OpenUI5 - Get Started

Originalartikel

Backup

<html> <h3>A quick Guide to OpenUI5</h3>

```
            <p>
                This tutorial will get you started with your very first
OpenUI5 application. It is an excerpt of the <a
href="https://openui5.hana.ondemand.com/#docs/guide/3da5f4be63264db99f2e5b04
c5e853db.html" target="_blank">Walkthrough Tutorial</a> that is part of the
OpenUI5 SDK documentation. The code samples for each step can be found here:
                <a
href="https://openui5.hana.ondemand.com/explored.html#/entity/sap.m.tutorial
.walkthrough/samples" target="_blank">Walkthrough Samples</a></p>
            <p>You will learn about the following topics:
                </p><ul><li><a
href="http://openui5.org/getstarted.html#step1">Step 1 - Bootstrapping
UI5</a></li>
                <li><a href="http://openui5.org/getstarted.html#step2">Step
2 - Using OpenUI5 Controls</a></li>
                <li><a href="http://openui5.org/getstarted.html#step3">Step
3 - XML Views</a></li>
                <li><a href="http://openui5.org/getstarted.html#step4">Step
4 - Controllers</a></li>
                <li><a href="http://openui5.org/getstarted.html#step5">Step
5 - Databinding</a></li>
                <li><a href="http://openui5.org/getstarted.html#deeper">Dig
deeper</a></li></ul>
            <p>
                A very simple application would look like this:<br/></p>
            <pre class="codeblock">
```

&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;head&gt; &lt;meta http-equiv='X-UA-Compatible' content='IE=edge' /&gt; &lt;title&gt;Hello World&lt;/title&gt; &lt;script id='sap-ui-bootstrap'

```
  src='resources/sap-ui-core.js'
  data-sap-ui-theme='sap_belize'
  data-sap-ui-libs='sap.m'
  data-sap-ui-compatVersion='edge'
  data-sap-ui-preload='async'&gt;
```

&lt;/script&gt; &lt;script&gt;

```
   sap.ui.getCore().attachInit(function () {
       jQuery("#content").html("Hello World - UI5 is ready");
   });
```

&lt;/script&gt; &lt;/head&gt; &lt;body class='sapUiBody'&gt;

```
  &lt;div id='content'&gt;&lt;/div&gt;
```

&lt;/body&gt; &lt;/html&gt;</pre>

```
            <p>In this step we load the OpenUI5 framework from our local
webserver and initialize the core modules with the following configuration
options:
                </p><ul><li>We specify the required UI library "sap.m"
containing the UI controls we need for this tutorial</li>
                    <li>The OpenUI5 controls support different themes, we
choose "sap_belize"</li>
                    <li>To make use of the most recent functionality of
OpenUI5 we define the compatibility version as "edge"</li>
                    <li>We configure the process of bootstrapping to run
asynchronously</li>
                </ul><p>The "src" attribute of the first &lt;script&gt; tag
tells the browser where to find the OpenUI5 core library &#8211; it
initializes the OpenUI5 runtime and loads additional resources such as the
libraries specified in the "data-sap-ui-libs" attribute.
            When all resources and libraries are loaded OpenUI5 is ready.
Then the "init" event of the core is fired and all registered handlers are
executed. This process of loading and initializing OpenUI5 is called
"bootstrapping".
            In our example the resources are loaded asynchronously. This
means that the resources can be loaded simultaneously in the background,
which improves performance.</p><p>Once the OpenUI5 core and the
corresponding resources have been loaded, the core fires the init event to
signal that the library is ready and we can start using it.
            We listen for this event in order to trigger the application
logic once all required resources are available.
            In the example above we get a reference to the OpenUI5 core by
calling </p><pre>sap.ui.getCore()</pre> and register an anonymous callback
function by calling attachInit(...) on the core.
            In OpenUI5 callback functions are often referred to as handlers,
listener functions, or simply listeners. The core is a Singleton and can be
accessed from anywhere in the code.
            Our anonymous callback function is executed when the bootstrap
of OpenUI5 is finished and displays a native JavaScript alert.
            <p>
                Try it out yourself - feel free to modify and extend this
example!
            </p>
            <div class="resultIE" readability="8">
                <p>
                    Your browser doesn't support Cross-Origin Resource
Sharing (CORS). Thus, you cannot see the result of the above sample here.
                </p>
            </div>
            <p><a href="http://openui5.org/getstarted.html#walkthrough">Back
```

```
to top</a></p>
            <h3>Step 2 - Using OpenUI5 Controls</h3>
            <p>Now it is time to build our first little UI by replacing the
"Hello World" text in the HTML body by the OpenUI5 control sap.m.Text.
            For the time being we use the JavaScript control API to set up
the UI. The control instance is then "placed" into the HTML body.
            Don&#8217;t forget to remove the line starting with
"jQuery("#content")" from step 1.</p>
            <p>Here&#8217;s the code for this step:</p>
            <pre class="codeblock">
```

