Tutorial - Undo

This page will walk you through an example of enabling the Fluid Undo functionality in your implementation of a Fluid component. This example will use the Inline Edit component, but the process is the same for any component that supports Undo.

This tutorial assumes that:

- you are already familiar with HTML, Javascript and CSS,
- you are have integrated the Inline Edit (or another Undo-compatible) component into your application,
- now you just want to know how to enable Undo.

For more general information about Undo, see the Undo API page.

Tutorial: How to Enable the Undo Functionality

Scenario

This tutorial assumes that you already have the Inline Edit component integrated into your application. For a tutorial on how to do this, see Tutorial - Simple Text Inline Edit.

On This Page

See Also

- Undo API
- Tutorial - Simple Text Inline Edit
- Simple Text Inline Edit API

Still need help?

Join the infusion-users mailing list and ask your questions there.

Starting point

Let's say you're adding Undo to the Inline Edits you've put into the web interface for your CD database application. Your markup will have the necessary classes:

```
<table id="catalog-table">
  <tr>
    <th>Artist</th>
    <th>Album</th>
    <th>Year</th>
  </tr>
  <tr>
    <td>Bootsy Collins</td>
    <td class="flc-inlineEditable">
      <span class="flc-inlineEdit-text">Ultra Wave</span>
    </td>
    <td>1980</td>
  </tr>
  <tr>
    <td>New York Dolls</td>
    <td class="flc-inlineEditable">
      <span class="flc-inlineEdit-text">One Day It Will Please Us To Remember Even This</span>
    </td>
    <td>2006</td>
  </tr>
  ...
</table>
```

and your script will create the Inline Edit components:

```
jQuery(document).ready(function () {
    fluid.inlineEdits("#catalog-table");
});
```

Step 1: Enabling Undo

Undo can be enabled for your Inline Edit components by specifying configuration parameters for the Undo decorator in the `options` that you pass to the Inline Edit creator function.

To turn Undo on using the options, you can specify the type of `componentDecorators` to be "fluid.undoDecorator":

```
jQuery(document).ready(function () {

    ...

    fluid.inlineEdits("#catalog-table", {componentDecorators: {
            type: "fluid.undoDecorator"
        }
    });

    ...
});
```

This is enough to enable the Undo functionality.

Step 2: Add the script to your HTML

You will need to declare the dependency on the Undo subcomponent by including the minified `InfusionAll.js` file in the header of the HTML file:

```
<script type="text/javascript" src="InfusionAll.js"></script>
```

Alternatively, if you are including individual files, you must include `Undo.js`:

```
<... other dependencies ...>
<script type="text/javascript" src="components/undo/js/Undo.js"></script>
```

Step 3: Customzing Undo

At this point, you can run your application using the default renderer built in to the Undo decorator. The default renderer will give the visual appearance of the "undo/redo" links to the end of each editable item.

Suppose you would like to change the default appearance of the "undo/redo" links by customize it with a different style, some personalized text or maybe even some snazzy icons. This is accomplished in 3 steps:

1. specify the new markup to render by creating a `renderer`
2. specify the new classnames as selectors
3. add the customized Undo to the Inline Edit

Specify a Renderer

Let's change the way the Undo subcomponent looks by adding a different classname (for CSS styling to be done later), an undo/redo icon, as well as some new text specific to the application. Here's how the new renderer code would look like...

```
var myUndoRenderer = function (that, targetContainer) {
    var markup =
      "<span class='flc-undo'>" +
      "<span class='myUndo-undoContainer'><a href='#' class='myUndo-undoControl'>
           <img src='./undo.png' alt='Undo edit'>Undo title edit</a></span>" +
      "<span class='myUndo-redoContainer'><a href='#' class='myUndo-redoControl'>
           <img src='./redo.png' alt='Redo edit'>Redo title edit</a></span>" +
      "</span>";
    var markupNode = $(markup);
    targetContainer.append(markupNode);
    return markupNode;
};
```

The value of `markup` can be any valid HTML markup as long as it follows the Container-Control structure seen in the above code snippet.

Note: if you are sticking with the default Infusion style, then your classnames would be: `flc-undo-undoContainer`, `flc-undo-undoControl`, `flc-undo-redoContainer`, and `flc-undo-redoControl`.

Specify New Classnames as Selectors

Now that you have added our own classname to the Undo, you now need to specify the roles of the new classnames. This can be done by making the following declarations:

```
var mySelectors = {
    undoContainer: ".myUndo-undoContainer",
    undoControl: ".myUndo-undoControl",
    redoContainer: ".myUndo-redoContainer",
    redoControl: ".myUndo-redoControl"
};
```

Note: If you are using the default Infusion style, you can omit this step because you do not need to specify new selectors.

Adding Undo to the Inline Edit

Now that we have a new renderer and new selectors, the last step is to hook up our customized Undo to the Inline Edit.

```
    fluid.inlineEdits("#catalog-table", {componentDecorators: {
            type: "fluid.undoDecorator",
            options: {
                selectors: mySelectors,
                renderer: myUndoRenderer
            }
        }
    });
```

That's it!

Note: If you are using the default Infusion style, then you will not need to pass `selectors` in the `options`.

Putting it Together

So combined the customized Undo subcomponent will look like this:

```
jQuery(document).ready(function () {

    //Other required code (e.g. undo component,

    // Custom renderer
    var myUndoRenderer = function (that, targetContainer) {
        var markup =
          "<span class='flc-undo'>" +
          "<span class='myUndo-undoContainer'><a href='#' class='myUndo-undoControl'>
               <img src='./undo.png' alt='Undo edit'>Undo title edit</a></span>" +
          "<span class='myUndo-redoContainer'><a href='#' class='myUndo-redoControl'>
               <img src='./redo.png' alt='Redo edit'>Redo title edit</a></span>" +
          "</span>";
        var markupNode = $(markup);
        targetContainer.append(markupNode);
        return markupNode;
    };

    // Specify new selectors
    var mySelectors = {
        undoContainer: ".myUndo-undoContainer",
        undoControl: ".myUndo-undoControl",
        redoContainer: ".myUndo-redoContainer",
        redoControl: ".myUndo-redoControl"
    };

    fluid.inlineEdits("#catalog-table", {componentDecorators: {
            type: "fluid.undoDecorator",
            options: {
                selectors: mySelectors,
                renderer: myUndoRenderer
            }
        }
    });
});
```

At this point you add styles for `myUndo-undoContainer`, `myUndo-undoControl`, `myUndo-redoContainer`, and `myUndo-redoControl` in your CSS if you wish.