SXMLua: Definindo estilos de documentos XML em Lua

Lucas Gomes da Silva, Rodrigo C. M. Santos,
Roberto Gerson de Albuquerque Azevedo
Departamento de Informática—PUC-Rio
Rio de Janeiro—RJ—Brasil
{lucas, rodrigocosta, robertogerson}@telemidia.puc-rio.br

ABSTRACT

This paper proposes SXMLua (Styling XML through Lua), a pre-processor and a library supporting the definition of stylesheets for XML documents using Lua. SXMLua was inspired by the CSS language, and, similar to it, allows the separation between the presentation and the content specification of XML documents. It also provides (through the Lua language) facilities that are not natively supported by CSS, such as variables and functions. Besides presenting SXMLua, in this paper, we also discuss its use for specifying stylesheets for interactive digital TV applications using NCL (Nested Context Language).

Keywords

Stylesheets; XML; NCL; CSS; Lua

1. INTRODUÇÃO

A criação de documentos XML (*EXtensible Markup Language*) é muitas vezes verbosa e redundante—frequentemente é necessário reusar um conjunto de atributos em diferentes elementos, em um ou mais documentos [9]. O uso de linguagens de estilo é uma forma de abordar esse problema, uma vez que elas promovem a separação entre a especificação do conteúdo e da apresentação de documentos estruturados—favorecendo assim a modularização e reúso de código.

CSS (*Cascading Style Sheet*) [6] é uma das linguagens de estilo mais conhecidas, sendo utilizada principalmente no desenvolvimento web. A linguagem permite que diferentes profissionais (e.g., designers e programadores) trabalhem colaborativamente, cada um em seu devido nível de abstração. Embora CSS tenha sido projetado inicialmente para definir o estilo de páginas HTML, ela também pode ser usada em outros ambientes, como, por exemplo, para o desenvolvimento de interface gráficas [2] ou para o desenvolvimento de aplicações de TV Digital. Mais ainda, também é possível extrapolar o modelo de CSS, focado especialmente na especificação de atributos de leiaute, para permitir a especificação de qualquer conjunto de atributos de documentos XML em um documento de estilo [8].

Este artigo descreve a ferramenta SXMLua (*Styling XML through Lua*). SXMLua possibilita que conceitos similares aos de CSS (e.g., atributos, seletores etc.) possam ser utilizados para especificar o estilo de documentos XML por meio da linguagem Lua. A vantagem dessa abordagem é permitir o uso de construções de Lua na definição do estilo. Algumas dessas construções, e.g., variáveis, laços e funções, não são suportadas nativamente por CSS. Além disso, SXMLua também promove o emprego da mesma forma de autoria já explorada por desenvolvedores web (utilizando documentos de estilo) em linguagens e ambientes XML que não suportam nativamente CSS. Como exemplo, neste artigo exploramos o uso de SXMLua para a especificação de leiautes em ambientes de TV Digital, usando a linguagem NCL (*Nested Context Language*) [10].

O restante do artigo está organizado como segue. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve a proposta e a arquitetura da ferramenta SXMLua. A Seção 4 apresenta um exemplo de cenário de uso da ferramenta para a especificação de leiautes para documentos NCL, para ambientes de TV Digital interativa e no qual CSS não é suportada nativamente. Além disso, também são discutidas funcionalidades para a especificação de estilos de documentos XML não suportadas nativamente por CSS. Por fim, a Seção 5 traz as conclusões e trabalhos futuros.

2. TRABALHOS RELACIONADOS

XSLT (*EXtensible Stylesheet Language Transformations*)[7] é uma linguagem XML para a definição de transformações em documentos XML. Um cenário de uso típico de XSLT é a definição da apresentação de dados armazenados em documentos XML na forma de páginas HTML. Assim como XSLT, SXMLua também pode ser utilizada para a definição da apresentação de documentos XML. XSLT, entretanto, utiliza XPath [7] para selecionar os subconjuntos de elementos da árvore do documento de origem, enquanto SXMLua usa uma sintaxe, baseada em tabela Lua, similar aos seletores CSS. Enquanto XPath é de fato mais poderoso do que os seletores CSS, esse último, possivelmente devido à sua simplicidade, tem sido mais utilizado—principalmente por usuários não especialistas.

