

JadaSite Template Tutorial 1.0

Version 1.0, 2007-2008

Copyright @ 2007-2008 JadaSite

www.jadasite.com

License

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means electronic, mechanical, photocopying, recording or otherwise without the prior written permission of the owner.

Permissions may be sought directly by emailing admin@jadasite.com.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY

THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE.

Table of Contents

Preface.....	4
About this Guide.....	4
Chapter 1 – Introduction.....	5
Chapter 2 – How a web page is produced in JadaSite.....	6
Chapter 3 – Template	7
Use of Apache Velocity.....	7
Structure of templates.....	8
Location of template.....	13
Chapter 4 – Creating template.....	14
Branding public and secure site by simply overriding styles.....	14
Brand public and secure site by creating a new /template.vm and/or /templateSecure.vm file.....	15
Further branding the public site by overriding more template pages.....	15
Branding secure site by overriding stylesheets.....	15
Appendix.....	17

Preface

Welcome to JadaSite. JadaSite is an easy to use and feature-rich content management and e-commerce system. More importantly, JadaSite is an open sourced project and the code is freely available for download. We believe open source is a better model since users can obtain the code, see the code and change it. They can also distribute the code that they have modified. If customers do not like the vendor that is serving them, they can freely switch to another vendor that they prefer or decide to support the product themselves.

Our philosophy is to ensure that JadaSite is feature-rich, easy to use and maintain, and does not require an I.T. Team for support and to make constant changes to the system. We believe in simplicity, and we use simple tools and frameworks to ensure that future changes by other parties outside of our core development team can be done easily. We constantly monitor the market to understand the trend and bring new features and ideas into JadaSite as long as they help our users to promote their sites and products.

Many sites are simply information only or mainly targeted on collaboration. However, a lot of commercial sites are for more than just information. Some sites need to sell, and selling is their business. JadaSite invests a lot of effort to help companies to promote their products and generate sales.

About this Guide

The JadaSite Template Tutorial is intended for template designer who already have concept on JadaSite content management system. If not, please refer to **JadaSite User Guide**.

Reader should already has basic understanding of authoring web site using HTML and CSS style. Since JadaSite use Apache Velocity to create template, it is desirable to understand the features of VTL (Velocity Template Language) and how to put VTL into action.

Chapter 1 – Introduction

The right design for one web site does not mean it is the right design for another. Companies or web sites have their branding strategy and they usually have their own preference of colors, styles, fonts, etc. There is no one right design that is good for all companies or web sites. To accomplish this, JadaSite allows the use of templates and templates can be uploaded to the site and the look and feel of the site can be skinned instantly. In addition, JadaSite provides functionality to create new templates and to fine-tune existing templates.

A template is a tool for enforcing a standard layout and look and feel across multiple pages of the web site. It separates data from the front-end design and makes it easier to use the same template for different sites regardless of the data delivered to the template.

There are multiple ways to make use of the templates. Some are very simple and some can be more involved and require a bit of technical skills. We are going to explore into the different ways of creating or maintaining existing templates.

Use default, 'basic', template bundled with JadaSite.

JadaSite is bundled with a default template named as 'basic'. As the name suggests, the layout and color coordination is rather neutral and simple. But, it has all the functions and features that JadaSite supports.

Leverage default, 'basic', template by overriding specific template file

JadaSite allows template designers to predominantly use the default, 'basic', template and only override the template files that need to be modified.

Upload external templates.

There are templates that are already created and ready for download. Once the template is downloaded, it can be uploaded directly to JadaSite and assigned to the site that you want to skin. A template uploaded may still not be 100% satisfactory due to the unique requirements of the site. In this case, template designers can modify the uploaded template via the JadaSite.

Create new templates.

Sometimes the design of the template that is required is so unique that no existing template can be found. In this case, a new template can be created from scratch.

Note: To learn how to operate the interface to upload, update and create templates, please refer to **JadaSite Template Tutorial**.

Chapter 2 – How a web page is produced in JadaSite

JadaSite is template based. Within JadaSite, there is a template engine that handles processing of web pages once they are requested.

The following is the sequence how a web page is generated.

