

SASS4J

UM COMPONENTE SASS PARA A PLATAFORMA JAVA EE

Guilherme Corrêa Morgado

INTRODUÇÃO

O desenvolvedor front-end executa diversas tarefas repetitivas no seu dia a dia. Para transformar a repetição em otimização, este projeto busca a compilação de códigos escritos em Sass utilizando a plataforma Java EE.

INTRODUÇÃO

- Motivação
- Histórico
- Para que serve?

OBJETIVO GERAL

- O objetivo do projeto é analisar e desenvolver um componente Java EE utilizando jRuby para processar o arquivos escritos com a sintaxe Scss.

OBJETIVOS ESPECÍFICOS

- Analisar e desenvolver um módulo Java EE utilizando filtros;
- Analisar e portar o módulo já existente do Sass, escrito em Ruby, para a plataforma Java EE utilizando jRuby;
- Analisar e desenvolver um mecanismo de “cacheamento” de arquivos manipulados pelo componente criado.

WEB

- World Wide Web
- HTTP
- HTML
- JavaScript
- CSS

WORLD WIDE WEB

- Arquivos manipulados e executados na Internet.
- Navegadores.
- Preceito de navegação.

HTTP

- Protocolo de comunicação padrão da Web.
- Modelo computacional cliente-servidor.
- MIME

HTML

- Linguagem de Marcação
- Tags
- Navegadores

```
4  <html>
5  <head>
6      <meta charset="UTF-8">
7      <title>TODO</title>
8  </head>
9  <body>
10     <h1>TODO</h1>
11
12     <a href="nova.php">Nova Tarefa</a>
13     <a href="lista.php">Lista de Tarefas</a>
14
15     <a href="http://av2.local/acad/index.php">Link</a>
16
17 </body>
18 </html>
```

JAVASCRIPT

- Linguagem de script
- Bibliotecas JavaScript
- Uso no lado do servidor e do cliente

```
1 $(document).ready(function(){  
2  
3     var nav = $('.nav-container');  
4  
5     $(window).scroll(function () {  
6         if ($(this).scrollTop() > 136) {  
7             nav.addClass("f-nav");  
8         } else {  
9             nav.removeClass("f-nav");  
10        }  
11    });  
12  
13 });
```

CSS

- Folhas de estilo em cascata
- Estilizar códigos HTML

```
29  .tag:hover {  
30      top: -2px;  
31      margin: 0.5em 0.5em 0.5em 1.4em;  
32      background: #f2dffb; }  
33  
34  .tag:active {  
35      top: 1px; }  
36  
37  .big {  
38      font-size: 1.5em; }  
39  
40  .giant {  
41      font-size: 2.5em; }
```

PREPROCESSADORES CSS

- Conversões léxicas em textos
- Evitar repetições excessivas
- Utilização de variáveis
- Facilidade de leitura do código escrito
- Folhas de estilo melhor administráveis

SASS

- Syntactically Awesome StyleSheets
- Linguagem declarativa
- Compilador Ruby

SASS

SCSS SYNTAX

```
$font-stack: Helvetica, sans-serif;  
$primary-color: #333;  
  
body {  
  font: 100% $font-stack;  
  color: $primary-color;  
}
```

CSS SYNTAX

```
body {  
  font: 100% Helvetica, sans-serif;  
  color: #333;  
}
```

SASS

Resultado CSS do código Scss do último slide

CSS SYNTAX

```
body {  
  font: 100% Helvetica, sans-serif;  
  color: #333;  
}
```

LESS

- Sintaxe própria
- Possibilidade de rodar o interpretador no cliente e no servidor

```
1  @verde: #6BAF2A;  
2  @fonte_h1: bold 18px "Times New Roman", Arial;  
3  
4  h1{  
5      color: @verde;  
6      font: @fonte_h1;  
7  }  
8  a{  
9      color: @verde;  
10 }
```


