Utveckling av modul för hantering av kvalitetsdokument i innehållshanterings-systemet DotNetNuke

Simon Nordman

Examensarbete för ingenjörsexamen (YH)

Utbildningsprogrammet för informationsteknik

Vasa 2010

YRKESHÖGSKOLAN NOVIA

EXAMENSARBETE

Författare: Simon Nordman
Utbildningsprogram och ort: Informationsteknik, Vasa
Handledare: Mats Braskén

Titel: *Utveckling av modul för hantering av kvalitetsdokument i innehållshanteringssystemet DotNetNuke.*

Datum 25.11.2010 Sidantal 24 Bilagor 3

Sammanfattning

Detta examensarbete utfördes åt Oy Ibiworks Ab. Syftet med arbetet var att skapa en lösning för hantering av kvalitetsdokument. Lösningen kommer att kunna erbjudas som ett alternativ till kunder som inte från förut har IBM Lotus Notes, som deras nuvarande system kräver. Tyngdpunkten för detta projekt låg i att skapa ett arbetsflöde samt versionshantering för dokumenten. Arbetet utfördes till största del med hjälp av webbapplikationsramverket ASP.NET och DotNetNuke som underliggande innehållshanteringssystem. Resultatet blev en fungerande versionshantering samt halvdynamiska arbetsflöden för att i framtiden enkelt lägga till flera arbetsflöden.

Språk: svenska

Nyckelord: DotNetNuke, modul, dokumenthantering, ärendehantering

Förvaras: Tritonia, Vasa vetenskapliga bibliotek.

YRKESHÖGSKOLAN NOVIA

OPINNÄYTETYÖ

Tekijä: Simon Nordman
Koulutusohjelma ja paikkakunta: Tietotekniikka, Vaasa
Ohjaaja: Mats Braskén

Nimike: *Moduulin kehittäminen asiakirjahallinnalle sisällönhallintajärjestelmässä DotNetNuke*

Päivämäärä 25.11.2010 Sivumäärä 24 Liitteet 3

Tiivistelmä

Opinnäytetyö suoritettiin Oy Ibiworksille Ab:lle. Työn tarkoitus oli luoda selainpohjainen hallintaratkaisu eri yrityksien laatudokumenteille. Tätä ratkaisua tulee yritys tarjoamaan vaihtoehdoksi asiakkaille, joilla ei ole IBM Lotus Notes ohjelmaa, jota yrityksen nykyinen laatudokumenttien hallintaratkaisu vaatii. Projektin painopisteenä oli luoda sovellukseen työnkulkutoiminto ja dokumenttien versionhallinta. Ohjelmaa kehitettäessä on pääsäätöisesti käytetty ohjelmistokomponenttikirjastoa ASP.NET sekä DotNetNuke sisällönhallintajärjestelmää. Projektin tuloksena on toimiva versionhallinta sekä osittain dynaamisia työnkulkutoimintoja, joita jatkossa voidaan helposti käyttää muissa samantapaisissa työnkulkusovelluksissa.

Kieli: ruotsi Avainsanat: dotnetnuke, moduuli, asianhallintajärjestelmä, asiakirjahallinta

---

Arkistoidaan: Tritonia, Vaasan tiedekirjasto.

YRKESHÖGSKOLAN NOVIA

BACHELOR'S THESIS

Author: Simon Nordman
Degree Programme: Information Technology
Supervisor: Mats Braskén

Title: *Development of a module for managing quality documents in the DotNetNuke content management system.*

Date 25.11.2010 Number of pages 24 Appendices 3

Summary

This thesis was made for Oy Ibiworks Ab. The purpose of the thesis was to create a solution for managing quality documents. The solution will be offered as an alternative to customers who don't have IBM Lotus from before, which their current solution requires. The main issue for this project lies in creating a workflow as well as version management for the documents. The work was mainly done with the ASP.NET web application framework and DotNetNuke as the underlying content management system. The result is a working version management with semi-dynamic workflows which in the future will make it easier to add several workflows.

Language: Swedish Key words: dotnetnuke, module, case management system, document management

---

Filed at: The Tritonia Academic Library, Vaasa.

Innehållsförteckning

Figurförteckning

Bilageförteckning

1 Inledning

Detta examensarbete utfördes som projekt åt Ab Ibiworks Oy, som är ett mindre mjukvaruföretag stationerat i Jakobstad. Projektet gick ut på att förverkliga en alternativ lösning till hantering av kvalitetsdokument. Denna typ av produkt riktar sig till alla som vill hantera och arkivera kvalitetsdokument elektroniskt.

1.1 Företaget

Ibiworks har specialiserat sig på kundanpassade systemlösningar, systemunderhåll och programmering. Företaget utför uppdrag i hela Norden och med över 10 års erfarenhet utvecklar och effektiviserar Ibiworks verksamhetsstödjande datasystem åt företag och organisationer. Kunderna består såväl av internationella så som nationella företag. Ibiworks erbjuder lösningar inom dokumenthantering, informations- och kunskapssystem, webbutveckling, systemintegration och programmering. Därtill installerar, underhåller och konfigurerar man servrar, datorer och kommunikationsutrustning. /7/

1.2 Vad är kvalitetsdokument?

Ett kvalitetsdokument innehåller t.ex. föreskriven dokumentation om hur en process går till vid tillverkning av en produkt. Denna handling kan vara till hjälp för berörd personal och ligger till grund för att processen utförs korrekt och resulterar i en produkt som uppfyller en viss kvalitetsstandard. Kvalitetsdokumenten hör även ofta till ett kvalitetssystem där kvalitetssystemet är en standard med krav vad gäller kvalitetsarbete i organisationer. Uppfyllande av kraven enligt ett kvalitetssystem leder till certifiering eller godkännande. Det finns många kvalitetssystem som utgör standarder för kvalitetsarbete inom olika branscher. Den vanligaste standarden för ett kvalitetssystem är ISO 9000 och den syftar till att underlätta leverantörssamverkan och att skapa framgångsrika samarbeten mellan företag /15/.

1.3 Dokumenthantering

Kvalitetsdokument sparas av företag på olika sätt. Många sparar dem i pappersformat eller elektroniskt i form av dokument som ligger på en arbetsstation eller server. Ett

dokumenthanteringssystem (DMS) utgörs av dokument i elektronisk form och har ofta ett innehållshanteringssystem under sig. På detta sätt kan man exempelvis dra nytta av hanteringen av användare och säkerhet som innehållshanteringssystemet använder sig av från förut. Det som kännetecknar ett dokumenthanteringssystem är versionshantering, säkerhet och åtkomsten till dokumenten, möjligheten att gå tillbaka till en version samt arbetsflöden med revidering /18/.

1.4 Bakgrund och uppgiftsbeskrivning

Syftet med projektet är att skapa en modul för det webbaserade innehållshanteringssystemet DotNetNuke (eng. WCM eller CMS). Denna modul kommer sedan att tillsammans med DotNetNuke kunna erbjudas som ett alternativ till de kunder som inte har eller inte vill ha IBM Lotus Notes. Alternativa webbaserade innehållshanteringssystem är t.ex. Drupal eller Joomla. Båda är fri programvara med öppen källkod men som ligger under GNU General Public Licence. Denna licens kräver att man gör källkoden tillgänglig för andra om man gör en modifikation eller ett tillägg som körs under eller är direkt kopplad till Drupal /6/. Detta är orsaken till att DotNetNuke valdes som plattform.

1.4.1 Beskrivning av den nuvarande lösningen

Den nuvarande lösning som erbjuds för hantering av kvalitetsdokument är gjord för Lotus Notes. En närmare beskrivning av denna lösning är inte nödvändig eftersom att den nya lösningen i stor utsträckning kommer att likna den nuvarande vad gäller funktionalitet.

Lotus Notes är en grupprogramvara från IBM som omfattar bland annat dokumenthantering, e-post och webblösningar. Dessutom finns kalenderfunktion och stöd för snabbmeddelanden via något som kallas Sametime. En av fördelarna med Lotus Notes är att det är enkelt att bygga applikationer eller att anpassa de standardapplikationer som medföljer programvaran. /8/

Därtill får man en server som innehåller alla funktioner, som intern och extern e-post samt en databashanterare.

Lotus Notes utvecklas av Lotus Software som sedan 1995 ägs av IBM. Nuförtiden består mjukvaran av fyra komponenter: Klienten Lotus Notes, utvecklingsverktyget

Domino Designer, administrationsverktyget Domino Administrator och servern Lotus Domino.

Programmet används oftast av större företag, organisationer, föreningar och kommuner. Då Lotus Notes är en grupprogramvara kan det för kunden ibland kännas överväldigande och rent av onödigt om kunden inte har det från tidigare. Lotus Notes kan även vara för dyrt för mindre företag. Dessa är orsakerna till att Ibiworks vill kunna erbjuda en alternativ lösning för hantering av kvalitetsdokument.

