

Taavi Rouhiainen

Sisällöntuottojärjestelmä EpiServer CMS 6 R2 sovelluskehittäjän näkökulmasta

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Taavi Rouhiainen

Insinööriyö

Tekijä Otsikko Sivumäärä Aika	Taavi Rouhiainen Sisällöntuottojärjestelmä EpiServer CMS 6 R2 sovelluskehittäjän näkökulmasta 43 sivua 19.3.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	ohjelmistotekniikka
Ohjaajat	tiimin vetäjä Mikko Ojamo lehtori Olli Hämäläinen
<p>Insinööriyön aiheena oli sisällöntuottojärjestelmä EpiServer CMS 6 R2 sovelluskehittäjän näkökulmasta. Työn tavoitteena oli luoda selkeä asennus- ja sovelluskehitysohje uusille EpiServerin sovelluskehittäjille.</p> <p>Insinööriyössä tehtiin EpiServer CMS 6 R2:lle sovelluskehitysohje, jossa esiteltiin EpiServerin ohjelmistovaatimukset ja EpiServerin asennus kuvia ja taulukoita apunakäyttäen. Tämän lisäksi työssä käsiteltiin EpiServerille www-komponenttien tekeminen alusta loppuun myös kuvina apuna käyttäen. Varsinainen sisällöntuotto esiteltiin työssä luoduille komponenteille. Työ tehtiin Suomessa toimivalle IT-yritykselle osana asiakasprojektia.</p> <p>Työn tuloksena oli sovelluskehitysohje, jota lukemalla saa kokonaiskuvan kuinka EpiServer asennetaan ja kuinka sille luodaan www-komponentteja sekä kuinka www-komponenteille luodaan sisältöä.</p> <p>Tämän kaltaista EpiServer sovelluskehitysohjetta, jossa käsiteltiin asennus, käyttöönotto, komponenttien teko ja sisällöntuotto samassa paketissa ei ole aikaisemmin tehty. Työtä voidaan hyödyntää uusien EpiServerin sovelluskehittäjien koulutuksessa, mikä on merkittävää ajan säästöä, jos vertaa kokonaisuutta hajallaan toisistaan oleviin verkkomateriaaleihin.</p>	
Avainsanat	EpiServer, Sisällöntuottojärjestelmä, IIS, SQL, CMS, .NET

Author(s) Title Number of Pages Date	Taavi Rouhiainen Content management system EpiServer CMS 6 R2 from the developer's point of view 43 pages 19th March 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Mikko Ojamo, Team Leader Olli Hämäläinen, Senior Lecturer
<p>The theme of the thesis was the content management system Episerver CMS 6 R2, focusing on the developer's point of view. The goal was to create an easy-to-read installation and software development manual for users that are new to the Episerver system.</p> <p>Creating a manual, showing the system requirements for the Episerver system and how to install it, using charts and pictures as an aid, were an important part of the project. In addition, the whole creation cycle of web components and content for them is portrayed in the thesis. The essential content production was introduced by using the components created in the project. The whole project was made for an IT company based in Finland.</p> <p>The result of the project was an application development manual for the system, which provides the reader with the necessary information of the installation, as well as component creation and even how to produce content for the components.</p> <p>A similar manual for the Episerver content management system, introducing installation, initialization, component and content creation, all in the same package, has never been made before. This work can be used, for example, to train new users for the Episerver system. If compared to the entities shattered all over the internet, this manual will without a doubt lead to a substantial decrease in the time used for the training.</p>	
Keywords	EpiServer, Content Management system (CMS), IIS, SQL, .NET

Sisällys

Lyhenteet

1	Johdanto	1
2	EpiServer CMS:n esittely	2
3	EpiServerin vaatimukset, asennus ja käyttöönotto	3
3.1	Järjestelmävaatimukset	3
3.2	EpiServerin asennus	8
3.3	EpiServerin käyttöönotto	15
4	EpiServerin www-osien teko	16
4.1	Komponenttien luonnista periaatteet	16
4.2	Sivutyypit ja UserControl	17
4.3	Twitter feedin luonti dynaamiseksi sisältökomponentiksi	24
4.4	Composer-tyyppinen sisällöntuottokomponentti	28
5	Sisällön tuottaminen EpiServeriin	33
5.1	Iframen sisällöntuotto	33
5.2	Sisällöntuotto Dynamic Content -komponentilla	35
5.3	Composer	41
6	Yhteenveto	42
	Lähteet	43

Liitteet

Liite 1. Sivupohjan graafinen määrittäminen (Iframe.aspx)

Liite 2. Sivupohjan toiminnallinen määrittäminen (Iframe.aspx.cs)

Liite 3. User Controlin graafinen määrittäminen (Iframe.ascx)

Liite 4. User Controlin toiminnallinen määrittäminen (Iframe.ascx.cs)

Liite 5. Dynaamisen sisältökomponentin graafinen määrittäminen (TwitterFeedControl.ascx)

Liite 6. Dynaamisen sisältökomponentin toiminnallinen määrittäminen
(TwitterFeedControl.ascx.cs)

Liite 7. EpiServerin englanninkielisen kielitiedoston määrittäminen
(ComposerSampleTemplates_EN.xml)

Liite 8. Composer-komponentin graafinen määrittäminen (StockBlock.ascx)

Liite 9. Composer-komponentin toiminnallinen määrittäminen (StockBlock.ascx.cs)

Lyhenteet

.NET	Microsoftin kehittämä ohjelmistokomponenttikirjasto, joka helpottaa ylemmän tason ohjelmointia. Muun muassa C# -ohjelmointikieli hyödyntää .NET -kirjastoja.
C#	Microsoftin kehittämä ohjelmointikieli, jossa on Javan ominaisuuksia C-kielen syntakseilla.
CMS	<i>Content management system</i> , yleisnimitys tietojärjestelmille, joilla on tarkoitus hallita organisaatioiden sisällöntuottoa.
IIS	<i>Internet information service</i> , Windows-pohjaisille palvelimille Microsoftin kehittämä palvelinohjelmistokokonaisuus.
SQL	<i>Structured Query Language</i> , standardoitu kyselykieli, jolla voidaan ohjata tietokannan sisältöä.
XML	<i>Extensible Markup Language</i> : kuvauskieli, jolla voidaan kuvata laajoja tietomassoja.

1 Johdanto

Tässä insinööriyössä tutustutaan EpiServer CMS 6 R2:n sisällöntuottojärjestelmään. EpiServer on yksi nopeimmin kasvavista palveluntarjoajista sisällönhallintajärjestelmien (WCM), sosiaalisten verkkoyhteisöjen ja sähköisen kaupankäynnin alueilla. EpiServer-alustaa toimitetaan 590 partnerin voimin kolmessakymmenessä maassa. Siinä yhdistyvät luotettavuus ja kaupallisen tuotteen tuki EpiServer World -yhteisössä, jossa kehittäjiä on yli viisitoistatuhatta. [1.]

Insinööriyössä esitellään yleisesti lukijalle EpiServer-sisällöntuottojärjestelmä, EpiServerin asennus, järjestelmän vaatimukset, käyttöönotto ja erilaisten komponenttien tekeminen sekä sisällöntuotto. Komponenttien tekemisessä käydään läpi kolmen erityyppisen komponentin luonti, joka pitää sisällään komponenttien esittelyn, tiedostojen luonnin, tiedostoihin kirjoitettavan koodin ja tapauskohtaiset asiat riippuen komponentista. Lopuksi esitellään, kuinka luotuihin komponentteihin luodaan sisältöä.

Insinööriyö tehtiin Suomessa toimivalle IT-yritykselle, joka tarvitsi EpiServer-sovelluskehitysohjeen. Tarkoituksena on tuottaa EpiServer-sovelluskehitysohje uusille EpiServer-sovelluskehittäjille, mikä helpottaa EpiServerin kokonaiskuvan hahmottamista. Työn tavoitteena on luoda helposti lähestyttävä ja selkeä ohje EpiServer CMS:n kehitystä varten.

Oma roolini projektissa oli pääosin sovelluskehittäjä, mutta tein myös komponenteille hieman ulkoasuakin. Iframe- ja TwitterFeed-komponentit ovat omaa käsialaa. Composer-komponentti on otettu toisesta projektista, mutta laitoin sen yhteensopivaksi tähän projektiin.

2 EpiServer CMS:n esittely

EpiServer-sisällöntuottojärjestelmä on toteutettu Microsoftin .NET 3.5:n arkkitehtuurilla, ja tietokantana EpiServer käyttää Microsoftin SQL-Serveriä tai Oraclea. Monet valitsevat EpiServerin sen tehokkuuden ja helppokäyttöisyyden vuoksi. Lisäksi se on todella nopea omaksua. Sisällöntuottojärjestelmän käyttöliittymästä on versiot kahdelletoista eri kielelle, ja sisällön kieliversiointi on helposti hallittavissa. Lisäksi tuotteeseen on implementoitu valmiiksi muun muassa dokumentinhallinta, sisällönhallinta ja lomakkeiden toteutus. [2.]

EpiServer CMS:n projektien hinnat ovat arviolta 30 000 eurosta 80 000 euroon. Hinnat voivat olla myös suurempiakin, ja niihin vaikuttavat projektin työmäärät ja haastavuus. Lisenssejä on kahdelle eri ohjelmistoversiolle, EpiServer CMS Professionalille ja EpiServer CMS Enterpriselle. Lisenssi maksetaan vain kerran, ja Professional-version hinta on noin 11 000 euroa ja Enterprisen hinta noin 15 000 euroa. [2.]

EpiServer-alustaa on käytetty monien suomalaisten yritysten Internet-ratkaisuissa kuten mm. Keskon, Orionin ja Blue1:n. Suomessa EpiServerin kilpailijoita ovat Meteor ja Innofactor Prime ja Pohjoismaissa Sitecore. Microsoftin SharePointia pidetään myös EpiServerin kilpailijana. [2.]

3 EpiServerin vaatimukset, asennus ja käyttöönotto

3.1 Järjestelmävaatimukset

EpiServerillä on neljä erityyppistä käyttöalustatarkoitusta, joille on olemassa erilaiset ohjelmistovaatimukset.

- Palvelinvaatimukset, jotka vaaditaan palvelimelta, johon EpiServer CMS asennetaan.
- Editointikäyttäjän ohjelmistovaatimukset, jotka vaaditaan sisällöntuottoa varten.
- Selaaajakäyttäjän ohjelmistovaatimukset, jotka vaaditaan valmiiden EpiServer CMS:n sivustojen selaamiseen.
- Kehitys- ja esitysympäristöjen ohjelmistovaatimukset, jotka vaaditaan päätteeltä, jolla rakennetaan ja testataan komponentteja CMS-järjestelmälle.

EpiServer CMS 6 R2, palvelimen vaatimukset

Taulukossa 1 on esitelty, millaiset käyttöjärjestelmät, Web-palvelimet ja tietokannat vaaditaan EpiServer CMS 6 R2 -palvelimelle. Edellisten lisäksi on esitelty muut vaatimukset eli käytännössä vaadittavat .NET-arkkitehtuurin versiot.

Taulukko 1. EpiServer CMS 6 R2-palvelinvaatimukset. [3]

Käyttöjärjestelmä	Jokin seuraavista käy: Microsoft Windows Server 2003 R2 32/64 bit Microsoft Windows Server 2008 32/64 bit Microsoft Windows Server 2008 R2 Kaikki Service pack:it ovat tuettuja
Web-palvelin	Jokin seuraavista pitää asentaa osana Windowsin palvelinkokonaisuutta Microsoft Internet Information Services (IIS) 6.0 Microsoft Internet Information Services (IIS) 7.0 Microsoft Internet Information Services (IIS) 7.5
Tietokanta	Jokin seuraavista käy: Microsoft SQL Server 2005 32/64 bit Microsoft SQL Server 2008 32/64 bit Microsoft SQL Server 2008 R2 Oracle 10g R2 32/64 bit (Windows-palvelin) Oracle 11g 32/64 bit (Windows-palvelin) Kaikki Service pack:it ovat tuettuja
Muut vaatimukset	Jokin seuraavista käy: Microsoft .NET Framework 3.5 SP1 Microsoft .NET 4.0 – (Muunnetuilla asetuksilla, löytyy liitteet kohdasta nro.1)

Taulukossa 1 on käyty läpi, millaisia alustavaatimuksia EpiServer CMS:llä on, jotta se voidaan asentaa toimimaan luotettavasti. Microsoftin palvelintuotteet ovat yleensä paras ratkaisu toimivalle EpiServer-alustalle.

EpiServer CMS 6 R2, editointikäytön ohjelmistovaatimukset

Taulukossa 2 on mainittu, mitkä ovat järjestelmävaatimukset EpiServer CMS 6 R2:n editointi- ja sisällöntuottotarkoitusta varten. Siinä esitellään käyttöjärjestelmä, Web-selain ja vaadittava versio Microsoft Officesta integrointia varten.

Taulukko 2. EpiServer CMS 6 R2, Editointikäyttäjän vaatimukset. [3]

Käyttöjärjestelmä	Jokin seuraavista käy: Microsoft Windows XP Microsoft Windows Vista Microsoft Windows 7 Mac OSX 10.6
Web-selain editointia varten	Jokin seuraavista käy: Microsoft Internet Explorer 7.0 Microsoft Internet Explorer 8.0 Microsoft Internet Explorer 9.0 Firefox 3.6, 4.0 Microsoft Internet Explorer (32-bittinen versio) tarvitaan drag and drop-tiedostojen serverille siirtoa varten ja julkaisuja Microsoft Officesta EpiServer CMS:iin.
Microsoft Officen integrointiin	Jokin seuraavista käy: Microsoft Office 2003 SP3 Microsoft Office 2007 SP1 Microsoft Office 2010 – vain 32-bittinen

Editointikäyttäjältä ei vaadita juuri muuta kuin käyttöjärjestelmä ja selain. Mikäli on tarve Officen intergrointiin, niin Office pitää olla asennettuna

EpiServer CMS 6 R2, selaajakäytön ohjelmistovaatimukset

Taulukossa 3 on esiteltyä, millä selaimilla pystyy luotettavasti selaamaan EpiServer CMS 6 R2:llä tehtyjä sivustoja.

Taulukko 3. EpiServer CMS 6 R2, Selaajakäyttäjän vaatimukset. [3]

Web-selain selausta varten	Kaikki yleisimmät selaimet ovat tuettuja: Microsoft Internet Explorer Mozilla Firefox Google Chrome, Opera, Safari, Netscape Selainten katselutuki riippuu EpiServer CMS pohjien(template) rakenteesta.
Live-seuranta	Silverlight 3 tai 4 vaaditaan.

Taulukossa 3 esitellään EpiServer-sivustoa selaavan käyttäjän vaatimuksia. Riittää, kun koneessa on selain ja Silverlight.

EpiServer CMS 6 R2, kehitys- ja esitysympäristöjen vaatimukset

Taulukko 4 kertoo, mitä käyttöjärjestelmiä, kehitystyökaluja ja tietokantoja EpiServer CMS 6 R2 -sovelluskehittäjällä pitää olla.

