HD3C13 – FlexBox Control

|  |  |
| --- | --- |
| **Product and Focus**  HANA Platform/SAPUI5 | **MOTIVATION**  The FlexBox control and its variants are a workhorse for creating layouts in SAPUI5 applications.  **PREREQUISITES**  HD3C06 – The Base Application |
| **Target Audience**  Undergrduate/Graduate Beginner to Intermediate |
| **Author**  Ross Hightower |
| https://bgoerke.files.wordpress.com/2013/05/section1.png | |

# The FlexBox Control

The Flex Box control is a versatile control. We will only show a couple of its uses but you can find more [here](http://hana.ucc.uwm.edu:8004/sap/ui5/1/sdk/explored.html#/entity/sap.m.FlexBox/samples) and [here](http://help.sap.com/saphelp_uiaddon10/helpdata/en/98/8d2c7652684dea98f9d6dbc94000c0/content.htm?frameset=/en/0e/c1d91487aa43058914ba80ccefbc9b/frameset.htm&current_toc=/en/e4/843b8c3d05411c83f58033bac7f072/plain.htm&node_id=499). The control can be used to organize interface elements either horizontally or vertically. The XML template for the control is shown in the box below.

|  |
| --- |
| <FlexBox xmlns=*"sap.m"*  busy=*"false"*  busyIndicatorDelay=*"1000"*  visible=*"true"*  height=*""*  width=*""*  displayInline=*"false"*  direction=*"Row"*  fitContainer=*"false"*  renderType=*"Div"*  justifyContent=*"Start"*  alignItems=*"Stretch"*>  <tooltip></tooltip> <!-- sap.ui.core.TooltipBase -->  <dependents></dependents> <!-- sap.ui.core.Control, since 1.19 -->  <items></items> <!-- sap.ui.core.Control -->  </FlexBox> |

The direction of orientation is determined by the direction attribute. By default this is set to Row for horizontal layout but it can also be set to Column. The direction of orientation is called the main axis. A simple example is shown in the box below:

|  |
| --- |
| <FlexBox>  <Button text=”Button 1” />  <Button text=”Button 2” />  </FlexBox> |

This would arrange the two buttons side-by-side. Changing the direction as shown below, will cause the buttons to be arranged vertically.

|  |
| --- |
| <FlexBox direction=”Column”>  <Button text=”Button 1” />  <Button text=”Button 2” />  </FlexBox> |

Some of the other attributes of the FlexBox control are described below.

|  |  |  |
| --- | --- | --- |
| Attribute | Description | Usage |
| fitContainer | Determines whether the FlexBox will completely fill its container. | true or false |
| justifyContent | Sets the alignment along the main axis | Left, Center and Right |
| alignItems | Sets the alignment along the cross axis. | Start, Center and End  Stretch – Items take up the whole space along the cross axis.  Baseline - takes the first line of text of each flex item and aligns their baselines. |

In addition to the attributes of the FlexBox control, you can also use layoutData control to modify how the items within the FlexBox are depicted. The example below shows the layoutData control being used to control the order in which the buttons are depicted. The Button 1 control will be displayed after the Button 2 control.

|  |
| --- |
| <FlexBox>  <Button text=”Button 1” >  <layoutData> <FlexItemData order ="1" />  </layoutData>  </Button>  <Button text=”Button 2” />  </FlexBox> |

The table below shows the attributes which can be used with [FlexItemData](https://openui5.hana.ondemand.com/docs/api/symbols/sap.m.FlexItemData.html):

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Usage** |
| growFactor | A floating point value that determines how an element grows to fill the available space | growFactor=”2” (see below) |
| [alignSelf](https://openui5.hana.ondemand.com/docs/api/symbols/sap.m.FlexAlignSelf.html) | Determines how an element is aligned horizontally | sap.m.FlexAlignSelf.Center  sap.m.FlexAlignSelf.Left  sap.m.FlexAlignSelf.Right |
| order | An integer that determines the order of elements in a FlexBox | order=”1” |
| shrinkFactor | A floating point value that determines how an element shrinks | shrinkFactor=”1” |

The explanation for growFactor from SAP Help: If, for example, a horizontal flex container is 300px wide and contains two elements of 100px each, 100px would remain. If the growFactor for both flex items is set to 1, both get 50px extra, thus making them 150px wide. If the growFactor is set to 3 for one item and to 1 for the other item, the first item gets additional 75px (¾ of 100px) of the remaining space and the second item 25px (¼ of 100px). If the growFactor is set to its default value of 0, the item is inflexible and both items would keep their width of 100px.

