Angular en NativeScript {N}

Bouw je eigen mobiele apps

Steeds meer Java ontwikkelaars komen de laatste jaren in aanraking met front-end ontwikkeling. Voor deze groep ontwikkelaars is het nog maar een kleine stap om tot mobiele app-ontwikkeling over te gaan door middel van NativeScript. In dit artikel proberen we duidelijk te maken wat de mogelijkheden zijn van native hybride app ontwikkeling met behulp van NativeScript.

Wat is NativeScript {N}?

{N} is een open source framework (onder de Apache 2 licence) om native IOS en Android apps te bouwen met behulp van Typescript en Angular. {N} is een andere technologie dan de hybride-frameworks, zoals Ionic en Phonegap. {N} is een runtime, geen web technologie. Je app zal niet draaien als een mini website in een WebView en is daarom meer performant. Met {N} heb je direct toegang tot al de Native APIs van je device.

TypeScript:

Voor Javanen is het interessant om te beseffen dat TypeScript (TS) heel veel overeenkomsten met Java heeft. Je kunt code kloppen waarvan je je afvraagt of het TS of Java is. TS is de beste manier om {N} apps te schrijven, al dan niet gecombineerd met Angular. Verder is het schrijven van native code in TS voor Android vrij eenvoudig, omdat je in de native Java code veel één-op-één kunt overnemen (bijvoorbeeld: new io.java.File() is zowel geldige Java als JS/TS. Dat ligt bij Objective C net even wat lastiger). Het is heel fascinerend om te zien hoe NativeScript het voor elkaar krijgt om met puur JavaScript alle native constructies te implementeren. Van String Arrays tot interfaces en implementeren van abstract classes.

Waarom NativeScript?

- Eén van de argumenten om NativeScript {N} te gebruiken, is hergebruik van 'Skills'. Dat wil zeggen iemand met kennis van JavaScript / Typescript kan direct aan de slag met {N}. {N} wordt namelijk geschreven in JavaScript of Typescript.
- Hergebruik van code. Schrijf Native Mobile apps voor iOS en Android met één enkele codebase en één gezamelijke interface. Dit is bijvoorbeeld niet het geval bij Xamarin waar je toch twee interfaces moet bouwen voor IOS en Android met één common layer. Het is niet zo vreemd dat de slogan van {N} luidt: 'Write once, run everywhere'.
- Het is tevens gemakkelijk uit te breiden met behulp van {N} modules (zie de voorbeelden in dit artikel) en npm modules. Eigenlijk de plug-ins van Cordoba / Phonegap.
- Tenslotte wordt geen gebruik gemaakt van WebViews, zoals bij vele hybride frameworks. Met JavaScript hebt je direct toegang tot de Native APIs en is daardoor beter performant.

Wat is er nodig voor NativeScript?

Op **https://docs.nativescript.org/start/quick-setup** staat een duidelijke beschrijving hoe {N} moet worden geïnstalleerd. Omdat deze installatie niet altijd feilloos verloopt, is het verstandig om 'NativeScript Sidekick' te installeren.

Met Sidekick is het mogelijk om apps te builden in de cloud. Als je ervoor kiest om je apps te builden in de cloud, dan kan je onafhankelijk van je operating system waarop je werkt je apps ontwikkelen. Je kan dan uiteraard zelfs iOS apps builden op een Windows machine. Het is ook mogelijk om met Sidekick lokaal aan de slag te gaan, maar daarvoor moet je je eigen omgeving inrichten met iOS Xcode en Android SDK. Sidekick heeft nog veel known issues, maar werkt al aardig. Het mooie aan {N} is ook Live Sync. Dus als je een regel in de code aanpast, zal dit zelfs met de cloudbuild direct op de telefoon worden 'gepushed' en zo is het resultaat van de wijziging direct zichtbaar.

Peter Eijgermans is een enthousiaste Java Web Developer bij Ordina JTech. Onlangs geridderd tot Codesmith.