1.4.2 Beskrivning av uppgiften

Modulen kommer så långt som möjligt att byggas och implementeras enligt DotNetNukes alla regler, för att få en så bra integrerad modul som möjligt. Det här betyder kort sagt att modulen ska kunna dra så mycket nytta som möjligt av vad DotNetNuke redan erbjuder. I och med detta kommer examensarbetet att gå hand i hand med hur man utvecklar en modul för DotNetNuke.

Exempel på bra integration är installations- och avinstallationsskript för de tabeller och procedurer som troligtvis behöver skapas i databasen när man installerar en modul via DotNetNukes gränssnitt. Ett annat exempel är lokalisering, som även byggs in för modulen. Med lokalisering underlättar man t.ex. översättningar till andra språk genom att sammanställa allt som kan behöva översättas, till en fil per språk /9/. I DotNetNukes gränssnitt kan sedan en användare med administratörsrättigheter enkelt översätta något till ett annat språk, såvida utvecklaren för modulen har angivit vilka texter och ord det ska gälla.

Förutom de grundläggande egenskaperna som att visa, redigera och lägga till dokument ska även versionshantering och arbetsflöden för dokumenten implementeras. Ett dokument ska inte direkt gå att radera utan i stället tillkommer en ny version vid ändringar. Alla versioner ska sparas i databasen för att göra det möjligt att återgå till en tidigare version.

Den största utmaningen och tyngdpunkten i detta arbete ligger i att implementera arbetsflöden för dokumenten. I kapitlet problemprecisering tas en närmare titt på hur arbetsflödet samt versionshanteringen ska fungera.

1.5 Arbetets upplägg

Detta examensarbete kommer inte att fokusera på den grafiska utformningen av modulen, då den inte behöver vara slutgiltig. Att skapa någon form av vy eller sida var man kan mata in och spara data från olika fält till en databas är oftast inget större problem och kommer därför inte att tas med. Detta betyder i sin tur att läsaren behöver ha någon form av kunskap vad gäller programmering i C# och webbutveckling i ASP.NET. Däremot ges kodexempel på någon rad för de klasser, egenskaper och metoder som finns.

Kapitel 2 innehåller information om de program och utvecklingsverktyg som använts. I problempreciseringen tas en kort men grundlig genomgång av hur de två huvuduppgifterna, arbetsflöde och versionshantering, ska fungera. I kapitel 4 framförs en kort genomgång för installeringen av DotNetNuke, vilket rekommenderas för nya modulutvecklare och för att kontinuerligt kunna testköra sin modul. I kapitlet utförande kommer utvecklandet av denna modul att gå hand i hand med den allmänna praxis som finns för modulutveckling i DotNetNuke. Därefter presenteras resultatet av arbetet och slutligen avslutas examensarbetet med en diskussion.

1.6 Begränsningar

När projektet startades låg den enda begränsningen i att undersöka huruvida det går att använda de dynamiska arbetsflöden som kommer med de andra utgåvorna av DotNetNuke. Modulen borde hinna utvecklas så långt att man kan dra en slutsats om användningen av dessa dynamiska arbetsflöden är möjligt innan prövotiden på 30 dagar går ut. Prövotiden har höjts till 60 dagar från och med version 5.5.

2 Program och utvecklingsverktyg

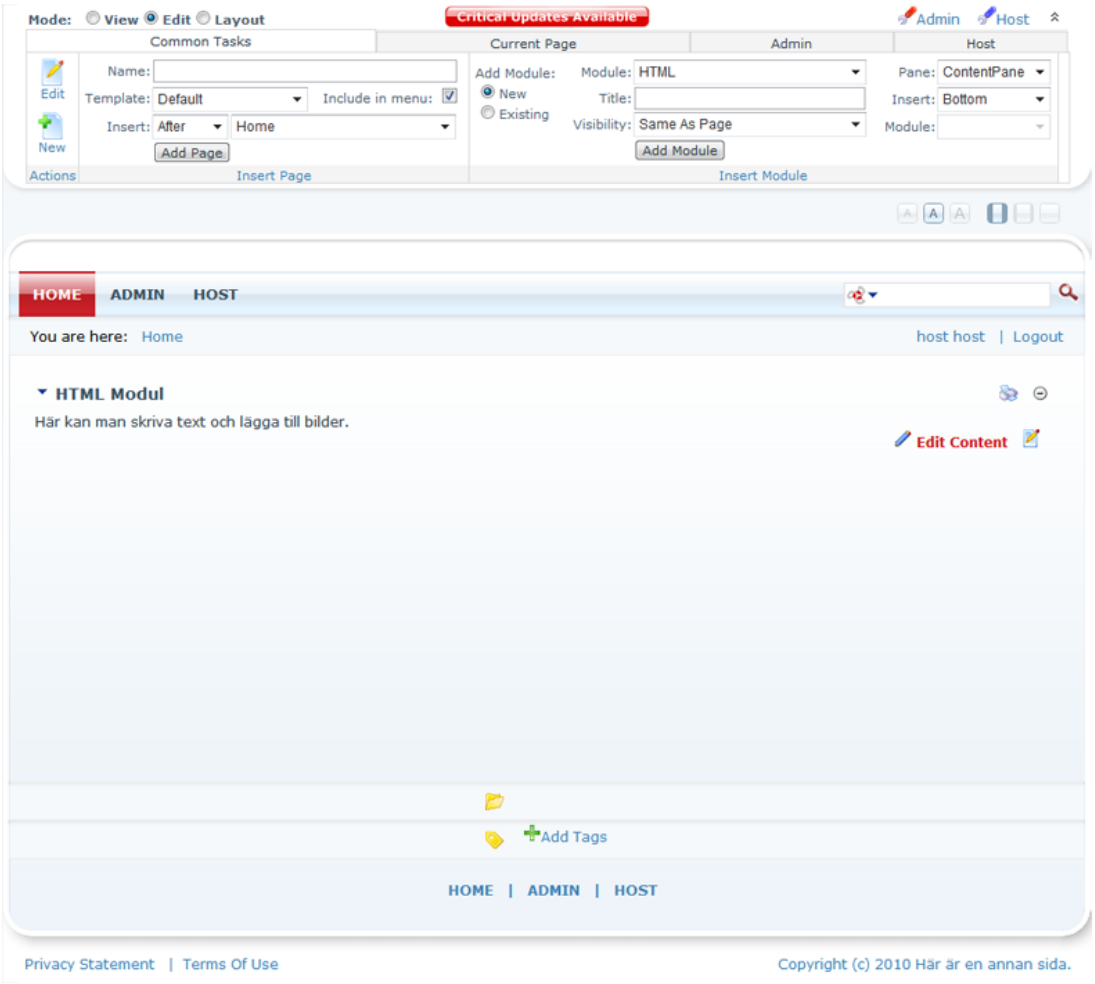
Det språk och utvecklingsverktyg som använts är Visual C# i ASP.NET. DotNetNuke är skriven i Visual Basic men klarar även av C#. C# valdes som språk p.g.a. egen kunskap och erfarenhet. ASP.NET är ett ramverk för att ta fram och skapa dynamiska webbsidor och koden har utformats i Visual Studio 2008 Professional Edition, som är en programutvecklingsmiljö från Microsoft. För att spara undan data används Microsoft SQL Server 2008 R2.

2.1 Vad är DotNetNuke?

DotNetNuke är den ledande webbaserade innehållshanteringsplattformen för Microsoft .NET (uttalas dotnet). DotNetNuke kan användas som webbinnehållshanteringssystem för enkla webbsidor eller som ett gediget ramverk för applikationsutveckling, vilket möjliggör för företag att snabbt bygga upp och lägga till funktionsrika, interaktiva webbsidor och applikationer i Microsoft .NET.

Ett menydrivet gränssnitt tillåter användare att enkelt skapa nya sidor eller utöka funktionaliteten och egenskaperna hos deras existerande webbsidor, när den används som webbinnehållshanteringssystem.

Ett öppet gränssnitt för applikationsprogrammering och tillgängligheten för över 8 000 tredjepartstillägg tillåter professionella webbutvecklare att skapa komplexa webbsidor för krävande applikationer. Att installera nya moduler eller skal (eng. skin) går snabbt och enkelt och tillåter användare att endast inom några minuter lägga till ny funktionalitet på deras webbsidor (figur 1).



Figur 1. En vy i DotNetNuke och för host i edit-läge.

