

Code Mã

Trương Hường

♥ for sharing

codemai.blogspot.com - truonghuong.code3@gmail.com

Thiết kế blogspot cơ bản

Tài liệu được tổng hợp và viết theo kinh nghiệm cá nhân.

Blogger.Com

Blogger.Com	1
Giới thiệu ban đầu	2
Mục đích ban đầu	2
Yêu cầu	2
Chú ý	2
Một số khái niệm cơ bản	8
Thẻ đóng	8
Dữ liệu của blogspot	8
Thuộc tính của thẻ	8
Thẻ gọi dữ liệu	8
Section (Mục)	9
Thẻ tạo mục	9
Tên gọi widget xác định	9
Tiện ích widget	10
Blog	10
Blog Archive	10
Blogger Button	10
Blog List	10
Search	10
Contact Form	11
Featured Post	11
Feed	11
Follow By Email	11
Header	11
HTML	11
Image	12
Label	12
Link List	12
Page List	12
Popular Posts	12
Profile	12
Report Abuse	12
Stats	13
Subscribe	13
Text	13
TextList	13
	3

Translate	13
Wikipedia	13
Mảng	14
Giao diện cơ bản đầu tiên	15
Yếu tố bắt buộc của mỗi giao diện	15
Giao diện đơn giản nhất	15
Cấu trúc một giao diện blogspot	16
Thẻ b:skin trong thẻ head	16
Thẻ b:section nâng cao	17
Cú pháp thẻ b:section	18
Đặc điểm thẻ b:section	21
Ví dụ sử dụng	22
Thẻ tiện ích b:widget	23
Cú pháp thẻ tiện ích b:widget	23
Đặc điểm của thẻ b:widget	25
Ví dụ sử dụng	27
Thẻ chứa dữ liệu hiển thị b:includable	27
Đặc điểm thẻ b:includable	29
Cấu trúc của thẻ b:includable	29
Ví dụ sử dụng	30
Thẻ gọi dữ liệu hiển thị b:include	30
Đặc điểm thẻ b:include	30
Ví dụ sử dụng	31
Một số loại trang trong blogspot	34
Thẻ dữ liệu và điều kiện trong blogspot	34
Cấu trúc thẻ gọi dữ liệu	34
Thẻ dữ liệu chung	34
Thẻ dữ liệu tiện ích Blog, PopularPosts, và FeaturedPost	36
Thẻ dữ liệu cho tiện ích Header	43
Thẻ dữ liệu cho tiện ích HTML	45
Thẻ dữ liệu cho tiện ích Image	46
Thẻ dữ liệu cho tiện ích PageList	48
Thẻ dữ liệu cho tiện ích TextList	48
Thẻ dữ liệu cho tiện ích LinkList	49
Thẻ dữ liệu tiện ích Labels	50
Thẻ dữ liệu tiện ích Translate	50

Thẻ dữ liệu tiện ích Profile	51
Thẻ dữ liệu cho tiện ích Subscribe	52
Một số thẻ căn bản của blogspot	53
Thẻ điều kiện trong blogspot	53
Thuộc tính xuất dữ liệu trong blogspot	54
Thẻ b:eval trong blogspot	55
Cấu trúc thẻ b:eval	56
Ví dụ sử dụng	56
Thẻ b:tag trong blogspot	56
Thẻ b:attr trong blogspot	56
Thẻ b:class trong blogspot	57
Thẻ b:comment trong blogspot	57
Thẻ b:template-script trong blogspot	58
Thẻ b:with trong blogspot	58
Thẻ b:message và b:param	58
Thẻ b:swith, b:case và b:default	60
Thẻ b:loop trong blogspot	60
Thẻ <![CDATA[]]> trong blogspot	61
Thẻ b:widget-settings và b:widget-setting trong blogspot	65
Thẻ cài đặt trong tiện ích PopularPosts	67
Thẻ cài đặt trong tiện ích Blog	67
Thẻ cài đặt trong tiện ích FeaturedPost	69
Thẻ cài đặt trong tiện ích Labels	69
Thẻ b:skin trong blogspot	70
Thẻ b:defaultmarkups và b:defaultmarkup	70
Một số loại toán tử trong blogspot	72
Toán tử logic và toán tử thường trong blogspot	72
Toán tử mảng trong blogspot	73
Toán tử mảng nâng cao	74
Toán tử làm việc với đường dẫn	76
Ví dụ sử dụng toán tử liên kết	77
Toán tử làm việc với chuỗi	77
Ví dụ sử dụng	78
Toán tử làm việc với hình ảnh.	79
Ví dụ sử dụng toán tử hình ảnh	80
Toán tử format làm việc với ngày tháng	81

Trước khi đi vào tìm hiểu các tiện ích widget nâng cao	85
Tiện ích widget nâng cao	89
Tiện ích Blog	89
Hiển thị bài viết ra trang chủ:	89
Tiện ích PopularPosts	93
Tiện ích FeaturedPost	93
Tiện ích Labels	94
Tổng kết	95
Phụ lục	96
Lưu ý ban đầu:	96
Làm thế nào để tạo một form comment trong bài viết	100
Xác định khung bình luận và các chức năng các nút cần có.	101
Lời cảm ơn	103

Gửi tặng những người tớ yêu quý và nhất và cậu  người tớ thích nhưng không biết có thích tớ hay không

Một số khái niệm cơ bản

Một số khái niệm cần phải biết và hiểu để phòng gặp trong tài liệu

Việc thiết kế một template blogspot tương tự như khi các ông viết một giao diện html. Tuy nhiên nó chỉ khác ở một số quy tắc mà trong tài liệu này tớ sẽ chỉ ra.

Thẻ đóng

Giao diện của blogspot được viết theo một quy tắc rất nghiêm ngặt, mọi thẻ html đều bắt buộc phải được đóng đúng cách. Ví dụ, với cú pháp `` là chính xác với html, tuy nhiên với xml của blogspot, nó bắt buộc phải là ``

Dữ liệu của blogspot

Là những dữ liệu có thể chỉnh sửa được quy định sẵn cho blogspot và dữ liệu đó có thể được gọi ra để sử dụng thông qua “**thẻ gọi dữ liệu**”. Những dữ liệu này có thể là tiêu đề bài viết, mô tả, hình ảnh, bình luận, hoặc những dữ liệu tương tự mà các ông đã thiết lập thông qua mục “**Settings**” hoặc trong mỗi bài viết, trang tĩnh mà các ông tạo. Mặc dù vẫn có một số dữ liệu mặc định không thể chỉnh sửa, tuy nhiên ta sẽ bỏ qua nó.

Thuộc tính của thẻ

Mỗi thẻ html đều có thể chứa một hoặc nhiều thuộc tính khác nhau. Ví dụ thẻ `<div class='codemai'/>` thì `class` được xem là một thuộc tính và `codemai` là một giá trị của thuộc tính `class`.

Thẻ gọi dữ liệu

Là những thẻ được dùng để gọi dữ liệu ra từ blog. Cấu trúc chung của những thẻ này là:

```
<data:name1.name2/>
```

Trong đó `data:name1` là phần bắt buộc phải có, `name2` hoặc thậm chí `name3 name4` ... chỉ cần thiết nếu `name1` là một mảng có chứa nhiều phần tử con và các ông muốn gọi phần tử con có tên `name2` thuộc mảng `name1` ra ngoài.

Section (Mục)

Mục là nơi chứa một nhóm các phần tử xác định nào đó. Ví dụ các ông có ba cuốn sách toán cùng để chung vào một ngăn. Vậy ngăn đó có thể gọi là mục toán. 😊

Còn có cách hiểu đơn giản như sau:

- Các ông có một mẫu layout gồm bốn phần cơ bản đó là header, main, sidebar và footer như sau

HEADER - nơi hiển thị logo, menu và linh tinh	
MAIN - hiển thị nội dung chính	SIDEBAR - hiển thị thanh bên
FOOTER - nơi chứa tuyên ngôn chống trộm cắp	

- Vậy thì mỗi phần như vậy có thể hiểu là một mục. Và mục có thể chứa mục con.

Thẻ tạo mục

Là từ viết tắt tên gọi của thẻ `b:section`

Là thẻ có nhiệm vụ tạo, khai báo rằng nó chứa các tiện ích widget. Và nó phải là chính nó, một thẻ `b:section` không được phép chứa một thẻ `b:section` khác và càng không được phép ở trong một thẻ `b:section` khác (tất nhiên 😊). Cụ thể các ông có thể chuyển sang phần thứ hai của tài liệu.

Tên gọi widget xác định

Tên gọi widget xác định hay tên gọi xác định là định nghĩa của tớ, những tên gọi này được sử dụng khi làm việc với “**Tiện ích widget**”, được dùng xác định loại chức năng tiện ích.

Tiện ích widget

Tiện ích widget được dùng khi các ông muốn tạo một phần tử cụ thể cho trang. Và mỗi tiện ích widget sẽ có một chức năng riêng. Hiện tại blogger phân các tiện ích theo chức năng bao gồm những phần sau đây :

Blog

Đây là phần quan trọng nhất khi thiết kế một giao diện cho blogspot. Nó đảm nhiệm hiển thị nội dung của các trang như trang tĩnh, trang bài viết, hiển thị bài viết lên trang chủ, trang tìm kiếm,...

Tên gọi xác định: **Blog**

Blog Archive

Dùng để hiển thị những bài viết có trên Blog theo thời gian. Có thể xem nó là một mục lục bài viết của các ông

Tên gọi xác định: **BlogArchive**

Blogger Button

Không cần thiết, dùng để hiển thị logo của blogger (không phải logo của các ông mà là logo blogger của chú google).

Tên gọi xác định: **BloggerButton**

Blog List

Dùng để hiển thị danh sách các blog mà hiện các ông đang quản lí.

Tên gọi xác định: **BlogList**

Search

Như tên gọi, dùng để hiển thị khung tìm kiếm

Tên gọi xác định: **BlogSearch**

Contact Form

Khung liên hệ để người xem có thể gửi thư cho các ông

Tên gọi xác định: **ContactForm**

Featured Post

Dùng hiển thị bài đăng nổi bật, tức là cái bài sẽ hiển thị thù lù trước mặt người xem của các ông cho dù nó được đăng từ thế kỉ trước miễn là nó được chọn làm bài đăng nổi bật.

Tên gọi xác định: **FeaturedPost**

Feed

Hiển thị nguồn cấp dữ liệu từ RSS hoặc Atom nếu được thiết lập

Tên gọi xác định: **Feed**

Follow By Email

Cho phép người xem đăng kí nhận bài viết của các ông mỗi khi các ông post bài mới bằng email

Tên gọi xác định: **FollowByEmail**

Header

Đơn giản là nó dùng để hiển thị header của các ông bao gồm logo của các ông và một số thứ linh tinh như tiêu đề khác.

Tên gọi xác định: **Header**

HTML

Nếu các ông muốn tạo một tiện ích mới chứa các loại mã như [HTML](#), [Javascript](#), [CSS](#) hoặc đơn giản là các ông muốn tạo một tiện ích mới nhưng không biết phân loại nó như nào.

Tên gọi xác định: **HTML**

Image

Dùng để hiển thị một hình ảnh mà các ông muốn lên blog

Tên gọi xác định: **Image**

Label

Liệt kê tất cả nhãn mà các ông đã gán cho toàn bộ bài viết

Tên gọi xác định: **Label**

Link List

Hiển thị danh sách liên kết đến những siêu văn bản mà các ông muốn

Tên gọi xác định: **LinkList**

Page List

Hiển thị danh sách các trang tính mà blog của ông có

Tên gọi xác định: **PageList**

Popular Posts

Hiển thị những bài viết có lượt xem lớn nhất

Tên gọi xác định: **PopularPosts**

Profile

Hiển thị thông tin về các ông, tương tự như mục giới thiệu về bản thân ông vậy đó

Tên gọi xác định: **Profile**

Report Abuse

Dùng hiển thị liên kết để người xem có thể báo cáo nội dung khi họ cảm thấy nội dung đó không phù hợp với thuần phong mỹ tục.

Tên gọi xác định: **ReportAbuse**

Stats

Hiển thị số lượt xem blog của các ông

Tên gọi xác định: **Stats**

Subscribe

Cho phép người xem đăng kí bài viết, bình luận mới thông qua những trình đọc nguồn cấp dữ liệu phổ biến đơn cử như là **RSS**

Tên gọi xác định: **Subscribe**

Text

Hiển thị một văn bản nào đó mà các ông muốn lên blog

Tên gọi xác định: **Text**

TextList

Tương tự như tiện ích "**Text**" nhưng hiển thị nhiều text hơn

Tên gọi xác định: **TextList**

Translate

Cho phép thêm nút Google dịch vào blog của các ông để người nước ngoài có thể hiểu được nội dung bài viết của các ông

Tên gọi xác định: **Translate**

Wikipedia

Cho phép người xem của các ông tìm kiếm nhanh một nội dung nào đó trên Wikipedia

Tên gọi xác định: **Wikipedia**

Mảng

Là loại dữ liệu chứa nhiều phần tử (đối tượng) cùng chung kiểu trong nó. Và các phần tử có thể chứa các phần tử con.

Mảng và object là tương đương nhau. Tuy nhiên chỉ có mảng mới có thể sử dụng cho vòng lặp và mảng có thể chứa nhiều object. Các ông nhớ chú ý.

Nhớ đơn giản cứ là mảng gồm nhiều object (đối tượng). Còn object chỉ chứa các thông tin liên quan đến nó. Ví dụ, lớp ông là một mảng còn các ông mỗi ông là một object. Trong đó các ông có chứa thông tin riêng của mình bao gồm họ tên, chỉ số iq,...

Mảng gọi dữ liệu bằng cách tham chiếu chỉ số. Ví dụ thầy giáo gọi học sinh thứ 24 thì ông thứ 24 trong danh sách lên bảng. Còn object tham chiếu đến dữ liệu bằng tên của dữ liệu. Ví dụ khi ông 24 lên bảng, thầy giáo bảo "chiều cao" thì đó là ông đang tham chiếu đến dữ liệu chiều cao của ông 24 đó.

Và một số tiện ích khác, tuy nhiên trong tài liệu này tớ sẽ chỉ giới thiệu một chút về những tiện ích được sử dụng thường xuyên nhất. Và tùy theo mục đích sử dụng mà các ông sẽ chọn tiện ích phù hợp trong quá trình viết giao diện cho blogspot.

Đôi lời của tớ: Thực ra ngoài trừ tiện ích Blog và Popular Posts ra thì tớ đều chọn tiện ích HTML để sử dụng.

Giao diện cơ bản đầu tiên

Những điều cần thiết phải có khi thiết kế một template blogspot

Yếu tố bắt buộc của mỗi giao diện

Không kể trường hợp nào, khi thiết kế giao diện cho blogspot. Phải đảm bảo ít nhất rằng:

- Bên trong thẻ `<head>` phải có thẻ `<b:skin><![CDATA[]]></b:skin>`

```
<head>
  <b:skin><![CDATA[
    /* Đây sẽ là nơi chứa CSS */
  ]]></b:skin>
</head>
```

- Bên trong thẻ `<body>` phải có chứa ít nhất một thẻ `b:section`.

`<b:skin><![CDATA[]]></b:skin>` được dùng để chứa css. Mặc dù các ông vẫn có thể chứa css bên trong thẻ `<style>`. Mặc dù vậy, sau này, một số css bắt buộc sẽ phải đặt bên trong thẻ `<b:skin>`

Giao diện đơn giản nhất

Dưới đây là code của một giao diện đơn giản nhất mà blogger có thể hiểu và hiển thị được:

```

<?xml version="1.0" encoding="UTF-8" ?>
<html>
  <head>
    <b:skin>
      <![CDATA[
        /* nơi chứa css */
      ]]>
    </b:skin>
  </head>
  <body>
    <b:section class='codemai' id='codemai' showaddelement='yes' />
  </body>
</html>

```

Chú ý : `<?xml version="1.0" encoding="UTF-8" ?>` có thể có hoặc không. Blogger sẽ tự thêm vào cho các ông.

Cấu trúc một giao diện blogspot

Một giao diện blogspot được thiết kế theo cấu trúc chung nào?

Đọc xong mục “**Giao diện có bản đầu tiên**” là các ông đã nắm trong mình cấu trúc cơ bản nhất khi thiết kế một giao diện cho blogspot. Nhưng nó mới chỉ đơn giản là hiển thị một trang trắng tinh không có nội dung.

Vậy trong mục này tớ sẽ đi vào phần hiển thị nội dung web, trong đó tập trung vào thẻ `b:section`, tiện ích widget. Chỉ cần nắm chắc hai nội dung này là các ông cơ bản đã thiết kế được một giao diện cơ bản hoàn chỉnh như những giao diện ngoài kia và tự tin khi SEO, phát triển nội dung cạnh tranh với đối thủ.

Thẻ `b:skin` trong thẻ `head`

Như đã nói **b:skin** là một phần bắt buộc khi thiết kế giao diện blogspot. Nó có nhiệm vụ chứa CSS. Và khi người dùng truy cập vào, Blogger sẽ hiển thị thẻ **<b:skin** dưới dạng một thẻ **<style>**

Thẻ b:section nâng cao

Nắm chắc khái niệm "thẻ **b:section**" và "mục" trước khi đi vào phần này

HEADER - nơi hiển thị logo, menu và linh tinh	
MAIN - hiển thị nội dung chính	SIDEBAR - hiển thị thanh bên
FOOTER - nơi chứa tuyên ngôn chống trộm cắp	

Khi các ông muốn sử dụng một tiện ích widget nào đó thì thẻ **b:section** bắt buộc phải có. Thẻ **b:section** có nhiệm vụ tạo một mục chứa các widget đó. Và thẻ **b:section** không được phép chứa cũng như bị chứa trong các thẻ **b:section** khác.

