# Extending Contribute

Advanced users and web administrators of Macromedia Dreamweaver MX can customize and enhance the capabilities of Macromedia Contribute to meet user requirements. Contribute inherits from Dreamweaver MX many of the features that you can extend.

Contribute also has the following intrinsic features that you can extend:

- The Welcome page

  The Welcome page is an HTML page that you can customize to add branding information and let users quickly identify and launch new third-party features.

- Starter pages

  Contribute provides a set of starter pages, which are prototype pages that you can quickly modify to develop new web pages of your own. You can also create and add pages to the set of Contribute starter pages, thereby developing templates that are specifically designed for your website.

- The How Do I panel

  The How Do I panel provides a Contribute tutorial and information on how to get started, how to create and modify web pages, and how to administer a website. A third party can add task information to the How Do I panel to provide branding, launch new features, and teach users how to use new features.

Extending inherited Dreamweaver MX features

Changing the Welcome page

Adding starter pages

Extending the How Do I panel

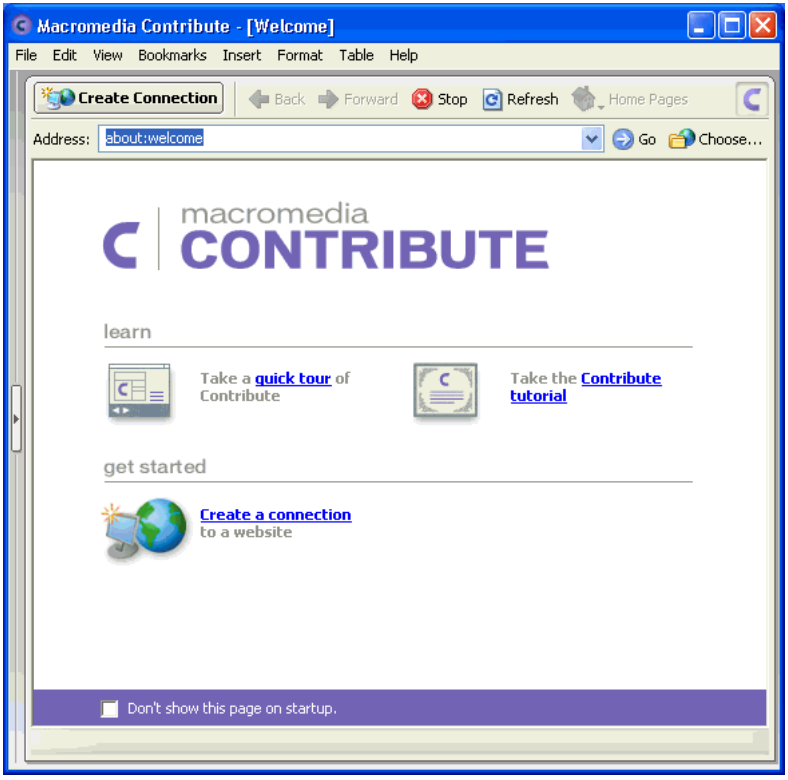
## Extending inherited Dreamweaver MX features

Contribute inherits from Dreamweaver MX the following features that you can extend. You can implement these features for Contribute in the same way that you implement them for Dreamweaver MX. For information on how to create these types of extensions, see Extending Dreamweaver on the Dreamweaver MX Help menu.

