Entity Framework Optimizations

Entity framework does not have great performance when dealing with large batches of data. We've compiled this list as a reference so that our batch operations can be efficient as possible.

Please Read these Articles

https://weblog.west-wind.com/posts/2014/dec/21/gotcha-entity-framework-gets-slow-in-long-iteration-loops

https://weblog.west-wind.com/posts/2013/Dec/22/Entity-Framework-and-slow-bulk-INSERTs

Disable Auto Detecting Changes

see https://docs.microsoft.com/en-us/ef/ef6/saving/change-tracking/auto-detect-changes

```
using (var context = new BloggingContext())
{
   try
   {
      context.Configuration.AutoDetectChangesEnabled = false;

      // Make many calls in a loop
      foreach (var blog in aLotOfBlogs)
      {
         context.Blogs.Add(blog);
      }
   }
   finally
   {
      // make sure you re enable AutoDetectChangesEnabled before calling SaveChanges
      context.Configuration.AutoDetectChangesEnabled = true;
      await context.SaveChangesAsync();
```

```
    }
}
```

Bloated DbContext

With Entity Framework, you have a dbContext object that tracks all the entities you've read from the database and changes that have been made to them. If it starts to track too many entities you can start getting pretty slow performance.

There a several thanks that can be done to address this issue.

.AsNoTracking()

`List<DbNodes> childNodes = UtopiaDB.DbNodes.Where(i => i.ParentID = 1).AsNoTracking().ToListAsync();`

When you use .AsNoTracking() the entities that get returned will not be tracked by the dbContext. You should use this when you are querying data from the db that is read only.

Re-Create DbContext Object

The easiest way to get past a bloated dbContext is to create a new dbContext object.

This is difficult in our Utopia solution because create one dbContext object per session, so unless a new session is created, a new dbContext object cannot be created.

To address this we have added code to to detach all entities from the dbContext

`UtopiaDB.DetachEntitiesFromUtopiaDB();`

The intent behind this code was to have the same effect as creating a new DbContext object.

Smaller Batches

Typically we've seen better performance when we process things in smaller batches (500 items or less), call 'SaveChangesAsync()', then detach all entities from the DbContext object, and process another small batch.

Revision #2
Created 18 July 2022 23:51:46 by Bryce Holloway
Updated 17 April 2023 17:29:19 by Bryce Holloway