# group (seskupování) (ItNetwork)

V dotazech často využíváme seskupování (grouping). Můžeme tak jednoduše prvky seskupit podle určitých kritérii. Ukažme si příkad jak bychom seskupili uživatele do skupin podle jejich věku:

**var** dotaz = **from** u **in** uzivatele

**group** u by u.Vek **into** vekovaSkupina

**select** **new** { Vek = vekovaSkupina.Key, Uzivatele = vekovaSkupina };

Co že jsme to provedli? Seskupili jsme uživatele podle jejich věku do vekovaSkupina, což je kolekce, která obsahuje vždy uživatele se stejným věkem. Je tam tedy např, skupina 15, skupina 16, 17 atd. Dále vybíráme jak bude skupina vypadat. Bude obsahovat věk, ten vezmeme z klíče skupiny, kterým je právě věk, jelikož podle něj seskupujeme. Druhou vlastností skupiny bude kolekce uživatelů, tam uložíme tu aktuální skupinu uživatelů pro daný věk.

Přejděme k výpisu:

**foreach** (**var** skupina **in** dotaz)

{

Console.WriteLine(skupina.Vek);

**foreach** (**var** uzivatel **in** skupina.Uzivatele)

Console.WriteLine(uzivatel.Jmeno);

}

Nejprve proiterujeme všechny skupiny a pro každou skupiny vypíšeme její věk a poté uživatele v ní obsažené.

foreach (var group in Columns.GroupBy(c => new { c.LocId, c.SecId }))

{

# Jak Se da zgroupovat kolekce podle všech propert soucasne:

* [Home](http://vmsdurano.com/)
* [A bit About Me](http://vmsdurano.com/a-bit-about-me/)

public class Product

{

public int ProductID { get; set; }

public string Make { get; set; }

public string Model { get; set; }

}

Now let’s create a method that would create a list of Products. For example:

private List<Product> GetProducts() {

List<Product> products = new List<Product>();

products.Add(new Product { ProductID = 1, Make = "Samsung", Model = "Galaxy S3" });

products.Add(new Product { ProductID = 2, Make = "Samsung", Model = "Galaxy S4" });

products.Add(new Product { ProductID = 3, Make = "Samsung", Model = "Galaxy S5" });

products.Add(new Product { ProductID = 4, Make = "Apple", Model = "iPhone 5" });

products.Add(new Product { ProductID = 5, Make = "Apple", Model = "iPhone 6" });

products.Add(new Product { ProductID = 6, Make = "Apple", Model = "iPhone 6" });

products.Add(new Product { ProductID = 7, Make = "HTC", Model = "Sensation" });

products.Add(new Product { ProductID = 8, Make = "HTC", Model = "Desire" });

products.Add(new Product { ProductID = 9, Make = "HTC", Model = "Desire" });

products.Add(new Product { ProductID = 10, Make = "Nokia", Model = "Lumia 735" });

products.Add(new Product { ProductID = 11, Make = "Nokia", Model = "Lumia 930" });

products.Add(new Product { ProductID = 12, Make = "Nokia", Model = "Lumia 930" });

products.Add(new Product { ProductID = 13, Make = "Sony", Model = "Xperia Z3" });

return products;

}

The method above returns a list of Products by adding a dummy data to the Listjust for the simplicity of this demo. In a real scenario you may want to query your database and load the result to your model. Now let’s bind the Products data in GridView.

protected void Page\_Load(object sender, EventArgs e) {

if (!IsPostBack) {

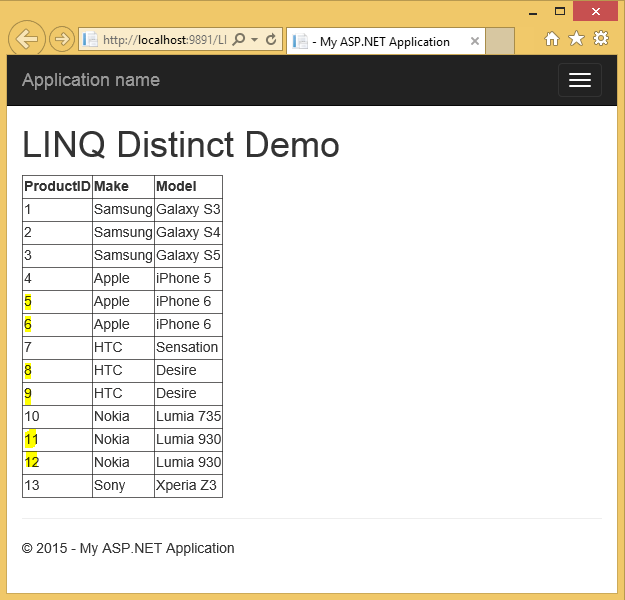
GridView1.DataSource = GetProducts();

