VueCliMiddleware unter IIS installieren

Dies ist eine Fortsetzung zu

https://hosting149637.a2ebb.netcup.net/devandy.de/vuecli-installieren/

1. Über Windows Cmd Console zum Ordner navigieren wo das neue Zertifikat erstellt werden soll (cd…)
2. in die Console folgendes eingeben: mkcert -pkcs12 localhost 127.0.0.1
3. Im Ordner wird nun eine localhost+1.p12 Datei erstellt. Diese Datei umbenennen zu localhost.pfx
4. In Die Console mkcert – install eingeben und mit Ja bestätigen
5. Die localhost.pfx Datei doppelt anklicken und „Lokaler Computer auswählen“. Ganz wichtig. Sonst funktioniert es nicht
6. Bei Kennwort „changeit“ eingeben und „Schlüssel als exportierbar machen“ anhaken
7. Zu IIS wechseln, ASP Core App mit VueCliMiddleware auswählen, und auf der rechten Seite auf „Bindungen…“ klicken.
8. Doppelt auf die https Sitebindung klicken
9. Auf den Button „Auswählen“ klicken und gerade installiertes Zertifikat auswählen und mit OK bestätigen
10. Bei Hostname noch localhost eintragen
11. Zur VueApp wechsel und in die VueConfig folgendes eintragen:
12. [crayon-6768287d43f45130844998/]
    Der Port 8080 sollte gleich sein, wie in VueCliMiddleware angegeben. Die localhost.pfx Datei muss im angegebenen Pfad vorhanden sein

Hot reloading funktioniert leider noch nicht

JavaScript CSV Download – Excel + Umlaute

```
var downloadCsv = function(content, fileName) {
var a = document.createElement('a');
mimeType = mimeType || 'application/octet-stream';

var mimeType = 'text/csv;encoding:utf-8';
var BOM = "\uFEFF";
content = BOM + content;

if (navigator.msSaveBlob) { // IE10
navigator.msSaveBlob(new Blob([content], {
type: mimeType
}), fileName);
} else if (URL && 'download' in a) { //html5 A[download]
a.href = URL.createObjectURL(new Blob([content], {
type: mimeType
}));
a.setAttribute('download', fileName);
document.body.appendChild(a);
a.click();
document.body.removeChild(a);
} else {
location.href = 'data:application/octet-stream,' + encodeURIComponent(content); // only this mime type is supported
}
```

```
// Example data given in question text
var data = [
['A', 'B', 'C', 'Größe', 'Maße', 'ÄÜÖ']
, ['C', 'D', 'E', 'F', 'G', 'H']
];

// Building the CSV from the Data two-dimensional array
// Each column is separated by ";" and new line "\n" for next row
var csvContent = '';
data.forEach(function(infoArray, index) {
dataString = infoArray.join(';');
csvContent += index < data.length ? dataString + '\n' : dataString; }); downloadCsv(csvContent, 'MyFilename.csv', );
```

VueCli installieren

Begriffe Definition
Babel
Ein JavaScript Kompiler. Übersetzt Code in EcmaScript, welcher auch mit alten JavaScript Browsern kompatibel ist und vieles mehr
PostCss
Beim kompilieren werden Prefixe an Css Klassen / Ids hinzugefügt, damit in einer SinglePage Applikation nicht plötzlich ein und dieselbe Id für unterschiedliche Komponente genutzt werden
Lint
Ein Prozess wo der Source Code nach potentiellen Fehler durchsucht wird
package.json
Beschreibt die Abhängigkeiten (dependencies) inkl. Version zu anderen Bibliotheken

Jede App in NodeJs hat im Root Verzeichnis der App so eine Datei.
Beim Abrufen von einer App, werden die Abhängigkeiten automatisch mit herunter geladen
sass / less
Dateien werden von z.B. Webpack zu einer Css Datei intepretiert
TypeScript (.ts)
Dateien werden von z.B. Webpack zu einer Js Datei intepretiert
Dadurch sind z.B. Klassen und Vererbungen in JavaScript möglich