# Create the FlexBox View

### Add the View to the App Navigation

Add a new object for the FlexBox view to the views.json file. You can find and icon [here](https://openui5.hana.ondemand.com/iconExplorer.html). Also, add a route to the Component.js file.

### FlexBox.view.xml

Create a file called **FlexBox.view.xml** in the **view** package. Enter the code below into the file.

|  |
| --- |
| <mvc:View  controllerName="ui5.controller.FlexBox"  xmlns:core="sap.ui.core"  xmlns:mvc="sap.ui.core.mvc"  xmlns="sap.m">  <Page title="Start Coding FlexBoxes!"  showNavButton = "true"  navButtonPress = "handleNavButtonPress">  <IconTabBar class="iconTabBarPaddingTop" id="idIconTabBarMulti"  expanded="{device>/isNoPhone}">  <items>  <IconTabFilter icon="sap-icon://hint" text="Flex Box">  <Text text="Flex Box content goes here ..." />  </IconTabFilter>  <IconTabFilter icon="sap-icon://attachment" text="Columns">  <Text text="Column content goes here ..." />  </IconTabFilter>  <IconTabFilter icon="sap-icon://notes" text="Nested FlexBox">  <Text text="Nested FlexBox content goes here ..." />  </IconTabFilter>  </items>  </IconTabBar>  </Page>  </mvc:View> |

Listing 1

This code uses the IconTabFilter control which is one of the more iconic looks for SAPUI5. There is a later case that describes this control in more detail. For this case, it is just used as a way to organize the layout content.

### FlexBox.controller.js

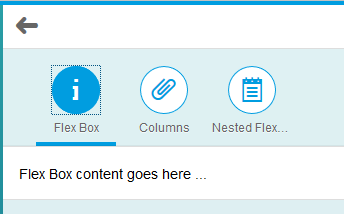
Create a file called **FlexBox.controller.js** andcopy the code shown below into the file:

|  |
| --- |
| sap.ui.define([  "sap/ui/core/mvc/Controller"  ], function(Controller) {  "use strict";  return Controller.extend("ui5.controller.FlexBox", {  onInit: function() {  this.router = sap.ui.core.UIComponent.getRouterFor(this);  },  handleNavButtonPress: function() {  this.router.navTo("Master", {  from: "FlexBox"  });  }  });  }); |

Listing 2

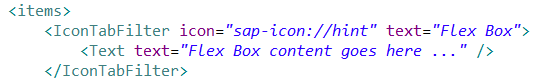
You should be familiar with this code by now. It handles the back navigation button press and navigates back to the Master view.

Now you can run the application. It doesn’t do much yet so let’s add some code to the tabs.



## Create the FlexBox Tab Contents

### FlexBox.view.xml

Use the code shown in the box below to replace the <Text> element in the first IconTabFilter block. 

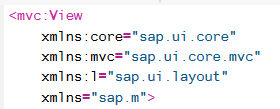
|  |
| --- |
| <l:VerticalLayout class=*"panelBoxContainer sapUiFioriObjectPage"*  width=*"100%"*>  <l:content>  <Panel width=*"80%"* height=*"200px"* class=*"panelShadow"*>  <headerToolbar>  <Toolbar height=*"3rem"*>  <Label text=*"Flex Box - Upper Left"* />  </Toolbar>  </headerToolbar>  <content>  <FlexBox height=*"120px"* alignItems=*"Start"*  justifyContent=*"Start"*>  <items>  <Image src=*"http://go.sap.com/dam/application/shared/logos/sap-logo.png.adapt.72\_36.false.png"* />  </items>  </FlexBox>  </content>  </Panel>  <Panel width=*"80%"* height=*"200px"* class=*"panelShadow"*>  <headerToolbar>  <Toolbar>  <Label text=*"Flex Box - Lower right"* />  </Toolbar>  </headerToolbar>  <content>  <FlexBox height=*"120px"* alignItems=*"End"*  justifyContent=*"End"*>  <items>  <Image src=*"http://go.sap.com/dam/application/shared/logos/sap-logo.png.adapt.72\_36.false.png"* />  </items>  </FlexBox>  </content>  </Panel>  <Panel width=*"80%"* height=*"200px"* class=*"panelShadow"*>  <headerToolbar>  <Toolbar height=*"3rem"*>  <Label text=*"Flex Box - Center"* />  </Toolbar>  </headerToolbar>  <content>  <FlexBox height=*"120px"* alignItems=*"Center"*  justifyContent=*"Center"*>  <items>  <Image src=*"http://go.sap.com/dam/application/shared/logos/sap-logo.png.adapt.72\_36.false.png"* />  </items>  </FlexBox>  </content>  </Panel>  </l:content>  </l:VerticalLayout> |