STYLUS

- Sintaxe intuitiva
- Ausência de pontuação nos códigos
- Importação de códigos

PLATAFORMA JAVA EE

- Utilização de APIs
- Criação de Servlets para a aplicação

```
236 public String showStories() {  
237     setCurrentSprint(sprints.getRowData());  
238     return "showStories";  
239 }  
240  
241 public String showDashboard() {  
242     setCurrentSprint(sprints.getRowData());  
243     return "showDashboard";  
244 }  
245  
246 public Sprint getCurrentSprint() {  
247     return currentSprint;  
248 }
```

SERVLETS

- Classes de extensões de APIs
- Extensão do filtro para interceptação dos arquivos
- Utilização na plataforma Java EE

FILTER

- Responsável pela interceptação dos arquivos
- Necessário a identificação de um perfil de arquivos para interceptação
- Capacidade de modularização de componentes

CONTAINERS WEB

- Componentes do servidor que realiza a comunicação com os Servlets
- Responsável pela manipulação das requisições
- Ambiente em qual é montado a aplicação
- GlassFish

JRUBY

- Implementação da linguagem de programação Ruby em Java
- Contém o núcleo completo de classes de Ruby
- Contém as bibliotecas padrões de Ruby

RUBY

- Linguagem de Script
- Orientada a objetos
- Linguagem Interpretada
- Executada em tempo real

RUBY

Exemplo de código em Ruby:

```
# Output "I Love Ruby"  
say = "I love Ruby"  
puts say  
  
# Output "I *LOVE* RUBY"  
say['love'] = "*love*"   
puts say.upcase  
  
# Output "I *Love* Ruby"  
# five times  
5.times { puts say }
```


SASS4J

- Componente Sass para a plataforma Java EE
- Utiliza o compilador Sass em Ruby
- Utiliza como comunicador entre camadas o jRuby
- Sistema de gerenciamento de arquivos já processados

SASS4J – REQUISITOS FUNCIONAIS

- Os arquivos tratados pelo componente, deverão estar escritos usando a sintaxe Scss e serão convertidos para arquivos CSS.

SASS4J – REQUISITOS NÃO FUNCIONAIS

- O componente compilará o arquivo Scss em tempo de execução sob demanda;
- A aplicação deverá prezar pela performance utilizando um mecanismo de armazenamento de informações já processadas;
- Tornar o componente possível de integração com uma aplicação Java EE existente;
- O componente poderá ser incluso em uma aplicação com no mínimo, a versão número seis da aplicação Java EE.

SASS4J - FILTER

- Filtro para interceptação dos arquivos CSS/SCSS
- Responsável pelo retorno do arquivo CSS já existente ou compilado.
- Responsável pela checagem dos arquivos em memória

```
36 @WebFilter(urlPatterns = "*.css", dispatcherTypes = DispatcherType.REQUEST)
```

SASS4J - CONSTRUTOR

Para configurar a comunicação entre jRuby e Java EE, foi criado um construtor com as informações:

```
45 public SassFilter() {  
46     config = new RubyInstanceConfig();  
47     config.setCompatVersion(CompatVersion.RUBY2_0);  
48     manager = new ScriptEngineManager();  
49     ScriptEngine script = manager.getEngineByName("jruby");  
50     engine = (Invocable) script;  
51     String rubyFile = SassFilter.class.getResource("../..sass4j.rb").getFile();  
52     String sassScript = SassFilter.class.getResource("../..sass.rb").getFile();  
53     script.put("sassScript", sassScript);  
54     InputStreamReader isr;  
55     try {  
56         isr = new InputStreamReader(new FileInputStream(rubyFile), "UTF8");  
57         script.eval(isr);  
58     } catch (FileNotFoundException | UnsupportedEncodingException | ScriptException ex) {  
59         throw new RuntimeException(ex);  
60     }  
61 }
```

SASS4J – CRIAÇÃO DE STRING ÚNICA

- Transformar o arquivo em uma String.

```
113 public StringBuilder appendSass(File sassFile) {
114
115     Scanner scanSass = null;
116     try {
117         scanSass = new Scanner(sassFile, "UTF-8");
118     } catch (FileNotFoundException ex) {
119         Logger.getLogger(SassFilter.class.getName()).log(Level.SEVERE, null, ex);
120     }
121     StringBuilder sass = new StringBuilder();
122     while (scanSass.hasNextLine()) {
123         sass.append(scanSass.nextLine());
124     }
125     return sass;
126 }
```

