

openSAP Evolved Web Apps with SAPUI5

Week 1 Unit 3: Local Setup with the UI5 Tooling

Exercises

PUBLIC



TABLE OF CONTENTS

LOCAL SETUP WITH THE UI5 TOOLING 3

Working with the UI5 Command Line Interface (CLI) 4

Connecting to a Service 5

RELATED MATERIAL.....10

LOCAL SETUP WITH THE UI5 TOOLING

Summary

In this unit you will learn how to develop, run and build a UI5 application in local development environment with a template app cloned from a GitHub repository. For this, you will use open source tools like node.js, git and an editor. You will establish a connection to an external OData service and retrieve data about directions in your app.

Preview

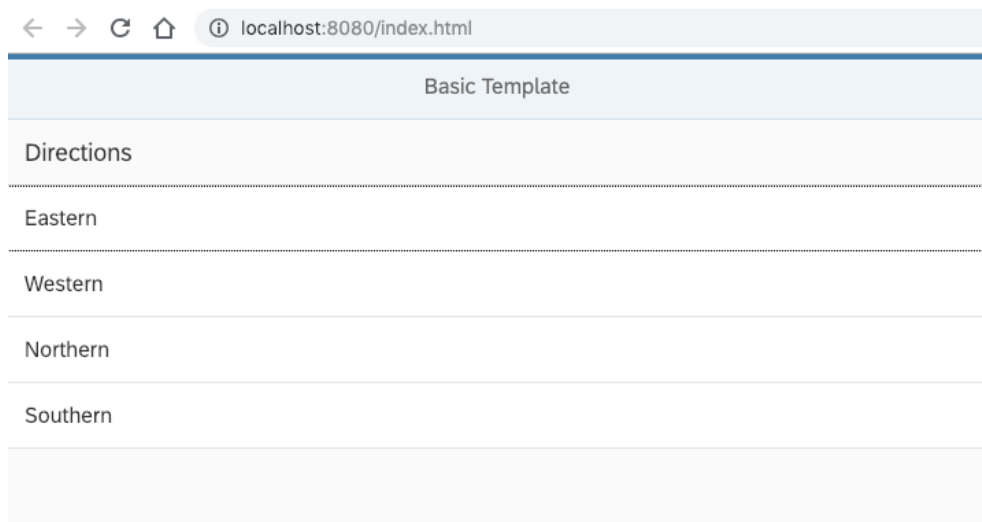


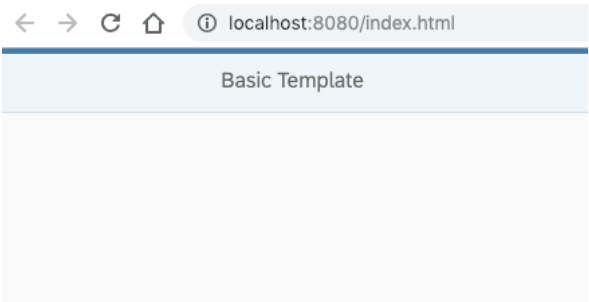
Figure 1: Preview of the Result

Prerequisites

You have installed:

- a code editor you feel familiar with
- Node.js (<https://nodejs.org/download>)
- Git (<https://git-scm.com/downloads/>)

Working with the UI5 Command Line Interface (CLI)

Explanation	Commands
<p>1. Open a new terminal window, and make sure all prerequisites are installed properly by checking the versions.</p> <p>Note: Please make sure that at least the following versions of the required software are installed:</p> <ul style="list-style-type: none"> Node.js: 8.5 npm: 5 Git: 2.1 	<pre># Verify installations node -v npm -v git --version</pre>
<p>2. Install the UI5 Tooling.</p> <p>Note: You need a working internet connection for this step.</p>	<pre>npm install -global @ui5/cli # Verify installation ui5 --help</pre>
<p>3. Clone the UI5 template project from GitHub and enter its directory</p>	<pre>git clone https://github.com/SAP/openui5- basic-template-app/ cd openui5-basic-template-app</pre>
<p>4. Let npm install the necessary dependencies.</p> <p>Note: Ignore all deprecation Warnings.</p>	<pre>npm install</pre>
<p>5. Run the local development server. Triggered by the <code>-o index.html</code>, argument a browser window should open showing the application.</p> <p>Note: The default URL of the server is: http://localhost:8080/index.html</p>	<pre>ui5 serve -o index.html</pre>  <p>The screenshot shows a web browser window with the address bar displaying 'localhost:8080/index.html'. The page content shows a header with the text 'Basic Template'.</p>