Listing 3

Notice the l: in the <l:VerticalLayout> and <l:content> tags. This is referencing a library that is not declared yet in the view code. Add the following to the included libraries at the top of the view.

|  |
| --- |
| xmlns:l="sap.ui.layout" |

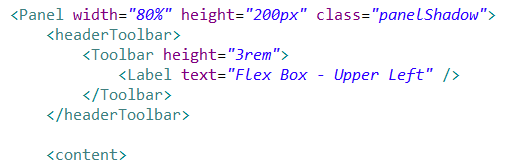
Listing



The three panels are organized using a [VerticalLayout](https://sapui5.hana.ondemand.com/sdk/test-resources/sap/ui/layout/demokit/VerticalLayout.html) control which arranges the content vertically. There is also a [HorizontalLayout](https://sapui5.hana.ondemand.com/sdk/test-resources/sap/ui/layout/demokit/HorizontalLayout.html) control that isn’t used here.



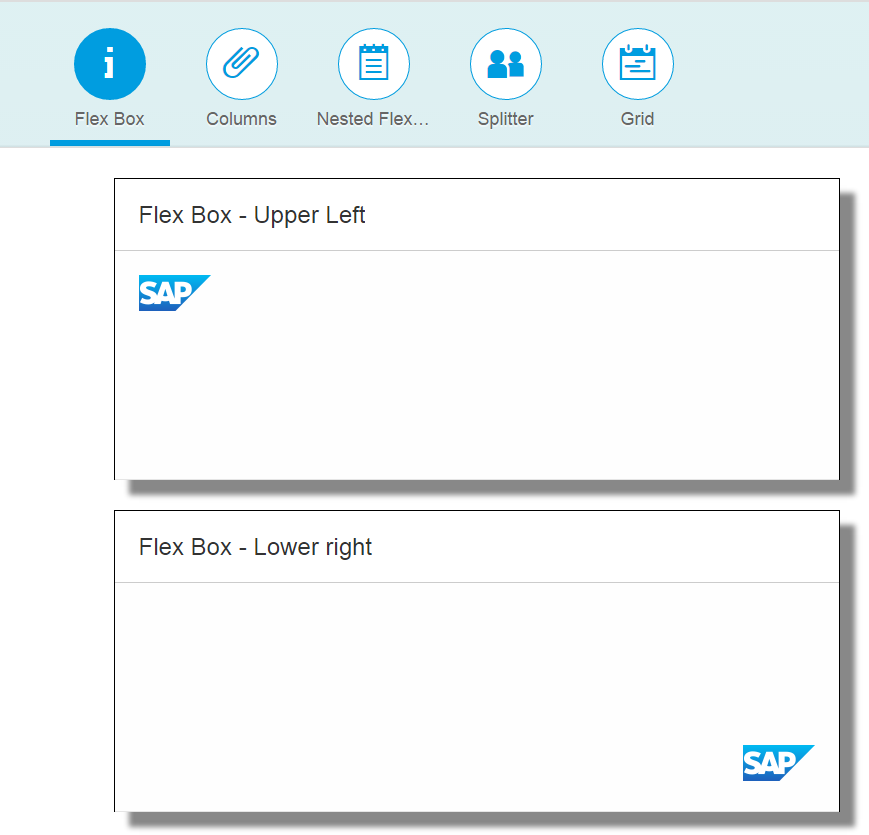
Each of the FlexBox controls is embedded in a [Panel](https://sapui5.hana.ondemand.com/sdk/test-resources/sap/ui/commons/demokit/Panel.html) control. The panel includes a headerToolbar that contains a Label control with the panel title. Notice that the class attribute is referring to a style defined in the Buttons and Toolbars case.



