Handelswetenschappen en Bedrijfskunde
Toegepaste Informatica
Programmatie - Analyse

Publieke website als communicatiemiddel voor het Europese netwerk ACTINET in eZ publish

CAMPUS
Geel

Gwen Vansonhoven

Academiejaar 2004-2005

De houder van dit diploma is gerechtigd tot het voeren van de titel van Bachelor

Woord vooraf

Na drie maanden kan ik met een “gerust” hart zeggen dat het zeer intensieve maanden waren. In het bedrijfsleven kan je het kalme schoolleven echt wel vaarwel zeggen. Ik ben met volle moed begonnen aan de stage en kan aan het einde van de meet zeggen dat het de moeite waard was.

Deze stage ging niet enkel om het ontdekken van het bedrijfsleven, maar ook van mezelf. Ik moest al mijn sociale vaardigheden naar boven halen om goed in team te kunnen functioneren. Er kwam dan nog de werkdruk, de frustraties en het eindwerk bij waardoor het op sommige momenten echt hectisch werd. De stage is zeer leerrijk geweest. Ik leerde er technische dingen bij die veel verder reiken dan de leerstof op school. Gecombineerd met een toffe werksfeer en de steun van vele mensen, was het allemaal beslist de moeite waard. Een woordje van dank is hier zeker op zijn plaats.

Allereerst wil ik Marie-Laure Ruyssen, mijn externe begeleidster, bedanken voor de toffe en leerrijke stageopdracht. Daarnaast zijn er zeker de informatici van het Knowledge Management: Hans Melis, Tom Couwberghs en Kristof Coomans, die mij met raad en daad hebben bijgestaan in het volbrengen van mijn stageopdracht. Ook wil ik Arnout Ulenaers bedanken voor de vlotte samenwerking. De collega’s op de werkvloer, Mario Bens, Kris Pennemans en Wendy Machiels zorgden voor een aangename sfeer, waarvoor mijn dank. Ook met mijn medestagiair, Wim Konings, was het aangenaam samenwerken.

Buiten mijn stageplaats op het SCK•CEN wil ik ook nog een aantal mensen bedanken, in het bijzonder Kristine Mangelschots, die mij raad gaf in verband met de stage en voor mijn eindwerk het naleeswerk verrichtte.

Tenslotte wil ik graag mijn familie en vooral mijn vriendin Leen bedanken voor de steun en het begrip tijdens mijn stage.

Bedankt allemaal!

Gwen Vansonhoven

Samenvatting

Mijn stageperiode begon met het aanleren van het open source Content Management System eZ publish. Toen ik een zekere basis had aangelegd over het systeem, ben ik gestart met mijn stageopdracht. Deze was het maken van een publieke website voor het Europese netwerk ACTINET.

Allereerst ben ik begonnen met de basislay-out van de website. De moeilijkheid hiervan was een goede lay-out te vinden die zeer duidelijk was en tegelijk aangenaam was om naar te kijken. Dit verliep niet eenvoudig; tijdens de stageperiode zijn er heel wat aanpassingen gebeurd. Na het bouwen van de lay-out, werd het tijd om aan de structuur van de website vorm te geven.

Vervolgens ben ik begonnen aan het opbouwen van de klassen. Deze zorgen ervoor dat er nieuwsitems, evenementen, partners, ... kunnen gemaakt worden die alle nodige informatie tonen op het scherm.

Het maken van templates die voor de opmaak zorgden van de website, nam het grootste deel van mijn tijd in beslag. Zo moest elke klasse een opmaak krijgen zodat alle informatie op een mooie en gestructureerde manier op het scherm wordt getoond. Soms moest er voor deze templates een templateoperator geschreven worden, omdat deze zich niet in het systeem bevond. Ook heb ik een editor-interface gemaakt voor de websiteverantwoordelijke, zodat hij op een eenvoudige manier inhoud kan toevoegen, bewerken of verwijderen.

Ook heb ik ervoor gezorgd dat het voor de gebruiker mogelijk is zich te registreren op de website. Hiervoor heb ik een module geschreven die de registratie controleert, opvangt en indien nodig weigert, en aan de hand van deze keuze een mail verstuurt.

Inhoudstafel

Illustraties

Alfabetische lijst van gebruikte afkortingen en symbolen

API	Application Programming Interface
BR1	Belgian Reactor 1
BR2	Belgian Reactor 2
BR3	Belgian Reactor 3
CSS	Cascading StyleSheets
DBX	Database aBstraction eXtension
GPL	General Public License
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IBL	InterBibliothecair Leenverkeer
IMAP	Internet Message Access Protocol
ISO	International Organization for Standardization
KMKC	Knowledge Management Knowledge Centre
LDAP	Lightweight Directory Access Protocol
MySQL	My Structured Query Language
NEMO	Network of Excellence on Micro-Optics
NF-PRO	Near Field Key Processes for Nuclear Waste Disposal
NNTP	Network News Transfer Protocol
ODBC	Open DataBase Connectivity
PDF	Portable Document Format
PHP	PHP: Hypertext Preprocessor
POP3	Post Office Protocol
PWR	Pressurised Water Reactor
SCK•CEN	Studiecentrum voor Kernenergie • Centre d'étude de l'Energie Nucléaire
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
STK	Studiecentrum voor de Toepassingen van de Kernenergie
SVN	Subversion
URL	Uniform Resource Locator
VENUS	Vulcain Experimental Nuclear Study
Vito	Vlaamse Instelling voor Technologisch Onderzoek
W3C	World Wide Web Consortium
XHTML	eXtensible Hypertext Markup Language
XML	eXtended Markup Language
XSLT	eXtensible Stylesheet Language Transformations

Inleiding

In dit eindwerk krijgt u een kort overzicht aangeboden over mijn stage op het SCK•CEN. Als opdracht kreeg ik het ontwikkelen van een publieke website voor het Europese netwerk ACTINET.

De bedoeling van deze website is informatie te verschaffen aan het grote publiek, de wetenschappers en de Europese Gemeenschap.

In het eerste hoofdstuk ga ik een korte voorstelling geven van het SCK•CEN en de afdeling Knowledge Management Knowledge Centre (KMKC). De software die ik tijdens mijn stage heb gebruikt, komt aan bod in hoofdstuk twee. Hier wordt voornamelijk eZ publish uit de doeken gedaan. In het derde hoofdstuk leg ik u kort alle standaarden uit die ik gebruikt heb bij het maken van mijn stageopdracht. Daarna wordt er in hoofdstuk vier de stageopdracht toegelicht. Om u toch een beetje voor te bereiden op de unieke templatecode van eZ publish, geef ik een korte introductie in hoofdstuk vijf.

Dan begint het meer technische deel van dit eindwerk. In hoofdstuk zes worden alle klassen uitgelegd die ik heb gebruikt bij het bouwen van de website. Daarop volgen de extensies in hoofdstuk zeven die nodig waren bij het maken en verzenden van mails en bij het aanmaken van klassen, templateoperatoren en nieuwe gebruikers. Als laatste wordt er in hoofdstuk acht de belangrijkere stukken code, die voor de opmaak van de website zorgen, benaderd en uitgelegd.

1 SCK•CEN

Het Studiecentrum voor Kernenergie, SCK•CEN, is een instelling die belangrijke bijdragen levert tot de nucleaire veiligheid, stralingsbescherming en veilige verwerking/opslag van radioactief afval. Het SCK•CEN biedt ook bescherming tegen verspreiding van nucleair materiaal voor niet-vreedzame doeleinden. Het SCK•CEN is een toonbeeld van de Belgische kracht op het vlak van kernenergie.

1.1 Geschiedenis

Het begon allemaal met Wilhelm Conrad Röntgen aan het einde van de 19de eeuw. De uitvinding van de X-stralen, ook Röntgenstralen genoemd, bracht een ware technologische revolutie in de geneeskunde. Kort daarop merkte Marie Curie dat sommige grondstoffen een hoge energie vrijgaven, de radioactieve energie. Deze voorvallen leidden tot ons huidig atoommodel.

Zodra in 1932 de neutron werd ontdekt, kwam de kernfysica en het kernonderzoek pas echt op gang. De Duitsers Otto Hahn en Fritz Strassman brachten voor het eerst een kernsplijting tot stand.

In de jaren ‘30 bezat België uraniummijnen in zijn Kongolese kolonie. Niemand vermoedde dat dit een grote rol zou spelen in de ontwikkeling van de nucleaire sector in België. Het uranium werd in de eerste plaats gebruikt voor de aanmaak van radium voor medische toepassingen. In 1942 trachtten de Amerikanen België te overhalen om de Kongolese uraniumreserves te verkopen. Op 26 september 1944 werd het “Memorandum of Understanding” getekend en leverde België 1560 ton uraniumerts aan de geallieerden. De USA en het Verenigd Koninkrijk kregen voor 10 jaar lang het alleenrecht op de uraniumvoorraden en België kreeg toegang tot de nucleaire know-how in commerciële en niet-militaire toepassingen. De levering van al dat uraniumerts leidde wel tot desastreuze gevolgen. Zo zijn er de voorbeelden van Hiroshima op 6 augustus 1945 en Nagasaki op 9 augustus 1945.

Maar er kwam een keerzijde. In 1951 gaf Pierre Ryckmans aan een groep wetenschappers de opdracht een nieuw organisme voor de studie van de toepassingen van de kernenergie op te richten. De oprichters behoorden tot diverse kringen van de wetenschappelijke wereld, de universiteiten, de overheid en de industrie. Zo kwam er de oprichting van een vzw die “Studiecentrum voor de Toepassingen van de Kernenergie”, kortweg STK, genoemd zou worden. Het onderzoek zou de nucleaire energie aanwerven als de energiebron van de toekomst. De bouw van het STK werd gedaan in Mol omdat daar het terrein voldeed aan alle eisen. Het was er voldoende groot, het lag op voldoende afstand van de bewoonde gebieden en het had een stabiel klimaat.

Tussen 1954 en 1962 groeiden de terreinen van het Centrum uit tot één van de grootste naoorlogse bouwwerven in België. Hierdoor begonnen de inwoners van Mol zelfs te praten over “den Atoom” of “het Atoomdorp”. Deze termen worden nu nog gebruikt.

Op vrijdag 11 mei 1956 was de BR1, Belgian Reactor 1, voor het eerst kritisch. Hij werkte met grafiet als moderator, natuurlijk uranium als splijtstof en lucht als koelmiddel. De BR1 werd voor 40% gebruikt voor de productie van isotopen, 30% voor onderzoeken door het SCK•CEN zelf en 30% voor rekening van externe klanten. De reactor is de dag van vandaag nog steeds in gebruik.

Op 6 juli 1961 werd BR2, Belgian Reactor 2, kritisch. Deze reactor was een stuk groter dan BR1. Hij werd tot eind 1962 getest en werd dan operationeel. Deze reactor heeft een hoge neutronenflux om het gedrag van allerlei materialen onder hoge bestraling na te gaan. Hij werkt op hoog verrijkt uranium en wordt gemodereerd en gekoeld door water. De BR2 is nog steeds de meest performante onderzoeksreactor van West-Europa.

Acht jaar voor de indienststelling van BR3 , Belgian Reactor 3, overwoog België de bouw van een elektriciteitscentrale. Deze elektriciteitscentrale zou uiteindelijk de BR3 worden. BR3 is een Pressurised Water Reactor (PWR). Deze is gekoeld en gemodereerd door water onder druk. De kriticiteit was op 19 augustus 1962 en de aansluiting op het Belgische elektriciteitsnet vond plaats op 25 oktober van hetzelfde jaar. De BR3 zou dienen als demonstratie-eenheid voor de bouw en uitbating van een industriële elektriciteitscentrale en diende als proefstation voor prototype splijtstoffen. Deze reactor is een voorbeeld van een "kruisbestuiving" tussen onderzoek en industrie geworden. Uiteindelijk werd de BR3 op 30 juni 1970 definitief stopgezet omdat men niet langer kon voldoen aan de uitbatingsvergunning. Volgens de planning zal BR3 volledig ontmanteld zijn in 2007.

VENUS is de enige reactor die een naam heeft die niet met "BR" begint. Dit was een project dat kaderde in het Vulcainproject. In het Vulcainproject onderzocht men hoe de vloeistof van de reactor afgewisseld kon worden om het verbruik van splijtstoffen te compenseren. Op 30 april 1964 werd VENUS voor het eerst kritisch. De reactor voerde van 1964 tot 1966 allerlei experimenten uit. De omgebouwde vulcainkern werd na 1966 in BR3 geïnstalleerd. Deze installatie eindigde in 1968. VENUS zelf werd na 1966 omgebouwd om neutronenstudies uit te voeren.

Op 16 oktober 1991 werd het SCK•CEN gesplitst. De niet-nucleaire activiteiten werden overgeheveld naar Vito. Vito zou zich concentreren op onderzoek rond het leefmilieu, energie, grondstoffen en materialen.

1.2 Doelstellingen SCK•CEN

Een van de belangrijkste doestellingen van het SCK•CEN is de bijdrage tot nucleaire veiligheid. Dit onderzoek bepaalt wat een veilige levensduur is van een kerncentrale en wat de optimale versplijtingsgraad van de splijtstoffen is. Ook belangrijk hierin is de vermindering van de dosisbelasting voor het personeel en het verminderen van afvalproductie.

Een volgende doelstelling is de stralingsbescherming. Dit houdt vooral de verbetering van de nucleaire noodplannen en de sanering van radioactief besmette sites in. Er zijn kleine onderzoeksprojecten zoals het onderzoek naar het behoud van de competentie op het vlak van radiobiologie, nucleaire metingen, verspreiding van radioactief materiaal in het leefmilieu, radiologische evaluatie en natuurlijke radioactiviteit.

Verwerking en opslag van radioactief afval is een doelstelling die niet mag vergeten worden. De studies hiervoor gebeuren zowel binnen als buiten het ondergrondse laboratorium dat zich in de Boomse kleilaag bevindt.

1.3 Knowledge Management Knowledge Centre

Sinds de oprichting is het SCK•CEN een zeer kennisgericht bedrijf. Omdat er op het SCK•CEN zoveel variëteit aan kennis is, is er een Knowledge Management nodig om al deze informatie op goede manier te bewaren en te centraliseren.

Door het Knowledge Management werd een portal aangemaakt om informatie op een goede manier beschikbaar te stellen aan al de werknemers. Hiervoor is er de bibliotheek portal ontwikkeld. Hier kunnen werknemers van het SCK•CEN informatie op aanvragen die dan verwerkt worden door het personeel van het Knowledge Centre. Deze portal krijgt met de tijd een grotere rol binnen het SCK•CEN en vergroot zo zijn rol tot een knowledge management portal.

Er wordt ook gewerkt aan andere projecten zoals:

- Network of Excellence on Micro-Optics (NEMO) (zie www.micro-optics.org)
- Network of Excellence on Actinides (ACTINET)
- Near Field Key Processes for Nuclear Waste Disposal (NF-PRO)

Op het SCK•CEN is er ook nog het Knowledge Centre. Deze was voorheen de bibliotheek die gesticht was in 1953 om alle nucleaire literatuur beschikbaar te stellen voor de werknemers. Maar voor het gewone publiek zijn de boeken niet toegankelijk. Vandaag is het concept van bibliotheek geplaatst in een bredere context. Omdat deze context informatie- en kennisbeheer omvat, heeft de bibliotheek de naam ‘Knowledge Centre’ gekregen in 2002.

Het personeel van het Knowledge Centre behandelt vragen naar informatie. Ze zetten boeken in de rekken, fotokopiëren artikels en printen microfiches af. Het Knowledge Centre kan natuurlijk niet alle informatie bevatten. Als er een boek of artikel wordt opgevraagd dat niet aanwezig is, kan het personeel deze aanvragen bij externe bibliotheken. Dit gebeurt met behulp van een IBL-account. IBL staat voor InterBibliothecair Leenverkeer. Deze laat een bibliotheek toe . Er is ook de mogelijkheid om een abonnement aan te vragen op een bepaald tijdschrift of boek. De betaling hiervan zal geregeld worden door het personeel van het Knowledge Centre. Maar er moet niet noodzakelijk een abonnement worden genomen op tijdschriften. Het Knowledge Centre bevat een leeszaal waar veel tijdschriften, kranten en andere informatiebronnen up-to-date worden gehouden. Deze leeszaal is vrij toegankelijk voor het personeel. Alle informatie die in het Knowledge Centre binnenkomt wordt ook gecatalogeerd door het personeel in een on-line cataloog.

2 Software

In dit hoofdstuk bespreek ik de software die tijdens mijn stageperiode aan bod gekomen is. Bij deze software hoort Apache, PHP, MySQL, eZ publish en SVN.

2.1 Apache

Apache is de meest gebruikte webserver wereldwijd sinds april 1996. De makers van Apache, de 'Apache Group' (www.apache.org, 2005), streven naar het maken van een veilige, efficiënte en uitbreidbare webserver die de huidige HTTP-standaarden ondersteunt.

De naam 'Apache' is er gekomen met 2 verschillende redenen. Ten eerste wil men respect tonen voor de Apaches, een indianenstam, die bekend waren voor hun superieure kennis in oorlogsstrategieën en hun uithoudingsvermogen. Ten tweede staat Apache voor 'A PAtCHy server'. Hoewel deze laaste reden niet volledig correct is, is ze toch overal geaccepteerd. 'A PAtCHy server' is ontstaan in het beginstadium van Apache. Het programma had veel te kampen met bugs. Deze bugs werden dan opgelost door middel van patches.

Maar wat houdt Apache nu eigenlijk in?

Apache is een sterke en flexibele webserver die de meeste protocollen ondersteunt. Een webserver is een programma dat verzoeken naar webpagina's ontvangt en dan de juiste documenten naar de verzoeker stuurt.

2.2 PHP

Dit onderdeel is vooral gebaseerd op informatie van de website van PHP (www.php.net, 2005).

PHP is een server-side scripting taal. Dit betekent dat de code niet door de webbrowser zelf wordt uitgevoerd zoals Javascript, maar op de server zelf waar de opgevraagde bestanden zich bevinden. Het voordeel is dat de mensen thuis de code niet te zien krijgen. PHP is een taal die eenvoudig is voor beginners en nog steeds krachtig is voor professionele gebruikers. PHP kan alles doen wat elk ander CGI script kan. Maar er zijn uiteraard nog andere grote doelen voor PHP, zoals server-side scripting en command line scripting.

- Server-side scripting

Dit is het meest traditionele doel van PHP. Het wordt ook aanzien als het hoofddoel van PHP. Om dit werkende te krijgen hebt u 3 zaken nodig. Het eerste is de PHP-parser die de PHP scripts verwerkt. Het tweede is een webserver die ervoor zorgt dat u de PHP-pagina kunt opvragen. Het derde is een webbrowser die ervoor zorgt dat u de webpagina's kunt gebruiken.

- Command line scripting

Deze scripts kunt u draaien zonder dat er een webserver of webbrowser aan te pas komt, enkel de PHP-parser voldoet. PHP is heel handig in de command line scripting taal voor het regelmatig uitvoeren van scripts via cron op Linux systemen of via Taak Planner op Windows systemen. Command line scripting is ook handig voor taken waar tekstverwerking aan te pas komt.

Met PHP hebt u de keuze om gebruik te maken van functioneel programmeren, object georiënteerd programmeren of de combinatie van beide. PHP4, de PHP versie die eZ publish gebruikt, ondersteunt nog niet alle object georiënteerde standaarden. Dit is uiteindelijk uit de wereld geholpen met PHP 5 die alle standaarden wel ondersteunt. PHP 5 is sterk verbeterd op 3 grote vlakken: het object georiënteerd programmeren, MySQL en XML. Voor MySQL is de extensie volledig herschreven. Voor XML zijn er nieuwe tools bijgekomen.

Een heel sterke kant van PHP is de ondersteuning voor databases. Het is heel simpel om PHP-pagina's te schrijven die gekoppeld zijn aan een database. Nu volgt er een lijstje met de meest gebruikte databases die momenteel worden ondersteund door PHP:

dBase	Ingres	Oracle (OCI7 and OCI8)
IBM DB2	Direct MS-SQL	PostgreSQL
ODBC	MySQL	Solid
Sybase	Unix dbm	SRDBMS

Ik wil u er op wijzen dat deze lijst met databases niet volledig is. Wanneer u toch met een database werkt die niet in de lijst voorkomt, kunt u kijken of de database ODBC ondersteunt. Als dit zo is, kan PHP hier gebruik van maken om toch een connectie te leggen. Nog een andere mogelijkheid is dat PHP een database abstractie extensie (DBX) heeft. Zo is PHP in staat een connectie te leggen met elke database die DBX ondersteunt.

PHP ondersteunt ook communicatie tussen verschillende systemen. Zo worden er protocollen als LDAP, IMAP, SNMP, NNTP, POP3, HTTP en nog vele anderen ondersteund.

2.3 MySQL

MySQL is 's werelds snelst groeiende en populairste open source databasesysteem. MySQL heeft een aantal sterke punten.

U kunt in MySQL een zeer betrouwbaar en performant databasesysteem vinden. Voor het afgewerkt product werd vrijgegeven mocht de MySQL community het systeem onderwerpen aan maandenlange en intensieve testen. Uiteindelijk werd het zeer betrouwbaar bevonden door de community om MySQL in de bedrijvenwereld te introduceren.

Het is zeer eenvoudig om MySQL te installeren en er mee te werken. Door de architectuur van het systeem kan het databasesysteem snel aangepast worden. Door de unieke kern van MySQL is het systeem snel in omgang, zeer compact en stabiel.

Een sterk punt van MySQL is de platformonafhankelijkheid. Het wordt op meer dan 20 verschillende platformen ondersteund. Zo kan u MySQL gebruiken op alle grote Linux distributies, Mac, Unix en Microsoft Windows.

Er zijn veel tools uitgebracht die het gebruiksgemak verhogen van MySQL. Zo bevat de website van MySQL (www.mysql.com) tools zoals MySQL Administrator en MySQL Query Browser. MySQL Administrator geeft een grafische interface om de MySQL servers te onderhouden en geeft een duidelijk zicht op de status van het systeem. MySQL Query Browser is een eenvoudige visuele tool die de mogelijkheid geeft om SQL-queries te maken en uit te voeren.

Tijdens mijn stage gebruikte ik voornamelijk de tool SQLyog. SQLyog lijkt zeer sterk op MySQL Query browser. Ik gebruikte deze tool voornamelijk om een tabel in de MySQL database te voegen en om de tabellen te bekijken. SQLyog kunt u vinden op de website van WebYog (www.webyog.com).

2.4 eZ publish

Tijdens mijn stageperiode op het SCK•CEN heb ik vooral gewerkt met eZ publish. eZ publish is een open source content management systeem en een ontwikkelingsframework. eZ publish staat toe om professionele, dynamische websites te bouwen. eZ publish heeft twee licenties, een GPL (General Public License) en een professionele licentie. Door de GPL te gebruiken kunnen mensen eigen open source applicaties maken en zo hun steentje bijdragen aan de ontwikkeling van open en vrije software. De professionele licentie laat bedrijven toe om commerciële software te maken en te verkopen. eZ publish is platformonafhankelijk zodat het kan gebruikt worden op een Windows-systeem of andere UNIX varianten zoals OS X, Linux, FreeBSD, Solaris, IRIX, … eZ publish is ook database onafhankelijk. Wanneer een database niet wordt ondersteund, kan er nog altijd een driver voor worden geschreven zonder eZ publish te moeten veranderen.

2.4.1 Content management system

In een content management systeem wordt alles opgeslagen als content. De rol van een content management systeem is het organiseren van deze content. Een content management systeem zorgt ervoor dat u de inhoud van een website op een eenvoudige manier kunt aanpassen.

