**Video speed: 1.25 (I recommend)**

**Week 1**

Diagram

Description automatically generated

**To gain access to SAP Web IDE Fullstack you need a SAP Nemo account like.**

ES5 configuration file – 3 settings described:

WebIDEEnabled=true => makes our destination visible to SAP Web IDE

WebIDESystem=ES5 => system ID the service runs on

WebIDEUsage => contains usages and you can enter multiple usages for a destination, separated by commas without spaces

But… the config settings above are not that useful, are they? Since “Portal” cannot be enabled.

Moreover, this IDE is not used anymore, or at least not like in the video anymore, so we can just skip.

**Not skip**, but just watch the videos until you see some ui5 content, and focus there.

Seems like you can run an application on the “cloud” anyway, from the SAP Web IDE:

Graphical user interface, text, application, email

Description automatically generated

Click the Green PLAY round button positioned in the top bar.

*Quiz related. (got 4.5/6 points on this)*

What do you need destinations for?

* To connect to cloud services or on premise systems in a secure and standardized way
* For accessing remote OData services, for example on the SAP Gateway demo system ES5

Unit 3 – Local setup

Setting up the development environment:

npm install -g @ui5/cli (to check if you have it, just use ui5 –help)

git clone <https://github.com/SAP/openui5-basic-template-app.git>

npm install

npm run will tell you the commands you can use

* start = starts a server
* test = executes the test suite which you can use for your own CI. (gr8 to keep quality high)
* lint = executes a static code check
* karma = Is actually more awfseq
  + (if you don’t believe me, check for yourself on unit3 video, at min 04:38 :D)
* watch = task supporting your test-driven development
* build = to deploy the application in the end, somewhere

**Related to the code editor:** Visual Studio Code

**Recommends the plugin:** UI5 Snippets & Extensions, UI5 Explorer, ESLint

**There is one obstacle you need to overcome for your local development:**

Browsers normally follow the same origin policy and blocks the so called CORS requests by default.

**CORS = cross-origin resource sharing**

* enables the flexibility to fetch data from other hosts than the actual web server the app is running on.

**Sooo, we gonna use a proxy:**

In **package.json**:

scripts ->

"proxy": "node proxy.js"

devDependencies ->

"cors-anywhere": "^0.4.1"

Then, we must prefix our service application/URL, with the actual proxy URL, to be able to reroute the CORS via that proxy.

<http://localhost:8081/>

uri: "http://localhost:8081/services.odata.org/V2/Northwind/Northwind.svc"

npm install cors-anywhere

npm run proxy

ui5 serve

Doesn’t matter which one you start first. (proxy/server)

Got error when trying to run the server: errno: -\*\*\*,

(after some searching, I’ve started all again with a fresh copy, and it worked just fine…)

Then, got a “require Headers” error in browser console.

**After googling it up and down, I remembered something proxy related in the Walkthrough.**

**Link:** [**https://sapui5.hana.ondemand.com/sdk/#/topic/5bb388fc289d44dca886c8fa25da466e.html**](https://sapui5.hana.ondemand.com/sdk/#/topic/5bb388fc289d44dca886c8fa25da466e.html)

**Note**

By default, you can't run the request in your browser with the proxy.js script. It throws the following exception: exception Missing required request header. Must specify one of: origin,x-requested-with. If you want to test the service in your browser, you can temporarily comment out the parameter requiring the headers ['origin', 'x-requested-with'] from your proxy.js.

* In order to fix it, you must just comment the requireHeader from proxy.js.

PS: Wanted to take a screenshot of the “require Headers” error for a proper documentation, so uncommented the line back, but it works just fine now... I have no idea what’s going on. :)

*Quiz related. (got 6/12 points on this, as I thought the development is much recommended to be done on the WEB IDE, not locally, haha)*

*Unit 4*

Note: Always check for available controls/etc for your ui5 project version. You can change the version in top right of ui5.sap.com to see relevant content.

**There are 2 versions of UI5:** OpenUI5 (open-source version of SAPUI5) and SAPUI5 (has additional libraries like charts, or SAP Fiori elements)

SAP Fiori elements = apps that are automatically configured by the OData annotations from the backend.

