Content Packaging is het proces om bij elkaar behorende content samen te pakken in een zip file. Deze wordt voorzien van een beschrijvend bestand (het manifest), zodat de ontvangende partij kan zien wat het krijgt. Vooral bij educatieve content is dit een veel gebruikt fenomeen.
Dit artikel geeft een overzicht van de praktijkervaringen rondom het realiseren van een Content Packaging applicatie met het Java open source framework Cocoon. Dit blijkt een uitermate geschikt platform hiervoor te zijn.
Dit artikel blijft op overzichtsniveau en gaat niet de technische diepte in. Kennis van Cocoon is niet noodzakelijk.

1. CONTENT PACKAGING

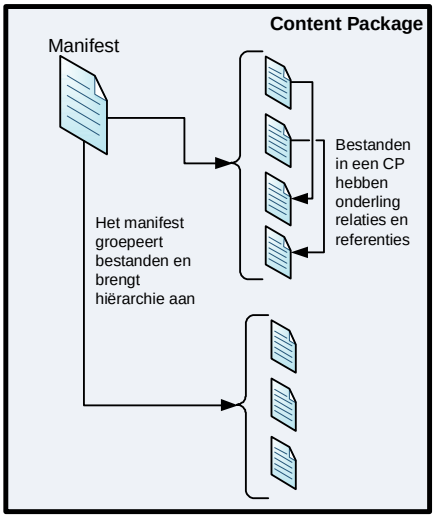
Het is een algemeen probleem: Hoe verplaats ik grote hoeveelheden aan elkaar gerelateerde bestanden van het ene systeem naar het andere? Hoe hou ik alles bij elkaar en zorg ik ervoor dat er onderweg niets wegraakt? En dan ook nog zodanig dat de ontvangende partij er weer chocola (of iets anders) van kan maken.
Een veelgebruikte oplossing hiervoor is al deze bestanden samenpakken in één overkoepelend zip bestand. De Java wereld kent bijvoorbeeld zijn `.jar` en `.war` files, tekstverwerkers de `.odt` en `.docx` bestandsformaten. Allemaal gestoeld op hetzelfde idee.

In de educatieve wereld heet de hiervoor bedachte oplossing “Content Packaging”: Stop al die gerelateerde bestanden bij elkaar in een zip bestand en voeg er een eenvoudig vindbare pakbon bij. In Content Packaging terminologie heet zo’n pakbon een manifest.
Deze pakbon is meer dan alleen maar een directory- of bestandsoverzicht: Het beschrijft o.a. welke bestanden uit de zip file bij elkaar horen (samen een eenheid vormen) en hoe de onderlinge hiërarchische structuur is. Belangrijk is ook de aanwezigheid van metadata in de pakbon, zodat direct duidelijk wordt wat voor pakketje dit eigenlijk is. Uiteraard is het manifest een XML bestand.
Deze ijzersterke combinatie van zipfile plus manifest komt uit de stal van de internationale educatieve standaardenorganisatie IMS (`www.imsglobal.org`). Bovenop de basale Content Packaging standaard zijn weer andere standaarden gemaakt zoals IMS Common Cartridge en SCORM (`www.adlnet.gov`). Voor Nederland is er een afgeleide vastgelegd bij Edustandaard (`www.edustandaard.nl/afspraken/002`). Overigens is de Edustandaard beschrijving van Content Packaging erg duidelijk en een verademing voor als je de taaie originele documentatie gewend bent. Een absolute aanrader voor iedereen die hierover meer informatie nodig heeft.

Content Packaging wordt in de educatieve wereld gebruikt om content van het ene systeem naar het ander te transporteren. Een beetje educatieve content bestaat al gauw uit een heleboel bestanden van divers pluimage:

- Teksten voor bijvoorbeeld instructies;
- Vragen in een (meestal XML) formaat dat het mogelijk maakt ze geautomatiseerd te presenteren en na te kijken. Hiervoor heeft IMS een andere veel gebruikte standaard: QTI;
- De plaatjes, geluiden en filmpjes (assets) die gebruikt worden;
- Eventuele applicatie software (players) die dit alles kunnen presenteren;

Al die bestanden hebben weer relaties met elkaar: Vragen verwijzen naar plaatjes, een filmpje heeft een afspeler nodig, onderdelen hebben een volgorde, etc. Anders gezegd: De bestanden hebben *onderling* relaties maar er zijn ook *overkoepelende* relaties en ordeningen.



