Getting started with Eclipse code templates

Move beyond snippets to increase your productivity

Jeremy Wischusen

October 07, 2008

This tutorial covers the basic use of Eclipse code templates. It takes a broad approach to expose the key concepts and skills that apply to templates regardless of the particular plug-in they're associated with. You will learn about the benefits of templates, how to create and edit them, and how to use them within the Eclipse IDE to increase your productivity.

Before you start

About this tutorial

This tutorial covers the benefits of Eclipse templates, including direct integration with auto-competition, the ability to use predefined and custom variables, context sensitivity, and the enhanced predictability and validity of code. It also shows how to create, edit, and delete templates, how to create a new file using a template, and how to use templates within the editor. Finally, the tutorial provides and some tips and tricks for getting the most out of your templates.

Template functionality varies by plug-in. It would be impossible to cover all the specifics for every plug-in that supports templates in a single tutorial, so to keep things as general as possible, we examine templates in the context of the Web HTML editor plug-in. I chose this because HTML is a simple language and the templates that come with this plug-in are simple and easy to use. Even if you don't have the plug-in installed, the concepts you learn will still be applicable to using templates in general. You can then takes these concepts and apply them to the templates for your specific plug-in.

Objectives

After completing this tutorial, you will be able to:

- Locate templates in the Preferences pane.
- Explain what a template context is.
- Use a template to create a new file.
- Use templates in the Editor window.
- Create you own templates.
- Use variables in templates.

Prerequisites

This tutorial assumes basic familiarity with the Eclipse IDE and that you know how to perform tasks such as setting up a workspace, creating projects and files, and switching perspectives and views.

System requirements

The examples were created in Eclipse V3.4 (Ganymede), but the instructions apply to V3.2 (Europa) and are most likely applicable to versions older than that. In addition to the Eclipse IDE, this tutorial requires the Web Tools Platform (WTP) plug-in (see Resources to find Eclipse and WTP).

Overview of templates

Templates in Eclipse go beyond the simple snippet functionality found in many editors. Eclipse templates allow for quick generation of commonly used code, as well as easy customization. This section outlines some of the more beneficial features of templates in Eclipse.

Benefits of templates

Eclipse templates have many benefits, including integration, variables, and predictability and consistency.

Integration

Eclipse templates are integrated directly into the Eclipse workflow. Templates are available when performing such tasks as creating a new document or working in the editor panel. In a way, they combine the functionality of new document templates and snippet functionality found in many editors. However, as you see in this tutorial, they go far beyond the functionality provided by templates and snippets in most editors.

For example, unlike many editors, in which you have to open a snippets panel, find the snippet, and insert it, in Eclipse, all you have to do is enable auto-completion, start typing the name of the template, and press **Enter**. This is much more efficient.

Variables

Perhaps the most impressive and useful feature of Eclipse is the ability to use variables in templates. It is this feature that allows for the customization within the automation. Each plug-in that includes template functionality comes with its own set of predefined variables. These variables can insert things such as the current date and time up to the name of the method or class that houses the template. The types and how many predefined variables there are vary from plug-in to plug-in, but perhaps the most useful feature of variables in templates is the ability to create custom variables. These custom variables act as placeholders in your template for which you provide the values when you insert the template.

To illustrate the importance of custom variables, let's return to the snippets comparison. With snippets, you may have insert a placeholder in the template such as `[userName]`, then either manually replace all the occurrences within the snippet or use a find-and-replace operation. Although this is a step up from manually typing the whole thing out, it does leave something to be desired. With Eclipse templates, you can define a variable in multiple locations and, when you update the first occurrence in your template, all variable placeholders with the same name are updated. So, regardless of whether you have one occurrence of the variable or 1,000, you have to type it only once. Beats find and replace, hunh?

Predictability and consistency

Finally, aside from these major conveniences, there are a couple of other benefits of templates worth noting: consistency and validity. If you're using a template, you know you're writing the same code in the same way each time you use it. This consistency gives your code enhanced predictability. Furthermore, if you write the template and test it and the template is valid the first time, you have a pretty good chance that each time you use the template, your code will be valid (barring any mistakes while customizing).

