

# Displaying XML Data in Web Pages

You can use Macromedia Dreamweaver 8 to create web pages that display XML data. Displaying XML data involves retrieving information stored in a local or remote XML file and rendering that information in a web page. Dreamweaver provides methods for displaying information from XML files, as well as built-in XSLT objects and design features that let you enhance the presentation of your XML data.

This chapter contains the following sections:

About using XML and XSL with web pages . . . . .	735
About server-side XSL transformations . . . . .	737
About client-side XSL transformations . . . . .	740
About XML data and repeating elements . . . . .	743
About previewing XML data . . . . .	744
Performing XSL transformations on the server . . . . .	746
Performing XSL transformations on the client . . . . .	760
Applying styles to XSLT fragments . . . . .	763
Troubleshooting XSL transformations . . . . .	764

## About using XML and XSL with web pages

Extensible Markup Language (XML) is a language that lets you structure information. Like HTML, XML lets you structure your information using tags, but XML tags are not predefined as HTML tags are. Instead, XML lets you create tags that best define your data structure. Tags are nested within others to create a schema of parent and child tags. Like most HTML tags, all tags in an XML schema have an opening and closing tag.

The following example illustrates the basic structure of an XML file:

```
<?xml version="1.0">
<mybooks>
  <book bookid="1">
    <pubdate>03/01/2004</pubdate>
    <title>Displaying XML Data with Macromedia Dreamweaver</title>
    <author>Charles Brown</author>
  </book>
  <book bookid="2">
    <pubdate>04/08/2004</pubdate>
    <title>Understanding XML</title>
    <author>John Thompson</author>
  </book>
</mybooks>
```

In this example, each parent `<book>` tag contains three child tags: `<pubdate>`, `<title>`, and `<author>`. But each `<book>` tag is also a child tag of the `<mybooks>` tag, which is one level higher in the schema. You can name and structure XML tags in any way you like, provided that you nest tags accordingly within others, and assign each opening tag a corresponding closing tag.

XML documents do not contain any formatting — they are simply containers of structured information. Once you have an XML schema, you can use the Extensible Stylesheet Language (XSL) to display the information. In the way that Cascading Style Sheets (CSS) let you format HTML, XSL lets you format XML data. You can define styles, page elements, layout, and so forth in an XSL file and attach it to an XML file so that when a user views the XML data in a browser, the data is formatted according to whatever you’ve defined in the XSL file. The content (the XML data) and presentation (defined by the XSL file) are entirely separate, providing you with greater control over how your information appears on a web page. In essence, XSL is a presentation technology for XML, where the primary output is an HTML page.

Extensible Stylesheet Language Transformations (XSLT) is a subset language of XSL that actually lets you display XML data on a web page, and “transform” it, along with XSL styles, into readable, styled information in the form of HTML. You can use Dreamweaver to create XSLT pages that let you perform XSL transformations using an application server or a browser. When you perform a server-side XSL transformation, the server does the work of transforming the XML and XSL, and displaying it on the page. When you perform a client-side transformation, a browser (such as Internet Explorer) does the work.

The approach you ultimately take (server-side transformations versus client-side transformations) depends on what you are trying to achieve as an end result, the technologies available to you, the level of access you have to XML source files, and other factors. Both approaches have their own benefits and limitations. For example server-side transformations work in all browsers while client-side transformations are restricted to modern browsers only (Internet Explorer 6, Netscape 8, Mozilla 1.8, and Firefox 1.0.2). Server-side transformations let you display XML data dynamically from your own server or from anywhere else on the web, while client-side transformations must use XML data that is locally hosted on your own web server. Lastly, server-side transformations require that you deploy your pages to a configured application server, while client-side transformations only require access to a web server.

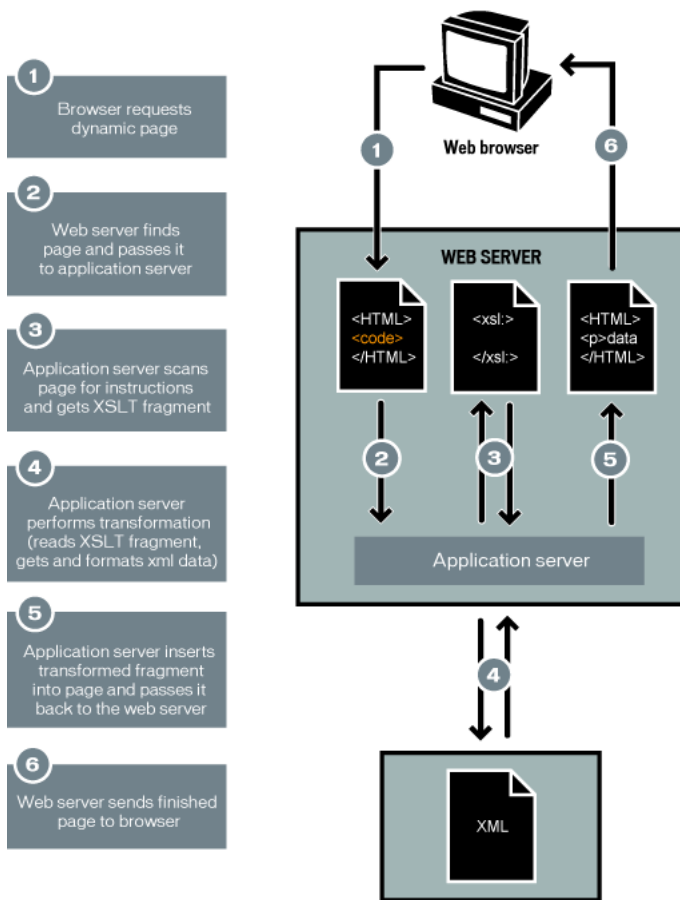
For more information, see [“About server-side XSL transformations” on page 737](#), and [“About client-side XSL transformations” on page 740](#).

## About server-side XSL transformations

Dreamweaver provides methods for creating XSLT pages that let you perform server-side XSL transformations. When an application server performs the XSL transformation, the file containing the XML data can reside on your own server, or anywhere else on the web. Additionally, any browser can display the transformed data. Deploying pages for server-side transformations, however, is somewhat complex, and requires that you have access to an application server.