DotNetNuke är världens mest använda plattform för skapande av webblösningar på Microsoft .NET. DotNetNuke driver över 600 000 portaler, intranät, extranät och allmänna webbsidor. En stor grupp med över 800 000 registrerade användare stöder plattformen. /20/

DotNetNuke utvecklades ursprungligen ur ett annat projekt, IBuySpy Workshop. IBuySpy Workshop utvecklades i sin tur av Shaun Walker som en förbättring till IBuySpy portalen, som startades som en testapplikation för .NET ramverket. De tidiga utgåvorna av DotNetNuke släpptes av Shauns företag, Perpetual Motion Inc. och som senare skulle expanderas med hjälp av open source gemenskapen. Namnet DotNetNuke var ett sammanslag av termen .NET och ordet "nuke". /16/ Ordet nuke hade blivit populärt i samband med befintliga ramverk så som PHP-Nuke och PostNuke /5/.

2.2 SQL Server 2008 Express

Microsoft SQL Server är Microsofts databashanterare. Liksom de flesta andra databashanterare är den av relationstyp med SQL (Structured Query Language) som frågespråk. SQL-dialekten som används heter Transact-SQL (T-SQL). /14/ Ett frågespråk är ett språk som man använder för att ställa frågor till en databashanterare, d.v.s. göra sökningar i en databas /3/.

SQL Server installerades med Management Tools. Detta för att kunna hantera data, tabeller och procedurer vid behov. Som färdig modul behövs dock inte detta verktyg utan tabellerna och procedurerna installeras med hjälp av ett skript som kommer att följa med modulen.

SQL Server valdes på basen av tidigare kunskap samt för att den är enkel att hantera och har mångsidig funktionalitet. Express versionen av SQL Server är i sin tur gratis. En begränsning i denna version är exempelvis att den endast tillåter fem samtidiga användare. Detta påverkar dock inte projektet då det endast finns en användare, vilket används av DotNetNuke.

2.3 Visual Studio 2008 Professional Edition

Microsoft Visual Studio är en avancerad programutvecklingsmiljö från Microsoft. Med Visual Studio kan man utveckla både PC-baserade applikationer för Microsoft Windows samt internetanpassade distribuerade applikationer. I Visual Studio ingår bland annat kompilatorer för språken Visual Basic, Visual C++, Visual C#, Visual J# och Visual F#. /21/ En kompilator är ett program som översätter skriven kod i ett högnivåspråk, som t.ex. C# eller Java, till maskinkod /17/.

2.4 Programmeringsspråket Visual C-Sharp

Visual C# (uttalas C-sharp) är ett programmeringsspråk som är utformad för att skapa en mängd olika applikationer som körs på .NET ramverket. C# är enkel, typsäker och objektorienterad. Det C# möjliggör, gentemot t.ex. C++, är snabbare programutveckling samtidigt som det bibehåller finesserna av C-liknande språk.

Visual C# är en implementation av Microsofts C# språk. Visual Studio stöder Visual C# med en kodredigerare, kompilator, projektmallar, designers, en kraftfull och

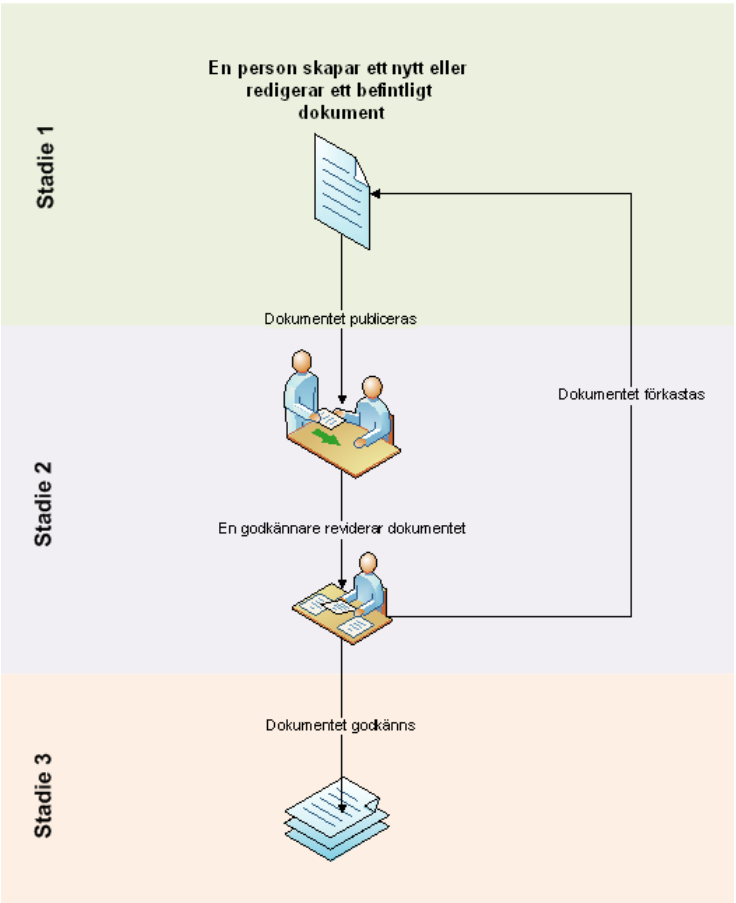
lättanvänd debugger samt andra verktyg. .NET ramverkets klassbibliotek ger tillgång till många operativsystemstjänster och andra användbara, väl utformade klasser som märkbart snabbar upp utvecklingscykeln /19/.

3 Problemprecisering

De två viktigaste tyngdpunkterna i det praktiska arbetet låg i att skapa ett så kallat arbetsflöde samt versionshantering. DotNetNuke finns i flera utgåvor, där den enklaste får användas fritt. I de mer avancerade utgåvorna finns redan dynamiska arbetsflöden inbyggt. Som en mindre del av projektet hörde att reda ut om dessa gick att utnyttja.

3.1 Arbetsflöden

Det arbetsflöde som planerats för detta ändamål är uppdelat i tre stadier (figur 2). Då en användare skapar ett nytt dokument är dokumentet i stadie ett, ett utkast. Om användaren direkt, eller senare, väljer att publicera dokumentet ska det skickas vidare för granskning, stadie två. När av användaren vald granskare väljer att godkänna dokumentet blir det slutligen publicerat och hamnar i stadie tre. Dokumentet kan även förkastas av granskaren varvid det återgår till stadie ett. Arbetsflöden kan för övrigt innehålla flera stadier men för hantering av kvalitetsdokument behövs endast tre.



Figur 2. Arbetsflöde och stadier.

3.2 Versionshantering

Om en användare väljer att redigera ett befintligt dokument så går dokumentet igenom hela processen för arbetsflödet men som en ny version. Men det är först när det blir godkänt som det ska ersätta den tidigare versionen. Vidare ska även användaren kunna granska alla tidigare versioner av ett dokument samt ges möjlighet att återgå till en tidigare version. Om man återgår till en tidigare version så går dokumentet igenom stadierna för arbetsflödet och även här som en ny version.

4 Konfigurationer och installationer

Att installera DotNetNuke för första gången kan för många nya användare kännas komplicerat. DotNetNuke behövs inte för att utveckla en modul, men är nödvändig då man vill kunna felsöka eller se hur modulen beter sig. Det finns tre viktiga punkter vad gäller installeringen av DotNetNuke. Till att börja med bör man skapa ett login konto för DotNetNuke och för den databas man kommer att använda. DotNetNuke är en dynamisk webbapplikation och en SQL Server databas används för att spara den information som behövs för att använda DotNetNuke. Det går att använda DotNetNuke med dynamiskt bundna databaser i SQL Server 2005. Detta är dock inte att rekommendera då framtida underhåll av databasen kan bli komplicerade och en korrekt konfiguration inte tar mycket extra tid /10/.

Den andra punkten gäller filrättigheter för IIS arbetsprocessen och tillåter DotNetNuke att skapa och radera mappar eller att kopiera några eller alla filer som behövs. Denna "användare" bör få fulla rättigheter till alla filer där DotNetNuke har packats upp. Man ska även se till att dessa filer inte är skrivskyddade. /11/

Det som återstår innan man kan exekverar installationen är att konfigurera Internet Information Services (IIS). Här gäller lite olika tillvägagångssätt beroende av vilken version man har. En beskrivning av hur man närmare konfigurerar och installerar DotNetNuke finns till exempel på den officiella hemsidan /4/.

4.1 Utvecklingsmallar

För att enklast komma igång finns det färdiga mallar för att börja utveckla en modul, vilket har använts för detta projekt. Dessa mallar finner man som ett startpaket på sidan www.codeplex.com. Mallarna finns för närvarande endast för språken Visual Basic och C#.

5 Utförande

Fokuset i detta kapitel sätts på allmän praxis för modulprogrammering samt de delar som togs upp i problempreciseringen.

5.1 Krav på modulen

I stället för att använda olika sidor för att visa formulär och data så används användarkontroller, Web User Controls. Genom att ha olika kontroller kan en administratör enkelt ange läs och skriv rättigheter till dem även om de finns på en och samma sida.