The FlexBox control has two attributes that define the alignment of the content. The **alignItems** attribute determines the vertical alignment and the **justifyContent** attribute determines the horizontal alignment. Each of these has three possible values: Start, Center and End.



Now when you run the application the first icon will show three panels with FlexBoxes.



## FlexBox Columns

Another use of the FlexBox control is to created columns.

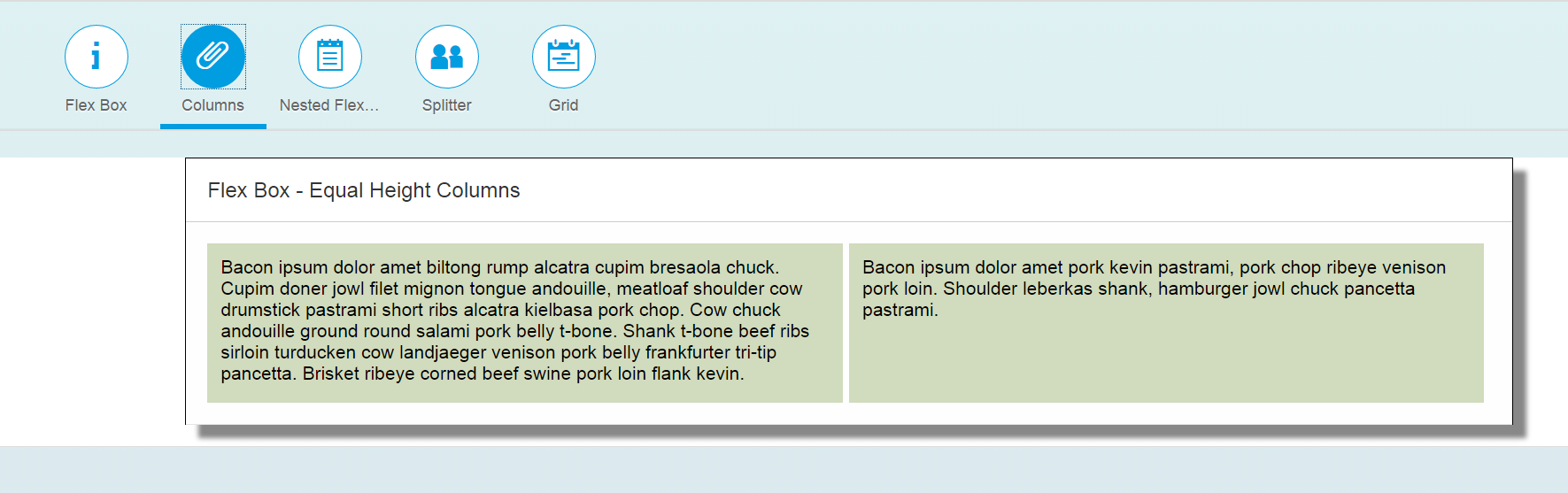
### FlexBox.view.xml

Replace the Text element in the second IconTabFilter block with the code shown below:

|  |
| --- |
| <Panel width="80%" height="200px"  class="panelShadow marginVerticalContent equalColumns">  <headerToolbar>  <Toolbar height="3rem">  <Text text="Flex Box - Equal Height Columns" />  </Toolbar>  </headerToolbar>  <content>  <FlexBox class="columns" height="120px" alignItems="Stretch">  <items>  <Text class="column1" text="Bacon ipsum dolor amet biltong rump alcatra cupim bresaola chuck. Cupim doner jowl filet mignon tongue andouille, meatloaf shoulder cow drumstick pastrami short ribs alcatra kielbasa pork chop. Cow chuck andouille ground round salami pork belly t-bone. Shank t-bone beef ribs sirloin turducken cow landjaeger venison pork belly frankfurter tri-tip pancetta. Brisket ribeye corned beef swine pork loin flank kevin.">  <layoutData> <FlexItemData growFactor="1" />  </layoutData>  </Text>  <Text class="column1" text="Bacon ipsum dolor amet pork kevin pastrami, pork chop ribeye venison pork loin. Shoulder leberkas shank, hamburger jowl chuck pancetta pastrami.">  <layoutData>  <FlexItemData growFactor="1" />  </layoutData>  </Text>  </items>  </FlexBox>  </content>  </Panel> |

Listing 5

When you run the application, the second tab shows a panel with two columns of text.



