Validity controls

Validity controls

The GInterval class

GENIUS gives the possibility to manage validity intervals (consistent with **SIRIUS** requirements):

- Only for GEntryReal, GEntryInt, GEntryRealVector et GEntryIntVector widgets
- Possibility to get an error and/or warning information
- For real values, these validity controls of course take into account units management

To do it, we will have to instantiate a GInterval object. The example below shows how to create an [0.,100.] validity interval.

```
GInterval myInterval = new GInterval(0., 100.);
```

It is possible to precise if the interval is opened or closed on each side (by default it is inclusive): below, we define the validity interval as [0., 100.[.

```
GInterval myInterval = new GInterval(0., 100., GInterval.Rule.INCLUSIVE,
GInterval.Rule.EXCLUSIVE);
```

At last, it is possible to add an intermediate "warning" interval meaning that the value is OK but to be careful with it:

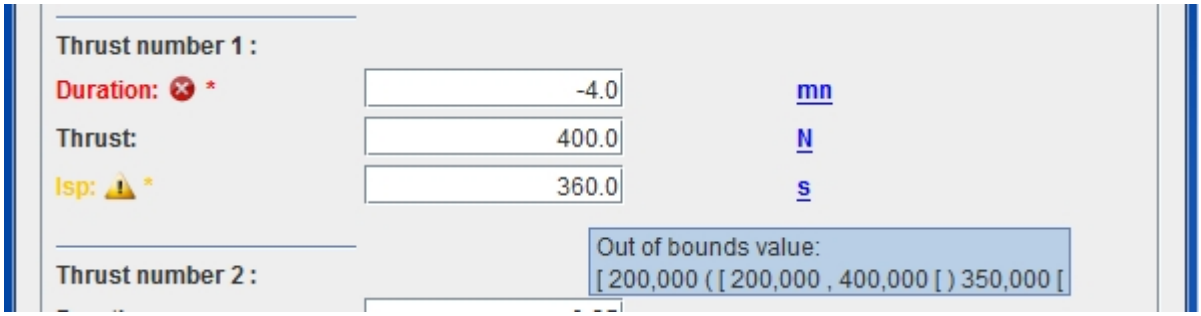
```
GInterval myInterval = new GInterval(10.,  90., GInterval.Rule.EXCLUSIVE,
GInterval.Rule.EXCLUSIVE,
                                     0., 100., GInterval.Rule.INCLUSIVE,
GInterval.Rule.INCLUSIVE);
```

With that example, the data will be OK if included in the [0.,100.] interval but a warning message will appear if the data will be comprised in the [0.,10.[or]90.,100.] intervals.

Note that we can use the infinity definitions given by the Java language to define, for example the [0,∞[interval:

```
GInterval myInterval = new GInterval(0., Double.POSITIVE_INFINITY);
```

At last, in case of an data out of the validity interval (or the warning one), a tooltip is available when the mouse will stay above the input field (see below).



The corresponding code is the following one:

```
GUnit[] unitDuration = {new GMetricUnit("mn"), new GMetricUnit("s")};
GUnit[] unitThrust = {new GMetricUnit("N")};
GUnit[] unitIsp = {new GMetricUnit("s")};

// Error control validity
durationIhm =  new GEntryReal("Duration:", val1, unitDuration);
durationIhm.addInterval( new GInterval(0., Double.POSITIVE_INFINITY) );

// No validity control
thrustIhm =  new GEntryReal("Thrust:", val2, unitThrust);

// Error and warning control validity
// Error if ]-Inf,200[ or[400,+Inf[
// Warning if [200,250[ or[350,400[
// OK if[250,350[
ispIhm    =  new GEntryReal("Isp:", val3, unitIsp);
ispIhm.addInterval( new GInterval(250., 350., GInterval.Rule.INCLUSIVE,
GInterval.Rule.EXCLUSIVE,
                                  200., 400., GInterval.Rule.INCLUSIVE,
GInterval.Rule.EXCLUSIVE) );
```

Status management

This validity interval check is not only a graphical one. Indeed, we may also get the status of a widget using the getStatus() method. This method will return a GStatus enumeration : GStatus.OK, GStatus.WARN or GStatus.ERR.

Moreover, from 1.4 version, it is now possible to use a mechanism allowing to globally manage the whole data status. This mechanism uses the GCondensedStatusInterface as explained below.

First, the widgets in which we want to test data validity must implement this interface and thus we have to define the updateCondensedStatus method where we will check the data to be considered:

```
public class TestForCondensedStatusWidget extends GPanel implements
GCondensedStatusInterface {

    private final GEntryReal valAngle;
    private final GEntryReal valDist;
```

```
    public final GInterval angleInterval =
        new GInterval(30., 60., GInterval.Rule.INCLUSIVE,
GInterval.Rule.EXCLUSIVE,
                      0., 90., GInterval.Rule.INCLUSIVE,
GInterval.Rule.EXCLUSIVE);
    public final GInterval distInterval =
        new GInterval(100.e+3, 1000.e+3, GInterval.Rule.INCLUSIVE,
GInterval.Rule.INCLUSIVE,
                       0.,   36000.e+3, GInterval.Rule.INCLUSIVE,
GInterval.Rule.INCLUSIVE);

    public TestForCondensedStatusWidget() throws GIntervalException {
        valAngle = new GEntryReal("Angle entry", 45.);
        valAngle.addGInterval(angleInterval);
        valDist = new GEntryReal("Distance entry", 500.e+3);
        valDist.addGInterval(distInterval);
    }

...

    @Override
    public void updateCondensedStatus(GCondensedStatus arg0) {
        // valAngle and valDist are checked
        arg0.update(valAngle, valDist);
    }

}
```

Note that none of the widgets must correspond to a null address otherwise an exception will be throwed!

