twister discussion

discussion interface customization guide

1/05/00

notices

trademarks

www.bcandid.com

contents

1: introduction

bCandid's Twister news server application brings newsgroup discussion to the Web. Twister provides:

- newsgroup-specific XML elements that simply plug into existing Web designs; Web sites can now display newsgroups as part of the site, without requiring a special news browser
- existing NNTP news servers handle NNTP/HTTP through Twister's management of HTTP transmission.
- cost control through bandwidth consumption management
- ISPs and their customers can display discussion, customizing the discussion "look" so that it seamlessly integrates into each Web site
- extensive content that can be brought to each site via the Twister XML discussion elements
- defense against spam for ISP subscribers
- scalability to support millions of users

about this guide

This guide reviews the use of Twister to customize the newsgroup/discussion interface. With this guide, those who use Twister to power newsgroups and who wish to integrate newsgroups into Web sites, can learn how to use the Twister templates and XML elements to add newsgroup discussion to their Web sites. The resulting Web site integrates discussion without requiring special news reader software or expertise with the sometimes arcane art of news reading.

This guide assumes that you are familiar with HTML coding and comfortable editing HTML code by hand.

Note that Twister interface customization can be handled as desired on your site. This guide describes core Twister interface elements and how they work, and includes some sample coding fragments. Typically, you will use this guide only if you are customizing the discussion interface; you do not need to review this guide if you are outsourcing customization work to bCandid. (For more information on this service, contact sales@bCandid.)

If you are customizing the interface, then you may want to consider initially using the templates bCandid provides, along with the library of "theme" files provided with Twister. We recommend that, regardless of how you customize the provided templates, you do not edit the theme files we provide; instead, create companion theme files that you reference as needed. This simplifies maintenance of your customized bCandid template. Otherwise, you'll need to edit the bCandid library of theme files every time you update the Twister template/libraries. Templates are in the Twister directory, templates subdirectory; libraries are in the templates/lib subdirectory.

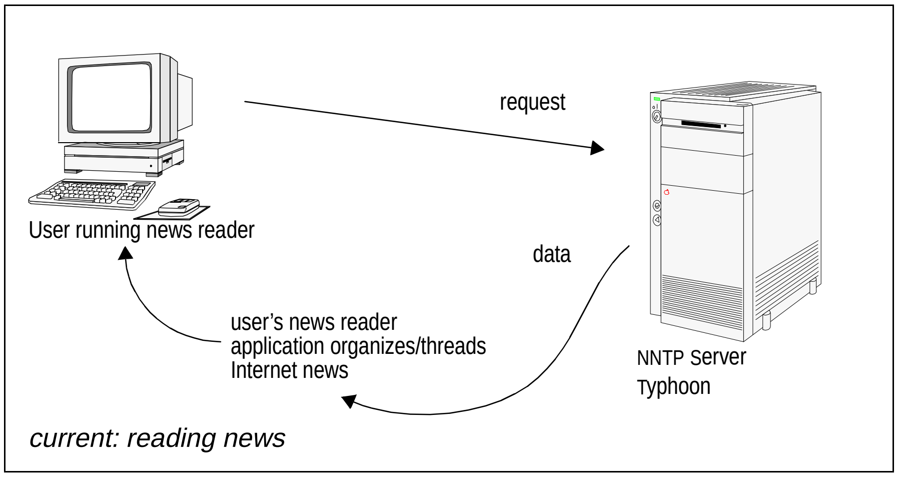
If, after you are familiar with the bCandid template files, you decide to build your own interface, consider creating your own version of the library files rather than editing the bCandid provided library, as this protects you from relying on files that will probably continue to change as features are added to the Twister interface command set.

Terms used in this guide are defined in glossary on page 8. This guide uses the following conventions:

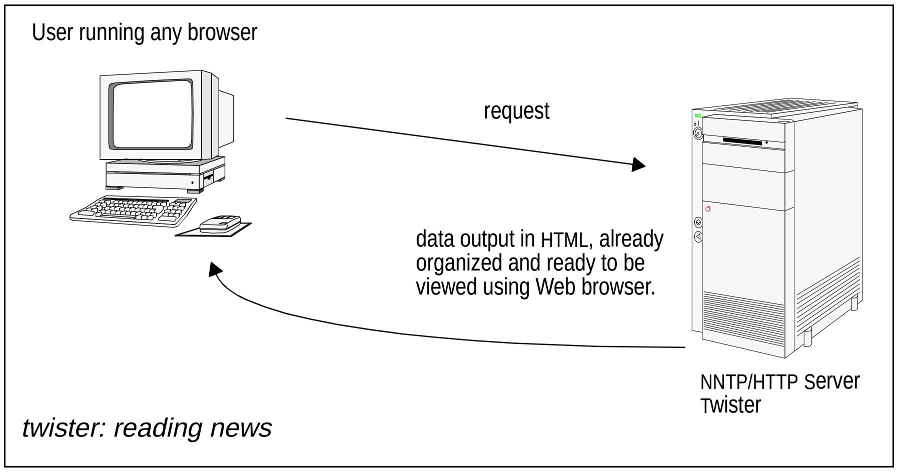
|	specify only one
[ ]	enclose multiple elements, from which you choose one
< >, /, other HTML markers	as used with HTML
" "	text you supply
@	separator used in Twister-enhanced URLs

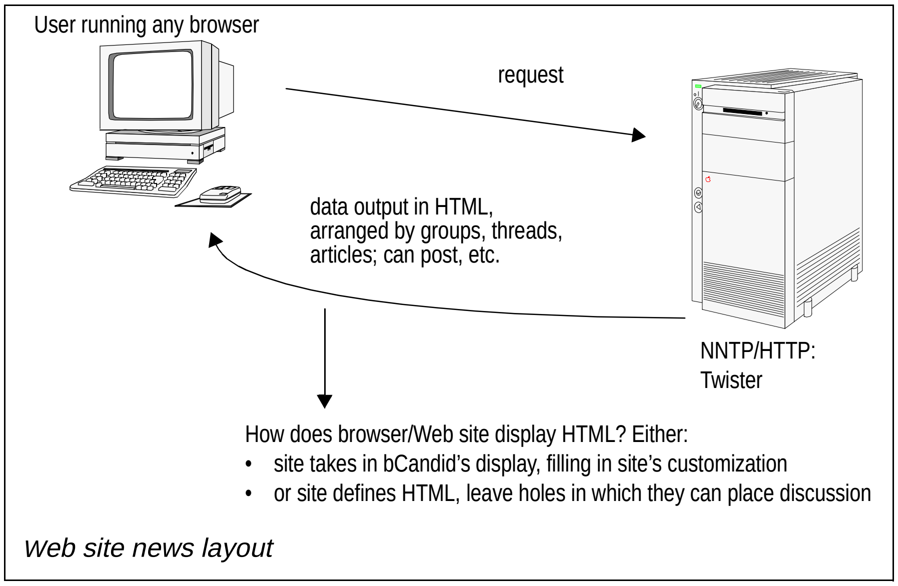
accessing discussion

Before Twister, users viewed discussion primarily through special news reader applications, such as Outlook Express and Agent. This has prevented Internet newsgroups from reaching the large audience already captured by the Web.



Twister changes this. By integrating the Twister news reader application with a flexible, and robust NNTP Internet news server software application, Internet discussion moves beyond the limits of news reader display, into the realm of the Web.


User running any browser
request
data output in HTML, already organized and ready to be viewed using Web browser.
NNTP/HTTP Server
Twister
twister: reading news

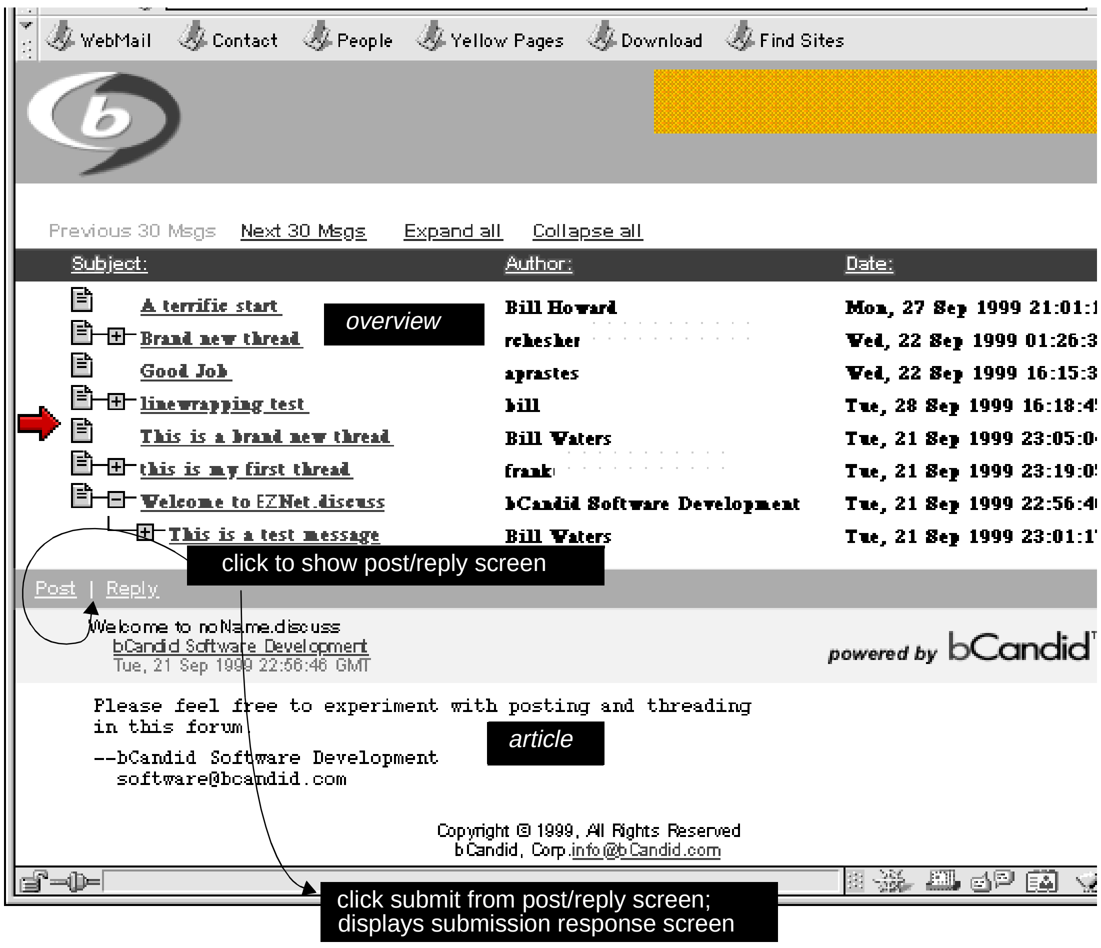

User running any browser
request
data output in HTML, arranged by groups, threads, articles; can post, etc.
NNTP/HTTP: Twister
How does browser/Web site display HTML? Either:
• site takes in bCandid's display, filling in site's customization
• or site defines HTML, leave holes in which they can place discussion
Web site news layout

Twister also provides something rare in the software application universe—a chance for you, the Web site user, to select and display the data you want, rather than having to use the layout handed to you by the software application.

Using special Twister-specific components, edit an HTML Web page so that it tells Twister the data to hand over, and how that data looks. This involves:

- adding Twister-specific tags and variables to the discussion Web pages.
- adding a link to the discussion Web page; this link is an extended URL whose format is defined by Twister. The link tells Twister the kind of data to be displayed (for example, an article or a list of articles).
- adding a line to discussion Web pages that tells Twister another special file to use: a dtd—document type definition, shipped with Twister.

a tour of web-based discussion


WebMail
Contact
People
Yellow Pages
Download
Find Sites
overview
click to show post/reply screen
article
click submit from post/reply screen; displays submission response screen
powered by bCandid

Previous 30 Msgs Next 30 Msgs Expand all Collapse all

Subject:	Author:	Date:
A terrific start	Bill Howard	Mon, 27 Sep 1999 21:01:1
Brand new thread	rchesher	Wed, 22 Sep 1999 01:26:3
Good Job	aprastes	Wed, 22 Sep 1999 16:15:3
linewrapping test	bill	Tue, 28 Sep 1999 16:18:4
This is a brand new thread	Bill Waters	Tue, 21 Sep 1999 23:05:0
this is my first thread	frank	Tue, 21 Sep 1999 23:19:0
Welcome to EZNet.discuss	bCandid Software Development	Tue, 21 Sep 1999 22:56:4
This is a test message	Bill Waters	Tue, 21 Sep 1999 23:01:1

Post | Reply

Welcome to noName.discuss
bCandid Software Development
Tue, 21 Sep 1999 22:56:46 GMT

```
Please feel free to experiment with posting and threading
in this forum.

--bCandid Software Development
  software@bcandid.com
```

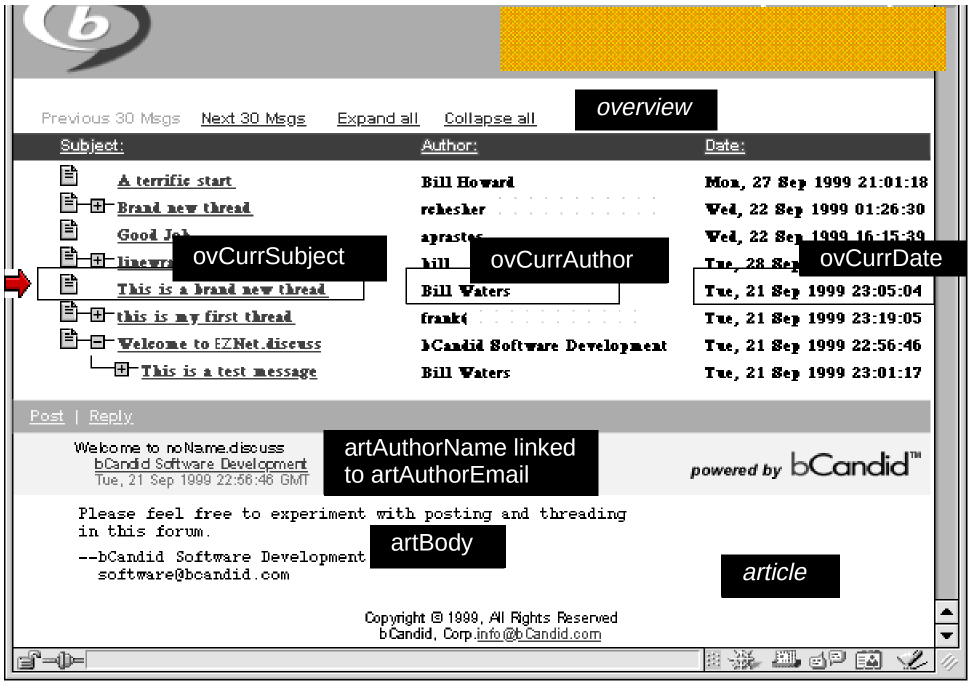
Copyright © 1999, All Rights Reserved
bCandid, Corp info@bCandid.com

The elements of discussion are:

- Overview: list of articles in this discussion—shown above.
- Article—text of article, used also in post/reply screen; to display and create a post or reply, click the post/reply button that displays in the overview/article screen.
- Submit the post/reply message—click the submit button shown on the post/reply screen; a screen showing the status of the submission then displays.