Taulukko 4. EpiServer CMS 6 R2, Kehitys ja esitysympäristöjen vaatimukset. [3]

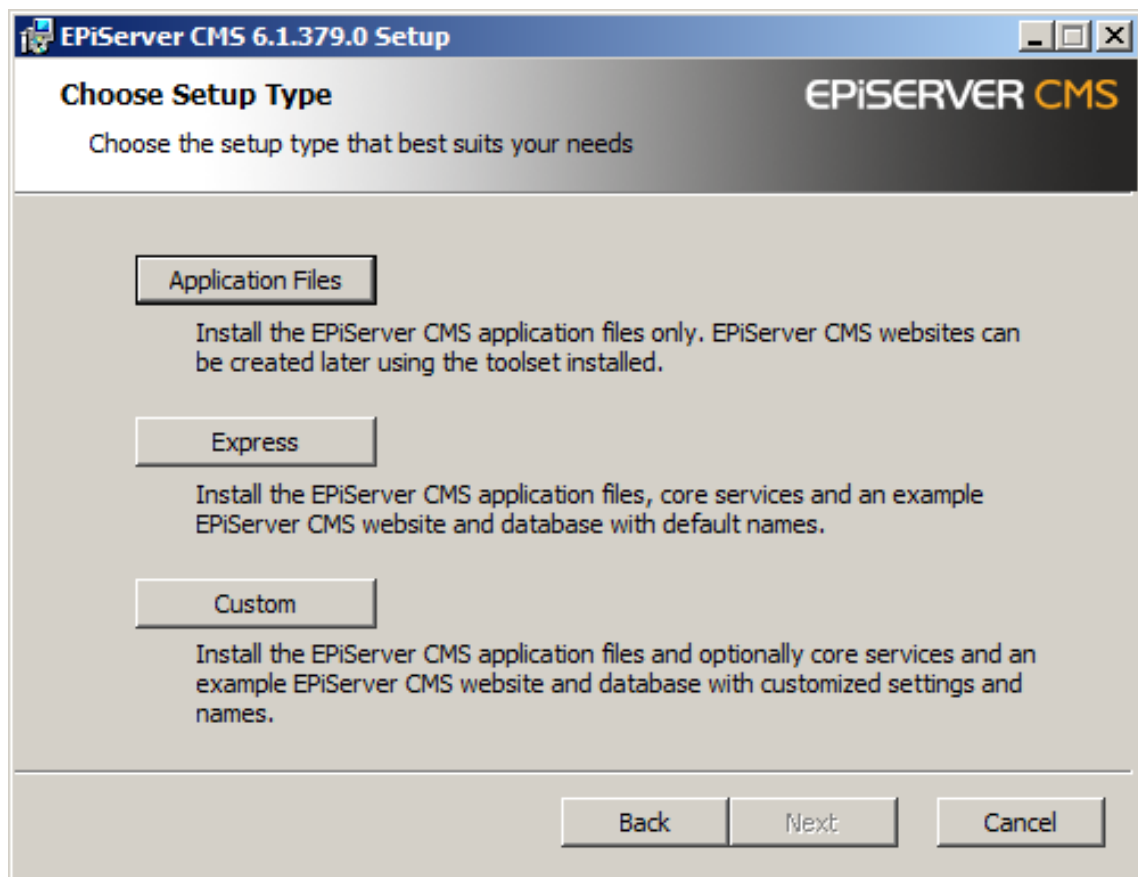
Käyttöjärjestelmä	Jokin seuraavista käy: Microsoft Windows Server 2003 R2 32/64 bit Microsoft Windows Server 2008 32/64 bit Microsoft Windows Server 2008 R2 Microsoft Windows XP SP3 32 bit Microsoft Windows Vista SP2 32/64 bit Microsoft Windows 7 SP1 32/64 bit
Kehitystyökalu	Jokin seuraavista käy: Microsoft Visual Studio 2008 SP1 Microsoft Visual Studio 2010 Visual Web Developer 2008 Express Edition SP1 Visual Web Developer 2010 Express Edition
Tietokanta	Jokin seuraavista käy: Microsoft SQL Server 2005 SP4 Microsoft SQL Server 2005 SP4 Express Microsoft SQL Server 2008 SP2 Microsoft SQL Server 2008 SP2 Express Microsoft SQL Server 2008 R2 Oracle 10g R2 32/64 bit* Oracle 11g 32/64 bit * Oracle 10g R2 Express Edition * EpiServerin Visual studio-integraatio SDK:n asennus vaatii Microsoft SQL Server Expressin. The EpiServer Visual Studio integration SDK installation requires Microsoft SQL Server Express.

Taulukossa 4 on esimerkkejä Windows-käyttöjärjestelmän versioista, kehitystyökaluista ja tietokannoista, jotka soveltuvat käytettäväksi EpiServer-kehitysympäristöön.

3.2 EpiServerin asennus

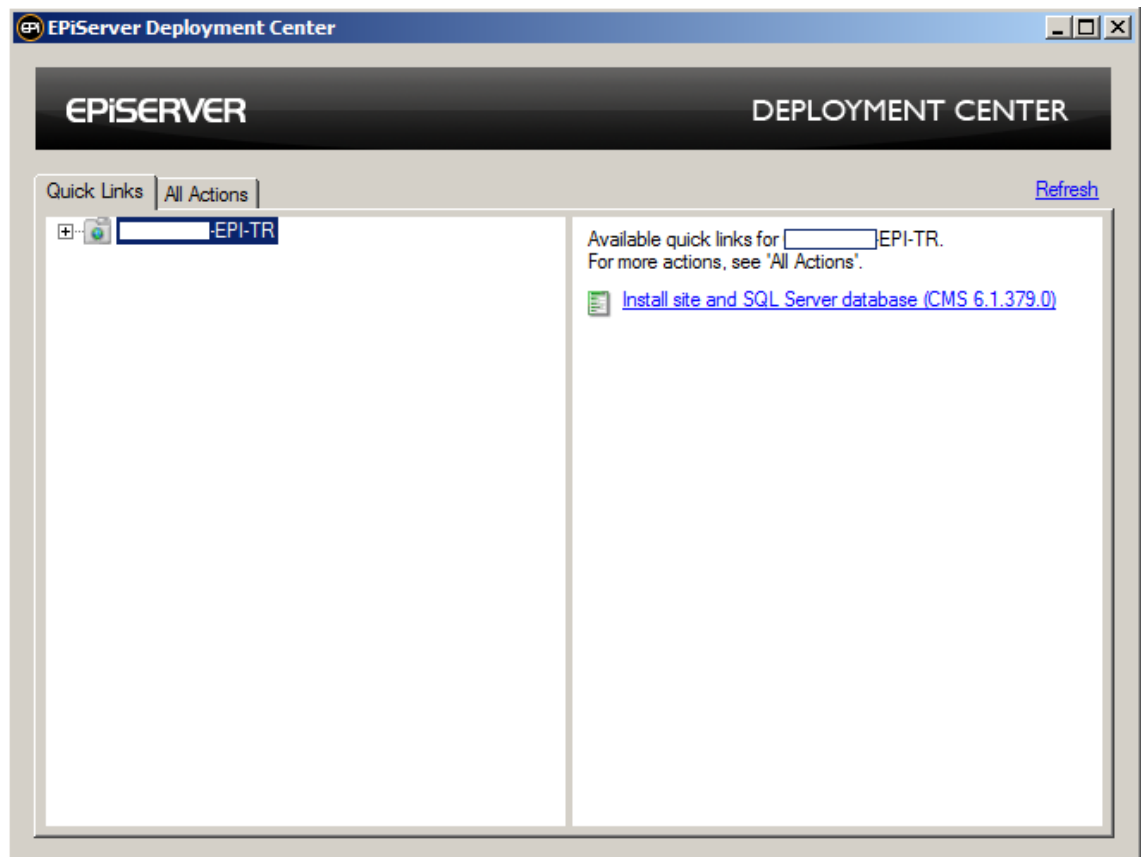
Seuraavaksi demonstroidaan EpiServerin asennus virtuaalikoneelle, jossa käyttöjärjestelmänä on Windows server 2008 R2 standard ja tietokantana SQL Server Express. Lisäksi Web-palvelimena on Microsoft Internet Information Services (IIS) 7.0, kehitystyökaluna Microsoft Visual Studio 2010 ja koodipohjana Microsoft .NET framework 3.5 SP1.

Aluksi ladataan EpiServer CMS 6 r2:n zip-tiedosto EpiServerin sivuilta. Sisältö puretaan omaan kansioonsa ja käynnistetään kansioista setup.exe. Asennus pyytää käyttäjää hyväksymään ehdot, minkä jälkeen siirrytään seuraavanlaiseen näkymään (kuvio 1), jossa valitaan Express-asennus ja painetaan Next-painiketta.



Kuvio 1. EpiServerin asennusikkuna ehtojen hyväksynnän jälkeen.

Seuraavassa valikossa, kuvion 1 jälkeen, painetaan Install-painiketta, jolloin asennusprosessi lähtee käyntiin. Kun Finish-painiketta on painettu, aukeaa EpiServerin Deployment Center, josta valitaan "Install site and SQL Server database". Tämä esitellään kuviossa 2.



Kuvio 2. Deployment Center.

Kuviossa 3 on esitetty, miten määritellään sivuston nimi, jolle EpiServer CMS asennetaan. Nimen voi päättää itse tai valita viereisestä pudotusvalikosta. Jos nimeä ei ole aiemmin olemassa, se luodaan uutena IIS:een. Lisäksi määritellään sivuston fyysinen polku. IIS:n käyttämä Application Pool (kokoelma osoitteista, jotka linkittyvät yhteen tai usempaan ”worker” -prosessiin, jotka ovat Windows prosesseja (w3wp.exe) ja vastaavat web-sovelluksista [8]) määräytyy automaattisesti sivuston polun määrittämään kohteeseen. Web-palvelinsidoksia voi määrittellä useamman, mutta ainakin yksi vaaditaan ja asetetaan sivustoa varten IIS:een. Relative Path URL:ia käytetään käyttöliittymään pääsemiseen. Tässä vakiopolku ei kelpaa, vaan pitää itse luoda uusi polku. [4.]

EPISERVER CMS DEPLOYMENT CENTER

Create new EpiServer CMS site (with SQL Server database) - Step 1 of 6

Site

Name: Epi_Esimerkki

Application (Optional):

Path: C:\EpiServer\Sites\Epi_Esimerkki Browse...

Application Pool Name: Epi_EsimerkkiAppPool

Web Server Bindings

	Protocol	Host Address (* = all)	Port	Default	UI Binding
▶	HTTP	*	17001	<input checked="" type="checkbox"/>	<input type="checkbox"/>
*				<input type="checkbox"/>	<input type="checkbox"/>

EpiServer User Interface

Relative Path: /Epi/

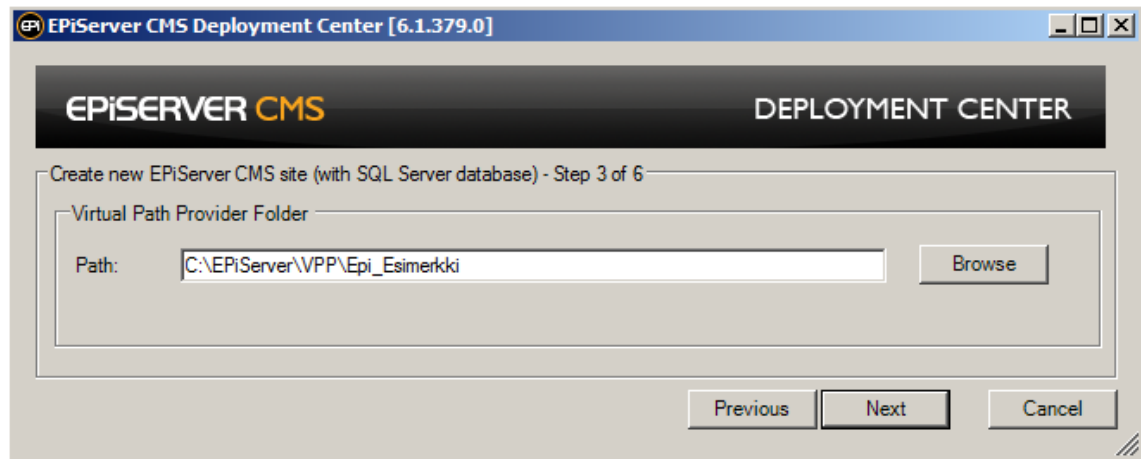
Previous Next Cancel

Kuvio 3. EpiServer IIS:n konfigurointia.

Seuraavaksi siirrytään tietokannan asetuksiin, jotka esitellään kuviossa 4. Ylimpään kenttään voi itse määrittää tietokantapalvelimen osoitteen tai valita sen alasvetovalikosta. Myös portin voi valita — vakioportti on 1433. Seuraavaksi valitaan SQL-palvelimeen tunnistautumistapa. Tämän jälkeen on vuorossa tietokannan valinta. Voidaan valita joko jo olemassa oleva tietokanta tai sitten määritellä itse uusi tietokanta. Käyttäjätunnus ja salasana valitaan edeltävän valinnan eli jo olemassa olevan tietokannan tunnusten perusteella tai sitten keksitään itse määriteltyyn tietokantaan uudet tunnukset. [5.]

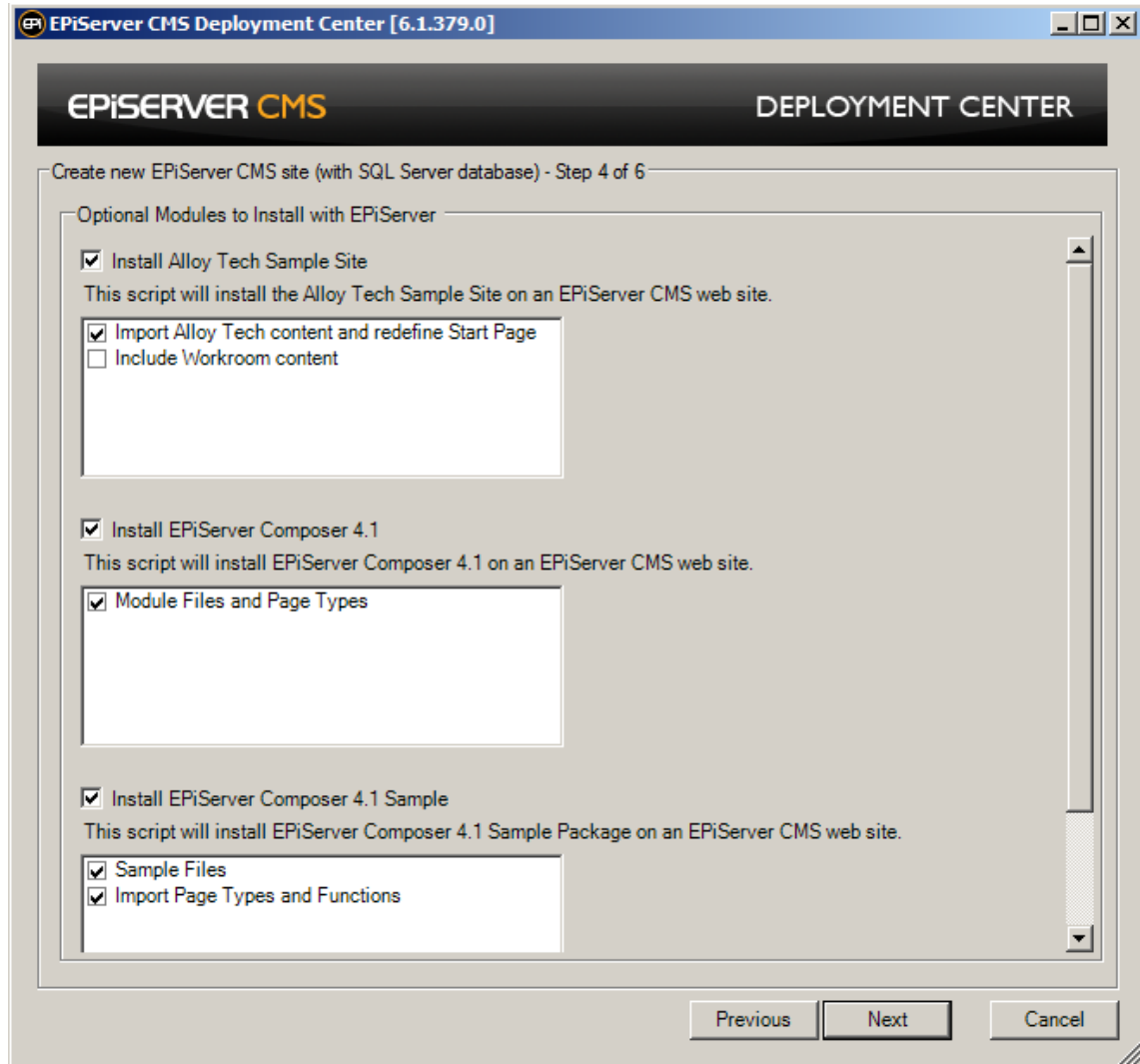
Kuvio 4. EpiServer-sivuston tietokantayhteyksien määrittäminen.