Rogier van Apeldoorn is een innovatieve Mobile Developer en Codesmith bij Ordina JTech.

Hoe werkt NativeScript met behulp van JavaScript?

In **Listing 1** volgt een simpel voorbeeld van JavaScript in {N}, die een Objective-C based IOS UIAlertView control instantieert.

```
var myAlert = new UIAlertView();
myAlert.message = "NativeScript rocks!";
myAlert.show();
```

Listing 1

Omdat web developers geen iOS en Android specifieke APIs willen aanleren, biedt {N} een set van {N} modules aan. Die modules abstraheren de iOS en Android details in simpele JavaScript APIs. De bovenstaande UIAlertView-gebaseerde code kan herschreven worden met de {N} 'Dialog module' (zie **Listing 2**).

```
var dialogs = require("ui/dialogs");
dialogs.alert({ message: "NativeScript rocks!" });
```

Listing 2

Deze dialogs.alert()call levert ons ook de android.app.AlertDialog voor je Android app. Ondanks dat dit 'dialog'' voorbeeld eenvoudig is, kan dezelfde techniek worden ingezet voor het bouwen van robuuste apps. Hiervoor kun je gebruik maken van de reeds bestaande en volwassen native iOS en Android UI componenten.

Wat kan Angular toevoegen aan NativeScript?

{N} kan nu ook geschreven worden in Angular. Als je kennis hebt van Angular, dan is het een kleine stap om Angular in {N} te gebruiken. Het grote verschil met Angular is dat de browser-based HTML-elementen, zoals en , niet beschikbaar zijn in {N}. Je dient daarvoor in de plaats {N} UI componenten te gebruiken. In {N} wordt geen DOM of browser gebruikt. {N} UIs zijn native UIs en dus losgekoppeld van de DOM. Omdat Angular een agnostisch framework is en losgekoppeld is van de DOM, kan dit framework gemakkelijk geïntegreerd worden met {N}. Onderstaand voorbeeld gaat hierop in. *AngularJS* is in tegenstelling tot Angular niet geschikt voor {N}, omdat dit framework gekoppeld is aan de DOM.

Door gebruik te maken van Angular in {N}, heb je de mogelijkheid om code te delen tussen je bestaande webapplicatie en je Native apps. Laten we kijken naar een voorbeeld (zie **Afbeelding 2**).

Hergebruik van code tussen web en mobile apps

Laten we als voorbeeld een 'Grocery List' tonen in een webapplicatie (zie **Listing 3**). In **Listing 3** is een Angular Component gedefinieerd, die een array van 'groceries' vult, via de constructor. Er wordt gebruik gemaakt van TypeScript. Dit is een 'typed superset' van JavaScript en is de standaard voor het schrijven van Angular applicaties.

```
import {Component} from '@angular/core';

@Component({
    selector: 'grocery-list'
    templateUrl: 'grocerylist.template.html'
})

export class GroceryListComponent {
    groceries: string[];

    constructor()  {
        this.groceries =['milk, 'bread']
    }
}
```

Listing 3

```
....
    <grocery-list></grocery-list>
....
```

Listing 4

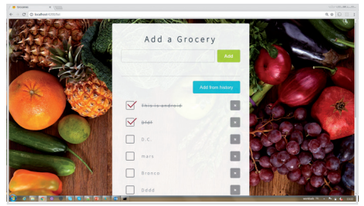
```
<p>Groceries: </p>
  <ul>
      <li *ng-for = "let grocery of groceries">
                {{grocery}}
      </li>
  </ul>
```

Listing 5

Hoe werkt dit component?

Via de *selector* behorend bij het Component, kan je Angular vragen om dit component te instantiëren en te renderen, waar het een <grocery-list> - tag vindt in de HTML (zie **Listing 4**).