2.4.2 Development Framework

eZ publish biedt een brede waaier aan ingebouwde functionaliteiten. Omdat eZ publish niet aan alle noden van de gebruiker kan voldoen, is het systeem zo opgebouwd dat het eenvoudig uitbreidbaar en aanpasbaar is. Om het systeem uit te breiden met functionaliteiten is er enige PHP kennis vereist. Doordat het systeem zo uitbreidbaar en aanpasbaar is, is het mogelijk om het systeem vlot te laten samenwerken met andere systemen.

2.4.3 Basis van eZ publish

eZ publish is een volledig object georiënteerde applicatie die geschreven is in PHP. Het systeem bestaat uit drie delen: libraries, de kernel en modules.

De libraries zijn herbruikbare PHP klassen die gebruikt worden als de bouwstenen van het systeem. Het voordeel van deze libraries is dat ze niet afhankelijk zijn van eZ publish en kunt u ze dus overal voor gebruiken.

De kernel kan beschreven worden als de kern van het systeem. Deze handelt alle zaken af zoals het omgaan met content, workflows, toegang tot de database. De kernel maakt gebruik van de libraries om goed te kunnen werken.

De modules van eZ publish kunnen bekeken worden als de interfaces van de kernel. Ik zal u nu twee kleine voorbeelden geven wat bepaalde modules als functionaliteit aanbieden:

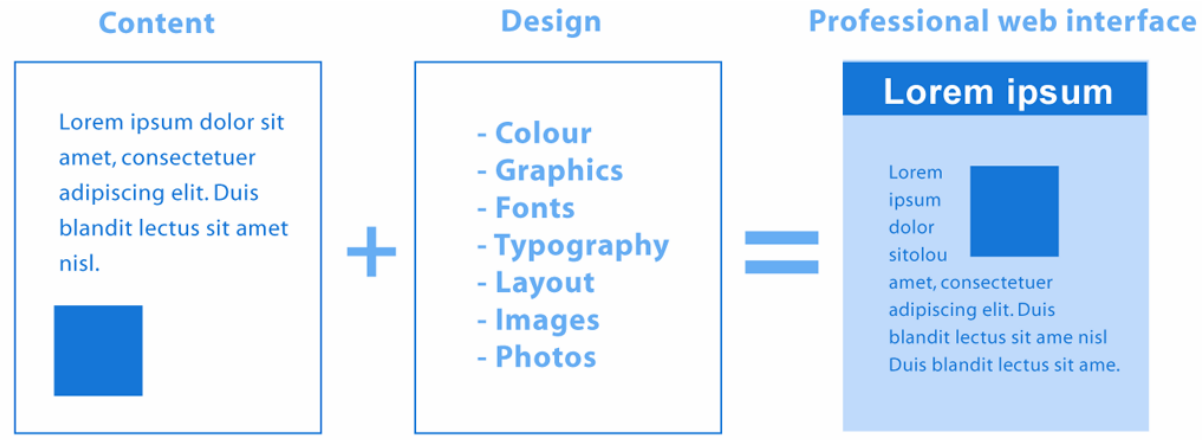
- De content module biedt functionaliteiten voor het onderhouden van content
- De user module zorgt onder andere voor de functionaliteit in verband met toegangsrechten.

Elke module in eZ publish is ontworpen om vlot te kunnen samenwerken met andere modules. Door de modules te gebruiken, kunnen ontwikkelaars snel professionele systemen maken.

2.4.4 Content en Design

In eZ publish zijn content en design volledig van elkaar gescheiden. Met content bedoel ik de informatie die wordt gestructureerd en opgeslagen zodat die later makkelijk terug opvraagbaar is. De content is maar een ruwe structuur van informatie. Het design zorgt ervoor dat het op een aangename wijze wordt getoond op het scherm. Dit gebeurt door middel van stylesheets, images, lay-outs, enz.

De volgende afbeelding toont hoe eZ publish de content en het design samenvoegt om een professionele webinterface te verkrijgen.



figuur 2.1 Content & Design

Dit is één van de grootste kwaliteiten van eZ publish. Door de scheiding van content en design kan één content object meerdere opmaken krijgen. De content wordt ook veilig gesteld omdat die niet kan veranderd worden wanneer er eens een klein foutje wordt gemaakt in het design.

2.4.5 De content structuur

In tegenstelling tot andere content management systemen, heeft eZ publish geen voorgedefinieerd content model. eZ publish laat de ontwikkelaar toe om zijn eigen content structuur te maken door middel van een uniek object georiënteerde aanpak. Zo wordt het makkelijker voor de ontwikkelaar om data te structureren, op te slaan, op te zoeken en te presenteren. Maar voor het gebruiksgemak van de gewone thuisgebruikers heeft eZ publish al een paar kant-en-klare structuren. Natuurlijk is het mogelijk om deze structuren uit te breiden en aan te passen.

2.4.6 De content klasse

De content klasse kan op een heel simpele manier uitgedrukt worden. Een content klasse in eZ publish is een datastructuur. Het bevat geen data maar zorgt er wel voor dat de informatie op een gestructureerde manier wordt opgeslagen in de database. eZ publish heeft standaard al een paar content klassen klaargestoomd voor de gewone gebruikers. Een paar voorbeelden van deze klassen zijn artikels, folders, user accounts,… Deze standaard content klassen kunnen gebruikt worden in het dagelijkse leven. Om zelf een goede website te bouwen, kunt u zelf uw eigen klassen maken of aanpassen zodat ze helemaal voldoen aan uw eigen behoeftes.

Een content klasse bevat ook attributen. Een attribuut kan bijvoorbeeld een string, een integer, een afbeelding, enz. zijn. Maar de datatypes blijven niet beperkt tot simpele datatypes zoals de string of integer. In eZ publish kunt u zelf uw eigen datatypes schrijven die dan in de content klasse kunnen gebruikt worden. Het attribuut wordt bepaald door het datatype dat is gekozen voor dat specifieke attribuut. Hieruit kunt u ook besluiten dat een content klasse bestaat uit één of meerdere datatypes.

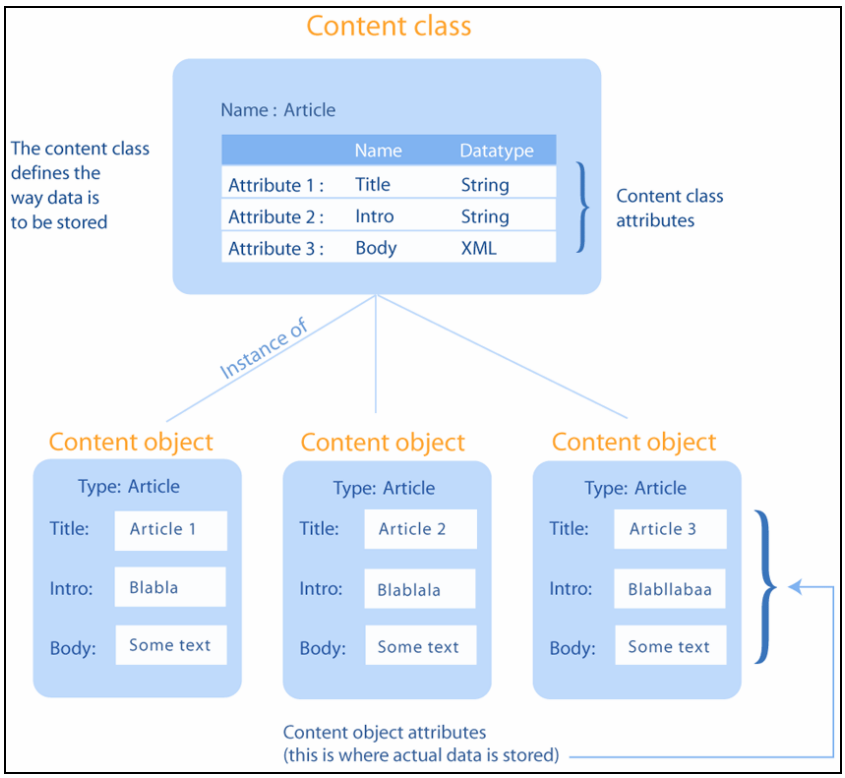
U kunt een paar voorbeelden van content klassen vinden in het deel met de klassen van de publieke website ACTINET (zie 6). Deze content klassen worden op deze plaats ook uitgelegd.

2.4.7 Het content object

Een content object is een instantie van de content klasse. De content klasse definieert de structuur van een content object door het gebruik van klasse attributen. Deze content objecten bevatten alle data die wordt gecreëerd in eZ publish. Na het maken van een content klasse kunt u zoveel content objecten maken van die content klasse als u wilt.

Een content object bevat ook content object attributen. Deze komen overeen met de attributen van de content klasse. Een content object kan één of meerdere content object attributen bevatten.

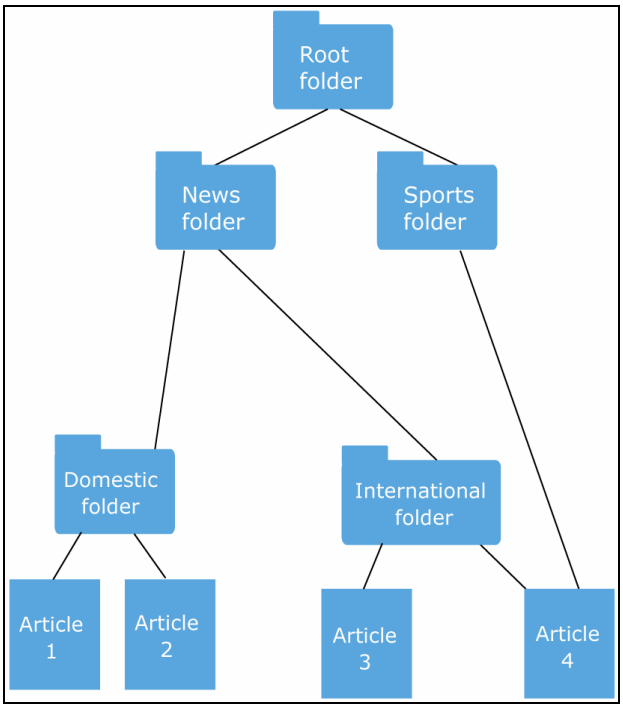
In de afbeelding (figuur 2.2) kunt u zien hoe u van een content klasse meerdere content objecten kunt maken.



figuur 2.2 Content klasse en content objecten

2.4.8 Nodes en de content node tree

Een node kan gezien worden als een soort capsule die verwijst naar een content object. De content node tree is gebouwd uit vele nodes. Deze content node tree structuur toont de achterliggende hiërarchie van de content objecten op een visuelere manier. Deze samenhang van nodes zorgt ervoor dat de content objecten gestructureerd blijven voor de gebruiker. Op figuur 2.3 kunt u een voorbeeld zien van een content node tree. Toch is deze afbeelding niet helemaal correct. De root folder is niet de echte root folder van eZ publish. De echte root folder heeft 3 onderverdelingen: content folder, user folder, media folder. De root folder op figuur 2.3 is dus eigenlijk de content folder.

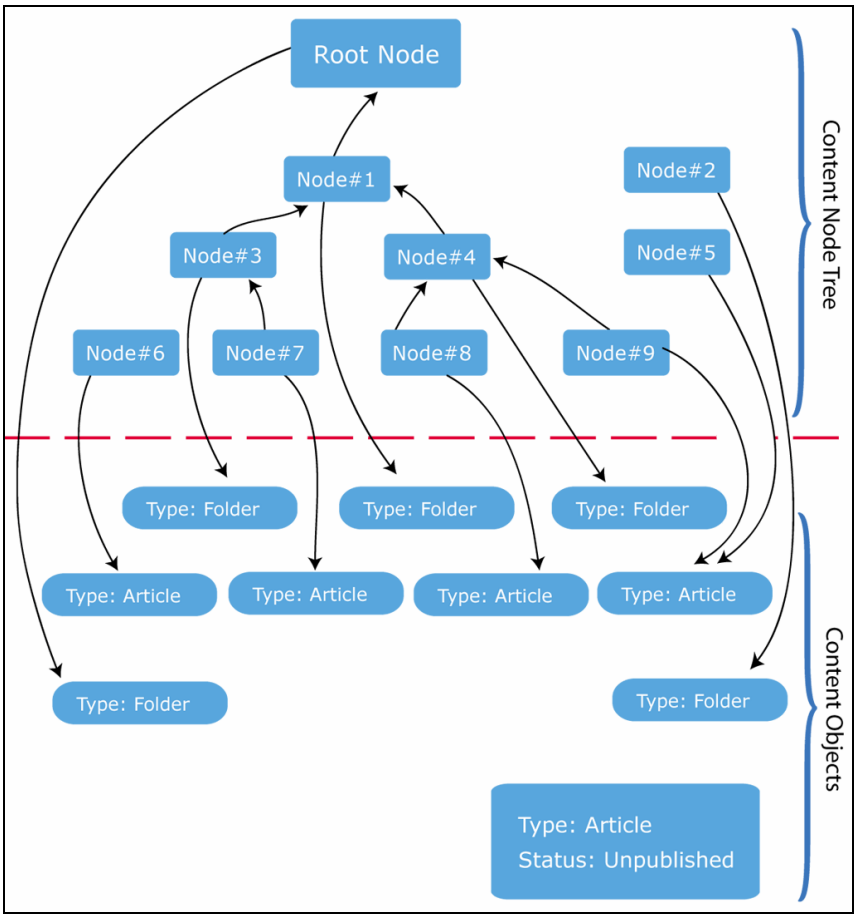


Figuur 2.3 Content Node Tree

U kunt nog een voorbeeld van een content node tree bekijken in bijlage 1. Deze content node tree komt van de publieke website die ik tijdens stage moest maken. De content node tree is de basis van de publieke website.

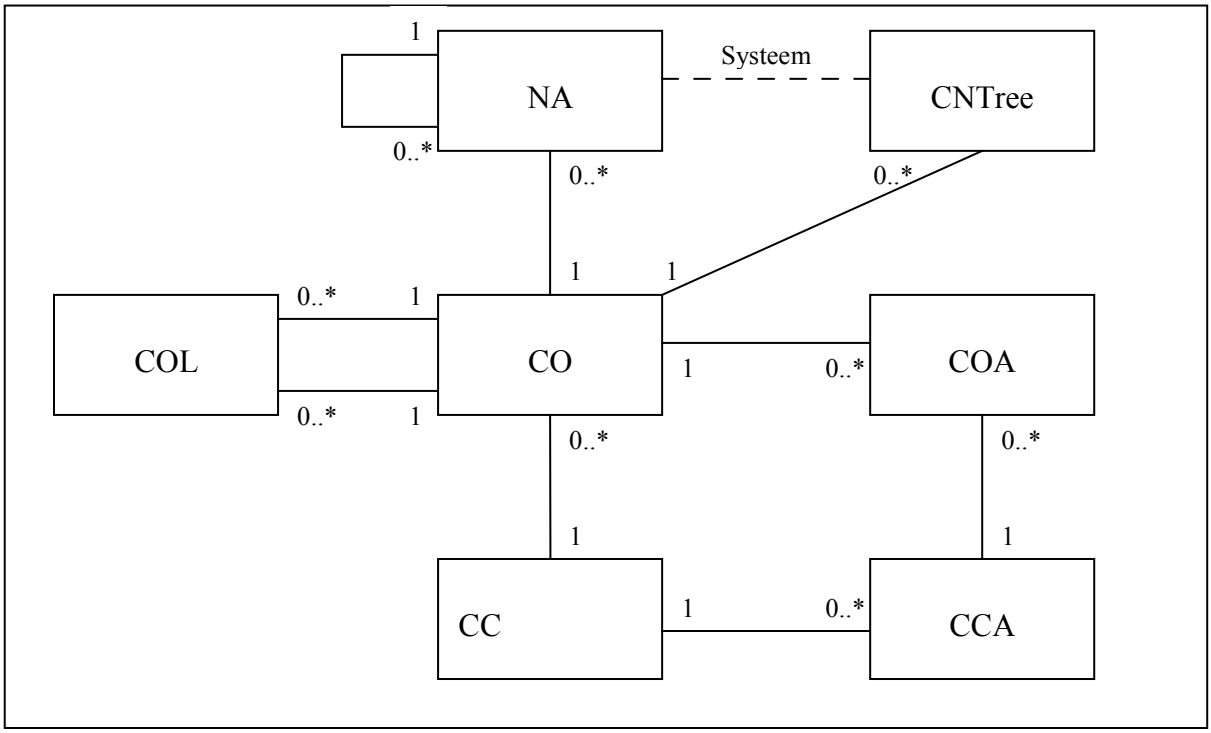
Een content node tree bestaat uit nodes. Het totale minimum dat een content node tree kan bevatten is één node en deze node wordt de rootnode genoemd. Elke node heeft een node assignment. Dit is een soort van kaartje waar alle gegevens op staan van de node. De node assignment bevat onder andere twee verwijzingen: eentje naar zijn parent node en eentje naar een content object. Er is maar één enkele node die geen verwijzing naar een parent node kan hebben en dat is de root node. Het is mogelijk dat bijvoorbeeld twee nodes naar hetzelfde content object verwijzen. Zo kan een content object op verschillende plaatsen verschijnen in de content node tree. Figuur 2.4 toont een content node tree die relaties legt met content objecten. In deze content node tree kunt u zien dat node 9 en node 5 een relatie hebben met één content object.

Als u de structuur van een content tree node vergelijkt met deze van de node assignments kunt u zelfs zien dat deze bijna hetzelfde is. Eigenlijk wordt de content tree node opgebouwd met behulp van de node assignments.



Figuur 2.4 Content Node Tree & Content Objecten

2.4.9 Simpel relationeel model van eZ publish



In dit deeltje ga ik kort de structuur van de database schetsen. Dit zal niet de hele database zijn maar de hoofdtabellen.

Ik ga beginnen met de tabel van de content klassen, de tabel *ezcontentclass* (CC). Omdat een content klasse meerdere attributen kan hebben, heeft deze tabel een één op meer relatie met de tabel *ezcontentclass_attribute* (CCA). Van elke content klasse kunnen er meerdere content objecten gemaakt worden. Dus krijgt de tabel *ezcontentclass* (CC) een één op meer relatie met de tabel *ezcontentobject* (CO). Elk content object kan net zoals de content klasse meerdere attributen hebben. Dus krijgt de tabel *ezcontentobject* een één op meer relatie met de tabel *ezcontentobject_attribute* (COA). Als we de lijn doortrekken van de relatie tussen content klassen en content objecten, is het logisch dat er een één op meer relatie wordt geplaatst tussen de tabel *ezcontentclass_attribute* en *ezcontentobject_attribute*.

Nu ga ik uitleggen hoe de node assignment, content node tree en de content objecten tegenover elkaar staan. Ik heb al uitgelegd dat een node een soort van bijhorend kaartje bevat met alle gegevens op. Deze kaartjes worden door het systeem opgeslagen in de tabel *eznode_assignment* (NA). Omdat een content object bij meerdere nodes kan horen, dus automatisch ook bij meerdere node assignments, wordt een één op meer relatie gelegd tussen de tabel *ezcontentobject* (CO) en de tabel *eznode_assignment* (NA). Een node assignment kan ook een link naar zichzelf hebben. Dit is de link tussen parent en child. Dus de tabel *eznode_assignment* (NA) heeft een één op meer relatie met zichzelf. De relatie tussen content object en content node tree zal dezelfde zijn als de relatie tussen content object en node assignment: een één op meer relatie. Deze relatie is hetzelfde omdat de content node tree wordt gegenereerd door het systeem uit de node assignments.

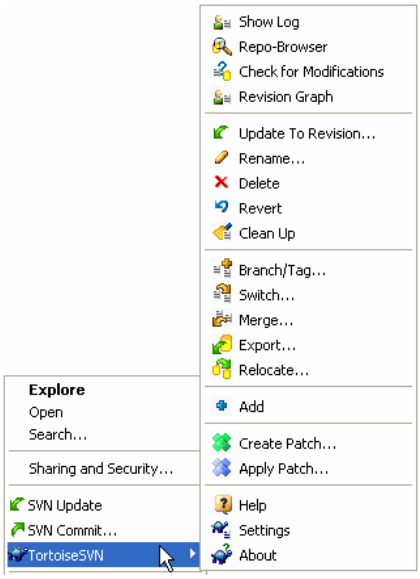
eZ publish heeft ook de mogelijkheid om relaties te leggen tussen objecten. Deze worden related objects genoemd. Deze relatie wordt opgeslagen in de tabel *ezcontentobject_link* (COL). Op het diagram staan twee één op meer relaties tussen content object en content object link. Dit komt doordat eerste relatie van het type *from* is en de tweede relatie van het type *to*. Het type *to* betekent dat elk object links krijgt waar hij naar toe verwijst. Het type *from* betekent dat een object ook te weten krijgt welke objecten naar hem verwijzen. Hier komen dus de twee relaties vandaan.

Voor diegene die geïnteresseerd is in de volledige databasestructuur van eZ publish, kan het volledige databasemodel bekijken op de website van eZ publish (http://www.ez.no/ez_publish/documentation/reference/database_diagram).

2.5 SVN

Subversion, ofwel SVN, is een open source Version Control System. Om dit Version Control Programma te bedienen gebruik ik TortoiseSVN. TortoiseSVN is de grafische schil van Subversion. Zo wordt de bediening van Subversion een stuk eenvoudiger. Samen vormen de twee programma's één groot geheel.

Een Version Control System is een programma dat ervoor zorgt dat er versies worden gemaakt van een bepaald bestand. Per versie worden de veranderingen bijgehouden die er gebeurd zijn. Het voordeel is dat u altijd naar een oudere versie van het bestand kunt terugkeren. Het kan wel eens voorvallen dat bij het programmeren ergens een fout in de code sluipt en u de fout niet meer terugvindt. Dan is het handig om in SVN de laatst werkende versie op te halen en zo terug opnieuw te beginnen.



figuur 2.5 TortoiseSVN

TortoiseSVN heeft een heel groot voordeel. Het programma is geïntegreerd in de Shell van Windows. Dit merkt u niet tot u in Windows Verkenner het snelmenu oproept. Een voorbeeld van dit snelmenu kunt u zien op figuur 2.5. Met TortoiseSVN is het ook mogelijk om met andere filemanagers te werken. Het programma is niet enkel beperkt tot Windows verkenner. Om te zien in welke staat een bestand zich bevindt, toont TortoiseSVN een aangepast icoontje bij het bestand.

3 Standaarden

In dit hoofdstuk zal ik wat meer uitleg geven over de technieken die ik gebruikt heb bij het maken van de website. Zo zullen XHTML, CSS en de validator aan bod komen.

3.1 XHTML

HTML was de populairste document markup taal in heel de wereld. Maar bij de intrede van XML werd er een tweedaagse conferentie georganiseerd. Hier heeft men gediscussieerd over de mogelijke introductie van een nieuw soort HTML. Het antwoord was een duidelijke 'ja'. Zo is men begonnen aan de markup taal XHTML. Met XHTML werd het eenvoudiger om HTML en XML te combineren met elkaar. Bij het maken van XHTML werd van de gelegenheid gebruik gemaakt om een paar slordige delen van HTML op te kuisen en om nog extra functionaliteiten toe te voegen.

Het verschil tussen HTML en XHTML is nauwelijks merkbaar. De voordelen van XHTML komen pas tevoorschijn bij het gebruiken van meer en meer tools die XML gebaseerd zijn. Een voorbeeld van een tool is XSLT die instaat voor het transformeren van documenten.

XHTML is een strengere en meer opgekuiste versie van HTML. Een belangrijke verandering is het afsluiten van tags. Ik zal dit demonstreren met een klein voorbeeldje.