5.1.1 Databasstruktur

I databasdiagrammet i bilaga 1 visas strukturen hos modulens viktigaste tabeller. Diagrammet är en första version och kommer troligtvis att ändras. Som första version har t.ex. tillräckligt med fält för tabellerna tagits med för att slippa mindre ändringar i databasen under utvecklingen. Det som slutligen anses vara onödigt tas sedan bort. För resultatet som uppnåtts har dock denna struktur använts. För många av tabellerna har en referens satts till PortalID för DotNetNukes Portal. Detta för att kunna skilja åt data i tabellerna om det finns flera portaler, vilket är fallet om företaget även erbjuder lösningen som en tjänst på deras egen server.

5.1.2 Användarkontroller

För modulen krävs endast tre användarkontroller. Användarkontroller kan ses som en sida men är egentligen en egengjord kontroll, som t.ex. vilken annan kontroll som helst. Den första kontrollen ska dels visa alla publicerade dokument men även dokument som väntar på en användares godkännande samt dokument som för användaren ännu är sparade som utkast. Den andra kontrollen ska innehålla alla textboxar och fält för att visa, skapa eller redigera ett dokument. Här ska även dokument- och versionshistorik visas. Den tredje och sista kontrollen ska vara för modulen specifika inställningar. Här ska en administrator t.ex. ha möjlighet att lägga till de referensstandarder, avdelningar eller processer, som företaget använder sig av.

5.1.3 Dataisolering

Dataisolering används för att hantera och ge tillgång till data i olika nivåer, eller lager. Isolering på portalnivå är den högsta, där alla instanser av en modul har tillgång till datat. Denna typ av isolering kommer att användas för denna modul, då alla dokument ska skiljas åt portalsvis. Det finns även isolering på lägre nivåer och ett exempel är isolering på modulnivå. Det är den mest använda och informationen finns tillgänglig för endast den modulinstansen.

5.2 Skapande av projekt

När man skapar en modul där man använder sig av den färdiga mallen så är det viktigt att man skapar den under DotNetNukes ’desktopmodules’ mapp. DotNetNuke förväntar sig att alla utomstående eller egna moduler ska finnas här. När man skapat projektet med mallen skapas även en del färdigt genererade mappar och filer.

5.2.1 Genererade komponenter

App_LocalResources mappen innehåller .resx-filer för lokalisering och i components-mappen ska all kod som inte är direkt knuten till kontrollernas front-end vara. Documentation-mappen finns till endast som information och många väljer att ta bort innehållet i mappen då det kan förorsaka förvirring mellan de färdigt genererade filerna och de egna. Grundfilerna som genererats är tre till typen. Den första är .SqlDataProvider. Dessa typer av filer är skript som körs när en modul installeras. Den andra typen är .dnn filen. Denna fil är ett manifest som berättar för DotNetNuke den nödvändiga informationen om modulen så som vilka användarkontroller den har. Den tredje och sista typen är .ascx. Dessa är de användarkontroller som används för att visa modulens innehåll till användare. /12/

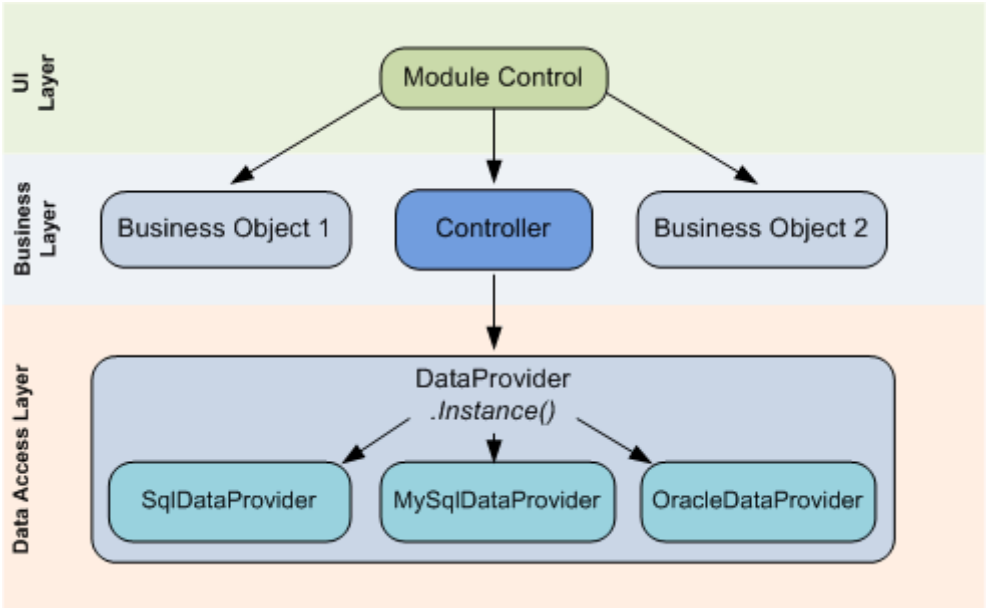
En begränsad överblick av projektets klasser och filer kan ses i bilaga 2.

5.3 Skapande av modulens back-end

En back-end är en programdel eller ett fristående program som sköter hämtning och lämning av data eller endast lagring av data. Det matchas oftast av en programdel eller ett program som sköter interaktionen med användaren, dvs. front-end. Ofta finns

även en mellannivå som hanterar affärslogiken eller bearbetningsdelen från den bakomliggande modellen för applikationen i sin helhet. /2/

Modulstrukturen för en modul i DotNetNuke ser ut som i figur 3 och är en del av den praxis som följs /1/.



Figur 3. Allmän praxis över modulstruktur för en modul i DotNetNuke.

På detta sätt går det exempelvis enkelt att migrera till andra datakällor. För mindre implementationer där t.ex. endast få metoder används, kan detta upplägg tänkas lämnas bort.

5.3.1 DataProvider.cs

DataProvider klassen som skapas är en abstrakt klass och som definierar de metoder som måste implementeras av den specifika konkreta klassen, i detta fall SqlDataProvider.cs. Genom att ha en abstrakt klass och abstrakta metoder försäkrar man att alla metoder kommer att vara stödda, även om en annan provider implementeras.

En metod i den abstrakta klassen som uppdaterar en referensstandard ser ut så här:

```
public abstract void UpdateRefStand(int PortalId, int RefStandId,
string Type);
```

Figur 4. Abstrakt metod för att lägga till en referensstandard.

5.3.2 SqlDataProvider.cs

SqlDataProvider klassen är i sin tur lite mer komplex då den implementerar de behövliga metoderna samt hämtar information om inställningar från web.config-filen. Web.config filen innehåller t.ex. information om hur DotNetNuke ansluter till databasen. Värt att notera är att SqlDataProvider ärver från klassen DataProvider. I och med att SqlDataProvider ärver från DataProvider så är det här som de behövliga metoderna ska köras över, eng. override.

Exemplet ovan skulle för SqlDataProvidern se ut enligt:

```
public override void UpdateRefStand(int PortalId, int RefStandId,
string Type)
{
    SqlHelper.ExecuteNonQuery(ConnectionString,
    GetFullyQualifiedName("UpdateRefStand"),PortalId,RefStandId,
    Type);
}
```

Figur 5. Den metod som kör över den abstrakta och beskriver hur data ska hämtas från databasen.

Det som eventuellt avviker sig i denna kod är SqlHelper. Genom att använda SqlHelper, som finns i Microsoft Application Blocks Data komponenten, görs exekveringen av proceduren snabbt och enkelt. GetFullyQualifiedName() är en metod som sätter till ett prefix till procedurnamnet, om man så angivit ett vid installationen av DotNetNuke. Detta för att hålla reda på tabeller och procedurer.

5.3.3 xx.xx.xx.SqlDataProvider-filen

Denna och xx.xx.xx.Uninstall.SqlDataProvider-filen ska inte förväxlas med de som har .cs som extension då de är de egentliga .NET kodmetoderna som tillhandahåller åtkomst till databasen. X:ena i filnamnen representerar versioner av modulen. Filerna används för att skapa tabeller och procedurer under installation eller uppdatering. Skriptena liknar till stor del SQL:s frågespråk. Hit kommer alltså text som ska sätta in tabellernas definition samt nödvändiga procedurer. Nedan är ett förkortat exempel för tabellen med referensstandarder.

```
CREATE TABLE {objectQualifier}RefStands
(
    [PortalID] INT NOT NULL,
    [RefStandID] INT NOT NULL IDENTITY(1,1),
    [Type] NVARCHAR(20) NOT NULL,
    [CreatedByUserID] INT,
    [CreatedOnDate] DATETIME
)
ALTER TABLE  {objectQualifier} RefStands ADD CONSTRAINT
    [PK_{objectQualifier} RefStands PRIMARY KEY CLUSTERED
    ([RefStandID])
END
GO
```

Figur 6. Skript för att skapa tabellen för referensstandarder i databasen.