GridView1.DataBind();

}

}

Running the code will give you the following output below:



If you noticed there are few items above that contain the same values or commonly called "duplicate" values. Now let’s try to get the distinct row values from the list using the LINQ [Distinct](https://msdn.microsoft.com/en-us/library/vstudio/system.linq.enumerable.distinct(v=vs.100).aspx) function. The code now would look like this:

if (!IsPostBack) {

GridView1.DataSource = GetProducts().Distinct();

GridView1.DataBind();

}

Unfortunately, running the code will **still** give you the same output. This means that the Distinct LINQ function doesn’t work at all. I was surprised and my first reaction was like…



**What??? Really???**

Yes, it doesn’t work as expected! This is because the Distinct method uses the Default equality comparer to compare values under the hood. And since we are dealing with reference type object then the Distinct will threat the values as unique even if the property values are the same.

#### So how are we going to deal with this?

There are few possible ways to accomplish this and these are:

###### Option 1: Using a combination of LINQ GroupBy and Select operators

if (!IsPostBack) {

GridView1.DataSource = GetProducts()

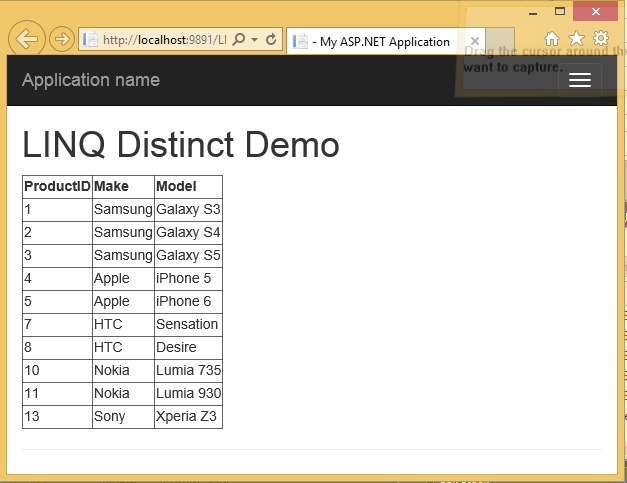
.GroupBy(o => new { o.Make, o.Model })

.Select(o => o.FirstOrDefault());

GridView1.DataBind();

}

###### Output



###### Option 2: Using a combination of LINQ Select and Distinct operators

if (!IsPostBack) {

GridView1.DataSource = GetProducts()

.Select(o => new { o.Make, o.Model })

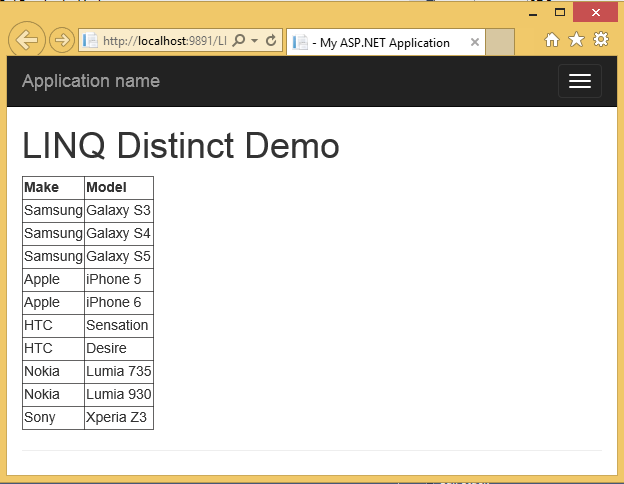
.Distinct();

GridView1.DataBind();

}

The approach above creates a collection of an anonymous types. Doing a Distinct on the anonymous types will automatically override the Equals and GetHashCode to compare each property.