&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;head&gt; &lt;meta http-equiv='X-UA-Compatible' content='IE=edge' /&gt; &lt;title&gt;Hello World&lt;/title&gt; &lt;script id='sap-ui-bootstrap'

```
  src='resources/sap-ui-core.js'
  data-sap-ui-theme='sap_belize'
  data-sap-ui-libs='sap.m'
  data-sap-ui-compatVersion='edge'
  data-sap-ui-preload='async'&gt;
```

&lt;/script&gt; &lt;script&gt;

```
  sap.ui.getCore().attachInit(function () {
      new sap.m.Text({
          text: "Hello World"
      }).placeAt("content");
  });
```

&lt;/script&gt; &lt;/head&gt; &lt;body class='sapUiBody'&gt;

```
  &lt;div id='content'&gt;&lt;/div&gt;
```

&lt;/body&gt; &lt;/html&gt;</pre>

```
        <p>Instead of using native JavaScript to change the content of the
body tag, we want to use a simple OpenUI5 control. Controls are used to
define appearance and behavior of parts of the screen.
```

In the example above the content of the init event handler is now replaced with an OpenUI5 text control. The name of the control is prefixed by the namespace of its control library „sap.m“ and the options are passed to the constructor with a JavaScript object. For our control we set the „text“ property to the value „Hello World“. We concatenate the constructor call of the control with the standard method „placeAt“ that is used to position OpenUI5 controls inside a node of the DOM. As the target node we use the body tag of the HTML document with the id „content“. The class „sapUiBody“ adds additional theme dependent styles for displaying OpenUI5 apps. All controls of OpenUI have a defined set of properties and methods to interact with, the public API of the control. You can it look up in the API reference that is linked below.</p>

```
            <p>
                Try it out yourself - feel free to modify and extend this
```

```
example!
            </p>
            <div class="resultIE" readability="8">
                <p>
                    Your browser doesn't support Cross-Origin Resource
Sharing (CORS). Thus, you cannot see the result of the above sample here.
                </p>
            </div>
        <h4>Read more</h4>
            <ul><li><a
href="https://openui5.hana.ondemand.com/#docs/api/symbols/sap.ui.html"
target="_blank">OpenUI5 API Reference</a></li>
                <li><a
href="https://openui5.hana.ondemand.com/explored.html#/entity/sap.m.Text/sam
ples" target="_blank">sap.m.Text - API Overview &#160; Samples</a></li>
                <li><a
href="https://openui5.hana.ondemand.com/#docs/api/symbols/sap.m.Text.html"
target="_blank">sap.m.Text - API Reference</a></li>
            </ul><p><a
href="http://openui5.org/getstarted.html#walkthrough">Back to top</a></p>
        <h3>Step 3 - XML Views</h3>
        <p>Putting all our UI into the index.html file will very soon result
in a messy setup and there is quite a bit of work ahead of us.
        So let&#8217;s do a first for modularization by putting the
sap.m.Text control into a dedicated "view".
        OpenUI5 supports multiple view types (XML, HTML, JavaScript and
more).
        We choose XML as this produces the most readable code and will force
us separate the view declaration from the controller logic.
        Yet the look of our UI will not change.</p>
        <p>We create a new file for our XML view. The root node of the XML
structure is this view.
        Here we reference the default namespace "sap.m" where the majority
of our UI assets are located.
        We define an additional "sap.ui.core.mvc" namespace where the
OpenUI5 views and all other Model-View-Controller (MVC) assets are located
with an alias "mvc".
        The view code will look like this:</p>
        <pre class="codeblock">
```

&lt;mvc:View

```
  xmlns="sap.m"
  xmlns:mvc="sap.ui.core.mvc"&gt;
  &lt;Text text="Hello World" /&gt;
```

