Integration von Anwenderdefinierten Steuerungen in WinCC Unified (Custom Web Controls)

SIMATIC WinCC Unified V18

https://support.industry.siemens.com/cs/ww/de/view/109779176

Siemens Industry Online Support

Rechtliche Hinweise

Nutzung der Anwendungsbeispiele

In den Anwendungsbeispielen wird die Lösung von Automatisierungsaufgaben im Zusammenspiel mehrerer Komponenten in Form von Text, Grafiken und/oder Software-Bausteinen beispielhaft dargestellt. Die Anwendungsbeispiele sind ein kostenloser Service der Siemens AG und/oder einer Tochtergesellschaft der Siemens AG ("Siemens"). Sie sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit und Funktionsfähigkeit hinsichtlich Konfiguration und Ausstattung. Die Anwendungsbeispiele stellen keine kundenspezifischen Lösungen dar, sondern bieten lediglich Hilfestellung bei typischen Aufgabenstellungen. Sie sind selbst für den sachgemäßen und sicheren Betrieb der Produkte innerhalb der geltenden Vorschriften verantwortlich und müssen dazu die Funktion des jeweiligen Anwendungsbeispiels überprüfen und auf Ihre Anlage individuell anpassen.
Sie erhalten von Siemens das nicht ausschließliche, nicht unterlizenzierbare und nicht übertragbare Recht, die Anwendungsbeispiele durch fachlich geschultes Personal zu nutzen. Jede Änderung an den Anwendungsbeispielen erfolgt auf Ihre Verantwortung. Die Weitergabe an Dritte oder Vervielfältigung der Anwendungsbeispiele oder von Auszügen daraus ist nur in Kombination mit Ihren eigenen Produkten gestattet. Die Anwendungsbeispiele unterliegen nicht zwingend den üblichen Tests und Qualitätsprüfungen eines kostenpflichtigen Produkts, können Funktions- und Leistungsmängel enthalten und mit Fehlern behaftet sein. Sie sind verpflichtet, die Nutzung so zu gestalten, dass eventuelle Fehlfunktionen nicht zu Sachschäden oder der Verletzung von Personen führen.

Haftungsausschluss

Siemens schließt seine Haftung, gleich aus welchem Rechtsgrund, insbesondere für die Verwendbarkeit, Verfügbarkeit, Vollständigkeit und Mangelfreiheit der Anwendungsbeispiele, sowie dazugehöriger Hinweise, Projektierungs- und Leistungsdaten und dadurch verursachte Schäden aus. Dies gilt nicht, soweit Siemens zwingend haftet, z.B. nach dem Produkthaftungsgesetz, in Fällen des Vorsatzes, der groben Fahrlässigkeit, wegen der schuldhaften Verletzung des Lebens, des Körpers oder der Gesundheit, bei Nichteinhaltung einer übernommenen Garantie, wegen des arglistigen Verschweigens eines Mangels oder wegen der schuldhaften Verletzung wesentlicher Vertragspflichten. Der Schadensersatzanspruch für die Verletzung wesentlicher Vertragspflichten ist jedoch auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht Vorsatz oder grobe Fahrlässigkeit vorliegen oder wegen der Verletzung des Lebens, des Körpers oder der Gesundheit gehaftet wird. Eine Änderung der Beweislast zu Ihrem Nachteil ist mit den vorstehenden Regelungen nicht verbunden. Von in diesem Zusammenhang bestehenden oder entstehenden Ansprüchen Dritter stellen Sie Siemens frei, soweit Siemens nicht gesetzlich zwingend haftet.
Durch Nutzung der Anwendungsbeispiele erkennen Sie an, dass Siemens über die beschriebene Haftungsregelung hinaus nicht für etwaige Schäden haftbar gemacht werden kann.

Weitere Hinweise

Siemens behält sich das Recht vor, Änderungen an den Anwendungsbeispielen jederzeit ohne Ankündigung durchzuführen. Bei Abweichungen zwischen den Vorschlägen in den Anwendungsbeispielen und anderen Siemens Publikationen, wie z. B. Katalogen, hat der Inhalt der anderen Dokumentation Vorrang.
Ergänzend gelten die Siemens Nutzungsbedingungen (https://support.industry.siemens.com).

Securityhinweise

Siemens bietet Produkte und Lösungen mit Industrial Security-Funktionen an, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen.
Um Anlagen, Systeme, Maschinen und Netzwerke gegen Cyber-Bedrohungen zu sichern, ist es erforderlich, ein ganzheitliches Industrial Security-Konzept zu implementieren (und kontinuierlich aufrechtzuerhalten), das dem aktuellen Stand der Technik entspricht. Die Produkte und Lösungen von Siemens formen einen Bestandteil eines solchen Konzepts.
Die Kunden sind dafür verantwortlich, unbefugten Zugriff auf ihre Anlagen, Systeme, Maschinen und Netzwerke zu verhindern. Diese Systeme, Maschinen und Komponenten sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn und soweit dies notwendig ist und nur wenn entsprechende Schutzmaßnahmen (z.B. Firewalls und/oder Netzwerksegmentierung) ergriffen wurden.
Weiterführende Informationen zu möglichen Schutzmaßnahmen im Bereich Industrial Security finden Sie unter https://www.siemens.com/industrialsecurity.

Die Produkte und Lösungen von Siemens werden ständig weiterentwickelt, um sie noch sicherer zu machen. Siemens empfiehlt ausdrücklich, Produkt-Updates anzuwenden, sobald sie zur Verfügung stehen und immer nur die aktuellen Produktversionen zu verwenden. Die Verwendung veralteter oder nicht mehr unterstützter Versionen kann das Risiko von Cyber-Bedrohungen erhöhen.
Um stets über Produkt-Updates informiert zu sein, abonnieren Sie den Siemens Industrial Security RSS Feed unter https://www.siemens.com/cert.

Inhaltsverzeichnis

1 Allgemeine Informationen zu Custom Web Controls

WinCC Unified bietet eine Option für Custom Web Controls (oder kurz CWC)
Das bedeutet, dass Ihre Visualisierung nicht auf die Standard-Steuerung von WinCC Unified beschränkt ist.

Sie haben die Möglichkeit, eine eigene, benutzerdefinierte Steuerung mit Web-Technologie zu erstellen, oder bereits bestehende Steuerelemente zu verwenden und diese auf weitere Stationen von
WinCC Unified Runtime anzuwenden.

Dieses Anwendungsbeispiel soll Ihnen erklären, wie Sie eine mit Visual Studio Code erstellte "Custom Web Control" in WinCC Unified integrieren.

Hinweis

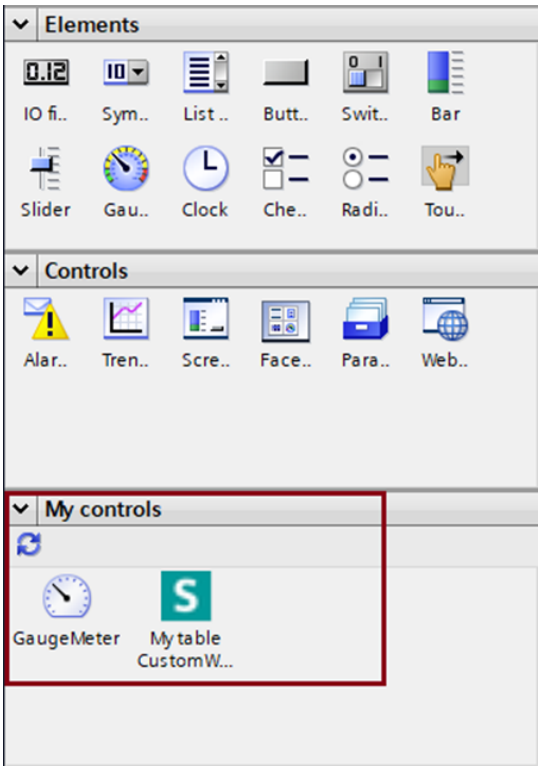
Custom Web Controls und die folgenden Konfigurationsschritte funktionieren mit WinCC Unified PC Runtime und den Unified Comfort Panels.

Bitte beachten Sie die begrenzte Leistung von Unified Comfort Panels. Custom Web Controls mit 3D-Darstellungen sind zum Beispiel für Unified Control Panels nicht empfohlen.

WinCC Unified Steuerung und Elemente

Die folgende Abbildung zeigt die Elemente, Standardsteuerung und benutzerdefinierte Steuerung ("Eigene Controls") in WinCC Unified für Unified Comfort Panel. Der Bereich "Eigene Controls" enthält die Custom Web Controls.

Abbildung 1-1



2 Custom Web Control "Gauge"

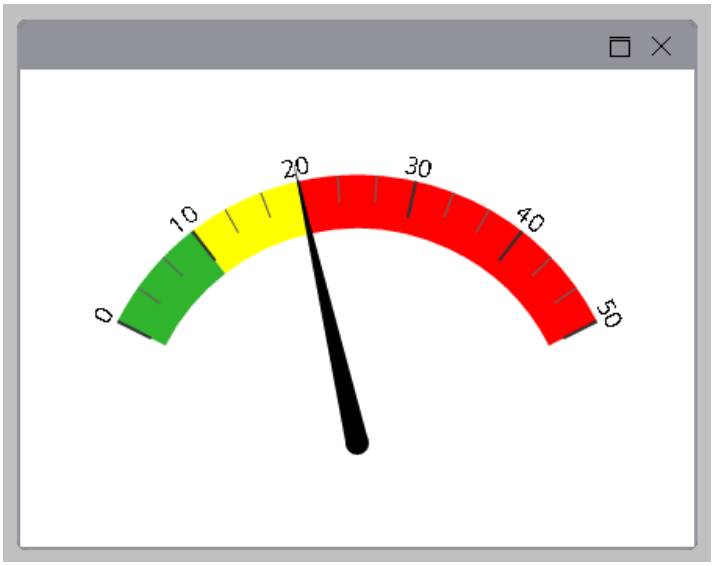
2.1 Einleitung

2.1.1 Überblick

Abbildung 2-1 zeigt das Custom Web Control "Gauge", das als Beispiel verendet wird, um die einzelnen Konfigurationsschritte zur Integration in WinCC Unified zu demonstrieren.

Das Beispiel-Control stammt aus der OpenSource-Bibliothek "Gauge.js" eines Drittanbieters und wird um Code für die Verwendung in WinCC Unified erweitert.

Abbildung 2-1



2.1.2 Funktionsprinzip

Das vollständige Custom Web Control hat die folgenden Funktionen:

- Anzeigen eines WinCC Unified Variablen-Werts auf einem Gauge-Control
- Definieren von Ober- und Untergrenzen
- Definieren verschiedener Stilparameter für eine bessere Darstellung
- Definieren von Wertbereichen mit verschiedenen Farben
- Methode zum Aufblinken des Bereichs, in dem sich der Wert gerade befindet
- Ereignis, das beim Übergang von einer Zone auf eine andere auftritt

Vorausgesetzte Kenntnisse

Das Anwendungsbeispiel setzt die folgenden Grundkenntnisse voraus:

- Konfiguration mit WinCC Unified
  Grundlagen werden im SITRAIN-Kurs "WinCC Unified & Unified Comfort Panels" vermittelt.
  Siehe Eintrags mit der ID 109773211.
- Microsoft Visual Studio CODE
- Website-Programmierung mit HTML5 und JavaScript

2.1.3 Verwendete Komponenten

Zur Erstellung des vorliegenden Anwendungsbeispiels wurden folgende Hardware- und Softwarekomponenten verwendet:

Tabelle 2-1

Komponenten	Artikelnummer	Hinweis
WinCC Unified V18	6AV2102-0AA08-0AA7	
Microsoft Visual Studio CODE	-	Siehe hier
Gauge.js 1.3.7		Siehe hier \5\

2.2 Engineering

2.2.1 Konfiguration und Implementierung des Custom Web Control

In diesem Abschnitt wird anhand eines Beispiels gezeigt, wie ein Custom Web Control mit Visual Studio Code erstellt wird.