The example above shows two parts of discussion: overview and article. A few elements make up each portion of the layout:

layout portion	data displayed	twister variables
overview	subject author date	ovCurrSubj ovCurrAuthor ovCurrDate
article	text author author's e-mail address	artBody artAuthorName artAuthorEmail



Subject:	Author:	Date:
A terrific start	Bill Howard	Mon, 27 Sep 1999 21:01:18
Brand new thread	rehesher	Wed, 22 Sep 1999 01:26:30
Good Jo	aprastes	Wed, 22 Sep 1999 16:15:39
linewra	bill	Tue, 28 Sep
This is a brand new thread	Bill Waters	Tue, 21 Sep 1999 23:05:04
this is my first thread	frank(	Tue, 21 Sep 1999 23:19:05
Welcome to EZNet.discuss	bCandid Software Development	Tue, 21 Sep 1999 22:56:46
This is a test message	Bill Waters	Tue, 21 Sep 1999 23:01:17

Welcome to noName.discuss
bCandid Software Development
Tue, 21 Sep 1999 22:56:46 GMT

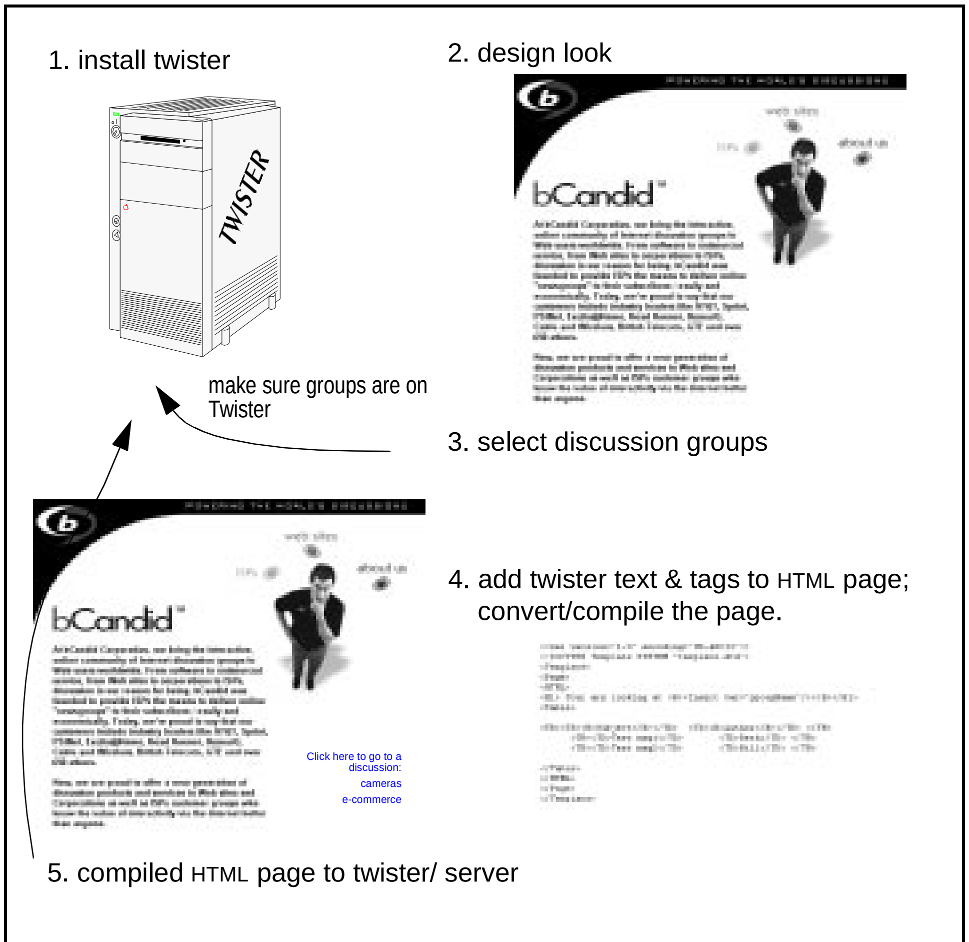
```
Please feel free to experiment with posting and threading
in this forum.
--bCandid Software Development
  software@bcandid.com
```

Copyright © 1999, All Rights Reserved
bCandid, Corp.info@bCandid.com

Each element of data in a discussion can be called using a specific Twister/XML tag and variable, and inserted into the Web page HTML text. This guide reviews the discussion-specific Twister XML variables you can use to introduce discussion into a Web site.

This guide also reviews:

- where to place files, so that Twister can find them
- how to create a link from a Web page to the discussion pages
- how to compile/process the file for use with Twister


1. install twister
TWISTER
2. design look
bCandid
make sure groups are on Twister
3. select discussion groups
4. add twister text & tags to HTML page; convert/compile the page.
bCandid
Click here to go to a discussion:
cameras
e-commerce
5. compiled HTML page to twister/ server

setting up a twister news page

glossary

action-handlers. Twister-specific processes that handle specific kinds of data.

article. Any message or reply in discussion. See also discussion.

discussion. Exchange of messages/information within a subject area; these messages are sent to a central Internet site, then redistributed using Usenet. Discussion is also referred to as Internet news. See also discussion.

group. An area of common interest, which has specific group names that post messages relating to specific subjects.

list. A specific news group, set up in Usenet; see also discussion.

news. See discussion.

post. Any message sent to a newsgroup.

reply. A message sent in response to an article in a discussion.

thread. The way in which related messages within a discussion are connected.

tpt. Extension used to denote a compiled Twister file.

URLs for Twister. URLs (Uniform Resource Locator) enhanced to provide additional information specific to Twister.

Usenet. A network of systems dedicated to handling news/discussion; available around the world.

2: twister components

Twister uses multiple components to set up a Web-based newsgroup display. These elements include:

- standard Web page HTML.
- Twister tags and variables, including optional imported files and user-defined variables.
- Twister-style URLs and associated templates, with some required lines and file structure.
- Twister programs that generate “well-formed” XML and compiled data “*.tpt” files that Twister uses.

This chapter reviews, at a structural level, what the elements are and how they fit together.

As reviewed in chapter 1: introduction, Twister models its tags, and its display of discussion, on the elements common to news discussion:

- overview of newsgroup articles from which to choose.
- article handling; this includes displaying article text selected from the list shown in the overview; creating a reply to a specific article; and creating a new article to be submitted to the news group.
- submit a post (a new thread, or subject of discussion), or submitting a reply to an article (or to a reply within an existing thread).

This chapter reviews information in appropriate to each of the kinds of data Twister supports.

twister-specific data & handlers

Twister supplies newsgroup data using an action handler for each kind of news, or discussion, data. Each action handler:

- Looks at a user-defined, Twister-specific URL to learn details about the data to show (for example, show an overview that lists the first five articles).
- Supplies data appropriate to that handler—overview data, article data, etc.
- Interprets variables specific to the action handler, that have been embedded in the Twister files.
- Knows what to do with special Twister functions that display complex data, but require one function call with only a few Twister tags.

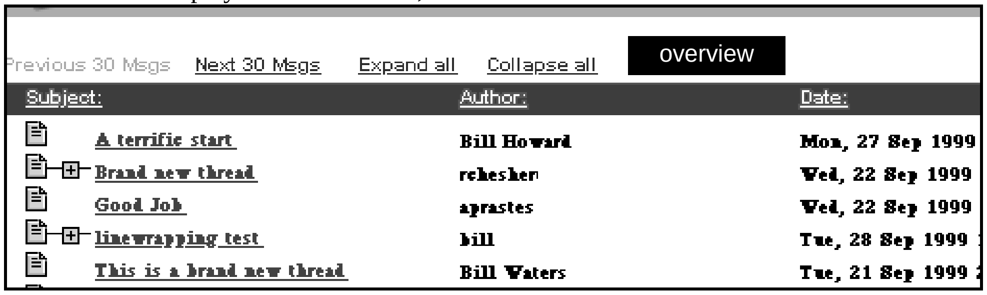
For example, to display five articles at a time in the overview section of the discussion, you:

- Define the twister variables, one for each column (such as subject, author, and date), and use the Twister Insert element to call them.
- Define a URL that specifies discussion group, the overview action-handler, and the number of articles to display at a time. The URL specifies other information, as well; this is described fully in chapter 4: reference, page 23.
- Call a special Twister function that fills multiple lines with a single definition of the elements in one line.

So, to display five articles at a time in the overview, you only have to specify the three variables for the kind of data to display for each article, the Insert element to insert these variables, and one Twister function; with this, the user can scroll through entire discussion lists, for example 10 or 30 items at a time.

what the overview looks like

The overview displays a list of articles, like those shown below:

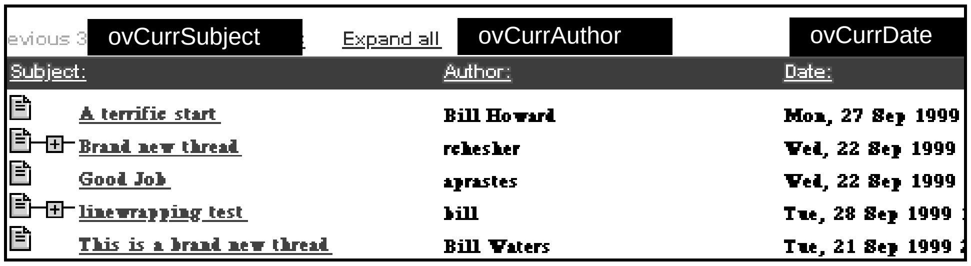


Subject:	Author:	Date:
A terrific start	Bill Howard	Mon, 27 Sep 1999
Brand new thread	rcheshen	Wed, 22 Sep 1999
Good Job	aprastes	Wed, 22 Sep 1999
linewrapping test	bill	Tue, 28 Sep 1999
This is a brand new thread	Bill Waters	Tue, 21 Sep 1999

twister overview

example: overview variables

To tell Twister what data to display for each article, use these variables in the Web page's HTML:



Subject:	Author:	Date:
A terrific start	Bill Howard	Mon, 27 Sep 1999
Brand new thread	rcheshen	Wed, 22 Sep 1999
Good Job	aprastes	Wed, 22 Sep 1999
linewrapping test	bill	Tue, 28 Sep 1999
This is a brand new thread	Bill Waters	Tue, 21 Sep 1999

twister overview variables

twister-enhanced URLs

Twister uses an enhanced URL for each type of discussion data shown on the discussion Web pages (one for each type of discussion, as mentioned earlier: overview, article, submit). Each URL requires information describing the data it needs to present.

In the source files, you can define a shorthand to refer to elements of the URL. These are then reviewed each time Twister renders a page (that is, for each user action). The data from the current URL is pulled into these variables, with the URL changing as necessary, sometimes subtly, to reflect user action.

The overview URL tells Twister the discussion group, the action handler, the number of articles to display, and how many to display. The following shows an example and description of an overview URL.

```
http://svr.com/group/comp.lang.tcl/yourtemplate.tpt/
@overview@first@F@10@S@all
```

where:

- `svr.com/group/comp.lang.tcl` is the fully specified path to the Usenet discussion group, available on the Twister server.
- `yourtemplate.tpt` is the compiled HTML/XML template to use to display data returned by Twister; this is found in the svr.com template directory, defined in Twister's feeds.conf file.
- `overview` tells Twister the kind of data to supply—that is, the action handler Twister is to use.
- `first` is the start of the range; you can also select last, or a specific article number.
- F displays articles forward, from this item to items received subsequently; you can also select R, for reverse.
- `10` defines the number of items to display; for example, this displays ten items at a time.
- S sort the displayed items by subject
- `all` expands posts that have replies associated with them; all

example twister-URL

Twister URLs are reviewed, in detail, in chapter 4: reference, page 23.

twister functions

Newsgroup data provides specific kinds of data with specific requirements in managing and displaying it. For example, you need a list of articles, a way to expand and collapse the list, a way to reply to the data, and a way to thread the data. Twister functions simplify these tasks.

For example, the Overview screen, illustrated on page 10, shows a list of five articles. Twister uses a special function, the overview function, to let you use Twister variables to define the format of only one line of the overview display; the overview function then knows to display one line for each article header retrieved. Functions are reviewed in more detail in chapter 4: reference.

xml and tpt files

Twister tags are, in fact, simple XML, where each tag is defined to handle a specific kind of data. The difference between HTML and Twister tags is the XML requires that the Web page be "well-formed;" that is, every tag must have its own close tag. Technically, HTML follows these rules. Realistically, however, Web page developers typically don't write clean HTML, because well formed pages are not strictly required—browsers are clever enough to figure out what to do with HTML, even if every start tag does not have a corresponding close tag.

For example, unordered lists start with `<UL>` and close with `</UL>`. However, <P> typically doesn't close with a slashed version of the tag </P> (although, technically speaking, it should).