SASS4J – COMPILADOR SASS

- Para compilar Sass, há um método para invocar a função na camada Ruby.

```
104 public String compileSass(StringBuilder sass) {  
105  
106     try {  
107         return engine.invokeFunction("compile", sass.toString()).toString();  
108     } catch (ScriptException | NoSuchMethodException ex) {  
109         throw new RuntimeException(ex);  
110     }  
111 }
```

SASS4J – MÉTODO DOFILTER

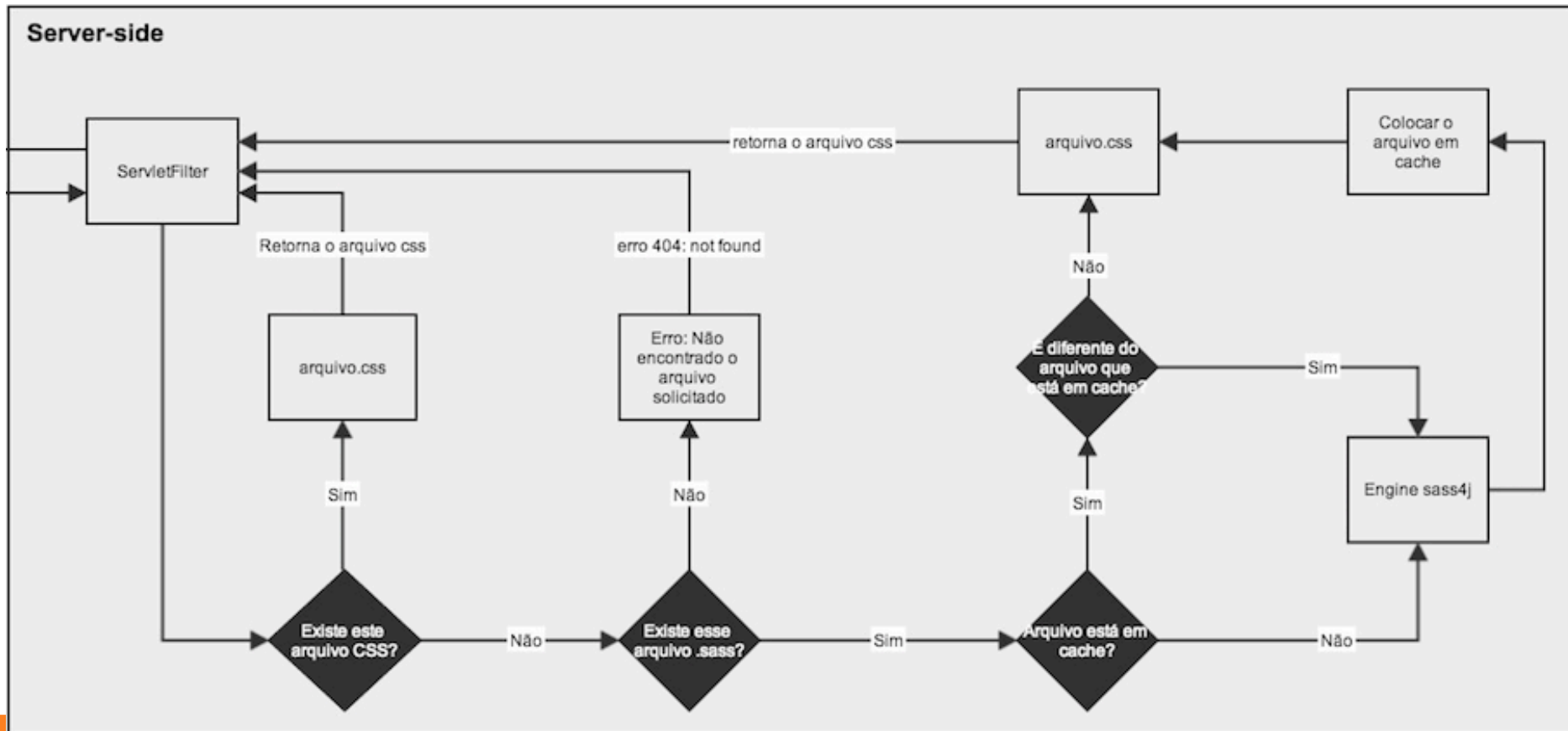
```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {  
  
    HttpServletRequest httpReq = (HttpServletRequest) request;  
    HttpServletResponse httpResp = (HttpServletResponse) response;  
    String sassPath = httpReq.getRequestURI().replace("/sass4j", "");  
    File cssFile = new File(httpReq.getServletContext().getRealPath(sassPath));  
  
    if (cssFile.exists()) {  
        chain.doFilter(request, response);  
    } else {  
        File sassFile = new File(httpReq.getServletContext().getRealPath(sassPath).replace(".css", ".scss"));  
        if (sassFile.exists()) {  
            byte[] bytesCss;  
            if (map.containsKey(cssFile.toString()) && map.get(cssFile.toString())[0].equals(String.valueOf(sassFile.lastModified())))  
                bytesCss = map.get(cssFile.toString())[1].getBytes();  
            } else {  
                StringBuilder sass = appendSass(sassFile);  
                String sassCompilado = compileSass(sass);  
                bytesCss = sassCompilado.getBytes();  
                map.put(cssFile.toString(), new String[]{String.valueOf(sassFile.lastModified()), sassCompilado});  
            }  
            httpResp.setHeader("Content-Type", "text/css");  
            OutputStream os = httpResp.getOutputStream();  
            os.write(bytesCss);  
            os.flush();  
            os.close();  
        } else {  
            httpResp.sendError(404, "SCSS and CSS files not founded.");  
        }  
    }  
}
```


