Exemple de Classe : (Classe d’un bien immobilier)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Adviser.Web.HomeRealEstate.Library;
using Adviser.Library;

namespace Adviser.Web.HomeRealEstate.Library
{
    [MetaDataOnly]
    public class RealEstate : HREBaseObjectImpl
    {
        [Storage]
        [MetaData]
        internal String ProjectId { get; set; }
        private Project project = null;

        /// <summary>
        /// Obtient ou défini le projet associé
        /// </summary>
        public Project Project
        {
            get
            {
                if (this.project == null)
                {
                    if (String.IsNullOrEmpty(this.ProjectId))
                    {
                        return null;
                    }
                    MetaDataQuery query = new MetaDataQuery();
                    query.Add(new MetaDataFilteredValue("Id", this.ProjectId,
MetaDataValueType.String, OperatorType.Equals));
                    this.project = query.First<Project>();
                }
                return this.project;
            }
            set
            {
                if (value == null)
                {
                    this.project = null;
                    this.ProjectId = string.Empty;
                }
                else
                {
                    value.Update();
                    this.project = value;
                    this.ProjectId = value.UniqueId;
                }
            }
        }
```

```
        /// <summary>
        /// Obtient ou défini le type de bien
        /// </summary>
        [Storage]
        [MetaData]
        public RealEstateType Type { get; set; }

        [Storage]
        [MetaData]
        public String CommercialId { get; set; }
        private Account commercial = null;
        /// <summary>
        /// Obtient ou défini le commercial associé;
        /// </summary>
        public Account Commercial
        {
            get
            {
                if (this.commercial == null)
                {
                    if (!String.IsNullOrEmpty(this.CommercialId))
                    {
                        this.commercial =
AccountFactory.GetAccountById(this.CommercialId);
                    }
                }
                return this.commercial;
            }
            set
            {
                if (value == null)
                {
                    this.commercial = null;
                    this.CommercialId = string.Empty;
                }
                else
                {
                    value.Update();
                    this.commercial = value;
                    this.CommercialId = value.UniqueId;
                }
            }
        }

        /// <summary>
        /// Obtient ou défini la première ligne d'adresse
        /// </summary>
        [Storage]
        [MetaData]
        public String Address1 { get; set; }

        /// <summary>
        /// Obtient ou défini la seconde ligne d'adresse
        /// </summary>
        [Storage]
        [MetaData]
        public String Address2 { get; set; }
```

```
internal String CityId { get; set; }
private City city = null;
/// <summary>
/// Obtient ou défini la ville
/// </summary>
[Storage]
[MetaData]
public City City
{
    get
    {
        if (this.city == null)
        {
            if (!String.IsNullOrEmpty(this.CityId))
            {
                this.city = CityFactory.GetCityById(this.CityId);
            }
        }
        return this.city;
    }
    set
    {
        if (value == null)
        {
            this.city = null;
            this.CityId = string.Empty;
        }
        else
        {
            value.Update();
            this.city = value;
            this.CityId = value.UniqueId;
        }
    }
}

/// <summary>
/// Obtient ou défini la Latitude
/// </summary>
[Storage]
[MetaData]
public decimal Latitude { get; set; }

/// <summary>
/// Obtient ou défini la Longitude
/// </summary>
[Storage]
[MetaData]
public decimal Longitude { get; set; }

/// <summary>
/// Obtient ou défini le nombre de pièces
/// </summary>
[Storage]
[MetaData]
public int RoomNumber { get; set; }

/// <summary>
```

```
/// Obtient ou défini la surface totale habitale en m2
/// </summary>
[Storage]
[MetaData]
public int LivingArea { get; set; }

/// <summary>
/// Obtient ou défini la surface du terrain en m2
/// </summary>
[Storage]
[MetaData]
public int LandArea { get; set; }

[Storage]
[MetaData]
public String HeatingSystemIds { get; set; }
private List<Item> heatingSystem = null;
private List<Item> heatingSystemList
{
    get
    {
        if (this.heatingSystem == null)
        {
            this.heatingSystem = new List<Item>();
            if (!String.IsNullOrEmpty(this.HeatingSystemIds))
            {
                String[] lines = this.HeatingSystemIds.Split(';');
                foreach (String line in lines)
                {
                    Item item = ItemFactory.GetItemById(line);
                    if (item == null) continue;
                    this.heatingSystem.Add(item);
                }
            }
        }
        return this.heatingSystem;
    }
}
/// <summary>
/// Obtient ou défini le type de chauffage
/// </summary>
[Storage]
[MetaData]
public IEnumerator<Item> HeatingSystems
{
    get
    {
        return this.heatingSystemList.GetEnumerator();
    }
}

/// <summary>
/// Obtient ou défini la disponibilité
/// </summary>
[Storage]
[MetaData]
public String Availability { get; set; }
```

```
/// <summary>
/// Obtient ou défini le lien de la video du bien
/// </summary>
[Storage]
[MetaData]
public String Video { get; set; }

