MOST Web Documentation

Release 1.0.1

Healthcare Flows - CRS4

May 02, 2017

Contents

The MOST project aims to achieve an open, modular and scalable solution for the creation, execution and management of remote clinical consultations with direct interaction between specialists.

The project consists of a set of frameworks that deal with different aspects and technologies useful for the creation of telemedicine applications.

CHAPTER 1

Report Framework

The Report Framework allows you to load, display and edit clinical records in your mobile applications, automatically built by json structures representing specific OpenEHR Archetypes.

TREE:

Most report Dependences

The Most-Report Freamework depends on the SuperTooltips Lib (https://github.com/nhaarman/supertooltips/) .

Basic Concepts

The Most EhrLib allow you to build, display and edit Open EHR based archetypes (http://www.openehr.org/) into your android applications. The lib uses specific json schemas for describing the structure, the ontology and the content of each archetype.

For example, assuming you want to build the well known Blood Pressure archetype, at least the following json schemas must be provided:

- *adl_structure__blood_pressure.json*: it contains the structure of the archetype, corresponding to the ADL structure defined by the Open EHR standard
- *datatypes__blood_pressure.json*: it contains the description of all datatypes used by this archetype (e.g: DV_QUANTITY, DV_CLUSTER and so on)
- *ontology__blood_pressure.json*: it contains the ontology of this archetype (i.e a textual title and description of each item of the archetype)

Note that for representing *any* archetype, the previous json schemas are mandatory. Optionally, if you need to customize the visual layout of the archetype, you can also specify a layout json schema, *layout__blood_pressure.json*, as explained in the next section.

Basic Example: how to build a single archetype: Blood Pressure

This section explains how to build, step by step, a Blood Pressure archetype and how to display it in an Android Application

First of all you need to build the json schemas. Let start with the *adl_structure__blood_pressure.json*:

```
{
 "archetype_class": "openEHR-EHR-OBSERVATION.blood_pressure.v1",
 "archetype_details": {
     "data": {
         "at0001": [
             {
                 "events": [
                     {
                         "at0006": {
                             "data": {
                                 "at0003": [
                                     {
                                         "items": {
                                             "at0004": "DATA_NODE::DV_QUANTITY",
                                             "at0005": "DATA_NODE::DV_QUANTITY",
                                             "at1006": "DATA_NODE::DV_QUANTITY",
                                             "at1007": "DATA_NODE::DV_QUANTITY",
                                             "at0033": "DATA_NODE::DV_TEXT"
                                         }
                                     }
                                 ]
                             },
                             "state": {
                                 "at0007": [
                                     {
                                         "items": {
                                             "at0008": "DATA_NODE::DV_CODED_TEXT",
                                             "at1052": "DATA_NODE::DV_TEXT"
                                         },
                                         "at1030": "ARCHETYPE_NODE",
                                         "at1043": "DATA_NODE::DV_CODED_TEXT",
                                         "at1005": "DATA_NODE::DV_QUANTITY"
                                     }
                                 ]
                             }
                         },
                         "at1042": "DATA_NODE::MATH_FUNCTION"
                     }
                 ]
             }
         ]
     }
 }
}
```

Note that the structure is very similar to the corresponding ADL structure defined by the Open EHR adl structure (you can find the adl structure of the Blood Pressure Archetype here: http://openehr.org/ckm/). Each item of the structure has a unique identifier (such as "at0004") that represents a specific item of the archetype. Each item is of a specific type (for example, the item at0004 is a DV_QUANTITY item).