Để ví dụ cho việc thẻ **b:section** không được phép chứa cũng như bị chứa trong các thẻ **b:section** khác thì các ông hãy nhìn lên layout phía trên. Thẻ **b:section** có thể tạo ra mục **Header**, mục **Main**, mục **Sidebar**, mục **Footer**,... Vì nó là các mục riêng biệt, không ai chứa ai. Tuy nhiên với thẻ **b:section** như sau sẽ báo lỗi:

THẺ b:section		
THẺ b:section		THẺ b:section
THẺ b:section	THẺ b:section	

THẺ b:section

Cú pháp đúng:

```

4
5 <?xml version="1.0" encoding="UTF-8" ?>
6 <html>
7   <head>
8     <b:skin><![CDATA[
9       ]]></b:skin>
10  </head>
11  <body>
12    <b:section class='main-content' id='blog' showaddelement='yes'>
13    </b:section>
14  </body>
15 </html>

```

Cú pháp sai:

```

4
5 <?xml version="1.0" encoding="UTF-8" ?>
6 <html>
7   <head>
8     <b:skin><![CDATA[
9       ]]></b:skin>
10  </head>
11  <body>
12    <b:section class='main-content' id='blog' showaddelement='yes'>
13      <!-- Sai do có lòng thêm một thẻ b:section -->
14      <b:section class='error' id='error' showaddelement='yes' />
15    </b:section>
16  </body>
17 </html>

```

Để ví dụ cho việc sử dụng thẻ b:section, các ông chỉ cần nhớ là thẻ b:section được sử dụng khi các ông muốn tạo một mục trong layout mà nó có chứa một hoặc nhiều tiện ích widget. Ví dụ các ông muốn sử dụng tiện ích **Popular Post** thì bắt buộc phải đặt tiện ích widget **Popular Posts** trong thẻ b:section.

Cú pháp thẻ b:section

Một thẻ `b:section` có cú pháp đầy đủ như sau:

```
<b:section class='tên_lớp'
    cond='thẻ_chứa_điều_kiện'
    id='tên_id'
    maxwidgets='số'
    name='tên_hiển_thị_trong_bố_cục'
    preferred='YES|NO|TRUE|FALSE'
    showaddelement='YES|NO'>

</b:section>
```

Trong đó thuộc tính **id** là bắt buộc, **id** này phải tuân thủ theo quy tắc đặt tên của HTML, một trang không được phép có hơn 1 id với tên giống nhau, tức tên của id phải là duy nhất những thuộc tính khác có thể có hoặc không.

Giá trị thuộc tính	Mô tả	Đặc hiệu
tên_lớp	<ul style="list-style-type: none"> - Là tên lớp tuân thủ quy tắc đặt tên tương tự khi thiết kế html. - Là một chuỗi kí tự bất kì không được phép đứng đầu bằng một số. - Chịu ảnh hưởng của CSS - Chịu ảnh hưởng của Javascript 	<ul style="list-style-type: none"> - Tên class mặc định là section
thẻ_chứa_điều_kiện	<ul style="list-style-type: none"> - Là điều kiện để mục có thể hiển thị. - Là một trong những thẻ điều kiện [Xem mục sau] - Là thẻ có dữ liệu trả về là true hoặc false 	<ul style="list-style-type: none"> - Mục hiển thị khi giá trị là true và ngược lại - Mặc định: thuộc tính này không có mặc định, chỉ có hiệu lực khi được thiết lập
id	<ul style="list-style-type: none"> - Là một id bình thường như khi viết HTML. - Là một chuỗi kí tự bất kì không 	<ul style="list-style-type: none"> - Là thuộc tính bắt buộc

	<p>được phép đứng đầu bằng một số.</p> <ul style="list-style-type: none"> - Chịu ảnh hưởng của CSS và Javascript - Id là duy nhất 	
số	<ul style="list-style-type: none"> - Là một số tự nhiên 	<ul style="list-style-type: none"> - Mặc định là "1" - Là số tiện ích tối đa mà thẻ b:section được phép chứa. - Nếu số tiện ích đạt bằng "số" thì nút thêm tiện ích trong phần bộ cục sẽ tự ẩn.
tên_hiển-thị_trong_bố_cục	<ul style="list-style-type: none"> - Là tên sẽ hiển thị trong phần bố cục 	<ul style="list-style-type: none"> - Với giá trị thuộc tính tên_hiển-thị_trong_bố_cục là halo thì trong phần bố cục. Mục sẽ được hiển thị là: <div style="border: 1px solid #ccc; padding: 5px; text-align: center; margin: 10px 0;"> <p>halo</p> </div> <ul style="list-style-type: none"> - Mặc định: là tên của id
preferred='YES NO'	<ul style="list-style-type: none"> - Xác định xem mục này có phải là mục chính không 	<ul style="list-style-type: none"> - Không cần thiết và không có mặc định
showaddelement='YES NO'		<ul style="list-style-type: none"> - Nếu "YES" thì mục sẽ được hiển thị cho người xem và ngược lại

Chỉ có **id** với **class** được hiển thị. Các thuộc tính khác đều là giá trị thiết lập và không được hiển thị ra do đó khi người xem truy cập blog các ông, xem như bỏ qua các thiết lập **cond** hoặc **showaddelement** thì thẻ **b:section** sẽ được blog hiển thị dưới dạng sau:

```
<div class="section tên_class" id="tên_id" ></div>
```

Như vậy, cho đến sau cùng, **b:section** sẽ được chuyển về là một thẻ **div**, vì vậy lời khuyên là các ông hãy chỉ dùng thẻ **b:section** khi muốn hiển thị một tiện ích widget. Ngược lại, hãy dùng những thẻ khác, ví dụ **div**.

Chỉ dùng **b:section** khi có ít nhất một tiện ích cần hiển thị

Đặc điểm thẻ **b:section**

Bên trong thẻ **b:section** chỉ được phép chứa các thẻ tiện ích **b:widget**. Nếu không phải là một thẻ tiện ích. Blogger sẽ tự loại bỏ nó khi cập nhật giao diện.

Ví dụ cú pháp đúng:

```

3
4
5 <?xml version="1.0" encoding="UTF-8" ?>
6 <html>
7   <head>
8     <b:skin><![CDATA[
9       ]]></b:skin>
10  </head>
11  <body>
12    <b:section class='main-content' id='blog' showaddelement='yes'>
13      <b:widget id='Blog1'
14        locked='true'
15        title='Danh sách bài viết và bài viết'
16        type='Blog'
17        version='2'
18        visible='true'>
19    </b:section>
20  </body>
21 </html>

```

Ví dụ cú pháp sai:

```

5  <?xml version="1.0" encoding="UTF-8" ?>
6  <html>
7      <head>
8          <b:skin><![CDATA[
9              ]]></b:skin>
10     </head>
11     <body>
12         <b:section class='main-content' id='blog' showaddelement='yes'>
13             Dòng text này sẽ bị loại bỏ khi cập nhật giao diện
14             <b:widget id='Blog1'
15                 locked='true'
16                 title='Danh sách bài viết và bài viết'
17                 type='Blog'
18                 version='2'
19                 visible='true'>
20         </b:section>
21     </body>
22 </html>

```

Ví dụ sử dụng

Mã code khi viết:

```

5  <?xml version="1.0" encoding="UTF-8" ?>
6  <html>
7      <head>
8          <b:skin>
9              <![CDATA[
10                 /* nơi chứa css */
11                 ]]>
12          </b:skin>
13     </head>
14     <body>
15         <b:section class='codemai' cond='data:view.isError' id='codemai' />
16     </body>
17 </html>

```

Mã code khi người xem truy cập vào trang 404 (`data:view.isError` sẽ trả về true nếu trang là trang 404):

```

3
4
5 <?xml version="1.0" encoding="UTF-8" ?>
6 <html>
7   <head>
8     <style id='page-skin-1' type='text/css'>
9
10    </style>
11  </head>
12  <body>
13    <div class='section codemai' id='codemai' />
14  </body>
15 </html>

```

Tên_class và **tên_id** đều chịu ảnh hưởng của CSS và JS như trong HTML thông thường.

Thẻ tiện ích b:widget

Là thẻ có nhiệm vụ khai báo và hiển thị các tiện ích đã được liệt kê ở mục “**Một số khái niệm cơ bản**” phần “**Tiện ích widget**”.

Cú pháp thẻ tiện ích b:widget

Cú pháp đầy đủ của thẻ tiện ích widget:

```

<b:widget id='widget_id'
  cond='true|false'
  locked='true|false'
  version='1|2'
  mobile='true|false'
  title='widget_title'
  visible='true|false'
  type='widget_type'>

```

</b:widget>

Trong đó, ngoại trừ những thuộc tính tương tự thẻ b:section thì :

Giá trị thuộc tính	Mô tả	Đặc hiệu
widget_id	<ul style="list-style-type: none"> - Để xác định id của tiện ích - Tuân thủ theo quy tắc đặt tên của id trong html - Chịu ảnh hưởng của CSS và JS - Tên id là duy nhất - Cấu trúc tên là "tên gọi xác định của tiện ích" + "một số duy nhất từ 1 đến 999" 	
locked='true false'	<ul style="list-style-type: none"> - Cho phép gỡ tiện ích thông qua "Bố cục" hay không 	<ul style="list-style-type: none"> - Khi true được đặt, ngoài việc chỉnh sửa cơ bản ra thì không thể xóa hay di chuyển tiện ích thông qua phần "Bố cục" được nữa và ngược lại. Nhưng vẫn có thể trong phần "chỉnh sửa html"
version='1 2'	<ul style="list-style-type: none"> - Không cần quan tâm cái này nó chỉ là xác định phiên bản của tiện ích 	
widget_type	<ul style="list-style-type: none"> - Là tên xác định của widget đã nêu trong phần "Một số khái niệm cơ bản" 	
mobile="true false"	<ul style="list-style-type: none"> - Cho phép tiện ích này hiển thị trên di động hay không 	<ul style="list-style-type: none"> - True là có và ngược lại - Chỉ áp dụng được khi "giao diện tùy chỉnh cho di động" trong phần cài đặt được bật
widget_title	<ul style="list-style-type: none"> - Tương tự name trong thẻ b:section 	
visible='true false'	<ul style="list-style-type: none"> - Tương tự như showaddelement của thẻ b:section 	

Khi truy cập, tương tự như thẻ **b:section**, Blogger sẽ hiển thị thẻ **b:widget** dưới dạng là:

```
<div class="widget widget_type" data-version="widget_version" id="widget_id">
```

Đặc điểm của thẻ **b:widget**

- Thẻ **b:widget** phải được chứa trong thẻ **b:section**
- Thẻ **b:widget** chỉ có thể chứa thẻ **b:widget-settings** và các thẻ **b:includable**

Ví dụ cú pháp đúng:

```
5 <?xml version="1.0" encoding="UTF-8" ?>
6 <html>
7   <head>
8     <b:skin><![CDATA[
9       ]]></b:skin>
10  </head>
11  <body>
12    <b:section class='main-content' id='blog' showaddelement='yes'>
13      <b:widget>
14        <b:includable/>
15      </b:widget>
16    </b:section>
17  </body>
18 </html>
```

Ví dụ cú pháp sai:


```

4
5 <?xml version="1.0" encoding="UTF-8" ?>
6 <html>
7   <head>
8     <b:skin><![CDATA[
9     ]]></b:skin>
10  </head>
11  <body>
12    <b:section class='main-content' id='blog' showaddelement='yes'>
13      <b:widget>
14        <span>Một thẻ html ãt ơ nào đó</span>
15        hoặc một dòng text ãt ơ nào đó
16        <b:includable/>
17      </b:widget>
18    </b:section>
19  </body>
20 </html>

```

Ở cả hai cú pháp tớ đã lược bỏ tất cả những thuộc tính bắt buộc trên các thẻ `b:section` và `b:widget`. Mặc dù vậy là sai nhưng sẽ tiết kiệm không gian ảnh làm ví dụ cho các ông.

- Mỗi thẻ `b:widget` với mỗi loại tiện ích xác định trong số những tiện ích đã được liệt kê đều có một mẫu hiển thị mặc định riêng. Do đó chỉ với cấu trúc đơn giản như sau là các ông đã có thể liệt kê được danh sách bài viết trong blog và hiển thị bài viết.

```

4
5 <?xml version="1.0" encoding="UTF-8" ?>
6 <html>
7   <head>
8     <b:skin><![CDATA[
9     ]]></b:skin>
10  </head>
11  <body>
12    <b:section class='main-content' id='blog' showaddelement='yes'>
13      <b:widget      id='Blog1'
14                  locked='true'
15                  title='Danh sách bài viết và bài viết'
16                  type='Blog'
17                  version='2'
18                  visible='true' />
19    </b:section>
20  </body>
21 </html>

```

Theo mặc định như vậy, Blogger sẽ thêm mọi thứ cần thiết liên quan đến tiện ích các ông đang đề cập. Và các ông chỉ cần dùng CSS để tái định hình lại. Tuy nhiên, trong tài liệu này tớ chỉ viết cách viết lại hoàn toàn giao diện cho các ông.

Một số tiện ích là mặc định không thể chỉnh sửa hay ghi đè được

Ví dụ sử dụng

Như trên

Thẻ chứa dữ liệu hiển thị b:includable

Thẻ **b:includable** là thẻ dùng để chứa dữ liệu và các ông có thể dùng thẻ b:include để tham chiếu và gọi dữ liệu từ thẻ b:includable ra. Mỗi widget đều có một thẻ b:includable bất di bất dịch đó là:

```
<b:includable id='main' />
```

Thẻ này sẽ là thẻ hiển thị nội dung của tiện ích widget. Có nghĩa là những thẻ b:includable khác dù cùng thuộc một tiện ích nhưng nó sẽ không hiển thị cho đến khi nó được gọi ra từ bên trong

thẻ `<b:includable id='main'/>` hoặc từ thẻ `b:includable` được gọi bên trong `<b:includable id='main'/>`

Ví dụ, trong widget Blog, có những thẻ `b:includable` như sau:

```
<b:includable id='main'>
  <b:include name='codemai-1'/>
</b:includable>

<b:includable id='codemai-1'>
  Tôi được gọi nên dòng text này sẽ hiển thị
</b:includable>

<b:includable id='codemai-2'>
  Tôi không được gọi nên tôi nằm im ở đây
</b:includable>
```

Như vậy nội dung chứa bên trong thẻ `b:includable` có id là `codemai-1` sẽ hiển thị còn `codemai-2` thì không. Còn như sau thì nội dung ở cả hai thẻ `b:includable` mang id là `codemai-1` và `codemai-2` đều hiển thị.

```
<b:includable id='main'>
  <b:include name='codemai-1'/>
</b:includable>

<b:includable id='codemai-1'>
  Tôi được gọi nên dòng text này sẽ hiển thị
  <b:include name='codemai-2'/>
</b:includable>

<b:includable id='codemai-2'>
  Tôi đã được gọi từ codemai-1 và codemai-1
  được gọi ra từ main nên tôi sẽ hiển thị
</b:includable>
```

Và ta có kết quả là :

Tôi được gọi nên dòng text này sẽ hiển thị Tôi đã được gọi từ codemai-1 và codemai-1 được gọi ra từ main nên tôi sẽ hiển thị

Đặc điểm thẻ `b:includable`

- Các ông có thể xem thẻ `b:includable` như là một container chứa những đoạn mã dùng thực hiện một chức năng nào đó, và khi được gọi bởi thẻ `b:include` thì đoạn mã đó sẽ được trả về cùng dữ liệu truyền vào
- Thẻ `b:includable` phải được chứa bên trong thẻ `b:widget`
- Mặc định mỗi tiện ích đều có một mẫu hiển thị riêng nhưng các ông có thể thay đổi được
- Thẻ `b:includable` không được chứa thẻ `b:includable` khác

Cấu trúc của thẻ `b:includable`

Cấu trúc đầy đủ của một thẻ `b:includable` là:

```

<b:includable id='includable_id'
    var='tên_biến'>
</b:includable>
  
```

Giá trị thuộc tính	Mô tả
<code>includable_id</code>	<ul style="list-style-type: none"> - Là id được dùng để xác định và gọi thẻ <code>b:includable</code> từ thẻ <code>b:include</code> - <code>Includable_id</code> phải là duy nhất trong mỗi tiện ích widget
<code>tên_biến</code>	<ul style="list-style-type: none"> - Là biến chứa dữ liệu được truyền vào từ thẻ <code>b:include</code> . - Là một biến cục bộ.

Ví dụ sử dụng

Thẻ **b:includable** sau sẽ hiển thị dòng text 'Hello Vietnam' khi được gọi:

```
<b:includable id='main'>
  <b:include name='hi' />
</b:includable>

<b:includable id='hi'>
  Hello Vietnam
</b:includable>
```

Thẻ gọi dữ liệu hiển thị b:include

Nếu **b:includable** là container chứa các đoạn mã thì **b:include** sẽ là người lấy mã từ container ra để sử dụng.