To convert your HTML page to "well-formed" XML, use your standard method of developing your Web page HTML. Use a text editor or XML editor to insert Twister tags. At this point, you rename the file so that it has a .cf extension, then run the .cf file through the appropriate Twister program. One program generates "well-formed" XML, another generates both *.xml and *.tpt files, and another runs on *.xml files to generate *.tpt.

Twister uses compiled version of .xml files. These compiled files are named using the extension *.tpt. To convert data to .xml format, you can use the appropriate program or programs, listed here:

program name	use	example
cf2tpt	run on *.cf file to create both *.xml and *.tpt files Note that this program is the one typically used, because it creates all the files Twister may need.	cf2tpt overview.cf creates: • overview.xml • overview.tpt
escapeHTML	run on *.cf file to create *.xml file Note that this program is occasionally used to convert imported files to *.xml format.	escapeHTML ovTheme.cf creates: • ovTheme.xml
makeTPT	run on *.xml file to create *.tpt file	makeTPT overview.xml creates: • overview.tpt (cf2tpt uses makeTPT as an intermediate step)

file structure/elements

Edit Twister .cf files containing HTML tagging by inserting the following, as appropriate or required:

component	description	required
xml introductory lines	provide required XML data **example**: <?xml version="1.0" encoding="US-ASCII"?> <!DOCTYPE Template SYSTEM "template.dtd">	yes
xml <Page> </Page> tags	surround page	no
xml <Template> </Template> tags	surround document, following introductory xml lines	yes
insert tag	use to specify variables to display	no
constant tag	use to define values that don't change	no
import tag	inserts whole file; useful to separate look/feel from function	no, but very useful
Twister variables	use to specify data Twister is to retrieve; use with insert or constant tag	no
user-defined variables	define before body of text	no
preliminary definitions of constants, variables, and imported files	inserted below introductory <Template> tag and before <Page> <html> tags	no, but recommended

process: creating web-based discussion

The following summarizes the steps to follow in creating discussion pages:

1 Select the discussion group or groups to put on your Web site, then make sure the Twister server carries these groups. Refer to the Twister installation and configuration documentation for more information.

2 Create a link on an introductory page that sends the user to the Twister discussion overview URL. For example:

```
...
<TD><font face="Arial,Tahoma,Verdana" size=-1> Welcome to the bCandid Twister Demo Site.
Please try one of these (usenet) forums:<P>
</P>
<A href="/group/twister.dicuss/reader.tpt/@overview@first@F@30@S@all">twister.discuss</A>
<A href="/group/comp.lang.tcl/reader.tpt/@overview@first@F@30@S@all">comp.lang.tcl</A>
```

3 Create a Web page using your favorite Web page editor.
4 Make a copy of the page, renaming it so that it ends in .cf.
5 Edit the file using a text editor; include the required Twister lines and tags. Define any user-defined variables in the top portion of the page, after the intro lines and before the standard HTML encoding. Import any file, such as the theme file, that the discussion page uses.
6 On the command line, run:
cf2tpt yourfile.tpt
This creates both .xml and .tpt versions of the file.
7 Put the compiled *.tpt file into the proper directory, defined in the feeds.conf file
8 Run a browser and point at the *.tpt page.

Tip: This process is reviewed in chapter 3: walk through.

process: running twister

The following summarizes Twister processes, as it displays data using templates and interprets user actions.

Twister:
1 Accepts the connection.
2 Reads the URL. The following is an example URL:
/svr.com/group/groupname/tmpl.tpt/@overview@first@F@10@S@all
3 Sets any data that is not action-specific, putting it into the template context, which is temporary storage. In this example, /svr.com/group/groupname are not specific to the overview action.
4 Separates remaining, action-specific URL data into individual actions, where action may be preceded with an @ sign.
  - For example, /tmpl.tpt/@overview@first@F@10@S@all is split into template file to use, overview (the action handler), first (article to retrieve), forward (next set of items retrieved), 10 (number of items), subject (the way in which the data is sorted), all or none (all is expanded, none is collapsed).
5 The action-handler retrieves that data putting the data into the template context, too. At this point, the data to display has been retrieved.
6 Twister now loads the compiled template into memory and renders it, displaying it filled with data from the template context storage. In addition, Twister has the data it needs to run functions; the output from those functions is displayed, as well.
7 With every user action, Twister re-renders the page, using and updating the URLs as appropriate for the action. For example, if the user clicks on the link that expands a thread, the URL then matches the overview URL, except that it also displays the article number:
/svr.com/group/groupname/tmpl.tpt/@overview@first@F@10@S@4
If, for example, the user then collapses the thread, the URL is again changed:
/svr.com/group/groupname/tmpl.tpt/@overview@first@F@10@S@4-4

3: walk through

Twister displays discussion using standard Web browsers, rather than browsers that support XML or Usenet news. This requires that the Web page supply Twister with a description of the data the Web page needs (variables using Twister elements), with well-formed XML files (created using cf2tpt), the *.tpt compiled version of the xml file, and with some files in the correct location (template-dir on the Twister virtual server). Although each part of this is straightforward, a walk through Twister-discussion Web page creation clarifies how to use Twister in setting up Web-based discussion.

getting started

The Twister-usable file that results from this walk through is a simple HTML layout that does not use frames.

To start:

1. Make sure Twister is already installed, and the newsgroup that is to display through this Web page is installed on the Twister server.
2. Create a layout in HTML using your favorite tools.
3. Leave spots for the Twister data. The code fragments below show some text that serves as Twister place holders. For example, you could use the following HTML as a table for a list of articles, for the overview portion of the discussion page:

```
<HTML>
<H1> Welcome to <B>groupName</B></H1>
<Table>

<TR><TD><B>Subject</B></TD>  <TD><B>Author</B></TD> </TR>
<TR><TD>Test msg1</TD>       <TD>author1</TD> </TR>
<TR><TD>Test msg2</TD>       <TD>author2</TD> </TR>

</Table>
</HTML>
```

4. Copy your HTML into a .cf file—for example, if the file is

    my_overview.html

simply make a copy, then name it:

    my_overview.cf

5 Start a text editor or XML editor and bring up your .cf file.

6 Insert the lines shown below:

```
<?xml version=”1.0” encoding=”US-ASCII”?>
<!DOCTYPE Template SYSTEM “template.dtd”>
<Template>
<Page>
  <HTML>
  <H1> Welcome to <B>groupName</B></H1>
  <Table>

  <TR><TD><B>Subject</B></TD>  <TD><B>Author</B></TD> </TR>
  <TR><TD>Test msg1</TD>       <TD>author1</TD> </TR>
  <TR><TD>Test msg2</TD>       <TD>author2</TD> </TR>

  </Table>
  </HTML>
</Page>
</Template>
```

7 Save this. Every Twister-usable file has these lines. (This is probably one of the only times you need to enter this text; since it’s used in every Twister file, you will most like cut and paste this text.)

cleaning up the .cf file

You’ve now created the first portion of the Twister file. As mentioned in an earlier chapter, Twister requires “well-formed” XML. To create well-formed XML without hand-coding a lot of HTML, simply run the .cf file through the Twister program cf2tpt. This creates an .xml and a .tpt file.

1 Make sure cf2tpt is in your path, then on a command line, generate .xml and .tpt files by typing:

```
cf2tpt my_overview.cf
```

This creates the files:

```
my_overview.xml
my_overview.tpt
```

2 Put the resulting .tpt file in the Twister template directory—Twister uses compiled *.tpt files as templates that identify the data to populate the page. The template directory resides on your company’s Twister server, in the template directory as defined in

Twister's feeds.conf file. Twister recognizes this location as a URL; note that the examples throughout this chapter refer to this path as follows.

http://svr.com/template/my_overview.tpt/

> **Note: Twister is case-sensitive**, so make sure the URL reflects the combination of upper and lower case in the template file name (and, if used, in the relative path to the template file). In addition, make sure the file name listed in the URL matches exactly the case and spelling of the .tpt file.

3 Access the URL with any Web browser to display the page layout

Okay, so it's not very interesting, because it has no Twister tags, and therefore no dynamic content from Twister.

getting content into the page

To get content from Twister, you need to let it know what group and what articles from that group the page needs. As reviewed earlier, this is handled through a Twister-enhanced URL. The template file does not tell Twister what the display needs. To supply data to the page:

1 Figure out how you want the data to display, then edit the URL. In this example, the URL uses the following values:

- group to look at—comp.lang.tcl
- the template to use—mytemplate.tpt
- the action-handler to use—overview
- the article in that group to start with—the first one
- which direction to go from the starting article—forward (F), rather than reverse (R)
- how many to display at once—count of 10
- how to sort those items—by subject
- threads to expand —all

The Twister-enhanced URL is now:

http://svr.com/group/comp.lang.tcl/my_overview.tpt/@overview@first@F@10@S@all

> **Note: You haven't touched the file—just the URL that accesses the page.**

2 Access the URL with any Web browser to display the page layout.

Well, it's still not that interesting, but at least Twister is ready to give you the data if you ask for it properly—as shown in the next step.

3 Edit the .cf file with a text editor to tell Twister to supply the name of the discussion group—comp.lang.tcl.

Change:

```
<H1> Welcome to <B>groupName</B></H1>
```

To:

```
<H1> Welcome to<B><Insert var=”groupName”/></B></H1>
```

Now, the .cf file looks like this:

```
<?xml version=”1.0” encoding=”US-ASCII”?>
<!DOCTYPE Template SYSTEM “template.dtd”>
<Template>
<Page>
  <HTML>
  <H1> Welcome to <B><Insert var=”groupName”/></B></H1>

  <Table>
  <TR><TD><B>Subject</B></TD>  <TD><B>Author</B></TD> </TR>
  <TR><TD>Test msg1</TD>      <TD>author1</TD> </TR>
  <TR><TD>Test msg2</TD>      <TD>author2</TD> </TR>

  </Table>
  </HTML>
</Page>
</Template>
```

4 Run the .cf file through cf2tpt to generate the .xml and .tpt files

```
cf2tpt my_overview.cf
```

5 Put the resulting .tpt file in the Twister template directory.

6 Check out the latest .tpt file in your Web browser via the URL:

http://svr.com/group/comp.lang.tcl/my_overview.tpt/@overview@first@F@10@S@all

Now it says

Welcome to comp.lang.tcl.

(Of course, this works only if comp.lang.tcl is available on the Twister server.)

But this still doesn’t display article data....

add the overview function: see real data!

Now, some more complicated HTML/XML:

1 Edit the .cf file with a text editor to tell Twister to supply real article data from comp.lang.tcl. To do this, change:

```
<TR><TD>Test msg1</TD>      <TD>author1</TD> </TR>
<TR><TD>Test msg2</TD>      <TD>author2</TD> </TR>
```

To:

```
<Function name=”overview”>
<Param name=”lineFormat”>
       <TR>
       <TD><Insert var=”ovCurrSubject”/></TD>
       <TD><Insert var=”ovCurrAuthor”/></TD>
       </TR>
</Param>
</Function>
```

The .cf file now looks like this:

```
<?xml version=”1.0” encoding=”US-ASCII”?>
<!DOCTYPE Template SYSTEM “template.dtd”>
<Template>
      <Page>
            <HTML>
                  <H1> Welcome to <B><Insert var=”groupName”/></B></H1>
                        <Table>
                  <Function name=”overview”>
                        <Param name=”lineFormat”>
                        <TR>
                              <TD><Insert var=”ovCurrSubject”/>
                              </TD><TD><Insert var=”ovCurrAuthor”/></TD>
                        </TR>
                        </Param>
                  </Function>

                        </Table>
            </HTML>
      </Page>
</Template>
```

2 Run the .cf file through cf2tpt to generate the my_overview.tpt and *.xml file:

```
~/cf2tpt my_overview.cf
```

3 Put the resulting .tpt file in the Twister template directory.

4 Check out the latest .tpt file in your Web browser via the URL
http://svr.com/group/comp.lang.tcl/my_overview.tpt/
@overview@first@F@10@S@all

You now display real messages.

want to display an article?