###### Output



###### Option 3: Using the IEqualityCompare interface

class ProductComparer : IEqualityComparer<Product>

{

public bool Equals(Product x, Product y) {

if (Object.ReferenceEquals(x, y)) return true;

if (Object.ReferenceEquals(x, null) || Object.ReferenceEquals(y, null))

return false;

return x.Make == y.Make && x.Model == y.Model;

}

public int GetHashCode(Product product) {

if (Object.ReferenceEquals(product, null)) return 0;

int hashProductName = product.Make == null ? 0 : product.Make.GetHashCode();

int hashProductCode = product.Model.GetHashCode();

return hashProductName ^ hashProductCode;

}

}

The [Distinct](https://msdn.microsoft.com/en-us/library/bb338049.aspx) operator has an overload method that lets you pass an instance of [IEqualityComparer](https://msdn.microsoft.com/en-us/library/ms132151.aspx). So for this approach we created a class “ProductComparer” that implements the IEqualityCompaper. Here’s the code to use it:

if (!IsPostBack) {

GridView1.DataSource = GetProducts()

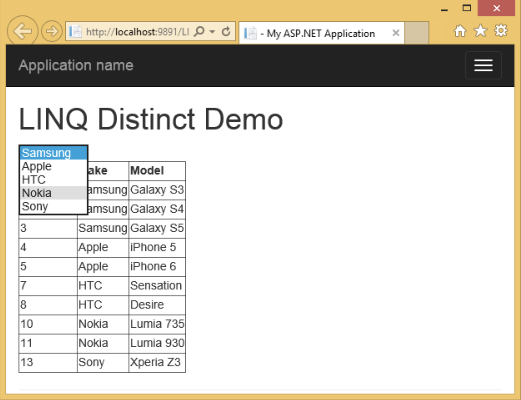
.Distinct(new ProductComparer());

GridView1.DataBind();

}

The approach above is my preferred option because it allows me to implement my own GetHashCode and Equals methods for comparing custom types. Also getting into a habit of making interfaces makes your code more reusable and readable.

###### Output



As you observed the duplicate values are now gone. Now here’s another scenario. What if we want to get the distinct values for a certain field in the list? For example get the distinct “Make” values such as Samsung, Apple, HTC, Nokia and Sony and then populate the result to a DropDownList control for filtering purposes. I was hoping that the Distinct function has an overload that can compare values based on a property or field like GetProducts().Distinct(o => o.PropertyToCompare) but then again it doesn’t seem to have that overload. So I came up with the following workarounds:

###### Option 1: Using GroupBy and Select operators

if (!IsPostBack) {

DropDownList1.DataSource = GetProducts()

.GroupBy(o => o.Make)

.Select(o => o.FirstOrDefault());

DropDownList1.DataTextField = "Make";

DropDownList1.DataValueField = "Make";

DropDownList1.DataBind();

}

###### Option 2: Using Select and Distinct operators

if (!IsPostBack) {

DropDownList1.DataSource = GetProducts()

.Select(o => new { Make = o.Make } )

.Distinct();

DropDownList1.DataTextField = "Make";

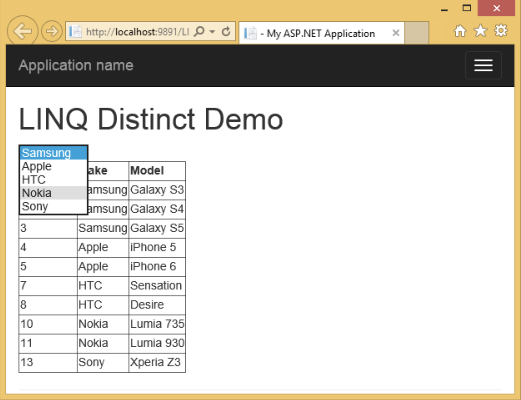
DropDownList1.DataValueField = "Make";

DropDownList1.DataBind();

}

###### Output

Running the code for both options above will give this output below:



That’s it! I hope you will find this post useful. ;)

# Left Join z vesmiru

public IEnumerable<Planeta> LoadPlanetsBasedOnGalaxyId(int id)

{

using (UniverseDataModel model = CreateDbContext())