[Storage]
[MetaData]
public decimal Price { get; set; }

/// <summary>
/// Obtient ou défini la description du bien
/// </summary>
[Storage]
[MetaData]
public String Description { get; set; }

/// <summary>
/// Obtient ou défini les informations complémentaires
/// </summary>
[Storage]
[MetaData]
public String FurtherInformation { get; set; }


//TODO Mettre l'element de mobule Visite qui a été réalisé par le commercial
//En bas de page 11 => R.A.
private List<Room> rooms = null;
private List<Room> roomList
{

    get
    {
        if (this.rooms == null)
        {
            this.rooms = RoomFactory.GetRoomsByRealEstate(this);
        }
        return this.rooms;
    }

}
/// <summary>
/// Obtient la liste des pièces du bien
/// </summary>
public IEnumerator<Room> Rooms
{
    get
    {
        return this.roomList.GetEnumerator();
    }
}

private List<Document> documents = null;
private List<Document> documentList
{
    get
    {
```

```
                if (this.documents == null)
                {
                    MetaDataQuery query = new MetaDataQuery();
                    query.Add(new MetaDataFilteredValue("OwnerId", this.UniqueId,
MetaDataValueType.String, OperatorType.Equals));
                    query.Add(new MetaDataFilteredValue("OwnerType",
((int)OwnerType.RealEstate), MetaDataValueType.Int, OperatorType.Equals));
                    this.documents = query.Find<Document>();
                    if (this.documents == null)
                    {
                        this.documents = new List<Document>();
                    }
                }
                return this.documents;
            }
        }

        /// <summary>
        /// Obtient tous les documents associés au bien.
        /// </summary>
        public IEnumerator<Document> Documents
        {
            get
            {
                return this.documentList.GetEnumerator();
            }
        }

        [Storage]
        [MetaData]
        public String SectorId { get; set; }
        private Sector sector = null;
        /// <summary>
        /// Obtient ou défini le Secteur du bien
        /// </summary>
        public Sector Sector
        {
            get
            {
                if (this.sector == null)
                {
                    if (String.IsNullOrEmpty(this.SectorId))
                    {
                        return null;
                    }
                    MetaDataQuery query = new MetaDataQuery();
                    query.Add(new MetaDataFilteredValue("Id", this.SectorId,
MetaDataValueType.String, OperatorType.Equals));
                    this.sector = query.First<Sector>();
                }
                return this.sector;
            }
            set
            {
                if (value == null)
                {
                    this.SectorId = string.Empty;
                    this.sector = null;
```

```
        }
        else
        {
            value.Update();
            this.sector = value;
            this.SectorId = value.UniqueId;
        }
    }
}

public RealEstate()
{
    this.Address1 = string.Empty;
    this.Address2 = string.Empty;
    this.Availability = string.Empty;
    this.City = null;
    //this.HeatingSystemType = Library.HeatingSystemType;
    this.LandArea = 0;
    this.Latitude = 0;
    this.LivingArea = 0;
    this.Longitude = 0;
    this.Name = string.Empty;
    this.Project = null;
    this.RoomNumber = 0;
    this.TimeCreated = DateTime.Now;
    this.TimeLastModified = DateTime.Now;
    this.Type = RealEstateType.HRERealEstateTypeUnknown;
    this.UniqueId = string.Empty;
    this.Availability = string.Empty;
    this.Video = string.Empty;
}

/// <summary>
/// Ajoute un document à la liste des documents
/// </summary>
/// <param name="document"></param>
/// <returns>Statut de l'opération</returns>
public SystemUpdateStatus AddDocument(Document document)
{
    Security.Check(document);
    this.documentList.Add(document);
    document.Owner = this;
    return document.Update();
}

/// <summary>
/// Supprime un document de la liste des documents
/// </summary>
/// <param name="document"></param>
/// <returns>Statut de l'opération</returns>
public SystemUpdateStatus RemoveDocument(Document document)
{
    Security.Check(document);
    this.documentList.Remove(document);
    document.Owner = null;
    return document.Update();
}
```

```
        public SystemUpdateStatus AddHeatingSystem(Item i)
        {
            if (i == null) return SystemUpdateStatus.InvalidParams;
            Security.Check(i);
            this.heatingSystemList.Add(i);
            this.LoadHeatingSystemString();
            return SystemUpdateStatus.Success;
        }

        public SystemUpdateStatus RemoveHeatingSystem(Item i)
        {
            if (i == null) return SystemUpdateStatus.InvalidParams;
            Security.Check(i);
            this.heatingSystemList.Remove(i);
            this.LoadHeatingSystemString();
            return SystemUpdateStatus.Success;
        }

        private void LoadHeatingSystemString()
        {
            this.HeatingSystemIds = string.Empty;
            foreach (Item item in this.heatingSystemList)
            {
                this.HeatingSystemIds += item.UniqueId + ";";
            }
            this.HeatingSystemIds = this.HeatingSystemIds.Trim(';');
        }

        public String GetMainPicture()
        {
            return string.Empty;
        }









    }
}
```