Contents

A **paged query** is a query which returns a subset, or page, of query results at a time. At its simplest level, paging is performed using the LINQ *Take* and *Skip* operators.

For example, if we think of a page size as being 50 entities, then we can fetch any page *n* of *Customers*, using the following kind of query logic:

```
var pageNumber = 1;
var pageSize = 50;
var pagedQuery = _em1.Customers.Skip((pageNumber-1)*pageSize).Take(pageSize);
var customersOnFirstPage = pagedQuery.ToList();
pageNumber++
var customersOnNextPage = pagedQuery.ToList();
pageNumber = 17;
var customersOnPage17 = pagedQuery.ToList();
```

```
Dim pageNumber = 1
Dim pageSize = 50
Dim pagedQuery = _em1.Customers.Skip((pageNumber - 1) * pageSize).Take(pageSize)
Dim customersOnFirstPage = pagedQuery.ToList()
pageNumber = pageNumber + 1
Dim customersOnNextPage = pagedQuery.ToList()
pageNumber = 17
Dim customersOnPage17 = pagedQuery.ToList()
```

EntityQueryPager

The *EntityQueryPager<T>* is an integrated paging engine that can fetch pages from a combined view of the server-side datasource plus the client-side cache. It can operate either synchronously or asynchronously, and can be used in both WinClient and Silverlight applications.

To use the *EntityQueryPager* you provide it with a base EntityQuery, sort information, and page size. You then call its various "MoveTo*" or "MoveTo*Async" paging methods to move forwards and backwards within the overall result set. The *EntityQueryPager* handles the difficult problems which arise when paging through an editable collection, as it can determine the appropriate query strategy based on what has been cached, and also performs page marking.

See the code samples for a sample application.