



ARASHI *project*

L'alternativa semplice in salsa .NET
per la gestione di contenuti web

Mission

- Arashi nasce con l'idea di avere un framework di web publishing modulare ed espandibile, rispettoso degli standard web, e con la massima aderenza possibile ai principi di usabilità.
- Esistono tanti framework di tal genere in tecnologia sviluppati con Microsoft .NET, ma molti sono deludenti quanto a facilità e immediatezza d'uso, e spesso sono realizzati “**da sviluppatori per sviluppatori**”.

Project Mission

- L'obiettivo ambizioso è quello di avere un prodotto equivalente prima e superiore poi a WordPress, attualmente lo stato dell'arte del web publishing.
- Ma anche un framework che sia una base di partenza per qualsiasi applicazione web.
- Il primo obiettivo è quello di ...realizzare il mio blog personale! 😊

WordPress Clone: why?

- Arashi nasce come un “fork” del framework di una parte del progetto Cuyahoga, differenziandosi poi completamente nella modalità di gestione dei contenuti, mirando ad offrire le stesse funzionalità di WordPress.
- WordPress è realizzato con PHP e MySQL. Arashi utilizza .NET 3.5 ed è multi-database ⁽¹⁾.
- WordPress ha un parco di templates enorme, perchè non riutilizzarli, invece di creare l'ennesimo sistema di templates?

(1) Questo è reso possibile dall'utilizzo dell' O.R.M. NHibernate

Technology

ASP.net
MVC

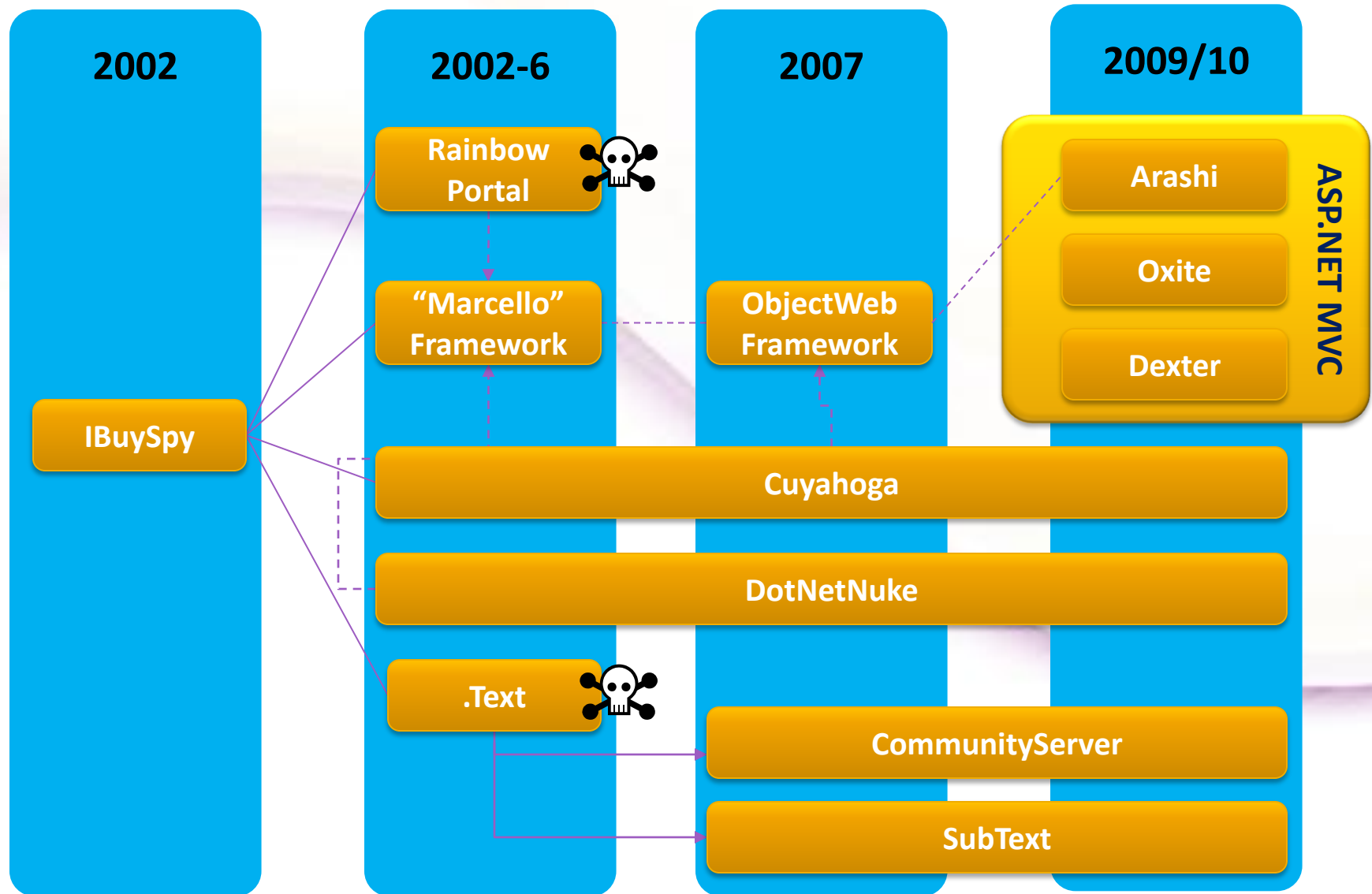
 **jQuery**
write less, do more.