| Feature | Extensibility | Example |
|---|---|---|
| Menus | You can customize and extend all product menus and context menus in Contribute. Menu items can trigger any Contribute API function currently within the menu structures, and can launch new extensions, such as objects, commands, and floating panels. | Add a new item to the Help menu to launch a browser that displays information specifically for your company. |
| Toolbars | Similar to menus, you can modify Contribute toolbars and define additional toolbars. Toolbars can do anything that menus can do. In addition, toolbars can provide visual icons to trigger any action, including objects, commands, and floating panels. | Add a toolbar of icons that users can click to insert images that are designed for your pages. |
| Objects | You can add new objects to Contribute. The new object can optionally have a modal user interface built using HTML, and can insert any block of code or text within the current document. By default, new objects appear at the bottom of the Insert menu. | Add a new object to insert some common HTML code, such as copyright information or a standard set of links. |
| Commands | You can add new Commands, which are similar to objects, but more powerful. Commands are typically used to modify the current page using the JavaScript Document Object Model. Command user interfaces are also modal. By default, new Commands appear at the bottom of the Format menu. | Add a new command to let users change selected text to uppercase or lowercase. |
| Floating Panels | You can add floating panels, which are similar to commands, but which are modeless. You can configure floating panels to receive events based on user interactions with the page. You can also use floating panels to display information or modify the page. | Create a floating panel that warns the user when the number of words on a page exceeds a certain limit. |
| Translators | You can write a translator to automatically change the appearance of a page, and to control the editing experience of the page. Translators are typically used to display tags with special meaning, and to prevent editing of portions of a page. | Write a translator to cause ‹ourtag›, a company-designed tag, to display as an image. |
| Third Party Tags | You can define third-party tags using objects to insert them and translators to display them. Contribute recognizes such tags and displays them correctly. | Design your own tag, ‹ourtag›, to implement company-specific functionality. |
| JavaScript Extensions | You can add JavaScript extensions, which are extensions that you can write in any language that can be compiled as a binary dynamic link library (DLL). Extensions can provide new APIs that you can call from JavaScript in any of the features described in this table, and can display custom user-interface elements. | Develop a dynamic link library to access the company database and return specific information for an object to insert. |

## Changing the Welcome page

The Welcome page is an HTML page that appears when you launch Contribute. It initially appears as shown in the following figure. When you create a connection to a website, however, Contribute adds a link for the website to the Welcome page.




The welcome.htm file contains the Welcome page. Contribute generates the welcome.htm file from the Dreamweaver template file, welcome.dwt, and from the previous generation of the welcome.htm file, if one exists.

***Tip:*** Before changing the Welcome page, save the welcome.dwt file with a different name so that you can restore the original file, if necessary.

To change the Welcome page, use Dreamweaver to edit the welcome.dwt file that is located in the Contribute Configuration/Content/Welcome folder. After you make your changes, save them to the welcome.dwt file. The next time you start Contribute, the program constructs a new welcome.htm file that includes your changes to the template. To distribute your new Welcome page to multiple users, you must copy the welcome.dwt file to the Contribute Configuration/Content/Welcome folder of each user who needs the new page.

When Contribute generates the Welcome page from the welcome.dwt file, it creates a welcome.htm file and a copy of the welcome.dwt file in the user's Welcome folder. On a single-user operating system, these files are located only in the Contribute Configuration/Content/Welcome folder. On multiuser operating systems, these files are located in the following locations:

| Platform | User Configuration Folder |
|---|---|
| Macintosh OS X | MacHD:Users:username:Library:Application Support:Macromedia:Contribute:Configuration: Content:Welcome |
| Windows 2000, Windows XP | C:/Documents and Settings/username/Application Data/Macromedia/Contribute/Configuration/Content/Welcome |
| Windows NT | C:/WinNT/profiles/username/Application Data/Macromedia/Contribute/Configuration/Content/Welcome |

***Note:*** To change the Welcome page on both single-user and multiuser operating systems, you must edit the welcome.dwt file in the Contribute Configuration/Content/Welcome folder.

## Adding starter pages

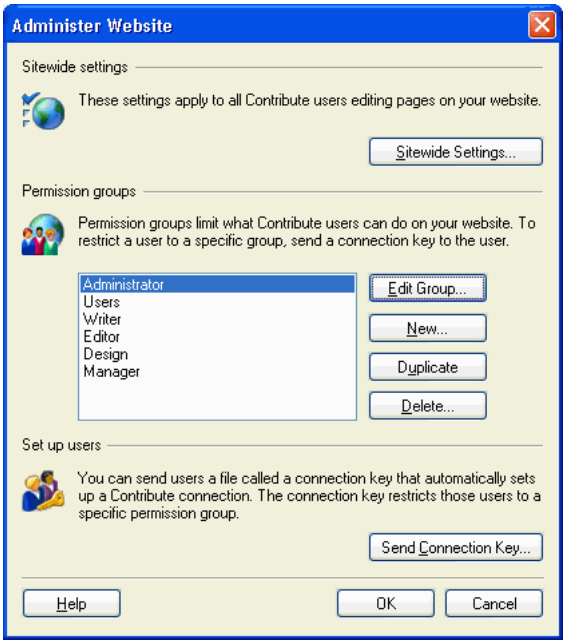
When you create a web page with Contribute, you can begin with a blank page or a copy of the current page, or choose a template or a page from a predefined set of sample pages. You can extend this set of starter pages by adding your own pages to it.