Hinweis

Es können auch andere textbasierte Entwicklungsumgebungen verwendet werden.

Wenn Sie mit dem Thema nur wenig vertraut sind oder z. B. die Software "Visual Studio Code" nicht kennen, finden Sie unter Abschnitt 6 Hilfreiche Informationen eine kurze Einführung.

Ein Custom Web Control erstellen

1. Ordnerstruktur anlegen:

   (Vgl. auch das Handbuch, auf das in "Allgemeine Konfiguration und Ordnerstruktur" Bezug genommen wird.)

   Öffnen Sie den Windows Explorer und erstellen Sie die folgenden Elemente in einem Arbeitsordner Ihrer Wahl:

   a. Datei mit dem Dateinamen "manifest.json"

   b. Ordner mit dem Namen "assets".
   Legen Sie in diesem Ordner ein Symbol in einem beliebigen Grafikformat ab. Es wird im TIA Portal für diesen Control angezeigt.

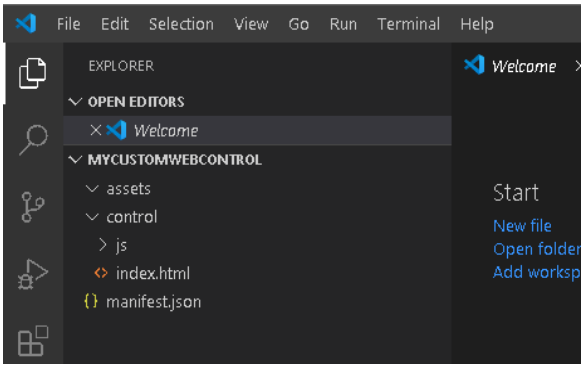
   c. Ordner mit dem Namen "control".
   Erstellen Sie dort die Dateien: "index.html", "code.js", "styles.css" und einen Ordner "js", in dem Sie die Datei "webcc.min.js" aus den beigefügten Dateien in diesem Anwendungsbeispiel und die Datei "gauge.min.js" mit dem heruntergeladenen Code (Quelle: https://bernii.github.io/gauge.js/ \5\) ablegen.

2. Öffnen Sie Visual Studio Code:

   a. Öffnen Sie Visual Studio Code (oder einen anderen Editor).

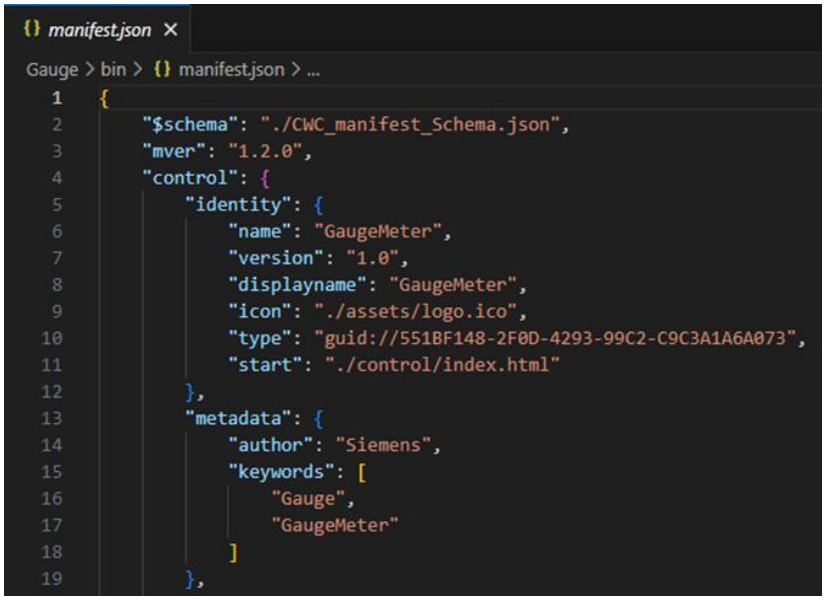
   b. Wählen Sie "Datei" → "Ordner öffnen", um den Ordner zu öffnen, in dem die von Ihnen gerade erstellte Ordnerstruktur hinterlegt ist.

Abbildung 2-2



3. Bereiten Sie die Datei "Manifest.json" vor:
   (Vgl. auch das Handbuch, auf das in "Vertragsbasierte Interaktion und die Manifest-Datei" Bezug genommen wird)
   a. Öffnen Sie die Datei Manifest.json in Visual Studio Code (oder in einem anderen Editor) Diese Datei benötigt eine gewisse Struktur, die im Handbuch genauer beschrieben wird. Für dieses Anwendungsbeispiel reicht es aus, eine bestehende Datei zu kopieren und einige Anpassungen vorzunehmen.
   b. Um ein neues Custom Web Control zu erstellen, kopieren Sie den Inhalt der manifest.json-Datei des Gauge-Beispiels in Ihre manifest.json.

Abbildung 2-3



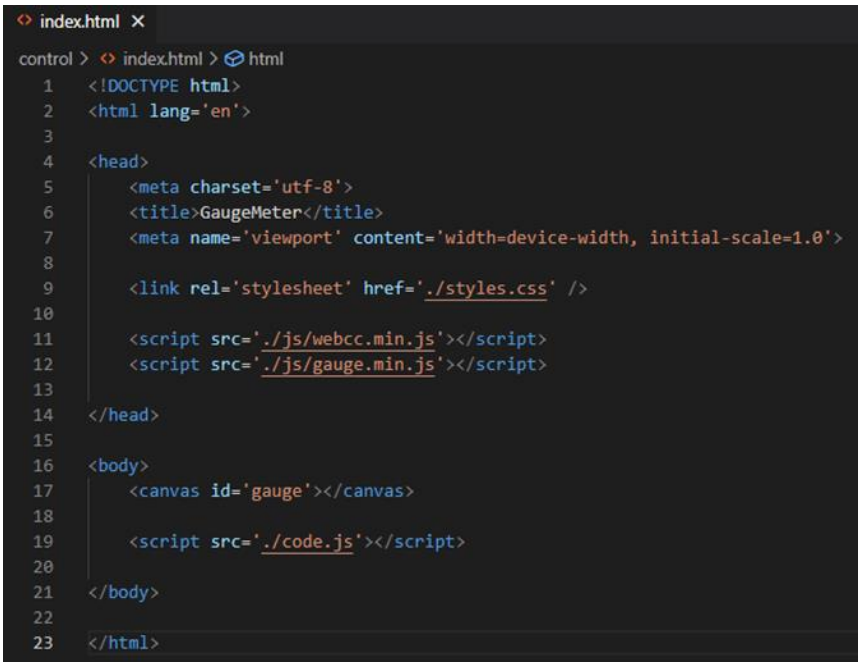
   c. Geben Sie Ihrem Custom Web Control unter dem Attribut "Name" in den Zeilen 6 und 8 einen benutzerdefinierten Namen.
   d. Passen Sie den Namen Ihres Logos in Zeile 9 an (alle üblichen Grafikformate, wie JPG, PNG, BMP usw. sind möglich).
   e. Weisen Sie in Zeile 10 einen neuen, benutzerdefinierten GUID zu. Sie können einen mit einem Online-Generator, wie https://www.guidgenerator.com/ erstellen.
   f. Passen Sie bei Bedarf die Metadaten in Zeile 13 an.
   g. Ändern Sie bei Bedarf ab Zeile 20 die Schnittstelle Ihres Custom Web Controls. Zu den möglichen Datentypen zählen:
   "boolesch", "Zahl", "String", "Array" oder einer Ihrer eigenen definierten Datentypen ab Zeile 80
   (Vgl. das enthaltene Manifest.json-Datei für Informationen zur Verwendung.)

Hinweis

Um sicherzustellen, dass Sie eine gültige JSON-Datei erstellt haben, referenzieren Sie Visual Studio Code darauf oder kopieren Sie den Inhalt der Datei in ein Online-Validierungstool (wie https://jsonlint.com: Fügen Sie den Inhalt ein und klicken Sie unten links auf "JSON validieren".)

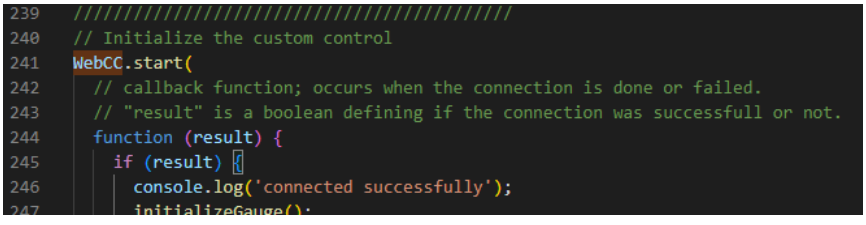
4. Index.html-Datei (Vgl. auch das Handbuch, auf das in "Interaktion zwischen Control und Container über die API".):
   a. Öffnen Sie die Datei Index.html in Visual Studio Code (oder in einem anderen Editor). Diese Datei ist das Portal zu Ihrer Website. Hier stellen Sie zum Datenaustausch auch eine Verbindung zum WinCC Unified Runtime-Server her.
   b. Die Verbindungsdaten für WinCC Unified befinden sich in der beigefügten Datei "webcc.min.js". Verschieben Sie sie in den "js"-Ordner und referenzieren Sie die Datei wie folgt in index.html:
   `<script src='./js/webcc.min.js'></script>`
   Diese Referenzierung wird am besten in Ihrer <Head>-Variable vorgenommen (siehe auch Zeile 11 des angefügten Beispiels).

Abbildung 2-4



   c. Die Verbindung wird mit der Funktion `WebCC.start()` in der Datei code.js (in Zeile 241) hergestellt, die in den angehängten Dateien enthalten ist.

Abbildung 2-5



Dazu ist es wichtig, dass die Verbindung korrekt hergestellt wird, wenn die Seite aufgerufen wird. Dies lässt sich am besten erreichen, wenn sich die Funktion (wie im Beispiel) direkt in der Variable <Script> befindet und nicht in einer tiefen verschachtelten Funktion, die möglicherweise erst zu einem späteren Zeitpunkt abgerufen wird (siehe Abbildung 2-4 Zeile 19).

5. Datenaustausch zwischen Custom Web Control und WinCC
(Vgl. auch das Handbuch, auf das in "Verwendung des Controls über WinCC"):
   a. Sie können nun von überall in Ihrer Anwendung auf die Daten zugreifen, die in der Manifest.json-Datei definiert sind.
   b. Der Zugriff wird mit dem API-Objekt "`WebCC`" bereitgestellt. Sie haben es bereits zur Herstellung der Verbindung verwendet. Jetzt haben Sie weitere Optionen.
   c. Wenn Sie Eigenschaften lesen oder schreiben möchten (in der Datei Manifest.json unter "properties" ab Zeile 42), können Sie z. B. mit Schreibzugriff wie folgt auf die Eigenschaft "GaugeValue" zugreifen: `WebCC.Properties.GaugeValue = 5` setzt den Wert auf 5. Der Wert der verknüpften WinCC-Variable wird ebenfalls auf 5 gesetzt.
   d. Wenn Sie die verbundenen WinCC-Variablen (vielleicht eine PLC-Variable) ändern möchten, verwenden Sie hierzu die Funktion "`WebCC.onPropertyChanged.subscribe()`". Sie sollten diese Funktion unmittelbar nach der erfolgreichen Herstellung der Verbindung aufrufen, um von Beginn an alle Änderungen zu erhalten (siehe Zeile 245 und 259). Nutzen Sie als Übergabe-Parameter eine von Ihnen definierte Funktion (Callback-Funktion). Im vorliegenden Beispiel ist es die Funktion "setProperty" in der Datei code.js in Zeile 125.
   Bei jeder Datenänderung wird die Funktion "setProperty" aufgerufen und ein "data"-Objekt übergeben, das einen "key" und einen "value" enthält. Der "key" ist der Name der geänderten Eigenschaft und der "value" der neue Wert. Es wird daher empfohlen, zur korrekten Verarbeitung des neuen Werts Verzweigungspunkte mit Switch-Case zu programmieren.

