**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)*