

Building tutorials with the Toot-O-Matic

Presented by developerWorks, your source for great tutorials

ibm.com/developerWorks

Table of Contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

1. Getting started	2
2. Creating your first tutorial	5
3. developerWorks editorial guidelines	12
4. Toot-O-Matic tag guide	15
5. Toot-O-Matic tag reference	16
6. Troubleshooting	26
7. Resources and feedback	27

Section 1. Getting started

Overview

Tutorials are the most popular type of content on developerWorks. In the past, tutorials were created with anything from HTML editors to word processors. Once the content was created, we spent too much time making sure the look and feel of the tutorial panels was consistent, and we had to build and test all the links between panels by hand. As a site committed to open standards and cross-platform technologies, we knew there was a better way.

We've been working on Toot-O-Matic, a standards-based tool that uses XML and Java to build our tutorials. Using this tool, we author a relatively simple XML document that describes the content and structure of the tutorial. The tool takes the XML file (and any graphics we want to use) and converts it into a series of HTML files, JPEG graphics for the title texts of the tutorial and its sections, a ZIP file that contains everything you need to run the tutorial on your machine, and a PDF file that gives you a high-quality printable version of the tutorial. Best of all, the tool is completely standards-based, so you can run it on any machine that has a Java development kit (Java 2 1.3.0_02 works best).

To learn more about Toot-O-Matic and all of its wonders, read [XML training wheels](#), the developerWorks article that describes the structure and design of Toot-O-Matic.

This tutorial shows you how to use Toot-O-Matic to create your own tutorials. It also includes the [Content guidelines for developerWorks tutorials](#) on page 12 for creating tutorials. When you've completed this tutorial, you'll have the Toot-O-Matic tool installed on your machine, and you'll be ready to create your own tutorials!

Installing the Toot-O-Matic

The files you need to install are listed below. Be sure to put all the files in the same directory.

- * **tootomatic.zip** - The latest version of the Toot-O-Matic tool.
- * **jars.zip** - All the .jar files you'll need to run Toot-O-Matic, including:
 - * bsf.jar - IBM's Bean Scripting Framework, used by the XSLT processor
 - * fop.jar - Apache's FOP (Formatting Objects to PDF) tool, used to generate the PDF files
 - * w3c.jar - Contains code used by the FOP tool
 - * xerces.jar - Apache's XML parser
 - * xalan.jar - Apache's XSLT processor
 - * xml.jar - Contains code used by the FOP tool

You'll need the username and password you used for this tutorial (apologies for any inconvenience) to [download the tootomatic.zip and jars.zip files](#). When you've downloaded the files, unzip the files `tootomatic.zip` and `jars.zip`. **Do not** unzip any of the JAR files.

Toot-O-Matic uses the Java 2 SDK, 1.3.0_02. You can download it from <http://java.sun.com/products/archive/index.html>.

Note: To run Toot-O-Matic on Linux, X-Windows must be running. The Toot-O-Matic tool uses graphic libraries that are present in X-Windows.

Checking your installation

To make sure you've installed Toot-O-Matic correctly, follow these steps:

- * Go to a command prompt.
- * At the prompt, go to the directory where you installed the Toot-O-Matic files.
- * Type the command `tootomatic brainsurgery3.xml`.

If everything has been installed correctly, the tool will parse your XML file, then build a number of XML, JPEG, PDF, and ZIP files on your disk. When the tool is finished, you'll see a number of messages scroll down the screen as the various components of your tutorial are built. When it's done, you should be able to open the file `bsfyp\index.html`.

Acknowledgements

The current state of the Toot-O-Matic tool owes a great deal to the many members of the developerWorks tutorial family and their many suggestions and ideas. In alphabetical order, they are:

- * Dan Amerson
- * Laura Bennett
- * Barry Busler
- * Dave Clark
- * Tom Coppedge
- * Nancy Dunn
- * Jeanette Fuccella
- * Leah Ketring
- * Gretchen Moore
- * Jeanne Murray
- * Caroline Newsome
- * Lou Shannon
- * Andy Smith
- * Chris Stackel
- * Mike Tormey
- * Jackie Wheeler
- * Janet Willis

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).



About the author

My name is Doug and this is my picture. I got it tooked at the big park and I rode a great big ferris wheel that was as big as the sky and it went round and round and round and we got stopped at the top way up high when somebody else got on it. And I got a big blue balloon shaped like a bear head shape and I held on to it really tight until I forgot to and it flew away higher than the ferris wheel even and I got really really really really sad.

Senior Programmer Doug Tidwell is IBM's evangelist for Web Services. He was a speaker at the first XML conference in 1997, and has been working with markup languages for more than a decade. He built and [poorly] documented the Toot-O-Matic tool to simplify the process of writing tutorials. He holds a Bachelors Degree in English from the University of Georgia and a Masters Degree in Computer Science from Vanderbilt University.

In his spare time, he enjoys reading, traveling, and writing ludicrous autobiographies that are published, unedited, on developerWorks. He can be reached at dtidwell@us.ibm.com.