HTML

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  <li>Milk</li>
</ul>
```

XHTML

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```

Het is ook belangrijk dat XHTML goed wordt genest zoals XML dat doet. Hier volgt een klein voorbeeldje

```
Fout: <b><i>This text is bold and italic</b></i>

Goed: <b><i>This text is bold and italic</i></b>
```

Andere veranderingen zijn:

- Tags moeten in kleine letters worden geschreven
- Alle attributen moeten quotes krijgen
- Het name-attribuut wordt vervangen door het id-attribuut

3.2 CSS

De Cascading Style Sheets (CSS) zorgen ervoor dat u XHTML elementen een opmaak kunt geven. De tags van XHTML zijn ontworpen om iets te zeggen over de inhoud die ze bevatten en niet over de opmaak. Om toch op een goede wijze een opmaak aan webpagina's te geven, kunt u CSS gebruiken.

Maar CSS werkt niet altijd even vlot op elke webbrowser. Tijdens het ontwikkelen en testen van webpagina's, gebruik ik Internet Explorer (IE) en Mozilla Firefox. Waar Firefox probleemloos een webpagina toont, gaat Internet Explorer helemaal in de fout. Ik zal dit nu aantonen met een van de ergste bugs die ik ben tegengekomen tijdens mijn stageperiode.

Deze bug noemt de peek-a-boo bug. De peek-a-boo bug wordt in IE uitgelokt door het gebruik van floats. De bug kunt u herkennen aan het verdwijnen van alle inhoud op uw webpagina. Zelfs wanneer u uw webpagina herlaadt blijft de inhoud van de webpagina op een mysterieuze wijze verdwenen. Soms komt de inhoud van uw webpagina plots tevoorschijn wanneer u met de muis eens over de webpagina sleept. Scrollen in een webpagina lijkt de inhoud ook te doen verschijnen.

Nu zal ik een paar tips geven die ervoor zorgen dat u minder last krijgt van peek-a-boo bugs.

- Zorg ervoor dat een div niet in aanraking komt met een element die de eigenschap 'float' bezit
- Een element met de eigenschap 'float' moet u een specifieke breedte of hoogte geven

3.3 Validator

Op mijn stage heb ik de validator van W3C gebruikt. Deze zorgde ervoor dat mijn XHTML bestanden voldeden aan de standaarden en normen van het web. W3C is de afkorting voor 'World Wide Web Consortium'. Dit consortium werkt samen met het publiek om web standaarden te ontwikkelen.

Wanneer u een volledige website hebt gemaakt die werkt in Internet Explorer, is er toch een kans dat de website niet valid is. Dit wil zeggen dat de kans zeer groot is dat uw website in striktere webbrowsers, zoals Firefox en Opera, er niet uit zal zien. Zelfs als de code valid is, kan het zijn dat de webpagina in andere webbrowser verkeerd wordt getoond. Dit komt doordat niet elke webbrowser hetzelfde met XHTML bestanden omgaat. Geen enkele webbrowser zal een XHTML-pagina perfect tonen. Maar er zijn webbrowser die toch dicht in de buurt komen. Om er toch voor te zorgen dat uw XHTML code voldoet aan de standaarden van W3C, zijn er twee dingen die u kunt doen. U kunt alle standaarden uit uw hoofd leren of u kunt uw code door een validator halen.

Voor mijn stageopdracht heb ik gekozen om mijn code te laten controleren door de validator van W3C. De validator van W3C is een online service die door iedereen gratis te gebruiken is. Door de validator te gebruiken kunt u ervoor zorgen dat uw webpagina's van uitstekende kwaliteit zijn. Wanneer uw webpagina toch fouten bevat, zal de validator deze aanduiden met een duidelijke foutmelding.

4 Stageopdracht

Mijn stageopdracht bestaat uit het maken van een publieke website voor het Europese 'ACTINET – Network of Excellence for Actinide Sciences'. De ACTINET website heeft vier belangrijke doelen.

Het eerste doel is het grote publiek bewust maken hoe belangrijk het onderzoek van de Actinides is en hoe belangrijk dit is voor het aanhoudende gebruik van nucleaire energie in de toekomst. Het is de bedoeling dat de ACTINET website het vertrouwen van het grote publiek terugwint in verband met nucleaire energie. Sinds het ongeval in Tsjernobyl is het wantrouwen in de nucleaire energie sterk gestegen.

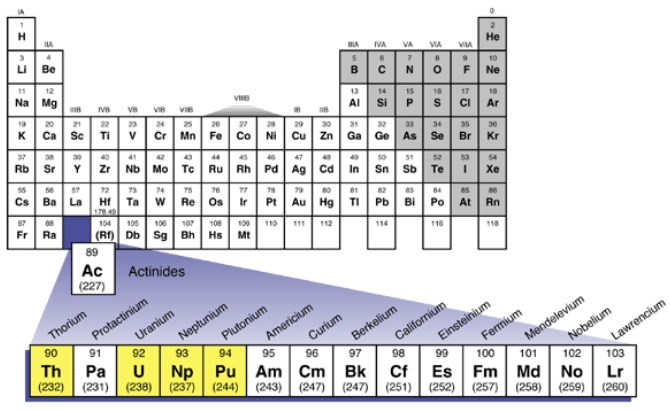
Het tweede doel is het ACTINET netwerk promoten als een toegevoegde waarde voor bijvoorbeeld de samenwerking op Europees vlak, voor nationale regeringen.

Het derde doel is het aanmoedigen van alle getalenteerde Europese studenten en wetenschappers om zich aan te sluiten bij een ACTINET partner.

Het laatste doel is de samenwerking aanmoedigen met niet-Europese partners zoals Rusland, USA, Japan, enz.

4.1 ACTINET

In de sector van de nucleaire energie spelen de Actinide elementen een grote rol. Twee van de Actinide elementen, uranium en plutonium, worden bekeken als sleutelelementen in de brandstof die gebruikt wordt voor een nucleaire reactor. Wanneer deze twee elementen gebombardeerd worden met neutronen, laten deze een grote hoeveelheid energie vrij. Het grote probleem is het afval dat door dit proces wordt geproduceerd. Dit afval is langdurig hoog radioactief en onstabiel.



Figuur 4.1 De Actinide groep

Een van de grootste opdrachten van de ACTINET partners is het vinden van een goede aanpak in het omgaan met radioactief afval, vooral het hoog radioactief afval. Er is veel onderzoek nodig om de afvalproductie te verminderen en beter gebruik te maken van de bestaande materialen.

Er zijn weinig laboratoria in Europa die over de kennis en gereedschappen beschikken voor het beoefenen van de Actinide wetenschap. Geen enkel laboratorium omvat het geheel van de Actinides. Daarom zou de toekomstige kennis en gereedschappen een proces moeten ondergaan om geïntegreerd en geoptimaliseerd te worden in elk laboratorium. 'Network of Excellence' geeft dit al een stapje doordat het kennis vergaart op één centrale plaats en dit gebeurt nu via het netwerk ACTINET.

4.2 Network of Excellence

ACTINET is een 'Network of Excellence'. 'Network of Excellence' geeft drie eigenschappen aan een netwerk. De eerste eigenschap is het coördineren en integreren van de activiteiten die worden beoefend door de leden van het netwerk. De tweede eigenschap is het verdelen van de taken tussen de verschillende leden van het ACTINET netwerk voor een bepaalde onderzoeken. Zo een samenwerking wordt ook Joint Research Program genoemd. Alle informatie wordt dan centraal gezet en gedeeld met de meewerkende leden. Dit is zeer handig omdat de meeste leden van een netwerk geografisch gezien ver van elkaar verwijderd liggen. De derde eigenschap is het organiseren van activiteiten zoals trainingen voor onderzoekers en ander personeel.

Een 'Network of Excellence' wordt voor 4 of 5 jaar financieel ondersteund door de Europese Commissie. Wanneer deze financiële steun wordt stopgezet is het de bedoeling dat het 'Network of Excellence' blijft bestaan en zijn vruchten blijft afwerpen.

4.3 The Sixth Framework Programme

ACTINET hoort bij het Sixth Framework Programme. Het Sixth Framework Programme of FP6 is een Europees kaderprogramma voor onderzoek, technologische ontwikkeling en demonstratie. Het is een collectie van acties op Europees niveau om dit onderzoek financieel te steunen.

Het hoofddoel van FP6 is het verbeteren van de integratie en coördinatie van wetenschappelijke onderzoeken in Europa. Momenteel zijn de meeste wetenschappelijke onderzoeken verspreid over Europa. Ook willen ze een structuur brengen in de Europese onderzoeken. Zo kunnen ze de grondbeginselen versterken van het Europese onderzoek en komen tot één geheel.

5 Korte introductie tot eZ publish template code

In dit deeltje ga ik een korte introductie geven tot de eZ publish template code. Hier staat enkel een korte basis van de eZ publish template code. Om de uitgebreide informatie te bekijken van eZ publish code, kunt u naar de website kijken van eZ publish (www.ez.no, 2005)

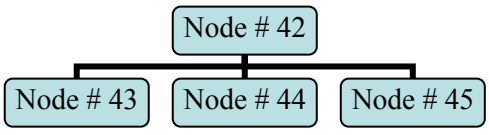
5.1 Fetch

De fetch operator spreekt de de modules van eZ publish aan die data uit de database halen en aan de template doorgeven. Tijdens mijn stageperiode heb ik de fetch voornamelijk gebruikt om data op te halen via de content module.

```
fetch( <module>, <function>, <parameters> )
```

Bij het opvragen van een fetch moeten er 3 parameters worden meegegeven. De eerste parameter is de naam van de module die de fetch moet gebruiken. De tweede parameter is de functie die moet worden gebruikt. Deze functie moet uit de module komen die is meegegeven met de eerste parameter. De derde parameter moet een array van parameters meegeven die de functie nodig heeft om een selectie te maken.

Ik zal nu een kort en simpel voorbeeld geven voor het ophalen van objecten.




```
{fetch( 'content',
        'list',
        hash( 'parent_node_id', 42 )}
```

Dit voorbeeldje zal zorgen voor het ophalen van de child nodes van de node met id 42.

In de fetch wordt de functie *list* van de content module gebruikt. Deze functie zorgt ervoor dat u een array met nodes terugkrijgt. De parameter die *list* meekrijgt is de parent node id. Bij *parent_node_id* wordt de id van de parent node geplaatst.

5.2 Section

Met een section krijgt u de mogelijkheid om arrays te doorlopen of een conditie te stellen.

Een simpel voorbeeld van een loop:

```
{section var=loop loop=array( 'appel', 'banaan', 'peer' )}
      {$loop}
{/section}
```

Het resultaat is:

```
appel banaan peer
```

In dit voorbeeldje overloopt de section elk item van de array en drukt het item dan af op het scherm.

Een simpel voorbeeld van een conditie:

```
{section var=loop loop=array( 'appel', 'banaan', 'peer' )}
      {section show=eq( $loop , 'banaan' )}
            <u>{$loop}</u>
      {section-else}
            {$loop}
      {/section}
{/section}
```

Resultaat:

```
appel banaan peer
```

In dit voorbeeld wordt elk item in de loop gecontroleerd of het gelijk is aan de string *banaan*. U kunt het eigenlijk vergelijken met de IF-ELSE-conditie.

5.3 Let

Met let krijgt u de mogelijkheid om variabelen te declareren. Deze variabelen zullen alleen bruikbaar zijn binnen de start-tag en de eind-tag van deze let. Ik zal dit even aantonen met een klein voorbeeldje.

```
{let  varhello='Hello'
      varworld='World'
}
      {$varhello}{$varworld}
{/let}
```

Het resultaat van het voorbeeld zal zijn:

```
Hello World
```

5.4 Switch

De switch functie laat toe om variabelen te controleren zoals een geneste IF-functie. Ik zal dit aantonen met een klein voorbeeldje.

```
{let myvar=1 message='Hello World'}
      {switch match=$myvar}
            {case match=1}
                  {$message}
            {/case}

            {case match=2}
                  Dit is geen Hello World.
            {/case}

            {case}
                  Deze ook niet.
            {/case}
      {/switch}
{/let}
```

Het resultaat van dit voorbeeld zal uiteindelijk 'Hello World' opleveren. Maar mocht de variabele *myvar* nu het getal 2 bevatten, dan zou het resultaat 'Dit is geen Hello World' zijn. Als geen van beide cases zouden overeenkomen met de variabele *myvar*, dan zou de standaard case worden getoond. In dit geval is dan het resultaat 'Deze ook niet'.

6 Klassen

In dit hoofdstuk komen de klassen van de ACTINET website aan bod. Deze klassen worden door eZ publish gebruikt om content objecten te maken. Voor ik een klasse kon maken moest ik eerst kijken welke attributen er allemaal nodig zijn voor een bepaalde klasse.

6.1 Member

De klasse *Member* zorgt voor het maken van content objecten die de informatie bevat van de ACTINET partners. Deze partners zijn vooral bedrijven, universiteiten of instituten die deel uitmaken van het Europese netwerk ACTINET. Volgende eisen waren gesteld zodat deze klasse ten volle wordt benut:

- de naam en de afkorting van de partner
- een korte beschrijving van de partner
- de mogelijkheid om een presentatie van de partner te downloaden
- het logo van de partner
- het adres van de partner
- de gegevens van de contactpersonen van de partner

In de tabel hieronder kunt u alle attributen van de klasse *Member* terugvinden. U kunt zien dat er per attribuut een aantal specificaties worden meegegeven. De voornaamste specificaties zijn *name* en *identifier*. De identifier zal ik later voornamelijk in de template van partners gebruiken om een attribuut van een content object op te roepen. U kunt ook aanduiden of een attribuut verplicht is. Door het verplicht maken van attributen, kunt u bepaalde informatie afdwingen van de gebruiker. In de klasse *Member* is alleen de shortname een verplicht attribuut. Ook kunt u ervoor kiezen of het item *searchable* is. Dit dient voor de ingebouwde zoekfunctionaliteit van eZ publish.

Klasse Member			
Name	**Identifier**	**Datatype**	**Options**
Shortname	shortname	Text line	verplicht
Name	name	Text line	
Address	address	Text line	
Postalcode	postalcode	Text line	
City	city	Text line	
Country	country	Country Selection	
Description	description	XML Text field	
Logo	logo	Image	
Presentation	presentation	Binary file	

Let wel op dat er in de klasse *Member* geen gegevens staan over de contactpersonen. Deze worden later door de klasse *Contactperson* toegevoegd (zie 6.2). Om meerdere contactpersonen toe te voegen, heb ik voor deze methode gekozen. Daarom krijgt de klasse *Member* de eigenschap van container mee. Zo kan een content object van deze klasse childnodes bevatten.

6.2 Contactperson

De klasse *Contactperson* maakt deel uit van partners. Omdat een ACTINET partner meerdere contactpersonen kan hebben, moet er onder elk content object van de klasse *Member* meerdere content objecten geplaatst worden van de klasse *Contactperson*. Bij elke contactpersoon moeten er bepaalde gegevens worden opgeslagen. Hiervoor moeten er attributen worden voorzien die het volgende omvatten:

- naam en voornaam van de contactpersoon
- de functie die de contactpersoon uitoefent bij de ACTINET partner
- het departement waar de contactpersoon zijn functie uitoefent
- het telefoonnummer van de contactpersoon
- het faxnummer van de contactpersoon
- het e-mailadres van de contactpersoon

De attributen die uiteindelijk zijn gemaakt, zijn:

Klasse Contactperson			
Name	**Identifier**	**Datatype**	**Options**
Name	name	Text line	
Firstname	firstname	Text line	
Position	position	Text line	
Department	department	Text line	
Mail	mail	Email	
Telephonenumber	telephonenumber	Text line	
Fax	fax	Text line	

6.3 Imagemap

De klasse *Imagemap* dient voor het maken van de clickable map bij Partners. Hier kunt u een titel, ondertitel en een kleine uitleg waarvoor de map dient, opgeven. De attributen die uiteindelijk zijn gemaakt, zijn:

Klasse Imagemap			
Name	**Identifier**	**Datatype**	**Options**
Title	title	Text line	
Subtitle	subtitle	Text line	
Map explanation	map_explanation	XML Text field	
Map	map	Image	

6.4 Webpage

Deze klasse zal dienen om een standaard webpagina te maken. Het enige wat deze klasse bevat is een titel, een intro en een body. Voor de intro en de body zijn XML text fields gebruikt. Dit wil zeggen dat ze allebei een opmaakwerkbalk krijgen waar de extensie *ezdhtml* voor zorgt. Deze extensie zal in dit werk nog worden besproken (zie 7.6). Hieronder kunt u de attributen vinden die in de klasse *Webpage* worden gebruikt.

Klasse Webpage			
Name	**Identifier**	**Datatype**	**Options**
Title	title	Text line	
Intro	intro	XML Text field	
Body	body	XML Text field	

6.5 Job Offer User

Job Offer User is een speciale klasse die niet voor gewone content objecten zorgt, maar voor content objecten die dienen als user accounts. Met deze user account zal het mogelijk zijn om op de website in te loggen en er advertenties op te plaatsen. De user account heeft een paar attributen nodig om in te loggen. Dit zijn:

- de naam en de voornaam
- het e-mailadres

Deze gegevens moet u verplicht invullen omdat hieruit een login en een paswoord worden gegenereerd. Er moeten ook nog gegevens van het bedrijf worden voorzien. Hier krijgt u 2 keuzes:

- een combobox met alle ACTINET partners

of

- naam van het bedrijf
- het adres van het bedrijf

Daarna zijn er nog een paar optionele gegevens:

- Telefoonnummer
- Faxnummer

De attributen die uiteindelijk zijn gemaakt kunt u hieronder vinden. Onder al de andere attributen bevinden zich twee speciale attributen. Het eerste speciale attribuut is van het datatype *User Account*. In dit attribuut komt de login en het paswoord die automatisch worden gegenereerd. Ook zal hier het e-mailadres in worden geplaatst. Het tweede speciale attribuut is van het datatype *Detailed Object Relation*. Deze is standaard niet in eZ publish aanwezig, maar wordt aangeboden door de extensie *objrel*. *Objrel* wordt besproken verder in dit eindwerk (zie 7.9).

Klasse Job Offer User			
Name	**Identifier**	**Datatype**	**Options**
Name	name	Text line	Required
Firstname	firstname	Text line	Required
Company	company	Text line	
Fax	fax	Text line	
Telephonenumber	telephonenumber	Text line	
Actinet partner	actinet_partner	Detailed Object Relation	
User account	user_account	User account	Required
Street	street	Text line	
Number	number	Text line	
PO box	po_box	Text line	
Postalcode	postalcode	Text line	
City	city	Text line	
Country	country	Country Selection	

6.6 Resume User

Resume User is net zoals Job Offer User. Het is een klasse die dient om user accounts aan te maken. Een Resume User kan inloggen als werkzoekende om zijn CV op de website te plaatsen. Voor de registratie zijn er de volgende gegevens nodig:

- de naam en voornaam (verplicht)
- het e-mailadres (verplicht)
- het telefoonnummer
- het faxnummer

Uit de naam, de voornaam en het e-mail adres worden de logingegevens gegenereerd. Alle attributen die gekozen zijn staan in de tabel hieronder. Net zoals bij Job Offer User is er hier gekozen voor het attribuut *User Account* om de login, het paswoord en het e-mailadres op te slaan.

Klasse Resume User			
Name	**Identifier**	**Datatype**	**Options**
Name	name	Text line	Required
Firstname	firstname	Text line	Required
User account	user account	User Account	Required
Fax	fax	Text line	
Telephonenumber	telephonenumber	Text line	

6.7 Job Offer

De klasse *Job Offer* zal voor de advertenties zorgen waarop werkzoekenden kunnen reageren. Deze advertenties zullen niet zo uitgebreid worden zoals de advertenties op de website van Vacature, maar enkele basisgegevens zijn wel nodig om een degelijke advertentie op te bouwen.

Daarvoor hebben we nodig:

- een titel voor een kernachtige beschrijving
- een datum die dient voor de deadline
- een tekst waarin een bedrijf zichzelf kan voorstellen
- een tekst waarin de objectieven van de job komen te staan
- een tekst waarin het profiel komt te staan waar het bedrijf naar zoekt
- de mogelijkheid om een bestand toe te voegen

De attributen die uiteindelijk gekozen zijn, zijn:

Klasse Job Offer			
Name	**Identifier**	**Datatype**	**Options**
Title	title	Text line	Required
Deadline	deadline	Date field	Required
Presentation	presentation	Text field	
Description	description	Text field	
Objectives	objectives	Text field	
Profile	profile	Text field	
File	file	Binary file	

6.8 News

De klasse *News* heeft als doel content objecten te maken voor het nieuwsdeel van de website. Deze nieuwsartikels moeten voornamelijk informatie geven aan de gebruikers. Om deze informatie over te brengen, heb ik de klasse *News* nodig. Deze zal het volgende moeten bevatten:

- de mogelijkheid om een nieuws-item onder de categorie *ACTINET* of *The Global Actinide Community* te zetten
- een titel
- een datum
- de auteur van het artikel
- de bron van het artikel
- een kleine intro
- een body om een tekst in te plaatsen.

De attributen die uiteindelijk zijn gekozen, zijn:

Klasse News			
Name	**Identifier**	**Datatype**	**Options**
Type	type	Selection	Options: • ACTINET • The Global Actinide Community
Title	title	Text line	

Date	date	Date field	
Author	author	Text line	
Source	source	URL	
Abstract	abstract	XML Text field	
Full text	full_text	XML Text field	

6.9 Activity

Deze klasse *Activity* zorgt voor het maken van content objecten in de subcategorieën van het deel General Info. Deze subcategorieën zijn Joint Research Projects en Pooling Facilities.

Omdat Joint Research Project een eenvoudigere versie is van Pooling Facilities, zullen beide gebruik maken van de klasse *Activity*. Ik zal daarom de klasse *Activity* bespreken uit het standpunt van Pooling Facilities.

Bij het maken van de klasse *Activity* moest ik rekening houden met volgende zaken:

- de naam van de faciliteit
- de naam van het instituut
- de contactpersonen en hun gegevens
- een beschrijving van de activiteit
- de categorie waarin deze activiteit zich bevindt
- de startdatum van een activiteit
- het toevoegen van een presentatie van de activiteit

Omdat er meerdere contactpersonen bij een activiteit kunnen voorkomen, krijgt de klasse *Activity* de eigenschap van container mee. Zo kunnen er content objecten gemaakt worden van de klasse *Contactperson Activity* onder elke activiteit. De klasse *Contactperson Activity* wordt later nog besproken (zie 6.10). De uiteindelijke attributen zijn geworden:

Klasse Activity			
Name	**Identifier**	**Datatype**	**Options**
Title	title	Text line	Required
Intro	intro	XML Text field	
Description	description	XML Text field	
Presentation	presentation	Binary file	
Scope	scope	Selection	Options: • Actinides in solution and solid phase • Actinides in the geological environment • Actinide materials under and after irradiation
Starting Date	starting_date	Date field	
Presentation	presentation	Binary file	

Het enige verschil dat er tussen Joint Research Projects en Pooling Facilities zal zijn, zijn de attributen *Starting Date* en *Scope*. Deze attributen zullen niet te zien zijn bij Joint Research Projects.

6.10 Contactperson Activity

De klasse *Contactperson Activity* lijkt zeer sterk op de klasse *Contactperson* die zich bij de ACTINET partners bevindt. Bij contactperson activity had men nodig:

- de naam en de voornaam
- het instituut waar de contactpersoon werkt
- het adres van het instituut
- het telefoon- en faxnummer
- het e-mailadres

Ik heb uiteindelijk gekozen om volgende attributen te maken:

Klasse Contactperson Activity			
Name	**Identifier**	**Datatype**	**Options**
Name	name	Text line	Required
Firstname	firstname	Text line	Required
Institute	institute	Text line	
Street	street	Text line	
Streetnumber	streetnumber	Text line	
Postalcode	postalcode	Text line	
City	city	Text line	
PO Box	po_box	Text line	
Country	country	Country Selection	
Telephonenumber	telephonenumber	Text line	
Fax	fax	Text line	
Mail	mail	Email	

6.11 Presentation

De klasse *Presentation* zorgt ervoor dat er content objecten kunnen gemaakt worden om presentaties in op te slaan. Deze klasse zal tot uiting komen in Presentation Material en in Summer School. In Presentation Material zal deze klasse de hoofdrol spelen. In Summer School wordt deze klasse gebruikt om meerdere presentaties aan één Summer School toe te wijzen. Om een presentatie te maken moest ik volgende informatie kunnen opslaan:

- een titel
- een introductie
- een afbeelding die dient als thumbnail
- de presentatie

Uit deze informatie heb ik de volgende attributen kunnen maken:

Klasse Presentation			
Name	**Identifier**	**Datatype**	**Options**
Title	title	Text line	Required
Introduction	introduction	XML Text field	
Image	image	Image	
Presentation	presentation	Binary file	

6.12 Event

Net zoals bij de klasse *Activity* zal de klasse *Event* voor meerdere soorten content objecten zorgen. De klasse *Event* zal objecten maken voor Events en voor Summer School. Ik heb deze klasse gebruikt voor events en summer school omdat deze twee ook bijna hetzelfde zijn buiten enkele attributen.

Voor Events en Summer School moest ik rekening houden met het volgende:

- het opgeven van het type van event (alleen voor Events)
- een start- en einddatum
- een titel
- een locatie
- door welke organisatie het event of summer school is georganiseerd
- de naam en de voornaam van de organisator
- het adres van de organisatie
- het telefoon- en faxnummer van de organisator
- het e-mailadres van de organisator
- een kleine introductie
- een tekst over het event of summer school
- een link naar een website
- het adres van de locatie waar het event of summer school zal plaatsvinden
- informatie over de accomodatie die er zal zijn tijdens een summer school (alleen voor summer school)

Het uiteindelijke resultaat is geworden:

Klasse Event			
Name	**Identifier**	**Datatype**	**Options**
Type	type	TextLine	
Startdate	startdate	Date Field	
Enddate	enddate	Date Field	
Title	title	TextLine	
Location	location	TextLine	
Organised by	organised_by	TextLine	
Name	name	TextLine	
Firstname	firstname	TextLine	
Street	street	TextLine	

Housenumber	housenumber	TextLine	
Postalcode	postalcode	TextLine	
City	city	TextLine	
Country	country	Country selection	
Telephonenumber	telephonenumber	TextLine	
Fax	fax	TextLine	
Mail	mail	Email	
Introduction	introduction	XML Text field	
Text	text	XML Text field	
URL	url	URL	
Location Street	location_street	TextLine	
Location Number	location_number	TextLine	
Location Postalcode	location_postalcode	TextLine	
Location City	location_city	TextLine	
Location Country	location_country	Country selection	
Accomodation Information	accomodation_information	XML Text field	

6.13 Summer School Register

Er moest voor summer school ook de mogelijkheid zijn om zich te kunnen inschrijven. Dit zal gebeuren met de klasse *Summer School Register*. Ook moet veel informatie worden afgedwongen van de gebruiker. De registraties zullen te bekijken zijn door de verantwoordelijke van de huidige summer school. Deze verantwoordelijke kan zich inloggen op de website met een aangemaakte user account en krijgt dan de lijst met registraties.

Voor deze registratie moet de verantwoordelijke van een summer school te zien krijgen:

- de naam en de voornaam
- het e-mailadres
- het telefoon- en faxnummer
- de status van een persoon. Dit kan zijn: PhD Student, Post-Doc of Permanent position
- de interesses van een persoon
- de organisatie waar de persoon bij werkt
- de stad en het land van die organisatie
- het paspoortnummer of het identiteitskaarnummer van de persoon
- de nationaliteit van de persoon
- de geboortedatum
- welke accomodatie de persoon wenst te hebben
- de datum van aankomst
- de datum van vertrek
- plaats voor opmerkingen en vragen

Het uiteindelijke resultaat is:

Klasse Summer School Registration			
Name	**Identifier**	**Datatype**	**Options**
Name	name	TextLine	Verplicht
First Name	first_name	TextLine	Verplicht
E-mail	e_mail	Email	Verplicht
Telephonenumber	telephonenumber	TextLine	Verplicht
Fax	fax	TextLine	
Status	status	Selection	Verplicht Options: • PhD Student • Post-Doc • Permanent position
Researchfield	researchfield	TextLine	
Organisation	organisation	TextLine	Verplicht
City	city	TextLine	Verplicht
Country	country	Country selection	Verplicht
Passport number	passport_number	TextLine	Verplicht
Nationality	nationality	TextLine	Verplicht
Date of Birth	date_of_birth	DateField	Verplicht
Accomodation Choice	accomodation_choice	Selection	Verplicht Options: • no room • single room • double room
Date of Arrival	date_of_arrival	DateField	Verplicht
Date of Departure	date_of_departure	DateField	Verplicht
Remarks & Questions	remarks_questions		

7 Gebruikte extensies

7.1 Countryselection

Countryselection is een extensie die gemaakt is door de informatici van het Knowledge Management. Deze zorgt ervoor dat u het datatype *Country Selection* kunt gebruiken voor elke content klasse. Dit datatype is een combobox die alle landen van de wereld bevat. Het datatype zal ik in mijn klassen gebruiken telkens er een land moet worden opgegeven. Om Countryselection te gebruiken moet u eerst de tabel *sckcountries* in de MySQL database plaatsen. In de tabel *sckcountries* staat de informatie van alle landen. Elk land heeft een aantal vaste gegevens: id, name, twocharcode, threecharcode en number. De twocharcode en de threecharcode komen van de standaard ISO 3166 die uitgebracht is door ISO (International Organization for Standardization), een organisatie die zorgt voor standaardisatie. Zelf zal ik enkel name gebruiken.

Ik heb aan eZ Countryselection de templateoperator *get_country* toegevoegd. Deze zorgt voor het ophalen van de naam van elk land uit de database. Dit kunt u opvragen in de templates als templateoperator.