Genom att använda en {objectQualifier} token så ersätts det med det rätta prefixet i klassen SqlDataProvider.cs och med metoden GetFullyQualifiedName(string name), där strängen "name" är namnet på den procedur man vill åt i databasen.

5.3.4 xx.xx.xx.Uninstall.SqlDataProvider-filen

Uninstall.SqlDataProvider fungerar på samma vis som SqlDataProvider men körs när man avinstallerar en modul, dvs. tar bort tabeller och procedurer. Värt att tänka på är att se till att allt körs i rätt ordning, t.ex. så att man raderar en foreign key före tabellen. Om inte allt körs i rätt ordning så stoppas skriptet där ett fel uppstår och data lämnar kvar i databasen.

5.3.5 Information Object-klasserna

Information Object klasserna används för att sätta in det data man hämtar från databasen och för att sedan underlätta den bindning i front-end som oftast görs. Klassen ser helt vanlig ut med variabeldeklarationer och publika get- och set-egenskaper. Det som är viktigt är att namnen för egenskaperna stämmer exakt överens med de kolumner man hämtar från databasen. Exempel för klassen med referensstandarder är:

```
private int _RefStandId;


public int RefStandId
{
   get { return _RefStandId; }
   set { _RefStandId = value; }
}
```

Figur 7. Exempel på variabel deklaration och dess set- och get-egenskap.

5.3.6 Kontrollerklasserna

Den underliggande koden för att underlätta dataåtkomst är det sista steget, ur ett .NET-perspektiv, att skapa kontrollerklassen. Kontrollerklassen fungerar som ett mellan lager i en applikation och tillåter åtkomst till de underliggande datametoderna. Vidare så konverterar den det som returneras från databasen till formbara objekt och listor som sedan kan användas av modulen i fråga /13/. För att lägga till en referensstandard så finns en metod AddRefStand(DMModuleRefStandInfo refInfo), som tar en referensstandards informationsobjekt som inparameter. Koden för denna metod visas i figur 8.

```
public void AddRefStand(DMModuleRefStandInfo refInfo)
{
    DataProvider.Instance().AddRefStand(refInfo.PortalId,
      refInfo.Type, refInfo.CreatedByUserId);
}
```

Figur 8. Metod som lägger till en referensstandard.

5.4 Skapande av modulens front-end

Modulens front-end består av de användarkontroller som visas i bilaga 2. De tre andra användarkontrollerna som inte nämnts skapades av mallen och är inte väsentliga men kan användas eller tas bort. Den mark-up kod och koden bakom användarkontrollerna tas inte med då det till största del är kod som genereras när man lägger till kontroller, t.ex. etiketter och textfält. CustomEditor.cs som även syns i bilaga 2 är en texteditor-kontroll som ärvts från Ajax HTML Editor för att kunna bestämma vilka knappar som ska finnas med. I bilaga 3 ges en första version av hur EditDocument.ascx användarkontrollen ser ut. Detta är som tidigare sagts en användarkontroll där man kan lägga till, redigera eller öppna ett dokument.

5.5 Arbetsflöde

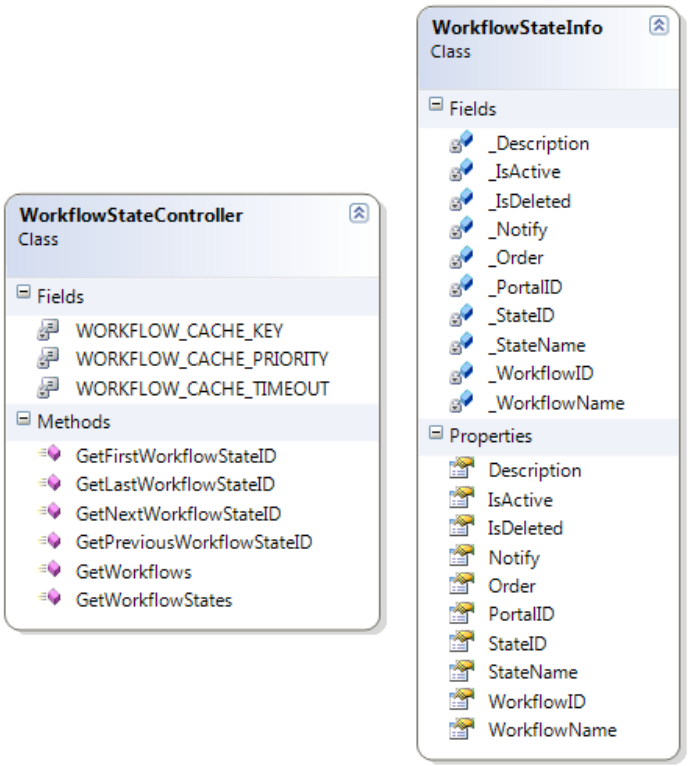
5.5.1 Undersökning av de dynamiska arbetsflödena

De dynamiska arbetsflöden som kommer till från och med den professionella utgåvan av DotNetNuke verkade lovande efter en testkörning av dem i en annan modul. Men efter att ha undersökt källkoden för den modulen och arbetsflödena, kunde man konstatera att det innebar en hel del ändringar i kod för att ens försöka dra nytta av dem. Det största problemet uppenbarade sig dock efter att det framkommit att

arbetsflödena inte direkt kunde tillämpas på en lägre nivå än modulnivå. För projektets modul ska dataisolering på portal- samt dokumentnivå tillämpas. I och med detta fastställdes att ett eget arbetsflöde ska göras.

5.5.2 Arbetsflödets uppbyggnad

För modulen behövs egentligen endast ett arbetsflöde och kan därmed hårdkodas. Men för att hålla kvar möjligheterna att lägga till nya eller ändra ett arbetsflöde så skapades tabeller för dem i databasen. De två tabellerna för arbetsflödet (syns i nere till höger i bilaga 1) liknar till stor del DotNetNukes egna. En referens finns mellan dem som säger att ett arbetsflöde kan ha flera stadier, och skiljs på detta sätt åt ifall det i framtiden läggs till flera. "Information Object"-klassen för arbetsflöde(na), WorkflowStateInfo.cs, innehåller endast de nödvändiga deklarationerna samt get- och set-egenskaper. Kontrollerklassen, WorkflowStateController.cs, är den som via Data Access Layer hämtar nödvändig data. Klassen innehåller de nödvändiga metoder som behövs. Metoderna och dess kod är i sig relativt enkla och arbetsflödets business layer, dvs. "Information object-" och kontrollerklassen, visas med ett klassdiagram i figur 9.



Figur 9. Arbetsflödets business layer.

5.5.3 Implementation

I koden bakom användarkontrollen EditDocument.ascx så sätts en variabel WorkflowID till ett värde med hjälp av metoden GetWorkflows(int PortalId). Som första version av modulen kommer värdet för WorkflowID att alltid bli 1 och fungerar i och med att det för tillfället endast finns ett enda arbetsflöde. Om det i framtiden läggs till möjligheten att ha flera arbetsflöden så borde detta skötas på ett annat sätt. Till exempel kunde en administrator, i användarkontrollen ViewSettings, välja vilket arbetsflöde som ska användas. I användarkontrollens OnLoad() hämtas sedan de stadier som tillhör det arbetsflödet med GetWorkflowStates(int WorflowID). Namnet på arbetsflödet och stadierna sätts, under sektionen dokumenthistorik, i diverse etiketter. Denna sektion hörde från början inte till projektet men togs med för att kunna ge användaren ytterligare information om vad och när någonting gjorts med dokumentet. Sektionens data hämtas från tabellen DocumentLog där alla ändringar som gör att stadiet ändras, loggas. När användaren slutligen väljer att publicera dokumentet så kallas metoden GetNextWorkflowStateID och det ID:et sätts för dokumentet.

5.6 Versionshantering

Som tidigare nämnts så ska inte några publicerade dokument, d.v.s. dokument som är i ett annat stadie än stadie 1, gå att radera. I och med att alla versioner av ett dokument sparas i databasen kan man enkelt hämta och visa äldre versioner på basen av DocumentID.

5.6.1 Implementation

För att visa versioner av ett dokument så används GetDocumentVersions(int PortalId, int DocumentId). Denna metod returnerar en lista och listan sätts som datakälla för en datagrid komponent. Proceduren som finns i databasen hämtar helt enkelt alla dokument som hör till den rätta portalen, har samma DocumentID som det öppnade dokumentet och slutligen där IsPublished = 1, d.v.s. alla publicerade dokument.

För att återgå till en tidigare version kan man trycka på "roll back"-knappen varvid en ny version av den gamla skapas. Knappen kan ses under sektionen "Version history" i bilaga 3.