*Unit 5*

**Access UI5 Diagnostics:** CTRL + ALT + SHIFT + S

**Inspect an element** by holding CTRL + ALT + SHIFT and clicking on the element

**Support Assistant:** CTRL + ALT + SHIFT + P *(seems like this basic app template doesn’t have it)*

**Extra:** Google extension – UI5 Inspector

**Linting** is a process that can be used for analyzing our JavaScript code. It can help us to avoid typos and syntactical errors.

SAPUI5 and the SAP Web IDE use ESLint for this purpose.

ESLint is a very flexible, open source tool, with lots of community support on the web.

As hidden files, inside SAP Web IDE, we have: .eslintrc, .eslintrc.ext

They can be used now to also set up ESLint for local development, or inside your continuous integration build, and you will have the same rules checked as in your SAP Web IDE.

*Quiz related. (got 6/6 points on this)*

*Unit 6 (and the last one)*

To access **Rufus**, we need to load this script:

<script src="https://sap.github.io/openSAP-ui5-course/Rufus.js"></script>

Then, we can **click** on the **Rufus mascot** or press **F9**

*Quiz related. (got 6/9 points on this)*

*Week 2*

*Unit 1*

MVC = Model View Controller scheme

Diagram

Description automatically generated

In order to check if your code is good according to their example/code, you must load Rufus script.

As for the exercise, this is what I’ve done:

1. I’ve put the <Link> in <content>, and link appeared. (checked the ui5.sap.com)
2. Checked with Rufus, but asked for a <Toolbar>. (checked the ui5.sap.com)
3. Put the link inside the toolbar, and everything was accepted.

(seen then, in Unit2, that the toolbar was placed in the footer, in their e.g., so I did it too)

**Finds:** <ToolbarSpacer/> put before and after the content of the toolbar, will center what’s inside.

*Unit 2*

UI Element / Control => UI definition and behavior encapsulated as a reusable asset

It’s an independent part of the UI, which is independent from the rest and can placed in other components.

For each control, several configurations are available:

* each can have properties, aggregations, associations and events.

**When putting in the Labels and Search Fields**, beware to not add the <ListItem>s, if you are copying the code from the online Material. Always try to get a similar code like the one from the video.

**Just a reminder:** An aggregation is just like the <f:content> placed in our <f.SimpleForm>

Most UI5 controls are responsive out-of-the-box.

**Regarding the MessageToast of the search button:**

**(better say: Regarding a content which might or might not be accessed)**

We can preload it in the sap.ui.define(), but we don’t know if the user will actually click on the button, so we can load our MessageToast asynchronously, using the sap.ui.require() function. (reduces the size of the app and potentially improves the loading time)

**Setting up the debugger** – index.html

<data-sap-logLevel="debug">

Then, in the controller, we load it sap.base.Log, put it as a parameter, and then use it’s method/s:

Log.info( “my message” );

We can use the logs in functions like: onInit, onExit, onBeforeRendering, onAfterRendering.

Quiz related. *(got 6/6 points on this)*

*Unit 3*

We use {movies>/initDate} with the / as it is a top node.

We use complex binding as well {path: movies>/initDate, formatter: ‘.formatter.formatDate’}

When using info that is related to the top nodes, we don’t need the / anymore.

**PS:** If you gonna write all the bindings yourself, to get some practice, be aware to not write “startData” instead of “startDate” like I did :)

**We just did a nested aggregation binding:** the row is an aggregation of the calendar, and the appointments is an aggregation of the row.

Aggregation binding is one of the three binding types:

1. aggregation binding for lists,
2. element binding for entities,
3. property binding for properties of an entity.

Graphical user interface, text, application, chat or text message

Description automatically generated

**Aggregation binding** can be used to automatically create a list of child controls according to data in the model.

The control defined for the aggregation in the XML view, is just a template. It will be cloned as many times as needed based on the data occurrence in the model.

**Property binding** allows connecting properties of a control to one or more paths in the model data.

(in this visualization, the page titles are bound like this)

**Element binding** allows us to bind elements to a specific control so that all children can also bind their attributes or aggregations in the same context

(often useful for container controls in the main area of the app)

**Expression binding** when we make certain content available only on certain devices. (is it?!)

It’s an enhancement of the UI5 binding syntax and allows to define simple logic directly in the XML views.

