**Object Queries**

**.NET Framework 4**

[Other Versions](javascript:;)

http://i3.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635533092500576233

3 out of 6 rated this helpful - [Rate this topic](http://msdn.microsoft.com/en-us/library/vstudio/bb896241%28v=vs.100%29.aspx#feedback)

The [ObjectQuery](http://msdn.microsoft.com/en-us/library/vstudio/bb345303%28v=vs.100%29.aspx) generic class represents a query that can return a collection of zero or more typed objects. An **ObjectQuery** belongs to an [ObjectContext](http://msdn.microsoft.com/en-us/library/vstudio/system.data.objects.objectcontext%28v=vs.100%29.aspx) that contains the connection and metadata information that is necessary to compose and execute the query. You can construct an **ObjectQuery** with a **new** operator and pass a query string and the object context to the constructor. However, a more common scenario is to use properties on an **ObjectContext** derived class to get an **ObjectQuery** instance that represents a collection of entity sets. Typically, the **ObjectContext** is subclassed, either by a class generated by the Entity Framework tools or by your POCO classes, and the properties on the object context return entity sets as either an **ObjectQuery** (in .NET Framework version 3.5 SP1) or as an [ObjectSet](http://msdn.microsoft.com/en-us/library/vstudio/dd412719%28v=vs.100%29.aspx) (in .NET Framework version 4). The **ObjectSet** class extends the **ObjectQuery** class to provide functionality, such as adding and deleting objects, in the context of a typed entity set.

The default **ObjectQuery** provides a starting query that returns all entities of the specified type. This query can be further refined by using LINQ to Entities or query builder methods.

The following example queries the object context for the collection of Products.

C#

[VB](http://msdn.microsoft.com/en-us/library/vstudio/bb896241%28v=vs.100%29.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

using (AdventureWorksEntities context = new AdventureWorksEntities())

{

IQueryable<Product> productsQuery = from product in context.Products

select product;

Console.WriteLine("Product Names:");

foreach (var prod in productsQuery)

{

Console.WriteLine(prod.Name);

}

}

**Query Execution**

An object query is executed when:

* It is enumerated by a **foreach** (C#) or **For Each** (Visual Basic) statement.
* It is enumerated by a collection operation such as [ToArray](http://msdn.microsoft.com/en-us/library/vstudio/bb298736%28v=vs.100%29.aspx), [ToDictionary](http://msdn.microsoft.com/en-us/library/vstudio/system.linq.enumerable.todictionary%28v=vs.100%29.aspx) or [ToList](http://msdn.microsoft.com/en-us/library/vstudio/bb342261%28v=vs.100%29.aspx).
* The [Execute](http://msdn.microsoft.com/en-us/library/vstudio/bb343787%28v=vs.100%29.aspx) method is explicitly called.
* LINQ operators such, as [First](http://msdn.microsoft.com/en-us/library/vstudio/bb291976%28v=vs.100%29.aspx) or [Any](http://msdn.microsoft.com/en-us/library/vstudio/bb337697%28v=vs.100%29.aspx) are specified in the outermost part of the query. For more information, see [Query Builder Methods](http://msdn.microsoft.com/en-us/library/vstudio/bb896238%28v=vs.100%29.aspx).

Note, if as a result of a query execution, nothing was returned from the data source, the results will contain an empty collection and not a **null**.

Queries executed by the Entity Framework are evaluated against the data in the data source and the results will not reflect against the new objects in the object context. If an entity with the same identity as the one being queried for is already attached to the context, the data coming from the data source and the data already in the context are merged according to the [MergeOption](http://msdn.microsoft.com/en-us/library/vstudio/system.data.objects.mergeoption%28v=vs.100%29.aspx) of the query. To get the data that is in the cache, use the [GetObjectStateEntries](http://msdn.microsoft.com/en-us/library/vstudio/bb738497%28v=vs.100%29.aspx) method on the [ObjectStateManager](http://msdn.microsoft.com/en-us/library/vstudio/system.data.objects.objectstatemanager%28v=vs.100%29.aspx) class. The **ObjectStateManager** manages the state of objects inside an object context, so if you want to get all the objects that were added, modified and unchanged, you can pass a bitwise OR of the following [EntityState](http://msdn.microsoft.com/en-us/library/vstudio/system.data.entitystate%28v=vs.100%29.aspx) values to the **GetObjectStateEntries** method: [Added](http://msdn.microsoft.com/en-us/library/vstudio/system.data.entitystate%28v=vs.100%29.aspx), [Modified](http://msdn.microsoft.com/en-us/library/vstudio/system.data.entitystate%28v=vs.100%29.aspx), [Unchanged](http://msdn.microsoft.com/en-us/library/vstudio/system.data.entitystate%28v=vs.100%29.aspx). For more information see the [blog](http://go.microsoft.com/fwlink/?LinkId=182371) that demonstrates how to perform local queries.