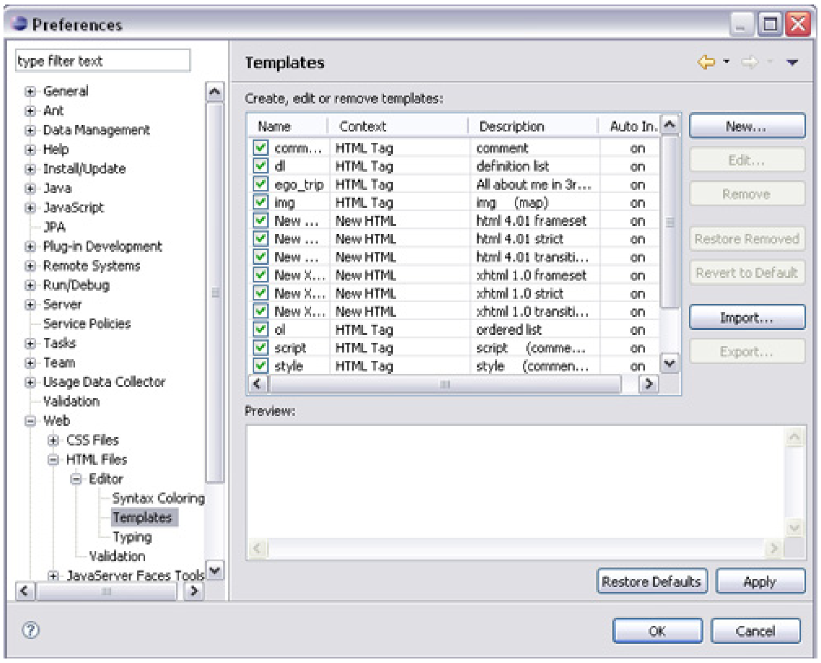
Locating templates

Templates are defined in the Preferences pane under the specific plug-in with which they're associated — for example, **Eclipse > Preferences > *Plug-inName* > *EditorName* > Templates**. To the access the HTML templates:

1. From the **Window** menu (Microsoft® Windows®) or the **Eclipse** menu (Apple Mac OS X V10), choose **Preferences**.
2. In the left pane, expand Web, expand HTML Files, then expand Editor.
3. Click **Templates**.

A grid similar to Figure 1 appears in the details pane.

Figure 1. The Templates folder



Create, edit or remove templates:

Name	Context	Description	Auto In.
✓ comm…	HTML Tag	comment	on
✓ dl	HTML Tag	definition list	on
✓ ego_trip	HTML Tag	All about me in 3r…	on
✓ img	HTML Tag	img (map)	on
✓ New …	New HTML	html 4.01 frameset	on
✓ New …	New HTML	html 4.01 strict	on
✓ New …	New HTML	html 4.01 transiti…	on
✓ New X…	New HTML	xhtml 1.0 frameset	on
✓ New X…	New HTML	xhtml 1.0 strict	on
✓ New X…	New HTML	xhtml 1.0 transiti…	on
✓ ol	HTML Tag	ordered list	on
✓ script	HTML Tag	script (comme…	on
✓ style	HTML Tag	style (commen…	on

This grid has several columns, but pay particular attention to **Name**, **Context**, and **Description**. **Description** is exactly as it sounds: It describes the purpose of the template and is there mostly so you have a quick reference to the purpose of the template. However, the other two headings have some implications in regards to how templates are used.

Template names

The **Name** column contains the name of the template (no surprise there). However, there are a few things about the template name you should be aware of. The name is how you reference the template: It is the name displayed in the code completion window. So, when the code-completion window is displayed, this is the name you must type to access that template.

Template contexts

The *context* of a template determines where a template is available. For example, *New HTML* means that the template will be available when creating a new HTML file. You can also see examples of other contexts, such as HTML Tag and All HTML. (You learn about these further when you get to creating your own templates.) The available contexts vary according to the plug-in. The point to remember is that templates are context-sensitive, meaning that you only get the templates relevant to your current location.

So, now that you have an idea of some of the benefits of templates and where they reside, let's take a closer look at a specific example.

Template anatomy

A template can be as simple as string, but, more often, they're a bit more complex. The first template in the list should be named **comment**. (You may have to expand the **Name** column to see the full names). Click **comment** in the data grid. You should see a preview of the comment template in the bottom of the window similar to `<!-- ${cursor} -->`.