Note the use of the alignItems attribute. The Stretch value causes the contents to take up the entire space along the cross axis, which in this case is vertically. The Stretch value causes the second column to extend the full height of the panel even though the text contents is less than the left column.

## Nested FlexBoxes

FlexBoxes can be nested within one another to create arbitrary grids.

### FlexBox.view.xml

Replace the Text element in the third IconTabFilter with the code shown below:

|  |
| --- |
| <Panel width=*"80%"* height=*"280px"*  class=*"panelShadow marginVerticalContent equalColumns"*>  <headerToolbar>  <Toolbar height=*"3rem"*>  <Text text=*"Flex Box - Equal Height Columns"* />  </Toolbar>  </headerToolbar>  <content>  <l:VerticalLayout class=*"marginVerticalContent nestedFlexboxes"*  width=*"95%"*>  <l:content>  <HBox fitContainer=*"true"* alignItems=*"Stretch"*>  <items>  <core:HTML content=*"&lt;h2&gt;1&lt;/h2&gt;"*>  <core:layoutData>  <FlexItemData growFactor=*"2"* styleClass=*"item1"* />  </core:layoutData>  </core:HTML>  <core:HTML content=*"&lt;h2&gt;2&lt;/h2&gt;"*>  <core:layoutData>  <FlexItemData growFactor=*"3"* styleClass=*"item2"* />  </core:layoutData>  </core:HTML>  <VBox fitContainer=*"true"*>  <layoutData>  <FlexItemData growFactor=*"7"* />  </layoutData>  <items>  <core:HTML content=*"&lt;h2&gt;3&lt;/h2&gt;"*>  <core:layoutData>  <FlexItemData growFactor=*"5"* styleClass=*"item3"* />  </core:layoutData>  </core:HTML>  <HBox fitContainer=*"true"* alignItems=*"Stretch"*>  <layoutData>  <FlexItemData growFactor=*"3"* />  </layoutData>  <items>  <core:HTML content=*"&lt;h2&gt;4&lt;/h2&gt;"*>  <core:layoutData>  <FlexItemData growFactor=*"1"* styleClass=*"item4"* />  </core:layoutData>  </core:HTML>  <core:HTML content=*"&lt;h2&gt;5&lt;/h2&gt;"*>  <core:layoutData>  <FlexItemData growFactor=*"1"* styleClass=*"item5"* />  </core:layoutData>  </core:HTML>  </items>  </HBox>  </items>  </VBox>  <core:HTML content=*"&lt;h2&gt;6&lt;/h2&gt;"*>  <core:layoutData>  <FlexItemData growFactor=*"5"* styleClass=*"item6"* />  </core:layoutData>  </core:HTML>  </items>  </HBox>  </l:content>  </l:VerticalLayout>  </content>  </Panel> |

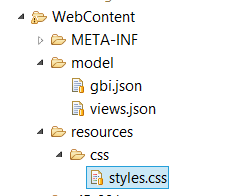
Listing 6

The code looks complex but it’s just a series of FlexBoxes nested within one another. The code will be explained below.

### styles.css

We also need to add the following styles to the **styles.css** file. If you haven’t already created it, create a package called **css** in the project package and create a file called **styles.css**.

|  |
| --- |
| This step is only necessary if you haven’t already added these in the Grid control case. |



Add the styles shown below the styles.css file.

