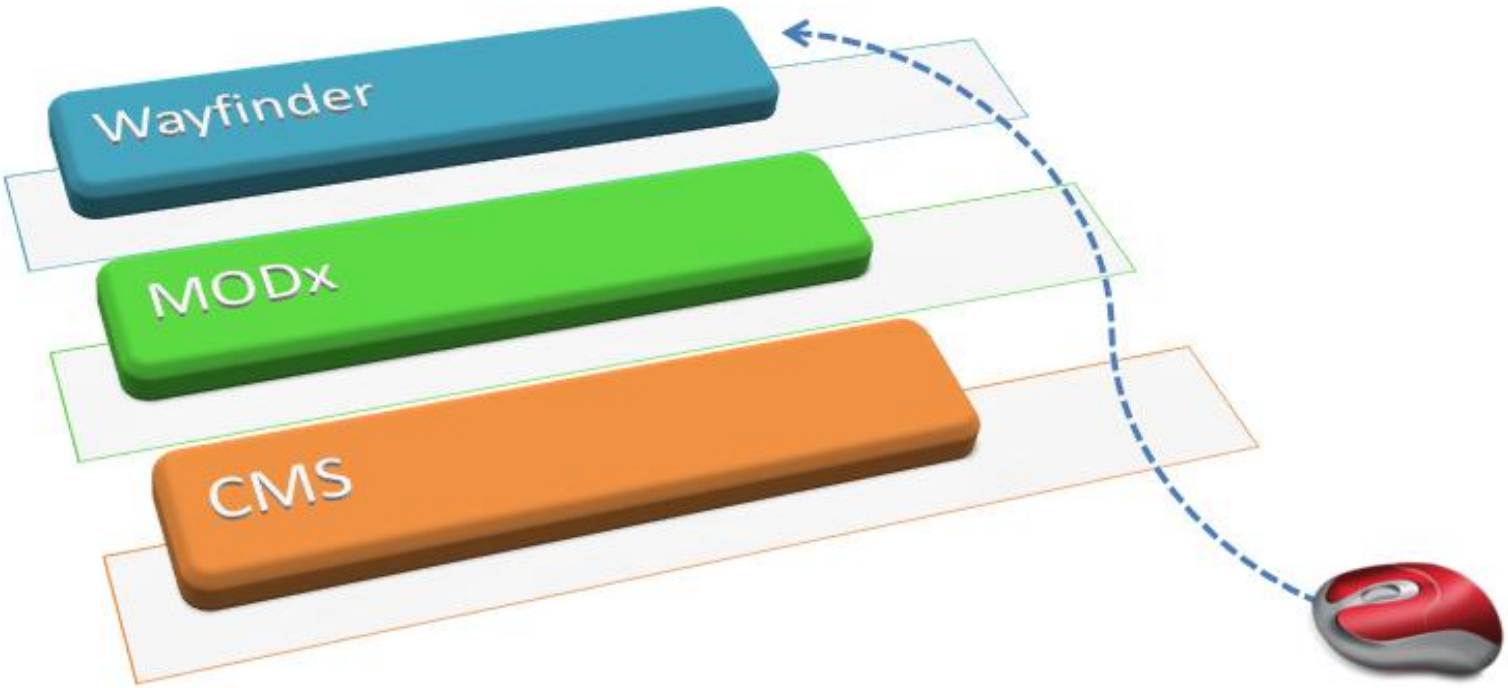
The almost Complete Guide to Creating Menus in MODx using Wayfinder



A guide by Kongondo

First Edition
March 2009

The (almost) Complete Guide to Creating Menus in MODx Using Wayfinder

Dedication

Dedicated to the MODx community – probably the best online community in the world...

Acknowledgements

Kyle Jaebker: For creating the clever snippet that Wayfinder is. Thank you so much.

Bill Fernandez: Thank you for the "Wayfinder 2.0 documentation". This guide is inspired by the work you did.

MODx developers: Thank you for creating the best CMS in the world. You guys are simply genius.

To the MODx community/forums: This guide would not have been possible without your input, questions and solutions alike.

Developers of Blueprint CSS: The fictitious website used in this guide is 99% styled using Blueprint CSS. Thanks.

Iconspedia: Thank you for the icons.

Images: All the "food" images used in this guide are copyright of their owners over at www.flickr.com. Thank you for sharing your work.

Others: To all the designers and developers whose works have been used and credited in this guide - thank you.

Contents

Chapter 1: getting started

Introduction, requirements & things to know

Hi there! Welcome to "The (almost) complete guide to creating menus in MODx using Wayfinder".

Wayfinder is a MODx snippet that easily creates highly dynamic and flexible navigational menus using unordered lists. You can even use Wayfinder to output tables, divs, definition lists, ordered lists, etc. In this guide, however, we will only be concerned with unordered lists.

Before we start there are a couple of requirements and instructions.

- You will need at least a basic/working knowledge of XHTML, CSS and MODx.
- To follow the lessons, you will need to work on either a test or local server. Please don't work on a live site! I will be using a local installation of MODx. If you don't know how to install MODx please visit the MODx website: http://modxcms.com
- I would recommend installing MODx with Wayfinder BUT without the sample content (you can tick this when installing MODx) on a local server such as WAMP, XAMPP, MAMP or similar. I am working on a Windows XP professional environment with MODx version 0.9.6.3 installed on XAMPP. This guide utilizes Wayfinder 2.0. If you didn't install Wayfinder when you installed MODx follow the following instructions to install it

    1. Download Wayfinder from either the MODx repository

    http://modxcms.com/extras.html?view=package/view&repository=10&package=487

    OR from the developer's website

    http://www.muddydogpaws.com/development/wayfinder/download.html

    2. Create a folder under your snippets directory named wayfinder.
    3. Copy the file wayfinder.inc.php into that directory
    4. Create a new snippet named Wayfinder
    5. Copy the contents of snippet.wayfinder.tpl.php into the snippet

- We'll be creating our own content from scratch. However, you will need to have installed a template for the tutorial. A simple two column XHTML template with a header, navigation/menu and footer sections will do just fine. I will not be getting into the details of how to install a MODx template so you will need to read up on this if you don't know how to do it.

- You will most likely need a text editor such as Notepad++, etc. This will help if you will be copying and pasting text from the examples provided. Please don't use Word or similar word processors for this. They might introduce unwanted code in your MODx code.

- The examples shown in this guide are meant to show you how someone can create dynamic menus with Wayfinder and are not meant to be in depth tutorials on how to make CSS menus. There are excellent tutorials out there for this *(Google CSS menus)*. So, although we will be applying styles to our menus, I will not go into detail why this or that menu is not displaying properly in a particular browser since those are browser issues and not Wayfinder ones.

OK, let's get started already!

Chapter 2: Lesson #1 – understanding the basics

Lesson #1 at a glance
&startId **[+wf.classes+]** **[+wf.classnames+]** **[+wf.wrapper+]** **&outerClass** **&outerTpl**

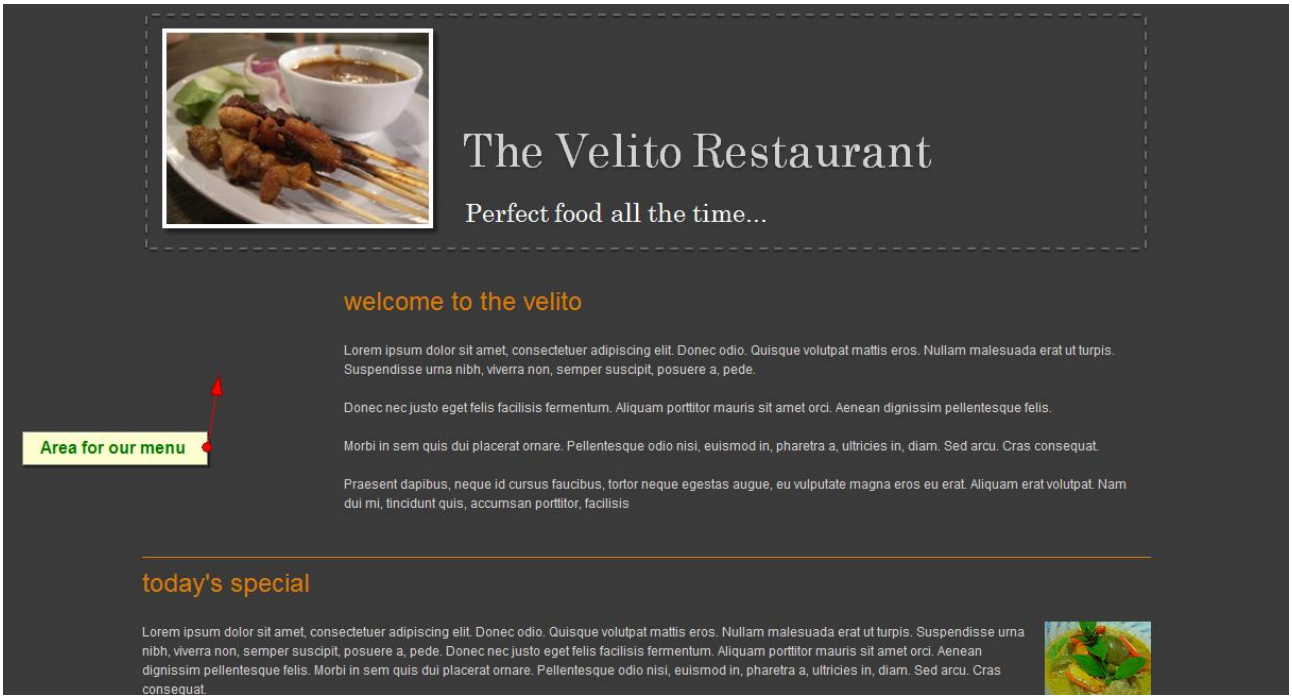
Welcome to lesson #1 of this Wayfinder guide. Before proceeding with this lesson, you should have read the introduction (chapter 1) to get a feel of MODx and Wayfinder. If you have not done so already, read that chapter first.

In this lesson I will be introducing Wayfinder concepts and parameters in general and lay the groundwork for all the lessons in this guide. It is very important for you to read and understand this chapter before you proceed to the other lessons. We will specifically look at a few key Wayfinder parameters and finish by creating a simple menu using Wayfinder.

For most of the lessons, we will be using a fictitious restaurant website to practice using Wayfinder. You are, however, not restricted to using my fictitious restaurant's website and may choose to apply what you are learning here in a different setting.

The Task

The Velito Restaurant has contracted us to build a website for them. We have chosen to use MODx for this job since it is highly flexible, powerful and easy for the end-user to edit and update. We have gone through the design process, created a template and installed it in MODx. We are now dealing with the menu for the site (the navigation menu, i.e., not the restaurant menu!) and that is where Wayfinder comes in. In the introduction section I asked you to ensure you have a simple template installed in your MODx site for the purposes of this guide. Here is how my Velito Restaurant website looks like without a navigation menu.



The Velito restaurant website template

Before we start creating the menu, there are a few things we need to know first. Wayfinder creates an unordered list of either all or a portion of your MODx document tree structure according to how you instruct it. In order for Wayfinder to do this, it uses a number of set parameters. By assigning certain values to these parameters, you can use Wayfinder to create an almost limitless variety of menus.

Wayfinder parameters CAN BE GROUPED into the following categories.

1. Parameters that determine the documents to list, how they are listed and their attributes
2. CSS class names parameters
3. Placeholder parameters
4. Template parameters
5. Utility parameters

For each of the categories above, Wayfinder has several parameters. Do not worry about what these parameters mean or do at this stage. I will introduce each of them gradually, over several lessons in this guide. This will allow you to learn what each of these parameters do and the logic behind how they work at a pace which, hopefully, is not overwhelming.

Let us first look at each of the parameter categories in turn to see what they do and afterwards we will look at how and when to use them.

Parameters that determine the documents to list, how they are listed and their attributes

The parameters in this category tell Wayfinder which documents in the MODx document tree it should include in the menu(s)/list items you tell it to create, how those documents should be shown, for instance, how they should be arranged (e.g. alphabetically) and the criteria to use for that arrangement. They also tell Wayfinder the text to use for the menu titles, i.e. the clickable text of your menu (e.g. "Home", "About", etc).

CSS class names parameters

In XHTML, one can assign class names to tags such as ** **, **<ul class="navigation">**, **<li class="submenu" >**, etc. In Wayfinder to assign such class names to your XHTML tags, you use class names parameters. You are not limited to only using these class name parameters though. You can use your own custom classes alongside Wayfinder ones.

Wayfinder has several pre-defined class parameters (that can be used in different areas of the menu structure) that a user has to assign values to. In the examples given above, in the context of Wayfinder, "**logo**", "**navigation**" and "**submenu**" would all be values declared by the user and could be assigned to the pre-defined Wayfinder class parameters. This is important to remember. Wayfinder will not force you to use certain class names for your menus. Using these classes as selectors, you can then use CSS to style your menus or use a JavaScript library like jQuery to do some really fancy stuff with your menus.

Placeholder parameters

Wayfinder has several pre-defined placeholder parameters. Using these parameters, you tell Wayfinder where to place certain "portions" of the menu. For instance, you can tell Wayfinder where you want the class parameters mentioned above to appear within the menu. You also use placeholders to tell Wayfinder where to place the links of your menus (the URL your menus are pointing to) and the texts for your menu titles. Do not worry if this is a bit confusing at the moment; it will become clear very soon.

Template parameters

Template parameters are very important in Wayfinder. They tell Wayfinder how the structure of your menu should be, i.e. how Wayfinder should build the menu. Do not confuse Wayfinder templates with your normal website template! The templates referred to here are simple code, for instance XHTML code, which are placed in what MODx refers to as chunks. Chunks are reusable pieces of XHTML. Please refer to the MODx documentation for more information. *(Note that in MODx such templates are sometimes referred to as tpl; this is just shorthand for templates).*

Once you have created your template chunks, you then tell Wayfinder which template chunks to use for which part of the menu structure. These will become clearer once we create a menu using Wayfinder. At this stage, do not worry how this code looks like and how it is created. We will be doing that in a few minutes!

Utility parameters

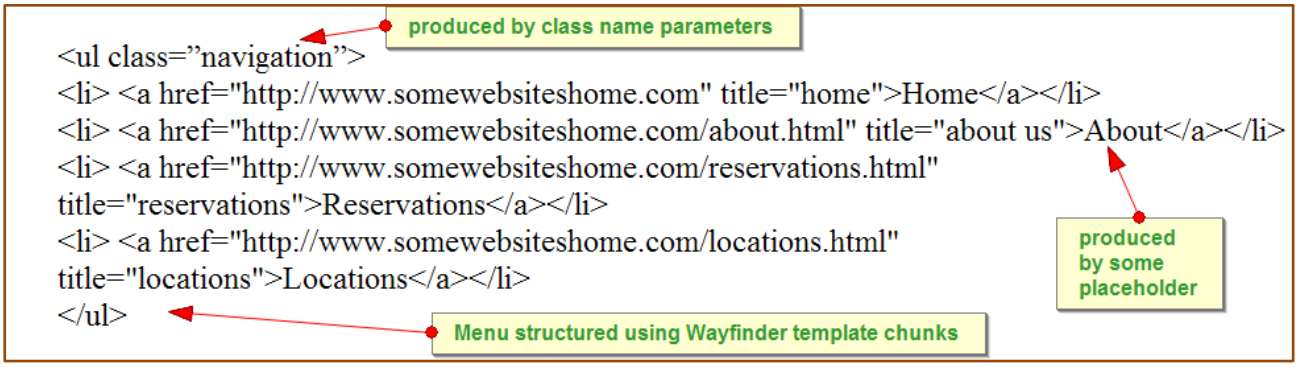
The final type of Wayfinder parameters are utility parameters. There is only a few of these and they help in such things as debugging (troubleshooting) errors you may have in your menu. These will be covered in more advanced lessons of this guide.

To assist you better understand what Wayfinder parameters do, let us look at how a basic XHTML/CSS menu looks like. An XHTML/CSS menu is nothing more than styled XHTML list items. Commonly, it is unordered lists that are used for these purposes.

The XHTML code for a menu might resemble something like this:

```
<ul class="navigation">
<li> <a href="http://www.somewebsiteshome.com" title="home">Home</a></li>
<li> <a href="http://www.somewebsiteshome.com/about.html" title="about
us">About</a></li>
<li> <a href="http://www.somewebsiteshome.com/reservations.html"
title="reservations">Reservations</a></li>
<li> <a href="http://www.somewebsiteshome.com/locations.html"
title="locations">Locations</a></li>
</ul>
```

As we can see, this menu is made up of different parts. What Wayfinder does is to dynamically put all these parts of the menu together behind the scenes and present its output on the front end of your website where you want the menu to appear. With Wayfinder, however, we will not have to manually enter each menu item and its links! Wayfinder will do these for us dynamically. Using the same menu structure above, let us now look at how each of the above mentioned parameter categories plays a role in creating this menu using Wayfinder.



A typical menu structure

As we can see from the above, a menu created by Wayfinder is a combination of several parameters. Once you get to understand these parameters, building menus using Wayfinder becomes very simple.

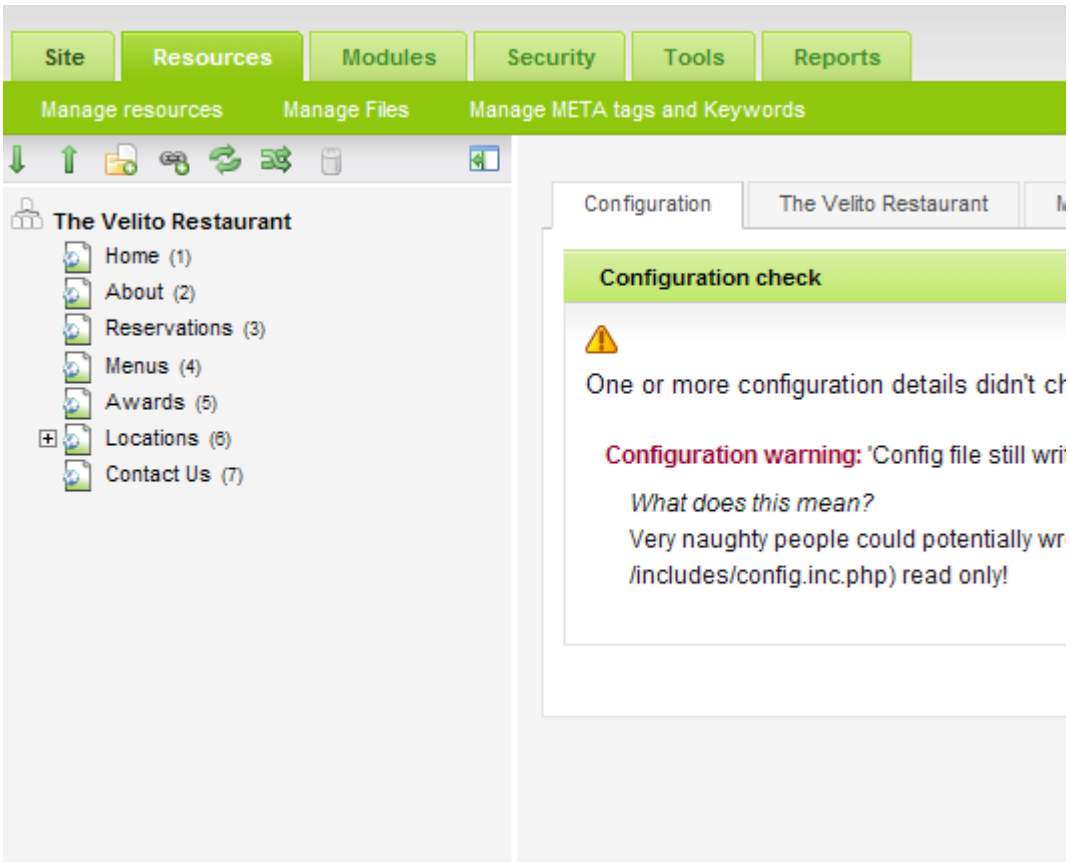
It’s now time to look at the syntax used in Wayfinder, learn a few parameters and create our first menu!

Fire-up your local server if you have not already done so and navigate your browser to your MODx installation that you set up for the purposes of these lessons. Log in to your MODx manager. If the browser you are using supports tabbed browsing, open up the home page (frontend) of your MODx site in another tab so that you can easily view the menu we will be creating as it will appear to the public or open up another instance of your browser in order to preview your finished menu if you do not have tabbed browsing capabilities in your browser.

I will assume you have already configured your MODx install (e.g. defined a site title, etc). For our restaurant website, for starters, we will need to create and publish the following 7 documents in MODx.

1. Home
2. About
3. Reservations
4. Menus
5. Awards
6. Locations
7. Contact

You can create some dummy content for these documents if you wish but this is not important. Make sure that "show in menu" is ticked for all the documents and that you have given them menu titles by filling in their "menu title" fields. Once you are done, your MODx document tree should resemble the following:



Document tree for the Velito restaurant website

The Wayfinder snippet call

Now for the next step we want to create a vertical menu whose items when clicked will load the pages we have created above into our browser. So how do we do create our menu in MODx using Wayfinder? This may be easier than you think. Remember in the introduction section I told you that Wayfinder is a snippet? Well, a snippet is of no use to us if we cannot tell that snippet to do something. So, we are going to do what is referred to as calling our snippet. In our case, we will make a Wayfinder call. Since we want our menu to appear in all our Velito Restaurant's web pages, we will place this snippet call in our web site's template exactly where we want it to appear. In my case, I will place this Wayfinder call in a **div** with the **id="navigation"**. To do this, I accessed the

template in the MODx manager and typed the following text in my template exactly as shown below where I wanted the menu to appear. Please remember that parameter names are case sensitive. Also note carefully the syntax especially the part **`0`.** On either side of the zero (it is a zero not a letter o) are what are called backticks. They are not apostrophe marks or single quotes. On a standard UK or American English keyboard this ` is located on the first key just before the 1-key *(the 1-key on the main keyboard, not the numeric pad).*

```
[[Wayfinder? &startId=`0`]]
```

Save your template. Open a new tab where we can preview your work in the front end. Navigate to our restaurant's website. In my case the path is: **http://localhost/velito/**

Congratulations! You have just created your first Wayfinder menu! That was not too hard, was it? OK, I know the menu does not look very appealing at the moment but that is because we have not applied any styling to it. If you don't see any menu make sure you typed the above code correctly and have also published your documents in addition to ticking their "show in menu" boxes. Before we do any styling, let us first understand the above Wayfinder call.

What we typed above in our template is Wayfinder's minimum snippet call. If you have read about snippets in MODx you should know that snippets can either be called cached or uncached. The above syntax may as well have been written uncached like this:

```
[!Wayfinder? &startId=`0`!]
```

However, we do not need to call this Wayfinder snippet uncached *(there are some cases where we would need to do this. This will be covered later)*. Wayfinder is clever enough to know when new documents that are set to be shown in the menu are created in the MODx manager and will add them to our menu.

Let's decipher this minimum Wayfinder call.

The &startId parameter

The **&startId** is perhaps the most important Wayfinder parameter you need to know about since it is always required in a Wayfinder call. Without it Wayfinder will not know where in the document tree you want your menu to start from. By telling Wayfinder that the value of **&startId** is equal to **`0`** we are instructing it to create an unordered list of all the descendant documents in our document tree. Put another way, to list all the descendant documents of the root. All MODx documents in the tree are descendants of the root. The root is our site. Therefore, in our case the root is "The Velito Restaurant".

Wayfinder will normally create an unordered list of all the descendant documents *(no matter how deeply nested)* of the document declared as the **&startId** as long as these documents are published,

are set to be viewed by the current user and with their "show in menu box" ticked. To put it in other words, Wayfinder will mirror the MODx document tree. This is what will happen if you use the minimum Wayfinder call. However, it is important to remember that Wayfinder has parameters that can be added to the Wayfinder call in order to control the number and level (hierarchy) of documents it outputs. These will be covered in lessons to come.

Another important thing to understand is the term descendant; it means exactly that. It means a child-parent-grand-parent etc, relationship. A child is a descendant of both the parent and the grand-parent. The parent is a descendant of the grand-parent, and so on. This is important so let us explain it further.

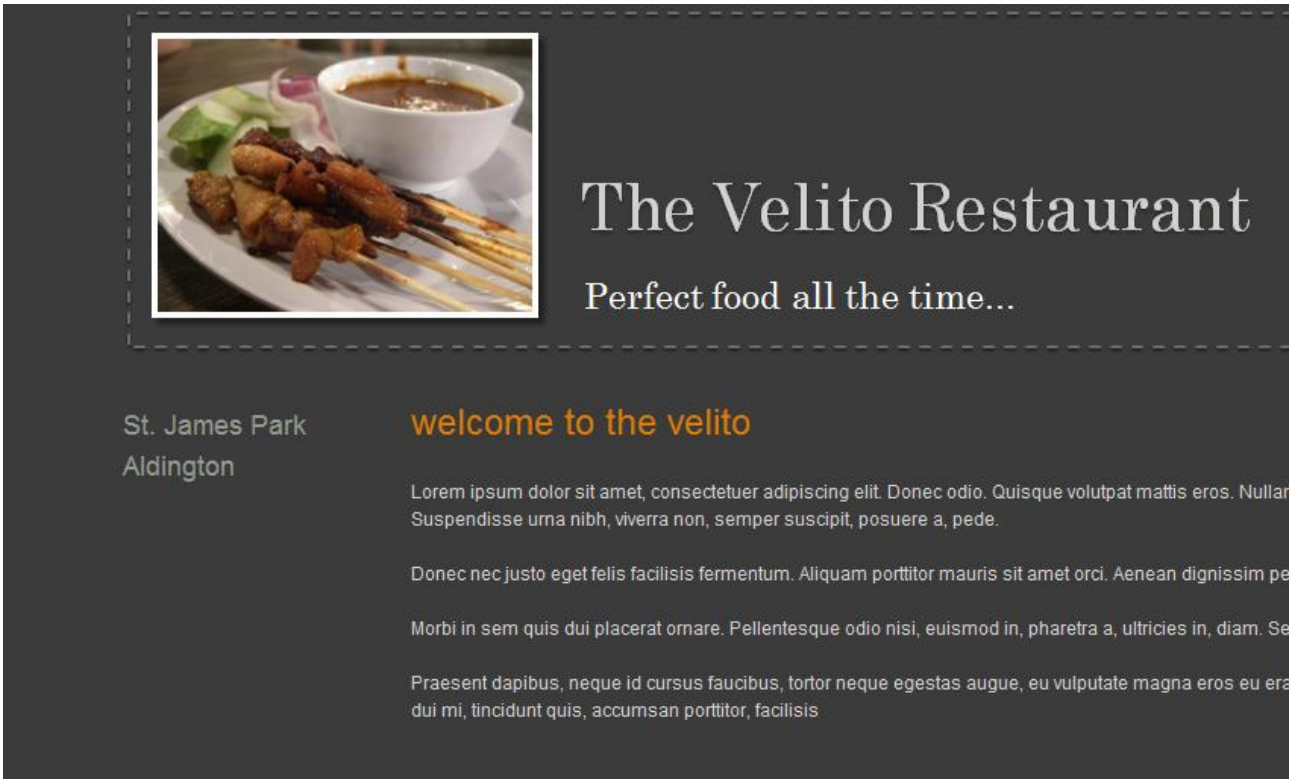
Imagine you have a document tree that looks like the following.

- Home (1)
- About (2)
- Reservations (3)
- Menus (4)
- Awards (5)
- Locations (6)
    - St. James Park (8)
    - Aldington (9)
- Contact (7)

All the above documents are the descendants of the root. The documents "Home", "About", "Reservations", "Menus", "Awards", "Contact", "St. James Park" and "Aldington" DO NOT have any descendants. However, the document "Location" has two descendants; "St. James Park" and "Aldington" *(these two can also be described as the grand-child documents of the root)*. Any of the above documents can have multiple descendants which can continue deep in the hierarchy, i.e. grand-children, great-grandchildren, great-great-grandchildren, etc.

As we will later find out, under certain conditions, we can override Wayfinder's default behaviour of listing the descendants of the **&startId** document and instead instruct Wayfinder to create a menu that starts with the document whose **id** is assigned to the **&startId** parameter *(i.e. include the **&startId** document in the output)*.

The value of **&startId** does not need to be **0**. Your menu can start from anywhere on the tree. Just remember Wayfinder will output a menu starting from the descendant document of the **&startId.** For instance, using our example document tree above if we had entered the value of **&startId** as **&startId=`1`** we would get not output. Why? This is because the document with the **id=1** is "home" and it has no descendants. On the other hand, if we had declared **&startId=`6`** *(i.e. the document called "Location")* Wayfinder would output a menu with the two documents "St. James Park" and "Aldington" as the menu items as follows:



The Velito restaurant website showing menu generated from &startId=`6`

Just to emphasize the point about **&startId** not being part of the menu itself, in the above example where **&startId=`6`,** the document "Location" (**id=6**) is NOT PART of the menu; only its descendants are.

This ability of Wayfinder to output a menu starting from anywhere on the document tree offers great flexibility in creating websites with multiple menus, ranging from simple to very complex menus.

Placeholders, classes and templates

A Wayfinder placeholder parameter looks something like this

```
[+wf.placeholder_name+]
```

A placeholder does exactly what the term placeholder suggests. It temporarily "holds" the place for a particular element or entity or variable. Think of it like someone holding a place for you in a queue. When you turn up, you take your place in the queue, replacing whoever was holding your place in the queue.

Wayfinder placeholders are always placed in Wayfinder templates (template chunks). In themselves, Wayfinder placeholders have no values. Instead, they get their values from other Wayfinder and MODx parameters and MODx values.

Creating a Wayfinder template

Let us now create a Wayfinder template. Create a new chunk and give it the name **menuContainer.** You can name it what you want *(as longs as it is a permissible name for a chunk in MODx).* Save the chunk but keep it open *(click on the radio button "continue editing")* because we will be adding content to it shortly.

As mentioned above, a Wayfinder template is just a short piece of code placed in MODx chunks. One typical but incomplete Wayfinder template chunk could look like this:

```
<ul>

</ul>
```

Type the above code in the chunk we have just created.

Does the code look familiar? It should. These are just opening and closing tags of an unordered list. This code is what the Wayfinder template chunk would contain. However, the above Wayfinder template is incomplete. This is because there are certain important pieces missing in our template.

First, we want to make sure that Wayfinder will attach a class or classes to our unordered list so that we can style it using CSS. We need something to reserve a place, so to speak, for the classes we will assign our unordered list **<ul>** tags (or our unordered list container tags). This looks like the job of a placeholder! We therefore need to add this placeholder where in a normal unordered XHTML list we would place the **class="".** Do you recall our unordered list above with the **<ul class="navigation">**? Our placeholder will stand in for the part **class="navigation".** The placeholder we are going to use is one of Wayfinder's pre-defined ones, the **[+wf.classes+]** placeholder.

The [+wf.classes+] placeholder parameter

All classes you assign your menu items will be inserted where **[+wf.classes+]** appears. As we will soon find out, Wayfinder will assign these classes depending on the properties of the menu item the class(es) are being assigned to. By property here I mean: is the menu item in the main menu or a sub-menu? Is the document currently active, i.e. being browsed)? etc. I will touch more on this later on. The **[+wf.classes+]** will include in its output **class=""** with the classes correctly inserted between the **""**.

The next step is to add this placeholder parameter **[+wf.classes+]** in our chunk. Type **[+wf.classes+]** within the opening **<ul>** tags as shown below. Make sure you have typed it exactly as shown here. Save the chunk but keep it open because we are not done yet.

```
<ul[+wf.classes+]></ul>
```

We need at least one more Wayfinder parameter, another placeholder.

Think of this: Wayfinder has to insert the list items themselves, i.e. the **
** tags, between the **
** tags in order to create a valid XHTML unordered list. We can think of these **
** tags

(and their content e.g. "Home") as the rows of the unordered list or rows that make up the menu. In order to tell Wayfinder where to place these rows, we will use the pre-defined Wayfinder placeholder **[+wf.wrapper+].** Before we look at the placeholder, let us look at a placeholder related to the **[wf.classes+]**. This is the **[+wf.classnames+]**.

The [+wf.classnames+] placeholder parameter

There might be times when alongside Wayfinder classes, you want to add your own custom classes to a particular template/XHTML element. In such a case you would not want Wayfinder to output **class=""** because Wayfinder would not output your custom classes in between the double quotes. The solution here would be for you to type in the **class=""** yourself and let Wayfinder only output the class names themselves alongside which you can add your own. In such a case, you would need the parameter **[+wf.classnames+]**. This placeholder will cause Wayfinder to only output the class names without the **class=""**. This would enable us to do something like the following:

```
<ul class="[+wf.classnames+] mycustomclass"></ul>
```

The above code will output the relevant Wayfinder classes alongside the class "**mycustomclass**". If we had used **[+wf.classes+]** instead, we would not have been able to achieve this. OK, let us now move to the **[+wf.wrapper+]**.

The [+wf.wrapper+] placeholder

This is a very important Wayfinder placeholder and is always required in a Wayfinder template. The inner contents of the menu (the row items) are inserted where the **[+wf.wrapper+]** appears in the Wayfinder template. We will explore this further in lesson #2.

Continuing with our template, now type **[+wf.wrapper+]** between the **
** tags exactly as shown below.

Our Wayfinder template now looks like this:

```
<ul[+wf.classes+]>[+wf.wrapper+]</ul>
```

The &outerTpl template parameter

Notice that we have placed the **[+wf.wrapper+]** placeholder in between **
** tags since that is where **
** need to be placed in a valid unordered XHTML list. The template we have just created above is so important for Wayfinder to create menus properly that it has an exact, default, in-built one just in case you forget to create this template. This is why Wayfinder was able to create our first menu above; it used its in-built template to structure the menu we asked it to create. In Wayfinder, this template is a required parameter and is referred to as the **&outerTpl**.
The **&outerTpl** forms the outermost part of the menu structure, i.e. it is the outermost container for the rows that constitute the menu.

Please note that the **&outerTpl** you define does not have to be exactly as shown above. For instance, your **&outerTpl** can contain other XHTML tags such as a **div** to wrap around your menus. For instance, it may look like the following

! As we will see in a later lesson, you will need to be cautious when you define your **&outerTpl** in order to get the correct XHTML output

```
<div id="nav">
<ul [+wf.classes+]>[+wf.wrapper+]</ul>
</div>
```

The thing to remember here is that you must define an **&outerTpl** otherwise Wayfinder will use its default **&outerTpl**. In addition, you can see from the above example that we have given the ** ** wrapping our menu an **id="nav".** We can obviously do this directly in the website's template instead of including this ** ** in our Wayfinder template.

We are almost done with setting up the foundation for our menu. However, there is still something missing. Can you notice what it is? So far we have not actually told Wayfinder the name or names of our classes (in our example the **ul class="navigation"**). We have also not told Wayfinder the name of the chunk that contains our **&outerTpl**. In other words, we have not made a connection between the Wayfinder call and the template. A MODx website will normally contain several chunks. We need to tell Wayfinder very specifically which chunk contains what template. In this present case, we need to tell Wayfinder which chunk contains the **&outerTpl**. To do this we need to add some parameters to our Wayfinder call. We need to tell Wayfinder the name of the chunk containing our **&outerTpl**. Recall that I named the chunk containing my **&outerTpl menuContainer**. You might have named yours differently. We will be giving Wayfinder this information in a minute. Here again is the **&outerTpl** we will be using.

```
<ul[+wf.classes+]>[+wf.wrapper+]</ul>
```

The &outerClass class name parameter

We also have to tell Wayfinder the class name we want to be assigned to our **&outerTpl.** From our example menu above, the class we gave to the **<ul>** which is the container of our menu was **"navigation".** This class **"navigation"** will be the value of a parameter. The parameter we use in this case is **&outerClass**. Wayfinder will use the value you give **&outerClass** as the class for **&outerTpl**. This is important; whatever value (i.e. string) you give as the value of the parameter **&outerClass** WILL ONLY BE ASSIGNED to **&outerTpl** and not any other Wayfinder template.

Putting it all together

With the above information, we can now modify our original Wayfinder call. Before you become very comfortable with using Wayfinder, it is a good idea not to just copy and paste code. Typing the code yourself will help you remember the parameters and Wayfinder's snippet call syntax.

Go to your website's Velito Restaurant template and delete the Wayfinder call we originally had in there for our first menu. Then, type the following code where you have just deleted the Wayfinder call and save the template.