NodeJs
Ist eine JavaScript Serverumgebung
Npm
Node-package-manager verwaltet Node Applikationen (packages)
Vue.js
Vue.js ist ein clientseitiges JavaScript-Webframework zum Erstellen von Single-Page-Webanwendungen
Vuex
Nutzt man mehrere Komponenten, die untereinander noch verschachtelt sind, dann müssen Events immer nach oben gereicht werden.
Mit Vuex wird ein Datenstore geschaffen, der alle Daten abruft und bereit stellt. Dadurch erhält man eine Zustandsmaschine
Webpack
Module Bundler. Packt mehrere JavaScript, Css usw. Dateien zu einer minified zusammen
VueCLI
Vue ist das JavaScript Framework und CLI steht für Command Line Interface

Installation VueCLI

1. NodeJs installieren x64 -> https://nodejs.org/en/download/
von nun an werden die Befehle in der NodeJs cmd eingegeben.
Node.js Command prompt:
C:\Program Files\nodejs\nodevars.bat

2. Vue Cli installieren -> npm i -g @vue/cli
Die Apps werden hier gespeichert: %UserProfile%/node_modules

3. vue ui -> startet den Service für Vue Cli Ui Interface http://localhost:8000/dashboard

4. Vue global startbar machen
npm install -g @vue/cli-service-global

5. In PowerShell mkcert installieren
choco install mkcert
Damit erstellt man für die eigene Entwicklungsmaschine ein SSL Zertifikat. Zertifikate von CA können nicht mit localhost arbeiten und Self Signed werden von Applikationen nicht anerkannt
6. Über cd [… Pfad zu meiner Applikation] navigieren
7. mkcert -install
8. Ipconfig -> Eigene IP rausfinden (IPv4-Adresse . . . .
9. (Durch eigene IPv4 ersetzen) mkcert localhost 127.0.0.1 192.168.178.47 ::1

Ordnerstruktur Vue App
node_modules Ordner
Hier liegen alle Bibliotheken (Modules), die für das Vue Projekt benötigt werden
public/Index.html
Die index.html ist die einzige Html Seite, die an den Client übertragen wird. Der Inhalt wird in sie rein gerendert. Und zwar hier zwischen:

Sobald man buildet, werden dort auch 2 Skript Tags generiert, die den JavaScript Teil enthalten
src/assets (Engl. Teile, Zubehör)
Hier liegen die Bilddateien, Fonts etc.
src/components
Hier liegen eigene Vue Komponente
src/App.vue

Beispiel Komponente
src/main.js
Diese JavaScript Datei ist die Haupt Datei, die sagt, nehme die und die JavaScript Komponente und rendere sie in Vue hinein
.gitignore
Regeln für Git, welche Dateien in das Projekt kommen und bei Änderungen hochgeladen werden sollen
babel.config.js
Einstellung von Babel, z.B welche Version von EcmaScript genutzt werden soll
package.js
Konfigurationsdatei vom Vue Projekt.
• Name, Version
• „scripts“ welcher NodeJs Service beim builden, deployen und linten genutzt werden soll
• „dependencies“ welche Bibliotheken in welcher Version genutzt werden sollen
• „devDependencies“ welche Bibliotheken ausschließlich im Entwicklungsmodus genutzt werden sollen
Postcss.config.js
Konfigurationsdatei für PostCss
README.md
Dokumentationsdatei. Wird in GitHub z.B. als Doku angezeigt: https://guides.github.com/features/mastering-markdown/
Tsconfig.json
Konfigurationsdatei für TypeScript
Tslint.json
Konfigurationsdatei für TypeScript Lint

dist Ordner
Mit dem Befehl „npm run build“ wird die App deployed und in das Verzeichnis dist gepackt. Auf einen produktiven Server kommt also nur der Inhalt vom dist Ordner

