CaArrayPanel.java

After establishing a connection, we will use the *ExperimentSearchCriteria* class to obtain a listing of ExperimentImpl objects, each of which will be wrapped in a *CaArrayExperiment* object.

1. ExperimentSearchCriteria **esc** = SearchCriteriaFactory.
   new_EXPERIMENT_EXPERIMENT_SC();
2. SearchResult **results** = **esc**.search();
3. ExperimentImpl[] **result** = (ExperimentImpl[]) **results**.getResultSet();
4. for each ExperimentImpl **exp** in **result** instantiate a wrapped object:
   CaArrayExperiment **caexp** = new CaArrayExperiment(**exp**);
   Specifically:
   caexp.e = exp;
   caexp.m_id = e.getId();
   caexp.m_name = e.getName();

When an experiment node is clicked in the experiments tree, the associated experiment into as well as the measured and derived assays are obtained and wrapped around CaArrayBioassay objects (using the stored pointer to the ExperimentImpl object):

1. **caexp**.experimentinfo = **caexp**.exp.getDescriptions()[0].getText().
2. **caexp**.measuredNum, **caexp**.derivedNum
   first get all *BioAssay*[] objects: BioAssay[] bioassays = exp.getBioAssays()
   for each BioAssay b in bioassays inspect it and increase a count:
   if (b instanceof MeasuredBioAssayImp)
   ++ **caexp**.measuredNum
   caexp.measuredAssays[i] = new CaArrayBioassay(b)
   if (b instanceof DerivedBioAssayImpl) → ++ **caexp**.derivedNum
   ++ **caexp**.derivedNum
   caexp.derivedAssays[i] = new CaArrayBioassay(b)

Each *CaArrayBioassay* object captured the following info:

1. BioAssay **ba**: reference to the source BioAssay object.
2. String **m_id = ba**.getIdentifier().
3. int **dataCount** // not user anywhere
4. BioAssayData[] **baData** // not used anywhere

Then, when a bunch of Array nodes are selected in the tree, a new *CaArrayResource* object is created for each selected CaArrayBioassay, containing a reference to the source BioAssay as well as to the containing *CaArrayExperiment* object.

ProjectPanel.java

At the next step we will go over each BioAssay in the CaArrayExperiment, retrieve the BioDataCube associated with it and extract the values for the QuantitationType "Affymetrix:QuantitationType:CHPSignal" (for derived assays) or

“Affymetrix:QuantitationType:CELIntensity” (for measured assays) for all derived and measured arrays linked to the BioAssay.

The method
  remoteFileOpenAction(CaArrayResource[] mRes)
is invoked to create a CSMicroarraySet containing the remote arrays described by each entry in mRes[]. Specifically, the following method is called for each BioAssay r = mRes[i].bioAssay:
  DSMicroarray CaARRAYParser. getMicroarray (int ser, BioAssay r, CSExprMicroarraySet maSet)

which will read in BioAssay r as the ser-th array in maSet. The following sequence of events take place for each **dba** DerivedBioAssayImpl (and a similar sequence for each mba MeasuredBioAssayImpl)

1. DerivedBioAssayData[] **dbd** = **dba**. getDerivedBioAssayData()
2. for each DerivedBioAssayData **dbad** = **dbd**[i]
    a. Get the quantitation types for **dbad**: QuantitationType[] qtypes = dbdd.getQuantitationTypeDimension().getQuantitationTypes()
    b. Count the design elements (aka, markers) in **dbad**:
       DesignElementDimension **ded** = **dbad**.getDesignElementDimension()
       DesignElement[] **de** = (ReporterDimension) ded.getReporters() or
                = (FeatureDimensionImpl) ded. getContainedFeatures() or
       = ((CompositeSequenceDimensionImpl) ded).getCompositeSequences()
    c. Get the associated DataCube:
       BioDataValues **bdv** = **dbad**.getBioDataValues()
    d. Cast **bdv** to an obect of type BioDataCubeImpl and proceed to read the [ser][marker][quantType] entries:
       (Double) cube[ser][marker][quantType]