Entity Object is Used in A-Board

# Introduction

In order to best decide how to realign Alignment Board with MongoDB Documents, it is necessary to know exactly how the Entity API is used.

# Class Breakdown

**RenderableBean**: Entity for type name. Uses AlignedItem which is an EntityWrapper

**AlignedItem**: Depends on EntityWrapper (is-a), sets value on entities/EntityConstants.ATTRIBUTE\_COLOR. Rendering characteristics based on EntityConstants.ATTRIBUTE\_RENDER\_METHOD.

**EntityWrapper**: Defines child/parent relationships. Uses Entity for id, type, role, owner, creation date.

**RenderableBeanCollection**: Unit Test that builds RenderableBeans.

**ConfigurableColorMapping**: Must collect entity to get/set its color value, which may be based on its Entity Type.

**RenderablesLoadWorker**: checks RenderableBean’s type to see if it is neuron fragment, for purposes of deciding if it is in or out of display.

**ABContextDataSource**: Looks at EntityWrapper classes Sample, Neuron, VolumeImage, CompartmentSet, Compartment, make MaskChanRenderableData.

**UserSettingSerializer**: uses AlignmentBoard Entity to anchor saved settings.

**LayersPanel**: uses a common root/ALIGNMENT\_BOARDS\_FOLDER folder. Makes rooted entity on this. Looks up the alignment board, using this folder (it should be a child). It makes an Alignment Board Context (EntityWrapper), to get the ancestors of the Alignment Board Context(??). Uses AlignmentBoardContext also to get the “compartment set” child. If that does not exist, it will add one of appropriate characteristics. Now launches the worker. LP uses aligned items, to find things by EntityID. Using ID-traversal. Uses the find-item just mentioned, to invalidate entities, in response to subscribed events. If stuff was invalidated, it recreates its model. This is an external-event response. Uses entity names for building a table (for table cell renderer component). Doing a lot with Entity data in its row model.

**AlignmentBoardCreator**: has a RootedEntity as part of its state. Uses entity type name for compatibility check of incoming entity. Calls EntityWrapperFactory to make the domain objects. Moves up tree looking for NeuronSeparatorPipelineResult (uses these for alignment space and resolutions). Examines AlignmentContext objects, looking for a compatible one. Calls Model Manager to create an alignment board (back through all layers). It finally launches the board by its Entity ID.

**AlignmentBoardPopulator**: This is a Drag-N-Drop target, and a NetBeans ServiceProvider. It uses AlignmentBoardContext, which is-a AlignedItem, which is-a EntityWrapper. Therefore, the AlignmentBoardContext is a candidate domain object. The drop call will hand in a list of entities-to-add. These are added as RootedEntity, to the AlignmentBoardContext.

**ABTargetedSearchDialog**: Constructed with AlignmentBoardContext. Calls SearchWorker with AlignmentBoardContext.

**SearchWorker**: Filters results based on EntityConstants types Sample and Neuron Fragment. Treats Sample and NeuronFragment results differently.

**SampleTreeModel**: Constructed with AlignmentBoardContext. Posts an alignment board change event for updating tree. Works with EntityWrapper and AlignedItem. Works at the EntityWrapper level.

**FragmentSizeSetterAndFilter**: Looks for an aligned item with an entity id. Works at level of EntityWrapper and AlignedItem.

**VolumeWritebackHandler**: writes a metadata file for volumes (separate text). Uses EntityWrapper objects to do so.

# Summary of Use Cases for Entity Above

Here are the ways I see Entity and its satellites being used:

* EntityWrapper, which is really a generalized domain object. At this level, it’s just a holder for entity’s data, which can live locally. EntityWrapper is extended by all the AlignedItem classes, which are specific to AlignmentBoard. This, and AlignedItem are the “existing domain objects”. Much of this code can remain as-is, or at most just will require a re-pointing to a different class name, to avoid confusion with Entity objects.
* Using an entity for its type, ID or name. These *might* be gleaned from target objects.
* Using the entity as an anchor point to save EntityData objects. These values should now be stored in the domain model objects.
* Traversal of Entity/EntityData based parent-child hierarchies (as in LayersPanel). This could be the most complex case to deal with. It will partly depend on how things like Sample, and NeuronSeparatorPipelineResult have been handled in the current model.
* Using entity type name to distinguish among all the many ‘flexibility-inspired’ alternatives. These used to all be Entity objects, so the choice could be made late in flight. Examples include selecting things off of menus (and setting up menus based on Entity type as context). I need to consult with the experts on how this has been changed. See AlignmentBoardPopulator and AlignmentBoardCreator.

# Challenges

How to solve these? I need to find out about how EntityWrapper is treated. I need info on what is going on with the hierarchy of objects being added to the board (Sample and Neuron Fragment, along with the Compartments, and the Reference Channels). I need to know about the original containers of Neuron Fragment and Sample (Neuron Separator Pipeline Result). I need to see how KR and David have stored all these objects in the MongoDB database.