When working with server-side XSL transformations, you can use Dreamweaver to create XSLT pages that generate full HTML documents (entire XSLT pages), or XSLT fragments that generate a portion of an HTML document. An entire XSLT page is similar to a regular HTML page. It contains a `<body>` tag and a `<head>` tag, and lets you display a combination of HTML and XML data on the page. An XSLT fragment is a piece of code, used by a separate document, that displays formatted XML data. Unlike an entire XSLT page, it is an independent file that contains no `<body>` or `<head>` tag. If you want to display XML data on a page of its own, you would create an entire XSLT page, and bind your XML data to it. If, on the other hand, you wanted to display XML data in a particular section of an existing dynamic page—for example, a dynamic home page for a sporting goods store, with sports scores from an RSS feed displayed on one side of the page—you would create an XSLT fragment and insert a reference to it in the dynamic page. Creating XSLT fragments, and using them in conjunction with other dynamic pages to display XML data, is the more common scenario.

The first step in creating these types of pages is to create the XSLT fragment: It is a separate file that contains the layout, formatting, and so on of the XML data that you eventually want to display in the dynamic page. Once you create the XSLT fragment, you insert a reference to it in your dynamic page (for example, a PHP or Macromedia ColdFusion page). The inserted reference to the fragment works much like an Server Side Include (SSI) — the formatted XML data (the fragment) resides in a separate file, while in Design view, a placeholder for the fragment appears on the dynamic page itself. When a browser requests the dynamic page containing the reference to the fragment, the server processes the included instruction and creates a new document in which the formatted contents of the fragment appear instead of the placeholder.



You use the XSL Transformation server behavior to insert the reference to an XSLT fragment in a dynamic page. When you insert the reference, Dreamweaver generates an `includes/MM_XSLTransform/` folder in the site's root folder that contains a runtime library file. The application server uses the functions defined in this file when transforming the specified XML data. The file is responsible for fetching the XML data and XSLT fragments, performing the XSL transformation, and outputting the results on the web page.

The file containing the XSLT fragment, the XML file containing your data, and the generated run-time library file must all be on the server for your page to display correctly. (If you select a remote XML file as your data source — one from an RSS feed, for example — that file must of course reside somewhere else on the Internet.)

You can also use Dreamweaver to create entire XSLT pages for use with server-side transformations. An entire XSLT page works in exactly the same way as an XSLT fragment, only when you insert the reference to the entire XSLT page using the XSL Transformation server behavior, you are inserting the full contents of an HTML page. Thus, the dynamic page (the `.cfm`, `.php`, `.asp`, or `.net` page that acts as the container page) must be cleared of all HTML before you insert the reference.

Dreamweaver supports XSL transformations for ColdFusion, ASP, ASP.NET, and PHP pages.

NOTE

Your server must be correctly configured to perform server-side transformations. For more information, contact your server administrator, or visit [www.macromedia.com/go/dw\\_xsl](http://www.macromedia.com/go/dw_xsl).

For procedures on creating server-side XSL transformations, see “[Performing XSL transformations on the server](#)” on page 746.

### Related topics

- “[About previewing XML data](#)” on page 744
- “[Applying styles to XSLT fragments](#)” on page 763

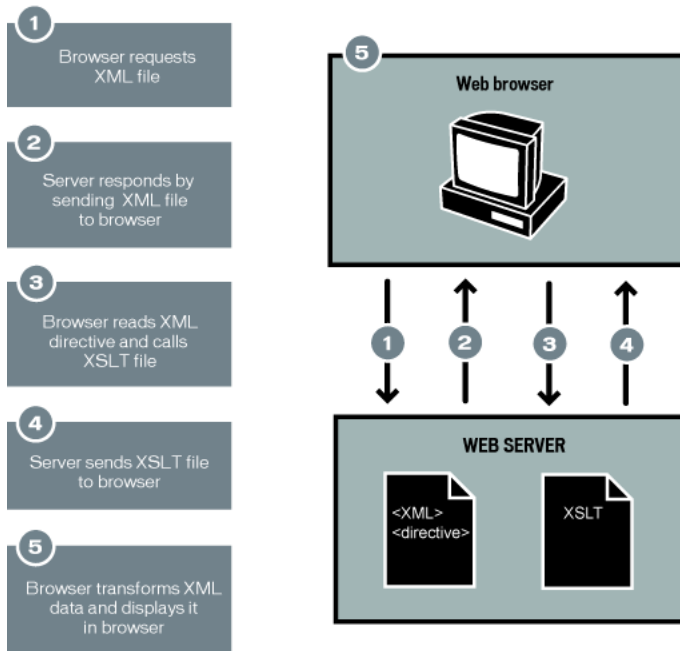
# About client-side XSL transformations

You can also perform XSL transformations on the client without the use of an application server. You can use Dreamweaver to create an entire XSLT page that will do this; however, client-side transformations require manipulation of the XML file that contains the data you want to display. Additionally, client-side transformations will only work in modern browsers (Internet Explorer 6, Netscape 8, Mozilla 1.8, and Firefox 1.0.2). For more information on browsers that do and don't support XSL transformations, see [www.w3schools.com/xsl/xsl\\_browsers.asp](http://www.w3schools.com/xsl/xsl_browsers.asp).

You begin by creating an entire XSLT page and attaching an XML data source. (Dreamweaver prompts you to attach the data source when you create the new page.) You can use Dreamweaver to create an XSLT page from scratch, or you can convert an existing HTML page to an XSLT page. When you convert an existing HTML page to an XSLT page you must attach an XML data source using the Bindings panel (Window > Bindings).

Once you've created your XSLT page, you must link it to the XML file containing the XML data by inserting a reference to the XSLT page in the XML file itself (much like you would insert a reference to an external CSS style sheet in the <head> section of an HTML page). Your site visitors must view the XML file (not the XSLT page) in a browser. When your site visitors view the page, the browser performs the XSL transformation and displays the XML data, formatted by the linked XSLT page.

The relationship between the linked XSLT and XML pages is conceptually similar, yet different from the external CSS/HTML page model. When you have an HTML page that contains content (such as text), you use an external style sheet to format that content. The HTML page determines the content, and the external CSS code, which the user never sees, determines the presentation. With XSLT and XML, the situation is reversed. The XML file (which the user never sees in its raw form), determines the content while the XSLT page determines the presentation. The XSLT page contains the tables, layout, graphics, and so forth that the standard HTML usually contains. When a user views the XML file in a browser, the XSLT page formats the content.



When you use Dreamweaver to link an XSLT page to an XML page, Dreamweaver inserts the appropriate code for you at the top of the XML page. If you own the XML page to which you're linking (that is, if the XML file exclusively lives on your web server), all you need to do is use Dreamweaver to insert the appropriate code that links the two pages. When you own the XML file, the XSL transformations performed by the client are fully dynamic. That is, whenever you update the data in the XML file, any HTML output using the linked XSLT page will be automatically updated with the new information.