Section 2. Creating your first tutorial

Development process

The development process consists of several steps:

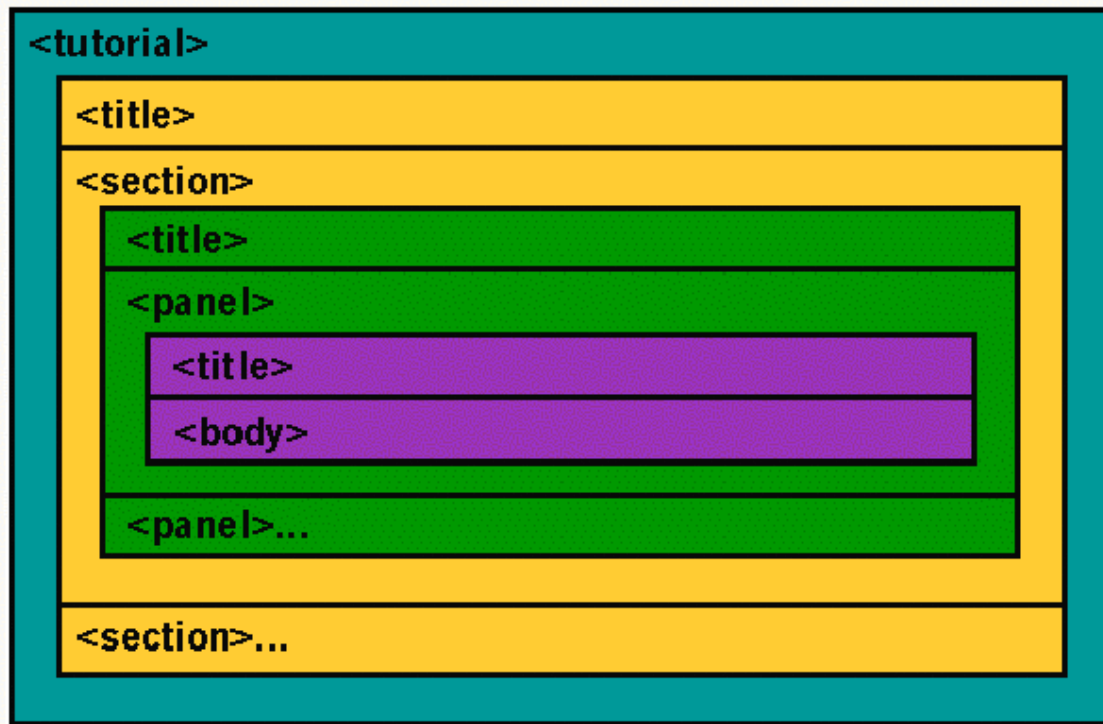
1. The subject matter expert creates the XML file. The best writing process starts with an outline of the topic; each topic in the outline becomes a section of the tutorial. The completed XML file contains all of the text of the tutorial. As the subject matter expert creates the XML file, she uses the Toot-O-Matic to make sure the generated tutorial is organized the way she wants.
2. A graphic designer works with the subject matter expert to decide what images the tutorial needs. The XML file references those files with ordinary household `` tags. As the graphic designer creates the images, he uses the Toot-O-Matic to make sure the generated tutorial looks the way he wants.
3. A production editor works with the subject matter expert to make sure the tutorial content meets the developerWorks editorial guidelines. As the production editor perfects the tutorial content, she uses the Toot-O-Matic to make sure the generated tutorial is ready for the Web.
4. When the subject matter expert, the graphic designer, and the production editor are done with the source files (the XML file and any supporting graphics), the Toot-O-Matic is run one last time. The HTML, JPEG, PDF, and ZIP files generated by the tool are then published to the Web.

Tutorial structure

The structure of a tutorial is fairly simple. The XML source document is structured as follows:

- * The entire document is contained in a `<tutorial>` element.
- * A `<tutorial>` contains a `<title>` and one or more `<section>`s.
- * A `<section>` contains a `<title>` and one or more `<panel>`s.
- * A `<panel>` contains a `<title>` and a `<body>`.

Here's a diagram that illustrates the structure of the XML source:



XML basics

As you create your XML source file, there are several things to keep in mind:

- * **XML is case-sensitive.** If you end a `<panel>` tag with a `</PANEL>` tag, you'll get an error. Use lower case for all tags. Otherwise, the PDF will not build correctly.
- * You can't leave out any end tags. That means you have to code the horizontal rule (`hr`) tag as `<hr />` and the break (`br`) tag as `
`.
- * All tags must be nested. If a tag begins inside another tag, it has to end inside that tag as well.
- * All attribute values must be quoted. You can use single quotes or double quotes, but you must quote the attribute's value.
- * There are five predefined entities in XML: Less than (`<`), greater than (`>`), ampersand (`&`), the single quote (`'`), and the double quote (`"`). If you need to use any of these characters inside an attribute value, you must use the entities instead.
- * Code samples must not be more than 80 characters in width. Anything over 80 won't appear in the PDF version of the tutorial.

If you break any of these rules, the Toot-O-Matic won't work. You'll get a typically cryptic error message; see [Troubleshooting](#) on page 26 for more information.

Of course, if you're writing your source file in an XML-aware editor like XML Spy, Adept or XMetal, the editor prevents you from making these mistakes.

