Documentation

De nieuws app is gebouwd in symfony 3.4,
ik heb gekozen voor Symfony 3.4 omdat ik hier de meeste ervaring mee heb.

De app bestaat hoofdzakelijk uit:

- twee controller classes `DefaultController` en `UserController`.
- Doctrine entities voor de feeds, items en gebruikers
- Een service `NewsService`.
- Vier twig views `index.html.twig`, `news.html.twig`, `settings.html.twig` en `login.html.twig`. Deze views extenden een base template.
- Een console command om nieuwe feeds toe te voegen.

De volledige source code staat in een private GIT repository op Github, indien hier interesse in is kan ik deze public maken of delen.

Online demo

De online demo is bereikbaar op:
http://news.keesgerbers.nl

er kan ingelogd worden met de demo gebruiker:
`email: demo@example.com`
`wachtwoord: Demo`

Ik heb geen restricties gemaakt op het bewerken van de gebruiker, dus alle gegevens zijn aan te passen via het gebruikers settings screen in de GUI, inclusief het wachtwoord.
Indien het niet meer werkt kan ik eenvoudig een backup terugzetten

gebruikersregistratie heb ik niet gemaakt omdat ik dit niet belangrijk achte voor een demo applicatie

Controllers

DefaultController

De `DefaultController` class verzorgt beide routes van de home pagina en de nieuws pagina. ondanks dat de doeleinden van de twee pagina's verschillend zijn heb ik ze toch in een controller ondergebracht omdat ze bijna dezelfde code bevatten en maar enkele regels groot zijn.

Specifiek bevat deze class de volgende methoden:

`indexAction`

Verzorgt de weergave van de homepage route, ook voorziet het de views van een lijst met alle feeds voor het menu en welkomst pagina.
Dit had ook via een twig extensie mogelijk geweest, maar heb dit in de controller gelaten omdat de controllers al heel ‘thin’ zijn en het voor weinig overhead en uiteindelijk weinig dubbele code oplevert.

‘newsAction`

Verzorgt de weergave van de news route en de koppeling naar de `NewsService`.
Pakt de volgende parameters:

- - `slug` → feed slug die gebruikt wordt om de juiste feed te laden (verplicht)
- - `categorie` → om binnen een feed alleen een bepaalde categorie te tonen (optioneel)

Aan de view wordt meegegeven:

- lijst met alle feeds,
- het opgevraagde Feed object welke verkregen is via de service `NewsService`
- categorie, deze zal `null` zijn indien er geen categorie is gezet.

UserController

In de `UserController` heb ik alle gebruikers gerelateerde zaken ondergebracht, het `settings` scherm en het login scherm.

De class bevat de volgende twee methoden:

`settingsAction`

Verzorgt de weergave van de `settings` route, maar ook het verwerken van de form gegevens.

Deze method haalt het het User object op van de huidig ingelogde gebruiker aan de hand van het email adres dat uniek moet zijn.

In eerste instantie wordt er via de Forms bundle een formulier gegenereerd en meegegeven en meegegeven aan de response.
Wanneer dit forumulier wordt ingevuld zal deze terug gestuurd worden naar dezelfde route en zal Forms automatisch herkennen dat er form data in de request zit en dit verwerken.
Terugkoppeling over de status hiervan wordt gegeven door een flash melding weer te zetten.

Op dit moment kan de gebruiker zijn naam, email en wachtwoord veranderen. Daarnaast als een extra voorkeur kan de achtergrondkleur gezet worden.

`loginAction`

Deze method verzorgt het weergeven inlog formulier en eventuele authenticatie fouten, de afhandeling van de authenticatie wordt intern door Symfony’s security systeem geregeld.

Doctrine entities

Ik heb ervoor gekozen om alle feeds en nieuws items in een mariaDB database te plaatsen.
Op deze manier kan ik zorgen voor caching.
De volgende entities zijn aanwezig:

- `Feed` (voor feed naam, url en overige meta data over de feed zelf)
- `Item` (voor alle nieuws items)
- `User` (voor alle gebruikers data en settings)

Ik heb ervoor gekozen om feeds en items apart op te slaan, met een mapping tussen deze twee entities. De mapping is ‘extra lazy’ waardoor bij het ophalen van het Feed object alleen item metadata wordt mee ingeladen, maar niet de actuele data.

Dit maakt het mogelijk dat ik de feeds los kan ophalen om weer te geven in het menu, zonder ook alle items in te laden, wat zorgt voor een snellere laadtijd van de pagina.

Door de mapping tussen de Feed en Item kan ik vanuit het Feed object wel gemakkelijk alle bijbehorende items bewerken en ophalen.
Hiervoor zijn een paar methods toegevoegd aan de Feed entity class.

Voor het inloggen van gebruikers heb ik een database provider geconfigureerd,
hiervoor is ook een user entity benodigd, met enkele verplichte methoden, Deze entity is uitgebreid met een extra veld om instellingen op te slaan.
Ik heb hier voor een vrij tekstveld gekozen waarin een JSON object wordt opgeslagen met alle instellingen, hierdoor kan ik gemakkelijk instellingen toevoegen in de toekomst zonder dat een schema wijziging nodig is in de database.

Service ‘NewsService’

Alle zaken gerelateerd aan het laden van nieuws items zijn ondergebracht in een service.