SASS4J – OTIMIZAÇÃO DE ARQUIVOS

Criação de um sistema de otimização de arquivos:

- Mapa(WeakHashMap) com duas posições
- Chave: String com o arquivo CSS requisitado
- Valor: Lista de Strings com a data de modificação e o CSS já compilado.

SASS4J – DIAGRAMA DE FLUXO



SASS4J – ESTUDO DE CASO

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title></title>
5      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6      <link href="style.css" rel="stylesheet" type="text/css">
7
8    </head>
9    <body>
10     <div id="content">
11       <a href="#" class="tag giant">sass4j</a><br />
12       <a href="#" class="tag">tiny tags</a>
13       <a href="#" class="tag">that</a><br />
14       <a href="#" class="tag big">scale</a>
15       <a href="#" class="tag">perfectly</a>
16     </div>
17   </body>
18 </html>
```

SASS4J – ESTUDO DE CASO

```
1 $fontsize: 13px; $bg: #E885B6; $tag: #FFFFFF; $taghover: #F2DFF2; $text: #000000; $bg-white: white; $dg: 45deg;
2
3 body {
4   color: $text;
5   margin: 0;
6   font-family: Droid Sans, sans-serif;
7   font-size: $fontsize;
8   word-spacing: 2px;
9   text-align: center;
10  background: $bg;
11 }
12 .tag:hover {top: -2px; margin: 0.5em 0.5em 0.5em 1.4em;background: $taghover;}
13 .tag:active{top: 1px;}
14 .big{font-size: 1.5em;}
15 .giant{font-size: 2.5em;}
16 #content{margin: 100px auto 0;}
17
```

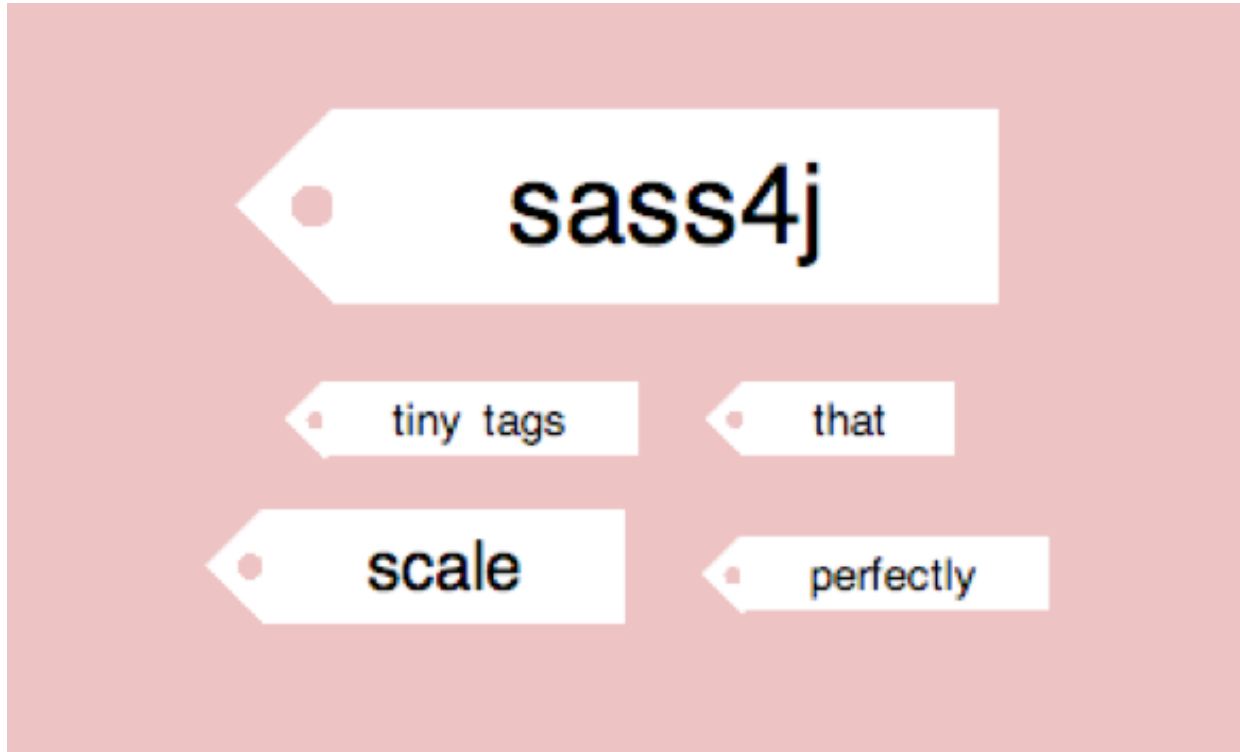
SASS4J – ESTUDO DE CASO

```
18 .tag {  
19     display: inline-block;  
20     padding: 0.3em 1.6em;  
21     background-color: $bg-white;  
22     position: relative;  
23     margin: 0 0 0 0.9em;  
24     line-height: 1.17;  
25     color: black;  
26     text-decoration: none;  
27     margin: 0.5em 0.5em 0.5em 1.4em;  
28 }  
29
```

SASS4J – ESTUDO DE CASO