To add your own page to the set of Contribute starter pages, create the page that you want to add. (For information on how to create a page in a website, see the instructions in the How Do I panel.) After you create the page, perform the following steps to add it to the starter pages as a template.

**To add a page to the Contribute starter pages:**

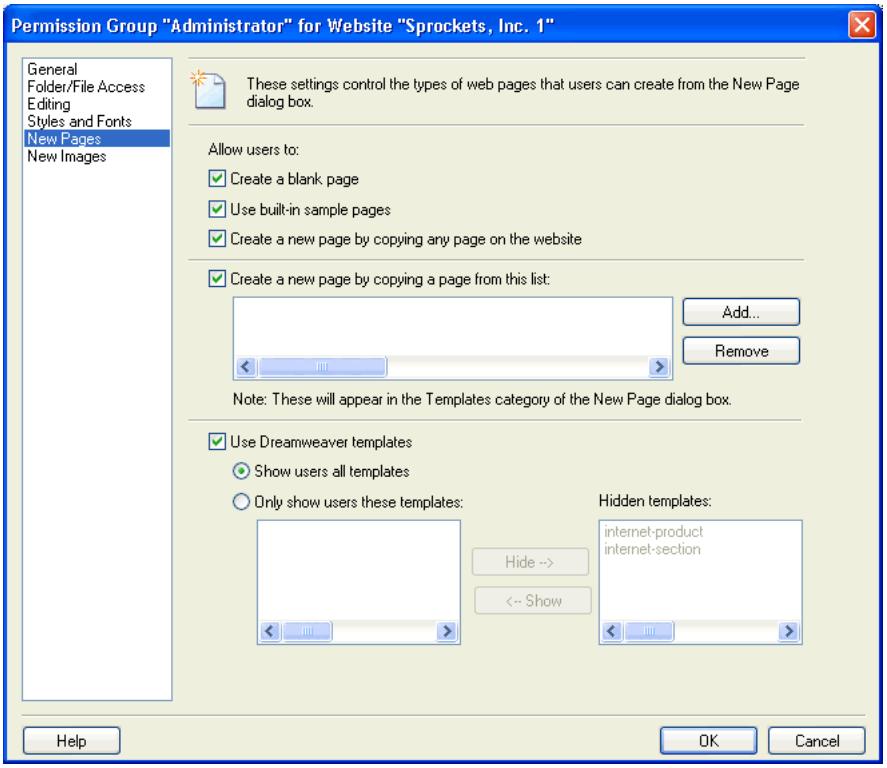
1 From the Edit menu, select **Administer Websites** and select the website to which you want to add the new starter page.

The Administer Website dialog box appears.

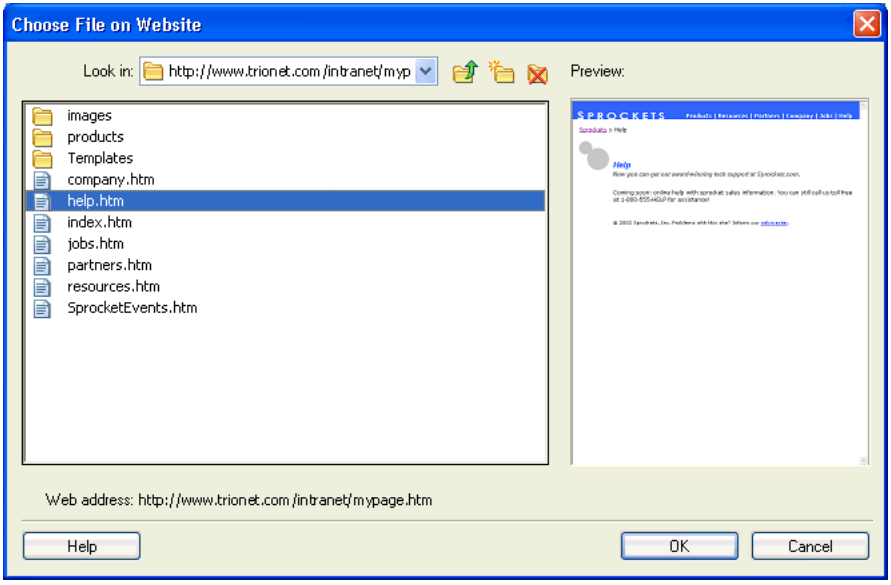



2 In the Administer Website dialog box, select the target group in the Permission groups list and click the **Edit Group** button to invoke the Permission Group Administrator dialog box for your website.

