Actions Tech. Note 16

Draft 1.0

Writing 'Block Actions'

(some template sample-code)

Writing actions that are intended for use in Dynamic pages frequently requires you to encase sections of HTML in markup. Typically this is required when the web-server is required to repeat a block of code to display some results. If you are writing actions for php, FileMaker, or other actions then you will need this sort of functionality to produce lists of results. Typically you will want to generate code in the form of:

```
<table>
  <tr>
    <td height=12 width=154 valign=top>
    <p>First Name</td>
    <td width=154 valign=top>
    <p>Last Name</td>
  </tr>
  [Repeat]
    <tr>
      <td height=16 width=154 valign=top>
      <p>[Value 1]</td>
      <td width=154 valign=top>
      <p>[Value 2]</td>
    </tr>
  [/Repeat]
</table>
```

This sort of structure is also useful if you want to encase blocks of HTML output in a conditional statement. e.g.

```
<table>
  <tr>
    <td height=12 width=154 valign=top>
    <p>First Name</td>
    <td width=154 valign=top>
    <p>Last Name</td>
  </tr>
  [if customer==good]
    <tr>
      <td height=16 width=154 valign=top>
      <p>You are a good customer!</td>
      <td width=154 valign=top>
      <p>And we like you</td>
    </tr>
  [end if]
</table>
```

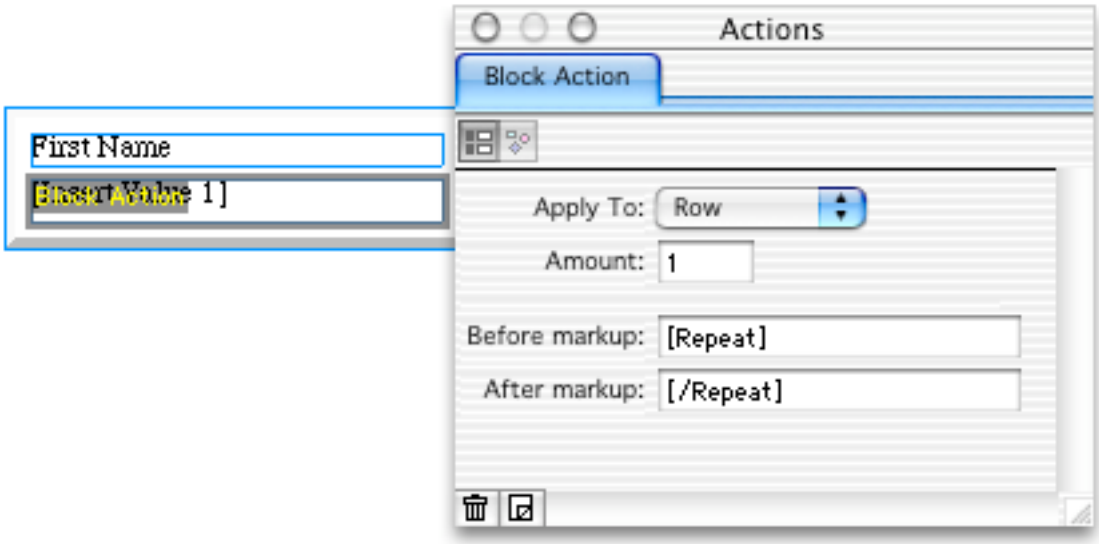
This Tech Note describes a simple (cut and paste) recipe by which you can add this sort of functionality to your actions.

The idea behind 'Block' style actions is to create a single action that you can apply and it will behave moderately intelligently depending on what it is applied to. There are some distinct cases that you may want a block to be applied to.

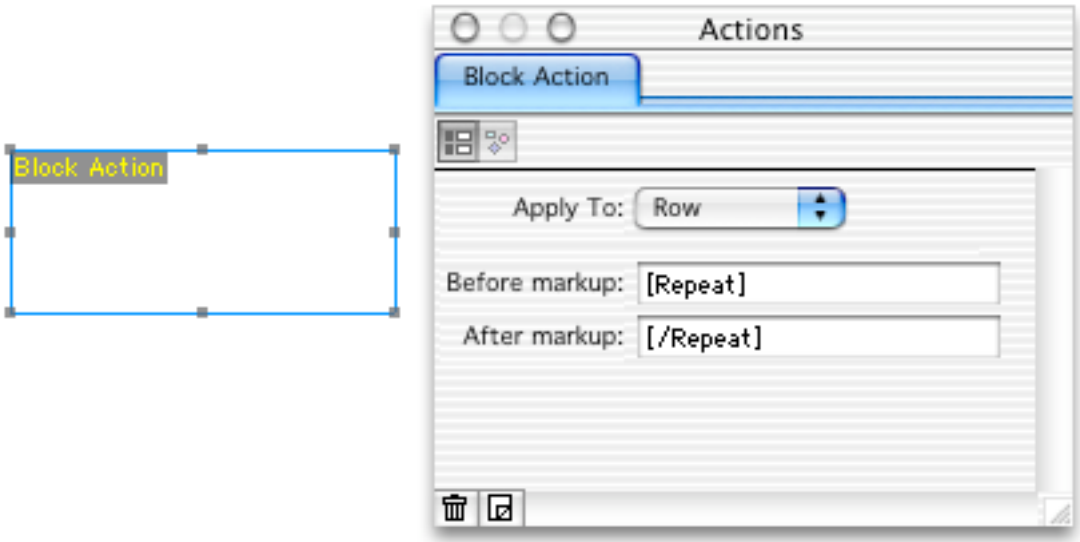
1 Table row - this is the example, whose HTML, is listed above.