Đặc điểm thẻ b:include

- Thẻ **b:include** không được phép gọi thẻ **b:includable** có id là **abc** bên trong chính nó.
- Thẻ **b:include** không được gọi dữ liệu từ thẻ **b:includable** có id là **main**
- Thẻ **b:include** chỉ gọi được những thẻ **b:includable** cùng thuộc một tiện ích (Có ngoại lệ và tứ sẽ trình bày ở phần sau)

Cấu trúc cú pháp của thẻ b:include

```
<b:include cond='điều_kiện'
data='dữ_liệu_cần_truyền_cho_b:includable'
```

```
name='id_thẻ_b:includable_cần_gọi'/>
```

Giá trị thuộc tính	Mô tả	Đặc hiệu
điều_kiện	<ul style="list-style-type: none"> - Điều kiện để thẻ b:includable được gọi. - Nếu điều kiện không thỏa mãn thì dữ liệu sẽ không được lấy ra 	<ul style="list-style-type: none"> - Không bắt buộc
dữ_liệu_cần_truyền_cho_b:includable	<ul style="list-style-type: none"> - Là một trong số những dữ liệu được lấy từ “thẻ dữ liệu” nhưng loại đi tiền tố “data:” mà tớ sẽ trình bày trong phần tiếp - Là một mảng, một string, ... bất kì - Thẻ b:includable sẽ nhận dữ liệu và lưu nó vào tên_biến 	<ul style="list-style-type: none"> - Không bắt buộc
id_thẻ_b:includable_cần_gọi	<ul style="list-style-type: none"> - Là id của thẻ b:includable mà các ông cần tham chiếu đến để gọi mã từ nó. 	<ul style="list-style-type: none"> - Bắt buộc

Ngoài ra các ông còn có thể truyền dữ liệu từ thẻ **b:includable** chứa thẻ **b:include** tới một thẻ **b:includable** khác.

```

37
38 <b:includable id='code1' var='bien1'>
39     <b:include name='code2' data='bien1' />
40 </b:includable>
41
42 <b:includable id='code2' var='bien2'>
43     Mã
44 </b:includable>
45

```

Như hình trên thì thẻ **b:include** sẽ lấy dữ liệu từ **bien1** của thẻ **b:includable** có id là **code1** để truyền tới cho thẻ **b:includable** có id là **code2**. Và lúc này **bien2** của thẻ **b:includable** có id là **code2** sẽ nhận dữ liệu và mang dữ liệu của **bien1**.

Ví dụ sử dụng

Tớ sẽ gọi thẻ **b:includable** có id là **codemai** và truyền cho nó dữ liệu là một mảng các phần tử.

```
<b:include data='["chuối","đào","ớt"]' name='codemai' />
```

Tớ sẽ gọi thẻ **b:includable** có id là **codemai** và truyền cho nó dữ liệu là một mảng các phần tử với điều kiện trang hiển thị là trang chủ.

```
<b:include cond='data:view.isHomepage' data='["chuối","đào","ớt"]' name='codemai' />
```

Ví dụ cụ thể:

```

4
5 <?xml version="1.0" encoding="UTF-8" ?>
6 <html>
7 <head>
8 <b:skin><![CDATA[
9 ]]></b:skin>
10 </head>
11 <body>
12 <b:section class='main-content' id='blog' showaddelement='yes'>
13   <!-- Sai do có lỗi thêm một thẻ b:section -->
14   <b:widget id='Blog1' locked='false' title='Code Vô Đới (Tiêu đề)'
15     type='Blog' version='2' visible='true'>
16     <!-- Hiển thị chính -->
17
18     <b:includable id='main'>
19       <b:include name='hi' />
20     </b:includable>
21
22     <b:includable id='hi'>
23       Hình doremon sẽ hiển thị:
24       <img src='https://file.hstatic.net/1000159991/file/doremon2-min_208c99d4e9ab4a1e98e5d8ba58a9d6e0_grande.jpg' />
25     </b:includable>
26
27   </b:widget>
28 </b:section>
29 </body>
30 </html>

```

Kết quả:

 Apps  Kill tab  Popular  AliExpress  Import bookmarks

Hình doremon sẽ hiển thị:



Một số loại trang trong blogspot

Blogspot phân thành là những loại trang sau:

- Trang chủ, trang tìm kiếm : **index**
- Trang lưu trữ: **archive**
- Trang bài viết : **post**
- Trang tĩnh: **page**
- Trang lỗi: **error_page**

Thẻ dữ liệu và điều kiện trong blogspot

Thẻ dữ liệu nguồn sống của mọi blog

Tùy vào từng tiện ích mà mỗi thẻ dữ liệu có thể gọi được từ tiện ích ấy hay không. Hay các ông có thể hiểu đơn giản là mỗi tiện ích để chỉ cho phép một số thẻ gọi dữ liệu nhất định. Và thực chất, thẻ điều kiện đa số được gọi ra từ thẻ dữ liệu, nhưng giá trị trả về là **true** và **false**. Nên trong những thẻ dữ liệu mà tớ trình bày sau đây, nếu bắt gặp cụm từ “**đúng hay không**” hoặc tương tự thì ắt hẳn nó chính là thẻ điều kiện.

Cấu trúc thẻ gọi dữ liệu

Cú pháp chung để gọi dữ liệu là:

```
<data:name1.name2/>
```

Và tùy thuộc vào độ sâu của dữ liệu mà số **namex** có thể tăng lên. Ở đây tớ trình bày độ sâu của dữ liệu dưới dạng cây và chỉ trình bày một số thẻ dữ liệu thuộc những tiện ích phổ biến - những tiện ích còn lại khá ít người sử dụng hoặc sử dụng mặc định của blogger và tớ cũng không am hiểu rõ về chúng nên tớ không thể trình bày được .

Thẻ dữ liệu chung

Đây là những thẻ dữ liệu có thể gọi từ bất kì đâu trong blogspot.

blog

- title: tiêu đề của blog.
- pageType: Loại của trang hiện tại
- url: địa chỉ trang hiện tại
- languageDirection : ngôn ngữ của blog đang dùng
- blogspotFaviconUrl : địa chỉ hình thu nhỏ của trang web
- homepageUrl : địa chỉ trang chủ
- metaDescription : mô tả của trang
- pageName: tiêu đề của trang hiện tại.
- encoding: mã hóa đang được sử dụng trong blog
- feedLinks: các liên kết nguồn cấp dữ liệu hiện có.
- enabledCommentProfileImages : người bình luận có mở hồ sơ cho mọi người hay là không

Vậy cú pháp gọi dữ liệu ở đây sẽ là: `<data:blog.title/>` để lấy tiêu đề của blog

Với những trường hợp, ví dụ tờ chỉ ghi đơn như "blog" hoặc "view" mà không ghi thêm bất kì chú thích nào khác. Có nghĩa nó là dữ liệu không được đọc lập, bắt buộc cần tham chiếu đến một dữ liệu cụ thể. Ví dụ `<data:blog.title/>` là đúng nhưng `<data:blog/>` là sai.

view

- title : tiêu đề trang hiện tại
- Url : địa chỉ trang hiện tại
- featuredImage : hình ảnh nổi bật của trang và thường là hình ảnh đầu tiên trong mỗi bài viết
- isHomepage : trang có phải là trang chủ hay không
- isPost: trang có là trang bài viết hay không
- isPage: trang là trang tĩnh đúng hay không
- isMultipleItems : trang có phải là trang thuộc trang chủ, trang lưu trữ và trang nhãn hay không
- isLabelSearch : trang có phải là trang tìm kiếm nhãn hay không
- isSearch : có phải là trang tìm kiếm nhãn hoặc trang tìm kiếm hay không

- isError: trang có phải là trang lỗi hay không.
- isSingleItem : trang có phải là trang thuộc trang bài viết hoặc trang tĩnh hay không
- isArchive : trang có phải là trang lưu trữ hay không
- search
 - label : trả về tên nhãn đang tìm kiếm (**search/label/hacking => hacking**)
 - query: trả về tên nội dung mà người xem đang tìm kiếm (**search?q=hacking => hacking**)

Thẻ dữ liệu tiện ích Blog, PopularPosts, và FeaturedPost

Với những tiện ích **Blog**, **PopularPosts**, và **FeaturedPost**. Là những tiện ích dùng để hiển thị bài viết lên blog đó đó các ông buộc phải sử dụng vòng lặp để có thể sử dụng được toàn bộ thẻ dữ liệu. Nếu không, các ông chỉ có thể gọi được những dữ liệu sau ở tất cả tiện ích **Blog**, **PopularPosts**, và **FeaturedPost**:

posts [posts ở đây là một mảng chứa các bài viết được trả về]

- size : kích thước mảng
- length :số lượng phần tử mảng
- empty : kiểm tra có bài viết nào được trả về hay không
- notEmpty : kiểm tra xem có đúng là có bài viết trả về hay không
- any : kiểm tra xem có bất kì bài viết được trả về hay không
- first : lấy bài viết đầu tiên được trả về (không bài là bài các ông post lần đầu khi viết blog đâu nhé)
 - [đọc tiếp ở dưới]
- last : lấy bài viết cuối cùng được trả về (không bài là bài các ông post gần đây nhất khi viết blog đâu nhé)
 - [đọc tiếp ở dưới]

Các ông có thể hiểu ở đây là, nếu không có vòng lặp thì các ông chỉ có thể chọn hoặc bài viết thứ nhất (**first**) hoặc bài viết cuối cùng (**last**) để lấy dữ liệu và hiển thị. Nhưng có vòng lặp thì các ông có thể lặp gọi dữ liệu toàn bộ các bài viết để hiển thị.

Ví dụ các ông có 3 bài viết, nếu không có vòng lặp các ông chỉ hiển thị được duy nhất tiêu đề của một trong ba bài. Nhưng khi có vòng lặp, ông có thể hiển thị được tiêu đề của cả ba bài.

Đối với có vòng lặp:

```
<b:loop values='data:posts' var='tên_biến'>
  <data:tên_biến.name2/>
</b:loop>
```

`data:posts` ở đây đang chứa danh sách những bài viết chuẩn bị được hiển thị. Và vòng lặp ở đây sẽ lặp qua từ bài từng bài một trong danh sách những bài viết đó. Và do vậy, `tên_biến` (tên tùy chọn) ở đây sẽ đại diện cho bài viết hiện đang được vòng lặp chỉ tới.

Ví dụ cho có vòng lặp:

```
<b:loop values='data:posts' var='tên_biến' >
  <data:tên_biến.title/>
</b:loop>
```

Như vậy toàn bộ tiêu đề các bài viết được trả về sẽ hiển thị

Như vậy, `first`, `last`, `tên_biến` ở đây, chúng đều đại diện cho bài viết đang được chỉ tới và vì vậy chúng đều có thể gọi được những dữ liệu sau:

```
<data:posts.first.id/>
<data:posts.last.id/>
<data:tên_biến.id/>
```

Thẻ dữ liệu chỉ dành cho tiện ích Blog

`posts` [Mảng chứa các bài viết]

`tên_biến`, `last`, `first`: [Trong đó `tên_biến` là ở vòng lặp]

- `id`: id của bài viết

- title: tiêu đề bài viết
- body: nội dung của bài viết
- url: địa chỉ của bài viết,
- link: trả về liên kết tiêu đề của bài viết,
- thumbnailUrl: liên kết đến hình ảnh thu nhỏ của bài viết,
- featuredImage: đường dẫn đến hình ảnh đầu tiên của bài viết
 - isResizable: hình ảnh có thể được resize hay không
 - isYoutube: có phải là hình ảnh từ video youtube hay không (Vì blogger là nền tảng của Google nên nếu các ông nhúng video youtube vào đầu bài viết, thì nó sẽ tự động lấy hình ảnh thu nhỏ của video đó làm featuredImage)
 - youtubeMaxResDefaultUrl: liên kết đến hình ảnh lớn nhất của video youtube được nhúng (cho các ông chưa biết thì mỗi video youtube đều được chia làm khá nhiều hình ảnh khác nhau từ mờ mờ cho đến HD và youtubeMaxResDefaultUrl ngụ ý hình ảnh có chất lượng lớn nhất có thể có của video đó)
 - width: chiều rộng ảnh
 - height: chiều dài ảnh
- date: hiển thị ngày đăng bài
 - iso8601: hiển thị theo định dạng iso8601
 - year: năm bài được đăng
 - month: tháng bài được đăng
 - day: ngày bài được đăng
 - dayOfWeek: ngày thứ tuần bài được đăng
 - dayOfMonth: ngày thứ tháng bài được đăng
 - dayOfYear: ngày thứ năm được đăng
- lastUpdated: ngày cập nhật gần đây nhất
 - Gồm tất cả những thứ có tương tự trong thẻ dữ liệu **date** phía trên
- author:
 - name: tên tác giả
 - profileUrl: liên kết đến hồ sơ tác giả
 - aboutMe: lời giới thiệu của tác giả ghi trong hồ sơ

- authorPhoto
 - image: liên kết đến hình ảnh của tác giả
 - width: chiều rộng ảnh
 - height: chiều dài ảnh
- hasJumpLink: boolean,
- adminClass: trả về id của admin như sau : “blog-admin pid-admin-id”. admin-id là một chuỗi số
- postAuthorClass: trả về id của tác giả như sau: “pid-author-id”
- allowComments: bài viết có cho phép bình luận hay không
- allowNewComments: bài viết có cho phép thêm bình luận mới hay không
- noNewCommentsText: hiển thị thông báo là một dòng text với nội dung là chưa có cái bình luận nào hết nếu đúng là chưa có cái bình luận nào hết
- numberOfComments: số lượng bình luận của bài viết hiện tại
- commentsUrl: liên kết đến nơi để bình luận (thường là nó thêm #comments vào phía sau đuôi link)
- commentsUrlOnClick: để lại bình luận cho bài viết đó mà không cần truy cập vào **commentsUrl**
- commentPagingRequired: bài viết có phải đang có hơn 200 bình luận hay không
- hasOlderLinks: trả về true nếu bài viết còn thêm bình luận cũ nữa. Và false nếu ngược lại
- oldLinkClass: đơn giản là nó chỉ trả về một chuỗi đại diện cho một class và có tên trả về là “**unneeded-paging-control**”
- oldestLinkUrl: liên kết đến bình luận cũ nhất có thể.
- olderLinkUrl: liên kết đến bình luận cũ tiếp theo
- hasNewerLinks: trả về true nếu bài viết còn thêm bình luận mới. Và false nếu ngược lại
- newLinkClass: trả về chuỗi “**paging-control**” có thể làm đại diện cho một class
- newerLinkUrl: liên kết đến bình luận mới hơn.
- newestLinkUrl: liên kết đến bình luận mới nhất

Ví dụ bài viết các ông có khoảng 10 bình luận và mỗi lần hiển thị chỉ ba bình luận. Vậy thì newerLinkUrl sẽ tiến đến ba bình luận mới tiếp theo nếu có và olderLinkUrl sẽ lùi về ba bình luận tiếp theo nếu có.

- commentRangeText: hiển thị xem hiện tại đang ở khoảng nào (**number1** - **number2**)
- commentFormIframeSrc: trả về một chuỗi là địa chỉ nhúng của khung bình luận. (Khung bình luận này tương tự như khung bình luận facebook mà các ông nhúng vào trang web. Nó là nơi để các ông thêm bình luận mới cho bài viết)
- embedCommentForm: có phải khung bình luận nhúng đang được bật trong cài đặt blogspot hay không
- showThreadedComments: có phải bình luận đa cấp đang được bật (ý là bình luận mà có thể reply cái bình luận đó)
- commentFeed: liên kết đến feed bình luận
- includeAd: trả về **true** hoặc **false** dựa trên điều kiện có nhúng được quảng cáo hay không
- adNumber: trả về số lượng quảng cáo hiển thị
- emailPostUrl: trả về một chuỗi là địa chỉ nhúng của khung gửi bài viết qua email
- shareUrl: trả về một chuỗi là địa chỉ nhúng của khung chia sẻ bài viết
- reactionsUrl: trả về địa chỉ nhúng của khung thể hiện cảm xúc
- appRpcRelayPath: string,
- location:
 - mapsUrl: liên kết nhúng của bản đồ (nếu bản đồ được cài đặt hiển thị trong bài viết của các ông)
 - name: tên địa chỉ mà các ông chọn trên bản đồ

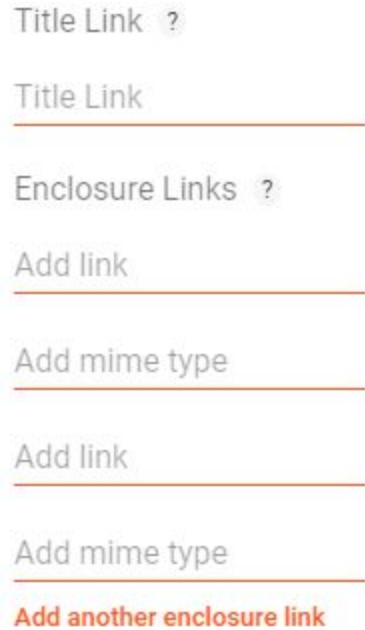
Những dữ liệu được trình bày dưới đây được trả về là một mảng. Do đó các ông cần thêm một vòng lặp nữa để hiển thị toàn bộ dữ liệu. Ví dụ:

```
<b:loop values='data:tên_biến.labels' var='tên_biến_2' >
    <data:tên_biến_2.name/>
</b:loop>
```

- labels : trả về mảng gồm những nhãn của bài viết đang được hiển thị
 - name : tên nhãn
 - url : địa chỉ liên kết đến nhãn
- feedLinks : trả về mảng gồm liên kết đến các feed của bài viết

- name : tên feed
- feedType : loại feed
- mimeType : đuôi feed
- url : liên kết đến feed
- comments : trả về mảng các bình luận hiện có của bài viết
 - id : id của bình luận
 - inReplyTo : trả về true nếu bình luận này được một ai đó trả lời lại
 - cmtBodyIdPostfix : trả về một chuỗi có dạng **_cmt-id** được dùng để nhận dạng bình luận sau này.
 - url : liên kết đến bình luận này
 - body : nội dung bình luận
 - timestamp : thời gian bình luận dạng chuỗi
 - timestampValue : thời gian bình luận dạng số
 - timestampAbs : thời gian bình luận dạng số tuyệt đối
 - author : tên người bình luận
 - authorUrl : liên kết đến hồ sơ người bình luận
 - authorUserType : là người dùng bình luận bằng blogger hay anonymous bằng cách trả về chuỗi **blogger** hoặc **anonymous**
 - authorPhoto
 - url : liên kết đến hình ảnh của người bình luận
 - width : rộng ảnh
 - height : cao ảnh
 - authorAvatarSrc : liên kết đến hình ảnh avatar của tác giả bình luận
 - authorAvatarImage: khá tương tự authorAvatarSrc
 - anchorName : trả về một chuỗi số dùng để nhận dạng tác giả của bình luận
 - deleteUrl : liên kết để xóa bình luận này
 - isDeleted : true nếu bình luận này đã bị xóa
 - adminClass : trả về một chuỗi là tên một class tên **blog-admin** để xác định ai là quản trị và ai là khách truy cập
- enclosures : trả về mảng các liên kết đính kèm hiện có của bài viết
 - url : liên kết đến liên kết đính kèm

- mimeType : loại liên kết đính kèm.