 **NHibernate**

Lucene

 **xVal**

Basic .NET “CMS” history



Features

CMS Features

- Role based membership
- Backend Control Panel with fine grained security
- Content First
- Content Indexing (Lucene.Net)
- Friendly Url (Asp.Net Routing)
- Simple workflow for content
- Multiple site management with multiple hostname
- Localization
- Services modularity with IoC
- Install & Upgrade Wizard
- Media Manager with Silverlight MultiFile Uploader
- Mini tutorial (for the backend)

Blog Features

- SEO friendly
- Comment moderation
- Pingback & Trackback
- Feed syndication with RSS 2.0 & ATOM 1.0
- WordPress Templates compatible
- MetaWeblogAPI (in progress...) for offline editing
- Spam fighting with ReCaptcha

Install Wizard

 **ARASHI** *project*

SETUP

Welcome to the installation procedure!

With few simple steps we'll prepare the system for first use.

Once the setup has finished, you can customize all the settings and add your content.

Important: make sure that in the Web.config file located in the application root the connection string is properly specified.



Start the setup

 **ARASHI** *project*

SETUP

**Step 1**

**Step 2**

**Step 3**

Install Database
The installer will first install the database.

The database for the following components will be installed:
Arashi Core 1.1.0

Install database

 **ARASHI** *project*

SETUP

**Step 1**

**Step 2**

**Step 3**

Create site
Now it's time to create the default site.
In the textbox below, please insert the default host name under which Arashi is configured to run in the web server:

Host name

Create a site

 **ARASHI** *project*

SETUP

**Step 1**

**Step 2**

**Step 3**


Set administrator password
Please enter a valid email address and a password that will be used for the administrator account.

Email


Password

Confirm password

Create the administrator user

 **ARASHI** *project*

SETUP



Setup completed!
Arashi is successfully installed!

Log in to the site administration with the account you just created

Log in

Upgrade Wizard

 **ARASHI** *project*







Welcome to the upgrade procedure!
We have detected that some components of Arashi needs to be upgraded to the latest version.

Component	Current Version	New Version	
Arashi.Web	1.0.0	1.1.0	💡 needs upgrade
Arashi.Core	1.0.0	1.1.0	💡 needs upgrade
Arashi.Core.Domain	0.0.0	1.1.0	💡 needs upgrade
Arashi.Services	0.0.0	1.1.0	💡 needs upgrade
Arashi.Web.Mvc	0.0.0	1.1.0	💡 needs upgrade
Arashi.Core.NHibernate	0.0.0	1.1.0	💡 needs upgrade
Arashi.Web.Plugins	0.0.0	1.1.0	💡 needs upgrade
Arashi.Web.Widgets	0.0.0	1.1.0	💡 needs upgrade

Upgrade!

 **ARASHI** *project*





Upgrade completed!
Arashi has been successfully upgraded to the latest version!