**3** Select **New Pages** from the list in the sidebar panel, and select the **Create a new page by copying a page from this list** option.



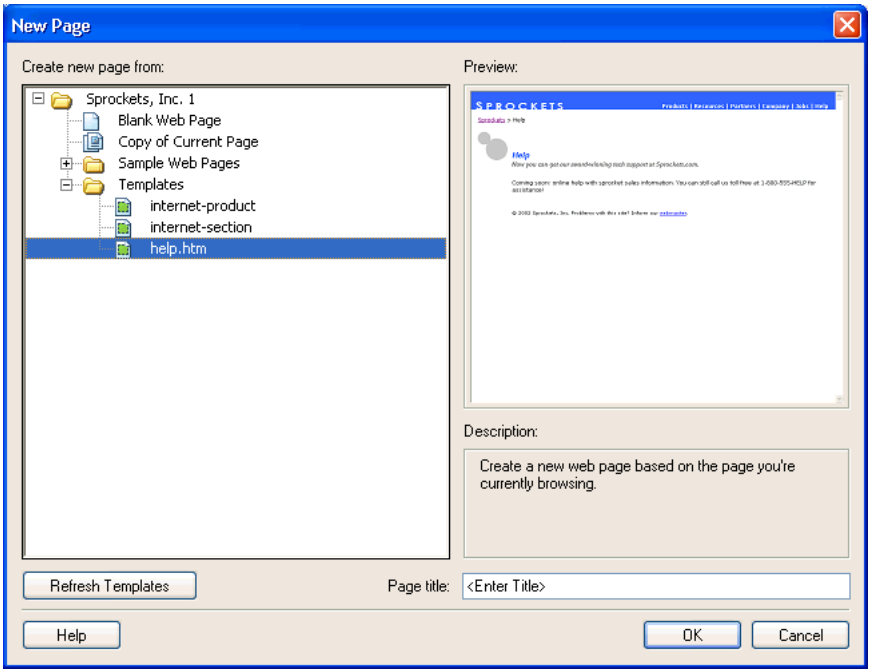
4 If the page that you want to add does not appear in the list box, click the **Add** button and locate the page in the Choose File on Website dialog box.




5 Click **OK** to add the selected page to the list box.

6 Click ***OK*** in the Permission Group Administrator dialog box to add the new page to the set of starter pages.

Your new page appears in the Templates folder of the New Page dialog box. The new page is available as a starter page to users in the specified group who subsequently contribute content to the website.

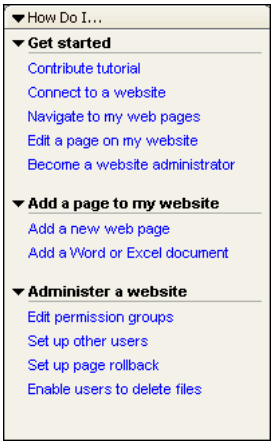



The HTML file for the page is located in the website folder.

To distribute the new starter page to other websites, you must repeat this procedure for each website.

# Extending the How Do I panel

The How Do I panel tells the user how to get started with Contribute and how to accomplish common tasks, such as connecting to a website, adding a page to a website, or editing a page. The How Do I panel appears in the lower section of the sidebar in the Contribute workspace. You can enable or disable the Sidebar by clicking the Sidebar entry on the View menu. The following figure shows the list of topics that appear when you open the How Do I panel.




You can extend the How Do I panel to add information that you developed for your organization. You can also change some aspects of the default style and layout for the How Do I panel.

The following files determine the content, style, and layout of the How Do I panel.

| File | Purpose |
|---|---|
| Tasks.xml | Lists the categories and tasks that display in the How Do I panel |
| Task files | Contain the procedural instructions that display when a user clicks a task in the How Do I panel |
| Tasks.css | Defines the styles that determine how text displays in the How Do I panel |
| TasksLayout.xml | Defines the structural layout of the How Do I panel |

These files are located in the Contribute Configuration/Content/Tasks folder.