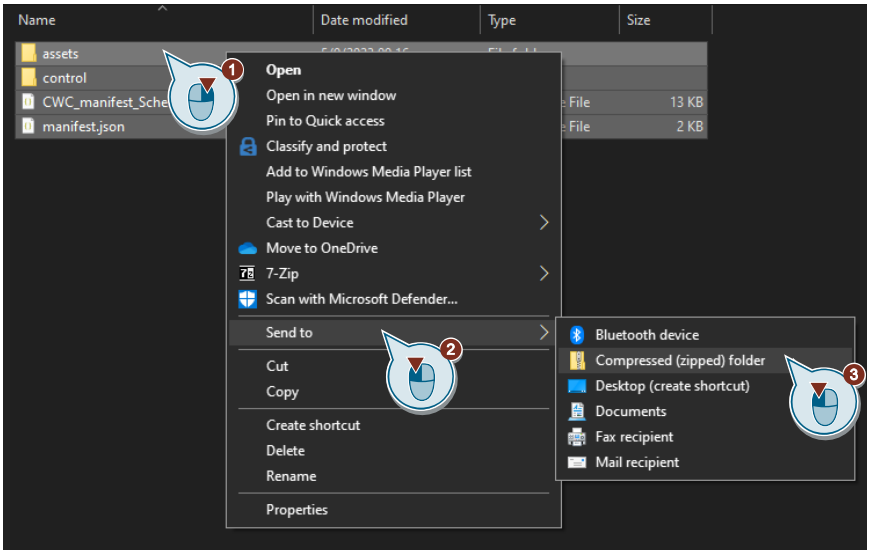
   Abbildung 2-6

```
121  // This is a callback function that is called every time a contrac
122  // other functions so you can see the new value in the control.
123  // - data: object containing a key and a value property. The "key"
124  //         the "value" contains the new value.
125  function setProperty(data) {
126    // console.log('onPropertyChanged ' + data.key);  // uncomment t
127    switch (data.key) {
128      case 'GaugeValue':
129        updateValue(data.value);
130        break;
131      case 'GaugeBackColor':
132        document.body.style.backgroundColor = toColor(data.value);
```

   e. Wenn Sie in der Manifest.json-Datei Methoden deklariert haben (Zeile 22 in der Datei "manifest.json"), kann WinCC diese Methoden aufrufen und Sie können im Custom Web Control auf sie reagieren. WinCC kann diese Methoden jederzeit aufrufen. Das bedeutet, dass Sie immer vor Herstellung der Verbindung definieren müssen, was geschehen soll. Sie definieren eine Funktion mit exakt dem Namen, den Sie in der Datei Manifest.json angegeben haben, und auch denselben Parametern. Sie finden ein Beispiel hierfür in der beigefügten code.js-Datei in Zeile 268.
   f. Wenn Sie in der Datei Manifest.json ein Ereignis definiert haben (siehe Zeile 32), können Sie dieses Ereignis an einer beliebigen Stelle in Ihrem Code mit "`WebCC.Events.fire()`" auslösen, so dass WinCC benachrichtigt wird. Der erste Übergabe-Parameter ist in diesem Fall immer der Name des Ereignisses, das Sie auslösen wollen, gefolgt von allen Übergabe-Parametern in der richtigen Reihenfolge, die Sie in der Manifest.json-Datei vorgegeben haben. Sie finden ein Beispiel hierfür in der beigefügten code.js-Datei in Zeile 69.

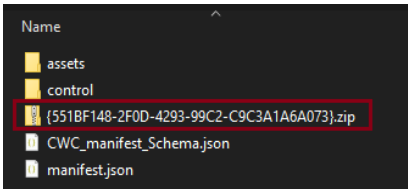
6. Bereitstellungsprozess zur Verwendung des Custom Web Controls im TIA Portal (Vgl. auch das Handbuch, auf das in "Erstellen der ZIP-Datei" Bezug genommen wird):
   a. Wenn Sie mit dem Programmieren fertig sind, müssen Sie den Code noch so verpacken, dass das TIA Portal Ihren Custom Web Control korrekt erkennt.
   b. Öffnen Sie hierzu den Windows Explorer und gehen Sie zu Ihrem Projektordner. Wählen Sie die ursprünglich erstellten Ordner "assets", "control, die Datei "CWC_manifest_Schema.json" (zu dieser Datei siehe Kapitel 2.2.4) und die Datei "manifest.json" und archivieren Sie sie mit Rechtsklick (1) →"Senden an" (2) → "Komprimierter (gezippter) Ordner" (3).

Abbildung 2-7



   c. Nutzen Sie Ihre GUID-Datei aus der Manifest.json-Datei, um den Namen Ihrer erstellten .zip-Datei wie folgt zu ändern (die X stehen dabei für Ihre GUID):
   `{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}.zip`
   Bitte beachten Sie die geschweiften Klammern vor und hinter den GUID.

Abbildung 2-8



   d. Ihr Custom Web Control ist nun fertiggestellt und bereit zur Benutzung im TIA Portal.

2.2.2 Installation und Integration in das Anwenderprojekt

Installation in das TIA Portal

Bevor Sie das Custom Web Control im TIA Portal verwenden können, stehen Ihnen zwei Vorgehensweisen für dessen Installation zur Verfügung (vgl. auch das offizielle Handbuch unter "Installation eines Custom Web Control"):

1. Nur für ein bestimmtes Projekt verfügbar machen:
   Fügen Sie Ihr Custom Web Control in Ihren Projektordner ein:

   "...Project_1\UserFiles\CustomControls\ {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}.zip"
2. Für alle Projekte verfügbar machen
   Fügen Sie Ihr Custom Web Control in den Installationspfad des TIA Portals ein:

   "C:\Program Files\Siemens\Automation\Portal Vxx\Data\Hmi\CustomControls\ {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}.zip"

Hinweis Wenn Sie das Projekt auf einen anderen PC kopieren, werden die Custom Web Controls nicht übermittelt und im TIA Portal Projekt entsteht ein Kompilierungsfehler. Sie müssen auch die Custom Web Controls auf den neuen Computer und in das oben angegebene Verzeichnis kopieren.

Hinweis Wenn Sie ein Projekt auf den Operator Panel laden, übermittelt das TIA Portal auch Ihre Custom Web Control. Es findet keine Installation auf dem Runtime-Server statt.

Integration in das Projekt des Benutzers

Gehen Sie wie folgt vor, um Ihr Custom Web Control im TIA-Portal-Projekt zu konfigurieren:

1. Öffnen Sie ein Bild Ihres Unified Comfort Panel oder Ihrer PC-Station.
2. Klicken Sie auf Ihr Custom Web Control unter "Tools > Eigene Controls" und ziehen Sie es per Drag & Drop auf das Bild.

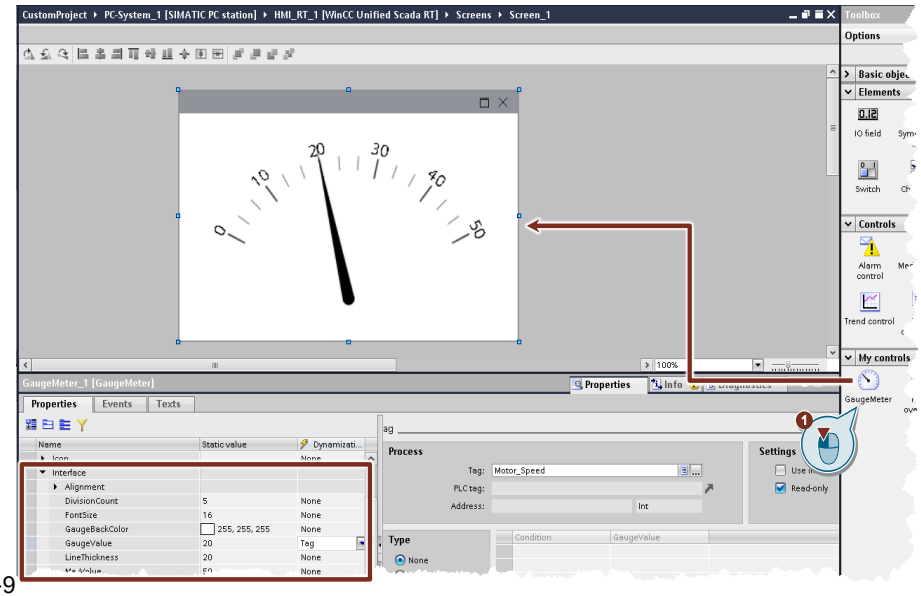
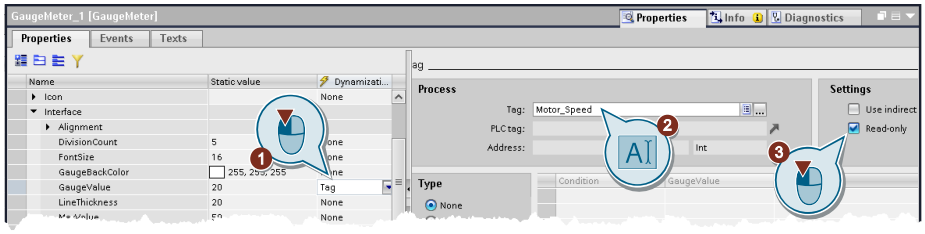


Abbildung 2-9

In den Eigenschaften Ihres Custom Web Controls finden Sie unter "Interface" all die Eigenschaften, die Sie in der Manifest.json-Datei definiert haben. Sie können diesen, wie auch allen anderen Eigenschaften von Bildobjekten einen statischen Wert zuweisen oder eine Dynamisierung definieren.

3. Custom Web Control dynamisieren
   a. Erstellen Sie eine Dynamisierungs-"Variable" für "GaugeValue".
   b. Wählen Sie eine geeignete SPS oder einen geeigneten HMI-Tag.
   c. Wenn Ihr Custom Web Control nur Lesezugriff auf die Variable hat, lassen Sie das Häkchen gesetzt. Wenn Ihre Implementierung jedoch vorsieht, dass das Custom Web Control Ihre Variablen verändert, dann deaktivieren Sie das Kontrollkästchen. In diesem Fall zeigt der "GaugeMeter" nur die Variable an (lesen).

Abbildung 2-10



2.2.3 Debugging

In diesem Abschnitt erfahren Sie, wie Sie Custom Web Controls in Runtime debuggen.

Ein Debugging ist im TIA Portal nicht möglich. Wenn Sie den Custom Web Control nicht auf Runtime übertragen können, weil Sie zuvor Probleme festgestellt haben, lesen Sie Kapitel 4 Bedienung Beispielprojekt für mögliche Lösungen.

Debugging in Google Chrome

Die WinCC Unified Runtime sendet das Custom Web Control direkt an den Web-Client bei Verwendung, sodass Sie es auch am Web Client mit der Standard-Entwicklerkonsole des Browsers debuggen können.

Öffnen Sie das Unified Runtime Bild, in dem sich das Custom Web Control befindet, mit dem Google Chrome Browser und drücken Sie auf Ihrer Tastatur F12, um die Entwicklerkonsole zu öffnen.