5.7 Testning

Testningen utförs i tre stadier. Det första stadiet är den testning jag som utvecklare kontinuerligt utför för att se till att de resultat som önskas uppnås. I det andra stadiet kommer modulen att grundligt testköras och med mer verklig data på en server i företaget. I det tredje stadiet tillägnas en testpilot (företag). Före den sista testningen ska en utförligare kontroll av installations- och avinstallationsskripten, samt diverse kontroller för inmatning av rätt data, göras. Även layout och utseende kommer att ändras för att ge ett mer professionellt intryck.

6 Resultat

Det som i detta skede uppnåtts är en modul med nödvändiga användarkontroller för visning och inmatning av dokument. Huvudpunkterna arbetsflöde och versionshantering infördes och med goda resultat. Det som senare kan ändras är möjligheten till flera arbetsflöden eller arbetsflödesstadier. Modulen är dock inte färdig utan kommer att vara mer eller mindre under ständig utveckling, då så kallade features eller tilläggsfunktioner läggs till och några av dessa har redan implementerats och återspeglas i bilagorna.

7 Diskussion

Genom att grundligt ha studerat modulprogrammering i DotNetNuke och utan desto större insikt i andra innehållshanteringssystem, kan jag säga att det är ett värdigt och omfattande system som enkelt går att skräddarsy helt enligt ens behov. Arbetet med projektet har framskridit i stadig takt men i och med att jag har mer eller mindre jobbat ensam, samt då DotNetNuke var något nytt för mig, har det dragit ut något på tiden. Å andra sidan vann jag en del då Ibiworks redan har ett liknande system och vet hurudan funktionalitet modulen ska ha.

Jag skulle utan tvekan rekommendera DotNetNuke som innehållshanteringssystem om någon frågade. Förutom enkelheten i att utveckla moduler så har företaget en oerhört god kundservice och support samt en stor gemenskap med andra utvecklare med vilka man kan rådgöra med.

Exempel på vidareutveckling vore tilläggsmoduler som går hand i hand med kvalitetshandböcker och -dokument, t.ex. en modul för avvikelsehantering, reklamationshantering och auditering.

Företaget, och även jag, har varit nöjd med hur utvecklingen av modulen gått och vad gäller modulens framtidsutsikter så återstår att färdigställa det sista så att en pilottestning kan komma igång.

8 Källförteckning

/1/ *Creating a DNN Module and Understanding DNN Architectural Approach.* The Code Project http://www.codeproject.com/KB/applications/LCTDNNModDev.aspx (hämtat: 9.11.2010)

/2/ Dahl, S. (2009) *Vad är en back-end*. Kungliga tekniska högskolan - CSC http://www.csc.kth.se/utbildning/kth/kurser/DD2471/moddb09/lankar/f11.pdf (hämtat: 17.10.2010)

/3/ *Databaser: Introduktion till frågespråket SQL*. Databasteknik. http://www.databasteknik.se/webbkursen/sql/index.html (hämtat: 6.9.2010)

/4/ *DotNetNuke.* DotNetNuke http://www.dotnetnuke.com/ (hämtat: 13.11.2010)

/5/ *DotNetNuke Community Edition*. DotNetNuke http://www.dotnetnuke.com/Products/CommunityEdition/tabid/2006/Default.aspx (hämtat: 13.11.2010)

/6/ *Drupal Licensing FAQ published | drupal.org.* Drupal http://drupal.org/node/272652 (hämtat: 28.10.2010)

/7/ *Ibiworks Ab Oy - Företaget.* Ibiworks. http://www.ibiworks.com/dnn/SV/Företaget.aspx (hämtat: 6.9.2010)

/8/ *IBM Lotus Notes.* IBM. http://www-01.ibm.com/software/lotus/products/notes (hämtat: 13.11.2010)

/9/ *Language localization.* Wikipedia http://en.wikipedia.org/wiki/Language_localisation (hämtat: 20.9.2010)

/10/ Sellers Mitchel. (2009). *DotNetNuke Module Programming.* Indianapolis: Wiley Publishing, Inc, 15-18

/11/ Sellers Mitchel. (2009). *DotNetNuke Module Programming.* Indianapolis: Wiley Publishing, Inc, 19-20

/12/ Sellers Mitchel. (2009). *DotNetNuke Module Programming.* Indianapolis: Wiley Publishing, Inc, 53

/13/ Sellers Mitchel. (2009). *DotNetNuke Module Programming.* Indianapolis: Wiley Publishing, Inc, 89

/14/ *SQL Server 2008 R2 Express.* Microsoft. http://www.microsoft.com/express/Database/ (hämtat: 13.11.2010)

/15/ *Vad är kvalitetssystem?* Bokföringstips http://www.bokforingstips.se/artikel/ekonomistyrning/kvalitetssystem.aspx (hämtat: 28.10.2010)

/16/ Walker S, Scarbeau B, Hardy D, Schultes S & Morgan R. (2009). *DotNetNuke 5 - Open Source Web Application Framework for ASP.NET* Indianapolis: Wiley Publishing, Inc, 2-8

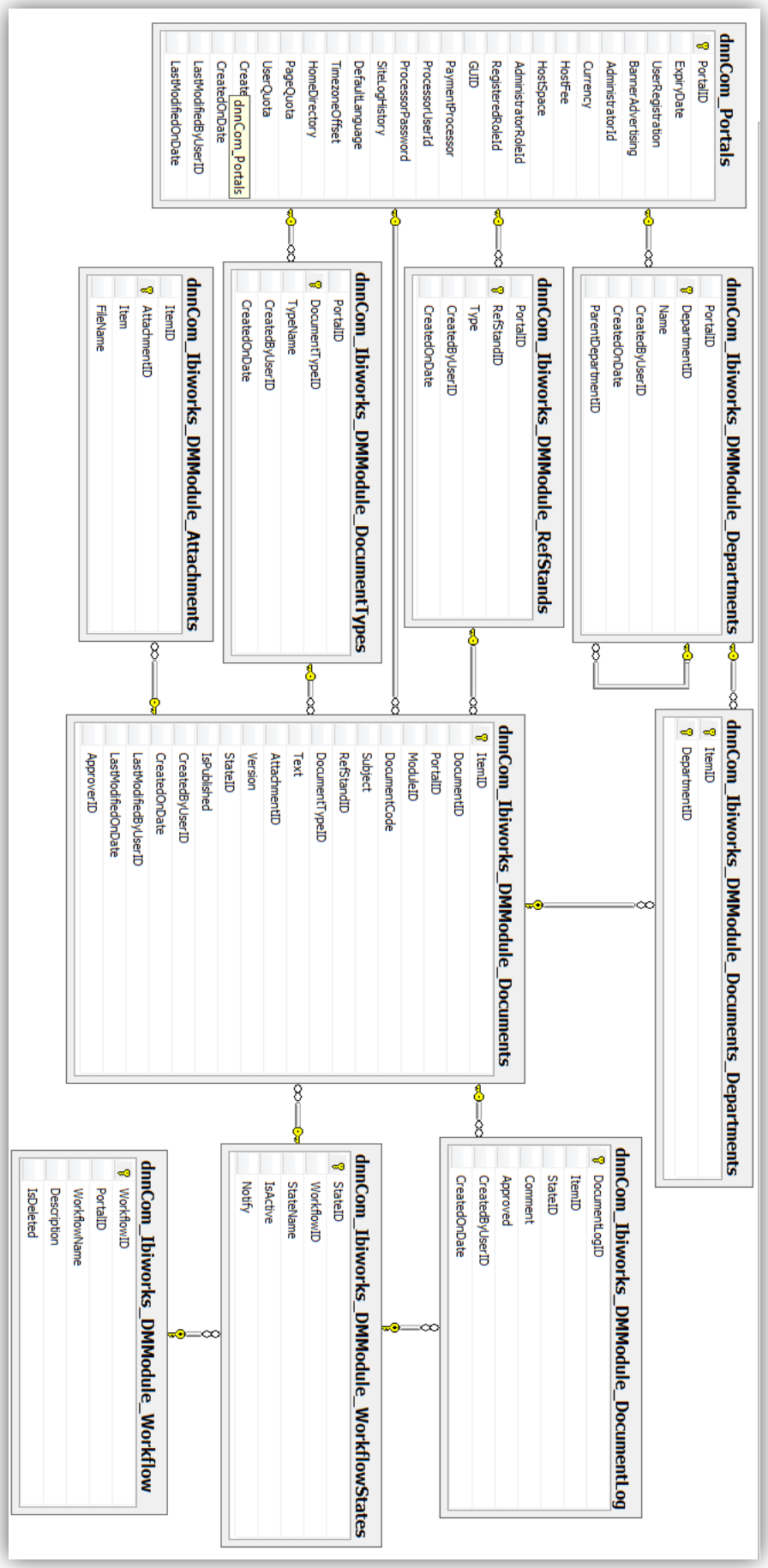
/17/ *What is a Compiler*. University of Texas at Arlington http://lambda.uta.edu/cse5317/notes/node3.html (hämtat: 20.9.2010)