```
function namedParameterList()
{
    return array( 'get_country' => array() );
}

function modify( &$tpl, &$operatorName, &$operatorParameters, &$rootNamespace,
&$currentNamespace, &$operatorValue, &$namedParameters )
{
    switch( $operatorName )
    {
        case 'get_country':
        {
            $db = &eZDB::instance();
            $rows = $db->arrayQuery( 'SELECT ThreeCharCode,
                                              Name FROM sckcountries' );
            $operatorValue = $rows ;
        }break;
    }
```

In de functie *namedParameterList* wordt opgegeven welke parameters de templateoperator *get_country* zal krijgen. Er moet geen parameter worden meegegeven om deze templateoperator te doen werken. In de functie *modify* wordt de code voor get_country uitgevoerd. Hier wordt eZDB geïnstancieerd om zo de functie *arrayQuery* te kunnen gebruiken. eZDB is een script dat in eZ publish is ingebouwd. *ArrayQuery* zal de SQL uitvoeren en de resultaten in de variabele *rows* steken. *rows* wordt aan de variabele *operatorValue* toegekend. operatorValue zorgt ervoor dat het resultaat in de template verandert.

7.2 ezsckmultiplexer en workflow Job Offer

Wanneer Job Offer Users en Resume Users zich registreren op de website, worden de registraties opgevangen door een workflow. De registraties worden niet gepubliceerd tot de websiteverantwoordelijke ze goedkeurt. Dit is de taak van de workflow *Job Offer*. Deze

workflow is pre-publish ingesteld. Dit betekent dat hij alles opvangt voor het gepubliceerd wordt. Het publiceren van een content object betekent dat een content object wordt opgenomen in de content node tree. Als de websiteverantwoordelijke toch verkiest om de gebruiker af te wijzen, wordt de user account ook verwijderd uit het systeem. Wanneer dit gebeurt, krijgt de gebruiker een mail dat hij niet is geaccepteerd. Bij goedkeuring wordt er een mail gestuurd met zijn login en paswoord.

Het probleem schuilt echter bij de pre-publish van de workflow. Hij vangt alle content objecten op die worden gepubliceerd en niet enkel Job Offer User en Resume User. Hier komt de SCK Multiplexer aan te pas. Deze zorgt ervoor dat enkel Job Offer User en Resume User worden opgevangen en doorverwezen naar de workflow. Alle andere klassen worden doorgelaten en gepubliceerd.

7.3 scksingleapprove

scksingleapprove is een extensie die bruikbaar is bij het maken van een workflow. Hier krijgt u de optie om een *SCK-CEN Single Approve* event aan te maken. Dit event zorgt ervoor dat er iemand van de users wordt aangesteld om nieuwe content objecten goed of af te keuren. Wanneer er een content object wordt aangemaakt, wordt de websiteverantwoordelijke verwittigd met een mail. U kunt hier ook kiezen of er bij goed- of afkeuring een mail wordt gestuurd naar de gebruiker. Deze extensie is geprogrammeerd door de informatici van het Knowledge Management.

7.4 sckpendingactions

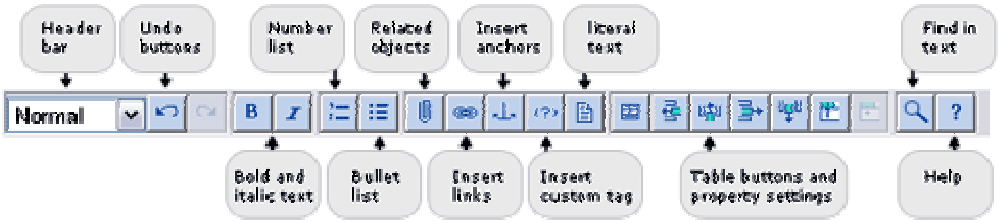
sckpendingactions bevat de cronjob die ervoor zorgt dat de mails worden verstuurd. Deze mails zijn bijvoorbeeld de mails met de logingegevens van Job Offer User en Resume User. De cronjob gaat elke mail die in de wachtrij staat, controleren. Als het een geldige mail is, wordt deze verstuurd. Anders blijft deze in de wachtrij staan. Alle verstuurde mails worden opgeslagen in een tabel die dient als logboek in de database. U kunt voor de mails van sckpendingactions een template schrijven zoals voor eender welke pagina in eZ publish. Het voordeel hiervan is dat u de mails mooi kunt opmaken zodat ze er deftig uitzien naar de gebruikers toe. Deze extensie is geprogrammeerd door de informatici van het Knowledge Management.

7.5 extendedattributefilters

Deze extensie zorgt ervoor dat u kunt filteren op de attributen van elke klasse naar keuze. Dit bestond al in eZ publish maar dan niet zoals de *like* van SQL. Met deze extensie kunt u bijvoorbeeld zoeken naar alle partners waarvan de naam begint met de letter ‘S’. Deze extensie is ontwikkeld door de informatici van het Knowledge Management.

7.6 ezdhtml

ezdhtml is een tool die de websiteverantwoordelijke een handige interface geeft voor XML Text fields. Hierdoor wordt het opmaken van tekst, het plaatsen van URL's, het plaatsen van afbeeldingen en nog veel meer, een stuk eenvoudiger. De XML-editor zal geïntegreerd worden in de website en elk XML Text field verzorgen van deze interface. Het is ook mogelijk om nieuwe XML-tags te voorzien. U maakt eerst de tag aan en daarna een bijhorende template die de tag moet vervangen. Deze is direct te gebruiken door de gebruiker. Hierdoor zijn XML Text fields zeer flexibel in gebruik. De opmaakwerkbalk van de XML-editor zal er als volgt uit zien:



figuur 6.1 ezdhtml opmaakwerkbalk

7.7 CustomRound

De extensie *CustomRound* heb ik zelf geschreven omdat eZ publish geen templateoperator bevat die getallen kan afronden na de komma. Dit is een heel simpele extensie waarmee u kunt bepalen tot op hoeveel getallen na de komma u uw getallen wilt afronden.

```
function namedParameterList()
{
       return array( 'customround' => array(  'number' => array( 'type' =>
'integer',
                                                                  'required' => true,
                                                                  'default' => ''),
                                                'decimal' => array( 'type' => 'integer',
                                                                    'required' => false,
                                                                    'default' => 0 ) ) );
}

function modify( &$tpl, &$operatorName, &$operatorParameters, &$rootNamespace,
                 &$currentNamespace, &$operatorValue, &$namedParameters )
{

    switch ( $operatorName )
    {
        case 'customround':
        {
            $number = $namedParameters['number'];
            $decimal = $namedParameters['decimal'];
            $integer = round( $number , $decimal );
            $operatorValue = $integer ;
        } break;
    }
}
```

Er is de keuze tussen 1 of 2 parameters die u meegeeft met de templateoperator. In de functie *namedParameterList* wordt er beschreven welke parameters allemaal mogelijk zijn. De 1ste parameter is verplicht en van het type integer. Deze parameter zal het getal doorgeven dat moet worden afgerond. De 2de parameter is optioneel, van het type integer en zal standaard op 0 worden gezet. Deze parameter zal beslissen tot op hoeveel getallen na de komma het getal moet worden afgerond.

Nu volgt er een voorbeeld hoe de templateoperator zal worden gebruikt in een template.

```
{customround( $getal , 2 )}
```

7.8 Sort

Deze extensie heb ik geschreven om een templateoperator te maken die voor de sortering van arrays zorgt. U hebt de keuze om een array op een gewone manier te sorteren of omgekeerd te sorteren.

```
function namedParameterList()
{
    return array( 'sortarray' => array( 'sortarray' => array( 'type' => array(),
                                                              'required' => true,
                                                              'default' => '' )),
                  'reversesortarray' => array(
                                  'reversesortarray' => array( 'type' => array(),
                                  'required' => true,
                                  'default' => '' ) ) );
}

function modify( &$tpl, &$operatorName, &$operatorParameters, &$rootNamespace,
                 &$currentNamespace, &$operatorValue, &$namedParameters )
{
    switch ( $operatorName )
    {
        case 'sortarray':
        {
            $array = $namedParameters['sortarray'];
            sort( $array );
            $operatorValue = $array ;
        } break;
        case 'reversesortarray':
        {
            $array = $namedParameters['reversesortarray'];
            rsort( $array );
            $operatorValue = $array ;
        } break;
    }
}
```

In de functie *namedParameterList* worden twee templateoperators beschreven. Dit zijn *sortarray* en *reversesortarray*. De templateoperator *sortarray* krijgt de parameter *sortarray* mee. Deze parameter is van het type array en is verplicht in te vullen. De templateoperator *reversesortarray* krijgt de parameter *reversesortarray* mee van het type array. Ook bij deze templateoperator is de parameter verplicht. In de functie *modify* wordt gekeken welke templateoperator wordt aanroepen. Bij allebei de templateoperators wordt de array gesorteerd via *sort* of omgekeerd gesorteerd via *rsort.* U kunt de templateoperators in de templates aanspreken met volgende code:

```
{sortarray( $array )}

{reversesortarray( $array )}
```

7.9 Objrel

Deze extensie is gemaakt door de informatici van het Knowledge Management. Door het implementeren van deze extensie, kreeg ik datatype *Detailed Object Relation*
erbij. Tijdens het maken van de klasse *Job Offer User* zorgde dit datatype ervoor dat ik een combobox kreeg met alle ACTINET partners die op de website staan.

7.10 Usersignup

Deze extensie heb ik speciaal geschreven voor het registreren van Job Offer Users en Resume Users. Deze extensie maakt de registratie pagina's van Job Offer User en Resume User. Bij het submitten van de informatie wordt er allereerst een controle gedaan op de input zodat al de nodige en juiste informatie wordt opgeslagen in de database. De controle wordt opgeroepen via volgende code:

```
//check the name, first name, fax, phonenumber, company, partnercompany
$isvalidname = checkVariable( 'name',
                              'username',
                              true,
                              'emptyFieldName' );
$isvalidfirstname = checkVariable( 'firstname',
                                   'userfirstname',
                                   true,
                                   'emptyFieldFirstName' );

checkVariable( 'fax', 'userfax' );
checkVariable( 'telephonenumber', 'userphone' );
$isvalidcompany = checkCompany( 'partnercompany', 'userpartnercompany' , 'company',
'usercompany' );
```

Deze code roept de functie *checkVariable* op. Er moeten 2 parameters verplicht worden opgegeven en dat is de naam van de postvariabele en de naam van de templatevariabele. De postvariabele wordt door de template gezonden wanneer er op de submitknop wordt geklikt. De templatevariabele dient voor de foutboodschappen die gebruiker te zien krijgt wanneer er een foute input wordt gegeven. Een optionele parameters is de parameter *required*. Wanneer u informatie wilt afdwingen, moet u deze parameter mee doorgeven aan de functie. Alle foutboodschappen worden in de template geplaatst en opgevraagd. Deze foutboodschappen hebben een bepaalde naam waarmee u ze kunt oproepen. Deze naam wordt meegegeven met de laatste parameter.

De functie *checkVariable* gaat als volgende:

```
function checkVariable( $getPostVariableName, $templatevariablename, $required =
false, $errormessage = '' )
{
    /* params =
        - the name of the postvariable
        - the name of the variable you use in the template
        - boolean if the field is required
        - the variable to give the errormessage in the template
    */
```

```
    $tpl = &templateInit();
    $http =& eZHTTPTool::instance();
    $valid=false;

    //check if the variable has a postvariable
    if( $http->hasPostVariable( $getPostVariableName ) )
    {
        // get the variable
        $varpost = $http->postVariable( $getPostVariableName );

        // check if the variable has to be filled in
        if( $required )
        {
            //check if it's empty
            if( $varpost == '' )
            {
                $tpl->setVariable( $errormessage, $errormessage );
                end;
            }
            else
            {
                $valid = true;
            }
        }

        $tpl->setVariable( $templatevariablename, $varpost );
        if( $valid )
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}
```

Eerst wordt er gekeken of er wel een postvariabele bestaat. Als deze bestaat, wordt de waarde ervan in de variabele *varpost* gestoken. Wanneer u de optie required hebt mee doorgegeven aan de functie, dan wordt er gekeken of de postvariabele een lege string bevat. Als dit zo blijkt te zijn, wordt de foutboodschap in de template geactiveerd en stopt het verdere verloop van de functie *checkVariable*. Wanneer de waarde toch een string bevat, wordt deze valid bevonden en wordt deze ook terug naar de template gestuurd.

Bij Job Offers wordt er dan nog eens gecontroleerd of de gegevens van het bedrijf voldoen. U kunt kiezen tussen een combobox met alle ACTINET-partners in en het invullen van uw bedrijfsgegevens.

Voordat alle gegevens van het bedrijf wordt gecontroleerd, wordt er gecontroleerd of de naam van het bedrijf is ingevuld of er een partner is geselecteerd. Dit gebeurt via de functie *checkCompany*. checkCompany gaat als volgende:

```
function checkCompany( $postpartner ,  $templatepartner, $postcompany,
$templatecompany)
{
    $tpl = &templateInit();
    $http =& eZHTTPTool::instance();
    $valid=false;

    if( $http->hasPostVariable( $postpartner ) )
    {
        if( $http->hasPostVariable( $postcompany ) )
        {
```

```
            $varpartner = $http->postVariable( $postpartner );
            $varcompany = $http->postVariable( $postcompany );
            switch( true )
            {
                case !$varpartner && !$varcompany :
                {
                    $valid = false;
                    $tpl->setVariable( 'choosecompanyField' , 'choosecompanyField'
);
                }break;
                case $varpartner && $varcompany :
                {
                    $valid = false;
                    $tpl->setVariable( 'onecompanyField' , 'onecompanyField' );
                }break;
                default:
                {
                    $valid=true;
                }
            }
        }
        $tpl->setVariable( $templatepartner, $varpartner );
        $tpl->setVariable( $templatecompany, $varcompany );
        return $valid;
    }
}
```

Bij deze functie wordt de post- en templatevariabelen van de combobox van de ACTINET partners en van het inputveld van de bedrijfsnaam meegegeven als parameters. Hier wordt er gecontroleerd of er maar één van de twee is ingevuld. Wanneer er geen enkele is ingevuld, krijgt u een foutmelding dat er moet worden gekozen tussen de twee. Er wordt ook de boolean *false* teruggestuurd. Wanneer ze allebei zijn ingevuld ontstaat er een foutmelding dat er maar ééntje mag worden gekozen en wordt de boolean *false* teruggestuurd. Anders wordt er de boolean *true* terug gestuurd.

Om het adres te controleren wanneer er toch voor het invullen van de bedrijfsgegevens is gekozen, wordt volgende code uitgevoerd:

```
$partnerchecked = false;

if( $http->hasPostVariable( 'partnercompany' ) )
{
    $var = $http->postVariable( 'partnercompany' );
    if( $var != '' )
    {
        $partnerchecked = true;
    }
}

if( !$partnerchecked )
{
    if( $http->hasPostVariable( 'company' ) )
    {
        $var = $http->postVariable( 'company' );
        if( $var != '')
        {
            $isvalidstreet = checkVariable( 'street', 'userstreet' );
            $isvalidnumber = checkVariable( 'number', 'usernumber' );
            $isvalidpobox = checkVariable( 'pobox', 'userpobox' );
            $isvalidpostalcode = checkVariable( 'postalcode',
                                                'userpostalcode',
                                                true,
                                                'emptyFieldPostalcode' );
            $isvalidcity = checkVariable( 'city',
```

```
                                        'usercity',
                                        true,
                                        'emptyFieldCity' );
            $isvalidcountry = checkVariable( 'country',
                                             'usercountry',
                                             true,
                                             'emptyFieldCountry' );

            if( $http->hasPostVariable( 'street' ) )
            {
                $var = $http->postVariable( 'street' );
                if( $var != '' )
                {
                    $isvalidstreet = true;
                }
            }
            if( $http->hasPostVariable( 'number' ) )
            {
                $var = $http->postVariable( 'number' );
                if( $var != '' )
                {
                    $isvalidnumber = true;
                }
            }
            if( $http->hasPostVariable( 'pobox' ) )
            {
                $var = $http->postVariable( 'pobox' );
                if( $var != '' )
                {
                    $isvalidpobox = true;
                }
            }

            if( $isvalidstreet && $isvalidnumber )
            {
                $isvalidstreetnumber = true;
            }

            if( $isvalidstreetnumber or $isvalidpobox )
            {
                $errorStreetNumberPO = true;
            }
            else
            {
                $tpl->setVariable( 'emptyFieldAddress' , 'emptyFieldAddress' );
            }

            if( $errorStreetNumberPO && $isvalidpostalcode && $isvalidcity &&
$isvalidcountry )
            {
                $isvalidaddress = true;
            }
        }
    }
}

if( $isvalidcompany && $partnerchecked )
{
    $varcompanyisvalid = true;
}
if( $isvalidaddress && !$partnerchecked )
{
    $varcompanyisvalid = true;
}
```

Eerst wordt er gekeken of er een ACTINET partner is geselecteerd in de combobox. Wanneer er eentje is geselecteerd wordt de variabele *partnerchecked* op true gezet. Als in de combobox met partners niets is geselecteerd, wordt het adres helemaal gecontroleerd. Wanneer het adres fouten bevat, wordt de gebruiker hiervan op de hoogte gesteld met een foutboodschap. Wanneer het adres correct is, wordt de boolean *isvalidaddress* op true gezet. Wanneer alles correct is ingevuld van de bedrijfsgegevens, wordt de variabele *varcompanyisvalid* op true gezet.

Daarna wordt er gekeken of de naam en de voornaam valid zijn. Als dit een positief resultaat geeft wordt de variabele *varuservalid* op true gezet.