2 Table cell - you may wish to create a table whose cells are repeated.
3 Table cell contents. - you may simply want to repeat the contents of a table cell.
4 Item - you may wish to apply it to any item on the Freeway page.

The template action can be applied to a table cell or any item on the Freeway page. When applied to a table cell it will provide control for weather it is applied to the table cell contents, the table cell, or the row in which that cell appears.



If it is applied to normal item you get a simpler interface.



The Template Code

The template code that appears at the end of this Tech Note is simple cut and paste code. There are two functions that you need to consider:

`BlockMarkupInterface` - this is intended to be called in your `fwInterface` function.

`AddBlockMarkup`- this is intended to be called during output, at `fwBeforeEndBody` or later.

```
<item-action  name="Block  Action">
```

```
<action-popup name="Repeat" title="Apply To">
  <value name="Row">
  <value name="Cell">
  <value name="Cell Contents">
</action-popup>

<action-number name="Amount" default=1 minimum=1 width=40 >

<action-label name="l2" title="-" >
<action-text name="before" title="Before markup" default="[Repeat]" width =140>
<action-text name="after" title="After markup" default="[/Repeat]" width =140>

<action-javascript>

  // options in the "Repeat" popup
  var kRow = "Row";
  var kCell = "Cell";
  var kCellContents = "Cell Contents";

</action-javascript>

<action-javascript>

  function EncloseInMarkup(firstTag, lastTag, beforeMarkup, afterMarkup)
  // Enclose tags in mark-up so that the 'beforeMarkup' goes before the
  // 'firstTag', and the 'afterMarkup' goes after the 'lastTag'
  {
    if (firstTag && lastTag)
    {
      // add the beforeMarkup if there is any
      beforeMarkup = beforeMarkup.toString();
      if (beforeMarkup)
      {
        // add it after 'firstTag' tag
        var tag = firstTag.fwEnclosing.fwAddRaw(firstTag, beforeMarkup);

        // and move the tag so that it is before
        firstTag.fwEnclosing.fwMove(firstTag, tag);
      }

      // add the afterMarkup if there is any
      afterMarkup = afterMarkup.toString();
      if (afterMarkup)
      {
        // add it after 'lastTag' tag
        lastTag.fwEnclosing.fwAddRaw(lastTag, afterMarkup);
      }
    }
  }

  function AddBlockMarkup(item, repeatParam, amountParam,
    beforeMarkup, afterMarkup)
  // Enclose a block of this item in markup
  {
    var tags=fwDocument.fwTags.fwFindAll("", item);
    if (tags.length == 0)
      return;

    var isTableCell = item.fwIsTableCell;

    if (isTableCell)
    {
      // make sure that we have a sane number
      var theAmount = parseInt(amountParam.toString());
      if (isNaN(theAmount))
        theAmount = 1;

      // find the parent 'table'
```

```
      var table = tags[0].fwFindEnclosing("table");

      if (repeatParam == kRow)
      {
        var rows = table.fwFindAll("tr");
        for (var i = 0 ; i < rows.length ; i++)
        {
          if (rows[i].fwFind("", item))
          {
            var j = i + theAmount -1;
            if (j >= rows.length)
              j = rows.length - 1;
            EncloseInMarkup(rows[i], rows[j], beforeMarkup, afterMarkup);
          }
        }
      }
      else if (repeatParam == kCell)
      {
        var cells = table.fwFindAll("td");
        for (var i = 0 ; i < cells.length ; i++)
          if (cells[i].fwOwner==item)
          {
            var j = i + theAmount -1;
            if (j >= cells.length)
              j = cells.length - 1;
            EncloseInMarkup(cells[i], cells[j], beforeMarkup, afterMarkup);
          }
      }
      else
      {
        var tag = table.fwFind("", item);
        if (tag)
        {
          EncloseInMarkup(tag, tag, beforeMarkup, afterMarkup);
        }
      }
    }
    else
    {
      for (var i in tags)
      {
        var tag = tags[i];
        EncloseInMarkup(tag, tag, beforeMarkup, afterMarkup);
      }
    }

  }

  function BlockMarkupInterface(item, repeatParam, amountParam)
  {
    var isTableCell = item.fwIsTableCell;
    amountParam.fwVisible = isTableCell && repeatParam != kCellContents;
  }


  function fwBeforeEndBody()
  {
    AddBlockMarkup(fwItem, fwParameters.Repeat, fwParameters.Amount,
      fwParameters.before, fwParameters.after);
  }

  function fwInterface()
  {
    BlockMarkupInterface(fwItem, fwParameters.Repeat, fwParameters.Amount);
  }

</action-javascript>
```

```
</item-action>
```