Starting with an outline

The best way to start your tutorial is with an outline. To create the outline, start with some number of `<section>` elements, one for each section of your eventual tutorial. The code listing below is a good place to start. We'll take a look at what this source file generates, then we'll add more things to our XML source file as we go along.

To save yourself some typing, you can use the source files (`brainsurgery1.xml`, `brainsurgery2.xml`, and `brainsurgery3.xml`) for the examples we'll work through in this section.

BTW, notice the attributes of the `<tutorial>` tag. The `filename` attribute defines the prefix for all of the generated filenames. In this example, the third panel in the second section would be named `bs-2-3.html`. The `img` attribute is the filename of the graphic that appears on the menu panel.

```
<?xml version="1.0"?>
<tutorial
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file:///tootomatic.xsd"
  filename="gsfyp"
  img="images/tutorial.jpg"
  exit-url = "http://www.ibm.com/developerworks/web/>
  <title>Brain Surgery for Young People</title>
  <section>
    <title>Introduction</title>
  </section>
  <section>
    <title>Finding patients</title>
  </section>
  <section>
    <title>Dealing with lawsuits</title>
  </section>
  <section>
    <title>Consoling patients' relatives</title>
  </section>
```

So far, so good...

The file `brainsurgery1.xml` contains the tutorial skeleton we examined in the previous panel. To generate a tutorial from this file, type `tootomatic brainsurgery1.xml`. (Note: If you click on any of the links in the menu panel, you'll get an error message because none of our sections contain any panels. We'll fix that in a minute.)

The menu panel of our tutorial looks like this:



Brain Surgery for Young People

[Main menu](#)
[Section menu](#)
[Feedback](#)
[< Previous](#)
[Next >](#)


Main menu

Select any of these links to start the tutorial.

- > [1. Introduction](#)
- > [2. Finding patients](#)
- > [3. Dealing with lawsuits](#)
- > [4. Consoling patients' relatives](#)

[Privacy](#)
[Legal](#)
[Contact](#)

Next, we'll add some panels to all of the sections.

Adding some panels

Now we want to add some panels to our tutorial. We'll begin by adding a single `<panel>` to each of the `<section>`s in our XML file; you can look at `brainsurgery2.xml` to see the source. Here's what a sample panel looks like:

```
<section>
  <title>Introduction</title>
  <panel>
    <title>Getting Started</title>
    <body>
      <text-column>
        <p>Once the exclusive domain of trained surgeons, brain surgery
        has now become a profitable avocation for thousands of people
        around the world. Thanks to advances in medical instruments,
        persons of even below-average intelligence can now perform
        dangerous, life-threatening procedures on patients hoping
        to save money on a traditionally expensive operation.</p>
      </text-column>
    </body>
  </panel>
</section>
```

Filling in the `<tutorial>` tag

At this point, we've built a tutorial with a number of panels. The Toot-O-Matic has built a number of HTML and PDF files for us, handling all of the issues of navigation, look and feel, and formatting. To give this tutorial a finished look, we need to add some more attributes to the `<tutorial>` tag. Here's what the completed `<tutorial>` tag looks like:

```
<tutorial
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="file:///tootomatic.xsd"
  filename="bsfyp"
  img="images/tutorial.jpg"
  alt="Brain surgery graphic"
  feedback-link="feedback"
  zone="Web Architecture"
  abstract="Once the exclusive domain of trained surgeons..."
  exit-url="http://www.ibm.com/developer/web/"
  email-link="http://www-106.ibm.com/developerworks/education/r-wa-bs.html"
```

In this sample, the `feedback-link` attribute defines where the user should go when they click the Feedback button; the value of this attribute must match the value of an `id` attribute on a `<panel>` somewhere in your script. The `zone` attribute defines the developerWorks zone where the tutorial appears; this is used to generate the bread crumb trail at the top of each page of the tutorial. The `abstract` is a brief overview of the tutorial. This text is used in the "e-mail this tutorial to a friend" panel. The `e-mail-link` is the url where the tutorial is located (used for the e-mail to a friend function also). Finally, `exit-url` defines the URL the user should go to when they exit the tutorial.

Filling in the `<tutorial>` tag (continued)

After completing the `<tutorial>` tag, our navigation bar looks like this:



The links on the navigation bar go to the ZIP version of the tutorial, the letter-sized and A4-sized PDF files, and the "e-mail this tutorial to a friend" function. These links give users several different ways to view the content of your tutorial, and allow them to share the content with a friend.