Connecting to a Service

To illustrate how to connect to backend systems in this scenario, we are going to consume a publicly available demo service and show a list of directions in our app.

webapp/view/Home.view.xml

```
<mvc:View
  controllerName="sap.ui.demo.basicTemplate.controller.Home"
  displayBlock="true"
  xmlns="sap.m"
  xmlns:mvc="sap.ui.core.mvc">
  <Page
    id="page"
    title="{i18n>title}">
    <content>
      <List headerText="Directions">
        <StandardListItem/>
      </List>
    </content>
  </Page>
</mvc:View>
```

To be able to bind some data in your application, you start by adding a `sap.m.List` to your Home view. Start your editor, and open the `webapp/view/Home.view.xml` file. Add the yellow passages to the existing file.

webapp/manifest.json

```
{
  "_version": "1.12.0",
  "sap.app": {
    "id": "sap.ui.demo.basicTemplate",
    ...
    "ach": "ach",
    "dataSources": {
      "mainService": {
        "uri": "https://services.odata.org/V2/Northwind/Northwind.svc/",
        "type": "OData",
        "settings": {
          "odataVersion": "2.0",
          "localUri": "localService/metadata.xml"
        }
      }
    }
  },
  ...
  "sap.ui5": {
    "rootView": {
      "viewName": "sap.ui.demo.basicTemplate.view.App",
      "type": "XML",
      "async": true,
      "id": "app"
    },
    ...
    "models": {
      "i18n": {
        "type": "sap.ui.model.resource.ResourceModel",
        "settings": {
          "bundleName": "sap.ui.demo.basicTemplate.i18n.i18n"
        }
      },
      "data": {
        "type": "sap.ui.model.odata.v2.ODataModel",
        "dataSource": "mainService",
        "preload": true,
        "settings": {
          "useBatch": false
        }
      }
    }
  },
  ...
}
```

You now have a *list*, but no data is shown, because you haven't bound any values. Therefore you enhance your application by adding a model via the `manifest.json`. You have to define a `dataSource` first, which will then be used in the model's description.

The setting for `preload` optimizes the loading time of the model. `useBatch` is set to `false`, because the service doesn't support it. Normally, you should make use of batch calls, but this is explained in detail in another unit.

webapp/view/Home.view.xml

```
<mvc:View
  controllerName="sap.ui.demo.basicTemplate.controller.Home"
  displayBlock="true"
  xmlns="sap.m"
  xmlns:mvc="sap.ui.core.mvc">
  <Page
    id="page"
    title="{i18n>title}">
    <content>
      <List headerText="Directions" items="{data>/Regions}">
        <StandardListItem title="{data>RegionDescription}"/>
      </List>
    </content>
  </Page>
</mvc:View>
```

Now add the data binding to the Home view.

proxy.js

```
var cors_proxy = require('cors-anywhere');

// Listen on a specific IP Address
// 0.0.0.0 equals localhost
var host = '0.0.0.0';

// Listen on a specific port, adjust if necessary
var port = 8081;

cors_proxy.createServer({
  originWhitelist: [], // Allow all origins
  requireHeader: ['origin', 'x-requested-with'],
  removeHeaders: ['cookie', 'cookie2']
}).listen(port, host, function() {
  console.log('Running CORS Anywhere on ' + host + ':' + port);
});
```

You might have noticed that the app reports a technical error when you run it at this point in time. This is due to the browser blocking the CORS request. Therefore, you will introduce a CORS proxy, which will provide the network requests with the necessary CORS headers. Create a new script in the project folder which makes use of CORS Anywhere.