In the following example, the **Execute** method is called to execute a query:

C#

[VB](http://msdn.microsoft.com/en-us/library/vstudio/bb896241%28v=vs.100%29.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

using (AdventureWorksEntities context =

new AdventureWorksEntities())

{

ObjectSet<Product> query = context.Products;

// Execute the query and get the ObjectResult.

ObjectResult<Product> queryResult = query.Execute(MergeOption.AppendOnly);

// Iterate through the collection of Product items.

foreach (Product result in queryResult)

Console.WriteLine("{0}", result.Name);

}

**Query Projection**

Object queries are often used to return conceptual model data as entity objects, but they may also return a [DbDataRecord](http://msdn.microsoft.com/en-us/library/vstudio/system.data.common.dbdatarecord%28v=vs.100%29.aspx) object for nested results and anonymous projections, or they can return primitive CLR types for sets of single values.

LINQ to Entities and Entity SQL both support query projection. The following considerations apply to query projections:

* Some extension methods require collection of many results as input. If a **ObjectQuery** represents a query that returns a collection with a single scalar result, and one of these extension methods is called, an [ArgumentException](http://msdn.microsoft.com/en-us/library/vstudio/system.argumentexception%28v=vs.100%29.aspx) is thrown, as in the following example.

C#

[VB](http://msdn.microsoft.com/en-us/library/vstudio/bb896241%28v=vs.100%29.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-3)

// Define a query projection that returns

// a collection with a single scalar result.

ObjectQuery<Int32> scalarQuery =

new ObjectQuery<Int32>("100", context);

// Calling an extension method that requires a collection

// will result in an exception.

bool hasValues = scalarQuery.Any();

* If an **ObjectQuery** might return a **null** value when projected to a primitive type, you should use the nullable version of the type. The following query uses a nullable [DateTime](http://msdn.microsoft.com/en-us/library/vstudio/system.datetime%28v=vs.100%29.aspx) because the **ShipDate** property of the **SalesOrderHeader** object might return a **null** value.

C#

[VB](http://msdn.microsoft.com/en-us/library/vstudio/bb896241%28v=vs.100%29.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-4)

ObjectQuery<Nullable<DateTime>> shipDateQuery =

context.SalesOrderHeaders

.Where("it.CustomerID = @contactId",

new ObjectParameter("contactId", contactId))

.SelectValue<Nullable<DateTime>>("it.ShipDate");

For more information, see [Nullable Types (Visual Basic Programming Guide)](http://msdn.microsoft.com/en-us/library/vstudio/ms235245%28v=vs.100%29.aspx) or [Nullable Types (C# Programming Guide)](http://msdn.microsoft.com/en-us/library/vstudio/1t3y8s4s%28v=vs.100%29.aspx).

**Viewing Store Commands**

When you query a conceptual model, the Entity Framework transforms the LINQ to Entities and Entity SQL query based on the conceptual model into an equivalent query against the data source. The Entity Framework provides the [System.Data.Objects.ObjectQuery.ToTraceString](http://msdn.microsoft.com/en-us/library/vstudio/system.data.objects.objectquery.totracestring%28v=vs.100%29.aspx) and [System.Data.EntityClient.EntityCommand.ToTraceString](http://msdn.microsoft.com/en-us/library/vstudio/system.data.entityclient.entitycommand.totracestring%28v=vs.100%29.aspx) methods, which enable you to view these store commands at runtime without having to run a trace against the data source. For more information, see [How to: View the Store Commands](http://msdn.microsoft.com/en-us/library/vstudio/bb896348%28v=vs.100%29.aspx).

**Retrieving an Object by Its EntityKey**

If you know the key value of an entity, you can retrieve it from the data source without explicitly creating and executing an object query. The [GetObjectByKey](http://msdn.microsoft.com/en-us/library/vstudio/system.data.objects.objectcontext.getobjectbykey%28v=vs.100%29.aspx) and [TryGetObjectByKey](http://msdn.microsoft.com/en-us/library/vstudio/bb738728%28v=vs.100%29.aspx) methods on **ObjectContext** will return an object with the specified [EntityKey](http://msdn.microsoft.com/en-us/library/vstudio/system.data.entitykey%28v=vs.100%29.aspx) into the object context. When you use **GetObjectByKey**, you must handle an [ObjectNotFoundException](http://msdn.microsoft.com/en-us/library/vstudio/system.data.objectnotfoundexception%28v=vs.100%29.aspx) when the provided **EntityKey** does not correspond to an existing entity. For more information, see [How to: Return a Specific Object Using its Key](http://msdn.microsoft.com/en-us/library/vstudio/bb896251%28v=vs.100%29.aspx).