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`]]
```

Notice that each of the values for the Wayfinder parameters in the above call is enclosed in backticks. Also note that all our parameters and their associated values have been declared in one Wayfinder call. Because we are only creating one menu, we DO NOT need to make several Wayfinder calls. In later lessons we will see that we can make more than one Wayfinder call to create multiple menus.

What the above call does is to tell Wayfinder to output all documents in our MODx tree and place these documents in an unordered list created partly using the template **&outerTpl** which is located in a chunk called **menuContainer** and assign the class **"navigation"** to the top most **<ul>** tag in that output list.

It is important to note that if we declare a certain parameter in our Wayfinder call and DO NOT give that parameter a value, i.e. we do not write anything between the `` , Wayfinder will assume that we want the value of that parameter to be blank. For instance, look at the following call:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=``]]
```

The parameter **&outerClass** has not been given a value. In the resultant menu, this will appear as follows:

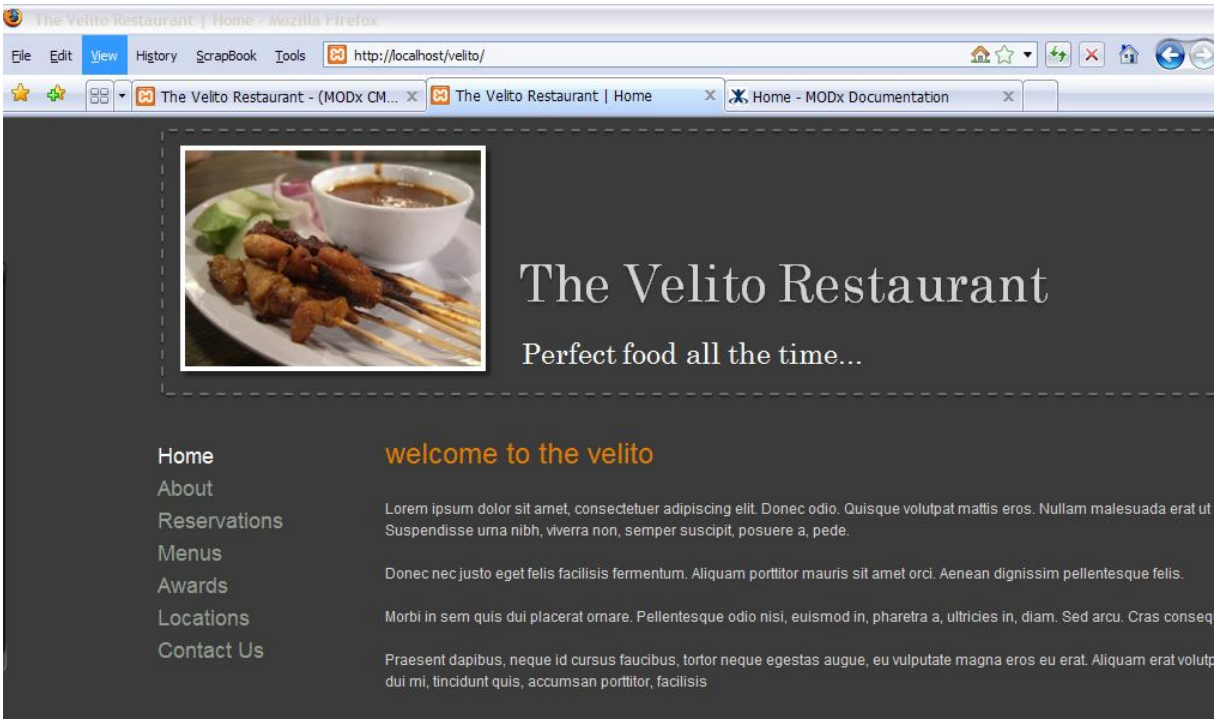
```
<ul class="">
```

Styling the menu

Back to our menu, to style it, we obviously need to apply some CSS rules that will target this list. We can do this by applying some rules to the **<ul>** with the class **"navigation"**. We can go further and target the **<li>** and the **<a>** within our menu as we wish. I will leave this up to you.

Show me the menu!

Let us now preview our menu. This is how it looks like when viewed in Firefox 3. It looks fine and it validates. Did yours come out fine? If not, do some troubleshooting - check your code, did you use backticks? Did you use square brackets correctly? Did you use the correct spellings (case sensitive), etc?



The Velito restaurant with menu generated from &startId=`0`

OK, let us bring this lesson to a close. It has been a long one but we have covered some very important Wayfinder concepts. Before we finish though, let us sum up what we have learnt in this lesson.

Summary of lesson #1

In this lesson we learnt the following:

1. We learnt about Wayfinder parameter categories

2. We looked at the minimum Wayfinder call, how it works and 6 Wayfinder parameters (**&startId**, **[+wf.classes+]**, **[+wf.classnames+]**, **[+wf.wrapper+]**, **&outerTpl** and **&outerClass**)

3. Using the knowledge above, we created a simple menu using Wayfinder

Here's the step by step process we followed to create our vertical menu:

1. We typed the following Wayfinder snippet call in our website's template and saved the template.

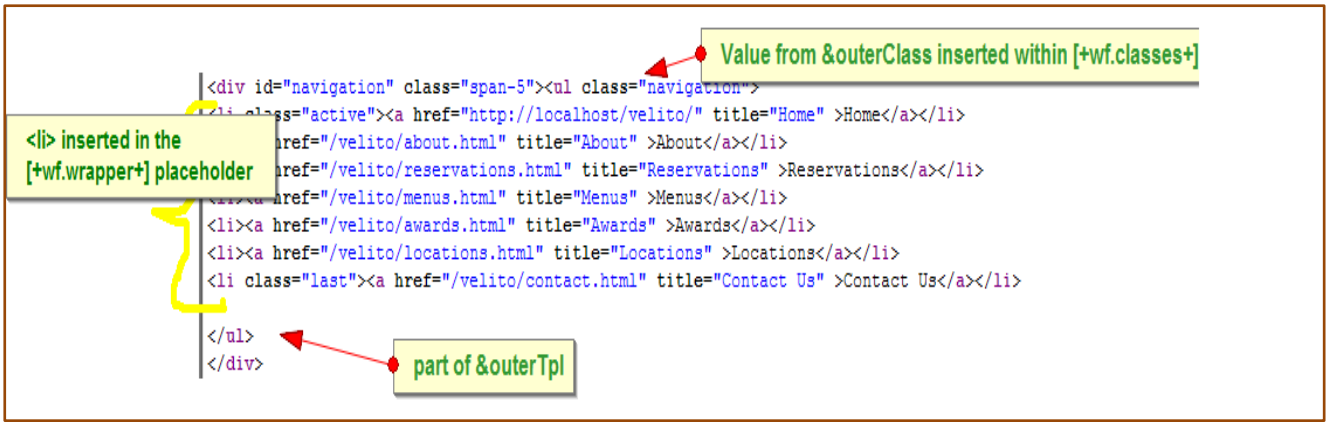
```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`]]
```

2. We created a Wayfinder template chunk, named it **menuContainer**, typed the following code in it and saved it

```
<ul[+wf.classes+]>[+wf.wrapper+]</ul>
```

3. In our CSS file we applied some styling to the **<ul>** with the class **"navigation"** and its descendant elements, i.e. the **<li>**.

The following image shows the source code of our menu and the Wayfinder parameters that have been substituted for the menu content. There are several ways to view the source code depending on which browser you are using so I will not get into details about this. I have friendly URLs turned on in my MODx install so my source code may be somewhat different from yours.



Before we close, we need to note a couple of things:

Wayfinder dynamically created each menu item and its link. Have a look at the source code of the menu. The code reveals a number of "values" that Wayfinder seems to have added to our menu items. For instance, the **<a href="URL">** was dynamically generated by Wayfinder. What about the names of our menu items, e.g. "Home", "About", "Reservations", etc? We did not really give Wayfinder that info or did we? Where did it get that info from? Come to think of it, we did not even tell Wayfinder where we want the links of our menu items to appear. So how did Wayfinder know these things? In the next lesson we will provide answers to these questions in addition to looking at other Wayfinder parameters.

That is it for this lesson folks. See you in the next lesson!

TIP
Placeholders are always placed within the contents of a Wayfinder template
All the other parameters are always declared/referenced in the Wayfinder call

Chapter 3: Lesson #2 – rows & links

Lesson #2 at a glance
&rowTpl **&rowClass** **&textOfLinks** **[+wf.link+]** **[+wf.linktext+]** **&titleOfLinks** **[+wf.title+]**

Welcome to lesson #2 of this Wayfinder guide. A requirement for this lesson is that you should have read lesson #1 and the introduction section. If you have not done so you need to do this first in order for you to effectively follow what we will be addressing in this lesson.

We will continue from where we left off in lesson #1. Remember at the end of that lesson we looked at the source code of our menu and found out that Wayfinder had put in some "additional" stuff there which led to us to ask where it got that information from? Well, I will be answering those questions in this lesson. I will also be introducing a couple of other Wayfinder parameters so let us get started.

Start your local server and log in to your MODx manager if you have not done so already. We will continue with our Velito Restaurant website.

We will start by looking at the source code of the menu we built in lesson #1. Here's how it looks like:

```
<ul class="navigation">
<li class="active"><a href="http://localhost/velito/" title="Home" >Home</a></li>
<li><a href="/velito/about.html" title="About" >About</a></li>
<li><a href="/velito/reservations.html" title="Reservations" >Reservations</a></li>
<li><a href="/velito/menus.html" title="Menus" >Menus</a></li>
<li><a href="/velito/awards.html" title="Awards" >Awards</a></li>
<li><a href="/velito/locations.html" title="Locations" >Locations</a></li>
<li class="last"><a href="/velito/contact.html" title="Contact Us" >Contact Us</a></li>
</ul>
```

Some of the things in the source code we are already familiar with. For instance, we know where the **<ul class="navigation">** comes from. We covered this in lesson #1. Wayfinder is replacing the placeholder **[+wf.classes+]** with the value of the relevant parameter. Specifically, Wayfinder replaced **[+wf.classes+]** with the value of the class for **&outerTpl.** Remember that the class parameter for **&outerTpl** is **&outerClass** and the value we gave our **&outerClass** parameter is **navigation** hence the **<ul class="navigation">**. If all these do not make sense you probably need to review lesson #1.

Now, let us look at the things we have not talked about in the above source code.

The [+wf.link+] placeholder parameter

We see from the source code that Wayfinder has put values in the **href** part of the menu links for each row/list item. For instance, in the case of the document "Locations", Wayfinder has put **"/velito/locations.html"** as the **href** value. The link is correct and is pointing to the URL of the document "Locations" *(your URL might be different depending on whether you are using friendly URLs or not)*. This brings us to the first parameter we will learn in this lesson - the **[+wf.link+]**.

The "links or URL of the menu items" are placed in the **[+wf.link+]** placeholder. In other words, the URL of the document corresponding to the menu item is placed in this placeholder. We notice that Wayfinder has correctly placed the link within the opening **<a>** tags. From lesson #1, we learnt that a placeholder stands in for some other parameter or as in this case, some other value. Therefore, in a Wayfinder template, [**+wf.link+**] stands in for the value of the href in a link as follows:

```
<li><a href="[+wf.link+]" > </a></li>
```

Using our example of the document "Locations", for this particular menu item Wayfinder grabbed the URL of the document and inserted it where the **[+wf.link+]** placeholder was located in the Wayfinder template code. We will be looking at this particular template further along this lesson.

The [+wf.title+] placeholder parameter

Still looking at the source code, coming next after the **href** attribute of the menu links in each row/list item, we see Wayfinder has added a link title attribute and given it a value. In Wayfinder templates the value of the link title attribute is placed in the **[+wf.title+]** placeholder. The following code shows where this placeholder will be substituted for the value of the link title attribute:

```
<li><a title="[+wf.title+]" > </a></li>
```

Now, let us talk about the value that will eventually substitute the [**+wf.title+**] placeholder when the menu is processed and output. Going by our definition of a placeholder, it follows that **[+wf.title+]** is getting its value from somewhere else. This placeholder receives its value from the value of the Wayfinder parameter **&titleOfLinks.**

The &titleOfLinks parameter

The value of the **&titleOfLinks** parameter determines what the title attribute of your menu links will be.

In the source code, looking at the title attribute of the link for the document "Home", we see that it has the value "Home". In the backend, in your MODx manager, if you check you will see that we stated that the "page title" for this document is "Home". That is the same value Wayfinder is using as the value for the title attribute for the menu link for this document. The same is true for the title attribute values of the other row/list items in our menu.

To take it a step further, what happens is that when creating a document in MODx, what you enter as your page title is stored in your website's MODx database **pagetitle** field. So, in essence, by default Wayfinder grabs the value of the **pagetitle** field in the database and uses it as the value for the title attribute of your menu link. Because Wayfinder does this by default, YOU DO NOT NEED to declare **&titleOfLinks** in your Wayfinder call if you want Wayfinder to use the value of **pagetitle** as the value of your link title attribute

! A Please note that the **[+wf.title+]** placeholder will still have to be in your relevant Wayfinder template otherwise the link title attribute value will not be output!

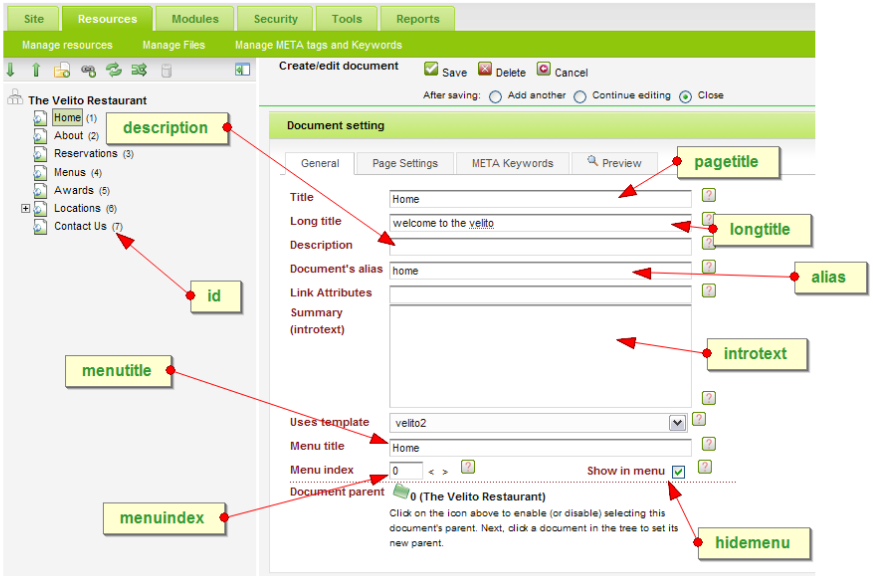
However, IF YOU DO NOT want the **pagetitle** to be used as the title attribute value of your menu links, by using the **&titleOfLinks** parameter, you can instruct Wayfinder to use some other value. Acceptable values are any of the following MODx document variables and attributes.

id | menutitle | pagetitle | introtext | menuindex | published | hidemenu | parent | isfolder | description | alias | longtitle | type | template

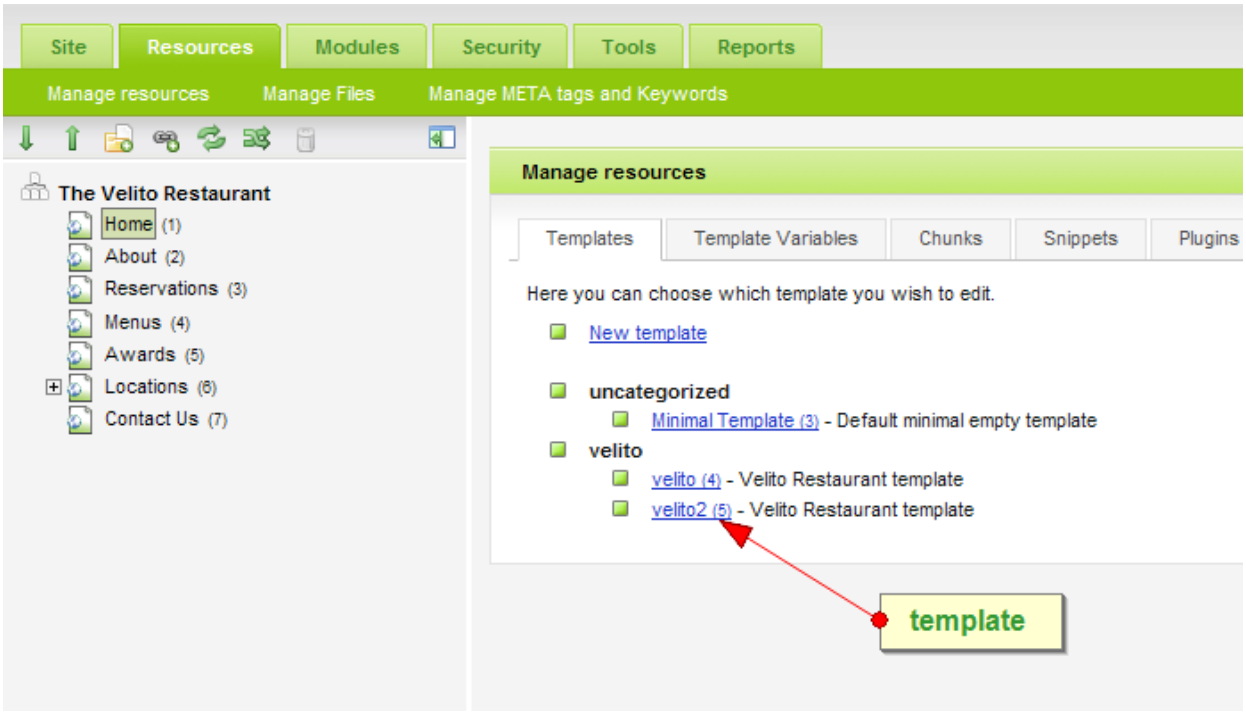
As an example, using our Wayfinder call from lesson #1, if we wanted to use the "long title" of our document as the value for the title attribute of our menu links, we would declare this parameter in our Wayfinder call as follows.

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&titleOfLinks=`longtitle`]]
```

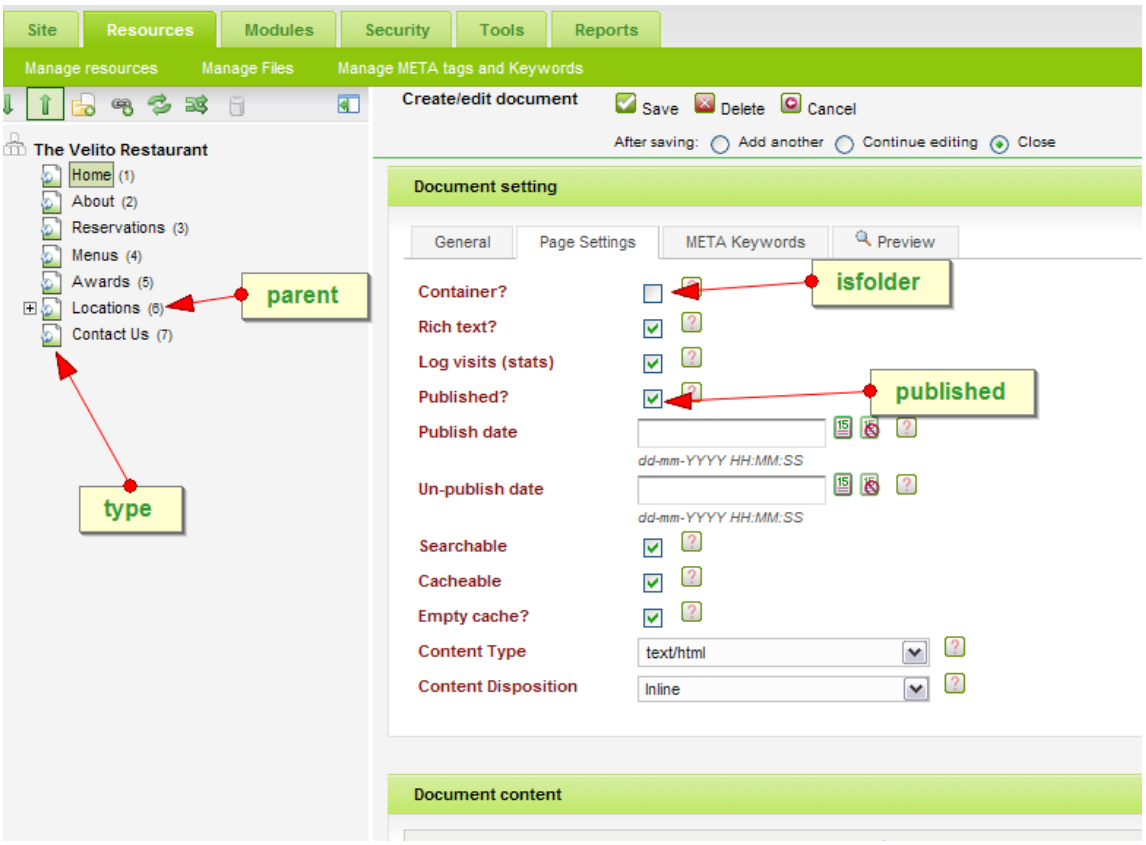
The following images show where the values that you can assign your **&titleOfLinks** parameter are set or appear in your MODx document or site (see the table below for an explanation of the output Wayfinder returns for each of the variables/attributes).



Values that can be assigned to the &titleOfLinks



More values that can be assigned to the &titleOfLinks



Even more values that can be assigned to the &titleOfLinks

In a moment I will show you how the title attributes of our links would look like if we were to use other values other than the **pagetitle**. Before this, let us look at another Wayfinder parameter.

The [+wf.linktext+] placeholder parameter

Still looking at the source code (and also the output menu), we see that Wayfinder has given each of our menu items some clickable text that corresponds to the document being referenced in the menu. So, our document "Home" appears in the menu as "Home"; the document "About" appears in the menu as "About", and so on and so forth. In other words, Wayfinder has correctly created the row items of our menu. We also see from the code that Wayfinder has correctly placed each of these row items between link tags, i.e. between **<a> ** tags which in turn are placed within the list tags **
**. Putting these together, and using the document "Home" as an example, they look like this.

```
<li><a>Home</a></li>
```

This brings us to the next Wayfinder placeholder we will learn about, namely the **[+wf.linktext+]**. This parameter determines where the text of our link/menu item will be placed, i.e. the text in the menu that we click as we navigate the site.

The **[+wf.linktext+]** placeholder stands in for what will be output between the **
<a> **, as follows:

```
<li><a> [+wf.linktext+] </a></li>
```

Just like for the other parameters we have looked at, Wayfinder gets the value of **[+wf.linktext+]** from another parameter. It gets this from the value of the **&textOfLinks** parameter.

The &textOfLinks parameter

The value of the **&textOfLinks** parameter determines what the text of your menu items will be. Remember in lesson #1 we gave each of our documents a menu title? This is where Wayfinder is getting this value from. What you enter as your menu title is stored in your website's MODx database **menutitle** field. By default, Wayfinder uses the value stored in the **menutitle** field as the text of your menu. Because Wayfinder does this by default, YOU DO NOT NEED to declare **&textOfLinks** in your Wayfinder call if you want Wayfinder to use the value of **menutitle** as the text of your menu items

! Please note that the **[+wf.linktext+]** placeholder will still have to be in your relevant Wayfinder template otherwise the text of the menu will not be output!

Similar to what we saw above, IF YOU DO NOT want the **menutitle** to be used as the text of your menu items, by using the **&textOfLinks** parameter, you can tell Wayfinder to use some other value. In this case too, the acceptable values are any of the following MODx document variables and attributes.

id | menutitle | pagetitle | introtext | menuindex | published | hidemenu | parent | isfolder | description | alias | longtitle | type | template

Continuing with our now expanded Wayfinder call from lesson #1, if we wanted, for example, to use the **description** field of our document as the text of our menu items, we would declare this parameter in our Wayfinder call as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&titleOfLinks=`longtitle` &textOfLinks=`description`]]
```

Below is an explanation of how a menu would look like if you used the other acceptable values mentioned above as your **&titleOfLinks** and **&textOfLinks** parameter values. Try out the different values and see the different outputs created by Wayfinder.

pagetitle	Page title will be used
id	Returns the document id of the documents output in the menu
introtext	Summary/Introtext text will be used
menuindex	Returns the integer value (if greater than 0) of the menu index field
description	Description text will be used
hidemenu	If a particular document is set to be hidden AND you also declare the parameter **&ignoreHidden=`TRUE`**, then for that document the menu title will be returned as 1 meaning - true. This means that the document is set to be hidden (i.e. its "show in menu" box is un-ticked) but you are overriding this using the **&ignoreHidden** parameter. This parameter will be covered in coming lessons
parent	This returns the id of the parent of the document
isfolder	Returns 1 (i.e. "true") if the document in the menu is a folder. this 1 will be used as the document's menu title. If it is not a folder, it will show it's menu title or page title if the document doesn't have a menu title
published	Similar to above (isfolder) - returns 1 if document is published. Note that if document is not published it is not shown in the menu anyway
alias	The document alias will be used
longtitle	Long title text will be used
type	Will return the name of the type of the document - e.g. if it is a document it will return "document". If it is a weblink it will return "reference"
template	Returns the template number of the document (note: doesn't apply to weblink - a weblink cannot have a template). This is the number shown in () after the template name

Before we move on to a couple more Wayfinder parameters, you need to note the following:

1. If in your MODx document the menu title field is blank (i.e. you have not entered any text there) **AND** you have not declared an alternative value to be used for **&textOfLinks** in your Wayfinder call, then Wayfinder by default will use the **pagetitle** of that document as the value for the **&textOfLinks** for the menu item for that particular document. This means that whatever you gave this document as its page title will be the text of the menu for this particular document. This ensures that your menus will always have visible text of menus.

2. A similar situation will occur if you declare **&textOfLinks** in your Wayfinder call and give it a value **BUT** the field you have given as the value of **&textOfLinks** is blank (i.e. no text entered in

your MODx document for that particular field). In this case Wayfinder will again default to using the **pagetitle** of that document as the value for the **&textOfLinks** for the menu item for that particular document.

So far, looking at the above XHTML code where we have inserted the three Wayfinder placeholders we have looked at in this lesson, a pattern seems to be emerging. Combing the three Wayfinder placeholders in the same XHTML code we now have the following:

```
<li><a href="[+wf.link+]" title="[+wf.title+]"> [+wf.linktext+] </a></li>
```

Is this beginning to look familiar? It should. It is beginning to look like a Wayfinder template, similar in some respects to the one we saw in lesson #1

The &rowTpl template parameter

Similar to the template Wayfinder uses to construct the outermost part of an unordered list/menu, i.e., the **&outerTpl**, Wayfinder also has a default in-built template to construct the row/list items of a menu, i.e. the **
** part of the unordered list. The template Wayfinder uses to construct the row/list items of a menu is known as the **&rowTpl** template and is always required. You need to create a **&rowTpl** template chunk and tell Wayfinder the name of that chunk in your Wayfinder call. If you do not create one, Wayfinder will use its default one. In the menu we constructed in lesson 1#, we did not create a **&rowTpl** template so Wayfinder defaulted to using its in-built one to correctly output our menu. If we are happy for Wayfinder to use its default **&rowTpl** template we obviously do not need to create one.

The default Wayfinder **&rowTpl** template is:

```
<li [+wf.id+] [+wf.classes+] >
<a href="[+wf.link+]" title="[+wf.title+]" [+wf.attributes+]>[+wf.linktext+]</a>
[+wf.wrapper+]
</li>
```

In the above template there are two placeholders we have not tackled so far. These are the **[+wf.id+]** and **[+wf.attributes+]** placeholders which we will cover in later lessons.

Another thing I need to mention is the **[+wf.wrapper+]** placeholder in the above **&rowTpl** code. Remember in lesson #1 I told you that the **[+wf.wrapper+]** that is placed in the **&outerTpl** template code is a very important Wayfinder placeholder and is always required since the inner contents of the menu (the row items) are inserted where this placeholder appears. We need to distinguish what the **[+wf.wrapper+]** placeholder does depending on whether we are talking about the **&outerTpl** template or the **&rowTpl** template. So, take a note of the following:

1. It is very important to remember that both these templates, the **&outerTpl** and the **&rowTpl** require this **[+wf.wrapper+]** placeholder. The placeholder must be specified and correctly placed within the XHTML code in the chunks containing each of these templates in order for the menu to be correctly output.

2. In the case of the **&outerTpl** template code, Wayfinder will insert the inner content of the menu where the **[+wf.wrapper+]** placeholder is located within the **&outerTpl** template code. By inner content I mean the rows/list items of the menu. In other words, we can say that Wayfinder will place the contents of the **&rowTpl** template in the **[+wf.wrapper+]** placeholder of the **&outerTpl** template. So, without this placeholder, we would have no menu; just empty **
** tags!

3. In the case of **&rowTpl** template code, Wayfinder will process and insert any sub-menus present in your menu where the **[+wf.wrapper+]** placeholder appears in the **&rowTpl** template code. Any sub-menus no matter how deeply nested will be placed within this placeholder. Without this placeholder, no sub-menus will be output in case your menu has any!

We will look at creating menus that have sub-menus in the next lesson but for now let us continue and create the **&rowTpl** template.

Creating the &rowTpl template

Apart from the content of the chunk and the name we give it, there is really no difference between creating the **&rowTpl** template and the **&outerTpl** template and for that matter any template in Wayfinder.

Create a new chunk; name it **menuRows**, type the default **&rowTpl** template code in it and save it. You could give your **&rowTpl** a different name if you wanted but make sure to use that same name in your Wayfinder call.

```
<li [+wf.id+] [+wf.classes+] >
<a href="[+wf.link+]" title="[+wf.title+]" [+wf.attributes+]>[+wf.linktext+]</a>
[+wf.wrapper+]
</li>
```

Using this information, in our Wayfinder call, the **&rowTpl** would be specified as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows`]]
```

With regard to class names, similar to the **&outerTpl** template parameter, the **&rowTpl** template parameter also has a class parameter whose value WILL ONLY be assigned to row level templates and no other templates. By row level templates I mean Wayfinder templates that are only applicable to the rows/list items that Wayfinder produces, i.e. templates for the **
** part of the menu. The corresponding class for **&rowTpl** is the **&rowClass**. In the coming lessons we will look at other row

level templates available in Wayfinder including their matching/relevant row level Wayfinder class name parameters.

The &rowClass class name parameter

The **&rowClass** class parameter is assigned ONLY to row level templates. So, this class will not be applied within **<ul>** tags since **<ul>** tags are not "row tags" but are "container tags". Similar to other class parameters, if you want to use this parameter you need to specify it in your Wayfinder call. Using the Wayfinder call from above, we would specify **&rowClass** as shown below by giving it a value that will correspond to the class name we want to assign the row/list items in our menu. I have given my **&rowClass** the value **"navitems".** This name is probably not very intuitive so by all means, please come up with better names for your CSS classes!

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &rowClass=`navitems`]]
```

Have a look below at the **&rowTpl** template code. This is the same template that we created above to be our **&rowTpl**:

```
<li [+wf.id+] [+wf.classes+] >
<a href="[+wf.link+]" title="[+wf.title+]" [+wf.attributes+]>[+wf.linktext+]</a>
[+wf.wrapper+]
</li>
```

In the above template, when Wayfinder outputs our menu, it will replace the **[+wf.classes+]** placeholder with **class="navitems"**.

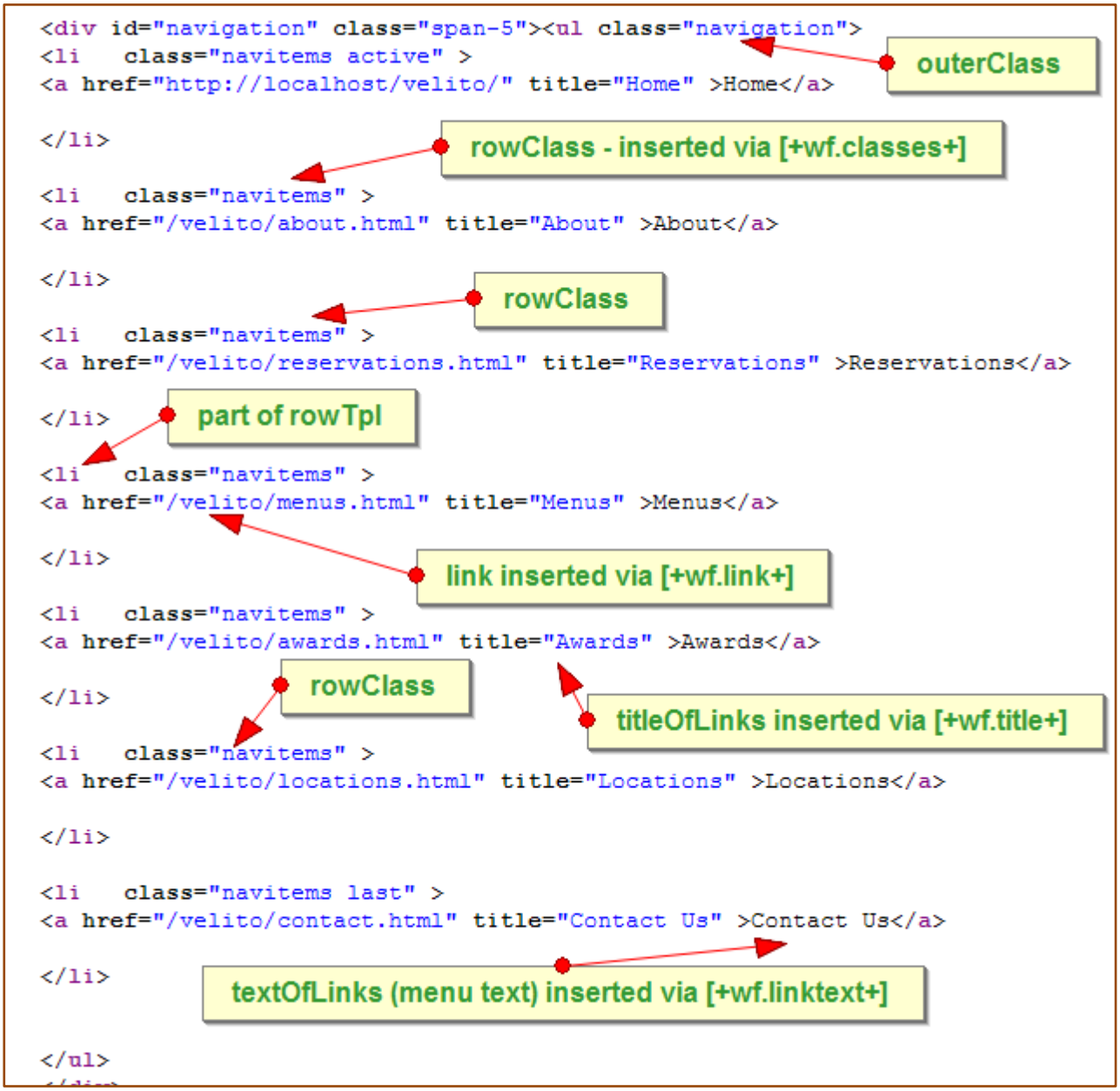
Putting it all together

Since there will be several row/list items in our menu, each of them will have the **class="navitems"** applied to their opening **<li>** tags as shown below when the menu is output.