Kuviossa 5 esitellään seuraava valikkonäkymä, jossa valitaan polku VPP-kansioille (Virtual Path Provider eli VPP on tapa integroida tiedostoja muiden ulkopuolisten sovellusten kanssa), jotka ovat Documents, Global ja PageFiles. Asennuskripti luo kansiot asetettuun polkuun. [4.]



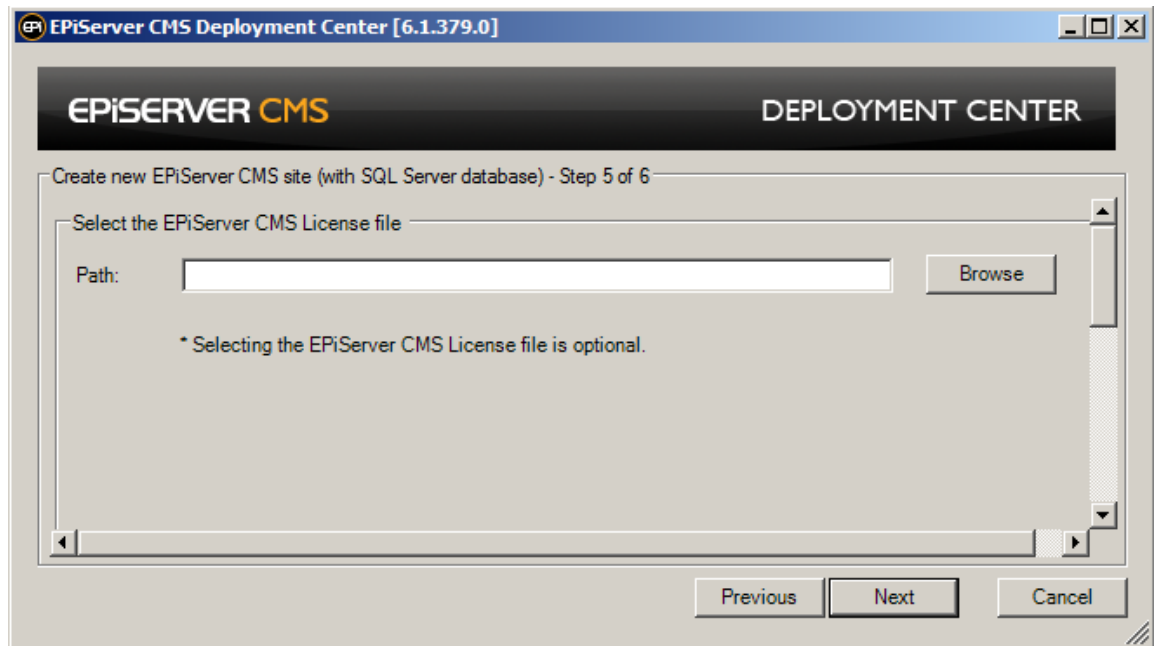
Kuvio 5. EpiServerin Virtual Path Provider: polun asetus.

Kuvion 6 näkymässä valitaan asennuksen yhteydessä asennettavat lisäosat eli moduulit. On mahdollista valita esimerkkisivusto, Composer-moduuli (josta lisää myöhemmin) sekä siihen esimerkit. [4.]



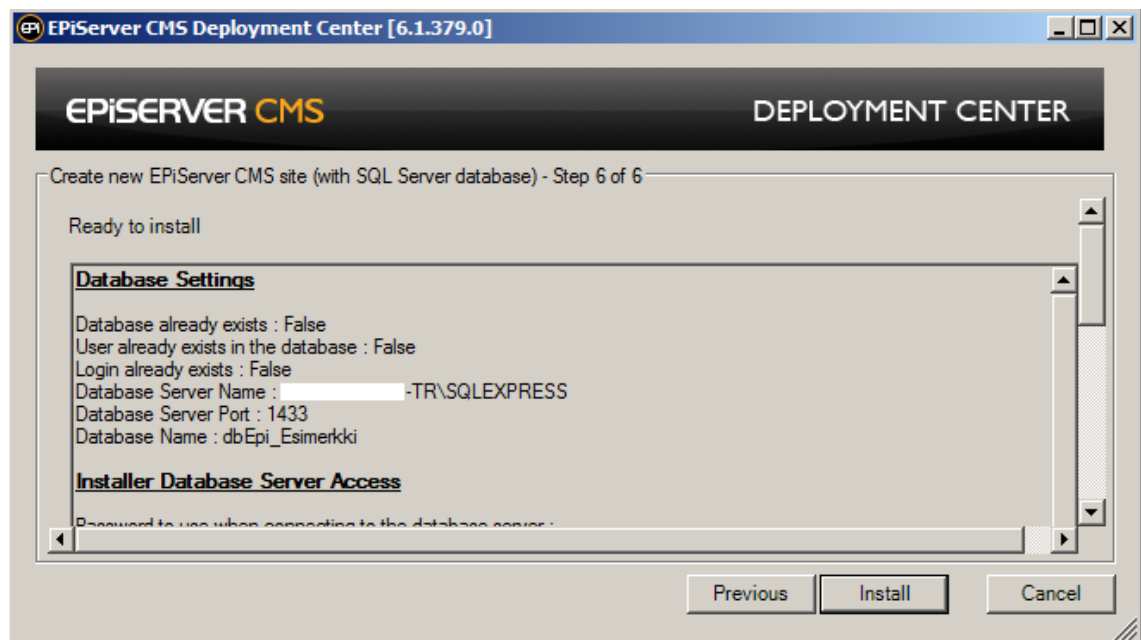
Kuvio 6. EpiServer-moduulien valinta.

Kuvion 7 valikosta valitaan EpiServerin CMS-lisenssi. Sitä ei kuitenkaan ole pakko valita, mutta lisenssin puuttuminen aiheuttaa virheilmoituksen. Lisenssin voi tilata EpiServerin sivuilta. [4.]



Kuvio 7. EpiServer-lisenssin valinta.

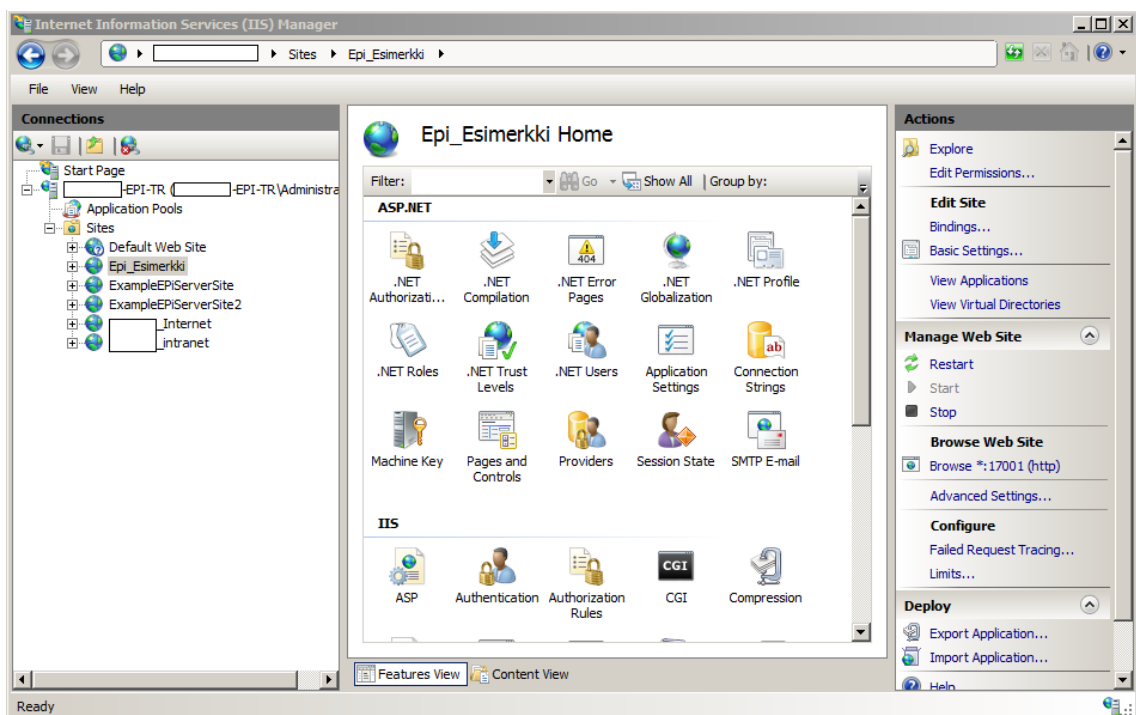
Kuviossa 8 esitellään näkymä, jossa voi selata asetukset lävitse ennen asennuksen hyväksymistä. [4.]



Kuvio 8. EpiServer CMS -asennusikkuna.

3.3 EpiServerin käyttöönotto

Seuraavaksi esitellään pikainen käyttöönotto. Ensin navigoidaan IIS:een ja valitaan asennusvaiheessa luotu sivusto, mikä on esiteltynä kuviossa 9. Tästä näkymästä valitaan "Explore" oikealta yläkulmasta. Tämä ohjaa sivuston juureen. Valitaan "episerver.config"-tiedosto ja etsitään tiedostosta seuraavat kohdat: `siteUrl="http://<yritys>-epi-tr:17001/"` `uiUrl="~/epi/CMS/"`. Näillä kahdella tiedolla päästään EpiServerin ylläpito- ja sisällöntuottonäkymään. Ylläpito- ja sisällöntuottonäkymä esitellään tarkemmin myöhemmin.



Kuvio 9. EpiServerin IIS-näkymä.

Asennetun EpiServerin juurikansioon ilmestyy Visual Studio -projekti asennuksen yhteydessä. Kun projektin käynnistää, se pyrkii muuttamaan itsensä sovellukseksi. Kyseessä on siis EpiServer-sovellus, johon voidaan rakentaa useasta projektista toimiva kokonaisuus. Esimerkki projektista voisi olla erillinen kokoelma composer-komponenteista tai web-sivustot. Rakenteet esitellään myöhemmin.

4 EpiServerin www-osien teko

4.1 Komponenttien luonnin periaatteet

Tässä luvussa tutustutaan EpiServerin erilaisiin sisällöntuottamismenetelmiin. Idea lyhykäisyydessään on rakentaa sivupohjien päälle toimivia komponentteja, joita voi tarpeen tullen käyttää useammissa kohdissa.

Pääosin työt aloitetaan määrittelemällä tarkasti, mitä halutaan, jonka jälkeen voidaan luoda kooditiedostot projektin puurakenteeseen oikeille paikoille. Kun tiedostot on luotu, voidaan aloittaa koodin kirjoitus. Tämän jälkeen määritellään oikeantyyppiset sivutyypit EpiServerin ylläpidon puolelta eli selaimen kautta valitsemalla. Lisäksi ohjelmistokoodit pitää linkittää niille tarkoitettuihin sivupohjiin. Joissain tilanteissa tulee myös määrittää muuttujille kieliasetukset resurssitiedostoihin, jotta sivuston jokaiselle kieliversiolle saataisiin omat kielet.

Seuraavissa alaluvuissa esitellään kolme erilaista komponenttia, Iframe, Twitter-feed ja StockBlock. Iframen tarkoitus on nostaa kyseiselle sivulle joltain muulta sivulta määritellyn kokoinen lohko, esimerkiksi kuva tai taulukko. Twitter-feed puolestaan on dynamic content -komponentti, eli se voidaan nostaa vapaaseen tekstikenttään millä tahansa sivulla. Sen ideana on näyttää esimerkiksi yrityksen viimeisimmät twiitit eli Twitteriin kirjoitetut päivitykset. Viimeisenä on composer-tyyppinen StockBlock-komponentti. Composer-komponentteja voi käyttää ainoastaan composer-sivupohjien päällä ja niiden näyttäminen sivulla tapahtuu yksinkertaisesti vetämällä kyseinen komponentti valikosta haluttuun paikkaan sivulla. StockBlockin idea on näyttää määritellyn yrityksen osaketietoja.

Tässä EpiServer-sovelluksessa keskitytään kahteen eri projektiin: Composer ja WebSites. WebSites-projektin alta löytyvät Iframen ja Twitter-feedin komponentit ja Composer-projektin alta luonnollisesti composer-komponentti, StockBlock. Projektien tekemiseen on käytetty Visual Studiota ja visuaaliseen varmentamiseen eri selaimia.

Edellä mainituista asioista kerrotaan tarkemmin seuraavissa alaluvuissa.

4.2 Sivutyypit ja UserControl

Sivutyypit ovat malleja siitä, miten sivut rakentuvat, ja ne käyttävät samaa pääsivupohjaa säilyttääkseen saman ulkoasun muiden sivutyypin kanssa. Sivutyypeille voidaan tehdä kiinteitä kontrolleja, jotka tulevat aina vakiopaikalle, kun sivu luodaan. Tässä osiossa havainnollistetaan Iframe-sivutyypin ja sen kontrollien luonti.

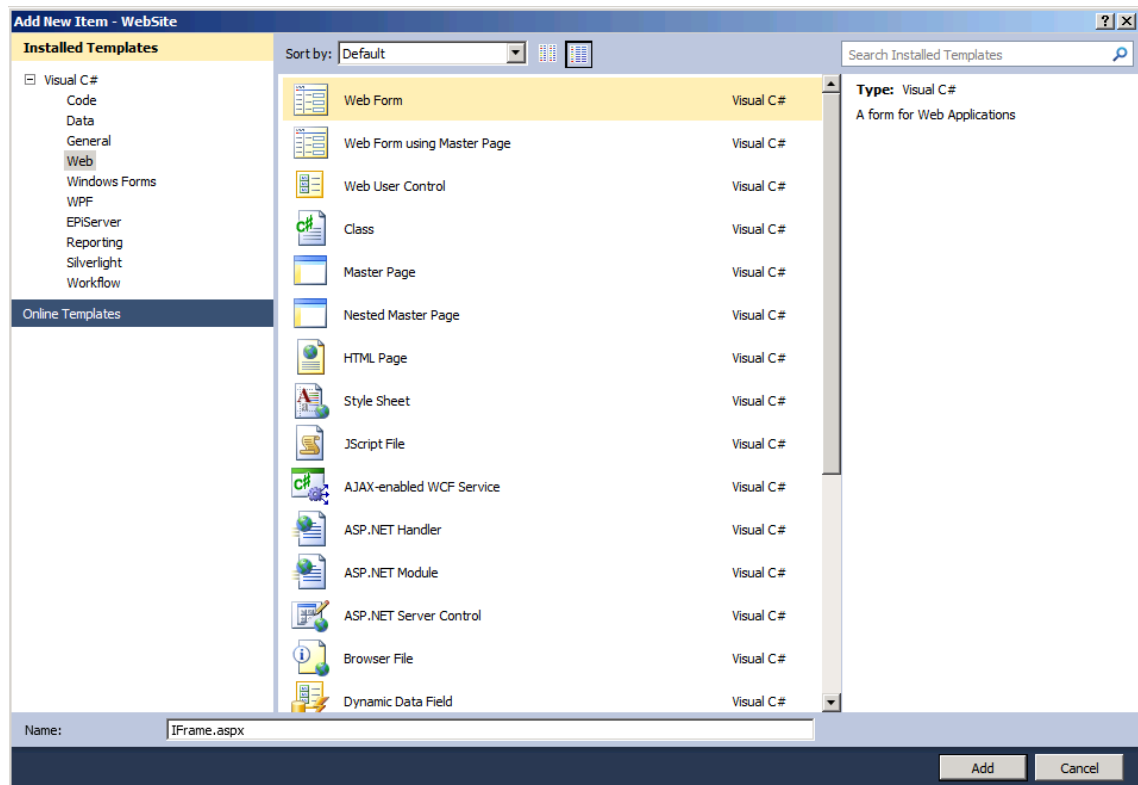
Tiedostojen luonti Iframe-komponenttia varten

Kuviossa 10 luodaan Visual Studioon uusi tiedosto Iframe-sivupohjaa varten. Tiedosto lisätään Visual Studio-projektissa Pages-hakemiston alle.