This Web page is about ready to display article text. To do that:

- create an article template file—Twister needs something to hand the data to.
- invest in tidying up the line format; then you can make a single change that pulls the article text into the article template.

make an article template

To display the article using Twister, Twister needs a template. The following creates a simple article template:

1 Use the text editor to create an article template, in this example, my_article.cf. The following can be used as a simple article template:

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE Template SYSTEM "template.dtd">

<Template>
    <Page>
        <HTML>
            <B>Subject:</B><Insert var="artSubject"/><P>
            <Insert var="artBody"/>
        </HTML>
    </Page>
</Template>
```

2 Run the .cf file through cf2tpt to generate my_article.tpt.tpt and *.xml files:

```
~/cf2tpt my_article.cf
```

3 Put the resulting .xml and .tpt files in the Twister template directory.

4 If you want, you can use a Web browser to check out the article template, but it's really pretty dull right now.

make your own line format

To wire together the subject with an article, first simplify the lineFormat.

Tip: (Really, do it now. You'll be glad you did.)

To do this:

1 Once again, edit the overview.cf file. This time, insert the following after the <Template> tag:

```
<Variable name="myLineFormat"><TR><TD><Insert
var="ovCurrSubject"/></TD><TD><Insert var="ovCurrAuthor"/></TD>
</TR></Variable>
```

2 Also, change:

```
<Param name=”lineFormat”><TR><TD><Insert
var=”ovCurrSubject”/></TD>
<TD><Insert var=”ovCurrAuthor”/></TD>
</TR></Param>
```

to

```
<Param name=”lineFormat”><Insert var=”myLineFormat”/></Param>
```

The .cf file now reads:

```
<?xml version=”1.0” encoding=”US-ASCII”?>
<!DOCTYPE Template SYSTEM “template.dtd”>
<Template>

    <Variable name=”myLineFormat”><TR><TD><Insert
    var=”ovCurrSubject”/></TD><TD><Insert var=”ovCurrAuthor”/></TD></TR></Variable>

    <Page>
        <HTML>
            <H1> Welcome to <B><Insert var=”groupName”/></B></H1>

            <Table>
                <TR><TD><B>Subject</B></TD>  <TD><B>Author</B></TD> </TR>

                <Function name=”overview”>
                    <Param name=”lineFormat”>
                        <Insert var=”myLineFormat”/>
                    </Param>
                </Function>
            </Table>
        </HTML>
</Page>
</Template>
```

Now that you’ve tidied up the HTML, link the subject field to its article. Each line in the subject column of the table corresponds to an article. Display the article using the article template you just created:

/group/groupname/templatename/@article@artNum

3 Continuing with the text editor, change myLineFormat to display an article:

```
<Variable name=”myLineFormat”>
    <TR>
        <TD>
        <A href=”/group/<Insert var=”groupName”/>/my_article.tpt/@article
        @<Insert var=”ovCurrArtNum”/>”><Insert var=”ovCurrSubject”/></A>
        </TD>
        <TD><Insert var=”ovCurrAuthor”/>
        </TD>
    </TR>
</Variable>
```

The subjects are now hyperlinks.

4 Run the .cf file through cf2tpt to generate the my_overview.tpt and *.xml files:

```
~/cf2tpt my_overview.cf
```

5 Put the resulting files in the Twister template directory.

6 Use the Web browser to view the URL
http://server.company.com/group/comp.lang.tcl/my_overview.tpt/
@overview@first@F@10@S@all

7 Click on an article listed in the overview, and the text displays using the URL that is something like:
/group/comp.lang.tcl/articletemplate.tpt/@article@1
and the subject and text of the article displays in the newly created template!

You’ve now created two Twister templates. Refer to the sample templates and other sample files in the Twister templates directory to check out the post and submit templates.

4: reference

This chapter provides more technical detail about Twister:

Tip: Remember that Twister is case-sensitive. Use proper initial capitalization when creating Twister variables, constants, and importing files.

elements

Insert

Use the Twister Insert element to include Twister variables. The overview, article, and submit variables are defined later in this chapter. The format of this element is:

```
<Insert var=”ovCurrSubject”/>
```

where var is the Twister variable; in this case, the var is set to an overview variable ovCurrSubject. Note that Insert does not require a closing /Insert tag; instead, this element has been defined so that it requires only a final / preceding the close angle bracket >. You can use Insert in the body of the HTML and in the header section of the .cf file, as illustrated on page 27.

Constant

Use the Twister Constant element to include values that need to be set only once. This simplifies calling these values. The format of the Constant element is:

```
<Constant name="selectedText"><img src="/selected.gif" width="16" height="13" border="" alt="">
</Constant>
```

where selectedText is text assigned to an image, selected.gif. Constant requires a closing tag, </Constant>, as shown in the example.

Import

Use the Twister Import element to import xml files; note that Import does not import *.tpt files. Optionally, use Import to separate commonly used functions in one file, and define the look of the Web page in another file, a template *.tpt file. By dividing look from functionality, you can simplify maintenance of the data, and more easily create different looks for discussion pages while retaining identical functionality. The format of the Import element is:

```
<Import file="readerTheme.xml"/>
```

where the Import element takes file as an argument, along with an XML file (not a *.tpt file). When the .cf file is compiled, Twister imports the entire contents of the .xml file, to make it available in the compiled *.tpt file. Note that when the contents of the Import element are used anywhere throughout the page, you must define the Import element in the Twister template file's header section of the .cf file, as illustrated on page 27.

Twister processes data in the order listed in a file. For example, if you put an Import element in the body of the file, the data defined in that Import element is available only to tags below the Import element.

Template

Use the Template element before the Page and HTML elements. Files with this tag call a template into which Twister inserts data. The use of this element is shown on page 27.

Variable

The Variable element defines a value that can be reset multiple times in a single page. The lineFormat entry in the table on page 30 uses the Variable element.

Param

The Param element defines data handed to a function. Review the example in IFEqual function on page 25 to see how Param is used.

Page

Use the Page element to defines the HTML portion of a Twister template page. The use of this element is shown on page 27.

Function

Use the Function tag preceding a Twister function. Functions are defined in more detail below.

functions

Twister provides functions to handle instances that regularly occur in Twister-enhanced Web pages. Functions can get data from Twister's Param elements.These functions include:

- URLEscape
- IFEqual
- quoteBody
- Overview
- getHeader

Tip: To review the use of some of these functions, refer to the tagging examples in displaying threaded list of articles on page 28.

URLEscape function

Use the URLEscape function to handle special (non-alphanumeric) characters. For example, if the Web page needs to display "a b", then use URLEscape to encode it as "a%20b" so that the text displays properly.

```
URLEscape
   <Function name="URLEncode">
      <Param name="data">something with non-alphanumeric (://?%) characters in it </Param>
   </Function>
```

quoteBody function

Use the quoteBody function to insert text, quoted from the original, in a reply. To review an example using quoteBody, refer to page 59.

IFEqual function

Use this to compare a variable to a value. Using an if/else construct, define appropriate actions to take (typically, one action if the two are equal, another action if they are no equal). The following example compares the value of the Twister variable ovCurrArtNum and artNum. If they are the same, then Twister puts an image (such as an arrow) next to the item; otherwise, Twister puts a different image (such as a blank gif, that indents the item the same amount as the arrow).

```
<Variable name="selPtr">
        <Function name="ifEqual">
                <Param name="a"><Insert var="ovCurrArtNum"/></Param>
                <Param name="b"><Insert var="artNum"/></Param>
                <Param name="ifEq"><Insert var="selectedImage"/></Param>
                <Param name="ifNotEq"><Insert var="unselectedImage"/></Param>
        </Function>
</Variable>
```

Overview function

Use the Overview function to iteratively gather article data to display in the overview template. For example, by calling the Overview function, you can display ten items in the overview portion of the Web page, by calling all the data for only one item. The overview function calculates how many items to display, using the overview URL, and retrieves and displays the appropriate data. Review displaying threaded list of articles on page 28 for examples of coding using the Overview function.

For example, in the following simple overview template HTML code, each overview line contains the subject and author of each article. This is defined once; then the overview function displays the line as many times as directed in the overview URL.

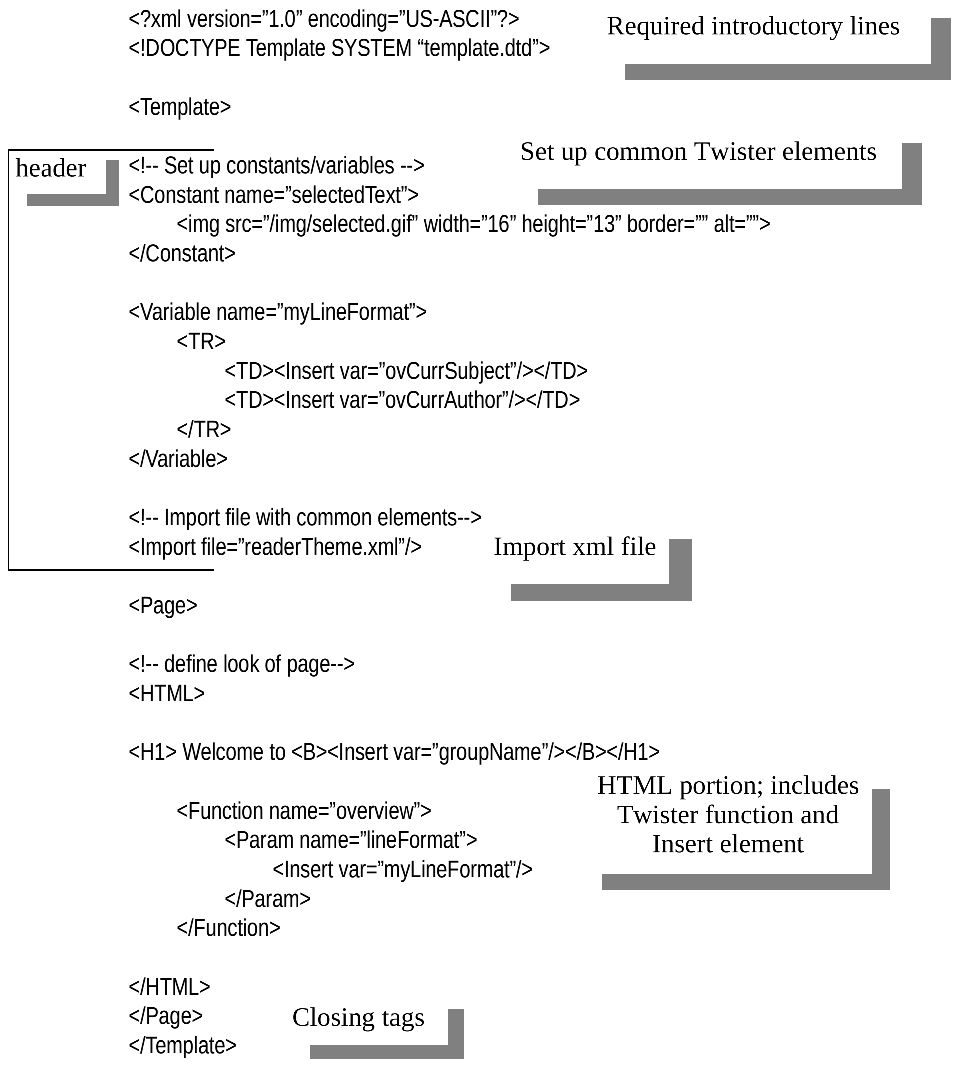
getHeader function

Use the getHeader function to retrieve article header from an article context. For example, to retrieve a UserID header:

```
<Function name=”getHeader”> <Param name=”name”> UserID </Param> </Function>
```

file structure

Twister relies on a known file structure in part to meet XML requirements. The following summarizes Twister file structure:



```
<?xml version=”1.0” encoding=”US-ASCII”?>
<!DOCTYPE Template SYSTEM “template.dtd”>

<Template>

<!-- Set up constants/variables -->
<Constant name=”selectedText”>
        <img src=”/img/selected.gif” width=”16” height=”13” border=”” alt=””>
</Constant>

<Variable name=”myLineFormat”>
        <TR>
                <TD><Insert var=”ovCurrSubject”/></TD>
                <TD><Insert var=”ovCurrAuthor”/></TD>
        </TR>
</Variable>

<!-- Import file with common elements-->
<Import file=”readerTheme.xml”/>

<Page>

<!-- define look of page-->
<HTML>

<H1> Welcome to <B><Insert var=”groupName”/></B></H1>

        <Function name=”overview”>
                <Param name=”lineFormat”>
                        <Insert var=”myLineFormat”/>
                </Param>
        </Function>

</HTML>
</Page>
</Template>
```

Required introductory lines
header
Set up common Twister elements
Import xml file
HTML portion; includes Twister function and Insert element
Closing tags

displaying threaded list of articles

This section reviews one way to display news in an overview format. To display a list of news articles, indent them properly, and thread the discussion so that articles and replies are properly associated, Twister exercises:

- overview URL
- overview variables
- user-defined lineFormat variable
- Overview and IFEqual functions.

overview URL

The overview URL defines a lot of details about how the specific articles are to display and how many, including:

- start—the number of the article that starts the range to be displayed
- direction—F (forward) or R (reverse); F is valid for the value first and for a specific article number; R is valid for the value last for a specific article number.
- count—the total number of articles displayed

The following is an example overview URL:
http://server.company.com/group/comp.lang.tcl/tmpl.tpt/@overview@first@F@10@T,S@all

Twister has to track more than simply a straight sequence of numbers, because some article numbers may have expired. Twister tracks this information, along with other data, to display articles properly.

using overview variables

To determine the number of articles actually displayed, Twister uses these variables.

example 1. Assume Twister returns articles numbered:

- 1,2,3,7,8,9

The overview URL asks for 1,F,5:

- articles 1, 2, 3, 7, 8 display

Overview tags, defined in the template, then have these values:

- ovRequestedCount = 5
- ovActualCount = 5
- ovScreenHi = 8
- ovScreenLo = 1
- ovScreenHi+1 = 9
- ovScreenLo-1 = 0 (none)
- ovRangeHi = 8
- ovRangeLo = 1

example2. For example, assume Twister returns articles numbered:
- 1,2,3,7,8,9

The overview URL asks for 9,F,5:
- article 9 displays

Overview tags then have these values:
- ovRequestedCount = 5
- ovActualCount = 1
- ovScreenHi = 9
- ovScreenLo = 9
- ovScreenHi+1 = 0 (none)
- ovScreenLo-1 = 8
- ovRangeHi = 8
- ovRangeLo = 1

example3. For example, assume Twister returns articles numbered:
- 1,2,3,7,8,9

The overview URL asks for 2,F,2:
- articles 2 and 3 display

Overview tags then have these values:
- ovRequestedCount = 2
- ovActualCount = 2
- ovScreenHi = 3
- ovScreenLo = 2
- ovScreenHi+1 = 7 (none)
- ovScreenLo-1 = 1
- ovRangeHi = 8
- ovRangeLo = 1

simple example

In the following simple code fragment, each overview line contains the subject and author of each article. This is defined once; then the overview function displays the line as many times as directed by the overview URL.

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE Template SYSTEM "template.dtd">
<Template>
    <Variable name="myLineFormat">
        <TR>
        <TD><Insert var="currOvSubject"/></TD>
        <TD><Insert var="currOvAuthor"/></TD>
        </TR>
    </Variable>
```

```
<Page>
<HTML>
        <H1> Welcome to <B><Insert var=”groupName”/></B></H1>
        <Table>
                <TR><TD><B>Subject</B></TD>  <TD><B>Author</B></TD> </TR>
                <Function name=”overview”>
                        <Param name=”lineFormat”><Insert var=”myLineFormat”/></Param>
                </Function>
        </Table>
</HTML>
</Page>
</Template>
```

displaying images next to article list

The Overview function shown next uses these parameters, that in this case are defined as variables on the page that uses them, between the Template and Page tags:

parameter	definition	example
indentSpacer	how far over to indent text to permit use of pointer or other icon	<Constant name="myIndentSpacer"><img src="/img/p.gif" HEIGHT="13" WIDTH="15"></Constant>
replyMarker	the icon to indicate a reply to the thread listed above	<Constant name="myReplyMarker"><img src="/img/reply-15x13.gif"> </Constant>
newThreadMarker	the icon to indicate a new thread	<Constant name="myNewThreadMarker"> <img src="/img/active-15x13.gif" HEIGHT="13" WIDTH = "15"> </Constant>
lineFormat	the data displayed in the overview line	<Variable name=”myLineFormat”> <TR><TD> <Insert var=”currOvSubject”/></TD><TD><Insert var=”currOvAuthor”/> </TD></TR></Variable>

These variables are called as follows:

```
<Function name=”overview”>
        <Param name=”indentSpacer”><Insert var=”myIndentSpacer”/></Param>
        <Param name=”replyMarker”><Insert var=”myReplyMarker”/></Param>
        <Param name=”newThreadMarker”><Insert var=”myNewThreadMarker”/></Param>
        <Param name=”lineFormat”><TR><Insert var=”myOverviewLine”/></TR></Param>
</Function>
```

overview and IFEqual functions

The following example code shows the use of spacers and thread markers with four messages:

```
A
RE: A
RE: RE: A
RE: A
```

The following shows elements used as spacers and markers, and a user variable lineFormat that contains the ovCurrLeftMarkers Twister variable:

```
indentSpacer = "---"
replyMarker = "-->";
newThreadMarker = "__]";
lineFormat = "<Insert ovCurrLeftMarkers> <Insert subject>"
```

Twister sorts by subject, as instructed by the overview URL, simultaneously figuring out indentation. In this example, Twister uses the IFEqual function to insert the proper images, along with the indent levels.