**NOTE**

The XML and XSL files you use for client-side transformations must reside in the same directory. If they don't, the browser will read the XML file and find the XSLT page for the transformation, but will fail to find assets (style sheets, images, and so on) defined by relative links in the XSLT page.

If you don't own the XML page to which you're linking (for example, if you want to use XML data from an RSS feed somewhere out on the web), the workflow is a bit more complicated. To perform client-side transformations using XML data from an external source, you must first download the XML source file to the same directory where your XSLT page resides. Once the XML page is in your local site, you can use Dreamweaver to add the appropriate code that links it to the XSLT page, and post both pages (the downloaded XML file and the linked XSLT page) to your web server. When the user views the XML page in a browser, the XSLT page formats the content, just like in the previous example.

The disadvantage to performing client-side XSL transformations on XML data that comes from an external source is that the XML data is only partially "dynamic." The XML file that you download and alter is merely a "snapshot" of the file that lives elsewhere on the web. If the original XML file out on the web changes, you must download the file again, link it to the XSLT page, and repost the XML file to your web server. The browser only renders the data that it receives from the XML file on your web server, not the data contained in the original XML source file.

For procedures on creating client-side XSL transformations, see ["Performing XSL transformations on the client" on page 760](#).

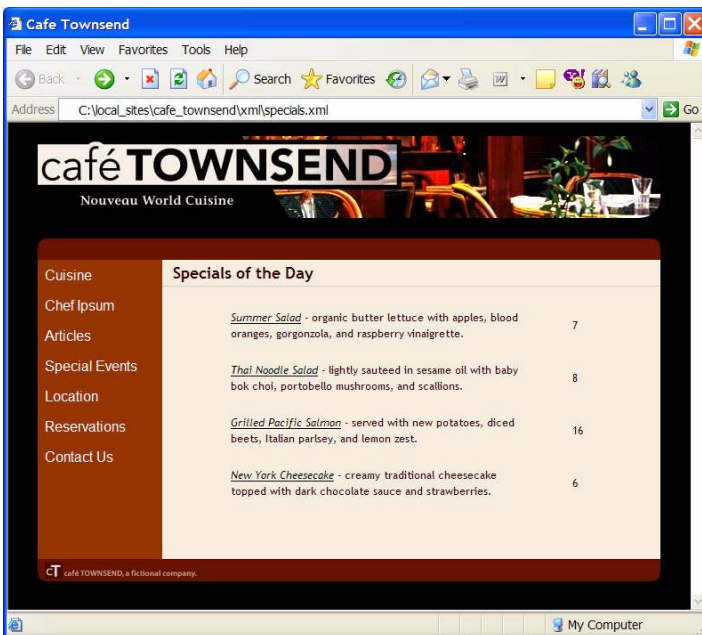
### Related topics

- ["About previewing XML data" on page 744](#)
- ["Applying styles to XSLT fragments" on page 763](#)

# About XML data and repeating elements

The Repeat Region XSLT object lets you display repeating elements from an XML file within a page. Any region containing an XML data placeholder can be turned into a repeated region. However, the most common regions are a table, a table row, or a series of table rows.

The following example illustrates how the Repeat Region XSLT object is applied to a table row that displays menu information for a restaurant. The initial row displays three different elements from the XML schema: item, description, and price. When the Repeat Region XSLT object is applied to the table row, and the page is processed by an application server or a browser, the table is repeated with unique data inserted in each new table row.



When you apply a Repeat Region XSLT object in the Document window, a thin, tabbed, gray outline appears around the repeated region. When you preview your work in a browser (File > Preview in Browser), the gray outline disappears and the selection expands to display the specified repeating elements in the XML file, as in the previous illustration.

You might also notice that when you add the Repeat Region XSLT object to the page, Dreamweaver truncates the length of the XML data placeholder in the Document window. This is because Dreamweaver updates the XPath for the XML data placeholder so that it is relative to the path of the repeating element.

For example, the following code is for a table that contains two dynamic placeholders, without a Repeat Region XSLT object applied to the table:

```
<table width="500" border="1">
  <tr>
    <td><xsl:value-of select="rss/channel/item/title"/></td>
  </tr>
  <tr>
    <td><xsl:value-of select="rss/channel/item/description"/></td>
  </tr>
</table>
```

The following code is for the same table with the Repeat Region XSLT object applied to it:

```
<xsl:for-each select="rss/channel/item">
  <table width="500" border="1">
    <tr>
      <td><xsl:value-of select="title"/></td>
    </tr>
    <tr>
      <td><xsl:value-of select="description"/></td>
    </tr>
  </table>
</xsl:for-each>
```

In the previous example, Dreamweaver has automatically updated the XPath for the items that fall within the Repeat Region (title & description) to be relative to the XPath in the enclosing `<xsl:for-each>` tags, rather than the full document.

Dreamweaver generates context-relative XPath expressions in other cases as well. For example, if you drag an XML data placeholder to a table that already has a Repeat Region XSLT object applied to it, Dreamweaver automatically displays the XPath relative to the existing XPath in the enclosing `<xsl:for-each>` tags.

To learn how to apply the Repeat Region XSLT object, see [“Displaying repeating XML elements” on page 753](#).

## About previewing XML data

When you use Preview in Browser (File > Preview in Browser) to preview XML data that you’ve inserted in an XSLT fragment or an entire XSLT page, the engine that performs the XSL transformation differs from situation to situation. For dynamic pages containing XSLT fragments, the application server always performs the transformation. At other times, either Dreamweaver or the browser might be performing the transformation.

The following table summarizes the situations when using Preview in Browser, and the engines that perform the respective transformations:

Type of page previewed in browser	Data transformation performed by
Dynamic page containing XSLT fragment	Application server
XSLT fragment or entire XSLT page	Dreamweaver
XML file with link to entire XSLT page	Browser

The following topics provide guidelines for helping you determine the appropriate previewing methods, based on your needs:

- [“Previewing pages for server-side transformations” on page 745](#)
- [“Previewing pages for client-side transformations” on page 745](#)
- [“Previewing entire XSLT pages and XSLT fragments” on page 746](#)

## Previewing pages for server-side transformations

In the case of server-side transformations, the content the site visitor ultimately sees is transformed by your application server. When building XSLT and dynamic pages for use with server-side transformations, it is always preferable to preview the dynamic page that contains the XSLT fragment instead of the XSLT fragment itself. In the former scenario, you make use of the application server, which ensures that your preview is consistent with what your site visitors will see when they visit your page. In the latter scenario, Dreamweaver performs the transformation, and could provide slightly inconsistent results. You can use Dreamweaver to preview your XSLT fragment while you are building it, but you’ll be able to see the most accurate results of the data rendering if you use the application server to preview your dynamic page after you’ve inserted the XSLT fragment.