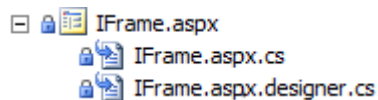
Kuvio 10. Sivupohjan luontia Visual Studioon.

Kuviossa 11 valitaan oikea tiedostotyyppi puurakenteen Pages-kohdan alle. Tiedosto on muotoa .aspx, ja sille annetaan nimeksi Iframe.aspx, jonka jälkeen tiedosto luodaan napsauttamalla Add-painiketta.



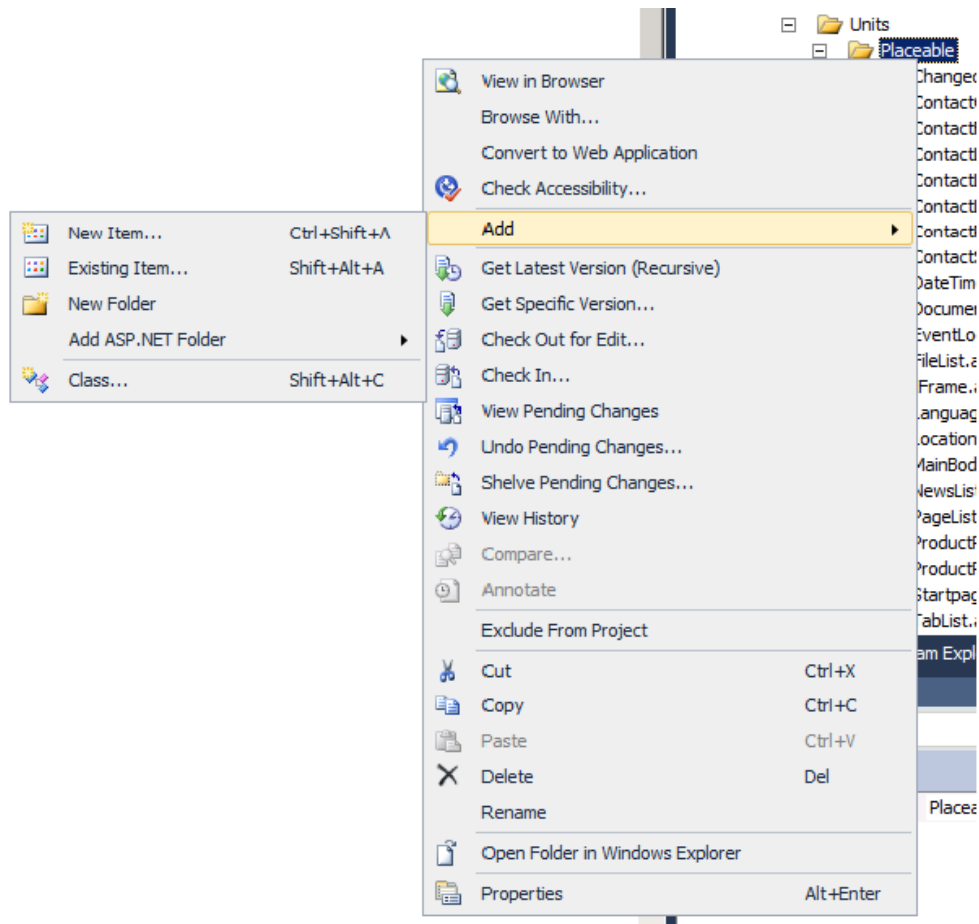
Kuvio 11. Sivustopohjan tiedostotyypin valinta.

Puurakenteeseen Pages-hakemiston alle ilmestyy IFrame.aspx, jonka alle luodaan automaattisesti kaksi muuta tiedostoa, joista IFrame.aspx.cs:llä luodaan toiminnallisuus IFrame.aspx:lle, kuten kuviossa 12 on esitelty.



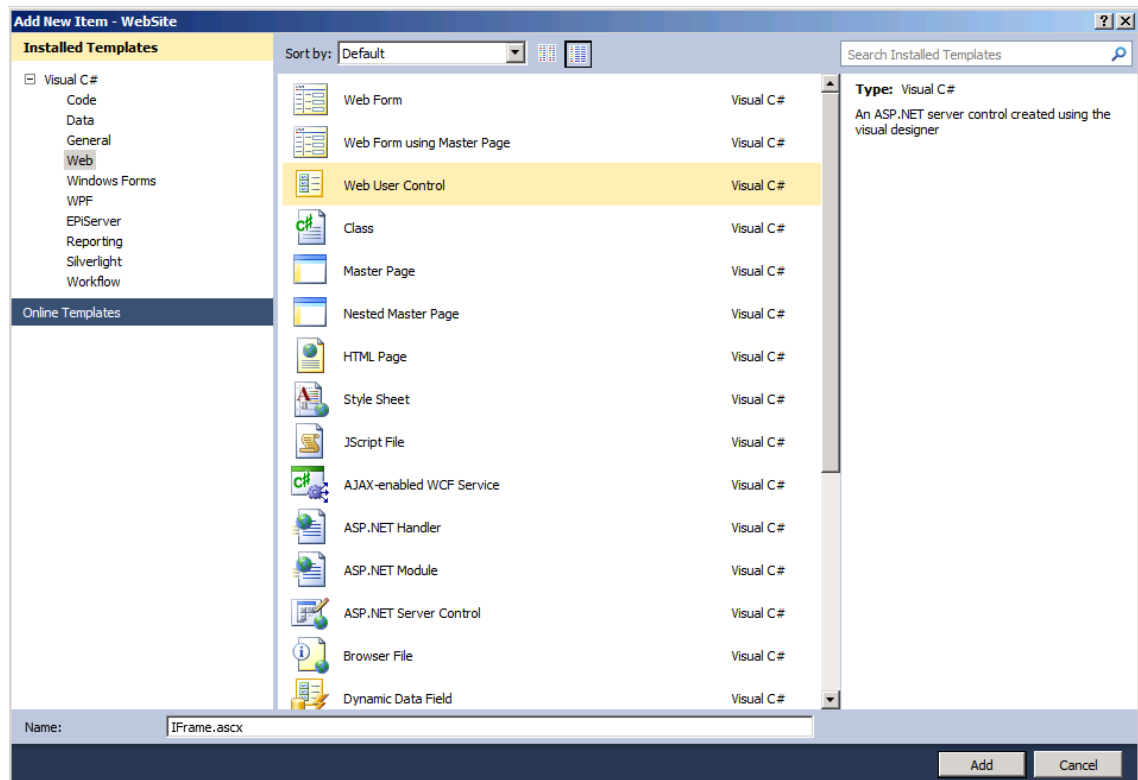
Kuvio 12. Sivupohjatiedostot hakemistorakenteessa.

Kun sivupohjatiedostot on luotu, luodaan Usercontrol-tiedostot. Nämä luodaan projektin puurakenteessa hakemistojen "Units" ja "Placeable" alle. Usercontrol on toiminnallisuus, joka upotetaan edellä luotuun sivupohjaan. Kuviossa 13 on esitelty tiedostojen luontia.



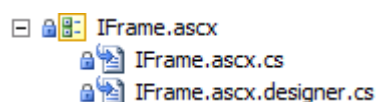
Kuvio 13. UserControlin luonti.

Kuviossa 14 valitaan varsinainen UserControl-tiedostotyyppi Visual Studiosta. Sen päätte on .ascx, ja sille annetaan nimeksi lframe.ascx, jonka jälkeen lisätään tiedosto projektiin painamalla Add-painiketta.



Kuvio 14. Tiedostotyypin valinta.

Nyt puurakenteen hakemistojen Units- ja Placeable-kohdan alle on ilmestynyt IFrame.ascx ja sen alle automaattisesti luodut kaksi tiedostoa, joista toinen on toiminnallisuus (IFrame.ascx.cs) tiedostolle IFrame.ascx. Tiedostot on esiteltyinä kuviossa 15.



Kuvio 15. IFrame Usercontrol -tiedostot

Tiedostojen kuvaus

EpiServerin ohjelmointikielenä on Microsoftin kehittämä Javaan perustuva kieli C#, jossa käytetään .NET-kirjastoja hyödyksi. C# -tyyppiset tiedostot ovat niin sanottuja codebehind-tiedostoja, eli niiden avulla käsitellään graafiselta puolelta tulevaa dataa. Codebehind-tiedostojen päätte on .cs, joka määrittelee tiedoston C#-kieliseksi

tiedostoksi. Käyttöliittymän visuaaliseen toteutukseen käytetään html-kieltä. Näihin visuaali-tiedostoihin sidotaan edellä mainittu codebehind-tiedosto.

Tiedostojen kuvaus aloitetaan graafisella `Iframe.aspx`-tiedostolla. Tässä tiedostossa määritellään, mitä kyseisellä sivupohjalla tullaan näkemään ja missä sivupohjaan asetetut komponentit sijaitsevat. Sivupohjaan liitetään tekstikenttä sekä `Iframe` usercontrol, joka esitellään myöhemmin. Lisäksi tiedostoon määritellään codebehind-tiedosto nimeltä `Iframe.aspx.cs`, joka esitellään seuraavassa kappaleessa. Käytännössä määrittely tapahtuu automaattisesti tiedoston luonnin yhteydessä. Esimerkkikoodi `Iframe.aspx`-tiedostosta on kommentoituna liitteessä 1.

`Iframe.aspx`:n codebehind-tiedostossa, `Iframe.aspx.cs`, ei tehdä muuta kuin periytetään tiedostot `TemplatePage`sta (määritellään tietynlaiseksi riippuen projekteista) ja määritellään käytettävät kirjastot. `Iframe.aspx.cs`:n koodi on kommentoituna liitteessä 2.

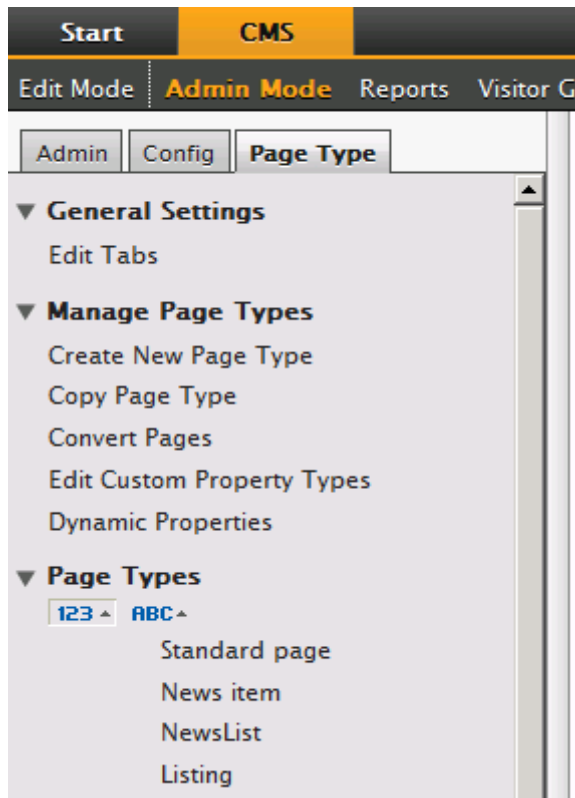
Graafinen usercontrol-tiedosto `Iframe.ascx` kertoo, miltä `Iframe`en nosto näyttää `Iframe.aspx`:n määrittelemällä sivulla. Tähän kuuluvat nostettavan objektin polku, korkeus ja leveys. `Iframe.ascx` käyttää codebehind-tiedostona `Iframe.ascx.cs`, jossa käsitellään erilaisten muuttujien välittämistä `Iframe.ascx`-tiedostoon. `Iframe.ascx`:n koodi on kommentoituna liitteessä 3.

`Iframe.ascx.cs` on `Iframe.ascx`:n codebehind-tiedosto, jossa määritellään toiminnallisuus `Iframe.ascx`:ä varten. Tiedosto hakee sille annetusta polusta sivun tai sen osan ja näyttää sen sisällön määritellyn kokoisena `Iframe.ascx`:ssä. `Iframe.ascx.cs`:n koodi on kommentoituna liitteessä 4.

Sivutyypin linkitys koodattuun sivupohjaan `EpiServer`in käyttöliittymän puolella

`EpiServer`in ylläpitopuoli on käytettävissä selaimen kautta. Käytännössä se on tavallinen sivusto, johon kirjaudutaan sisään, ja näin ollen saadaan ylläpitotyökalut näkyville. Ylläpitotyökalujen avulla voidaan rakentaa sivustolle erilaisia sivupohjia sekä muokata sivujen sisältöä. Erilaisten sivupohjien idea on saada vaivattomasti luotua uusi sivu. Esimerkiksi tässä tapauksessa luodaan `Iframe`-sivupohja, jotta sivustoon voidaan jatkossa upottaa useita `Iframe`-tyyppisiä sivuja.

Seuraavaksi käydään läpi, kuinka EpiServerin ylläpitopuolella linkitetään sivutyyppi edellä koodattuun sivupohjaan. Aluksi kirjaudutaan sisään EpiServeriin ylläpitäjän tunnuksilla. Tämän jälkeen valitaan CMS-valikosta Admin Mode, josta navigoidaan Page Typeen. Täältä valitaan Manage Page Types ja sen jälkeen Create New Page Type. Edellä mainittu näkymä esitellään kuviossa 16.



Kuvio 16. EpiServerin Admin Moden sivustotyyppinäkymä.

Seuraavaksi valitaan sivutyyppille nimi, kuvaus ja polku, joka viittaa aiemmin luotuun IFRAME.aspx-tiedostoon. Kun kentät on täytetty, painetaan Save-painiketta oikealta alareunasta. Edellistä havainnollistaa kuvio 17.

Create New Page Types ?

Edit the basic information about the page type.

Information | Default Values | Available Page Types

Name: []IFrame

Description: IFRAME page

File name: /Templates/ /Pages/IFrame.aspx

Sort index: 100

Available in Edit mode

Access level

Create Select all

Everyone

Add Users/Groups

Save

Kuvio 17. EpiServerin sivustotyyppin luonti.

Seuraavaksi valitaan yllä luotu sivutyyppi Page Types -kohdasta (katso kuvio 16). Valinnan jälkeen aukeaa valikko, jossa voi lisätä IFRAME-sivutyyppille ominaisuuksia. IFRAME-sivu vaatii kuviossa 18 esitellyt ominaisuudet, jotka ovat Link, FrameWidth ja FrameHeight. Nämä ominaisuudet ovat tätä kautta yhteydessä koodiin, eli ne näkyvät sivua luotaessa ja välittyvät sitä kautta koodin muuttujiin.

[] IFRAME ?

