16 分页检索业务对象

在“11.业务对象生命周期及其状态”的“Fetch 业务对象”章节中提到了当需要 Fetch 海量业务数据时，应该实现分页 Fetch 的机制。

分页 Fetch 经常应用在浏览器界面的开发上，这主要由浏览器应用架构的性能和其展现方式决定的。在桌面系统的应用上，分页 Fetch 主要应用在万条级别以上的海量业务数据（此时性能往往会成为一个瓶颈），以提高界面交互的友好性、优化 Fetch 性能。

分页 Fetch 涉及到如下几个概念：

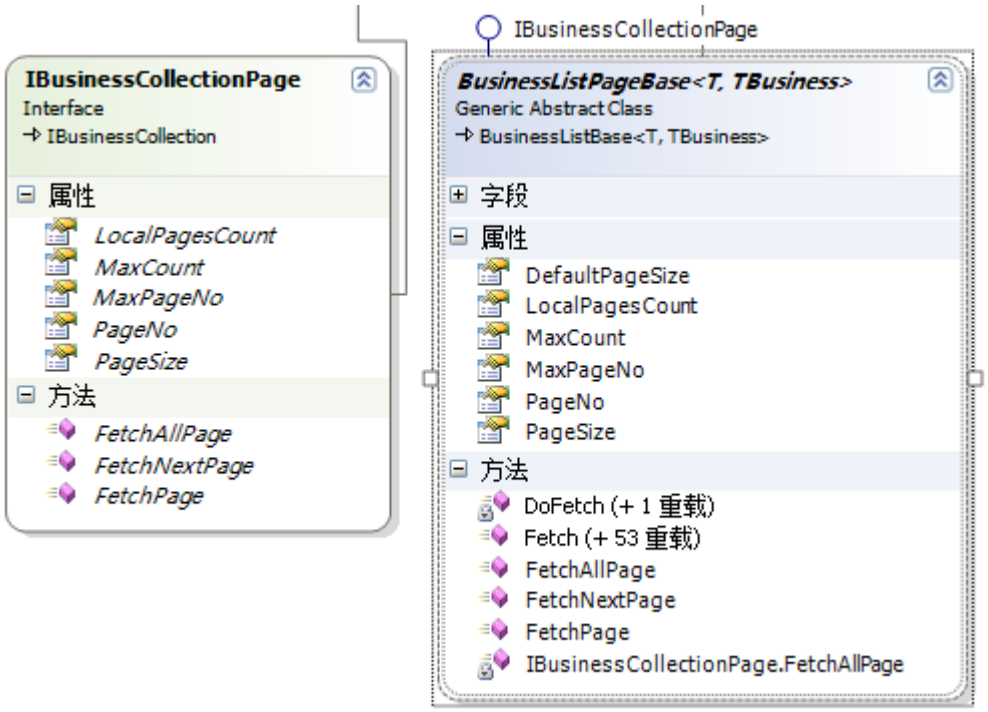
概念	说明	备注
MaxCount	最大记录数	指全部 Fetch 后可能的记录数（完整 Fetch 岂止期间，如果发生相关记录的增删，则会与实际 Fetch 到的记录数不一致）；
PageSize	页大小	每次分页 Fetch 的记录数；在第一次分页 Fetch 时需明确并在完整 Fetch 岂止期间不再更改；如不指明，则默认为 1000；
MaxPageNo	最大分页号	算法是 MaxCount % PageSize == 0 ? MaxCount/PageSize : MaxCount/PageSize + 1；
PageNo	分页号	业务对象上的分页号是指其所在的分页，而业务对象集合上的分页号是指最近一次分页 Fetch 时的页号，范围在 1…MaxPageNo 之间，如果此时做 Add 操作的话，添加进来的业务对象也将拥有相同的分页号；
LocalPagesCount	本地分页数量	当前 Fetch 到本地的分页数量，范围在 0…MaxPageNo 之间；
Order By	排序字段	只有排序的记录才能较为正确地被分页 Fetch(完整 Fetch 岂止期间，如果发生相关记录的增删，则无法保证会不会在结果集合中发生重复和遗漏）；在第一次分页 Fetch 时需明确并在完整 Fetch 岂止期间不再更改；如不指明，则默认以主键或唯一键作为排序字段；

