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

---

Revision #15
Created 17 September 2020 01:56:59 by Theuns Pretorius
Updated 1 October 2020 05:33:42 by Theuns Pretorius