Web Addon

Sprache: **Deutsch** · English

Web Addon

> **Den Web Client per Angular/TypeScript erweitern – Dev-Setup, Live-Reload und ein Beispiel für berechnete Felder.**

HOW-TO | ANWENDUNGSENTWICKLER | STAND: JUL 2026 | GILT FUER NUCLOS 4.2026.X

Auf dieser Seite

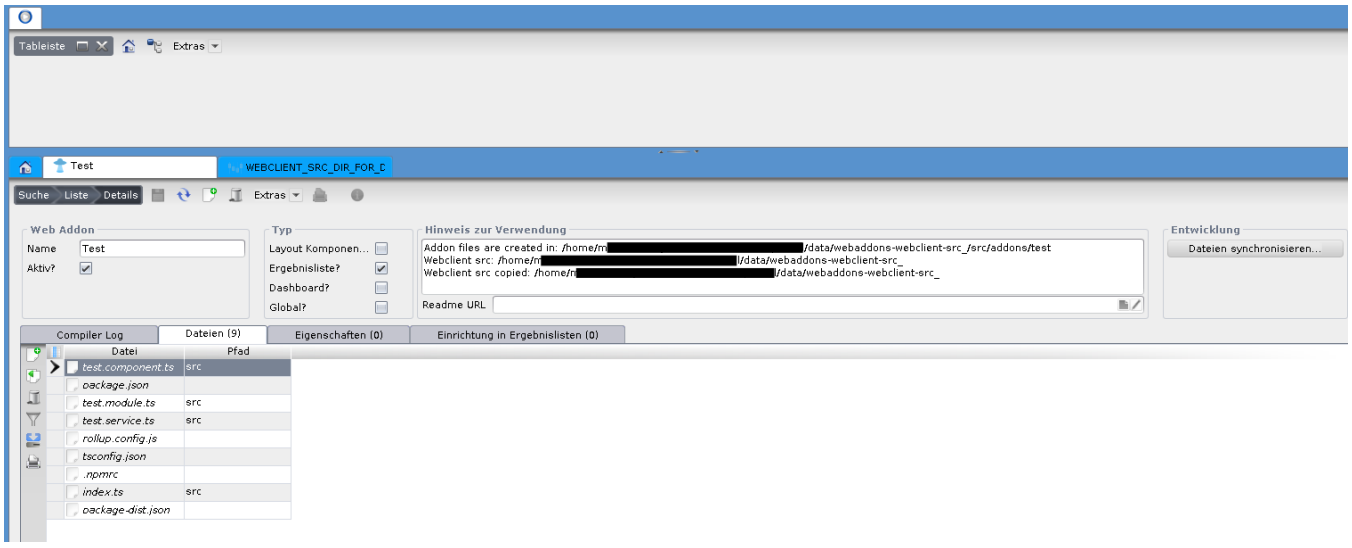
Was sind Web Addons?

Menüaufruf: Konfiguration Web Addon.

Mit **Web Addons** erweiterst du den Web Client flexibel per Angular/TypeScript – z.B. um clientseitige Berechnungen in Unterformularen.

Voraussetzungen

- Lokale Dev-Instanz mit **Node.js** (ab 4.2024.x v18.20.2; davor v12) – z.B. über `nvm`, plus `npm`.
- Editor (z.B. Visual Studio Code).
- Nuclos-Server im **DEV-Modus**.



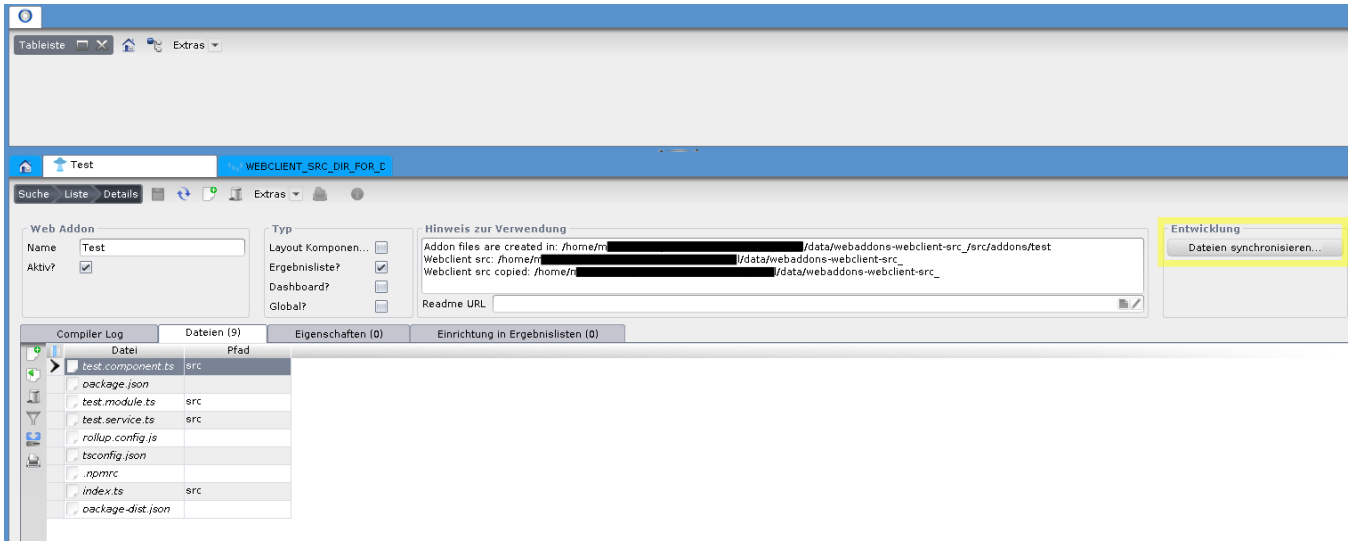
Web-Addon in der Verwaltung anlegen.