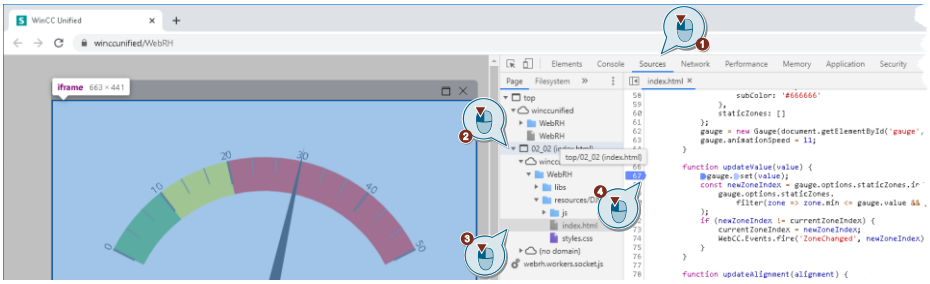
Schritt-für-Schritt-Anleitung

In Abbildung 2-11 sehen Sie, wie das Debugging nach dem Öffnen der Entwicklerkonsole mit F12 funktioniert.

1. Wählen Sie die Registerkarte "Sources" in der Entwicklerkonsole.
2. Bewegen Sie den Mauszeiger langsam auf der linken Seite der Entwicklerkonsole über die Ordner unter dem Knoten "top". Ihr Custom Web Control hat hier keinen sprechenden Namen, er wird aber von Ihrem Browser blau markiert, wenn Sie ihn ausgewählt haben. (hier finden Sie für jede Custom-Web-Control-Instanz einen eigenen Ordner).
3. Öffnen Sie den entsprechenden Ordner und gehen Sie zu der Datei, die Sie debuggen wollen. (hier ist sämtlicher Code in der index.html-Datei enthalten.)

4. Scrollen Sie im rechten Fenster zur entsprechenden Position und klicken Sie auf die Zeilennummer, um einen Haltepunkt einzusetzen. Wenn der Code diese Position wieder erreicht, stoppt er und Sie erhalten detaillierte Informationen zu Programmsequenz. In der Abbildung befindet sich ein Haltpunkt am Anfang der Funktion "updateValue". Das Skript wird nun immer hier stoppen, wenn Sie den verknüpfen Wert aktualisieren. Entfernen Sie den Haltepunkt, indem Sie nochmals auf die Zeilennummer klicken.

Abbildung 2-11



Hinweis Weitere Informationen über die Arbeit mit der Entwicklerkonsole und die Informationen, die Sie auslesen können, finden Sie in der offiziellen Dokumentation von Google: https://developers.google.com/web/tools/chrome-devtools/javascript#sources-ui

Hinweis Sie können ein Debugging auch mit jedem anderen Browser durchführen, lesen Sie dazu einfach die Dokumentation des jeweiligen Browsers.

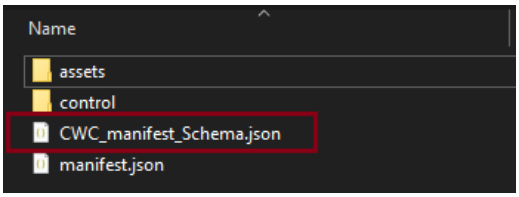
2.2.4 Auto-Vervollständigen bei der Programmierung

In diesem Kapitel erfahren Sie, wie Sie Auto-Vervollständigen aktivieren.

JSON Manifest-Schema

Zur Unterstützung bei der Erstellung der "manifest.json" benötigen Sie eine Datei, die das Schema vorgibt um die Auto-Vervollständigung zu ermöglichen (siehe Schritt 3 in Kapitel 2.2.1). Diese wird dann auf derselben Dateiebene platziert, wie die "manifest.json"-Datei. Kopieren Sie hierzu die Datei "CWC_manifest_Schema.json" zur "manifest.json"

Abbildung 2-12



Jetzt müssen Sie den Pfad des Templates in der "manifest.json" angeben. Es genügt, den Pfad zum Template direkt nach dem Start der Datei anzugeben. Hierzu wird der Parameter "$schema" verwendet. In diesem Fall: fügen Sie folgende Zeile hinter der ersten öffnenden geschweiften Klammer ein:

```
"$schema": "./CWC_manifest_Schema.json"
```

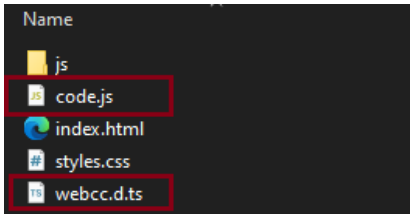
Speichern Sie die Änderungen. Sie können die Auto-Vervollständigen nun benutzen.

Abbildung 2-13

Skripting-Unterstützung mit webcc.d.ts

Für die Auto-Vervollständigung im Skript-Teil benötigen Sie die webcc.d.ts-Datei (Siehe Schritt 4 und 5 von Kapitel 2.2.1). Sie enthält die relevanten Informationen für das WebCC-Objekt und muss sich im selben Ordner wie die "index.html"-Datei befinden. Dieser Skriptteil ist in der "code.js"-Datei enthalten, weil die Skripting-Unterstützung von Visual Studio Code nur in einer .js-Datei funktioniert und nicht in der "index.html".

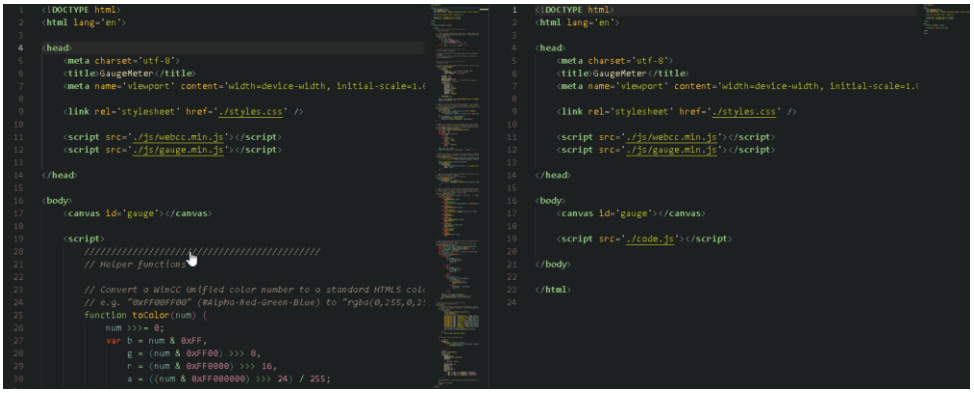
Abbildung 2-14



Das CWC muss referenziert sein, um Skripte in der "code.js" lokalisieren zu können. Das tun Sie mit dem folgenden Codeschnipsel:

```
<script src='./code.js'></script>
```

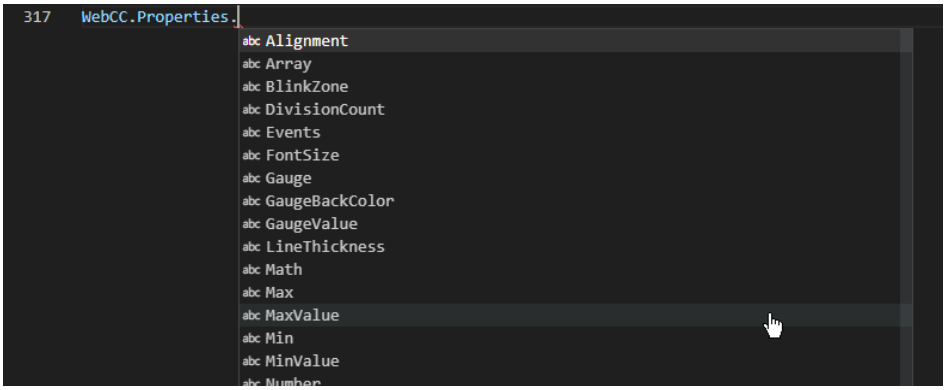
Abbildung 2-15



Ergebnis

Als Ergebnis erhalten Sie Auto-Vervollständigen und eine eindeutige Trennung der Skripte.

Abbildung 2-16



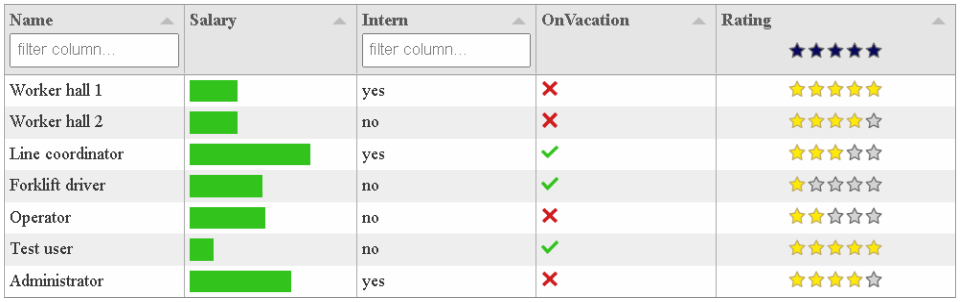
3 Custom Web Control "Table"

3.1 Einleitung

3.1.1 Überblick

Im Folgenden ist ein weiteres Anwendungsbeispiel für die Erstellung von Custom Web Controls aufgeführt. Das Custom Web Control "Table" ist eine tabellarische Darstellung von Werten. Sie können die Werte nicht nur anzeigen, sondern auch grafisch formatieren (siehe nachfolgende Abbildung).

Abbildung 3-1



Für dieses Beispiel wurde die JavaScript-Bibliothek "Tabulator" (siehe http://tabulator.info) verwendet und mit Code ergänzt, damit sie in WinCC Unified verwendet werden kann.

3.1.2 Funktionsspektrum

Mit dem Custom Web Control "Table" haben Sie die folgenden Optionen:

- Erstellen einer Tabelle auf Grundlage übermittelter WinCC Unified Variablenwerte (z. B. Prozesswerte, Parameter-Sollwerte)
- Alle Tabellenwerte in einer String-Variablen speichern
- Tabellenspalten einzeln anpassen (Breite, Anzeigeformat der Werte, Filteroptionen für die Werte)

3.1.3 Verwendete Komponenten

Bei der Erstellung dieses Anwendungsbeispiels wurde neben den bereits dargelegten Hard- und Softwarekomponenten die folgende Komponenten verwendet:

Tabelle 3-1

Komponenten	Artikelnummer	Hinweis
WinCC Unified V18	6AV2102-0AA08-0AA7	
Microsoft Visual Studio CODE	Siehe im Internet	
Gauge.js 1.3.7	Siehe im Internet \5\	

3.1.4 Anwendungsbeispiele

Die Tabellen des Custom Web Control eignen sich u. a. für die folgenden Zwecke:

- Ausgabe der Werte einer CSV-Datei als Tabelle in WinCC Unified Runtime.
- Grafische Darstellung der Prozesswerte innerhalb einer Tabelle.
- Exportieren der Prozesswerte als CSV-Datei.

3.2 Engineering

3.2.1 Konfiguration und Implementierung des Custom Web Control

In diesem Abschnitt erfahren Sie, wie Sie mit Visual Studio Code das Custom Web Control "Table" erstellen und implementieren.

Custom Web Control erstellen

1. Erstellen Sie eine Ordnerstruktur (Siehe auch im Handbuch unter: "Allgemeiner Aufbau und Ordnerstruktur").

   Öffnen Sie den Windows Explorer und erstellen Sie die folgenden Elemente in einem Arbeitsordner Ihrer Wahl:

   a. Datei mit dem Dateinamen "manifest.json"
   (Diese Datei benötigt eine gewisse Struktur, die im Handbuch genauer beschrieben wird. Für dieses Anwendungsbeispiel reicht es aus, eine bestehende Datei zu kopieren und bei Bedarf Anpassungen vorzunehmen.)

   b. Ordner mit dem Namen "assets":
   Legen Sie in diesem Ordner ein Symbol in einem beliebigen Grafikformat ab. Es wird im TIA Portal für diesen Control angezeigt.

   c. Ordner mit dem Namen "control".

