



Stampare con Java facile e gratuito

di Andrea Musumeci
<http://www.javalab.it>

Java™
Italian
Portal

www.JavaPortal.it



Stampare con Java: facile e gratuito

- Relatore: Andrea.Musumeci@javalab.it
- Data: Padova 07.05.04
- Area: Programming & Development
- Target: Tecnici
- Difficoltà: media

Il seminario si propone di fornire le basi per utilizzare la tecnologia open source del progetto Apache-Jakarta FOP che permette di gestire i formati delle stampe in java; in particolare ci occuperemo delle stampe in PDF. Durante il seminario verrà presentato un programma gestionale freeware per la fatturazione di consulenze di liberi professionisti.



Obiettivi

Dopo aver completato questo seminario sarai in grado di:

- Descrivere le tecnologie esistenti per la stampa in Java
- Descrivere le tecnologie coinvolte in una stampa con FOP
- Creare un semplice gestionale con funzionalità di salvataggio dati in XML e stampa in formato PDF



Tecnologie di stampa esistenti in Java

- `java.awt.print` – un insieme di tre interfacce e quattro classi che permettono di descrivere dei documenti da mandare direttamente in stampa
- `IText` – classi free che permettono di descrivere degli oggetti per stampare anche in formato PDF
- `JasperReport` – un framework creato per collegare direttamente i risultati di una query su database ad un template formattato in xsl
- `FOP` – Driver di trasformazione di documenti XML in grado di esportare in PDF sposando il paradigma MVC



FOP

- Formatting Object Protocol
- Grazie a The Apache Software Foundation e il team di sviluppatori del Apache XML Project è stato creato un Driver di trasformazione XML in grado di stampare su vari formati tra i quali PDF, PCL, RTF, SVG, XML, PRINT, AWT e TXT.
- Il principale output è PDF.
- Punti di forza: l'insieme delle features di Java e XML
- Tecnologie coinvolte:
XML – XPATH –
XSL e FO – JAVA





XML – introduzione

- eXtensible Markup Language
- è un linguaggio di definizione di strutture di dati
- è lo standard per definire dati su web
- supporta la definizione di tags personalizzati



XML – esempio

```
<?xml version='1.0' encoding='windows-1252'?>
```

```
<LISTACLIENTI>
```

```
  <CLIENTE id='11'>
```

```
    <NOME>CMC Srl</NOME>
```

```
    <VIA>via Mameli 6</VIA>
```

```
    <CITTA>28100 Novara</CITTA>
```

```
  </CLIENTE>
```

```
</LISTACLIENTI>
```



XML – caratteristiche [1 / 2]

- Pochi componenti di linguaggio:
 - Elemento: definisce un oggetto tramite start tag ed end tag [ad es: `<NOME>Cmc Srl</NOME>`]
 - Attributo: indica il nome di un attributo di un oggetto e il suo valore inserito tra apici [ad es: `<CLIENTE id='10'>`]
 - Elemento vuoto: un elemento senza valori [ad es: `<CAP valore='28100' />`]
 - Entità: un meccanismo per definire variabili di sostituzione in un documento XML [ad es. per scrivere un carattere di sintassi utilizzeremo una stringa particolare: il carattere `<` infatti si valorizza in `<`]



XML – caratteristiche [2 / 2]

- Case sensitive
- Poche regole per essere well-formed:
 - inizio del file di testo XML con definizione di documento XML
 - Tutti gli start tag devono avere una end tag (a meno che non si stia definendo un Elemento vuoto)
 - Presenza di una unica definizione di Elemento di root
 - Commenti tra `<!-- commento -->`
 - Nomi di elemento e attributo contenenti solo caratteri (e non numeri)
- Tree View testabile con un browser



XPATH [1 / 2]

- XML Path Language – linguaggio per puntare o filtrare nodi o foglie di un albero XML
- Disegnato per esser utilizzato con XSL o per interrogare basi di dati su XML DB (SQLX ad esempio)
- Case sensitive e tutto lower case
- Necessario per definire XPath Expressions su tipi di dato Nodo, Boolean, Number e String

ESEMPI:

elemento restituisce il valore di quell'elemento
nell'albero XML con nome 'elemento'

@attributo restituisce l'attributo con nome indicato
sul nodo corrente



XPATH [2 / 2]

- `x // y` restituisce il valore del primo elemento `y` partendo dal nodo `x`
- `/elemento[QE]` restituisce il valore dell'elemento se e solo se si verifica la Query Expression;
`es cliente/[nome='Gigi']` restituisce i dati del cliente se si chiama Gigi
- `position()` restituisce la posizione nell'insieme di nodi figli sullo stesso livello.
- `count()` restituisce il numero di nodi figli
- `concat()` restituisce come unico valore la somma di valori di due nodi passati come argomento