Linguagens dinâmicas para a definição de estilo que estendem as funcionalidade de CSS—e que podem ser compiladas para CSS—também têm ganhado popularidade nos últimos anos. Geralmente, essa linguagens estendem CSS com funcionalidades como variáveis, operações matemáticas, funções, aninhamentos de regras. Exemplos dessas linguagens são: *Less* [1], *SASS/SCSS*[3] e *Stylus* [4].

Semelhante às linguagens acima, SXMLua também per-

In: Workshop de Trabalhos em Iniciação Científica (WTIC), 13., 2016, Teresina. Anais do XXII Simpósio Brasileiro de Sistemas Multimídia e Web. Porto Alegre: Sociedade Brasileira de Computação, 2016. v. 2.

ISBN: 978-85-7669-332-1

mite estender CSS com funcionalidades como variáveis e funções. Como principal diferença, as linguagens acima se propõem a compilar para CSS enquanto SXMLua se propõe a gerar o XML final diretamente. Essa abordagem favorece o uso de estilos mesmo em ambientes que não possuem um interpretador para tais linguagens. Além disso, as linguagens acima são específicas de domínio específico (DSLs), enquanto SXMLua é uma DSL em Lua. Nesse sentido, SXMLua pode ser usado tanto por designers, que podem se restringir à sintaxe básica similar à CSS, como por programadores, que podem tirar proveito da expressividade de Lua para facilmente estender SXMLua com novas funcionalidades.

Seguindo um caminho similar à SXMLua, em [5] também é definida uma DSL para gerar CSS. Assim como SXMLua, a DSL proposta em [5] permite ao usuário gerar um código CSS utilizando Lua e seguindo um esquema definido por tabelas. Assim como SXMLua, ao usar Lua como a linguagem *host* para a DSL proposta, em [5] também é proposta uma ferramenta baseada em CSS que pode ser facilmente estendida em Lua. Como diferença, e similar às linguagens discutidas acima, a DSL proposta em [5] tem como objetivo gerar um documento CSS, enquanto SXMLua transforma o documento XML.

3. SXMLUA

A Figura 1 ilustra o fluxo de trabalho da SXMLua. Como entrada, SXMLua recebe um documento XML (1) e uma tabela Lua (2), contendo as regras a serem aplicadas no documento XML. A sintaxe da tabela de regras segue as mesmas regras sintática de tabelas Lua. As chaves da tabela devem ser seletores CSS, e o conteúdo de cada campo é uma outra tabela com um conjunto de propriedades e valores. Como saída, SXMLua produz um novo documento XML (4) aplicando as regras definidas na tabela Lua de entrada.

Figure 1: Funcionamento da SXMLua.

O código a seguir mostra um exemplo de tabela de regras:

```
["rect"] = {
  width  = 150,
  height = 150
},
[".mycircle"] = {
  stroke = "green",
  fill = "yellow"
}
```

Caso a tabela de regras acima seja aplicada no seguinte arquivo XML (e.g. *input.svg*):

```
<svg width="400" height="180">
 <rect x="0" y="20"/>
 <rect x="50" y="50"/>
 <circle class="mycircle" cx="50" cy="50"
 ↪  r="40"/>
</svg>
```

o resultado será o seguinte XML processado (e.g. *out.svg*):

```
<svg width="400" height="180">
 <rect x="0" y="20" width="150" height="150"/>
 <rect x="50" y="50" width="150"
 ↪  height="150"/>
 <circle class="mycircle" cx="50" cy="50"
 ↪  r="40" stroke="green" fill="yellow"/>
</svg>
```

Há duas formas de utilizar a SXMLua: como uma ferramenta *standalone* ou como uma biblioteca Lua. No primeiro caso, deve-se passar como argumento um documento XML e um arquivo Lua contendo a tabela de regras. Esse modo de uso não requer conhecimentos profundos em Lua, apenas da sintaxe de criação de tabelas.