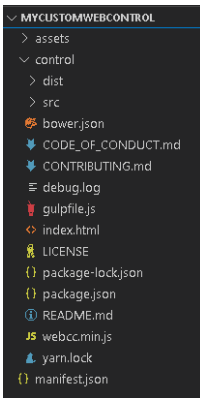
   d. Erstellen Sie im Ordner "control" die Dateien "index.html" und "code.js" und fügen Sie die Datei "webcc.min.js" aus den beigefügten Dateien zu diesem Verzeichnis hinzu. Entpacken Sie die heruntergeladene "Tabulator"-Bibliothek (Quelle: https://github.com/olifolkerd/tabulator) ebenfalls in den Ordner "control".

2. Visual Studio Code:

   a. Öffnen Sie Visual Studio Code oder einen anderen Editor Ihrer Wahl.

   b. Wählen Sie "Datei" → "Ordner öffnen", um den Ordner zu öffnen, in dem die von Ihnen gerade erstellte Ordnerstruktur hinterlegt ist.

   Abbildung 3-2



3. Erstellen Sie die Manifest.json-Datei:

   a. Öffnen Sie die Manifest.json-Datei.

   b. Richten Sie Ihre Manifest-Datei wie in Abschnitt 2.2.1 beschrieben ein.

   c. Nehmen Sie keine Änderungen an der Eigenschaft "TableDataString" vor. Die Tabelleninhalte werden hier zurückgeschrieben.

4. Index.js konfigurieren
   a. Referenzieren Sie die erforderlichen Skripte.

      Abbildung 3-3

```
<script src='webcc.min.js'></script> <!-- mandatory dependency -->
<script type="text/javascript" src="./dist/js/tabulator.min.js"></script>
<link href="./dist/css/tabulator.css" rel="stylesheet">
```

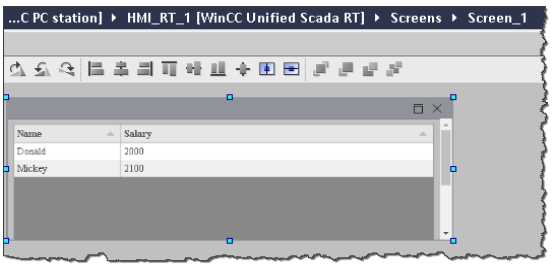
5. Konfigurieren Sie code.js:
   a. Starten Sie das Verbindungs-Setup mit der Funktion "WebCC.start();".
   b. Sobald die Verbindung hergestellt ist, wird der aktuelle, hinter "TableDataString" verstecke Wert in das Array "ArrayData" geschrieben.
   c. Zuletzt sehen Sie die Funktion "subscribe". Sie prüft, ob sich der Wert "TableDataString" ändert. Dies geschieht automatisch, wenn die Seite geladen wird. Wenn die Seite neu geladen wird, wird der Wert auf den "statischen" Wert aus dem TIA Portal-Projekt gesetzt. Sobald dies geschieht, wird der Wert erneut mit den aktuellen Werten von "ArrayData" überschrieben.

6. Arbeiten mit "Tabulator"
   a. Das Feld "ArrayData" enthält jetzt die Datenpunkte aus der Tabelle. Sie können diese verwenden, um Tabellen mit der "Tabulator"-Funktion zu erstellen. (Weitere Informationen erhalten Sie unter http://tabulator.info/)
   b. Die Interface-Variable "ColumnStyleString" definiert die Eigenschaften der Tabellenspalten.

7. Bereitstellungsprozess und Installation:
   a. Wenn Sie die Programmierung abgeschlossen haben, erstellen Sie eine *.zip-Datei, wie im vorherigen "Gauge"-Beispiel gezeigt.
      Die .zip-Datei sollte die Order "assets" und "control", die Datei "CWC_manifest_Schema.json" (siehe Kap. 2.2.4 für weitere Informationen zu dieser Datei) und die Datei "manifest.json" enthalten. Geben sie der *.zip-Datei einen eigenen GUID-Namen `({xxxxxxxx-xxxx-xxxx-xxxxxxxx}.zip`).
   b. Stellen Sie diese Datei nun Ihrem Benutzer-Projekt zur Verfügung, indem Sie es in den Ordner "CustomControls" Ihres Projekts kopieren.
      `…Project_1\UserFiles\`
      `CustomControls\{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}.zip`

      Öffnen Sie Ihr TIA-Portal-Projekt und ziehen Sie Ihr Custom Web Control auf Ihr gewünschtes Bild.

      Abbildung 3-4



   c. Übergeben Sie dann die entsprechenden Daten per Skript an die Eigenschaften "ColumnStyleString" und "TableDataString" des Custom Web Control.

      Weitere Informationen hierzu finden Sie im Abschnitt Erforderliche Daten sowie in Abschnitt Funktionsweise des Custom Web Control "Table".

3.2.2 Installation und Integration ins Anwenderprojekt

Unter Abschnitt Installation und Integration ins Anwenderprojekt ist beschrieben, wie Sie das Custom Web Control im TIA Portal installieren und in Ihr Benutzer-Programm integrieren.

3.2.3 Debugging

Im Abschnitt Debugging erfahren Sie, welche Möglichkeiten zur Fehlerdiagnose und -behebung Ihnen zur Verfügung stehen.

3.2.4 Erforderliche Daten

Um eine Tabelle erstellen zu können, müssen zwei Strings an die folgenden beiden Eigenschaften der "Table" Custom Web Control übergeben werden:

- "ColumnStyleString": String zur Formatierung der Spalten
- TableDataString String mit Dateneinträgen (Tabelleninhalt)

ColumnStyleString

"ColumnStyleString" (String in JSON-Format) definiert das Layout der Tabellenspalten.
Ein Beispiel für eine Struktur des "ColumnStyleString" für die Spalte "Name" ist in der folgenden Abbildung dargestellt.

Abbildung 3-5

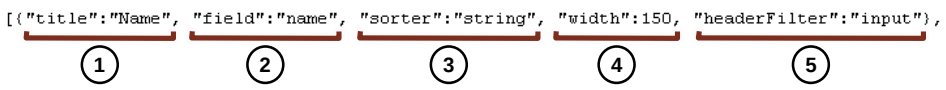
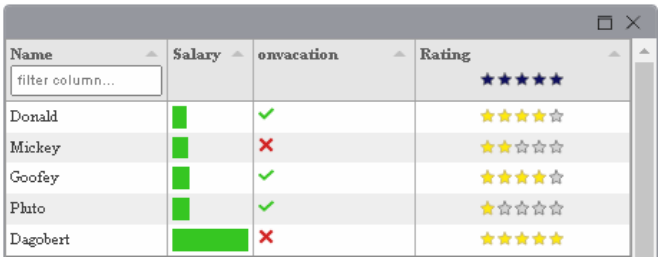


Tabelle 3-2

Nr.	Parameter	Beschreibung
1.	title	Definiert den Namen der Spalte im Custom Web Control
2.	field	Dient der Zuordnung der Werte (aus dem "TableDataString") beim Befüllen der Tabelle. **Hinweis:** Achten Sie darauf, dass die Schreibweise der Werte korrekt ist, andernfalls kann keine Zuordnung zur Spalte "title" erfolgen.
3.	sorter	Gibt an, nach welchem Attribut die Liste sortiert wird.
4.	width	Definiert die Spaltenbreite.
5.	headerFilter	Optional: Erstellt einen Filter im Spaltenkopf.

Sie können die Werte nicht nur einfach anzeigen, sondern auch grafisch formatieren. Geben Sie dies bei Bedarf über zusätzliche Parameter an, wie z. B. "hozAlign", "formatter" und "formatterParams".

Abbildung 3-6



Hinweis

Unter dem folgenden Link finden Sie eine detaillierte Beschreibung der Formatierungsoptionen: http://tabulator.info/docs/4.9/format

TableDataString

Der "TableDataString" (String im JSON-Format) enthält die Werte, mit denen die einzelnen Spalten befüllt werden sollen. Seine Struktur besteht aus dem "field"-Namen des "ColumnDataString" und dem Wert, die in die entsprechende Tabellenzeile eingesetzt werden soll.

Hinweis

In Abschnitt 4.1 Custom Web Control "Gauge"-Verwendung wird der Aufbau von "ColumnStyleString" und "TableDataString" in der Anwendung anhand des Beispielprojekts nochmals erläutert.

Weitere Informationen zu den Eigenschaften finden Sie unter http://tabulator.info/.

Hinweis

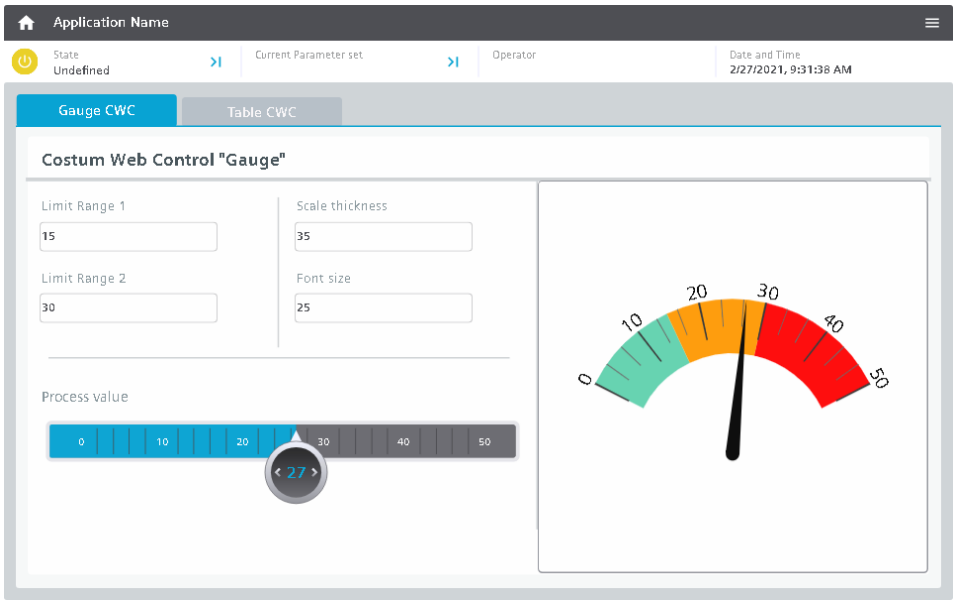
Wenn ein Parameter leer oder eine Control nicht korrekt verbunden ist, zeigt das Custom Web Control die Tabelle nicht korrekt an (siehe Abbildung 5-3).

4 Bedienung Beispielprojekt

4.1 Custom Web Control "Gauge" - Bedienung

Das Beispielprojekt besteht aus zwei Registerkarten. In der ersten Registerkarte finden Sie das CWC "Gauge", in dem vier Eigenschaften (über E/A-Felder) sowie der Prozesswert (über Schieberegler) als Beispiel dynamisiert werden können.