```
30 .tag::before {
31     content: '';
32     position: absolute;
33     height: 1.25em;
34     width: 1.25em;
35     top: 0;
36     left: 0;
37     -webkit-transform-origin: top left;
38     -moz-transform: rotate($dg);
39     -ms-transform: rotate($dg);
40     -webkit-transform: rotate($dg);
41     -webkit-transform: rotate($dg);
42     background: -moz-radial-gradient(circle 1em at 40% 60%, rgba(0, 0, 0, 0) 0.19em, $tag 0.2em, $tag);
43     background: -webkit-radial-gradient(circle 1em at 40% 60%, rgba(0, 0, 0, 0) 0.19em, $tag 0.2em, $tag);
44     background: radial-gradient(circle 1em at 40% 60%, rgba(0, 0, 0, 0) 0.19em, $tag 0.2em, $tag);
45     z-index: -1;
46 }
```

SASS4J – ESTUDO DE CASO - RESULTADO



CASOS EXISTENTES: JSASS

Jsass é um compilador de Sass utilizando Java. É baseado na utilização de ANTLR(ANother Tool for Language Recognition) para a interpretação de expressões em Sass. Apesar das poucas versões de códigos enviados, pode-se perceber que é um código de compilação simples, que utiliza de tokens para o processo de criação de códigos CSS.

CASOS EXISTENTES: SASS-JAVA

Sass-java é um compilador utilizando a sintaxe Sass, em tempo real, utilizando Compass via um *servlet* feito em Java EE. Sua funcionalidade é a geração de CSS usando *templates* na sintaxe Sass. Suas principais características são a utilização de Compass para compilar os arquivos Sass e a portabilidade para usar com Maven.

CONCLUSÃO

- Criação de um componente usual para compilar Sass em Java EE;
- Fácil preparação de ambiente da plataforma Java EE;
- Entendimento avançado na usabilidade de preprocessadores CSS;
- Estudos de APIs e Servlets para aplicação Java EE;
- Estudos na camada HTTP para interceptação de arquivos

REFERÊNCIAS

- BROWN, K; IBM PRESS. Enterprise Java Programming with IBM WebSphere. 2nd ed. Boston: Addison-Wesley, 2003. p. 79-80.
- DUCKETT, J. HTML and CSS: Design and Build Websites. Hoboken: John Wiley & Sons, 2011. p. 20
- EDELSON, J; LIU, H. JRuby Cookbook. Sebastopol: O'Reilly, 2008. p. 1.
- FLANAGAN, D. JavaScript: The Definitive Guide. 6th ed. Sebastopol: O'Reilly, 2011. p. 1
- FLANAGAN, D.; MATSUMOTO, Y. The Ruby Programming Language. Sebastopol: O'Reilly, 2008. p. 1-2.
- FOSTER, J. CSS for Windows 8 App Development. New York: Apress, 2013. p. 261.

REFERÊNCIAS

FRAIN, B. Sass and Compass for Designers. Birmingham: Packt Publising, 2013. p. 22.

GONCALVES, A. Beginning Java EE 6 with GlassFish 3. 2nd ed. New York: Apress, 2010. p. 35.

GOURLEY, D; TOTTY, B; SAYER, M; AGGARWAL, A; REDDY, S. HTTP: The Definitive Guide: The Definitive Guide. Sebastopol: O'Reilly, 2002. p. 3-5

HUNTER, J.; CRAWFORD, W. Java Servlet Programming. 2nd ed. Sebastopol: O'Reilly, 2001. p. 5-6.

KOSMACZEWSKI, A. Sencha Touch 2 Up and Running. Sebastopol: O'Reilly, 42 2013. p. 179.

REFERÊNCIAS

LIBBY, A. Instant SASS CSS How-to. Birmingham: Packt Publising, 2013. p. 1

MELONI, J. Sams Teach Yourself HTML, CSS, and JavaScript All in One.
Indianapolis: Sams Publishing, 2011. p. 1

YAAPA, H. Express Web Application Development. Birmingham: Packt
Publising, 2013. p. 34.

CATLIN, H.; WEIZENBAUM, N.; EPPSTEIN, C. Sass Basics. Junho 2014.
Disponível em: <<http://sass-lang.com/guide>>. Acesso em: 10 jun. 2014.

EIS, D. Pré processadores: usar ou não usar?. Julho 2013. Disponível em:
<<http://tableless.com.br/pre-processadores-usar-ou-nao-usar/>>. Acesso
em: 12 jun. 2014.

REFERÊNCIAS

LESS. Getting Started. Maio 2012. Disponível em: <<http://lesscss.org/>>. Acesso em: 20 jun. 2014.

PILON, A. SASS/SCSS. Outubro 2009. Disponível em: <<https://drupal.org/project/sass>>. Acesso em: 10 jun. 2014.

POPLADE, T. Sass - Um outro método de escrever CSS. Junho 2013. Disponível em: <<http://tableless.com.br/sass-um-outro-metodo-de-escrever-css>>. Acesso em: 12 jun. 2014.

STYLUS. LearnBoost/Stylus. Dezembro 2010. Disponível em: <<https://github.com/learnboost/stylus>>. Acesso em: 20 jun. 2014.

W3C. Sobre o W3C. Novembro 2011. Disponível em: <<http://www.w3c.br/Sobre>>. Acesso em: 15 jun. 2014.

Obrigado!