Entity Framework

Table of contents

General

- All methods accept and use ISessionHandler

```
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
   {
      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;
      }

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

```
            var results = ValidSamples(dbContext, searchParms, eagerLoadParms);

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

- Eager loading used responsibly

```
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 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;
      }
   }
}
```

- IQueryable declared as **protected**

```
protected static IQueryable<exSample> ValidSamples(SignifyHRDAL dbContext, EagerLoadParameters eagerLoadParms = null)
protected static IQueryable<exSample> FilterSamples(SignifyHRDAL dbContext, SearchParameters searchParms, EagerLoadParameters eagerLoadParms = null)

public static exSample Fetch(ISessionHandler sessionHandler, int id, EagerLoadParameters eagerLoadParms = null)
public static exSample TryFetch(ISessionHandler sessionHandler, int id, EagerLoadParameters eagerLoadParms = null)
```

- Create a POCO Object when you want to call a **Stored Procedure** or **View** from Entity Framework

```
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
    {
□□public class POCOPreview
        {
            public int ExampleId { get; set; }
            public string ExampleDescription { get; set; }
            public bool ExampleBool { get; set; }
        }
    }
}
```

- Search parameter array used

```
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 Protected Methods

        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 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
    }
```

```
}
```

- No DateTime values are passed to the database (different servers = different time = different results).
- Domain Convention were followed.
- First, FirstOrDefault, Single, SingleOrDefault used for the correct purpose:
  - **First** - when one or more entities may be returned but only the first one is used (Remember to use OrderBy to return the correct entity)
  - **FirstOrDefault** - when none, one or more entities are returned, but only the first one is used (Remember to use OrderBy to return the correct entity).
  - **Single** - when only one entity will ALWAYS be returned
  - **SingleOrDefault** - when one or no entities are expected

TO DO : Add database date fetch method

Use of sp's vs LINQ and limitations

Do not use Views and SP's directly in entity framework, use POCO classes to map objects

Naming conventions

Methods for EF are written in Pascal Case

Examples of naming conventions that need to be followed for EF Methods:

Method
Fetch
TryFetch
FetchAll
FetchAllBy<Association>
TryFetchFirst
Create
Update
Delete

TODO: Examples, when is a postfix added to the above method names or not?

Which method is used for paging?

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

LINQ

IQueryable

This is used when you want to add and/or filter results from the database, without the query being executed already. This is generally used with methods like **ValidSamples()** and **FilterSamples()**

The IQueryable interface inherits the IEnumerable interface so that if it represents a query, the results of that query can be enumerated.

IEnumerable

Use when you deal with in process memory object collections and loop through the collection objects. IEnumerable uses Func objects that result in the query being **executed immediately and completely**, your application will see a performance boost.

Example:

```
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 IEnumerable<exSample> ValidSamples(SignifyHRDAL dbContext, EagerLoadParameters
eagerLoadParms = null, SearchParameters searchParms)
        {
            //Result not executed yet
            var samples = dbContext.exSamples.AsQueryable();

            //Result not executed yet
            if (eagerLoadParms != null)
            {
                if (eagerLoadParms.IncludeSampleDocuments )
                    samples = samples.Include(item => item.exSampleDocuments);

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

            //Result not executed yet
            if (searchParms!= null)
            {
                if (!String.IsNullOrWhiteSpace(searchParms.Description))
                    samples = samples.Where(item =>
item.Description.ToLower().Contains(searchParms.Description.ToLower()));

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

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

        //Result executed immediately and completely
        return samples.AsEnumerable();
    }
  }
}
```

TODO: Add example for IQueryable

As many calculations, filtering should occur in IQueryable

Naming convention for IQueryable e.g. Valid{ObjectNames}

EF Core

https://docs.microsoft.com/en-us/ef/