla colonna mentre il numero alla riga.

Nell'esempio che segue, invece, si può notare come sia possibile selezionare all'interno di una formula più celle. È sufficiente utilizzare due coordinate



divise dal simbolo matematico.

Oltre ai classici operatori +, -, /, \*, esiste una serie di operatori selezionabili attraverso il pulsante 

Inserita una funzione all'interno di una cella è possibile modificarla attraverso il tasto **F2**. Allo stesso modo, se la cella contiene testo, può essere sovrascritto con una formula o un numero semplicemente cliccandovi sopra e premendo il tasto **F2**.

## Modifica di una Tabella:

Nella **Barra dei simboli/Tabella** sono raccolti i pulsanti relativi ad alcune funzioni di formattazione della tabella.



*NOTA: questa barra apparire automaticamente quando si seleziona una tabella.*

Posizionando il cursore in una cella, con i pulsanti di eliminazione si possono cancellare le righe sottostanti al cursore o le colonne successive. La procedura è analoga per aggiungerne.

Inoltre, è possibile puntare sul bordo della colonna finché il puntatore non assume una forma diversa (una doppia freccia) e poi trascinare il bordo fino alla larghezza desiderata.

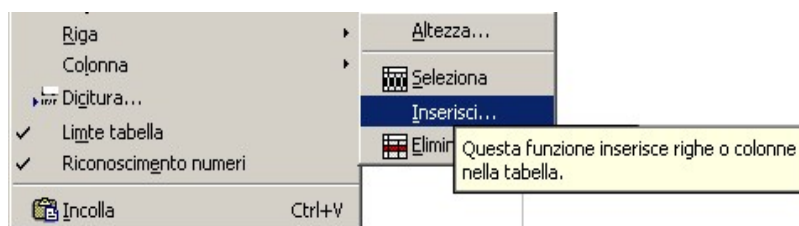
### Aggiunta di righe e colonne

Per aggiungere più di una riga o una colonna:

- selezionare il numero di righe o colonne che si desidera inserire;
- fare clic sul pulsante **Inserisci righe** o **Inserisci colonne** della **Barra dei simboli/Tabella**.

*NOTA: Le nuove righe verranno inserite sopra quelle selezionate e le nuove colonne a sinistra di quelle selezionate.*

Oppure, posizionare il cursore nel punto interno alla tabella dove si vuole che venga inserita la riga o la colonna, premere il tasto destro del mouse,



selezionare dal menù contestuale **Riga** o **Colonna** poi **Inserisci**.

Compare una finestra nella quale si può impostare il numero di righe o colonne, e la modalità di inserimento (**davanti** significa prima e **dietro** significa dopo la riga o la colonna correnti).

Per aggiungere una riga alla fine di una tabella:

- posizionare il punto di inserimento nell'ultima cella a destra e premere **TAB**.

## 11 GLOSSARIO

### **Body:**

letteralmente: corpo, intesa come parte centrale di un documento o di una sezione; di solito viene associato anche ai termini "header" e "footer", rispettivamente intestazione e piè di pagina.

### **Browser:**

un browser web (in italiano: navigatore) è un [programma](#) che consente agli utenti di visualizzare e interagire con testi, immagini e altre informazioni, tipicamente contenute in una [pagina web](#) di un [sito](#) , per approfondimenti:

<http://it.wikipedia.org/wiki/Browser>

### **CNIPA:**

Il Centro Nazionale per l'Informatica nella Pubblica Amministrazione (CNI-PA) opera presso la Presidenza del Consiglio per l'attuazione delle politiche del Ministro per le riforme e le innovazioni nella PA.

<http://www.cnipa.gov.it/site/it-IT/>

### **codice sorgente:**

Il codice sorgente altro non è che un file costituito da istruzioni che possono essere interpretate da un personal computer, un insieme di istruzioni costituisce un programma .