Then, to test the global status (for example just before launching an execution), we only have to write something like this:

```
    TestForCondensedStatusWidget widget = new TestForCondensedStatusWidget();

    ...

    GCondensedStatus status = new GCondensedStatus(widget);

    // We print the global status ...
    System.out.println("Global status: "+status.getStatus());

    // We print the list of data with an ERROR status ...
    for (int i = 0; i < status.getErrorComponentList().size(); i++) {
        System.out.println("Error on
"+status.getErrorComponentList().get(i).getNameInConfigFile());
        System.out.println("Error on
"+status.getErrorComponentList().get(i).getPathInConfigFile()); // Sinve V1.8
```

```
    }

    // We print the list of data with a WARNING status ...
    for (int i = 0; i < status.getWarningComponentList().size(); i++) {
        System.out.println("Warning on
"+status.getWarningComponentList().get(i).getNameInConfigFile());
        System.out.println("Warning on
"+status.getWarningComponentList().get(i).getPathInConfigFile()); // Sinve
V1.8

    }
```

Complex status management

Since V1.8, using the setForcedStatus() method, it is now possible to manage more complex status, for example above several entries.

Let us imagine, the following test : we have three real entries whose values must be included between 0 and 1:

```
    GEntryReal[] vals = new GEntryReal[3];
    vals[0] = new GEntryReal("Real entry 1:", 0.);
    vals[0].addGInterval(new GInterval(0., 1.));
    vals[1] = new GEntryReal("Real entry 2:", 0.);
    vals[1].addGInterval(new GInterval(0., 1.));
    vals[2] = new GEntryReal("Real entry 3:", 0.);
    vals[2].addGInterval(new GInterval(0., 1.));
```

... but their sum must also be equal to 1! To do it, we will just have to call for the following example method, using the setForcedStatus() method to force the status between these values:

- GStatus.ERROR
- GStatus.WARNING
- GStatus.OK
- null

Depending on this status, the individual ones will be overwritten. Setting null will go back to the classical way. Note also that it is mandatoty to call for the updateStatus() method after it. Then, it is also possible to change the message as it is shown below.

```
private void testValues () {
    double sum = 0.;
    for (int i = 0; i < vals.length; i++) {
        sum = sum + vals[i].getValue();
    }
    if ( sum != 1. ) {
        System.out.println("Not allowed values: sum equals to "+sum);
        for (int i = 0; i < vals.length; i++) {
```

```
            vals[i].setForcedStatus(GStatus.ERROR);
            vals[i].updateStatus();
            vals[i].getGTextField().getJTextField().setToolTipText("Not
allowed value: sum equals to "+sum);
        }
    } else {
        for (int i = 0; i < vals.length; i++) {
            vals[i].setForcedStatus(null);
            vals[i].updateStatus();
        }
    }
}
```

Return to the introduction ↑ Go to the next page →

Menu de navigation

GENIUS

- Welcome
- Quick Start
- News

Basic principles

- GFrame and GPanel
- Main widgets
- Links with Swing
- GLayout
- Conditional Display
- GListener interface

More deeper in the concept

- Units management
- GContainer
- GReadWrite interface and data files management
- Modified data
- Process management

Still more ...

- Validity controls
- Menu bar
- Icons
- GClear interface

Still more again ...

- Tooltips
- Shortcuts
- Copy & paste
- Plots
- Results File Management
- GPlotPanel
- GGroundPlotPanel
- Internationalization
- Log file
- Update data

Some other widgets

- GTabbedPane

- GTable1D
- GTable2D
- GComponentList
- GDialog and GDetachedPanel
- GContextFileManagement
- How to build a standard application
- GPanTest
- Create your own widget

Evolutions

- Main differences between V1.11.4 and V1.12.1
- Main differences between V1.10.1 and V1.11.4
- Main differences between V1.10 and V1.10.1
- Main differences between V1.9.1 and V1.10
- Main differences between V1.9 and V1.9.1
- Main differences between V1.8 and V1.9
- Main differences between V1.7 and V1.8
- Main differences between V1.6.2 and V1.7
- Main differences between V1.6.1 and V1.6.2
- Main differences between V1.6 and V1.6.1
- Main differences between V1.5 and V1.6
- Main differences between V1.4.1 and V1.5
- Main differences between V1.3 and V1.4.1

Training

- Training slides
- Tutorials package for V1.12.1
- Tutorials package for V1.11.4
- Tutorials package for V1.10.1
- Tutorials package for V1.9.1
- Training & tutorials package for V1.8
- Training & tutorials package for V1.7
- Training & tutorials package for V1.6

Links

- CNES freeware server

Outils

- Pages liées
- Suivi des pages liées
- Pages spéciales
- Adresse de cette version
- Information sur la page

- Citer cette page

- Dernière modification de cette page le 6 mai 2019 à 13:21.

- Politique de confidentialité
- À propos de Wiki
- Avertissements
-