1. Visitor hits a URL.
2. Template engine interprets the URL to determine the type of pages that the visitor is hitting. These pages can be the home page, section display, item details, content details, etc.
3. Based on what is defined on site maintenance, template engine knows which template to apply.
4. Template engine extracts required information from database based on the URL.
5. Template engine locate the template file to be used for this URL.
6. If the site is using a user defined template and the required template page is not defined in the user defined templates, template engine use the one in the 'default' template.
7. The extracted information is merged with the template file to produce the web page.

URLs that template engine supports

Home page - /jada/web/fe/home

Section page - /jada/web/fe/section/{section info}/{section info}

Item details - /jada/web/fe/item/{item number}

Content details - /jada/web/fe/content/{content title}

Contact us - /jada/web/fe/contactUs

Search result - /jada/web/fe/search

Chapter 3 – Template

Use of Apache Velocity

JadaSite templates are implemented based on the use of Apache Velocity engine. Velocity use VTL (Velocity Template Language) to author templates. VTL is meant to provide the easiest, simplest and cleanest way to incorporate dynamic content in the web page. Even a web developer with little or no programming experience should soon be capable of using VTL to incorporate dynamic content in a web site. Please reference to Apache Velocity web site for full documentation of VTL.

Data delivered by template engine to the template pages are done in VTL context.

There are three types of references in VTL: variables, properties and methods. All VTL variables, properties and methods are associated with a Java class.

Variables

Variables consists of a leading “\$” character followed by a VTL identifier.

Examples

\$foo

Properties

The second flavor of VTL references are properties of variables. The shorthand notation consists of a leading \$ character followed a VTL identifier, followed by a dot character (“.”) and another VTL identifier.

Examples

\$itemInfo.itemDesc

\$contentInfo.contentTitle

Methods

A method is defined in the template engine and is capable to handling complicated logic. Methods are references that consist of a leading “\$” character followed by a VTL identifier, followed by a VTL Method Body. A VTL Method Body consists of a VTL Identifier followed by an left parenthesis character (“(“), followed by an optional parameter list, followed by right parenthesis character (“)”).

Examples

\$template.getHorizontalMenu('MAIN', "")

In addition, Apache Velocity provides the following script element to help authoring templates pages and they are widely used in JadaSite.

Include

The #include script element allows the template to import a local file, which is then inserted into the location where the #include directive is defined.

Examples

```
#include("file.htm")
```

Parse

The `#parse` script element allows the template designer to import a local file that contains VTL. Velocity will parse the VTL and render the template specified.

Examples

```
#parse("me.vm")
```

Note: The above discussion is to allow us to have enough background for this tutorial. Apache Velocity offers a lot more than the above and they can be used in JadaSite template authoring. In fact, some of them are already used in the standard, 'default', template.

Structure of templates

Template does not consist of one single template file. A template is made up of a number of template files.

Some template files contain common code and they are imported into other template files via `#include` or `#parse` directives.

Template file may also contain zero or many widgets. These widgets usually require special logic. When a widget is requested, the template engine does necessary processing to extract the data, merge the data with the widget's template file and deliver the result to the requesting template.

The following are the two template files that are absolutely required. One is for the public site and the other is for the secure site. They have the layout and the look and feel of the whole site. In these two files, it layout where specific information for each page should be shown as well as where different widgets should be shown.

1. `/template.vm`
2. `/templateSecure.vm`

The following is the code for `/template.vm` from the default, 'basic', template.

```
<html>
<head>
<title>${pageInfo.pageTitle}</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="Keywords" content="${pageInfo.pageTitle}">
<meta name="Description" content="${siteInfo.pageTitle}">

```

Load the following CSS stylesheets.