Title Link ?

Title Link

Enclosure Links ?

Add link

Add mime type

Add link

Add mime type

Add another enclosure link

Title link thường được sử dụng khi các ông muốn, thay vì khi click vào tiêu đề bài viết ở trang chủ, trang nhà, trang tìm kiếm,.. thay vì nó dẫn các ông đến bài viết đó, nó sẽ dẫn các ông đến trang web được liên kết bằng title link đó.

Enclosure Links được sử dụng khi các ông muốn thêm một hoặc nhiều hình ảnh để làm một bộ sưu tập cho bài viết nhưng thay vì chèn nó vào trong bài viết thì các ông sử dụng thông qua **Enclosure Links**

Thẻ dữ liệu chỉ dành cho tiện ích PopularPosts

Đối với thẻ tiện ích dành cho **PopularPosts**, những thẻ sau cùng với các phần tử con của nó như đã trình bày ở tiện ích **Blog** là khả dụng có thể gọi dữ liệu:

- Id
- title
- body
- featuredImage
- url
- shareUrl
- emailPostUrl
- commentsUrl
- commentsUrlOnClick
- numberOfComments
- allowComments
- hasJumpLink
- date
- lastUpdated
- labels
- author

Thẻ dữ liệu tiện ích dành riêng cho FeaturedPost

Tương tự như thẻ tiện ích dành cho **PopularPosts**

Thẻ dữ liệu cho tiện ích Header

Cấu trúc dùng: `<data:name1.name2/>`

Ví dụ: `<data:image/>`

- title: tiêu đề trang web được thiết lập trong tiện ích Header

- description: mô tả trang web được thiết lập trong tiện ích Header
- useImage: hình ảnh có được thêm trong tiện ích Header
- imagePlacement: hiển thị một chuỗi mô tả cách mà hình ảnh và tiêu đề được hiển thị bao gồm **BEHIND, REPLACE, BEFORE_DESCRIPTION**
- image: liên kết đến hình ảnh được thiết lập trong tiện ích Header
- sourceUrl: tương tự như **image**
- backgroundPositionStyleStr: một số nước có cách viết từ phải sang trái, và dữ liệu ở thẻ này trả về là **left** hoặc **right** tùy thuộc vào ngôn ngữ mà blog các ông đang sử dụng.
- shrinkToFit: trả về **true** nếu như tùy chọn resize image được thiết lập trong tiện ích Header
- sectionWidth: trả về chiều rộng của thẻ tạo mục đang chứa tiện ích này
- width: chiều rộng tiện ích
- height: chiều cao tiện ích

Configure Header


Blog Title

Blog Description

Image

From your computer:
 No file chosen

From the web. Paste an image URL below.

 **Invalid image url.**

Placement

Behind title and description

Instead of title and description

Have description placed after the image

Shrink to fit
Image will be shrunk to 1059 pixels wide.



Thẻ dữ liệu cho tiện ích HTML

- title: tiêu đề của tiện ích
- content: nội dung của tiện ích - đây chính là phần mã mà các ông thêm vào tiện ích sẽ được hiện ra.

Configure HTML/JavaScript

Title

Content

b *i*   Rich Text

```
<span>Nội dung html sẽ hiển thị</span>
```

Save Cancel Back

Thẻ dữ liệu cho tiện ích Image

- title: tiêu đề hình ảnh
- sourceUrl: địa chỉ hình ảnh
- sourceSet
- link: địa chỉ sẽ mở khi click vào hình ảnh
- width: chiều rộng ảnh
- height: chiều cao ảnh
- sectionWidth
- caption: mô tả hình ảnh

- shrinkToFit

Blogger: aaa - Configure Image - Cent Browser

https://draft.blogger.com/rearrange?blogID=600946353830014230§ionId=bl...

Blogger

Configure Image

Title:

Caption:

Link:
URL that clicking this image will link to (optional)

Image:

From your computer:
 No file chosen

From the web. Paste an image URL below.

⚠ Invalid image url.

Shrink to fit
Image will be shrunk to 1059 pixels wide.

Thẻ dữ liệu cho tiện ích PageList

- title: tiêu đề tiện ích
- links - trả về một mảng gồm
 - href: địa chỉ trang tĩnh
 - title: tiêu đề trang tĩnh
 - id: id trang
 - isCurrentPage: trả về **true** nếu trang đang xem là một trong các trang có trên blog. (trang khác với trang bài viết)

Thẻ dữ liệu cho tiện ích TextList

- title: tiêu đề tiện ích
- sorting: trả về cách mà các item sẽ được sắp xếp (theo bảng chữ cái, chiều dài,...)
- shownum: số lượng các item tối đa sẽ hiển thị (mặc định là không giới hạn)
- items: trả về kết quả là một mảng gồm các item (như hình là mảng gồm : "đối tượng 1", "đối tượng 2")

Configure Text List

Title

Number of items to show in list Leave blank to show all items

Sorting

Add List Item

↑ ↓ đối tượng 2

↑ ↓ đối tượng 1

Thẻ dữ liệu cho tiện ích LinkList

- title
- sorting
- shownum
- links - trả về mảng là danh sách các liên kết
 - target: địa chỉ liên kết
 - name: tên liên kết

Configure Link List

Title	<input type="text" value="tiêu đề tiện ích"/>
Number of items to show in list	<input type="text" value="3"/> Leave blank to show all links
Sorting	<input type="text" value="Don't Sort"/>
New Site Name	<input type="text" value="tiêu đề liên kết"/>
New Site URL	<input type="text" value="địa chỉ liên kết"/>
	<input type="button" value="Add Link"/>
Edit Delete	↑ ↓ tiêu đề liên kết
<input type="button" value="Save"/>	<input type="button" value="Cancel"/> <input type="button" value="Back"/>

Thẻ dữ liệu tiện ích Labels

- title: tiêu đề tiện ích,
- display: trả về chuỗi là thông tin cách hiển thị nhãn (gồm list,cloud)
- showFreqNumbers: có hiển thị số lượng bài viết thuộc một nhãn hay không
- labels - trả về mảng gồm thông tin các nhãn
 - url: địa chỉ nhãn
 - name: tên nhãn
 - count: số lượng bài viết thuộc nhãn (nếu showFreqNumbers là true)

Configure Labels

Title	<input type="text" value="Labels"/>
Show	<input checked="" type="radio"/> All Labels <input type="radio"/> Selected Labels
Sorting	<input checked="" type="radio"/> Alphabetically <input type="radio"/> By Frequency
Display	<input checked="" type="radio"/> List <input type="radio"/> Cloud
	<input type="checkbox"/> Show number of posts per label

Thẻ dữ liệu tiện ích Translate

- title: tiêu đề tiện ích
- pageLanguage: trả về ngôn ngữ của blog các ông
- layout: trả về chuỗi là cách hiển thị của tiện ích

Configure Translate

Title: Translate

Style:

- Vertical
- Horizontal
- Dropdown Only

Preview: Select Language ▼
Powered by Google™ Translate

Buttons: Save, Cancel, Back

Thẻ dữ liệu tiện ích Profile

- title: string,
- team: là một team hay là một tác giả
- showlocation: có hiển thị cá nhân của tác giả hay không
- showaboutme: có hiển thị thông tin giới thiệu về tác giả hay không
- userUrl: địa chỉ đến profile của tác giả chính
- displayName: trả về tên hiển thị của tác giả
- profileLogo: trả về liên kết đến logo của tác giả
- location: địa chỉ các nhân của tác giả chính
- aboutme: thông tin tự giới thiệu của tác giả chính
- viewProfileMsg: chỉ là một chuỗi text thông báo.
- authorPhoto
 - image: hình ảnh tác giả chính
 - width: chiều rộng hình ảnh tác giả chính
 - height: chiều cao hình ảnh tác giả chính

- authors - trả về một mảng gồm thông tin các tác giả trong team
 - userUrl
 - display-name: string,
 - profileLogo: string,
 - authorPhoto
 - image
 - width
 - height

Thẻ dữ liệu cho tiện ích Subscribe

- title: tiêu đề tiện ích
- widgetId: id tiện ích
- isPublic: có công khai hay là không
- atomFeedTxt: chỉ là một thông báo text bình thường
- arrowDropdownImg: trả về liên kết đến hình ảnh cái mũi tên - hình ảnh mặc định - không cần quna tâm
- feedIconImg: trả về logo feed
- imagePathBase: cũng không cần quan tâm nốt
- feeds - trả về mảng các nguồn cấp dữ liệu hiện có
 - type: loại feed
 - title: tên feed
 - url: địa chỉ feed

Nếu dữ liệu trả về là một mảng. Lặp là cần thiết

Như vậy, đến đây, các ông đã biết cơ bản về thẻ gọi dữ liệu và phạm vi hiệu lực của thẻ gọi dữ liệu trong một số tiện ích căn bản và đây sẽ là nền tảng để các ông tiếp tục đến với phần tiếp

theo của tài liệu. Tuy nhiên, trước khi đi vào phần tiếp theo đó, thì các ông cần phải nắm được một số thẻ cơ bản của blogspot.

Một số thẻ căn bản của blogspot

Danh sách những thẻ luôn luôn có trong khi thiết kế blogspot

Thẻ điều kiện trong blogspot

Là một thẻ tối quan trọng trong blogspot. Chức năng tương tự như cấu trúc điều kiện trong các ngôn ngữ lập trình khác.

Cấu trúc thẻ điều kiện là:

```
<b:if cond='điều_kiện'>  
  <!-- khối code sẽ làm việc khi điều kiện đúng -->  
</b:if>
```

điều_kiện đã được nhắc đến ở các phần trước. Sau đây là một số câu điều kiện nâng cao:

```
<b:if cond='điều_kiện'>  
  <!-- khối code làm việc khi điều kiện là đúng-->  
<b:else/>  
  <!-- khối code làm việc khi điều kiện sai -->  
</b:if>
```

Hoặc cấu trúc thẻ đa điều kiện

```

<b:if cond='điều_kiện_1'>
  <!-- khối code sẽ chạy khi điều kiện 1 đúng -->
  <b:elseif cond='điều_kiện_2' />
  <!-- khối code chạy khi điều kiện 1 sai nhưng điều kiện hai đúng -->
  <b:else />
  <!-- khối code chạy khi điều kiện 1 và điều kiện hai cùng sai -->
</b:if>

```

Có thể có nhiều hơn 1 thẻ **b:elseif**

Và điều kiện có thể là một toán tử

Cấu trúc xuất dữ liệu có thể hiểu đúng là **data:name1.name2** và hiểu ngược là thẻ xuất dữ liệu nhưng lược bỏ đi **<** và **/>**

Thuộc tính xuất dữ liệu trong blogspot

Thuộc tính xuất dữ liệu ở đây có cú pháp là **expr** và được dùng trong thẻ **b:eval**, các thuộc tính của các thẻ **html**, và một số khác. Tuy nhiên, trong tài liệu này sẽ chỉ gói gọn trong hai trường hợp đó là thẻ **b:eval** và thẻ **html**.

Với thẻ **b:eval** các ông hãy đọc phần dưới. Còn với thẻ **html** cú pháp sử dụng như sau:

```
<thẻ_html expr:thuộc-tính='dữ_liệu_cần_xuất'></thẻ_html>
```

Trong đó

- **thẻ_html** là các thẻ chuẩn của html5 như **a, meta, div, span, ...**
- **thuộc_tính** là các thuộc tính mà thẻ_html có thể sở hữu. Ví dụ thẻ **a** thì có thuộc tính là **href**

- `dữ_liệu_cần_xuất` có thể là dữ liệu lấy từ các cấu trúc xuất dữ liệu từ các thẻ dữ liệu đã nêu ở phần trước. Có thể kết hợp cùng các toán tử sẽ trình bày ở phần sau hoặc không, hoặc dữ liệu tương tự

Ví dụ:

```
<a expr:href='data:view.url'>Liên kết đến trang này</a>
```

Thì khi truy cập, ví dụ trang chủ, kết quả sẽ là:

```
<a href='https://homepage.com'>Liên kết đến trang này</a>
```

Ví dụ 2 có sử dụng toán tử :

```
<div expr:class=' ( ( data:view.isHomepage) ? "home" : "not-home" ) '/>
```

Như vậy khi truy cập vào trang chủ, thẻ div trên sẽ có class là home, ngược lại là not-home

```
<div class='home' /> hoặc <div class='not-home' />
```

Thẻ b:eval trong blogspot

Thẻ b:eval được xem như là một thẻ dùng để xuất dữ liệu từ thẻ dữ liệu hoặc đơn giản là để xuất dữ liệu.

Thẻ này xuất hiện để giải quyết một số bất cập của thẻ dữ liệu. Đó là như các ông đã biết thì thẻ xuất dữ liệu có dạng như này `<data:name1.name2/>` . Lấy ví dụ các ông muốn xuất dữ liệu của bài viết. Như vậy các ông sẽ dùng thẻ dữ liệu sau trong tiện ích Blog:

```
<data:tên_biến.body/>
```

Tuy nhiên, nếu các ông muốn sử dụng toán tử **"snippet"** (được trình bày ở mục **"toán tử với chuỗi"** ở phần sau) để xuất một đoạn ngắn của thẻ dữ liệu phía trên. Thì cấu trúc sẽ là như này:

```
data:tên_biến.body snippet {length:150,links:false}
```

Mặc dù cấu trúc là đúng. Tuy nhiên nếu đặt cấu trúc này vào thẻ xuất dữ liệu:

```
<data:tên_biến.body snippet {length:150,links:false}/>
```

Đó sẽ là một lỗi. Vì vậy **b:eval** được ra đời để giải quyết vấn đề này.

Cấu trúc thẻ **b:eval**

```
<b:eval expr='{cấu trúc dữ liệu xuất}'/>
```

Trong đó {cấu trúc dữ liệu xuất} là cấu trúc gọi dữ liệu kết hợp với các toán tử hoặc không.

Ví dụ sử dụng

Xuất ra dữ liệu là một đoạn cắt ngắn của dữ liệu lấy từ cấu trúc xuất dữ liệu **data:tên_biến.body**

```
<b:eval expr='data:tên_biến.body snippet {length:150,links:false}'/>
```

Xuất ra tiêu đề trang cùng chuỗi **"Hello Việt Nam"**

```
<b:eval expr=' data:view.title + "Hello Việt Nam" '/>
```

Thẻ **b:tag** trong blogspot

Cấu trúc thẻ **b:tag** được trình bày như sau:

```
<b:tag cond='điều kiện' name='tên thẻ'></b:tag>
```

Thẻ này có chức năng tạo một thẻ html có tên là **tên thẻ** nếu **điều kiện** thỏa mãn. Ví dụ

Cú pháp sau sẽ tạo một thẻ span bao quanh chuỗi **"Hello Việt Nam"** nếu trang hiện tại là trang chủ.

```
<b:tag cond='data:view.isHomepage' name='span'>Hello Việt nam</b:tag>
```

Khi trang hiện tại là trang chủ mã được trả về cho người xem sẽ là:

```
<span>Hello Việt Nam</span>
```

Ngược lại:

```
Hello Việt Nam
```

Thẻ **b:attr** trong blogspot

Cú pháp thẻ `b:attr`

```
<b:attr cond='điều kiện' name='tên thuộc tính' value='giá trị thuộc tính'/>
```

Thẻ này có nhiệm vụ thêm một thuộc tính có tên là `tên thuộc tính` mang `giá trị thuộc tính` vào lớp cha của nó nếu điều kiện là đúng. Ví dụ với cú pháp sau:

```
<div>
  <b:attr cond='data:view.isHomepage' name='id' value='home'/>
  <b:attr cond='data:view.isHomepage' name='style' value='color:red'/>
</div>
```

Thì khi truy cập vào trang chủ, thẻ `div` sẽ mang một thuộc tính `id` có giá trị là `home`, và thẻ `div` này còn mang thêm một style inline là `color:red` ngược lại thì không. (`<div id='home' style='color:red'>`)

Thẻ `b:class` trong blogspot

Cú pháp:

```
<b:class cond='điều kiện' name='tên class'/>
```

Thẻ này có chức năng thêm một class cho lớp cha của nó nếu điều kiện là đúng. Cách thêm của thẻ này tương tự như thẻ `b:attr`

Thẻ `b:comment` trong blogspot

Cú pháp:

```
<b:comment render='TRUE|FALSE'> Ghi chú ở đây </b:comment>
```

Thẻ này có chức năng tạo một thẻ comment html khi thuộc tính `render` là `true` (mặc định là `true`).

Ví dụ: `<b:comment> Ghi chú ở đây </b:comment>`

Sẽ được chuyển thành `<!-- Ghi chú ở đây -->` khi truy cập. Nếu `render` là `false` thì sẽ không.

Thẻ b:template-script trong blogspot

Thẻ này có nhiệm vụ tạo một liên kết js đến blog các ông khi truy cập.