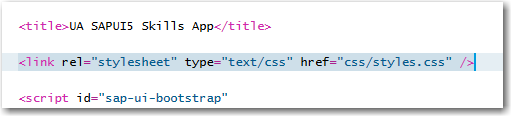
|  |
| --- |
| *.item1 {*  *padding: 1em;*  *background-color: #d1dbbd;*  *}*  *.item2 {*  *padding: 1em;*  *background-color: #7D8A2E;*  *}*  *.item3 {*  *padding: 1em;*  *background-color: #C9D787;*  *}*  *.item4 {*  *padding: 1em;*  *background-color: #FFFFFF;*  *}*  *.item5 {*  *padding: 1em;*  *background-color: #FFC0A9;*  *}*  *.item6 {*  *padding: 1em;*  *background-color: #FF8598;*  *}* |

Listing 7

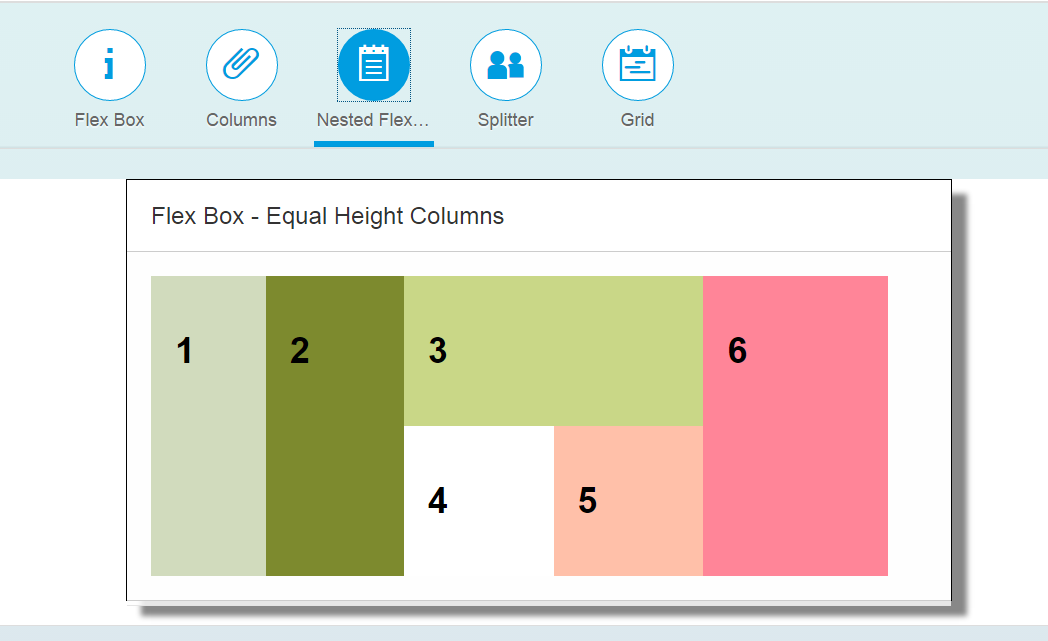
Make sure you have the reference to the styles.css file in your index.html file.