- styles.css
- yui/menu/assets/menu.css

If a user defined template is defined for the site and the stylesheets exists in the user defined template, they will be loaded from the user defined template. Otherwise, they will be loaded from the default, 'basic', template.

```
<link rel="stylesheet" type="text/css" href="${template.getResourcePrefix('yui/menu/assets/menu.css')}">
<link rel="stylesheet" type="text/css" href="${template.getResourcePrefix('styles.css')}">
<script type="text/javascript" src="${componentInfo.servletResourcePrefix/basic/jcCommon.js}"></script>
<script type="text/javascript" src="${componentInfo.servletResourcePrefix/basic/yui/yahoo/yahoo.js}"></script>
<script type="text/javascript" src="${componentInfo.servletResourcePrefix/basic/yui/dom/dom.js}"></script>
<script type="text/javascript" src="${componentInfo.servletResourcePrefix/basic/yui/event/event.js}"></script>
<script type="text/javascript" src="${componentInfo.servletResourcePrefix/basic/yui/container/container_core.js}"></script>
```

```

<script type="text/javascript" src="$componentInfo.servletResourcePrefix/basic/yui/menu/menu.js"></script>
<script type="text/javascript" src="$componentInfo.servletResourcePrefix/basic/yui/connection/connection.js"></script>
</head>
<body bgcolor="#FFFFFF" text="#000000" leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<div align="center" style="background-color: #e0e0e0">
<div style="margin:0pt auto; min-width:950px; width:80%;">
<br>
<div align="left" style="height:10px;">
<div align="left" style="height:10px;">
Include horizontal menu widget and use menu set 'MAIN' as the data. This
widget is align by the <div> tags and is set at the top of the page.
$template.getHorizontalMenu('MAIN', ")
</div>
<br><br>
</div>
</div>

<div align="center">
<div style="margin:0pt auto; min-width:950px; width:80%;">
<table width="100%" border="0" cellspacing="0" cellpadding="5">
<tr>
<td align="center" style="width:60%; height:30px; vertical-align: top;">
Show the name of the site.
<td nowrap><font size="36px" face="Arial, Helvetica, sans-serif">$siteInfo.siteName</font>
</td>
<td align="right" style="width:40%; height:30px; vertical-align: top;">
Include the search input box.
#parse("search/query.vm")
</td>
<td align="right" style="width:40%; height:30px; vertical-align: top;">
Show the site logo as the image link.
<a href="$componentInfo.contextPath/web/fe/home">

</a>
</td>
</tr>
</table>
<hr>
<br>
<table width="100%" border="0" cellspacing="0" cellpadding="5">
<tr>
<td align="center" style="width:100%; height:30px; vertical-align: top;">
This section shows the body of the page. Depending on the URL the visitor
hits., it can be the home page, item details page, content detail page, etc.
<td align="top">$pageInfo.pageBody</td>
</tr>
</table>
</div>
</div>
<br>
<table class="t1_template_site_footer" style="width:100%; border:0px; padding:3px;">
<tr>
<td align="left" style="width:100%; height:30px; vertical-align: top;">
Show the footer of the site.
<div align="left">$siteInfo.siteFooter</div>
</td>
</tr>
</table>
</body>
</html>

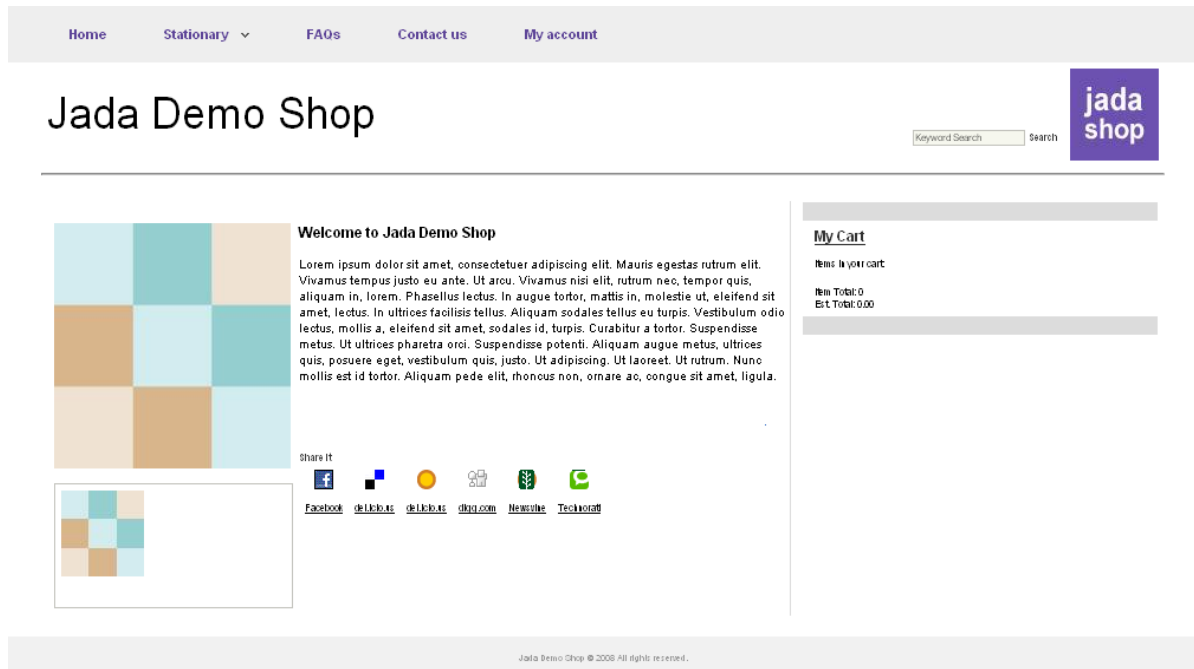
```