综上所述，分页 Fetch 的应用场景往往是浏览海量业务数据，并且不会过于计较在分页 Fetch 的一个完整过程中数据库的记录会不会发生变化。所以，Phenix丶采取了从数据库中分批摘取排序序列区间记录的方法，以达到高性能的界面交互目的，故意忽视了展现业务数据的完整性，这是我们在应用本功能时需注意的一点。

Phenix丶为分页 Fetch 提供了专门的业务集合分页基类 Phenix.Business.BusinessListPageBase<T, TBusiness>和业务对象分页基类 Phenix.Business.BusinessPageBase<T>，以方便和规范应用系统中分页 Fetch 功能的实现。

16.1 Phenix.Business.BusinessListPageBase<T, TBusiness>

业务集合分页基类实现了 Phenix.Business.IBusinessCollectionPage 接口：



实现分页 Fetch 功能所需的参数由 IbusinessCollectionPage 的属性表达，含义可参考上文中的概念表述，下文将逐一介绍分页 Fetch 的方法。

16.1.1启动分页 Fetch

第一次分页 Fetch 时需指明 PageSize，检索得到的是第一页的业务集合对象，接下来就可以通过它继续 Fetch 其他页。

16.1.1.1 一般方法

```
/// <summary>
/// 构建业务对象集合
/// 条件类的字段映射关系请用Phenix.Core.Mapping.CriteriaFieldAttribute标注
/// </summary>
/// <param name="criteria">条件对象</param>
/// <param name="pageSize">分页大小</param>
public static T Fetch(ICriteria criteria, int pageSize)

/// <summary>
/// 构建业务对象集合
/// </summary>
/// <param name="criteriaExpression">条件表达式</param>
/// <param name="pageSize">分页大小</param>
```

```
    public static T Fetch(Expression<Func<TBusiness, bool>> criteriaExpression, int pageSize)

    /// <summary>
    /// 构建业务对象集合
    /// </summary>
    /// <param name="criteriaExpression">条件表达式</param>
    /// <param name="pageSize">分页大小</param>
    public static T Fetch(CriteriaExpression criteriaExpression, int pageSize)

    /// <summary>
    /// 构建业务对象集合
    /// </summary>
    /// <param name="criterions">条件集</param>
    /// <param name="pageSize">分页大小</param>
    public static T Fetch(Criterions criterions, int pageSize)
```

16.1.1.2 嵌入到现有事务中获取业务对象的方法，不可跨物理域

```
    /// <summary>
    /// 构建业务对象集合
    /// 条件类的字段映射关系请用Phenix.Core.Mapping.CriteriaFieldAttribute标注
    /// </summary>
    /// <param name="transaction">数据库事务</param>
    /// <param name="criteria">条件对象</param>
    /// <param name="pageSize">分页大小</param>
    public static T Fetch(DbTransaction transaction, ICriteria criteria, int pageSize)

    /// <summary>
    /// 构建业务对象集合
    /// </summary>
    /// <param name="transaction">数据库事务</param>
    /// <param name="criteriaExpression">条件表达式</param>
    /// <param name="pageSize">分页大小</param>
    public static T Fetch(DbTransaction transaction, Expression<Func<TBusiness, bool>>
criteriaExpression, int pageSize)

    /// <summary>
    /// 构建业务对象集合
    /// </summary>
    /// <param name="transaction">数据库事务</param>
    /// <param name="criteriaExpression">条件表达式</param>
    /// <param name="pageSize">分页大小</param>
    public static T Fetch(DbTransaction transaction, CriteriaExpression criteriaExpression, int pageSize)

    /// <summary>
    /// 构建业务对象集合
```

```
/// </summary>
/// <param name="transaction">数据库事务</param>
/// <param name="criterions">条件集</param>
/// <param name="pageSize">分页大小</param>
public static T Fetch(DbTransaction transaction, Criterions criterions, int pageSize)
```