```
if( $isvalidname && $isvalidfirstname )
{
    $varuservalid=true;
}
```

Er wordt ook gecontroleerd of er een gebruiker bestaat met hetzelfde e-mailadres.

```
$mailcheck = checkMail( 'mail' );
if( $mailcheck != '' )
{
    $tpl->setVariable( $mailcheck , $errormessages[$mailcheck] );
}
else
{
    $isvalidmail = true;
    $userFieldEmail = $http->postVariable( 'mail' );
    $tpl->setVariable( 'usermail', $userFieldEmail );
}
```

Eerst wordt er de functie *checkMail* aangesproken. Wanneer de mail al bestaat, wordt er een foutboodschap doorgegeven aan de template. Wanneer de mail uniek is, wordt de variabele *isvalidmail* op true gezet en wordt de mail terug naar de template gestuurd. Deze variabele wordt teruggestuurd naar template omdat de gebruiker dit tekstveld niet meer zou moeten invullen wanneer er toch ergens nog een foute input is. Ik ga nu even kort de code tonen van *checkMail* en deze een beetje uitleggen.

```
/*!
Check the mail
*/
function checkMail( $postVariableName )
{
    $tpl = &templateInit();
    $http =& eZHTTPTool::instance();
    if ( $http->hasPostVariable( $postVariableName ) )
    {
    $mail = $http->postVariable( $postVariableName );
    $validatemail = eZMail::validate( $mail );
        if( $mail != '' )
        {
            if( $validatemail )
            {
                $user = &eZUser::fetchByEmail( $mail );
                $is_object = is_object( $user );
                if( !$is_object )
                {
                    return false;
                }
                else
                {
```

```
                    return 'notuniqueFieldMail';
                }
            }
            else
            {
                return 'wrongFieldMail';
            }
        }
        else
        {
            return 'emptyFieldMail';
        }
    }
    return false;
}
```

In deze code wordt de functie *validate* aanroepen van eZMail. eZMail is een ingebouwde functionaliteit van eZ publish. Eerst wordt er gekeken of er een e-mailadres is ingevuld. Als dit niet zo is, wordt er een string met *emptyFieldMail* terug gestuurd. Daarna gaat men kijken of de mail juist is opgebouwd. Als het e-mailadres niet juist opgebouwd is, wordt er de string *wrongFieldMail* terug gestuurd. Wanneer het e-mailadres wel juist is opgebouwd, wordt er gekeken of het e-mailadres zich al in de database bevindt via de functie *fetchByEmail* van eZUser. Wanneer er een object wordt teruggegeven, betekent dit dat het e-mailadres al bestaat en wordt er de variabele *notuniqueFieldMail* teruggestuurd. Anders wordt de boolean false teruggestuurd.

Als volgende zal ik uitleggen hoe de login en het paswoord worden gegenereerd, het content object wordt aangemaakt en opgeslagen en de mail wordt gemaakt en verstuurd.

De login en het paswoord worden gegenereerd door volgende code:

```
/*!
    create the loginname
*/
function createLogin( $inputFirstName, $inputName )
{
    $firstname = strtolower( $inputFirstName );
    $firstname = preg_replace( array( "/[^a-z]/" ), array( '' ), $firstname );
    $firstcharfirstname = substr( $firstname, 0, 1 );

    $name = strtolower( $inputName );
    $name = preg_replace( array( "/[^a-z]/" ), array( '' ), $name );

    $login = strtolower( substr( $firstcharfirstname . $name, 0, 8 ) );

    //check if generated login already exists
    $number = 1;
    $loginBase = $login;
    $existingUser = eZUser::fetchByName( $login );

    while( $existingUser != null )
    {
        $login = $loginBase . $number;
        $number++;
        $existingUser = &eZUser::fetchByName( $login );
    }
    return $login;
}

/*!
    create a password and useraccount
*/
function createUser( $userId, $login, $userFieldEmail )
```

```
{
    $ini = &eZINI::instance();

    $user = &eZUser::create( $userId );

    //generate password
    $passwordLength = $ini->variable( 'UserSettings', 'GeneratePasswordLength' );
    $password = &eZUser::createPassword( $passwordLength , $login . microtime() );

    $passwordConfirm = $password;

    $user->setInformation( $userId, $login, strtolower( $userFieldEmail ),
$password, $passwordConfirm );
    $isEnabled = 1;
    $userSetting = &eZUserSetting::create( $userId, $isEnabled );
    $userSetting->store();

    $user->store();
    return array( 'user' => $user, 'password' => $password );
}
```

Voor het creëren van de login, wordt de voornaam en de naam van de persoon gebruikt. Ten eerste wordt de voornaam omgezet in kleine letters en worden alle speciale karakters uit de voornaam gefilterd. Daarna wordt enkel de eerste letter van de voornaam genomen. Ten tweede wordt ook de naam naar kleine letters omgezet en alle speciale karakters eruit gefilterd. Ten derde wordt de eerste letter van de voornaam aan de achternaam geplakt. Hier worden de eerste acht letters van genomen. Als laatste wordt er gecontroleerd of deze login al bestaat in de database. Als dit zo is dan wordt er een getalletje achter geplaatst. Als de login met dit getalletje al bestaat wordt het getal met één verhoogd tot de login uniek is.

Na het aanmaken van de login wordt er een user account en een paswoord aangemaakt. Er moet eerst een user account worden aangemaakt omdat deze een functie bevat om paswoorden te maken. Het paswoord wordt dan gegenereerd uit de login. Uiteindelijk wordt de login, het paswoord en het e-mailadres in de user account gestoken. Deze user account zal later nog aan het content object worden toegevoegd.

Volgende code zorgt voor het aanmaken en opslaan van het content object in de database:

```
$contentObject = &$class->instantiate( $userCreatorId, $sectionID );

$nodeAssignment = &eZNodeAssignment::create( array(
    'contentobject_id' => $contentObject->attribute( 'id' ),
    'contentobject_version' => $contentObject->attribute( 'current_version' ),
    'parent_node' => $parentContentObjectTreeNode->attribute( 'node_id' ),
    'is_main' => 1 ) );
$nodeAssignment->store();

$contentObject->setAttribute( 'name', $login );
$contentObject->store();

$version = &$contentObject->version( $contentObject->attribute( 'current_version' )
);
$version->setAttribute( 'modified', mktime( ) );
$version->setAttribute( 'created', mktime( ) );
$version->setAttribute( 'status', EZ_VERSION_STATUS_DRAFT );
$version->store();

$version = &$contentObject->attribute( 'current' );
$contentObjectAttributes = &$version->dataMap();
//make the class objcollection
$partner = &new SckObjCollection();
$partner->addrow( $userFieldPartnerCompany );
```

```
//content object attributes
$contentObjectAttributes['name']->setAttribute( 'data_text', $userFieldName );
$contentObjectAttributes['name']->store();

$contentObjectAttributes['firstname']->setAttribute( 'data_text',
$userFieldFirstName );
$contentObjectAttributes['firstname']->store();

//store the right company data
if( $partnerchecked )
{
  $contentObjectAttributes['actinet_partner']->setAttribute( 'data_text', $partner-
>xmlString() );
  $contentObjectAttributes['actinet_partner']->store();
}
else
{
  //addressdata company
  $contentObjectAttributes['company']->setAttribute( 'data_text' ,
$userFieldCompany );
  $contentObjectAttributes['company']->store();

  $contentObjectAttributes['street']->setAttribute( 'data_text', $userFieldStreet
);
  $contentObjectAttributes['street']->store();

  $contentObjectAttributes['number']->setAttribute( 'data_text', $userFieldNumber
);
  $contentObjectAttributes['number']->store();

  $contentObjectAttributes['po_box']->setAttribute( 'data_text', $userFieldPOBox );
  $contentObjectAttributes['po_box']->store();

  $contentObjectAttributes['postalcode']->setAttribute( 'data_text',
$userFieldPostalcode );
  $contentObjectAttributes['postalcode']->store();

  $contentObjectAttributes['city']->setAttribute( 'data_text', $userFieldCity );
  $contentObjectAttributes['city']->store();

  $contentObjectAttributes['country']->setAttribute( 'data_text', $userFieldCountry
);
  $contentObjectAttributes['country']->store();
}

$contentObjectAttributes['user_account']->setContent( $user );
$contentObjectAttributes['user_account']->store();

$contentObjectAttributes['fax']->setAttribute( 'data_text', $userFieldFax );
$contentObjectAttributes['fax']->store();

$contentObjectAttributes['telephonenumber']->setAttribute( 'data_text',
$userFieldPhone );
$contentObjectAttributes['telephonenumber']->store();

$contentObject->store();

$operationResult = eZOperationHandler::execute( 'content', 'publish',
                array
                (
                    'object_id' => $contentObject->attribute( 'id' ),
                    'version' => $contentObject->attribute( 'current_version' )
                ) );

//send a mail to the user with his login and his password
//prepare parameters for notification
```

```
sendUserMail( $user, 'pending_notifications_job_offer', 'approve', $login,
$password );
```

Eerst zorgt deze code ervoor dat er een node assignment wordt aangemaakt voor het content object. Door de node assignment zal het content object worden opgenomen in de Content Node Tree. Daarna wordt er een versie van het content object aangemaakt. Na het maken van de versie, worden alle content object attributes meegegeven. Nu alle gegevens voor het content object goed zijn ingevuld, wordt dit content object opgeslagen door de functie *execute* van het ingebouwde script *eZOperationHandler* van eZ publish. Na het publiceren van het content object wordt er een mail gestuurd met de nodige gegevens in. Het versturen van de mail gebeurt door volgende code:

```
/*!
    send the mail to the user
*/
function sendUserMail( $user, $templatename, $status, $login, $password )
{
    //send a mail to the user with his login and his password
    //prepare parameters for notification
    $ini = &eZINI::instance();

    $emailSender = $ini->variable( 'MailSettings', 'EmailSender' );
    if ( !$emailSender )
    {
        $emailSender = $ini->variable( 'MailSettings', 'AdminEmail' );
    }

    $receiver = $user->attribute( 'email' );

    $notify_params = array  ( 'templatename' => $templatename,
                              'status' => $status,
                              'author' => $receiver,
                              'approver' => $emailSender,
                              'login' => $login,
                              'password' => $password );

    $row = array
          (
            'action_name' => 'notify',
            'param_text' => serialize( $notify_params )
          );

    ext_class( 'sckpendingactions', 'SckPendingAction' );
    $action = &new SckPendingAction( $row );
    $action->store();
}
```

Eerst worden de ini-settings opgehaald zodat het systeem weet door wie de mail wordt gestuurd. Daarna wordt de mail opgemaakt en in *notify_params* gestoken. In *notify_params* staat welke template moet worden gebruikt voor de opmaak van de mail, de status van de mail die in dit geval *approve* is, het e-mailadres van de ontvanger, het e-mailadres van de verzender, de login en het paswoord. Wanneer dit is gebeurd, wordt de mail opgeslagen in de database en wordt achteraf door de cronjob sckpendingactions verstuurd.

8 Webpagina's

De onderverdeling van de publieke website kunt u vinden in de bijlagen (bijlage 1). In dit deel zal ik de belangrijkste onderdelen van de website toelichten.

8.1 Standaard Webpagina's

De standaard webpagina's zijn content objecten van de klasse *Webpage*. De standaardwebpagina's op de website zijn de volgende:

- homepage
- contact
- about us
- background, hoort bij general info
- copyright
- terms & conditions

Bij deze webpagina's krijgt de websiteverantwoordelijke de mogelijkheid om de inhoud te veranderen naar believen. De inhoud kan tekst zijn met opmaak die door de XML tags wordt gemaakt. De websiteverantwoordelijke zal geen rekening moeten houden met deze tags omdat de extensie *ezdhtml* een handige interface voorziet. Meer informatie over ezdhtml kunt u in dit eindwerk onder het deel *extensies* lezen (zie 7.6).

8.2 Editor

Tijdens mijn stage is het de bedoeling dat ik de structuur van de website maak en ervoor zorg dat alle informatie juist op het scherm komt. Maar inhoudelijk wordt alles overgelaten aan de websiteverantwoordelijke. Deze persoon moest voorzien worden van een user account en enkele functies op de website zodat hij de website helemaal kan aankleden met inhoud.

De editor is een compleet aparte webpagina die buiten alle andere webpagina's staat. U kunt de editor in een webpagina implementeren door de include functie en een aantal parameters op te geven.

De volgende code toont de parameters en de include:

```
{let    varclassname=$node.object.class_name
        varclassid=$node.object.contentclass_id
        varclassfakename='valse klassenaam'
        varclasscreate=true
        varclassedit=true
        varclassremove=true
        vardraft=true
        vardisplayinline=true
}
{include uri="design/actinet/templates/editor.tpl"}
{/let}
```

Ik leg nu kort in het volgende tabelletje uit waarvoor elke variabele dient.

varclassname	Deze variabele bevat de naam van de content klasse die gebruikt moet worden.
varclassid	Deze variabele bevat het id van de content klasse die gebruikt moet worden.
varclassfakename	Soms moet er op een knop een andere naam staan dan de klasse naam, daarom kunt u een valse naam meegeven om op de knop te plaatsen.
varclasscreate	Deze variabele zorgt ervoor dat u een content object kunt creëren van de gekozen content klasse.
varclassedit	Deze variabele zorgt ervoor dat u een content object kunt bewerken van de gekozen content klasse.
varclassremove	Deze variabele zorgt ervoor dat u een content object kunt verwijderen van de gekozen content klasse.
vardraft	Deze variabele zorgt ervoor dat de websiteverantwoordelijke al zijn voorlopige content objecten kan bekijken. Dit is een functie van eZ publish.
vardisplayinline	Deze variabele zorgt ervoor dat alle knoppen op 1 lijn worden geplaatst.

Deze variabelen worden allemaal gebruikt in de editor code die via de include regel in de webpagina wordt toegevoegd.

De editor code ziet er als volgt uit:

```
{* Check if the user is an editor *}

{* Check if the user may create, edit, remove *}
{section show=and(  $node.object.can_create_class_list,
                    $node.object.can_edit,
                    $node.object.can_remove ) }

<div class="editor_global_div">

    {* beginning of create *}

    {section show=eq( true , $varclasscreate )}
        {section show=count( $node.object.can_create_class_list )}
            {section show=eq( true , $vardisplayinline)}
                <form class="inlineform" method="post"
action={"/content/action"|ezurl}>
            {section-else}
                <form method="post" action={"/content/action"|ezurl}>
            {/section}
                <input type="hidden" name="NodeID" value="{$node.node_id}" />
                <input type="hidden" name="ClassID" value="{$varclassid}"/>

                {section show=is_set($varfakeclassname)}
                    {section show=$varfakeclassname}
                        <input class="buttonlayout" type="submit" name="NewButton"
value="Create new {$varfakeclassname}" />
                    {section-else}
                        <input class="buttonlayout" type="submit" name="NewButton"
value="Create new {$varclassname}" />
                    {/section}
                {section-else}
                    <input class="buttonlayout" type="submit" name="NewButton"
value="Create new {$varclassname}" />
                {/section}
            </form>
        {/section}
    {/section}
```

```
    {* end of create *}

    {* beginning of edit *}
    {section show=eq( true , $varclassedit )}
        {section show=$node.object.can_edit}
            {section show=eq( true , $vardisplayinline)}
                <form class="inlineform" method="post"
action={concat("/content/edit/", $node.object.id)|ezurl} >
            {section-else}
                <form method="post" action={concat("/content/edit/",
$node.object.id)|ezurl} >
            {/section}
                {section show=is_set($varfakeclassname)}
                    {section show=$varfakeclassname}
                        <input type="submit" class="buttonlayout" value="Edit this
{$varfakeclassname}"/>
                    {section-else}
                        <input type="submit" class="buttonlayout" value="Edit this
{$varclassname}"/>
                    {/section}
                {section-else}
                    <input type="submit" class="buttonlayout" value="Edit this
{$varclassname}"/>
                {/section}

             </form>
        {/section}
    {/section}
    {* end of the edit*}

    {* beginning of remove *}

    {section show=eq( true , $varclassremove )}
        {section show=$node.object.can_remove}
            {section show=eq( true , $vardisplayinline)}
                <form class="inlineform" method="post"
action={"/content/action"|ezurl}>
            {section-else}
                <form method="post" action={"/content/action"|ezurl}>
            {/section}
                <input type="hidden" name="ContentNodeID"
value="{$node.object.main_node_id}" />
                <input type="hidden" name="ContentObjectID"
value="{$node.object.id}" />
                <input type="submit" class="buttonlayout" name="ActionRemove"
value="Remove this {$varclassname}" />
            </form>
        {/section}
    {/section}
    {* end of the remove *}

    {* begin button draft *}
        {section show=eq( true , $vardraft )}
            {section show=eq( true , $vardisplayinline)}
                <form class="inlineform" method="post"
action={"/content/draft"|ezurl}>
            {section-else}
                <form method="post" action={"/content/draft"|ezurl}>
            {/section}
                <input type="submit" class="buttonlayout" name="Draft" value="My
Drafts"/>
            </form>
        {/section}
    {* end button draft *}
</div>
{/section}
```

figuur 8.1 Editor

Eerst wordt er gecontroleerd of de huidige gebruiker wel bevoegd is om deze knoppen te zien. Alleen de websiteverantwoordelijke en de administrator krijgen deze knoppen te zien (figuur 8.1).

De eerste knop die wordt geplaatst, is de knop *create*. Hier wordt nog eens gecontroleerd of de gebruiker bevoegd is om content objecten te creëren. Er wordt ook een controle gedaan wanneer er een valse naam wordt opgegeven. Als er een valse naam wordt opgegeven, dan wordt deze valse naam gebruikt in plaats van de naam van de gebruikte klasse. Deze knop doet een submit naar de content module met het id van de huidige node en het id van de klasse waarvan een content object moet worden gemaakt.

De tweede knop is een knop om content objecten te bewerken. Deze controleert ook of de websiteverantwoordelijke content objecten mag bewerken. Net zoals bij create wordt er een controle gedaan of er een valse naam is die kan gebruikt worden. Hier wordt beroep gedaan op de edit functie van de content module. De edit zal uitgevoerd worden op de huidige node.

De derde knop is diegene die een content object kan verwijderen. Er wordt net zoals bij de vorige twee knoppen gecontroleerd of de websiteverantwoordelijke hier rechten voor heeft. Als hij die rechten heeft, worden er twee variabelen via een hidden field meegegeven. Deze variabelen zijn het huidige node id en het huidige object id. De submit zal net zoals bij create de content module aanspreken.

Dan is er de laatste knop die naar de drafts van de websiteverantwoordelijke verwijst. Dit is een functie van de content module van eZ publish en dient enkel opgeroepen te worden. Hier kan de websiteverantwoordelijke al zijn voorlopige versies in bewaren.

8.3 Navigatie

Een website is geen website als er geen efficiënt middel bestaat om tussen de webpagina's te surfen. Daarom heb ik een aantal navigatiesystemen aan de website toegevoegd. Deze zijn het linkermenu, het topmenu en het navigatiepad.

8.3.1 Linkermenu

figuur 8.2 Het linkermenu

Het linkermenu kunt u op figuur 8.2 zien. Het linkermenu is opgebouwd uit 2 niveau's. Het eerste niveau zijn de hoofdmappen en zijn altijd zichtbaar. Het tweede niveau, de submappen, worden pas zichtbaar wanneer er op de hoofdmap is geklikt. Dus het menu past zich altijd aan waar u zich op de website bevindt. De map waar u zich in bevindt krijgt ook een donker paarse kleur.

Het linkermenu wordt door volgende code gegenereerd:

```
<div class="menutable">
<ul>
{let menuchildren=fetch( content, list, hash(
                                        parent_node_id, 59,
                                        class_filter_type, include,
                                        class_filter_array, array( 'folder' ) ) )}
    {section var=menu loop=$menuchildren}
        {section show=$module_result.path.2.node_id|eq( $menu.node_id )}
            {* the link of the selected folder only can view his children*}
            <li>
                {section show=ne( $module_result.path.3.node_id , '' )}
                    <div class="menu_level_0">
                        <a href={$menu.url_alias|ezurl}>{$menu.name}</a>
                    </div>
                {section-else}
                    <div class="menu_level_0_selected">
                        <a href={$menu.url_alias|ezurl}>{$menu.name}</a>
                    </div>
                {/section}
                {* all the children of the selected folder *}
                {let children=fetch( content, list, hash( parent_node_id,
$menu.node_id,
                                                          class_filter_type,
include,
                                                          class_filter_array,
array( 'folder','webpage' ),
                                                          sort_by,
$menu.sort_array ) )}
                    {section var=menulevel1 loop=$children}
                        {section show=ne( $menulevel1.node_id , 262 )}
                            {section show=$module_result.path.3.node_id|eq(
$menulevel1.node_id )}
                                <div class="menu_level_1_selected">
                                    <a
href={$menulevel1.url_alias|ezurl}>{$menulevel1.name}</a>
                                </div>
                            {section-else}
                                <div class="menu_level_1">
                                    <a
href={$menulevel1.url_alias|ezurl}>{$menulevel1.name}</a>
```

```
                                </div>
                            {/section}
                        {/section}
                    {/section}
                {/let}
             </li>
        {section-else}
            {*all other links that are not selected of level 0*}
            <li>
                <div class="menu_level_0"><a
href={$menu.url_alias|ezurl}>{$menu.name}</a></div>
            </li>
        {/section}
    {/section}
{/let}
</ul>
</div>
```

Allereerst worden alle childnodes van het type *folder* opgehaald uit de hoofdmap van de website via een fetch. Deze hoofdmap heeft als id het getal 59. Daarna wordt het resultaat dat uit de fetch komt in een loop geplaatst. Er wordt gekeken of de webpagina waar u zich in bevindt gelijk is aan de link die u gaat plaatsen. Als dit zo is, dan wordt er een link geplaatst met een paarse kleur, anders wordt er een gewone link met een blauwe kleur geplaatst. Daarna worden de childnodes van de geselecteerde map opgehaald. Deze childnodes kunnen van het type *Folder* en *Webpage* zijn. Wanneer er een childnode wordt geselecteerd, wordt deze childnode paars en verliest de parent node zijn paarse kleur. Natuurlijk zijn er altijd een paar nodes die speciaal moeten worden behandeld in een menu en dit is hier de node met node id 262 of de pagina met de inschrijvingen van Summer School.

8.3.2 Topmenu

figuur 8.3 Het topmenu

Het topmenu is iets eenvoudiger. Dit is enkel het eerste niveau van het linkermenu maar dan op een heel andere wijze op het scherm gebracht.