Let us create our updated menu. Go to your website's Velito Restaurant template and modify the Wayfinder call we typed in there from lesson #1 to include what we have learnt in this lesson as shown in the call below. We will not be declaring the **&textOfLinks** and **&titleOfLinks** parameters since we want Wayfinder to use the default values for these two parameters, i.e. **menutitle** and **pagetitle** respectively. Remember to save the template. If you wish, you can tweak your CSS file to apply some CSS rules to the class **"navitems"**. Here's our now updated Wayfinder call:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &rowClass=`navitems`]]
```

Now let us view our menu in the web browser. If you followed the instructions correctly you will see that Wayfinder has output the menu just fine. Let us look at the source code. Mine looks like this:




```
<div id="navigation" class="span-5"><ul class="navigation">
<li   class="navitems active" >
<a href="http://localhost/velito/" title="Home" >Home</a>

</li>

<li   class="navitems" >
<a href="/velito/about.html" title="About" >About</a>

</li>

<li   class="navitems" >
<a href="/velito/reservations.html" title="Reservations" >Reservations</a>

</li>

<li   class="navitems" >
<a href="/velito/menus.html" title="Menus" >Menus</a>

</li>

<li   class="navitems" >
<a href="/velito/awards.html" title="Awards" >Awards</a>

</li>

<li   class="navitems" >
<a href="/velito/locations.html" title="Locations" >Locations</a>

</li>

<li   class="navitems last" >
<a href="/velito/contact.html" title="Contact Us" >Contact Us</a>

</li>

</ul>
```

As can be seen from the above source code of our menu, all of our row/list items have the class **"navitems"** applied to them. This is not really necessary because in this particular case, we do not really need each of our menu's list items to have a class applied to it. This is because we can easily target these **<li>** elements using CSS descendant rules. For instance, using CSS, we can target the **<li>** elements in our menu using their parent element **<ul>** that has the class **"navigation"** in the same way we did in lesson #1.

I am not saying that the parameter **&rowClass** is of no use. There might be situations where you might want to use this parameter. Similarly, depending on how you structure your **&outerTpl**, in some situations, you can avoid using the **&outerClass** parameter. For instance, if your gave your **&outerTpl** a class within the template code as shown below you would have no need to use the **&outerClass** to specify a class name for the outermost part of your menu:

```
<ul class="navigation">[+wf.wrapper+]</ul>
```

Another example is when for instance, in your website template you include your Wayfinder call within a ** ** and gave that ** ** an **id** or a **class**. Using CSS descendant selectors, we can easily target all the elements that are contained within that ** **.

The important lesson for us here is that we need to carefully think how we construct our menus without being redundant or using unnecessary code. Wayfinder is a very robust snippet with many possibilities but this does not mean we have to use all the features it offers in one go! If you carefully and thoughtfully plan your projects you may be able to use as few Wayfinder parameters as possible. With that said, let us now modify our Wayfinder snippet call in the Velito Restaurant website template to remove the **&rowClass** parameter. Our call now looks like this:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows`]]
```

Save the template and also delete any rules you might have included in your CSS files for the class **"navitems"**. Remember to save the CSS file. Refresh your browser and have a look at the menu again. Check the source code too. The **"navitems"** class should have disappeared from the output.

Summary of lesson #2

Wow, that has been a long one! The good news though is that we have covered a lot of ground with regard to the basics. Before we move on, let us recap.

1. In this lesson, we looked at 4 Wayfinder parameters and 3 Wayfinder placeholders. We saw how each of these affects the output of our menu.

2. In the **[+wf.link+]** placeholder is where Wayfinder will insert the **URL** of the menu items

3. The value we give the **&titleOfLinks** parameter determines the title attribute of the links in the menu items. Wayfinder will use this value and insert it where we the **[+wf.title+]** placeholder is located in the template when it outputs our menu. By default, Wayfinder uses the **pagetitle** as the value of the title attribute of links. The **&titleOfLinks** parameter does not have to be declared in the Wayfinder call if you want it to use this default.

4. The value we give the **&textOfLinks** parameter determines the text of the menu items. Wayfinder will use this value and insert it where we the **[+wf.linktext+]** placeholder is located in the template when it outputs the menu. By default, Wayfinder uses the **menutitle** as the value of the text of menu items. The **&textOfLinks** parameter does not have to be declared in the Wayfinder call if you want it to use this default.

5. We looked at other values that we can use for our **&titleOfLinks** and **&textOfLinks** parameters.

6. We learnt about the **&rowTpl** template parameter and its associated class parameter, the **&rowClass** class name parameter. We learnt that **&rowTpl** is a row level template and that Wayfinder has other row level templates.

7. We concluded by modifying the Wayfinder call in our website's template to include the parameters and placeholders we learnt about in this lesson .We looked at the output and re-modified the Wayfinder call to delete redundant code.

That's enough for this lesson. Please re-read the lesson if there is something you are not clear about. In the next lesson we will look at creating multi-level menus, a.k.a. menus that have sub-menus. See you there!

TIP

Always troubleshoot your code. If your menu is not output, in my experience CSS is "normally" the culprit and not Wayfinder. Check the source code to see whether your menu is being output at all. If it is, the problem is not on Wayfinder's side. Check you CSS and refresh your browser

Chapter 4: Lesson #3 – levels, hidden things and the inner stuff

Lesson #3 at a glance
&level **&hideSubMenus** **&innerTpl** **&innerClass** **&innerRowTpl**

Hey there! Welcome to lesson #3. By now you know that if you have not done so already you should have read the previous lessons before reading this one. In this lesson we will be looking at other Wayfinder goodies - building multi-level menus a.k.a. menus with sub-menus. We will be building on what we have already learnt from lesson#1 and #2. By now you should be familiar with terms like "menu container" or "row/list items" with respect to Wayfinder. Because of this I will not spend time explaining what they mean. The same goes for terms like Wayfinder templates and parameters. OK, now that we have that out of the way, let us dive right in into the lesson!

Before we start I would like to clarify one thing; I may use the term "container" in two different contexts. I may use the term to refer to parent documents with respect to the MODx document tree. I may also use the same term to refer to the outermost part of the unordered list menu, i.e. the **
** container that is created by Wayfinder. When referring to parent documents I will use container document to avoid confusion. OK, let us carry on.

In this lesson we will be looking at 5 Wayfinder parameters that we have not used before. Don't let that scare you; they are all quite easy and this is where the fun really starts. As you are well aware, our menu from lesson #2 was a single level menu. We now want to build upon it and create some sub-menus. In order to do this we first need to understand the arrangement of documents in the MODx document tree.

The MODx document tree

In MODx it is possible to specify that a document is a "container". This means that such a container document can have other documents inside it. Put another way, container documents can act as folders - similar to the way files are organised in your computer - and therefore contain other documents. Because of this, documents that are containers are also referred to as parent documents. The documents they contain can also be referred to as child documents. Child documents can themselves be container documents and therefore contain other documents.

If your MODx document tree has container documents with child documents in them, it is then possible to create multi-level menus. If you were to instruct Wayfinder to output a menu using such a tree hierarchy, the topmost parent documents, i.e. the topmost container documents, will appear as the topmost level of your menu. Any child documents and grand-child documents etc will appear in subsequent levels of the menu and reflect your MODx document tree hierarchy. The best way to understand the MODx document tree hierarchy is to visualise it so let us create some documents!

We are going to build upon what we already have in our document tree. Your server should be fired up and you should be logged in to your MODx manager. We currently have 7 documents in our

restaurant website. In a couple of minutes we will be creating other documents to nest inside the ones we already have.

Below is the kind of document hierarchy we want for our restaurant website. The names in italics are the documents we are going to create:

- Home (1)
- About (2)
- Reservations (3)
- Menus (4)
  - *Monday - Thursday (10)*
  - *Friday (11)*
  - *Saturday - Sunday (12)*
- Awards (5)
- Locations (6)
  - *St. James Park (8)*
  - *Aldington (9)*
    - *Maine Street (13)*
    - *Rampant Horse (14)*
- Contact (7)

Let us look at the above MODx document tree hierarchy for a minute. Notice that we have documents at 3 levels. The documents in this tree are coloured for illustration purposes only. The blue coloured documents are at level 1 of the hierarchy. The red coloured documents are at level 2 and the green coloured ones are at level 3.

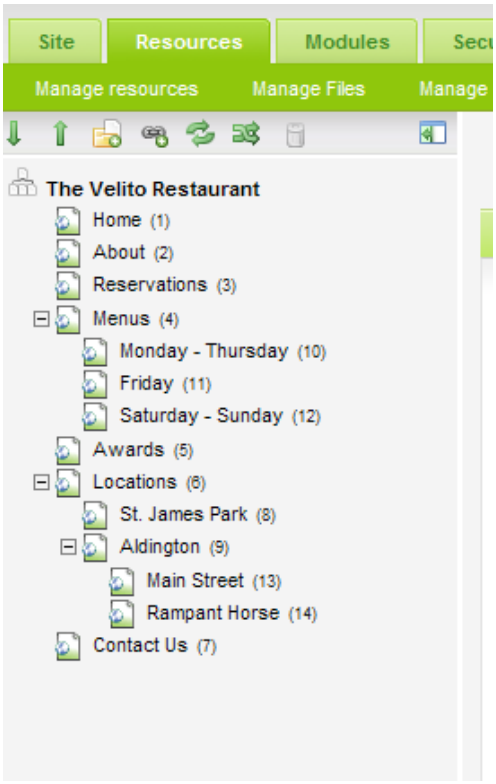
It is important to note that with respect to the MODx document tree, the levels I refer to above are relative to the root. I hope you remember that in lesson #1 when we were talking about the **&startId** we said that the root of the MODx document tree is actually the site itself; i.e. the highest level in the hierarchy. In contrast, when processing menus, Wayfinder will count document levels relative to the **&startId**. What I am trying to say here is that depending on where you want your menu to start from, in the menu that Wayfinder creates, a document that may be at level 2 of the MODx document tree hierarchy might actually be at level 1 of the menu that Wayfinder outputs. Keep this in mind because I will shortly be illustrating what I mean by this and why it is important with respect to a certain Wayfinder parameter. OK, let us go ahead and replicate the above hierarchy in our MODx document tree.

Right-click on the document "Menus" and select "create document here". This will open the create/edit document editor. Give this document the page title "Monday - Thursday". Give it the long title "Monday to Thursday's Menu". Give it a description if you wish. Also give it the menu title "Mon - Thur". Make sure the "show in menu" box is ticked, publish the document and save it.

Repeat the above process for the "Friday" and "Saturday - Sunday" documents in each case remembering to complete the appropriate settings (menu title, etc). By creating these documents inside the document titled "Menu" MODx has automatically converted this document "Menu" to be a container. If you followed the instructions correctly you should now have 3 new documents inside the "Menu" document. Let us create the others.

Follow the above process to create the "St. James Park" and "Aldington" documents inside the document "Locations". Once you have these documents in place, go ahead and create the "Main

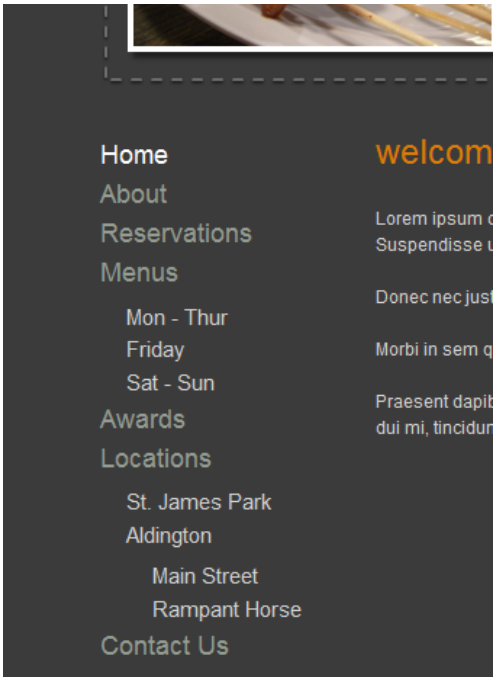
Street" and "Rampant Horse" documents inside the "Aldington" document. Your document tree hierarchy should now resemble the following.



New documents added to the Velito website

If any of your documents in the document tree seem greyed out and in italics it means the documents are not published. If this is the case right click on such documents and choose "Publish document".

From lesson #2 we already had our Wayfinder call in place in our website's template so go ahead and preview the site. You may need to refresh the page if you were already previewing the page. Here is how the resulting menu looks like:



Generated menu showing the new documents added to the Velito website

Wow! Our menu is all over the place! Yes, the menu has been output but we probably do not want to see all those menu levels – that is 3 levels of menus. Secondly, we probably do not want to see all the sub-menus shown at the same time unless their parent documents are active (i.e. current/being browsed in the menu). Let us deal with these issues one at a time.

By default, unless you instruct it differently, starting with the topmost descendant documents of the **&startId**, Wayfinder will display documents at all levels of the hierarchy in its output. Let me clarify. Using the example of our now modified document tree, if we changed the **&startId** value so that our menu begins from the document "Locations", i.e. **&startId=`6`** Wayfinder will output a menu that resembles the following:

St. James Park
Aldington
    Maine Street
    Rampant Horse

In this example, we now have a menu with two levels. Two important things to note are:

1. Wayfinder has created a menu that is showing all levels of the documents below the **&startId**. The **&startId** in this case being the document with the **id 6**.

2. In our document tree in the MODx manager, the documents "St. James Park" and "Aldington" are at level 2 of the document tree hierarchy relative to the root. However, in our menu output, they are *at level 1* of the *menu hierarchy, relative to the* **&startId** document. In addition, the documents "Main Street" and "Rampant Horse" are, relative to the root, at level 3 with respect to the MODx document tree in the backend. In the output, with regard to the menu, they are *sub-menus* and are at *level 2 of the menu, relative to the* **&startId**. Now you see why I said that with regard to the menu created by Wayfinder, the level a document is at is actually relative to the **&startId**?

I hope this has enlightened you rather than confused you!

Back to our problem, there is a Wayfinder parameter that can be used to limit the maximum number of levels of document hierarchy that are output to a menu. This is the **&level** parameter.

The &level parameter

The **&level** parameter limits how many levels of document hierarchy will be output to the menu. If you do not include this parameter in your Wayfinder call, or if you include it and give it a value of zero, i.e. **&level=`0`** Wayfinder will output all the levels of document hierarchy in the MODx tree to your menu. So, if this is what you want, you do not need to type in this parameter in your Wayfinder call. However, if like in our present case you want to limit the number of document levels that are output to the menu, you need to use this parameter and specify a value for it. That value must be an integer. For example, using the Wayfinder call above where we had our menu starting from the document "Locations", if we wanted the menu to have only one level, our Wayfinder call would look like this:

```
[[Wayfinder? &startId=`6` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &level=`1`]]
```

The resulting menu would look like this:

Menu showing the effect of the &level parameter

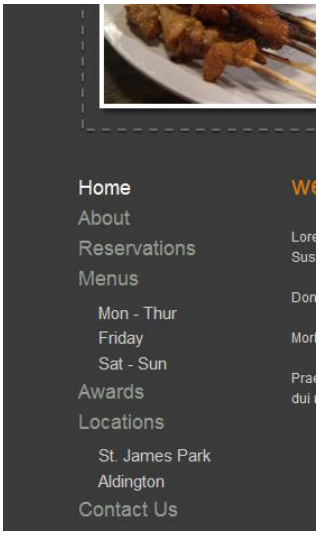
As we can see from the above image, the parameter **&level=`1`** has ensured that only one level of the document hierarchy is output to the menu - i.e. the documents "St. James Park" and "Aldington" while the "Maine Street" and "Rampant Horse" documents have been "trimmed" from the output.

This was just an example and is not the way we want the menu to appear in our Velito Restaurant website. Yes, we still want to limit the number of levels displayed in our menu but we want the menu to start from the root of the tree. Let us now modify our Wayfinder call from lesson #2 by including the **&level** parameter and giving it a value of 2 since we want only the first and second

levels of the menu to be displayed. Type the Wayfinder call to resemble the following. By now you know where we type the Wayfinder call so I do not need to go into details about this. I can mention in passing though that your Wayfinder call does not have to be placed in a template; Wayfinder can as well be called within a document or a chunk. Remember it is a snippet.

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &level=`2`]]
```

Let us preview the menu. Mine looks like this:



Menu showing effect of setting &level=`2`

We see that Wayfinder has output a 2 level menu. We still have not solved our second issue though; the one about hiding the sub-menus unless their parent document is active in the menu. For this we need to use another Wayfinder parameter. This is the **&hideSubMenus** parameter. It is pretty obvious from the name what it does. Let us see how it works.

The &hideSubMenus parameter

The **&hideSubMenus** parameter instructs Wayfinder to either show or hide sub-menus. Please do not confuse this with the **&level** parameter. We are talking about hiding/revealing the sub-menus, **AND NOT** about **EXCLUDING** them from the menu output. So, whilst the **&level** parameter will exclude documents at certain levels of the tree hierarchy from the menu output, the **&hideSubMenus** parameter may be used to temporarily hide/reveal documents that are sub-menus.

The values we can pass the **&hideSubMenus** parameter are **Boolean** in nature. This means there is only two possible values for it; either TRUE or FALSE. We can specify the values of this parameter as follows:

If we want Wayfinder to show sub-menus at all times:

```
&hideSubMenus=`FALSE`
```

OR if we want Wayfinder NOT TO show sub-menu unless their parent documents are active:

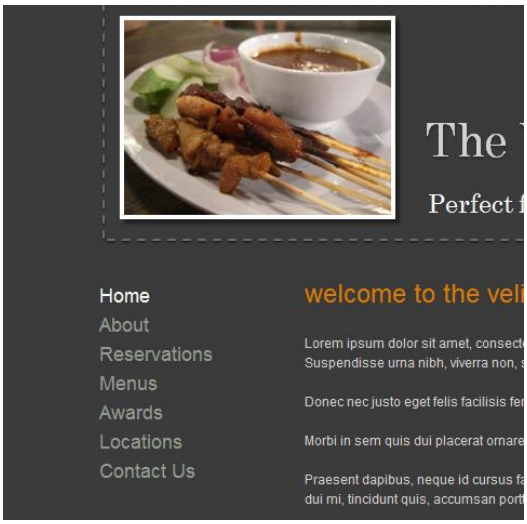
&hideSubMenus=`TRUE`

The default value for the **&hideSubMenus** parameter is **`FALSE`**. This means that by default, unless you tell it otherwise, Wayfinder will display your sub-menus whether their parent documents are currently active or not (i.e. whether they are currently being browsed or not). If this is what you desire then you do not need to declare this parameter in our Wayfinder call.

In our case, we want the sub-menus to be hidden in the output unless their parent documents are active. Let us then modify our Wayfinder call by replacing it with the code below:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &level=`2` &hideSubMenus=`TRUE`]]
```

Your menu should now resemble the following:



Menu showing the result of using &hideSubMenus=`TRUE`

Note that no sub-menus are currently being shown on the home page despite it being active since the document "Home" does not have sub-documents. Try clicking the document "Menus" or "Locations". When you click the parent documents, the sub-menus are revealed. Play around with different values for both the **&hideSubMenus** and the **&level** parameters to see how your menu output changes.

In lesson #1 we looked at the template parameter **&outerTpl**. We learnt that this parameter is always required in Wayfinder and is used to specify the outermost container for the rows that make up the menu. I am assuming that you are aware that multi-level menus are (at least commonly) nothing more than nested lists (in our case unordered lists). Have a look at the source code of the menu we have created above. This pattern of nested unordered lists will continue for each and every sub-menu no matter how deep the hierarchy continues.

By default, unless you instruct it differently, Wayfinder will use the **&outerTpl** template to construct the container for all levels of your menu, including containers for any sub-menus no matter how deeply nested they are. Please note that I am referring to the container portion of the menu - i.e. the **
** portion **AND NOT** the row level portion - i.e. the **
** parts. However, Wayfinder has a template parameter that can be used as the container for the rows of the sub-menus. This template parameter is called the **&innerTpl**.

The &innerTpl template parameter

The **&innerTpl** is a container level template and is used as the container for the rows in the sub-menus no matter how many they are and no matter how deeply nested they are. If you define the **&innerTpl** then **&outerTpl** will only be used for the highest, i.e. outermost, level of the menu structure.

```
<li    >
<a href="/velito/awards.html" title="Awards" >Awards</a>

</li>

<li    class="active" >
<a href="/velito/locations.html" title="Locations" >Locations</a>
<ul>
<li    >
<a href="/velito/st.-james-park.html" title="St. James Park" >St. James Park</a>

</li>

<li    class="last" >
<a href="/velito/aldington.html" title="Aldington" >Aldington</a>

</li>


</ul>
</li>

<li    class="last" >
<a href="/velito/contact.html" title="Contact Us" >Contact Us</a>

</li>
```

innerTpl - i.e. nested within <li> - i.e. sub-items of list starts here

end of innerTpl in this example. Note that innerTpl will be used multiple times if necessary - i.e. for all "containers" of sub-items

There is no default Wayfinder in-built template for the **&innerTpl**. If you want to use this template you will have to create one. It basically may look like the default Wayfinder **&outerTpl** template. You might want to create this template for example if you want to give the container of your sub-menus a unique class different from the main menu. You might want to do this so that you can specifically style your sub-menus differently. Remember though that I mentioned in lesson #2 that we need to put some thought into how we create our menus in order to avoid using unnecessary code or in order not to complicate things. This is important so let me reiterate. Consider the following scenario:

In lesson #1 I said that you can construct your **&outerTpl** differently from the default, to include, say, a ** ** to wrap around the whole menu. Because of this you need to be careful with the code in your template if your menu will contain any sub-menus. For instance, if your **&outerTpl** is enclosed within a ** **, and you have not defined an **&innerTpl**, then it would mean that the **&outerTpl** will be used to construct the containers of your sub-menus and that each of your sub-menus will be enclosed in a ** **! This is not an output that you would like I suppose. It would even be worse if that ** ** was given an **id**. It would mean that one **id** is used multiple times in the XHTML, which is obviously wrong and can cause all sorts of problems. See the following code which illustrates what I mean.

Assuming your **&outerTpl** looks like this:

```
<div id="nav">
<ul [+wf.classes+]>[+wf.wrapper+]</ul>
</div>
```

Using the above code as your **&outerTpl**, if you do not define an **&innerTpl** it means that Wayfinder will use the **&outerTpl** to construct your sub-menus. Bottom line - you will end up with a messed-up menu!

That said, let us now create an **&innerTpl** template. By now you know what the process is. Create a chunk and give it a unique name. I have called mine **innerContainer**. Type the following code in it and save it.

```
<ul [+wf.classes+]>[+wf.wrapper+]</ul>
```

Alternatively you can give the **<ul>** above a class directly in the **&innerTpl** as follows:

```
<ul class="submenu">[+wf.wrapper+]</ul>
```

If we were to use this second method, all our sub-menus' containers will be wrapped in **<ul>** tags with the class **"submenu".** For our restaurant project, let us stick with the first method because I want to show you the Wayfinder class parameter that is applied to the **&innerTpl**. This class parameter is known as the **&innerClass**.

The &innerClass class name parameter

The **&innerClass** parameter if used will only be applied to the **&innerTpl** and no other template. Since **&innerTpl** if defined will be used to construct all the containers for the rows of the sub-menus, it follows that the value you give **&innerClass** will be assigned to all those sub-menu containers. As with the other class parameters this value will be inserted where the **[+wf.classes+]** or the **[+wf.classnames+]** placeholder is placed in the **&innerTpl** template code. You will also need to declare this **&innerClass** parameter in your Wayfinder call. Let us add it to our Wayfinder call together with the **&innerTpl** parameter. I have given my **&innerClass** the value "**subnav**". Our Wayfinder call now looks like this:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &innerTpl=`innerContainer` &innerClass=`subnav` &level=`2`
&hideSubMenus=`TRUE`]]
```

Before we preview the menu let us have a look at the last parameter to be covered in this lesson. We found out in lesson #2 that Wayfinder has a default in-built row level template that it uses to create the rows or list items of your menus. This template parameter is the **&rowTpl** and is used to construct all the rows of your menu regardless of whether they are in the main menu or sub-menu levels. However, we also learnt that Wayfinder has other row level templates. The first one we will look at is the **&innerRowTpl**.

The &innerRowTpl template parameter

The name of this parameter gives us a pretty good idea about what it does. The **&innerRowTpl** template parameter if defined will be used by Wayfinder to construct all the rows of your sub-menus no matter how deeply nested they are. If you use the **&innerRowTpl** then the **&rowTpl** will only be used to construct the rows of the top level menu. All the placeholders available to the **&rowTpl** are also available to the **&innerRowTpl**. See the appendix for a list of all the placeholders available for row level templates. Since it is a row level template, the **&innerRowTpl** template code will look pretty much similar to **&rowTpl**. This is not to say it is of no use. There might be cases where you might want to use this parameter to give you greater flexibility in constructing your multi-level menus. The value given to the **&rowClass** class parameter will also be assigned to the rows created by the **&innerRowTpl**. In other words, there is no special class only applicable to the **&innerRowTpl**. To illustrate this, we will add the **&rowClass** to the Wayfinder call and look at the output. In practice you will probably not need to do this since as mentioned earlier, you can use the cascading aspect of CSS to style those nested items of the menu.

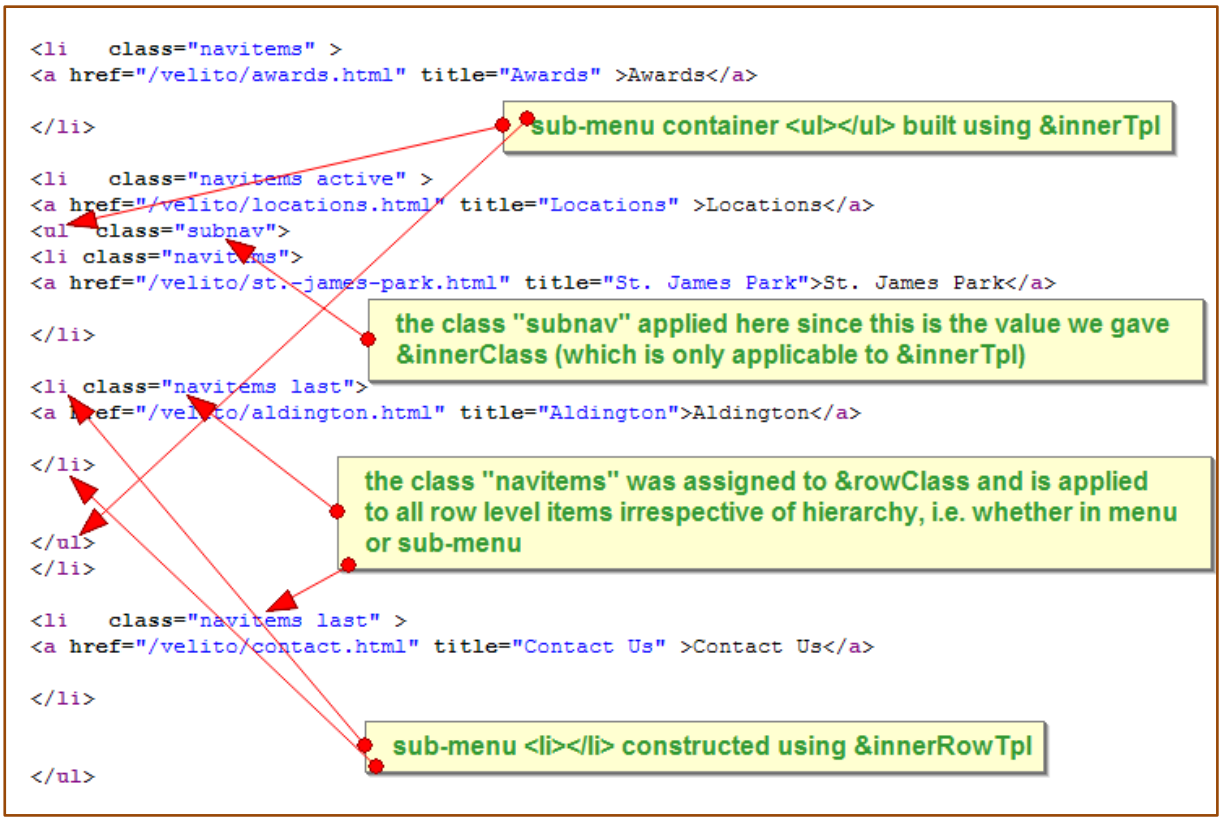
Let us create an **&innerRowTpl** for our restaurant website. The process is the same as for the other Wayfinder templates. Create a chunk and give it a name. I have called mine **submenuRows**. Type the following code in the chunk and save it. I have chosen not to use some placeholders in my template chunk. Please note, however, that the **[+wf.wrapper+]** placeholder cannot be left out since this is where any sub-menus at all levels of the hierarchy will be placed.

```
<li[+wf.classes+]><a href="[+wf.link+]" title="[+wf.title+]">[+wf.linktext+]</a>[+wf.wrapper+]
</li>
```

Now let us add the **&innerRowTpl** to our Wayfinder call. It should now look like this:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &innerTpl=`innerContainer` &innerClass=`subnav`
&innerRowTpl=`submenuRows` &rowClass=`navitems` &level=`2` &hideSubMenus=`TRUE`]]
```

Preview the menu in your browser. Click on the menu item "Locations" and then have a look at the source code (remember that Wayfinder dynamically generates the menus on the fly so the source code might look different depending on the document that is currently being viewed). Here is how my source code looks like with the document "Locations" selected in the menu.




```
<li    class="navitems" >
<a href="/velito/awards.html" title="Awards" >Awards</a>

</li>

<li    class="navitems active" >
<a href="/velito/locations.html" title="Locations" >Locations</a>
<ul    class="subnav">
<li class="navitems">
<a href="/velito/st.-james-park.html" title="St. James Park">St. James Park</a>

</li>

<li class="navitems last">
<a href="/velito/aldington.html" title="Aldington">Aldington</a>

</li>


</ul>
</li>

<li    class="navitems last" >
<a href="/velito/contact.html" title="Contact Us" >Contact Us</a>

</li>


</ul>
```

As we can see, all the row/list items of the menus AND sub-menus have been given the class **"navitems"** which is the value we assigned to **&rowClass**. The sub-menus are hidden until one clicks on their parent menu item as per the instructions we gave the **&hideSubMenus** parameter. In addition, the container tags for the sub-menus, i.e. the **<ul>** for the sub-menus have been constructed using the **&innerTpl** and have the class **"subnav"** which we assigned the **&innerClass**. Finally, the sub-menu items, i.e. the **<li>** of the sub-menus, have been constructed using the **&innerRowTpl** that we created and instructed Wayfinder to use.

That is it for this lesson folks. As usual, before we finish, let us sum up what we have learnt today.

Summary of lesson #3

1. In this lesson, we learnt that by default, Wayfinder will output all the levels of the document hierarchy unless you instruct it differently. In order to limit the number of levels of hierarchy that Wayfinder outputs to the menu, we use the **&level** parameter

2. Wayfinder will normally show the main menu items and their sub-menus at all times unless we change this behaviour. To do this, we use the **&hideSubMenus** parameter. If the value of this parameter is set to "TRUE", this will ensure that sub-menus are hidden until their parent documents are currently active, i.e. are clicked.

3. If defined, the **&innerTpl** template will be used to construct the outermost container of all sub-menus. If **&innerTpl** is not defined, the **&outerTpl** template parameter will instead be used to do this.

4. The **&innerClass** class name parameter is used to assign the class name for the **&innerTpl**, i.e. the outermost containers of sub-menus

5. The **&innerRowTpl** template parameter if defined will be used to construct all the rows of sub-menus. Conversely, if the **&innerRowTpl** is not defined, then the **&rowTpl** will be used to construct all the rows of the menu at all levels.

6. We also learnt that we need to thoughtfully craft our Wayfinder templates so that we can achieve the desired menu output and also create valid XHTML menus.

In the next lesson we will look at parameters that can be assigned to documents that have children - i.e. are parent row menu items. See you there!

Chapter 5: Lesson #4 – parents and limits

Lesson #4 at a glance
[+wf.attributes+] **&limit** **&parentRowTpl** **&parentClass** **&categoryFoldersTpl**

Welcome to lesson #4 of this Wayfinder guide. I am glad you are still here and hope that you are finding the lessons useful. What we have covered so far should be enough to get you on your way to constructing some nice looking menus using Wayfinder. There is still more cool stuff though and I want to show you some of these. Unfortunately this lesson may lack the flow of the previous lessons since we will be looking at some Wayfinder parameters that may not be very much related to each other. Nonetheless, let us dig in.

In lesson #2 I showed you the default Wayfinder **&rowTpl** template (which we replicated). There were two placeholders in this template that we did not cover. In this lesson I would like us to look at one of these placeholders. The remaining one we will cover in a later lesson.

The default Wayfinder **&rowTpl** template is:

```
<li [+wf.id+] [+wf.classes+] >
<a href="[+wf.link+]" title="[+wf.title+]" [+wf.attributes+]>[+wf.linktext+]</a>
[+wf.wrapper+]
</li>
```

In lesson #2, we did not cover the **[+wf.attributes+]** placeholder and the **[+wf.id+]** placeholder. Today we will be looking at the former.

The [+wf.attributes+] placeholder

The **[+wf.attributes+]** placeholder is where, in the output menu, Wayfinder will insert the link attributes of the document. Link attributes include **rel="alternate"**; **target="_blank"**, etc. Please note that the "target" attribute is deprecated in strict XHTML 1.0 and is only shown here as an example.

If you would like to use the **[+wf.attributes+]** placeholder, you must have given your document a link attribute. Let us do this for one of our documents. I assume you are already logged into your MODx manager. If not, please do so now.

Right-click on the document "Home" and click "Edit document". While on the "General" tab, just below the "Document's alias" and above the "Summary (introtext)", there is a field titled "Link Attributes".

Enter some valid link attributes in this field. The doctype of the template for my restaurant website is specified as XHTML 1.0 transitional so I can use the attribute **target="_blank"**. Save your document and preview the menu to see if the link attribute was applied by clicking the menu item "Home". It should open in a new browser window is you used the **target="_blank"** attribute. Also have a look at the source code. Remember in lesson #2 we created a **&rowTpl** that is similar to the default Wayfinder one and in there we had the **[+wf.attributes+]** placeholder. If you followed the instructions correctly the menu should work just fine and you should have seen the changes brought about by using this placeholder. Here is how the source code of my menu looks after I applied a **target="_blank"** attribute to the document "Home".

```
<li    class="navitems active" >
<a href="http://localhost/velito/" title="Home" target="_blank">Home</a>
</li>
```

link attribute of the document applied where we inserted the [+wf.attributes+]

Next, we will have a look at a Wayfinder parameter that can be used to limit the number of documents that Wayfinder will include in its menu output. This is the **&limit** parameter.

The &limit parameter

The **&limit** parameter is used to tell Wayfinder the maximum number of items it should include in the menu. This parameter can only take integer values. For instance, we can specify the **&limit** parameter as follows:

&limit=`5`

The above means that Wayfinder will only output the first 5 documents in the MODx document tree to the menu. The counting starts from the root downwards and takes into account the menu index assigned to each document. For instance, if a document at level 1 of the document tree has a menu index of say "**1**" and a child document (of another parent document) at level 2 of the document tree has a menu index of say "**0**", the child document will be given precedence over the document at level one with the menu index 1.

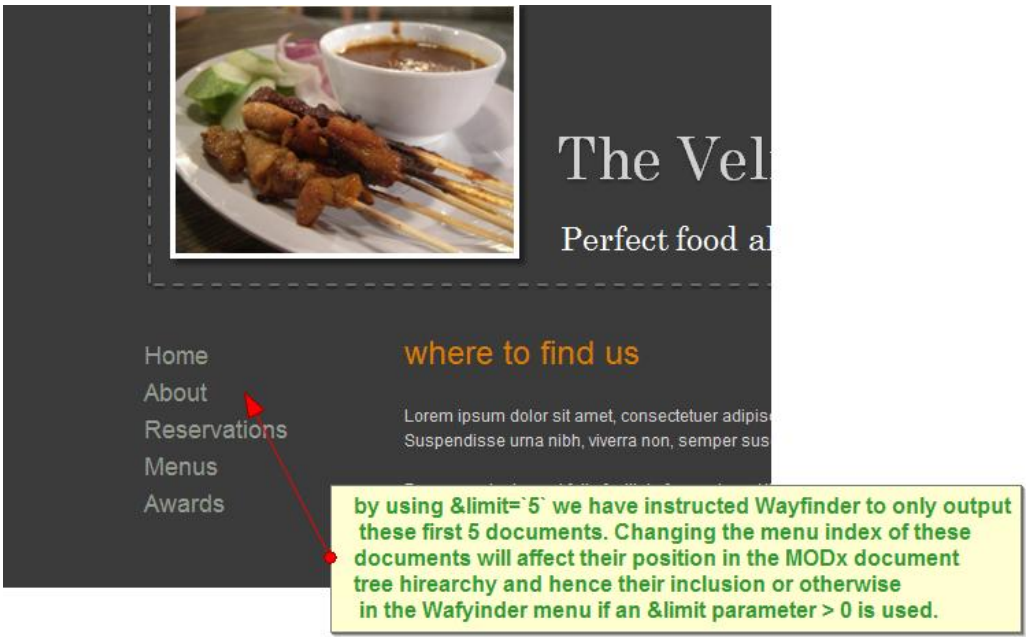
The absence of this parameter or a value of "**0**" (this is the default so this parameter does not have to be declared) will instruct Wayfinder not to limit the number of documents to be output to the menu. Let us try it to see how it works.