Cú pháp:

```
<b:template-script name='file js' />
```

Ví dụ :

```
<b:template-script name='https://example.com/main.js' />
```

Sẽ được chuyển thành:

```
<script type="text/javascript" src="https://example.com/main.js" />
```

Thẻ b:with trong blogspot

Thẻ này là một thẻ chứa dữ liệu trung gian. Và có thể lồng vào nhau

Cú pháp:

```
<b:with value='dữ liệu' var='biến đại diện'>
</b:with>
```

- **dữ liệu** phải là một dữ liệu cụ thể, có thể là một cấu trúc gọi dữ liệu (đúng hơn là một biểu thức blogspot) hoặc kết hợp với toán tử,...
- Và **biến đại diện** sẽ là biến chứa dữ liệu đó
- Gọi dữ liệu bằng cú pháp `data:biến đại diện`

Ví dụ cách sử dụng:

```
4
5 <b:with value=' data:view.isHomepage ? "Bạn đang ở" : "Buồn thật, bạn ở trang" ' var='page'>
6   <b:with value=' data:view.isHomepage ? "trang chủ" : "không phải là trang chủ" ' var='cond'>
7     <b:eval expr='data:page + data:cond' />
8   </b:with>
9 </b:with>
```

Như vậy khi truy cập vào trang chủ. Các ông sẽ nhận được thông báo là: **Bạn đang ở trang chủ**

Thẻ b:message và b:param

Là thẻ tạo một text thông báo mặc định theo ngôn ngữ được cài đặt trong blog của ông. Và thẻ **b:param** phải lót bên trong thẻ **b:message**

Cấu trúc:

```
<b:message name='tên thông báo'>
    <b:param value='giá trị đi kèm' />
</b:message>
```

Trong đó, **tên thông báo** chỉ được là những tên sau:

- data:messages.authorSaid
- data:messages.authorSaidWithLink
- data:messages.byAuthor
- data:messages.byAuthorLink
- data:messages.numberOfComments
- data:messages.postedByAuthor
- data:messages.postedByAuthorLink
- data:messages.poweredByBloggerLink
- data:messages.templatelImagesBy
- data:messages.templatelImagesByLink

Với mỗi giá trị trên, tùy vào ngôn ngữ blog của các ông mà nó sẽ hiển thị. Ví dụ **data:messages.postedByAuthor** với blog có ngôn ngữ tiếng Việt thì dữ liệu trả về đó là “**Đăng bởi**”. Còn tùy thuộc vào từng tiện ích widget mà giá trị trên có thể gọi được hay không. Ví dụ các ông không thể gọi giá trị **data:messages.numberOfComments** ở tiện ích Header được.

Và **giá trị đi kèm** có thể là một cấu trúc gọi dữ liệu hoặc là tương tự.

Cú pháp sau sẽ xuất ra một dòng text là “**5 bình luận các ông ơi**” (với giả sử bài viết đang xem có 5 bình luận và ngôn ngữ blog các ông là tiếng Việt)

```
<b:message name='data:messages.numberOfComments'>
    <b:param value=' bình luận các ông ơi' />
```

```
</b:message>
```

Thẻ b:switch, b:case và b:default

Chức năng thì tương tự thẻ **b:if b:else** và **b:elseif**, tuy nhiên ngắn gọn và dễ dùng hơn.

```
<b:switch var='biến'>
  <b:case value='đối tượng' />
  <b:default />
</b:switch>
```

Có thể có nhiều hơn một thẻ **b:case** tuy nhiên chỉ được phép có một thẻ **b:default** trong thẻ **b:switch**. Trong đó **biến** và **đối tượng** bắt buộc phải có.

Các ông có thể hiểu đơn giản, **đối tượng** sẽ là phần tử được so sánh với **biến**. Và **biến** có thể là một dữ liệu từ cấu trúc gọi dữ liệu hoặc tương tự, kết quả sẽ là **true** nếu **đối tượng** tương đương với **biến**. Và nếu không có bất kì **đối tượng** nào tương đương với **biến**, thì **b:default** sẽ là **true** và được thực hiện.

```
<b:switch var='data:posts.length'>
  <b:case value='0' />
    Đoạn code này sẽ được thực hiện nếu số
    lượng bài viết trả về bằng 0
  <b:case value='1' />
    Đoạn code này sẽ được thực hiện nếu số
    lượng bài viết trả về bằng 1
  <b:default />
    Đoạn code này sẽ được thực hiện nếu số
    lượng bài viết trả về không bằng 0
    và cũng chả bằng 1
</b:switch>
```

Thẻ b:loop trong blogspot

Như đã định nghĩa mảng ở phần trước, vậy thì thẻ **b:loop** này sẽ cao nhiệm vụ lặp qua các đối tượng trong mảng để lấy dữ liệu. Thẻ b:loop sẽ lặp quan toàn bộ phần tử trong mảng được truyền vào. Không có cách nào dừng nó lại cho đến khi nó lặp đến phần tử cuối cùng.

Cấu trúc cú pháp đầy đủ:

```
<b:loop index='kí_tự' reverse='true|false' values='một_mảng_dữ_liệu' var='tên_biến'>
</b:loop>
```

Trong đó những thuộc tính được tô tô nền trắng là không cần thiết còn lại thì bắt buộc phải có.

Giá trị	Mô tả
kí_tự	Là một chuỗi kí tự hoặc kí tự không được bắt đầu bằng một số. Có nhiệm vụ lưu vị trí phần tử hiện đang được vòng lặp chỉ tới. Ví dụ một mảng có 10 phần tử. Thì kí_tự của index sẽ chỉ hiện tại các ông đang ở phần tử thứ 1 hay 5 hay 10.
true false	Mặc định reverse có giá trị là false tuy nhiên nếu true được thiết lập thì vòng lặp sẽ đảo ngược. Ví dụ thay vì lặp từ 1 đến 10 nó sẽ lặp từ 10 đến 1.
một_mảng_dữ_liệu	Là một mảng dữ liệu có thể là được lấy từ thẻ dữ liệu có thể kết hợp với các toán tử mảng hoặc không, hoặc là một mảng do các ông tự tạo.
tên_biến	Nó làm đại diện cho phần tử đang được lặp tới. Và dữ liệu sẽ được gọi ra từ nó. Ví dụ nếu hiện tại vòng lặp đang chỉ tới học sinh thứ 5 thì tên_biến sẽ là đại diện cho học sinh thứ 5. Và thông qua tên_biến các ông có thể gọi các thông tin liên quan đến học sinh thứ 5 đó ra ngoài thông qua thẻ gọi dữ liệu. Ví dụ <data.tên_biến.dữ_liệu/>

Thẻ <![CDATA[]]> trong blogspot

Blogspot được thiết kế dựa trên ngôn ngữ xml. Có quy định và yêu cầu rất chặt chẽ về cấu trúc của từng cú pháp và cách các kí tự đặc biệt xuất hiện trong mã.

Hầu như mọi thẻ, kể cả thẻ html bình thường hay là thẻ gọi dữ liệu và thậm chí là dữ liệu được chứa đều sẽ được chuyển về dạng mới. Ngoại trừ nó là kí tự cấu trúc đóng mở của thẻ html. Thì mọi kí tự đặc biệt khác đều bị chuyển về kiểu mới.

Thế nào là một kí tự cấu trúc đóng mở thẻ?

` Ví dụ `

``

Nếu là các kí tự `<`, `>`, `/` đứng trước và sau tên của một thẻ html chuẩn (tô màu hồng). Thì các ông cứ tạm hiểu nó là một kí tự cấu trúc đóng mở thẻ. (Thuật ngữ do tớ tự nghĩ đấy 🤪) thì nó sẽ không bị chuyển sang dạng mới. Còn lại chúng đều bị chuyển. Và sau đây là bảng những kí tự đặc biệt sẽ bị chuyển thành dạng mới, còn lại thì không.

**Gọi là dạng mới nhưng thực chất chúng giống nhau chỉ khác nhau về cách viết.
Ví dụ trong tiếng Việt gọi là quả trứng nhưng tiếng anh là egg**

Kí tự	Dạng bị chuyển
>	>
<	<
&	&
“ hoặc ”	"

Ngoài ra trong một số trường hợp. Kí tự “ sẽ bị chuyển thành ‘ trong trường hợp nó là ký tự đứng đầu của giá trị thuộc tính của thẻ.

Ví dụ:

```
<a href="href"/>
```

Sẽ được tự động chuyển thành:

```
<a href='href'/>
```

Như vậy, nó đúng trong mọi trường hợp. Giả sử các ông có một cấu trúc JS, một cấu trúc HTML và một cấu trúc CSS như sau:

```

5
6 ▼ <script type="text/javascript">
7     if ( 8 > 9 ) {
8         alert ( " True " )
9     }
10 </script>
11 ▼ <style type="text/css">
12     a:before {
13         content: " 😊 ";
14     }
15 </style>

```

Khi cập nhật lên Blog nó sẽ bị chuyển thành:

```

18
19 <script type='text/javascript'>
20     if ( 8 &gt; 9 ) {
21         alert ( &quot; True &quot; ; )
22     }
23 </script>
24 <style type='text/css'>
25     a:before {
26         content: &quot; 😊 &quot;;
27     }
28 </style>

```

Rất khó khăn trong việc đọc hiểu đúng không/ Vì vậy thẻ `<![CDATA[]]>` được ra đời để giải quyết vấn đề đó.

Cấu trúc sử dụng :

<cú pháp comment của mã> <![CDATA[

Mã được viết tại đây

<cú pháp comment của mã>]]>

<cú pháp comment của mã> ở đây được hiểu là cấu trúc được dùng để viết comment trong mã đó. Ví dụ CSS là `/* */` và JS là `//`

Ví dụ:

```
<script type="text/javascript">
  // <![CDATA[
  if ( 8 > 9 ) {
    alert ( " True " )
  }
  // ]]>
</script>
<style type="text/css">
  /* <![CDATA[ */
  a:before {
    content: "😁";
  }
  /* // ]]> */
</style>
```

Như vậy mã sẽ không bị chuyển và sẽ giữ nguyên như vậy.

Chú ý: Trong mã js tránh trường hợp viết như này. `// <![CDATA[if (8 > 9) {`
 ... Vì lúc này sẽ bị hiểu toàn bộ đó là một bình luận mã JS chứ không phải là một mã JS

```
<script type='text/javascript'>
    if ( 8 &gt; 9 ) {
        alert ( &quot; True &quot; ; )
    }
</script>
<script type='text/javascript'>
    // <![CDATA[
    if ( 8 > 9 ) {
        alert ( " True " )
    }
    // ]]>
</script>
```

Thẻ `b:widget-settings` và `b:widget-setting` trong blogspot

Đây là thẻ mặc định được tự động thêm mỗi khi các ông nhấn nút cập nhật giao diện. Chức năng của thẻ này là xác định những thông số mà các ông đã cài đặt trong phần **layout**.

Trong đó thẻ **b:widget-settings** sẽ chứa các thẻ **b:widget-setting** và các thẻ `b:widget-setting` sẽ chứa các cài đặt

Ví dụ ở tiện ích PopularPosts thì các cài đặt trong mục layout sẽ là:

Configure Popular Posts

Title	<input type="text" value="Blog"/>
Most viewed	<input type="radio"/> All time <input checked="" type="radio"/> Last year <input type="radio"/> Last 30 days <input type="radio"/> Last 7 days
Show	Post title and <input checked="" type="checkbox"/> image thumbnail <input checked="" type="checkbox"/> snippet Display up to <input type="text" value="10"/> post(s)
<input type="button" value="Save"/> <input type="button" value="Cancel"/> <input type="button" value="Remove"/>	

Vậy, tương ứng với các mục cài đặt trên sẽ là:

```
<b:widget-settings>
  <b:widget-setting name='numItemsToShow'>10</b:widget-setting>
  <b:widget-setting name='showThumbnails'>>true</b:widget-setting>
  <b:widget-setting name='showSnippets'>>true</b:widget-setting>
  <b:widget-setting name='timeRange'>LAST_YEAR</b:widget-setting>
</b:widget-settings>
```

Các ông không cần thiết phải quan tâm đến những thẻ này, vì nó có thể dễ dàng chỉnh sửa trong mục **layout** và được thêm một cách tự động dựa trên các cài đặt mặc định.

Và **b:settings** phải được đặt bên trong thẻ **b:widget** tương ứng.

Cấu trúc chung của những thẻ này là:

```
26
27 <b:widget-settings>
28   <b:widget-setting name='<tên-cài-đặt'>'{giá-trị}'</b:widget-setting>
29 </b:widget-settings>
```

Và các thẻ cài đặt này chứa các dữ liệu cài đặt sẽ được trả về khi gọi bằng cú pháp :

data:<tên cài đặt>

Thẻ cài đặt trong tiện ích PopularPosts

<tên cài đặt>	{giá trị}
numItemsToShow	Là một số nguyên dương thể hiện số lượng bài viết phổ biến sẽ được hiển thị
showThumbnails	Có giá trị là true hoặc false dùng cho phép có hiển thị hình ảnh của bài viết đó hay không.
showSnippets	Giá trị true hoặc false dùng xác định có hiển thị đoạn cắt ngắn nội dung bài viết hay không
timeRange	Xác định chọn những bài viết phổ biến trong thời gian nào <ul style="list-style-type: none"> - LAST_YEAR : năm trước - ALL_TIME : toàn bộ thời gian - LAST_MONTH : tháng trước - LAST_WEEK : tuần trước

Ví dụ **data:showThumbnails** sẽ trả về **true** nếu thẻ **b:setting** là **true**. Và các ông có thể dùng dữ liệu được trả về đó để dùng cho thẻ **b:if** hoặc tương tự. Ví dụ

<b:if cond='data:showSnippets'>

Đoạn mã này sẽ chạy nếu như **showSnippets** có giá trị là **true**

</b:if>

Thẻ cài đặt trong tiện ích Blog

<tên cài đặt>	{giá trị}
showReactions	True hoặc false dùng để cho biết có hiển thị nút cảm xúc hay không
showInlineAds	True hoặc false cho biết có hiển thị quảng

	cáo hay không
showBacklinks	True hoặc false cho biết có hiển thị backlinks hay không
showTimestamp	True hoặc false cho biết có hiển thị thời gian đăng bài hay không
showLocation	True hoặc false cho biết có hiển thị vị trí nếu nó được thiết lập trong bài viết hay không
showLabels	True hoặc false cho biết có hiển thị nhãn của bài viết hay không
showAuthorProfile	True hoặc false cho biết có hiển thị hồ sơ tác giả hay không
disableGooglePlusShare	True hoặc false cho biết có hiển thị nút chia sẻ lên Google+ hay không
showAuthor	True hoặc false cho biết có hiển thị tác giả hay không
showCommentLink	True hoặc false cho biết có hiển thị liên kết đến bình luận hay không
showShareButtons	True hoặc false cho biết có hiển thị nút chia sẻ hay không
showDateHeader	True hoặc false cho biết có hiển thị ngày đăng bài hay không
postsPerAd	Là một số nguyên cho biết số lượng quảng cáo sẽ hiển thị trong mỗi bài viết
style.textcolor	Giá trị là một mã màu html cho biết màu của văn bản
style.linkcolor và style.urlcolor	Giá trị là một mã màu html cho biết màu của link / url
style.bgcolor	Giá trị là một mã màu html cho biết màu của nền
style.bordercolor	Giá trị là một mã màu html cho biết màu của viền

Thẻ cài đặt trong tiện ích FeaturedPost

Cài đặt	Giá trị
showSnippet	Có hiển thị đoạn cắt ngắn nội dung hay không (true/false)
showPostTitle	Có hiển thị tiêu đề bài viết hay không(true/false)
postId	Id bài viết sẽ được chọn làm bài viết nổi bật
showFirstImage	Có hiển thị hình ảnh cho bài viết nổi bật hay không(true/false)
useMostRecentPost	Có sử dụng bài viết gần đây nhất để làm bài viết nổi bật hay không (phải là false nếu postId được thiết lập)(true/false)

Thẻ cài đặt trong tiện ích Labels

Cài đặt	Giá trị
sorting	Kiểu sắp xếp nhãn: <ul style="list-style-type: none"> - ALPHA : theo kiểu bảng chữ cái - FREQUENCY : nhãn nào nhiều bài nhất xếp trước
display	Kiểu hiển thị của nhãn: <ul style="list-style-type: none"> - CLOUD : kiểu cloud - LIST : kiểu danh sách
selectedLabelsList	Danh sách nhãn sẽ hiển thị, các nhãn cách nhau bằng dấu phẩy. Thẻ cài đặt này chỉ dùng nếu như showType là USER_SELECTED . Ngược lại, chỉ cần như sau: <b:widget-setting name='selectedLabelsList'/>
showType	<ul style="list-style-type: none"> - ALL : hiển thị tất cả các nhãn hiện có

	- USER_SELECTED : hiển thị các nhãn do người dùng chọn
showFreqNumbers	Có hiển thị số lượng bài viết thuộc nhãn này hay không (true/false)

Những tiện ích còn lại hoặc cần hoặc không cần hoặc cần nhưng các ông cũng không cần quan tâm trừ khi các ông muốn viết giao diện blog theo cách chuyên nghiệp.

Thẻ b:skin trong blogspot

Đây là loại thẻ hiếm khi được tác động tới nhưng bắt buộc phải có khi thiết kế giao diện cho blogspot. Nhất là nếu - hiện tại - các ông muốn sử dụng comment nhiều bậc (comment mà người khác có thể trả lời lại - reply). Chức năng chính của chúng đều là chứa CSS và chứa một số cài đặt liên quan đến CSS (các cài đặt liên quan này tương tự như **:root** trong CSS và các ông có thể gọi cài đặt này bằng thẻ gọi dữ liệu **data:skin**).

Tớ không tìm hiểu và không hay sử dụng thẻ này cộng thêm việc các chức năng này chỉ hữu ích khi các ông **"code đúng chuẩn google"**. Còn nếu không có thể bỏ qua.

Thẻ này sẽ được chuyển thành dạng một thẻ **style** khi hiển thị.

```
<style id='page-skin-1' type='text/css'><!--
--></style>
```

Thẻ b:defaultmarkups và b:defaultmarkup

Đầu tiên, tớ sẽ giới thiệu cấu trúc thẻ trước:

```

25
26 <b:defaultmarkups>
27     <b:defaultmarkup type='tiện_ích_widget'>
28
29     </b:defaultmarkup>
30 </b:defaultmarkups>

```

Trong đó **tiện_ích_widget** là tên gọi xác định các tiện ích đã được trình bày. Cách nhau bằng dấu phẩy (,) nếu có hơn một tiện ích được sử dụng.