```
<div id="topmenuglobaldiv">

    {section show=$module_result.path.2.node_id|eq( 195 )}
        <div class="topmenudivselectedfirst"><a href={"/"|ezurl}>Home</a></div>
    {section-else}
        <div class="topmenudivfirst"><a href={"/"|ezurl}>Home</a></div>
    {/section}
    {let children=fetch(content, list, hash( parent_node_id, 59,
                                                class_filter_type, include,
                                                class_filter_array, array(
'folder' ) ) ) }

        {section var=topmenu loop=$children}
            {section show=$module_result.path.2.node_id|eq( $topmenu.node_id )}

                <div class="topmenudivselected"><a
href={$topmenu.url_alias|ezurl}>{$topmenu.name}</a></div>

            {section-else}

                <div class="topmenudiv"><a
href={$topmenu.url_alias|ezurl}>{$topmenu.name}</a></div>
```

```
            {/section}
        {/section}
    {/let}
</div>
<div class="cleardivclean"></div>
```

Deze code gaat gewoon net zoals het linker menu alle folders uit de hoofdfolder halen. Er wordt wel eerst een knop *Home* geplaatst zodat de gebruiker op een eenvoudige manier de homepage kan terugvinden. Wanneer de gebruiker zich in een bepaalde hoofdmap bevindt, zal deze knop paars worden.

8.3.3 Navigatiepad

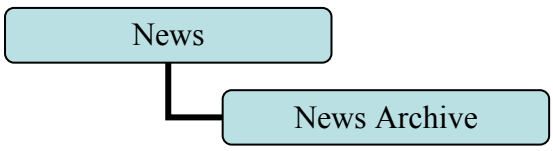
figuur 8.4 Het navigatiepad

Wanneer een gebruiker zich te ver begeeft in de website kan hij wel eens door de bomen het bos niet meer zien. Daarom is er het handige navigatiepad zodat hij altijd kan volgen waar hij zich bevindt op de website. Dit pad wordt gemaakt door volgende code:

```
{section show=ne( $module_result.path.2 , '')}
    <a href="/">Home</a>
    {section var=pathitem loop=$module_result.path offset=2}
        {section show=eq( $module_result.path.2.node_id , 195 )|not()}
            <span style="color: black"> / </span>
            {section show=$pathitem.url}
                <a href={cond( is_set($pathitem.url_alias), $pathitem.url_alias,
$pathitem.url)|ezurl}>{$pathitem.text|shorten(18)|wash}</a>
            {section-else}
                {$pathitem.text|shorten(18)|wash}
            {/section}
        {/section}
    {/section}
{/section}
```

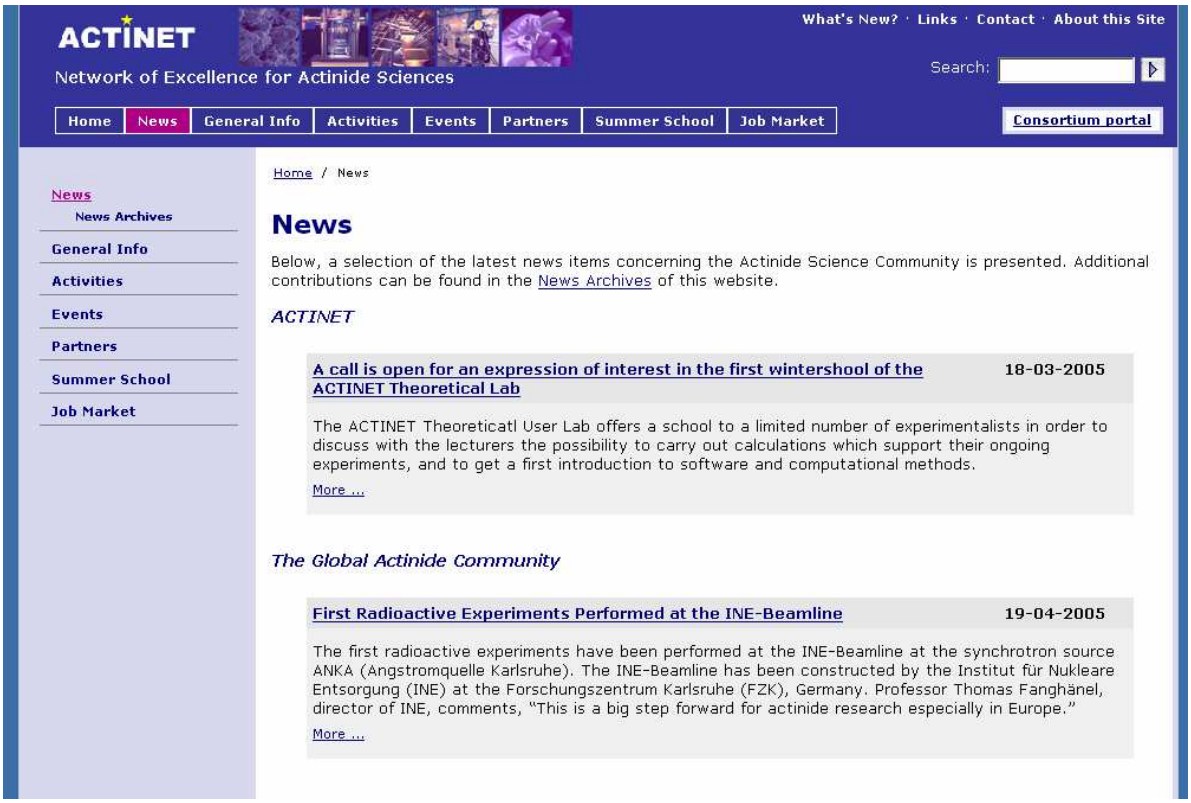
Eerst wordt er gekeken of er wel een pad kan worden gegeven. Als er een pad bestaat, wordt er als eerste de homepage geplaatst. Daarna komt elk niveau van het pad aan bod met telkens een link zodat u altijd eenvoudig kunt terug navigeren. Het laatste niveau, het niveau waar u zich bevindt, zal in het zwart zonder link worden aangeduid. Wanneer een naam te lang wordt, zal deze korter worden gemaakt.

8.4 News



In dit deel van de website kunt u de laatste vijf nieuwsartikels bekijken. De nieuwsartikels zijn onderverdeeld in twee categorieën: *ACTINET* en *The Global Actinide Community*. De

websiteverantwoordelijke krijgt hier ook de mogelijkheid om een inleiding te schrijven. Zo wordt aan de gebruiker duidelijk gemaakt waarvoor de nieuwsartikels dienen. Ook krijgt de websiteverantwoordelijke de mogelijkheid om nieuwsartikels toe te voegen. Bij elk artikel worden er de knoppen *edit* en *remove* geplaatst.



Figuur 8.5 Screenshot News

```
{* begin editor *}

{let    varclassname=$node.object.class_name
        varclassid=$node.object.contentclass_id
        varclasscreate=false
        varclassedit=true
        varclassremove=false
        vardraft=true
}
    {include uri="design/□ctinet/templates/editor.tpl"}
{/let}
{let    varclassname='news'
        varclassid='45'
        varclasscreate=true
        varclassedit=false
        varclassremove=false
        vardraft=false
}
    {include uri="design/□ctinet/templates/editor.tpl"}
{/let}
{* end editor *}

<h1>{$node.data_map.name.data_text|wash()}</h1>

{attribute_view_gui attribute=$node.data_map.description}

<h3>ACTINET</h3><br />

{let children=fetch( content, list, hash( parent_node_id, $node.node_id,
                                          sort_by, $node.sort_array,
```

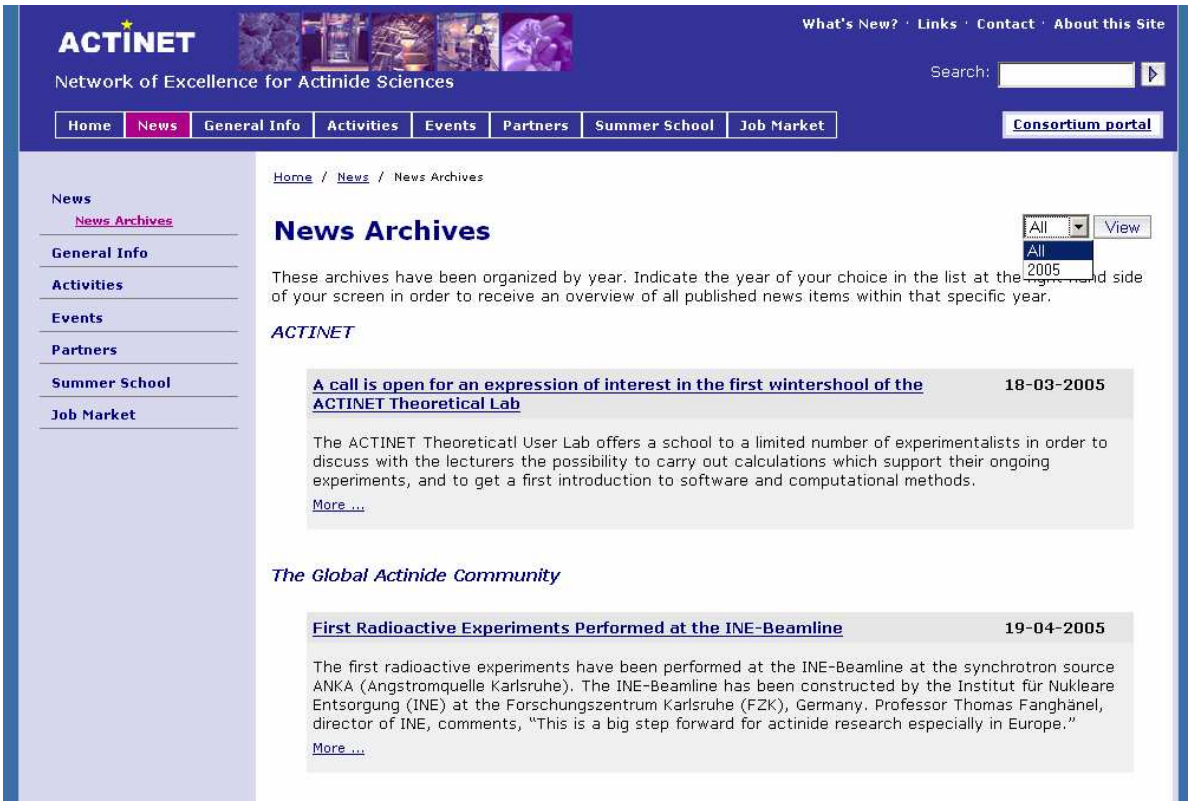
```
                                        limit, 5,
                                        class_filter_type, include,
                                        class_filter_array, array( 'news' ) ) )}
    {let item=false()}
        <div>
            {section var=Child loop=$children}
                {section show=eq($Child.data_map.type.data_text, '0')}
                    {*Display the content of the child using a line-view template*}
                    <table class="linenewstable" cellspacing="0" >
                        {node_view_gui view=line content_node=$Child}
                    </table>
                    {set item=true()}
                {/section}
            {/section}
        </div>
        {section show=eq( $item , false() ) }
            <p>There are no news items</p>
        {/section}
    {/let}
    <br />
    <h3>The Global Actinide Community</h3><br />

    {let item=false()}
        <div>
            {section var=Child loop=$children}
                {section show=eq($Child.data_map.type.data_text, '1')}
                    {*Display the content of the child using a line-view template*}
                    <table class="linenewstable" cellspacing="0" >
                        {node_view_gui view=line content_node=$Child}
                    </table>
                    {set item=true()}
                {/section}
            {/section}
        </div>

        {section show=eq( $item , false() )}
            <p>There are no news items</p>
        {/section}
    {/let}
{/let}
<br /><br />
```

Helemaal in het begin wordt de editor interface aangeroepen. Dit gebeurt via een paar gepaste parameters en een include van de editor-code. De editor-code werd eerder al besproken (zie 8.2).

In de code kunt u zien dat er een fetch gebeurt die de laatste 5 nieuwsartikels ophaalt. Om alleen de laatste 5 artikels te krijgen wordt er een parameter *limit* meegegeven. De twee parameters die beginnen met *class_filter* zorgen ervoor dat alleen nodes van de klasse *News* worden gefetched. Na het ophalen van de 5 laatste artikels wordt er per categorie de fetch overlopen. In deze loop wordt elke node gecontroleerd of ze van de juiste categorie zijn. Hiervoor zorgen de sections. Er wordt per categorie ook de variabele *item* gedeclareerd. Deze houdt bij of er een artikel wordt geplaatst onder die categorie. Wanneer er toch geen nieuwsartikel onder die categorie hoort, wordt er een gepaste boodschap getoond op het scherm.

8.4.1 News Archives

News Archives is eigenlijk hetzelfde als News, maar er zijn op sommige plaatsen een toevoeging aan de code. Daarom zal ik in dit deel de belangrijke stukjes code uitleggen die van toepassing zijn voor *News Archives*. De fetch die in deze webpagina gebeurt, is dezelfde als die van news, maar dan zonder de limiet van de laatste vijf nieuwsartikels. Er wordt ook een combobox gemaakt op deze pagina zodat u alle artikels van een jaar naar keuze kunt bekijken. In deze combobox zullen enkel de jaren staan die bij de datums horen van de artikels. Dus als u alleen maar artikels hebt van het jaar 2004 en 2005, zullen enkel 2004 en 2005 voorkomen in de combobox.



figuur 8.6 Screenshot News Archives

De volgende code plaatst de combobox op de webpagina.

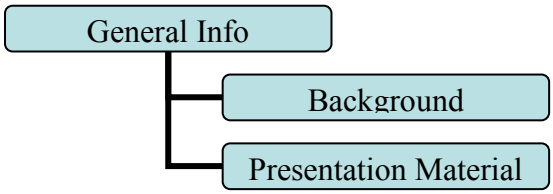
```
{* dropdownlist with the years that the user can select, generated *}
<table>
    <tr>
        <form method="post" action={"/content/action"|ezurl}>
            <td>
                {let year=array()}
                    {section var=Child loop=$children}
                       {set year=$year|append(
$Child.data_map.date.data_int|datetime( 'custom' , '%Y' ) )}
                    {/section}
                    {set year=$year|unique}

                    {set year=reversesortarray( $year )}

                    <select name="(fromyear)">
                        {section show=is_set($view_parameters.fromyear)}
                            <option selected='selected' value="all">All</option>
                        {section-else}
```

```
                            <option value="all">All</option>
                        {/section}
                        {section var=jaar loop=$year}
                            {section show=is_set($view_parameters.fromyear)}
                                {section show=eq( $view_parameters.fromyear, $jaar
)}
                                    <option selected="selected"
value="{$jaar}">{$jaar}</option>
                                {section-else}
                                    <option value="{$jaar}">{$jaar}</option>
                                {/section}
                            {section-else}
                                <option value="{$jaar}">{$jaar}</option>
                            {/section}
                        {/section}
                    </select>
                {/let}
            </td>
            <td>
                <input type="hidden" name="DestinationURL"
value="{$node.url_alias}"/>
                <input type="submit" class="buttonlayout" value="View"/>
            </td>
        </form>
    </tr>
</table>
```

De combobox wordt in een form geplaatst omdat er bij het klikken op de knop *View* terug naar dezelfde webpagina moet worden verwezen. Er wordt dan een URL parameter meegegeven die ervoor zorgt dat u kunt zien welke nieuwsartikels er moeten worden opgehaald. Deze URL-parameter wordt als volgt gemaakt:

1. Eerst worden alle attributen van *date* opgehaald en in de array *year* gestoken. Er wordt voor gezorgd dat alleen het jaar in de array wordt opgenomen.
2. Dan wordt de array *year* uniek gemaakt om dubbels te vermijden.
3. De array *year* wordt omgekeerd gesorteerd met de templateoperator *reversesortarray*. Deze templateoperator bevindt zich in de extensies en wordt besproken in het onderdeel *sort* in dit eindwerk (zie 7.8).
4. Daarna wordt de select gemaakt met als name *(fromyear)*.
5. In de select wordt er gekeken of er al een URL-parameter bestaat. Wanneer er al een URL-parameter staat, wordt ervoor gezorgd dat het juiste jaar wordt getoond in de combobox. De array *year* wordt ook doorlopen om alle jaren in de combobox te plaatsen.
6. Na de select wordt er een hidden variabele geplaatst met de juiste URL in. Deze URL is een URL alias van de huidige node. Er wordt gewerkt met een URL alias om de gebruiker een duidelijkere URL te laten zien.
7. Na de hidden variabele wordt er een submit-knop gezet die ervoor zorgt dat u terug naar dezelfde pagina gaat.

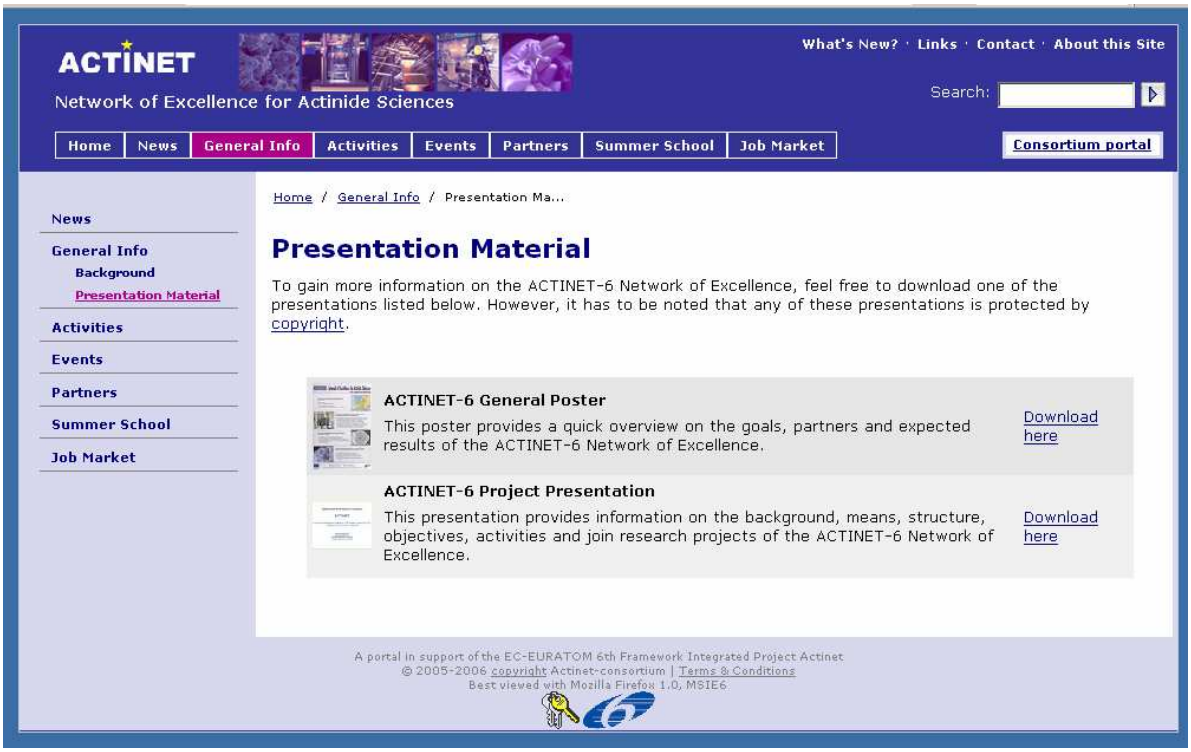
8.5 General Info



Bij het deel *General Info* wordt er wat meer informatie gegeven over de website en ACTINET. In dit deel zijn er 2 onderdelen: *Background* en *Presentation Material*. Het deeltje *Background* is een standaardwebpagina. Omdat de standaardpagina's al besproken zijn in een vorig deel van dit eindwerk (zie 8.1), zal ik deze webpagina niet meer bespreken. In dit deel zal enkel Presentation Material aan bod komen.

8.5.1 Presentation Material

In Presentation Material zullen zich nodes bevinden die verbonden zijn met content objecten van de content klasse *Presentation Material*. Er wordt een editor interface voorzien voor de websiteverantwoordelijke, zodat deze een inleidend stukje kan schrijven. Hij kan ook een nieuwe presentatie toevoegen. Bij elke presentatie wordt er een knop *edit* geplaatst om de presentatie te bewerken en een knop *remove* om de presentatie te verwijderen.



figuur 8.7 Screenshot Presentation Material

```
{let children=fetch( content, list, hash( parent_node_id , $node.node_id,
                                          class_filter_type, include,
                                          class_filter_array, array( 'presentation'
) ) )}
    <table class="presentationtable" cellspacing="0" >
```

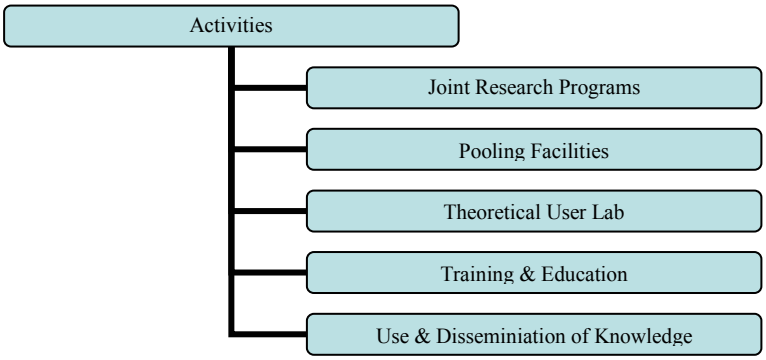
```
        {section var=presentation loop=$children sequence=array( 'rowgray' ,
'rowlightgray' )}
            <tr class={$presentation.sequence}>
                {node_view_gui view=line content_node=$presentation}
            </tr>
        {/section}
    </table>
{/let}
```

Dit stukje code is de kern van deze pagina die ervoor zorgt dat alle presentaties worden weergegeven. Hier wordt een fetch gedaan naar alle nodes die van de klasse *Presentation* zijn. Deze fetch wordt dan overlopen om elke presentatie afzonderlijk af te drukken op het scherm. Hiervoor wordt beroep gedaan op n*ode_view_gui*. Deze zorgt ervoor dat een line-template wordt opgeroepen. Een line-template is eigenlijk een verkleinde vorm van een full-template. De code van deze line-template komt zodadelijk aan bod. De line-templates worden in een tabel geplaatst om een goede structuur te creëren op het scherm.

