Um parser XML*em Common Lisp

David Fernandes
daugfernandes@aim.com

26 de Janeiro de 2011

Resumo

Com o advento da Internet e com o consequente aumento, por um lado, do mercado potencial consumidor de informação e, por outro, da facilidade de acesso a fontes de dados diversas, operou-se uma mudança significativa de paradigma no que toca ao desenvolvimento de aplicações: a produção e a utilização da informação tornaram-se actividades autónomas, cada uma delas objecto de negócio por si mesma, cada uma delas objecto de investigaçãp e desenvolvimento autónomos.

Deste modo, as aplicações consumidoras de dados necessitam de um mecanismo que lhes permita aceder de forma controlada ao conteúdo estruturado de um bloco de informação representada em XML.

Este trabalho apresenta o desenvolvimento e implementação de um tal mecanismo, chamado de 'parser', em Common Lisp.

1 Introdução

1.1 Perspectiva histórica[2]

A XML é uma linguagem de 'marcas', de certa forma descendente da GML criada pela IBM, que permite descrever de forma estruturada os mais variados tipos de informação.

2 XML

Apresentaremos de seguida a gramática independente de contexto formal da XML tal como definida pelo W3C, tendo o cuidado de colocar após cada grupo de produções um exemplo XML concreto da utilização das mesmas.

2.1 Gramática da XML

2.1.1 Documento

Um documento XML consiste no prólogo e no elemento raíz. (Ver exemplo em 3.1)

1 document ::= (#prolog element Misc*) - (Char* RestrictedChar Char*)

*Extensible Markup Language - http://www.w3.org/standards/xml/core

2.1.2 Carácteres

2	Char	::=	[#x1-#xD7FF] | [#xE000-#xFFFD] | [#x10000-#x10FFFF]
2a	RestrictedChar	::=	[#x1-#x8] | [#xB-#xC] | [#xE-#x1F] | [#x7F-#x84] | [#x86-#x9F]

2.1.3 Espaço em branco

3	S	::=	(#x20 | #x9 | #xD | #xA)+

2.1.4 Nomes e 'tokens'

4	NameStartChar	::=	":" | [A-Z] | "_" | [a-z] | [#xC0-#xD6] | [#xD8-#xF6] | [#xF8-#x2FF] | [#x370-#x37D] | [#x37F-#x1FFF] | #x200C-#x200D] | [#x2070-#x218F] | [#x2C00-#x2FEF] | [#x3001-#xD7FF] | [#xF900-#xFDCF] | [#xFDF0-#xFFFD] | [#x10000-#xEFFFF]
4a	NameChar	::=	NameStartChar | "-" | "." | [0-9] | #xB7 | [#x0300-#x036F] | [#x203F-#x2040]
5	Name	::=	NameStartChar (NameChar)*
6	Names	::=	Name (#x20 Name)*
7	Nmtoken	::=	(NameChar)+
8	Nmtokens	::=	Nmtoken (#x20 Nmtoken)*

2.1.5 Literais

9	EntityValue	::=	" ([^%&"] | PEReference | Reference)* " | '([^%&'] | PEReference | Reference)* '
10	AttValue	::=	" ([^<&"] | Reference)* " | ' ([^<&'] | Reference)* '
11	SystemLiteral	::=	(" [^"]* ") | (' [^']* ')
12	PubidLiteral	::=	" PubidChar* " | ' (PubidChar - ')* '
13	PubidChar	::=	#x20 | #xD | #xA | [a-zA-Z0-9] | [-'()+,./:=?;!*#$_%]

2.1.6 Dados carácter

14	CharData	::=	[^<&]* - ([^<&]* ']]>' [^<&]*)

2.1.7 Comentários

15	Comment	::=	'<!–' ((Char - '-') | ('-' (Char - '-')))* '–>'

2.1.8 Instruções de processamento

16	PI	::=	'<?' PITarget (S (Char* - (Char* '?>' Char*)))? '?>'
17	PITarget	::=	Name - (('X' | 'x') ('M' | 'm') ('L' | 'l'))

2.1.9 Secções CDATA

18	CDSect	::=	CDStart CData CDEnd
19	CDStart	::=	'<![CDATA['
20	CData	::=	(Char* - (Char* ']]>' Char*))
21	CDEnd	::=	']]>'

2.1.10 Prolog

22	prolog	::=	XMLDecl Misc* (doctypedecl Misc*)?
23	XMLDecl	::=	'<?xml' VersionInfo EncodingDecl? SDDecl? S? '?>'
24	VersionInfo	::=	S 'version' Eq ('"'VersionNum '"' | "'" VersionNum "'")
25	Eq	::=	S? '=' S?
26	VersionNum	::=	'1.1'
27	Misc	::=	Comment | PI | S

2.1.11 Definição DOCTYPE

28	doctypedecl	::=	'<!DOCTYPE' S Name (S ExternalID)? S? ('[' intSubset ']' S?)? '>'
28a	DeclSep	::=	PEReference | S
28b	intSubset	::=	(markupdecl | DeclSep)*
29	markupdecl	::=	elementdecl | AttlistDecl | EntityDecl | NotationDecl | PI | Comment

2.1.12 Subconjuntos externos

30	extSubset	::=	TextDecl? extSubsetDecl
31	extSubsetDecl	::=	(markupdecl | conditionalSect | DeclSep)*