## Previewing pages for client-side transformations

In the case of client-side transformations, the content the site visitor ultimately sees is transformed by a browser. You accomplish this by adding a link from the XML file to the XSLT page. If you open the XML file in Dreamweaver and preview it in a browser, you force the browser to load the XML file and perform the transformation. This provides you with the same experience as that of your site visitor.

One disadvantage of this approach is that it makes it harder for you to debug your page because the browser transforms the XML and generates the HTML internally. If you select the browser's View Source option to debug the generated HTML, you will only see the original XML that the browser received, not the full HTML (tags, styles, and so forth) responsible for the rendering of the page. To see the full HTML when viewing source code, you must preview the XSLT page in a browser instead.

## Previewing entire XSLT pages and XSLT fragments

When creating entire XSLT pages and XSLT fragments, you'll want to preview your work to make sure that your data is being displayed correctly. If you use Preview in Browser to display an entire XSLT page or an XSLT fragment, Dreamweaver performs the transformation using a built-in transformation engine. This method gives you quick results, and makes it easier for you to incrementally build and debug your page. It also provides a way for you to view the full HTML (tags, styles, and so forth) by selecting the View Source option in the browser.

NOTE

You will most likely use this method when you begin building XSLT pages, regardless of whether you use the client or the server to transform your data.

### Related topics

- [“About using XML and XSL with web pages” on page 735](#)
- [“About server-side XSL transformations” on page 737](#)
- [“About client-side XSL transformations” on page 740](#)
- [“Applying styles to XSLT fragments” on page 763](#)

## Performing XSL transformations on the server

You can use Dreamweaver to create entire XSLT pages or XSLT fragments for use in dynamic web pages. An entire XSLT page is a page that, when transformed, generates a full HTML page. An XSLT fragment is a piece of code, used by a separate document, that, when transformed, displays XML data.

Macromedia recommends that you read [“About server-side XSL transformations” on page 737](#) before proceeding with any of the following procedures.

NOTE

Your server must be correctly configured to perform server-side transformations. For more information, contact your server administrator, or visit [www.macromedia.com/go/dw\\_xsl](http://www.macromedia.com/go/dw_xsl).

This section contains the following topics:

- “Workflow for performing server-side XSL transformations” on page 747
- “Creating XSLT pages” on page 748
- “Converting HTML pages to XSLT pages” on page 750
- “Attaching XML data sources” on page 750
- “Displaying XML data in XSLT pages” on page 751
- “Displaying repeating XML elements” on page 753
- “Editing a Repeat Region XSLT object” on page 755
- “Inserting XSLT fragments in dynamic pages” on page 755
- “Deleting XSLT fragments from dynamic pages” on page 757
- “Editing XSL Transformation server behaviors” on page 757
- “Using parameters with XSL transformations” on page 757
- “Creating conditional XSLT regions” on page 759
- “Editing a Conditional Region XSLT object” on page 759
- “Inserting XSL comments” on page 760

## Workflow for performing server-side XSL transformations

This section provides a list of steps you need to follow to perform server-side XSL transformations, and refers you to the sections in the documentation that elaborate on each procedure.

Macromedia recommends that you read “[About using XML and XSL with web pages](#)” on page 735, “[About server-side XSL transformations](#)” on page 737, and “[About client-side XSL transformations](#)” on page 740 before building pages that display XML data.

To perform server-side XSL transformations, follow these steps:

- Set up a Dreamweaver site. See [Chapter 2, “Setting Up a Dreamweaver Site,”](#) on page 79.
- Choose a server technology and set up an application server. See “[Setting up an application server](#)” on page 601.
- Test the application server to make sure it is functioning correctly. For example, create a page that requires processing, and make sure that the application server processes the page. For a tutorial on how to do this, visit [www.macromedia.com/go/dw\\_xsl](http://www.macromedia.com/go/dw_xsl).

- Do one of the following:
  - In your Dreamweaver site, create an XSLT fragment or an entire XSLT page. See [“Creating XSLT pages” on page 748](#).
  - Convert an existing HTML page to an entire XSLT page. See [“Converting HTML pages to XSLT pages” on page 750](#).
- If you haven’t already done so, attach an XML data source to the page. See [“Attaching XML data sources” on page 750](#).
- Bind your XML data to the XSLT fragment or to the entire XSLT page. See [“Displaying XML data in XSLT pages” on page 751](#).
- If appropriate, add a Repeat Region XSLT object to the table or table row that contains the XML data placeholder(s). See [“Displaying repeating XML elements” on page 753](#).
- Do one of the following:
  - Use the XSL Transformation server behavior to insert a reference to the XSLT fragment in your dynamic page. See [“Inserting XSLT fragments in dynamic pages” on page 755](#).
  - Delete all of the HTML code from a dynamic page, and then use the XSL Transformation server behavior to insert a reference to the entire XSLT page in the dynamic page.
- Post both the dynamic page and the XSLT fragment (or entire XSLT page) to your application server. If you are using a local XML file, you will need to post that as well.
- View the dynamic page in a browser. When you do so, the application server transforms the XML data, inserts it in the dynamic page, and displays it in the browser.

## Creating XSLT pages

You can create XSLT pages that let you display XML data on web pages. You can create either an entire XSLT page — an XSLT page that contains a `<body>` tag and a `<head>` tag — or you can create an XSLT fragment. When you create an XSLT fragment, you create an independent file that contains no body or head tag — a simple piece of code that is later inserted in a dynamic page.

**NOTE**

If you are starting with an existing XSLT page, and need to attach an XML data source to it, see [“Attaching XML data sources” on page 750](#).

### To create an XSLT page:

1. Select File > New
2. On the General tab of the New Document dialog box, select Basic page from the Category column and do one of the following:
  - Select XSLT (Entire page) from the Basics page column to create an entire XSLT page.
  - Select XSLT (Fragment) from the Basics page column to create an XSLT fragment.
3. Click Create.

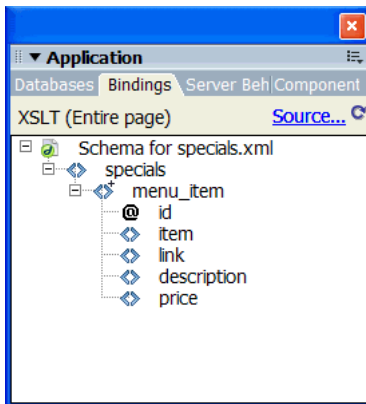
The Locate XML Source dialog box appears, asking you to attach an XML data source.