- **b:defaultmarkups** chỉ có thể chứa các thẻ **b:defaultmarkup**
- Và thẻ **b:defaultmarkup** chỉ có thể chứa các thẻ **b:includable**

Như các ông đã biết, thẻ **b:includable** được đặt ở tiện ích nào, thì chỉ có tiện ích đó được phép gọi và sử dụng mã bên trong thẻ **b:includable** đó.

Tuy nhiên, lấy trường hợp các ông thẻ **b:includable** chứa mã có chức năng **a** được đặt trong tiện ích **Blog**, và thẻ **b:includable** chứa mã có chức năng **a** nhưng lần này lại ở trong tiện ích **PopularPosts**. Các ông thấy gì chưa nào? Cùng một chức năng nhưng phải viết lại hai lần ở cả hai tiện ích. Quá tốn kém đúng không.

Lúc này **b:defaultmarkups b:defaultmarkup** ra đời để giúp các ông giải quyết vấn đề này.

Giả dụ thẻ **b:includable** thực hiện chức năng **a** có id là **codemai**.

```

<b:defaultmarkups>
  <b:defaultmarkup type='Blog,PopularPost'>
    <b:includable id='codemai'>
      Tôi thực hiện chức năng a
    </b:includable>
  </b:defaultmarkup>
</b:defaultmarkups>

```

Như vậy, lúc này tiện ích **Blog** và tiện ích **PopularPosts** đều có thể gọi và sử dụng thẻ **b:includable** có chức năng **a**.

- Nếu id của thẻ **b:includable** chứa trong thẻ **b:defaultmarkup** trùng với id của thẻ **b:includable** (có thể là thẻ **b:includable** mặc định được tự động thêm vào của Blogger hoặc là do các ông viết) có trong các tiện ích được liệt kê trong **type** (trên ảnh là Blog và PopularPosts) thì nó sẽ ghi đè cái cũ ở các tiện ích được liệt kê đó.
- Trong trường hợp các ông muốn gọi được thẻ **b:includable** ở bất kì tiện ích và vị trí nào thì sử dụng **type='Common'**

```

25
26 <b:defaultmarkups>
27   <b:defaultmarkup type='Blog,PopularPost'>
28     <b:includable id='codemai'>
29       Chỉ có Blog,PopularPost mới gọi được
30       codemai là tôi
31     </b:includable>
32   </b:defaultmarkup>
33   <b:defaultmarkup type='Common'>
34     <b:includable id='common'>
35       Tôi có thể được gọi ở bất kì vị trí nào
36     </b:includable>
37   </b:defaultmarkup>
38 </b:defaultmarkups>

```

Một số loại toán tử trong blogspot

Những loại toán tử thường gặp và hữu ích khi thiết kế blogspot

Toán tử logic và toán tử thường trong blogspot

Trong blogspot, các toán tử logic sau được hỗ trợ và các ông có thể kết hợp nhiều toán tử lại với nhau để tạo nên một điều kiện phù hợp với mục đích của mình. Ở bảng sau đây, giá trị sẽ trả về **true** nếu thỏa mãn toàn bộ điều kiện có trong ví dụ:

Tên / Kí hiệu	Ví dụ
and (&&)	điều_kiện and điều_kiện - true nếu các điều kiện cùng đúng Ở các ví dụ sau: những kí hiệu bên trong (và) có thể dùng để thay thế cho tên gọi của chúng. Ví dụ điều_kiện && điều_kiện tương đương với điều_kiện and điều_kiện
or ()	điều_kiện or điều_kiện - true nếu một trong các điều kiện là đúng
not (!)	not điều_kiện - true nếu điều_kiện sai
+	"string" + "string" hoặc "string" + dữ_liệu từ các thẻ gọi dữ liệu hoặc các loại dữ liệu khác string và dữ_liệu . Ví dụ "code" + "mã" trả về chuỗi "codemai"
+, -, *, /	Đơn giản nó là các phép cộng (+), trừ (-), nhân (*) và chia (/) các số
?:	điều_kiện ?: thực_thi , thực_thi sẽ được gọi nếu như điều kiện là đúng. Còn một cách khác đó là: điều_kiện ? thực_thi_1 : thực_thi_2 Nếu điều kiện là đúng thì thực_thi_1 được thực hiện và ngược lại là thực_thi_2
gt (>)	a gt b - So sánh xem a có lớn hơn b hay không
lt (<)	a lt b - so sánh xem a nhỏ hơn b không
eq (==)	a eq b so sánh a có bằng b hay không
gte (>=)	a gte b xem xem a có lớn hơn hoặc bằng b hay không
lte (<=)	a lte b xem xem a có bé hơn hoặc bằng b hay không
neq (!=)	a neq b xem xem có phải là a khác b hay không

Toán tử mảng trong blogspot

Toán tử mảng ở đây là các toán tử làm việc với các dữ liệu liên quan đến mảng. Đây là một loại toán tử khác khó hiểu. Các ông muốn học nâng cao thì hãy tìm hiểu, nếu không có thể bỏ qua.

Toán tử	Ví dụ mô tả
---------	-------------

contains (đối với mảng chứa các phần tử riêng)	<p>a contains b a và b phải cùng loại với nhau. Ví dụ a là mảng có kiểu dữ liệu chuỗi thì b phải là chuỗi. Điều này cũng đúng với các trường hợp tiếp theo. Kiểm tra xem mảng a có chứa phần tử b hay không. Ví dụ sau sẽ trả về false <code>["crush","lover"] contains "learn"</code></p>
contains (đối với chuỗi)	<p>a contains b Kiểm tra xem chuỗi a có chứa chuỗi b hay không. Ví dụ sau sẽ là false <code>"Codemai" contains "blog"</code></p>
in	<p>b in a Kiểm tra xem b có phải thuộc a hay không. Ví dụ sau sẽ trả về false: <code>"code" in ["blog","js"]</code></p>
to	<p>n1 to n2 Tạo một mảng chứa các số từ n1 đến n2</p>
take (limit)	<p>mảng take number Chỉ lấy và đọc number phần tử đầu của mảng <code>["1","2","3"] take 2</code> thì dữ liệu trả về sẽ là một mảng chỉ gồm <code>["1","2"]</code></p>
skip (offset)	<p>mảng skip number Bỏ qua number phần tử đầu của mảng và chỉ đọc những phần tử sau phần tử thứ number</p>

Toán tử mảng nâng cao

Đây là những toán tử mảng thường được dùng để tạo một mảng mới từ mảng cũ khi điều kiện được thỏa mãn.

Cấu trúc chung của toán tử nâng cao dạng này đó là:

`<mảng> <toán tử> (<biến đại diện> => <dữ liệu / dữ liệu có điều kiện cho mảng mới>)`

Trong đó :

- **<mảng>** thì tớ sẽ không giới thiệu thêm nữa. Vì những gì chứa nhiều phần tử thì nó là một mảng. Ví dụ `data:posts` trong tiện ích Blog là một mảng.
- **<toán tử>** là những toán tử tớ sẽ trình bày ở dưới
- **<biến đại diện>** là một chuỗi ký tự hoặc kí tự bất kí không được phép đứng đầu là một số. Có vai trò đại diện cho phần tử đang được toán tử mảng chỉ tới. Hiểu đơn giản hơn là toán tử mảng nâng cao này giống như một vòng lặp và **<biến_đại_diện>** sẽ chỉ đến phần tử đang được lặp tới
- Nếu điều kiện của toán tử đúng thì **<dữ liệu / dữ liệu có điều kiện cho mảng mới>** sẽ được đọc và lấy từ mảng cũ **<array>**. Và tùy theo từng toán tử mà dữ liệu hoặc dữ liệu có điều kiện sẽ được sử dụng và cũng tùy vào mục đích các ông và chúng thường là các thẻ gọi dữ liệu đã được trình bày ở phần trước

Toán tử	Ví dụ mô tả
map (select)	tạo và trả về kết quả một mảng mới chứa tất cả dữ liệu cho mảng mới. Ví dụ <code>data:posts map (p => p.title)</code> Kết quả trả về là một mảng mới chứa tất cả tiêu đề của tiêu đề các phần tử có trong mảng data:posts
where (filter)	Tạo và trả về một mảng mới chứa tất cả dữ liệu cho mảng mới nếu dữ liệu có điều kiện được thỏa mãn. Ví dụ <code>data:posts where (p => p.numComments gt 5)</code> Thì mảng trả về sẽ chỉ chứa các bài viết thuộc data:posts với điều kiện bài viết đó có số lượng bình luận lớn hơn 5 (<code>p.numComments gt 5</code>)
first	Chọn và trả về phần tử thoả mã điều kiện dữ

	liệu đầu tiên của mảng . Ví dụ: <code>data:posts first (p => p.title contains "hack")</code> Thì mảng trả về sẽ là phần tử đầu tiên trong mảng thỏa mãn rằng tiêu đề của nó chứa chuỗi "hack"
any	Trả về true nếu ít nhất một phần tử trong mảng thỏa mãn điều kiện. Ví dụ: <code>data:posts any (p => p.title contains "hack")</code> Sẽ trả về true nếu có ít nhất một phần tử trong mảng data:posts có tiêu đề chứa cụm từ "hack"
all	Tương tự như any nhưng điều kiện trả về true là toàn bộ phần tử thỏa mãn điều kiện
none	Tương tự như all nhưng trả về true nếu toàn bộ phần tử đều không thỏa mãn điều kiện
count	Dữ liệu trả về là số các phần tử thỏa mãn điều kiện. <code>data:posts count (p => p.title contains "hack")</code> Sẽ trả về số lượng phần tử thuộc mảng data:posts mà tiêu đề của nó chứa chuỗi "hack"

Toán tử làm việc với đường dẫn

Đây là những toán tử có cũng được không có cũng không sao, vì ít ai làm việc với các toán tử này, tuy nhiên các ông có thể tham khảo qua.

Những toán tử sau đây được sử dụng khi các ông muốn thêm, bớt các đối số của đường dẫn (Tất nhiên đường dẫn thì chủ yếu và thẻ **a**). Và các toán tử này được xuất ra thông qua thuộc tính **expr** , thuộc tính này được trình bày ở mục **"Một số thuộc tính căn bản của thẻ trong blogspot"**.

Cấu trúc chung của các toán tử này có thể được viết dưới hai dạng:

`<phần tử 1> <toán tử> <phần tử hai>`

Hoặc:

<toán tử> (<phần tử 1>, <phần tử hai>)

Trong đó <phần tử 1> thông thường là các thẻ dữ liệu mà kết quả trả về là một liên kết.

Toán tử	Ví dụ mô tả
path	Thêm vào phía sau <phần tử 1> <phần tử 2> Ví dụ phần tử một là một thẻ gọi dữ liệu có dữ liệu trả về là https://google.com và phần tử hai là /codemai thì kết quả trả về sẽ là https://google.com/codemai
params (appendParams)	Thêm vào phía sau <phần tử 1> một tham số là <phần tử 2>. Trong đó cấu trúc <phần tử 2> sẽ là {a : "b"}. Ví dụ : <code>data:view.url params { id: "codemai" }</code> Thì dữ liệu trả về sẽ là <data:view.url>?id=codemai. Và tùy thuộc vào liên kết hiện tại mà ? sẽ được dùng hoặc là & sẽ được dùng.
fragment	<code>data:view.url params comments</code> Sẽ trả về <data:view.url>#comments

Ví dụ sử dụng toán tử liên kết

```
<a expr:href='data:view.url params { id: "codemai" }'/>
```

Kết quả trả về khi truy cập vào trang chủ:

```
https://example.com/#codemai
```

Toán tử làm việc với chuỗi

Dùng toán tử này trong trường hợp các ông muốn cắt ngắn nội dung của một chuỗi. Ví dụ các ông muốn cắt ngắn `data:tên_biến.body` (xem lại thẻ dữ liệu để biết chức năng của thẻ này) để hiển thị như là một đoạn mô tả ngắn khi hiển thị bài viết ở trang chủ.

Cấu trúc của toán tử này là:

```
<dữ liệu chuỗi> snippet { <các thuộc tính cài đặt> }
```

Trong đó các thuộc tính cài đặt cách nhau bởi dấu phẩy (,) và chúng có thể có hoặc không. Với những thuộc tính có chứa **true false** thì mặc định đều **true**.



Thuộc tính cài đặt	Mô tả
length	Cắt một chuỗi gồm length kí tự (lớn nhất là 1000 và nhỏ nhất là 50)
ellipsis	Có hiển thị dấu ... ở cuối mỗi đoạn cắt khi trả về hay không (giá trị phải là true hoặc false)
linebreaks	Có bao gồm cả dấu ngắt dòng trong đoạn trích hay không
links	Có bao gồm liên kết nếu có trong đoạn cắt hay không. Ví dụ ông muốn cắt 150 kí tự từ bài viết của mình nhưng rùi thay trong 150 kí tự đầu có chứa một liên kết thì các ông có muốn bỏ qua liên kết đó để lấy text khác hay là tính luôn đoạn link đó vào trong 150 kí tự sẽ được cắt

Ví dụ sử dụng

```
<b:eval expr='data:tên_biến.body snippet { length:20,links:false}'/>
```

Sẽ trả về chuỗi gồm 20 kí tự (**length:20**), không chứa liên kết (**links:false**).

Nếu **data:tên_biến.body** có giá trị là 'Anh ơi sao <https://codemai.blogspot.com> hôm nay em đói thế, hay mình đi ăn gì đi' kết quả trả về sẽ là 'Anh ơi sao hôm nay em '.

Toán tử làm việc với hình ảnh.

Nếu dữ liệu trả về từ thẻ gọi dữ liệu là một liên kết đến một hình ảnh. Và hình ảnh này là một hình ảnh có thể resize (xác định thông qua isResizable đã nêu ở phần trước) hoặc hình ảnh là một hình ảnh được lưu trên các host của google. - thường có tên miền là x.bp.blogspot.com thì các toán tử sau đây có thể được sử dụng.

Cấu trúc chung của toán tử này được thể hiện qua hai cách:

<dữ liệu/liên kết đến hình ảnh> <toán tử> <số / mảng số> <toán tử> <tỉ lệ>

Hoặc:

<toán tử>(<dữ liệu/liên kết đến hình ảnh>,<số / mảng số>,<tỉ lệ>

Trong đó có thể là một thẻ dữ liệu mà kết quả trả về là một đường dẫn đến hình ảnh (ví dụ: `data:view.featuredImage`) hoặc một đường dẫn cụ thể (ví dụ: <https://1.bp.blogspot.com/abc/image.jpg>)

Toán tử	Cấu trúc ví dụ mô tả
resizeImage	<p>Thay đổi kích thước ảnh được truyền tới thông qua <dữ liệu/liên kết đến hình ảnh> thành một hoặc một mảng nhiều hình ảnh mới có chiều rộng là <số / mảng số> với tỉ lệ hình ảnh là <tỉ lệ></p> <p>Nếu <mảng số> được thiết lập thì dữ liệu trả về sẽ là một mảng gồm nhiều hình ảnh theo chiều rộng lần lượt là các giá trị trong mảng và tỉ lệ như đã thiết lập tại <tỉ lệ>. Vì vậy một vòng lặp ở đây là cần thiết</p>
sourceSet	<p>Đọc qua bài: https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Responsive_images</p> <p>Thì các ông sẽ hiểu sourceSet sẽ trả về một</p>

	giá trị srcSet tương ứng theo <mảng số> đã được thiết lập cho các ông. Xem ví dụ dưới
--	---

Ví dụ sử dụng toán tử hình ảnh

Hiển thị hình ảnh nổi bật của trang dưới chiều rộng là 250 và hình ảnh có tỉ lệ là 4:3

```
<img expr:src='resizeImage(data:view.featuredImage, 250, "4:3")'/>
```

Tạo và hiển thị một mảng hình ảnh gồm các hình ảnh có kích thước chiều rộng lần lượt là 250, 350, 450 với tỉ lệ 4:3

```
<b:loop values='resizeImage (data:view.featuredImage, [250,350,450] , "16:9")'
var='image'>
  <img expr:src='data:image.url'/>
</b:loop>
```

Hiển thị hình ảnh theo kiểu srcset cho màn hình có kích thước lần lượt là 1024, 720, 420 với tỉ lệ hình ảnh là 4:3:

```
<img expr:src='data:view.featuredImage'
expr:srcset='sourceSet(data:view.featuredImage, [420,720,1024], "4:3")'/>
```

Giả sử data:view.featuredImage có giá trị là <https://abc.com/image.jpg> thì kết quả trả về sẽ là:

```
<img src='https://abc.com/image.jpg' srcset="https://abc.com/image420-4-3.jpg 420w,
https://abc.com/image720-4-3.jpg 720w,https://abc.com/image1024-4-3.jpg 1024w"/>
```

Tất nhiên thực tế, liên kết trả về sẽ ở dạng khác.

Có thể kết hợp nhiều toán tử với nhau thành một toán tử theo mục đích của các ông. Những hãy chú ý đến các toán tử mảng, thông thường chúng không được kết hợp với mảng, vì đa số kết quả trả về sẽ là một lỗi

```
data:posts count (p => p.numComments gt 5)
```

Sẽ trả về kết quả là số lượng các bài viết trong mảng `data:posts` mà số lượng bình luận của bài viết đó lớn hơn 5

Và thực tế còn khá nhiều loại toán tử khác được hỗ trợ. Và để sử dụng thuần thạo các toán tử này đòi hỏi các ông phải gõ hơi bị nhiều.