IFRAME page

Add Property Settings

Move Up	Move Down	Name	Type	Tab	Unique value per language	Field name	Help Text
	↓	Heading	String (<= 255)	Content	Yes	Heading	Specify a heading that will present the main title of this page. If this property is not set, the page name will be used as a heading instead.
↑	↓	MainBody	XHTML string (>255)	Content	Yes	Main body	The main body will be shown in the main content area of the page. You can insert for example text, images and tables, using the XHTML-editor.
↑	↓	Link	URL to page/external address	Content	No	Url	
↑	↓	FrameWidth	Integer	Content	Yes	Width of IFRAME	
↑		FrameHeight	Integer	Content	Yes	Height of IFRAME	

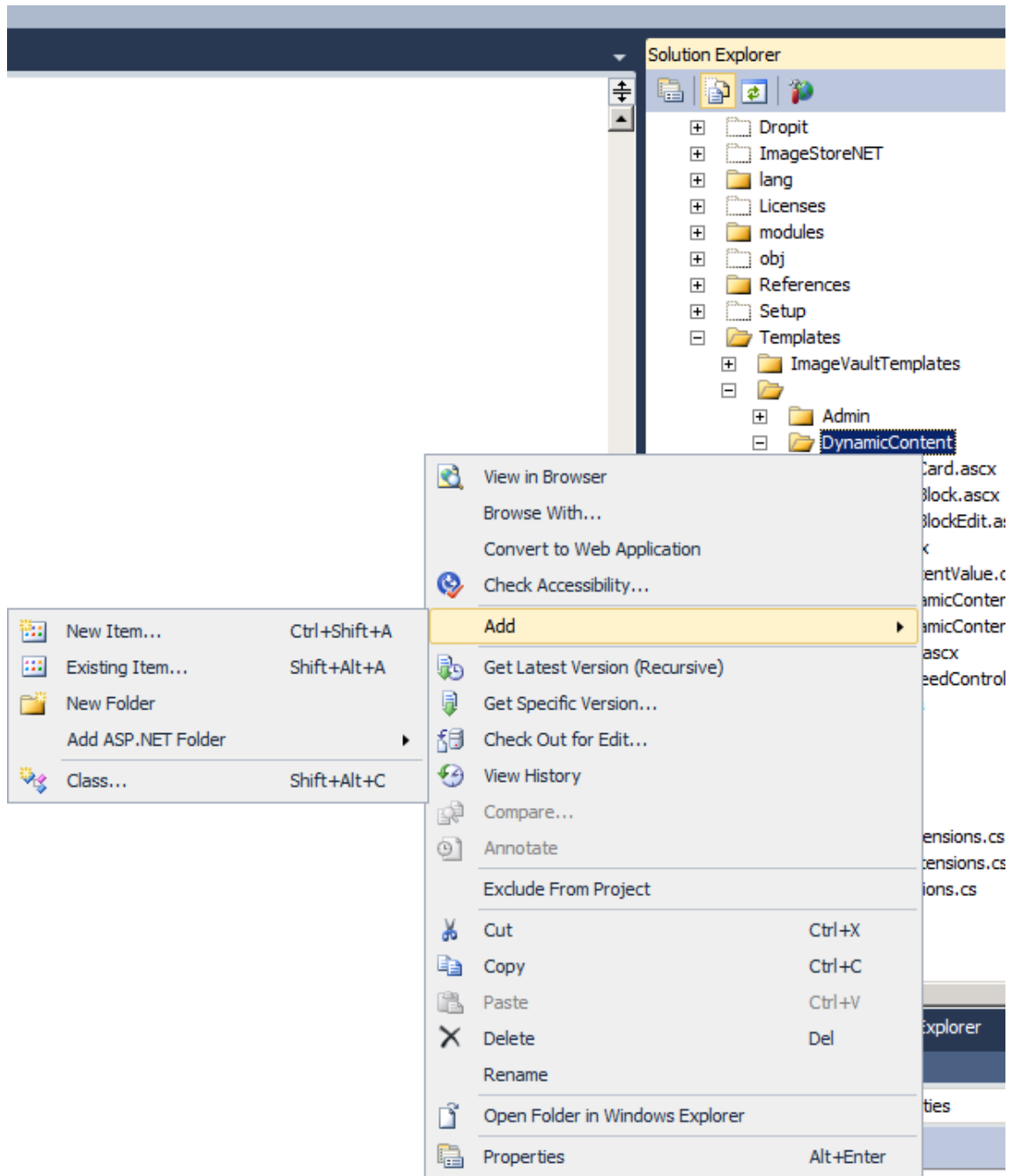
Kuvio 18. Luodun IFRAME-sivun ominaisuudet.

4.3 Twitter feedin luonti dynaamiseksi sisältökomponentiksi

Dynaamisen sisällön eli Dynamic Contentin ideana on tehdä uudelleenkäytettäviä sisällöntuottolohkoja. Niitä voidaan hyödyntää lähes missä vain, mistä löytyy vapaa tekstikenttä, ja sen joustavuuden ansiosta ne ovatkin hyvin suosittuja komponentteja. Dynamic Contentilla on oma paikkansa ohjelmistosovelluksen puuhierarkiassa, jonne kaikki Dynamic Contentin osat tulevat. Seuraavaksi esitellään yksinkertaisen TwitterFeed Dynamic Contentin teko. Sitä havainnollistetaan kuvilla ja koodilla. [5.]

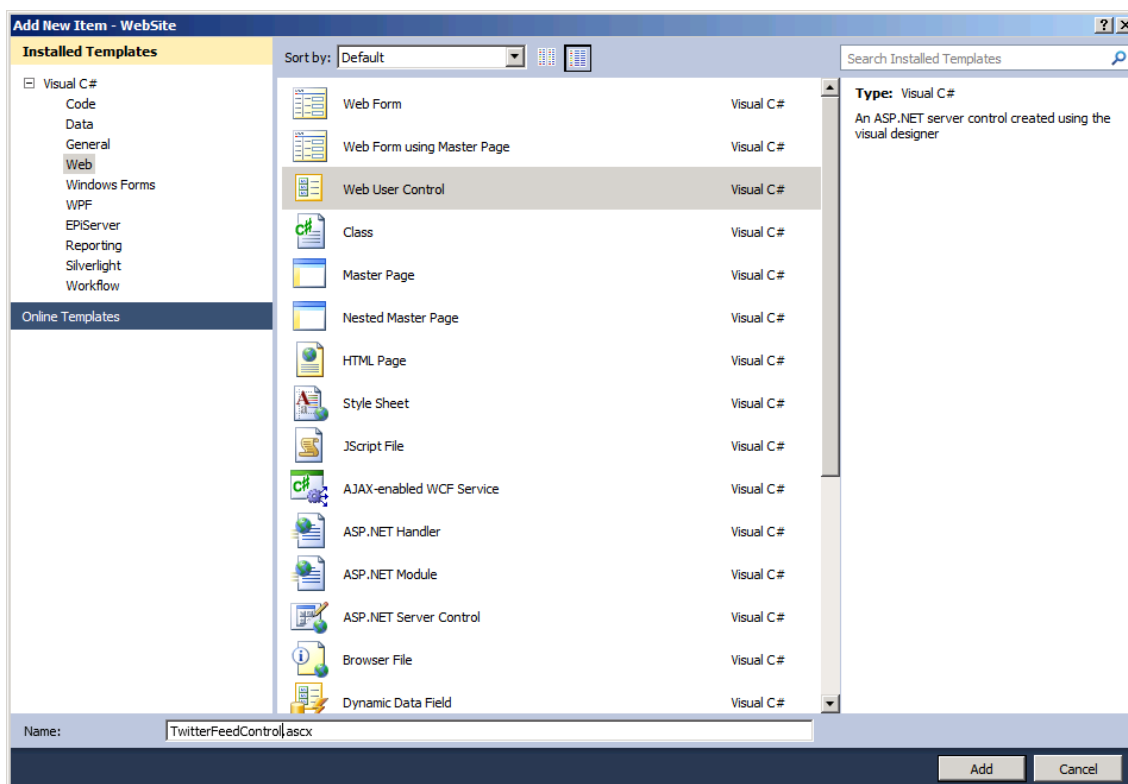
Tiedostojen luonti

Seuraavaksi käydään läpi Dynamic Content -tiedostojen luonti. Kuviossa 19 luodaan Visual Studio -projektiin Templates, <yritys>, Dynamic Content -polkuun uusi tiedosto napsauttamalla kansiota oikealla ja valitsemalla Add, New item.



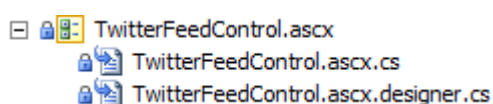
Kuvio 19. Dynamic Content -tiedostojen luonti.

Seuraavaksi valitaan tiedostotyyppi Web User Control ja nimetään se `TwitterFeedControl.ascx`:ksi, jonka jälkeen painetaan Add-painiketta. Tämä on esiteltynä kuviossa 20.



Kuvio 20. Tiedostotyypin valinta Dynamic Contenttia varten.

Puurakenteessa Dynamic Content -hakemistoon ilmestyy TwitterFeedControl.ascx usercontrol, jonka alle luodaan automaattisesti kaksi muuta tiedostoa, joista TwitterFeedControl.ascx.cs:llä luodaan toiminnallisuus TwitterFeedControl.ascx-tiedostolle, kuten kuviossa 21 on esitelty.



Kuvio 21. Dynamic Content -tiedostot.

Tiedostojen kuvaus

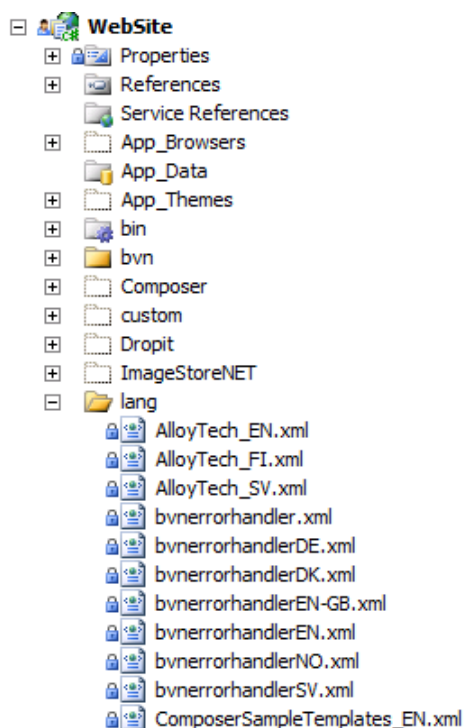
Tiedostojen codebehind ja graafiset ominaisuudet ovat samat kuin edellisessä tiedostojen kuvauksessa (s. 20). TwitterFeedControl.ascx-tiedostoon luodaan paneeli, jonka sisälle upotetaan java script -koodia. Java scripttiin otetaan arvoja codebehindista get- ja set-funktioiden avulla. Skriptiin määritellään, mikä on TwitterFeedin käyttäjä,

jonka päivityksiä näytetään, kuinka monta päivitystä se näyttää ja minkä kokoinen paneeli on. TwitterFeedControl.ascx:n koodi on kommentoituna liitteessä 5.

TwitterFeedControl.ascx.cs codebehind-tiedostossa määritellään useita get- ja set-funktioita, joiden avulla tiedot otetaan ylläpidon puolelta ja välitetään graafiseen Twitter Feed -paneeliin. Tästä esitellään havainnollistava esimerkki myöhemmin. TwitterFeedControl.ascx.cs:n koodi on kommentoituna liitteessä 6.

Kieliasetukset

Kieliasetukset määritellään ohjelmistosovelluksen WebSite-projektin juuresta löytyvän lang-hakemiston alla olevilla .xml-tiedostoilla. Kuviossa 22 näkee, missä sijaitsee ComposerSampleTemplates_EN.xml, johon tehdään lisäyksiä.



Kuvio 22. Kielitiedostot puurakenteessa.

Edellä esiteltyyn englanninkieliseen kielitiedostoon lisätään Twitterin funktioille halutut englanninkieliset nimet common sulkeiden sisään. Vastaavasti, jos halutaan tehdä suomenkielinen versio, niin tehdään kopio englanninkielisestä versiosta ja muutetaan `<language name="English" id="en">` seuraavanlaiseksi: `<language name="suomi" id="fi">`. Tiedostossa on paljon muitakin asetuksia, mutta esittelyssä on ainoastaan

Twitteriä koskevat asetukset. Muutokset näissä kentissä näkyvät www-sivulla olevassa TwitterFeed settings-osiossa, joka esitellään myöhemmin. ComposerSampleTemplates_EN.xml:n koodia on esitelty liitteessä 7.

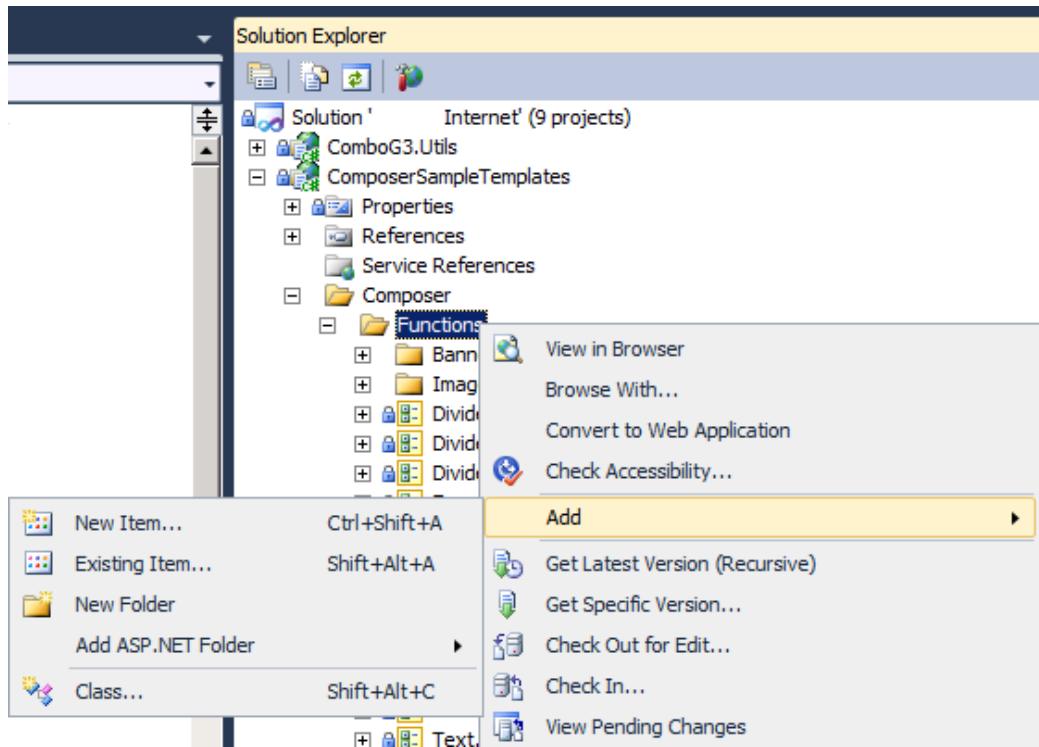
4.4 Composer-tyyppinen sisällöntuottokomponentti

Composer on moduuli EpiServer CMS:lle, ja se tarjoaa sivujen editointiin uuden ominaisuuden. Sen avulla pystyy luomaan nopeasti ja tehokkaasti erilaisia sivukokonaisuuksia. Composer-sivustoilla on työvälineistö (ToolBox), jossa on kaikki moduulin mukana tulleet ja itse luodut lohkot. Composer-sisällöntuottomenetelmän idena on vedä ja pudota -tyylinen (drag and drop) sisällön luonti, sen avulla voi käyttää samoja lohkoja useammassa sivuissa ja myös vaihtaa sivustoilla esiintyvää ulkoasua sen mukaan, mille paikalle lohkot pudotetaan.

Composer-moduulin voi ladata EpiServerin sivuilta ja sen voi lisätä projektiksi Deployment Centerin kautta osaksi Visual Studiossa tehtävää ohjelmistokokonaisuutta. [6.]

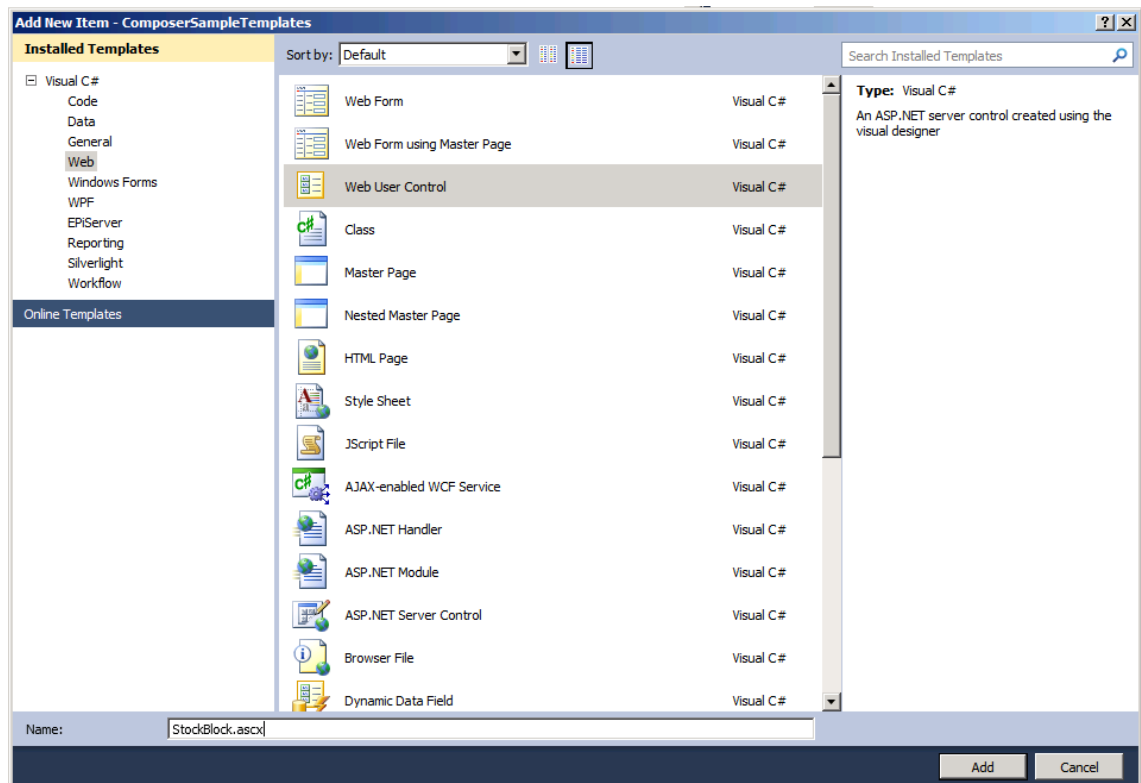
Tiedostojen luonti

Seuraavaksi esitellään yksittäisen Composer-lohkon luonti Visual Studioon. Tiedostot luodaan Composer-projektin alle Composer- ja Functions- kansioon. Tämä esitellään kuviossa 23.