XSL – Introduzione

- eXtensible Stylesheet Language – linguaggio di programmazione che permette di definire fogli di stile in linguaggio XML
- Può essere utilizzato per trasformare o riordinare documenti XML
- Il suo interprete viene chiamato XSLTransformer (in Java – C – perl)
- Per effettuare trasformazioni di formattazione dei valori di XML si usa un'estensione di XSL detta :FO (formatting object)



XSL – Cenni di sintassi [1 / 2]

- Template: definisce uno stile per ogni elemento contenuto nell'albero XML da trasformare
- Manipolazione dati: si possono modificare i contenuti e l'ordine dei dati mappati nell'XML

```
<xsl:template match="FATTURA">...</xsl:template>
```

Puo' essere richiamato all'interno del file XSL tramite l'istruzione `<xsl:apply-templates/>`

```
<xsl:sort select="nome"/>
```

ordina gli elementi in base al valore di nome

- Condizionali

```
<xsl:if test="BE"> .. </xsl:if>
```

```
<xsl:choose>
```

```
<xsl:when test="BE"> .. </xsl:when>
```

...

```
<xsl:otherwise>... </xsl:otherwise>
```

```
</xsl:choose>
```



XSL – Cenni di sintassi [2/2]

- Cicli:

```
<xsl:for-each select="CLIENTE">
```

```
<tr>
```

```
<td>nome</td><td><xsl:value-of select="nome"/>
```

```
<td>cognome</td><td><xsl:value-of select="cognome"/>
```

```
</tr>
```

```
</xsl:for-each>
```

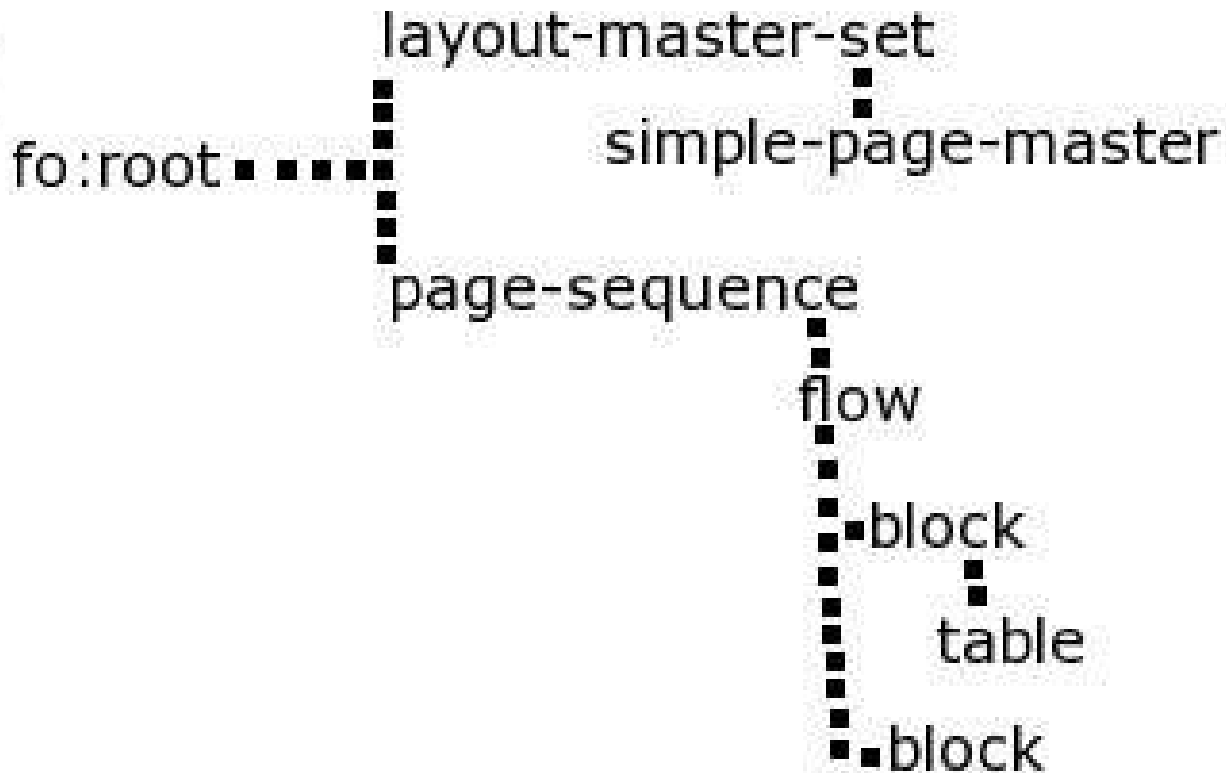


XSL:FO – Introduzione

- Estensione del linguaggio XSL che permette operazioni di Formatting Object
- Sintassi interpretabile da un driver FOP (Formatting Object Processor) per effettuare trasformazioni da XML tree view a PDF



