Making drag and drop work in XPages - Part 3

Coding drag-and-drop

There is a demo site accompanying this series of posts on drag and drop. The text is mostly the same, but you get working examples on the demo site.

Coding drag and drop

The basic examples use HTML5 Sortable as it is easier to use. We'll deal with jQuery UI widgets when we get to more advanced examples.

Implementation of drag and drop is usually done using unordered lists, ul, but you could use almost any other HTML element.

There is a simple example on the demo site, with two unordered lists. You simply grab one of the elements from the left-hand side list and drop it on the placeholder on the right-hand side.

Once the element is on the right-hand side, you can change its position relative to the other elements. You can also grab it and drop it back onto the source list on the left-hand side, thus effectively removing it from the target list.

Let's take a look at the code. The HTML code is nothing special:

We can ignore div tags as they are only used to position lists. We then have two unordered lists, one with some items, and the other empty. The important thing to note here are the classes attached to these lists: source, target and connected.

JavaScript code required to make this example work is extremely simple:

```
<script type="text/javascript">
  $(function () {
    $(".source, .target").sortable({
      connectWith: ".connected"
    });
  });
</script>
```

On line 3 we select all HTML elements with class source and target (i.e. our unordered lists) and attach sortable() to them which enables drag, drop and sort functionality.

On line 4, we also pass connectWith: ".connected" option to the sortable() which enables dragging and dropping between elements of class connected .

Finally, this is the CSS used to style lists in the above example:

```
ul.source, ul.target {
  min-height: 50px;
  margin: 0px 25px 10px 0px;
  padding: 2px;
```

```
  border-width: 1px;
  border-style: solid;
  -webkit-border-radius: 3px;
  -moz-border-radius: 3px;
  border-radius: 3px;
  list-style-type: none;
  list-style-position: inside;
}
ul.source {
  border-color: #f8e0b1;
}
ul.target {
  border-color: #add38d;
}
.source li, .target li {
  margin: 5px;
  padding: 5px;
  -webkit-border-radius: 4px;
  -moz-border-radius: 4px;
  border-radius: 4px;
  text-shadow: 0 1px 0 rgba(255, 255, 255, 0.5);
}
.source li {
  background-color: #fcf8e3;
  border: 1px solid #fbeed5;
  color: #c09853;
}
.target li {
  background-color: #ebf5e6;
  border: 1px solid #d6e9c6;
  color: #468847;
}
.sortable-dragging {
  border-color: #ccc !important;
  background-color: #fafafa !important;
  color: #bbb !important;
}
.sortable-placeholder {
  height: 40px;
}
.source .sortable-placeholder {
  border: 2px dashed #f8e0b1 !important;
  background-color: #fefcf5 !important;
}
.target .sortable-placeholder {
  border: 2px dashed #add38d !important;
  background-color: #fefcf5 !important;
```

```
}
```

You can choose your own class names instead of source and target, simply replace them in the HTML and the JavaScript code.

As you can see, there are no styles defined for the connected class, as it is only used to tell the HTML5 Sortable code which lists to connect.

There are also two class names, sortable-dragging and sortable-placeholder, which are hard-coded in the HTML5 Sortable. These two classes are used for styling items in dragging state and drop placeholders.

Making it XPages friendly

Static lists are great for examples, but I doubt you'll find them useful in your XPages application.

You would probably want to read list items from a view or perhaps some field. The easiest way to implement it is by using the Repeat control, similar to this:

```
<ul class="source connected">
  <xp:repeat id="repeat1" rows="30" var="manufacturer"
removeRepeat="true">
    <xp:this.value>
      <![CDATA[${javascript:
        return ["Alfa Romeo", "Audi", "BMW", "Jaguar", "Mercedes",
"Porsche", "Tesla", "Volkswagen", "Volvo"];
      }]]>
    </xp:this.value>
    <xp:text escape="true" id="sourceLI" value="#{manufacturer}"
tagName="li"></xp:text>
  </xp:repeat>
</ul>
```

I am using a static array as data source for the repeat - obviously, you would use your application specific code instead.

Two things to note here:

- Use removeRepeat="true" in the repeat control in order to remove div tag that the repeat control generates.
- Use tagName="li" in the computed text control to make it output li tag, instead of the usual span.

All of the JavaScript code should be at the end of your XPage, like this:

```
<xp:view>
  <!-- rest of the code -->
```

```
    <script type="text/javascript" src="js/jquery-
3 1.9.1.min.js"></script>
4   <script type="text/javascript"
5 src="js/jquery.sortable.min.js"></script>
6   <script type="text/javascript">
7     $(function () {
8       $(".source, .target").sortable({
9         connectWith: ".connected"
10      });
11    });
12  </script>
  </xp:view>
```

Courtesy:
http://blog.squareone.ba/2013/07/making-drag-and-drop-work-in-xpages_29.html