Voor het renderen van de 'Grocery list', dient er tenslotte een template gedefinieerd te worden met de naam: grocerylist.template.html (zie **Listing 5**). Deze template staat ook gedefinieerd onder de *templateUrl* van het Component. Er wordt door de lijst met 'groceries' geïte-



Afbeelding 2

Afbeelding 1

```
<StackLayout *ngFor = “let grocery of groceries”>
                <Label [text] = “grocery”></Label>
</StackLayout>
```

Listing 6

```
var http = require(“http”);  à import HTTP module

http.getJSON(‘https://api.groceries.com’)
        .then(function (result)) {
                …
       });
```

Listing 7

reerd, met behulp van de *ng-for* directive van Angular. De vraag is nu: wat moeten we anders programmeren om bovenstaande code werkend te krijgen voor je NativeScript iOS en Android apps?

Het uitgebreide antwoord is: het component in **Listing 3** blijft hetzelfde. Deze TypeScript-code kan je dus hergebruiken tussen de webapplicatie en mobile apps. Het enige wat je dient te wijzigen is de template voor dit component. Je dient {N} UI componenten te gebruiken **in plaats van** HTML-elementen in je template (zie **Listing 6**). Wat opvalt is dat je nog steeds de *ng-for* directive van Angular kan gebruiken voor het itereren door de lijst.

Het omzetten van een template naar {N} UI componenten is een eenvoudige exercitie, omdat voor ieder HTML-element wel een Native UI component te vinden is. Dit voorbeeld laat ook zien dat Angular losgekoppeld is van de DOM en dus met {N} UI componenten kan omgaan.

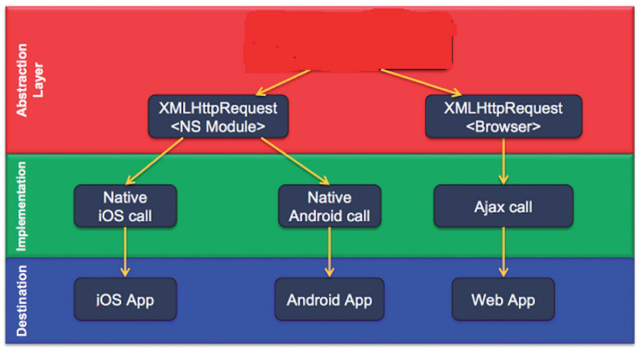
HTTP Module

Een ander voorbeeld voor code hergebruik is het volgende. {N} biedt support voor web APIs, zoals XMLHttpRequest en fetch(). Je kan hiervoor de HTTP module gebruiken binnen {N}. Deze module maakt het mogelijk om web requests te verzenden en web responsies te ontvangen (zie **Listing 7**).

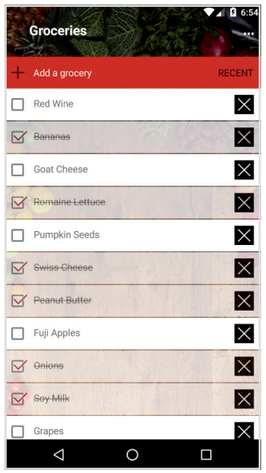
In **Afbeelding 4** zie je aan de linkerzijde dat {N} de XMLHttpRequests web API vertaalt naar Native code voor iOS en Android. Tevens wordt aan de rechterzijde van dit schema de implementatie van de XMLHttpRequest getoond die gebruikt wordt voor de webapplicatie. Dit is een andere implementatie dan de NativeScript variant.

Het probleem hier is dat we de XMLHttpRequest implementaties niet kunnen hergebruiken tussen de webapplicatie en mobiele app. Er is hier een ''missing link'', namelijk een generieke HTTP-module voor de webapplicatie en de mobiele app. Dit kan opgelost worden met Angular. Angular voegt een extra abstractielaag toe voor het afhandelen van HTTP-requests/responses. Dit kan met de HTTP-module van Angular. Deze module kunnen we hergebruiken tussen de webapplicatie en de mobiele app.