Om dit alles op een correcte en begrijpelijke manier samen te bundelen valt nog niet mee. Bronbestanden zijn niet altijd in het gewenste formaat, gerefereerde bestanden moeten er bij worden gezocht, referenties aangepast, de structuur moet worden vastgesteld, een manifest opgebouwd, etc. Een flinke klus voor een pittig stukje software. Ervaring leert dat dit in traditionele programmeeromgevingen (Java, .NET) vaak lastiger te realiseren is dan het op het eerste gezicht lijkt.

2. COCOON

Cocoon is een open source Java framework waarmee je op een relatief eenvoudige en intuïtieve wijze XML kunt verwerken (`cocoon.apache.org`). De basis van Cocoon zijn "pipelines": XML transformatiestraten opgebouwd uit achter elkaar geplaatste losse transformaties, meestal XSLT. Deze pipelines kun je naar hartelust combineren tot complexe samenstellingen. Doordat Cocoon slim is opgebouwd "stroomt" de XML zonder veel overhead (voor de intimi: als SAX events) door de pipelines en wordt een zeer behoorlijke performance gehaald.

Cocoon is oorspronkelijk opgezet voor het maken van websites. IMHO (en nu schop ik de Cocoon puristen tegen de schenen) is het daarvoor echter minder geslaagd. Voor normale websites zijn PHP of ASP.NET veel geschikter. Maar wat de makers wel (onbedoeld?) gecreëerd hebben is een fantastische omgeving voor het manipuleren van grote hoeveelheden XML bestanden, ook als het uiteindelijke resultaat geen webpagina's zijn.

Cocoon maakt het mogelijk de XML verwerking op een hoger abstractieniveau te beschrijven. Waar traditionelere programmeertalen lijden onder veel low-level geneuzel als bestanden openen en sluiten, handles, verwerking van tussenresultaten, etc., hoef je in Cocoon "slechts" je transformatiestraat te beschrijven. Voor Content Packaging precies wat je nodig hebt.

3. PACKAGING PRINCIPES

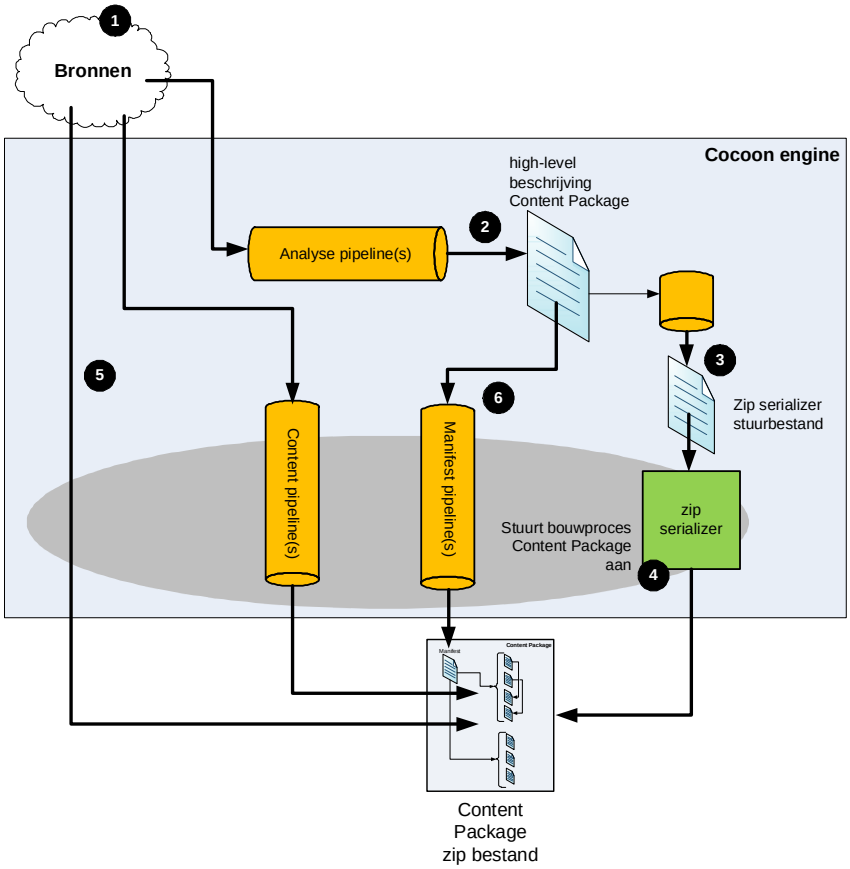
Wat moet een Content Packaging applicatie nu zoal doen om tot een resultaat te komen?

