Domain convention

Methods

Method	Parameters	Linq / Entity Operation	Return Type	State	Dal Sync Mode
Fetch	(id, optional filter params, eagerLoaded = false / eagerLoadParms = null)	Single	T	Static	Default
TryFetch	(id, optional filter params, eagerLoaded = false / eagerLoadParms = null)	SingleOrDefault	T?	Static	Default
FetchAll	(optional filter params)	Where	IEnumerable<T> , IPagedList<T>	Static	Default
FetchAllBy<Association>	(<Association>Id , optional filter params, eagerLoaded = false / eagerLoadParms = null)	Where	IEnumerable<T> , IPagedList<T>	Static	Default
TryFetchFirst	None	FirstOrDefault	T?	Statuc	Default
Create	None	AddObject(this)	Void	Non-Static	Sync
Update	None	Attach(this) / Modified Entity State	Void	Non-Static	Sync
Delete	None	Attach(this) / Deleted Entity State	Void	Non-Static	Sync

Queryable<T>{All}{By<Association>	SignifyHRDAL	IQueryable	IQueryable<T>	Static	Default
Is<Condition>, Can<Condition>, Has<Condition>, etc.	None / Property not method	Boolean Check	Bool	Non-Static	Default
<Enum>Type	None / Property not method	<T>TryParseEnum	<T>	Non-Static	Default
CreateEdit	Call Create / Update separately from method. Is Wrapper Method	Call Methods Separately	Void	Non-Static	Sync

Meta Data

Meta data is used when rendering MVC Razor views and are defined in the domain if display name differs from specified name in base table, values are required and/or format of value.

Table 2 : Meta Data Properties

Property	Reason	Example
Required	Mark value as required and performs validation when submitting from form on view. An error message can be defined to display if validation fails.	[Required(ErrorMessage = "The {0} field is required")]
Display(Name=<string>)	Text to display when using DisplayFor() on views.	[Display(Name = "Start Date")]
DataType(<DataType>)	Formats and validate input according to specified type.	[DataType(DataType.DateTime)]

Examples

Template

Below is an example of a basic domain. This can be used as a template when creating new domains; copy and paste the code, replace all the required words (exSample, sample, smpl) according to applicable entity name.

Note: For domains with a single eager loaded objects, the following can be used:

```
//- Replace "EagerLoadParameters eagerLoadParms = null"
bool useEagerLoading = false
```

```
using SignifyHR.Core;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;

namespace SignifyHR.Data.Domain
{
   public partial class exSample : IAuditable
   {
      #region Search Parameters

      public class SearchParameters : BaseSearchParameters
      {
         public int? SomeId { get; set; }
         public string Description { get; set; }
         public bool IsUsed { get; set; }
      }

      #endregion

      #region Eager Load Parameters

      public class EagerLoadParameters : BaseSearchParameters
      {
         public bool IncludeSampleDocuments { get; set; }
         public bool IncludeSampleComments { get; set; }
      }

      #endregion
```

```
        #region Protected Methods

        protected static IQueryable<exSample> ValidSamples(SignifyHRDAL dbContext, EagerLoadParameters
eagerLoadParms = null)
        {
            var samples = dbContext.exSamples.AsQueryable();

            if (eagerLoadParms != null)
            {
                if (eagerLoadParms.IncludeSampleDocuments )
                    samples = samples.Include(item => item.exSampleDocuments);

                if (eagerLoadParms.IncludeSampleComments )
                    samples = samples.Include(item => item .exSampleComments);
            }

            return samples;
        }

        protected static IQueryable<exSample> FilterSamples(SignifyHRDAL dbContext, SearchParameters
searchParms, EagerLoadParameters eagerLoadParms = null)
        {
            var result = ValidSamples(dbContext, eagerLoadParms);

            if (searchParms!= null)
            {
                if (!String.IsNullOrWhiteSpace(searchParms.Description))
                    result = result.Where(item =>
item.Description.ToLower().Contains(searchParms.Description.ToLower()));

                if (searchParms.SomeId.HasValue)
                    result = result.Where(item => item.SomeId == searchParms.SomeId.Value);

                if (searchParms.IsUsed.HasValue)
                    result = result.Where(item => item.IsUsed == searchParms.IsUsed.Value);
            }

            return result;
        }
```

```
        #endregion

        #region Public Methods

        public static exSample Fetch(ISessionHandler sessionHandler, int id, EagerLoadParameters
eagerLoadParms = null)
        {
            using (var dbContext = new SignifyHRDAL(sessionHandler))
            {
                return ValidSamples(dbContext, eagerLoadParms).Single(item => item.Id == id);
            }
        }

        public static exSample TryFetch(ISessionHandler sessionHandler, int id, EagerLoadParameters
eagerLoadParms = null)
        {
            using (var dbContext = new SignifyHRDAL(sessionHandler))
            {
                return ValidSamples(dbContext, eagerLoadParms).SingleOrDefault(item => item.Id == id);
            }
        }

        public static IEnumerable<exSample> FetchAll(ISessionHandler sessionHandler, SearchParameters
searchParms, EagerLoadParameters eagerLoadParms = null)
        {
            using (var dbContext = new SignifyHRDAL(sessionHandler))
            {
                var results = FilterSamples(dbContext, searchParms, eagerLoadParms);

                return results.OrderByDescending(item => item.Id)
                        .Skip(searchParms.Skip)
                        .Take(searchParms.Take)
                        .ToList();
            }
        }

        #endregion
```

```
        #region Public CRUD Actions

        public void Create(ISessionHandler sessionHandler)
        {
            using (var dbContext = new SignifyHRDAL(sessionHandler))
            {
                dbContext.exSamples.Add(this);
                dbContext.SaveChanges();
            }
        }

        public void Update(ISessionHandler sessionHandler)
        {
            using (var dbContext = new SignifyHRDAL(sessionHandler))
            {
                dbContext.exSamples.Attach(this);
                dbContext.Entry(this).State = EntityState.Modified;
                dbContext.SaveChanges();
            }
        }

        public void Delete(ISessionHandler sessionHandler)
        {
            using (var dbContext = new SignifyHRDAL(sessionHandler))
            {
                dbContext.exSamples.Attach(this);
                dbContext.exSamples.Remove(this);
                dbContext.SaveChanges();
            }
        }

        #endregion
    }
}
```