This is a nice, simple template, but it demonstrates some of the key concepts this tutorial aims to get across. Basically, this template is simply a string (`<!-- -->`) with a single variable in it (`${cursor}`). Templates can contain varying combinations of string elements and variables, but these are the basic components that make up templates.

In this example, you see the basic syntax for template variables. They start with the dollar sign (`$`), and the variable name is contained within curly brackets (`{}`). Variables come in two basic varieties: Those defined by the template plug-in (that is, *predefined*—`cursor` is one of these), and those you create yourself. The sections that follow examine both of these types in more detail. But for now, just be aware that when you see something in a template that has `$` and `{}`, you're looking at a variable.

Working with existing templates

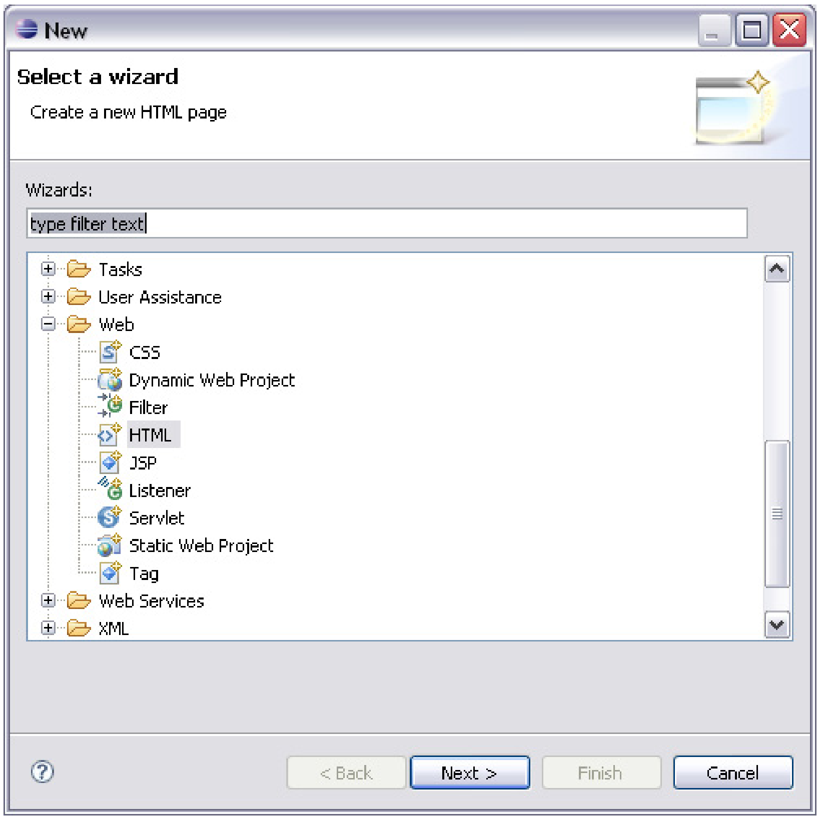
Create your file

To use a template, you must create a document of the type that your templates are associated with. So, because this tutorial deals with HTML code templates, create an HTML file. Close the Preferences pane by clicking **OK**. If you have not done so, create a project so you can create a file to work with. The name of the project doesn't really matter, but you can call it `templates`, if you like.

You will create a file to work with, but you want to do so in a specific way so that you can see the template context:

1. Click **File > Other**. The **New** window appears.

Figure 2. The New window



New

Select a wizard

Create a new HTML page

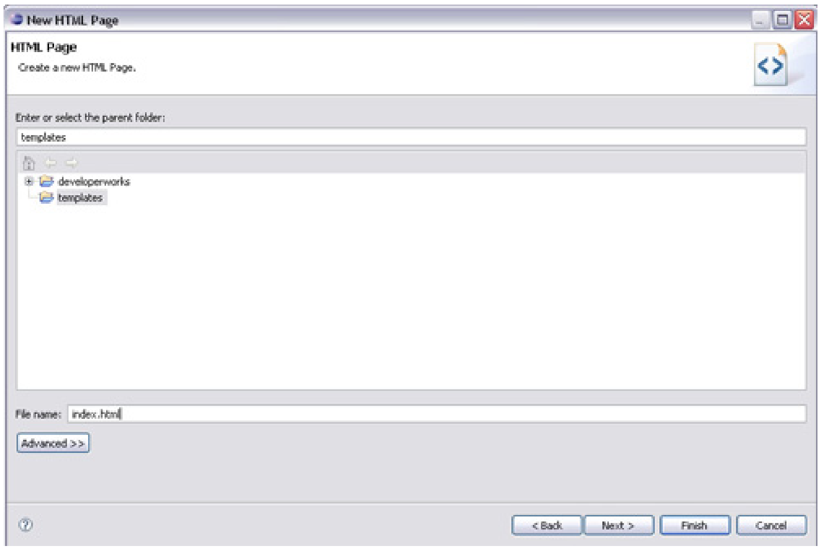
Wizards:

type filter text

- Tasks
- User Assistance
- Web
  - CSS
  - Dynamic Web Project
  - Filter
  - HTML
  - JSP
  - Listener
  - Servlet
  - Static Web Project
  - Tag
- Web Services
- XML

2. In the Wizards tree, expand Web.
3. Click **HTML** and click **Next**. The **New HTML Page** window appears.

Figure 3. The New HTML Page window



HTML Page

Create a new HTML Page.

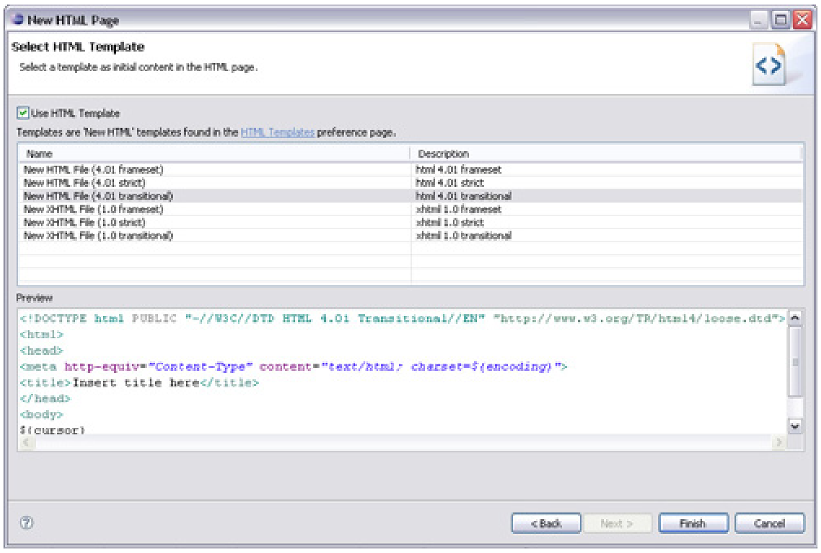
Enter or select the parent folder: templates

File name: index.html

4. Type a name for the file. You can use any name you like, as you're only going to be working with one file.
5. Click **Next**.

You should now be looking at a data grid with several templates in it.

Figure 4. Select an HTML template



Select HTML Template

Select a template as initial content in the HTML page.

[✓] Use HTML Template

Templates are 'New HTML' templates found in the HTML Templates preference page.

Name	Description
New HTML File (4.01 frameset)	html 4.01 frameset
New HTML File (4.01 strict)	html 4.01 strict
New HTML File (4.01 transitional)	html 4.01 transitional
New XHTML File (1.0 frameset)	xhtml 1.0 frameset
New XHTML File (1.0 strict)	xhtml 1.0 strict
New XHTML File (1.0 transitional)	xhtml 1.0 transitional