## Using the XML tags of the tasks.xml file

To add to or change the content of the How Do I panel, you must add or change XML tags in the Tasks.xml file. You use three tags to create the content of the Tasks.xml file: the `tasks` tag, the `category` tag, and the `task` tag. The `tasks` tag groups the set of tasks being defined under an identifier. The `category` tag groups related tasks under a heading, such as “Get Started”. The `task` tag specifies the name of a task, such as “Connect to a website”, and is a link to the page that explains how to accomplish the task.

The following excerpt from the Tasks.xml file shows the relationship between these tags.

```
<?xml version='1.0' encoding='utf-8'?>
<tasks id="tasks">
  <category name="Get started" id="0">
    <task name="Contribute tutorial " file="task2.htm" id="1"/>
    <task name="Connect to a website" file="task14.htm" id="2"/>
    <task name="Navigate to my web pages" file="task17.htm" id="3"/>
    <task name="Edit a page on my website " file="task18.htm" id="4"/>
    <task name="Become a website administrator" file="task19.htm" id="5"/>
  </category>
...
</tasks>
```

The following sections describe each of these tags in more detail.

## <tasks>

### Description

The `tasks` tag groups a set of categories and tasks and assigns an identifier to the set.

### Attributes

`id` is a required string that names the set of categories and tasks that are defined in the file. The `id` must be unique within the file.

### Contents

`category` tags

### Container

None.

### Example

```
<tasks id="tasks">
```

## <category>

### Description

The `category` tag groups a set of related tasks. You can expand or collapse a category in the How Do I panel to show or hide the tasks within the category. You can also enable or disable a category to cause the entire category to either appear or not appear within the How Do I panel.

### Attributes

```
{enabled}, id, name, {stylecollapsed}, {styleexpanded}, {update}
```

`enabled` is an optional attribute. The `enabled` attribute specifies whether this section is shown in the How Do I panel. The value of this attribute is a JavaScript expression that evaluates to a value of `true` or `false`. If you do not specify a value, the default value is `true`. For example, the following `category` tag specifies that the category appears in the How Do I panel only when you edit a web page.

```
  <category name="Add content to a web page"
    enabled="CCWorkspaceManager.getManager(dw.getDocumentDOM()).getState() ==
    'edit'" id="9">
```

`id` is a required string that identifies the category. The `id` must be unique within the file.

name is required. Contribute displays the category name in the How Do I panel next to a right-arrow that you can expand to show a list of tasks, or a down-arrow that you can collapse to hide the list of subordinate tasks.

stylecollapsed is an optional attribute that has a default value of clsCategoryCollapsed. The style clsCategoryCollapsed is defined in the Tasks.css file. To specify a different style, you must define the style in the Tasks.css file and then set this attribute to use it.

styleexpanded is an optional attribute that has a default value of clsCategoryExpanded. The style clsCategoryExpanded is defined in the Tasks.css file. To specify a different style, you must define the style in the Tasks.css file and then set this attribute to use it.

update is an optional attribute that specifies a comma-separated list of values that describes when this item enabler runs. The possible values for the update attribute, in order from least to most common (and from least to most processor intensive) are onWorkspaceChange, onURLChange, onEdit, onSelChange, and onEveryIdle. The onWorkspaceChange event occurs when the user switches between browse and edit modes. The onURLChange event occurs when the user browses to a different location. The onEdit event occurs when the user makes a change to a draft. The onSelChange event occurs when the user changes the selection within a draft. The onEveryIdle event occurs when Contribute is idle. The default value for the update attribute is onWorkspaceChange.

#### Contents

task tags

#### Container

tasks tag

#### Example

```
<category name="Finish up "
  enabled="CCWorkspaceManager.getManager(dw.getDocumentDOM()).getState() ==
  'edit'" id="20">
```

### <task>

#### Description

The task tag describes a particular task and acts as a link to a page that tells you how to accomplish the task.

#### Attributes

{command}, {commandfile}, {enabled}, {file}, id, name, {style}, {update}

command is an optional attribute, but you must specify one of the following attributes per task tag: the command attribute, the commandfile attribute, or the file attribute. The command attribute is a string that specifies a segment of JavaScript to run when the user clicks the link.

commandfile is an optional attribute, but you must specify one of the following attributes per task tag: the command attribute, the commandfile attribute, or the file attribute. The commandfile attribute is a string that names a command file that executes JavaScript, and any arguments that the command file requires. For example, commandfile="'commandFile.htm', arg1, arg2". The command file must be in the Contribute Configuration/Commands folder.

enabled is an optional attribute that Contribute uses to determine if this task appears in the How Do I panel. This attribute applies only if the enabled attribute is true for the parent category. The default value is true.

