TopLeaf 7 How To:

Handling PIs

1. Introduction

TopLeaf works by providing a simple, intuitive, point-and-click series of style managers to map XML tags to formatting actions. The source data is never altered. And, through custom markers, XML pseudo-tags can be injected into the XML data stream while it is being processed by TopLeaf to add additional format control over what is capable through tag mappings alone. But, sometimes, some XML environments use processing instructions (PIs) to insert unstructured information into the XML source. Processing instructions can be used by XML editing environments to maintain state information between edit sessions, or other added-value features. In fact, some XML composition systems rely on PIs inserted into the XML data to retain clues as to how the data is to be formatted.

But, TopLeaf's style managers do not currently provide a method for setting up formatting instructions pertaining to PIs. Still, TopLeaf can be directed to act on them. In this how-to, we'll take a look at a simple example of handling a PI with TopLeaf.

2. A Common PI Example

A frequent question is: ***How can I force a page break?*** *The standard answer is:* ***Use the Map Manager to map the start or end tag behavior to force a new page.*** *But, sometimes that's insufficient, because we want to only force it in some circumstances, not others. So, we say:* ***Create a custom tag that controls page breaking, and use TopLeaf's flow control custom programming to call the marker when required.***

Ah, but there are also cases when you don't want such behavior to be always performed, but only when absolutely required, so you want something in the data itself, separate from the structured data, to trigger the page break.

That's when we need a PI. And so, that's when TopLeaf needs to be able to handle it.

Step 1: Decide On Your PI

TopLeaf knows about only one PI: <?TL?>. This PI is used for very specific, obscure, tasks when converting legacy loose leaf systems to TopLeaf. Therefore, don't use it. Beyond that, you can create any PI you like to manage manually-inserted page breaks. For the sake of discussion, let's use the PI:

```
<?PiNewPage?>
```

When this PI is added to the XML source data, the ultimate action will be to cause a style change in TopLeaf that results in a new page being forced.

Step 2: Create a Custom Marker

Yes, you need a custom marker. This is the safest way to handle a dangerous situation. By calling a custom marker, you minimize the risk of breaking anything else in the processing stream, so it's the best safety-net you can buy when doing unrecommended things like this.

Let's say you want your custom marker to be:

```
%PiNewPage
```

When you create this tag in the Map Manager, set the start tag's "New Page" check-box on.

Step 3: Programming the Handler

TopLeaf has a built-in **directive** called **<pimap/>**. The **<pimap/>** directive provides a way to map the PIs you need to custom markers easily. You can have as many instances of the **<pimap/>** directive as you want, for as many PIs as you need to handle.

In our example, we want to associate the <?PiNewPage?> PI to the %PiNewPage custom marker, so that the custom marker is processed when the PI is encountered in the XML data stream. To do this, we open the map manager, and select the **$document** mapping. In the custom box, we add:

```
<pimap
       name="PiNewPage"
       custom-marker="PiNewPage"
       attribute="args"
       data="string"
 />
```

The **name** attribute identifies the PI we want to handle. The **custom-marker** identifies the name of the custom marker TopLeaf will run when the PI is encountered.

You can optionally identify a custom marker attribute by defining the **attribute** attribute. When you do this, any content of the PI will be passed into the custom marker as the value associated with that attribute.

For example, supposing you have a PI that contains:

```
<?PiNewPage landscape ?>
```

The custom marker will be called like this:

```
<PiNewPage args="landscape"/>
```

Within the custom marker's custom code, you could then check the attribute value (refrencing "{@args}", and process the PI according to the value of the attribute.

Two values for the **data** attribute can be specified for the **<pimap/>** directive: "string" or "attributes"

If the data type is set to "attributes", you do not need to specifiy a value for the **attribute** attribute. In this case, the content of the PI is parsed and passed as a series of attributes with their associated values. This is for the case where your PI content looks like this:

```
<?PiNewPage orientation="landscape" size="legal" ?>
```

When handled this way, the custom marker is called like this:

```
<PiNewPage orientation="landscape" size="legal" />
```

Once you have completed the definition of the **<pimap/>** directive, TopLeaf will respond to the PI in your XML data and apply the <PiNewPage> custom marker as though it were called from the Map Manager's custom boxes.

3. Conclusion

TopLeaf supports processing PIs through its style managers. With a bit of care, you can design PIs to run Map Manager custom markers, to achieve the stylistic control you require.

About the Author

John Barker is the co-founder and president of Metaformix Information Systems Inc. He has developed integrated publishing solutions around TopLeaf since 1998.

Contact him at john[_at_]metaformixis.com