Preview

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=${encoding}">
<title>Insert title here</title>
</head>
<body>
${cursor}
```

This is an example of the context sensitivity of templates. Here, Eclipse knows you're creating a new document of type HTML. Because the HTML plug-in has a context called *New HTML*, the IDE presents those templates relevant to creating a new document. This would be the same for any other plug-in that defines templates for a new document. To use one of these templates, simply click it in the data grid and click **Finish**. (Other plug-ins may have additional steps, but as far as selecting the template goes, it is generally the same process.)

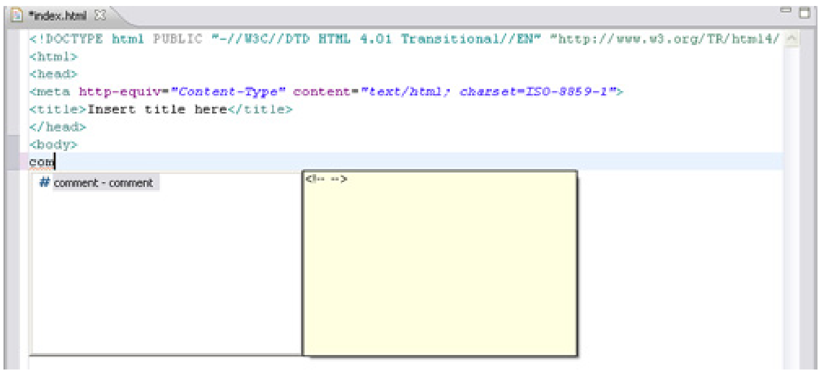
You should now be looking at a document that contains the code in the template you selected, which means that you're ready to see how templates work in the editor.

Use the editor

To work with your new page in the template:

1. Click in the editor so that the cursor is in between the HTML `<body>` tags.
2. Press **Ctrl+Space** (or **Cmd+Space** on the Mac) to trigger the auto-completion box.
3. Start typing the word `comment`. By the time you get to the first `m`, the word *comment* should be highlighted in the auto-completion box.

   Figure 5. The text in the auto-completion box

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
com
```

```
<!-- -->
```

4. Press **Enter** to insert the text.

You should now have an HTML comment in your page, and the cursor should be blinking between the starting and ending `<comment>` tags. So, what just happened? Because you're now within the editor window and within an HTML tag, you're in the HTML Tag context. Hence, the IDE presents those templates related to that context. Remember that the comment template looked like this: `<!-- ${cursor} -->` and that `${cursor}` is a predefined variable. This variable causes the cursor to be located between the comment tags when the template is inserted.

As far as using templates goes, that's basically it. Based on the context of where you are and what you're doing, the IDE presents you with appropriate templates. All you have to do is select the template you want.

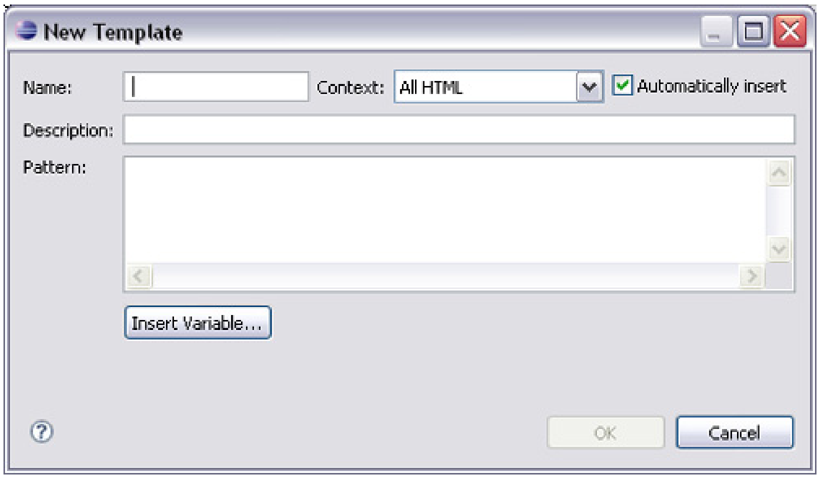
Creating custom templates

Custom templates

Predefined templates can be a great time-saver, but chances are, you have you own pieces of code you use on a regular basis, and none of the predefined templates quite fit. Fortunately, the developers of Eclipse templates thought of this and have given you the ability to create your own templates.

Open the **Preferences** pane and navigate to the HTML templates. When you arrive at the data grid with the templates list, click **New**. You should now be looking at a window that has input text fields for the template name, description, the template itself (pattern), and a drop-down menu for the context.

Figure 6. The New Template window



Before you create the template itself, you need a bit of information.

Set the context for your template

First, note the **Context** drop-down list, which contains the available contexts for your template. In the following examples, the word *cursor* in the italic font indicates where the cursor must be for the template to be available.

All HTML
These are available anywhere within an HTML document.

New HTML
These are the templates that are available when creating a new document.