&lt;/mvc:View&gt;</pre>

```
        <p>Inside the view tag we add the declarative definition of our text
control with the same properties as in the previous step.
        The XML tags are mapped to controls and the attributes are mapped to
```

```
the properties of the control.
        In OpenUI5 each control has its own id.
        In the XML View above we did not specify an id attribute and
therefore the OpenUI5 runtime generates a unique id and assigns it to the
control.
        However, it is a good practive to set the ids of controls explicitly
so controls can be identified easily.
        </p>
        <p>Next, we change our application code in index.html again:</p>
        <pre class="codeblock">
```

&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;head&gt; &lt;meta http-equiv='X-UA-Compatible' content='IE=edge' /&gt; &lt;title&gt;Hello World&lt;/title&gt; &lt;script id='sap-ui-bootstrap'

```
  src='resources/sap-ui-core.js'
  data-sap-ui-theme='sap_belize'
  data-sap-ui-libs='sap.m'
  data-sap-ui-compatVersion='edge'
  data-sap-ui-resourceroots='{
      "sap.ui.demo.wt": "./"
  }'
  data-sap-ui-preload='async'&gt;
```

&lt;/script&gt; &lt;script&gt;

```
  sap.ui.getCore().attachInit(function () {
      new sap.ui.xmlview({
          viewName: "sap.ui.demo.wt.App"
      }).placeAt("content");
  });
```

&lt;/script&gt; &lt;/head&gt; &lt;body class='sapUiBody'&gt;

```
  &lt;div id='content'&gt;&lt;/div&gt;
```

&lt;/body&gt; &lt;/html&gt;</pre>

```
        <p>We replace the instantiation of the sap.m.Text control by our new
"App" XML view.
        The view is created by a factory function of OpenUI5 which makes
sure that the view is correctly configured.
            The name is prefixed with the namespace "sap.ui.demo.wt" in
order to uniquely identify this resource.
            Furthermore, we use the "data-sap-ui-resourceroots" attribute to
tell the OpenUI5 core that resources in the "sap.ui.demo.wt" namespace are
located in the same folder as the index.html.
            This is important and necessary for resources to be found by the
UI5 runtime.
        </p>
        <p>
```

```
            Try it out yourself - feel free to modify and extend this
example!
        </p>
        <div class="resultIE" readability="8">
            <p>
                Your browser doesn't support Cross-Origin Resource Sharing
(CORS). Thus, you cannot see the result of the above sample here.
            </p>
        </div>
        <h4>Read more:</h4>
        <ul><li><a
href="https://openui5.hana.ondemand.com/#docs/guide/91f233476f4d1014b6dd926d
b0e91070.html" target="_blank">Model View Controller (MVC) in
OpenUI5</a></li>
                <li><a
href="https://openui5.hana.ondemand.com/#docs/guide/91f27e3e6f4d1014b6dd926d
b0e91070.html" target="_blank">Model View Controller(MVC) &gt;
Views</a></li>
                <li><a
href="https://openui5.hana.ondemand.com/#docs/guide/91f292806f4d1014b6dd926d
b0e91070.html" target="_blank">XML Views</a></li>
            </ul><p><a
href="http://openui5.org/getstarted.html#walkthrough">Back to top</a></p>
        <h3>Step 4 - Controllers</h3>
        <p>In this step we replace the text with a button and show the
"Hello World" message once the button gets pressed. The handling of this
event will be implemented on the controller of the view.</p>
        <p>Here is the new code for our view:</p>
        <pre class="codeblock">&lt;mvc:View
  controllerName="sap.ui.demo.wt.App"
  xmlns="sap.m"
  xmlns:mvc="sap.ui.core.mvc"&gt;
  &lt;Button
      text="Say Hello"
      press="onShowHello" /&gt;
```

&lt;/mvc:View&gt;</pre>

```
        <p>
            We replace the text control with a button that carries the text
"Say Hello" and calls the "onShowHello" function when being pressed. We also
have to state the name of the controller that holds the "onShowHello"
function.
            We create a new file - the controller for our view, which we
will call App.controller.js.
        </p>
        <pre class="codeblock">sap.ui.define([
  "sap/ui/core/mvc/Controller"
```

], function (Controller) {

```
  "use strict";
  return Controller.extend("sap.ui.demo.wt.App", {
      onShowHello: function () {
          // show a native JavaScript alert
          alert("Hello World");
      }
  });
```