Abbildung 4-1



4.2 Custom Web Control "Table" - Bedienung

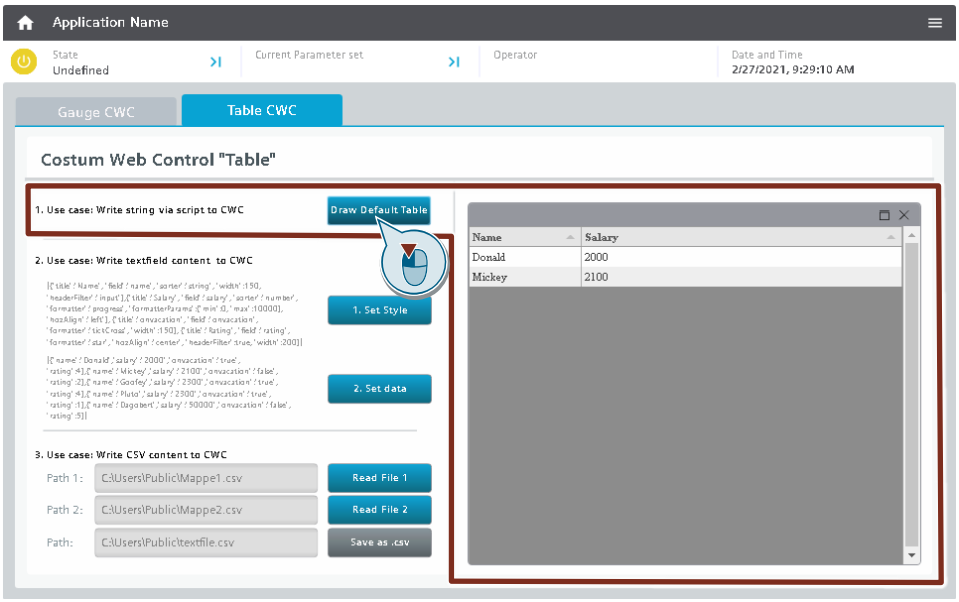
In der zweiten Registerkarte finden Sie das Custom Web Control "Table". Es werden Ihnen drei Möglichkeiten angezeigt, wie Sie das Custom Web Control mit den erforderlichen Daten versorgen können (vgl. Abschnitt 3.2.4 Erforderliche Daten).

- Skript: Tabelleninhalte sind in einem Skript hinterlegt.
- Bildobjekt: Tabelleninhalte sind in einem Bildobjekt gespeichert (z. B. Textfeld).
- CSV-Datei: Tabelleninhalte sind in einer CSV-Datei enthalten oder müssen exportiert werden.

4.2.1 Übermittlung von Tabelleninhalten über ein Skript

Im ersten Beispiel werden die Werte einfach in zwei Spalten angezeigt. Die beiden Strings ("ColumnTableString" and "TableDataString") werden an das CWC über die Schaltfläche "Draw default table" übergeben.

Abbildung 4-2



ColumnStyleString

Der "ColumnStyleString" setzt sich für dieses Beispiel wie folgt zusammen:

```
'[{"title":"Name", "field":"name", "sorter":"string", "width":150},
{"title":"Salary", "field":"salary", "sorter":"number","hozAlign":"left"}]'
```

TableDataString

Der "TableStyleString" setzt sich für dieses Beispiel wie folgt zusammen:

```
'[{"name":"Donald","salary":"2000"},
{"name":"Mickey","salary":"2100"}]'
```

Hinweis

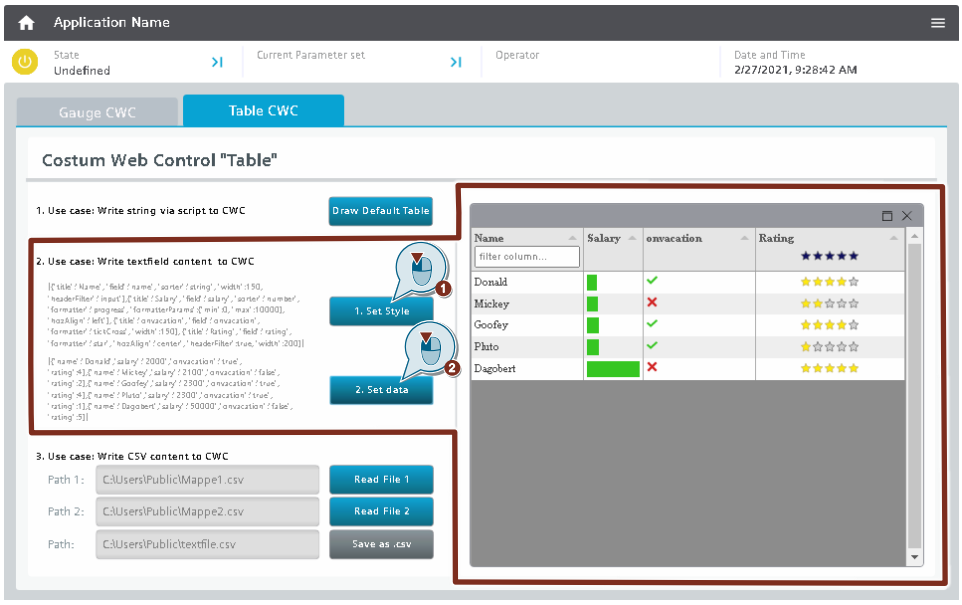
Für eine bessere Lesbarkeit wurden den beiden Strings in der Dokumentation Zeilenumbrüche hinzugefügt.

4.2.2 Übermittlung von Tabelleninhalten über ein Bildobjekt

Im zweiten Beispiel werden die Tabelleninhalte grafisch formatiert angezeigt. Zusätzlich wurde in den Spalten "Name" und "Rating" je ein Filter hinzugefügt, um die Einträge filtern zu können. Die beiden Strings ("ColumnTableString" und "TableDataString") wurden außerdem in einem Bildobjekt hinterlegt (im vorliegenden Beispiel als Textfeld).

Über die Schaltfläche "1 Set Style" wird das Format ("ColumnStyleString") zunächst an das CWC übertragen. Anschließend wird der Tabelleninhalt ("TableDataString") über die Schaltfläche "2. Set data" übertragen.

Abbildung 4-3



ColumnStyleString

Der "ColumnStyleString" setzt sich für dieses Beispiel wie folgt zusammen:

```
[{"title":"Name", "field":"name", "sorter":"string", "width":150,
"headerFilter":"input"},
{"title":"Salary", "field":"salary", "sorter":"number", "formatter":"progress",
"formatterParams":{"min":0, "max":10000}, "hozAlign":"left"},
{"title":"onvacation", "field":"onvacation", "formatter":"tickCross", "width":150},
{"title":"Rating", "field":"rating", "formatter":"star", "hozAlign":"center",
"headerFilter":true, "width":200}]
```

TableDataString

Der "TableStyleString" setzt sich für dieses Beispiel wie folgt zusammen:

```
[{"name":"Donald","salary":"2000","onvacation":"true", "rating":4},
{"name":"Mickey","salary":"2100","onvacation":"false", "rating":2},
{"name":"Goofey","salary":"2300","onvacation":"true", "rating":4},
{"name":"Pluto","salary":"2300","onvacation":"true", "rating":1},
{"name":"Dagobert","salary":"50000","onvacation":"false", "rating":5}]
```

Hinweis

Für eine bessere Lesbarkeit wurden den beiden Strings in der Dokumentation Zeilenumbrüche hinzugefügt.

4.2.3 Übermittlung von Tabelleninhalten über eine CSV-Datei

Im dritten Anwendungsfall werden die Werte sowohl unformatiert ("Read File 1") als auch formatiert ("Read file 2") angezeigt.

Hinweis

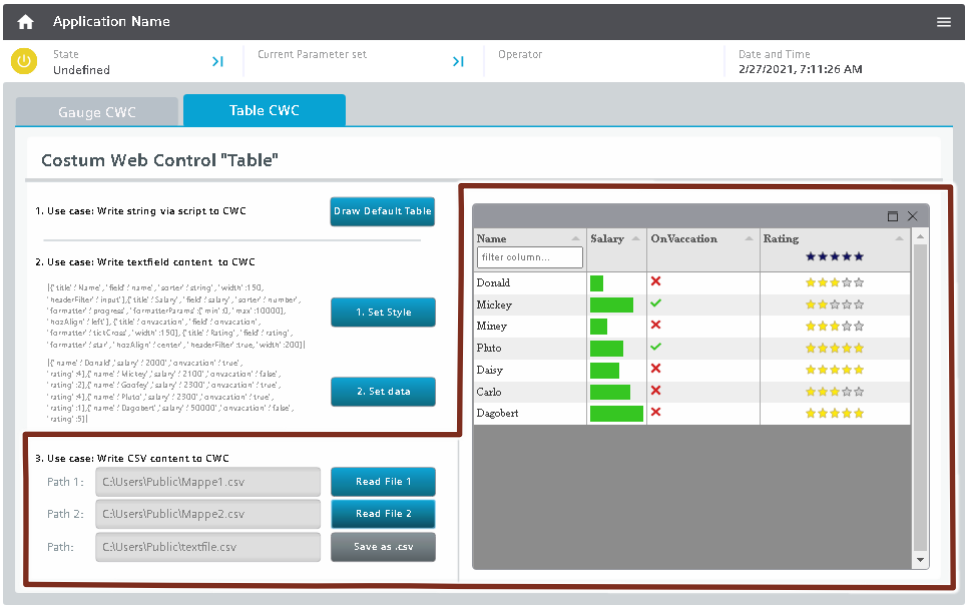
Im Projektverzeichnis des Beispielprojekts ("109779176_CustomWebControl_V30_PROJ\UserFiles") liegen bereits zwei vorkonfigurierte CSV-Dateien. Wenn Sie diese im Beispielprojekt verwenden wollen, müssen Sie beide Dateien (Ordner 1 und Ordner 2) an folgendem Ort Ihres verwendeten Bediengeräts ablegen.

- Unified PC Runtime: "C:\Users\Public"
- Unified Comfort Panel: "\home\industrial\"

Der String "ColumnTableString" wird im Skript der jeweiligen Schaltfläche gespeichert. Es definiert die unformatierte oder formatierte Anzeigeform.

Die Tabellendaten "TableDataString" werden aus einer CSV-Datei gelesen und von einem Skript vor Weitergabe an das CWC in das JSON-Format konvertiert.

Abbildung 4-4



ColumnStyleString

Der "ColumnStyleString" setzt sich für das unformatierte Beispiel wie folgt zusammen:

```
'[{"title":"Name", "field":"name", "sorter":"string", "width":150},
{"title":"Salary", "field":"salary", "sorter":"number", "hozAlign":"left"},

{"title":"Intern", "field":"intern", "sorter":"boolean", "hozAlign":"left"}]'
```

Der "ColumnStyleString" setzt sich für das formatierte Beispiel wie folgt zusammen:

```
'[{"title":"Name", "field":"name", "sorter":"string", "width":150,
"headerFilter":"input"},
{"title":"Salary", "field":"salary", "sorter":"number", "formatter":"progress",
"formatterParams":{"min":0, "max":10000}, "hozAlign":"left"},
{"title":"OnVaccation", "field":"onvaccation", "formatter":"tickCross", "width":150},
{"title":"Rating", "field":"rating", "formatter":"star", "hozAlign":"center",
"headerFilter":true, "width":200}]'
```

TableDataString

Für dieses Beispiel lautet der "TableStyleString" wie folgt:

```
HMIRuntime.FileSystem.ReadFile("C:\\Users\\Public\\Mappe1.csv", "utf8").then(
  function(text) {
//read file and convert to a string in JSON format
  var lines = text.split("\n");
    var result = [];
    var headers;
    headers = lines[0].split(";");

    for (var i = 1; i < lines.length; i++) {
        var obj = {};
        if(lines[i] == undefined || lines[i].trim() == "") {
            continue;
        }
        var words = lines[i].split(";");
        for(var j = 0; j < words.length; j++) {
            obj[headers[j].trim()] = words[j];
        }
        result.push(obj);
    }

//write to Interface
Screen.Items('MyCWC').Properties.TableDataString = JSON.stringify(result);
```

5 Häufige Fehlerbilder

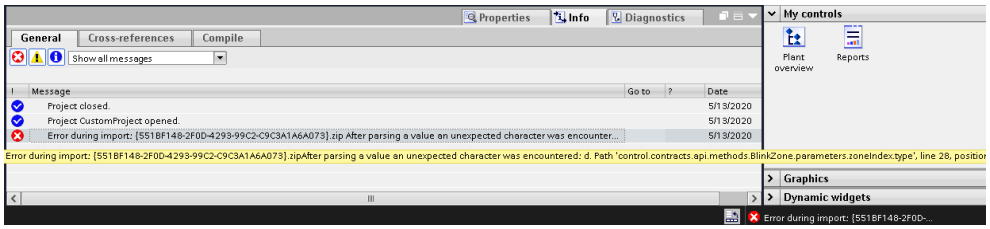
In diesem Abschnitt werden häufige Fehlerbilder, die bei der Erstellung und Integration eines Custom Web Controls auftreten können, sowie deren Lösungen beschrieben.