4. Do one of the following:
  - Select Attach a local file, click the Browse button, browse to a local XML file on your computer, and click OK.
  - Select Attach a remote file, enter the URL of an XML file on the Internet (such as one coming from an RSS feed), and click OK.

NOTE

Clicking the Cancel button will generate a new XSLT page with no attached XML data source. For information on attaching XML data sources, see [“Attaching XML data sources” on page 750](#).

Dreamweaver populates the Bindings panel with the schema of your XML data source.



The following table provides an explanation of the various elements in the schema that might appear:

Element	Represents	Details
◇	Required nonrepeating XML element	An element that appears exactly once within its parent node
◇+	Repeating XML element	An element that appears one or more times within its parent node
◇?	Optional XML element	An element that appears zero or more times within its parent node
Element node in boldface type	Current context element	Normally the repeating element when the insertion point is inside a repeat region
@	XML attribute	

5. Save your new page (File > Save) with the .xsl or .xslt extension (.xsl is the default).

## Converting HTML pages to XSLT pages

You can also convert existing HTML pages to XSLT pages. For example, if you have a predesigned static page to which you want to add XML data, you can convert the page to an XSLT page, instead of creating an XSLT page and redesigning the page from scratch.

### To convert an existing HTML page to an XSLT page:

1. Open the HTML page that you want to convert.
2. Select File > Convert > XSLT 1.0.

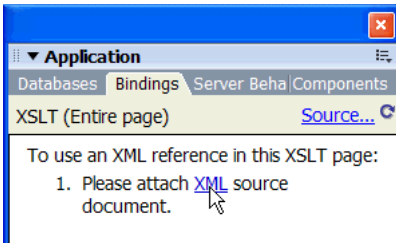
Dreamweaver opens a copy of the page in the Document window. The new page is an XSL style sheet, saved with the .xsl extension.

## Attaching XML data sources

If you are starting with an existing XSLT page, or if you don't attach an XML data source when creating a new XSLT page with Dreamweaver, you will need to attach an XML data source using the Bindings panel.

### To attach an XML data source:

1. In the Bindings panel (Window > Bindings), click the XML link.



NOTE

You can also click the Source link at the upper-right corner of the Bindings panel to add an XML data source.

2. Do one of the following:
  - Select Attach a Local File, click the Browse button, browse to a local XML file on your computer, and click OK.
  - Select Attach a Remote File, enter the URL of an XML file on the Internet (such as one coming from an RSS feed).
3. Click OK to close the Locate XML Source dialog box.

Dreamweaver populates the Bindings panel with the schema of your XML data source. For a guide to the symbols in the schema, see [“Creating XSLT pages” on page 748](#).

## Displaying XML data in XSLT pages

Once you’ve created an XSLT page and attached an XML data source, you can bind data to the page.

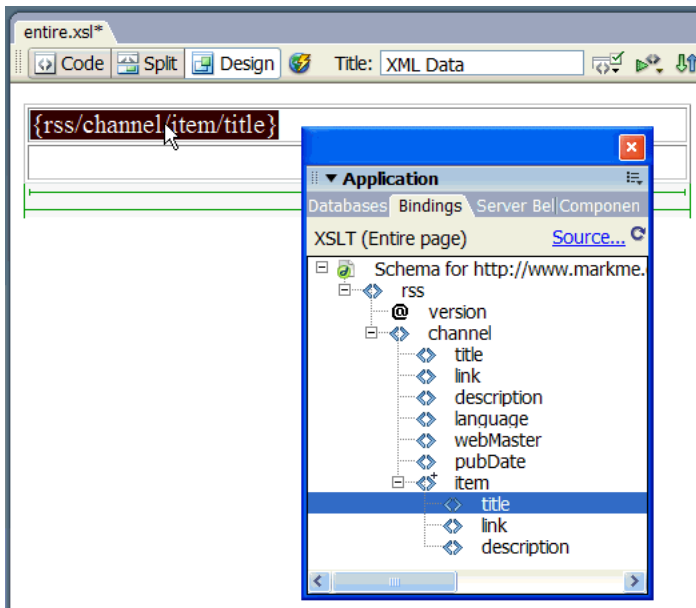
### To display XML data:

1. Open an XSLT page with an attached XML data source. For instructions, see [“Creating XSLT pages” on page 748](#).
2. (Optional) Select Insert > Table to add a table to the page. A table helps you organize your XML data. For more information, see [Chapter 8, “Presenting Content with Tables,” on page 233](#).

NOTE

In most cases, you will want to use the Repeat Region XSLT object to display repeating XML elements on a page. If this is the case, you might want to create a single-row table with one or more columns, or a two-rowed table if you want to include a table header. For more information, see [“Displaying repeating XML elements” on page 753](#).

3. In the Bindings panel, select an XML element and drag it to the place on the page where you want to insert data.



An XML data placeholder appears on the page. The placeholder is highlighted and in curly brackets. It uses the XPath (XML Path language) syntax to describe the hierarchical structure of the XML schema. For example, if you drag the child element “title” to the page, and that element has the parent elements “rss,” “channel,” and “item,” then the syntax for the dynamic content placeholder will be `{rss/channel/item/title}`.

Once an XML data placeholder is on the page, you can double-click it to open the XPath Expression Builder. The XPath Expression builder lets you format selected data, or select other items from the XML schema. For more information, click the Help button in the XPath Expression builder.

4. (Optional) Apply styles to your XML data by selecting an XML data placeholder and applying styles to it like any other piece of content using the Property inspector or the CSS Styles panel. Alternatively, you can use Design-time style sheets to apply styles to XSLT fragments. Each of these methods has its own set of benefits and limitations. For more information, see “Applying styles to XSLT fragments” on page 763.

5. Preview your work in a browser (File > Preview in Browser)

NOTE

When you preview your work using Preview in Browser, Dreamweaver performs an internal XSL transformation without the use of an application server. For more information, see [“About previewing XML data” on page 744](#).

## Displaying repeating XML elements

The Repeat Region XSLT object lets you display repeating elements from an XML data source in a web page. For example, if you are displaying article titles and descriptions from a news feed, and that news feed contains between 10 and 20 articles, each title and description in the XML file would probably be a child element of a repeating element.