This is the code for /templateSecure.vm

Simply include template.vm. This really means there is no difference between public and secure site.

```
#parse("template.vm")
```

The above template produce the following result for the content detail page.



You can clearly see from this page how the template drive the layout and the look and feel of the page.

In the middle of this page, there is block of information that says “Welcome to Jada Demo Site’ pn the left as well as My Cart section on the right. These are being positioned on the page by the following VTL snippets.

```
$pageInfo.pageBody
```

Based on the URL of the page, template engine knows a content details page for a particular content is to be rendered. It prepares the content, reads template file, /content/content.vm, and merge the content and the template file to produce an HTML block. This HTML block is delivered to the template via \$pageInfo.pageBody.

While parsing /content/content.vm, the shopping cart summary gadget is requested. Template engine prepares data for the shopping cart summary gadget, loads template file, /components/shoppingCartSummary.vm, and merge the two to produce an HTML block.

Code snippet from /content/content.vm.

```
<td style="width:300px" valign="top">
  To determine if check-out process is enabled.
  #if ($template.isShoppingCart())
  <div class="jc_horizontal_line_seperator"></div>
  <table border="0" cellpadding="0" cellspacing="0" width="100%">
  <tr>
  To render the shopping cart summary gadget.
  <td> $template.getShoppingCartSummary() </td>
  </tr>
  </table>
  #end
```

```
<div class="jc_horizontal_line_seperator"></div>
</td>
```

As you can see, templates files goes hand in hand with template data. Template engine not only loads template files, it also prepares template data to merge with template files.

There is template data that is available for all pages. There are also template data that are prepared only for specific template pages.

Template data are delivered to the template in VTL variables. These variables usually has a number of properties.

The following is a list of variables that can be access in all template pages.

- SiteInfo \$siteInfo
- ComponentInfo \$componentInfo

The following is the list of template files and the VTL variables that it may access.

Note: This list contains top level template files that are accessed directly by template engine. These files can access other template files via `#parse` and the VTL variables can also be accessed in the files that they parse. Files may also be loaded by the top level template files via `#include`. Files that are loaded via `#parse` and `#include` (non-top level) are not included in this list.

Template page	Class and variable	Description
/home/home.vm	HomeInfo \$homeInfo	Home page
/section/section.vm	SectionInfo \$sectionInfo	Section page
/item/item.vm	ItemInfo \$itemInfo	Item detail page.
/content/content.vm	ContentInfo \$contentInfo	Content detail page.
/contactUs/contactUs.vm	ContactUsInfo \$contactUsInfo	Contact us page.
/search/search.vm	SearchInfo \$searchInfo	Search result page.
/components /menus/horizontalMenu.vm	String \$horizontalMenuCode String \$menuDivId	Render the horizontal menu.
/components /menus/verticalMenu.vm	String \$verticalMenuCode String \$menuDivId	Render the vertical menu.
/components/poll/poll.vm	n/a	Render the poll component.
/components /shoppingCart /shoppingCartSummary.vm	ShoppingCartSummaryInfo \$shoppingCartSummaryInfo	Render the My Cart component.
/components /syndication /syndication.vm	Vector \$syndications	Render the syndication component.
/mail/adminSaleNotification.vm	String \$adminSaleNotification	Render notification email for system administrator once a sale is completed.
/mail/custSaleConfirmation.vm	String \$custSaleConfirmation	Render sale confirmation email for customer once a sale is completed.