5.1 Custom Web Control erscheint nicht in der TIA Portal Toolbox

Fehlerbeschreibung

Wenn Sie im TIA Portal ein Bild öffnen und Ihr Custom Web Control erscheint nicht in der Toolbox auf der rechten Seite, dann enthält Ihre Manifest.json-Datei einen Fehler. Zusätzlich erscheint eine Fehlermeldung im Infofeld, wie in Abbildung 5-1 gezeigt.

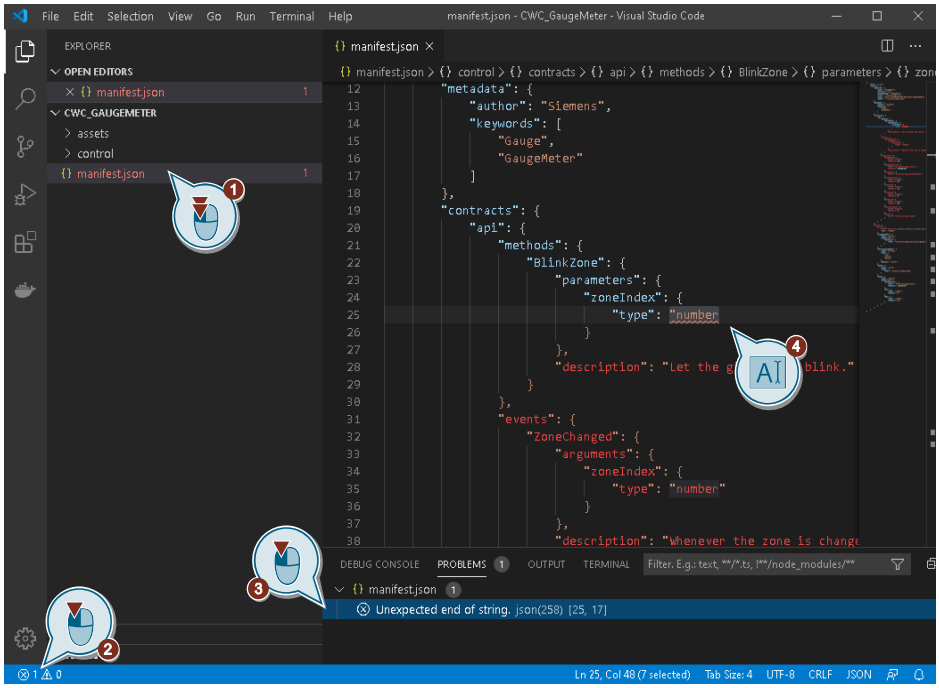
Abbildung 5-1



Lösung

Um das Problem zu lösen, öffnen Sie Ihre Manifest.json-Datei mit Visual Studio Code und beheben Sie die Fehler wie in Abbildung 5-2 gezeigt.

Abbildung 5-2



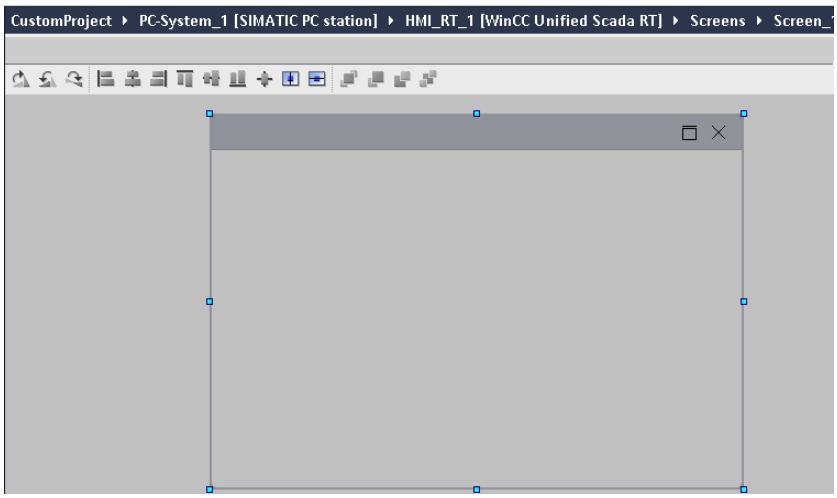
1. Öffnen Sie die Manifest.json-Datei mit einem Doppelklick. Visual Studio Code wird sie dann automatisch prüfen und den Dateinamen in Rot sowie die Anzahl Fehler auf der rechten Seite anzeigen.
2. Unten links sehen Sie dann die Fehleranzahl. Klicken Sie darauf, um auf der rechten Seite ein Fenster mit einer Detailbeschreibung aller Fehler zu öffnen.
3. Klicken Sie auf einen Fehler. Visual Studio Code springt im Dokument direkt an die Stelle, an der sich der Fehler befindet.
4. Beheben Sie den Fehler mit Hilfe der Fehlerbeschreibung.

5.2 TIA Portal stellt das Custom Web Control im Bild nicht korrekt dar

Fehlerbeschreibung

Sie können Ihr Custom Web Control erfolgreich ins Bild einfügen und mit den Eigenschaften verknüpfen. Das funktioniert auch in Runtime problemlos, allerdings zeigt der Bildeditor des TIA Portals ein leeres oder fehlerhaftes Control an, wie z. B in Abbildung 5-3.

Abbildung 5-3



Erklärung: TIA Portal versucht, die Webinhalte anzuzeigen, aus Sicherheitsgründen deaktiviert es aber bestimmte Funktionen, mit dem Ergebnis, dass Ihr Custom Web Control falsch oder gar nicht angezeigt wird.

Lösung

Als Lösung können Sie eine Vorschau in Ihren Custom Web Control programmieren, die nur der TIA Portal Bildeditor anzeigt. In Runtime wird es jedoch wie gehabt funktionieren.

Dazu müssen Sie nach einer erfolgreichen Verbindung in JavaScript definieren, wie der Code fortfahren soll. Mit einer IF-Abfrage der Eigenschaft "isDesignMode" am API-Objekt "WebCC" können Sie herausfinden, ob sich Ihr Custom Web Control gerade im TIA Portal oder in der Runtime befindet.
Ihr Code sollte dem in Abbildung 5-4 ähneln, wobei hier eine neue Funktion "showDemoData()" definiert wurde, in der Sie Ihren Code programmieren, um für das TIA Portal eine Vorschau darzustellen.

Abbildung 5-4

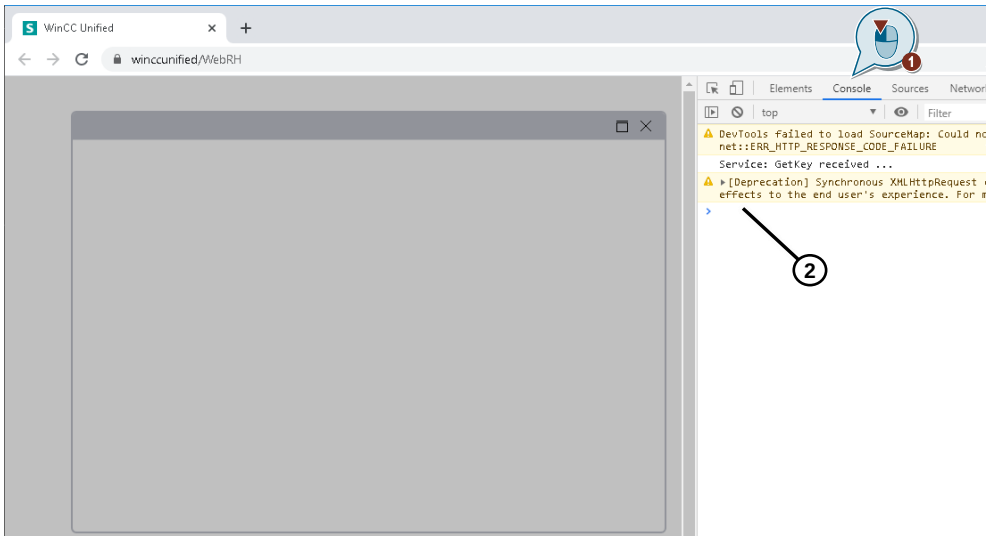
```
18        ////////////////////////////////////////////
19        // Initialize the custom control
20        WebCC.start(
21            // callback
22            function (result) {
23                if (result) {
24                    console.log('connected successfully');
25                    if (WebCC.isDesignMode) {
26                        // do not subscribe, only show some dummy data
27                        showDemoData();
28                    } else {
29                        // Subscribe for value changes
30                        WebCC.onPropertyChanged.subscribe(setProperty);
```

5.3 Custom Web Control bleibt in Runtime Leer

Fehlerbeschreibung

Sie konnten Ihr Custom Web Control erfolgreich aus dem TIA Portal in Runtime laden. Wenn Sie das Bild öffnen, sehen Sie seine Ränder, aber keinen Inhalt.

Abbildung 5-5



Lösung

Prüfen Sie zunächst, ob eine Verbindung zwischen Ihrem Custom Web Control und WinCC Unified hergestellt wurde. Öffnen Sie dazu die Entwicklerkonsole mit F12 und klicken Sie dann auf "Konsole" wie in Abbildung 5-5. Wenn Ihnen nicht die Erfolgsmeldung "erfolgreich Verbunden" wie in dar Abbildung dargestellt, angezeigt wird, wurde keine Verbindung hergestellt.

Überprüfen Sie in diesem Fall, dass die Funktion `"WebCC.Start()"` direkt beim Start des Custom Web Control aufgerufen wird.

Wurde eine Verbindung hergestellt, aber es erscheint immer noch nichts auf dem Bild, dann liegt ein Fehler im Code, den Sie geschrieben haben, vor. Debuggen Sie Ihren Code und beheben Sie den Fehler (vgl. Abschnitt **Fehler! Verweisquelle konnte nicht gefunden werden.**).

5.4 Custom Web Control enthält keine WinCC-Daten

Fehlerbeschreibung

Der Browser zeigt Ihren Custom Web Control korrekt an. Wenn Sie aber die verknüpften Variablen ändern, sehen Sie keinerlei Wertänderung im Custom Web Control.

Lösung

Überprüfen Sie zunächst, dass die Schreibweise der Variablen überall in Ihrem Code und der Manifest.json-Datei korrekt ist. Achten Sie dabei besonders auf eine übereinstimmende Groß- und Kleinschreibung.

Prüfen Sie als nächstes, dass Sie die richtige Variable verknüpft haben des Werts, den Sie anpassen möchten. Geben Sie bei Bedarf die Variable neben dem Custom Web Control in einem E/A-Feld aus.

Wenn das E/A-Feld aktualisiert wird und alle Variablen im TIA Portal korrekt verbunden sind, der Wert aber immer noch nicht im Custom Web Control angezeigt wird, dann liegt der Fehler im Code.

Kontrollieren Sie, ob Sie nach erfolgreicher Verbindung die Funktion "`WebCC.onPropertyChanged.subscribe()`" aufgerufen und eine Callback-Funktion übergeben haben (siehe Kapitel Konfiguration und Implementierung des Custom Web Controls, Schritt 5).

Setzen Sie mit der Debugging-Funktion (siehe Kapitel **Fehler! Verweisquelle konnte nicht gefunden werden.**) einen Haltepunkt am Beginn der Callback-Funktion (in Zeile 145 in diesem Beispiel).