Expression binding starts with = in the curly braces.

We add the ! to show an exception. (so it will be {=!})

$ for an object => so it will be {=!${device>/system/phone}}

Like: numberState=”{=${invoice>ExtendedPrice} > 20 ? ‘Error’ : ‘Success’ }”

(very basic formatting, such as setting colors for ‘Success’ to ‘Error’)

((more complex formatting should be implemented with a formatter))

Diagram

Description automatically generated

Quiz related. *(got 6/9 points on this)*

*Unit 4*

Diagram

Description automatically generated

We’ve learnt that when we want data binding, we use $, right?

But we use the $ when we are “calling” an UI5 model which we didn’t create. (exists by default)

<Flexbox wrap=”Wrap”> container (responsive)

<l:VerticalLayout>

**Recommendation:** Use padding class for the containers, and margin class for the controls.

**Note:** The order declared in sap.ui.define( [] must always match the order the parameters are put.

As for the exercise, I’ve noticed the mention in “Detail.controller” of “NotFound”

UIComponent.getRouterFor(this).getTargets().display("NotFound");

1. So I’ve created a new control called NotFound.view.js (firstly I tried with .fragment), and put there the base code.
2. Searched on ui5.sap.com for “message” and looked on the finds. There was this <IllustratedMessage> which I thought I can just simply attach a photo to it and that’s it.
3. Actually, just by inserting it, with no properties/attributes, I was getting the wanted result.

Graphical user interface, text, application

Description automatically generated

1. Added the showNavButton=”true” on the <Page> for the navigating back possibility, so needed a controller too.

Put everything up together, but the button triggers nothing, just like for the Detail.view from the openSAP example.. I tried different destination names than “Home”, like “App” or “TargetApp” but nothing. No errors, no nothing. (and died with it)

Here is my onNavBack function:

onNavBack : function () {

var oHistory = History.getInstance();

var sPreviousHash = oHistory.getPreviousHash();

if (sPreviousHash !== undefined) {

window.history.go(-1);

} else {

var oRouter = this.getOwnerComponent().getRouter();

oRouter.navTo("App", {}, true);

}

}

! In order to pass Rufus, you must use MessagePage. Quiz related. *(got 6/6 points on this)*

*Unit 5*

Key Product Standards

* User Experience consistency
* Accessibility (high-contrast themes, keyboard handling, screen reader support)
* Globalization
* Security

All UI5 controls are designed according to design guidelines, the SAP Fiori Design Guidelines.

ARIA = Accessible Rich Internet Application

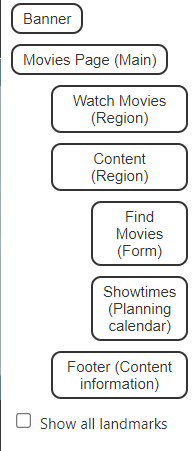
Recommending that all pages are withing a landmark.

The page control can hold an aggregation with the name landmarkInfo

Graphical user interface, treemap chart

Description automatically generated

This is what we’ve got after adding the landmarks Info, vs the old one.

Graphical user interface, text, application, chat or text message

Description automatically generated

Quiz related. *(got 6/9 points on this; didn’t know the code of Mexican Spanish)*

*Week 3*

*Unit1*

Fiori Design Guidelines main layouts: dynamic and flexible column.

Flexible column => it’s a solid base and has options for more extensibility for our application later on.

The main functionality of the SemanticPage is to predefine the placement, behavior and styles of the page elements.

Content specified in the semantic aggregations will be automatically positioned in dedicated sections of the footer or the header of the page.

**As for the exercise: I couldn’t make it.**

I have read the hints, and googled for Avatar control. After some tries to add it near the Title “Orders” we already had, to see it displayed, I’ve managed to add it.

I realize I’ve had huge gaps when about positioning controls in the proper containers, as I’ve firstly tried with <Toolbar> <Panel>, because <semantic:titleHeading> was accepting only one control inside (and more nested inside it?!).

Googled up and down the SemanticPage control, tried to understand the way it works, but couldn’t.