Add the **&limit** parameter to our Wayfinder call from lesson #3 as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &innerTpl=`innerContainer` &innerClass=`subnav`
&innerRowTpl=`submenuRows` &level=`2` &hideSubMenus=`TRUE` &limit=`5`]]
```

! Note that the above Wayfinder call has been modified to remove some parameters that we will not be using, e.g. the **&rowClass**.

Once you have added the parameter to the website template, save and preview it. You will see that only the first 5 documents have been output. Here is how my menu looks like after applying this parameter.



Menu showing the &limit parameter applied

Notice that a few documents from our original menu are missing from the menu. Play around with this parameter to better understand how it works. Try using different values to see how this affects the menu output. You can also try changing the menu index of your documents to see how this will affect their inclusion or otherwise, in the menu.

In our restaurant website we do not need to use this parameter so remove the **&limit** parameter from the Wayfinder call save the template.

Moving on, we have already learnt that Wayfinder has several row level templates. The one we are going to look at next only affects the menu items that are parents, i.e. documents that are containers, or put another way, documents that have children (when viewed in the MODx document tree hierarchy these documents have a "**+**" sign next to them).

The &parentRowTpl template parameter

The **&parentRowTpl** template parameter is used to process items that are containers/folders AND that have their children being displayed in the menu. It is important to note the following:

1. There are several Wayfinder parent templates. *(We will look at the others in coming lessons)*

2. Parent templates will only be used for parent documents that have child documents shown in the sub-menus. This means that if the child documents **are not** being shown in the menu then the parent template will not be used!

3. In addition, note that the template will be applied to all parent items meeting the above two specified conditions at all levels of the menu. This means that you could have a parent document in a sub-menu

Before we test this template parameter, let us look at its related parent class parameter. This is the **&parentClass** class name parameter.

The &parentClass class name parameter.

The **&parentClass** class name parameter is applied to each menu row item that is a parent. If the document has no children, it means it is not a parent and this class name parameter will not be applied to it. For our Restaurant's website menu, we will specify a parent class with the value **"navparents"**. As with the other class name parameters, the value given to the **&parentClass** parameter will be inserted where we have the **[+wf.classes+]** or the **[+wf.classnames+]** placeholder in our Wayfinder templates.

Creating the &parentRowTpl template parameter

You should by now be familiar with creating Wayfinder template chunks. Create a template chunk and give it a name. Mine is named "**menuParents**". Write the following code in this template and save it.

```
<li[+wf.classes+]><a href="[+wf.link+]" title="[+wf.title+]">[+wf.linktext+]+</a>
[+wf.wrapper+]
</li>
```

Look carefully at the above code and notice a couple of things:

The first one is that the code is pretty much similar to row level templates we have seen before. This should not come as a surprise because the **&parentRowTpl** is also a row level template.

The second thing to note is that I have added a **+** (plus sign) next to the text of menu items (i.e. next to **[+wf.linktext+]**). **This is not** a special Wayfinder parameter or anything to do with Wayfinder. It might as well have been a dollar sign or a hash sign. What I have done is just to add a visual aid that will appear in the output to intimate that the document with a + sign next to it has child documents beneath it. Since this is a **&parentRowTpl** template that we are defining, all parent row items will have this + sign next to them. Note that I have chosen to highlight this + sign in red just for visual purposes.

Before we preview the output, let us specify these parent level parameters in our Wayfinder call.

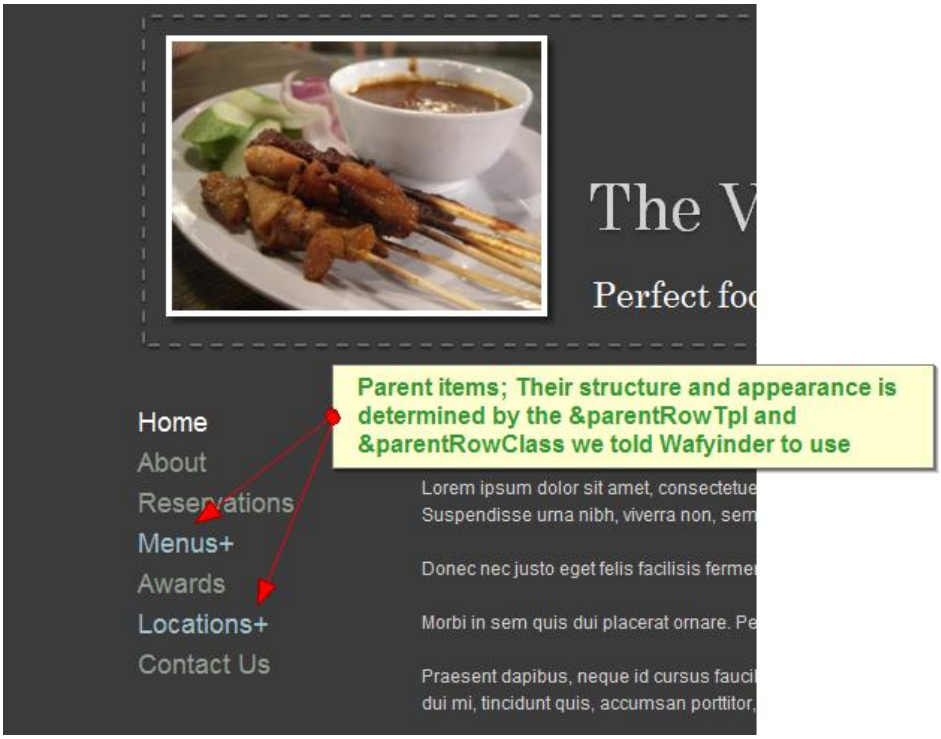
Specifying the &parentClass and the &parentRowTpl in the Wayfinder call

Add the above two parameters to your Wayfinder call so that it now looks as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &innerTpl=`innerContainer` &innerClass=`subnav`
&innerRowTpl=`submenuRows` &level=`2` &hideSubMenus=`TRUE` &parentRowTpl=`menuParents`
&parentClass=`navparents`]]
```

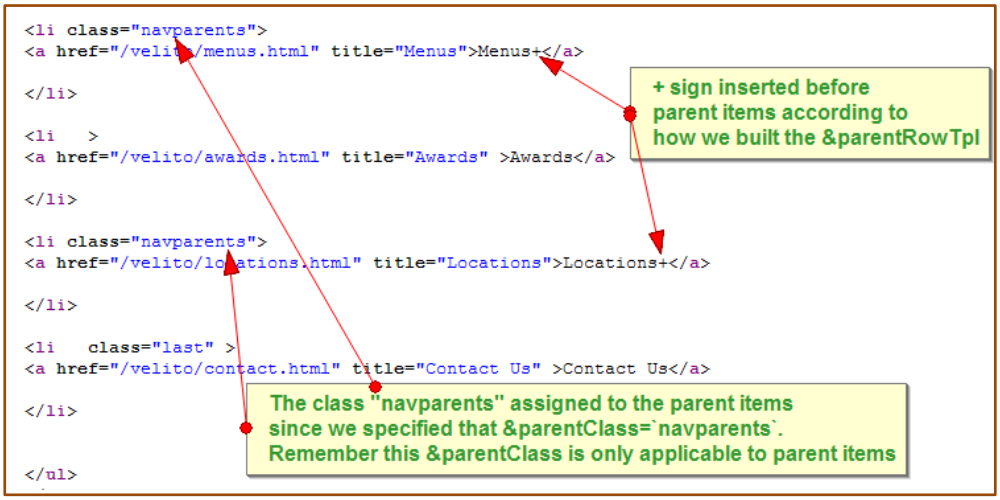
In addition you could add some CSS styles for the class **"navparents"**. This will style all document items in the menu that are parents. I have given mine a subtle blue colour.

We are now ready to preview the menu. Preview yours to see if everything turned out fine. Also look at the source code of the menu to see what classes were applied to what items. Here is how my menu looks like:



Parent menu items structured and styled using &parentRowTpl & &parentRowClass

And here is how the source code looks like:



You should notice a couple of things if everything went fine. These are:

1. All the documents that are parents have now been given the class "**navparents**". The menu items that are not parents DO NOT have this class applied to them. If we had specified a **&rowClass** in our Wayfinder call, it would not have been applied *to parent row* items since we have defined a **&parentClass**.

2. The parent menu items look different from the other menu items. This is due to the CSS styling I applied to them.

3. Despite the fact that parent items are also row level templates, only one template can be applied to them. In our Wayfinder call, we have a **&rowTpl**, **&innerRowTpl** and the **&parentRowTpl**. All these would not be applied to the same row item even if it met all the necessary criteria. The question is, which Wayfinder template would be applied to row items that meet the criteria for all these templates? Fortunately, Wayfinder has a template processing order. Refer to the appendix for Wayfinder's template processing order. According to this processing order, the **&parentRowTpl** has processing precedence over both **&rowTpl** and **&innerRowTpl** and will therefore be applied to documents that meet the relevant criteria; i.e. that are parents and have their children as part of the menu.

The final parameter we will be looking at in this lesson is the **&categoryFoldersTpl**.

The &categoryFoldersTpl template parameter

The **&categoryFoldersTpl** is a useful row level Wayfinder template parameter that you can use if you want certain parent row items of your menu to act as menu category or section headings of the child documents under them. Let me clarify this using an example. Let us say in our restaurant menu we did not want the menu item "Locations" to be clickable (i.e. it should not be a link). However, we still want it to appear in the menu item *only not as a link*; we want it to appear as a category heading for all its child documents under a category called "Locations". In our case, we would have two documents that would appear within this category heading "Locations". These are the "St. James Park" and the "Aldington" documents. The effect of this would be these two documents seemingly appearing under this category heading "Locations". Using the **&categoryFoldersTpl** template parameter we would be able to achieve this.

In order to use the **&categoryFoldersTpl** template parameter, the document that we want to be output as a category heading must, in addition to being a parent document with its child documents shown in the menu, meet one of the following conditions:

The document **must** either:

1. Have its template set to **"blank"** i.e. it has no template associated with it in the document settings. In the create/edit document editor in the MODx manager, you can set a document as having a "**blank**" template.

OR

2. Have its link attributes set to **category**. We looked at the link attributes at the beginning of this lesson. In order to set the link attribute as category, we would type the following in the link attributes field of the relevant document as follows:

rel="category"

If either of the above conditions are true, and the document we want to be a category is a parent document, then, if defined, Wayfinder will use the **&categoryFoldersTpl** to construct the parent row items of our menus that we want to be category headings.

Please note that Wayfinder will not automatically make these "category headings" *non-clickable*. You will need to properly construct the **&categoryFoldersTpl** to achieve this. To do this, you can remove the anchor links from the **&categoryFoldersTpl** template. Let us construct this template so that you can see what I mean.

Creating the &categoryFoldersTpl template

Create a new chunk; give it a name, type the following code in it and save it. I have called my chunk "**catHeadings**".

```
<li class="menuheading">[+wf.linktext+][+wf.wrapper+]</li>
```

Notice that I have only used two placeholders that I absolutely need in this template. I need the **[+wf.linktext+]** placeholder so that Wayfinder will insert a text for my menu category heading. I also need the **[+wf.wrapper+]** placeholder so that Wayfinder will have a place to insert the subsidiary content of my menu - i.e. the sub-menus. Finally, I have given my category heading items the class "**menuheading**" which I will be able to apply some CSS styles to.

Preparing the documents that will use the &categoryFoldersTpl template

The next step is to make sure that the document we want to act as a category is properly set up to meet either one of the latter two conditions described above. In our case, we only need to do this for one document, i.e. the "Locations" document. I prefer to use the second method above, i.e. give the document a link attribute of **rel="category"**. These are the steps to follow:

In your MODx manager right-click on the document "Locations" and choose **"Edit document".** We already know where the **"Link Attributes"** field is. In this field type the following:

rel="category"

Save the document.

Specifying the &categoryFoldersTpl template in the Wayfinder call

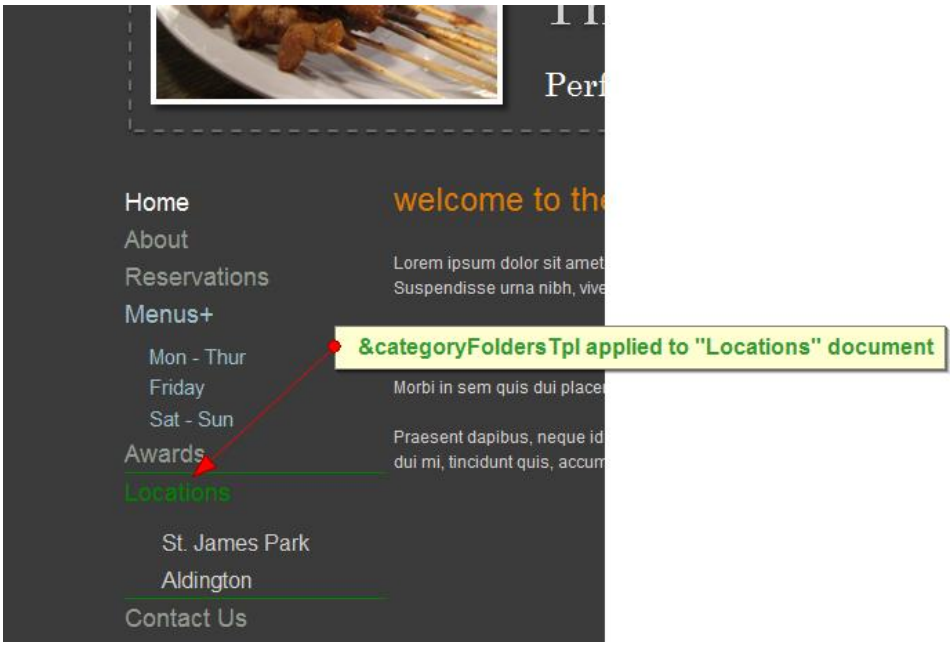
The final step is telling Wayfinder the name of the **&categoryFoldersTpl** template chunk. We do this in the Wayfinder call. Navigate to your Wayfinder call and modify it as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &innerTpl=`innerContainer` &innerClass=`subnav`
&innerRowTpl=`submenuRows` &level=`2` &hideSubMenus=`FALSE`
&parentRowTpl=`menuParents` &parentClass=`navparents` &categoryFoldersTpl=`catHeadings`]]
```

! Notice that I have also changed the value of **&hideSubMenus** to **FALSE**. I have done this just for this particular example. This is because if we hide sub-menus by default and the "**Locations**" document is now not clickable, we will not be able to see its sub-documents.

In addition, if you want, you can write some CSS styles for the class for your category heading items. My class for the category heading is "**menuheading**".

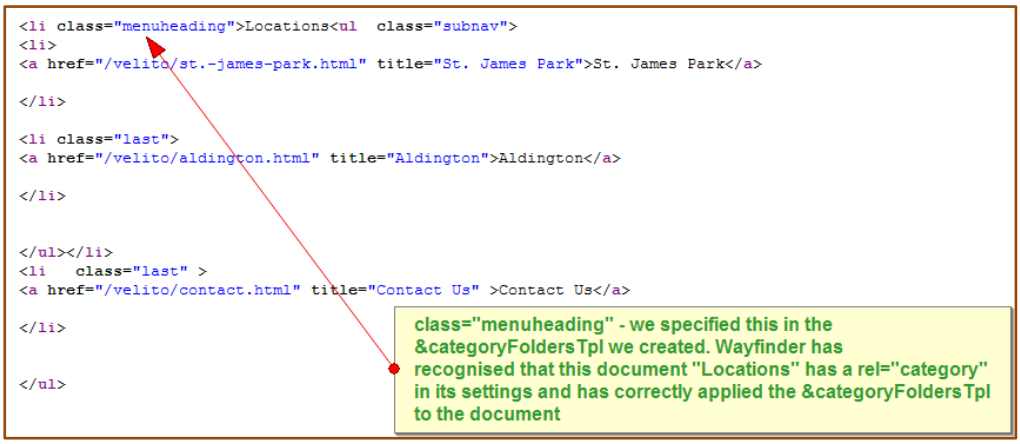
OK, we are all done. Preview the menu and also have a look at the source code. Since we changed the **&hideSubMenus** to **FALSE** all the sub-menus should be showing. You should also see that the menu item "Locations" still appears in the menu but *it is not clickable - i.e., it is not a link*. If you gave it some different CSS styling then this should also be apparent. So far so good - it does look like a menu category heading. This is how my menu looks:



The &categoryFoldersTpl applied to the menu item "Locations"

Let us now turn to the source code. Notice that the list item "Locations" has the class "**menuheading**" applied to it **AND NOT** the **&parentClass** class value, i.e. "**navparents**". If we had several category heading items they would all have this class applied to them. This is because the **&categoryFoldersTpl** has precedence over **&parentRowTpl** in Wayfinder's template processing order.

Here is how my source code looks:



That should be it for this lesson folks. Before we finish, let us modify our Wayfinder call since we will not be using the **&categoryFoldersTpl** template parameter for our restaurant's website. Revert the Wayfinder call back to how it was before we specified the **&categoryFoldersTpl** template parameter and save the template. The Wayfinder call should now look like this:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &innerTpl=`innerContainer` &innerClass=`subnav`
&innerRowTpl=`submenuRows` &level=`2` &hideSubMenus=`TRUE` &parentRowTpl=`menuParents`
&parentClass=`navparents`]]
```

You might probably find the **&categoryFoldersTpl** template parameter more useful for vertical side-menus rather than horizontal ones. In such a case, you could use this template to create a vertical menu that looks like it has category headings that divide your menu in some logical sections. You would of course need to have your sub-menus shown by default. Play around with the template to see what works for you. One thing to remember is that there are other ways of achieving this "category" effect in MODx without reverting to using this template. It is beyond the scope of this guide to discuss these methods so I would recommend that you dig around the MODx forums.

As usual, let us summarise to wrap up this lesson.

Summary of lesson #4

1. You can use the **[+wf.attributes+]** placeholder to instruct Wayfinder where to insert the link attributes of your documents

2. By using the **&limit** parameter, you can limit the maximum number of row items that Wayfinder will output to your menus

3. The **&parentRowTpl** template parameter can be used to define menu row items that are parents. Their child documents must be shown in the menu for this template to be applied.

4. The **&parentClass** class name parameter if defined will be applied to row items that are parents.

5. The **&categoryFoldersTpl** can be used to create menus that "seem" to have category headings. This can be useful for creating sections in your menus. For this parameter to work, the category heading items must be parent documents with their children being shown in the menu. In addition, the parent documents must either have their link attributes set to **rel="category"** OR have their **templates set to "blank"**.

In the next lesson, we will be looking at creating "bread-crumb" effects; i.e. the "you are here" type of effect on menus. Hope to see you there!

Chapter 6: Lesson #5 – here, here, who's active then?

Lesson #5 at a glance
&hereClass **&selfClass** **&hereTpl** **&innerHereTpl** **&parentRowHereTpl** **&activeParentRowTpl**

Welcome to lesson #5 of this Wayfinder guide. You have really covered a lot just by getting to this point; well done!

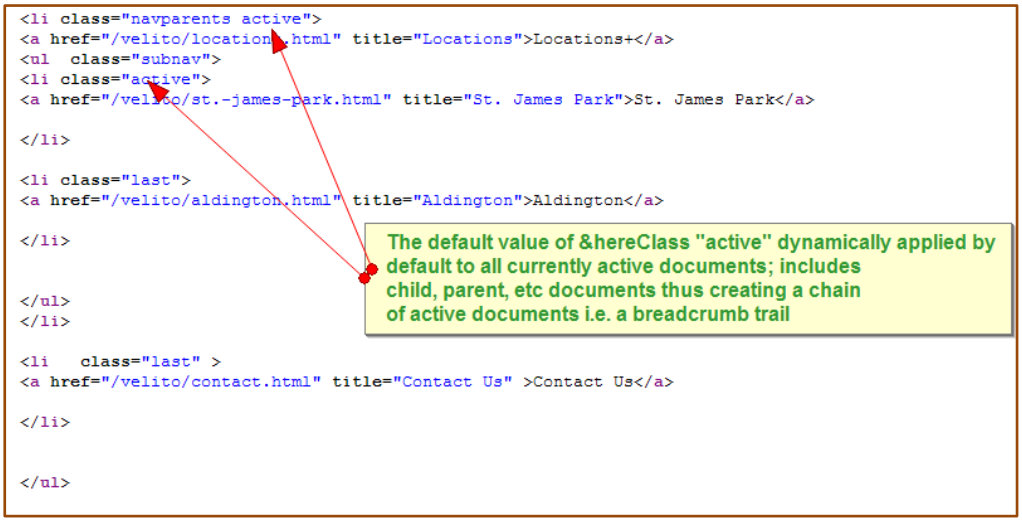
In this lesson we will be looking at some Wayfinder parameters that can aid in the navigation of your website. We will be looking at 2 Wayfinder class parameters and 4 template parameters in this regard. Let us get started.

Let us assume you wanted your website visitors to know which page(s) they were currently viewing with respect to your site's menu/navigation. One way of doing this would be to have a different (CSS) style for the active or currently being browsed menu items. Wayfinder offers us several options to achieve just this. Let us look at each of them in turn.

The &hereClass class name parameter

By default, Wayfinder dynamically applies the value of the **&hereClass** class name parameter to all menu items that are currently active. This is very handy in styling currently active pages. Basically, the **&hereClass** will be applied to the current item and to each parent, grandparent item, etc up the document tree thereby creating a "you are here" state all the way up the chain of documents. This creates a "breadcrumb trail" of the active documents in your menu. This makes this parameter a very powerful arsenal in Wayfinder's bag of tricks.

The default value for the **&hereClass** class parameter is **"active"**. Because this parameter's value is applied by default, you DO NOT need to declare it in your Wayfinder call. This is why, if you check the source code of the menus created by Wayfinder, you will notice that currently active documents have the class **"active"** applied to them. Have a look at the source code of the menu we created from previous lessons. Do you see the class **"active"** applied to the page you are currently viewing including its parent documents? If you navigate to another page and view the source code you will see that the class **"active"** has now been applied to *that* current document that you are browsing and any parent documents if applicable. Pretty neat! You obviously have to create some CSS rules for the class **"active"** in order to style these differently if you wish. Below is an example of the source code generated when browsing the document "St. James Park".




```
<li class="navparents active">
<a href="/velito/location.html" title="Locations">Locations+</a>
<ul  class="subnav">
<li class="active">
<a href="/velito/st.-james-park.html" title="St. James Park">St. James Park</a>

</li>

<li class="last">
<a href="/velito/aldington.html" title="Aldington">Aldington</a>

</li>


</ul>
</li>

<li   class="last" >
<a href="/velito/contact.html" title="Contact Us" >Contact Us</a>

</li>


</ul>
```

See how the class **"active"** has not only been assigned to the document "St. James Park" but also to its parent document, "Locations"**?**

One thing to note though is that because of the "inheritance" nature of CSS, the CSS styles you apply for this **"active"** (and may be other) class may "trickle" down and be applied to other elements, thereby creating a result that you did not intend. This is not a shortcoming in Wayfinder; it's just the way CSS was designed - to cascade. So, you need to be aware of this and work around it.

If you either wanted your current menu items to be given a class with *a different* name besides "active" or you did not want Wayfinder to automatically assign a class to the currently active documents in the menu, you could do this easily by specifying these in the Wayfinder call. Let us see how we can do this.

Assign a different value to the &hereClass class name parameter

Let us assume that for our restaurant website we want the currently active menu items to have the class **"current"**. To have Wayfinder assign a class other than **"active"** to currently active items in your menu, you will need to assign the class name you want for this parameter in the Wayfinder call. Using our Wayfinder call from lesson #4, we could add the parameter and the value we want as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &innerTpl=`innerContainer` &innerClass=`subnav`
&innerRowTpl=`submenuRows` &level=`2` &hideSubMenus=`TRUE` &parentRowTpl=`menuParents`
&parentClass=`navparents` &hereClass=`current`]]
```

Now that was pretty easy. By now I don't need to mention that as in all the other Wayfinder classes, this one too will be inserted where the **[+wf.classes+]** or **[+wf.classnames+]** placeholders appear in your Wayfinder template.

Once you have changed the Wayfinder call as shown above preview the results in your browser and have a look at the source code. You will see that the page you are currently viewing has been assigned the class **"current"**.

I notice our Wayfinder call is starting to get a little bit long. Remember though you do not have to use all the parameters available in Wayfinder. With some careful and thoughtful prior planning, you will be able to limit the number of parameters used in your Wayfinder call. Let us now see how we can instruct Wayfinder not to use the **&hereClass** class parameter at all.

Instruct Wayfinder not to assign the &hereClass class name parameter

This is really quite easy to do. All you have to do is to declare the parameter in your Wayfinder call but leave the value empty. See the following example on how to do this.

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &innerTpl=`innerContainer` &innerClass=`subnav`
&innerRowTpl=`submenuRows` &level=`2` &hideSubMenus=`TRUE` &parentRowTpl=`menuParents`
&parentClass=`navparents` &hereClass=``]]
```

In the above example, notice that the **&hereClass** parameter has no value assigned to it; the backticks are *"empty"*. If you do this, Wayfinder will not assign any class names to the currently active documents. Let us test the above code; change your Wayfinder call as indicated above - i.e. with no value assigned to the **&hereClass**. Have a look at the source code. Your currently active documents should no longer have the class **"active"** assigned to them. Here is how the source code for my menu looks like when I am browsing the document "St. James Park".

```
<ul  class="subnav">
<li>
<a href="/velito/st.-james-park.html" title="St. James Park">St. James Park</a>

</li>

<li class="last">
<a href="/velito/aldingt

</li>


</ul>
</li>

<li   class="last" >
<a href="/velito/contact.html" title="Contact Us" >Contact Us</a>

</li>


</ul>
```

no "active" class is applied to this currently active document since we left the value of &hereClass "blank"

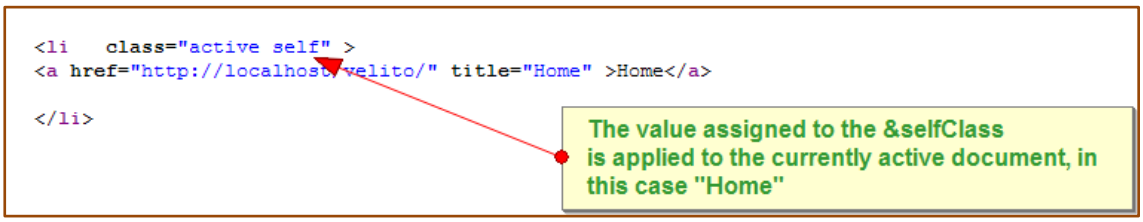
As you can see, the class "**active**" has not been assigned to the "St. James Park" menu item.

The &selfClass class name parameter

The **&selfClass** class name parameter is pretty much similar to the **&hereClass** parameter with one notable exception; it *will be applied* to the current document only AND NOT all the way up the chain of currently active child and parent documents. Another difference you need to note is that the **&selfClass** class name parameter will not be automatically assigned by Wayfinder. If you want to use this parameter you will have to add it to your Wayfinder call. This is how you would do it:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &innerTpl=`innerContainer` &innerClass=`subnav`
&innerRowTpl=`submenuRows` &level=`2` &hideSubMenus=`TRUE` &parentRowTpl=`menuParents`
&parentClass=`navparents` &selfClass=`self`]]
```

As you can see in the above example, I have given the value **"self"** to the **&selfClass** class name parameter. Do the same in your Wayfinder call and save your website template. Also add this class and some CSS rules to your CSS file to distinguish the menu item with this class from the rest. Refresh your preview browser page; you should now see that the menu item of the currently being browsed document *(and only this document)* is styled differently from the rest. Check the source code; the **<li>** tag of the document you are currently viewing should have the class **"self"** assigned to it. Here is the source code generated when the document "Home" is active.



OK, let us now see the other parameters we can use to assist with the navigation of our websites using Wayfinder.

The &hereTpl template parameter

The **&hereTpl** template parameter should not be confused with the **&hereClass** class name parameter despite the similarity in names. For one, the **&hereTpl** is a template parameter whereas the **&hereClass** is a class name parameter. The major difference goes beyond this though. Recall that the **&hereClass** will be applied to the current document and each parent document up the document tree hierarchy. This **IS NOT** the case for the **&hereTpl** template parameter; the **&hereTpl** template will **ONLY** be used for the current document. Do not let the **"here"** in its name confuse you. Basically, if you declare this parameter in your Wayfinder call, the template will be applied to the current document only. Notice that I am talking about a single document; not documents. Therefore, we can say that this template parameter will always be applied to the same document the **&selfClass** is applied to. Being a row level template, the **&hereTpl** will be applied to any document at any level of the menu hierarchy.

As I have previously mentioned, a case may arise when you would want to have the currently selected document in your menu to be structured differently. Using the **&hereTpl** template parameter you can do just this. Note that this is a row level template and will therefore only be applied to row level menu items.

Let us now create a chunk that we will use as our **&hereTpl** template. Type the following code in the chunk, give it a name and save it. I have called my chunk **menuheretpl**.

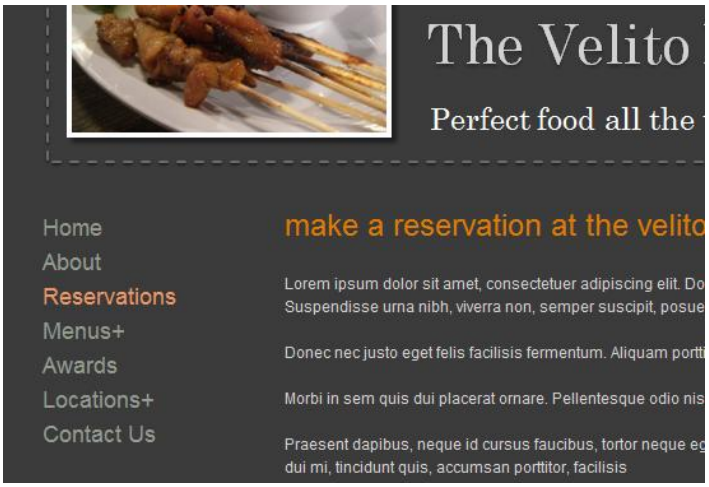
```
<li[+wf.classes+]><span class="menuhere">[+wf.linktext+]</span>[+wf.wrapper+]</li>
```

Notice that I have decided not to use the **[+wf.link+]** placeholder in the **&hereTpl** template. The reasoning is that once the user has clicked a particular menu item and hence that item is currently active, there is really no need for that menu item to be a link since clicking it will only reload the same page the user is viewing. You may decide to construct yours differently if you wish. Secondly, notice the above code has **** tags with the class **"menuhere"** enclosing the text of the menu. This is will make it easier to target the current selected menu item using CSS.

We now have to tell Wayfinder that we are using this template. We do this in the normal way by adding this information to the Wayfinder call. The modified call now looks as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &innerTpl=`innerContainer` &innerClass=`subnav`
&innerRowTpl=`submenuRows` &level=`2` &hideSubMenus=`TRUE` &parentRowTpl=`menuParents`
&parentClass=`navparents` &hereTpl=`menuheretpl`]]
```

You will also probably need to add some CSS rules to style the currently selected menu item. In my case, I created some rules for the **** tags with the class **"menuhere"**. The image below shows the output menu when the currently selected menu item is "Reservations".



Menu showing the &hereTpl applied

This is the generated source code

```
<li class="active"><span class="menuhere">Reservations</span></li>
<li class="navparents">
<a href="/velito/menus.html" title="Menus">Menus+</a>

</li>

<li   >
<a href="/velito/awards.html" title="Awards" >Awards</a>

</li>
```

menu item with &hereTpl applied to it. Notice it is surrounded by the tags with the class="menuhere" just as we constructed the &hereTpl template

You may be wondering what advantage we gain by using **&hereTpl** over **&selfClass**. One added advantage I see is that since **&hereTpl** is a template, we have the ability to structure it differently by adding additional XHTML tags to offer even greater control of the output. In the example above, by adding the **** tags, we have ensured we can easily target this part of the menu. If we wanted to have even greater control over our menu output by highlight the currently selected sub-menu item, we can use a template that is quite similar to the **&hereTpl**. This is the **&innerHereTpl** template and we will look at this next.

The &innerHereTpl template parameter

This template does exactly the same thing as the **&hereTpl** with one exception; it will only be applied to sub-menu items. Put another way, it will be applied to menu items that, for the purposes of a Wayfinder menu, are in a folder, i.e. are child documents. Note that if you have both the **&hereTpl** and the **&innerHereTpl** specified in your Wayfinder call, **&hereTpl** will only be applied if the currently selected document **IS NOT** in a sub-menu. Therefore, the **&innerHereTpl** always has precedence over the **&hereTpl** (see appendix for template processing order) and will be applied as long as the condition that the currently selected document is in a sub-menu is true.
Let us now see how this parameter works. Create a chunk with similar contents to the **&hereTpl** and give it a name. I called mine **innermenuheretpl**.

```
<li[+wf.classes+]><span class="innermenuhere">[+wf.linktext+]</span>[+wf.wrapper+]</li>
```

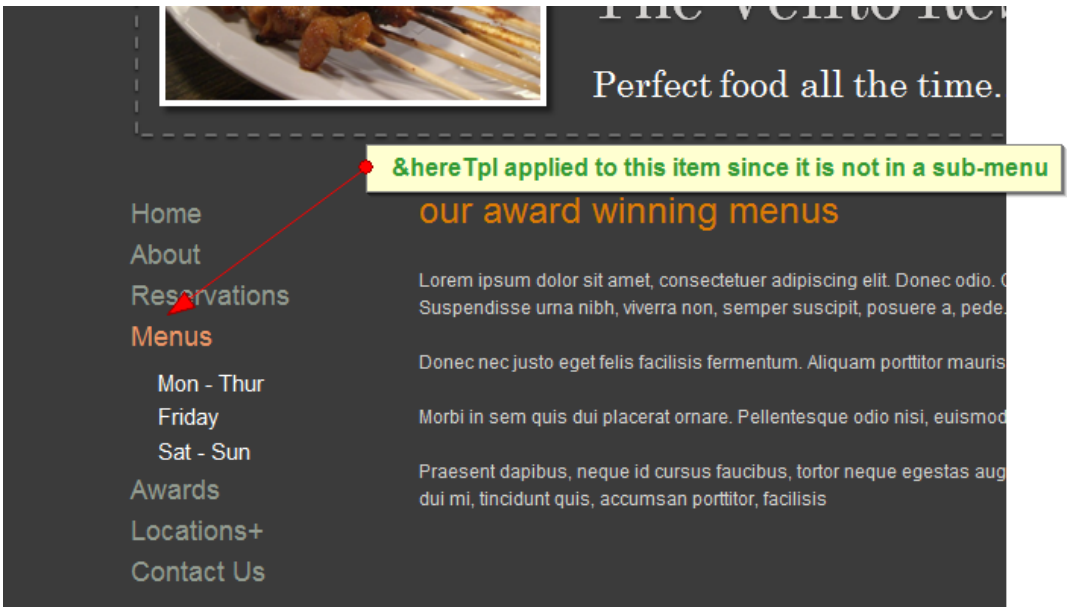
Notice that I have included the class **"innermenuhere"** so that I can style the output using CSS.