HTML Tag
For these to show up, the cursor must be between an opening and closing HTML tag, such as `<div>cursor</div>`.

HTML Attribute
For these to be triggered, the cursor must be within an opening HTML tag, such as `<div cursor>`.

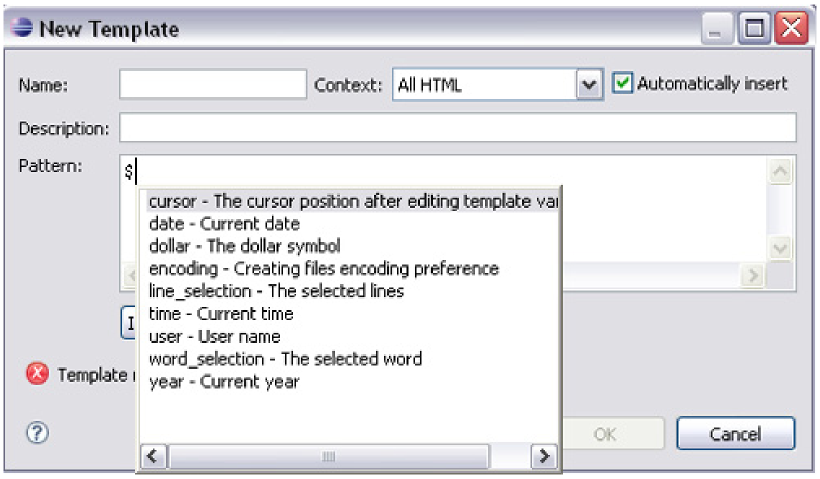
HTML Attribute Value
These are triggered when you're providing a value for an attribute such as `<div align="cursor">`.

Contexts provide a lot of control over where your templates show up and provide a pre-filtering mechanism so you don't have to search through all templates, but only those most relevant to what you're currently doing. The available contexts vary based on the plug-in, and you will become familiar with the contexts for the plug-ins you use as you try them. In general, the default context is the one with the broadest scope (the one available in the most places). The point to remember is that this is where you set the context for a template. If you create a template and it doesn't show up when you expect it to or it shows up where it shouldn't, make sure you selected the appropriate context.

Use the predefined variables

Next, you must be aware of the predefined variables. Click in the pattern box and type `$` (or you can click **Insert Variable**), and an auto-completion box appears.

Figure 7. Predefined variables



cursor - The cursor position after editing template va
date - Current date
dollar - The dollar symbol
encoding - Creating files encoding preference
line_selection - The selected lines
time - Current time
user - User name
word_selection - The selected word
year - Current year

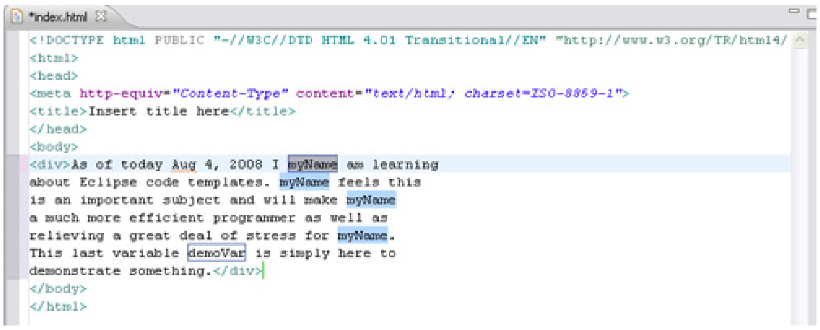
This box contains the predefined variables for the plug-in. Each has a description next to it, so I won't bother explaining what each one is. Just be aware that the names that you see are the predefined variables.

Create a custom template

This tutorial is more concerned with demonstrating the features of custom templates than in creating valid (or even useful) HTML code, so this example may seem a bit contrived and perhaps silly. But bear with me. To create a custom template:

1. In the **Name** box in the **New Template** window, type `ego_trip`.
2. Select **HTML Tag** from the **Context** list.
3. In the **Description** box, type `All about me in 3rd person`.
4. In the **Pattern** box, type:
```
<div>As of today ${date} I ${myName} am learning about Eclipse
    code templates. ${myName} feels this is an important subject and will make
    ${myName} a much more efficient programmer as well as relieving a great
    deal of stress for ${myName}. This last variable ${demoVar} is simply here
    to demonstrate something.</div>
```
5. Click **OK**, then click **OK** in the Preferences pane.
6. Go back to the document you created earlier and position the cursor between the HTML `<body>` tags.
7. Press **Ctrl+Shift** (**Cmd+Shift** on the Mac) to launch the auto-completion box.
8. Start typing the word `ego_trip`.
9. When the item is highlighted in the auto-completion box, press **Enter**. You should now see something like Figure 8.