`file` is an optional attribute, but you must specify one of the following attributes per `task` tag: the `command` attribute, the `commandfile` attribute, or the `file` attribute. The `file` attribute specifies the path, relative to the Configuration/Content/Tasks folder, and file name of the file that contains the task procedures.

`id` is a string identifier for the task. It is required by Contribute for expanding and collapsing `category` tags. Each `id` must be unique within the file.

`name` is a required attribute that describes a task and acts as a link in the How Do I panel.

`style` is an optional attribute that has a default value of `clsTask`, which is defined in the Tasks.css file. You can set this attribute so that this task looks different from other tasks.

`update` is an optional attribute that specifies a comma-separated list of values that describes when this item enabler runs. The possible values for the `update` attribute, in order from least to most common (and from least to most processor intensive) are `onWorkspaceChange`, `onURLChange`, `onEdit`, `onSelChange`, and `onEveryIdle`. The `onWorkspaceChange` event occurs when the user switches between browsing and editing. The `onURLChange` event occurs when the user browses to a different location. The `onEdit` event occurs when the user makes a change to a draft. The `onSelChange` event occurs when the user changes the selection within a draft. The `onEveryIdle` event occurs when Contribute is idle. The default value for the `update` attribute is `onWorkspaceChange`.

#### Contents

None.

#### Container

`category` tag

#### Example

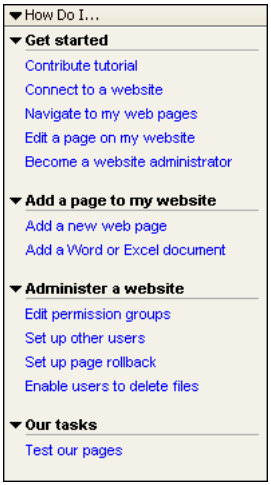
```
<task name="Add an image " file="task31.htm" id="10"/>
```

## Adding categories and tasks

To add a category and related tasks to the How Do I panel, add to the Tasks.xml file a category tag for each category that you want to add, and a task tag for each task that you want to add. For example, the following `category` and `task` tags define a new category, “Our tasks”, and a new task, “Test our pages”.

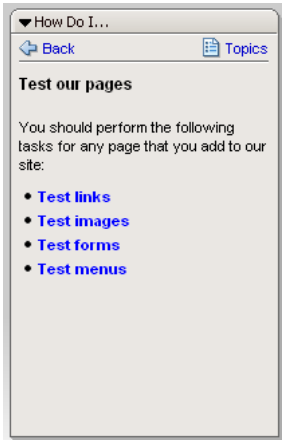
```
<category name="Our tasks" id="o_stuff">
     <task name="Test our pages " file="o_task1.htm" id="os1"/>
</category>
```

Adding these tags to the end of the Tasks.xml file (prior to the `/tasks` tag) causes the new category and task to appear after the category, “Administer a website”, as shown in the following example:




## Adding task procedures

When you click the link for the topic, “Test our pages”, the following content page appears. This page is from the o_task1.htm file, as specified by the `file` attribute in the `task` tag.




The Tasks.css file controls the style of the page. Include the following line in your task file to apply the styles that are defined in the Tasks.css file:

```
<link href="Tasks.css" rel="stylesheet" type="text/css">
```

To create the icons for Back and Topics, which appear at the top of the task pages, use the following table cell definitions (`td` tags), which include the images taskback.gif and taskhome.gif, defined as links in `a` tags:

```
<td width="50%" class="tdcustom" >
  <a class="alinkcustom" href="#" onMouseUp="dw.tasksPalette.back()">
  <img src="images/taskback.gif" alt="Back"  border="0"
    align="absmiddle">Back</a></td>
<td width="50%" align="left" valign="middle" class="tdcustom1">
  <a class="alinkcustom" href="#"
    onMouseUp="dw.tasksPalette.browseToPage('')">
  <img src="images/taskhome.gif" alt="Topics" border="0"
    align="absmiddle">Topics</a></td>
```

The following HTML lines from the o_task1.htm file produce the text that appears in the How Do I panel for the task, "Test our pages". Notice that the text of the four bullets are embedded inside `a` tags, which defines them as links. When the user clicks one of these links, the `onMouseUp` event handler calls the JavaScript function `dw.tasksPalette.browseToPage()`, passing the name of the file to open (o_task2.htm, otask3.htm, and so on).