No segundo caso, programadores Lua podem usar a biblioteca SXMLua (`require "sxmlua"`) em seus *scripts*. A função `SXMLua.process` é responsável por ler o documento XML e aplicar as regras definidas na tabela de regras. O código a seguir mostra como o exemplo anterior poderia ser utilizado em um script Lua:

```
local SXMLua = require "sxmlua"
local tam = 1080
local style = {
  ["rect"] = {
    width  = tam,
    height = tam/3
  },
  [".mycircle"] = {
    stroke = "green",
    fill = "yellow"
  }
}
SXMLua.process(style, "input.svg", "out.svg")
```

Usar a SXMLua como uma biblioteca permite que programadores Lua utilizem as construções nativas da linguagem para realizar computações (e.g., operações matemáticas, variáveis e funções) na definição da tabela de regras. No exemplo acima, uma variável Lua está sendo usada para definir as propriedades *width* e *height*.

SXMLua também define o comportamento padrão de alguns tipos de dados, em especial, tabelas e funções. Caso um campo seja uma tabela, SXMLua expande os valores dessa tabela. Isso permite agrupar um conjunto de propriedades em uma variável Lua e reusá-las na especificação de mais de uma regra. Caso o valor de um campo seja uma função, essa função será chamada (e o valor retornado será atribuído a este campo) para cada um dos elementos selecionados.

Por exemplo, caso o exemplo anterior seja alterado para:

```
local SXMLua = require "sxmlua"
local n_rect = 0
local tam = 150
local colors = {
  stroke = "green",
  fill = "yellow"
```

```
}
local style = {
  ["rect"] = {
    colors,
    width = tam,
    height = function ()
               n_rect = n_rect+1
               return n_rect * 100
             end
  },
  [".mycircle"] = {
    colors
  }
}
SXMLua.process(style, "input.svg", "out.svg")
```

o arquivo de saída gerado será:

```
<svg width="400" height="180">
 <rect x="0" y="20" width="150" height="100"
 ↪  stroke="green" fill="yellow"/>
 <rect x="50" y="50" width="150" height="200"
 ↪  stroke="green" fill="yellow"/>
 <circle class="mycircle" cx="50" cy="50"
 ↪  r="40" stroke="green" fill="yellow"/>
</svg>
```

No exemplo, a tabela `colors` é reusada na definição das duas regras `"rect"` e `".circle"`. Portanto, todos os elementos selecionados (os dois retângulos e o círculo) terão os atributos `stroke` e `fill` preenchidos. A função definida pela propriedade `height` será chamada uma vez para cada um dos elementos selecionados pelo seletor `"rect"`. Por isso, o valor do atributo `height` do primeiro retângulo é igual a 100, enquanto o do segundo retângulo é 200.

3.1 Estendendo SXMLua

Como é possível notar nos exemplos anteriores, por padrão, SXMLua aplica as propriedades da tabela de entrada Lua como atributos XML dos elementos selecionados. Dessa forma, SXMLua pode ser utilizado para qualquer documento XML para separar a apresentação (ou quaisquer outros atributos) do conteúdo de documentos XML.

Em alguns casos, aplicar o estilo ao documento XML não é tão simples quanto apenas adicionar as propriedades como atributos XML dos elementos selecionados. Caso necessário, também é possível customizar como SXMLua aplica as propriedades definidas na tabela de regras no documento XML. Isso é feito por meio de uma função definida pelo usuário, e que deve ser passada como parâmetro para a função `SXMLua.process`. A função definida pelo usuário será chamada para cada um dos elementos selecionados (em cada uma das regras), e recebe como parâmetros a tabela com as propriedades a serem aplicadas e o elemento que foi selecionado.

Duas funções adicionais já estão implementadas na SXMLua: (1) `applyAsAttrStyle` e (2) `applyAsElemProperty`.