|  |
| --- |
| <link rel="stylesheet" type="text/css" href="css/styles.css" /> |

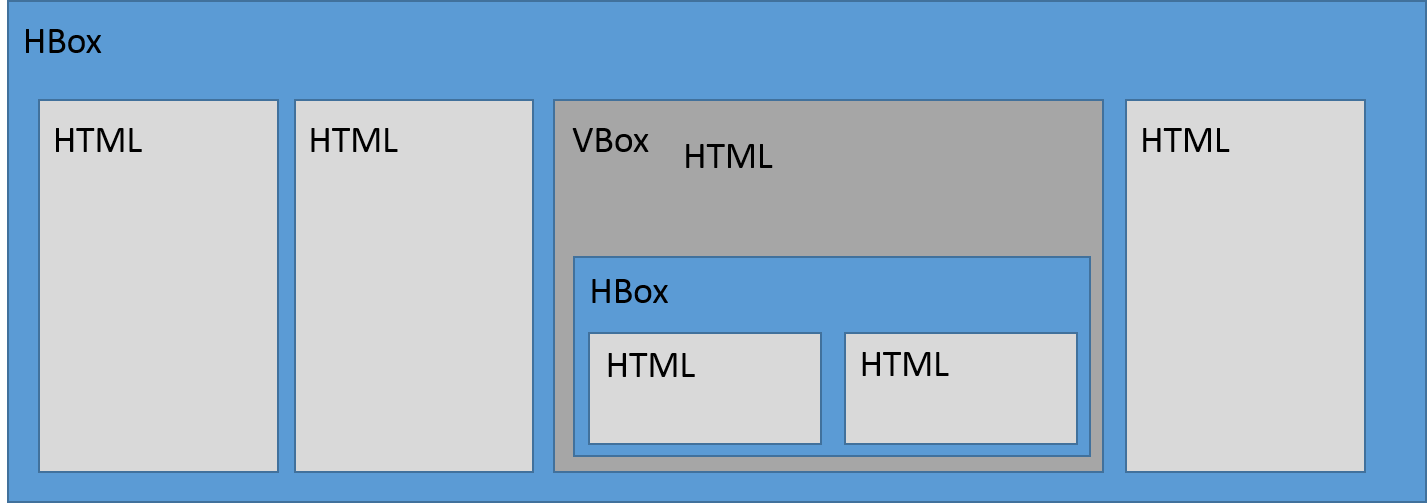
Listing 8



When you run the application, the third tab looks like the screenshot shown below. The code uses two specialized versions of the FlexBox control called HBox and VBox. These controls organize their contents horizontally and vertically respectively.



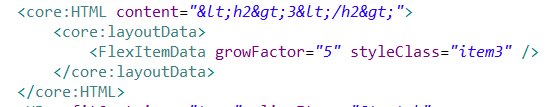
The entire contents are encapsulated in an HBox control. The items in the HBox are two HTML controls (1 and 2 in the figure), a VBox (3, 4 and 5) and another HTML control (6). The VBox control contains an HTML control (3) and an HBox which contains two HTML controls (4 and 5).



Note the use of the alignItems and fitContainer attributes.

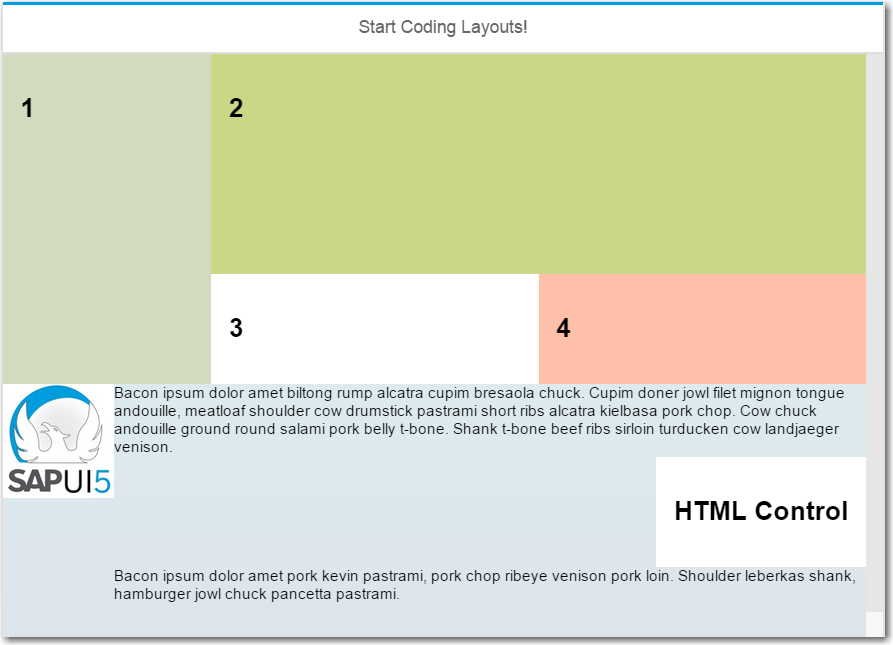


Angle brackets cannot be used in the content of HTML controls so these are replaced by &lt; for a < and &gt; for a >.



# Exercise

Create a view that uses VBox, HBox and FlexBox controls to create this interface. Use any image you wish.



### Adding the View to the Application

To add the view to the application:

1. Add an object to the ExerciseCollection in the views.json file. Remember that the info property specifies the name of the view.
2. Add the route to Component.js.