ASP Core 3.1 inkl VueCLI mit mehreren VueApps verwenden

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.SpaServices;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.FileProviders;
using Microsoft.Extensions.Hosting;
using VueCliMiddleware;

namespace ASPMultipleVueSPA
{
public class Startup
{
// This method gets called by the runtime. Use this method to add services to the container.
// For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940
public void ConfigureServices(IServiceCollection services)
{
}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
```

```
{
if (env.IsDevelopment())
{
app.UseDeveloperExceptionPage();
}

app.UseRouting();

app.UseEndpoints(endpoints =>
{
endpoints.MapGet(„/“, async context =>
{
await context.Response.WriteAsync(„Hello World!“);
});

/*
* // Wichtig: In vue.config.js publicPath eintragen! z.B.
module.exports = {
publicPath: ,/en/‘
};
*/

endpoints.MapToVueCliProxy(
pattern: „en/{*path}“,
options: new SpaOptions { SourcePath = „ClientApp“ },
npmScript: (System.Diagnostics.Debugger.IsAttached) ? „serve“
: null,
port: 8080,
https: false,
runner: ScriptRunnerType.Npm,
regex: „Compiled successfully“,
forceKill: true,
wsl: false
);

endpoints.MapToVueCliProxy(
pattern: „fr/{*path}“, // Wichtig: In vue.config.js publicPath
eintragen: publicPath: ,/fr/‘
```

```
options: new SpaOptions { SourcePath = „ClientApp2“ },
npmScript: (System.Diagnostics.Debugger.IsAttached) ? „serve“
: null,
port: 8081,
https: false,
runner: ScriptRunnerType.Npm,
regex: „Compiled successfully“,
forceKill: true,
wsl: false
);

});