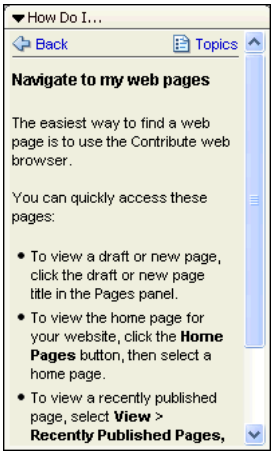
```
<table width="100%" border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td class="heading2"> <p>Test our pages </p>
      <IMG height=1 src="images/pixel_white.gif" width=1></td>
  </tr>
</table>
<table width="100%" border="0" cellspacing="0" cellpadding="1">
  <tr>
    <td>
     <p class="bodytext">You should perform the following tests for any web
     pages that you add to our site.</p>
      <p class="bodytext"></p>
<table width="95%"  cellspacing="2" border="0" cellpadding="1">
<tr valign="top">
  <td align="left" width="8"><img src="images/bullet.gif" vspace="2"></td>
  <td colspan="2"><a  class="alinkcustom" href="#"
  onMouseUp="dw.tasksPalette.browseToPage('o_task2.htm')">Test links</a></td>
</tr>
<tr valign="top">
  <td align="left" width="8"><img src="images/bullet.gif" vspace="2"></td>
  <td colspan="2"><a  class="alinkcustom" href="#"
  onMouseUp="dw.tasksPalette.browseToPage('o_task3.htm')">Test images</a>
  </td>
</tr>
<tr valign="top">
  <td align="left" width="8"><img src="images/bullet.gif" vspace="2"></td>
  <td colspan="2"><a  class="alinkcustom" href="#"
  onMouseUp="dw.tasksPalette.browseToPage('o_task4.htm')">Test forms</a></td>
</tr><tr valign="top">
  <td align="left" width="8"><img src="images/bullet.gif" vspace="2"></td>
  <td colspan="2"><a  class="alinkcustom" href="#"
  onMouseUp="dw.tasksPalette.browseToPage('o_task5.htm')">Test menus</a>
  </td>
</tr>
</table>
</table>
</body>
</html>
```

## Changing the default style

To change default style characteristics for tasks, you must modify the Tasks.css file. The Tasks.css file is a cascading style sheet that defines styles for the various text elements in the How Do I panel. For example, the following style settings specify the appearance of text that has the `body` tag applied.

```
body {
  font-family: Arial;
  font-size: 11px;
  margin: 2px;
  background-color: #F0F0F0;
}
```

You can see the effect of these settings in the following figure, which shows the instructions for the task, "Navigate to my web pages".



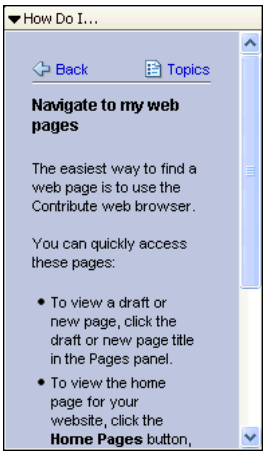

You can change the appearance of text in the How Do I panel by creating a new style and applying it. For example, the following style, called `alt`, has a background color of #BDC6DE, and a margin of 20 pixels (px):

```
.alt {
  font-size: 11px;
  margin: 20px;
  background-color: #BDC6DE;
}
```

To apply the `alt` style, add the `class` attribute to the `body` tag for the task.

```
<body leftmargin="0" topmargin="0" class="alt">
```

These changes produce the following effect on the appearance of the task, “Navigate to my web pages”.