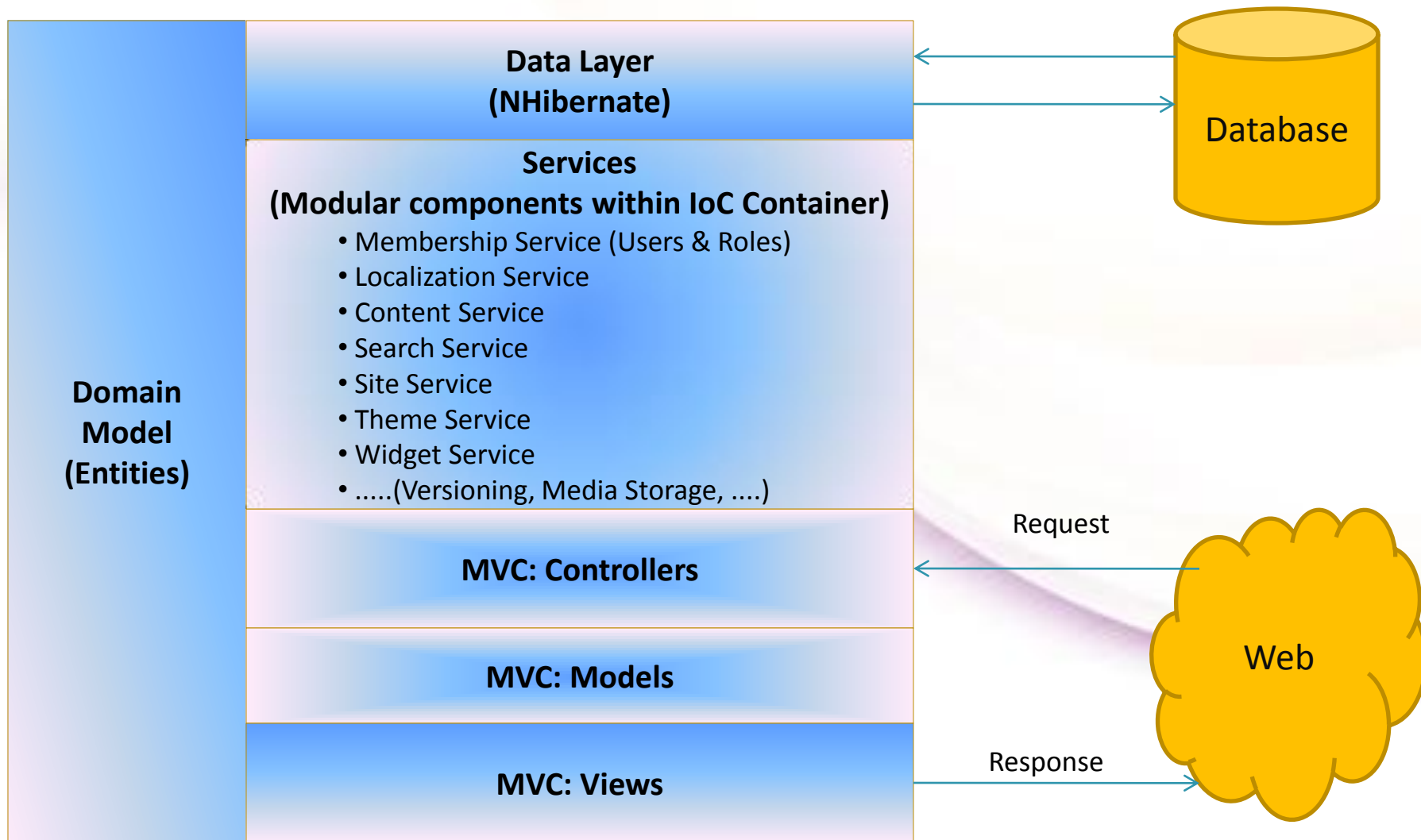
Now you can log in to the site administration with the account you just created

Log in



Demo...