Methods

FetchAll - PagedList.IPagedList

OrderBy or *OrderByDescending* **must** be applied prior to *ToPagedList*.

Use *PagedList.IPagedList* for Bootstrap 3 Pager

```
using PagedList;


public static IPagedList<exSample> FetchAll(ISessionHandler sessionHandler, SearchParameters searchParms,
EagerLoadParameters eagerLoadParms = null)
{
  using (var dbContext = new SignifyHRDAL(sessionHandler))
  {
     var results = FilterSamples(dbContext, searchParms, eagerLoadParms);


      return results.OrderByDescending(item => item.Id)
            .ToPagedList(searchParms.CurrentPage.Value, searchParms.PageSize.Value);
  }
}
```

FetchAll - X.PagedList.IPagedList

OrderBy or *OrderByDescending* **must** be applied prior to *ToPagedList*.

Use *X.PagedList.IPagedList* for Bootstrap 3 Pager when returning POCO class data.

```
using X.PagedList;



       public static IPagedList<exSample> FetchAllPreview(ISessionHandler sessionHandler, SearchParameters
searchParms, EagerLoadParameters eagerLoadParms = null)
       {
          using (var dbContext = new SignifyHRDAL(sessionHandler))
          {
             var samples = FilterSamples(dbContext, searchParms, eagerLoadParms);


             var results = samples.Select(item => new SamplePreview
             {
                XSPreviewId = item.Id,
                XSDescription = item.Description,
                XSPath = Path.GetFileNameWithoutExtension(item.Title)
             })
```

```
            return results.OrderByDescending(item => item.Id)
                    .ToPagedList(searchParms.CurrentPage.Value, searchParms.PageSize.Value, results.Count());
        }
    }
```

Create & Update (Non-*IAuditable* Table)

Use the following where table definitions does not contain all columns as required by *IAuditable*.

```
public void Create(exSample sample)
{
    using (var dbContext = new SignifyHRDAL(true))
    {
        sample.SomeValue = 123;

        dbContext.exSample.AddObject(sample);
        dbContext.SaveChanges();
    }
}

public void Update(exSample sample)
{
    using (var dbContext = new SignifyHRDAL(true))
    {
        sample.SomeValue = 123;

        dbContext.exSample.Attach(sample);
        dbContext.ObjectStateManager.ChangeObjectState(this, EntityState.Modified);
        dbContext.SaveChanges();
    }
}
```

Delete Multiple

Use the following for multiple items.

```
public void DeleteMany(IEnumerable<exSample> sampleList)
{
    using (var dbContext = new SignifyHRDAL(true))
```

```
    {
        foreach (var sample in sampleList)
        {
            sample.Delete();
        }
    }
}
```

Meta Data

Remember to specify *MetadataType* for partial class created.

```
[MetadataType(typeof(exSampleMeta))]
public partial class exSample : IAuditable { ... }

public class exSampleMeta : DefaultColumnsMeta
{
    [Required(ErrorMessage = "The {0} field is required")]
    [Display(Name = "Sample Name")]
    public string Description { get; set; }

    [Required(ErrorMessage = "The {0} field is required")]
    [Display(Name = "Start Date")]
    [DataType(DataType.DateTime)]
    public DateTime StartDate { get; set; }

    [Required(ErrorMessage = "The {0} field is required")]
    [Display(Name = "End Date")]
    [DataType(DataType.DateTime)]
    public DateTime EndDate { get; set; }

    [Display(Name = "Enabled")]
    public bool Enabled { get; set; }
}
```

TODO : AddRange, UpdateRange, FetchPaged

Revision #10
Created 17 September 2020 01:57:49 by Theuns Pretorius
Updated 1 October 2020 05:40:45 by Theuns Pretorius