XSL:FO – Tree View semplificata dei tag





XSL:FO – Cenni sui tag [1 / 2]

- a) fo:root – Nodo principale di un xsl formatting objects tree.
DDL: (layout-master-set,declarations?,page-sequence+)
- b) fo:layout-master-set – Contenitore di tutti gli oggetti master usati nel documento. DDL: (simple-page-master|page-sequence-master)+
- c) fo:simple-page-master – Elemento che descrive la struttura delle pagine che possono esser suddivise fino a cinque regioni. L'unica obbligatoria è la region-body (corpo della pagina)
DDL: (region-body,region-before?,region-after?,region-start?,region-end?)
- b) -fo:page-sequence – Elemento utilizzato per dichiarare come deve esser creato il corpo delle pagine del documento. Per esempio potrebbe esser pensato per un capitolo di un libro o nel nostro gestionale per le pagine della fattura.
DDL (title?,static-content*,flow)



XSL:FO – Cenni sui tag [2/2]

- c) fo:flow – Contenitore per gli oggetti (paragrafi) che costituiscono il flusso di dati da stampare
DDL: (block;)+
- d) fo:block – Comunemente utilizzato come elemento per formattare paragrafi, titoli, tabelle o immagini
DDL: (#PCDATA|inline;|table;|block;)*
- e) fo:table – Contenitore di una tabella
DDL: (table-column*,table-header?,table-footer?,table-body+)
- ...and so on :) ...



XSL:FO – Esempio

```
<?xml version="1.0" encoding="windows-1252" ?>
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:svg="http://www.w3.org/2000/svg">
  <xsl:output method="xml" indent="yes" />
- <xsl:template match="FATTURA">
  - <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    - <fo:layout-master-set>
      - <fo:simple-page-master margin-right="1.5cm" margin-left="1.5cm" margin-bottom="2cm" margin-top="1cm" page-
        width="21cm" page-height="29.7cm" master-name="first">
        <fo:region-before extent="1cm" />
        <fo:region-body margin-top="1cm" />
        <fo:region-after extent="1.5cm" />
      </fo:simple-page-master>
    </fo:layout-master-set>
  - <fo:page-sequence master-reference="first">
    - <fo:flow flow-name="xsl-region-body">
      - <fo:block margin-left="14pt" font-size="20pt" font-weight="bold" margin-right="3pt" space-before.optimum="3pt" text-
        align="left">
        <xsl:value-of select="INTESTAZIONE/NOME" />
      </fo:block>
      - <fo:block margin-left="3pt" font-size="12pt" margin-right="3pt" space-before.optimum="3pt" text-align="left">
        <xsl:value-of select="INTESTAZIONE/VIA" />
      </fo:block>
      - <fo:block margin-left="3pt" font-size="12pt" margin-right="3pt" space-before.optimum="3pt" text-align="left">
        <xsl:value-of select="INTESTAZIONE/CITTA" />
      </fo:block>
      - <fo:block margin-left="3pt" font-size="12pt" margin-right="3pt" space-before.optimum="3pt" text-align="left">
        <xsl:value-of select="INTESTAZIONE/CELLULARE" />
      </fo:block>
      - <fo:block margin-left="3pt" font-size="12pt" margin-right="3pt" space-before.optimum="3pt" text-align="left">
        <xsl:value-of select="INTESTAZIONE/CODFISCALE" />
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```



JAVA e FOP

Per utilizzare il Driver FOP da Java occorre importare le definizioni corrette, istanziare il Driver FOP e impostare i file di dati XML, il template XSL e indicare il file di output PDF

