sablo-tabseq

This directive helps provide dynamic tab sequences on the client. All it needs is a set of so called 'design' values on DOM nodes inside your document (<input ... sablo-tabseq="2" ...) and an optional config attribute (<div ... sablo-tabseq-config:{container: true} ...).

Based on only these two things that you can add to a new custom component as needed, the sablo-tabseq directive will generate and dynamically manage the "tabIndex" attributes for all the DOM elements displayed in the browser that use this directive.

IMPORTANT

One must always have in the DOM structure a top-most sablo-tabseq element marked with

```
<someTag ... sablo-tabseq="1" sablo-tabseq-config="{root: true}"/>
```

. This can be for example the <body> tag.

IMPORTANT

sablo-tabseq requires you to include a full jQuery lib in your page, before angular.js. It will not work with angular shipped jQLite only (because it relies on jquery trigger() which is not available in jQLite). Servoy already does this, so this note is only relevant for sablo users.

For example:

```
<!DOCTYPE html>
<html ng-app="myApp" sablo-tabseq="1" sablo-tabseq-config="{root: true}">

  <head>
    <title>Tabseq</title>
    <link href="css/style.css" rel="stylesheet" />
    <script src="../lib/jquery-1.11.1.js"></script>
    <script src="../lib/angular-1.3.0-beta.2/angular.js"></script>
    <script src="controllers/impl.js"></script>
  </head>

  <body ng-controller="myController">
        <mycustomcomponent sablo-tabseq="model1.tabSeq"></mycustomcomponent>
        <div sablo-tabseq="20" sablo-tabseq-config="{container: true, reservedGap: 50}">
            <input sablo-tabseq="1" value="a"/>
            <input sablo-tabseq="-2" value="b"/>
            <input sablo-tabseq="1" value="c"/>
        </div>
        <mycustomcomponent sablo-tabseq="model2.tabSeq"></mycustomcomponent>
  </body>

</html>
```

Here we have two custom web components 'mycustomcomponent' (could be anything, for example you could replace them with inputs as well) and 3 inputs that are meant to have a specific tab traversal sequence.

Let's assume that

```
model1.tabSeq === 30
model1.tabSeq === 10
```

In this case this case the wanted tab sequence is: [last custom component, input field 'a', input field 'c', first custom component]. And indeed when you would run this example, the tabIndex attribute will automatically get set like this (ignoring the fact that mycustomcomponent would probably be expanded by angular depending on what that web component does):

```
        <mycustomcomponent sablo-tabseq="model1.tabSeq" tabIndex='53'></mycustomcomponent>
        <div sablo-tabseq="20" sablo-tabseq-config="{container: true, reservedGap: 50}">
            <input sablo-tabseq="1" value="a" tabIndex='2'/>
            <input sablo-tabseq="-2" value="b" tabIndex='-1'/>
            <input sablo-tabseq="1" value="c" tabIndex='2'/>
        </div>
        <mycustomcomponent sablo-tabseq="model2.tabSeq" tabIndex='1'></mycustomcomponent>
```

Notice that I intentionally used 10, 20, 30 as 'design' tab indexes for the 2 custom components and the parent div of the inputs. This is just to show that only their order matters, as at runtime the real tabIndex that is generated by sablo-tabseq directive and used by the browser will start from 1 and increase accordingly.

Also notice that "**-2**" is a special 'design' value that means 'this tag should be skipped completely by tab sequence'. It will always generate a "tabIndex='-1'" for that DOM element and for any child DOM elements that it might have using sablo-tabseq directive.

"**0**" is a special 'design' value as well and it means 'this tag should use default tab sequence'. It will remove "tabIndex" attributes for that DOM element and for any child DOM elements that it might have using sablo-tabseq directive.

Duplicate 'design' tab indexes on the same level are permitted and will generate duplicate tabIndex attribute values (it lets the browser decide which one gets focus first).

Another thing to notice is the div's configuration as tab sequence container (sablo-tabseq-config="{**container: true,** reservedGap: 50}). This means that child DOM elements of that div are considered to be traversed by tab sequence at the 'design' tab index value of that div/container. So at runtime all child DOM elements of that div will get a tabIndex value that sets them between the parent div's siblings, according to the 'design' tab index value of the parent and it's siblings. Also notice the child 'design' values are not related to the ones on the container level. They only logically make sense as compared to the ones on the same level - in this case the 'input' siblings.

About '**reservedGap**': it is an optional configuration option; it will work without it as well. It tells sablo-tabseq that it should 'reserve' a number of tabIndexes for that container on top of the ones it currently needs. That can help later on, if more elements are added to that container and it needs more tabIndexes assigned - it can just use them without recalculating the tabIndexes of the parents (so less calculations to be done in the browser) - at least for a while, until it runs out of reserved indexes.

For example let's say (that div is actually an angular expanded smarter custom web component) that at runtime decides to add 10 more input children in the div. Only the tabIndex values inside the div will be updated as needed, because of the reservedGap:

```
<mycustomcomponent sablo-tabseq="model1.tabSeq" tabIndex='53'></mycustomcomponent>
<div sablo-tabseq="20" sablo-tabseq-config="{container: true, reservedGap: 50}">
    <input sablo-tabseq="1" value="a" tabIndex='3'/>
    <input sablo-tabseq="-2" value="b" tabIndex='-1'/>
    <input sablo-tabseq="1" value="c" tabIndex='3'/>
    <input sablo-tabseq="0" value="new1" tabIndex='2'/>
    <input sablo-tabseq="2" value="new2" tabIndex='4'/>
    <input sablo-tabseq="3" value="new3" tabIndex='5'/>
      (...)
    <input sablo-tabseq="10" value="new10" tabIndex='12'/>
 </div>
<mycustomcomponent sablo-tabseq="model2.tabSeq" tabIndex='1'></mycustomcomponent>
```

But then, if you add 80 more inputs (a real life example could be a table having all sorts of components that can grow depending on data model content) then one sibling of the parent div will be updated as well, as the initial reservedGap is not enough:

```
<mycustomcomponent sablo-tabseq="model1.tabSeq" tabIndex='143'></mycustomcomponent>
<div sablo-tabseq="20" sablo-tabseq-config="{container: true, reservedGap: 50}">
    <input sablo-tabseq="1" value="a" tabIndex='3'/>
    <input sablo-tabseq="-2" value="b" tabIndex='-1'/>
    <input sablo-tabseq="1" value="c" tabIndex='3'/>
    <input sablo-tabseq="0" value="new1" tabIndex='2'/>
    <input sablo-tabseq="2" value="new2" tabIndex='4'/>
    <input sablo-tabseq="3" value="new3" tabIndex='5'/>
      (...)
    <input sablo-tabseq="90" value="new90" tabIndex='92'/>
 </div>
<mycustomcomponent sablo-tabseq="model2.tabSeq" tabIndex='1'></mycustomcomponent>
```