A função `applyAsAttrStyle` aplica os estilos e transforma as propriedades definidas na tabela de regras em um único atributo *style* nos elementos que foram selecionados. Essa função foi especialmente projetada para trabalhar com HTML, mas também pode ser útil em outras linguagens (e.g. SVG). Como exemplo, caso a função `applyAsAttrStyle` seja utilizada com o exemplo anterior, a saída (arquivo *out.svg*) será:

```
<svg width="400" height="180">
 ...
 <circle class="mycircle" cx="50" cy="50"
 ↪  r="40" style="stroke: green; fill:
 ↪  yellow;"/>
</svg>
```

A função `applyAsElemProperty` aplica os estilos transformando as propriedades em elementos `<property>`, filhos dos elementos selecionados e que possuem como atributos *name* e *value*. Essa função foi especialmente projetada para trabalhar com NCL. Um exemplo de uso dessa função é discutido na seção a seguir.

4. EXEMPLO DE USO: SXMLUA + NCL

Em NCL, propriedades visuais de uma mídia (`<media>`) são definidas por meio de regiões (`<region>`), descritores (`<descriptor>`) ou propriedades (`<property>`). O exemplo a seguir (baseado na aplicação interativa "Primeiro João" [1], ilustrada na Figura 4) explora como definir as propriedades dos objetos de mídia usando SXMLua e seguindo uma abordagem similar à CSS, onde seletores definem quais regras aplicam-se a quais objetos de mídia. Esse exemplo é composto pelo objeto de mídia *anim* (vídeo principal) e quatro imagens (opções do menu). A listagem a seguir apresenta um trecho de código da aplicação.

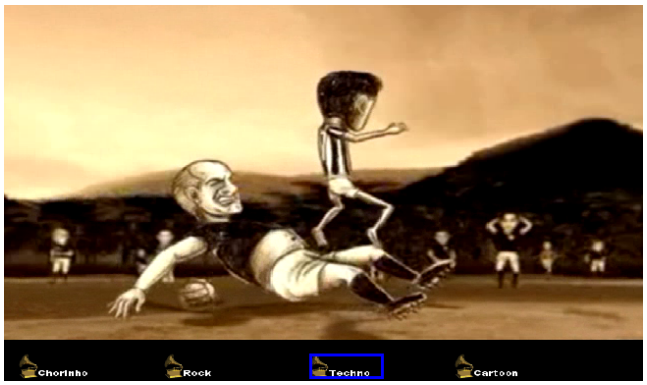

Figure 2: Exemplo de execução da aplicação "Primeiro João"

```
<ncl>
 ...
 <body>
  <media id="anim" ... />

  <media id="imgChoro" class="menu" ... />
  <media id="imgRock" class="menu" ... />
  ...
 </body>
</ncl>
```

O estilo do documento acima pode ser definido pelo código Lua a seguir.

[1]Disponível em: `http://www.clube.ncl.org.br`

```
1  local SXMLua = require ("sxmlua")
2  local left = 2.5, dist = 22.5
3  local n_items = 0
4  local menu = { top = "91.7%", width = "11.7%",
   ↪  height = "6.7%", zIndex = "3" }
5  local style = {
6    ["#anim"] = {
7      width = "100%",
8      height = perc_diff("100%", menu.height)
9      zIndex = "2"
10   },
11   [".menu"] = {
12     menu,
13     focusIndex = function ()
14                    n_items = n_items+1;
15                    return n_items
16                  end,
17     left = function()
18              return
   ↪  (left+(n_items-1)*dist).."%"
19            end
20   },
21   ...
22 }
23 SXMLua.process(style, "11menu.ncl", "out.ncl",
   ↪  SLAXML.applyAsElemProperty)
```