16.1.2按页检索业务数据

16.1.2.1 获取下一页

```
/// <summary>
/// 获取下一页
/// </summary>
public IList<IBusinessObject> FetchNextPage()
```

需注意的是：

- 其返回的 IList 对象，是本页中 Fetch 到的业务对象清单内容，它的 Item 对象和被整合到全集（this）里的 Item 对象是同一个对象。
- 已检索过的页，不会再重复从数据库中获取，而是从本地缓存中获取。

16.1.2.2 获取指定页

```
/// <summary>
/// 获取页
/// </summary>
/// <param name="pageNo">分页号</param>
public IList<IBusinessObject> FetchPage(int pageNo)
```

需注意的两点，同上。

16.1.3检索完整业务数据

调用本函数返回的业务集合对象就是本全集自己（this）。采取的方法是逐一 Fetch 每页的业务对象，只要不在本地缓存中的就从数据库中获取，并填充到本全集（this）中。也就是说，当执行完本函数时，其返回的本全集对象的内容是完整无缺的。

```
/// <summary>
/// 获取全部页
/// </summary>
public T FetchAllPage()
```

16.1.4缺省分页大小

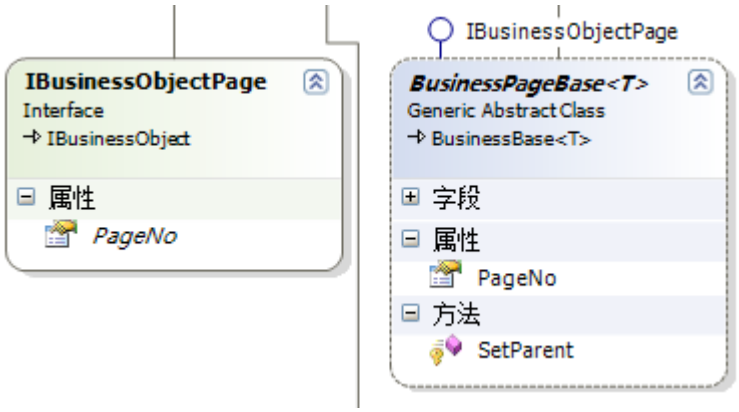
Phenix.Business.BusinessListPageBase<T, TBusiness>提供了可本地配置的“缺省分页大小”属性：

```
/// <summary>
/// 缺省分页大小
/// 缺省为 1000
/// </summary>
public static int DefaultPageSize
{
  get { return AppSettings.GetProperty(ref _defaultPageSize, 1000); }
  set { AppSettings.SetProperty(ref _defaultPageSize, value); }
}
```

第一次分页 Fetch 时，如未指明 PageSize 则以本属性作为默认的参数。

16.2 Phenix.Business.BusinessPageBase<T>

业务对象分页基类实现了 Phenix.Business.IBusinessObjectPage 接口：



它有一个 PageNo 属性，类似于 Selected 属性，是操作集合 Item 的一个很好的抓手。比如我们要是在界面上显示某一页 Item 清单的话，只要编写类似于以下的代码就可以实现：

```
this.assemblyInfoPageListBindingSource.Filter = String.Format("PageNo = {0}",
showPageNoTextBox.Text);
```

16.3 案例

16.3.1设计分页类

```
  /// <summary>
  /// 程序集分页清单
  /// </summary>
  [Serializable]
  public class AssemblyInfoPageList : Phenix.Business.BusinessListPageBase<AssemblyInfoPageList,
AssemblyInfoPage>
  {

  }
```

16.3.2操作方法

```
      //检索按10条记录分页的第一页数据
      AssemblyInfoPageList assemblyInfoPageList = AssemblyInfoPageList.Fetch(10);
      //检索最大记录数
      long i = assemblyInfoPageList.MaxCount;
      //检索最大分页号
      long j = assemblyInfoPageList.MaxPageNo;
      //检索下一页
      assemblyInfoPageList.FetchNextPage();
      //检索全部页
      assemblyInfoPageList.FetchAllPage();
```