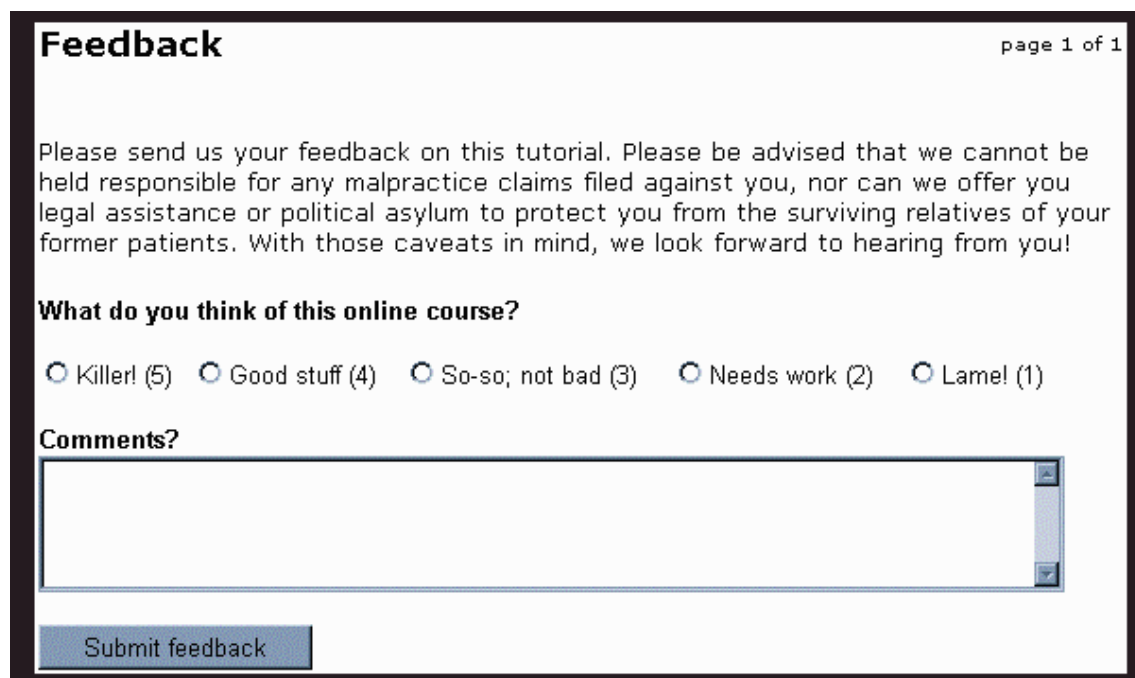
Adding a feedback form

The final thing we'll add is a feedback form. We'll create a new section that contains a single panel, and add the `<feedback-form>` tag to it. The source for this final version of the tutorial is `brainsurgery3.xml`. Notice that the value of the `id` attribute matches the value of the `feedback-link` attribute on the `<tutorial>` tag. Here's how the XML source looks:

```
<section>
  <title>Feedback</title>
  <panel id="fb">
    <title>Feedback</title>
    <body>
      <text-column>
        <p>Please send us your feedback on this tutorial.
        Please be advised that we cannot be held responsible for
        any malpractice claims filed against you, nor can we
        offer you legal assistance or political asylum to protect
        you from the next-of-kin of your former patients. With
        those caveats in mind, we look forward to hearing from
you!</p>
        <feedback-form
          action-url="http://www.ibm.com/developer/ratings.nsf/RateCourse?CreateDocument
            zone="Web"
            redirect-url="http://www.ibm.com/developer/thankyou/feedback-java.html"
        />
      </text-column>
    </body>
  </panel>
</section>
```

Adding a feedback form (continued)

Here's what the feedback form looks like:



The screenshot shows a web page titled "Feedback" with a page number "page 1 of 1" in the top right corner. The main content area contains a paragraph of text: "Please send us your feedback on this tutorial. Please be advised that we cannot be held responsible for any malpractice claims filed against you, nor can we offer you legal assistance or political asylum to protect you from the surviving relatives of your former patients. With those caveats in mind, we look forward to hearing from you!". Below this text is a section titled "What do you think of this online course?" with five radio button options: "Killer! (5)", "Good stuff (4)", "So-so; not bad (3)", "Needs work (2)", and "Lame! (1)". Underneath the radio buttons is a section titled "Comments?" with a large text input field. At the bottom of the form is a "Submit feedback" button.

Congratulations!

At this point, you've built a complete tutorial. The tutorial features a main menu panel, section indexes, a ZIP file that contains everything you need to run the tutorial on

another machine, and two PDF versions for high-quality printing. Users can easily tell someone else about the wonderful tutorial you've built. All you need to do now is replace these silly panels with useful content, and you'll have a professional tutorial that's ready for the Web.

Section 3. developerWorks editorial guidelines

Content guidelines for developerWorks tutorials

These guidelines will help you reach that goal. They begin with a discussion of developerWorks' objectives pertaining to tutorials. Then you'll find specific guidelines on how to decide if particular content is suitable for a tutorial and how to assess the characteristics of a good tutorial.

A tutorial teaches a specific skill, such as understanding a concept or how to do a task. A good tutorial states an objective and how the user will achieve it, demonstrates concepts with example code, and uses interaction where appropriate. developerWorks tutorials must be easy to use on the Web and must be easy to navigate. Tutorials must also meet audience expectation for professional material, especially in terms of visual appeal.

developerWorks tutorial objectives

dW's objective is to provide technical education to developers about key open-standard technologies. The material must:

- * Describe technologies and demonstrate concepts with example code
- * Be short (under 1 hour) and easy to enter and exit
- * Be easy to use directly from the dW Web site

Audience

The audience for dW tutorials is worldwide developers who have technical background and aptitude but varying levels of knowledge about these key technologies. They are technically sophisticated, would rather see code than read about it, and want information quickly.