});</pre>

```
        <p>We define the App controller in its own file by extending the
"Controller" object of the OpenUI5 core.
        For the beginning it holds only a single function called
"onShowHello" that handles the pressing of the button by showing an alert.
</p>
        <h4>Conventions</h4>
        <ul><li>Controller names are capitalized</li>
            <li>Controller carry the same name as the related view (if there
is a 1:1 relationship)</li>
            <li>Event handlers are prefixed with "on"</li>
            <li>Controller file names always end with *.controller.js</li>
        </ul><p>Try it out yourself - feel free to modify and extend this
example!</p>
    <div class="resultIE" readability="8">
        <p>
            Your browser doesn't support Cross-Origin Resource Sharing
(CORS). Thus, you cannot see the result of the above sample here.
        </p>
    </div>
    <h4>Read more:</h4>
    <ul><li><a
href="https://openui5.hana.ondemand.com/#docs/guide/121b8e6337d147af9819129e
428f1f75.html" target="_blank">Model View Controller (MVC) &gt;
Controllers</a></li>
    </ul><p><a href="http://openui5.org/getstarted.html#walkthrough">Back to
top</a></p>
    <h3>Step 5 - Databinding</h3>
    <p>
        Now that we have set up the view and controller it&#8217;s about
time to think about the M in MVC.
        We will add an input field to our app, bind its value to the model,
and show the user input when pressing the "Say Hello" button.
        We will start by modifying our controller. sap.ui.define allows us
to specify dependencies we would like to use in our controller and have the
        framework load them asynchronously. We have already made use of this
functionality in the previous step, and we will now add the JSON model to
these dependencies - "sap/ui/model/json/JSONModel".
        It needs to be added to the array of dependencies in our controller,
        and is then passed as parameter to the function.
    </p>
    <p>The new controller code will look like this:</p>
```

<pre class=„codeblock“>sap.ui.define([

```
  "sap/ui/core/mvc/Controller",
  "sap/ui/model/json/JSONModel"
```

], function (Controller, JSONModel) {

```
  "use strict";
  return Controller.extend("sap.ui.demo.wt.App", {
      onInit: function () {
          // set data model on view
          var oData = {
              recipient: {
                  name: "World"
              }
          };
          var oModel = new JSONModel(oData);
          this.getView().setModel(oModel);
      },
      onShowHello: function () {
          // show a native JavaScript alert
          alert("Hello World");
      }
  });
```

});</pre>

```
    <p>
        We add an init function to the controller. </p><pre>onInit</pre> is
one of OpenUI5&#8217;s lifecycle methods that is invoked by the framework
when the controller is created, similar to a constructor function of a
control.
        Inside the function we instantiate a JSON model. The data for the
model only contains a single property for the "recipient" and inside this
one another property for the name.
        To be able to use this model from within the XML view we call the
setModel function on the view and pass in our newly created model.
    <p>Next, let us adapt our view once more:</p>
```

<pre class=„codeblock“>&lt;mvc:View

```
  controllerName="sap.ui.demo.wt.App"
  xmlns="sap.m"
  xmlns:mvc="sap.ui.core.mvc"&gt;
  &lt;Button
      text="Say Hello"
      press="onShowHello" /&gt;
  &lt;Input
```

```
      value="{/recipient/name}"
      description="Hello {/recipient/name}"
      valueLiveUpdate="true"
      width="60%" /&gt;
```

&lt;/mvc:View&gt;</pre>

```
    <p>
        We add an sap.m.Input control to the view. With this, the user can
enter the recipient of our greetings.
        We bind its value to the JSON model by using the declarative binding
syntax for XML views:
        </p><ul><li>The curly brackets "{&#8230;}" indicate that data
binding shall be used</li>
        <li>The "/recipient/name" declares the path in the
model</li></ul><p>Last, we have to update our index.html again:</p>
```

<pre class=„codeblock“> &lt;!DOCTYPE html&gt; &lt;html&gt; &lt;head&gt; &lt;meta http-equiv='X-UA-Compatible' content='IE=edge' /&gt; &lt;title&gt;Hello World&lt;/title&gt; &lt;script id='sap-ui-bootstrap'

```
  src='resources/sap-ui-core.js'
  data-sap-ui-theme='sap_belize'
  data-sap-ui-libs='sap.m'
  data-sap-ui-compatVersion='edge'
  data-sap-ui-bindingSyntax='complex'
  data-sap-ui-resourceroots='{
      "sap.ui.demo.wt": "./"
  }'
  data-sap-ui-preload='async'&gt;
```