Modify the Wayfinder call as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &innerTpl=`innerContainer` &innerClass=`subnav`
&innerRowTpl=`submenuRows` &level=`2` &hideSubMenus=`TRUE` &parentRowTpl=`menuParents`
&parentClass=`navparents` &hereTpl=`menuheretpl` &innerHereTpl=`innermenuheretpl`]]
```

! Notice that I am still using the **&hereTpl** parameter in our call. This is so that I can demonstrate that when there is a situation whereby either of **&hereTpl** or **&innerHereTpl** can be used, Wayfinder will give precence to and use **&innerHereTpl**. I will also demonstrate that **&hereTpl** will "win" in case the selected document is not a sub-menu item.

With the document "Menus" selected, the menu and its source code looks as follows:



&hereTpl applied to menu item at the top level of the menu

```
<li class="navparents active"><span class="menuhere">Menus</span><ul  class="subnav">
<li>
<a href="/velito/10.html" title="Monday - Thursday">Mon - Thur</a>

</li>

<li>
<a href="/velito/11.html" title="Friday">Friday</a>

</li>

<li class="last">
<a href="/velito/12.html" title="Saturday - Sunday">Sat - Sun</a>

</li>
```

&hereTpl applied to this item since it is not a sub-menu item

With the document "Mon - Thur" selected, the menu and its source code looks as follows:

The template &innerHereTpl applied to the active sub-menu item

```
<li class="navparents active">
<a href="/velito/menus.html" title="Menus">Menus+</a>
<ul  class="subnav">
<li class="active"><span class="innermenuhere">Mon - Thur</span></li>
<li>
<a href="/velito/11.html" title="Friday">Friday</a>

</li>

<li class="last">
<a href="/velito/12.html" title="Saturday - Sunday">Sat - Sun</a>

</li>
```

the innerHereTpl applied to this sub-menu item

The &parentRowHereTpl template parameter

A template parameter that is very similar to the **&innerHereTpl** is the **&parentRowHereTpl**. This template has precedence over both the **&innerHereTpl** and the **&hereTpl** and if used will be applied to the currently selected parent document. As with the other parent row level templates, this template will only be used if the children of the parent document are also being output to the menu. If this is not the case, this template will not be used.

Construct a chunk for this template using the following code. Note that besides the different class name, this template is no different from the **&hereTpl** or the **&innerHereTpl**. This is just how I have decided to construct these templates. You may decide to construct yours differently.

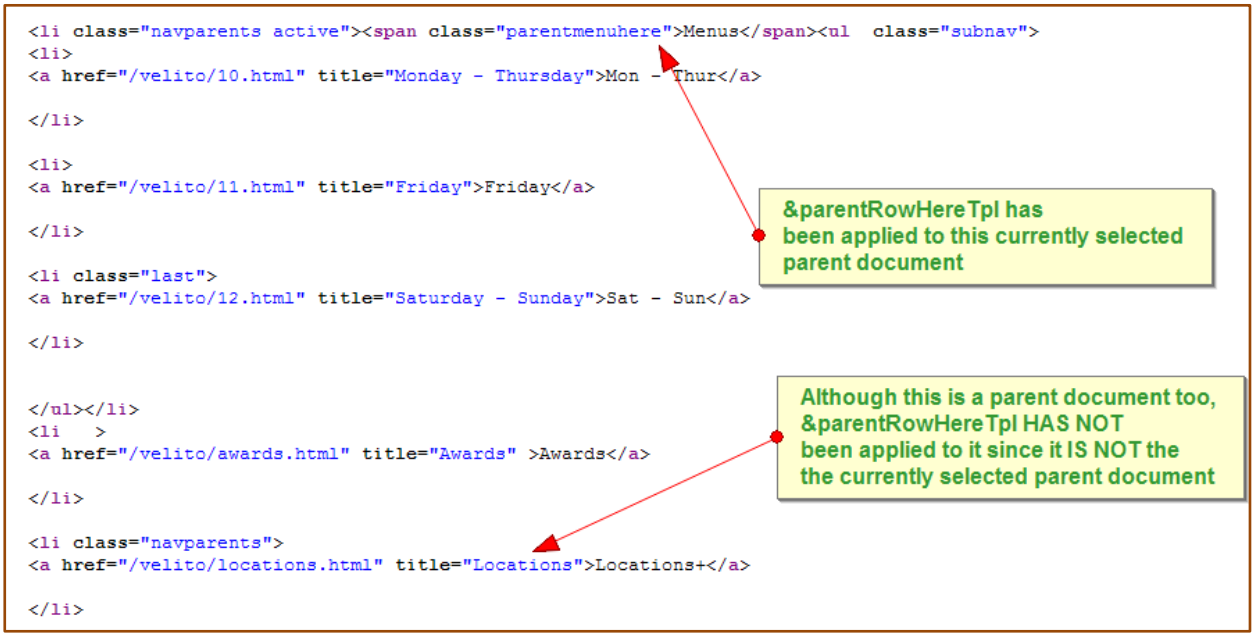
```
<li[+wf.classes+]><span class="parentmenuhere">[+wf.linktext+]</span>[+wf.wrapper+]</li>
```

I have named my chunk **parentmenuheretpl**. Let us now tell Wayfinder that we will be using this template by using the following call.

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &innerTpl=`innerContainer` &innerClass=`subnav`
&innerRowTpl=`submenuRows` &level=`2` &hideSubMenus=`TRUE` &parentRowTpl=`menuParents`
&parentClass=`navparents` &parentRowHereTpl=`parentmenuheretpl`]]
```

If you want, you can include the **&hereTpl** and/or the **&innerHereTpl** to your Wayfinder call if you would like to assure yourself that indeed this template will only be applied to the selected parent document and that it has a precedence over the former two templates.

Previewing my menu with the document "Menu" currently selected, the following code has been generated.




```
<li class="navparents active"><span class="parentmenuhere">Menus</span><ul  class="subnav">
<li>
<a href="/velito/10.html" title="Monday - Thursday">Mon - Thur</a>

</li>

<li>
<a href="/velito/11.html" title="Friday">Friday</a>

</li>

<li class="last">
<a href="/velito/12.html" title="Saturday - Sunday">Sat - Sun</a>

</li>


</ul></li>
<li    >
<a href="/velito/awards.html" title="Awards" >Awards</a>

</li>

<li class="navparents">
<a href="/velito/locations.html" title="Locations">Locations+</a>

</li>
```

As shown clearly from the image, Wayfinder has applied the **&parentRowHereTpl** to the selected parent document only and not any other.

The &activeParentRowTpl template parameter

The final and quite similar template to the **&parentRowHereTpl** is the **&activeParentRowTpl**. The difference between the two is that **&activeParentRowTpl** will be applied to all currently active parent documents in the menu. What this means is that the **&activeParentRowTpl** template will be used on the parent, grandparent, etc of the currently selected menu item. From this description you need to note that the template **IS NOT** applied to the selected menu item itself *(unless it is a parent item – hence the name, active parent);* it is applied to its parents instead, hence creating a chain of parent documents.

To test the parameter, we will construct a template for it and instruct Wayfinder to display 3 levels of our menu so that we can see how the template will be applied to parent and grandparent documents of a selected menu item. We will also set **&hideSubMenus** to **FALSE** so that child and parent documents will be open at the same time in our menu. Before we proceed, note that the

&activeParentRowTpl has a lower precedence than all the other 3 templates we have looked at in this lesson.

Here is the code we will use for the **&activeParentRowTpl** template. In my case I named the chunk **activeparenttpl**. I also included a class called "**activeparents**" for the **** tag.

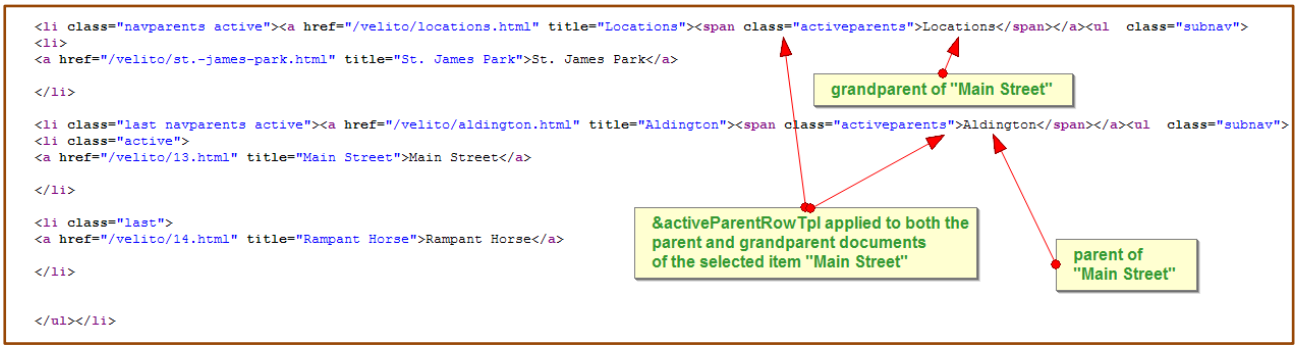
```
<li[+wf.classes+]><a href="[+wf.link+]" title="[+wf.title+]">
<span class="activeparents">[+wf.linktext+]</span></a>[+wf.wrapper+]</li>
```

Create a template with the above code. Note that I have decided to use **[+wf.link+]** placeholder since I want the active parent items to be clickable (links).

Now modify the Wayfinder call as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &outerClass=`navigation`
&rowTpl=`menuRows` &innerTpl=`innerContainer` &innerClass=`subnav`
&innerRowTpl=`submenuRows` &level=`3` &hideSubMenus=`TRUE` &parentRowTpl=`menuParents`
&parentClass=`navparents` &activeParentRowTpl=`activeparenttpl` &hideSubMenus=`0`]]
```

Your menu should now show 3 levels. Click on the menu item "Main Street" and view the source code. You should see that the **&activeParentRowTpl** template has been applied to its parent and grandparent documents, namely the "Aldington" and "Locations" menu items respectively. Here is the source code:



That does it for this lesson folks. In the next lesson we will look at of the how we can tell Wayfinder to also include the document specified as the **&startId** in its menu output. We will also look at how we can dynamically assign unique **ids** to menu items. Before that, let us recap.

Summary of lesson #5

1. Wayfinder applies the **&hereClass** class name parameter by default to the currently selected item and to each of its parents all the way up the document tree. The default value of this

parameter is "**active**". This parameter can be used to create a bread-crumb trail to show a hierarchy of active documents within the menu.

2. The **&selfClass** if used will be applied to currently active document only. This is a class name parameter.

3. If used, the template parameter **&hereTpl** will be applied to the currently active document only.

4. The template parameter **&innerHereTpl** if used will be applied to the currently selected sub-menu document.

5. The template parameter **&parentRowHereTpl** if used will be applied to the currently selected parent document.

6. If used, the **&activeParentRowTpl** will be applied to the parent, grandparent, etc documents of the currently selected menu item no matter how high up they ascend the document tree.

See you in the next lesson!

Chapter 7: Lesson #6 – i want an id and I want to be in the menu too!

Lesson #6 at a glance
&rowIdPrefix **[+wf.id+]** **&displayStart** **&startItemTpl** **&ignoreHidden**

Welcome to lesson #6!

We will be looking at some really cool stuff in this lesson so I hope you enjoy it. We will look at how we can use Wayfinder to dynamically assign an **id** to our menu items. We will also look at how we can instruct Wayfinder to include the document we specify as the **&startId** to be part of the menu it generates. OK, let us get started.

There will be instances when you will want to assign an **id** to each of your menu items. I am of course referring to the XHTML **id** attribute and not the MODx document identifier. For instance, some menus require you to style each menu item differently from the others. In such cases classes will not be of much help. You will need to assign each menu item an **id** in order to style them differently using CSS or uniquely target them using a js framework such as JQuery. Fortunately you can do this very easily in Wayfinder. Enter the **&rowIdPrefix**.

The &rowIdPrefix parameter

You can use the **&rowIdPrefix** to tell Wayfinder to assign unique **ids** to each of your menu items. This parameter is set in the Wayfinder call. To use it, you declare it in the Wayfinder call and give it a value. Since that value will be used as the prefix of the id that Wayfinder will dynamically generate, it must not start with an integer! This is because **ids** are not supposed to start with integers but with a text string. Wayfinder will then prepend that value to the MODx document id for each document that is part of the menu. This is easier to explain with an example. Assuming you declare the following in your Wayfinder call

&rowIdPrefix=`nav`

For each and every document in your menu, Wayfinder will prepend the value/prefix "**nav**" to the document's MODx document id and create a unique **id** for that menu item. For instance, using our Restaurant's menu items, the menu item "Home" will be assigned an **id** as follows:

```
<li id="nav1">Home</li>
```

In the above example, the "**nav**" is the prefix we told Wayfinder to prepend to the document ids, in this case for the document "home" it is "**1**". So, for the other documents, the **ids** will be:

```
<li id="nav2">About</li>
<li id="nav3">Reservations</li>
<li id="nav4">Menus</li>
<li id="nav5">Awards</li>
<li id="nav6">Locations</li>

<--Etc, etc.-->
```

Now, the **&rowIdPrefix** parameter is not used alone. It always requires a particular Wayfinder placeholder. This is the **[+wf.id+]**.

The [+wf.id+] placeholder

The unique **id** created by Wayfinder using the **&rowIdPrefix** will be inserted where the **[+wf.id+]** placeholder appears in you Wayfinder template. Hence, you need to be careful where you place this placeholder for your menu items' **ids** to be placed in the correct position. You also need to place it in the correct Wayfinder template. For instance, you probably do not want to have it in the **&outerTpl** since this template is not used to create menu row level items. It is also not very helpful to place it in a template that will only be used in certain cases and not others, for instance within the **&hereTpl**. You therefore want to probably use this placeholder in your **&rowTpl** or **&innerRowTpl**.

Let us now look at an example of the above two parameters in action.

Assigning unique ids to menu items

We will be revisiting the **&rowTpl** template we created in lesson #2. Below is the code we have in the chunk we named **menuRows** that forms the **&rowTpl** template. See where the **[+wf.id+]** placeholder is; that is where Wayfinder will place the unique **ids** for each of the documents that forms our menu.

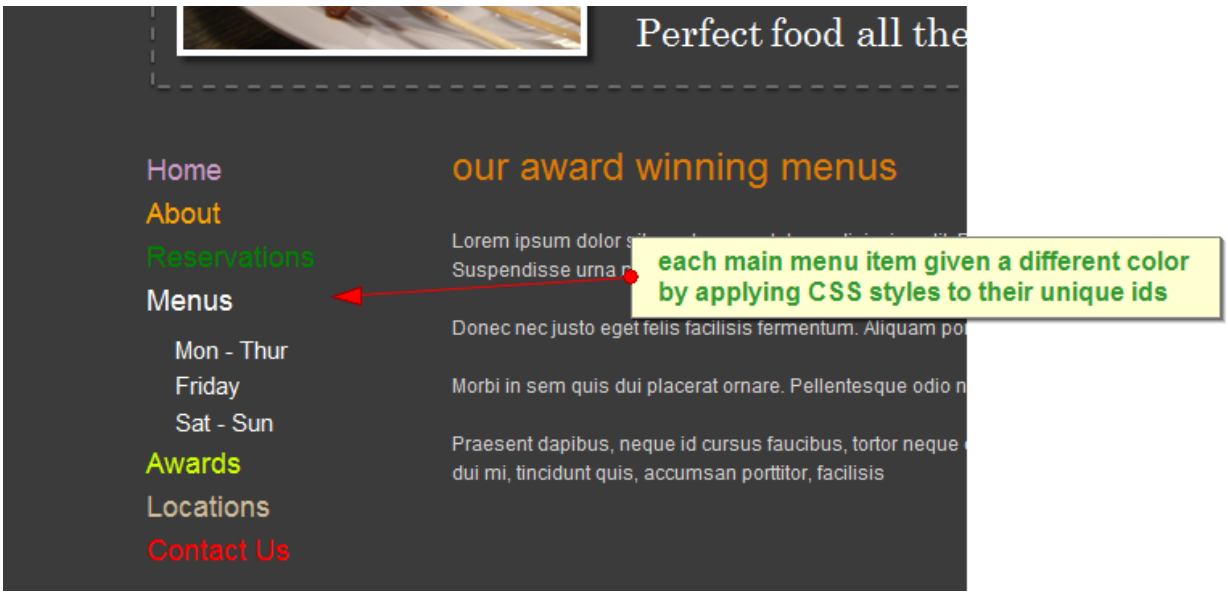
```
<li [+wf.id+] [+wf.classes+] >
<a href="[+wf.link+]" title="[+wf.title+]" [+wf.attributes+]>[+wf.linktext+]</a>
[+wf.wrapper+]
</li>
```

Modify the Wayfinder call so that it looks like follows. I have decided to strip it down to the bare essentials since I do no longer need to use some of the parameters. The prefix that will be prepended to the unique **id** for each document in the menu will be "**nav**". If you wish you can use a different prefix.

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`2`
&hideSubMenus=`TRUE` &rowIdPrefix=`nav`]]
```

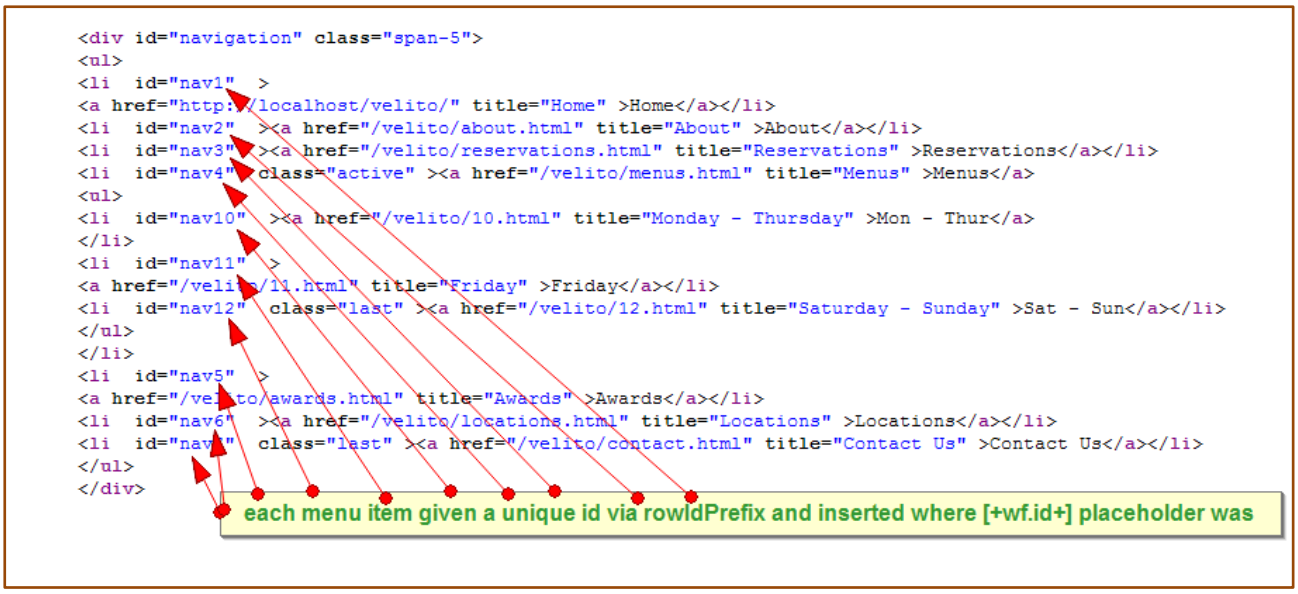
Since we know the **ids** that Wayfinder will assign our menu items, we can go ahead and create some CSS styles for the menu items by using those **ids** as selectors. In this example, we will only do some basic styles, i.e. give each of the main menu items a different colour. However, we have the possibility of creating more complex menus than this, e.g. the so called "fisheye" menu. In chapter 15 we will see other examples of using the **&rowIdPrefix** to assist in the creation of more complex menus.

The document ids of our menu items range from 1 ("Home") to 12 ("Saturday - Sunday") so the **ids** created will range from "**nav1**" through to "**nav12**" for each of our menu items respectively. I have added the **ids** of the main menu items (documents 1 – 7) to my CSS style and given each **id** a different background colour. The resultant menu looks like follows:



Menu showing the application of the &rowIdPrefix parameter

And here is the source code for the menu. Notice the unique **ids** added by Wayfinder.



The &displayStart parameter

As we learnt in lesson #1, by default the document specified as the **&startId** is not included in the menu generated by Wayfinder. For instance, using the documents in our Restaurant's website, if we specified **&startId=`6`**, only the child documents of the document "Locations" will be output to our menu. However, if we wanted the document "Locations" itself to be part of the menu, e.g. as the menu's heading, we can override this behaviour by using two Wayfinder parameters in conjunction. The first of these is the **&displayStart**. This parameter can take only two values, **TRUE** or **FALSE** and is set to **FALSE** by default thus does not need declaring unless you want to use the parameter wit ha value of **TRUE**. Setting this parameter to **TRUE** tells Wayfinder to also output to the menu the document that has been specified as the **&startId**. This is of course with the exception of the root document! So, this will only work if the **&startId** value is greater than 0.

If this parameter is set to **TRUE**, the **&outerTpl** will no longer be regarded as the top most container of the menu. Instead, there is another Wayfinder template that will be used for this purpose. This is the **&startItemTpl**. In Wayfinder's template processing order, the **&startItemTpl** has precedence over all the other Wayfinder templates.

&startItemTpl

As its name suggests, this template will be used as the template of the start item in the menu. Wayfinder comes with a default **&startItemTpl**. You are however free to construct your own. As with the other parameters with default values, if you do not specify the chunk containing this template, Wayfinder will use its default one. The default **&startItemTpl** is:

```
<h2>[+wf.linktext+]</h2>[+wf.wrapper+]
```

Notice the **<h2></h2>** tags. You can of course use different tags if you wish. They should be valid XHTML though for your code to validate. Notice too that only two very essential placeholders are used here. The use of the **[+wf.linktext+]** is pretty obvious. The **[+wf.wrapper+]** is the more

important of the two. Without it no inner content of the menu will be displayed. Without it you would end up only with the content you entered between the **<h2></h2>**!

We are going to create a **&startItemTpl** which is a slight modification of the default one. If you decide to create your own the most important thing to remember is not to forget the **[+wf.wrapper+]** since without it the contents of your menu will not be displayed.

Create a chunk, give it a name, and copy the following code in it. I named mine **menustart**.

```
<h2 class="menustart">[+wf.linktext+]</h2>[+wf.wrapper+]
```

Notice that I have added a class="**menustart**" so that I can easily target and style the content of the template's **<h2>** tags.

Now modify your Wayfinder call so that it looks like the following:

```
[[Wayfinder? &startId=`6` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`2`
&hideSubMenus=`TRUE` &displayStart=`TRUE` &startItemTpl=`menustart`]]
```

Save the template and if you wish add some CSS styles to style the menu's start item.

As per the conditions mentioned above, please note that here our **&startId** has a value greater than one. Since **&displayStart** is also set to TRUE, we expect that our menu will start from and include the document with MODx document id #6, i.e. "Locations".

This is the resultant menu using the above code.

The &displayStart and &startItemTpl in action in the menu

And here is the source code of the menu. Notice that the document "Locations" is enclosed in **<h2>** tags.

```
<div id="wrapper" class="container">
<div class="span-24 last"><h1 id="branding"></>The Velito Restaurant</h1></div>
<div id="navigation" class="span-5"><h2 class="menustart">Locations</h2><ul>
<li   >
<a href="/velito/st.-james-park.html" title="St. James Park" >St. James Park</a>

</li>

<li   class="last" >
<a href="/velito/aldington.html" title="Aldington" >Aldington</a>

</li>


</ul>

</div>
```

with &displayStart set to true, the &startId document is displayed in the menu and wrapped in the &startItemTpl template

Note that the first level of this menu is not the "Locations" menu item since it is being used a category or menu heading. The child documents are actually the main menu items.

It is also important to remember that since we can only have one value for **&startId** per Wayfinder menu, and that setting the value of **&displayStart=`TRUE`** will cause Wayfinder to construct a menu starting from and including that specified **&startId** document, it logically follows that the **&startItemTpl** can only be used once per Wayfinder menu.

The last parameter we will look at in this lesson is the **&ignoreHidden** parameter.

The &ignoreHidden parameter

This parameter can only take boolean values, **TRUE** or **FALSE**. It is set to **FALSE** by default and hence does not need to be declared in the Wayfinder call.

When Wayfinder constructs menus, it checks to see whether the documents to be included in the menu output have their "**show in menu**" box ticked or un-ticked, i.e. whether set or not set. If a document has its "**show in menu**" setting un-ticked, Wayfinder will by default not include that document in the menu output. In other words, we can think of this as Wayfinder asking itself, "should I ignore the fact this document does not want to be included in the menu and include it anyway or not?". By default, Wayfinder answers that question as "**FALSE**"; meaning, no, do not ignore the fact that "**show in menu**" is un-ticked. You have probably guessed what will happen if we use this parameter and give it a value of **TRUE**; if in the Wayfinder call we specify that **&ignoreHidden=`TRUE`**, Wayfinder will go ahead and include documents with their "**show in menu**" boxes un-ticked in the menu output. This parameter at times confuses people so it is worth understanding what it does. Let us try it out and see how it works.

Edit the documents "Awards" and "Reservations" and "un-tick" their "**show in menu**" boxes saving the documents in turn. Hence by default Wayfinder will not include these two documents in the output menu. Preview you menu; you will find this is true. Now let us force Wayfinder to include these documents in the menu despite the fact that we have indicated that we do not want them shown in the menu. Modify the Wayfinder call as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`2`
&hideSubMenus=`TRUE` &ignoreHidden=`TRUE`]]
```

Please note that in the above code we have reverted to our original **&startId** and are also no longer using the **&displayStart** parameter.

Preview the menu. You should now see that the documents "Awards" and "Reservations" despite having their "**show in menu**" boxes un-ticked have been output to the menu.

That is it folks for this lesson. Before we move on, let us do a recap.

Summary of lesson #6

1. By using the **&rowIdPrefix** you can instruct Wayfinder to dynamically create unique **ids** for menu items. The **ids** will be inserted where the **[+wf.id+]** placeholder is located in the Wayfinder template. To get the correct output, consideration has to be given with respect to correctly placing this placeholder in the "right" template and in the right location.

2. If **&displayStart** is used and set to **TRUE**, Wayfinder will include the document specified as the **&startId** in the menu output as long as the **&startId** value is greater than **0** (i.e. not the root). Wayfinder will use the **&startItemTpl** template to construct the container for this start item of the menu. It will use its default **&startItemTpl** unless you create and specify one.

3. Using the parameter **&ignoreHidden** and setting its value to **TRUE** will cause Wayfinder to include in the menu output the relevant documents whose "**show in menu**" boxes are un-ticked.

We have now covered all the template parameters available in Wayfinder. See you in the next lesson where we will be looking at some class name parameters and how to include or exclude certain documents from the Wayfinder menu output.

Chapter 8: Lesson #7 – hey look; I am included in the first class!

Lesson #7 at a glance
&firstClass **&lastClass** **&levelClass** **&includeDocs** **&excludeDocs**

Welcome to lesson #7! We have covered half the guide already.

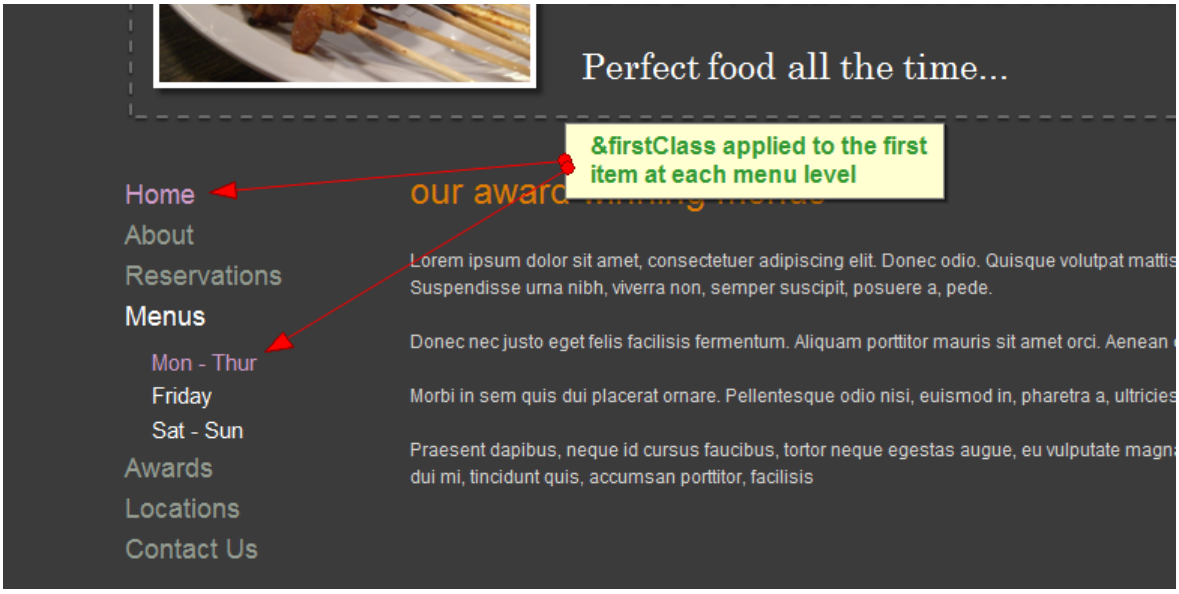
This lesson will be pretty easy and relatively short. We will be looking at 3 class name parameters and two other parameters that allow us to fine tune the menu output. Let us get started.

The &firstClass class name parameter

If you ever wanted to apply a CSS class to the first item in your menu, you can use the **&firstClass** class name parameter. When used, this parameter will be applied to all the first items in your menu at all levels of the menu. The parameter is therefore applied to the row items of your menu. The **&firstClass** parameter has no default value so will have to be declared in the Wayfinder call in order to be applied. You need to remember that the first item in your menu can change depending on how and with what value you order the menu output. We covered this in lesson #2. Let us test this parameter by modifying our Wayfinder call as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`2`
&hideSubMenus=`TRUE` &firstClass=`first`]]
```

In the above call, the **&firstClass** has been assigned the value "**first**". In my example output below, I have applied some styling to this class in order for the first item to be styled differently from the rest of the menu items.



The &firstClass applied to the menu


```
<li   class="first" >
<a href="http://localhost/velito/" title="Home" >Home</a>

</li>

<li   >
<a href="/velito/about.html" title="About" >About</a>

</li>

<li   >
<a href="/velito/reservations.html" title="Reservations" >Reservations</a>

</li>

<li   class="active" >
<a href="/velito/menus.html" title="Menus" >Menus</a>
<ul>
<li   class="first" >
<a href="/velito/10.html" title="Monday - Thursday" >Mon - Thur</a>

</li>

<li   >
<a href="/velito/11.html" title="Friday" >Friday</a>

</li>

<li   class="last" >
<a href="/velito/12.html" title="Saturday - Sunday" >Sat - Sun</a>

</li>
```

The &lastClass class name parameter

There **&lastClass** class name parameter is very similar to the **&firstClass** except for two differences. One, it will be applied to the last items at each menu level and; two it is applied by default and has the value "**last**". If you look at the source code of the menus we have been generating, you will notice that the last items at each menu level have a class "**last**" applied to them. Wayfinder has been doing this by default since we did not instruct it otherwise. In order to change or not use this

parameter, you would have to tell Wayfinder this just as we did in lesson #5 when we looked at the **&hereClass** class name parameter that is also applied by default.

If you would like to use this parameter with its default value you do not have to do anything other than give the class "**last**" some CSS rules in your style sheet. If you do not want Wayfinder to output this parameter you will have to specify this as follows in your Wayfinder call:

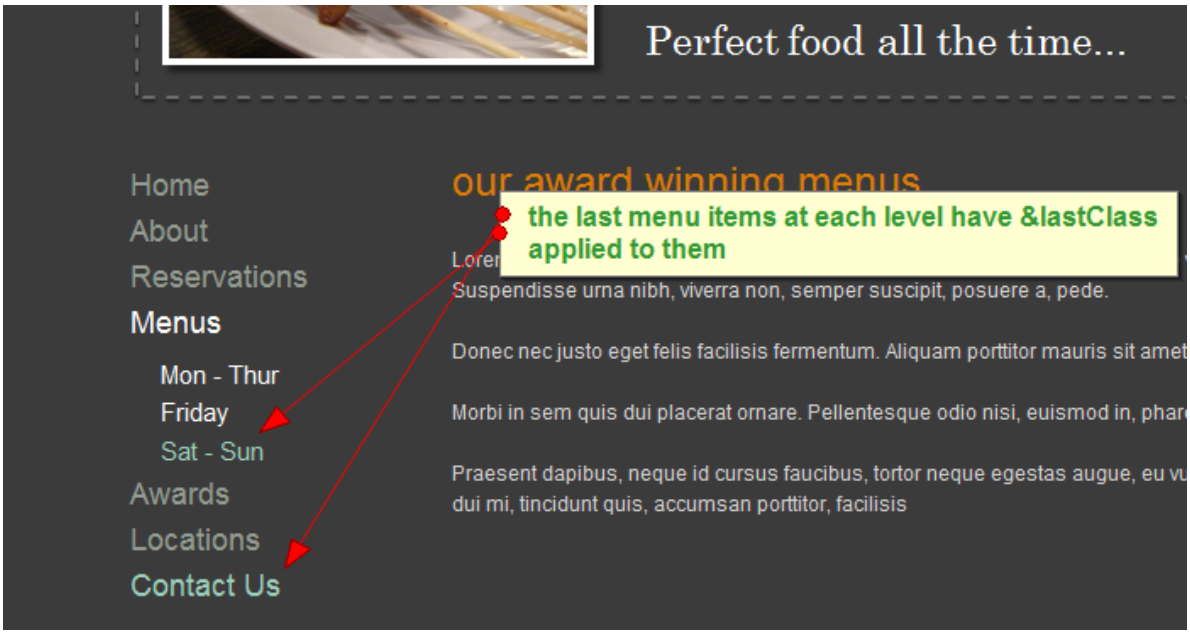
&lastClass=``

If on the other hand you want to use this parameter but assign it a different value you would do this in the Wayfinder call. As an example, let us give this parameter the value "**lastitem**"

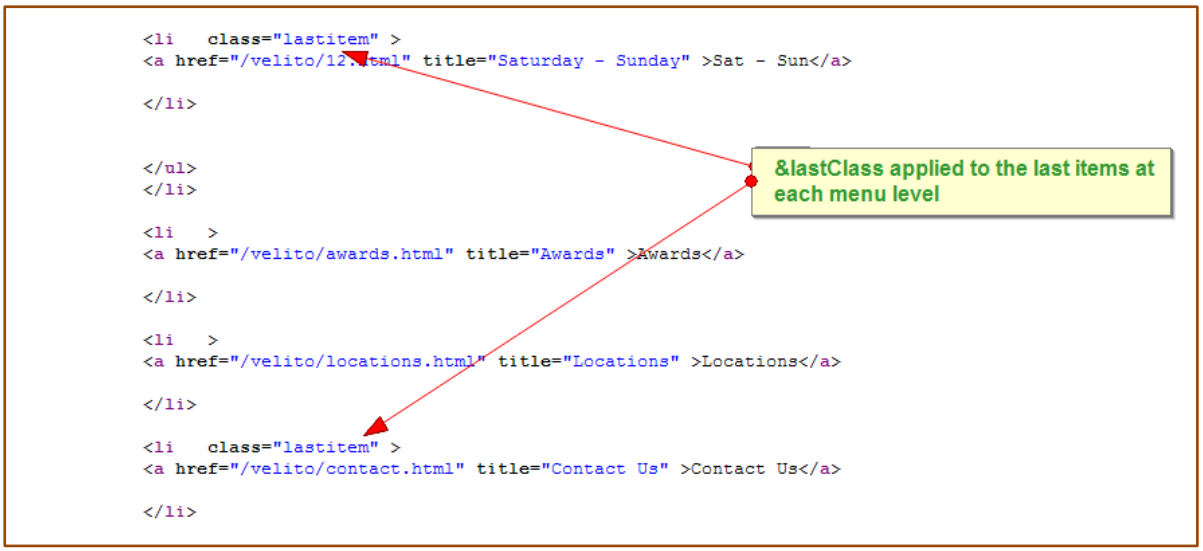
&lastClass=` lastitem `

The **&lastClass** can be handy say in a horizontal menu whose items have a right border that acts as a separator between the menu items. For the last menu item, you would probably not want a right border applied. Let us test this parameter by assigning it a different value other than the default.

