Single-page LINQ cheatsheet (C#)

May. 03, 2018 • 1 like • 4,295 views

Software

...heat-sheet

...tion .ToArray(), .ToList() and .ToLookup(), but I think they're not that hard

Download Now
Download to read offline

More Related Content

Slideshows for you (20)

ASP.NET MVC.
Ni • 955 views

Angular 8
Sunil OS • 527k views

Java Collections Framework
Sony India Softwa... • 4.3k views

MSDN - ASP.NET MVC
Maarten Balliauw • 6k views

ASP.NET Web API and HTTP
Ido Flatow • 26.3k views

Recently uploaded (20)

Sun-Product-line-Update-...
PencilData • 0 views

Fortinet_ProductGuide_NO...
AlonzoJames2 • 0 views

User Interface and Data...
PencilData • 0 views

Sun硬件产品介绍
PencilData • 0 views

Lesson_08_Continuous_Mo...
Minh Quân Đoàn • 0 views

Single-page LINQ cheatsheet (C#)

1. LINQ in C# Filtering var col = from o in Orders where o.CustomerID == 84 select o; var col2 = Orders .Where (o => o.CustomerID == 84) Return Anonymous Type var col = from o in orders select new { OrderID = o.OrderID, Cost = o.Cost }; var col2 = orders .Select (o => new { OrderID = o.OrderID, Cost = o.Cost }); Ordering var col = from o in orders orderby o.Cost ascending select o; var col2 = orders .OrderBy (o => o.Cost); var col3 = from o in orders orderby o.Cost descending select o; var col4 = orders .OrderByDescending (o => o.Cost); var col9 = from o in orders orderby o.customerID, o.Cost descending select o; var col = orders .OrderBy (o => o.CustomerID) .ThenByDescending (o => o.Cost); var col5 = from o in orders orderby o.Cost descending orderby o.CustomerID select o; Joining var col = from c in customers join o in orders on c.CustomerID equals o.CustomerID select new { c.CustomerID, c.Name, o.OrderID, o.Cost }; var col2 = customers .Join (orders, c => c.CustomerID, o => o.CustomerID, (c,o) => new { c.CustomerID, c.Name, o.OrderID, o.Cost }); Grouping var OrdCnt = from o in orders group o by o.CustomerID into g select new { CustomerID = g.Key, TotalOrders = g.Count() }; var OrdCnt1 = orders .GroupBy (o => o.CustomerID) .Select (g => new { CustomerID = g.Key, TotalOrders = g.Count() }); Paging (using Skip and Take) var col = (from o in orders where o.CustomerID == 84 select o) .Take(3); // selects top 3 var col2 = orders .Where (o => o.CustomerID == 84) .Take(3); var col3 = (from o in orders where o.CustomerID == 84 orderby o.Cost select o) .Skip(2) .Take(2); var col4 = orders .Where (o => o.CustomerID == 84) .OrderBy (o => o.Cost) .Skip(2) .Take(2); Element Operators (Single, Last, First, ElementAt, Defaults) // Throws exception if no elements var cust = (from c in customers where c.CustomerID == 84 select c) .Single(); var cust1 = customers .Single (c => c.CustomerID == 84); // Returns null if no elements var cust = (from c in customers where c.CustomerID == 84 select c) .SingleOrDefault(); var cust1 = customers .SingleOrDefault (c => c.CustomerID == 84); // Returns new customer instance if no elms var cust = (from c in customers where c.CustomerID == 85) .DefaultIfEmpty(new Customer()) .Single(); // First/Last/ElementAt used in same way var cust4 = (from o in orders where o.CustomerID == 84 orderby o.Cost select o) .Last(); var cust5 = orders .Where (o => o.CustomerID == 84) .OrderBy (o => o.Cost) .Last(); // Returns 0 if no elements var i = (from c in customers where c.CustomerID == 85 select c.CustomerID) .SingleOrDefault(); var j = customers .Where (c => c.CustomerID == 85) .Select (o => o.CustomerID) .SingleOrDefault(); Grouping key is the same type as grouping value. In this example, grouping key is an int because o.CustomerID is an int Single Last First ElementAt All of the above throw exceptions if source sequence is empty. SingleOrDefau lt LastOrDefault FirstOrDefaul t ElementAtOrDe fault The above ones return default(T) if source sequence is empty., i.e. null will be returned if T s a reference type or nullable value type; default(T) will be returned if T is a non-nullable value type (int, bool, etc.). This can be seen in the last example