Ändern Sie nun den verknüpften Wert und beobachten Sie, ob der Haltepunkt erreicht wird.

Wenn der Browser am Haltepunkt nicht stoppt, prüfen Sie, ob Sie beim Aufruf von "`WebCC.Start()`" die Eigenschaft mit dem richtigen Namen übergeben haben (siehe Zeile 263 im Beispiel).

Hier sehen Sie ein weiteres gültiges Beispiel für einen korrekten Aufruf der Funktion:

```
WebCC.Start(function(result) {
   if (result) {
      console.log('connected successfully');
      WebCC.onPropertyChanged.subscribe((data) => {
        console.log(data);
      });
    }
  }, {
    properties: {MyIntProperty: 0, MyStringProperty: 'test'}
  }, [], 10000 // timeout
);
```

Das Custom Web Control hat die Eigenschaften "MyIntProperty" und "MyStringProperty" von WinCC Unified angefordert. Diese müssen auch in der Manifest.json-Datei vorhanden sein. Beachten Sie die Groß- und Kleinschreibung und korrekte Schreibweise.

5.5 Custom Web Control kann keine WinCC-Daten schreiben

Fehlerbeschreibung

Der Browser zeigt Ihr Custom Web Control korrekt an.

Ihr Custom Web Control ist so programmiert, dass es Eigenschaften schreiben kann. Die Eigenschaften sind mit WinCC-Variablen im TIA Portal verknüpft. Das Custom Web Control sollte demnach die WinCC-Variablen direkt ändern.

Wenn Ihr Custom Web Control die Eigenschaft schreibt, sehen Sie keinerlei Änderung an den WinCC-Variablen.

Lösung

Überprüfen Sie zunächst, dass die Schreibweise der Variablen überall in Ihrem Code und der Manifest.json-Datei korrekt ist. Achten Sie dabei besonders auf eine übereinstimmende Groß- und Kleinschreibung.

Prüfen Sie als nächstes, dass Sie die richtige Variable verknüpft haben des Werts, den Sie anpassen möchten. Geben Sie bei Bedarf die Variable neben dem Custom Web Control in einem E/A-Feld aus.

Wenn das E/A-Feld nicht aktualisiert wird, liegt der Fehler im Code. Prüfen Sie, ob Sie die Codezeile erreichen, in die Sie den Wert schreiben.

Setzen Sie mithilfe der Debugging-Funktion (siehe Abschnitt **Fehler! Verweisquelle konnte nicht gefunden werden.**) einen Haltepunkt in der Zeile, in die Sie die Eigenschaft schreiben. Beobachten Sie, ob der Haltepunkt erreicht wird.

Wenn der Browser an dem Haltepunkt stoppt und sich der Variablenwert in WinCC nicht ändert, prüfen Sie nochmals, ob Sie die richtige Eigenschaft "Schreiben" verwendet haben.

Ein weiteres Beispiel für eine korrekte Schreibweise:

```
WebCC.Properties.MyIntProperty = 5;
```

Das Custom Web Control schreibt die Eigenschaft "MyIntProperty". Diese muss auch in der Manifest.json-Datei vorhanden sein. Beachten Sie die Groß- und Kleinschreibung und korrekte Schreibweise.

6 Hilfreiche Informationen

6.1 Tipps & Tricks

Versionsmanagement

Das Versionsmanagement bietet einige Vorteile, zum Beispiel, dass Ihr Code nochmals gespeichert wird. Ein weiterer Vorteil ist die Nachverfolgung von Änderungen, falls Sie Fehler nicht zurückverfolgen können und zurück zu einer stabilen Version wollen.

Zu den Plattformen für das Versionsmanagement zählt http://github.com/.

Aktualisierung des Custom Web Control bei seiner Erstellung

In Abschnitt "Installation und Integration in das Projekt des Benutzers” wird erklärt, wie das Custom Web Control in Runtime geladen wird. Wenn Sie bei der Erstellung viel experimentieren müssen, müssen Sie die Schritte häufiger durchlaufen. Mit den folgenden Vorgehensweisen können Sie Zeit sparen.

Es existiert eine weitere Methode zur Programmierung eines Custom Web Controls:

1. Erstellen Sie die Manifest.json-Datei so vollständig wie möglich.
2. Erstellen Sie eine leere "index.html" und "code.js"-Datei.
3. Erstellen Sie ein Custom Web Control aus diesen drei Dateien und integrieren Sie es (vgl. Kapitel "Installation und Integration in das ").
4. Starten Sie Ihren Browser und zeigen Sie Ihren leeren Custom Web Control an.
5. Gehen Sie zum Code der Online-Instanz des Custom Web Control. In Windows ist er hier hinterlegt: "C:\Users\Public\Documents\Siemens\WebUX_ResourceCache_CustomWebControls".
6. Passen Sie den Code an.
7. Drücken Sie F5, um die Unified Runtime Website im Browser zu aktualisieren.
8. Das Custom Web Control wurde nun aktualisiert, ohne dass er erneut heruntergeladen werden musste.
9. Wiederholen Sie die Schritte 6 und 7, bis Sie mit der Programmierung fertig sind.

VORSICHT	**Bei diesem Vorgang kann Ihr Code verloren gehen!** Wenn Sie die Programmierung abgeschlossen haben, speichern Sie den Code der Online-Instanz des Custom Web Controls separat. Bei einem erneuten Start von Runtime überschreibt WinCC Unified Ihren Code mit dem Original-Code. Wenn Sie das Manifest.json-Datei zu einem späteren Zeitpunkt ändern müssen, speichern Sie auch dann Ihren Code separat, da Sie sonst wieder von Schritt 3 beginnen müssten. Ein erneutes Herunterladen des Custom Web Controls führt ebenfalls zu einer Überschreibung Ihres Online-Codes.

6.2 Alternativlösungen

Runtime ODK und OpenPipe

Custom Web Controls werden verwendet, um Daten zwischen dem Operator und den Daten in WinCC Unified (wie SPS-Variablen) auszutauschen.

Wenn Sie das Custom Web Control verwenden, um Daten von anderen Webservices zu nutzen, sollten Sie sich fragen, ob Sie die Daten nur für einen bestimmten Operator benötigen, oder ob die Daten für alle Operators relevant sind.

Wenn die Daten für eine spezifische Zeitspanne vorgesehen sind und WinCC Unified diesem Operator die Daten direkt mit einem Custom Web Control zeigen muss, ist das Custom Web Control die richtige Wahl.

Wenn Sie hingegen externe Daten mit dem Custom Web Control abfragen und Variablen in WinCC mit Eigenschaften zurückschreiben, um die Daten mehreren oder allen Anwendern zur Verfügung zu stellen, dann haben Sie möglicherweise das Problem, dass mehrere Anwender Daten abfragen und Variablen in WinCC schreiben. Dies ist eine unnötige Mehrbelastung für Ihren anderen Webservice und auch für den WinCC Unified Runtime Server.

Es gibt jedoch eine schnellere und bessere Lösung: Schreiben Sie ein Programm, das auf dem WinCC Unified Runtime Server läuft und die Daten abgreift. Dieses Programm kann dann die Daten über die OpenPipe- oder Runtime ODK-Schnittstelle und Variablen in WinCC schreiben.

Wenn Sie trotzdem eine bestimmte Darstellungsform benötigen, die WinCC Unified derzeit nicht bietet, erstellen Sie ein Custom Web Control, das ausschließlich diese WinCC-Tags verwendet.

Auf diese Weise überlassen Sie dem WinCC Unified Server die Themen Authentifizierung, Autorisierung und künftige Redundanzen und haben gleichzeitig im Custom Web Control weniger Entwicklungsaufwand.

7 Anhang

7.1 Service und Support

Industry Online Support

Sie haben Fragen oder brauchen Unterstützung?

Über den Industry Online Support greifen Sie rund um die Uhr auf das gesamte Service und Support Know-how sowie auf unsere Dienstleistungen zu.

Der Industry Online Support ist die zentrale Adresse für Informationen zu unseren Produkten, Lösungen und Services.

Produktinformationen, Handbücher, Downloads, FAQs und Anwendungsbeispiele – alle Informationen sind mit wenigen Mausklicks erreichbar:

support.industry.siemens.com

Technical Support

Der Technical Support von Siemens Industry unterstützt Sie schnell und kompetent bei allen technischen Anfragen mit einer Vielzahl maßgeschneiderter Angebote
– von der Basisunterstützung bis hin zu individuellen Supportverträgen.

Anfragen an den Technical Support stellen Sie per Web-Formular:

https://support.industry.siemens.com/cs/my/src

SITRAIN – Digital Industry Academy

Mit unseren weltweit verfügbaren Trainings für unsere Produkte und Lösungen unterstützen wir Sie praxisnah, mit innovativen Lernmethoden und mit einem kundenspezifisch abgestimmten Konzept.

Mehr zu den angebotenen Trainings und Kursen sowie deren Standorte und Termine erfahren Sie unter:

siemens.de/sitrain

Hinweis

Trainings zu SIMATIC WinCC Unified und Unified Comfort Panels finden Sie unter diesen Links:

- SITRAIN System Course "WinCC Unified & Unified Comfort Panels"
  https://support.industry.siemens.com/cs/ww/en/view/109773211
- SITRAIN advanced course: SIMATIC WinCC Unified for PC systems
  https://support.industry.siemens.com/cs/ww/en/view/109781323
- SITRAIN access: WinCC Unified Scripting (JavaScript)
  https://support.industry.siemens.com/cs/ww/en/view/109782872

Serviceangebot

Unser Serviceangebot umfasst folgendes:

- Plant Data Services
- Ersatzteilservices
- Reparaturservices
- Vor-Ort und Instandhaltungsservices
- Retrofit- und Modernisierungsservices
- Serviceprogramme und Verträge

Ausführliche Informationen zu unserem Serviceangebot finden Sie im Servicekatalog:

support.industry.siemens.com/cs/sc

Industry Online Support App

Mit der App "Siemens Industry Online Support" erhalten Sie auch unterwegs die optimale Unterstützung. Die App ist für iOS und Android verfügbar:

support.industry.siemens.com/cs/ww/de/sc/2067

7.2 Industry Mall

Die Siemens Industry Mall ist die Plattform, auf der das gesamte Produktportfolio von Siemens Industry zugänglich ist. Von der Auswahl der Produkte über die Bestellung und die Lieferverfolgung ermöglicht die Industry Mall die komplette Einkaufsabwicklung – direkt und unabhängig von Zeit und Ort:

mall.industry.siemens.com

7.3 Links und Literatur

Tabelle 7-1

Nr.	Thema
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link auf die Beitragsseite des Anwendungsbeispiels https://support.industry.siemens.com/cs/ww/de/view/109779176
\3\	Microsoft Visual Studio Code 1.44 https://code.visualstudio.com/
\4\	Gauge.js 1.3.7 https://bernii.github.io/gauge.js/
\5\	Tabulator v4.9 http://tabulator.info/
\6\	SITRAIN System Course "WinCC Unified & Unified Comfort Panels" https://support.industry.siemens.com/cs/ww/en/view/109773211
\7\	SITRAIN advanced course: SIMATIC WinCC Unified for PC systems https://support.industry.siemens.com/cs/ww/en/view/109781323
\8\	SITRAIN: You can find out more about the training and courses as well as their locations and dates at: https://www.siemens.com/sitrain

7.4 Änderungsdokumentation

Tabelle 7-2

Version	Datum	Änderung
V1.0	06/2020	Erste Ausgabe
V2.0	04/2021	Custom Web Control "Table" hinzugefügt und Dokument neu strukturiert
V3.0	06/2023	Kapitel 2.2.4 "Autocompletion" neu hinzugefügt. Aktualisiert auf TIA V18 / WinCC Unified V18