No código acima é possível observar algumas funcionalidades de SXMLua. Na linha 12, a tabela `menu` é utilizada para reusar atributos estáticos dos elementos que fazem parte do menu. Nas linhas 13–19, funções Lua são utilizadas para definir os atributos `focusIndex` e `left`. Na linha 23, a chamada `SXMLua.process` usa a função `SXMLua.applyAsElemProperty` que, ao processar o documento original, gera o documento:

```
1  ...
2  <media id="anim" .../>
3    <property name="width" value="100%"/>
4    <property name="zIndex" value="2"/>
5    <property name="height" value="93.3%"/>
6  </media>
7  <media id="imgChoro" class="menu" ... />
8    <property name="focusIndex" value="2"/>
9    <property name="left" value="2.5%"/>
10   <property name="top" value="91.7%"/>
11   <property name="width" value="11.7%"/>
12   <property name="height" value="6.7%"/>
13   <property name="zIndex" value="3"/>
14 </media>
15 <media id="imgRock" class="menu" ... />
16   <property name="focusIndex" value="3"/>
17   <property name="left" value="25%"/>
18   ...
19 </media>
20 ...
```

5. CONCLUSÃO

Este artigo discute a ferramenta SXMLua. SXMLua é um pré-processador XML e uma biblioteca que permite a definição de estilos de documentos XML em Lua. Usando a notação de tabelas Lua, SXMLua possibilita a definição de estilos de forma similar à CSS. Mais ainda, SXMLua também possibilita o uso da expressividade de Lua para a definição desses estilos. O processador pode rodar tanto do lado do servidor quanto do cliente (quando um interpretador Lua estiver disponível no cliente). Um exemplo desse último é NCL, que utiliza Lua como linguagem de script.

Atualmente, a SXMLua suporta apenas parte dos seletores CSS. Sendo assim, um dos trabalhos futuros é suportar todos os seletores. Outros trabalhos futuros que destacamos são: estender a proposta para incluir regras aninhadas, funcionalidade suportada em outras linguagens como Sass e Stylus; e estudar o uso de recursos de cascata como os presentes em CSS/HTML para SXMLua/NCL.

Além disso, nossos testes iniciais indicam que em alguns casos o documento XML final gerado pode ser consideravelmente maior que o documento original. Em ambientes com pouca largura de banda disponível, como é o caso da transmissão de aplicações interativas de TV Digital ou no do Rádio Digital, em que a largura de banda é ainda mais reduzida, a utilização da SXMLua no cliente pode ser uma abordagem interessante para economizar recursos. Assim, pretendemos fazer uma análise quantitativa com experimentos envolvendo aplicações reais para avaliar a redução de recursos que nossa ferramenta pode proporcionar.

Finalmente, salientamos a necessidade de avaliar a usabilidade da nossa DSL em Lua com usuários com diferentes níveis de especialização (designers e programadores), e estudar o possível ganho da produtividade que SXMLua oferece.

6. REFERENCES

[1] LessCSS. Disponível em: `http://lesscss.org/`.

[2] Qt Documentation—Qt Style Sheets. Disponível em `http://doc.qt.io/qt-5/stylesheet.html`.

[3] Sass: Syntactically Awesome Style Sheets. Disponível em: `http://sass-lang.com/`.

[4] Stylus: Expressive, Dynamic, Robust CSS. Disponível em: `http://styles-lang.com/`.

[5] S. Donovan. A little Lua DSL for generating CSS, 2011. Disponível em: `http://steved-imaginaryreal.blogspot.com.br/2011/08/little-lua-dsl-for-generating-css.html`.

[6] T. A. Jr., E. J. Etemad, and F. Rivoal. CSS snapshot 2015. Technical report, W3C, 2015. Disponível em : `https://www.w3.org/TR/CSS/`.

[7] M. Kay. *XSLT 2.0 and XPath 2.0*. Wrox, 2008.

[8] R. Laiola Guimarães, D. Bulterman, P. Cesar, and J. Jansen. Synchronizing web documents with style. In *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web*, WebMedia '14, pages 151–158, New York, NY, USA, 2014. ACM.

[9] R. Prieto-Diaz. Status report: software reusability. *IEEE Software*, 10(3):61–66, May 1993.

[10] L. F. G. Soares and S. D. J. Barbosa. *Programando em NCL 3.0*. Elsevier, 2011.