The following IFEqual example summarizes that coding:

```
IFEqual (indent = 0) then ovCurrLeftMarkers = newThreadMarker
else
    (x = indent) then
    ovCurrLeftMarkers = indentSpacer (repeated X times) + replyMarker
```

The results of these Twister variables and functions are shown here:

indents	message	indentation looks like this:	graphics look like this:
0	A	A	__] A
1	RE: A	RE: A	-----> RE: A
2	RE: RE: A	RE: A	--------> RE: A
1	RE: A	RE: A	-----> RE: A

showing the selected item

The elements used as spacers and markers, along with a *** to indicated the selected article using a line format called ovCurrLeftMarkers, follow. If the user has moved the cursor to article 200, Twister indents, places images, and inserts a pointer (in this case, ***) adjacent to article 200:

article #	indents	message	indentation looks like this:	graphics look like this:
100	0	A	A	__] A
200	1	RE: A	RE: A	*** -----> RE: A
300	2	RE: RE: A	RE: A	--------> RE: A
400	1	RE: A	RE: A	-----> RE: A

Tip: For examples showing the use of selPtr, review the templates and examples included with Twister.

common URL components

All URLs contain some or all of these elements:

URL portion	description	legal values
`svr.com`	defines the server on which Twister is running	Twister server name on which data and templates reside.
`/group/<“groupname”>`	complete path and Usenet discussion group name	Fully specified path to a group; make sure it resides on the Twister server.
`/<”relative_path/ tmpl.tpt”>`	the xml file you created, that displays data returned by Twister; may include relative path to the template subdirectory, on the server; the template directory is defined in feeds.conf.	The exact name of the xml file you created, along with the relative path to it. This xml file must reside in the Twister server’s template directory or in a subdirectory located in the template directory. Note that Twister is case-sensitive.

overview URL & variables

overview URL

http://<“svr.com”>/group/<“groupname”>/<”relative_path/tmpl.tpt”>/@overview@[first | last | “number”] @[F | R]@<“count”>@[T,] A [+|-] | B [+|-] | D [+|-] | F [+|-] | L [+|-] | S [+|-] [, A [+|-] | B [+|-] | D [+|-] | F [+|-] | L [+|-] | S [+|-] @[all | none]

Each part of this URL is described in the following table:

URL portion	description	legal values
`overview`	the kind of data Twister is to supply—that is, the action handler Twister is to use.	overview
`[first | last | number]`	the article at the beginning of the range; select first, last, or a specific article number.	select one of these: • first • last • “number”— supply an integer of the article to start with.
`[F | R]`	whether to display articles forward or reverse, as compared to the value set as the beginning of the range.	select one, depending in part on the value set in first/last/number: • F—use with first, or with number • R—use with last, or with number
`“count”`	the number of items to display; for example, enter 10 to display ten items at a time.	an integer

URL portion	description	legal values

`[T, ]`
 	
`A[+-] |`
`B[+-] |`
`D[+-] |`
`F[+-] |`
`L[+-] |`
`S[+-]`
 	
[, `A[+-] |`
`B[+-] |`
`D[+-] |`
`F[+-] |`
`L[+-] |`
`S[+-] ]`
	Whether the items are threaded.
	
	Primary sort (what is sorted and in what direction); applies to entire list of articles. If threaded, applies to the first level (threads).
	
	If threaded, the secondary sort (what is sorted and in what direction) applies to each set of sub-articles below a top-level thread. If not threaded, any value supplied here is ignored.
		optionally set a value; if none is set, then threading is off and the articles are sorted by subject. The values are:
		THREADING:
		T to Thread, or nothing to disable threading; optional; default is no threading.
		
		PRIMARY SORT: by default, sorted by subject; otherwise, sorts according to option set, which is a letter as shown below, and optionally a direction (+ or -).
		
		SECONDARY SORT; only used if articles are threaded.
		
		ORDER OF SORT:
		+ represents forward sort; default used if no value supplied
		- represents backward sort
		
		VALUE SORTED ON:
		A: article number
		B: number of bytes
		D: date
		F: from
		L: number of lines
		S: subject; default used in no value specified.

`[all | none]`	how to display articles that have associated replies; whether to expand them (all) or collapse them (none).	select one: • all • none

Put the overview URL in the Web page displaying the option to go to this newsgroup; you may have several of these listed, if the Web site supports multiple discussion groups.

overview variables

overview variables	description	what's returned
templateName	overview	full path to template name, which ends in .tpt
groupName	discussion group name	Usenet discussion group
groupCount	total number of articles	integer
ovActualCount	number of articles returned/displayed	integer
groupHi	highest article number in the group	integer
groupLo	lowest article number in the group	integer
ovRangeHi	highest article number in requested range; independent of sort order	integer
ovRangeLo	lowest article number in requested range; independent of sort order	integer
ovScreenHi	the highest article number displayed	integer
ovScreenHi+1	the next article number to display	integer, or zero (0) for none
ovScreenLo	the lowest article number displayed	integer
ovScreenLo-1	the next article number to display, in descending order	integer, or zero (0) for none
ovStart	the beginning of the range of articles	first | last | "key-number"
ovDirection	direction in which to display messages after one is selected	"F" or "R"
ovReverseDirection	display in opposite direction	"R" or "F"
ovSortString	reports threading on/off; sort order of top-level articles; if threading on, also returns how sub-articles are sorted under each top-level item	*[threading on or off,] primary_sort [, secondary sort]* *for legal values, see* overview URL *on page 33*

overview variables	description	what's returned
ovFirstSortKey	primary sort key; how top level threads are sorted	A | B | D | F | S where A —sorted by article number B —sorted by number of bytes D —sorted by date F —sorted by from L —sorted by number of lines S —sorted by subject
ovSecondSortKey	secondary sort key, valid only if articles are threaded; how sub-articles beneath each top-level thread are sorted	
ovIsThreaded	whether threading is enabled	Y | N
ovFirstSortDirection	returns direction (forward or reverse) primary sort	+ | - where + is forward, - is reverse
ovSecondSortDirec-tion	returns direction (forward or reverse) secondary sort	+ | - where + is forward, - is reverse
overview (function)	reviews retrieved data, displays it using specific parameters (lineFormat—required; newThreadMarker, indentSpacer, replyMarker—optional)	lineFormat newThreadMarker indentSpacer replyMarker
ovCurrLeftMarkers	set by Overview function	
ovCurrSubject	subject of each item	alphanumeric characters
ovCurrAuthorName	author of each item	alphanumeric characters
ovCurrAuthorEmail	author's e-mail address, one per item	Internet-style e-mail address, using alphanumeric characters, @ (at), and . (dot)
ovCurrDateHHMM ovCurrDateMMDD ovCurrDateYYYY	date associated with each item	HHMM: HH is hours 0-23; MM is minutes 0-59 MMDD: MM is months 01-12;DD is days 01-31 YYYY: year 1999 (2000, etc.)
ovCurrMsgID	unique message ID associated with each item	standard NNTP/Usenet message ID format
ovCurrLines	number of lines of each item	integer
ovCurrBytes	number of bytes of each item	integer

overview variables	description	what's returned
ovCurrArtNum	article number associated with each item	integer
ovPrevArtNum	article number of the item preceding the selected item	integer
ovNextArtNum	article number of the item following the selected item	integer
ovExpandedNodes	enables expand and contract functionality for the threads and their replies	all or none
ovCurrExpanded	whether the current object has been expanded	Y | N
ovFetchedCount	the number of items retrieved, shown in the list of overview items	integer, or zero for none

article URL & variables

article URL

Article URLs may look like any one of the following:

URL	when displayed
http://<"svr.com">/group/<"groupname">/<"relative_path/tmpl.tpt">/@article@[number]	when article selected using multi-frame Web pages
http://<"svr.com">/group/<"groupname">/"<relative_path/tmpl.tpt">/@overview @.../article@<first | last | #>	from overview page

Each part of the article URLs is described in the following table:

URL portion	description	legal values
`overview` *or* `article`	the kind of data Twister is to supply—that is, the action-handler to call	overview and article, or simply article, depending on the action-handlers required to display the article
`number`	the number of the selected article; Twister uses this template to display the article	the number of the article Twister displays
`<first | last | #>`	the article that begins the range	first, last, or "number," where number represents the group's article number
`"msgID"`	the unique message ID assigned to this message	the article's message ID in standard NNTP/Usenet format

article variables

article variables	description	what's returned
artBody	the text of the article	alphanumeric characters
artCurrAttachName	the text of the current article and attachments	artBody and attachments
artSubject	the subject of the article	alphanumeric characters
artGroups	the group in which this article is listed	alphanumeric and special characters
artAuthorName	the author of this article	alphanumeric characters
artAuthorEmail	the email of the article's author	alphanumeric characters and special characters, such as @
artDate	complete date of the article	integers
artDateMMDD artDateHHMM artDateYYYY	date associated with each article	HHMM: HH is hours 0-23; MM is minutes 0-59 MMDD: MM is months 01-12;DD is days 01-31 YYYY: year 1999 (2000, etc.)
artMsgID	article's unique ID	standard NNTP/Usenet message ID format
artLines	number of lines in the article	integer
artBytes	length of the article in bytes	integer
artNum	number of the article	integer
artPrevArtNum	the number of the article that precedes the current article number	integer
artNextArtNum	the number of the article that follows this article	integer
artReferences	references supplied for the article	alphanumeric characters
artXRef	cross-references for cross-posted articles	cross reference using Internet conventions

submit action

Submit action (svr.com/action/submit) works specifically with forms. Submit gets the data from these form fields:

- from
- newsgroup
- subject
- body
- references

Twister then posts the message using this data.

submit message variable

You can insert a message variable to display data from Twister about the submission success (1) or failure (0). For example,

```
<Insert var=”message”\>
```

With this returned data, you can, for example, supply one message if the returned value is success, another if the returned value is failure.

twister DTD summary

The basic format of Twister’s document-type-definition is:

```
<Template>
      <Constant></Constant>
             ...more constants...
      <Variable></Variable>
             ...more variables...
      <Page>
            text
            <Function name="twister_if">
                  <Param name="a">aaaaaaa</Param>
                  ...more params...
            </Function>

            <Insert var="whatever"/>
      </Page>
</Template>
```

retrieving key value pairs

Twister can access HTTP URL key-value pairs using the HTTP standard ?key=value... format.

attachment decode

Twister displays any attachment associated with articles in a newsgroup using this decoder action handler. The form of the URL provided by Twister to display attachments is:

```
/action/decode/<“image.jpg”?group=compl.lang.tcl&artnum=<“int”>....
```

TZOffset

By default, Twister returns dates and times using GMT (Greenwich Mean Time). Use TZOffset to adjust the article’s date and time to a time zone other than GMT. If you supply the TZOffset with a value (plus or minus 24) to apply as an offset to GMT, the time is returned reflects the date and time using the offset. For example,

```
<Constant name=“TZOffset”>-5.5</Constant>
```

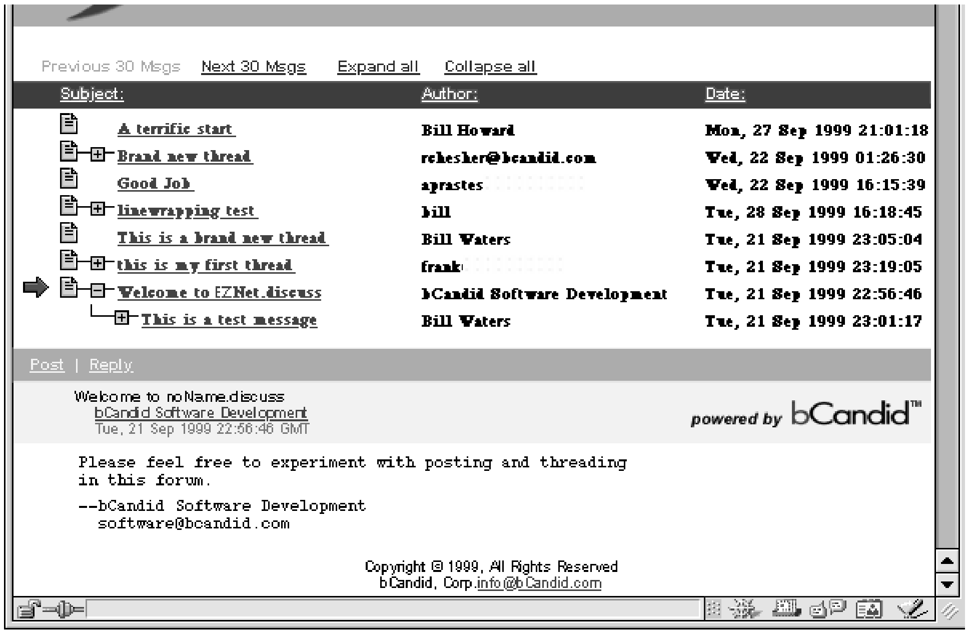
returns values adjusted for the time zone that is 5.5 hours west of Greenwich.