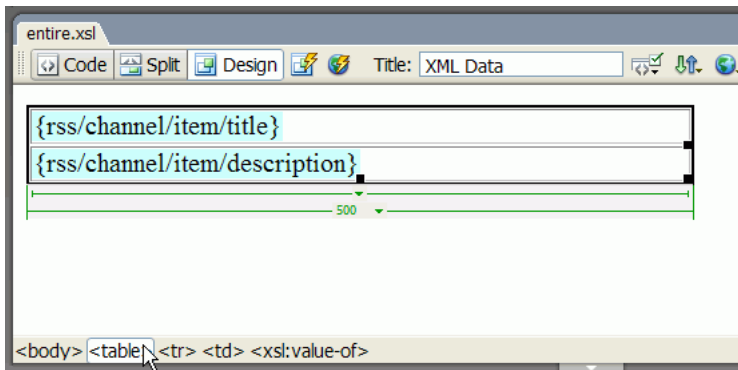
Any region in Design view containing an XML data placeholder can be turned into a repeated region. However, the most common regions are tables, table rows, or a series of table rows.

To learn more about how the Repeat Region XSLT object works with XML data, see [“About XML data and repeating elements” on page 743](#).

### To display repeating XML elements:

1. In Design view, select a region that contains an XML data placeholder or placeholders.

The selection can be anything, including a table, a table row, or even a paragraph of text.

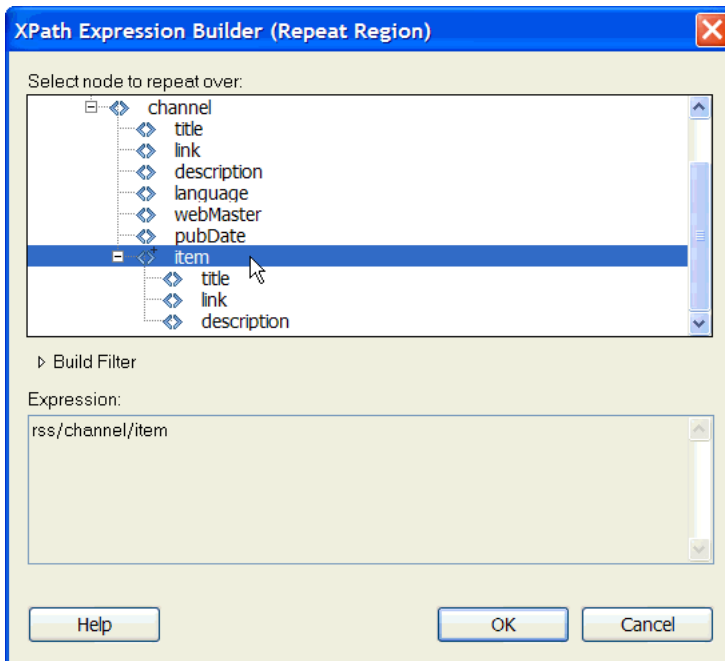


TIP

To select a region on the page precisely, you can use the tag selector in the lower-left corner of the Document window. For example, if the region is a table, click inside the table on the page, and then click the `<table>` tag in the tag selector.

2. Do one of the following
  - Select Insert > XSLT Objects > Repeat Region.
  - In the XSLT category of the Insert bar, click the Repeat Region button.

3. In the XPath Expression Builder, select the repeating element, indicated by a small plus sign.



For more information, click the Help button in the dialog box.

4. Click OK.

In the Document window, a thin, tabbed, gray outline appears around the repeated region. When you preview your work in a browser (File > Preview in Browser), the gray outline disappears and the selection expands to display the specified repeating elements in the XML file.

You'll also notice that when you add the Repeat Region XSLT object to the page, Dreamweaver truncates the length of the XML data placeholder in the Document window. This is because Dreamweaver updates the XPath for the XML data placeholder so that it is relative to the path of the repeating element.

For more information, see [“About XML data and repeating elements” on page 743](#).

## Editing a Repeat Region XSLT object

After you've added a Repeat Region XSLT object to a region, you can make changes to it using the Property inspector.

### To edit a Repeat Region XSLT object:

1. Select the object by clicking the gray tab that surrounds the repeated region.
2. In the Property inspector (Window > Properties), click the dynamic icon next to the Select text field.
3. In the XPath Expression Builder, make your changes and click OK.

## Inserting XSLT fragments in dynamic pages

Once you have created an XSLT fragment, you can insert it in a dynamic web page using the XSL Transformation server behavior. When you add the server behavior to your page and view the page in a browser, an application server performs a transformation that displays the XML data from the selected XSLT fragment. Dreamweaver supports XSL transformations for ColdFusion, ASP, ASP.NET, or PHP pages.

NOTE

If you want to insert the contents of an entire XSLT page in a dynamic page, the procedure is exactly the same. Before using the XSL Transformation server behavior to insert the entire XSLT page, delete all HTML code from the dynamic page. For more information, see [“About server-side XSL transformations” on page 737](#).

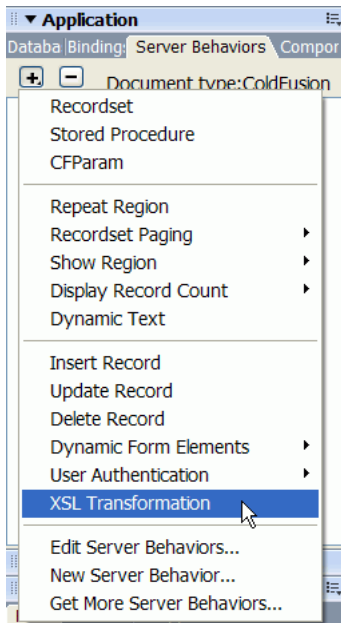
### To insert an XSLT fragment in a web page:

1. Open an existing ColdFusion, ASP, ASP.NET, or PHP page.
2. In Design view, place the insertion point in the location where you want to insert the XSLT fragment.

NOTE

When inserting XSLT fragments, you should always click the Show Code and Design view button after placing the insertion point on the page so that you can ensure that the insertion point is in the correct location. If it isn't, you might need to click somewhere else in Code view to place the insertion point where you want it.

3. In the Server Behaviors panel (Window > Server Behaviors), click the Plus (+) button and select XSL Transformation.



4. In the XSL Transformation dialog box, click the Browse button and browse to an XSLT fragment or an entire XSLT page. For more information, see [“Creating XSLT pages” on page 748](#).

Dreamweaver automatically populates the next text field with the file path or URL of the XML file that is attached to the specified fragment. To change it, click the Browse button and browse to another file.

5. (Optional) Click the Plus (+) button to add an XSLT parameter. For more information, see [“Using parameters with XSL transformations” on page 757](#).
6. Click OK.

Dreamweaver inserts a reference to the XSLT fragment in the page. The fragment is not editable. You can double-click the fragment to open the fragment’s source file and edit it. Dreamweaver also creates an `includes/MM_XSLTransform/` folder in the site’s root folder that contains a runtime library file. The application server uses the functions defined in this file to perform the transformation. For more information, see [“About server-side XSL transformations” on page 737](#).