In addition, template engine delivers useful information via VTL modules. VTL modules can be accessed via the following VTL variable.

- \$template

Location of template

The standard, 'default', template is bundled with the distribution. This template can be viewed via the admin console. However, it cannot be changed. No changes to this template is recommended either by patching the distributed file or any other means. Since other templates may be relying on the standard, 'basic' template, changes the 'basic' template may cause adverse impact to other templates.

User defined templates are located in the following directory. These templates can be changed via admin console.

```
/{UserDefinedDirectory}/p/{SiteName}/template/{TemplateName}
```

Chapter 4 – Creating template

JadaSite template engine makes creating new template very easy by allowing template designer to override default template, 'basic' or to create completely new template.

In this chapter, we are going to explore different ways in which new templates can be created.

Branding public and secure site by simply overriding styles

This is the simplest way of branding the site. Template designer, in general, satisfied with the layout of the default, 'basic', template as well as the information that is displayed. He/she simply would like to change the color scheme, font, etc.

JadaSite default template, 'basic', stores all CSS styles in the following stylesheets. There are 2 ways how styles can be overridden.

```
/styles.css  
/yui/menu/assets/menu.css
```

Create new stylesheets in the new template

Template designer can simply make a copy of the stylesheet from the default, 'basic', template and only modify the required styles. The new style file need to be located in the exact location under the new template directory.

Since the default template use the following statement to pick up the style file, it checks the new template directory for the style file. If no style file can be found, the one on the default template directory will be used.

```
<link rel="stylesheet" type="text/css" href="$template.getResourcePrefix('styles.css')">
```

Create new /template.vm and/or /templateSecure.vm to include inline CSS

Template designer can also create new /template.vm and/or /templateSecure.vm file and include inline CSS definition to overridden those that are defined in the stylesheets.

Once the following code is created in /template.vm and/or /templateSecure.vm, all horizontal menu will have red background.

```
<style type="text/css">  
.yuimenuubar {  
  background-color:red;  
}  
</style>
```

Brand public and secure site by creating a new /template.vm and/or /templateSecure.vm file

A lot of the times, template designer would like to do more than just changing the color and font of the site. He/she may want to simply change the layout of the major components that are displayed and he/she is generally satisfied with the information that is displayed by the components.

This can be achieved by simply creating a new /template.vm and/or /templateSecure.vm file in the new template directory. When rendering new pages, template engine use the new /template.vm and/or /templateSecure.vm created together with other existing template files from default, 'basic', template.

For example, in the default, 'basic', template, there is only a horizontal menu running across at the top of the page. In some cases, template designer would like to have a vertical menu on the left of the page as well. This can be done by creating a new /template.vm and/or /templateSecure.vm file (copied from default, 'basic', template) and inserts the following statement to the template file.

```
$template.getVerticalMenu('SECONDARY', '')
```

This statement creates a vertical menu using data in menu set SECONDARY defined via admin console.

Note: Some HTML code need to be created as well to properly align the vertical menu.

Further branding the public site by overriding more template pages

Sometime, customizing the site by simply changing the CSS styles and /template.vm is not enough.

For example, on the item details page, images of the items are shown on the top left corner of the template page. Template designer may like to align the images to the right of the page. This can be done by creating template file, /item/item.vm, in the new template directory to override the one in default, 'basic' template.

Other example is template designer would like to show item rating information for the item when the item preview is shown on the section page. In this case, template designer can clone /common/previewItem.vm from default, 'basic', template to the new template directory and make changes to it.

In other words, template designer can rely on the default, 'basic', template for most the pages. He/she can simply pick and choose the page that they like to do differently. If required, all pages can be overridden to come up with completely new design based on the data (VRL variables) deliver to them via template engine.

Branding secure site by overriding stylesheets

Since secure site involve a lot of logic and security is a major concern, there is limitation imposed on branding the secure site.

Secure site can only be branded by modifying stylesheets and /templateSecure.vm. Information delivered to \$pageInfo.pageBody is not allowed to be altered in order to

protect the logic as well to maintain high level of security. Since information delivered is a piece of HTML code with CSS styles, the look and feel can still be altered by modifying the stylesheets.