Access to high-speed communication lines may be limited, especially for non-U.S. users. For this reason, our tutorial format is easy to use from the Web, and we also offer PDF files (both letter-size and A4) and a ZIP file for download.

Delivery of tutorials

The primary medium for delivery is on the dW Web site. Tutorials must be compatible with multiple browsers and use readily available technology. A secondary delivery medium is on CD for distribution at trade shows.

How do I decide whether content is a tutorial or a

how-to article?

The line is admittedly blurry; they do have a lot in common. When determining whether to use tutorial format or a how-to article, consider:

- * **Interactivity** - Ideally, tutorials are hands-on and interactive.
- * **Tone** - Tutorials use a declarative tone.
- * **Objective** - Is there a clear learning objective? Can the user walk away and do something?
- * **Voice** - If there are multiple voices to convey, including alternative approaches from other sources, or if there is a need to highlight a particular voice (for example, a technical rock star), the how-to article may be the better vehicle.
- * **Universality** - "How I solved a problem" is not always suitable for a tutorial because it may not stand the test of time.
- * **Visual content** - Tutorials typically have a more visual presentation.

What are characteristics of a good tutorial?

All tutorials should follow the guidelines for clearly written technical material that apply to every piece of content on dW. In addition, a good tutorial has the following characteristics:

1. States a clear objective
2. Lists prerequisite knowledge (and where to obtain it)
3. Contains a set of specific tasks for achieving the objective
4. Is focused on the task at hand ("need to know" vs. "nice to know")
5. Takes the show-me approach, using real code
6. Is interactive where appropriate
7. Delivers information in short chunks (to help with comprehension and retention)
8. Meets audience expectation for professional material
9. Is visually appealing but not cutesy
10. Is easy to navigate, to know where you are in accomplishing the objective, and to exit

While the objective must be measurable, it may or may not be necessary to actually test the user. However, the user should always be able to tell whether they've achieved the objective. Whether the tutorial uses a summary or a quiz to do this is up to the author.

Content

Follow these guidelines for the content of your tutorials.

- * The abstract must be compelling and entice the reader to register for the tutorial.
- * The first section should be "Tutorial tips" which includes at least two panels:
 - * Should I take this tutorial?
 - * Contact, giving the author's e-mail and bio

- * End the tutorial with "Wrapup" or a similar section that may give a summary of the tutorial but will always solicit feedback.
- * Follow guidelines for clearly written technical material; in particular, use sentence-style capitalization and use serial commas.
- * Make the appropriate tool decisions that affect the appearance of the tutorial. For example, should the tutorial be in two-column format or one? Are graphics needed?

Section 4. Toot-O-Matic tag guide

Overview

This section of the tutorial covers tasks that you'll likely encounter as you're building a tutorial. If you're viewing this online, you can use the Section menu button to go directly to the description of a particular task. Remember that all tags must be in lower case.

If you prefer schema details, take a look at the [Toot-O-Matic-Schema.pdf](#) file.

Adding a feedback form

If you want to add a feedback form to your tutorial, simply add a `<feedback-form>` element to any panel. Typically a tutorial only has one feedback panel, and it is referenced by the `feedback-link` attribute of the `<tutorial>` element.

Common errors

Several errors can cause perplexing results:

- * Using any uppercase tags or attributes
- * Leaving out the abstract attribute of the `<tutorial>` tag

Section 5. Toot-O-Matic tag reference

Overview

This section describes the tags used in Toot-O-Matic. You can also refer to the schema documentation in [Toot-O-Matic-Schema.pdf](#) .

Symbols

The Toot-O-Matic defines several symbols you can use in your scripts. They are listed in the following table:

Symbol	Text
<code>&feedback;</code>	Feedback
<code>&mainmenu;</code>	Main menu
<code>&moreinfo;</code>	More info...
<code>&next;</code>	Next
<code>&nextsection;</code>	Next section
<code>&nbsp;</code>	[Non-breaking space]
<code>&previous;</code>	Previous
<code>&sectionmenu;</code>	Section menu

<a> - Define a hypertext anchor

Description: The <a> element defines a hypertext link.

Parents: The <a> element can appear inside the , <code>, <i>, , or <p> elements.

Children: The text of an anchor tag can contain , <code>, or <i> elements.

Attributes:

- * **href** - Defines the target URL of this hypertext link.
 - * **target** - Defines the target frame of this hypertext link. A value of `_blank` (the only valid value) means the target URL will appear in a new browser window.
-

 - Boldfaced text

Description: The element contains boldfaced text.

Parents: The element can appear in the <code>, <i>, , <p>, <prompt>, and <response> elements.

Children: The `` element contains text and optional `<i>` or `<code>` elements in any order. The markup "`This is bold text <i> with italics</i> mixed in.`" is valid.

Attributes: None.

`<body>` - The contents of a panel

Description: The `<body>` element defines the contents of a panel.

Parents: The `<panel>` element.

Children: `<image-column>`, `<example-column>`, and `<text-column>`. Of these three, `<text-column>` is required. The `<image-column>` and `<example-column>` elements are optional, and only one of the two may appear at the same time.