{

var query = (from pl in model.Planeta // Vybirame z tabulky Planeta

// VYTRIDIME SI KOLEKCI VLASTNOSTI

join vp in model.VlastnostiPlanet on pl.Id equals vp.PlanetaId into vpl // Z tabulky VlastnostiPlanet vyber ty polozky, kde se Id planety shoduje s PlanetaId, uloz si je prozatim do kolekce vpl. Kolekce obsahuje jen VlastnostiPlanet kde pl.Id == vp.PlanetaId

from vp in vpl.DefaultIfEmpty() // Z VlastnostiPlanet ve vpl, (pokud je prazdna (nejsou zadne vlastnosti) vyber defaultni),

join v in model.Vlastnost on vp.VlastnostId equals v.Id into vl // vyber z tabulky Vlastnosti ty, ktere maji Id shodne s VlastnostId polozek ve vpl a uloz si je prozatim do kolekce vl. Kolekce obsahuje jen Vlastnosti kde v.Id == vp.VlastnostId

from v in vl.DefaultIfEmpty() // Pokud neni kolekce vl prazdna, vyber z ni vlastnost,

where pl.GalaxieId == id // Vyber planety ve kterych GalaxieId odpovida parametru metody.

select new { Planet = pl, Property = v }); // Vytvor kolekci objektu, ktere budou mit vlastnosti Planet a Property.

// MAME KOLEKCI VE KTERE SE PLANETY OPAKUJOU PODLE TOHO KOLIK MAJI VLASTNOSTI MUSIME JE ROZTRIDIT DO KOLEKCI PODLE KLICE ID PLANETY.

var groupedByPlanet = query.ToList().GroupBy(x => x.Planet.Id);

List<Planeta> result = new List<Planeta>(); // Nove uloziste

foreach (var dato in groupedByPlanet) // Pro kazdou skupinu ve skupinach serazenych podle planety, (v kazde skupine se opakuje planeta tolikrat, kolikrat ma vlastnost.)

{

var planet = dato.First().Planet; // Planeta bude prvni polozka

planet.Properties = dato.Where(d => d.Property != null).Select(x => x.Property).ToList(); // Do vlastnosti teto nasi planety vlozime cely hotovy List. Vezmeme ho ze skupiny dato => Linq dotaz nam prosel vsechny polozky ve skupine a vratil seznam vsech objektu Property. (Vybirame z anonymniho typu ktery ma vlastnosti Planet a Property).

result.Add(planet); // Vloz do resultu.

}

return result; // Vrat result.

}

}

}

# Group by , then by

Ve zmenovadle bylo potreba seskupit najoinovanou tabulku mtm a tabulku motivlet nejprve podle mtm.MotiveVersion a pak podle propert na motivletu.

var query = from mtm in model.TempMotiveVersionToMotivlet **Nejprve si najoinuju potrebne tabulky**

join mot in model.TempMotivlet on mtm.MotivletId equals mot.Id

into gr

from mot in gr.DefaultIfEmpty()

select new { Mtm = mtm, Mot = mot }; **Ziskam anonymni typ**

var jedna = from c in query.ToList() **Musim brat z listu jinak to pada**

group c by c.Mtm.MotiveVersionId **Seskupim podle VersionId**

into g

select new

{

Group = g,

SubGroups = from c in g **Tady je figl: to druhe grupovani je uvnitr selectu**

group c by new

{ **Mam podskupiny zgrupovane podle stejnych propert**

c.Mot.OwnerId,

c.Mot.CompanyBrandId,

c.Mot.ProductBrandId,

c.Mot.ProductDetailId,

c.Mot.CategoryId,

c.Mot.RoleId,

c.Mot.MarketId,

c.Mot.GenderId,

c.Mot.PlatformId

}

into g2

where g2.Count() > 1 **Hledam duplicity, takze chci jen skupiny, kde je pocet polozek > 1**

select g2

};

var groups = jedna.Select(d => d.SubGroups.SelectMany(x => x)).ToList();  **Nejprve vytvorim list skupin**

var dal = groups.Select(d => d.Select(x => x.Mtm)).SelectMany(d => d).ToList(); **Z nej si vybiram vazby**

return dal.Select(d => new MotiveVersionToMotivletDataContract **A ty uz jen prevedu na dataContracty**

{

MotiveVersionId = d.MotiveVersionId,

MotivletId = d.MotivletId

}).ToList();