Exkurs: Integration des Wetter-Services in ein Angular-Projekt

Folgende Schritte müssen gesetzt werden:

1. Die mitgelieferten Klassen `Station`, `StationValley`, `Measurement`, `Station-Factory` und `WeatherService` müssen in den Ordner `shared` kopiert werden.

   `Station`
   Ist die Basisklasse von `StationValley` welche für eine Wetterstation im Tal steht. Sie enthält alle Eigenschaften, die allen Arten von Wetterstationen eigen sind sowie die Eigenschaften `descriptions` und `units` (siehe nachfolgende Tabelle) welche für eine Messwertabkürzung deren Beschreibung und Einheit zurück liefern.

   `StationValley`
   Diese Klasse enthält alle Messwerte welche für eine *Messstation im Tal* geliefert werden.

   `Measurement`
   Diese Klasse enthält die Links zu den historischen Diagrammen der einzelnen Messwerte.

   `StationFactory`
   Diese Klasse stellt die Methode `fromObject()` bereit, welche aus einer Station die im JSON-Format vorliegt ein *TypeScript-Objekt* generiert.

   `WeatherService`
   Diese Klasse ruft mit `getAll()` alle Wetterstationen die im Tal liegen mit ihren Daten ab und liefert diese als `StationValley`-Objekte zurück.

2. Es muss der Wetter-Service in der Anwendung registriert und das `HttpModule` eingebunden werden. Dazu müssen in der Datei `app.module.ts` folgende Einstellungen gemacht werden:

```
…
import { HttpClientModule } from '@angular/common/http';
import { WeatherService } from './shared/weather-service';

@NgModule({
  imports: [ … HttpClientModule[a] ],
  providers: [ WeatherService[b] ]
})
export class AppModule { }
```

   [a] Bindet das `HttpClientModule` ein, so dass die Klasse `HttpClient` verwendet werden kann und durch sie Daten von anderen Quellen nachgeladen werden können. Dabei werden von Angular auch das *http-* und *RxJS Framework* (Reactive Extensions for JavaScript) mit eingebunden und in den Dependencies angezeigt.

   [b] Dadurch wird der Service für die gesamte Anwendung bekannt gemacht, von der Anwendung ein `WeatherService`-Objekt angelegt, welches allen Komponenten mittels *Constructor Injection* übergeben werden kann (siehe nächster Punkt).

3. Durch *Constructor Injection* wird nun der Komponente die den Wetter-Service verwenden will, dieser im Konstruktor mitgegeben. Weiters wird nach erfolgreichem Laden der Komponente der Wetter-Service abgefragt und bei Vorhandensein des Response dieser in das Array `stations` geschrieben:

```
export class StationListComponent implements OnInit {
  stations!: StationValley[];

  constructor(private ws: WeatherService) { }

  ngOnInit() {
    this.ws.getAll().subscribe(res => {
      this.stations = res;

      // Es wird nach Name sortiert
      this.stations.sort((s1, s2) => s1.name.localeCompare(s2.name));

      // Es werden nur jene Stationen angezeigt die mit S beginnen
      this.stations = this.stations.filter(s => s.name.startsWith('S'));
    });
  }
}
```

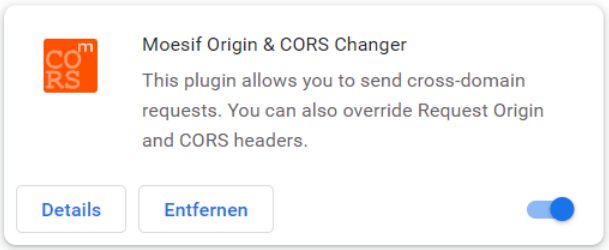
Das CORS-Problem

*(**Quelle:** https://medium.com/@dtkatz/3-ways-to-fix-the-cors-error-and-how-access-control-allow-origin-works-d97d55946d9)*

Bei der Verwendung des Web-Services, erscheint in der Konsole des Browsers folgende Fehlermeldung, und keine Wetterdaten werden geliefert:

```
Access to XMLHttpRequest at 'http://daten.buergernetz.bz.it/services/weather/station?c  localhost/:1
ategoryId=1&lang=de&format=json' from origin 'http://localhost:4200' has been blocked by CORS
policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
```

Diese besagt, dass der Browser aus Sicherheitsgründen Zugriffe auf entfernte Server unterbindet, die explizit dafür nicht die Berechtigung gegeben haben. Da Ihre Web-Anwendung auf `http://localhost:4200` läuft, der Web-Service aber unter `http://daten.buergernetz.bz.it` zu finden ist und dieser nicht die CORS-Berechtigung erteilt hat, wird der Zugriff unterbunden.



Um für *Testzwecke auf Ihrem lokalen PC* trotzdem den Zugriff zu erlauben, kann in Chrome das Plugin **Moesif Origin & CORS Changer** installiert und aktiviert werden.

Bemerkung

Im Kapitel zu Web-Services wird Ihnen gezeigt werden, wie Sie einen Web-Service programmieren und so konfigurieren können, dass er Zugriffe von anderen Servern aus erlaubt.

Nachfolgend finden Sie noch die Beschreibung der Eigenschaften der Klassen `Station` und `StationValley`.

Beschreibung der Eigenschaften der Klassen `Station` und `StationValley`

Station	StationValley
`altitude`	dd Windrichtung
`categoryId`	ff Mittlere Windgeschwindigkeit
`code`	bb Windböe
`id`	n Niederschlagsmenge seit Mitternacht
`lastUpdated`	p Luftdruck
`latitude`	rh Relative Luftfeuchtigkeit
`longitude`	t Lufttemperatur
`name`	sd Sonnenscheindauer
`measurements`	gs Globalstrahlung

Für Interessierte folgen noch die Eigenschaften der Berg- und Pegelstationen:

Wetterstation am Berg	Pegelstation
dd Windrichtung	q Durchfluss
ff Mittlere Windgeschwindigkeit	w Wassertsand
bb Windböe	wt Wassertemperatur
hs Schneehöhe	
rh Relative Luftfeuchtigkeit	
t Lufttemperatur	
sd Sonnenscheindauer	
gs Globalstrahlung	