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`2`
&hideSubMenus=`TRUE` &lastClass=`lastitem`]]
```



The &lastClass applied to the menu




```
<li    class="lastitem" >
<a href="/velito/12...tml" title="Saturday - Sunday" >Sat - Sun</a>

</li>


</ul>
</li>

<li    >
<a href="/velito/awards.html" title="Awards" >Awards</a>

</li>

<li    >
<a href="/velito/locations.html" title="Locations" >Locations</a>

</li>

<li    class="lastitem" >
<a href="/velito/contact.html" title="Contact Us" >Contact Us</a>

</li>
```

The final class we will be looking at in this lesson is the **&levelClass**.

The &levelClass class name parameter

This parameter is somewhat special. It has no default value so in order to be used it has to be specified in the Wayfinder call. If the **&levelClass** parameter is used, it will be applied to each item denoting each items row level with the level number added to the class specified as the value of the parameter. For instance, let us say we specify the parameter as follows:

&levelClass=`level`

The above instruction will cause Wayfinder, for each menu item, to prepend the class "level" to the items row level number. So, for items in the 1st level of the menu, the class "**level1**" will be added; for 2nd level items, the class "**level2**" will be added; for items in the 3rd level, "**level3**" will be added, and so on and so forth for menu items at other levels.

Hence, in our Restaurant website, if we used this parameter, we would expect that menu items such as "Home", "About", "Locations", etc will have "**level1**" applied to them as a class. Menu items such as "St. James Park" and "Aldington" would have the class "**level2**" applied to them, etc. Let us try it out by modifying the Wayfinder call as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`2`
&hideSubMenus=`TRUE` &levelClass=`level`]]
```

Here is the source code of the menu output by Wayfinder using the above call.


```
<li    class="level1 active" >
<a href="/velito/menus.html" title="Menus" >Menus</a>
<ul>
<li    class="level2" >
<a href="/velito/10.html" title="Monday" >

</li>

<li    class="level2" >
<a href="/velito/11.html" title="Friday" >Friday</a>

</li>

<li    class="last level2" >
<a href="/velito/12.html" title="Saturday - Sunday" >Sat - Sun</a>

</li>


</ul>
</li>

<li    class="level1" >
<a href="/velito/awards.html" title="Awards" >Awards</a>

</li>

<li    class="level1" >
<a href="/velito/locations.html" title="Locations" >Locations</a>
```

Notice the classes applied to the menu items at different levels of the menu. You need to note that the level of a menu item is counted relative to what you have specified as the **&startId** of the menu and not relative to the MODx document tree. This means that whilst a document may be on the 3rd level of the MODx document hierarchy, it could as well be at level 1 of Wayfinder's menu levels. We covered this in details in lesson #1 so I will not go into any more detail about it.

You can of course apply CSS rules to the different level classes as you wish.

The &includeDocs parameter

A case may arise where you want all or a number of your documents to be enabled to be shown in the menu while at the same time to output to the menu only some certain documents and not others. In such as case, un-ticking the "**show in menu**" boxes of the documents you do not want to be output to your menu may not be the answer since you may want those documents to be output to a different menu at some other section of your website. The answer to this situation is the **&includeDocs** parameter. Using this parameter, you can instruct Wayfinder to only include the documents you specify as the values for this parameter in the menu output. The parameter will accept a comma delimited list of MODx document ids. The **&startId** parameter is still required even in this case (remember in lesson #1 we learnt it is always required).

Let us say for instance that in our Restaurant website we want to only show the documents "Home", "Menus" and "Contact Us" in the menu output. We could do this as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`2`
&hideSubMenus=`TRUE` &includeDocs=`1,4,7`]]
```

This parameter therefore acts as a filter. You need to remember that whilst the order of the values in the parameter do not matter, i.e. **1,4,7** or **7,4,1** etc, the inclusion of child documents in a multi-level menu require the inclusion of their parent documents. Therefore, we could not, for instance, include the document "Aldington" without including its parent document "Locations". Try out the above code to see how Wayfinder will output only these documents to the menu.

The &excludeDocs parameter

You have probably guessed what this parameter does. Whilst it shares the same characteristics as its brother parameter above, it does the exact opposite of the **&includeDocs** parameter. The **&excludeDocs** parameter when used will instruct Wayfinder to exclude the documents specified in this parameter from the menu output. All the other documents you want in the menu will be output to the menu.

For instance, let us say we want to exclude the menu items "Reservations", "Saturday - Sunday" and "Awards" from the output. We could write the Wayfinder call as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`2`
&hideSubMenus=`TRUE` &excludeDocs=`3,5,12`]]
```

Preview the menu; you will see that all the other documents up to two levels deep are included in the menu with the exception of the ones we excluded.

Whilst these two filter parameters can be very handy, for a large site you would benefit from writing down notes why you included or excluded certain documents from the menu output. These would help when you or someone else comes to edit the site after a while.

There you are folks; we are done with this lesson. In the next one we will look at how we can tell Wayfinder to sort menu items using certain criteria and how we can tell from the menu output how many child documents are in a particular parent menu item. Let us recap before we move on.

Summary of lesson #7

1. You can use the **&firstClass** to apply a specified class to all the first items at all levels of your menu.

2. The **&lastClass** is applied by default to all the last items at all levels of the men. You can override this behaviour by giving the class an "empty" value or you could even assign a different value of your choice to this class.

3. The **&levelClass** can be used to prepend a specified value to a number corresponding to the row level of respective menu items.

4. To include only certain documents in your menu, you can use the **&includeDocs** parameter. Acceptable values are comma delimited list of MODx document ids.

5. To exclude certain documents from the menu, you can use the **&excludeDocs** parameter. This parameter shares the same characteristics as the **&includeDocs** but behaves differently. It excludes the specified documents from the menu.

OK, see you in the next lesson!

Chapter 9: Lesson #8 – order, order...how many children you got?

Lesson #8 at a glance
&sortBy **&sortOrder** **&showSubDocCount** **[+wf.subitemcount+]**

Welcome to lesson #8!

Similar to the previous lesson, this one too will be pretty easy to follow and relatively short. We will be looking at how to sort the menu output by Wayfinder. We will also look at how to show the number (count) of child documents that are contained in each parent document item in the menu.

The &sortBy parameter

By default, Wayfinder will sort the menu items output to the menu using the **menuindex**. You therefore do not need to declare this parameter if you are happy with Wayfinder's default sorting. However, if you want Wayfinder to sort the menu items using another criterion, you can choose from the following MODx document fields. Note that the menu will be sorted on a level by level basis.

pagetitle	Page title will be used as sorting criterion
id	The document id will be used as sorting criteria
introtext	Summary/Introtext text will be used to sort menu
menuindex	Sorting according to menu index field (default sorting criteria)
description	Description text will be used to sort menu
hidemenu	Hidden from menu items sorted to bottom of menu
parent	??
isfolder	Parents documents output at the bottom of the menu
published	??
alias	The document alias will be used to sort menu output
longtitle	Long title text will be used to sort menu output
type	Type output ("document" if document and "reference" if weblink) will be used as sorting criteria
template	Sort according to template number
random	If Wayfinder call is uncached, menu will be sorted randomly every time the page is loaded

If we used this Wayfinder call, the image below would be the resultant menu.

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`2`
&hideSubMenus=`TRUE` &sortBy=`pagetitle`]]
```

Menu sorted using &sortBy=`pagetitle`

The &sortOrder parameter

There **&sortOrder** parameter will simply order the menu items in an ascending or descending fashion using the criterion you selected for the **&sortBy** parameter. The values it takes are "**ASC**" (ascending) and "**DESC**" (descending). The default value is "**ASC**" and therefore does not need to be declared in the Wayfinder call. However, if you would like a descending order, you would declare this as follows

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`2`
&hideSubMenus=`TRUE` &sortOrder=`DESC`]]
```

The &showSubDocCount parameter

You can use this parameter to show the number (count) of child documents contained within each menu item. The **&showSubDocCount** parameter only accepts boolean values, "**TRUE**" or "**FALSE**" and is set to by "**FALSE**" default. In order to use it, you have to specify it in the Wayfinder call and give it the value "**TRUE**". If a menu item has no child documents or a parent document has either unpublished or not-to-be-shown in menu children, Wayfinder will return an output of "**0**". If the menu item has children, Wayfinder will return the count of the number of published to-be-shown-in-menu child documents contained within that parent menu item. The count of child documents returned by Wayfinder is placed within the **[+wf.subitemcount+]** placeholder.

The [+wf.subitemcount+] placeholder

The output of the parameter **&showSubDocCount** is inserted where you have the **[+wf.subitemcount+]** placeholder in your Wayfinder template. Similar to the other placeholders, you need to place this placeholder in the correct position within the correct template. In my case,

since I am using the **&rowTpl** to construct the row items of my menus, I have put this placeholder in this template as shown below to illustrate its use.

```
<li [+wf.classes+] >
<a href="[+wf.link+]" title="[+wf.title+]" [+wf.attributes+]>[+wf.linktext+]
[+wf.subitemcount+] </a>
[+wf.wrapper+]
</li>
```

Modify your **&rowTpl** as shown above and your Wayfinder call as shown below.

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`3`
&hideSubMenus=`TRUE` &showSubDocCount=`TRUE`]]
```

Preview the menu; you will see that Wayfinder has output the count of the number of child documents found within each menu item and at all levels of the menu. You could have of course enclosed the **[+wf.subitemcount+]** placeholder in say {} or () as a visual aid to the user. Try un-publishing and/or setting some child documents not-to-be-shown in the menu and see how this affects the output count of child documents.

Menu showing the effect of using &showSubDocCount=`TRUE`

There you have it folks. In the next lesson we will look at how we can influence how Wayfinder outputs our menu's links. Here is a brief recap of what we covered in this lesson

---

Summary of lesson #8

You can use the **&sortBy** to sort your Wayfinder menu items. This parameter accepts several values the default of which is "**menuindex**".

1. You can use **&sortOrder** to either sort your menu items in an ascending (**ASC**) or descending **DESC** order. The default value is "**ASC**".

2. The **&showSubDocCount** can be used to instruct Wayfinder to output the count of the number of child documents within each menu item. This parameter accepts only two values, **TRUE** or **FALSE** and is set to **FALSE** by default.

3. The output returned by Wayfinder with respect to the **&showSubDocCount** is inserted in the placeholder **[+wf.subitemcount+]**.

See you in the next lesson!

Chapter 10: Lesson #9 – linking with class

Lesson #9 at a glance
&useWeblinkUrl **&webLinkClass** **&fullLink**

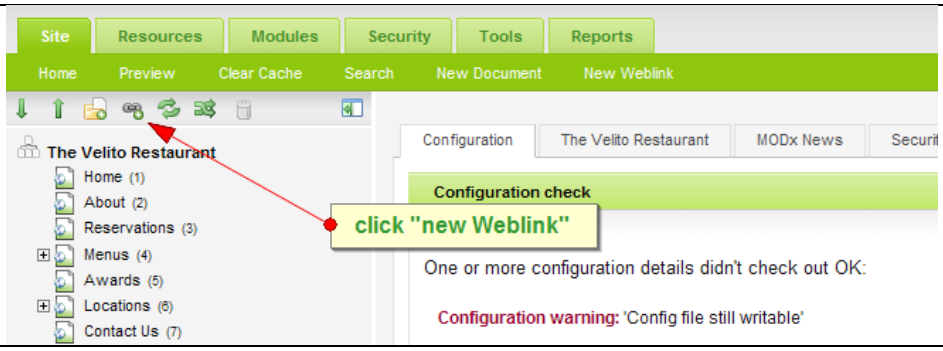
Welcome to lesson #9!

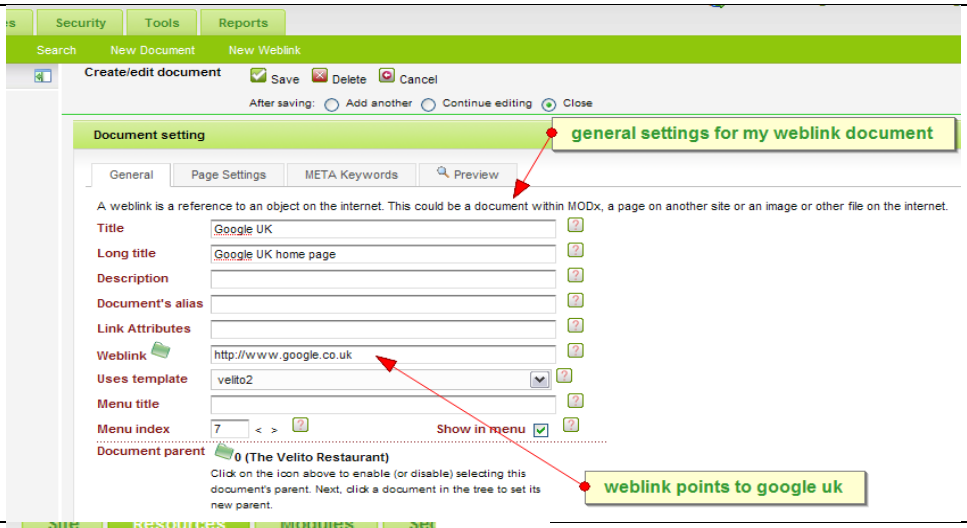
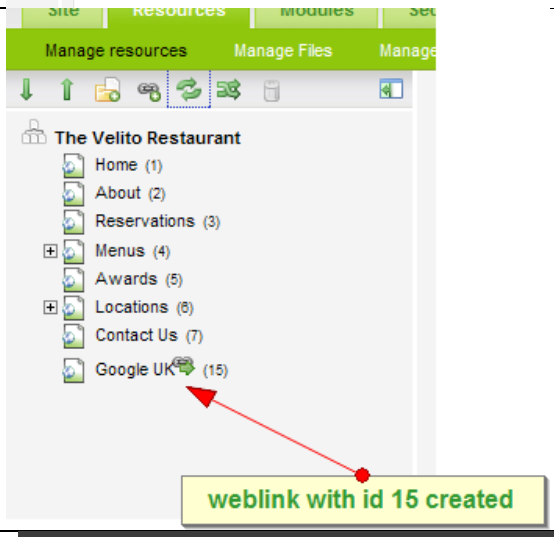
In this lesson we are going to be dealing with 3 parameters that influence how Wayfinder outputs links.

The &useWeblinkUrl parameter

By now you probably know that in MODx you can create two types of documents; "normal" documents and documents that are weblinks. The **&useWeblinkUrl** parameter is concerned with weblinks. This parameter only accepts the Boolean values "**TRUE**" or "**FALSE**" and is set to "**TRUE**" by default. When creating a weblink document, you have to enter a URL that you wish that document to reference, i.e. the weblink. At the same time, MODx will add this document to your document tree and also assign it an **id**. If this document is included in your menu item, clicking it will take you to the URL you entered as the weblink. By default, Wayfinder will output the link you specified in the weblink instead of the normal MODx document link. For instance, if your weblink was specified as http://www.google.co.uk, this is what Wayfinder will output instead of that document's **id**. However, using this parameter, you can instruct Wayfinder to output that documents MODx link instead. Note though that clicking on that document in the Wayfinder menu will still take you to the URL you specified as the weblink. Therefore, the only difference in setting the **&useWeblinkUrl** parameter to "**FALSE**" is that the link will be shown as a MODx document link whilst clicking on the menu item will take the user to the correct and intended weblink. This is easier to understand if you see an example so let us create a weblink document.

Create a new weblink document and fill in the necessary fields. The images below illustrate the steps I followed and the output with this parameter not declared, i.e. letting Wayfinder use its default value.

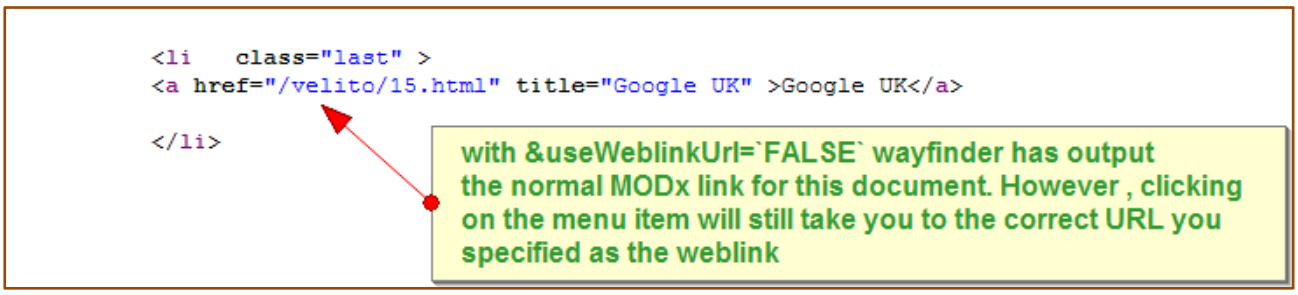
Creating the weblink	

The weblink settings	
Weblink created	
Weblink in Wayfinder menu	
Weblink in Wayfinder menu source code	

Now set the parameter to "**FALSE**" as follows

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`2`
&hideSubMenus=`TRUE` &useWeblinkUrl =`FALSE`]]
```

Preview the results; clicking on the menu item for the weblink document will still take you to the correct URL, Google UK in my case. However, look at the source code. You will see that Wayfinder has output the document's normal MODx link instead. Magic, eh?

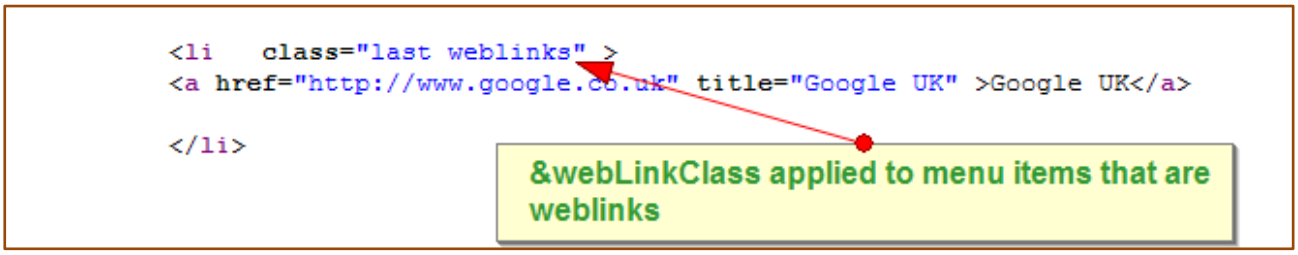


The &webLinkClass class name parameter

As the name suggests, the **&webLinkClass** will be applied to all documents that are weblinks in the menu output by Wayfinder. There is no default value for this parameter so to use it you will have to specify it in the Wayfinder call. Let us see how this works. Modify the Wayfinder call as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`2`
&hideSubMenus=`TRUE` &webLinkClass=`weblinks`]]
```

In the above example, I have assigned the value "weblinks" to my **&webLinkClass** parameter. You will obviously need to create some CSS rules for this class if you wish to style the weblinks differently from the "normal" menu items. The output of the above call results in the following source code with respect to the weblink menu item:



The &fullLink parameter

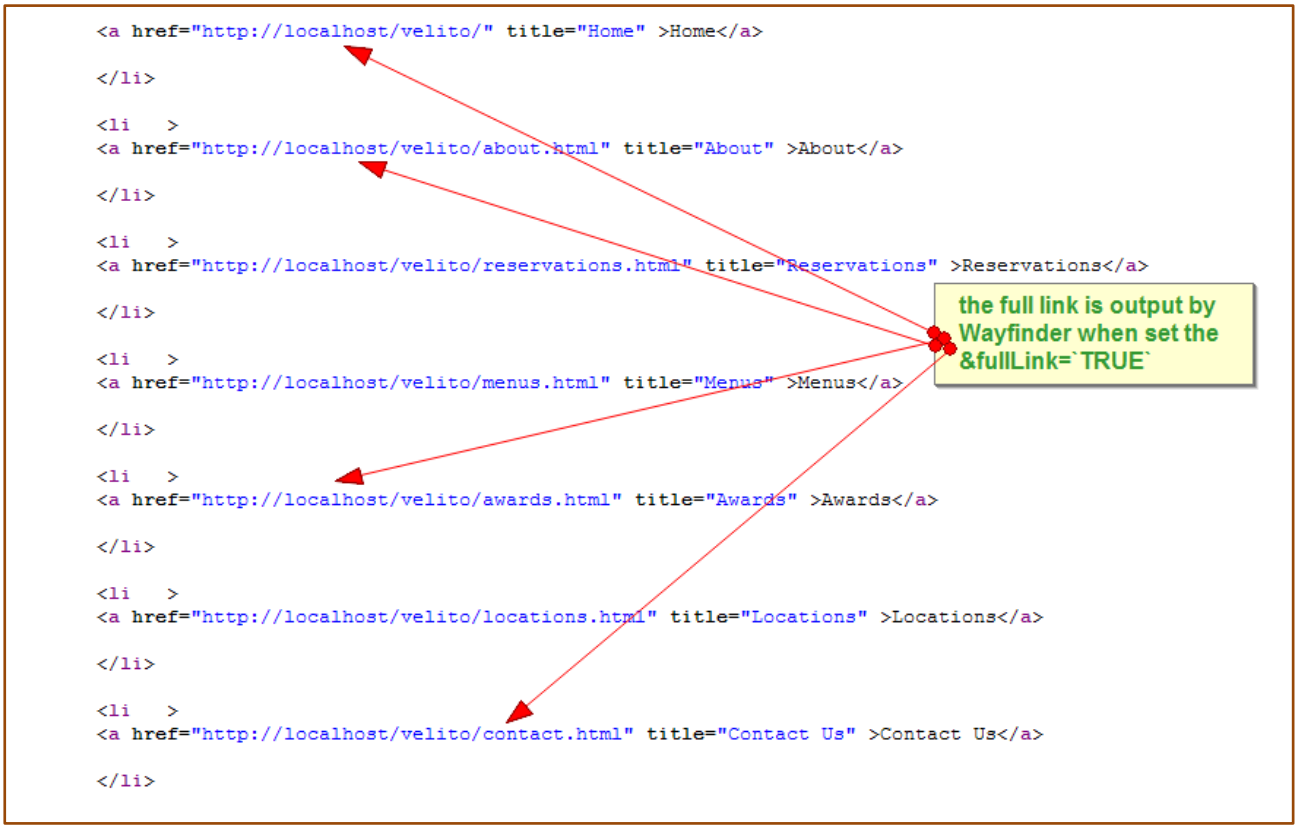
If this parameter is set to "**TRUE**", Wayfinder will output the full URL of the menu items instead of the relative URL. The default value is "**FALSE**". Have a look at the source code of the menu items. You

will notice that they are all relative, e.g. for the menu item "About", the URL is **"/velito/about.html"**. Yours could be different depending on whether you are using friendly URLs or not, etc.

To test this parameter, modify the Wayfinder call as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`2`
&hideSubMenus=`TRUE` &fullLink=`TRUE`]]
```

The following is the resultant source code generated by Wayfinder for our Restaurant's website when **&fullLink** is set to "**TRUE**"




```
<a href="http://localhost/velito/" title="Home" >Home</a>

</li>

<li   >
<a href="http://localhost/velito/about.html" title="About" >About</a>

</li>

<li   >
<a href="http://localhost/velito/reservations.html" title="Reservations" >Reservations</a>

</li>

<li   >
<a href="http://localhost/velito/menus.html" title="Menus" >Menus</a>

</li>

<li   >
<a href="http://localhost/velito/awards.html" title="Awards" >Awards</a>

</li>

<li   >
<a href="http://localhost/velito/locations.html" title="Locations" >Locations</a>

</li>

<li   >
<a href="http://localhost/velito/contact.html" title="Contact Us" >Contact Us</a>

</li>
```

That is all for this lesson. In the next lesson we will look at some more uses of MODx document fields within Wayfinder. We will also look at how we can incorporate MODx Template Variables (TVs) within Wayfinder calls.

Before we wrap up, here is a brief recap of what we covered in this lesson

---

Summary of lesson #9

1. When set to "**FALSE**" the **&useWeblinkUrl** parameter will output the MODx link of a weblink document rather than the weblink URL itself. Clicking on such menu items, however, will still take the user to the correct and intended weblink URL.

2. The **&webLinkClass** class name parameter can be used to apply a specified class for all menu items that are weblinks rather than "normal" MODx documents. This would enable the application of CSS rules to specifically target menu items that are weblinks.

3. Wayfinder will normally output relative links of the menu items. By setting the parameter **&fullLink** to "**TRUE**" you instruct Wayfinder to output the full URLs of the menu items. The default value is "**FALSE**"

See you in the next lesson!

Chapter 11: Lesson #10 – fields, fields and more fields

Lesson #10 at a glance
[+wf.docid+] [+wf.description+] [+wf.introtext+] [*document_field_name*] [+template_variable_name+]

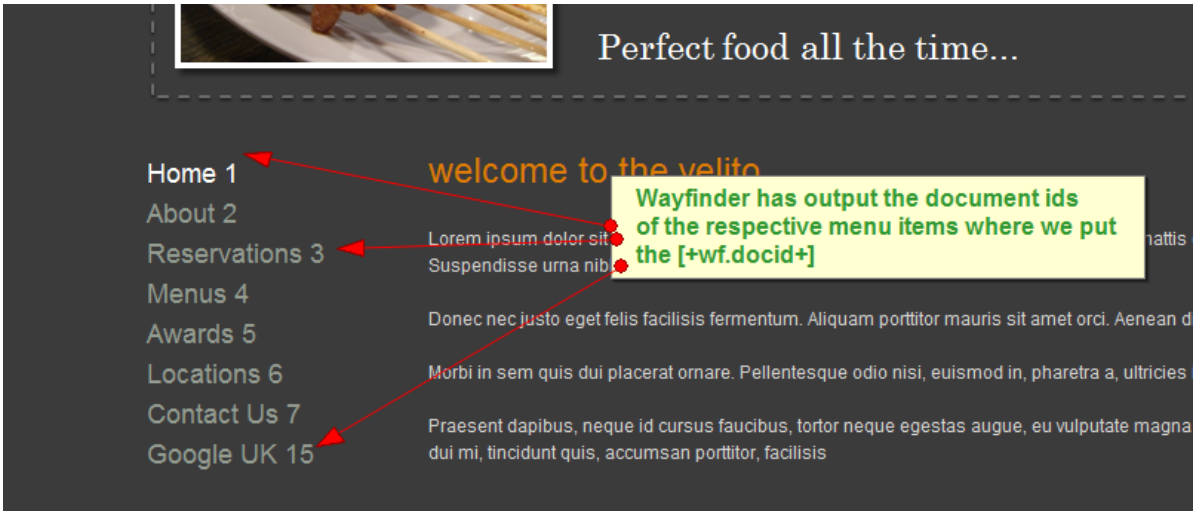
Welcome to lesson #10!

The [+wf.docid+] placeholder

If you wish to output the MODx document **id** of the items in your Wayfinder menu you can do this easily by using the placeholder **[+wf.docid+]**. Similar to most of the other Wayfinder placeholders, the **[+wf.docid+]** is only applicable in row level templates. Wayfinder will output the respective MODx document **id** of your menu items (at all levels) where you put the **[+wf.docid+]** placeholder in your template. In the example below, the placeholder has been added to the **&rowTpl**.

```
<li [+wf.id+] [+wf.classes+] >
<a href="[+wf.link+]" title="[+wf.title+]" [+wf.attributes+]>[+wf.linktext+][+wf.docid+]</a>
[+wf.wrapper+]
</li>
```

The resultant menu is as follows:



Use of [+wf.docid+] inserts document ids in the menu output

The [+wf.description+] placeholder

Using the **[+wf.description+]** placeholder, you can output the contents of your MODx documents' "Description" field to the Wayfinder menu output. This too is a row template placeholder. To illustrate how this placeholder works, we will create a mini menu for our Velito Restaurant. This menu will be placed not within our website's template but will be within the content of the document "Menus".

We will start by creating a **&rowTpl** for this mini menu since I would like to structure the menu differently from the main one. You can do this by either creating a new chunk or duplicating the existing **&rowTpl**, then renaming and editing it. I have called my chunk **menu2Rows** and it has the following as its content:

```
<li [+wf.id+] [+wf.classes+] >
<a href="[+wf.link+]" title="[+wf.title+]" [+wf.attributes+]>[+wf.description+]</a>
[+wf.wrapper+]
</li>
```

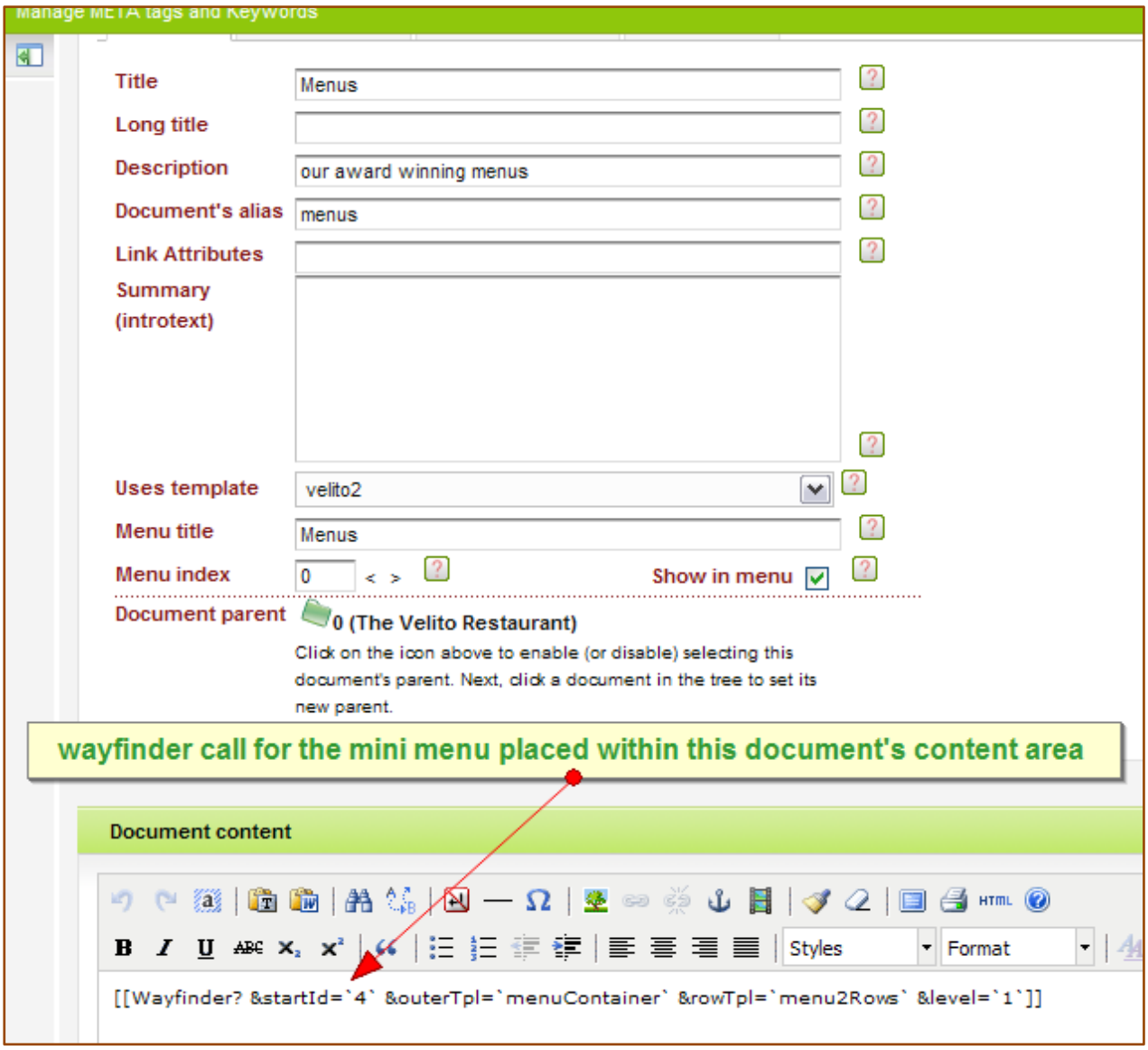
Notice that the chunk does not have a **[+wf.linktext+]** placeholder. I have opted to use **[+wf.description+]** placeholder instead. The contents of the description field for each menu item will be respectively inserted in this placeholder.

For each of the documents that will form our menu, make sure you add some text in their respective "Description" fields. I have already done this for my 3 documents.

Now open the document "Menus" for editing and add the Wayfinder call below within the Document content text area:

```
[[Wayfinder? &startId=`4` &outerTpl=`menuContainer` &rowTpl=`menu2Rows` &level=`1`]]
```

Notice that the **&startId=`4`** - this is of course the **id** of the document "Menus". Notice also the **&rowTpl** we are using for this mini menu, i.e. **menu2Rows**.



Wayfinder called within a document

In addition, modify the **&level** parameter within the Wayfinder call that *we have in our Velito website template* to **&level=`1`**. This is just a matter of preference for me since I do not want the child documents of the parent menu item "Menus" to be part of the main menu of the site. Since there will be no sub-menus in the main menu, I have also done away with the **&hideSubMenus** parameter.

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`1`]]
```

The final step is to add some CSS rules to target the menu items of the mini menu. I have created a 3 column list using the CSS below.

```
#main_content ul  { list-style-type: none; margin:0px; padding: 0; width: 675px; float:left; }
#main_content ul li a {color: #BEFF1F; text-decoration: none; font-size: 2em; float:left; margin: 0px
7px; padding:5px; width: 200px; outline: yellow dashed thin; background: black; text-align: center;}
#main_content ul li a:hover {color: white; background: #DE7C05;outline: white dashed thin;}
```

The image below illustrates the menu output

You could get even fancier by using the **&rowIdPrefix** within the mini menu. This would enable you to use the id selectors of the menu items to style them differently. We will see an example of this in chapter 15.

The [+wf.introtext+] placeholder

The **[+wf.introtext+]** placeholder will allow you to insert the contents of the "Summary (introtext)" field of your MODx documents in a Wayfinder menu. This could probably be used as a teaser for your menu items. This placeholder is only applicable to row level items. I do not need to provide an example of its use since the process is pretty much the same as for the **[+wf.description+]** placeholder; place the **[+wf.introtext+]** within your Wayfinder template where you want the contents of your documents' "Summary (introtext)" to appear.

Other document field names

You can include other MODx document field names (Document Variables) in your Wayfinder output by referencing them using the syntax **[*document_field_name*]** within your Wayfinder templates. For instance, to include the "longtitle" field in the **&rowTpl** of our main menu (or any menu for that matter), we would do this as follows:

```
<li [+wf.id+] [+wf.classes+] >
<a href="[+wf.link+]" title="[+wf.title+]" [+wf.attributes+]>[+wf.linktext+][*longtitle*]
</a>
[+wf.wrapper+]
</li>
```

This would output the "longtitle" of the respective documents where the **[*longtitle*]** was placed in the Wayfinder template.

For a list of MODx document fields visit this link:
http://svn.modxcms.com/docs/display/MODx096/Document+Variables

Using Template Variables in Wayfinder

Using MODx Template Variables (TVs) within your Wayfinder menu is really quite easy. You will need to reference the TVs you want to appear in your menu using the following syntax:

```
[+template_variable_name+]
```

! You will notice from the above syntax that this is not the normal MODx way of referencing a TV! However, this is how it is done in Wayfinder

Note that for each document in your menu, the respective values of their TVs will be output. To better understand this, let us modify the mini menu we earlier looked at in this lesson.

Create an image type TV and give it a name. I have called mine **menutv**. Make that TV accessible to the template you are using for your website. What we want to do is use images in our 3 mini menu items – i.e. "Monday – Thursday", "Friday" and "Saturday – Sunday". We want the value of the **menutv** of each of these documents to be output in their corresponding rows. Instead of using the **[+wf.description+]** as our text of menu, we will use the **menutv** instead. I have gone ahead and prepared three 200px by 200px images that I will use as the values of the **menutv** for each of my documents respectively. I have uploaded these images to my **assets** folder within my MODx install. These images are now available for the **menutv**.

I have then edited each of the above three documents and inserted an image in their respective **menutv** TVs.

It is now time to edit the Wayfinder template where I want this TV to be inserted. I will be using the **&rowTpl** of the mini menu, i.e. **menu2Rows**. I have edited it as follows:

```
<li [+wf.id+] [+wf.classes+] >
<a href="[+wf.link+]" title="[+wf.title+]" [+wf.attributes+]><img src="[+menutv+]"</a>
[+wf.wrapper+]
</li>
```

Please note *where* and *how* I have referenced my TV in the above Wayfinder template. What should happen is that the value (in our case the path to the image) that we gave each document should be output to the corresponding row of each document. In addition, that image should be a link since it will be enclosed within anchor tags **<a>**.

Here is the resultant menu using the above **&rowTpl**:

Use of Template Variables within Wayfinder menu

How far you can go into combining the power of TVs with Wayfinder's brilliance and robustness is perhaps limited only by your imagination.

Well, that is all for this lesson. In the next lesson we will look at how we can instruct Wayfinder to embed CSS or JS into the HEAD section of the page in which it is called. Let us do a recap before we move on.

Summary of lesson #10

1. The **[+wf.docid+]** placeholder if used will cause Wayfinder to insert the MODx document **ids** of the for each and every document that is part of the menu items.

2. Using the **[+wf.description+]** placeholder will cause Wayfinder to output the contents of the "Description" field of the respective documents in your menu.
3. The **[+wf.introtext+]** placeholder will allow you to insert the contents of the "**Summary** (**introtext**)" field of your MODx documents in a Wayfinder menu.

4. To output other Document Variables to your Wayfinder menu, you can use the syntax **[*document_field_name*]** in the row level templates of your Wayfinder menu. For instance, **[*longtitle*]**.

5. To use a TV within your Wayfinder menu, call it using the following syntax in your row level Wayfinder templates; **[+template_variable_name+]**.

Chapter 12: Lesson #11 – gimme some extra codes

Lesson #11 at a glance
&cssTpl **&jsTpl**

Wow...you have done quiet well to have come this far. This lesson will be somewhat advanced but do not let that scare you. It really is straightforward.

The &cssTpl parameter

There may be times when you want a particular menu to use a particular CSS style sheet instead of using your website's main style sheet. By using the **&cssTpl** parameter you can achieve just this. So far in the examples shown in this guide, the menu items have been styled using the external CSS style sheet that is referenced in the "HEAD section" of the template used by our Restaurant website. However, Wayfinder can be instructed to insert an internal (embedded) style sheet to be used by your Wayfinder menu or to output a link to a particular external style sheet that you want used with your menu. In both occasions, these will be inserted into the "HEAD section" of the generated page.

To use this parameter, you need to create a chunk and in that chunk either insert the CSS styles you want used with your menu (internal style sheet format) or insert a link to the CSS file that you want your menu to use. In both cases, you will need to enter the information correctly so that the CSS will be correctly referenced when Wayfinder inserts the contents of your **&cssTpl** chunk in the "HEAD section" of the generated page.

Let us try each of these two methods in turn

Method 1: Instruct Wayfinder to insert an internal CSS style sheet to your documents HEAD section

Create a chunk and give it a name. I have called mine **csstpl1**. Enter your CSS styles, enclosed within **<style></style>** tags into the chunk's code text area as follows:

```
<style type="text/css">
Your CSS style rules /*this is where you will enter your CSS styles*/
/* you need to include the style tags*/
</style>
```

You will obviously have to enter your specific CSS styles where shown by the **red text** in the above chunk. In my case, for this example, I have transferred all my menu's CSS styles from the external style sheet into my **csstpl1** so as to avoid repetition.

Now modify your main menu's Wayfinder call as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`1`
&cssTpl=`csstpl1`]]
```

! Please note that I am providing the above method of embedding styles as an example and not as a recommendation. CSS best practice recommends that you use external style sheets.