- Als eerste moet het de bron(nen) natuurlijk kunnen uitlezen. De content voor het package moet tenslotte ergens vandaan komen: Een content management systeem, speciale database, gewoon files op disk, etc. De meest voorkomende typen bronnen zijn XML bestanden en assets.
- De volgende stap om tot een Content Package te komen is het analyseren van de bronnen. Wat hoort bij elkaar, welke assets moeten meegenomen worden, welke XML conversies of "rits-en-splits" operaties zijn nog nodig, wat wordt de metadata?
  Voor deze analyse zijn er meestal stuurbestanden aanwezig ("neem dit mee", "dit is de eerste pagina", etc.). Ook wordt gekeken in de content zelf (aanwezige relaties met andere bestanden, metadatering, etc.).
- Met behulp van deze analyse kan het systeem beginnen het Content Package samen te stellen. Voor ieder bestand in het uiteindelijke Content Package is nu bekend hoe het gemaakt moet worden (bron, conversies) en waar het terecht moet komen (interne directory structuur, uiteindelijke bestandsnaam).
- Ergens in het proces moet ook nog een manifest worden gemaakt. Omdat de analyse het Content Package al volledig specificeert kan deze hiervoor uitstekend als bron dienen.

- Het totaal moet worden afgeleverd als zip file, op de juiste plek en met de juiste naam.

4. DE INZET VAN COCOON

Cocoon met zijn pipeline architectuur is uitermate geschikt voor bovenstaand karwijtje. Hoe dit in zijn werk gaat ziet u in het volgende schema:



1. Ergens (laten we dat niet verder uitdiepen) zijn er bronbestanden. Cocoon moet deze natuurlijk kunnen uitlezen. Het heeft daarvoor allerlei mogelijkheden aan boord, variërend van bestanden op disk tot database en webservice koppelingen.
2. Het eerste wat er vervolgens moet gebeuren is het creëren van een analyse of high-level beschrijving van het gewenste Content Package. De pipeline (of meestal serie van pipelines) hiervoor levert een XML bestand af waarin alle noodzakelijke informatie staat. Dit is een intern bestand en de opbouw ervan kan volledig door de programmeur bepaald worden. Door de opzet van Cocoon hoeft het ook niet ergens opgeslagen of bewaard te worden, het blijft intern. Voor debugging doeleinden kan het uiteraard wel bekeken worden.

Cocoon heeft een zogenaamde “zip serializer” component. Een zip serializer stelt een zip file samen op basis van een, XML, beschrijving. Het aardige van de zip serializer is nu dat de bron voor de bestanden in de zip file weer Cocoon pipelines kunnen zijn. Dit stelt je in staat om bij het vullen van het Content Package noodzakelijke conversies uit te voeren.

3. Vanuit de high-level beschrijving wordt een input file voor de zip-serializer gemaakt (met behulp van weer een pipeline). Deze beschrijft exact de inhoud (bestanden en directory structuur) van de zip file en waar deze bestanden vandaan moeten komen.
4. De zip serializer gaat aan het werk en bouwt de zip file op.
5. Bestanden die eerst nog geconverteerd, samengesteld, gesplitst of iets dergelijks moeten worden, worden via de aanroep van Cocoon pipelines gerealiseerd. Bestanden waarvoor dat niet geldt worden rechtstreeks uit de bron gelezen.

6. Ook voor het manifest is er een speciale pipeline. Deze gebruikt de high-level beschrijving als bron. Je zult er dus voor moeten zorgen dat alle noodzakelijke informatie hiervoor (o.a. de metadata) in stap 2 wordt geproduceerd en in de high-level beschrijving terecht komt.