Appendix

Class SiteInfo

String siteId	Site id.
String siteName	Name of the site.
String siteDomainName	Domain name of the site.
String publicURLPrefix	Reserved for future use.
String secureURLPrefix	Reserved for future use.
String siteFooter	Footer message of the site.

Class ComponentInfo

String requestURL	URL that visitor request.
String contextPath	Reserved for future use.
String resourcePrefix	Should be removed.
String servletResourcePrefix	Prefix of the template resource that is hosted in the servlet (JadaSite standard distribution). The prefix is likely to be /jada/content/template.
String templateResourcePrefix	Prefix of the template resource that is hosted on the local drive (user defined templates). The prefix is /p/{siteId}/template/{templateName}.
String templateName	Name of the template.

Class ContactUsInfo

String contactUsName	Contact name
String contactUsAddressLine1	Line 1 address
String contactUsAddressLine2	Line 2 address
String contactUsCityName	City name
String contactUsStateName	State name
String contactUsStateCode	State code
String contactUsCountryName	Country name
String contactUsCountryCode	Country code
String contactUsZipCode	Zip code
String contactUsDesc	Long description
String seqNum	Sort sequence of contact us
String contactUsEmail	Email address
String contactUsPhone	Phone number

Class DataInfo

String getAttribute(String key)	To get properties by key name.
---------------------------------	--------------------------------

Class PageInfo

String pageTitle	Page title and is intended to be used in the HTML <title> tag.
String pageBody	Body of the page. It can be the home page, item detail page, content detail page, etc. Content delivered in this properties depends on the URL of the visited page.

Class HomeInfo

Object[] homePageDatas	Contains a collection of DataInfo which can be either ItemInfo or ContentInfo. It contains all the content record and item record for the home page.
DataInfo homePageFeatureData	Contains the feature data for the home page. This data can be either ItemInfo or ContentInfo.
String pageTitle	Page title for the home page.

Class ContentInfo

String infoType	Hardcoded to be 'content'.
boolean feature	Specify if it is defined as the feature piece of information for the home page.
String contentNaturalKey	The key for this content that is made of the title of the content.
String contentId	Internal id of the content.
String contentTitle	Title of the content.
String contentShortDesc	Short description of the content.
String contentDesc	Description of the content.
String pageTitle	Title of the content detail page when this content is shown.
Stringing contentUrl	URL to the content detail screen for this content.
String contentDefaultImageUrl	URL of the default image for this content.
Vector contentImageUrls	A collection of all URLs for the images of this content.

Class ItemInfo

String infoType	Hardcoded to be 'item'.
boolean feature	Specify if it is defined as the feature piece of information for the home page.
boolean outOfStock	Specify if this item is currently out of stock.
String itemNaturalKey	The key of this item that is made up of the item number of this item.
String itemId	Internal id of this item.
String itemNum	Item Number.
String itemUpcCd	Item upc code.
String itemShortDesc	Item short description.
String itemShortDesc1	Second line of short description.
String itemDesc	Item long description.
String pageTitle	Title of the item detail page when this item is shown.
String itemPrice	Single price of the item.

String itemMultQty	Item quantity in the multiple pricing scenario.
String itemMultPrice	Item price in the multiple pricing scenario.
String itemSpecPrice	Special price of the item.
String itemSpecMultQty	Item quantity in the multiple pricing scenario when on special.
String itemSpecMultPrice	Item price in the multiple pricing scenario when on special.
String itemSpecPublishOn	Start date of the special price
String itemSpecExpireOn	End date of the special price
String itemHitCounter	Hit counter
String itemRating	Current rating of this item.
String itemRatingCount	Number of rating done for this item.
String itemQty	Current inventory of this item.
String itemBookedQty	Current booked (ordered and not shipped) quantity of this item.
String itemPublishOn	Publish date of this item.
String itemExpireOn	Expiry date of this item.
String itemUrl	URL to the item detail screen for this item.
String itemDefaultImageUrl	URL of the default image for this item.
Vector itemImageUrls	Collection of all URLs to the image of this item.
String itemOrderedQty	Number of items ordered in the current session.

Class MenuInfo

String menuName	Name of the menu set.
Int seqNo	Sequence number of menu items.
String menuAnchor	The HTML <a> tag.
String menuWindowTarget	The target of the following javascript command. window.open(URL, target, mode);
String menuWindowMode	The mode of the following javascript command. window.open(URL, target, mode);
String menuUrl	The URL of the following javascript command. window.open(URL, target, mode);
Vector menus	Collection of sub-menus for the current menu item.