Toán tử format làm việc với ngày tháng

Thông thường, với thẻ dọi dữ liệu ngày tháng sau:

```
<data:tên_biến.date/>
```

Thì dữ liệu trả về sẽ là một chuỗi ngày tháng năm theo định dạng mặc định của vùng ngôn ngữ mà các ông thiết lập trong blog. Một ví dụ của tớ sẽ là:

January 16, 2016

Tuy nhiên, các ông muốn hiển thị ngày tháng dưới dạng: **thứ - ngày - tháng - năm** thì sao?

Hoặc các ông sử dụng thẻ gọi dữ liệu đã được trình bày ở phần trước như sau:

- date: hiển thị ngày đăng bài
 - iso8601: hiển thị theo định dạng iso8601
 - year: năm bài được đăng
 - month: tháng bài được đăng
 - day: ngày bài được đăng
 - dayOfWeek: ngày thứ tuần bài được đăng
 - dayOfMonth: ngày thứ tháng bài được đăng
 - dayOfYear: ngày thứ năm được đăng

Như vậy để hiển thị theo dạng **thứ ngày tháng năm** thì các ông cần một thẻ b:eval kiểu này:

```
<b:eval expr='data:tên_biến.date.dayOfWeek data:tên_biến.date.day
data:tên_biến.date.month data:tên_biến.date.year/'>
```

Khá dài dòng đúng không nào. Vậy nên toán tử làm việc với ngày tháng đã được ra đời.

Cấu trúc toán tử format:

```
format( <thẻ xuất dữ liệu ngày tháng>, "[danh sách định dạng]" )
```

Trong đó [danh sách định dạng] bao gồm:

Nhóm	Định dạng	Mô tả ví dụ
Năm	YY	Xuất ra hai số cuối cùng của năm (2020 => 20)
	YYYY	Xuất ra năm luôn (2020 => 2020)
Tháng	M	Xuất ra tháng
	MM	Cũng là xuất ra tháng nhưng những tháng từ 1 đến 9 được thêm 0 vào đầu (1 => 01)
	MMM	Chỉ dùng nếu blog các ông dùng ngôn ngữ có hỗ trợ viết tắt tháng. Ví dụ tiếng Anh

		February => Feb
	MMMM	Tên tháng (Ví dụ tháng hai tháng ba,...) kết quả trả về tùy theo ngôn ngữ blog
	MMMMM	Chỉ dùng nếu blog các ông dùng ngôn ngữ có hỗ trợ viết tắt tháng. Ví dụ tiếng Anh : February = > F
Tuần	w	Tuần trong tháng (tuần thứ 4 => 4)
	ww	Tuần trong tháng như từ 1 - 9 có số 0 ở trước (1 => 01)
	W	Tuần trong năm (Tuần thứ 32 => 32)
Ngày	d	Ngày trong tháng (tối đa 31 ngày)
	dd	Ngày trong tháng nhưng từ 1 - 9 có thêm số 0 trước
	D	Ngày trong năm (tối đa 366 ngày)
	DD	Ngày trong năm nhưng từ 1 - 9 có 0 phía trước (1 => 01)
	DDD	Ngày trong năm nhưng từ 1 - 9 có thêm hai số 0 phía trước (1 => 001) từ 10 đến 99 có thêm một số 0 phía trước (10 => 010)

Ngày	F	Tên ngày thứ 10 trong tháng. Ví dụ ngày thứ 10 của tháng 8
-------------	----------	---

		là thứ hai vậy trả về “thứ hai”
	E	Tên ngày trong tuần (Monday => M)
	EE	Tên ngày trong tuần (Monday => Mo)
	EEE	Tên ngày trong tuần (Monday => Mon)
	EEEE	Tên ngày trong tuần (Monday => Monday)
Buổi	aaaa	Sáng thì trả về am và chiều tối thì pm
	bbbb	Trả về tên của buổi sáng, buổi trưa, buổi tối theo ngôn ngữ của Blog. Ví dụ blog tiếng Việt thì trả về “buổi sáng” nếu thời gian trong dữ liệu được trả về từ thẻ gọi dữ liệu là buổi sáng
	BBBB	
Giờ	h	Định dạng trả về là 12 giờ (Ví dụ 13 giờ chiều => 1 giờ)
	hh	Định dạng trả về là 12 giờ, từ 1 - 9 có số 0 phía trước (Ví dụ 13 giờ chiều => 01 giờ)
	H	Định dạng trả về là 24 giờ (Ví dụ 13 => 13)
	HH	Định dạng trả về là 24 giờ, từ 1 - 9 có số 0 phía trước (Ví dụ 1 => 01)
Phút	m	phút thứ 1 => 1
	mm	Phút thứ 1 => 01

Như vậy, dựa vào bảng trên, để hiển thị **thứ ngày tháng năm** ta dùng:

```
<b:eval expr='format ( data:tên_biến.date, "EEEE d M YYYY ")'/>
```

Trong đó :

- **EEEE** là trả về tên đầy đủ của thứ trong tuần (thứ hai đến chủ nhật)
- **d** là trả về ngày trong tháng
- **M** là trả về tháng trong năm
- **YYYY** là trả về năm

Trả về:

Thứ ba ngày 27 tháng 3 năm 2020

Kết hợp với các kí tự đặc biệt:

```
<b:eval expr='format ( data:tên_biến.date, "EEEE , d, M ,YYYY ")'/>
```

Trả về:

Thứ ba, ngày 27, tháng 3, năm 2020

Trước khi đi vào tìm hiểu các tiện ích widget nâng cao

Một số tips nhỏ cho các ông dễ hiểu

Các ông hãy nhìn vào một mảng dữ liệu của tiện ích Post sau:

```
- posts <- Là một mảng chứa các đối tượng post và các phân tử con của post
- post
  - id
  - title
  - labels <- Là một mảng chứa các đối tượng label1,label2,...
    - label1
    - label2
- post
  - id
  - title
  - labels
    - label1
    - label2
- post
  - id
  - title
  - labels
    - label1
    - label2
```

Để gọi title của từng post như trong ảnh việc đầu tiên các ông chắc chắn làm đó là lặp một vòng qua **posts**. Và dữ liệu mảng truyền đến values của vòng lặp sẽ là **data:posts** vì **posts** lúc này là ngang hàng với **con trở vị trí hiện tại** như ảnh (nếu các ông để ý ở mục thẻ tiện ích cho tiện ích Blog thì sẽ thấy **posts** là cao nhất nó chỉ chứa chứ nó không bị chứa):

```
[+] <- vị trí hiện tại

- posts <- Là một mảng chứa các đối tượng post và các phân tử con của post
  - post
    - id
    - title
    - labels <- Là một mảng chứa các đối tượng label1,label2,...
      - label1
      - label2
  - post
    - id
    - title
    - labels
      - label1
      - label2
  - post
    - id
    - title
    - labels
      - label1
      - label2
```

Khi các ông thực hiện vòng lặp. Thì con trỏ lặp **index** chỉ nhảy lần lượt qua từng object trong nó.
Như ví dụ sau:

```

- posts <- Là một mảng chứa các đối tượng post và các phân tử con của post
- post      [+] <- vị trí sẽ nhảy tới
  - id
  - title
  - labels <- Là một mảng chứa các đối tượng label1,label2,...
    - label1
    - label2
- post      [+] <- vị trí sẽ nhảy tới
  - id
  - title
  - labels
    - label1
    - label2
- posts     [+] <- vị trí sẽ nhảy tới
  - id
  - title
  - labels
    - label1
    - label2

```

Như vậy, khi con trỏ lặp index lặp tới vị trí đối tượng post, lúc này con trỏ ngang hàng với đối tượng này, nó không index sâu vào dữ liệu của đối tượng. Tuy vậy các ông hãy thấy mỗi đối tượng đều mang cho mình một mảng nữa đó là labels. Nhưng vì con trỏ index lặp không trở qua vị trí labels đó. Đó là lí do khi các ông muốn lặp thêm mảng **labels** bên trong mỗi đối tượng. Các ông cần phải truyền cho nó values đó là `data:tên_biến.labels` mà không phải là `data:labels`.

Như vậy có nghĩa là nếu con trỏ hiện đang ở vị trí nào thì các ông có quyền gọi trực tiếp dữ liệu ngang hàng với nó. Còn dữ liệu không ngang hàng thì cần phải có tham chiếu phụ.

Điều này giải thích vì sao một số tiện ích như Labels có thể sử dụng trực tiếp `data:labels` mà không cần thêm tham chiếu phụ.

data:labels trong tiện ích Labels khác với data:labels trong đối tượng thuộc mảng posts của tiện ích Blog,PopularPost,...

Thẻ dữ liệu tiện ích Labels

- title: tiêu đề tiện ích,
- display: trả về chuỗi là thông tin cách hiển thị nhãn (gồm list,cloud)
- showFreqNumbers: có hiển thị số lượng bài viết thuộc một nhãn hay không
- labels - trả về mảng gồm thông tin các nhãn
 - url: địa chỉ nhãn
 - name: tên nhãn
 - count: số lượng bài viết thuộc nhãn (nếu showFreqNumbers là true)

Như các ông thấy: **title.display,showFreqNumbers** và **labels** là ngang hàng và trước nó không bị chứa bởi đối tượng khác. Vì thế mặc định nó đều ngang hàng với con trỏ vị trí.

Toàn bộ khái niệm bao gồm con trỏ vị trí, con trỏ index ở đây đều là do tớ tự nghĩ ra để giải thích một cách dễ hiểu cách làm việc của vòng lặp và cấu trúc gọi dữ liệu cho các ông.

Tiện ích widget nâng cao

Gần kết thúc tài liệu

Đọc đến tận đây là coi như các ông đã có thể tự hoàn thành cho mình một giao diện rồi đúng không. Còn nếu không thì có lẽ là do tớ viết không tốt.

Đến phần này tớ sẽ trình bày khái quát về về cách hiển thị của từng widget. Và code mẫu đơn giản, còn việc sử dụng và code như thế nào đó sẽ là công việc của các ông. Vì trong tài liệu này, tớ đã giới thiệu về các thẻ dữ liệu ở phần trước , việc sử dụng chúng như nào bây giờ sẽ là công việc của các ông.

Tiện ích Blog

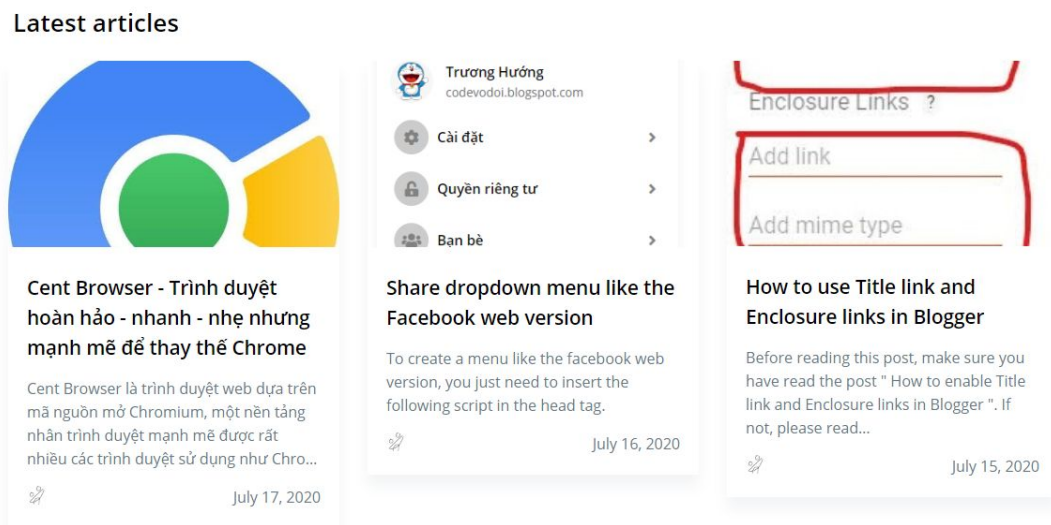
Để hiển thị bài viết và danh sách bài viết ra trang chủ các ông hãy dùng tiện ích Blog. Và đặt trong vòng lặp. Mặc định, số bài viết được trả về trong mảng `data:post` tương ứng với số lượng bài viết các ông cho hiển thị một lần trong **"cài đặt blog"**. Còn với trang bài viết và trang tĩnh, mặc định mảng có giá trị số lượng là 1.

Hiển thị bài viết ra trang chủ:

Điều này đúng cho cả trang chủ, trang nhấn, trang lưu trữ và trang tìm kiếm. Các ông chỉ cần hiểu là tiện ích Blog sẽ là nơi hiển thị :

- Các bài viết có trong blog cho người xem
- Hiển thị nội dung bài viết và trang tĩnh cho người xem

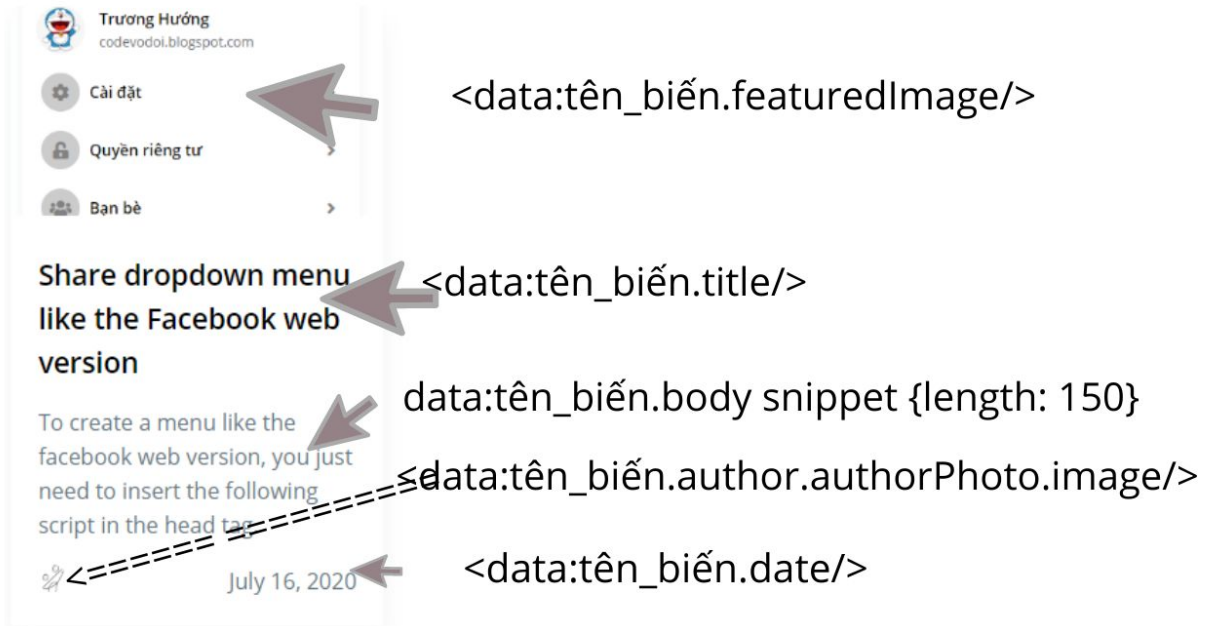
Các ông xem hình ảnh sau đây:



Một ví dụ đơn giản đây là danh sách bài viết được hiển thị ra trang chủ. Nhìn vào mỗi phần bài viết các ông dễ dàng nhận thấy chúng đều có.



Căn cứ vào thẻ gọi dữ liệu tiện ích Blog đã trình bày ở những phần đầu của tài liệu. Các ông dễ dàng nhận ra ngay rằng:



Như vậy cấu trúc để xuất được dữ liệu như trên ra đó tối thiểu nhất phải là (ở đây `tên_biến` tờ sẽ đặt là `post`) như sau (chưa kể phải đặt trong thẻ `b:includable` và `b:widget`, `b:section`)

```
<b:loop values='data:posts' var='post'>
  <data:post.featuredImage/>
  <data:post.title/>
  <b:eval expr='data:post.body snippet { length: 150 }' />
  <data:post.author.authorPhoto.image/>
  <data:post.date/>
</b:loop>
```

Đến đây thì các ông đã hiểu cách đơn giản nhất để hiển thị các bài viết rồi chứ nhỉ. Kết hợp với một số thẻ điều kiện nữa là các ông có thể hiển thị bài viết ở trang bài viết và trang tĩnh rồi. Ví dụ:

```

1
2 <b:section class='navbar' id='navbar' showaddelement='yes'>
3 <b:widget id='Blog1' title='Blog' type='Blog' version='2' visible='true'>
4   <b:includable id='main'>
5     <b:loop values='data:posts' var='post'>
6       <b:include name='trang-chu' cond='!data:view.isSingleItem' data='post' />
7       <b:include name='bai-viet' cond='data:view.isSingleItem' data='post' />
8     </b:loop>
9   </b:includable>
10
11   <b:includable id='trang-chu' var='post' >
12     <data:post.featuredImage />
13     <data:post.title />
14     <b:eval expr='data:post.body snippet { length: 150 }' />
15     <data:post.author.authorPhoto.image />
16     <data:post.date />
17   </b:includable>
18
19   <b:includable id='bai-viet' var='post'>
20     <data:post.title />
21     <data:post.body />
22   </b:includable>
23 </b:widget>
24 </b:section>

```

Dưới đây là một số thẻ b:includable mặc định của tiện ích Blog mà không thể xóa được, mỗi lần cập nhật giao diện từ “**edit HTML**”, blogger sẽ luôn kiểm tra xem giao diện của các ông đã chứa thẻ b:includable nào có id như các id sau đây chưa, nếu chưa nó sẽ tự lấy mặc định và thêm vào.

Những thẻ b:includable có id sau sẽ được tự động thêm vào, đối với những id mà trong mã của các ông đã xuất hiện, thì nó sẽ không thêm vào nữa (khác giống việc ghi đè). Vì vậy tớ khuyên các ông là khi viết giao diện cho blog hãy gắng đặt id cho các thẻ b:includable trùng với các id dưới đây theo chức năng. Tránh việc Blogger tự thêm mã vào mặc dù nó không được sử dụng.