Architecture

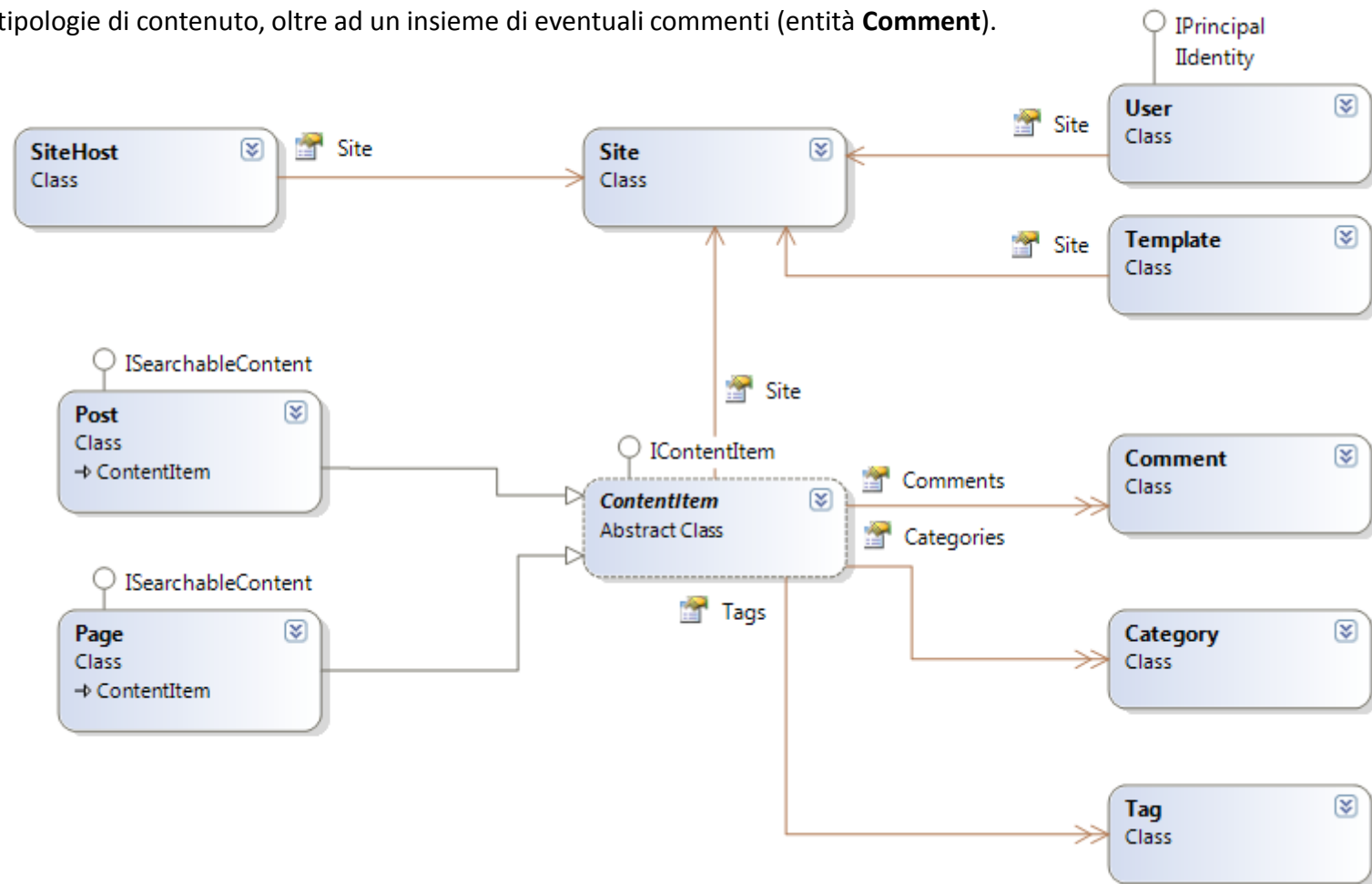


Architecture - Domain

Le due entità chiave sono Site e ContentItem. Ogni sito può avere uno o più **SiteHost** (nomi di dominio), uno o più **User** (utenti registrati), un **Template**, ossia il tema grafico che ne distingue l'aspetto sul web.

Infine, ciascun sito avrà diversi contenuti, rappresentati dall'entità astratta **ContentItem** e al momento implementata dalle due specializzazioni **Post** e **Page**.