Class SearchInfo

Int hitsCount	Number of items or contents located during the search.
String query	Search string that produce this result.
Object searchData[]	Collection of all items and contents for the current page.
int pageNum	Page number of this search page.
int pageStart	The start page of the navigation of this search.
int pageEnd	The end page of the navigation of this search.
int pageTotal	The total number of pages for this search.

Class SectionInfo

String sectionNaturalKey	The key to this section that is make up of the short title of the section.
String topSectionNaturalKey	Visitors can navigation the section hierarchical structure. This value determine the section where the visitor starts.
String sectionId	Internal id of this section.
String sectionShortTitle	Short title of this section.
String sectionTitle	Title of this section.
String sectionDesc	Description of this section.
String sectionUrl	The URL to this section page.
Int pageNum	Page number of this section page.
Int pageStart	The start page of the navigation of this section.
Int pageEnd	The end page of the navigation of this section.
Int pageTotal	The total number of pages for this section.
String sortBy	Display how this section is sorted.
Object sectionFeatureDatas[]	Not implemented.
Object sectionDatas[]	Collection of all items or contents for this section to be displayed in this page. Type of items is ItemInfo Type of contents is ContentInfo
SectionInfo titleSectionInfos[]	Collection of parent sections intended to be used as the bread crumb to the current section page. It is also in the form of SectionInfo and only the following properties are populated. <ul style="list-style-type: none">● sectionNaturalKey● sectionId● sectionShortTitle● sectionTitle● sectionDesc● sectionUrl
SectionInfo childSectionInfos[]	Collection of children sections for this section intended to be used as navigation to the children sections. It is also in the form of SectionInfo and only the following properties are populated. <ul style="list-style-type: none">● sectionNaturalKey● sectionId● sectionShortTitle● sectionTitle● sectionDesc● sectionUrl
Int childCount	Number of childSectionInfos.

Class ShoppingCartSummaryInfo

ItemInfo cartItems[]	Collection of items in the shopping cart.
String itemCount	Number of items in the shopping cart.
String priceSubTotal	Price total of all items in the shopping cart (excluding taxes and shipping charge).

Class SyndicationInfo

String link	URL to the external article.
String title	Title of the article.
String description	Description of the article.
String publishDate	Publish date.
String updatedDate	Updated date.

Class DefaultTemplateEngine

String getHorizontalMenu(String menuSetName, String styleClassSuffix)
Generate HTML and corresponding JavaScript code for horizontal menu.

Parameters

menuSetName – Name of the menu set to be used to generate this menu.

StyleClassSuffix – Menu code generated will be using using yuimenu-sty;eClassSuffix and yuimenuitemlabel-styleClassSuffix as the CSS style. This is to enable menus to be able to branded differently. Example, if styleClassSuffix = sample, the CSS styles used will be yuimenu-sample and yuimenuitem-sample.

String getVerticalMenu(String menuSetName, String styleClassSuffix)
Generate HTML and corresponding JavaScript code for vertical menu.

Parameters

menuSetName – Name of the menu set to be used to generate this menu.

StyleClassSuffix – Menu code generated will be using using yuimenu-sty;eClassSuffix and yuimenuitemlabel-styleClassSuffix as the CSS style. This is to enable menus to be able to branded differently. Example, if styleClassSuffix = sample, the CSS styles used will be yuimenu-sample and yuimenuitem-sample.

String getPoll()
Generate HTML block for the current poll.

String getResourcePrefix(String resource)
Return the URL of the resource. If a user defined template is assigned to the site, and the resource exists in the user defined template directory, URL referencing the resource in the user defined template is returned. Otherwise, URL referencing the resource in the default, 'basic', template is returned.

String getSyndication()
Generate HTML block for syndication.

String getShoppingCartSummary()
Generate HTML block for shopping cart summary information.

boolean isShoppingCart()
Returns if checkout process is enabled.

boolean isPoll()
Returns if there is active poll.

boolean isSyndication()
Returns if there is syndication currently defined.

String encodeURL()

Translates a string into application/x-www-form-urlencoded format.