Attributes: None.

`
` - Line break

Description: The `
` element defines a line break. Keep in mind that a Toot-O-Matic script is an XML file, so you have to code the break tag as either "`
</br>`" or "`
`".

Parents: The parents of `
` are ``, `<p>`, `<prompt>`, `<response>`, and `<title>`. The `
` element can appear any number of times in these elements.

Children: None.

Attributes: None.

`<code>` - A piece of code

Description: The `<code>` element defines a very short code listing. For longer code listings, use the `<code-listing>` element. Unlike `<code-listing>`, `<code>` does not preserve white space.

Parents: The `<code>` element appears inside `<a>`, ``, `<p>`, `<prompt>`, `<response>`, and `<title>` elements. `<code>` is an optional element that can occur any number of times.

Children: The `<code>` contains text, and can also contain the universally loved `` and `<i>` elements.

Attributes: None.

<code-listing> - A long piece of code

Description: The <code-listing> element defines a long piece of code. For shorter code listings, use the <code> element. The content of the <code-listing> element is formatted in a monospaced font and all white space is preserved (as it appears in the XML source file). Be sure to use spaces and not tab characters to indent the code samples; tabs cause problems in the pdf file. If a given line of code begins with 23 spaces, the formatted output will begin with 23 spaces as well.

Parents: The <code-listing> element can appear in the <example-column> or <text-column> elements.

Children: None.

Attributes: None.

<example-column> - Define a column for a code sample

Description: This element contains a code listing. The code is displayed in a monospaced font to the left of the text area.

Parents: The <body> element. <example-column> is an optional element that may occur once or not at all. There can be at most one <example-column> or <image-column> element; if you have one, you can't have the other.

Children: The <code-listing> element.

Attributes: None.

<feedback-form> - Create a feedback form

Description: The <feedback-form> element creates a form for user comments.

Parents: <feedback-form> must appear inside a <text-column> element.

Children: None.

Attributes:

- * **action-url** (required) - The URL to which user feedback should be submitted.
 - * **redirect-url** (required) - The URL to which the user should be redirected.
 - * **zone** - The developerWorks zone for this tutorial.
-

<fileref> - File reference

Description: The `<fileref>` element references an external file, typically a code listing. When rendered in either HTML or PDF, the element is replaced with the value of the `href` attribute. All files referenced with `<fileref>` elements are automatically added to the zip file created by Toot-O-Matic. The `href` attribute is required.

Parents: `<p>`, ``, `<table-cell>`

Children:

Attributes: `href`

`<i>` - Italicized text

Description: The `<i>` element contains italicized text.

Parents: The `<i>` element can appear in the ``, `<code>`, ``, `<p>`, `<code-listing>`, and `<table-cell>` elements.

Children: The `<i>` element contains text and optional `` and `<code>` elements in any order. The markup "`<i>This is italicized text with bold mixed in.</i>`" is valid.

Attributes: None.

`<image-column>` - Define a column for an image

Description: The `<image-column>` defines an area to the left of the text area that contains a graphic.

Parents: The `<body>` element.

Children: None.

Attributes:

- * **alt** - Defines the alternate text for the image file.
 - * **img** - Defines the URL of the image file.
-

`` - Define an image

Description: The `` element defines an image. If you use this tag, you must include the `alt` attribute for accessibility reasons.

Parents: Either ``, `<table-cell>` or `<p>`.

Children: None.

Attributes:

- * **alt** - Defines the alternate text for the image file.
- * **src** - Defines the URL of the image file.

 - Define an item in a list

Description: The element defines an item in a list.

Parents: Either or .

Children: A list item can contain text, <a>, ,
, <code>, <fileref>, <i>, , <p>, , or <xref> any number of times, in any order.

Attributes: None.

 - Define an ordered list

Description: The element defines an ordered list.

Parents: can appear inside the <p>, , , <table-cell>, <text-column>, or elements.

Children: An ordered list element must contain at least one element.

Attributes: None.

<p> - Define a paragraph

Description: The <p> element defines a paragraph. Nuff said.

Parents: <p> can appear inside and <text-column>.

Children: A <p> element can contain text and any combination of the following elements: <a>, ,
, <code>, <feedback>, <fileref>, <i>, , , <xref> or .

Attributes: None.

<panel> - Define a tutorial panel

Description: The <panel> element defines the contents of a tutorial panel. A tutorial panel consists of a <title> element, followed by a <body> element. If you need to refer to this panel from other parts of the tutorial, you can use the id attribute to name this

panel. You then use the <xref> element to refer back to this panel. The text of the <title> element is displayed in a larger font at the top of the panel.

Parents: The <section> element. The <panel> element must appear at least once, and can appear any number of times.

Children: One <title> element and one <body> element.

Attributes:

- * **id** - Defines an ID for this panel. The value of the ID must be unique across all of the panels and sections in the tutorial.

<section> - Define a tutorial section

Description: The <section> defines a major topic in a tutorial. It contains a <title> element, followed by one or more <panel> elements. As with the <tutorial> element, the text of the title is converted to a JPEG file used in the graphical version of the tutorial.