Ogni singolo contenuto può avere associati un insieme di **Category** e/o di **Tag**, che permettono di categorizzare e indicizzare le varie tipologie di contenuto, oltre ad un insieme di eventuali commenti (entità **Comment**).



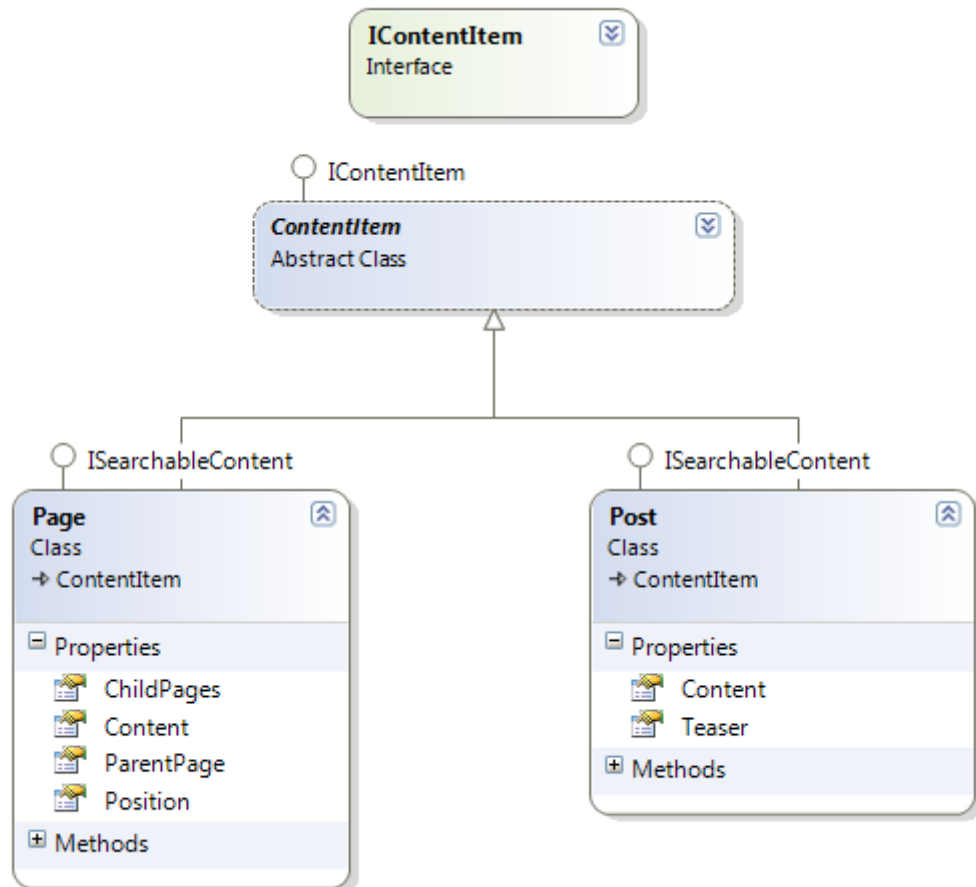
Architecture – Content Abstraction

Al fine di disporre di una infrastruttura comune che consenta di gestire operazione comuni per tutte le tipologie di contenuti, si è resa necessaria la costruzione di un livello di astrazione; in un ambito OOP tale astrazione si ottiene tramite l'ereditarietà, a tal scopo si è creata una classe base per la gestione dei contenuti: *ContentItem*, e il suo contratto è l'interfaccia *IContentItem*. Questa classe fornisce out-of-the-box tutta una serie di proprietà che **tutte** le classi derivate che gestiscono contenuti devono avere.

La classe *ContentItem* viene ereditata da:

- Post**: è un *ContentItem* specializzato il cui contenuto rappresenta un articolo (ad es. di un blog), con tags, categorie e commenti.
- Page**: è un *ContentItem* specializzato il cui contenuto serve per visualizzare del generico HTML statico.

Entrambe queste classi espongono una proprietà *Content* di tipo stringa, destinata a contenere il contenuto in formato html.



