Web Addon

Language: Deutsch · **English**

Web Addon

> **Extend the web client with Angular/TypeScript – dev setup, live reload and an example of calculated fields.**

HOW-TO | APPLICATION DEVELOPER | UPDATED: JUL 2026 | APPLIES TO NUCLOS 4.2026.X

On this page

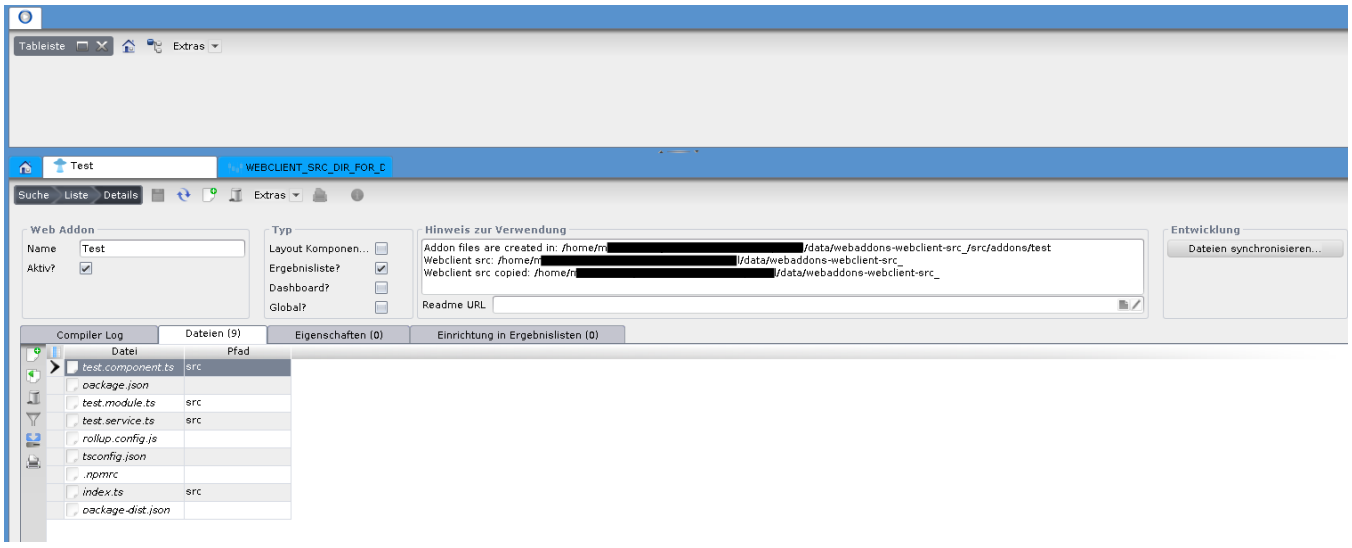
What are web addons?

Menu: Configuration Web Addon.

With **web addons** you flexibly extend the web client using Angular/TypeScript – e.g. with client-side calculations in subforms.

Prerequisites

- Local dev instance with **Node.js** (from 4.2024.x v18.20.2; earlier v12) – e.g. via `nvm`, plus `npm`.
- Editor (e.g. Visual Studio Code).
- Nuclos server in **DEV mode**.



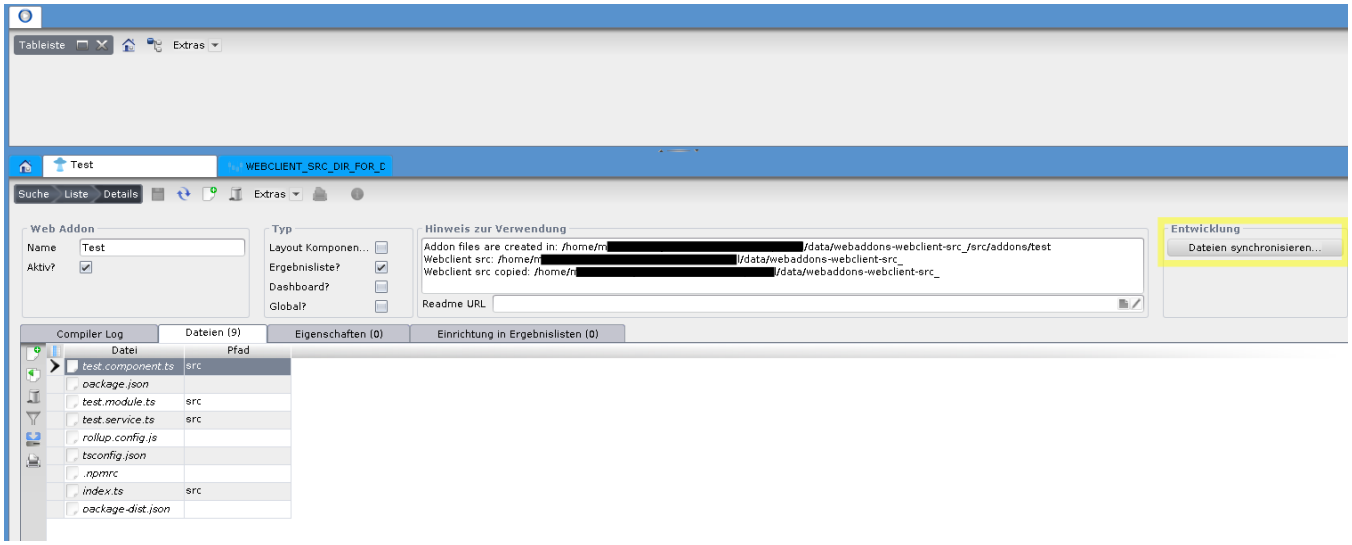
Create a web addon in the administration.

Set up local development

1. Create the addon and assign it to a nuclet. It is then located under `[instance]/data/webaddons-webclient-src/addons`.
2. Create a copy of this directory (e.g. `webaddons-webclient-src_`) and point the parameter `WEBCLIENT_SRC_DIR_FOR_DEVMODE` to it; restart the server.
3. In `src/assets/config.json` set `nuclosURL` to the local server.
4. Install dependencies and start the dev server:

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

Thanks to the npm watcher, changes are visible directly in the browser. Transfer finished changes into the nuclet in the web addon administration and export them.



Transfer the finished addon into the nuclet.

Example: calculated fields

A global addon calculates values client-side in subforms. `test_SubBo_refMainBo` is the subform's BO name; the business logic is in `calculate_Attribute()` and is called whenever the observed attributes change.

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;
}
```

Related pages

Nuclet

Nuclet.

Open

Client-side rules

Client-side.

Open