Parents: The <tutorial> element. The <section> element must appear at least once, and can appear any number of times.

Children: One <title> element, followed by one or more <panel> elements.

Attributes:

- * **id** - Defines an ID for this section. The value of the ID must be unique across all of the sections and panels in the tutorial.

<table> - Define a table

Description: The <table> element defines a table. To accommodate the special needs of the PDF generator, the Toot-O-Matic's table tags are slightly different than those you're used to in HTML. Here's what a Toot-O-Matic table looks like:

```
<table border="1">
  <table-column column-width="150pt" />
  <table-column column-width="150pt" />
  <table-header>
    <table-cell>Symbol</table-cell>
    <table-cell>Text</table-cell>
  </table-header>
  <table-row>
    <table-cell><code>&feedback;</code></table-cell>
    <table-cell>Feedback</table-cell>
  </table-row>
</table>
```

Formatted, the table looks like this:

Symbol	Text
<code>&feedback;</code>	Feedback

Parents: A table can appear inside a `<text-column>`, `<table-cell>` or another `<table>`.

Children: A table must have at least one `<table-column>`, an optional `<table-header>` (if there is a `<table-header>`, there can only be one), followed by one or more `<table-row>`s.

Attributes:

- * **border** - Defines whether or not this table will have a border. A value of 1 means the table will have a border; if this attribute is missing or has any other value, the table will not have a border.

`<table-cell>` - Define a table cell

Description: The `<table-cell>` element defines an individual cell in a table. It is equivalent to the HTML `<td>` tag.

Parents: Either the `<table-heading>` or the `<table-row>` element.

Children: This tag can contain text or any combination of `<a>`, ``, `
`, `<code>`, `<file-ref>`, `<i>`, ``, ``, `<table>`, `<p>`, `<xref>`, or `` elements.

Attributes:

- * **align** - Defines the alignment of the table cell. Allowable values are `left`, `center`, and `right`.

`<table-column>` - Define the width of a table column

Description: The `<table-column>` element defines the width of a column in the table. **You must have one `<table-column>` element for every column in your table.**

Parents: The `<table>` element. `<table>` must contain at least one `<table-column>`.

Children: None.

Attributes:

- * **column-width** - Defines the width of the column. You should define this in points, as in `column-width="200pt"`.

`<table-header>` - Define a header row for a table

Description: This element defines a header row for the columns of a table. All text in the header row appears in boldface font.

Parents: This can only appear in a <table> element.

Children: One or more <table-cell> elements, one for each <table-column> defined in the table.

Attributes: None.

<table-row> - Define a row in a table

Description: This element defines a row in a table.

Parents: This can only appear in a <table> element.

Children: One or more <table-cell> elements.

Attributes: None.

<text-column> - Define the text area of a panel

Description: The <text-column> element defines the content that should appear in the text area of a panel. If the panel does not contain any <image-column> or <example-column> elements, the text area occupies the entire width of the panel.

Parents: The <body> element.

Children: A <text-column> can contain any combination of <code-listing>, <feedback-form>, <table>, , , <p>, or elements.

Attributes:

- * **break** - Defines whether or not the text area can be split across two pages when it is rendered in the PDF file. Allowable values are `yes` and `no`, and the default value is `no`. This attribute is optional.
-

<title> - Define the title of a tutorial, section, or panel

Description: The <title> element defines the title of something, whether it's a tutorial, a section, or a panel.

Parents: The <title> element can appear in the <panel>, <section>, or <tutorial> elements. <title> must appear exactly once in these elements.

Children: None.

Attributes: None.

<tutorial> - Define the tutorial

Description: This is the root element of the tutorial document. A <tutorial> element contains a <title> element and some number of <section> elements; each <section> contains some number of <panel> elements. The text of the <title> element is converted into a JPEG file; the JPEG file is used in the tutorial display. The title text is also used as the HTML <title> element.

Parents: None.

Children: One <title> element, followed by one or more <section> elements.

Attributes:

- * **abstract** - A short description of the tutorial. This text appears inside the "e-mail this to a friend" panel when the user clicks the e-mail icon.
- * **email-link** - The e-mail-link is the url where the tutorial is located (used for the e-mail to a friend function also).
- * **alt** - Alternate text for the image. This attribute is required.
- * **discussion-link** - The URL of the discussion forum for this tutorial. **This is not currently used, but will be added when discussion forum support is added to the tool.**
- * **exit-url** - The URL of the document that should be referenced when the user exits the tutorial.
- * **feedback-link** - The `id` of the <panel> that contains the feedback form used by this tutorial. When viewing the HTML version of the tutorial, if you click on the Feedback link in the navigation area, the browser goes to the panel whose `id` matches the value you specified on this attribute.
- * **filename** - This defines the base filename for the tutorial. If you define a base filename of "bob," your files will be written to a directory named "bob", and they will be named bob-1-1.html, bob-1-2.html, etc. This attribute is required.
- * **img** - The image file used on the main panel of the tutorial. This attribute is required.
- * **zone** - The developerWorks zone in which this tutorial will appear. The value of this attribute is used to build the breadcrumb trail at the top of each panel. This attribute is required, and must be one of the following **case-sensitive** values: Components, Java, Linux, Open Source, Security, Unicode, Web, Web Architecture, Web Services, Wireless or xml.

