# Summary

[Summary 1](#_Toc252568119)

[Introduction 2](#_Toc252568120)

# 

# Introduction

# Info di login

Al primo avvio dell'applicazione viene lanciato il wizard di setup, che configurerà il sistema e creerà un web site di default.

Una volta configurato, il sistema sarà accessibile dai seguenti link:

* Pannello di Controllo: <http://admin.localhost:8080/Admin/>
* Sito web di default: <http://localhost:8080/>

# Architettura del progetto

* [Solution Structure](https://www.assembla.com/wiki/show/arashi/Solution_Structure)
  + [Arashi Core](https://www.assembla.com/wiki/show/arashi/Arashi_Core)
  + [Arashi Core Cms](https://www.assembla.com/wiki/show/arashi/Arashi_Core_Cms)
    - [Domain Entities](https://www.assembla.com/wiki/show/arashi/Domain_Entities)
    - [Content Abstraction](https://www.assembla.com/wiki/show/arashi/Content_Abstraction)
  + [Arashi Web](https://www.assembla.com/wiki/show/arashi/Arashi_Web)

Request life-cycle: from global.asax to View render (for ControlPanel & Frontend)

[Riferimenti alla documentazione di WordPress](https://www.assembla.com/wiki/show/arashi/Riferimenti_alla_documentazione_di_WordPress)

# Browser supportati

Questo progetto aderisce alla campagna [**IE6 no more**](http://www.ie6nomore.com/) e di fatto il backend non supporta Internet Explorer 6, mentre il supporto a IE7 è parziale; sui siti di frontend generati tramite Arashi, il supporto ai vari browser è demandato a ogni singolo template.

Si consiglia quindi di utilizzare un browser moderno quali Google Chrome, Mozilla Firefox, o Microsoft Internet Explorer 8.

Per maggiori info si veda [www.ie6nomore.com](http://www.ie6nomore.com/)

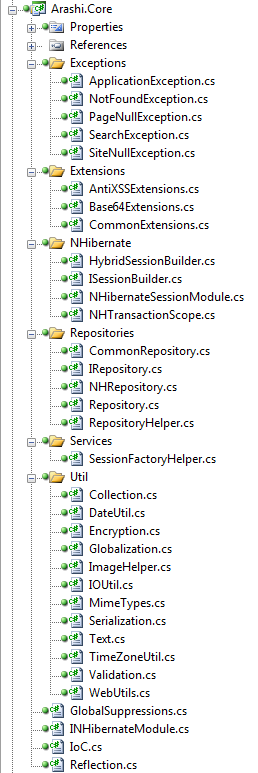
# Solution Structure

Solution Structure:

* Progetto Core: [Arashi Core](https://www.assembla.com/wiki/show/arashi/Arashi_Core)
* Progetto Core.Cms: [Arashi Core Cms](https://www.assembla.com/wiki/show/arashi/Arashi_Core_Cms)
  + [Domain Entities](https://www.assembla.com/wiki/show/arashi/Domain_Entities)
  + [Content Abstraction](https://www.assembla.com/wiki/show/arashi/Content_Abstraction)
* Progetto Web: [Arashi Web](https://www.assembla.com/wiki/show/arashi/Arashi_Web)
  + Pannello di Controllo: ND
  + Frontend & Templates: ND
* Progetto Arashi.Web.Mvc: MVC base classes, Filters, Models, WordPress Compatibility

## Arashi Core

Il progetto Arashi.Core è il progetto base di tutta la solution, e sostanzialmente contiene tutta una serie di classi di utilità generale; di seguito ne descrivo una panoramica.



### Arashi.Core.Exceptions

Contiene delle custom exceptions, usate principalmente nei progetti Arashi.Core.Cms e Arashi.Web per segnalare particolari situazioni nelle funzioni chiamanti.

### Arashi.Core.Extensions

Qui ci sono degli Extension Methods (C# 3) per:

* il supporto all'Anti XSS (Cross Site Scripting): UrlEncode/Decode, HtmlEncode/Decode,...
* Encode/Decode di stringhe da/per il formato Base64
* CommonExtensions contiene invece tutta una serie di estensioni per facilitare il lavoro con Collections varie, criptazioni SHAxx, e soprattutto per le stringhe.

