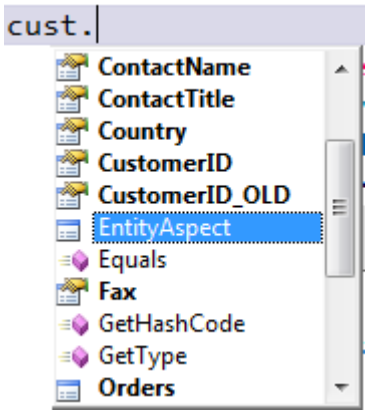
Contents

You **access the internals of an entity** through the entity's **EntityAspect** property. *EntityAspect* is your gateway to the DevForce infrastructure hidden within each entity. The *EntityAspect* tells you whether the entity has changed and how, and allows you to perform basic functions such as adding and removing the entity from its EntityManager. This topic covers what you can do with and learn from the *EntityAspect*.

Hiding entity infrastructure

An entity represents a thing in the application domain. A Customer represents "customer-ness". You modeled the Customer entity to suit the needs of programmers who write code about customers. The Customer entity should exhibit "customer-ness". Here is what a Customer instance looks like in IntelliSense:



Almost all members are Customer-specific. They are data properties, navigation properties, and custom methods (if we had some).

A few members concern infrastructure having nothing to do with Customer *per se*. These are members of the Customer's base classes such as Object.GetHashCode and Object.GetType ... and *Entity.EntityAspect*.

They are noise when our attention is on "customer-ness". They can distract us from our primary purpose - to do business with a customer.

In an ideal world they would be invisible. In practice, we need them. At various times we have to set aside our "Customer" perspective and see the Customer entity as a .NET object. Sometimes we need to work with it as *Object*. Sometimes we need to work with it as an *Entity*. When we need to shift our perspective, we need to shift quickly.

The architect's challenge is to keep "customer-ness" on center stage **and** have "object-ness" and "entity-ness" quietly ready in the wings.

You have to access the *entity* internals of Customer from time to time. Rather than clutter Customer with lots of "entity-ness" members, DevForce wraps them in a single helper object, the *EntityAspect*. The *EntityAspect* property is the place to go for "entity-ness" - the place to go for access to the inner entity nature of a Customer.

What's in *EntityAspect*

You'll most often use *EntityAspect* to determine the state of the entity: is it *Added*, has it been changed, is it null or pending? You'll also use *EntityAspect* to work directly with validation errors or to access entity metadata. In general, though, you don't often need to work with the *EntityAspect*, as DevForce will handle the entity for you as you query, modify and save your entities.

Learn the state of the entity

Member	Purpose
EntityKey	The value that uniquely identifies this entity.
EntityState	The state of the entity - whether unchanged, added, modified, deleted or detached.
HasChanges	If changes are pending for this entity.
IsChanged	If changes are pending for this entity, i.e., EntityState is added, modified, or deleted.
IsNullEntity	If this entity is the special "null entity" ("nullo"). A reference navigation property returns such a null entity when it can't return the related entity (rather than return null and risk an exception).
IsPendingEntity	If this entity instance is a placeholder for a real entity of this type. An entity will be pending when using asynchronous navigation and the retrieval from the database has not completed.
PendingEntityResolved	Raised when a pending entity has been retrieved from the database.
IsNullOrPendingEntity	If this entity is either the null entity or a pending entity. A quick way to tell if the entity is real and complete.

Change the state of the entity

Other than calling *Delete*, in most circumstances you won't need to use the methods described below, as DevForce will automatically handle the state of the entity as it's queried, modified, and saved.

Member	Purpose
Delete	Mark this entity for deletion; the entity will be deleted from the database when saved.
RejectChanges	Undo pending changes.
AcceptChanges	Commit pending changes locally as if the entity were saved and now unchanged. Beware! Does not update the database. You do not need to call *AcceptChanges* during normal save processing, as DevForce will do it for you when a save completes successfully.
SetAdded	Convert the state of the entity such that it appears to have been newly created. If saved, DevForce will try to insert it into the database. When you create a new entity and add it to an *EntityManager*, it will automatically be marked as *Added* and you do not need to call this method.
SetModified	Convert the state of the entity such that it appears to be a modified entity that came from the database. If saved, DevForce will try to find and update the corresponding database object. When you modify an entity in cache it will automatically be marked as *Modified* by DevForce and you do not need to call this method.
GetValue	Get the value of the named property. Could get the current value of the property or one of the other versions.
SetValue	Set the current value of the named property.
ForcePropertyChanged	Raise the entity's *PropertyChanged* event on demand.

Learn about the entity's validity

Member	Purpose
ValidationErrors	Get a modifiable collection of the currently detected validation errors for this entity.
VerifierEngine	Get the VerifierEngine used by this entity's auto-validating property setters. The engine always comes from the entity's EntityManager. It is null if the entity is detached; for this reason property setters don't validate input automatically when the entity is detached.

Work with the entity's EntityManager

Member	Purpose
EntityManager	Get the *EntityManager* to which the entity is attached. Null if detached.
AddToManager	Add the entity to its associated *EntityManager* as a new entity. An *EntityManager* is associated with an entity when the entity is created with the CreateEntity method.
RemoveFromManager	Detach the entity from its *EntityManager*. Removing an entity from its *EntityManager* is not the same thing as deleting an entity. The entity will not be deleted from the back end data source with *RemoveFromManager*.
FindRelatedEntities	Find cached entities related to this entity by a specific link.
GetRelatedEntities	Get entities related to this entity by a specific link; depending on circumstances and parameters DevForce could retrieve them from the database (asynchronously in Silverlight).

EntityGroup	Gets the collection of entities to which this entity belongs. When this entity is attached to an EntityManager, the EntityGroup contains this entity and every other entity of its type in the cache. EntityGroup events reveal when entities in the group are added, removed, or changed.

Get Metadata about the entity

Member	Purpose
EntityMetadata	Get model metadata about this entity's type.
GetProperty	Get an *EntityProperty* metadata object by name with which you can inspect or modify property values generically.
GetDataProperty	Get a data meta-property by name which you can inspect or use to get and set values generically.
GetNavigationProperty	Get navigation meta-property by name which you can inspect or use to add and remove related entities generically.

EntityAspect's own stuff

Member	Purpose
Entity	A back-reference to get the entity for which this is the *EntityAspect*.
Wrapper	A back-reference to the *EntityWrapper* for which this is the *EntityAspect*. For a DevForce generated entity the *Entity* and *Wrapper* properties return the same entity object because the entity inherits from both *Entity* and *EntityWrapper*. The distinction between *Entity* and *EntityWrapper* is relevant only for POCOs.
PropertyChanged	Raised by *EntityAspect* when certain of its **own** properties change such as *EntityState*, *EntityKey*, and *HasChanges*. Not to be confused with *PropertyChanged* event for the **entity**.