Drag'n'Drop between two lists

The goal of this tutorial is to detailled the steps for creating two lists whose elements can be moved from one list to the others.

Instantiation

The first step is the creation of the two collections which will be used as data for the lists.

```
collection1 := #(1 2 3 4 5).
collection2 := #(a b c d e).
```

The two `ListModel` widgets can then be created

```
list1 := ListModel new
    items: collection1;
    yourself.

list2 := ListModel new
    items: collection2;
    yourself.
```

Drag and Drop settings

In this section, we will see the settings needed for activating the drag and drop mechanism.

The settings will be only explicited for the list `list1`, but a symetrical implementation for `list2` is required.

For the sake of brievety, this code will not be exposed here.

```
list1
    dragEnabled: true; "activates the drag from this model"
    dropEnabled: true; "activates the drop to this model"
    wantDropBlock: [ :draggedItem :event :source |
     "this block is used to filter the possible objects that could be dropped to
        draggedItem isTransferable and: [ draggedItem source model model = list2
        "the first statement is to ensure only dragged items are expected (and no
ndows),
            when the second statement allows dragged items only if they come fron
     ];
    acceptDropBlock: [ :transfer :event :source :receiver :index |
    "this block is performed when a object is effectively dropped"
        | sourceList |
        "This is used to retreive the model where the dropped object comes from"
        sourceList := transfer source model model.
        "For each dropped element"
        transfer passenger
            do: [ :e |
                "the element in inserted before the targeted list item"
                list1 listItems add: e first beforeIndex: index.
                "and removed from the source list"
                sourceList listItems remove: e first ].
        "Finally both lists are updated"
        list1 updateList.
        sourceList updateList ].
```

The totality of the code can be found here (https://gist.github.com/BenjaminVanRyseghem/9974654).

Conclusion

The tutorial covers the basics of the drag and drop mechanism supported by Spec, and provides an example of how to use it.

I want to personally thank Thierry Goubier and Martin Walk whose discussion (http://forum.world.st/Drag-and-drop-items-between-list-views-td4752285.html) on the mailing-list inspired me for this tutorial.

Back (/docs/repositories_/) Next (/docs/repositories_/)

Getting Started

Welcome (/docs/home/)
Quick start (/docs/quickstart/)
Installation (/docs/installation/)
Example (/docs/example/)

Your UI

Where to start? (/docs/starting/)
Instantiating sub widgets (/docs/initializing/)
Defining the layout (/docs/layout/)
Opening your UI (/docs/open/)
Sub widgets interaction (/docs/interactions/)
Dynamic UI (/docs/dynamic/)
Using Pharo window (/docs/use-pharo-window/)
Inserting a Morph (/docs/insert-morph/)

Inside Spec

Where to find what I want (/docs/api/)
Creating new basic widgets (/docs/own-model/)
Spec Interpreter (/docs/interpreter/)

Tutorials

👉 **Drag'n'Drop between two lists**

(/docs/drag_n_drop/)

More

Repositories
(/docs/repositories/)