5: sample overview tagging

Twister uses a template file to determine the data to display, how to display it, the user actions that are possible, and how to interpret them. You can, optionally, define a separate file containing some common variables or functions. For example, set up the template file so that it handles the display of data (the “look”), and set up an imported “theme” file so that it handles functionality (the “feel”). Imported files may simplify Twister templates maintenance.

Typically, use the template files as is, so that subsequent Twister upgrades and templates can be readily installed. To customize your site, create and import your own theme files.

This chapter reviews some coding fragments that demonstrate one simply method Twister can use to display and thread discussion. These coding fragments are used to display a combination overview list of articles and the text of a selected article; this combined layout is referred to in this chapter as a reader layout. This layout is illustrated below:



Previous 30 Msgs   Next 30 Msgs   Expand all   Collapse all

Subject:	Author:	Date:
A terrific start	Bill Howard	Mon, 27 Sep 1999 21:01:18
⊞ Brand new thread	rchesher@bcandid.com	Wed, 22 Sep 1999 01:26:30
Good Job	aprastes	Wed, 22 Sep 1999 16:15:39
⊞ linewrapping test	bill	Tue, 28 Sep 1999 16:18:45
This is a brand new thread	Bill Waters	Tue, 21 Sep 1999 23:05:04
⊞ this is my first thread	frank	Tue, 21 Sep 1999 23:19:05
⊟ Welcome to EZNet.discuss	bCandid Software Development	Tue, 21 Sep 1999 22:56:46
└⊞ This is a test message	Bill Waters	Tue, 21 Sep 1999 23:01:17

Post | Reply

Welcome to noName.discuss
bCandid Software Development
Tue, 21 Sep 1999 22:56:46 GMT

powered by bCandid™

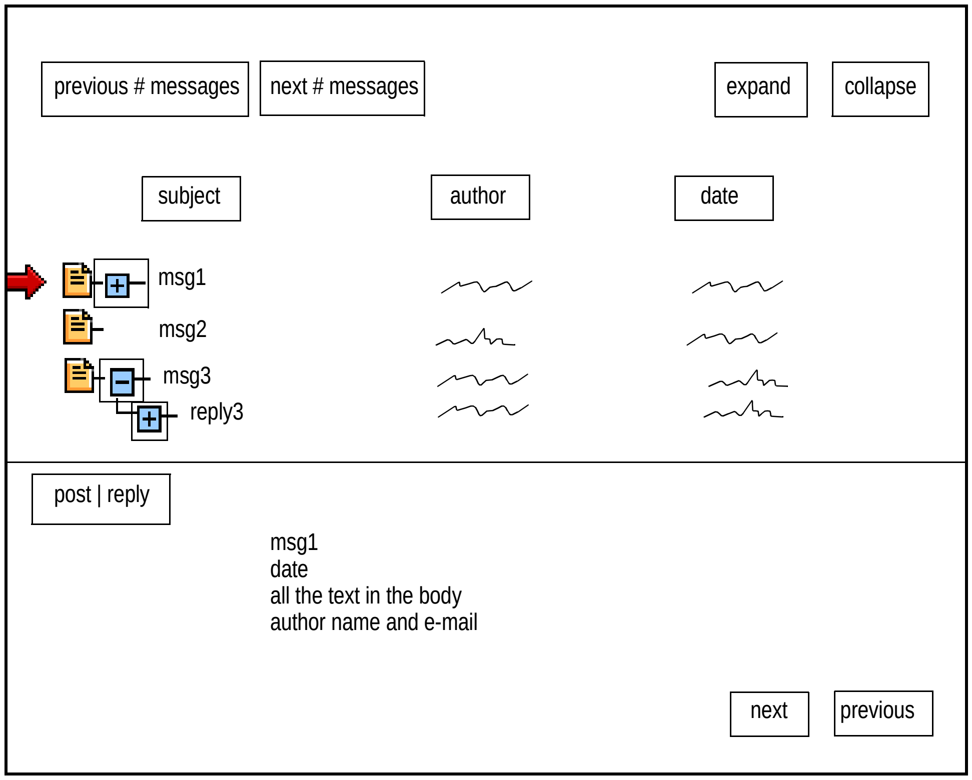
```
Please feel free to experiment with posting and threading
in this forum.
--bCandid Software Development
  software@bcandid.com
```

Copyright © 1999, All Rights Reserved
bCandid, Corp.info@bCandid.com

discussion reader: overview

The following diagram shows:

- three articles, one expanded to show one response
- a selected article, as indicated by the arrow
- the text of the selected article that displays in the bottom half of the window
- boxed text, that indicates buttons



The user can manipulate data using boxed buttons and other linked images and text. The buttons and boxed data on the top half of the illustration deal with the overview list; they include:

- a choice of how to sort the listed data—for example, by subject.
- next and previous list of discussion items to display in the top half of the illustration
- whether to expand or collapse the discussion items to show all items in each thread.

Graphics next to each item in the overview list indicate whether an item is a new thread, whether it has responses associated with it, and whether it corresponds to the text shown in the bottom half of the display. All of these actions are accomplished using Twister functions and variables.

defining some URLS

Twister refers to URLs to determine what to display. These can be abbreviated, using Twister's capacity to accept user-defined variables. The following are example URLs you can define, and that are used across many examples referred to in this chapter.

baseURL. shows the initial portion of the Twister overview URL.

```
<Constant name="baseURL">
    /group/<Insert var="groupName"/>/<Insert var="templateName"/>/@overview
</Constant>
```

currURLMinusSort. adds range data to the base URL; no sorting.

```
<Constant name="currURLMinusSort">
    <Insert var="baseURL"/>@<Insert var="ovRangeLo"/>@F@<Insert var="ovRequestedCount"/>@
</Constant>
```

currURLMinusExpanded. sorts, but no expand/collapse data in this URL fragment.

```
<Constant name="currOvURLMinusExpanded">
    <Insert var="currURLMinusSort"/><Insert var="ovSortString"/>@
</Constant>
```

currOvUR. URL including overview/expand/collapse.

```
    <Constant name="currOvURL"><Insert var="currOvURLMinusExpanded"/><Insert
var="ovExpandedNodes"/></Constant>
```

currOvURL. URL for the article that displays in a frameless "reader" file.

```
<!-- This is the URL of the current page for JUST the article part -->
    <Constant name="currArtURLFragment">/@article@<Insert var="artNum"/></Constant>
```

myURL. URL that currently displays.

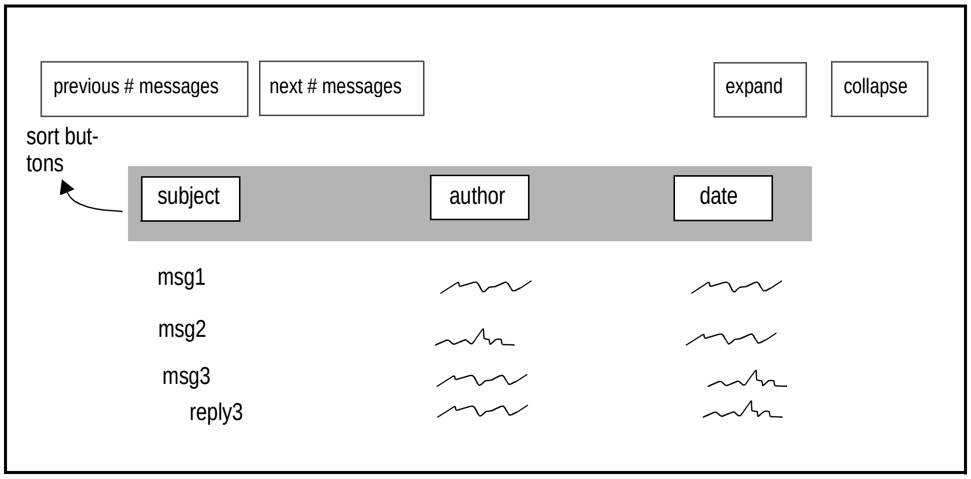
```
<!-- This is the URL we are at right now -->
<Constant name="myURL"><Insert var="currOvURL"/><Insert var="currArtURLFragment"/></Constant>
```

Tip: URLs are defined in the template libraries, in /templates/lib in the Twister directory. Two versions are available: one for framed interfaces, another for non-framed interfaces.

sorting articles

the look

The following defines the look of the “sort” buttons, which is what the user clicks on to sort by the specified heading. The sort buttons are shown here:



The look of these buttons are defined as follows:

```
subject:
    <Constant name="sortBySubjectLinkText"><font face="Arial,Tahoma,Verdana" size=-1 color="#ffffff">
    <b>Subject:</b></font>></Constant>
author:
    <Constant name="sortByAuthorLinkText"><font face="Arial,Tahoma,Verdana" size=-1 color="#ffffff">
    <b>Author:</b></font>></Constant>
text:
    <Constant name="sortByDateLinkText"><font face="Arial,Tahoma,Verdana" size=-1 color="#ffffff">
    <b>Date:</b></font>></Constant>
```

linking text

Using user-defined variables, the following coding fragments define the link between each item in the overview and the text of the article.

```
sortBySubjectLink:
    <Variable name="sortBySubjectLink">
        <A HREF="<Insert var="currURLMinusSort"/>subject@all/@article@
    <Insert var="artNum"/>"><title="Sort by Subject"><Insert var="sortBySubjectLinkText"/></A></Variable>
sortByAuthorLink:
    <Variable name="sortByAuthorLink"><A HREF="<Insert var="currURLMinusSort"/>from/@article@<Insert
```

```
        var="artNum"/>"><title="Sort by Author"><Insert var="sortByAuthorLinkText"/></A></Variable>
sortByDateLink:
        <Variable name="sortByDateLink">
                <A HREF="<Insert var="currURLMinusSort"/>date/@article@<Insert var="artNum"/>">
        <B><title="Sort by Date"><Insert var="sortByDateLinkText"/></A></Variable>
```

table with linked text

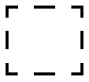
Typically, the overview list shows a table of items, such as that coded below. The hot links are coded into the table, then the table is populated using the overview function.

```
<table width_0% border="0" cellpadding="4" cellspacing="0">
        <tr bgcolor="#000099">
        <td width="4%">&nbsp;</td>
        <td width="*" align="left"><Insert var="sortBySubjectLink"/></td>
        <td width="2%">&nbsp;</td>
        <td width="28%"><Insert var="sortByAuthorLink"/></td>
        <td width="2%">&nbsp;</td>
        <td width="19%"><Insert var="sortByDateLink"/></td>
        </tr>
<Insert var="overviewLines"/>
</table>
<br>
```

Tip: For more information on the line format, in this example called overviewLines, refer to setting the line format on page 53. For more information on the Overview function, refer to using the overview function on page 54.

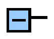
expand/collapse threads

not-expandable image: a blank gif image that displays next to all article that can't be expanded.



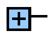
```
<Constant name="notExpandableLinkText">
        <img src="/img/p.gif" border=0 width="15" height="13" >
</Constant>
```

minus-link image: an image indicating that this thread can be collapsed.



```
<Constant name="minusLinkText">
        <img src="/img/minus-15x13.gif" border=0 width="15" height="13" alt="collapse">
</Constant>
```

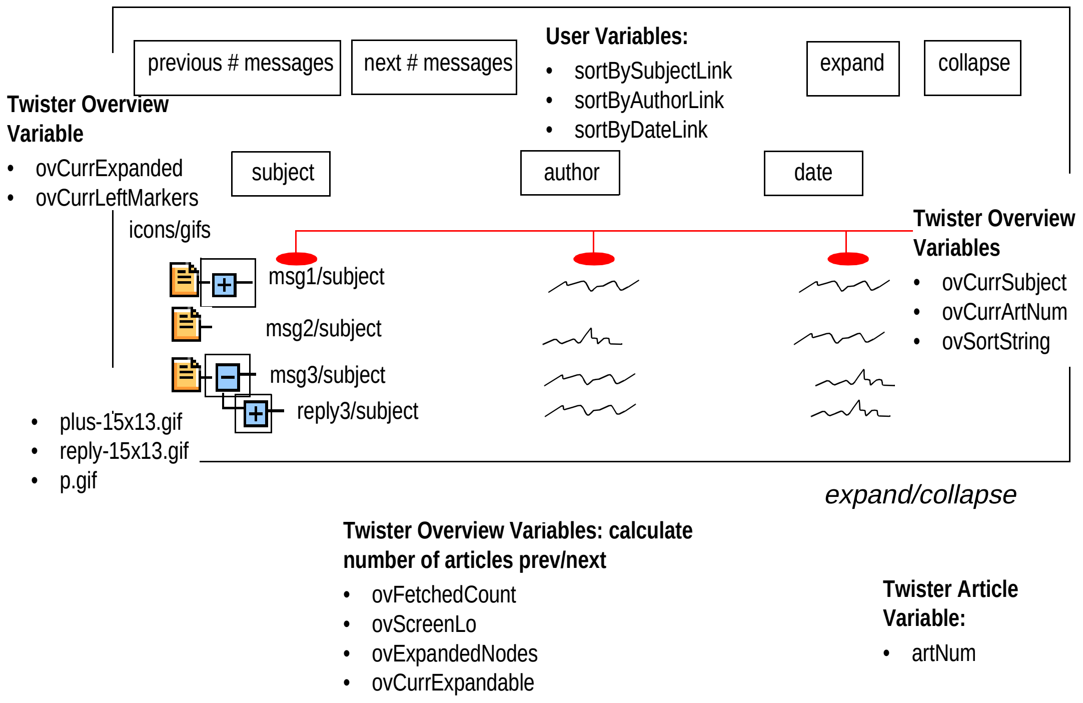
plus-link image: an image indicating that this thread can be expanded.



```
<Constant name="plusLinkText">
        <img src="/img/plus-15x13.gif" border=0 width="15" height="13" alt="expand">
</Constant>
```