}
}
}
```

Backup alle Nicht System Datenbanken

[crayon-6768287d442cd196186805/]
Quelle:
https://www.sqlshack.com/multiple-methods-for-scheduling-a-sql-server-backup-automatically/

Openmediavault + Nextcloud

auf Raspberry Pi installieren

Installation erfolgt von einem Windows Rechner.

Benötigt werden folgende Programme:

1. SD Card Formatter: https://www.sdcard.org/downloads/formatter/
2. Win32 Disk Imager: https://sourceforge.net/projects/win32diskimager/
3. Raspberry Pi OS (32-bit) Lite Image: https://www.raspberrypi.org/downloads/raspberry-pi-os/
4. KiTTY oder PuTTY (SSH Client) [Im folgenden Konsole genannt]: https://www.fosshub.com/KiTTY.html

Vorbereitung:

1. Mit SD Card Formatter SD Karte formatieren
2. Mit Win32 Disk Imager „Raspberry Pi OS (32-bit) Lite Image“ auf SD Karte schreiben
3. Auf der SD Karte im Root Verzeichnis eine Datei, mit Dateinamen „ssh“(ohne Dateiendung) anlegen
4. SD Karte in Raspberry schieben, RasPi neustarten und IP Adresse über Router herausfinden
5. In der Konsole mit „pi“ und „raspberry“ einloggen

Installation Webmin + Nextcloud:
[crayon-6768287d44442287737945/]
Web einloggen https://192.168.178.45:10000/
mit pi und raspberry

Servers -> MySQL Database Server -> „Create new database“ -> „nextcloud“
-> „User Permissions“ -> oben „Create new user“ ->
Radio auf leer und username eingeben „nextclouduser“
Password „Set to“ und password eingeben
Hosts Radio auf leer und „localhost“ eingeben
Permissions alles markieren

Navigations Menü: „Others“ -> „File Manager“ -> etc -> openal -> php -> 7.3 -> apache2 -> php.ini (r.Maustaste) -> editieren -> memory_limit 2048 -> oben rechte Save Diskette
-> home -> pi -> „File“ -> „Download from remote URL“ -> nextcloud.com/de/install/#instructions-server (r.Maustaste) auf download -> link kopieren
download -> r.Maustaste auf Datei und „Extract“
Ins Verzeichnis gehen -> Alle markieren -> Cut
var -> www -> html -> nextcloud Ordner erstellen und reinkopieren
html Ordner r.Maustaste -> Properties -> Change ownership -> Username, Group: „www-data“, recursive -> true

Navigations Menü: „System“ -> „Bootup and Shutdown“ -> apache2 -> „Restart“

####
Apache Port ändern auf z.B. 81. Damit OpenMediaVault auf Port 80 und 444 SSL installiert werden kann. Dies wird später zurück geändert
/etc/apache2/ports.conf

Nextcloud installieren
http://192.168.178.45/nextcloud/index.php

Installation OpenMediaVault:

1. [crayon-6768287d44445986728659/]
2. OpenMediaVault ist nun installiert. Wenn man die IP Adresse im Browser aufruft, sollte die WebUi erscheinen. Einzuloggen mit „admin“ und „openmediavault“
3. Standard Passowort ändern: Generel Settings -> Web Administrator Password
4. SSH Zertifikat: Certifikates -> Tab SSH -> Add -> Irgendetwas eingeben. z.B. SSHCert -> Apply
5. SSL Zertifikat: Certifikates -> Tab SSL -> Add ->

ausfüllen -> Apply
6. Generel Settings: Port -> 90, Auto logout -> disabled, Enable SSL auf true, Zertifikat auswählen, Force SSL auf true -> save & apply [WebUi ist ab jetzt nur mit https erreichbar]

In Webmin Port von Nextcloud zurück stellen:

/etc/apache2/ports.conf auf Port 80 bzw. 443

#Laufwerk dauerhaft einbinden:

sudo mount -t ext /dev/sda1 /media/Hdd1

https://confluence.jaytaala.com/display/TKB/Mount+drive+in+linux+and+set+auto-mount+at+boot

---

ASP Core Configuration vererben

public IConfiguration Configuration { get; }

in ServiceCollection

services.Configure<BaseConfiguration>(this.Configuration);
services.Configure<ChildConfiguration>(this.Configuration);

Die ChildConfiguration erbt von BaseConfiguration

Hat die ChildConfiguration Unterknoten, erben auch die Unterknoten. So wird die Konfiguration „erweitert“

Integrations Test mit Dependency Injection MsTest v2 .Net Core 3.1

Möchte man einen Integrationstest schreiben und dabei dependency injection nutzen, muss man folgendermaßen vorgehen:

1. Im Besten Fall teilt ma die Ausführende Applikation z.B. Web Applikation, Console etc und eine Bibliothek.
2. In die Bibliothek kommt die Auslagerung der StartUp: [crayon-6768287d445e4133598153/]
3. Im Testprojekt wird eine Basis Klasse definiert, die dieses Modul einliest. Nun kann in CofigureServices services.UseMyModule() aufgerufen werden oder der Host selbst gebaut werden: [crayon-6768287d445e7762627302/]
4. Die eigentliche Test Klasse erbt nun von diesen BaseUnitTest. Nun kann ein Propertiy erzeugt werden, welche einen Service aus der Dependency Injection ausliest: [crayon-6768287d445e8354323574/]

SQL CPU Last und Gesamtdauer

von Query testen

[crayon-6768287d447b0691802082/]

.gitignore SSIS dtsx Dateien hinzufügen

in der VisualStudio .gitignore: https://www.gitignore.io/api/visualstudio

wird das komplette Verzeichnis „obj“ exkludiert. Die packages (.dtsx) aus SSIS befindet sich aber gerade dort.

Daher muss man folgende Zeile auskommentieren:
[crayon-6768287d44919122581869/]
und unter diesen Block einen neuen Block hinzufügen:
[crayon-6768287d4491c608792681/]
meine komplette .gitignore sieht demnach so aus:
[crayon-6768287d4491d170682638/]