[http://it.wikipedia.org/wiki/Codice\\_sorgente](http://it.wikipedia.org/wiki/Codice_sorgente)

### **DEFAULT:**

Con questo termine si definisce una variabile od un campo a cui viene associato un particolare valore a meno che non ne venga specificato un altro.

[http://it.wikipedia.org/wiki/Default\\_%28informatica%29](http://it.wikipedia.org/wiki/Default_%28informatica%29)

### **flag:**

Letteralmente bandierina, indica in alcune accezioni, lo stato di una variabile o di un' etichetta che la funzione anche di "segnalare" il punto in cui

viene messa.

<http://it.wikipedia.org/wiki/Flag>

### **footer:**

Con questo termine viene definita la parte finale di un documento o di una tabella.

### **Framework:**

il framework è una struttura di supporto su cui un software può essere organizzato e progettato

<http://it.wikipedia.org/wiki/Framework>

### **header:**

intestazione: parte iniziale di un documento o di una tabella

### **Java:**

Il linguaggio Java è un [linguaggio di programmazione orientato agli oggetti](#), derivato dal [C++](#) (e quindi indirettamente dal [C](#)) e creato da [James Gosling](#) e altri ingegneri di [Sun Microsystems](#)

<http://www.java.com/it/>

### **Linux:**

Linux è un termine che ha diverse valenze, per convenzione quando non viene specificato altro si fa riferimento al [sistema operativo](#) basato su tale kernel.

<http://it.wikipedia.org/wiki/Linux>

### **namespace:**

in italiano spazio dei nomi, ha lo scopo di definire l'ambito di valenza di una variabile in modo da non rischiare confusione.

<http://it.wikipedia.org/wiki/Namespae>

### **NULL:**

Questa è l'etichetta che si usa per definire il caso in cui un dato campo non ha associato alcun valore.

<http://it.wikipedia.org/wiki/NULL>

**Open Source:**

In italiano "codice aperto" è una metodologia di sviluppo che da la possibilità all'utente finale di visionare il codice sorgente del software che utilizza, questo permette trasparenza e una migliore qualità del prodotto.

<http://www.opensource.org/>

**OpenOffice.org:**

E' una suite di programmi di ufficio *Open Source* che permette la creazione e la modifica di testi, fogli di calcolo, data-base e presentazioni.

<http://it.openoffice.org/>

**Parser:**

E' un analizzatore sintattico ossia verifica che il file o i files siano scritti seguendo precise regole sintattiche.

<http://it.wikipedia.org/wiki/Parsing>

**Pop-up:**

E' una finestra del browser aperta da un'altra finestra senza necessariamente avere effettuato operazioni specifiche. E' sfruttato come mezzo pubblicitario all'apertura di una qualsivoglia pagina.

**Python:**

E' un linguaggio di programmazione definito di scripting pseudocompilato, ossia necessita di un interprete per essere utilizzato, ma non ha bisogno di essere compilato come accade con altri linguaggi di programmazione con il C++.

<http://www.python.it/>

**query:**

in italiano domanda, con questo vocabolo si definisce una richiesta al un data-base, la quale ritornerà zero o più record.

**Sistema operativo:**

E' il programma che permette di utilizzare il computer facendo dialogare i vari componenti.

[http://it.wikipedia.org/wiki/Sistema\\_operativo](http://it.wikipedia.org/wiki/Sistema_operativo)

**Software:**

in italiano si può tradurre con "programma"

<http://it.wikipedia.org/wiki/Software>

**SQL DISTINCT:**

E' una tipologia di query che visualizza una sola occorrenza per ogni record non visualizzando eventuali doppi.

[http://www.w3schools.com/sql/sql\\_select.asp](http://www.w3schools.com/sql/sql_select.asp)

**W3C:**

Il World Wide Web Consortium (W3C) è un consorzio che promuove l'interoperabilità tra sviluppatori tramite la definizione di standard comuni.

<http://www.w3.org/>