This example uses multiple code fragments to link threads:

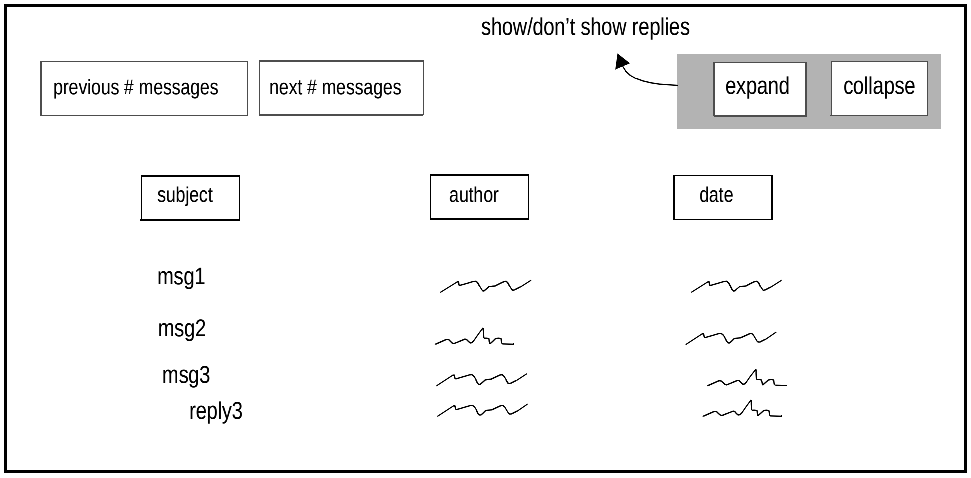
- linking each item in the overview list to any associated responses (plusMinusLink)
- putting the proper image before each line, depending on whether it can be expanded, marked by a plus sign, or whether it has already fully expanded, marked by a minus sign (plusMinus)
- determining whether the displayed data is sorted by subject.



This example connects a plus image with its associated response text; a minus image appears before fully expanded items. The following tagging fragment uses some shorthand abbreviations, defined here, and the images used for plus and minus.

the look

The expand and collapse text on the layout is defined in a very few lines. The location of these buttons is called out on the following diagram.



The look of these buttons are defined as follows:

expand:

```
<Constant name=”expandAllLinkText”>
    <font face=”Arial,Tahoma,Verdana” size=-1 color=”#000099”><b>Expand all</b></font>
</Constant>
```

collapse:

```
<Constant name=”collapseAllLinkText”>
    <font face=”Arial,Tahoma,Verdana” size=-1 color=”#000099”><b>Collapse all</b></font>
</Constant>
```

link to responses

This coding example links a line in the overview, and associated graphic, to a specific article.

```
<Variable name=”plusMinusLink”>
<Function name=”ifEqual”>
    <Param name=”a”><Insert var=”ovCurrExpanded”/></Param>
    <Param name=”b”>Y</Param>
    <Param name=”ifEq”><A href=”<Insert var=”currOvURL”/>-
        <Insert var=”ovCurrArtNum”/><Insert var=”currArtURLFragment”/>”>
        <Insert var=”minusLinkText”/></A></Param>
    <Param name=”ifNotEq”><A href=”<Insert var=”currOvURL”/>+
        <Insert var=”ovCurrArtNum”/>
        <Insert var="currArtURLFragment"/>">
        <Insert var="plusLinkText"/></A></Param>
</Function>
</Variable>
```

insert plus/minus

The following nested function calculates whether the overview list of items is threaded; if it is, then the plus/minus linking occurs (see preceding code fragment), and images are placed next to items as appropriate.

```
<Variable name="plusMinus">
<Function name="ifEqual">
    <Param name="a"><Insert var="ovIsThreaded"/></Param>
    <Param name="b">Y</Param>
        <Param name="ifEq"><Insert var="plusMinusLink"/></Param>
        <Param name="ifNotEq"><Insert var="notExpandableLinkText"/></Param>
</Function>
</Variable>
```

collapse/expand text

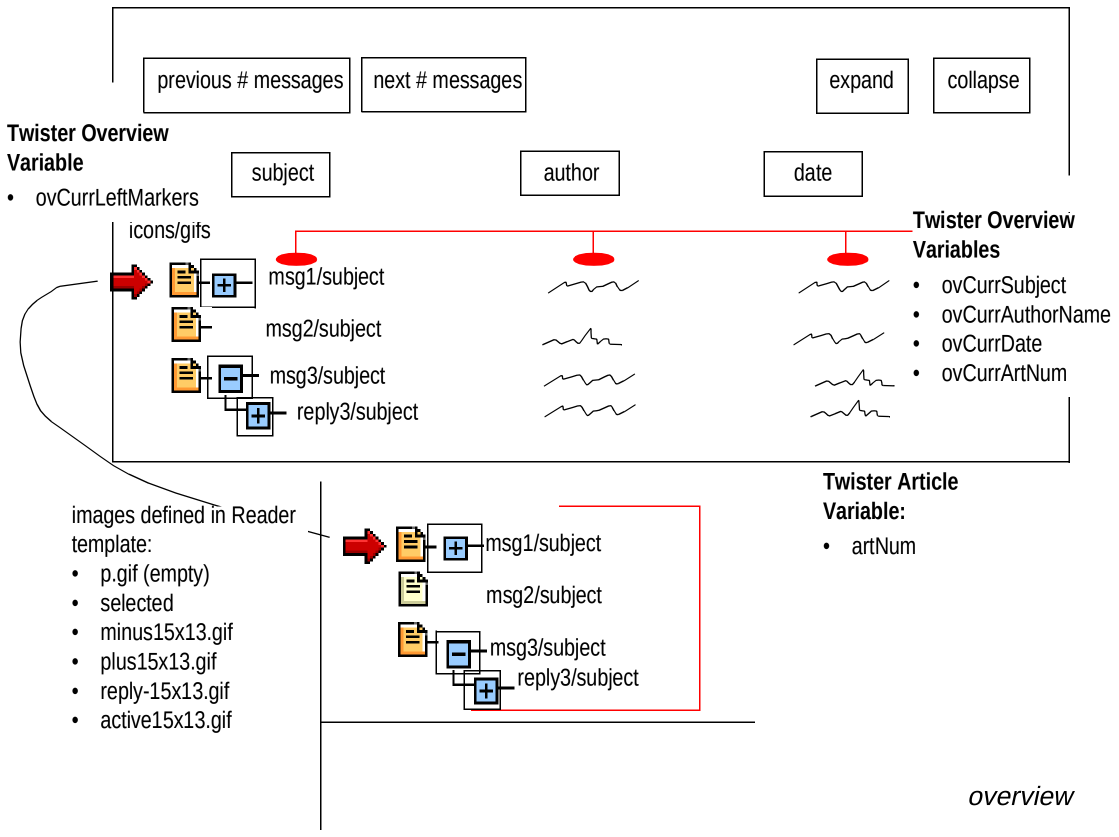
The following nested functions link to an article and display the text to click on for expand/collapse.

```
<Variable name="expandAllLink">
<Function name="ifEqual">
    <Param name="a"><Insert var="ovIsThreaded"/></Param>
    <Param name="b">Y</Param>
    <Param name="ifEq"><A href="<Insert var="currOvURLMinusExpanded"/>all<Insert
var="currArtURLFragment"/>">
        <Insert var="expandAllLinkText"/></A></Param>
    <Param name="ifNotEq"></Param>
</Function>
</Variable>

<Variable name="collapseAllLink">
<Function name="ifEqual">
    <Param name="a"><Insert var="ovIsThreaded"/></Param>
    <Param name="b">Y</Param>
    <Param name="ifEq"><A href="<Insert var="currOvURLMinusExpanded"/>none
        <Insert var="currArtURLFragment"/>">
        <Insert var="collapseAllLinkText"/></A></Param>
    <Param name="ifNotEq"></Param>
</Function>
</Variable>
```

overview list

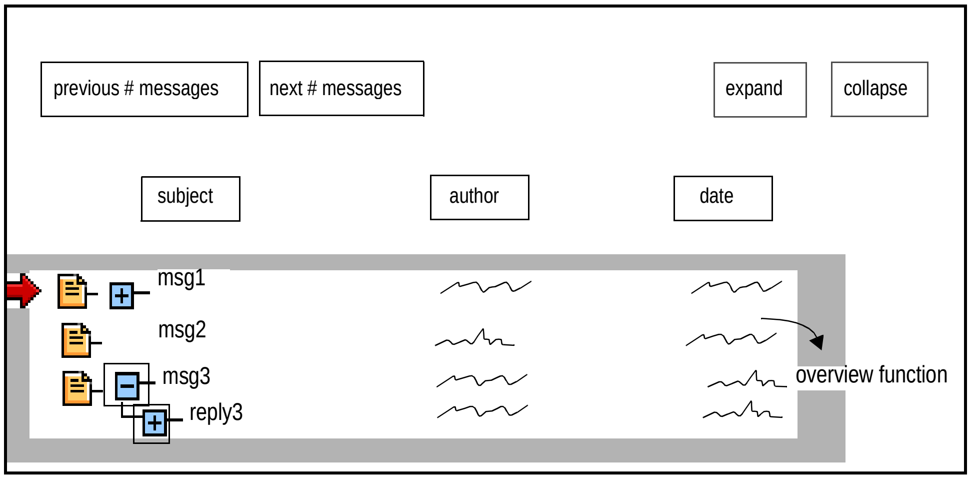
The overview diagram, below, calls out portions of the overview half of the reader layout, including ovCurrLeftMarkers, buttons that complete actions, and variables and images associated with each element.



using overview & ifEqual functions

The overview function retrieves multiple lines of data, each line containing the data you specified using Twister variables, and can display graphics next to items to show whether the items have any associated responses, etc. To set up a complex overview function, Twister needs:

- variables corresponding to the data returned
- format for each line the overview function is to provide
- images and instructions for placing them, depending on whether an item is a new thread or a response, etc.



variables and gifs

variables. For a brief definition of the variables used in this diagram, refer to the table listing overview variables on page 35.

gifs. Each of the gif images is assigned a constant name, so that it can easily be referenced by functions.

selected image: an arrow that displays next to the selected article.

```
<Constant name="selectedText">
     <img src="/img/selected.gif" width="16" height="13" border="" alt="">
</Constant>
```

unselected image: a blank gif image that displays next to all unselected articles; this aligns the selected and unselected articles.

```
<Constant name="unselectedText">
     <img src="/img/p.gif" width="16" height="13" border="" alt="">
</Constant>
```

indent-spacer image: a blank gif image that displays before indented items, aligning them.

```
<Constant name="myIndentSpacer">
     <img src="/img/p.gif" height="13" width="15">
</Constant>
```

reply marker image: a right-angle that links an object with its displayed response.

```
<Constant name="myReplyMarker">
     <img src="/img/reply-15x13.gif">
```

```
</Constant>
```

new-thread image: an image indicating a new thread.

```
<Constant name="myNewThreadMarker">
        <img src="/img/active-15x13.gif" height="13" width = "15">
</Constant>
```

subjectLink

This links each article (subject), shown in the overview list with its article number. With this data, you can compare the article number in the overview list to the number of the article displayed in the bottom half of the reader display. This lets you place a graphic next to the selected article.

```
<Variable name="subjectLink">
        <A href="<Insert var="currOvURL"/>/@article@<Insert var="ovCurrArtNum"/>">
        <Insert var="subjectText"/> </A>
</Variable>
```

selPtr

To identify the article in the overview list that corresponds to the article whose text is displayed in the article portion of the reader display, use the arrow graphic along with the Twister ifEqual function. In this example, the empty image is put next to all articles in the list that are not selected; the arrow is next to the article whose text is displayed in the bottom half of the display:

```
<Variable name="selPtr">
<Function name="ifEqual">
        <Param name="a"><Insert var="ovCurrArtNum"/></Param>
        <Param name="b"><Insert var="artNum"/></Param>
        <Param name="ifEq"><Insert var="selectedText"/></Param>
        <Param name="ifNotEq"><Insert var="unselectedText"/></Param>
</Function>
</Variable>
```

setting the line format

```
<Variable name="myOverviewLine"><tr>
        <tr>
        <td width="4%"><Insert var="selPtr"/></td>
        <td width="45%"><Insert var="ovCurrLeftMarkers"/>&nbsp;
                <Insert var="plusMinus"/><font face="Arial,Tahoma,Verdana" size=-1>
                <Insert var="subjectLink"/></font></td>
        <td width="2%">&nbsp;</td>
        <td width="28%"><font face="Arial,Tahoma,Verdana" size=-1><Insert var="author"/></font></td>
        <td width="2%">&nbsp;</td>
        <td width="19%"><font face="Arial,Tahoma,Verdana" size=-1><Insert var="date"/></font></td>
```

```
    </tr>
</Variable>
```

using the overview function

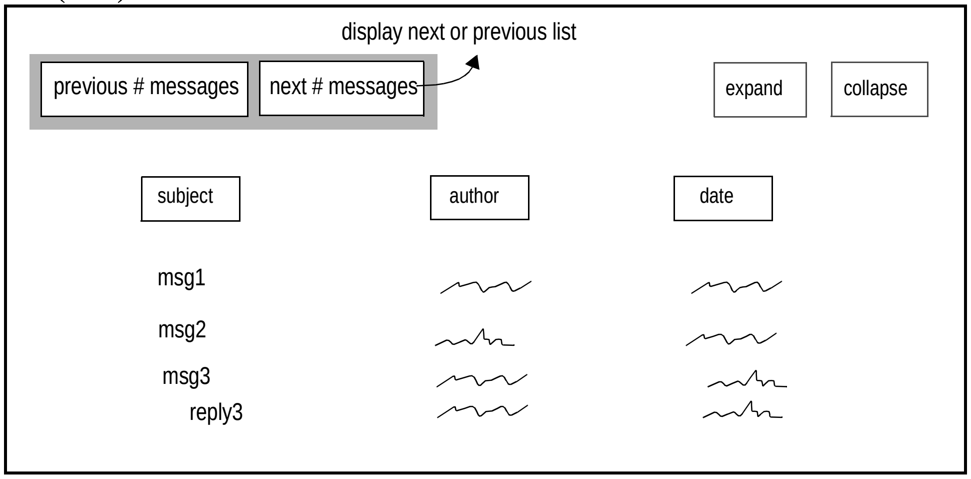
The overview function uses the line format to retrieve and display data:

```
<Variable name=”overviewLines”>
<Function name=”overview”>
    <Param name=”indentSpacer”><Insert var=”myIndentSpacer”/></Param>
    <Param name=”replyMarker”><Insert var=”myReplyMarker”/></Param>
    <Param name=”newThreadMarker”><Insert var=”myNewThreadMarker”/></Param>
    <Param name=”lineFormat”><TR><Insert var=”myOverviewLine”/></TR>
    </Param>
</Function>
</Variable>
```

previous/next list of items

the look

The following defines the look of the previous and next buttons, which is what the user clicks on to display the preceding or next screenful of messages. The previous/next buttons (links) are shown below:



the look

The text in the next and previous buttons is defined here; note that two sets of each are provided, because in some cases each button may not be available. For example, if you are at the beginning of the list of articles, the previous button has no meaning, because no messages precede those displayed. In that case, the disabled button “look” displays.