2.1.13 Declaração de documento isolado

32	SDDecl	::=	S 'standalone' Eq (("'"('yes' | 'no') "'") | ('"'('yes' | 'no') '"'))

2.1.14 Elemento

39	element	::=	EmptyElemTag | STag content ETag

2.1.15 TAG de início

40	STag	::=	'<' Name (S Attribute)* S? '>'
41	Attribute	::=	Name Eq AttValue

2.1.16 TAG de fim

42	ETag	::=	'</' Name S? '>'

2.1.17 Conteúudo de um elemento

43	content	::=	CharData? ((element | Reference | CDSect | PI | Comment) CharData?)*

2.1.18 TAG de elementos vazios

44	EmptyElemTag	::=	'<' Name (S Attribute)* S? '/>'

2.1.19 Declaraçã de tipo de elemento

45	elementdecl	::=	'<!ELEMENT' S Name S contentspec S? '>'
46	contentspec	::=	'EMPTY' | 'ANY' | Mixed | children

2.1.20 Modelos de conteúdo de elemento

47	children	::=	(choice | seq) ('?' | '*' | '+')?
48	cp	::=	(Name | choice | seq) ('?' | '*' | '+')?
49	choice	::=	'(' S? cp (S? '|' S? cp)+ S? ')'
50	seq	::=	'(' S? cp (S? ',' S? cp)* S? ')'

2.1.21 Declaração de conteúdo mist0

51	Mixed	::=	'(' S? '#PCDATA' (S? '|' S? Name)* S? ')*' | '(' S? '#PCDATA' S? ')'

2.1.22 Declaração de lista de atributos

52	AttlistDecl	::=	'<!ATTLIST' S Name AttDef* S? '>'
53	AttDef	::=	S Name S AttType S DefaultDecl
54	AttType	::=	StringType | TokenizedType | EnumeratedType
55	StringType	::=	'CDATA'
56	TokenizedType	::=	'ID' | 'IDREF' | 'IDREFS' | 'ENTITY' | 'ENTITIES' | 'NMTOKEN' | 'NMTOKENS'

2.1.23 Tipos de atributos enumerados

57	EnumeratedType	::=	NotationType | Enumeration
58	NotationType	::=	'NOTATION' S '(' S? Name (S? '|' S? Name)* S? ')'
59	Enumeration	::=	'(' S? Nmtoken (S? '|' S? Nmtoken)* S? ')'

2.1.24 Valores de atributos por omissão

60	DefaultDecl	::=	'#REQUIRED' | '#IMPLIED' | (('#FIXED' S)? AttValue)

2.1.25 Secção condicional

61	conditionalSect	::=	includeSect | ignoreSect
62	includeSect	::=	'<![' S? 'INCLUDE' S? '[' extSubsetDecl ']]>'
63	ignoreSect	::=	'<![' S? 'IGNORE' S? '[' ignoreSectContents* ']]>'
64	ignoreSectContents	::=	Ignore ('<![' ignoreSectContents ']]>' Ignore)*
65	Ignore	::=	Char* - (Char* ('<![' | ']]>') Char*)

2.1.26 Carácter

66	CharRef	::=	'&#' [0-9]+ ';' | '&#x' [0-9a-fA-F]+ ';'

2.1.27 Entidade

67	Reference	::=	EntityRef | CharRef
68	EntityRef	::=	'&' Name ';'
69	PEReference	::=	'%' Name ';'

2.1.28 Declaração de entidade

70	EntityDecl	::=	GEDecl | PEDecl
71	GEDecl	::=	'<!ENTITY' S Name S EntityDef S? '>'
72	PEDecl	::=	'<!ENTITY' S '%' S Name S PEDef S? '>'
73	EntityDef	::=	EntityValue | (ExternalID NDataDecl?)
74	PEDef	::=	EntityValue | ExternalID

2.1.29 Declaração de entidade externa

75	ExternalID	::=	'SYSTEM' S SystemLiteral | 'PUBLIC' S PubidLiteral S SystemLiteral
76	NDataDecl	::=	S 'NDATA' S Name

2.1.30 Declaração de texto

77	TextDecl	::=	'<?xml' VersionInfo? EncodingDecl S? '?>'

2.1.31 Entidade externa bem-formada

78	extParsedEnt	::=	(TextDecl? content) - (Char* RestrictedChar Char*)

2.1.32 Declaração de codificação

80	EncodingDecl	::=	S 'encoding' Eq ('"' EncName '"' | "'"EncName "'")
81	EncName	::=	[A-Za-z] ([A-Za-z0-9._] | '-')*

2.1.33 Declaração de notação

82	NotationDecl	::=	'<!NOTATION' S Name S (ExternalID | PublicID) S? '>'
83	PublicID	::=	'PUBLIC' S PubidLiteral

3 Exemplos

3.1 Documento

```
<?xml version="1.0"?>
<mensagem>
  <enviada timezone="UTC">12:54</enviada>
  <de>
    <nome>David Fernandes</nome>
    <endereco>daugfernandes@aim.com</endereco>
  </de>
  <para>
    <endereco>lc@univ-ab.pt</endereco>
  </para>
  <corpo>
    Gostaria de marcar encontro para apresentao de
    ideia para projecto de final de curso.
  </corpo>
</mensagem>
```

Figura 1: Document

Referências

[1] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F., Cowan, J. (Eds.). (2006). *Extensible Markup Language (XML) 1.1 (Second Edition) -W3C Recommendation 16 August 2006, edited in place 29 September 2006*, [html] Acedido em 26 de Janeiro de 2011, no *Web Site* do W3C: http://www.w3.org/TR/2006/REC-xml11-20060816/

[2] Connolly, R. (Rev.) (2006) *Development History*, [html] Acedido em 26 de Janeiro de 2011, no *Web Site* do W3C: http://www.w3.org/XML/hist2002