 - Define an unordered list

Description: The unordered list element defines an unordered list. All of the items in the list are preceded by a bullet.

Parents: can appear inside the <p>, , , <text-column>, or elements.

Children: An unordered list element must contain at least one `` element.

Attributes: None.

`<xref>` - Define a cross-reference

Description: This creates a cross-reference to another panel or section. This is an empty tag (it contains no text). In the HTML files, this tag is replaced by a hyperlink to the other panel or section, with the `<title>` of the panel or section as the text of the link. In the PDF file, this tag is replaced by the title of the other panel or section and the page number on which it appears (both of which are hyperlinks if you're viewing the PDF file online).

As an example, this `<xref>`:

```
<ul>
  <li><xref refid="feedback-form-tag"/></li>
</ul>
```

Produces this result:

- * [<feedback-form>](#) - [Create a feedback form](#) on page 18

Parents: `<xref>` can appear inside the ``, `<code>`, `<i>`, ``, and `<p>` tags.

Children: None. This is an empty element.

Attributes:

- * **refid** - Refers to the ID of the referenced panel. For the cross-reference to resolve correctly, the `refid` attribute must match the `id` attribute of one of the `<panel>`s in your XML source file.

Section 6. Troubleshooting

Overview

This section covers common problems and shows you how to deal with them. (This section is included, of course, in lieu of more robust error checking and validation code in the tool itself.) If you're viewing this online, you can use the Section menu button to go directly to the description of a particular error.

There's nothing wrong at the location of the error

Although the XML parser does its best to tell you where an error occurred, it isn't always terribly helpful. Consider this piece of markup:

```
<text-column>
  <p>Here's a paragraph, yada yada yada.</p></p>
</text-column>
```

The error message you'll get will tell you that the element `<text-column>` must be terminated by the correct end tag. If you just glance at the file, there's clearly a `<text-column>` tag and an end `</text-column>` tag. The problem here is that there are two `</p>` tags, which confuses the parser. When you encounter something like this, it's almost always an extra end tag that's causing the problem.

Another hint: be sure wrap is off if you are using a text editor that wraps text in the file while editing.

XML is case-sensitive

Unlike HTML, XML is case-sensitive. All tags must be in lower case. If you close a `<panel>` tag with a `</PANEL>` tag, you'll get an error message like this one:

```
file:///D:/projects/toot-o-matic/brainsurgery3.xml;
Line 21; Column 12
XSL Error: Could not parse brainsurgery3.xml document!
XSL Error: SAX Exception
Unfortunately, errors occurred:
Something went wrong when transforming your file:
org.apache.xalan.xslt.XSLProcessorException: The element type "panel" must be
terminated by the matching end-tag "</panel>".
```

There are a couple of hints to what's wrong here: First of all, we get the line and column number where the parser found an error. (That may or may not be where the actual error is, it's just where the parser got confused.) Our second hint here is the message that the `<panel>` tag wasn't terminated with the correct tag. The key piece of information is that the wrong end tag was used, so we have to examine the `<panel>` tag to find the error.

Section 7. Resources and feedback

Resources

- * Download the free [Toot-O-Matic package](#) , which includes:
 - * The sets of XML, XSL, Java, and .jar files that you'll download to your hard drive to run the tool. You'll need to be running the Java 2 SDK, 1.3.0_02.
 - * Sample tutorial files and Toot-O-Matic documentation.
 - * A readme file.

 - * Find out how developerWorks produced the Toot-O-Matic in [XML training wheels](#).
 - * Check out some other articles on developerWorks by Doug Tidwell that might be useful background as you work with the Toot-O-Matic:
 - * [Java developers: Fill your XML toolbox](#): The Toot-O-Matic package comes with all of the files you need to build tutorials. If you want to start working with the various technologies used by the Toot-O-Matic, this article will help you install the tools you'll need.
 - * [Converting XML into HTML, Converting XML into SVG, and Converting XML into PDF](#): The three-part series takes several XML documents and converts them into a variety of other formats. If you're just getting started with XSLT, these articles will help.

 - * [What kind of language is XSLT?](#) by Michael Kay, creator of the open-source Saxon XSLT processor, gives a good overview of XSLT as a language, covering its design goals and including some sample style sheet examples.
 - * For details about Xalan and Xerces, the open-source XSLT processor and XML parser used in the Toot-O-Matic, go directly to the source at [the Apache XML Project home page](#).
 - * If you want to know how IBM's WebSphere Application Server (WAS) supports XML development, see this [technical background info on XML in the WAS Advanced Edition 3.5 online help](#).
-

Feedback

Please send us your feedback on this tutorial and the Toot-O-Matic tool itself. The tool has benefitted greatly from the many suggestions and ideas contributed by the Toot-O-Matic user community. We look forward to hearing from you!

Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The Toot-O-Matic tool is an XSLT stylesheet and several XSLT extension functions that convert an XML file into a number of HTML pages, a zip file, JPEG heading graphics, and two PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML. (It also saves our production team a great deal of time and effort.)