&lt;/script&gt; &lt;script&gt;

```
  sap.ui.getCore().attachInit(function () {
      new sap.ui.xmlview({
          viewName: "sap.ui.demo.wt.App"
      }).placeAt("content");
  });
```

&lt;/script&gt; &lt;/head&gt; &lt;body class='sapUiBody'&gt;

```
  &lt;div id='content'&gt;&lt;/div&gt;
```

&lt;/body&gt; &lt;/html&gt;</pre>

```
    <p>The binding of the value attribute is a simple binding example that
contains only a binding pattern. We can also combine texts and binding
pattern to a more complex binding result as seen in the description
attribute.
    To be able to use the so-called complex binding syntax we have to
```

```
globally enable it by setting the bootstrap parameter "data-sap-ui-
bindingSyntax" to "complex".
    If this setting is omitted, then only standard binding syntax is
allowed, i.e. using "Hello {/recipient/name}" would not work anymore while
"{/recipient/name}" would work just fine.</p>
    <p>Try it out yourself - feel free to modify and extend this
example!</p>
    <div class="resultIE" readability="8">
        <p>
            Your browser doesn't support Cross-Origin Resource Sharing
(CORS). Thus, you cannot see the result of the above sample here.
        </p>
    </div>
    <h4>Read more</h4>
    <ul><li><a
href="https://openui5.hana.ondemand.com/#docs/guide/e1b625940c104b558e52f47a
fe5ddb4f.html" target="_blank">Model View Controller (MVC) &gt;
Models</a></li>
        <li><a
href="https://openui5.hana.ondemand.com/#docs/guide/68b9644a253741e8a4b9e427
9a35c247.html" target="_blank">Model And Data Binding</a></li>
        <li><a
href="https://openui5.hana.ondemand.com/#docs/api/symbols/sap.ui.html#.defin
e" target="_blank">sap.ui.define API Documentation</a></li>
    </ul><p><a href="http://openui5.org/getstarted.html#walkthrough">Back to
top</a></p>
            <h3>Now, Dig Deeper!</h3>
            <p>
                <a
href="https://openui5.hana.ondemand.com/#docs/guide/3da5f4be63264db99f2e5b04
c5e853db.html" class="downloadbutton">Continue with the Walkthrough
Tutorial...</a>
            </p>
            Read the
            <a
href="https://openui5.hana.ondemand.com/#docs/guide/Documentation.html">Deve
loper Guide</a> and refer to the
            <a
href="https://openui5.hana.ondemand.com/#docs/api/symbols/sap.ui.html">API
Reference</a>
            <p>Watch training videos on our <a
href="https://www.youtube.com/channel/UCOlLpeus2uAJhmxjKHHGTgA">YouTube
channel</a></p>
            <p>Enroll for the free openSAP course <a
href="https://open.sap.com/courses/ui51">Developing Web Apps with SAPUI5</a>
- it is also applicable for OpenUI5.</p>
            <p>Use <a
href="https://openui5.hana.ondemand.com/#docs/guide/a460a7348a6c431a8bd967ab
9fb8d918.html">App Templates</a> as a foundation for your developments
(available in SAP Web IDE or on <a
href="https://github.com/SAP?q=openui5-worklist-app%20OR%20openui5-masterdet
ail-app%20OR%20openui5-sample-app">github</a>).</p>
            <p>Discuss about OpenUI5 on <a
href="https://www.sap.com/community/tag.html?id=500983881501772639608291559920477">sap.com Community</a> and <a
href="http://stackoverflow.com/questions/tagged/sapui5%20or%20openui5">Stack
Overflow</a></p>
            <p>or look into more examples on jsbin.com:</p>
            <ul class="download"><li>
                    <a target="_blank"
href="http://jsbin.com/yowiwukoyo/edit?html,output">UI5 mobile list with
databinding</a>
                </li>
                <li>
                    <a target="_blank"
href="http://jsbin.com/zutamig/edit">A sample of using D3.js wrapped as a
UI5 control with a synced table</a>
                </li>
            </ul>
```

</html>