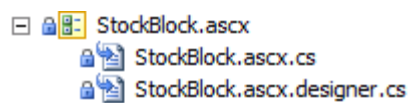
Kuvio 23. Composer-komponentin luonti.

Tiedostotyyppiä valitaan Web User Control ja sille annetaan nimeksi StockBlock.ascx, jonka jälkeen painetaan Add-painiketta. Tämä on esiteltyä kuviossa 24.



Kuvio 24. Tiedostotyypin valinta composer-blockia varten.

Puurakenteessa composer -projektiin Composer- ja Functions-hakemiston alle ilmestyy StockBlock.ascx usercontrol, jonka alle Visual Studio luo automaattisesti kaksi muuta tiedostoa, joista StockBlock.ascx.cs:llä luodaan toiminnallisuus StockBlock.ascx tiedostolle. Kuviossa 25 on esitelty tiedostot.



Kuvio 25. Composer-tiedostot.

Composer standard page -sivutyyppi tulee automaattisesti Composer-moduulin asennuksen yhteydessä EpiServerin ylläpitopuolelle. Sinne tulevat myös valmiit sivun ominaisuudet, kuten kuviossa 26 on esitelty.

The screenshot shows the CMS interface with the 'Page Type' configuration for '[ExtensionSys] Composer standard page'. The left sidebar contains a tree view of page types, with '[ExtensionSys] Composer standard page' selected. The main area displays a table of properties for this page type.

Move Up	Move Down	Name	Type	Tab	Unique value per language	Field name	Help Text
	↓	Heading	String (<= 255)	Content	Yes	Heading	Specify a heading that will present the main title of this page. If this property is not set, the page name will be used as a heading instead.
↑	↓	MainArea	Extension Content Area Property	Composer	No	MainArea	The main area for main content
↑	↓	RightArea	Extension Content Area Property	Composer	No	RightArea	The right area for banners, puffs
↑	↓	ExtensionPageProperty	Extension Page Property	Composer	Yes	Composer edit	Allow to edit Composer functions in edit-on-page mode
↑	↓	SEOTitle	String (<= 255)	SEO	Yes	Page title	The title of the page
↑	↓	SEODescription	String (<= 255)	SEO	Yes	Page description	Description of the page
↑		SEORobots	Drop-down list	SEO	Yes	Search robots	Specify how search robots should handle the page

Kuvio 26. EpiServerin sivustotyyppi Composer standard page.

Kuviossa 27 on esitelty kuvion 26 settings-painikkeen takana oleva näkymä. Tästä näkymästä voidaan vaihtaa tiedostopolku, jolla määritellään, mikä kooditiedosto linkitetään Composer Standard pageen.

The screenshot shows the 'Edit' configuration for the '[ExtensionSys] Composer standard page'. The left sidebar is the same as in Kuvio 26. The main area displays the 'Edit' configuration for the page type, with the 'Information' tab selected.

Edit "[ExtensionSys] Composer standard page"

Edit the basic information about the page type.

Information | Default Values | Available Page Types

Name: [ExtensionSys] Composer standard page

Description: Create standard Composer page

File name: /composer/pages/page.aspx

Sort index: 500

Available in Edit mode

Access level: Create Select all

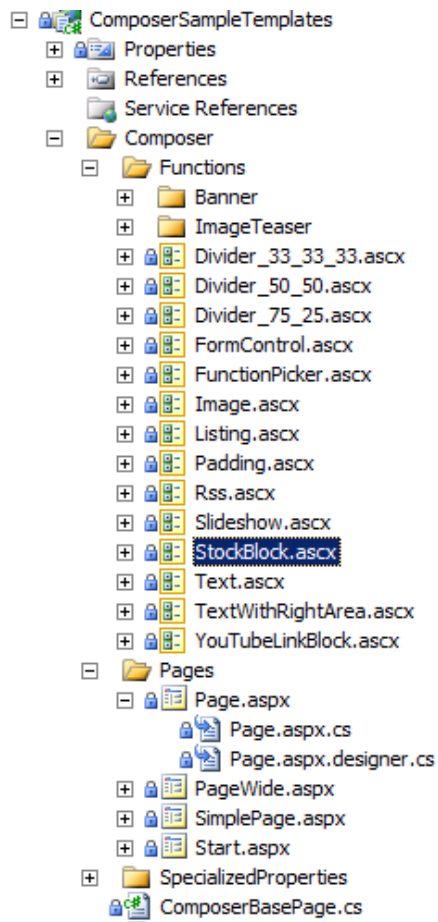
Everyone

Add Users/Groups

Save Delete Cancel

Kuvio 27. Composer standard pagen perusinformaatio.

Page.aspx, johon EpiServerin Composer standard page on linkitetty, löytyy projektin puurakenteesta Composer-projektin alta, Composer- ja Pages-hakemistojen alapuolelta. Kuvio 28 havainnollistaa edellä mainittua informaatiota.



Kuvio 28. Composerin puurakenne.

Tiedostojen kuvaus

StocBlock.ascx-tiedostossa määritellään, minkälaiselta kyseinen komponentti näyttää, kun se on asetettu sivulla. Se koostuu yksinkertaisista div-elementeistä, joihin nostetaan tietoa StockBlock.ascx.cs codebehind -tiedostosta. StockBlock.ascx:n koodia voi tarkastella liitteessä 8.

StockBlock.ascx.cs-tiedosto hakee määritetyltä sivulta osaketiedot ja sen jälkeen välittää ne StockBlock.ascx:lle näytettäväksi. Sivun, josta tiedot haetaan, määritellään EpiServerin web.config-tiedostossa (ks. s. 15 löytääksesi web.config-tiedoston). Esimerkkikoodi on kommentoituna liitteessä 9.

5 Sisällön tuottaminen EpiServeriin

Sisällön tuottamisen ideana on luoda uutta informaatiota sivustolle sekä päivittää jo olemassa olevaa dataa ajankohtaiseksi. Koska kyseessä on sisällöntuottojärjestelmä, on sisällöntuotto tehty mahdollisimman yksinkertaiseksi. Jokaista sivua sisältöineen ei tarvitse kirjoittaa erikseen, vaan valmiiksi ohjelmoiduista sivupohjista voi valita itselleen sopivimman vaihtoehdon ja luoda sivuun haluttua sisältöä. Tässä esitellään, kuinka sisältöä tuotetaan edellä esiteltyihin komponentteihin.

EpiServerissä sisällöntuottaminen tapahtuu selaimen kautta ensin kirjautumalla sisään sivustoon. Tämän jälkeen on mahdollista luoda uusia sivuja sivustorakenteeseen tietyillä sivustopohjilla sekä muokata ja päivittää vanhaa sisältöä. Sisällönluontia varten on kahdentyyppisiä sivupohjia, niin sanotut tavalliset sivupohjat ja Composer-tyyppiset sivupohjat. Tavalliset sivupohjat koostuvat user control -tiedostoista, jotka on aseteltu sivupohjille valmiisiin paikkoihin. User control voi olla esimerkiksi kalenteri tai aiemmin esitelty Iframe-nostokomponentti. Sen sijaan Composer-sivupohjiin voi vetää ja pudottaa sisältöä Composer-komponenttikirjastosta (ToolBox) mihin tahansa järjestykseen sivulle määrättyjen rajausten puitteissa. Composer-pohjiin on tehty erikokoisten komponenttien vuoksi rajauksia, jotta pudotetut komponentit eivät aiheuttaisi särkynyttä kokonaisuutta.

5.1 Iframen sisällöntuotto

Iframe-sivulle sisällöntuottaminen tapahtuu luodun Iframe-tyyppisen sivun edit-modessa. Ensin luodaan sivu napsauttamalla hiiren oikealla painikkeella puurakenteesta sivua, jonka alle kyseinen sivu halutaan luoda. Listasta valitaan Iframe-tyyppinen sivu, jonka jälkeen ilmestyy kuviossa 29 esitelty Iframe -sivun edit-valikko. Siitä löytyy vakiokentät, sivun nimi puurakenteessa ja sivun otsikko sekä tekstikenttä. Varsinaiseen Iframe usercontroliin liittyvät kohdat ovat kolme alinta tekstikenttää, www-osoite, leveys ja korkeus. Www-osoitekenttään (URL) valitaan osoite, josta iframe-syöte halutaan nostaa luotavalle sivulle. Tässä tapauksessa nostetaan Logican logo .jpg -tiedostona asettamalla osoitekenttään osoite, jossa kuva sijaitsee. Leveys- (Width of IFrame) ja korkeus (Height of IFrame) -kentissä määritellään nostettavan objektin, tässä tapauksessa kuvan, koko pikseleissä nostoa varten.

Creating New Page
Page Type: [] **Iframe** Status:

Edit

Save Save and View Save and Publish Cancel

Content Scheduling Settings Shortcut Categories

Name ("PageName")

Heading ("Heading")

Main body ("MainBody")

Tässä on MainBody tekstiä.

Path: p

Url ("Link")

Width of IFrame ("FrameWidth")

Height of IFrame ("FrameHeight")

Kuvio 29. Uuden sivun luonti Iframe-sivutyyppiä käyttäen.

Kun sivu julkaistaan Save and Publish -painikkeella, siirrytään selaimessa luodulle sivulle. Sivulle ilmaantuu Logican mustalla väritetty logo tietyillä leveys- ja korkeusehdoilla. Kuviossa 30 on esitelty sivu, jossa on Logican logo.

IFRAME_KOKEILU

Tässä on MainBody tekstiä.

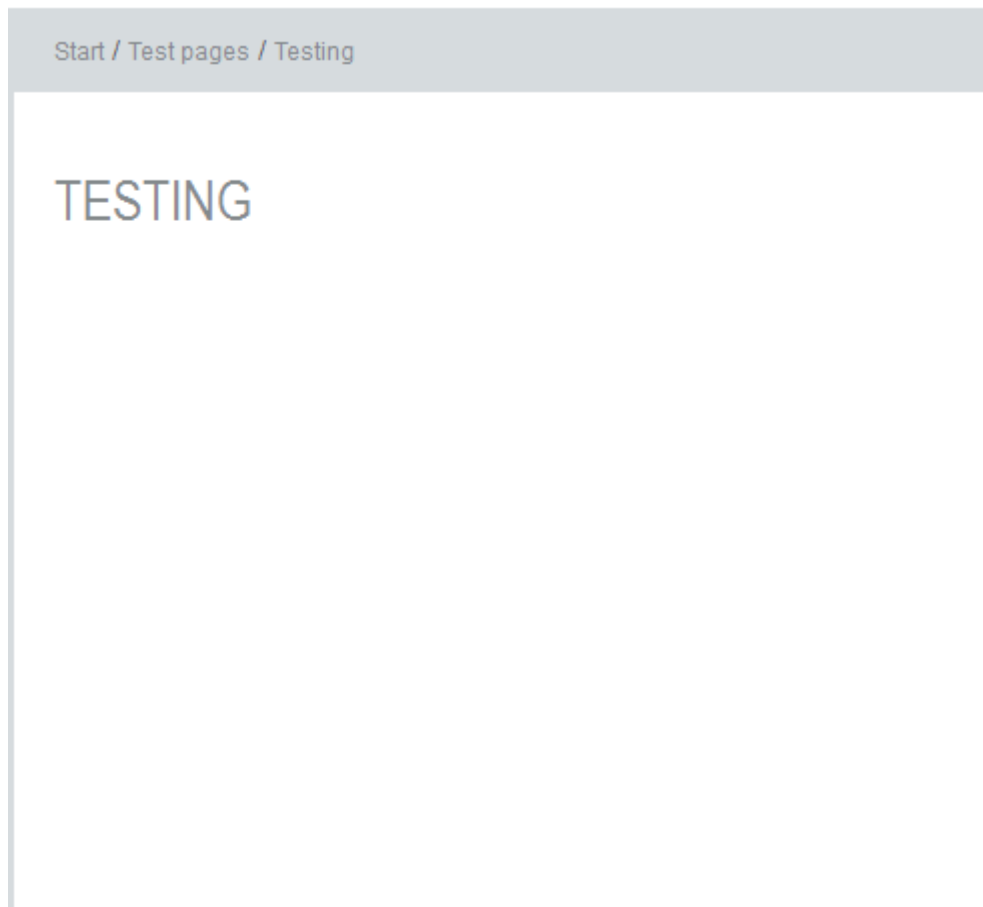


Kuvio 30. Esimerkki IFrame-sivusta, jossa Logican logo. [7]

5.2 Sisällöntuotto Dynamic Content -komponentilla

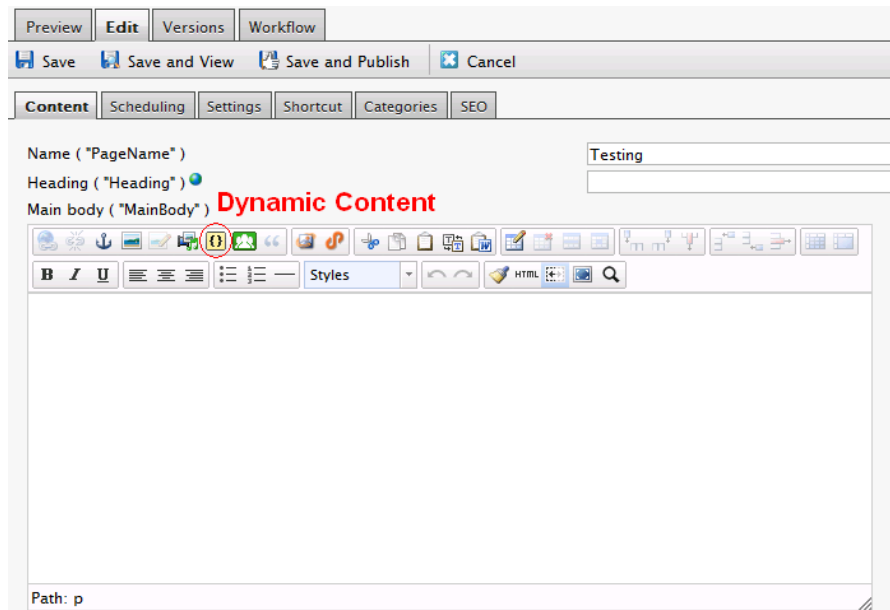
Dynaamisen sisällön luominen Web-sivulle tapahtuu ylläpidossa edit-tilassa. Dynaaminen sisältö voidaan lisätä osaksi vapaaseen sisällöntuottoalueeseen. Vapaaseen sisällöntuottoalueeseen voi dynaamisen sisällön lisäksi lisätä normaalia tekstiä, kuvia ja taulukoita. Seuraavaksi havainnollistetaan sisällöntuotto Dynamic Content -komponentilla kuvia apuna käyttäen.