With this script, the proxy listens to `localhost:8081`. To learn more about the other settings, please refer to the CORS Anywhere documentation.

package.json

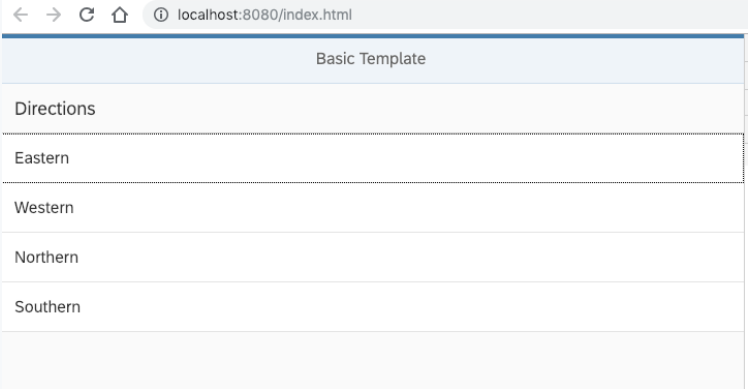
```
{
  "name": "openui5-basic-template-app",
  "version": "0.1.0",
  "description": "Best-practice starting point for building a freestyle app",
  "private": true,
  "scripts": {
    "start": "ui5 serve",
    ...
    "proxy": "node proxy.js"
  },
  ...
},
{
  "devDependencies": {
    "@ui5/cli": "^0.2.9",
    "chrome-headless-launcher": "^0.1.5",
    "cors-anywhere": "^0.4.1"
    ...
  }
}
```

The CORS Anywhere dependency needs to be declared in the `package.json`, and the new script needs to be declared there, to make it available as an `npm` command later on. Run `npm install` on the command line.

webapp/manifest.json

```
{
  "_version": "1.12.0",
  "sap.app": {
    "id": "sap.ui.demo.basicTemplate",
    ...
    "dataSources": {
      "mainService": {
        "uri": "http://localhost:8081/services.odata.org/V2/Northwind.svc",
        "type": "OData",
        "settings": {
          "odataVersion": "2.0",
          "localUri": "localService/metadata.xml"
        }
      }
    }
  },
  ...
}
```

Finally, after you added the proxy's address to your service URI, you should start it and see it in action.

Explanation	Commands
1. Start the proxy in another terminal window, so both (server and proxy) are running.	<code>npm run proxy</code>
2. Switch to the browser window and refresh to see the list populated with data from the service. Note: If the <code>cors-anywhere</code> proxy doesn't work as expected and you still see the CORS technical error in the console, try to use a different proxy like <code>local-cors-proxy</code> .	

package.json

```
{
  "name": "openui5-basic-template-app",
  "version": "0.1.0",
  "description": "Best-practice starting point for building a freestyle app",
  "private": true,
  "scripts": {
    "start": "ui5 serve",
    ...
    "local-proxy": "lcp --proxyUrl https://services.odata.org/V2/Northwind/Northwind.svc/"
  },
  ...
  "devDependencies": {
    "@ui5/cli": "^0.2.9",
    "chrome-headless-launcher": "^0.1.5",
    "local-cors-proxy": "^1.0.2",
    ...
  }
}
```

Add the script and the dependency to the `package.json` file, and execute the command `npm install` again to install the new dependency. Now execute the command `npm run local-proxy`.

The proxy will start on port 8010.

webapp/manifest.json

```
{
  "_version": "1.12.0",
  "sap.app": {
    "id": "sap.ui.demo.basicTemplate",
    ...
    "dataSources": {
      "mainService": {
        "uri": "http://localhost:8010/proxy",
        "type": "OData",
        "settings": {
          "odataVersion": "2.0",
          "localUri": "localService/metadata.xml"
        }
      }
    }
  },
  ...
}
```

Follow the instructions on the terminal and replace your service `uri` with `http://localhost:8010/proxy`. Now you should see your *list* showing some data from the service

RELATED MATERIAL

- UI5 Tooling: <https://github.com/SAP/ui5-tooling>
- Node.js: <https://nodejs.org/en/docs>
- Git: <https://git-scm.com/docs>
- Visual Studio Code: <https://code.visualstudio.com>
- CORS Anywhere: <https://www.npmjs.com/package/cors-anywhere>
- local-cors-proxy: <https://www.npmjs.com/package/local-cors-proxy>
- [Demo Kit: Request Fails Due to Same-Origin Policy \(Cross-Origin Resource Sharing - CORS\)](#)

Coding Samples

Any software coding or code lines/strings ("Code") provided in this documentation are only examples and are not intended for use in a production system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages caused by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

www.sap.com/contactsap

© 2018 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. See <http://www.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.