```
import java.io.*;
import javax.xml.transform.*; //classi JAXP
import org.apache.fop.apps.*; //classi FOP
....
Driver driver = new Driver();//Construct driver
driver.setRenderer(Driver.RENDER_PDF);
OutputStream out = new java.io.FileOutputStream(pdf);
driver.setOutputStream(out);
TransformerFactory factory = TransformerFactory.newInstance();
Transformer transformer =
    factory.newTransformer(new StreamSource(xslt)); //Setup XSLT
Source src = new StreamSource(xml);
Result res = new SAXResult(driver.getContentHandler());
transformer.transform(src, res);
```

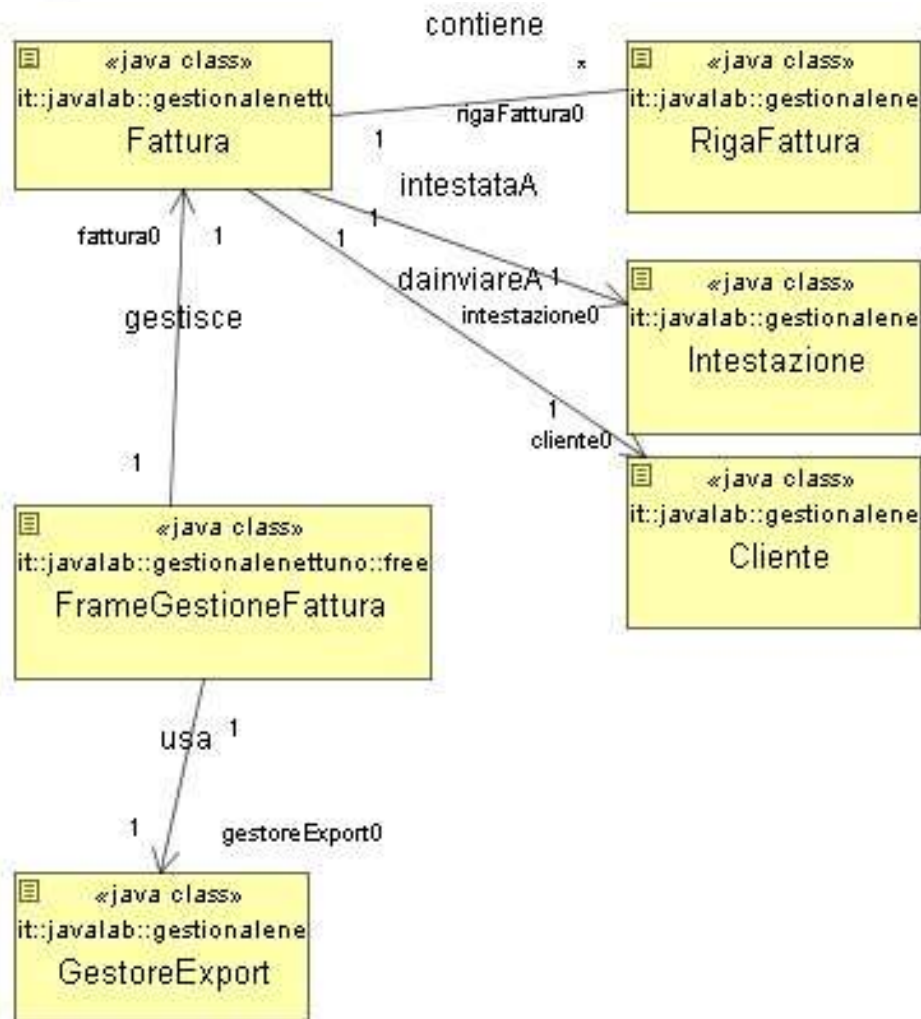


Gestionale Nettuno per Freelance – Requisiti

- Requisiti: mi occorre un software per esportare in PDF e stampare fatture da poter installare su una pendrive usb da 64 MB e per poterlo eseguire da qualunque sys operativo dato che presso alcuni clienti il mio portatile non può entrare per rigide regole atte a mantenere i segreti industriali
- Casi d'uso
Nuova fattura – Salva – Load – Esporta in PDF – Anteprima – Stampa
 - L'anteprima e la stampa le demando ad un viewer esterno configurabile (ad esempio Acrobat Reader)
 - Utilizzo il filesystem salvando i dati in xml per non dover istanziare un dbms (dato che il software è monoutente) e per poterli facilmente importare nel gestionale del mio commercialista.



Gestionale Nettuno per Freelance – Oggetti





Gestionale Nettuno per Freelance – Schermata

gn4fl 1.2 - freeware di Andrea.Musumeci@javalab.it - aprile 2004

File Help

Apri: Andrea Musumeci
Salva: via Paletta 6
Esporta PDF: 28100 Novara
Exit: +393470755781

gn4fl 1.2
da www.javalab.it

Notizie fiscali 1: MDDIEIIEIEIEEOEO
Notizie fiscali 2: 01234567891
Sito web riferimento: http://??
Contatto email: tuonome@tuodominio.est

Luogo emissione: inserire la tua citta' **progressivo**: 000001/04
Data emissione: 6 maggio 2004

Intestazione: NOME CLIENTE
Indirizzo 1: via
Indirizzo 2: CAP CITTA
Notizie fiscali 1: 12345678901

CODICE	DESCRIZIONE	GG	PREZZO	TOTALE
0011	corso di java base	4	300	1200,00

Totale Imponibile	1200,00
Imposta	240,00
Totale Prestazioni	1200,00
dedotta RitAcc	240,00
Totale Fattura	1440,00

notizie aggiuntive



Link utili

- <http://xml.apache.org/fop/index.html> – approfondimenti su FOP
- <http://www.javalab.it> – il gestionale



Q&A





GRAZIE

da

Andrea Musumeci

www.JavaPortal.it



Un grazie a chi ha voluto anche il mio nome tra gli speaker della comunità JIP.

KeyTECH
SOLUZIONI INFORMATICHE

www.JavaPortal.it