Lokale Entwicklung einrichten

1. Addon anlegen und einem Nuclet zuweisen. Es liegt dann unter `[instanz]/data/webaddons-webclient-src/addons`.
2. Eine Kopie dieses Verzeichnisses anlegen (z.B. `webaddons-webclient-src_`) und den Parameter `WEBCLIENT_SRC_DIR_FOR_DEVMODE` darauf zeigen lassen; Serverneustart.
3. In `src/assets/config.json` die `nuclosURL` auf den lokalen Server setzen.
4. Abhängigkeiten installieren und Dev-Server starten:

```
npm install
npm run install-addon-deps
npm run start --webclient-host="127.0.0.1" --webclient-port=4200
```

Änderungen werden dank npm-Watcher direkt im Browser sichtbar. Fertige Anpassungen in der Web-Addon-Verwaltung ins Nuclet übernehmen und exportieren.



Fertiges Addon ins Nuclet übernehmen.

Beispiel: Berechnete Felder

Ein globales Addon berechnet clientseitig Werte in Unterformularen. `test_SubBo_refMainBo` ist der BO-Name des Unterformulars; die Fachlogik steht in `calculate_Attribute()` und wird bei jeder Änderung der beobachteten Attribute aufgerufen.

Fehler beim Rendern des Makros 'code': Ungültiger Wert für den Parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
import { Injectable } from '@angular/core';
import {GlobalContext, ISubEntityObject, Logger} from "@nuclos/nuclos-addon-api";
import {
        IEntityObject,
        IEntityObjectDependents,
        IEntityObjectEventListener
} from "@nuclos/nuclos-addon-api/api/entity-object";

@Injectable({
        providedIn: 'root'
})

class CalculationRule {

        constructor(public attributeNames : string[], public targetAttributeName : string, public funcCalculate :
(eo : IEntityObject, attributeName : string) => number) {
        }
}

@Injectable()
export class BerechneteFelderService {
        private attributeChangeListener : IEntityObjectEventListener;
        private selectedEo : IEntityObject;
        private calculationRules : CalculationRule[];

        callForDependents(eo : IEntityObject, func: {(entity:ISubEntityObject):void}) :any {
                eo.getDependents("test_SubBo_refMainBo").asObservable().subscribe(subEntities => {
                        if( subEntities === undefined)
                                return;
                        for(let i in subEntities) {
                                let entity : ISubEntityObject = subEntities[i];
                                func(entity);
                                this.logger.log('BerechneteFelder: onEoSelection() ' + func.prototype.name);
                        }
                })
```

```
        }

        constructor(private addonCtx: GlobalContext, private logger: Logger) {
                this.logger.log('BerechneteFelder: constructor');
                this.calculationRules = [];
                this.calculationRules.push(new CalculationRule(["subbo_attribute1", "subbo_attribute2"],
"subbo_attribute3", calculate_Attribute));
                this.attributeChangeListener = {
                        afterAttributeChange: (entityObject: IEntityObject, attributeName: string, oldValue: any,
newValue: any) => {
                                this.logger.log("BerechneteFelder: entityObject: " + entityObject.
getEntityClassId() + ", attributeName: " + attributeName +
                                        ", oldValue: " + oldValue + ", newValue: " + newValue);
                                this.calculationRules.forEach(calculationRule => {
                                        if(calculationRule.attributeNames.includes(attributeName)) {
                                                let result = calculationRule.funcCalculate(entityObject,
attributeName);
                                                entityObject.setAttribute(calculationRule.
targetAttributeName, result);
                                        }
                                })
                        }
                }

                addonCtx.getEntityObjectApi().onEoSelection().subscribe(eo => {
                        this.logger.log('BerechneteFelder: onEoSelection()');
                        if (this.selectedEo !== undefined) {
                                this.selectedEo.getDependents("test_TestBo_ref1").asObservable().subscribe
((entities => {
                                        entities?.forEach((subEo) => subEo.removeListener(this.
attributeChangeListener));
                                }))
                        }

                        if (eo !== undefined) {
                                eo.getDependents("test_TestBo_ref1").asObservable().subscribe((entities => {
                                        entities?.forEach((subEo) => {
                                                subEo.addListener(this.attributeChangeListener)
                                        });
                                }))
                                this.selectedEo = eo;
                        }
                })
                addonCtx.getEntityObjectApi().onEoModification().subscribe(eo => {
                });
        }
}

function calculate_Attribute(eo : IEntityObject, attributeName : string) : number {

        let dAttribute1 = eo.getAttribute("subbo_attribute1");
        let dAttribute2 = eo.getAttribute("subbo_attribute2");

        return dAttribute1 * dAttribute12;
}
```

Verwandte Seiten

Nuclet

Nuclet.

Öffnen

Regeln (clientseitig)

Clientseitig.

Öffnen