```
<td>
    {section show=$node.data_map.image.has_content}
        <img class="presentationimage"
src={$node.data_map.image.content.articlethumbnail.url|ezurl} alt="" title=""/>
    {/section}
</td>
<td>
    <b>{attribute_view_gui attribute=$node.data_map.title}</b>
    {attribute_view_gui attribute=$node.data_map.introduction}
</td>
<td>
{section show=$node.object.data_map.presentation.has_content}
<a href={concat("content/download/",
         $node.object.id,
         "/",
         $node.object.data_map.presentation.id,
         "/file/",

$node.object.data_map.presentation.content.original_filename)|ezurl}>Download
here</a>
{section-else}
    <p>No file available</p>
{/section}
</td>
<td>
    {* begin editor *}
    {let    varclassname=$node.object.class_name
            varclassid=$node.object.contentclass_id
            varclasscreate=false
            varclassedit=true
            varclassremove=true
            vardraft=false
    }
    {include uri="design/actinet/templates/editor.tpl"}
    {/let}
    {* end editor *}
</td>
```

Deze code komt uit de line-template van presentation. Er wordt hier een kleine image geplaatst. Deze image zal als thumbnail geplaatst worden omdat hij anders veel te groot kan worden. Er komt een titel en introductie bij om een beetje uitleg te geven over de presentatie. Daarna wordt er een link geplaatst om de presentatie te downloaden. Als er geen file bestaat om te downloaden zal er een boodschap 'No file available' staan, anders krijgt u gewoon een line 'download here'. Als laatste komt er voor de websiteverantwoordelijke een knop *edit* en een knop *remove* bij.

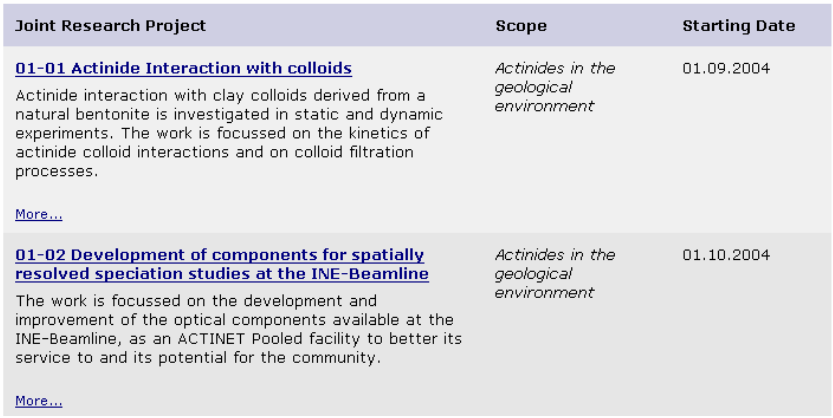
8.6 Activities



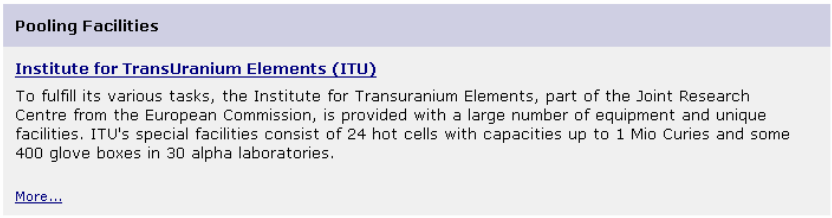
Bij Activities zijn er verschillende onderdelen. Ik zal alleen Joint Research Programs en Pooling Facilities uitleggen, omdat alle andere onderdelen gewone standaardwebpagina's zijn.

8.6.1 Joint Research Programs & Pooling Facilities

Joint Research Programs en Pooling Facilities wordt op dezelfde manier gemaakt. Ze komen ook van dezelfde content klasse *Activities*. Het enige verschil is dat er bij Pooling Facilities de attributen *Scope* en *Starting date* te zien zijn.



figuur 8.8 Screenshot Joint Research Projects



figuur 8.9 Screenshot Pooling Facilities

```
<h1>{$node.name}</h1>

{attribute_view_gui attribute=$node.data_map.description}
<br /><br />
<div class="pagetext">
<div class="subfolders">
{let children=fetch( content,
                     list,
                     hash( parent_node_id, $node.node_id,
                           sort_by, array( 'published', true() ),
                           class_filter_type, include,
                           class_filter_array, array( 'activity' ) ) )}
    <table class="jprtable" cellspacing="0">
        <tr class="jprtable_header">
            <th>Pooling Facilities</th>
            <th></th>
        </tr>
        {let item=false()}
            {section var=pf loop=$children sequence=array( 'rowlightgray' ,
'rowgray')}
                <tr class={$pf.sequence}>
                    {node_view_gui view=line content_node=$pf}
                    {set item=true()}
                </tr>
            {/section}
            {section show=eq( $item , false() )}
                <tr><td>There are no pooling facilities</td></tr>
            {/section}
        {/let}
    </table>
{/let}
</div>
</div>
```

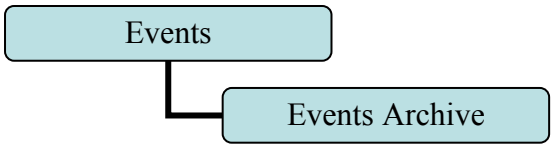
Deze code komt uit Pooling Facilities. Alle nodes worden opgehaald via een heel eenvoudige fetch die dan in de variabele *children* wordt opgeslagen. Daarna wordt er een tabel gemaakt met de juiste tableheaders. Hierop volgt een loop die de informatie van elke node plaatst met een line-template. Ook zoals bij het deeltje *News* zal de gebruiker een gepast bericht krijgen wanneer er geen activity is die getoond kan worden.

```
<td class="jprtable_middle">
    <b><a href={$node.url_alias|ezurl}>{attribute_view_gui
attribute=$node.data_map.title}</a></b>
    {attribute_view_gui attribute=$node.data_map.intro}
    <br />
    <div class="more"><a href={$node.url_alias|ezurl}>More...</a></div>
</td>
{section show=eq( $node.parent_node_id , 236 )}
    <td class="jprtable_left">
        {attribute_view_gui attribute=$node.data_map.scope}
    </td>
    <td class="jprtable_right">
        {$node.data_map.starting_date.data_int|datetime('custom' , '%d.%m.%Y' )}
    </td>
{/section}
<td>
    {* begin editor *}

    {let    varclassname=$node.object.class_name
            varclassid=$node.object.contentclass_id
            varclasscreate=false
            varclassedit=true
            varclassremove=true
            vardraft=false
    }
    {include uri="design/actinet/templates/editor.tpl"}
```

```
    {/let}
    {* end editor *}
</td>
```

Hier worden alle elementen van een rij afgedrukt. Dit zal in de eerste kolom een titel en een korte inleiding zijn. Voor de tweede en de derde kolom wordt er gecontroleerd op de node id van de parent node. Als die id gelijk is aan de node id van Joint Research Projects, dan zullen deze kolommen worden afgedrukt. Als laatste komt er nog een kolom bij voor de websiteverantwoordelijke waarin een knop *edit* en een knop *remove* staat voor het verwijderen van een activiteit.

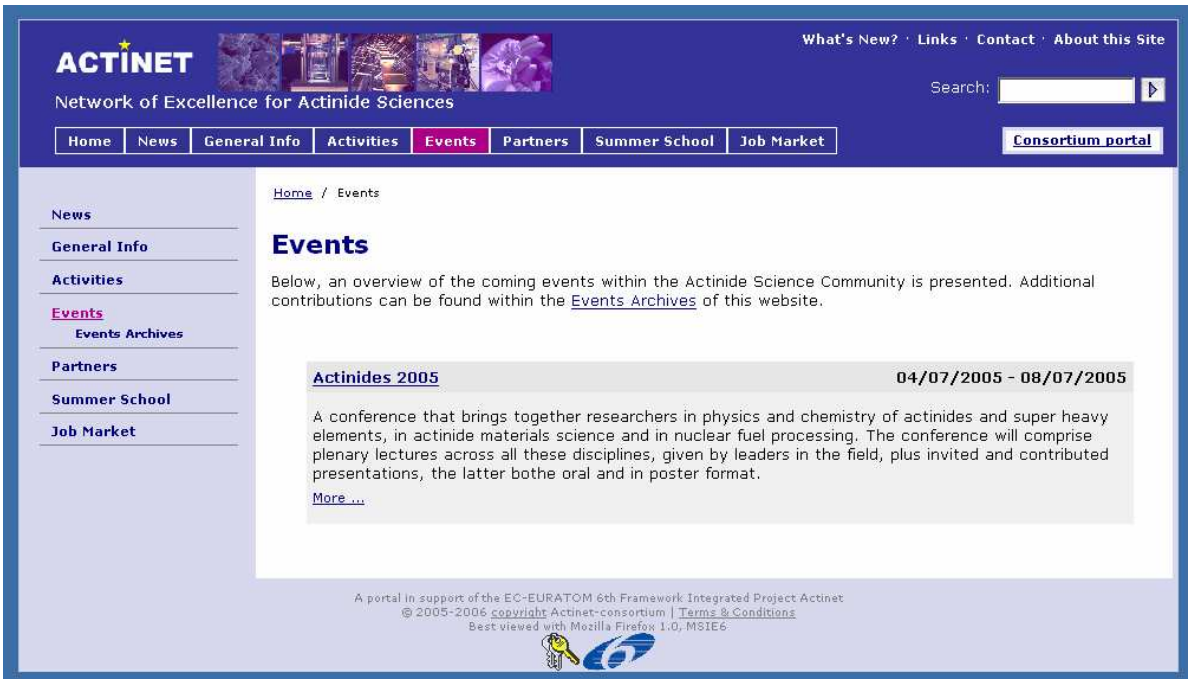
8.7 Events



In events wordt informatie van alle evenementen geplaatst. Het archief van events is iets uitgebreider dan de basisfolder van events. Daarom zal ik in dit deel enkel het archief van events uit de doeken doen.

8.7.1 Events Archives

In Events Archives worden alle events opgeslagen die niet meer getoond worden in Events zelf. Wanneer de einddatum van een event voorbij is, dan wordt deze gearchiveerd. Ook hier is er een combobox die ervoor zorgt dat alleen de events worden getoond van een bepaald jaar, maar deze code werkt hetzelfde als de code van de combobox van News Archives (zie 8.4.1).



Figuur 8.10 Screenshot Events

Event:

```
{let children=fetch( content, list, hash( parent_node_id, $node.node_id,
                                          sort_by, array( array(
'attribute',false(),'event/startdate' ) ),
                                          attribute_filter, array(array(
'event/enddate' , '>' , currentdate() ) ),
                                          class_filter_type, include,
                                          class_filter_array, array( 'event' ) ) )}
```

Event archief:

```
{let children=fetch( content, list, hash( parent_node_id, 63,
                                            sort_by, array( array( 'attribute' ,
false(), 'event/enddate' ) ),
                                            attribute_filter, array(array(
'event/enddate' , '<' , currentdate() ) ),
                                            class_filter_type, include,
                                            class_filter_array, array( 'event' ) ) )}
```

De twee vorige blokjes code zijn de fetches van de nodes. Het enige verschil is dat bij Events Archives alleen de events worden opgehaald waarvan de einddatum al voorbij is en bij het deel *Events* alle events waarvan de einddatum nog niet is verstreken.

```
{let item=false()}
    <div>
        <table class="eventfoldertable" cellspacing="0" >
            {section var=Child loop=$children}
                {* Display the content of the child using a line-view template. *}
                    {section show=is_set($view_parameters.fromyear)}
                        {section show=or(
                    eq( $Child.data_map.startdate.data_int|datetime( 'custom' ,
'%Y' ) , $view_parameters.fromyear),
                    eq( $view_parameters.fromyear , 'all' ) )}

                            {node_view_gui view=line content_node=$Child}
                        {/section}
                    {section-else}
                        {node_view_gui view=line content_node=$Child}
                    {/section}
```

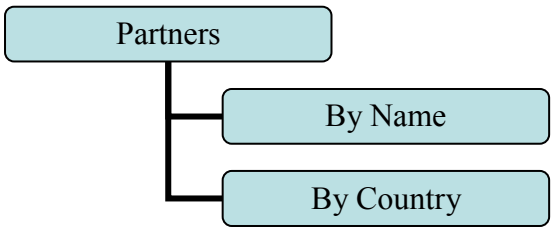
```
                {set item=true()}
            {/section}
        </table>
    </div>
    {section show=eq( $item , false() ) }
        <p>There are no events in the archive</p>
    {/section}
{/let}
```

Door dit stukje code worden alle events getoond die in het nieuwsarchief komen te staan. Ook hier is de controle of er events zijn om te tonen. Wanneer er geen events zijn, wordt er een gepaste boodschap getoond. Ook gebeurt er een controle of er een URL-parameter is opgegeven. Wanneer deze is opgegeven, worden alleen maar de events getoond van dat jaar. Het tonen van de events gebeurt met de line-template.

```
<tr class="rowgray" id="eventfoldertable_firstrow">
    <td id="eventfoldertable_leftcol">
        <a href={$node.url_alias|ezurl}>{attribute_view_gui
attribute=$node.object.data_map.title}</a>

    </td>
    <td id="eventfoldertable_rightcol">{attribute_view_gui
attribute=$node.object.data_map.startdate type='slashdate'} - {attribute_view_gui
attribute=$node.object.data_map.enddate type='slashdate'}
    </td>
</tr>
<tr class="rowlightgray">
    <td colspan="2">
        {attribute_view_gui attribute=$node.object.data_map.introduction}
        <div class="more">
            <a href={$node.url_alias|ezurl}>More ...</a>
        </div>
    </td>
</tr>
<tr class="rowlightgray">
    <td colspan="2">
        {* begin editor *}
        {let    varclassname=$node.object.class_name
                varclassid=$node.object.contentclass_id
                varclasscreate=false
                varclassedit=true
                varclassremove=true
                vardraft=false
                vardisplayinline=true
        }
        {include uri="design/actinet/templates/editor.tpl"}
        {/let}
        {* end editor *}
    </td>
</tr>
```

In de line-template staat de code voor het tonen van een paar gegevens van een event. In de line-template wordt er allereerst een titel gezet die een link bevat naar een webpagina waar alle informatie te vinden is van een event. Daarna wordt de startdatum en de einddatum afgedrukt. Vervolgens staat er een korte introductie van het event. Voor de websiteverantwoordelijke worden ook hier de knop *edit* en *remove* voorzien om een event te bewerken en te verwijderen.

8.8 Partners



In dit deel kunt u alle informatie bekomen over de ACTINET partners. Er zijn 3 verschillende manieren om aan uitgebreide informatie te raken van een partner. De eerste manier is via een clickable map, de tweede manier via *by name* en de derde manier via *by country*.

8.8.1 Partner

figuur 8.11 Screenshot Partner

In Partner staan alle gegevens van de ACTINET partner. Deze gegevens zijn dezelfde als diegene die in de klasse Member zijn besproken (zie 6.1).

Het speciale op deze pagina zijn de tabbladen die worden gegenereerd door Javascript. Om bij te houden op welk tabblad u zit bij een refresh, wordt er een cookie bijgehouden. Omdat de werking van de tabbladen de belangrijkste functie heeft in deze webpagina, zal ik deze hier uitleggen.

```
function show(page, tab, aantal) {
	hideall(aantal);

	pagina = document.getElementById(page);
	tabname = document.getElementById(tab);
	if (pagina)
	{
		pagina.style.display='block';
		tabname.className='outerselected';
	}
}

function hideall(aantal) {
	for( i = 1; i < (aantal+1) ; i++)
	{
		pagina = document.getElementById("page" + i);
		pagina.style.display='none';

		tabname = document.getElementById("tab" + i);
		tabname.className='outer';
	}
}
```

```
<script language="JavaScript" type="text/javascript"
src={'js/cookie/cookie.js'|ezdesign}></script>
<script language="JavaScript" type="text/javascript"
src={'js/tabfolder/tab.js'|ezdesign}></script>

<div id="membercontainer">
    <div id="tabmenu">
        <div id="tab1" class="outer"><a href="javascript:void 0;"
onclick="show('page1','tab1', 2); deleteCookie( 'tabpage'); setCookie( 'tabpage', 1
);"><span>General</span></a></div>

        <div id="tab2" class="outer"><a href="javascript:void 0;"
onclick="show('page2','tab2', 2); deleteCookie( 'tabpage'); setCookie( 'tabpage', 2
);"><span>Contact</span></a></div>

    </div>
    <div id="page1" class="tabpage">
    </div>
    <div id="page2" class="tabpage">
    </div>
</div>
<script language="JavaScript" type="text/javascript">
{literal}
    if( getCookie('tabpage') == null )
    {
        setCookie('tabpage', 1);
    }
    show( 'page' + getCookie( 'tabpage' ), 'tab' + getCookie( 'tabpage' ), 2);
{/literal}
</script>
```

```
function setCookie(name, value, expires, path, domain, secure) {
	var curCookie = name + '=' + escape(value) + ((expires) ? '; expires=' +
expires.toGMTString() : '') + ((path) ? '; path=' + path : '') + ((domain) ? ';
domain=' + domain : '') + ((secure) ? '; secure' : '');
	document.cookie = curCookie;
}

function getCookie(name) {
	var dc = document.cookie;

	// find beginning of cookie value in document.cookie
	var prefix = name + "=";
	var begin = dc.indexOf("; " + prefix);
```

```
	if (begin == -1) {
		begin = dc.indexOf(prefix);
		if (begin != 0) return null;
	}
	else begin += 2;

	// find end of cookie value
	var end = document.cookie.indexOf(";", begin);
	if (end == -1) end = dc.length;

	// return cookie value
	return unescape(dc.substring(begin + prefix.length, end));
}

function deleteCookie(name, path, domain) {
	var value = getCookie(name);
	if (value != null) document.cookie = name + '=' + ((path) ? '; path=' + path
: '') + ((domain) ? '; domain=' + domain : '') + '; expires=Thu, 01-Jan-70 00:00:01
GMT';
	return value;
}
```

Het eerste blokje code is het javascript die zorgt voor de werking van de tabbladen. De functie *show* zorgt voor het verbergen van de tabbladen en het tonen ervan. Het eerste wat de functie doet is het verbergen van alle tabbladen door de functie *hideall* op te roepen. Hier wordt er een loop gedaan die ervoor zorgt dat elk tabblad verborgen wordt. Daarna gaat de functie *show* verder door het tonen van het gekozen tabblad.

Het tweede blokje code zorgt voor het aanmaken van de tabbladen en dat de code van het javascript juist wordt aangesproken. Er wordt ook voor gezorgd dat er een cookie wordt gemaakt die onthoudt op welk tabblad de gebruiker zich bevond. Op het einde van de webpagina wordt een stukje javascript-code uitgevoerd zodat de gebruiker toch nog informatie krijgt te zien wanneer hij voor de eerste keer op deze pagina komt.

Het derde blokje code maakt en verwijdert de cookie voor de tabbladen. Deze werd gemaakt door de informatici van het Knowledge Management. De functie *setCookie* zorgt ervoor dat de cookie wordt gecreëerd. De functie *deleteCookie* zorgt ervoor dat de cookie wordt verwijderd. *DeleteCookie* heeft getCookie nodig om te kijken of de cookie dan ook daadwerkelijk bestaat.

8.8.2 Clickable map

figuur 8.12 Screenshot Partner Imagemap

Op deze pagina krijgt de gebruiker een Europese kaart te zien waar alle ACTINET partners zijn op aangeduid door middel van een bolletje. De gebruikers kunnen op de bolletjes klikken om naar de uitgebreide informatie te gaan van de partner waar ze op hebben geklikt. Dit heb ik verkregen door een map aan de afbeelding te koppelen. Er zijn ook via javascript tooltips aan elk bolletje bevestigd. Deze tooltips kunnen aangepast worden qua opmaak.

Het volgende stukje code staat in voor het plaatsen van de Europese kaart en de map die erbij hoort.

```
{* Imagemap *}
<div id="imagemapcontainer">
    <img id="imagemap" src={$node.object.data_map.map.content.original.url|ezurl}
title="European map with partners" alt="European map with partners"
usemap="#actinetmap"/>
</div>

<div>
    <map id="actinetmap" name="actinetmap">
        <area shape="circle" coords="178,267,4" alt=""
href={"public/partners/by_name/uniman"|ezurl} title="UNIMAN" onmouseover="return
overlib(xhtml['UNIMAN'], BORDER, 5, WIDTH, 200, ABOVE, OFFSETX, 1, OFFSETY, 7);"
onmouseout="return nd();"/>
        <area shape="default" nohref="nohref" alt="" />
    </map>
</div>
```

Aan de afbeelding van Europa wordt de map toegevoegd via het attribuut *usemap* van de tag *img*. In de code laat ik alleen maar de link naar één partner zien. Deze link wordt gemaakt door de tag *area*. Daar wordt de vorm en de coördinaten van een link geplaatst en de link zelf. *onmouseover* zorgt voor het laden van het javascript dat de tooltips laat zien op het scherm. *onmouseout* zorgt op zijn beurt voor het verdwijnen van de tooltips.

De gegevens van deze tooltip worden opgehaald uit een ander javascript. De gegevens de partner uit het vorige blokje code zijn de volgende:

```
xhtml["UNIMAN"]="<p style=\"padding:3px;\"><b>University of
Manchester</b><br/><br/><b>UNIMAN</b><br/>Manchester, UNITED KINGDOM</p>";
```

8.8.3 By Name

figuur 8.13 Screenshot Partners By Name

By Name is een deel van de website waar alle ACTINET partners worden opgehaald door middel van het alfabet.

```
{let    varclassname=Member
        varclassid=16
        varclasscreate=true
        varclassedit=false
        varclassremove=false
        vardraft=true
}
{include uri="design/actinet/templates/editor.tpl"}
{/let}

<h1>Partners By Name</h1>
<div>
    <div id="containeralfabet">
        {let letter=array()}
            {let children=fetch( content, list, hash(
                    parent_node_id, $node.node_id,
                    sort_by, array( 'attribute', true(), 'member/shortname'),
                    class_filter_type, include,
                    class_filter_array, array( 'member' ) ) ) }
                {section var=loop loop=$children}
```

```
                    {set
letter=$letter|append($loop.data_map.shortname.data_text|wash|extract(0,1)|upcase()
)}
                {/section}
            {/let}
            <br/>
            {set letter=$letter|unique}
            <br/>
            <br/>
            {let alfabet=array( A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z
)}
                {let test=false()}
                    {section var=loop loop=$alfabet}
                        {section var=letterloop loop=$letter}
                            {section show=eq($test, false())}
                                {section show=eq($letterloop.item, $loop.item)}
                                    {set test=true()}
                                {section-else}
                                    {set test=false()}
                                {/section}
                            {/section}
                        {/section}
                        {section show=eq($test,true())}
                            {let page=concat( $node.url_alias ,'/(offset)/',
$loop.item )}
                                <a class="alfabetclick"
href={$page|ezurl}>{$loop.item}</a> 
                            {/let}
                        {section-else}
                            <a class="alfabetnonclick">{$loop.item}</a> 
                        {/section}
                        {set test=false()}
                    {/section}
                {/let}
            {/let}
        {/let}
    </div>
    <br/>
    <br/>
    <div id="namecontainer">
        {section show=eq('', $view_parameters.offset )|not()}
        {let children=fetch( content, list, hash(
              parent_node_id, $node.node_id,
              sort_by, array( array( 'attribute' , true() , 'member/shortname' ) ),
              class_filter_type, include,
              class_filter_array, array( 'member' ),
              extended_attribute_filter, hash(id,
                                              'like',
                                              params,
                    hash(like_value,concat($view_parameters.offset,'%') ) ) ) ) }
            <div id="page{$loop.index}" class="memberlist">
                <h3>{$view_parameters.offset}</h3>
                <ul>
                    {section var=child loop=$children}
                        <li>
                            <a
href={$child.url_alias|ezurl}>{$child.data_map.shortname.content|wash}:
{$child.data_map.name.content|wash}</a>
                        </li>
                    {/section}
                </ul>
            </div>
        {/let}
        {/section}
    </div>
    <div class="cleardivclean"></div>
</div>
```

Eerst worden alle ACTINET partners gecontroleerd op de eerste letter van hun naam. Dit is om te kijken achter welke letter er een hyperlink mag schuilen. Daarna wordt ervoor gezorgd dat deze array uniek wordt gemaakt, dus alle dubbele letters worden eruit gefilterd. Vervolgens wordt elke letter van het alfabet overlopen en gecontroleerd. Deze letter moet overeenkomen met de letter van de array met unieke letters. Als de letter hetzelfde is, dan wordt er een hyperlink geplaatst. Er wordt ook met een boolean bijgehouden wanneer een letter is geplaatst zodat deze maar één maal voorkomt. De letter wordt geplaatst zonder hyperlink als de twee letters niet overeenkomen. Zo vermijdt u dat er letters zijn waar geen enkele ACTINET partner achter schuilt.

Nadat de gebruiker op een letter heeft geklikt, wordt hij terug naar dezelfde pagina verwezen, maar deze keer met een URL parameter. Er wordt een fetch gedaan aan de hand van deze parameter om zo de juiste ACTINET partners uit de database te halen. In deze fetch komt de extensie *extendedattributefilters* heel goed van pas omdat deze functioneert zoals een *like* in SQL. Van de ACTINET partners krijgt u enkel de afkorting en de volledige naam te zien. Deze informatie wordt als link getoond en verwijst naar de uitgebreidere informatie van de ACTINET partner.

Een voorbeeld: Ik wil alle ACTINET partners zien die hun naam begint met de letter *S*.
Ik klik op de letter *S* als die beschikbaar is en alle ACTINET partners met deze beginletter komen tevoorschijn.

In figuur 8.8 ziet u een screenshot van by name. De screenshot toont de situatie van het voorbeeld dat ik net heb gebruikt.

8.8.4 By Country

figuur 8.14 Screenshot Partners By Country

By Country lijkt sterk op *By Name*. Alleen worden de ACTINET partners hier geselecteerd per land. Op de webpagina zelf kunt u de vlag en de naam van elk land zien. Het is mogelijk om op de vlag en de naam van een land te klikken, omdat er een hyperlink achter schuilt. Wanneer op een vlag of naam van een land wordt geklikt, wordt men terug naar dezelfde pagina verwezen met een URL parameter en worden alle ACTINET partners uit de database gehaald door middel van het geselecteerde land.

