Contents

Querying for POCO entities from your *EntityManager* is performed the same as with non-POCO entities, with the single exception that DevForce has not auto-generated an EntityQuery for these entities on the *EntityManager* class. There are several ways to add these queries to **make your POCO query a first class citizen**.

Create the query directly

Your first choice is to simply create the *EntityQuery* directly:

```
var baseQuery = new EntityQuery<State>("States", anEntityManager);
var queryForstatesStartingWithA = query.Where(s => s.Abbrev.StartsWith("A");
```

```
Dim baseQuery = New EntityQuery(Of State)("States", anEntityManager)
Dim queryForstatesStartingWithA = query.Where(Function(s) s.Abbrev.StartsWith("A")
```

Remember that the *EntitySet* or query method name is passed to the *EntityQuery* constructor so that the appropriate service provider query method will be called to fulfill the query. Here "States" is the *EntitySet* name.

Create an extension method

The next choice is to create an extension method which adds a method returning the *EntityQuery* to your EntityManager class. Here we add an extension method named "States":

```
namespace DomainModel {
public static class EmExtensions {
  public static EntityQuery<State>States(this EntityManager em) {
    return new EntityQuery<State>("States", em);
   }
 }
}
```

```
Namespace DomainModel
Public Module EmExtensions
   <System.Runtime.CompilerServices.Extension> _
  Public Function States(ByVal em As EntityManager) As EntityQuery(Of State)
    Return New EntityQuery(Of State)("States", em)
  End Function
End Module
End Namespace
```

With this extension method you can now query for States with the following syntax:

```
var queryForstatesStartingWithA = anEntityManager.States().Where(s => s.Abbrev.StartsWith("A"));
```

```
Dim queryForstatesStartingWithA = _
  anEntityManager.States().Where(Function(s) s.Abbrev.StartsWith("A"))
```

Note that because we're using an extension method you must include parentheses when referencing *anEntityManager.States()*.

Extend your strongly typed EntityManager partial class

Another alternative to the above is to add a "States" EntityQuery property to your own custom partial class which extends your strongly-typed EntityManager. This makes querying for POCO objects syntactically identical to querying for non-POCO objects.

```
public partial class DomainModelEntityManager : IdeaBlade.EntityModel.EntityManager {
 public EntityQuery<State> States {
   get { return new EntityQuery<State>("State", this); }
  }
}
```

```
Partial Public Class DomainModelEntityManager
 Inherits IdeaBlade.EntityModel.EntityManager
 Public ReadOnly Property States() As EntityQuery(Of State)
   Get
     Return New EntityQuery(Of State)("State", Me)
```

```
    End Get
  End Property
End Class
```