Quick note

Whenever I say that “this member is not used in our case”, I mean that the member in question should normally not be used in Cresus.Core, but is probably used in the kind of outdated stuff with the form engine and the dialogs which auto binds themselves with the entities. It might happen that the member is not used at all. But the important point, is that it will never be called in Cresus.Core, unless some assumption that I made is wrong or has not been respected in the implementation of Cresus.Core. Those assumptions are mostly that we always have proper entities that inherit from AbstractEntity and to which corresponds a StructuredType, that these entities don’t mess with their internal state (and more generally that nothing messes with their internal state), and that we don’t use GenericEntity, SearchEntity, … and that we don’t use entities as IStrucuredData or IValueStore, etc. There are probably other assumptions that I made which I don’t remember. All this stuff is real messy so my brain is kind of upside down right now.

Thread safety concerns for EntityContext

|  |  |
| --- | --- |
| Member | Note |
| Constructor | Thread safe by default, it cannot be called by more than one thread at once on the same instance and we don’t leek the instance being constructed. |
| Static constructor | Thread safe by default, it cannot be called by more than one thread at once. |
| DataGeneration | This member is not supposed to be called by multiple threads on the same instance. |
| SkipConstraintChecking | The mechanism provided by this member is never really used in our case. |
| ExceptionManager | The mechanism provided by this member is never really used in our case. |
| PersistenceManager | This member is not supposed to be called by multiple threads on the same instance. |
| CompareEqual | This member is not used in our case. |
| GetPersistedId | This member is not used in our case. |
| GetPersistedEntity | This member is not used in our case. |
| NewDataGeneration | This member is not supposed to be called by multiple threads on the same instance. |
| SuspendConstraintChecking | This member is not used in our case. |
| EvaluateFunc | This member is not used in our case. |
| CreateValueStore | Thread safe because all the methods that it calls are thread safe and the state of the objects that it uses Is considered as immutable. |
| GetValueStoreDataIds | This member is not used in our case. |
| CreateSearchEntity | This member is not used in our case. |
| IsSearchEntity | This member is not used in our case. |
| CreateEmptyEntity | When this member is called in our case, we know that the lock has been acquired in write mode. |
| CreateEmptyEntity | When this member is called in our case, we know that the lock has been acquired in write mode. |
| InitializeDefaultValues | Thread safe because the main body of the method is surrounded by a call to the member AbstractEntity.DefineOriginalValues() which acquires the lock in exclusive mode. |
| CreateEntity | This member is not used in our case. |
| CreateEntity | This member is not used in our case. |
| CreateGenericEntity | This member is not used in our case. |
| CreateRelatedEntities | This member is not used in our case. |
| CreateRelatedEntities | This member is not used in our case. |
| CreateChildEntity | This member is not used in our case. |
| HandleGraphLoop | This member is not used in our case. |
| DisableCalculations | This member is not used in our case. |
| FindPropertySetter | This member is not used in our case. |
| FindPropertyGetter | This member is not used in our case. |
| FindPropertyInfo | This member is not used in our case. |
| NotifyEntityAttached | When this member is called in an EntityContext which can expect calls from multiple threads at once, the caller should already have acquired the lock in write mode. |
| NotifyEntityDetached | When this member is called in an EntityContext which can expect calls from multiple threads at once, the caller should already have acquired the lock in write mode. |
| NotifyEntityChanged | Ehen this member is called in an EntityContext which can expect calls from multiple threads at once, the caller should already have acquired the lock. |
| GetCaption | Thread safe because this member forwards the call to the underlying SafeResourceResolver which is thread safe. |
| GetStructuredType | Thread safe because this member forwards the call to the underlying SafeResourceResolver which is thread safe. |
| GetStructuredType | Thread safe because in our case we read immutable data and call the underlying SafeResourceResolver which is thread safe. |
| GetStructuredTypeField | Thread safe because we call the underlying SafeResourceResolver which is thread safe and we assume that we extract the structured type field from the structured type in a thread safe way. |
| GetEntityFieldIds | Thread safe because we call the underlying SafeResourceResolver which is thread safe and we assume that we extract the structured type field ids from the structured type in a thread safe way. |