Hierdoor is alles gecentreerd in een service die verantwoordelijk is voor het teruggeven van een Feed object aan de `DefaultController::newsAction` method.

Deze service is de drijvende kracht achter de nieuws app en bestaat uit vier methoden:

public `getFeed()`	Geeft een Feed object terug aan de controller
public `addFeed()`	Voegt een nieuwe feed to aan de database
private `refreshFeed()`	Haalt de RSS feed op.
private `saveFeed()`	Plaatst de opgehaalde data in de Feed en Item objecten en slaat dit op in de database.

GetFeed($slug, $categorie=null)

Deze method wordt aangeroepen vanuit de controller en haalt het juiste Feed object op aan de hand van de meegegeven slug.

Daarna wordt bepaald aan de hand van de feed’s `lastUpdated` veld of deze ververst moet worden of direct teruggegeven moet worden.

Indien er een categorie meegegeven wordt zal er een aparte query gedaan worden om alleen de items te laden die aan de categorie voldoen, dit gebeurd simpel op basis van een zoekopdracht (LIKE) in het categorie veld van de items. Hierdoor is het in zijn basis meer een zoekfunctie en kan dit ook in de url aangepast worden naar een algemene zoekterm, ik heb hier geen restricties aan gehangen omdat het voor sommige feeds (zoals tweakers) juist een toevoeging kan zijn omdat deze maar een gecombineerde categorie tag meegeven (bijvoorbeeld ‘Nieuws: -Gaming / Games’ en `Nieuws: - PC / Games) en wanneer de categorie dan aangepast wordt naar ‘Gaming’ zullen items uit beide categorieen worden weergegeven. De GUI heeft hier geen optie voor ingebouwd omdat het niet de beoorgde bedoeling is van de functie.

AddFeed($url)

Deze method verzorgt het aanroepen van de RSS url, dit kan vooralsnog niet vanuit de GUI gedaan worden, maar heb hier een console commando voor gemaakt, zodat ik dit ook kon demonstreren. De method haalt de xml file op en zal deze doorgeven aan de method `saveFeed`

RefreshFeed(Feed $feed)

Deze methode doet exact hetzelfde als de `AddFeed` method, echter wordt de rss url hier uit het meegegeven Feed object gehaald, en zal ook dit Feed object doorgegeven worden aan de `saveFeed` methode

SaveFeed(Feed $feed = null, $rss)

Deze methode heeft als taak het omzetten en splitsen van het `SimpleXml` object de doctrine entities. Deze method zorgt ook ervoor dat alle data gecast wordt naar het juiste type zodat dit later makkelijk te bewerken en tonen is.
Deze methode neemt optioneel een Feed object en de opgehaalde xml uit de rss feed.
De reden dat ik $feed optioneel heb gemaakt is zodat de `AddFeed` en `RefreshFeed` beide gebruik kunnen maken van deze methode, en binnen `AddFeed` is er nog geen Feed object bekend.

Views – Templates

De app templates zijn volledig gebouwd met Bootstrap 3, hier ben ik het bekendst mee en heb nog weinig ervaring nog met Bootstrap 4 waar toch de nodige wijzigingen inzitten.

Alle sub templates extenden `base.html.twig` waardoor elk scherm dezelfde opmaak heeft, en dit heel gemakkelijk aan te passen is.

Ik heb geleerd dat er maar een keer ge-extend kan worden, daarom zit er meer code in het basis template dan nodig zou zijn.
Tegenwoordig kan dit wel begreep ik uit de documentatie en ideaal zou hier nog een stap tussen zitten waarin logica wordt toegepast, zoals user styling en het bouwen van het feed menu, echter toen had ik de bestaande basis template al klaar en heb ik besloten deze zo te laten.

De volgende twig templates zijn beschikbaar:

- `base.html.twig`
  - `index.html.twig`
  - `news.html.twig`
  - `settings.html.twig`
  - `login.html.twig`

base.html.twig
Verzorgt het weergeven van de header, menu, flash notificaties en styling, elke view extend dit basis template, afhankelijk van de beschikbaarheid van sommige variablen worden UI componenten getoond.

index.html.twig
Deze view is verantwoordelijk voor het weergeven van de beschikbare feeds en de welkomst-tekst

news.html.twig
Het weergeven van de nieuws items in aparte panels

settings.html.twig
Een simpel bootstrap panel met de initialisatie code voor het Symfony Forms formulier weer te geven.

login.html.twig
Een simpel bootstrap panel met het login formulier

Console command - AddFeedCommand

Om nieuwe feeds toe te voegen moet het console commando worden gebruikt
`bin/console news:feed:add`

Het enige wat dit commando doet is het aanroepen van de `AddFeed` methode uit de `NewsService` met de opgegeven rss url als parameter.

De `AddFeed` methode is echter ook heel gemakkelijk in te voegen bijvoorbeeld in het gebruikers settings scherm waardoor het mogelijk zou zijn voor de gebruiker om nieuwe feeds toe te voegen.

Maar omdat ik feeds nog niet aan gebruikers heb gekoppeld en iedereen dus alle feeds kan zien en ik de functionaliteit toch nodig had om zelf makkelijk feeds toe te voegen heb ik hier een console commando van gemaakt. (en ook om dit te demonstreren)