7. Upload the dynamic page to your server (Site > Put). When Dreamweaver gives you the option of including dependent files, click Yes. The file containing the XSLT fragment, the XML file containing your data, and the generated run-time library file must all be on the server for your page to display correctly. (If you selected a remote XML file as your data source, that file must of course reside somewhere else on the Internet.)

## Deleting XSLT fragments from dynamic pages

You can remove an XSLT fragment from a page by deleting the XSL Transformation server behavior used to insert the fragment. Deleting the server behavior deletes the XSLT fragment only — it does not delete the associated XML, XSLT, or run-time library files.

### To delete an XSLT fragment from a dynamic page:

1. In the Server Behaviors panel (Window > Server Behaviors), select the XSL Transformation server behavior that you want to delete.
2. Click the minus (-) button.

**NOTE**

Macromedia recommends that you always remove server behaviors in this fashion. Manually deleting the generated code only partially removes the server behavior, even though the server behavior may disappear from the Server Behaviors panel.

## Editing XSL Transformation server behaviors

Once you've added an XSLT fragment to a dynamic web page, you can edit the XSL Transformation server behavior at any time.

### To edit an XSL Transformation server behavior:

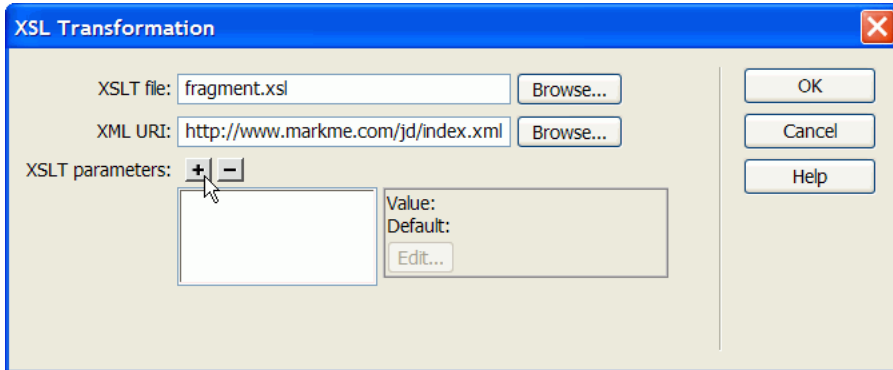
1. In the Server Behaviors panel (Window > Server Behaviors), double-click the XSL Transformation server behavior that you want to edit.
2. Make your changes and click OK.

## Using parameters with XSL transformations

You can define parameters for your XSL transformation when adding the XSL Transformation server behavior to a web page. A parameter controls how XML data is processed and displayed. For example, you might use a parameter to identify and list a specific article from a news feed. When the page loads in a browser, only the article you specified with the parameter appears.

### To add an XSLT parameter to an XSL transformation:

1. Open the XSL Transformation dialog box. You can do this by double-clicking an XSL Transformation server behavior in the Server Behaviors panel (Window > Server Behaviors), or by adding a new XSL Transformation server behavior. For instructions, see [“Inserting XSLT fragments in dynamic pages” on page 755](#).
2. In the XSL Transformation dialog box, click the Plus (+) button next to XSLT Parameters.



3. In the Add Parameters dialog box, enter a name for the parameter in the Name text box. The name can only contain alphanumeric characters. It cannot contain spaces.
4. Do one of the following:
  - If you want to use a static value, enter it in the Value text box.
  - If you want to use a dynamic value, click the dynamic icon next to the Value text box, complete the Dynamic Data dialog box, and click OK. For more information, click the Help button in the Dynamic Data dialog box.
5. In the Default Value text box, enter the value you want the parameter to use if the page receives no run-time value.
6. Click OK.

### To edit an XSLT parameter:

1. Open the XSL Transformation dialog box. You can do this by double-clicking an XSL Transformation server behavior in the Server Behaviors panel (Window > Server Behaviors), or by adding a new XSL Transformation server behavior. For instructions, see [“Inserting XSLT fragments in dynamic pages” on page 755](#).
2. Select a parameter from the XSLT parameters list.
3. Click the Edit button.
4. Make your changes and click OK.

### To delete an XSLT parameter:

1. Open the XSL Transformation dialog box. You can do this by double-clicking an XSL Transformation server behavior in the Server Behaviors panel (Window > Server Behaviors), or by adding a new XSL Transformation server behavior. For instructions, see [“Inserting XSLT fragments in dynamic pages” on page 755](#).
2. Select a parameter from the XSLT parameters list.
3. Click the minus (-) button.

## Creating conditional XSLT regions

You can use Dreamweaver to create simple conditional regions or multiple conditional regions on an XSLT page. You can make a selection in Design view and apply a conditional region to the selection, or you can just insert a conditional region wherever the insertion point is in the document.

For example, if you wanted to display the word “Unavailable” next to the price of an item when the item is unavailable, you could type the word “Unavailable” on the page, select it, and then apply a conditional region to the selected text. Dreamweaver surrounds the selection with `<xsl:if>` tags, and only displays the word on the page when the data match the conditions of the conditional expression.

### To create a conditional XSLT region:

1. Select Insert > XSLT Objects > Conditional Region, or Insert > XSLT Objects > Multiple Conditional Region.
2. In the Conditional Region or Multiple Conditional Region dialog box, enter the conditional expression you want to use for the region.  
For more information, click the Help button in the dialog box.
3. Click OK.

## Editing a Conditional Region XSLT object

After you’ve added a conditional XSLT region to your page, you can make changes to it using the Property inspector.

### To edit a Conditional Region XSLT object:

1. Select the object by clicking the gray tab that surrounds the conditional region.
2. In the Property inspector (Window > Properties), edit your conditional expression in the Test text box.

## Inserting XSL comments

You can add XSL comment tags to a document, or you can wrap a selection in XSL comment tags.

### To add XSL comment tags to a document:

- Do one of the following:
  - In Design view, select Insert > XSLT Objects > XSL Comment, type the contents of the comment (or leave the text box blank), and click OK.
  - In Code view, select Insert > XSLT Objects > XSL Comment.

### To wrap a selection in XSL comment tags:

1. Switch to Code view (View > Code)
2. Select the code you want to comment.
3. On the Coding toolbar, click the Apply Comment button and select Apply `<xsl:comment></xsl:comment>` Comment.

## Performing XSL transformations on the client

You can perform XSL transformations on the client without the use of an application server. When you do so, a browser does the work of transforming the XML data, instead of an application server. You can use Dreamweaver to create such a page, however, client-side transformations require manipulation of the XML file containing the data you want to display. Additionally, client-side transformations will only work in modern browsers.

