Redux uitbreiding gebaseerd op Redux-store

Onderzoeksvoorstel Bachelorproef

thibault.gobert.w1869@student.hogent.be[1]

Samenvatting
Er is weinig duidelijkheid rond het samenvoegen van 2 React-Redux projecten. Dit kan interessant zijn om te weten of het mogelijk is om onafhankelijke Redux stores van verschillende projecten te hebben in 1 hoofdproject. Dit moet onderzocht worden omdat er momenteel weinig documentatie beschikbaar is rond het samenvoegen van 2 projecten die Redux gebruiken. Om te onderzoeken of dit mogelijk is, zal er een uitbreiding op Redux geschreven worden om aan deze functionaliteit te voldoen. Indien dit lukt, kunnen deze toekomstige problemen snel en efficiënt worden opgelost met de nodige documentatie/bevindingen.

Sleutelwoorden
Webapplicatieontwikkeling. Redux — Store — React

Co-promotor
Piet Pieters[2] (Bedrijfsnaam)

Contact: [1] thibault.gobert.w1869@student.hogent.be;

Inhoudsopgave

1. Introductie

Vooraleer er dieper op het probleem kan worden ingegaan is er meer context nodig rond een aantal basisbegrippen van React en Redux. Volgens de website van **Redux02** React is een JavaScript library. Met behulp van React worden views gecreëerd voor elke state in onze applicatie. Redux is een state container voor JavaScript apps. Redux helpt om de afgebeelde data te beheren en beschrijft hoe er gereageerd wordt op user actions. Een store is een object die de volledige state-tree van de applicatie bevat. De enige manier om een state te veranderen in de store is door een action te dispatchen. Een action zal er eigenlijk voor zorgen dat er data van de applicatie verzonden wordt naar de store. Dit is tevens de enige vorm van input voor de store. Actions beschrijven dus het feit dat er iets gebeurd is, maar ze beschrijven niet hoe de state van de applicatie verandert. Dit is de taak van een reducer, deze toont aan hoe de state veranderd is ten gevolge van een action die verzonden werd naar de store. Een container is dan weer verantwoordelijk voor het verkrijgen van data. Om die data te verkrijgen wordt er gebruik gemaakt van de connect functie (voor de store) en een functie die de data uit de state neemt en deze mapt naar zijn eigen props. Een container is ook verantwoordelijk voor het dispatchen van actions. Het probleem zit bij het samenvoegen van 2 react-redux apps tot 1 app. Er is geen manier bekend om beide stores te behouden in 1 project. Dit is relevant omdat er toch een aantal hits op Stack Overflow te vinden zijn voor dit probleem. 2 React projecten die elk afzonderlijk een Redux store gebruiken kunnen als ze samengevoegd worden momenteel maar de volledige functionaliteit van 1 store gebruiken. Onderzoeksvraag:

- Op welke manier kunnen de stores van 2 react-redux apps onafhankelijk functioneren in 1 app?

2. State-of-the-art

Zoals eerder vermeld is er zeer weinig documentatie te vinden over dit probleem. Er zijn wel een aantal vragen in die richting op Stack Overflow gesteld. De antwoorden die op deze vragen gegeven zijn, bieden geen oplossing op lange termijn of bieden geen schaalbaarheid/uitbreidbaarheid. Er zal veel tijd gestoken worden in het verdiepen van de GitHub repository van Redux, geschreven door Abramov (2016). Er is wel documentatie te vinden over de Redux-store sharen in meerdere components binnen hetzelfde project. Hier staat een uitgewerkt voorbeeld van op Stack Overflow door b.g (2017). Het wordt echter onduidelijk wanneer we de Redux-store willen sharen tussen meerdere projecten.

3. Methodologie

Er zal geprobeerd worden een uitbreiding te schrijven op de bestaande Redux library. Zoals eerder vermeld houdt een re-

dux store eigenlijk de hele state-tree van de applicatie vast. De bedoeling van deze uitbreiding zal zijn om een draaiende app te hebben met 2 onafhankelijke stores van 2 projecten. Deze realisatie zal gebeuren door het ene project om te zetten naar een npm-package. Op deze manier kunnen geëxporteerde componenten makkelijk geïmporteerd worden in het andere project. Met de uitbreiding zal geprobeerd worden om een store toe te voegen aan het project, zonder dat de andere store daar enige invloed van ondervindt. Om dit te kunnen realiseren zal het react-redux patroon goed moeten bestudeerd worden. Daarbij kan ook naar Flux en MobX gekeken worden om eventuele analogieën door te trekken. In dit onderzoek zullen een aantal grafieken worden gemaakt met de gemiddelde rendertijd van verschillende oplossingen (met bv: nesting en reducers importeren). Anderszijds wordt er ook naar de lines of code gekeken. Een groot criteria hierbij is dan het dupliceren van code of het vervuilen van de applicatie door code toe te voegen die ze eigenlijk niet nodig zou moeten hebben.

4. Verwachte resultaten

Zoals eerder aangegeven worden er een aantal oplossingen verwacht. Enerzijds kan een nesting van de stores een mogelijke oplossing bieden, maar de vraag is dan of alle functionaliteit wordt behouden. Door het schrijven van de uitbreiding kan de rendertijd hoger zijn dan normaal. Een andere oplossing is door de reducers van het ene project samen te voegen met de reducers van het andere project, waarbij een hoger aantal lines of code verwacht wordt.

5. Verwachte conclusies

Een verwachte conclusie bij nesting is dat mogelijks niet alle functionaliteit behouden wordt. Een kopie nemen van de reducers van het ene project is geen efficiënte oplossing en wel om deze reden: stel dat er een update gepushed wordt naar de npm-package dan zou men alle reducers opnieuw moeten kopiëren naar het andere project. Tevens vervuilt dit ook de hoofdapplicatie.

Referenties

Abramov, D. (2016). Redux [Github]. Verkregen van https://github.com/reactjs/redux

b.g, S. (2017). How to share redux store in multiple components.