| IsNullable | Thread safe because we call the underlying SafeResourceResolver which is thread safe and we assume that we extract the structured type field and the info on whether it is nullable from the structured type in a thread safe way. |
| IsNullable | Thread safe because we call the underlying SafeResourceResolver which is thread safe and we assume that we extract the structured type field and the info on whether it is nullable from the structured type in a thread safe way. |
| IsNullable | Thread safe because we extract the info on whether the structured type field is nullable from in a thread safe way. |
| Private Data class | GetValue, SetValue and GetStructuredType are supposed to be accessed only when we already have the lock on the entity. GetStructuredType reads and returns immutable data. |
| Event members | They don’t really need synchronization because they are naturally as thread safe as possible. As usual, see comments in DataContext. |

Thread safety concerns for AbstractEntity

|  |  |
| --- | --- |
| Member | Note |
| Constructor | Thread safe by default, it cannot be called by more than one thread at once on the same instance and we don’t leek the instance being constructed. |
| IsReadOnly | This member is considered as immutable in the case where we want to have thread safety because it is written only once in the entity creation process in the DataContext (or in the EntityNullReferenceVirtualizer), which ensures that we have acquired the write lock and that there is no way that other threads can access the entity. So it is safe to read it afterwards because it will never change. |
| IsDefiningOriginalValues | In the case where we want thread safety, this member will only be called by caller which will have acquired the write lock on the entity. |
| IsUpdateSilent | In the case where we want thread safety, this member will only be called by caller which will have acquired the write lock on the entity. |
| AreEventEnabled | In the case where we want thread safety, this member will only be called by caller which will have acquired the write lock on the entity. |
| ReadOnlyChecksEnabled | In the case where we want thread safety, this member will only be called by caller which will have acquired the read or write lock on the entity. |
| GetSummary | Virtual method that when overridden access only fields of entities with the auto generated getter/setters so we don’t need to be thread safe. This assumes that the method does not mess with the internal state. |
| GetCompactSummary | Virtual method that when overridden access only fields of entities with the auto generated getter/setters so we don’t need to be thread safe. This assumes that the method does not mess with the internal state. |
| GetEntityKeywords | Virtual method that when overridden access only fields of entities with the auto generated getter/setters so we don’t need to be thread safe. This assumes that the method does not mess with the internal state. |
| IsEntityEmpty | Simply calls GetEntityStatus |
| IsEntityValid | Simply calls GetEntityStatus |
| GetEntityStatus | Virtual method that when overridden access only fields of entities with the auto generated getter/setters so we don’t need to be thread safe. This assumes that the method does not mess with the internal state. |
| Freeze | In the case where we want thread safety, this member will be called only once in the entity creation process in the DataContext (or in the EntityNullReferenceVirtualizer), which ensures that we have acquired the write lock and that there is no way that other threads can access the entity (because it has not yet leeked to the outside world). |
| CalculationDisabled | This member is not used in our case. |
| DebuggerDisplayView | Thread safe because it only calls this.GetType(). |
| DataContextId | This member is considered as immutable in the case where we want to have thread safety because it is written only once in the entity creation process in the DataContext, which ensures that we have acquired the write lock and that there is no way that other threads can access the entity. We don’t care about the case when the entity can be assigned to the DataContext from the EntityNullReferenceVirtualizer, because “null” entities will stay “null” entities forever in the case where we want to achieve thread safety, because of the readonly property of the entities. |
| GetStructuredTypeId | Should only read immutable data. |
| GetEntityStructuredTypeKey | Should only read immutable data. |
| GetEntityContext | This member acquires the lock in shared mode because it relies on mutable internal state but it does not alter it. |
| GetEntityDataState | This member is not used in our case. |
| GetEntitySerialId | This member only reads immutable data. |
| GetEntityDataGeneration | This member acquires the lock in read mode because it relies on mutable data. |
| ContainsDataVersion | This member is not used in our case. |
| ForEachField | This member is not used in our case. |