Kuviossa 31 näytetään tyhjä sivu, johon TwitterFeed dynamic content -komponentti lisätään.



Kuvio 31. Tyhjä standardisivu.

Seuraavaksi navigoidaan EpiServerin ylläpidon kautta aiemmin esitellyn sivun Edit-tilaan, josta näkee, mitä sivulle voi lisätä. Nimen ja otsikon lisäksi sivulla on Main Body -kenttä, joka on vapaa sisällöntuottoalue. Jotta voidaan lisätä dynaamista sisältöä sivulle, valitaan Main Bodyn sisältä Dynamic Contentin valikko Dynamic Contentille osoitetun painikkeen avulla. Kuviossa 32 esitellään standardisivun (standard page) Edit-tila.



Kuvio 32. Tyhjän standardisivun edit-tila.

Seuraavaksi valitaan haluttu Dynamic Content -komponentti Dynamic Content -valikosta, tässä tapauksessa valitaan tyypiksi Twitter Feed. Kyseisen komponentin teko on esitelty sivulla 24. Kun tietty komponentti on valittu, Dynamic Content -valikko laajenee alaspäin, minne ilmestyy valitun komponentin ominaisuudet. Twitter Feed -komponentissa voi valita, minkä Twitter-profiilin päivityksiä näytetään, mikä on sivulla näytettävän komponentin otsikko, kuinka monta päivitystä näytetään, mikä on paneelin korkeus sekä leveys ja mikä on sivun päivitystiheys. Kun asetukset ovat valmiit ja halutaan siirtyä eteenpäin, painetaan OK-painiketta. Kuviossa 33 on esitelty, miltä Twitter Feed -komponentin sisällöntuottoasetukset näyttävät.

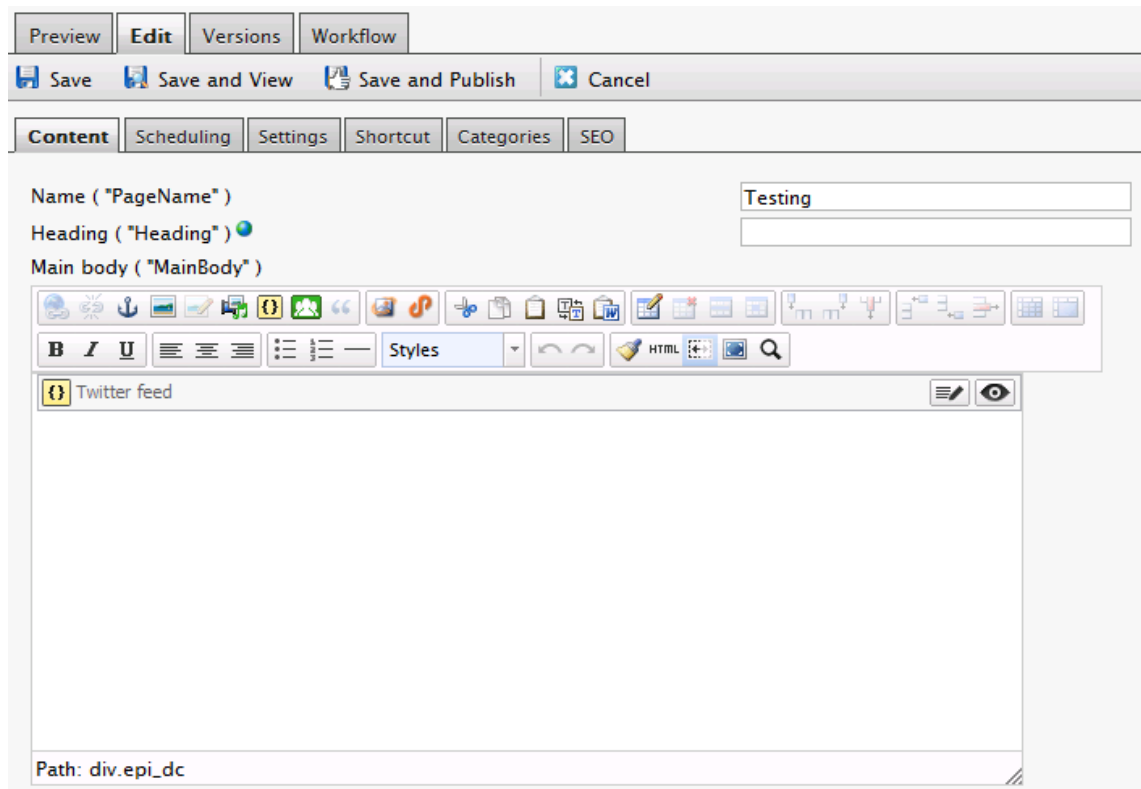
The screenshot shows a web browser window titled "Dynamic Content - Mozilla Firefox". The main content area is divided into three sections:

- Dynamic Content:**
 - Type: Twitter feed (selected in a dropdown menu)
 - Description: Displays a Twitter feed
- Settings:**
 - Twitter Profile: Logica
 - Panel Title: Logica
 - Panel Subject: (empty)
 - Number of Feeds: 4
 - Panel Height: 350
 - Panel Width: 300
 - Update Interval (ms): 3000
- Personalization Settings:**
 - Select the visitor group(s) that should see this content: No group selected (with a green plus icon)
 - Personalization groups are used to group personalized content for different visitor groups, or no visitor groups.
 - Include in a personalization group (with a dropdown menu and a green plus icon)
 - As an option, you can display fallback content to visitors not matching a visitor group. To do so, include the fallback content in a personalization group but leave the visitor group field empty.

At the bottom right, there are buttons for "OK", "Cancel", and a help icon (question mark).

Kuvio 33. Twitter Feed -komponentin sisällöntuottoasetukset.

Seuraavaksi vapaalle sisällöntuottoalueelle ilmestyy Twitter Feed -komponentti, jota ennen tai sen jälkeen voidaan lisätä tekstiä, taulukoita tai uusia Dynamic Content -komponentteja. Kun painetaan Save and Publish -painiketta, sivu julkaistaan. Kuvio 34 tarkentaa edellä kerrottua.

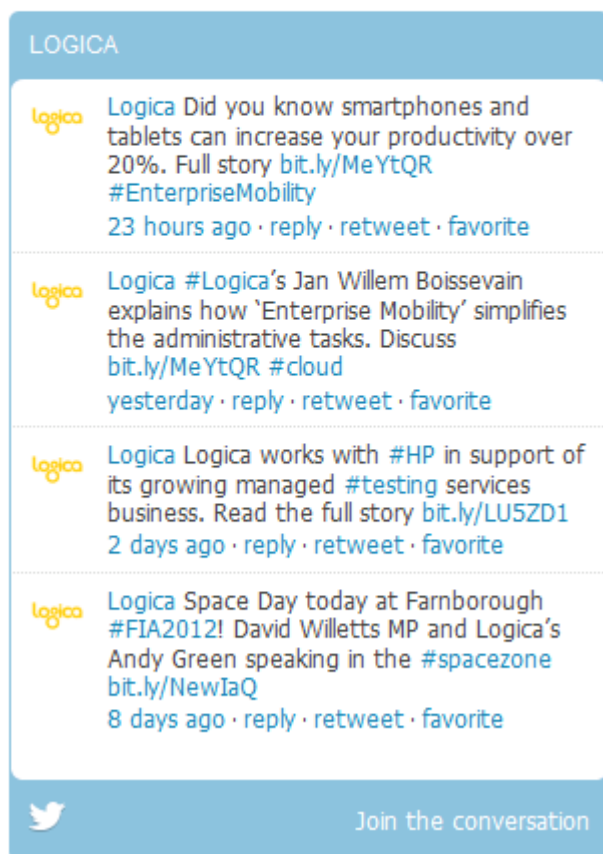


Kuvio 34. Standardisivun edit-tila Twitter Feed -komponentin kanssa.

Julkaisun jälkeen selain vie käyttäjän juuri luodulle sivulle, jossa nähdään, millaiselta Twitter Feed näyttää. Kuviossa 35 esitellään esimerkki Twitter Feedistä.

Start / Test pages / Testing

TESTING

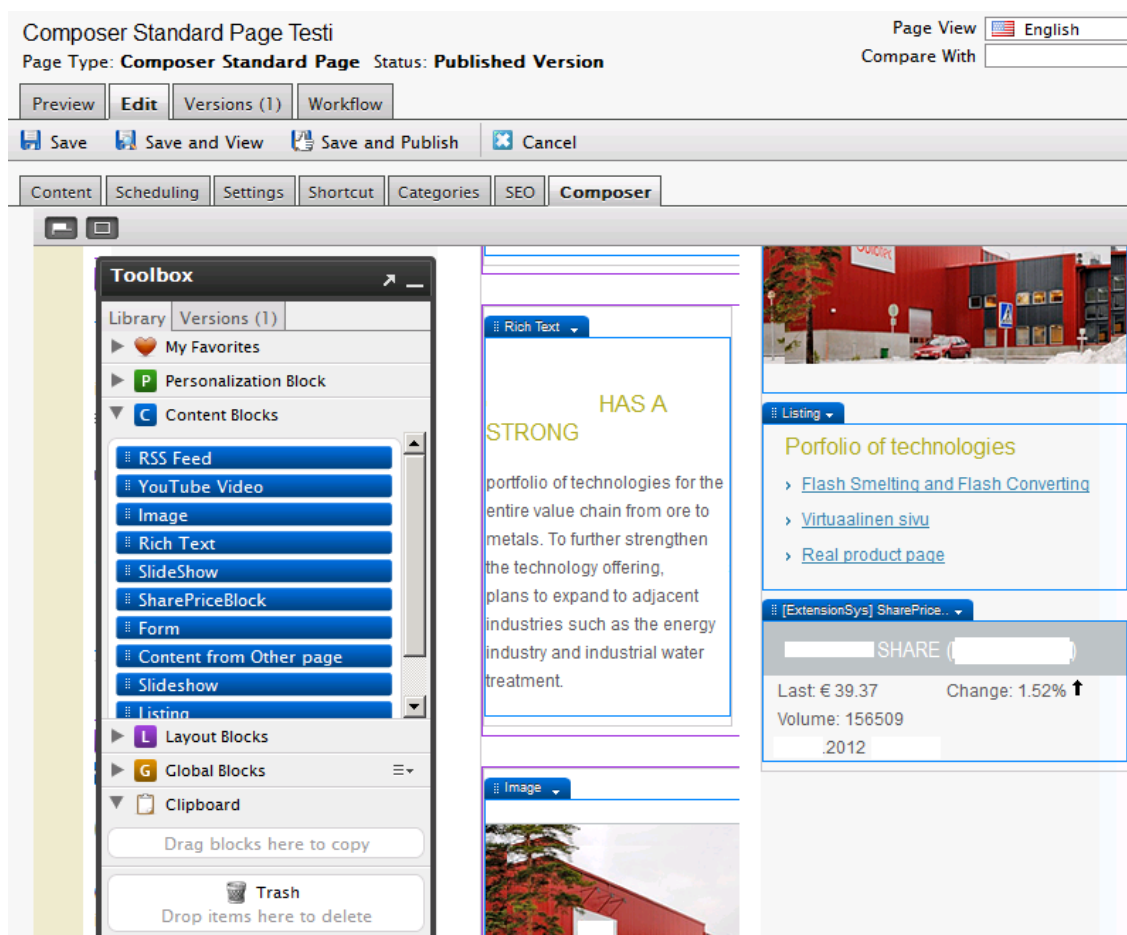


Kuvio 35. Twitter Feed -komponentti nostettuna sivulle.

Dynaamista sisältöä suositaan paljon silloin, kun halutaan näyttää samalla kaavalla sisältöä useammalla sivulla. Esimerkiksi Google maps -toiminto voidaan toteuttaa Dynamic Contentin menetelmällä. Google maps -toiminnon avulla voidaan näyttää esimerkiksi yritysten konttorien sijainnit. Vaikka komponentin pohja on molemmille sama, asetukset (tässä tapauksessa kartalle näytettävä osoitekenttä) näytettäville kartoille ovat kuitenkin erilaiset.

5.3 Composer

Composer-sivulle sisällöntuotto on hyvin yksinkertaista. Ensin navigoidaan Composer-tyyppiselle sivulle sivustohierarkiassa ja mennään Edit-tilaan. Edit-tilaan avautuu välilehtiä, joista yksi on Composer. Kun tämä valitaan, sivu avautuu Composer-sisällöntuottotilaan. Sivulla on esillä työkaluvalikko (Toolbox), josta voi vetää ja pudottaa Composer-komponentteja sivulle. Aiemmin esitelty StockBlock-komponentti (s. 28) esiintyy työkaluvalikossa nimellä ”SharePriceBlock”. Kun kyseinen komponentti vedetään ja pudotetaan sivulle työkaluvalikosta, tuloksena on osaketietolohko, jossa näkyy osakkeen tietoja, kuten hinta ja muutos. Lohkon ominaisuuksia voi säätää painamalla pientä nuolta alaspäin sinisellä pohjalla Composer-lohkon yhteydessä. Selitystä tarkentaa graafinen esitys kuviossa 36.



Kuvio 36. Composersivun -sisällöntuottonäkymä.

6 Yhteenveto

Insinööriyössä käytiin läpi EpiServerin esittely, erilaiset käyttötarkoituksesta riippuvat vaatimukset, asennus ja käyttöönotto. Pääpainona työssä oli kolmen erilaisen www-osan tekeminen. Aluksi kerrottiin yleisesti www-komponenteista, minkä jälkeen siirryttiin yksityiskohtaisemmin kolmen erityyppisen www-komponentin tekemiseen pienten johdantojen saattelemina. Työssä esiteltiin kyseisten komponenttiratkaisujen käyttömahdollisuuksia ja esimerkkikoodien avulla niiden ohjelmallista toteutusta. Varsinaisessa sisällöntuotto –osuudessa tarkasteltiin, kuinka insinööriyössä esitellyille komponenteille luodaan sisältöä, ja havainnollistettiin kuvien avulla, minkälaisia ovat komponenttien valmiit sisällöt.

Insinööriyö onnistui suunnitellusti, eli tuloksena oli selkeä ohjekokonaisuus komponenttien tekemisestä ja sisällöntuottamisesta. Vastaavanlaista ohjekokonaisuutta ei ole aiemmin ollut, verkosta löytyy pelkästään yksittäisiä suppeasti laadittuja ohjeita.

Työn tekoa rajoitti lisenssien tarve, näin ollen kotiloissa oli vaikea tehdä asiaan liittyvää testausta tai materiaalin tuottoa. Insinööriyötä voi hyödyntää parhaiten työpaikoilla, joissa tarjotaan EpiServer-ratkaisuja asiakkaille. Etenkin uusien kehittäjien on hyvä tutustua työhön, jotta he saisivat paremman kokonaiskuvan, millaisia vaiheita EpiServerin komponenttien kehitys ja sisällöntuotto sisältää.

Ennen insinööriyön aloittamista minulla ei ollut omakohtaista kokemusta EpiServerin sovelluskehityksestä, ainoastaan testaamisesta ja yksinkertaisesta sisällöntuotosta. Näin ollen työn tekemisen myötä kokonaiskuva EpiServeristä hahmottui selkeämmäksi, kun ohjelmistopuoli tuotiin mukaan kuvioihin. Tämän johdosta C#-ohjelmointitaidot kehittyivät entisestään.

