Technical Dept/Todos for AB2Mongo

# Introduction

This document is to keep track of things “owed” to the Alignment Board conversion. Certain items are being stubbed or otherwise bypassed so that an alignment board can be opened as rapidly as possibly, and especially so that the code can be committed and brought up to date with the rest of the NG (DataBrowser) branch.

# VolumeImage

The old, Entity implementation had a class called “VolumeImage”, which implemented three interfaces. These would return a 2d , 3d, and the Mask/Chan files. The class has not yet been modeled in Mongo/NG, so it has been implemented as a bean, but it never gets filled or created at this time.

# Inner View Dialog

Under Entity-Attribute-Value, any Entity could be used to populate a properties dialog. It was a side effect of the generic implementation. However, since objects all have true domain definitions now, it is not possible (or at least has not been done, yet) to make something that generic.

To get past this immediate need, I implemented a simplistic class called “org.janelia.it.workstation.gui.alignment\_board.swing. AlignmentBoardItemDetailsDialog”. This will need to be revisited.

# Drag and Drop

There is a new paradigm/suite of classes to deal with dragging and dropping things under NG. However, I have bypassed this. The old class “AlignmentBoardEntityTransferHandler” has been deprecated.

AlignmentBoardPanel, where this is normally established, is simply not doing this anymore, per the following code:

//TODO fill in these nulls, make a new custom version of this,

// or adapt the AlignmentBoardEntityTransferHandler for this purpose.

// THIS WILL BREAK! It will be fixed prior to release. LLF

setTransferHandler(new DomainObjectTransferHandler((ImageModel<DomainObject, Reference>) null, (DomainObjectSelectionModel) null) {

@Override

public JComponent getDropTargetComponent() {

logger.warn("Exploiting unimplemented feature: Drag-and-Drop.");

return AlignmentBoardPanel.this;

}

});

# Search

“org.janelia.it.workstation.gui.dialogs.search.alignment\_board.SearchWorker” which is meant to search up new candidates for Alignment Board inclusion, is currently in an in-between state, where it will try and do a SOLr search to fill in things as Domain Objects. It is broken, and needs further tweaking.

This block of code shows that there is a Search Parameter-based receiver that will have to be refactored for this new world.

@Override

protected void hadSuccess() {

// Accept results and populate.

RootedEntityReceiver receiver = param.getReceiver();

receiver.setRootedEntities(

rootedResults,

resultsMetaData

ABTargetedSearchDialog also harkens partially back to the old class structure.

for ( BaseballCard bbc: selected ) {

logger.info("Adding entity {}.", bbc.toString());

try {

DomainModel model = DomainMgr.getDomainMgr().getModel();

final String entityTypeName = bbc.getEntity().getEntityTypeName();

String domainObjectClass = null;

if (entityTypeName.equals(EntityConstants.TYPE\_NEURON\_FRAGMENT)) {

domainObjectClass = NeuronFragment.class.getName();

}

else if (entityTypeName.equals(EntityConstants.TYPE\_SAMPLE)) {

domainObjectClass = Sample.class.getName();

}

DomainObject dobj = model.getDomainObject(domainObjectClass, bbc.getEntity().getId());

context.addDomainObject(dobj, **"63x"**);

//context.addRootedEntity( new RootedEntity( bbc.getEntity() ) );

} catch ( Exception ex ) {

logger.error(

"Failed to add entity {} to alignment board context {}.",

bbc.getEntity(),

context.getAlignmentBoard().getName()

);

}

}

}

This mention of “63x”—should be taken care of either with true dredging, or by offering a menu to the user. Further, I am cheating here, by using the old entities, and exchanging them, by ID, for new DomainObjects.

# Layer Context Menu

This class has the following code. Note the TODO. Renaming, etc., needs to be re-implemented with Domain Objects.

protected JMenuItem getRenameItem() {

JMenuItem renameItem = new JMenuItem(" Set Alias");

renameItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent actionEvent) {

final String newName = (String) JOptionPane.showInputDialog(SessionMgr.getMainFrame(), "Alias:\n", "Set Alias For "

+ alignedItemTarget.getName()+" in this Alignment Board", JOptionPane.PLAIN\_MESSAGE,

null, null, alignedItemTarget.getName());

if ((newName == null) || (newName.length() <= 0)) {

return;

}

SimpleWorker worker = new SimpleWorker() {

@Override

protected void doStuff() throws Exception {

// DomainObject dObj = RenderUtils.getObjectForItem(alignmentBoardItem);

// ModelMgr.getModelMgr().renameEntity(alignedItemTarget, newName);

}

@Override

protected void hadSuccess() {

// Updates are driven by the entity model

}

@Override

protected void hadError(Throwable error) {

SessionMgr.getSessionMgr().handleException(error);

}

};

worker.execute();

}

});

//TODO: find modern code.