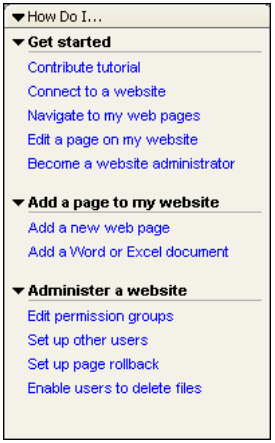



## Changing the default layout

The TasksLayout.xml file defines the layout and spacing of elements in the How Do I panel. You can alter the layout of the How Do I panel by changing settings within the TasksLayout.xml file. For example, the following `category_separator` tag defines the height of spacing between categories as 15:

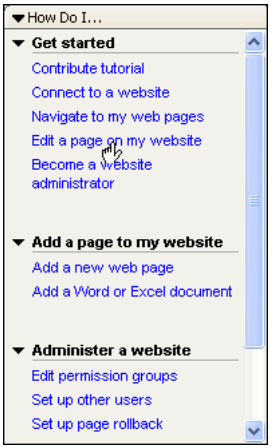
```
<category_separator>
  <![CDATA[
  <table border="0" cellpadding="0" cellspacing="0" width="100%"
  height="15"><tr><td></td></tr></table>]]>
</category_separator>
```

The following figure shows the default layout of categories in the How Do I panel.




***Note:*** Save a copy of the TasksLayout.xml file before making any changes to it so that you can restore the original settings, if necessary.

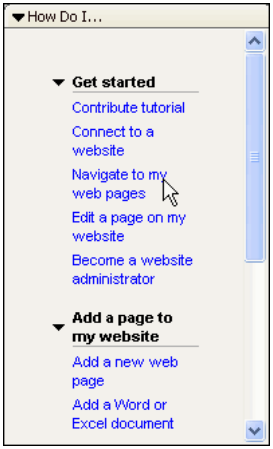
Changing the value of the `height` setting to `30` doubles the space between category sections. The following figure shows the effect of this new setting in the How Do I panel.




Similarly, you can change the space surrounding the text by changing the value for `cellspacing` that is contained in the `tasks_layout` tag, which is shown in the following example.

```
<tasks_layout>
      <main>
        <![CDATA[
        <table border="0" cellpadding="0" cellspacing="0" width="100%">
          <tr>
            <td>{category_sections+category_separators}</td>
          </tr>
        </table>
        ]]>
      </main>
```

Changing the value of `cellspacing` from `0` to `30` produces wide margins in Topics, as shown in the following figure:



## Using JavaScript API functions to navigate

Each of the links in a task procedure file should have an `onMouseUp` event handler that calls a JavaScript function when a user clicks the link. The following JavaScript API functions let you navigate between task procedure files, and expand and collapse categories from within the TasksLayout.xml file. You can call the `dreamweaver.tasksPalette.back()` function to display the previous page in the How Do I panel. The `dreamweaver.tasksPalette.browseToPage()` function lets you display a page from a specified file, and the `dreamweaver.tasksPalette.expandSection()` function lets you expand or collapse a category heading from within the TasksLayout.xml file.

### dreamweaver.tasksPalette.back()

#### Availability

Contribute 1

#### Description

The `tasksPalette.back()` function sets the content of the How Do I panel to the previous page in the How Do I panel history, if there is one. Otherwise, it sets the How Do I panel to Topics.

#### Arguments

None.

#### Returns

Nothing.

#### Example

```
dw.tasksPalette.back()
```

### dreamweaver.tasksPalette.browseToPage()

#### Availability

Contribute 1

#### Description

Displays a page in the How Do I panel from a file whose pathname is specified by the `strRelativePath` argument. If the `strRelativePath` argument is an empty string, the How Do I panel shows Topics with collapsible sections.

#### Arguments

*`strRelativePath`* is a string that specifies the path, relative to the Contribute Configuration/Content/Tasks folder, to a file that contains the page to display in the How Do I panel.

#### Returns

Nothing.

#### Example

The following table cell definition in the task0.htm file calls `dw.tasksPalette.browseToPage()` to open the `task1.htm` file.

```
<td colspan="2"><a class="alinkcustom" href="#"
  onMouseUp="dw.tasksPalette.browseToPage('task1.htm')">Connect to a website
  to edit</a></td>
```

## dreamweaver.tasksPalette.expandSection()

### Availability

Contribute 1

### Description

The `dw.tasksPalette.expandSection()` function expands or collapses a category. Contribute saves the state when the user quits the program. This function is only used from within TasksLayout.xml when the user clicks the category name.

### Arguments

*`bExpand, strItemIdentification`*

*`bExpand`* is `true` if the category should be expanded, `false` if the category should be collapsed.

*`strItemIdentification`* is a string that matches the `category.id` attribute value that must be collapsed or expanded. If the `category.id` attribute does not exist, nothing happens.

### Returns

Nothing.

### Example

```
dw.tasksPalette.expandSection(true, "9")
```