The resultant menu and its source code are as follows:

Menu styled using the CSS specified in &cssTpl

And the source code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>The Velito Restaurant | Home</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<base href="http://localhost/velito/"></base>
<link rel="stylesheet" href="assets/templates/velito/css/screen.css" type="text/css" media="screen, projection">
<link rel="stylesheet" href="assets/templates/velito/css/print.css" type="text/css" media="print">
<!--[if IE]>
    <link rel="stylesheet" href="assets/templates/velito/css/ie.css" type="text/css" media="screen, projection">
<![endif]-->
<link rel="stylesheet" href="assets/templates/velito/css/style.css" type="text/css" media="screen" />
<style type="text/css">
#navigation ul  { list-style-type: none; margin:0; padding-top: 0.5em;}
#navigation ul li a {color: #949C91; text-decoration: none; font-size: 1.5em;}
#navigation li.active a{color: white;}
#navigation ul li a:hover   {color: #A0C0CB;}
#navigation li ul {margin: 0 0 0 1.5em;}
#navigation li ul li a {color: #D4D2D0; text-decoration: none; font-size: 1.25em;}
/* you need to include the style tags*/</style>
</head>
```

Wayfinder has embedded/inserted the contents of the &cssTpl template within the HEAD section of the generated document

Method 2: Instruct Wayfinder to insert a link to an external CSS style sheet to your documents HEAD section

This method is much easier than the first one. All you have to do is to create a chunk and inside that chunk correctly reference a link to an external CSS style sheet. In this example, I have created a chunk and named it **csstpl2**. Here is the code for this chunk:

```
<link rel="stylesheet" href="assets/templates/velito/css/menucss.css" type="text/css"
media="screen" />
```

You will need to modify the value of "**href**" to point to the path to your CSS style sheet. You may also wish to modify the "**media type**" to suit your needs.

I then modified the main menu's Wayfinder call as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`1`
&cssTpl=`csstpl2`]]
```

The results show that Wayfinder has correctly output the menu. Here is the source code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>The Velito Restaurant | Home</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<base href="http://localhost/velito/"></base>
<link rel="stylesheet" href="assets/templates/velito/css/screen.css" type="text/css" media="screen, projection">
<link rel="stylesheet" href="assets/templates/velito/css/print.css" type="text/css" media="print">
<!--[if IE]>
    <link rel="stylesheet" href="assets/templates/velito/css/ie.css" type="text/css" media="screen, projection">
<![endif]-->
<link rel="stylesheet" href="assets/templates/velito/css/style.css" type="text/css" media="screen" />
<link rel="stylesheet" href="assets/templates/velito/css/menucss.css" type="text/css" media="screen" />
</head>
```

Wayfinder has embedded/inserted a link to the external CSS style sheet as we instructed it using the css&Tpl parameter

Please note that using this second method, you can include references to multiple CSS style sheets in this one **&cssTpl** chunk.

The &jsTpl parameter

This parameter works in very much the same way as the **&cssTpl** parameter with the exception that it is used to reference a Wayfinder template containing some JavaScript (internal script) or a link to a JavaScript file that you want to use with your menu. Similar to the previous case, the contents of the chunk will be inserted in the "HEAD section" of the generated page. Let us look at the templates for the two methods for inserting JavaScript information.

Method 1: Instruct Wayfinder to insert internal JavaScript to your documents HEAD section

I will not be showing examples for this parameter since it is very similar to the previous one. You would have to create a chunk and give it a name then enter your JavaScript enclosed within **<script></script>** tags into the chunk's code text area as follows:

```
<script language="javascript" type="text/javascript">
Some JavaScript
</script>
```

You can alter the "**language**" parameter as desired.

Method 2: Instruct Wayfinder to insert a link to an external JavaScript file to your documents HEAD section

You would have to create a chunk and give it a name then enter a correctly formatted link to your JavaScript file into the chunk's code text area. Here is an example:

```
<script src="assets/js/jquery-1.2.1.min.js" type="text/javascript"></script>
```

In this example, the above code pointing a JQuery library file will be inserted into the "HEAD section" of the generated page. Edit the **src=""** to suit your situation.

Please note that using this second method, you can include references to multiple JavaScript files in this one **&jsTpl** chunk.

That is all for this lesson folks. In the next lesson we will look at 3 advanced utility parameters. Let us recap before we wrap up.

Summary of lesson #11

1. The **&cssTpl** parameter can be used to reference an internal or external CSS style sheet. The contents of the chunk you specify as the value of **&cssTpl** will be inserted into the HEAD section of the generated page.

2. Similarly, using the **&jsTpl** parameter will cause Wayfinder to output the contents of the specified chunk into the HEAD section of the generated page.

Chapter 13: Lesson #12 – don't bug me! Debug me!

Lesson #12 at a glance
&ph **&removeNewLines** **&debug**

Welcome to lesson #12, the final lesson in this guide. Well done for making it this far. In this lesson, we will be looking at 3 advanced Wayfinder parameters.

The &ph parameter

Using the **&ph** parameter you can tell Wayfinder to place its output in a placeholder of your choice. The normal behaviour of Wayfinder (and indeed other MODx snippets) is to replace the snippet call with the snippet's output. Using the **&ph** parameter however, you can override this behaviour. What this means is that your menu will be output where you have placed the specified placeholder. You will need to create this placeholder yourself which is quite straightforward as you will find out in a minute. Using this method, you can have multiple outputs of the same menu in different locations, i.e. wherever you have that placeholder. For instance, you can call Wayfinder from within one document and have it output the menu in another document or template. Let us see how this works.

Start by "creating" a placeholder using the following syntax:

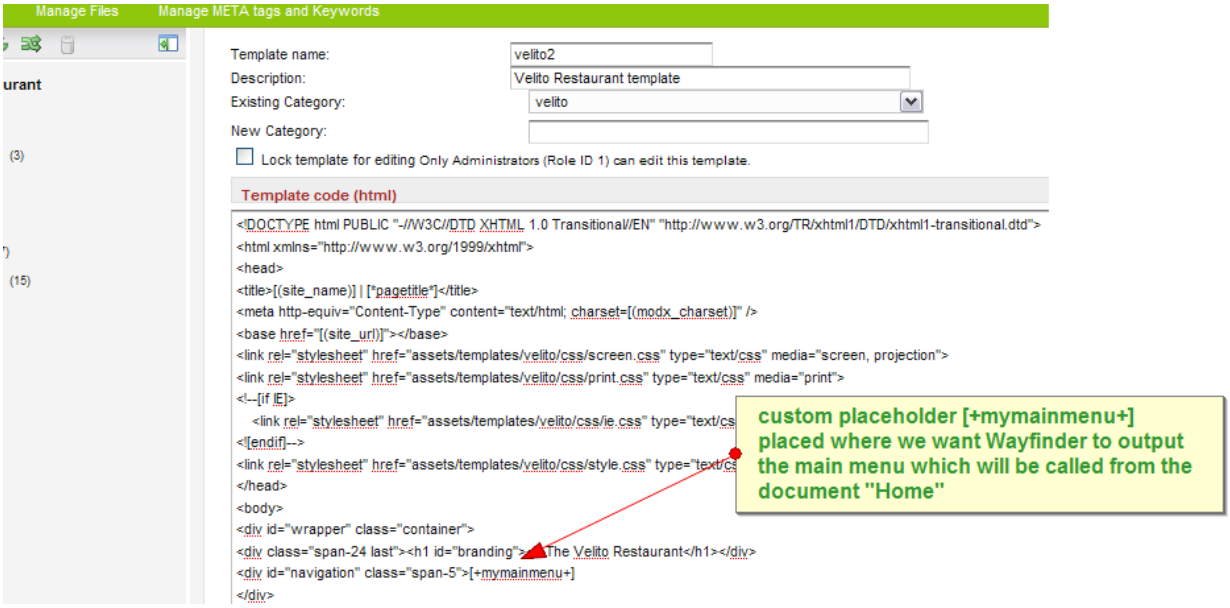
[+your_placeholder_name+]

Make sure you do not use the name of a MODx Document Variable or an existing TV name, etc.

I have created a placeholder with the name "**mymainname**". In order for it to be interpreted as a placeholder it will need to be written as follows whenever it is used:

[+mymainmenu+]

I then typed the name of my placeholder in my Website's template where I will soon remove the Wayfinder call and transfer it to another location. This will allow us to see whether this parameter actually works.



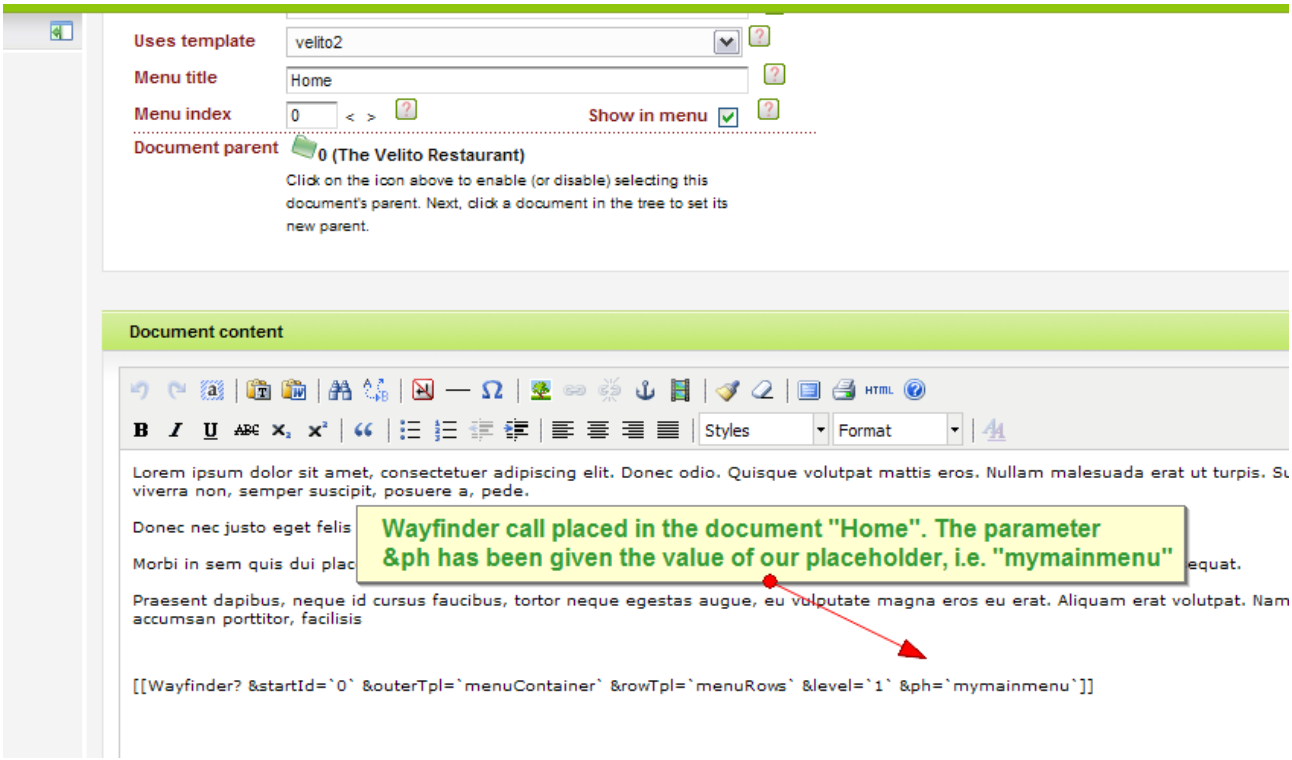
Custom placeholder placed within the website tempalte

The next step is to remove the Wayfinder call from the template and place it somewhere else. I have chosen to place it in the content area of the document "Home".

Finally I need to tell Wayfinder where to insert the menu by telling it the name of my placeholder. I edited the just transferred Wayfinder call as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`1`
&ph=`mymainmenu`]]
```

! Note that in the Wayfinder call you need to enter the name of the placeholder without the **[++]**!



Remember to save the document where you placed the Wayfinder call and preview the menu. It works just fine for me. I also tested with inserting my custom placeholder in the chunk that is used to create the "footer" of my Restaurant website and it also worked OK.

You can now revert the changes we made to our Wayfinder call and the template, etc since we will not be using the **&ph** parameter further on in this lesson.

The &removeNewLines parameter

The **&removeNewLines** parameter only accepts Boolean values, "TRUE" or "FALSE", the default value being "FALSE". For increased code readability, Wayfinder will normally automatically add line breaks into the generated output. If you wish to remove these line breaks you will need to set this parameter to "TRUE". This will cause Wayfinder to remove those line breaks.

To use the parameter you would write you call as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`1`
&removeNewLines=`TRUE`]]
```

The &debug parameter

The **&debug** parameter is a very useful utility for troubleshooting your Wayfinder menu output. The parameter only accepts "TRUE" or "FALSE" values, the default value being "FALSE". If set to true, Wayfinder will output information on how each document in the menu was processed. This very

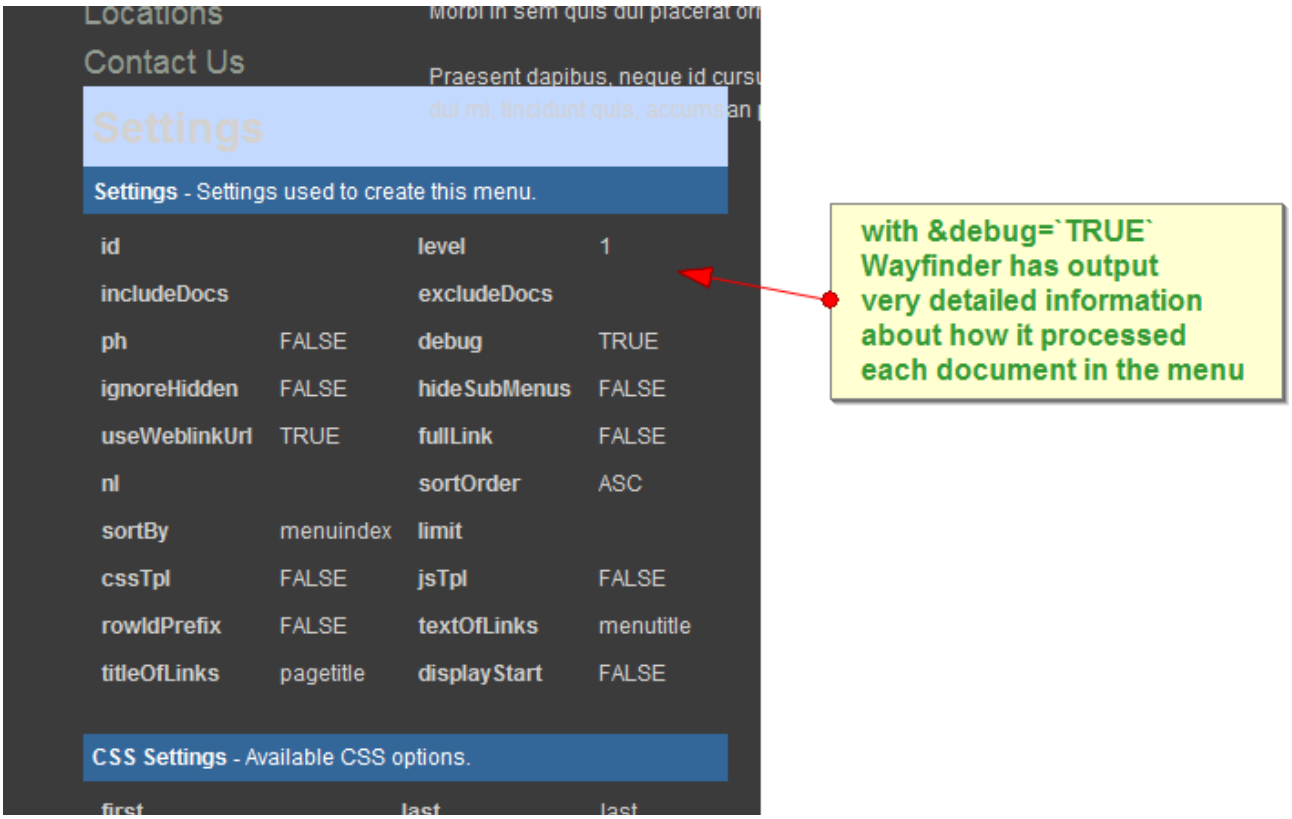
detailed information is output in the front end, so this is not something you want to do on a live site. Let us see how this works.

Modify the Wayfinder call for the main menu as follows:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`1`
&debug=`TRUE`]]
```

Save the template and preview the menu in the front end. You should get not only your menu output but also an overlay of some very detailed information on how your menu documents were processed.

Here is a snipped snapshot of how my output looks like:



The &debug parameter set to "TRUE"

That is all folks for this lesson. In the next chapter I will briefly mention 3 advanced Wayfinder parameters that this guide will not cover. Let us recap this lesson before we wrap up.

Summary of lesson #12

1. The **&ph** parameter can be used to instruct Wayfinder to output a menu to a different location other than where it has been called. The output will be inserted at the location of a custom made placeholder that you define.

2. The **&removeNewLines** parameter if used will instruct Wayfinder to remove its normally automatically inserted line breaks from the output code.

3. The **&debug** if set to TRUE will cause Wayfinder to output, besides the menu, long and detailed information on how each document in the menu was processed. It is a useful troubleshooting parameter.

CHAPTER 14: SOME ADVANCED WAYFINDER PARAMETERS

Chapter 14 at a glance
@CODE **@FILE** **&config**

There are three parameters that this guide will not be covering that I want to briefly mention.

The @CODE parameter

If you use the **@CODE** parameter as part of the value for a template parameter, Wayfinder will interpret whatever comes after **@CODE:** as the template code itself. For instance:

```
[[Wayfinder? &startId=`0` &outerTpl=`@CODE:<ul [+wf.classes+]>[+wrapper+]</ul>`
&rowTpl=`menuRows` &level=`1`]]
```

In the above example we are telling Wayfinder that the value of &outerTpl is all that comes after the **@CODE:**

The @FILE parameter

If you use the **@FILE** parameter as part of the value for a template parameter, Wayfinder will interpret whatever you have written after **@FILE:** as the directory path to a file which contains the template. For example:

```
[[Wayfinder? &startId=`0` &outerTpl=`@FILE: /assets/templates/velito/outertpl.txt`
&rowTpl=`menuRows` &level=`1`]]
```

In the above example we are telling Wayfinder to use the contents of the file outertpl.txt to construct the **&outerTpl** of the menu.

The &config parameter

Wayfinder allows you to package into a single file all the specifications for your menu. This configuration file can contain all your menu's template chunks, parameters and their corresponding values. In this way you can call Wayfinder with just a single **&config** parameter, and get a complete set of specifications for the call.

Have a look at the **configs** directory in the Wayfinder folder that comes with the MODx install (or in the Wayfinder package that you download) for example configuration files. If you want to use your own configuration file you will need to place it in this directory.

The name of your configuration file should be in the format **[name].config.php** where **[name]** is the name you have given your configuration file. For example, if you want to name your configuration file "**topmenu**" then the full name of the configuration file should be **topmenu.config.php**

Below is an example of how you would instruct Wayfinder to use the custom config file "**topmenu.config.php**"

```
[!Wayfinder? &config=`topmenu`!]
```

Note that you only need to include **[name]** part of your configuration file's name as the value of the parameter **&config**.

For more information about this parameter please refer to the Wayfinder sub-forum in the MODx forums.

CHAPTER 15: EXAMPLE MENUS

Chapter 15 at a glance

1. **Fisheye Wayfinder menu**
2. **Sitemap using Wayfinder**
3. **JQuery accordion Wayfinder menu**
4. **UltimateParent snippet and Wayfinder menu**
5. **CSS Sprite menu using Wayfinder**

Fisheye menu using Wayfinder

Adapting a fisheye menu for use in MODx using Wayfinder is really not hard.

Credits

ICONS: www.iconspedia.com

CSS Dock Menu
Developer: Nick La
Website: http://www.ndesign-studio.com/blog/mac/css-dock-menu

The task

Create a one level fisheye menu using Wayfinder for the Velito Restaurant website:

In this example, we will be adapting the fisheye menu developed by www.ndesign-studio.com for use in MODx (the developer calls it CSS Dock menu). The menu uses jQuery and the fisheye component by http://interface.eyecon.ro.

Start by visiting that website and downloading the menu packages. Unzip the files to your computer and open the **css-dock-top.html** in your browser. That is the menu we want to recreate. Here is its source code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>CSS Mac Dock</title>
<script type="text/javascript" src="js/jquery.js"></script>
<script type="text/javascript" src="js/interface.js"></script>

<!--[if lt IE 7]>
```

```
 <style type="text/css">
 div, img { behavior: url(iepngfix.htc) }
 </style>
<![endif]-->

<link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div class="dock" id="dock">
 <div class="dock-container">
 <a class="dock-item" href="#"><img src="images/home.png" alt="home"
/><span>Home</span></a>
 <a class="dock-item" href="#"><img src="images/email.png" alt="contact"
/><span>Contact</span></a>
 <a class="dock-item" href="#"><img src="images/portfolio.png" alt="portfolio"
/><span>Portfolio</span></a>
 <a class="dock-item" href="#"><img src="images/music.png" alt="music"
/><span>Music</span></a>
 <a class="dock-item" href="#"><img src="images/video.png" alt="video"
/><span>Video</span></a>
 <a class="dock-item" href="#"><img src="images/history.png" alt="history"
/><span>History</span></a>
 <a class="dock-item" href="#"><img src="images/calendar.png" alt="calendar"
/><span>Calendar</span></a>
 <a class="dock-item" href="#"><img src="images/rss.png" alt="rss"
/><span>RSS</span></a>
</div>
</div>
<script type="text/javascript">

        $(document).ready(
                function()
                {
                        $('#dock').Fisheye(
                                {
                                        maxWidth: 50,
                                        items: 'a',
                                        itemsText: 'span',
                                        container: '.dock-container',
                                        itemWidth: 40,
                                        proximity: 90,
                                        halign : 'center'
                                }
                        )
                }
        );

</script>
</body>
</html>
```

The process & solution

In order to implement the above code into MODx let us break the above code into smaller parts. This will help us know what to do and how to accomplish it.

The code	The thought process
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <html xmlns="http://www.w3.org/1999/xhtml"> <head> <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /> <title>CSS Mac Dock</title>	We already have something similar in our website's template so we do not need this code.

The code	The thought process
<script type="text/javascript" src="js/jquery.js"></script> <script type="text/javascript" src="js/interface.js"></script>	We will first need to copy the **jquery.js** and **interface.js** to the desired location in our MODx install. I decided to go with the latest jQuery package, i.e. **jquery-1.3.2.min.js** so did not use the **jquery.js** file
The modified code	**What I have done**
<script type="text/javascript" src="assets/js/ jquery-1.3.2.min.js "></script> <script type="text/javascript" src="assets/js/interface.js"></script>	1. I copied the file **interface.js** to MODx assets/js directory 2. I downloaded the latest jQuery, **jquery-1.3.2.min.js** and copied it to MODx assets/js directory 3. I then added and modified the references to the two js files within my website's template's **<head></head>** section

The code	The thought process
<!--[if lt IE 7]> <style type="text/css"> div, img { behavior:	1. I copied the file **iepngfix.htc** to my website's template's CSS folder.

url(assets/templates/velito/css/iepngfix.htc) } </style> <![endif]--> <link href=" assets/templates/velito/css/style.css" rel="stylesheet" type="text/css" /> </head>	2. I have added this to the website's template so as to force internet explorer 7 to play nice with my **png** images. 3. I also modified the path to point to the location my CSS files 4. I do not need all the styles in the CSS file **style.css** that comes with the fisheye menu package we downloaded. Hence, I copied and modified as necessary the styles that only apply to the fisheye menu to my own CSS file which is also named **style.css**. I also modified my websites template since I want a horizontal fisheye menu

The code	The thought process
<body> 	I already have a container **div** where I will place the Wayfinder call for my menu so I do not need the inner **divs**. I have, however, added the class "**dock-container**" to my menu container **div**. I could have of course used a different class name and change this in the configuration of the script and my stylesheet too. The menu also needs a **div** with the id "**dock**" to wrap around the **div** that will contain the menu. I have replicated this in my website template. I could have given the **div** an **id** with a different name and updated the script but I chose not to.

The code	The thought process
Home Contact Portfolio Music Video History Calendar RSS	This is where the fun really begins. Wayfinder will take care of this part for us. However, let us look at this code closer by further sub-diving this bit (see below). We only need to consider the code for one menu item, e.g. **Home**, since the rest are just replications but for different links.

The code	The thought process
Home	We will deal with this bit using a custom Wayfinder **&rowTpl** template chunk since this is a menu row item. Notice that the links in the above code are not wrapped in list ** ** tags. We are going to change this so that they can be list items similar to the menus we have been creating in this guide.
The code line by line	**MODx/Wayfinder solution**
<a	This will be part of the **&rowTpl**
class="dock-item"	I decided to add this as is directly to the **<a>** tags of the row items in my **&rowTpl** template
href="#">	The placeholder **[+wf.link+]** should take care of this, i.e. **href="[+wf.link+]"** *(see lesson #2)*

<img src="images/home.png"	1. I do not want to hardcode the path to the **png** images for each menu item in the fisheye menu. I can therefore use an image TV instead. 2. We already created an image TV for our website in lesson #10. The TV was called **menutv**. This will there be applied as: **<img src="[+menutv+]"** 3. I wanted to use other **png** images for the menu so I downloaded 7 128x128 **png** images from www.iconspedia.com. I then copied the images to my MODx assets/images folder 4. Finally, I edited each of the 7 menu item documents to include the **png** images using the TV "**menutv**"
alt="home" />	I added the "**alt**" attribute to the **&rowTpl** chunk. To give it a value I had 2 choices; 1. Either use the **[+wf.title+]** -> this would output the default value for **&titleOfLinks** i.e. "**pagetitle**". **OR** 2. Use **[+ wf.linktext+]** -> this would output the **&textOfLinks** default value, i.e. "**menutitle**" I opted to use the **[+wf.title+]**
****	I added this to the **&rowTpl** chunk as shown below in the final template chunk
Home	Wayfinder's **[+wf.title+]** should take care of this, i.e. it will

	output the value of **&textOfLinks**
****	I added this to the **&rowTpl** chunk as shown below in the final template chunk

Putting it all together, this is the code of the final **&rowTpl** I created for our fisheye menu:

```
<li [+wf.classes+]>
<a class="dock-item" href="[+wf.link+]" title="[+wf.title+]">
<img src="[+menutv+]" alt="[+wf.title+]" />
<span>[+wf.linktext+]</span>
</a>
[+wf.wrapper+]
</li>
```

I named the **&rowTpl** chunk "**fisheyeRows**". Notice that I am also using some placeholders that I may need for my menu, e.g. **[+wf.classes+]**.

The code	The thought process
 	These are just the closing tags of the **divs**. Nothing to do here if we have already properly closed the **divs** in our template.

The code	The thought process

<script type="text/javascript">
 	
$(document).ready(
function()
{
$('#dock').Fisheye(
{
maxWidth: 50,
items: 'a',
itemsText: 'span',
container: '.dock-container',
itemWidth: 80,
proximity: 90,
halign : 'center'
}
)
}
);
	This bit is very important and has to be added anywhere within the **<body></body>** tags of our template. This is the **js** configuration for the fisheye menu. I added it as is to my website's template. As mentioned above, I made sure that my template has the corresponding selectors "**dock**" and "**dock-container**". As a matter of personal choice, I also increased the " **itemWidth** " from **40** to **80**

</script>	

The code	The thought process
</body> </html>	This is already in our website's template.

And finally the Wayfinder call. I placed this in the website's template.

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`fisheyeRows` &level=`1`]]
```

Do not forget to insert the **png** images for each of the documents in the menu. As mentioned above, you do this via the image TV "**menutv**".

The final result

And here is the finished menu. Though there might be browser related issues, these really have nothing to do with Wayfinder.

Fisheye menu created using Wayfinder

Sitemap using Wayfinder

This is really quite easy to do.

The task

Create a simple sitemap for the Velito Restaurant website.

The process & solution

This will require a simple Wayfinder call. We can either include this call in the template or call it for a particular document only. I have gone with the latter.

1. The first step is to create the document that will hold our sitemap. I created a document called "**sitemap**" for this purpose. MODx gave this document id #**16**

2. I also did not want the document "**sitemap**" to appear in the main menu so I used the parameter **&excludeDocs** in our website's main menu Wayfinder call to achieve this.

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`fisheyeRows` &level=`1`
&excludeDocs=`16`]]
```

3. Next, I added a small menu in the footer section of the website since this is where I want the document "**sitemap**" to appear. I used the following call.

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`1`]]
```

Notice that I am using the chunk we created in lesson #2 for the **&rowTpl** of the footer menu.

4. I added some CSS rules for this footer menu in my style sheet.

5. I duplicated and slightly modified my website's template and assigned the document "**sitemap**" this new template. I did not want the "**side-bar**" content to appear in the document "**sitemap**". There are other methods I could have achieved to exclude the "**side-bar**" content without resorting to creating a new template but this is what I have chosen for illustration purposes.

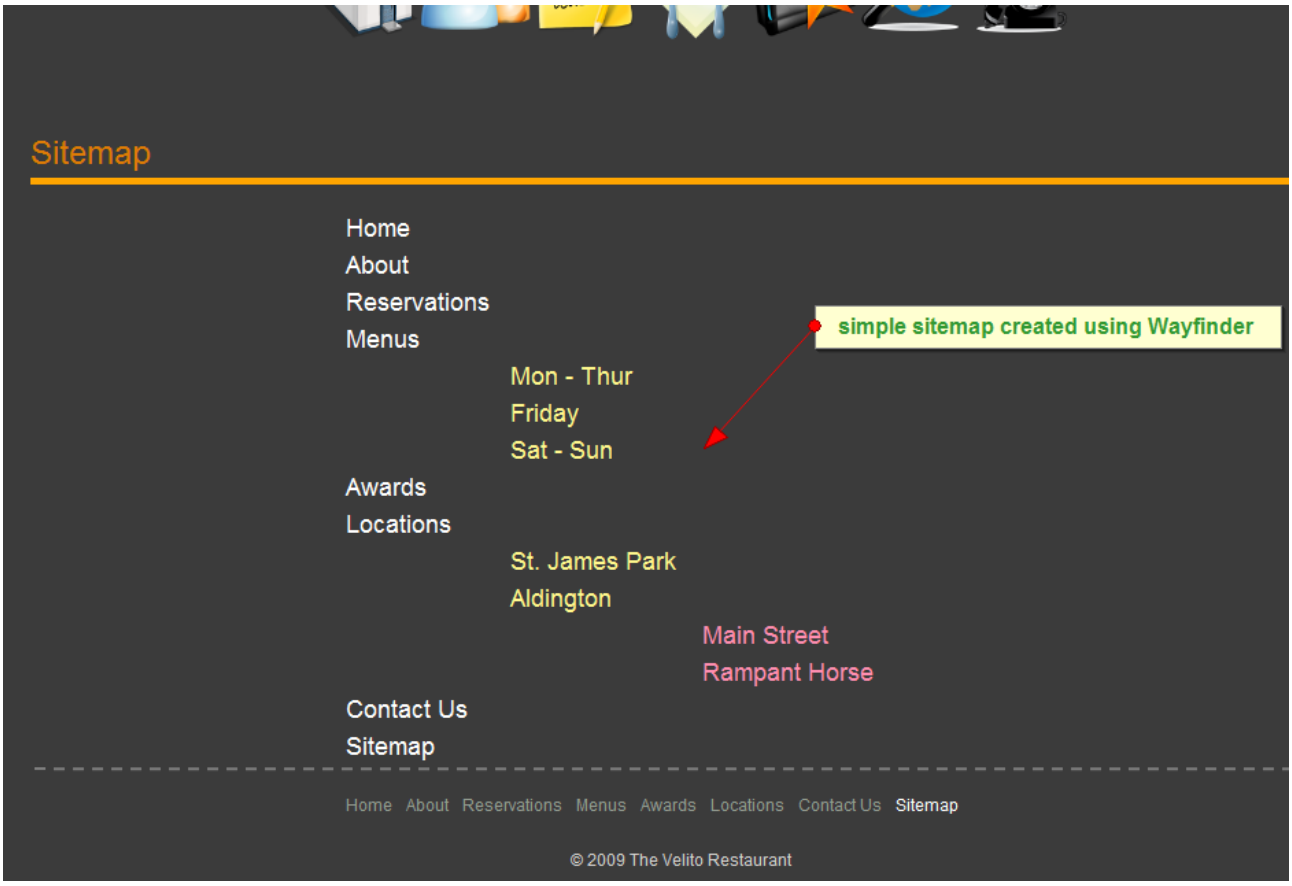
6. Finally, I added the following Wayfinder call to the content section of the document "**sitemap**" and added some CSS rules to my style sheet for the sitemap.