Lähteet

- 1 About Us 2012. Verkkodokumentti. EpiServer. <<http://www.episerver.com/About-Us/>>. Luettu 18.07.2012.
- 2 Esittelyssä EpiServer 6 julkaisujärjestelmä 2010. Verkkodokumentti. Vierityspalkki. <<http://vierityspalkki.fi/2010/10/11/esittelyss-episerver-6-julkaisujrjestelm/>>. Luettu 25.09.2012.
- 3 System Requirements for EpiServer CMS 6 R2 2010. Verkkodokumentti. EpiServer. <<http://world.episerver.com/Documentation/Items/System-Requirements/EPiServer-CMS/Version-6/System-Requirements---EPiServer-CMS-6-R2/>>. Luettu 18.07.2012.
- 4 Deployment Center 2011. Verkkodokumentti. EpiServer. <http://world.episerver.com/Documentation/Items/Tech-Notes/EPiServer-CMS-6/EPiServer-CMS-6-R2/EPiServer-Deployment-Center/#install_cms>. Luettu 18.07.2012.
- 5 EpiServer CMS Dynamic Content 2007. Verkkodokumentti. EpiServer. <<http://world.episerver.com/Documentation/Items/Tech-Notes/EPiServer-CMS-5/EPiServer-CMS-5-R2-CTP/EPiServer-Dynamic-Content/>>. Luettu 20.8.2012.
- 6 EpiServer Composer 4 2010. Verkkodokumentti. EpiServer. <<http://world.episerver.com/Articles/Items/EPiServer-Composer-4/>>. Luettu 20.8.2012.
- 7 Logica Public Server 2012. Verkkokuva. Logica. <<http://public.logica.com/>>. Luettu 20.8.2012.
- 8 IIS Application pool 2010. Verkkodokumentti. Microsoft <[http://technet.microsoft.com/en-us/library/cc735247\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc735247(v=ws.10).aspx)>. Luettu 29.12.2012

Sivupohjan graafinen määrittäminen (Iframe.aspx)

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Iframe.aspx.cs"
Inherits="EPiServer.Templates.<yritys>.Pages.IFrame" MasterPage-
File="~/Templates/<yritys>/MasterPages/MasterPage.master" %>
// Määritellään codebehind, kieli, perintä ja käytettävä masterpage.
```

```
<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assem-
bly="EPiServer" %>
// Määritellään nimiavaruuteen liittyvät asiat.
```

```
<%@ Register TagPrefix="<yritys>" TagName="MainBody"
Src="~/Templates/<yritys>/Units/Placeable/MainBody.ascx" %>
// Määritellään sivu käyttämään osanaan controllia nimeltä MainBody tietystä polusta
```

```
<%@ Register TagPrefix="<yritys>" TagName="IFrame"
Src="~/Templates/<yritys>/Units/Placeable/IFrame.ascx" %>
// Määritellään sivu käyttämään osanaan IFrame controllia tietystä polusta
```

```
<asp:Content ContentPlaceHolderID="MainBodyRegion" runat="server">
//Tässä viitataan masterpagen MainBodyRegioniin jonka sisälle tulevat MainBody –ja
//Iframe controllit ( keskelle sivua )
```

```
<div id="MainBody">

    <<yritys>.MainBody runat="server" />
    //Määritellään MainBody controllin sijainti koodissa
    <<yritys>.IFrame
        runat="server"
        IFrameProperty="IFrame"
        HeadingProperty="IFrameHeading"
        ShowStatistics="false">
    </<yritys>.IFrame>
    // Määritellään IFrame controllin sijainti koodissa sekä muita ominaisuuksia

</div>
</asp:Content>
```

Sivupohjan toiminnallinen määrittäminen (Iframe.aspx.cs)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using EPiServer;
using EPiServer.Core;

// valitaan tarvittava kirjasto

namespace EPiServer.Templates.<yritys>.Pages
{
    public partial class IFrame : TemplatePage
    {
        // Sivutyypit periytetään TemplatePage:sta
        protected void Page_Load(object sender, EventArgs e)
        {

        }
    }
}
```

User Control:n graafinen määrittäminen (Iframe.ascx)

```
<%@ Control Language="C#" AutoEventWireup="false" CodeBehind="Iframe.ascx.cs"
Inherits="EPiServer.Templates.<yritys>.Units.Placeable.IFrame" %>
// Määritellään codebehind, kieli ja perintä.

<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assem-
bly="EPiServer" %>

<asp:Panel ID="Paneeli" runat="server">
// Määritellään asp paneeli jonka sisälle iframe tulee
  <iframe id="iframe"
    name="iframe"
    src="<%=IframeUrl%>"
    width="<%=IframeWidth%>"
    height="<%=IframeHeight%>"
    frameborder="0">
  </iframe>
// Määritellään iframelle muuttujien avulla source ja framen koko.

</asp:Panel>
```

User Control:n toiminnallinen määrittäminen (Iframe.ascx.cs)

```
using System;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using EPiServer;
using ComboG3.Utils.AppParameters;

namespace EPiServer.Templates.<yrittys>.Units.Placeable
{
    public partial class IFrame : UserControlBase
    {
        public string IframeUrl    { get; set; }
        public string IframeWidth  { get; set; }
        public string IframeHeight { get; set; }
        public string LinkText     { get; set; }
        // yksinkertaiset getterit ja setterit joilla controloidaan iframen ulkoasua ja
        //kohdetta

        protected HtmlGenericControl PageHeaderP;

        private void Page_Load(object sender, System.EventArgs e)
        {
            // sivun latauksen yhteydessä suoritettavat koodit.
            IframeUrl = AddParametersToIframeUrl();
            // Kutsutaan funktiota jonka palautus arvo IframeUrl:iin

            if (CurrentPage.Property["FrameHeight"] != null && CurrentPage.Property["FrameHeight"].Value != null)
                IframeHeight = CurrentPage.Property["FrameHeight"].Value.ToString();
            // Otetaan arvo IframeHeight:iin sivun ominaisuuksissa olevasta
            // FrameHeight nimisestä muuttujasta (esitellään myöhemmin)
            else IframeHeight = "480";

            if (CurrentPage.Property["FrameWidth"] != null && CurrentPage.Property["FrameWidth"].Value != null)
                IframeWidth = CurrentPage.Property["FrameWidth"].Value.ToString();
            // Otetaan arvo IframeWidth:iin sivun ominaisuuksissa olevasta
            // FrameWidth nimisestä muuttujasta (esitellään myöhemmin)
            else IframeWidth = "100%";

            //Show/hide headline
            if (CurrentPage["Heading"] == null)
                PageHeaderP.Visible = false;
        }
    }
}
```

```
        this.DataBind();
        // Liitetään määritellyt arvot sivuun.
    }

private string AddParametersToIframeUrl()
{
    // Funktiossa laitetaan URL käyttäytymään niin kuin halutaan
    string newQueryString = "";
    string iFrameUrl;
    if (CurrentPage.Property["Link"] != null &&
        CurrentPage.Property["Link"].Value != null)
    {
        iFrameUrl = CurrentPage.Property["Link"].Value.ToString();
        //Otaa kaikki parametrit URL: sta paitsi ID:n
        IEnumerator qs = Request.QueryString.GetEnumerator();
        while (qs.MoveNext())
        {
            if (qs.Current.ToString().ToLower() != "id")
                newQueryString += qs.Current.ToString() + "=" + Request.QueryString[qs.Current.ToString()].ToString() + "&";
        }

        //Poista viimeinen "&"
        if (newQueryString.EndsWith("&"))
            newQueryString = newQueryString.Substring(0,
                newQueryString.Length - 1);

        //Tarkistaa kumpaa pitäisi käyttää "?" vai "&"
        if (newQueryString.Length == 0) return iFrameUrl;
        else if (iFrameUrl.IndexOf("?") > 0)
            return iFrameUrl + "&" + newQueryString;
        else
            return iFrameUrl + "?" + newQueryString;
    }
    else return "";
}

#region Web Form Designer generated code
override protected void OnInit(EventArgs e)
{
    InitializeComponent();
    base.OnInit(e);
}

private void InitializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
}
#endregion

}
}
```

Dynaamisen sisältökomponentin graafinen määrittäminen (TwitterFeedControl.ascx)

```
<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="TwitterFeedControl.ascx.cs" Inherits="EpiServer.Templates.<yritys>.DynamicContent.TwitterFeedControl" %>

<asp:Panel ID="TwitterFeedPanel" runat="server" Visible="true">
<script charset="utf-8" src="http://widgets.twimg.com/j/2/widget.js"></script>
<script>
    new TWTR.Widget({
        version: 2,
        type: 'search',
        search: 'from:"<%=TwitterProfile%>"',
        rpp: "<%=TwitterNumber_Of_Feeds %>",
        interval: "<%=TwitterInterval %>",
        title: '<%=TwitterTitle %>',
        subject: '<%=TwitterSubject%>',
        width: "<%=TwitterWidth %>",
        height: "<%=TwitterHeight %>",
        theme: {
            shell: {
                background: '#8ec1da',
                color: '#ffffff'
            },
            tweets: {
                background: '#ffffff',
                color: '#444444',
                links: '#1985b5'
            }
        },
        features: {
            scrollbar: true,
            loop: false,
            live: true,
            behavior: 'all'
        }
    }).render().start();
</script>
</asp:Panel>
```

Dynaamisen sisältökomponentin toiminnallinen määrittys (TwitterFeedControl.ascx.cs)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using EPiServer.Core;
using EPiServer.DynamicContent;

namespace EPiServer.Templates.<yritys>.DynamicContent
{
    [DynamicContentPlugIn(DisplayName = "Twitter feed", Description = "Displays a
    Twitter feed", ViewUrl = "~/Templates/<yritys>/ Dynamic-
    Content/TwitterFeedControl.ascx")]

    public partial class TwitterFeedControl : UserControlBase
    {

        protected override void OnLoad(EventArgs e)
        {
            base.OnLoad(e);
        }
        // tarkasteltava profiili
        public string TwitterProfile
        {
            get;
            set;
        }
        public string TwitterTitle
        {
            get;
            set;
        }
        public string TwitterSubject
        {
            get;
            set;
        }
        // feedien lukumäärä.
        public int TwitterNumber_Of_Feeds
        {
            get;
            set;
        }
        // paneelin korkeus pikseleissä
```



```
public int TwitterHeight
{
    get;
    set;
}
// paneelin leveys pikseleissä
public int TwitterWidth
{
    get;
    set;
}
// päivitystiheys millisekunnissa
public int TwitterInterval
{
    get;
    set;
}
}
}
```

**EpiServer:n englanninkielisen kielitiedoston määrittäminen
(ComposerSampleTemplates_EN.xml)**

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<languages>
  <language name="English" id="en">
    <function>
      <rss>
        <errorloading>Error when loading RSS data:
          &lt;br/&gt;&lt;b&gt;{0}&lt;/b&gt;
        </errorloading>
      </rss>
    </function>
    <pagetypes>
    <common>
      <property name="TwitterHeight">
        <caption>Panel Height</caption>
        <help>Panel Height</help>
      </property>
      <property name="TwitterWidth">
        <caption>Panel Width</caption>
        <help>Panel Width</help>
      </property>
      <property name="TwitterNumber_Of_Feeds">
        <caption>Number of Feeds</caption>
        <help>Number of Feeds</help>
      </property>
      <property name="TwitterProfile">
        <caption>Twitter Profile</caption>
        <help>Twitter Profile</help>
      </property>
      <property name="TwitterInterval">
        <caption>Update Interval (ms)</caption>
        <help>Update Interval (ms)</help>
      </property>
      <property name="TwitterSubject">
        <caption>Panel Subject</caption>
        <help>Panel Subject</help>
      </property>
      <property name="TwitterTitle">
        <caption>Panel Title</caption>
        <help>Panel Title</help>
      </property>
    </common>
    .....
  </pagetypes>
</language>
</languages>
```

Composer-komponentin graafinen määrittys (StockBlock.ascx)

```
<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="StockBlock.ascx.cs" Inherits="EPiServer.ComposerSampleTemplates.Composer.Functions.StockBlock" %>
<%@ Import Namespace="EPiServer.Templates.<yritys>" %>

<div>
  <div class="shareheading">
    <h2><Extension:Property ID="Property4" PropertyName="Heading"
      runat="server" /></h2>
  </div>

  <div>
    <div class="shareleft">Last: € <asp:Literal ID="Bid" runat="server" /></div>
    <div class="shareright"> Change: <asp:Literal ID="ChangePercent"
      runat="server" />% <asp:Image ID="Changelmg" runat="server"/>
    </div>
  </div>

  <div>
    <div class="shareleft">Volume: <asp:Literal ID="Volume" runat="server" /></div>
  </div>
  <div class="sharedate">
    <asp:Literal ID="Date" runat="server" />
  </div>
</div>
```

Composer-komponentin toiminnallinen määrittäminen (StockBlock.ascx.cs)

```
// koodissa käytettävät kirjastot
using System;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using EPiServer;
using EPiServer.Core;
using EPiServer.DataAbstraction;
using EPiServer.Web.WebControls;
using System.Xml;
using Dropit.Extension.Core;
using <yrittys>.Intranet.BL;
using EPiServer.Templates.<yrittys>;
using ComboG3.Utils.AppParameters;

namespace EPiServer.ComposerSampleTemplates.Composer.Functions
{
    public partial class StockBlock : BaseContentFunction
    {
        // koodissa käytettävät muuttujat
        protected Literal Bid;
        protected Literal ChangePercent;
        protected Literal Volume;
        protected Literal Date;
        protected Image ChangeImg;
        private XmlDocument _xmlDoc;
        private HttpAccess httpData;

        // Sivun latauksen yhteydessä tapahtuvaa
        protected void Page_Load(object sender, EventArgs e)
        {
            try
            {
                if ((System.Web.Configuration.WebConfigurationManager.AppSettings
                    ["ShareInfoUrl"]) != null)
                {
                    LoadShare();
                }
            }
            catch (Exception)
            {
                return;
            }
        }
    }
}
```

```
    }
  }
  // Lataa XML-dataa nykyisellä osaketiedoilla ja täyttää .ascx:än tiedot.
  private void LoadShare()
  {
    // ShareInfoUrl löytyy web.config tiedostosta <AppSettings> tagine sisältä
    // <add key="ShareInfoUrl"
    // value="http://www.euroland.com/tickerxml/?companycode%<XXXXXX>" />

    string customSetting = Decode(System.Web.Configuration.
      WebConfigurationManager.AppSettings["ShareInfoUrl"]);
    bool proxySetting = false;
    if(bool.TryParse(AppParameterController.GetAppParameterValue("UseProxy"),
      out proxySetting))
    {
      httpData = new HttpAccess((customSetting), proxySetting, 120, false, 10);
      _xmlDoc = httpData.Xml;
      Date.Text = FormatDate(DateTime.Parse(Get<yritys>Data("Date")));
      Bid.Text = ToPercentage(Get<yritys>Data("Last")).ToString();
      ChangePercent.Text = ToPercentage(Get<yritys>Data(
        "ChangePercent")).ToString();
      ChangeImg.ImageUrl = GetChangeImage(ToPercentage(
        Get<yritys>Data("ChangePercent")));
      Volume.Text = Get<yritys>Data("Volume");
    }
  }
  //parsitaan urlia
  private string Decode(string url)
  {
    string s = url.Replace("%3d", "=");
    return s.Replace("%3a", "&");
  }

  // Muuntaa stringin prosentiksi
  private Decimal ToPercentage(string data)
  {
    System.Globalization.NumberFormatInfo nfi = new System.
      Globalization.NumberFormatInfo();
    nfi.NumberDecimalSeparator = ".";
    nfi.NumberNegativePattern = 1;
    Decimal decData = Decimal.Parse(data, Sys
      tem.Globalization.NumberStyles.Float, nfi);
    return System.Math.Round(decData, 2);
  }

  // Palauttaa osaketiedot stringinä
  private string Get<yritys>Data(string element)
  {
    XmlNode node =
      _xmlDoc.ChildNodes[1].ChildNodes[1].SelectSingleNode(element);
    if (node != null)
      return node.InnerText;
    else
      return string.Empty;
  }
}
```

```
}  
  
// Palauttaa nuolen suunnan riippuen osakkeen muutoksesta  
private string GetChangelImage(Decimal change)  
{  
    if (change > 0)  
        return "/Templates/<yritys>/styles/default/images/icon_share_up.gif";  
    else if (change < 0)  
        return "/Templates/<yritys>/styles/default/images/icon_share_down.gif";  
    else  
        return "/Templates/<yritys>/styles/default/images/icon_share_zero.gif";  
}  
  
// Palauttaa halutuksi muokatun päiväyksen  
private string FormatDate(DateTime dt)  
{  
    if (dt != null)  
        return dt.ToString("dd.MM.yyyy HH:mm ");  
    else  
        return string.Empty;  
}  
}  
}
```