1 Conclusie

1.1 Adobe Experience Manager

AEM doet wat het moet doen, het biedt een schaalbare oplossing voor organisaties die op zoek zijn naar een CMS applicatie. De UI van de editor zorgt ervoor dat marketeers, zonder al te veel moeite, aan de slag kunnen met het maken, wijzigen en publiceren van pagina's. Doordat de leercurve zo laag is, zijn werknemers sneller ingewerkt waardoor er minder geÏnvesteerd moet worden in opleidingen.

De leercurve voor ontwikkelaars ligt des te hoger, dit komt omdat AEM bestaat uit een combinatie van verschillende frameworks. De kans dat een ontwikkelaar reeds met al deze frameworks te maken heeft gehad (indien deze nog niet met AEM gewerkt heeft) is nihil. Voornamelijk OSGi is een leerproces waar een redelijke hoeveelheid tijd aan besteed moet worden alvorens men hiermee aan de slag kan.

Ook al is de editor intuïtief in de omgang met componenten, het maken van deze componenten is iets ingewikkelder. Voornamelijk via een Maven project is het een kwestie van vallen en opstaan. Tijdens het leerproces was één van de uitdagingen om bruikbare documentatie hierover te vinden. Het maken van componenten via de author is beter gedocumenteerd en kan gebruikt worden om de gedachtegang beter te begrijpen. Toch is het even schrikken wanneer je je eerste (template) project opent en alle modules en packages ziet verschijnen. Zonder begeleiding met kennis hierover is het begrijpen van de onderdelen een moeilijke/frustrerende taak.

AEM is de CMS applicatie voor bedrijven die geen schrik hebben om te investeren in de opzet ervan, het aannemen van ervaren developers is een must. De vrijheid dat AEM biedt met betrekking tot het beheren van een grote hoeveelheid content en de out-of-the-box ondersteuning voor meerdere sites maakt het voor gewenst bij de internationale commerciële bedrijven.

1.2 Omgaan met dynamische data

Het is duidelijk dat AEM niet gelimiteerd is in het omgaan met deze data maar dat een duidelijke analyse een noodzaak is om een succesvolle applicatie te bouwen. Tijdens het ontwerpen van de designs is het cruciaal om deze te ontleden in logische segmenten. Deze segmenten moeten vervolgens geanalyseerd worden om te kunnen beslissing welke technieken waar toegepast kunnen worden. Een segment kan kan gebruik maken van een combinatie van technieken om tot een optimale oplossing te komen. Tijdens het analyseren moet men volgende vragen in achting nemen, deze helpen bij het scheppen van een duidelijk beeld.

Wordt het segment veelvuldig gebruikt? Indien het antwoord ja is, kan men ervoor kiezen dit deel te isoleren in een aparte component en op alle pagina's te injecteren via SSI. Dit zorgt dat alle pagina's dezelfde versie tonen ongeacht de tijd van caching.

Bevat het segment data specifiek voor deze pagina en die niet wijzigt door een actie van de gebruiker? Dan kan men kiezen uit twee technieken: het genereren van een pagina of het injecteren in een generieke pagina. Om tussen deze twee te kiezen moet men kijken naar twee variabelen, de dynamiek van de data en het aantal items in bijhorende tabel. De eerste variabele vertelt ons hoe vaak we dezelfde pagina opnieuw zouden moeten genereren. Hoe groter de frequentie, des te minder we geneigd zijn om de eerste techniek te kiezen. De tweede vertelt ons hoeveel pagina's we zouden moeten genereren bij een initiële load, ook hier geldt dezelfde regel als de vorig variabele.

Wijzigt er data via interactie van de gebruiker op de pagina? In dit geval moeten we er rekening mee houden dat we de componenten, waaruit het segment bestaat, voorzien van Javascript. Het is perfect mogelijk om een component die met SSI geïnjecteerd wordt te voorzien van Javascript, om zo het beste van beiden te combineren.

Indien deze vragen bij elk design gesteld worden, weet een ontwikkelaar perfect waar hij of zij rekening mee moet houden. Dit zorgt voor een kortere ontwikkelingstijd en performante componenten, wat leidt tot een performante webapplicatie.