// if (!ModelMgrUtils.hasWriteAccess(alignmentBoardItem.getInternalEntity())) {

// renameItem.setEnabled(false);

// }

return renameItem;

}

Here is another point of bypass. Here, we need to return a menu item for deleting things. I have stubbed this out for immediate compilation.

/\*\* TODO: This is no longer using the Entity Delete List action. But it does nothing. \*/

private JMenuItem getDeleteListItem(final List<DomainObject> domainObjects, String text) {

final Action action = new Action() { //new RemoveEntityAction(domainObjects, false, false, new Callable<Void>() {

@Override

public String getName() {

return "Dummy Delete List";

}

@Override

public void doAction() {

try {

call();

} catch (Exception ex) {

ex.printStackTrace();

}

}

//@Override

public Void call() throws Exception {

AlignmentBoardItemRemoveEvent abEvent;

if ( domainObjects.size() == 1 ) {

DomainObject nextDomainObject = domainObjects.get( 0 );

log.debug("The removed entity was an aligned item, firing alignment board event...");

abEvent = new AlignmentBoardItemRemoveEvent(

alignmentBoardContext, nextDomainObject, 0); // Q: order index required?

}

else {

abEvent = new AlignmentBoardItemRemoveEvent(

alignmentBoardContext, null, null

);

}

ModelMgr.getModelMgr().postOnEventBus(abEvent);

return null;

}

};

# LayersPanel

This has a lot of dependencies on Entity and RootedEntity. Here’s one example of a switch that might not be complete:

private void showPopupMenuImpl(final MouseEvent e, DomainObject domainObject) {

if ( domainObject == null ) {

return;

}

// Create context menu

// Need to find aligned multiSelectionItems for the current (multi) selection.

int[] rows = outline.getSelectedRows();

List<DomainObject> multiSelectionItems = new ArrayList<>();

for ( int row: rows ) {

DomainObject nextItem = getAlignedItemFromRow( row );

if ( nextItem != null ) {

multiSelectionItems.add( nextItem );

}

}

//TODO: reassess. This may not have proper child linkages, etc.

AlignmentBoardItem abi = new AlignmentBoardItem();

abi.setTarget(new Reference(domainObject.getClass().getName(), domainObject.getId()));

abi.setInclusionStatus(InclusionStatus.In.toString());

abi.setVisible(true);

final LayerContextMenu popupMenu = new LayerContextMenu(alignmentBoardContext, abi, multiSelectionItems);

popupMenu.addMenuItems();

popupMenu.show(outline, e.getX(), e.getY());

}

Here, too: we use icons, and in the past they were based on the entity. Where from now?

private class OutlineTreeCellRenderer extends DefaultOutlineCellRenderer {

public OutlineTreeCellRenderer() {

}

@Override

public Component getTableCellRendererComponent(JTable table, Object value, boolean selected,

boolean hasFocus, int row, int column) {

if (value==null) return null;

JComponent cell = (JComponent)super.getTableCellRendererComponent(

table, value, selected, hasFocus, row, column);

JLabel label = (JLabel)cell;

if (label==null) return null;

if (value instanceof AlignmentBoardItem) {

AlignmentBoardItem alignedItem = (AlignmentBoardItem)value;

DomainObject alignedItemTarget = DomainMgr.getDomainMgr().getModel().getDomainObject(alignedItem.getTarget());

if (alignedItem==null) {

label.setText("Item is null");

label.setIcon(null);

}

else {

label.setText(alignedItemTarget.getName());

//TODO workout where to get icon label.setIcon(Icons.getIcon(alignedItem));

}

}

else if (value instanceof String) {

}

else {

log.warn("Unrecognized value type in LayersPanel tree column: "+value.getClass().getName());

}

return label;

}

}

Further, we have parent loading…

@Subscribe

public void itemChanged(AlignmentBoardItemChangeEvent event) {

if (sampleTreeModel==null || alignmentBoardContext==null) return;

ChangeType change = event.getChangeType();

final OutlineExpansionState expansionState = new OutlineExpansionState(outline);

expansionState.storeExpansionState();

if (event.getDomainObject()!=null) {

log.debug("Aligned item changed {} with change type {}", event.getDomainObject().getName(), change);

}

else {

log.debug("All aligned items changed with change type {}", change);

}

// TODO: need to look at whether parent-add is needed now.