| ForEachField | This member is not used in our case. |
| Dump | This member is not used in our case. |
| DumpFlatData | This member is not used in our case. |
| Dump | This member is not used in our case. |
| DefineOriginalValues | This member locks the entity in write mode because it alters the way the entity is perceived in the outside world, so we want to avoid that one threads defines the original values while another thread does something with the entity. The write lock is released when the helper disposable object returned is disposed. |
| UseSilentUpdates | This member locks the entity in write mode because it alters the way the entity is perceived in the outside world, so we want to avoid that one threads defines the original values while another thread does something with the entity. The write lock is released when the helper disposable object returned is disposed. |
| DisableEvents | This member locks the entity in write mode because it alters the way the entity is perceived in the outside world, so we want to avoid that one threads defines the original values while another thread does something with the entity. The write lock is released when the helper disposable object returned is disposed. |
| DisableReadonlyChecks | This member locks the entity in write mode because it alters the way the entity is perceived in the outside world, so we want to avoid that one threads defines the original values while another thread does something with the entity. The write lock is released when the helper disposable object returned is disposed. |
| DisableCalculations | This member is not used in our case. |
| Resolve | This member is not used in our case. |
| GetField | Only calls InternalGetValue (which acquires the write lock) and processes its output. |
| GetFieldCollection | Locks the entity in write mode during the duration of the method because it might modify the internal state if the collection is not yet defined. Moreover, we return a wrapper around the collection if necessary, in order to ensure that the accesses on the collection are thread safe and acquire the lock if necessary or throws a readonly exception if necessary. This assumes that if we are in the special killer mode with a readonly entity without readonly checks, we don’t keep collections around for later usage, otherwise we will be allowed to access to the underlying collection without the thread safe/readonly wrapper. |
| SetField | This member acquires the lock in exclusive mode. This is probably overkill, but by doing this we ensure that we have the lock and that we will throw the exception if the readonly checks are enabled or that we will proceed normaly if the readonly checks are disabled (and that thus we already have the lock in exclusive mode). |
| SetField | This member acquires the lock in exclusive mode. This is probably overkill, but by doing this we ensure that we have the lock and that we will throw the exception if the readonly checks are enabled or that we will proceed normaly if the readonly checks are disabled (and that thus we already have the lock in exclusive mode). |
| GetCalculation | This member is not used in our case. |
| SetCalculation | This member is not used in our case. |
| IsFieldDefined | This member calls another one which is thread safe and the only processes its output. |
| IsFieldNotEmpty | This internal member is called only by caller which have already acquired the write lock or in cases where thread safety is not a concern. |
| InternalGetValueOrFieldCollection | This private member is only called from IsFieldNotEmpty which is itself called from a place where the write lock has been acquired or in cases where thread safety is not a concern. |
| InternalGetValue | This member acquires the write lock during its execution. |
| InternalSetValue | This member acquires the lock in exclusive mode. This is probably overkill, but by doing this we ensure that we have the lock and that we will throw the exception if the readonly checks are enabled or that we will proceed normaly if the readonly checks are disabled (and that thus we already have the lock in exclusive mode). |
| InternalGetFieldRelation | This member only calls a thread safe member and does only process its output. |
| InternalGetFieldSource | This member is not used in our case. |
| InternalGetFieldCollection | Locks the entity in write mode during the duration of the method because it might modify the internal state if the collection is not yet defined. Moreover, we return a wrapper around the collection if necessary, in order to ensure that the accesses on the collection are thread safe and acquire the lock if necessary or throws a readonly exception if necessary. This assumes that if we are in the special killer mode with a readonly entity without readonly checks, we don’t keep collections around for later usage, otherwise we will be allowed to access to the underlying collection without the thread safe/readonly wrapper. |