Macromedia recommends that you read [“About client-side XSL transformations” on page 740](#) before proceeding with any of the following procedures.

This section contains the following topics:

- [“Workflow for performing client-side XSL transformations” on page 761](#)
- [“Creating entire XSLT pages” on page 762](#)
- [“Linking XSLT and XML files” on page 762](#)

### Related topics

- [“About server-side XSL transformations” on page 737](#)

## Workflow for performing client-side XSL transformations

This section provides a list of steps you need to follow to perform client-side XSL transformations, and refers you to the sections in the documentation that elaborate on each procedure.

Macromedia recommends that you read [“About using XML and XSL with web pages” on page 735](#), [“About server-side XSL transformations” on page 737](#), and [“About client-side XSL transformations” on page 740](#) before building pages that display XML data.

To perform client-side XSL transformations, follow these steps:

- Set up a Dreamweaver site. See [Chapter 2, “Setting Up a Dreamweaver Site,” on page 79](#).
- Do one of the following:
  - In your Dreamweaver site, create an entire XSLT page. See [“Creating entire XSLT pages” on page 762](#).
  - Convert an existing HTML page to an entire XSLT page. See [“Converting HTML pages to XSLT pages” on page 750](#)
- If you haven’t already done so, attach an XML data source to the page. See [“Attaching XML data sources” on page 750](#). The XML file that you attach must reside in the same directory as the XSLT page.
- Bind your XML data to the XSLT page. See [“Displaying XML data in XSLT pages” on page 751](#).
- If appropriate, add a Repeat Region XSLT object to the table or table row that contains the XML data placeholder(s). See [“Displaying repeating XML elements” on page 753](#).
- Attach the XSLT page to the XML page. See [“Linking XSLT and XML files” on page 762](#).
- Post both the XML page and the linked XSLT page to your web server.
- View the XML page in a browser. When you do so, the browser transforms the XML data, formats it with the XSLT page, and displays the styled page in the browser.

## Creating entire XSLT pages

You must use an entire XSLT page for client-side transformations. (XSLT fragments don't work for this type of transformation.) For instructions on creating, binding XML data to, and formatting XSLT pages, see the following topics:

- “Creating XSLT pages” on page 748
- “Displaying XML data in XSLT pages” on page 751
- “Displaying repeating XML elements” on page 753
- “Applying styles to XSLT fragments” on page 763

## Linking XSLT and XML files

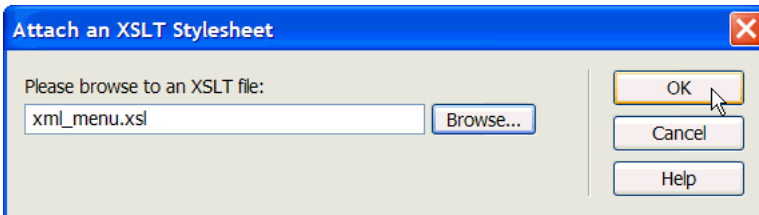
Once you have an entire XSLT page with dynamic content placeholders for your XML data, you must insert a reference to the XSLT page in the XML page.

NOTE

The XML and XSL files you use for client-side transformations must reside in the same directory. If they don't, the browser will read the XML file and find the XSLT page for the transformation, but will fail to find assets (style sheets, images, and so on) defined by relative links in the XSLT page.

### To link an XSLT page to an XML page:

1. Open the XML file that you want to link to your XSLT page.
2. Select **Commands > Attach an XSLT Stylesheet**.
3. In the **Attach an XSLT Stylesheet** dialog box, click the **Browse** button, browse to the XSLT page you want to link to, select it, and click **OK**.
4. Click **OK** to close the **Attach an XSLT Stylesheet** dialog box.



Dreamweaver inserts the reference to the XSLT page at the top of the XML document.

# Applying styles to XSLT fragments

When you create an entire XSLT page (that is, an XSLT page that contains `<body>` and `<head>` tags), you can display XML data on the page and then format the data like any other piece of content using the Property inspector or the CSS Styles panel. When you create an XSLT fragment for insertion in a dynamic page, however (for example, a fragment for insertion in an ASP, PHP, or Cold Fusion page), the rendering of styles in the fragment and in the dynamic page becomes more complicated. Although you work on an XSLT fragment separately from the dynamic page, it is important to remember that the fragment is intended for use within the dynamic page, and that the output from the XSLT fragment ultimately resides somewhere within the `<body>` tags of the dynamic page. Given this workflow, it is important to make sure that you do not include `<head>` elements (such as style definitions or links to external style sheets) in XSLT fragments. Doing so will cause the application server to place these elements into the `<body>` of the dynamic page, thereby generating invalid markup. For example, let's say you're creating an XSLT fragment for insertion in a dynamic page, and you want to format the fragment using the same external style sheet as the dynamic page. If you attach the same style sheet to the fragment, the resulting HTML page will contain a duplicate link to the style sheet (one in the `<head>` section of the dynamic page, and another in the `<body>` section of the page, where the content of the XSLT fragment appears). Instead of this approach, you should use Design-time style sheets to reference the external style sheet. When formatting the content of XSLT fragments, Macromedia recommends that you use the following workflow:

- First, attach an external style sheet to the dynamic page. (This procedure follows best practices for applying styles to the content of any web page).
- Next, attach the same external style sheet to the XSLT fragment as a Design-time style sheet. As the name implies, Design-time style sheets only work in the Dreamweaver Design view. For more information, see [“Using Design-Time style sheets” on page 403](#).

Once you have completed the previous steps you can apply existing styles or create new styles in your XSLT fragment using the same style sheet that you've attached to your dynamic page. You will have cleaner HTML output (because the reference to the style sheet is only valid while working in Dreamweaver), and the fragment will still display the appropriate styles in Design view. Additionally, all of your styles will be applied to both the fragment and the dynamic page when you view the dynamic page in Design view, or preview the dynamic page in a browser.

**NOTE**

If you preview the XSLT fragment in a browser, the browser does not display the styles. Instead you should preview the dynamic page in the browser to see the XSLT fragment within the context of the dynamic page.

For more information on using CSS to format XSLT fragments, see [www.macromedia.com/go/dw\\_xsl\\_styles](http://www.macromedia.com/go/dw_xsl_styles).

## Troubleshooting XSL transformations

If you are having trouble getting your XSL transformations to work, a troubleshooting guide with the answers to many frequently asked questions is available at [www.macromedia.com/go/dw\\_xsl\\_faq](http://www.macromedia.com/go/dw_xsl_faq).