De te hergebruiken code ziet er dan bijvoorbeeld als volgt uit in Angular (zie **Listing 8**).

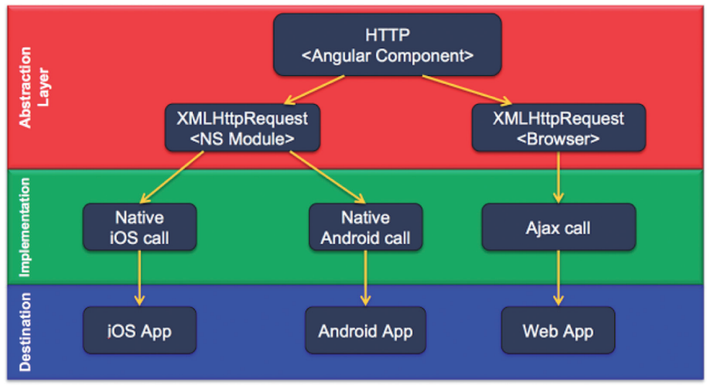


Afbeelding 4



Afbeelding 3

WRITE ONCE, RUN EVERYWHERE



Afbeelding 5

Meer tools en features

{N} biedt ons naast hierboven genoemde voorbeelden ook nog andere featues, zoals het gebruik van plug-ins, scaffolding en de {N} Playground. Verder kan je momenteel ook met het *Vue.js framework* een {N} app bouwen.

Plug-ins

Het interessante aan {N} is dat er veel plug-ins beschikbaar zijn in de marketplace. **https://market.nativescript.org**. Je hoeft dus niet alles zelf te bouwen. Denk aan Facebook, fingerprint authenticatie, 'text to speech', advertenties met Admob of in-app purchase. Mocht de plug-in er nog niet zijn, dan kan deze altijd zelf worden gebouwd.

NativeScript Playground

{N} Playground is een tool waarmee op eenvoudige wijze een app kan worden gebouwd met behulp van een web interface. Zie **https://play.nativescript.org**. Naast de website moet op een device twee apps worden geïnstalleerd:

- Nativescript Playground
- Nativescript Preview

Na het scannen van de QR-code van de **https://play.nativescript.org** website met behulp van de playground app, zal de app worden geïnstalleerd op het device. Met behulp van Live Sync zullen alle wijzigingen direct zichtbaar worden in de app. Supercool. Probeer het eens uit! Met Playground kan je eenvoudig je {N} code en issues delen met andere ontwikkelaars. Het is een middel om snel aan de slag te gaan met {N} en de voordelen ervan te ervaren.

Tot slot

Hergebruik van code is een zeer belangrijke uitdaging voor development in het algemeen. Op dit moment kan je met de *Angular CLI* een webapplicatie genereren. En voor {N} gebruik je de *NativeScript CLI*. Dit is een probleem, omdat de CLIs ons geen mogelijkheid geven om één project te maken voor de webapplicatie en de Native applicatie.

Natuurlijk is het mogelijk om twee aparte projecten te onderhouden en de gedeelde files tussen de twee projecten te 'copy-pasten'. Dit kan ook opgelost worden door bijvoorbeeld het volgende 'seed project' te gebruiken: **https://github.com/TeamMaestro/angular-native-seed**. Hiermee kan je met één codebase je webapplicatie en NativeScript onderhouden. Samenvattend, is {N} een krachtig framework om native cross-platform mobiele applicaties te bouwen met Angular, Vue.js, TypeScript of JavaScript. Je hebt direct toegang tot de Native platform API's. {N} werkt anders dan hybride frameworks, zoals Phonegap. Maar is uitermate geschikt voor webdevelopers. Het is beter performant en eenvoudig te leren.■

```
        import  {Http} from '@angular/http'

       export class GroceryListService {

          constructor(private http: Http)  {}

  getGroceries(){
  return this.http.get(`https://api.groceries.com`)
  .map((res:Response) => res.json());
          }

       }
```

Listing 8