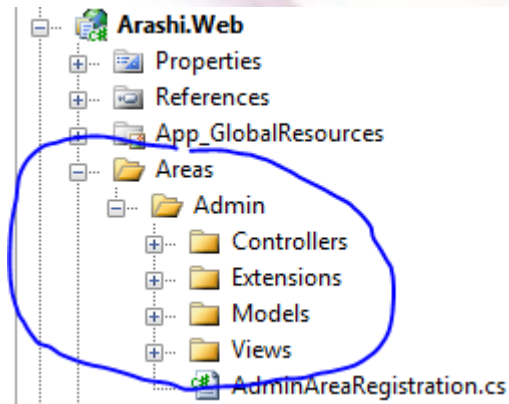
[Note: due to the fact that the Arashi Project is derived from the Cuyahoga project, most of the infos in this page is borrowed from the Cuyahoga documentation. Credits go to Martijn Boland and the cuyahoga dev team]

ASP.NET MVC 2 Features

- Area partitioning
- Strongly typed Models
- Custom HtmlHelper extensions
- Partial Views
- Custom ActionFilters
 - Localization
 - Exception
 - Permission
 - Seo Url Canonicalization
 - Maintenance
- Custom Routing Constraints
- Custom ActionResult (Syndication, Thumbnails, ...)

ASP.NET MVC 2 in Arashi - Backend

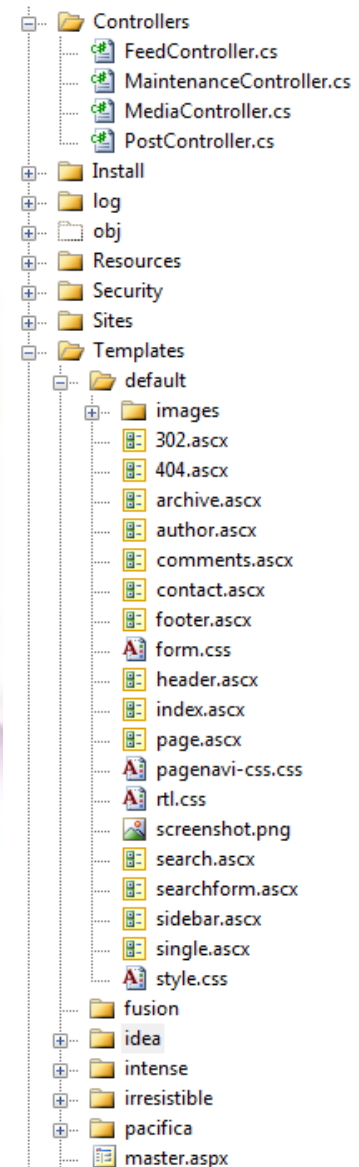
- Il backend è strutturato come una applicazione standard MVC



ASP.NET MVC 2 in Arashi - Frontend

- Il frontend è strutturato in modo da poter riutilizzare i themes di WordPress*
- Ogni theme è formato da più template files che corrispondono a delle Partial Views di ASP.NET MVC
- Dato che Wordpress mette a disposizione un ricco set di funzioni da usare nei Template Files, queste sono state portate in Arashi in una partial class da cui ereditano tutte le partial views di ogni theme.

* La compatibilità non è totale e ci sono ancora dei limiti; i template files originali vanno prima convertiti.



Frontend

- Arashi e WordPress hanno un motore di rendering “template-driven”.
- Entrambe le piattaforme effettuano il render delle richieste di frontend tramite file di template specifici per il tipo di richiesta:

Request	Template File
/	index.ascx
/2009/11/04/titolo-del-post/	single.ascx
/page/this-is-a-static-page/	page.ascx
/2009/10/	archive.ascx
....

Widgets

- Ma ci sono anche i Widgets: blocchi di UI che vengono renderizzati in modo dinamico.
- La configurazione è memorizzata su db.

TO DO (cosa c'è ancora da fare)

- Versioning dei contenuti
- Migliorare il workflow
- Completare il Pannello di Controllo (PdC)
 - Edit dei template
 - File/Media Manager con storage su db
 - Widgets management
- Content Ratings
- Aumentare la compatibilità con WordPress
- Estensibilità tramite plugins...

Thanks...

...anzi,

ありがとうございました

Arigatou!