// if (change==ChangeType.Added) {

// try {

// // Recreate the contextualized wrappers

// EntityWrapper parent = event.getAlignedItem().getParent();

// if (parent==null) {

// log.error("Aligned item has null parent: "+event.getAlignedItem().getName());

// }

// else {

// parent.loadContextualizedChildren(alignmentBoardContext.getAlignmentContext());

// }

// }

// catch (Exception e) {

// SessionMgr.getSessionMgr().handleException(e);

// }

// }

// else {

// // No need to do anything for other changes, they've all been handled through other means

// }

// Generating model events is hard (we don't know the UI indexes of what was deleted, for example),

// so we just recreate the model here.

recreateModel();

expansionState.restoreExpansionState(true);

}

I need to find out how “entity invalidated” plays out in the Domain Object world.

public void entityInvalidated(EntityInvalidationEvent event) {

//TODO: re-examine this.

// if (event.isTotalInvalidation()) {

// log.debug("Total invalidation, so we're refreshing the tree");

// refresh();

// return;

// }

//

// log.debug("Some entities were invalidated, let's check if we care...");

// if (alignmentBoardContext==null) return;

//

// final OutlineExpansionState expansionState = new OutlineExpansionState(outline);

// expansionState.storeExpansionState();

//

// final Collection<AlignmentBoardItem> invalidItems = new HashSet<>();

//

// Collection<AlignmentBoardItem> invalidated = event.getInvalidatedEntities();

//

// for(AlignmentBoardItem alignmentBoardItem : invalidated) {

// AlignmentBoardItem invalidItem = findAlignedItemByEntityId(alignmentBoardContext, alignmentBoardItem.getTarget().getTargetId());

// if (invalidItem!=null) {

// invalidItems.add(invalidItem);

// }

// }

//

// log.debug("Found {} aligned items with invalidated entities",invalidItems.size());

//

// if (invalidItems.isEmpty()) return;

//

// for(AlignmentBoardItem invalidItem : invalidItems) {

// try {

// log.debug("Updating invalidated entity {} on aligned item",invalidItem.getId());

// invalidItem.updateEntity(ModelMgr.getModelMgr().getEntityById(invalidItem.getId()));

// invalidItem.loadContextualizedChildren(alignmentBoardContext.getAlignmentContext());

// }

// catch (Exception e) {

// log.error("Error updating entity {} on aligned item",invalidItem.getId());

// }

// }

//

// recreateModel();

//

// expansionState.restoreExpansionState(true);

}

Smaller problems here: 1) do I need parent for alignment board item? 2) where is the Domain Object means of saving or updating values?

/\*\*

\* The value "set" here is the checbox selected state.

\*

\* @param node model for this checkbox.

\* @param column column number for the checkbox.

\* @param value T or F here.

\*/

@Override

public void setValueFor(Object node, int column, Object value) {

if (!(node instanceof AlignmentBoardItem)) {

return;

}

final AlignmentBoardItem alignedItem = (AlignmentBoardItem)node;

// Is this possible??? If so, would NOW be the AlignmentBoard,

// and I would need to ensure all the types of things in table

// are DomainObject, instead of AlignmentBoardItem.

final Boolean isVisible = (Boolean)value;

SimpleWorker worker = new SimpleWorker() {

private AlignmentBoardItem parent;

private DomainObject parentObject;

private DomainObject target = DomainMgr.getDomainMgr().getModel().getDomainObject(alignedItem.getTarget());

@Override

protected void doStuff() throws Exception {

alignedItem.setVisible(isVisible);

Collection<DomainObject> affectedEntities = new ArrayList<>();

for(AlignmentBoardItem child : alignedItem.getChildren()) {

DomainObject innerChild = DomainMgr.getDomainMgr().getModel().getDomainObject(child.getTarget());

affectedEntities.add( innerChild );

}

// HOW to get the parent?

AlignmentBoardItem parentWrapper = null; //alignedItem.getParent();

parentObject = RenderUtils.getObjectForItem(alignedItem);

if (parentWrapper!=null) {

if (parentWrapper instanceof AlignmentBoardItem) {

parent = (AlignmentBoardItem)parentWrapper;

if ( ! isVisible ) {

// Check children of this parent: any of them

// on?

boolean childVisible = false;

for (AlignmentBoardItem child : parent.getChildren()) {

if (child.isVisible()) {

childVisible = true;

break;

}

}

if ( ! childVisible ) {

affectedEntities.add(parentObject);

}

}

else if ( ! parent.isVisible() ) {

// May have to read uncached visibility flag.

// But could non-incur whole writeback cost.

affectedEntities.add(parentObject);

}

}

}

if ( affectedEntities.size() > 0 ) {

// TODO How to save the values?

// ModelMgr.getModelMgr().setOrUpdateValues(

// affectedEntities,

// EntityConstants.ATTRIBUTE\_VISIBILITY,

// Boolean.toString(isVisible)

// );

}

}