```
<h1>Partners By Country</h1>

<div>
    <div id="flagcontainer">
        {let country=array()}
            {let children=fetch( content, list, hash(
                 parent_node_id, 78,
                 sort_by , array( array( 'attribute', true(), 'member/country' ) ),
                 class_filter_type, include,
                 class_filter_array, array( 'member' ) ) ) }

                {section var=child loop=$children}
                    {set country=$country|append(
$child.object.data_map.country.data_text|wash ) }
                {/section}
            {/let}
        {set country=$country|unique}
            <div id="flagcontainer">
                {let count=0}
                <table class="flagtable">
                {section var=land loop=$country}

                    <td>
                    <div class="flag">
                        {let page=concat( $node.url_alias ,'/(offset)/',
$land|downcase() ) }
                            {let vlag=$land.item|append(".gif")}
                                {set vlag=concat( "flag/" , $vlag )|downcase()}
                                <a href={$page|ezurl}><img class="flagimage"
src={$vlag|ezimage} alt="{$land|downcase()|upfirst}"/></a><br />
                            {/let}
                            <a href={$page|ezurl}>{$land|downcase()|upfirst}</a>
                        {/let}
                    </div>
                    </td>
                    {section show=eq( mod( $count, 7 ) , 6 )}
                        </tr></table><table class="flagtable">
                    {/section}{set count=$count|inc}
                {/section}
                </table>
                {/let}
            </div>
        {/let}
    </div>
    <br />
    <div id="countrycontainer">
        {section show=eq( '0' , $view_parameters.offset )|not()}
        {let children=fetch( content,
                             list,
                             hash( parent_node_id, 78,
                                   sort_by, array( array( 'attribute' , true() ,
'member/shortname' ) ),
                                   class_filter_type, include,
                                   class_filter_array, array( 'member' ),
                                   attribute_filter, array(array( 'member/country',
'=', urldecode($view_parameters.offset ) ) ) ) ) }
            <div class="memberlist">
                <h3>{urldecode( $view_parameters.offset )|upfirst()}</h3>
                <ul>
```

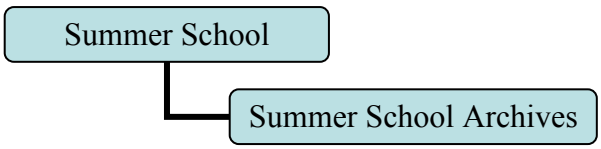
```
                    {section var=myloop loop=$children}
                        <li>
                            <a
href={$myloop.url_alias|ezurl}>{$myloop.data_map.shortname.data_text|wash}:
{$myloop.data_map.name.data_text|wash}</a>
                        </li>
                    {/section}
                </ul>
            </div>
        {/let}
        {/section}
    </div>
</div>
```

Om de webpagina te verkrijgen wordt er een array gevuld met alle landen van elke ACTINET partner. Deze array wordt uniek gemaakt om dubbels te vermijden. Daarna wordt er aan de hand van het land een image uit de mappenstructuur gehaald en op de webpagina geplaatst. De juiste naam wordt mee onder elke image gezet. Na zeven landen wordt er een nieuwe rij geplaatst om zo alle landen op de webpagina te krijgen. Zo bereikt men de navigatie voor deze webpagina. Het weergeven van elke ACTINET partner is net hetzelfde als in By Name, maar de fetch heeft één klein verschil. Er worden alleen ACTINET partners gefetchet die het juiste land hebben.

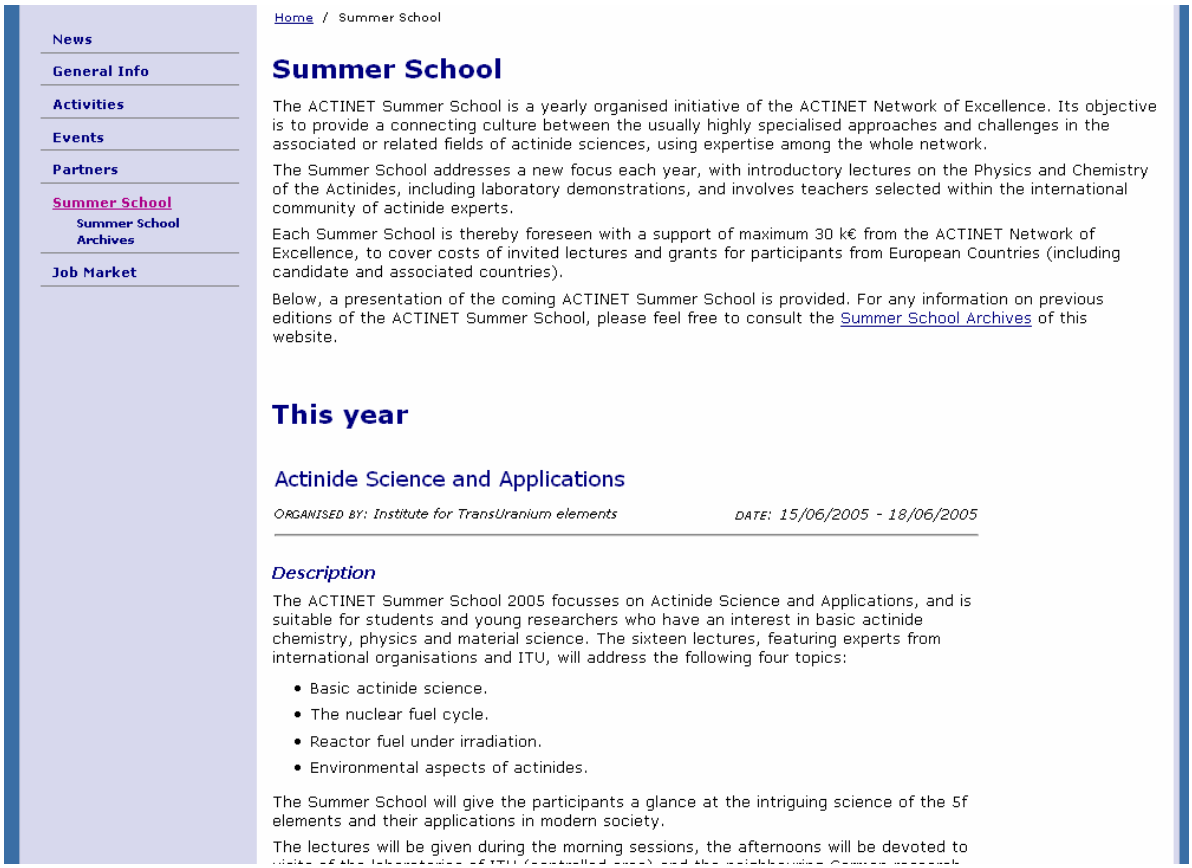
Een voorbeeld: Ik wil alle ACTINET partners zien die zich in België bevinden.
Ik klik gewoon op de Belgische vlag of naam en krijg al de Belgische ACTINET partners te zien.

In figuur 8.14 kunt u een screenshot zien van by country. Op deze screenshot is België geselecteerd zoals in het vorige voorbeeld.

8.9 Summer School



Op de webpagina van Summer School is een korte inleiding te vinden. Er is ook alle informatie van de huidige Summer School. Zo is er elk jaar een nieuwe Summer School. Alle Summer Schools worden in het archief geplaatst om toch de informatie na afloop te kunnen bekijken. Het wordt ook mogelijk om in te schrijven voor een Summer School. Dit is enkel mogelijk vanaf het jaar 2006. De verantwoordelijke van de Summer School zal een user account krijgen waarmee hij op de website kan inloggen. Zo kan hij de gegevens van alle mensen bekijken die zich hebben geregistreerd voor een Summer School.



figuur 8.15 Screenshot Summer School

```
{let children=fetch( content , list , hash(
                                        parent_node_id , 65,
                                        class_filter_type, include,
                                        class_filter_array, array( 'event' ) ) )}
    {section var=summerschool loop=$children}
        {section show=eq(
             $summerschool.data_map.startdate.data_int|datetime( 'custom' , '%Y') ,
             currentdate()|datetime( 'custom' , '%Y') )}
            {node_view_gui view=full content_node=$summerschool}
        {/section}
    {/section}
{/let}
```

Deze code heeft maar één functie en dat is het ophalen van de huidige Summer School.

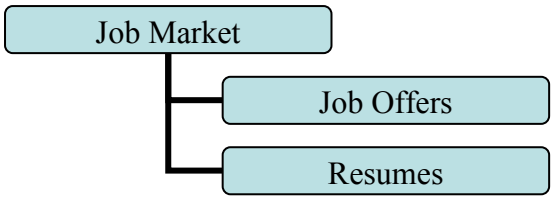
In het archief worden de voorbije Summer Schools getoond met een link naar de volledige beschrijving van de Summer School.

```
{let children=fetch( content, list, hash(
               parent_node_id, 65,
               sort_by, array( array( 'attribute' , false(), 'event/startdate' ) ),
               class_filter_type, include,
               class_filter_array, array( 'event' ) ) )}
    {let item=false()}
        <div>
            <table class="eventfoldertable" cellspacing="0" >
                {section var=Child loop=$children}
                    {section show=eq(
                      $Child.data_map.enddate.data_int|datetime( 'custom' , '%Y') ,
                      currentdate()|datetime( 'custom' , '%Y' ) )|not }

                        {node_view_gui view=line content_node=$Child}
```

```
                        {set item=true()}
                    {/section}
                {/section}
            </table>
        </div>
        {section show=eq( $item , false() ) }
            <p>There are no summer schools in the archive</p>
        {/section}
    {/let}
{/let}
```

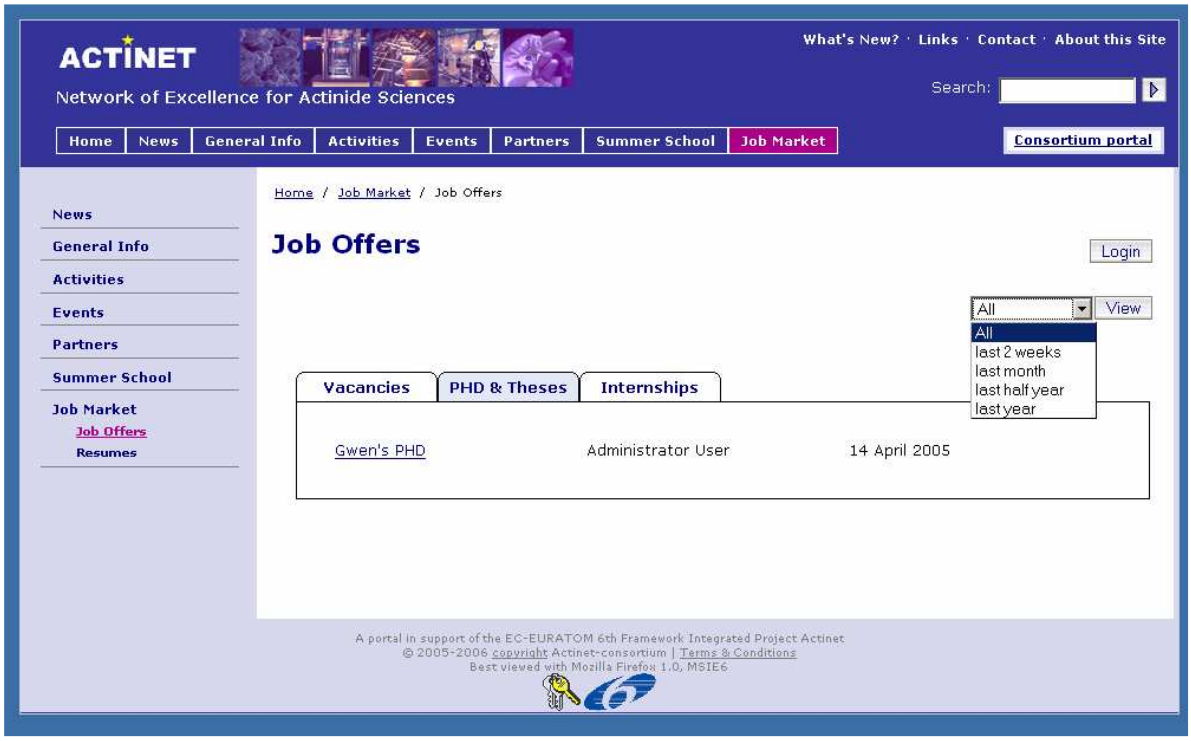
Dit stukje code zorgt ervoor dat alle Summer Schools worden gefetched uit de database. Daarna wordt deze fetch overlopen om ze allemaal te tonen op het scherm via een line-template. Enkel de huidige Summer School wordt niet getoond in het archief.

8.10 Job Market



Job Market zorgt ervoor dat bedrijven, instituten of universiteiten de mogelijkheid krijgen om advertenties te plaatsen. Werkzoekenden kunnen hun CV op de website zetten.

8.10.1 Job Offers en Resumes



figuur 8.16 Screenshot Job Offers

Het registreren van de Job Offer User is al eerder besproken (zie 7.10). Job Offer User en Resume User geven de mogelijkheid aan de gebruiker om zich in te loggen. Wanneer hij ingelogd is, krijgt hij allerlei mogelijkheden ter beschikking. De eerste mogelijkheid is het veranderen van zijn eigen paswoord. Dit is een ingebouwde functionaliteit van eZ publish en wordt aangeroepen met de volgende code:

```
<div class="useraccountdiv">
	{section show=eq( $user.login, 'anonymous' )|not()}
		<form method="post" action={"/user/password"|ezurl}>
			<input class="buttonlayout" type="submit" name="NewButton"
value="Change Password"/>
		</form>
	{/section}
</div>
```

De gebruiker krijgt ook een knop aangeboden om naar de login-pagina te gaan. Hier krijgt hij de mogelijkheid om zich in te loggen met de login en het paswoord die hij ontvangen heeft per mail. Wanneer hij zijn paswoord is vergeten, kan hij hiervoor een functie gebruiken die ervoor zorgt dat er een mailtje wordt gestuurd met een nieuw paswoord.
Wanneer de gebruiker nog geen user account heeft, kan hij zich hier registreren. De code die ervoor zorgt dat u naar deze pagina kunt surfen, gaat als volgt:

```
<div class="useraccountdiv">
	{section show=eq( $user.login, 'anonymous' )}
		<form method="post" action={"/user/login"|ezurl}>
			<input type="hidden" name="frompage" value="joboffers"/>
			<input class="buttonlayout" type="submit" name="NewButton"
value="Login"/>
		</form>
	{section-else}
		<form method="post" action={"/user/logout"|ezurl}>
			<input class="buttonlayout" type="submit" name="NewButton"
value="Logout"/>
		</form>
	{/section}
</div>
```

De Job Offer User kan ook een advertentie plaatsen op de website. Dit gebeurt via volgende code:

```
<div class="buttondiv">
	{section show=or(
			eq( $user.contentobject.class_name , 'Job Offer User'),
			eq( $user.contentobject_id, 85 ) ) }
		<form method="post" action={"/content/action"|ezurl}>
			<input type="hidden" name="NodeID" value="{$node.node_id}" />
			<input type="hidden" name="ClassID" value="21"/>
			<input class="buttonlayout" type="submit" name="NewButton"
value="Add a Job Offer"/>
		</form>
	{/section}
</div>
```

Natuurlijk willen we niet dat de gebruiker zomaar nepadvertenties op de website plaatst. Voor een Job Offer wordt gepubliceerd, wordt deze gecontroleerd door de websiteverantwoordelijke. Deze functie zal niet beschikbaar zijn voor de Job Offer User. Het goedkeuren en afkeuren van een advertentie is, net zoals het paswoord veranderen, een ingebouwde functie. Deze functie wordt aangeroepen via volgende code:

```
<div class="useraccountdiv">

      {section show=eq( $user.contentobject_id, 85 )}
            <form method="post" action={"/collaboration/view/summary"|ezurl}>
                  <input class="buttonlayout" type="submit" name="NewButton"
value="Approve new Job Offers"/>
            </form>
      {/section}
</div>
```

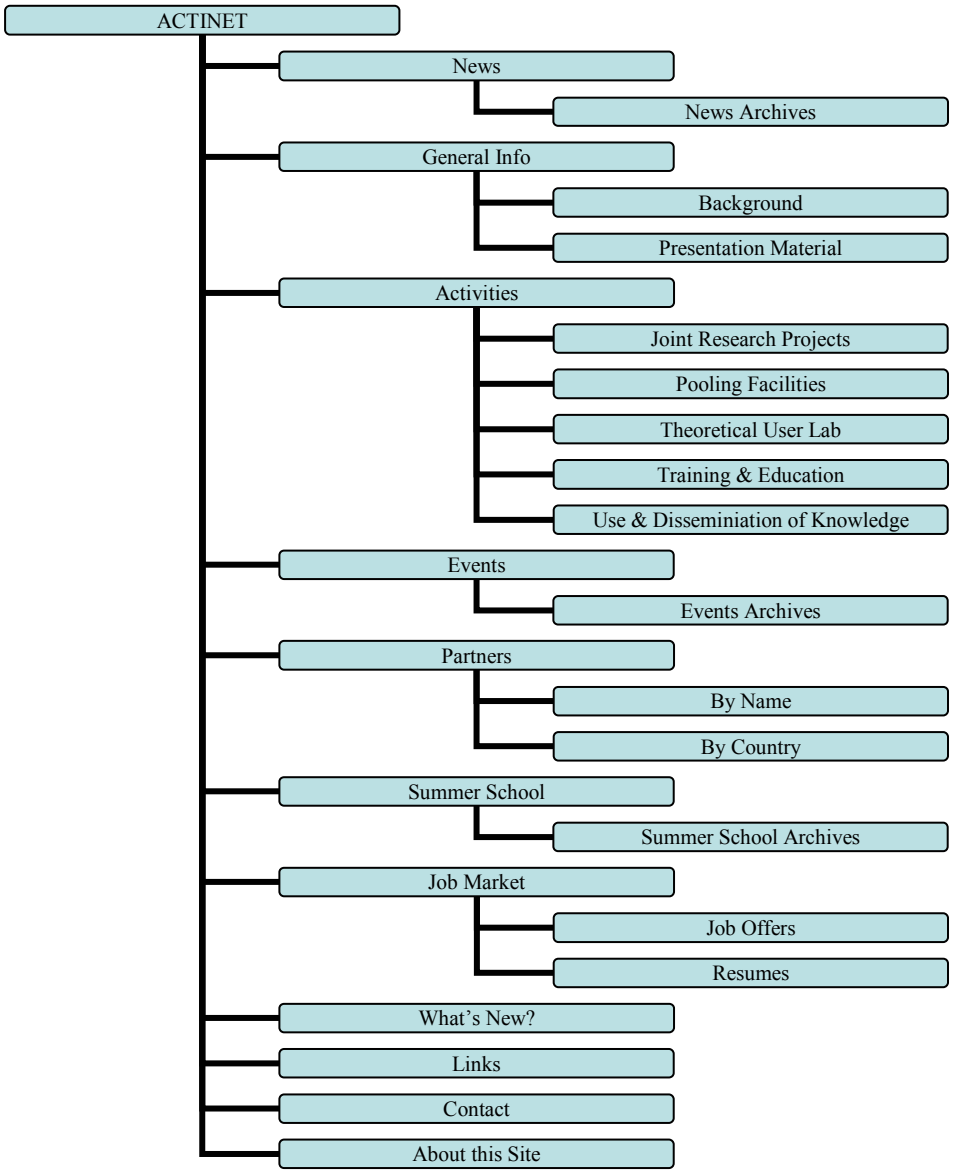
Besluit

De ACTINET website bezit alles wat ervoor kan zorgen dat hij kan uitgroeien tot een professionele website. Het deel van Job Market is nog niet volledig uitgewerkt door een gebrek aan tijd, maar de website zal nog verder ontwikkeld worden, zodat hij eind juni gelanceerd kan worden.

Ik kan na deze stage zeggen dat de opdracht voor mij persoonlijk een hele ervaring is. Ik heb er ongelooflijk veel van bijgeleerd. Het uitvoeren van mijn stageopdracht heeft zeker mijn interesse in de wereld van de informatica versterkt. Het werken met een (voor mij) totaal onbekend systeem zoals eZ publish leek in het begin schrikwekkend. Nu is dit gevoel totaal omgekeerd. Ik zou me niet meer kunnen inbeelden om in een ander systeem te werken. Na deze drie maanden heb ik een zekere basiskennis in eZ publish kunnen opbouwen. Wat mij vooral boeit, is de kracht en uitbreidbaarheid van het systeem. Ik hoop dat ik in de toekomst zeker met dergelijke systemen in aanraking blijf komen.

Bijlagen

Bijlage 1: Content Structuur publieke website ACTINET



Literatuurlijst

VERWIMP, L. en VERLEDENS, A., 2002. 1952-2002 SCK•CEN studiecentrum voor kernenergie Centre d'Etude de L'Energie Nucléaire. 57 p.

SCK•CEN, 1999. Klaar voor de 21ste eeuw. 32 p.

ACTINET - a Network of Excellence for Actinide Sciences. http://www.actinet-network.org/index-2.html

The Apache Software Foundation. http://www.apache.org/foundation/how-it-works.html#history

The PHP Group. http://www.php.net

SVN, 2005. Subversion Project Home. http://subversion.tigris.org/

KÜNG, S., LÜBBE, O., LARGE, S., 2005. TortoiseSVN A Subversion client for Windows. http://tortoisesvn.tigris.org/docs/TortoiseSVN_en/help-onepage.html

DANIELS, C., http://www.khlim.be/~cdaniels/XHTML.html

W3C HTML Working Group, 2002. XHTML 1.0 The Extensible HyperText Markup Language (Second Edition). http://www.w3.org/TR/xhtml1/

REFSNES, S. Introduction to CSS. http://www.w3schools.com/css/css_intro.asp.

MySQL. http://www.mysql.com

ONBEKEND. Apache Server Frequently Asked Questions. http://httpd.apache.org/docs/misc/FAQ.html#what

ONBEKEND, 2005. Networks of excellence in IST Priority. http://www.cordis.lu/fp6/faq_ist_noe.htm (13 mei 2005).

ONBEKEND, 2004. Actinet 6 Network for Actinide Sciences. http://dbs.cordis.lu/fep-cgi/srchidadb?ACTION=D&CALLER=FP6_PROJ&QM_EP_RCN_A=71466 (13 mei 2005).

TRACHTENBERG, A., 2004. Why PHP5 Rocks!. http://www.onlamp.com/pub/a/php/2004/07/15/UpgradePHP5.html (20 mei 2005).

HOLLY 'N, J., 2004. IE6 Peekaboo Bug. http://www.positioniseverything.net/explorer/peekaboo.html (24 mei 2005)

Sixth Framework Programme 2002 – 2006. http://europa.eu.int/comm/research/fp6/index_en.html

RUYSSEN, M.L., CHAIX, P., 2004. Communication Action Plan. Niet-gepubliceerd PDF-bestand, Mol, SCK•CEN, 5 p.

RUYSSEN, M.L., MOONS, F., LEGRAIN, CH. Managing Nuclear Knowledge: a SCK•CEN concern, Status of a practical Knowledge Management approach. PDF-bestand, Mol, SCK•CEN, 8 p.