### Arashi.Core.NHibernate

Qui ci sono le classi necessarie per il supporto all'ORM NHibernate, per accedere alla sia SessionFactory e Session tramite il pattern [Session-Per-Request](https://www.hibernate.org/42.html), più [rimpiazzate dall'assembly Castle.Facilities.NHibernateIntegration]

La classe NHTransactionScope che si può considerare equivalente alla System.Transaction.TransactionScope, usata per effettuare operazioni con NHibernate nel contesto di una transazione esplicita.

### Arashi.Core.Repositories

Queste classi costituiscono l'implementazione del pattern Repository; questo pattern è spiegato in dettaglio ai seguenti link:

* [Martin Fowler, Patterns of Enterprise Application Architecture: Repository](http://martinfowler.com/eaaCatalog/repository.html)
* [The NHibernate FAQ: The Repository Pattern](http://blogs.hibernatingrhinos.com/nhibernate/archive/2008/10/08/the-repository-pattern.aspx)

La classe Repository costituisce un wrapper sui metodi originali di NHibernate per le operazioni CRUD.

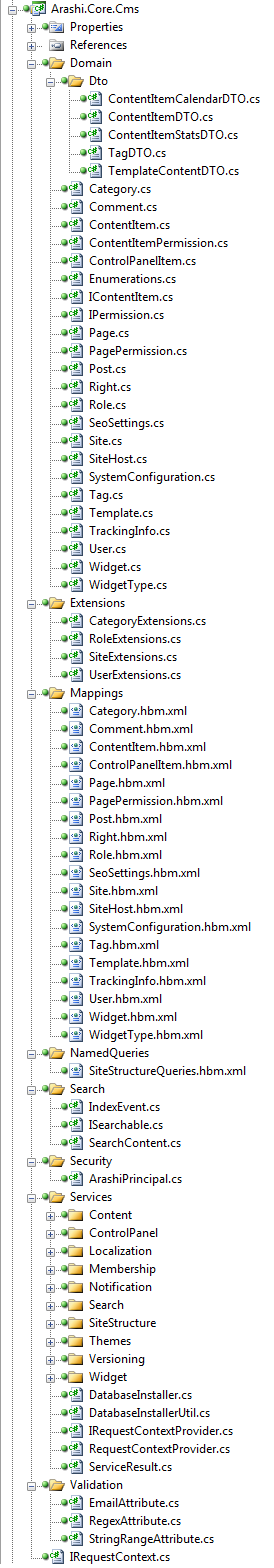
In tutti gli altri progetti della solution, l'implementazione della classe Repository avviene sempre tramite la sua interfaccia IRepository ottenibile tramite il contenitore di IoC (Inversion of Control).

### Arashi.Core.Util

Contiene varie classi di utilità, per lo più non statiche, alcune delle quali sono usate dagli extension methods sopra nominati.

Di particolare rilevanza, nella root del progetto è presente la classe **IoC** che rappresenta un wrapper per l'accesso ai metodi di gestione del contenitore di Inversion of Control.

# Arashi Core Cms

Questo progetto contiene principalmente:

* classi che rappresentano le entità del dominio.
* file \*.xml.hbm usati da NHibernate per il mapping tra le entità e il database.
* Servizi applicativi, ovvero il business layer.

Proprio per la tipologia delle classi che contiene, è molto probabile che durante la roadmap per la milestone 1.5 questo progetto verrà scisso in due:

* Arashi.Domain: per le entità e i mapping.
* Arashi.Business.Services: per i servizi applicativi

Vediamo ora una panoramica del progetto attuale:

|  |  |
| --- | --- |
|  | * \Domain: contiene le classi delle entità del dominio. * \Domain\Dto: contiene i DTO ([Data Transfer Object](http://en.wikipedia.org/wiki/Data_transfer_object)). * \Extensions: contiene Extension Methods per alcune entità. * \Mappings: contiene i file xml di mapping per NHibernate * \NamedQueries: contiene un file xml per NHibernate contenente le query HQL; per maggiori info sui vantaggi di questo approccio vedi questo [link](http://darioquintana.com.ar/blogging/2007/10/16/why-use-named-queries-with-nhibernate/). |

yyyyyyyyyyyy