next button

```
<Constant name="nextBunchLinkText">
        <font face="Arial,Tahoma,Verdana" size=-1 color="#000099">
        <b>Next<Insert var="ovRequestedCount"/> Msgs</b></font>
</Constant>
```

previous button

```
<Constant name="prevBunchLinkText">
        <font face="Arial,Tahoma,Verdana" size=-1 color="#000099">
        <b>Previous <Insert var="ovRequestedCount"/> Msgs</b></font>
</Constant>
```

disabled next button

```
<Constant name="nextBunchLinkTextDisabled">
        <font face="Arial,Tahoma,Verdana" size="-1" color="#999999">
        <b>Next <Insert var="ovRequestedCount"/> Msgs</b></font>
</Constant>
```

disabled previous button

```
<Constant name="prevBunchLinkTextDisabled">
        <font face="Arial,Tahoma,Verdana" size="-1" color="#999999">
        <b>Previous <Insert var="ovRequestedCount"/> Msgs</b></font>
</Constant>
```

URLS of the next/previous to display

The following code fragments are used to calculate whether there are any more previous/next items; if there are, to identify the next/previous set of items to display.

```
<Constant name="nextBunchURL">
        <Function name="ifEqual">
                <Param name="a"><Insert var="ovRangeHi"/></Param>
                <Param name="b"><Insert var="groupHi"/></Param>
        <Param name="ifEq"></Param>
        <Param name="ifNotEq"><Insert var="baseURL"/>@<Insert var="ovRangeHi+1"/>@F@
                <Insert var="ovRequestedCount"/>@<Insert var="ovSortString"/>@
                <Insert var="ovExpandedNodes"/>@article@first</Param>
</Function></Constant>


<Constant name="prevBunchURL">
        <Function name="ifEqual">
                <Param name="a"><Insert var="ovRangeLo"/></Param>
                <Param name="b"><Insert var="groupLo"/></Param>
```

```
    <Param name=”ifEq”></Param>
    <Param name=”ifNotEq”>
        <Insert var=”baseURL”/>@<Insert var=”ovRangeLo-1”/>@R@
        <Insert var=”ovRequestedCount”/>@<Insert var=”ovSortString”/>@
        <Insert var=”ovExpandedNodes”/>@article@last</Param>
    </Function>
</Constant>
```

what to display

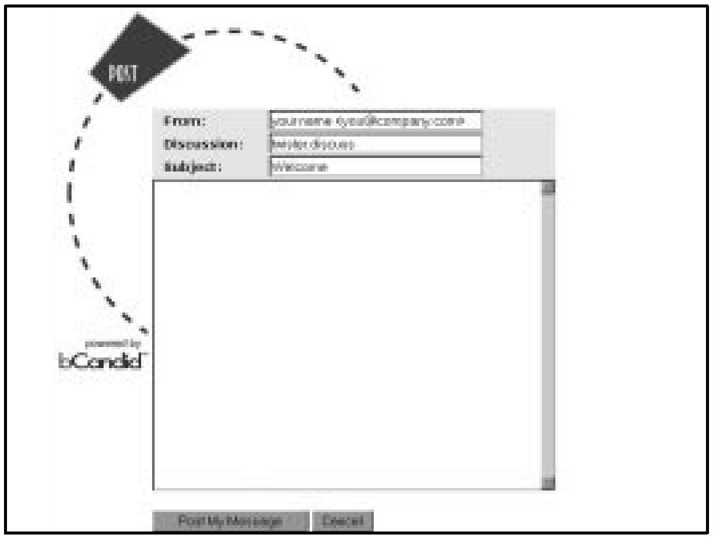
The following code fragments are used to determine what to display; if there are additional items to display (the next/previous set), display them and make sure the button displays properly.

```
<Constant name=”nextBunch”><Function name=”ifEqual”>
    <Param name=”a”><Insert var=”nextBunchURL”/></Param>
    <Param name=”b”></Param>
    <Param name=”ifEq”><Insert var=”nextBunchLinkTextDisabled”/></Param>
    <Param name=”ifNotEq”><a href=”<Insert var=”nextBunchURL”/>”>
        <Insert var=”nextBunchLinkText”/></a></Param>
</Function></Constant>

<Constant name=”prevBunch”>
<Function name=”ifEqual”>
    <Param name=”a”><Insert var=”prevBunchURL”/></Param>
    <Param name=”b”></Param>
    <Param name=”ifEq”><Insert var=”prevBunchLinkTextDisabled”/></Param>
    <Param name=”ifNotEq”><a href=”<Insert var=”prevBunchURL”/>”>
        <Insert var=”prevBunchLinkText”/></a></Param>
</Function>
</Constant>
```

6: sample message post tagging

This chapter reviews some coding fragments that demonstrate how Twister handles posting a new article or a response to a newsgroup. One possible layout to handle posting is illustrated below:

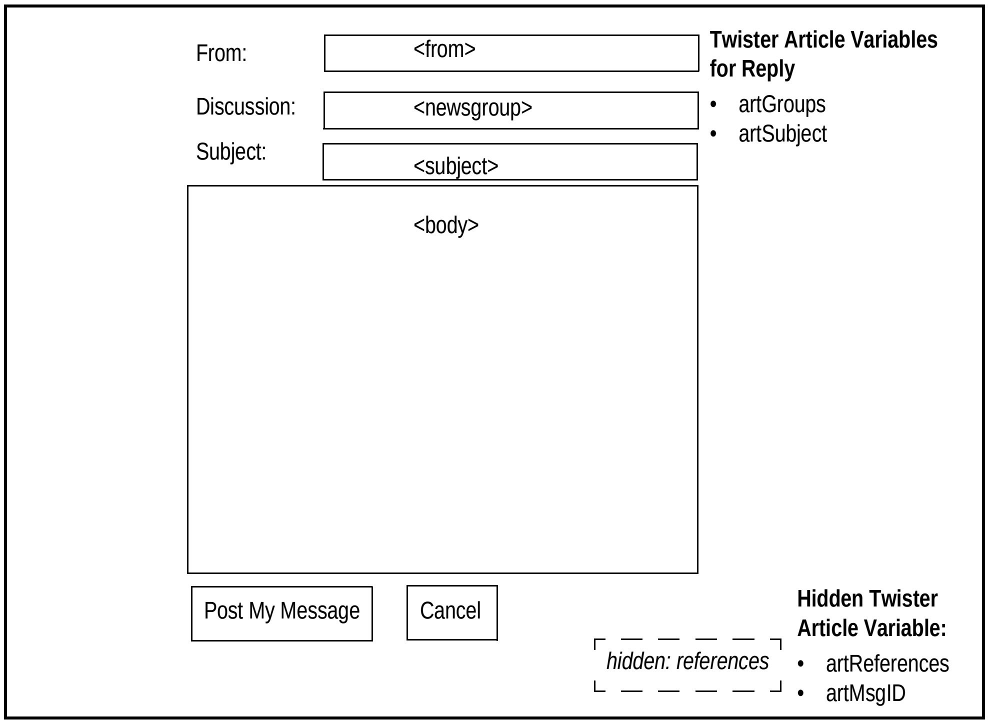


From: yourname <you@company.com>
Discussion: twister.discuss
Subject: Welcome

post/reply

The following diagram shows:
- a form
- three fields and the text area
- a hidden references field
- post and reply buttons

This screen uses HTML forms.



from, discussion, & subject

The look of these fields and the values that may be inserted can be defined using the following tagging:

the look of the from field:

```
<table border=0 cellpadding=0 cellspacing=0 width="1">
<tr bgcolor="#e5e5e5"><td width="9"><img src="/img/p.gif" width="9" height="1" border="0" alt=""></td>
<td width="70"><font face="Verdana,Tahoma,Arial" size=-1><b>From:</b></font></td>
```

the value in the from field:

```
<td width="*"><input type=text size="30" name="from" value="your name <you@company.com>">
    </td></tr>
<tr bgcolor="#e5e5e5"><td width="9"><img src="/img/p.gif" width="9" height="1" border="0" alt="">
</td></tr>
```

the look of the discussion field:

```
<td width="70"><font face="Verdana,Tahoma,Arial" size=-1><b>Discussion:</b></font></td>
```

the value in the discussion field:

```
<td width="*"><input type=text size="30" name="newsgroups" value="<Insert var="artGroups"/>"></td>
```

the look of the subject field:

```
<td width="70"><font face="Verdana,Tahoma,Arial" size=-1><b>Subject:</b></font></td>
```

the value in the subject field:

```
<td width="*"><input type=text size="30" name="subject" value="<Insert var="artSubject"/>">
</td>
```

body text, including reply's quoted original (no attachments)

For posts, rather than replies, the text box is empty. For replies, however, Twister inserts the original text, to which this is a reply, using the quoteBody function, as shown in this tagging fragment:

```
<Function name="ifEqual">
    <Param name="a"><Insert var="artNum"/></Param>
    <Param name="b"></Param>
    <Param name="ifEq"></Param>
    <Param name="ifNotEq"><Function name="quoteBody"/> </Param>
</Function>
```

references

Using a hidden input type that tracks the article variable artReferences, Twister can track any articles to which this post is associated.

```
<input type=hidden size0 name="references" value=<Insert var="artReferences"/>
    <Insert var="artMsgID"/>">
```

submit/cancel buttons

The following includes some HTML form tags:

```
<form action="/action/submit?backto=<Function name="URLEncode"><Param name="data">
    </Param></Function>" METHOD=POST>
```

results of submit

If you do not specify another destination, submitting data sends Twister to postmsg.tpt, the default "goto template." Twister indicates whether the submission was successful (1) or failed (0). On success, Twister renders success.xml; on failure, Twister renders error.xml (the error message). You can also use the message variable to display any message Twister returns on success or failure.

To create your own destination, use

```
submit?goto=XXXX.tpt
```

You can also specify encoding type as a parameter on the action/submit line. If you do not specify any encoding type, Twister uses UUencode. To request MIME-encoding for the posted article, insert:

```
?encoding_type=MIME
```

Optionally, you can request UUencoding:

```
    ?encoding_type=UUENCODE
```

postmsg.cf. The following is a sample postmsg.cf.

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE Template SYSTEM "shared/template.dtd">
<Template>
<Page>
<Function name="ifEqual">
    <Param name="a"><Insert var="success"/></Param>
    <Param name="b">1</Param>
    <Param name="ifEq"><Import file="success.xml"/></Param>
    <Param name="ifNotEq"><Import file="error.xml"/></Param>
</Function>
</Page>
</Template>
```

If you use postmsg.cf, you may want to edit success.xml and error.xml files to provide submission success/failure messages. You can also user postmsg.cf as both success and error pages.

attachments

To enable uploading attachments, add <input type=file name=attachment> to the HTML form on the post page. Also, make sure the form tag includes enctype="multipart/form-data". For example, the file may include:

```
<form enctype="multipart/form-data" input type=file name=attachment
    action="/action/submit?goto=postmsg.tpt" METHOD=POST>.
```

Note that the goto, although explicitly called out above, is the default that is used if goto is absent. For jpg and gif, the images are rendered inline; otherwise, an attachIcon image displays, linking to the item.

```
<Function name="getArtWithAttach">
    <Param name="image/gif*"><img src="/action/decode/<Insert var="encAttachName"/>?msgid=
        <Insert var="encMsgID"/>"></Param>
        <Param name="image/jpeg*"><img src="/action/decode/<Insert var="encAttachName"/>?msgid=
        <Insert var="encMsgID"/>"></Param>
<Param name="default"><A href="/action/decode/<Insert var="encAttachName"/>?msgid=
        <Insert var="encMsgID"/>"><img src="<Insert var="attachIconURL"/>"></A></Param>
</Function>
```

post/reply page tagging (no attachments)

The following shows simplified post/reply tagging:

```
<form action=”/action/submit?goto=postmsg.tpt” METHOD=POST>

<input type=text size0 name=”from” value=”your name < you@company.com <mailto:you@company.com> >”>
<input type=text size0 name=”newsgroups” value=”<Insert var=”artGroups”/>”>
<input type=text size0 name=”subject” value=”<Insert var=”artSubject”/>”>

<input type=hidden size0 name=”references” value=”<Insert var=”artReferences”/><Insert var=”artMsgID”/>”>

<textarea>
<Function name=”ifEqual”>
        <Param name=”a”><Insert var=”artNum”/></Param>
        <Param name=”b”></Param>
        <Param name=”ifEq”></Param>
        <Param name=”ifNotEq”><Function name=”quoteBody”/></Param>
</Function>

</textarea>
<input type=”submit” value=”Post My Message”>
```

index