Seen all the aggregations, but I just didn’t know how to properly use them.

**After watching the solution provided by the course repo, I’ve understood, and I hope for good.**

Quiz related. *(got 3/9 points on this) pretty bad start today huh?! Haha. Live to improve.*

*Unit2*

OData = Open Data Protocol

It is a RESTful protocol that is standardized at OASIS.

It defines a set of best practices for developing and building RESTful APIs.

OData helps you focus on business logic while you are building your RESTful APIs.

You don’t have to worry about the various approaches to define request and response headers, status codes, HTTP methods, and so on.

OData $metadata

* **Is a machine-readable description of the data model of the APIs.**
* **Decouples the backend from frontend, by providing a clear description of its resources, like collections, entities and properties. This allows us to consume and use additional information about the service centrally.**

**We can load the metadata file in the browser by entering the OData service URL followed by the $metadata keyword.**

**Diagram

Description automatically generated**

NavigationProperty defines a link from one EntitySet to another.

(this can be compared to relationships in a database)

! To see what properties are available in the BuisnessPartnerSet, we go to $metadata and search for EntityContainer. Here we find the BusinessPartnerSet, which is of type BusinessPartner. Sooo, we search for this entity type, and will see it’s available properties.

Batching is enabled by default in OData v2.

! Disabling batching and the query operators, like filtering, sorting, etc. and accessing the properties of the OData service via URL are explained in detail in their previous course and in their online tutorials

**OData v4**

* Relatively new standard

**Improvements:**

1. **Performance**
   1. The new JSON data format payload size is reduced by 10%-60% after compression.

(by removing the inline metadata)

* 1. Cross-Service Navigation => metadata of large service models are split into smaller files which can be loaded lazily when required.

(so, OData v4 loads part of data only when it is needed!)

1. **Features** (at least some of them; for more, check odata.org)
   1. Support for new data types, especially in the date/time area.
   2. Support for inheritance and collection values.
   3. The $filter query parameter enables support for lambda operators, like “any” and “all”, which allows query options to further refine the data request for navigation properties.
   4. For the Analytical Applications, the most interesting feature is the $apply.

It allows you to provide well defined data aggregation requests which can be the basis for a much better analytical solution.

OData v4 tutorial: <https://sapui5.hana.ondemand.com/sdk/#/topic/bcdbde6911bd4fc68fd435cf8e306ed0>

Quiz related. *(got 12/12 points on this)*

*Unit3*

Quiz related. *(got 7.5/9 points on this)*

*Unit4*

New fancy features which the SAPUI5 framework offers.

1. Stickiness
2. Floating Footer of the semantic page
3. Dynamic Header of the object page layout
4. InfoLabel and StatusIndicator – which goes with values 0-100 on a SVG – fills it up according to the value

Quiz related. *(got 5/6 points on this)*

*Unit5*

UI5 comes with 5 major concepts targeting reuse on different levels.

**Control**

* Encapsulates Rendering and Behavior
* Standardized Control Interface
* Smallest Reusable UI Asset
* Custom or Composite Control
* Bundled in Control Libraries

**Fragment**

* Declarative division in XML
* Light-weight reusable UI part
* No controller or behavior
* No footprint at runtime

**View**

* Declarative Definition in XML
* Comprehensive reusable UI part
* Defines a larger UI area and behavior
* Typically connected to a controller

**Component**

* Independent and reusable part
* Predefined appearance and functionality
* Application container
* Reuse component

**Library**

* Collection of reusable UI and code artifacts
* Most common use case: Control Library
* Consumed in views or components
* Supports Theming and Building

XML composite control

* Declarative UI definition in XML
* Special $this model for binding properties
* Control metadata description
* Special about XML composite:

$this => refers to the properties of the item we saw in the metadata

Is reflecting the managed object model which is created underneath in the XML composite implementation.

It is the connection between the .js and the .control.xml file, and provides easy access to all those properties and aggregations.

Quiz related. *(got 6/6 points on this)*

*Week 5*

*Unit1*

TwoWay binding = as we want to have a transfer of information from the model to the controls, and back from the controls to the model.

The default mode is “OneWay”