One question that comes up regularly is how does a developer **control the order in which his entities are saved** to the database.

The simple answer is: **He or she doesn't.**

The rules are actually a little different for POCO's.

This is actually a good thing. It turns out that DevForce and the Entity Framework do an excellent job of determining the order that entities need to be saved so as to avoid violating database constraints. In fact, this is probably something that most developers would rather not have to worry about. It isn't simply a matter of specifying that certain types get saved before other types, it turns out that correctly ordering a save may require type A to be saved before type B for one instance of type A but after it for another. The rules regarding whether to perform deletes before or after adds and modifications change in some cases when cascading deletes are involved. So, in general, let DevForce and the Entity Framework, do the save ordering for you.

Ok, but what if you have a really strange case that DevForce cannot handle? **This really shouldn't happen, so please let us know if it does so that we can understand the use case.**

You have two choices:

- Option 1) Break your saves into groups and call *SaveChanges* more than once. The problem with this is that you will lose the ability to treat the combined saves as a single transaction.
- Option 2) Create an *EntityCacheState* object with your changes and make an *InvokeServerMethod* call to a 'custom' server side method with this object. Within the server side implementation, create a new *TransactionScope* and inside this transaction scope do the following:
  - Create a new *EntityManager* and load the *EntityCacheState* object into it.
  - Perform **option 1** above. You are allowed to call *SaveChanges* within your server side method.
  - DevForce will 'notice' that its saves are being called within an 'outer' *TransactionScope* and will force all of its operations to participate in this scope. This is what gives you back a 'Transactional Save' even though you are doing it in pieces.