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows`]]
```

That is all there is to it. You do not have to go through all the above steps. The idea here is that we want to show all or most of the documents in our MODx document tree in our site map. Hence, the **&startId=`0`** is the key issue in creating the sitemap.

The final result

Here is the simple sitemap created using Wayfinder and CSS:



A simple sitemap created using Wayfinder

JQuery accordion Wayfinder menu

Credits

JQuery Accordion menu
Developer: Marco van Hylckama Vlieg
Website: http://www.i-marco.nl/weblog/archive/2008/05/08/simple_jquery_accordion_menu

The task

Recreate in MODx the simple jQuery accordion menu found in the above link using Wayfinder and CSS.

Visit the above site and download the simple jQuery accordion menu packages. Unzip the files to your computer and open the file **index.html** in your browser. This is the menu we will be replicating in MODx using Wayfinder. There are 4 menu examples in that package. In this example, we will be recreating a menu that has the "accordion" effect, is not open on load but is collapsible when an expanded item is clicked. As in the fisheye menu above, I have decided to use the latest jQuery package, version 1.3.2. Here is the (truncated) source code of the accordion menu we want to adapt and port over to MODx:

```
<ul id="menu4" class="menu collapsible expandfirst">
<li>
<a href="#">Weblog Tools</a>
<ul>
<li><a href="http://www.pivotx.net/">PivotX</a></li>
<li><a href="http://www.wordpress.org/">WordPress</a></li>
<li><a href="http://www.textpattern.com/">Textpattern</a></li>
<li><a href="http://typosphere.org/">Typo</a></li>
</ul>
</li>
<li>
<a href="#">Programming Languages</a>
<ul>
<li><a href="http://www.php.net/">PHP</a></li>
<li><a href="http://www.ruby-lang.org/en/">Ruby</a></li>
<li><a href="http://www.python.org/">Python</a></li>
<li><a href="http://www.perl.org/">PERL</a></li>
<li><a href="http://java.sun.com/">Java</a></li>
<li><a href="http://en.wikipedia.org/wiki/C_Sharp">C#</a></li>
</ul>
</li>
<li><a href="http://www.i-marco.nl/weblog/">Marco's blog (no submenu)</a></li>
<li>
<a href="#">Cool Stuff</a>
<ul>
<li><a href="http://www.apple.com/">Apple</a></li>
<li><a href="http://www.nikon.com/">Nikon</a></li>
<li><a href="http://www.xbox.com/en-US/">XBOX360</a></li>
<li><a href="http://www.nintendo.com/">Nintendo</a></li>
</ul>
</li>
<li>
<a href="#">Search Engines</a>
<ul>
<li><a href="http://search.yahoo.com/">Yahoo!</a></li>
<li><a href="http://www.google.com/">Google</a></li>
<li><a href="http://www.ask.com/">Ask.com</a></li>
<li><a href="http://www.live.com/?searchonly=true">Live Search</a></li>
</ul>
```

We will obviously need to replace those links with our own menu links. Please note that I have not included the code found in the **head** section of the file **index.html** in the above code. We will pick only what we need from there. OK, let us get cracking.

The process & solution

According to the developer, for the menu to work, we will need to do:

* Give each menu a unique ID
* Give each menu a CSS class named menu

Nothing difficult there; it is pretty straight forward. Let us break the above code into smaller parts. This will help us know what to do and how to accomplish it.

The code	The thought process
<script src="jquery-1.2.1.min.js" type="text/javascript"></script> <script src="menu.js" type="text/javascript"></script>	We will first need to copy the **jquery-1.2.1.min.js** and **menu.js** to the desired location in our MODx install. I decided to go with the latest jQuery package, i.e. **jquery-1.3.2.min.js** so did not use the **jquery-1.2.1.min.js** file
The modified code	**What I have done**
<script type="text/javascript" src="assets/js/ jquery-1.3.2.min.js "></script> <script type="text/javascript" src="assets/js/menu.js"></script>	1. I copied the file **menu.js** to MODx assets/js directory 2. I downloaded the latest jQuery, **jquery-1.3.2.min.js** and copied it to MODx assets/js directory 3. I then added and modified the references to the two js files within my website's template's **<head></head>** section

The code	The thought process
**<ul id="menu4" class="menu collapsible expandfirst"> Weblog Tools PivotX WordPress **	This part of the code should be very familiar to us by now. It is simply a nested two level unordered list We will apply what we have learnt and use Wayfinder template chunks. We will

Textpattern Typo <!-- removed what we do not need – i.e. links, lists items --> 	therefore need an **&outerTpl**, a **&rowTpl** and possibly an **&innerTpl**
The code line by line	**MODx/Wayfinder solution**
<ul id="menu" class="menu collapsible">	This bit we will recreate using **&outerTpl** because it is the outer most part of the menu. As we have already seen, the list items we will be recreating have two levels. Also, notice that the ** ** has an **id**. Remember in lesson #2 and #3 we learnt that we need to be careful how we construct the **&outerTpl** especially when we will be giving the ** ** and **id**. We also learnt that if an **&innerTpl** is not specified, Wayfinder will use the **&outerTpl** to create the container of the sub-menus. Since our **&outerTpl** will have an **id**, we do not want this to happen. We will need to create an **&innerTpl** as well for the sub-menus later on. I have changed id to "**menu**" Finally, personally I do not wish for the menu to be expanded on load so I have removed the class "**expandfirst**" from the ** **

Below is the code of the final **&outerTpl** I created for our jQuery accordion menu. If I wanted to use the Wayfinder class for the **&outerTpl**, i.e. **&outerClass** alongside the classes that are needed or are optional for this accordion menu ("**menu**", "**collapsible**", etc), I would have needed to use the placeholder **[+wf.classnames+]** *(see lesson #1)*. Since I do not wish to use **&outerClass** I typed in the classes I need myself.

```
<ul id="menu" class="menu collapsible">[+wf.wrapper+]</ul>
```

I named the **&outerTpl** chunk **accordionContainer**. Let us move on.

The code	The thought process
 Weblog Tools <!—inner/nested and will be recreated using **&innerTpl**	This part of the code will be created using **&rowTpl** *(see the final template chunk below)*.
The code line by line	**MODx/Wayfinder solution**
** **	This is the opening tag of the list items themselves will be part of the **&rowTpl**
<a	This will be part of the **&rowTpl**
href="#">	The placeholder **[+wf.link+]** should take care of this, i.e. **href="[+wf.link+]"** *(see lesson #2)*
Weblog Tools	Wayfinder's **[+wf.title+]** should take care of this, i.e. it will output the value of **&textOfLinks**
****	This will be part of the **&rowTpl**
** **	This is the opening tag of the sub-menus container which we will create using the **&innerTpl** *(see the final template chunk below)*.
** PivotX WordPress Textpattern Typo **	These are the row level items of the sub-menus. In lesson #3 we learnt that if we specify an **&innerRowTpl**, this template chunk will be used to create the row level items of the sub-menus. We also learnt that if we do not specify this template chunk, Wayfinder will by default use the **&rowTpl** to construct the sub-menu row level items. In this example we really do not

	need to create an **&innerRowTpl** so we will let Wayfinder use the **&rowTpl** to construct the sub-menu's row items.
** **	This the closing tag of the **&innerTpl**
** **	This the closing tag of the **&rowTpl**
** **	This the closing tag of the **&outerTpl**

Here is the **&rowTpl** which I named **accordionRows** and will be using for the jQuery menu:

```
<li [+wf.classes+] >
<a href="[+wf.link+]" title="[+wf.title+]" [+wf.attributes+]>[+wf.linktext+]</a>
[+wf.wrapper+]
</li>
```

And this is the **&innerTpl** which I named **accordionInnerContainer** and will be using for the jQuery menu:

```
<ul [+wf.classes+]>[+wf.wrapper+]</ul>
```

And finally the Wayfinder call. I placed this in the website's template.

```
[[Wayfinder? &startId=`0` &outerTpl=`accordionContainer` &rowTpl=`accordionRows`
&innerTpl=`accordionInnerContainer` &level=`2` &excludeDocs=`16`]]
```

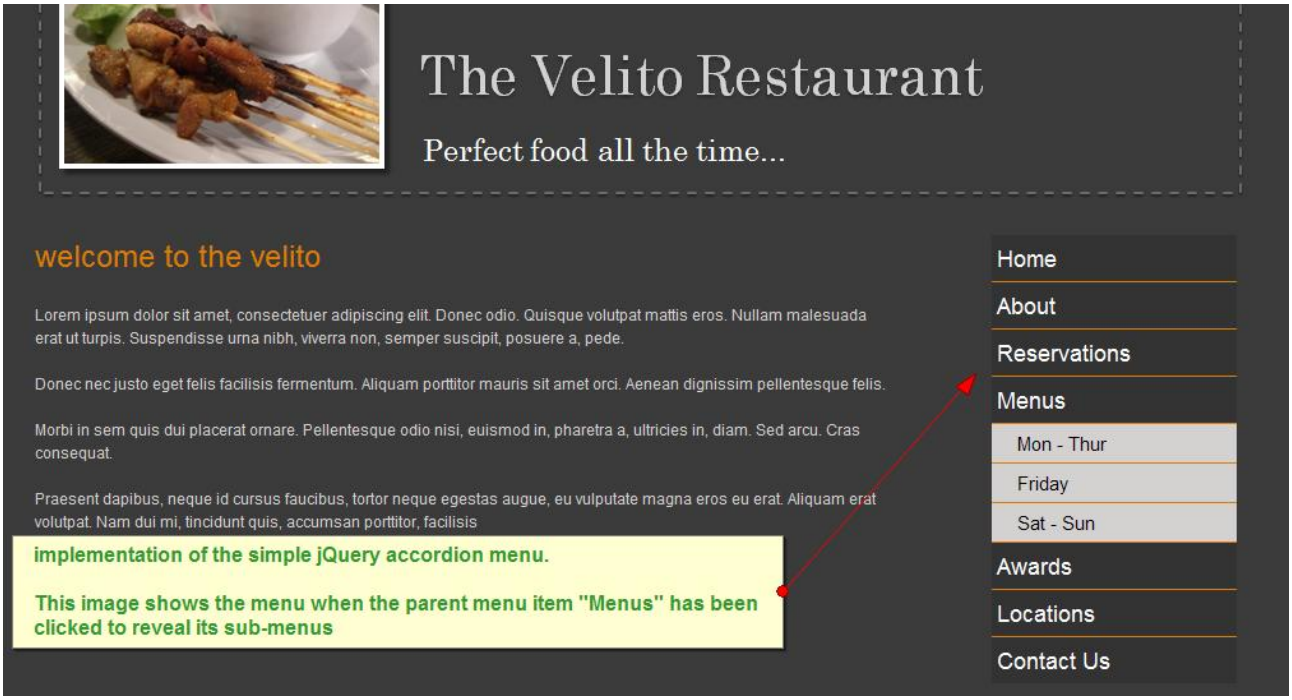
I reverted to and modified my earlier website's template since I want to have the jQuery accordion menu horizontally placed in the side-bar. I am still excluding the document "**sitemap**" from this menu though. I also modified the CSS that came with the menu and included it in my style sheet.

! We could have used the parameter **&hideSubMenus** so that in case javascript is disabled by a user, the sub-menus would not show unless their parent item was clicked. However, with this jQuery accordion menu, my tests showed that using this parameter meant that the user had to click twice on the parent item in order to reveal the sub-menus

Secondly, my tests also showed that clicking on a parent item revealed the sub-menus **BUT** did not load the parent item document itself

The final result

This is the finished menu with the parent menu item "**Menus**" clicked. There might be browser related issues that really have nothing to do with Wayfinder.



A simple jQuery accordion menu implemented in MODx using Wayfinder

UltimateParent snippet and Wayfinder menu

In this example we will be using the MODx snippet UltimateParent in conjunction with Wayfinder. There are times when you would want to show, for instance, a horizontal main menu near the top of your website with a second vertical sub-menu running either on the left or the right hand side of your website that displays the child documents of the main horizontal menu. In this scenario, if you click a menu item on the main menu, the side menu displays the sub-menus of the clicked main menu item. This is a classic example whereby the snippet UltimateParent is very handy. As described by the author of the snippet, "UltimateParent travels up the document tree to return the **id** of the 'ultimate' parent of a document." The snippet has two optional parameters that we will not get into in this example.

Credits

UltimateParent
Developer: Susan Sottwell
Website: http://www.sottwell.com

The task

Construct a one-level horizontal main menu whose child documents will be revealed in a vertical sub-menu located on the right hand side of the website. The sub-menu will be dynamic in that it will only reveal the child documents of the currently active main menu item.

The process & solution

This example will require two Wayfinder calls; one for the horizontal main men and the other for the vertical sub-menu. I have gone ahead and modified my website's template so as to accommodate these two menus.

1. The first step is to check whether you have the snippet UltimateParent installed. It comes with the default MODx installation. If you do not have it, grab a copy from here http://modxcms.com/extras.html?view=package/view&package=475 and install it.

2. I added a few more child documents to my Velito Restaurant website for the purposes of this example. You do not need to do this as long as you have a couple of child documents in your test website.

3. I also did not want the document "**sitemap**" to appear in the main menu so I used the parameter **&excludeDocs** in our website's main menu Wayfinder call to achieve this.

4. I placed the following Wayfinder call to display the horizontal main menu:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`1`
&excludeDocs=`16`]]
```

Note that this call is not the same as the one we used for the accordion menu we have just tackled hence I am using our original website's chunks.

5. I placed the following Wayfinder call in a **div** on the right hand side of my website's template.

```
[!Wayfinder? &startId=`[[UltimateParent]]` &outerTpl=`menuContainer` &rowTpl=`menuRows`
&level=`1` &excludeDocs=`16`!]
```

Look carefully at the above side-menu Wayfinder call. A couple of things to note:

- The above is an example of nested snippet calls in MODx. This means that one snippet is being called within another snippet. In this example, the UltimateParent snippet call is nested within the Wayfinder snippet call.

- In MODx, when you have nested snippet calls, you need to alternate their cached/uncached status. Therefore, whilst we are calling UltimateParent cached, we need to and have thus called Wayfinder uncached. This is very important! If both snippet calls were cached in all likelihood the menu would not have been output

- UltimateParent returns the **id** of the ultimate parent of a document. In our example, the value of **&startId** is being passed to Wayfinder by UltimateParent. You may be wondering which document is the ultimate parent of a document located at the top most hierarchy in the MODx document tree, for instance, the our main menu's documents. The answer is actually the document itself! To illustrate, when the user clicks on the document "Locations" in the main menu, there is action in the second Wayfinder call in the side-menu. UltimateParent runs up the MODx document tree to check which document is the ultimate parent of this currently active document "Locations". This document is at the top of the tree hierarchy so it has no parents and therefore UltimateParent return the **id** of the document, in this case, "**6**". Hence, in the side-menu Wayfinder builds a menu using **&startId=`6`** thereby displaying document's "**6**" child documents *(I hope you have not forgotten that Wayfinder will by default build a menu using the descendant documents of the document specified as the **&startId**)*. If you want to test the snippet UltimateParent, place the following call somewhere in your website's template and view the output in the front end:

```
[[UltimateParent]]
```

  You will notice that the above snippet call will return the **id** of the ultimate parent of the document you are currently viewing. For instance, in my Velito website, when viewing the document "Friday" the value returned is "**4**" which is the **id** of the parent document, i.e. "Menus". UltimateParent has two parameters that you can use to control how far up the tree it should go looking for the ultimate parent and which document's parents you want it to return.

6. Note that I am reusing the **&outerTpl** and **&rowTpl** chunks from the main menu since I do not need any special functions.

7. Finally I added some CSS rules to style both menus.

That is all there is to it.

The final result

Here is the final output:

UltimateParent working together with Wayfinder used to create separate but related menus

In the above image, the document "Locations" is currently selected hence its child documents are displayed in the side-menu.

UltimateParent working together with Wayfinder used to create separate but related menus

In this image, the document "Awards" is currently selected hence its child documents are displayed in the sub-menu on the right side.

CSS Sprite menu using Wayfinder

Creating a menu that utilizes so-called CSS sprites is really quite easy using Wayfinder. If you are not familiar with CSS sprites, the basic thing to know about them is that CSS sprites allow us to combine several images that we would have otherwise used individually and use that image as a background image for menu items. With CSS sprites, the key is in position the image so that it only reveals what you want it to reveal. By playing around with the position of the image, one can for example use the same image for the normal, hover and active states of a menu. The biggest advantage to CSS sprites is that they reduce server load because of fewer HTTP requests since only one image has to be loaded into memory. I suggest you Google CSS sprites if this is still not clear.

The task

Construct a Wayfinder menu that utilises CSS sprites. The single image to be used as the sprite should show unique images for each menu item's normal, hover and active states.

The process & solution

1. We start by preparing the image we will use as the sprite. In my case, I combined and modified the images I used in the fisheye menu into one big image that would show the different states of each of the menu items. Here is the image:

CSS sprite image

The above single image was made from the 7 unique images. The bottom part of the image will act as the active state of the menu items. The middle part will act as the hover images whilst the top part will be the normal, i.e. inactive menu items. They key thing is to make sure is that the images are properly aligned. You also need to take note of the widths of the individual images and their heights too. This information will help in correctly positioning the images for each menu item using CSS.

2. Whilst this is not a lesson in CSS sprites per se, I need to mention CSS selectors will help correctly place an image as a background of the relevant menu item. One can either use unique **classes** for each menu item or use an **id** for each menu items. The latter is really quite easy to use and is my preferred method so that is what I used.

3. Except for the single image being used as the background for all the main menu items, there is nothing special about the menu we will be constructing. It is still going to be made using an unordered list, i.e. **
. Since we want each of our 7 row menu items to have unique **id, we have a Wayfinder parameter that is just right for the job. We looked at it in lesson #6 and it is the **&rowIdPrefix**. Using this parameter I was able to assign unique **ids** to each of my 7 menu items.

4. I made sure that the template chunk for my **&rowTpl** had the placeholder **[+wf.id+]**. If you are still wondering why we need this placeholder then you really need to review lesson #6. We will also need an **&outerTpl** and that is all there is to it. We will be making use of Wayfinder's automatically applied CSS class "**active**". This will help us correctly position our sprite image to display the active states of each of the menu items. Below is the Wayfinder call I used:

```
[[Wayfinder? &startId=`0` &outerTpl=`menuContainer` &rowTpl=`menuRows` &level=`1`
&excludeDocs=`16` &rowIdPrefix=`nav`]]
```

Note that I am just reusing the **&outerTpl** and the **&rowTpl** that we used in our latest main menu example. I have given the **&rowIdPrefix** the value "**nav**". This means that my menu items will each get a unique id that consists of the prefix "**nav**" and the respective MODx document id, e.g. **id="nav1"**, **id="nav2"**, etc.

Armed with the above, I was now ready to start styling the menu items so that they would each have a unique background image depending on the state they were in. I had the information regarding the total height and width of the sprite image and also of the individual images that make up the sprite. Each individual image's dimension was about 123px high by 113px wide. The height of the image, in CSS sprites, plays a part in determining the vertical position of the image, i.e. the y-axis position. The same is true for the width with regard to the horizontal position of the image, i.e. the x-axis.

Therefore, to position the image to show the "house" icon as the background image of the document "Home" in the normal state (top part of sprite image), we would need, using CSS, to position the image by its top left corner. Hence, the coordinates, so to speak, of the image should be 0px 0px for the x and y axes respectively. For the same menu item, in the hover state, the image need not travel on the x-axis since we still want the "house" icon visible, only a different "house" icon, i.e. the one for the hover state. This is the one in the middle part of the sprite image. The image therefore needs to travel only on its y-axis. Since it needs to travel up, or in other words, it needs to be pulled up, we need a negative y-axis value. The question is, how high up it needs to travel to reveal the middle part of the sprite image, i.e. the hover part. The answer is that it needs to travel the length of the height of the image in the normal state, i.e. **123px**. In the CSS, we will write **-123px**. In order to get the active state, it will need to travel the total of the heights of the two images above it, i.e. the normal and the hover states of the image. It therefore needs to travel **123x2=246px** - which in our CSS we will write as **-226px** since it is being pulled back.

Using similar arguments, in order to show the normal state of the "people" icon for the menu item "About" the image needs to be both pulled toward the left to reveal the correct icon, hence a negative value. It needs to be pulled to the left a distance at least equal to the width of the preceding image, i.e. the "house" icon. It therefore needs to be positioned at **-113px**. On the y-axis it need not move on the normal state so it will remain at 0.

As mentioned earlier, this is not meant to be a lesson in how to make CSS sprites. With the above information you will hopefully understand how the sprite image should be positioned with respect to the menu items. Below is the CSS I used to correctly position the sprite image with regard to the menu items.

```
#main-navigation-sprite ul {list-style-type:none; margin: 0; padding: 0;}
#main-navigation-sprite ul li a {display:block;
height:123px; width:113px; text-indent: -9999px; float: left; text-decoration: none;}

/* menu item "home" */
```

```
li#nav1 a { background:url(images/sprite-image.png) no-repeat 0 0;}
li#nav1.active a{background-position: 0 -246px;}
li#nav1 a:hover {background-position: 0 -123px;}

/* menu item "about" */
li#nav2 a { background:url(images/sprite-image.png) no-repeat -113px 0;}
li#nav2.active a{background-position: -113px -246px;}
li#nav2 a:hover {background-position: -113px -123px;}

/* menu item "reservations" */
li#nav3 a { background:url(images/sprite-image.png) no-repeat -226px 0;}
li#nav3.active a{background-position: -226px -246px;}
li#nav3 a:hover {background-position: -226px -123px;}

/* menu item "menus" */
li#nav4 a { background:url(images/sprite-image.png) no-repeat -339px 0;}
li#nav4.active a{background-position: -339px -246px;}
li#nav4 a:hover {background-position: -339px -123px;}

/* menu item "awards" */
li#nav5 a { background:url(images/sprite-image.png) no-repeat -452px 0;}
li#nav5.active a{background-position: -452px -246px;}
li#nav5 a:hover {background-position: -452px -123px;}

/* menu item "locations" */
li#nav6 a { background:url(images/sprite-image.png) no-repeat -565px 0;}
li#nav6.active a{background-position: -565px -246px;}
li#nav6 a:hover {background-position: -565px -123px;}

/* menu item "contacts" */
li#nav7 a { background:url(images/sprite-image.png) no-repeat -678px 0;}
li#nav7.active a{background-position: -678px -246px;}
li#nav7 a:hover {background-position: -678px -123px;}
```

The final result

CSS sprite menu implementation in MODx using Wayfinder

CSS sprite menu implementation in MODx using Wayfinder

That's it folks. I sincerely hope that you have learnt something useful in these examples. In the next chapter we will look at some tips for troubleshooting our menus.

CHAPTER 16: TROUBLESHOOTING

1. Help, no menu is output!

- Check that you have actually called Wayfinder in your document/template
- Did you forget to specify **&startId**?
- Check your spellings for **&startId**. **&startid** will cause errors/not work
- Check that you are using backticks (``) around your parameter values and not single quotes ('')
- Try disabling CSS when viewing your site. If the menu shows then the problem is in your CSS. Check for things such as text-indent:9999px; and edit as appropriate
- If using a **&startId** value other than the root does the container parent have child documents? If not then there is nothing to output
- Did you create your Wayfinder template chunks correctly? Make sure you have **[+wrapper+]** in your templates
- Are your documents published and set to "show in menu"?
- Make sure you have the placeholder **[+wf.linktext+]** in your row level templates
- Use **&debug** to find out what exactly Wayfinder is doing/processing

2. My JavaScript (e.g. jQuery) menu does not show

- There could be a conflict between JavaScript libraries. This can happen for example if you have installed and are using QuickEdit in MODx and **ARE NOT** running jQuery in non-conflict mode
- Is JavaScript enabled in your browser?

3. My menu is not output the way I want (indentations, positions, etc are off)

- That's a CSS issue - not a Wayfinder one

4. Wayfinder is not using the template chunks I want

- Check the spellings of the chunk names - they are case sensitive too
- Make sure you have specified the chunks you want to use in your menu in your Wayfinder call

5. I cannot see my sub-menus

- Check the **&level** parameter value; is it **>1**?
- Does the document you are viewing have child documents? Are those documents published?

6. I want to sort my menu differently

- You can use menu index to order your menu items
- If you want to you may also use **&sortBy** and **&sortOrder** parameters to control the order

7. Using Wayfinder with another nested snippet does not work

- Make sure you alternate the caching status of the snippets. For instance call Wayfinder uncached **[!!]** and the nested snippet cached **[[]]**

8. I would like to apply a class to my row level menu items

- You can use the **&rowClass** parameter or include your class directly to the row level Wayfinder template

9. I would like the document specified as the &startId to also show in my menu

- Set **&displayStart=`TRUE`** and define a **&startItemTpl** if you don't want Wayfinder to use its default one

10. Wayfinder is using the &outerTpl to construct the containers of my sub-menu; I don't want this

- Make sure you define a **&innerTpl**

- Same argument applies to **&rowTpl** with respect to your sub-menu rows; create an **&innerRowTpl** to stop Wayfinder from using **&rowTpl** to construct to construct your sub-menu's row items

11. Wayfinder is applying the "wrong" template to my menu item

- This could be a case of the template processing order in Wayfinder. In case a menu item qualifies to be constructed using any of the templates you has specified, Wayfinder's template processing order kicks in. For instance if you specify both a **&parentRowHereTpl** and a **&parentRowTpl**, the former will have precedence and hence will be applied

12. I don't want certain documents to appear in a certain Wayfinder menu but would like to use them in another

- You can achieve this using **&excludeDocs** or **&includeDocs**

13. I want more than one level of my menu to be shown

- Use **&level** with a value **>1**

14. Wayfinder is not outputting my classes

- Make sure you are using the **[+wf.classes+]** or **[+wf.classnames+]** placeholders in your Wayfinder templates

15. The links in my menu are not clickable!

- Remember to include the **[+wf.link+]** placeholder in the relevant Wayfinder row level templates and that your row level templates are properly constructed to output links

16. I do not want my sub-menus shown unless their parent menu item is active

- Use and set **&hideSubMenus=`TRUE`**

17. I want to limit the number of documents output to the menu

- You could use the **&limit** parameter

18. What could I use to output my menu with category/heading information?

- You could use the **&categoryFoldersTpl** parameter

19. What if I want classes applied to my first and/or last menu items?

- Use **&firstClass** and **&lastClass** parameters respectively. The latter is applied by default with the value "**last**"

20. What is this "active" class being applied to my menu items by default?

- That is coming from the **&hereClass** which is applied by default. If you do no want it you can specify **&hereClass=``** in your Wayfinder call. You can change the class name if you like by doing this **&hereClass=`name_you_want`**

- The same principles can be applied to the other Wayfinder classes where applicable

21. I would like my parent menu items to be styled/constructed differently

- You have a choice of parent row level parameters to choose from. These are: **&parentClass**, **&parentRowTpl**, **&parentRowHereTpl** and **&activeParentRowTpl**

22. I would like my sub-menus to be styled/constructed differently

- You have a choice of sub-menu specific row level parameters to choose from. These are: **&innerTpl**, **&innerRowTpl**, **&innerClass** and **&innerHereTpl**

23. What do I use to create/achieve a breadcrumb-like menu?

- If you want this for all the active menu items *including the document* you are currently browsing together with its parent and grandparent documents, you need to take advantage of the **&hereClass** which applies the class "**active**" to all currently active documents inclusive of their parent and grandparent documents.

- If you want this *for only the parent, grandparent documents, etc* of the current document,

you need to use **&activeParentRowTpl** which will only be applied to the currently active parent documents

24. How can I style/apply a template to the current document/menu item ONLY and not any other?

- You can use the **&selfClass** and **&hereTpl** respectively.
- If you wish you can also use the **&innerHereTpl** which is an equivalent of the **&hereTpl** for current sub-menu items. Note that **&hereTpl** if defined will be used in all cases unless you specify **&innerHereTpl**
- The **&parentRowHereTpl** is the equivalent of **&hereTpl** for parent documents only. To use it you must specify it otherwise **&hereTpl** will be used if specified

25. How do I make Wayfinder output full links and not relative ones?

- Use **&fullLink=`TRUE`**

26. I want to give my menu items unique ids

- Use **&rowIdPrefix** and don't forget to use the **[+wf.id+]** placeholder in your row level templates

27. How do I make Wayfinder include in its output documents whose "show in menu" box is UN-TICKED (i.e. not to be shown in menu)

- Use **&ignoreHidden=`TRUE`**

28. I want to apply a Wayfinder class to my menu and sub-menu containers (i.e. &outerTpl and innerTpl respectively)

- Use **&outerClass** and **&innerClass** for the **&outerTpl** and **&innerTpl** respectively.
- You could also apply the class directly to the respective template chunks

29. Which Wayfinder templates are always required?

- The **&outerTpl** and the **&rowTpl** for the menu container and row level items respectively. In

addition, if **&displayStart=`TRUE`** then **&startItemTpl** is required. Wayfinder has in-built templates for all the three parameters that it will use by default when/if applicable, i.e. when you don't specify define your own

30. Can I apply a different class to my weblinks?

- Yes you can - use **&webLinkClass**

31. Can I use MODx document variables within Wayfinder?

- Sure. Just reference them using the syntax **[*document_field_name*]**. For instance **[*parent*]** or **[*longtitle*]**

32. Can I use MODx TVs with Wayfinder?

- Yes you can. Just reference the TV using the following syntax **[+TV_name+]**. For instance **[+myTV+]**

33. What about including the description and/or the introtext of my documents in my Wayfinder menu?

- Use the placeholders **[+wf.description+]** and **[+wf.introtext+]** respectively

34. How do I instruct Wayfinder to include a link to a CSS and/or a JavaScript file in the HEAD section of my document's template?

- Use the parameters **cssTpl** and **&jsTpl** respectively

35. How can I call Wayfinder in one location and have it place its output in another?

- You will need to use the **&ph** parameter

36. How can I show a main menu in one location of my template and have its sub-menu shown in another location of my template?

- You could achieve this using Wayfinder and the snippet UltimateParent

37. How do I debug Wayfinder's output/menus

- You use and set the parameter **&debug=`TRUE`**

38. How can find more information/help about Wayfinder?

- http://www.muddydogpaws.com
- http://modxcms.com/forums/index.php/board,182.0.html

CHAPTER 17: APPENDIX

Wayfinder Parameters

Parameter	Description	Default
&startId	The starting point for the menu (document ID)	current docId
&level	The depth to build the menu (0 goes through all levels)	0
&ignoreHidden	Ignore the "show in menu" checkbox for documents and include them in the menu	FALSE
&ph	Name of a placeholder to output the menu to instead of directly returning the output results where the Wayfinder snippet is called	none
&debug	Turn on debug mode for extra troubleshooting	FALSE
&displayStart	If set to TRUE will cause Wayfinder to output the document used as the **&startId** using the template **&startItemTpl**. The parameter only works if the **&startId** is not equal to zero	FALSE
&limit	The limit parameter causes Wayfinder to only process the number of items specified per level	0
&excludeDocs	Acts as a filter and will remove the documents specified in this parameter from the output. The **&startId** is still required	none
&includeDocs	Acts as a filter and will limit the output to only the documents specified in this parameter. The **&startId** is still required	none
&fullLink	If set to TRUE outputs the full URL instead of a relative link	FALSE
&config	Use to specify the name of a configuration file that contains the parameters and templates you want Wayfinder to use to output a menu	default
&hideSubMenus	If set to TRUE will only output the active sub-menu	FALSE
&removeNewLines	Set TRUE to remove newline characters from output	FALSE
&textOfLinks	The document field you want the actual link text to be. Use the following values: menutitle, id, pagetitle, description, parent, alias, longtitle, introtext	menutitle
&titleOfLinks	The document field you want the title of your links to be. Use the	pagetitle

	following values: menutitle, id, pagetitle, description, parent, alias, longtitle, introtext	
&rowIdPrefix	If set, creates a unique id for each item (id is value of rowIdPrefix + document id)	none
&useWeblinkUrl	If set to TRUE the link specified in the weblink will be output to the placeholder **[+wf.link+]**. If set to FALSE the document's MODx link will be output instead	TRUE
&showSubDocCount	If set to TRUE the number of documents in each folder will be output to the placeholder **[+wf.subitemcount+]**	FALSE
&sortOrder	Allows the menu to be sorted in either ascending or descending order.	ASC
&sortBy	Specify any of the following fields to sort the menu: id, menutitle, pagetitle, introtext, menuindex, published, hidemenu, parent, isfolder, description, alias, longtitle, type, template, random	menuindex
	Template Chunks for Menu Layout	
&outerTpl	The template chunk for the outer most container *placeholders:* **[+wf.classes+]** - where classes specified will be inserted (includes **class=" "**) **[+wf.classnames+]** - outputs just the class names (without **class=" "**) **[+wf.wrapper+]** - where inner content will be inserted *example:* **<ul [+wf.classes+]>[+wf.wrapper+] **	« See Example
&rowTpl	The template chunk for the row items *placeholders:* **[+wf.classes+]** - where classes specified will be inserted (includes **class=" "**) **[+wf.classnames+]** - outputs the just the class names (without **class=" "**) **[+wf.link+]** - the href value for your link **[+wf.title+]** - text for the link title **[+wf.linktext+]** - text for the link display **[+wf.wrapper+]** - where to insert a sub-menu **[+wf.id+]** - where to insert unique id **[+wf.attributes+]** - where to insert link attributes **[+wf.docid+]** - the document identifier for the current item **[+wf.subitemcount+]** - displays the number of items in a folder **[+wf.description+]** - output the description field **[+wf.introtext+]** - output the introtext field *example:* **<li[+wf.id+][+wf.classes+]><a href="[+wf.link+]"**	« See Example

	title="[+wf.title+]" **[+wf.attributes+]>[+wf.linktext+][+wf.wrapper+] **	
&parentRowHereTpl	The template chunk for the current document if it is a folder *placeholders:* same as **&rowTpl** *example:* **<li[+wf.classes+]>[+wf.linktext+]»[+wf.wrapper+] **	none
&parentRowTpl	the template chunk for any document that is a folder *placeholders:* same as **&rowTpl** *example:* **<li[+wf.classes+]>[+wf.linktext+]»[+wf.wrapper+] **	none
&hereTpl	The template chunk for the current document *placeholders:* same as **&rowTpl** *example:* **<li[+wf.classes+]>[+wf.linktext+][+wf.wrapper+] **	none
&innerTpl	The template chunk for the any subfolders listed *placeholders:* same as **&outerTpl** *example:* **<ul [+wf.classes+]>[+wf.wrapper+] **	none
&innerRowTpl	The template chunk for the row items in a subfolder *placeholders:* same as **&rowTpl** *example:* **<li[+wf.classes+]>[+wf.linktext+][+wf.wrapper+] **	none
&innerHereTpl	The template chunk for the current document if in a subfolder *placeholders:* same as **&rowTpl** *example:* **<li[+wf.classes+]>[+wf.linktext+][+wf.wrapper+] **	none
&activeParentRowTpl	The template chunk for the items that are folders and are currently active in the tree *placeholders:* same as **&rowTpl** *example:* **<li[+wf.classes+]>** **[+wf.linktext+][+wf.wrapper+] **	none
&categoryFoldersTpl	The template chunk for category folders, category folders are determined by setting the template to blank or by setting the link attributes field to **rel="category"** *placeholders:* same as **&rowTpl** *example:* **<li[+wf.classes+]>[+wf.linktext+][+wf.wrapper+] **	none
&startItemTpl	The template is used for the **&startId** document when the parameter **&displayStart** is set to TRUE *placeholders:* same as **&rowTpl**	« See Example

example: **<h2>[+wf.linktext+]</h2>[+wf.wrapper+]**

Class names to be assigned		
&firstClass	CSS class denoting the first item at a given menu level	none
&lastClass	CSS class denoting the last item at a given menu level	last
&hereClass	CSS class denoting the "you are here" state all the way up the tree	active
&selfClass	CSS class denoting the "you are here" state for current doc only	none
&parentClass	CSS class denoting that the menu item is a folder (has children)	none
&rowClass	CSS class applied to each output row	none
&levelClass	CSS class denoting each output rows level. The level number will be added to the specified class (i.e. level1, level2, level3, etc...)	none
&outerClass	CSS class for the **&outerTpl**	none
&innerClass	CSS class for the **&innerTpl**	none
&webLinkClass	CSS class for weblinks	none
CSS & JavaScript to include with menu		
&cssTpl	Name of a chunk containing CSS you would like added to the page	FALSE
&jsTpl	Name of a chunk containing JavaScript you would like added to the page	FALSE

Credits: adapted from the Wayfinder documentation available with the Wayfinder installation files and http://www.muddydogpaws.com/development/wayfinder/parameters.html.

Wayfinder Template processing order

1. **&startItemTpl**
2. **&parentRowHereTpl**
3. **&innerHereTpl**
4. **&hereTpl**
5. **&activeParentRowTpl**
6. **&categoryFoldersTpl**
7. **&parentRowTpl**
8. **&innerRowTpl**
9. **&rowTpl**

Decreasing priority

The higher on the list the template is the higher processing priority it has. Thus, **&startItemTpl** will be applied first (if applicable) then the **&parentRowHereTpl**, etc, etc.

Credits: Adapted from http://www.muddydogpaws.com/development/wayfinder/parameters.html.

Index