Contents

When the *EntityServer* receives a client query request, it looks on the server for a corresponding query method such as a named query or POCO query method. It recognizes a server-side query method by its signature and name. This topic explains the DevForce **query method naming convention** and how to use the *QueryAttribute* when you can't follow the convention.

The client query

At the root of a client *EntityQuery* is the name of the *EntitySet* that identifies the domain of entities to query. Here is an example client query for *Customers*.

```
var query = new EntityQuery<Customer>("Customers", anEntityManager).Where(...);
```

```
Dim query = New EntityQuery(Of Customer)("Customers", anEntityManager).Where(...)
```

"Customers" is the name of *EntitySet* for the *Customer* entity type.

The more familiar anEntityManager.Customers() property returns an *EntityQuery* that is defined in precisely this way.

If your query depends upon a specialized named query, the *EntitySet* name will be something else. It might be "GoldCustomers" if you've defined a named query that implements that concept.

Finding the server-side query method

When the *EntityServer* receives that client query, it looks for a method on the server that corresponds to the *EntityQuery*'s *EntitySet* name.

If it can't find a corresponding query method on the server there could be trouble. In most cases the *EntityServer* throws an exception reporting its inability to find a method to go with the requested *EntitySet* name.

There is one exception. The *EntityServer* can muddle through if the query's *EntitySet* name matches the *EntitySet* name of an Entity Framework entity type. Because "Customers" is the name of the *EntitySet* for the *Customer* EF type, DevForce doesn't **have** to find a matching query method on the server. It will try to find a server-side method ... but it can proceed without one.

That's why you do not have to write a *default named query method*. If you **do** write a default named query method, make sure DevForce can find it by following the rules described here.

If you wrote a server-side query method, you must help DevForce find it, either by naming the method in a way that conforms to DevForce conventions or by marking it with the DevForce *Query* attribute.

Query naming conventions

DevForce can interpret query method names that begin with any one of the following prefixes:

Prefix	Method name example	Matching *EntitySet* name
Get	GetCustomers	Customers
Query	QueryCustomers	Customers
Fetch	FetchGoldCustomers	GoldCustomers
Retrieve	RetrieveStates	States
Find	FindWaldo	Waldo
Select	SelectCandidates	Candidates

Using the *Query* Attribute

You don't have to follow the DevForce conventions when naming a specialized named query or POCO query.

You must follow the DevForce convention when naming the *default named query method*.

You can name the query method as you please as long as you adorn it with the *Query* attribute.

```
[Query]
```

```
public IQueryable<Customer> ReturnAllCustomers() { ... }
[Query]
public IQueryable<Customer> GrabMeSomeGoldCustomers() { ... }
[Query]
public IEnumerable<State>   BringBackTheStates() { ... }
```

```
<Query()> _
Public Function ReturnAllStates() As IQueryable(Of Customer) ...
<Query()> _
Public Function GrabMeSomeGoldCustomers() As IQueryable(Of Customer) ...
<Query()> _
Public Function BringBackTheStates() As IEnumerable(Of State) ...
```

Now create the client query using the full server-side method name as the *EntitySet* name:

```
var customersQuery     = new EntityQuery<Customer>("ReturnAllStates", anEntityManager);
var goldCustomersQuery = new EntityQuery<Customer>("GrabMeSomeGoldCustomers", anEntityManager);
var statesQuery        = new EntityQuery<State>("BringBackTheStates", anEntityManager);
```

```
Dim customersQuery     = New EntityQuery(Of Customer)("ReturnAllStates", anEntityManager)
Dim goldCustomersQuery = New EntityQuery(Of Customer)("GrabMeSomeGoldCustomers", anEntityManager)
Dim statesQuery        = New EntityQuery(Of State)("BringBackTheStates", anEntityManager)
```

Other discoverability requirements

It isn't enough to name the method properly. You must also

- Ensure the query method signature is appropriate for the root of the *EntityQuery*.
- Define the method within a class marked by the *EnableClientAccess* attribute.
- Locate that class in a discovered assembly.