- main - chứa các mã dùng hiển thị nội dung của widget , là các tối quan trọng
- aboutPostAuthor - chứa các mã hiển thị thông tin về tác giả
- commentAuthorAvatar - chứa các mã dùng hiển thị avatar của thằng bình luận
- commentDeletelcon - chứa các mã dùng hiển thị nút xóa bình luận
- commentForm - chứa các mã dùng hiển thị khung bình luận
- commentFormIframeSrc - chứa các mã dùng hiển thị liên kết nhúng khung bình luận
- commentItem - chứa các mã dùng hiển thị nội dung từng bình luận
- feedLinks - chứa các mã dùng hiển thị các liên kết feed của bài viết
- postMeta - chứa các mã dùng hiển thị các thông tin về bài viết
- postPagination - chứa các mã dùng hiển thị nút phân trang
- postTitle - chứa các mã dùng hiển thị tiêu đề bài viết
- previousPageLink - chứa các mã dùng hiển thị bài viết trước
- commentLists - chứa các mã dùng hiển thị danh sách bình luận theo commentItem

- commentPicker - chứa các mã dùng hiển thị bình luận theo cài đặt (nhúng, popup,...)
- comments - chứa toàn bộ những cái liên quan đến bình luận
- addComments - chứa các mã dùng hiển thị “thêm bình luận”
- feedLinksBody - chứa các mã hiển thị feed bài viết
- homePageLink - chứa các mã dùng hiển thị liên kết về trang chủ
- inlineAd - chứa các mã hiển thị quảng cáo
- nextPageLink - chứa các mã dùng hiển thị liên kết đến bài tiếp theo
- post - chứa các mã dùng hiển thị bài viết
- commentsTitle - chứa các mã dùng hiển thị tiêu đề bình luận
- threadedCommentForm - chứa các mã dùng hiển thị bình luận
- threadedComments - chứa các mã dùng hiển thị bình luận
- postBody - chứa các mã dùng hiển thị thân bài viết
- postBodySnippet - chứa các mã dùng hiển thị nội dung cắt ngắn của bài viết
- postCommentsAndAd - chứa các mã dùng hiển thị bình luận và quảng cáo
- postCommentsLink - chứa các mã dùng hiển thị liên kết đến bình luận
- postFooter - chứa các mã dùng hiển thị chân bài viết
- postFooterAuthorProfile - chứa các mã dùng hiển thị thông tin tác giả dưới chân bài viết
- postHeader - chứa các mã dùng hiển thị đầu của bài viết
- Và một số khác nữa

Chức năng của chúng tôi chỉ ghi tóm tắt như vậy, nó có thể mở rộng hoặc thu hẹp chức năng tùy theo mục đích của người viết.

Như vậy, các tiện ích tiếp theo đều có cách sử dụng tương tự nếu như các ông đã nắm được cách hoạt động của tiện ích Blog.

Sử dụng `<b:includable id='xxx'/>` để ngăn Blogger tự động thêm thẻ `b:includable` mặc định với id là `xxx`

Tiện ích PopularPosts

Thẻ `b:includable` mặc định của PopularPosts

- main - hiển thị nội dung chính

Hiển thị bài viết thuộc tiện ích PopularPost bằng đoạn mã đơn giản nhất:

```
7      <b:includable id='main'>
8          <b:loop values='data:posts' var='post'>
9              <!-- Hiển thị id bài viết -->
10             <data:post.id/>
11             <!-- Tiêu đề bài viết -->
12             <data:post.title/>
13             <!-- Hiển thị nội dung cắt ngắn của bài viết -->
14             <b:eval expr='data:post.body snippet { length:150 }' />
15             <!-- Hiển thị hình ảnh bài viết -->
16             <data:post.featuredImage/>
17             <!-- Địa chỉ bài viết -->
18             <data:post.url/>
19             <!-- Ngày đăng bài -->
20             <data:post.date/>
21             <!-- Nhân -->
22             <b:loop values='data:post.labels' var='label'>
23                 <!-- Tên nhân -->
24                 <data:label.name/>
25                 <!-- Địa chỉ nhân -->
26                 <data:label.url/>
27             </b:loop>
28             <!-- tên tác giả bài viết -->
29             <data:post.author.name/>
30         </b:loop>
31     </b:includable>
32
```

Tiện ích FeaturedPost

Thẻ `b:includable` mặc định của `FeaturedPost`

- `main` - hiển thị nội dung chính
- `content` - chứa nội dung cần hiển thị

```

5 <b:includable id='main'>
6   <b:loop values='data:posts' var='post'>
7     <!-- Hình ảnh bài viết -->
8     <img expr:src='data:post.featuredImage' />
9     <!-- Tiêu đề bài viết -->
10    <data:post.title />
11    <!-- Đoạn cắt ngắn nội dung -->
12    <b:eval expr=' data:post.body snippet { length: 350} ' />
13    <!-- Ảnh tác giả -->
14    
15    <!-- Ngày đăng -->
16    <data:post.date />
17    <!-- Tên tác giả -->
18    <data:post.author.name />
19  </b:loop>
20 </b:includable>

```

Tiện ích Labels

```

6 <b:includable id='main'>
7   <b:loop values='data:labels' var='label'>
8     <!-- Tên nhãn -->
9     <data:label.name />
10    <!-- Địa chỉ nhãn -->
11    <data:label.url />
12  </b:loop>
13 </b:includable>

```

Những tiện ích khác. Các ông căn cứ và thẻ dữ liệu đã được trình bày trong phần trước để sử dụng cho phù hợp.

Tổng kết

Tổng kết lại sau một quá trình dài dai dẳng

Một giao diện blog khi được thiết kế nhất thiết phải có:

- Thẻ **b:skin**
- Ít nhất là một thẻ **b:section**

Trong thẻ **b:section**

- Chỉ được chứa các thẻ **b:widget**
- Không được chép chứa thẻ **b:section** khác

Thẻ **b:widget**:

- Là thẻ hiển thị nội dung với chức năng nhất định
- Chỉ được phép chứa thẻ các **b:widget-settings**, **b:widget-setting** và **b:includable**

Trong thẻ **b:includable**:

- Có thể chứa mã HTML, CSS, JS, các thẻ gọi dữ liệu liên quan đến widget đang chứa nó để hiển thị nội dung
- Có thể chứa thẻ **b:include**

Sử dụng các thẻ tiện ích **widget** theo chức năng và căn cứ theo từng thẻ widget đó mà sử dụng các thẻ gọi dữ liệu để thiết kế blogspot cho phù hợp.

Nếu gặp một mảng, hãy sử dụng thẻ **b:loop** để lấy dữ liệu.

Phụ lục

Lưu ý ban đầu:

Để phòng trường hợp một số Blog khi cập nhật sẽ bị lỗi không hiển thị được khung bình luận như hướng dẫn sau. Vui lòng các ông thêm đoạn mã sau vào trong thẻ :

```
<b:skin><![CDATA[
```

Chèn tại đây

```
]]></b:skin>
```

Hoặc tại đây : <https://truong-huong.github.io/code/comment-b-skin.txt>

```
/* <Group description="Body">
```

```
<Variable name="body.background" description="Background" type="background"
color="$(body.background.color)" default="$(color) none repeat scroll top left"
value="#ffffff url(#) no-repeat scroll top center /* Credit: Mae Burke
(http://www.offset.com/photos/389967) */;"/>
```

```
<Variable name="body.gradient.first" description="First Color" type="color"
default="#f00" value="#ff0000"/>
```

```
<Variable name="body.gradient.second" description="Second Color" type="color"
default="#ffa500" value="#ffa500"/>
```

```
<Variable name="body.gradient.third" description="Third Color" type="color"
default="#ff0" value="#ffff00"/>
```

```
<Variable name="body.gradient.fourth" description="Fourth Color" type="color"
default="#008000" value="#008000"/>
```

```
<Variable name="body.gradient.fifth" description="Fifth Color" type="color"
default="#0591d6" value="#0591d6"/>
```

```
<Variable name="body.background.color" description="Background color" type="color"
default="transparent" value="transparent"/>
```

```
<Variable name="body.title.font.small" description="Title font (small)" type="font"
default="$(garamond20)" value="bolder 900 14px Inconsolata,monospace, sans-serif"/>
```



```
<Variable name="body.title.font.large" description="Title font (large)" type="font"
default="$(garamond24)" value="bolder 900 24px Inconsolata,monospace, sans-serif"/>
```

```
<Variable name="body.title.color" description="Title color" type="color" default="#000"
value="#ffffff"/>
```

```
<Variable name="body.action.font.small" description="Action font (small)" type="font"
default="$(Inconsolata,monospace12)" value="bolder 900 12px Inconsolata,monospace,
sans-serif"/>
```

```
<Variable name="body.action.font.large" description="Action font (large)" type="font"
default="$(Inconsolata,monospace14)" value="bolder 900 14px Inconsolata,monospace,
sans-serif"/>
```

```
<Variable name="body.action.color" description="Action color" type="color"
default="#e52e71" value="#e52e71"/>
```

```
<Variable name="body.text.font" description="Text font" type="font"
default="$(garamond20)" value="bolder 900 14px Inconsolata,monospace, sans-serif"/>
```

```
<Variable name="body.text.color" description="Text color" type="color" default="#000"
value="#ffffff"/>
```

```
<Variable name="body.link.color" description="Link color" type="color"
default="#4267b2" value="#4267b2"/>
```

```
<Variable name="body.widget.title.font.small" description="Gadget title font (small)"
type="font" default="$(Inconsolata,monospace12)" value="bolder 900 12px
Inconsolata,monospace, sans-serif"/>
```

```
<Variable name="body.widget.title.font.large" description="Gadget title font (large)"
type="font" default="$(Inconsolata,monospace14)" value="bolder 900 14px
Inconsolata,monospace, sans-serif"/>
```

```
<Variable name="body.widget.title.color" description="Gadget title color" type="color"
default="rgba(0, 0, 0, 0.54)" value="rgba(0, 0, 0, 0.54)"/>
```

```
<Variable name="body.filter.background.color" description="Filter background color" type="color" default="#302c24" value="#302c24"/>
```

```
<Variable name="body.filter.text.font.small" description="Filter text font (small)" type="font" default="$(Inconsolata,monospace12)" value="bolder 900 12px Inconsolata,monospace, sans-serif"/>
```

```
<Variable name="body.filter.text.font.large" description="Filter text font (large)" type="font" default="$(Inconsolata,monospace14)" value="bolder 900 14px Inconsolata,monospace, sans-serif"/>
```

```
<Variable name="body.filter.text.color" description="Filter text color" type="color" default="rgba(255, 255, 255, 0.54)" value="rgba(255, 255, 255, 0.54)"/>
```

```
<Variable name="body.filter.keyword.font.small" description="Filter keyword font (small)" type="font" default="$(Inconsolata,monospaceBold12)" value="normal 700 12px Inconsolata,monospace, sans-serif"/>
```

```
<Variable name="body.filter.keyword.font.large" description="Filter keyword font (large)" type="font" default="$(Inconsolata,monospaceBold14)" value="normal 700 14px Inconsolata,monospace, sans-serif"/>
```

```
<Variable name="body.filter.keyword.color" description="Filter keyword color" type="color" default="rgba(255, 255, 255, 0.87)" value="rgba(255, 255, 255, 0.87)"/>
```

```
<Variable name="body.filter.link.font.small" description="Filter link font (small)" type="font" default="$(Inconsolata,monospaceBold12)" value="normal 700 12px Inconsolata,monospace, sans-serif"/>
```

```
<Variable name="body.filter.link.font.large" description="Filter link font (large)" type="font" default="$(Inconsolata,monospaceBold14)" value="normal 700 14px Inconsolata,monospace, sans-serif"/>
```

```
<Variable name="body.filter.link.color" description="Filter link color" type="color" default="#4267b2" value="#4267b2"/> </Group> */
```

Làm thế nào để tạo một form comment trong bài viết

Trước khi tạo được một form comment trong bài viết. Điều đầu tiên các ông cần phải nhớ lại và quan tâm là các thẻ dữ liệu liên quan đến comment. Những thẻ cơ bản và cần thiết này bao gồm:

- allowComments
- allowNewComments
- enabledCommentProfileImages
- numberOfComments
- noNewCommentsText
- comments : trả về mảng các bình luận hiện có của bài viết
 - id : id bình luận
 - inReplyTo : trả về true nếu bình luận này được một ai đó trả lời lại
 - cmtBodyIdPostfix : trả về một chuỗi có dạng `_cmt-id` được dùng để nhận dạng bình luận sau này.
 - url : liên kết đến bình luận này
 - body : nội dung bình luận
 - timestamp : thời gian bình luận dạng chuỗi
 - timestampValue : thời gian bình luận dạng số
 - timestampAbs : thời gian bình luận dạng số tuyệt đối
 - author : tên người bình luận
 - authorUrl : liên kết đến hồ sơ người bình luận
 - authorUserType : là người dùng bình luận bằng blogger hay anonymous

 - authorPhoto
 - url : liên kết đến hình ảnh của người bình luận
 - width : rộng ảnh
 - height : cao ảnh
 - authorAvatarSrc : liên kết đến hình ảnh avatar của tác giả bình luận
 - authorAvatarImage: khá tương tự authorAvatarSrc
 - anchorName : trả về một chuỗi số dùng để nhận dạng tác giả của bình

- o luận
- o deleteUrl : liên kết để xóa bình luận này
- o isDeleted : true nếu bình luận này đã bị xóa
- o adminClass : trả về một chuỗi là tên một class tên blog-admin để xác định ai là quản trị và ai là khách truy cập

Những thẻ này là những thẻ dùng để xuất dữ liệu bình luận, còn để ghi bình luận thì chúng ta sẽ phải sử dụng một khung nhúng của Google.

Xác định khung bình luận và các chức năng các nút cần có.

Một khung hiển thị bình luận chuẩn thường có những thông tin sau:



Dựa vào các thẻ gọi dữ liệu liên quan đến bình luận. Ta có thể code được một khung bình luận như sau:

Các mã html đều phải ở bên trong:

```
<b:loop values='data:posts' var='tên_biến'>
    </b:loop>
```

Cửa tiện ích Blog

```

<b:includable id='main'>
  <b:include name='comments' />
</b:includable>

<b:includable id='comments'>
  <b:loop values='data:posts' var='post'>
    <b:if cond='data:post.allowComments'>
      <b:if cond='data:post.allowNewComments'>
        <!-- Bình luận tại đây -->
        <iframe allowtransparency='true'
          expr:data-comment="//www.blogger.com/comment-iframe.g?blogID=" + data:blog.blogId +
          "&" + (data:view.isPost ? "postID" : "pageID") + "=" + data:post.id + "&skin=soho"
          expr:src="//www.blogger.com/comment-iframe.g?blogID=" + data:blog.blogId + "&"
          + (data:view.isPost ? "postID" : "pageID") + "=" + data:post.id + "&skin=soho" />
        </iframe>
      </b:if>
      <b:loop values='data:post.comments' var='comment'>
        <b:if cond='!data:comment.inReplyTo'>
          <div class='comment'>
            <b:include data='comment' name='commentItem' />
            <div class='replies'>
              <b:loop values='data:post.comments' var='comment2'>

                <b:if cond='data:comment2.inReplyTo == data:comment.id'>
                  <b:include data='comment2' name='commentItem' />
                  <div class='replies'>
                    <b:loop values='data:post.comments' var='comment3'>
                      <b:if cond='data:comment3.inReplyTo == data:comment2.id'>
                        <b:include data='comment3' name='commentItem' />
                        <!-- Độ sâu của phân hội tối đa bằng cách thêm một vòng lặp
tạo đây
data:comment3.id'>
                          <b:include data='comment4' name='commentItem' />
                        </b:if>
                      </b:loop>
                    </div>
                  </b:if>
                </b:loop>
              </div>
            </b:if>
          </b:loop>
        </div>
      </b:loop>
    </b:if>
  </b:loop>
</b:includable>

```

```

        </div>
      </b:if>
    </b:loop>
  </b:if>
</b:includable>

<b:includable id='commentItem' var='comment'>
  <div class='comment_item' expr:id='"c" + data:comment.id'>
    <b:if cond='data:blog.enabledCommentProfileImages'>
      <!-- Ảnh bình luận -->
      <img expr:src='data:comment.authorAvatarSrc' />
    </b:if>
    <!-- Tên người bình luận -->
    <b:if cond='data:comment.authorUserType != "anonymous"'>
      <span expr:class='"is_" + ((data:post.author.name == data:comment.author) ? "author" :
"users")'>
        <data:comment.author/>
      </span>
    <b:else/>anonymous</b:if>
    <!-- Thời gian bình luận -->
    <data:comment.timestamp/>
    <!-- Nội dung bình luận -->
    <div expr:class='"comment-body" + (data:comment.isDeleted ?: " deleted")'>
      <data:comment.body/>
    </div>
    <!-- Trả lời tại đây -->
    <iframe src="//www.blogger.com/comment-iframe.g?blogID=" + data:blog.blogId + "&" +
(data:view.isPost ? "postID" : "pageID") + "=" + data:post.id + "&parentID=" + data:comment.id +
"&skin=soho"'></iframe>
    <a expr:href='data:comment.deleteUrl'>Xóa comment</a>
  </div>
</b:includable>

```

Xem cụ thể tại đây:

<https://raw.githubusercontent.com/truong-huong/code/master/comment-tai-lieu.html>

Tớ chỉ có thể hướng dẫn các ông đến đây. Việc vận dụng thẻ gọi dữ liệu nào, điều kiện ra sao,... và làm thế nào để có thể tối ưu được những đoạn code trên hay vận dụng JS và CSS là do tính sáng tạo của các ông tớ không thể can thiệp và hướng dẫn cụ thể được. Vì vậy các ông hãy tự làm và thiết kế cho riêng mình.