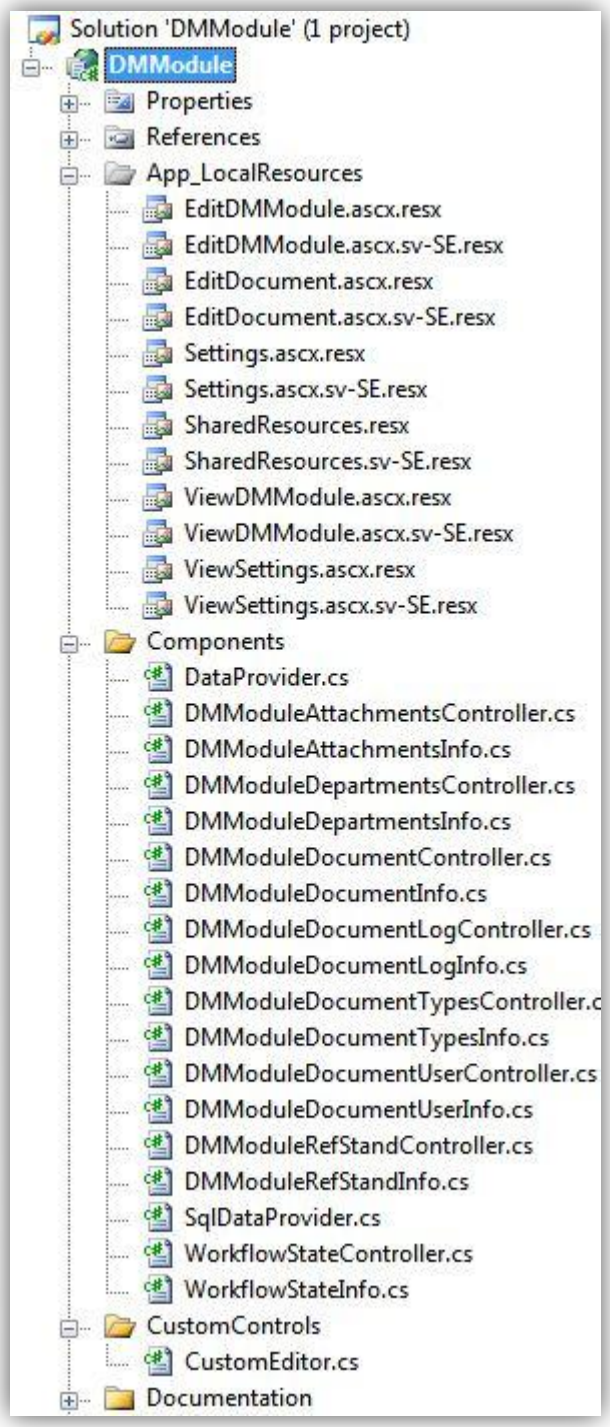
/18/ *What is Document Management?* Aiim http://www.aiim.org/What-is-Document-Management (hämtat: 9.11.2010)

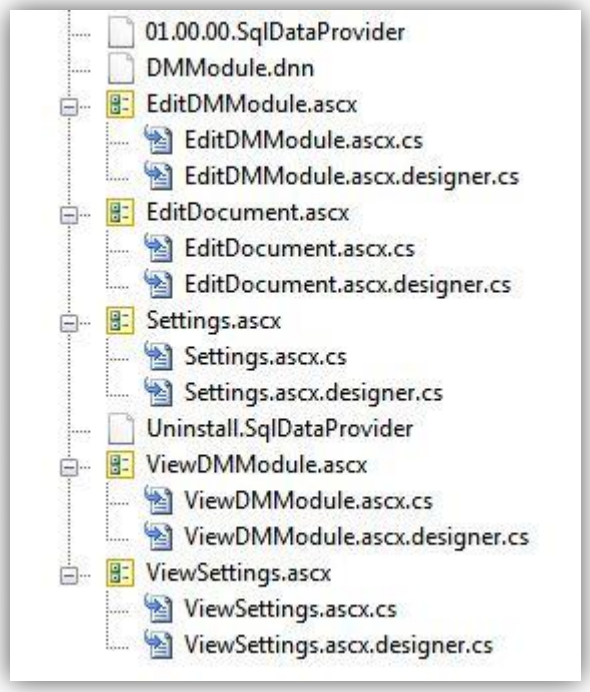
/19/ *Visual C#.* Microsoft http://msdn.microsoft.com/en-us/library/kx37x362.aspx (hämtat: 6.9.2010)

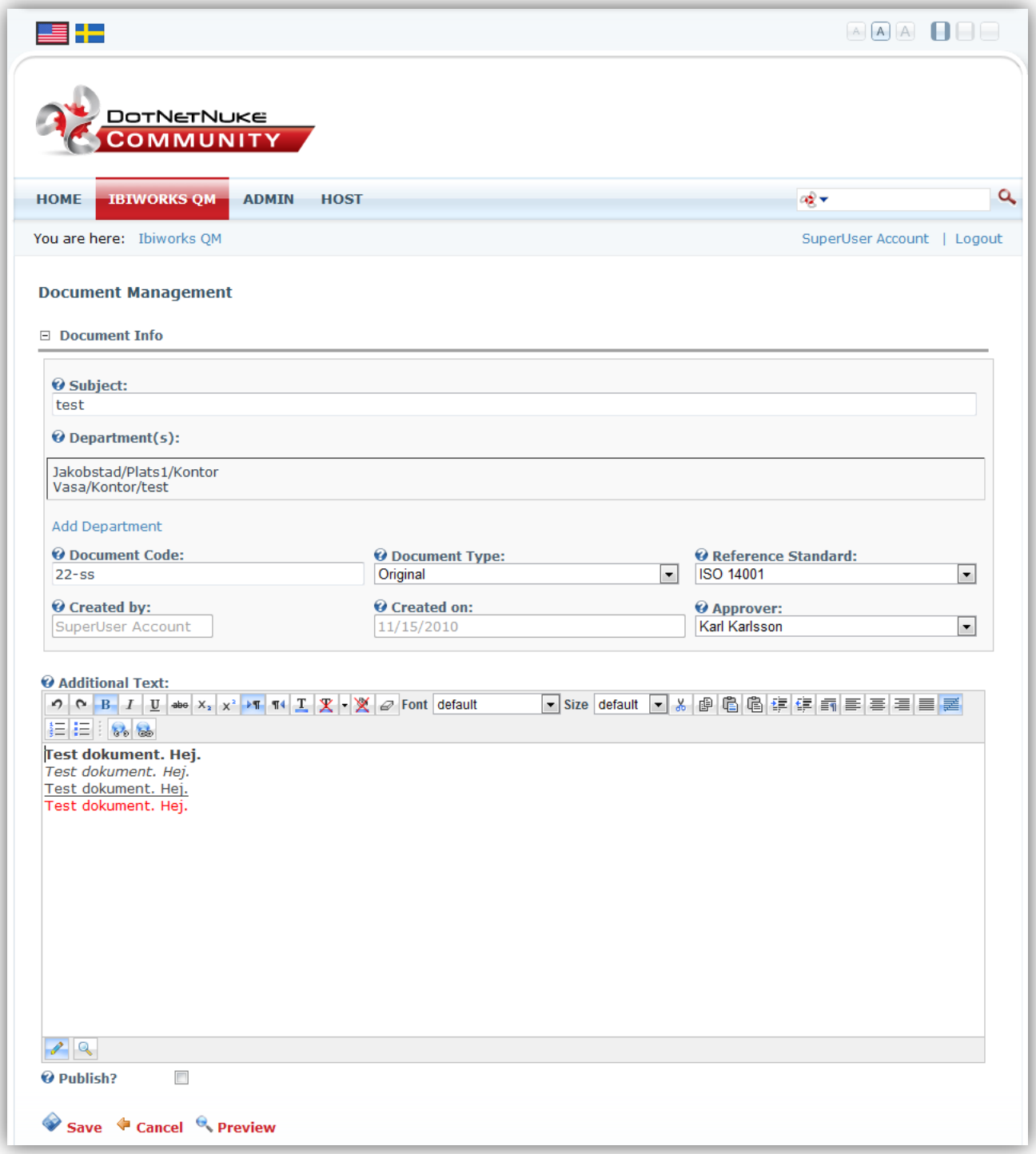
/20/ *Overview*. DotNetNuke. http://www.dotnetnuke.com/Products/Overview/tabid/1206/Default.aspx (hämtat: 6.9.2010)