Figure 8. Template output



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<div>As of today Aug 4, 2008 I myName am learning
about Eclipse code templates. myName feels this
is an important subject and will make myName
a much more efficient programmer as well as
relieving a great deal of stress for myName.
This last variable demoVar is simply here to
demonstrate something.</div>
</body>
</html>
```

Even though you've just inserted the template, several things have happened. The `${date}` variable has been replaced with the current date. All your variables are surrounded by boxes, but notice that all the instances of `$myName` are highlighted, while `${demoVar}` is not. Also notice the location of the cursor: It's blinking within the box that surrounds the first variable for which you must provide a value. It is not after the template, as it would be if you inserted something like a snippet or just copied and pasted some code.

Work in your custom template

Start typing your name. Notice that all the occurrences `myName` are being updated. This is one of — if not the greatest — features in Eclipse templates. Right now, if you were using a snippet, you would probably be doing a string replace on some placeholder you built into the template. This can be time-consuming even if the placeholder only occurs a few times within the snippet, as find-and-replace operations generally involve some sort of window and you have to address each occurrence. You could be brave and replace all, but many times, that operation ends up replacing things you didn't intend to replace. With Eclipse variables, it only takes as long as it takes you to type the value, which you would have to do anyway for a find-and-replace operation, and the entire template is updated only in the locations that contain the variable.

Now, press the **Tab** key. Notice that `${demoVar}` is highlighted. The **Tab** key allows you to move between variables within a template. If you were using a snippet and had two patterns to replace, you would have to perform the find-and-replace routine for each value you wanted to replace. Not only do you *not* have to do this with Eclipse templates but variables that have already been updated will be skipped, and you will be taken to the next variable in the template for which a value must be provided. In this example, all occurrences of the `${myName}` variable that had already been update were skipped, and you were taken directly to the `${demoVar}` variable.

Type any value you like for the `${demoVar}` variable and press **Enter**. Notice the location of the cursor: It has skipped to the end of the template. Pressing **Enter** key signals that you're done editing a template, and, as such, Eclipse assumes you want to go back to editing your document, with the most likely place from which to continue editing being after the template you just inserted. This is a convenience and a source of errors because if you accidentally press **Enter** while in the middle of editing the template, Eclipse keeps the default names of all the variables you provided or whatever you managed to type before pressing **Enter**. If you do this, you will have to go back and manually edit them or start over by reinserting the template.

You should really be starting to understand the power of Eclipse templates and how much thought went into their creation. Now that you can use the built-in templates as well as create and use your own templates, you're ready to learn how to manipulate (edit, rename, and delete) existing templates.

Manipulating templates

Edit templates

Chances are, you're going to create a few templates that are not exactly what you intended. Fortunately, editing templates is a simple process, and you can edit templates that you create, as well as predefined templates.

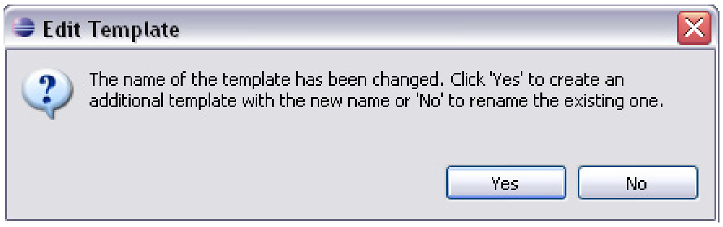
Navigate to the HTML templates in the **Preferences** pane. When you arrive at the data grid, click the ego_trip template and click **Edit**. You should be looking at the same window you saw when you created your template.

Editing the template is simply a matter of changing the values for any of the fields you desire and clicking **OK**. Because you have already created a template, I don't spend any more time explaining how to edit it, as the process is basically the same. However, there is a trick worth knowing about.

Renaming templates

In the **Name** field, change the name of the template from `ego_trip` to `ego_trip_2`, then click **OK**. You should see a window similar to Figure 9.

Figure 9. Confirming the rename command



The name of the template has been changed. Click 'Yes' to create an additional template with the new name or 'No' to rename the existing one.