5. PRAKTIJKERVARINGEN

Dit artikel is geschreven op basis van ervaringen met het daadwerkelijk bouwen van een Cocoon gebaseerd Content Packaging systeem. Dit systeem leest uit meerdere bronnen en kan momenteel zo'n vijf verschillende smaken Content Packages maken, waaronder een SCORM variant. Het bevat enkele tientallen pipelines en ruim vijfhonderd XSLT (meest V2.0) transformaties. De omvang van de resulterende Content Packages varieert van enkele tientallen Mb's tot, in een enkel geval, meer dan 1 Gb. Het is nu een jaar in productie en wordt frequent gebruikt.
Een aantal ervaringen rondom bouw en gebruik van dit systeem:

- Voor de ontwikkeling van een dergelijk systeem maakt Cocoon een wereld van verschil. Dat wil niet zeggen dat het daarmee eenvoudig wordt, het harde werk, analyse en uitvoering, moet nog steeds gedaan worden. Je kunt echter het probleem op het juiste niveau en met de juiste hulpmiddelen te lijf gaan, zonder afgeleid te worden door voor het probleem niet relevante zaken als bijvoorbeeld tijdelijke bestanden. Een XSL ergens invoegen is triviaal.
- De Cocoon leercurve is echter behoorlijk steil. Zoals veel open source technologieën lijdt het onder gebrekkige, onduidelijke en achterlopende documentatie. De boeken die erover gaan zijn verouderd.
- De verwerkingscapaciteit is ruim voldoende. De uiteindelijke applicatie bleek in staat om Content Packages van meer dan één GB moeiteloos te kunnen produceren. Ook XML bestanden van tientallen Mb's werden zonder problemen verwerkt.
- Wat niet wil zeggen dat het altijd even snel ging: Het maken van grote Content Packages kan makkelijk oplopen tot een uur of langer. Gelukkig hoeft Content Packaging zelden real-time plaats te vinden. Het is weliswaar vervelend steeds zo lang te moeten wachten maar het wordt niet als een groot probleem ervaren.
- Met bepaalde trucs (caching, aanpassen tussenresultaten) kan de performance nog flink worden verhoogd (in een uitzonderingsgeval is zelfs een factor vijf gehaald!). Dit verandert echter niets aan de principiële werking van de applicatie. Voor Cocoon applicaties geldt dan ook zeer beslist het: "First make it work and then make it fast".
- Qua hardware is Cocoon vooral geheugen en processor intensief. Voor productiedoeleinden is een grote server zeker aan te raden, maar ontwikkeling kan prima op een kleinere machine. Ik gebruik hiervoor een normale zakelijke laptop. Een dual core processor is dan wel handig: Cocoon pakt netjes één core zodat je ook tijdens intensieve verwerkingsslagen gewoon door kunt werken.
- Ondersteuning en kennis is in Nederland is slechts mondjesmaat verkrijgbaar. Er zijn maar enkele bedrijven en freelancers die voldoende kennis in huis hebben. Dit kan een gevaar zijn voor de continuïteit. De oplossing voor het hier beschreven systeem is gezocht in een samenwerkingsverband tussen verschillende partijen die een gezamenlijke SLA aanbieden.

6. CONCLUSIES

Dit artikel is voortgekomen uit het enthousiasme dat ontstond toen bleek dat eindelijk het juiste tool voor het lastige Content Packaging probleem gevonden was. Het relatieve gemak waarmee zo'n applicatie gebouwd is en onderhouden kan worden bewijst dit.
Wat voor Content Packaging geldt, geldt natuurlijk ook voor alle andere toepassingen waarin complexe zip bestanden moeten worden opgeleverd.
Wat niet wil zeggen dat er geen nadelen aan kleven: Met name de relatieve onbekendheid van het Cocoon platform baart zorgen, maar door middel van samenwerkingsverbanden is hier zeker een mouw aan te passen. Cocoon als platform is volop in ontwikkeling en in ieder geval nog lang geen end of life.
Kortom, wie een Content Packaging probleem heeft zou er goed aan doen eens naar Cocoon als ontwikkelplatform te kijken.