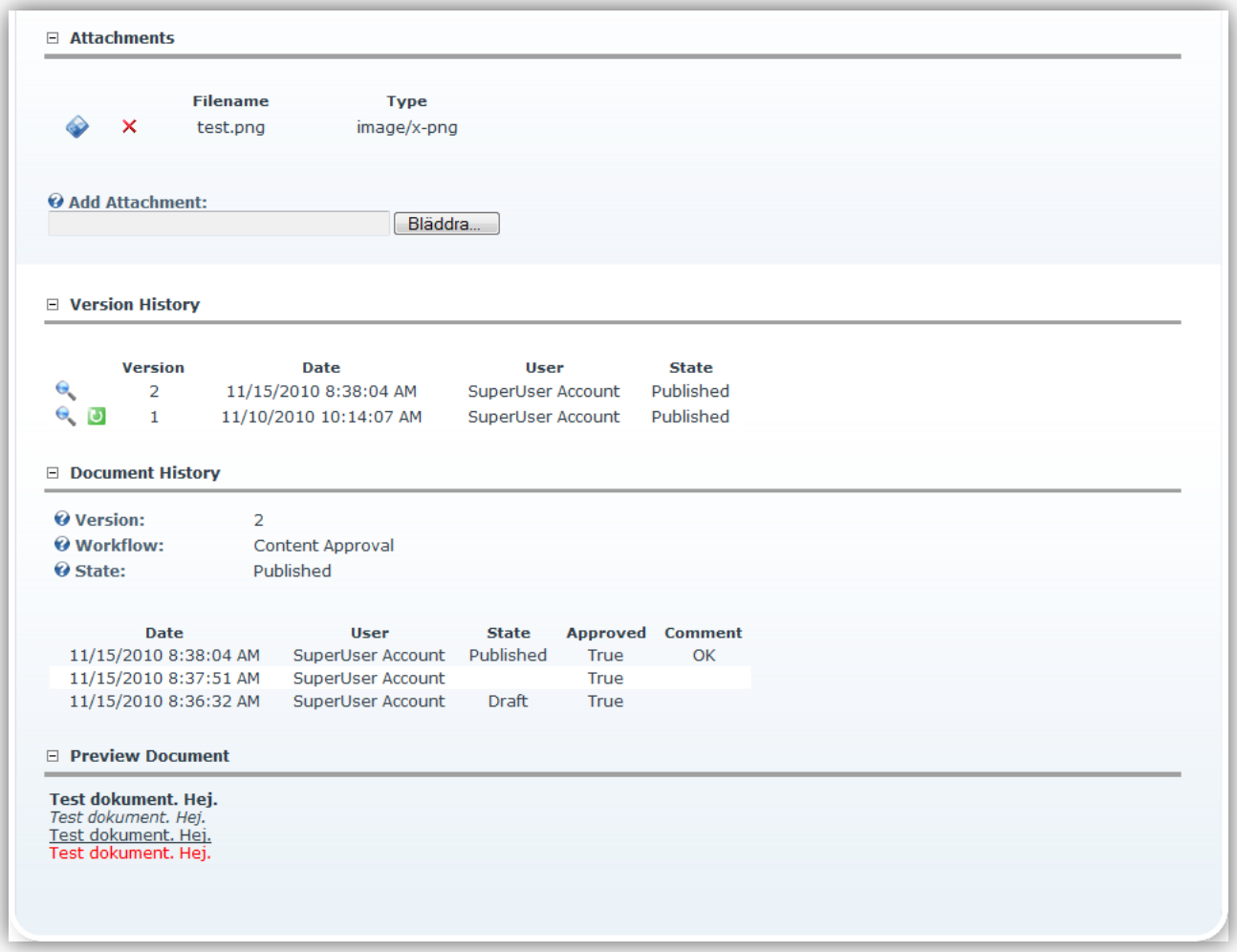
/21/ *Visual Studio Professional 2008*. Microsoft. http://msdn.microsoft.com/en-us/vstudio/aa700830.aspx (hämtat: 13.11.2010)


dnnCom_Portals
PortalID
ExpiryDate
UserRegistration
BannerAdvertising
AdministratorId
Currency
HostFee
HostSpace
AdministratorRoleId
RegisteredRoleId
GUID
PaymentProcessor
ProcessorUserId
ProcessorPassword
SiteLogHistory
DefaultLanguage
TimezoneOffset
HomeDirectory
PageQuota
UserQuota
CreatedByUserID
CreatedOnDate
LastModifiedByUserID
LastModifiedOnDate
dnnCom_Ibiworks_DMModule_Departments
PortalID
DepartmentID
Name
CreatedByUserID
CreatedOnDate
ParentDepartmentID
dnnCom_Ibiworks_DMModule_Documents_Departments
ItemID
DepartmentID
dnnCom_Ibiworks_DMModule_RefStands
PortalID
RefStandID
Type
CreatedByUserID
CreatedOnDate
dnnCom_Ibiworks_DMModule_DocumentTypes
PortalID
DocumentTypeID
TypeName
CreatedByUserID
CreatedOnDate
dnnCom_Ibiworks_DMModule_Attachments
ItemID
AttachmentID
Item
FileName
dnnCom_Ibiworks_DMModule_Documents
ItemID
DocumentID
PortalID
ModuleID
DocumentCode
Subject
RefStandID
DocumentTypeID
Text
AttachmentID
Version
StateID
IsPublished
CreatedByUserID
CreatedOnDate
LastModifiedByUserID
LastModifiedOnDate
ApproverID
dnnCom_Ibiworks_DMModule_DocumentLog
DocumentLogID
ItemID
StateID
Comment
Approved
CreatedByUserID
CreatedOnDate
dnnCom_Ibiworks_DMModule_WorkflowStates
StateID
WorkflowID
StateName
IsActive
Notify
dnnCom_Ibiworks_DMModule_Workflow
WorkflowID
PortalID
WorkflowName
Description
IsDeleted


Solution 'DMModule' (1 project)
DMModule
Properties
References
App_LocalResources
EditDMModule.ascx.resx
EditDMModule.ascx.sv-SE.resx
EditDocument.ascx.resx
EditDocument.ascx.sv-SE.resx
Settings.ascx.resx
Settings.ascx.sv-SE.resx
SharedResources.resx
SharedResources.sv-SE.resx
ViewDMModule.ascx.resx
ViewDMModule.ascx.sv-SE.resx
ViewSettings.ascx.resx
ViewSettings.ascx.sv-SE.resx
Components
DataProvider.cs
DMModuleAttachmentsController.cs
DMModuleAttachmentsInfo.cs
DMModuleDepartmentsController.cs
DMModuleDepartmentsInfo.cs
DMModuleDocumentController.cs
DMModuleDocumentInfo.cs
DMModuleDocumentLogController.cs
DMModuleDocumentLogInfo.cs
DMModuleDocumentTypesController.c
DMModuleDocumentTypesInfo.cs
DMModuleDocumentUserController.cs
DMModuleDocumentUserInfo.cs
DMModuleRefStandController.cs
DMModuleRefStandInfo.cs
SqlDataProvider.cs
WorkflowStateController.cs
WorkflowStateInfo.cs
CustomControls
CustomEditor.cs
Documentation


01.00.00.SqlDataProvider
DMModule.dnn
EditDMModule.ascx
EditDMModule.ascx.cs
EditDMModule.ascx.designer.cs
EditDocument.ascx
EditDocument.ascx.cs
EditDocument.ascx.designer.cs
Settings.ascx
Settings.ascx.cs
Settings.ascx.designer.cs
Uninstall.SqlDataProvider
ViewDMModule.ascx
ViewDMModule.ascx.cs
ViewDMModule.ascx.designer.cs
ViewSettings.ascx
ViewSettings.ascx.cs
ViewSettings.ascx.designer.cs


DotNetNuke COMMUNITY
HOME
IBIWORKS QM
ADMIN
HOST
You are here: Ibiworks QM
SuperUser Account | Logout
Document Management
Document Info
Subject:
test
Department(s):
Jakobstad/Plats1/Kontor
Vasa/Kontor/test
Add Department
Document Code:
22-ss
Document Type:
Original
Reference Standard:
ISO 14001
Created by:
SuperUser Account
Created on:
11/15/2010
Approver:
Karl Karlsson
Additional Text:
Font
default
Size
default
Test dokument. Hej.
Test dokument. Hej.
Test dokument. Hej.
Test dokument. Hej.
Publish?
Save
Cancel
Preview



Attachments

	Filename	Type
	test.png	image/x-png

Add Attachment:

Bläddra...

Version History

	Version	Date	User	State
	2	11/15/2010 8:38:04 AM	SuperUser Account	Published
	1	11/10/2010 10:14:07 AM	SuperUser Account	Published

Document History

Version: 2
Workflow: Content Approval
State: Published

Date	User	State	Approved	Comment
11/15/2010 8:38:04 AM	SuperUser Account	Published	True	OK
11/15/2010 8:37:51 AM	SuperUser Account		True	
11/15/2010 8:36:32 AM	SuperUser Account	Draft	True	

Preview Document

Test dokument. Hej.
Test dokument. Hej.
Test dokument. Hej.
Test dokument. Hej.