If your intention is to rename the existing template, simply click **No** and you're done. If you click **Yes**, a copy of the existing template is saved with the new name. You can use this to quickly create new templates from existing ones. So, if you have a template that is, say 90 percent of what you need, you can open it, edit it, and save it with a new name, which is much faster than creating a new one each time. Remember that the entire idea behind templates is that you should not have to perform repetitive tasks. You should only be creating a new template from scratch if no template exists that you can customize.

So far, you learned how to create, use, and edit templates. This leaves only one possible task you can perform with them: deleting them.

Delete templates

Deleting a template is as simple as selecting it in the templates data grid in the Preferences pane and clicking **Remove**.

Tips, tricks, and gotchas

Templates are workspace-specific

Custom templates are workspace-specific. This means that templates you create in one workspace will not be available if you switch workspaces. This has benefits and drawbacks. The major drawback is obvious: You have to recreate templates for each workspace. However, many things are like this in Eclipse. For example, preferences are also workspace-specific. The benefit of this is that you can create templates that are very specific to the projects for that workspace.

Misfire auto-completion entries

If a template doesn't show up in the auto-completion box when you press **Ctrl+Shift** (**Cmd+Shift** on the Mac), press **Backspace**, then type the first few letters of the template name, then press **Ctrl+Shift**.

Custom cursor location

Remember the `${cursor}` variable? It can give you precise control over where the cursor is located when you're using templates. Essentially, the `${cursor}` variable marks the spot that the cursor will jump to when you press **Enter**, indicating you're done editing the template. I find this variable particularly useful when creating templates for things like functions, where I want the cursor to end up in the body of the function after I have customized the parameters.

Show me the money

If you happen to be working in a programming language in which the dollar sign (`$`) has special meaning (such as in PHP), you must escape that character in your templates. To get a literal `$`, use two of them in a row —`$$`. If you want a template variable to follow a dollar sign, you must use three of them —`$$${`*`variable`*`}`.

Summary

It should be clear that Eclipse templates are far more versatile than simple snippet functionality. Unlike snippets, Eclipse templates are directly integrated into the workflow in Eclipse — from creating a new document to working in the editor. The ability to use variables in templates makes customization a breeze, whereas with snippets, you basically get a shell you can insert and edit manually.

This tutorial was purposefully broad, and you will have to spend some time learning the templates for each plug-in you end up using. However, most of the differences you will find among plug-ins that support templates have to do with the number of predefined templates, available contexts, and the number of predefined variables. You are already familiar with all these concepts from this tutorial. I used a wide variety of Eclipse plug-ins (HTML, PHP, the Java™ language, Python, and more) that support templates, and the skills you have learned here are applicable to all of them.

When you become accustomed to using templates, you'll wonder how you ever lived without them. On the rare occasions I find myself working in an editor other than Eclipse, I find myself trying to trigger the templates. When I realize that they're not there, I breathe a sigh of frustration.

Related topics

- The Eclipse help has more information about what's new, as well as additional tips and tricks.
- The Eclipse Java Development User Guide -- Code templates discusses Java code templates.
- The Eclipse Java Development User Guide -- Using code templates discusses Java code template use.
- The Eclipse C++ Development User Guide -- Code templates discusses C++ code template use.
- Download the Eclipse IDE and install it from the Eclipse Foundation.
- Download the Web Tools Platform (WTP) Project plug-in to follow the lessons found in this tutorial.
- Check out the latest Eclipse technology downloads at IBM alphaWorks.
- Check out the "Recommended Eclipse reading list."
- Download Eclipse Platform and other projects from the Eclipse Foundation.
- Browse all the Eclipse content on developerWorks.
- New to Eclipse? Read the developerWorks article "Get started with Eclipse Platform" to learn its origin and architecture, and how to extend Eclipse with plug-ins.
- Expand your Eclipse skills by checking out IBM developerWorks' Eclipse project resources.