| CopyFieldCollection | This member should never be called in the case where we want thread safety because it can only be called when an internal collection is copied on write and we don’t allow writes. |
| GetStructuredType | Only calls a thread safe member of EntityContext |
| GetStructuredTypeProvider | This member acquires the lock in read mode to be thread safe, because it relies on mutable internal state but does not alter it. |
| GetSyntheticStructuredType | This member is not used in our case. |
| AssignEntityContext | This member acquires the lock in excusive mode because it alters the internal state of the entity. |
| DynamicGetField | This member is not used in our case. |
| DynamicSetField | This member is not used in our case. |
| OriginalValues | This member acquires the lock in shared mode because it relies on mutable internal state but it does not alter it. |
| ModifiedValues | This member acquires the lock in shared mode because it relies on mutable internal state but it does not alter it. |
| InternalGetValueStores | This member acquires the lock in shared mode because it relies on mutable internal state but it does not alter it. |
| CreateOriginalValues | This member acquires the lock in excusive mode because it alters the internal state of the entity. |
| CreateModifiedValues | This member acquires the lock in excusive mode because it alters the internal state of the entity. |
| GetOriginalValues | This member acquires the lock in excusive mode because it alters the internal state of the entity. |
| SetOriginalValues | This member acquires the lock in excusive mode because it alters the internal state of the entity. |
| GetModifiedValues | This member acquires the lock in excusive mode because it alters the internal state of the entity. |
| SetModifiedValues | This member acquires the lock in excusive mode because it alters the internal state of the entity. |
| Resolve | This member relies only on immutable data, and we assume that it is not overridden in a descendent class. |
| GenericGetValue | This member is not used in our case. |
| GenericSetValue | This member acquires the lock in exclusive mode, just to be sure. This might be overkill. |
| NotifyEventHandlers | The underlying mechanism is not used in our case. The entity is never used as an instance of IStructuredData |
| NotifyContextEventHandler | This member acquires the lock in shared mode because it relies on mutable internal state but it does not alter it. |
| AssertIsNotRead Only | This member acquires the lock in shared mode because it relies on mutable internal state but it does not alter it. |
| AssertSimpleField | This member only calls a thread safe member and processes its data. We assume that it is not overridden. |
| AssertCollectionField | This member only calls a thread safe member and processes its data. We assume that it is not overridden. |
| AttachListener | The underlying mechanism is not used in our case. The entity is never used as an instance of IStructuredData. |
| DetachListener | The underlying mechanism is not used in our case. The entity is never used as an instance of IStructuredData. |
| GetValueIds | The underlying mechanism is not used in our case. The entity is never used as an instance of IStructuredData. |
| SetValue | The underlying mechanism is not used in our case. The entity is never used as an instance of IStructuredData. |
| EnsureEventHandlers | This member is never used in our case because it is only used when the entity is viewed as an instance of IStructuredData. |
| GetValue | The underlying mechanism is not used in our case. The entity is never used as an instance of IValueStore. |
| SetValue | The underlying mechanism is not used in our case. The entity is never used as an instance of IvalueStore. |
| InternalDefineProxy | This member is not used in our case. |
| GetEntityProxy | This member will always return null because InternalDefineProxy is never used in our case. |
| UpdateDataGeneration | This member acquires the lock in excusive mode because it alters the internal state of the entity. |
| ResetDataGeneration | This member acquires the lock in excusive mode because it alters the internal state of the entity. |
| UpdateDataGenerationAnd  NotifyEntityContextAboutChanges | This member is never used. |
| SetModifiedValuesAsOriginalValues | This member is only called at a place where the exclusive lock has been acquired. |
| SetModifiedValuesAsOriginalValues | This member is only called at a place where the exclusive lock has been acquired. |
| ResetValueStores | This member is only called at a place where the exclusive lock has been acquired. |
| ResetValue | This member is only called at a place where the exclusive lock has been acquired. |
| DefineLockFunction | This member is only called at a place where the exclusive lock has been acquired. |
| LockRead | This member is in charge of acquiring the lock in shared mode. |
| LockWrite | This member is in charge of acquiring the lock in exclusive mode. |
| MustAcquireCollectionLock | This member is only called at two places where the exclusive lock has been acquired. |