The datatype description of each item is specified in the *datatypes__blood_pressure.json* schema:

```
{
 "title": "at0000",
 "datatypes": {
     "data": {
         "at0004": {
             "path": "data[at0001]/events[at0006]/data[at0003]/items[at0004]",
             "type": "DV_QUANTITY",
             "attributes": {
                 "unit_of_measure": "mm[Hg]",
                 "precision": 2,
                 "range": {
                     "min": 10,
                     "max": 180
                 }
             }
         },
         "at0005": {
             "path": "data[at0001]/events[at0006]/data[at0003]/items[at0005]",
             "type": "DV_QUANTITY",
             "attributes": {
                 "unit_of_measure": "mm[Hg]",
                 "precision": 2,
                 "range": {
                     "min": 10,
                     "max": 180
                 }
             }
         },
         "at1006": {
             "path": "data[at0001]/events[at0006]/data[at0003]/items[at1006]",
             "type": "DV_QUANTITY",
             "attributes": {
                 "unit_of_measure": "mm[Hg]",
                 "precision": 2,
                 "range": {
                     "min": 10,
                     "max": 180
                 }
             }
         },
         "at1007": {
             "path": "data[at0001]/events[at0006]/data[at0003]/items[at1007]",
             "type": "DV_QUANTITY",
             "attributes": {
                 "unit_of_measure": "mm[Hg]",
                 "precision": 2,
                 "range": {
                     "min": 10,
                     "max": 180
                 }
             }
         },
         "at0033": {
             "path": "data[at0001]/events[at0006]/data[at0003]/items[at0033]",
             "type": "DV_TEXT",
             "attributes": {}
         }
     }
```

```
    }
}
```

All datatypes are specified into the *datatypes* json dictionary. In the example above you defined the *data* section of the archetype, so that, there is the corresponding *data* json dictionary containing the data type description of all items contained inside it. For example, there is the *item0004* with the following informations:

```
"at0004": {
            "path": "data[at0001]/events[at0006]/data[at0003]/items[at0004]",
            "type": "DV_QUANTITY",
            "attributes": {
                "unit_of_measure": "mm[Hg]",
                "precision": 2,
                "range": {
                    "min": 10,
                    "max": 180
                }
        }
    }
```

The item *at0004* is a *DV_QUANTITY* item, located inside the ADL structure of the archetype at the absolute path *data[at0001]/events[at0006]/data[at0003]/items[at0004]*. Note that each datatype has specific attributes that can be specified. In this example, the DV_QUANTITY item has its unit of measure, precision and a range of allowed numeric values. Again, see the Open EHR reference for getting more informations about Open EHR data types.

The description of each item is specified by the *ontology__blood_pressure.json* schema:

```
{
 "en":{
   "at0000":{
      "text":"Blood Pressure",
      "description":"The local measurement of arterial blood pressure which is a␣
↪surrogate for arterial. pressure in the systemic circulation. Most commonly, use of␣
↪the term 'blood pressure' refers to measurement of brachial artery pressure in the␣
↪upper arm."
   },
   "at0001":{
      "text":"history",
      "description":"History Structural node."
   },
   "at0003":{
      "text":"blood pressure",
      "description":"internal"
   },
   "at0004":{
      "text":"Systolic",
      "description":"Peak systemic arterial blood pressure  - measured in systolic or␣
↪contraction phase of the heart cycle."
   },
   "at0005":{
      "text":"Diastolic",
      "description":"Minimum systemic arterial blood pressure - measured in the␣
↪diastolic or relaxation phase of the heart cycle."
   },
   "at0006":{
      "text":"any event",
      "description":"Default event."
   },
```

```
   "at0007":{
      "text":"state structure",
      "description":"internal"
   },
   "at0008":{
      "text":"Position",
      "description":"The position of the subject at the time of measurement."
   },
   "at0011":{
      "text":"Tree",
      "description":"List Structure"
   },
   "at0013":{
      "text":"Cuff Size",
      "description":"The size of the cuff used for blood pressure measurement."
   },
   "at0033":{
      "text":"Comment",
      "description":"Comment on blood pressure measurement."
   },
   "at1006":{
      "text":"Mean Arterial Pressure",
      "description":"The average arterial pressure that occurs over the entire course␣
↪of the heart contraction and relaxation cycle."
   },
   "at1007":{
      "text":"Pulse Pressure",
      "description":"The difference between the systolic and diastolic pressure."
   },
   "at0033":{
      "text":"Comment",
      "description":"Comment on blood pressure measurement."
   },
   "at1000":{
      "text":"Standing",
      "description":"Standing at the time of blood pressure measurement."
   },
   "at1001":{
      "text":"Sitting",
      "description":"Sitting (for example on bed or chair) at the time of blood␣
↪pressure measurement."
   },
   "at1002":{
      "text":"Reclining",
      "description":"Reclining at the time of blood pressure measurement."
   },
   "at1003":{
      "text":"Lying",
      "description":"Lying flat at the time of blood pressure measurement."
   },
   "at1014":{
      "text":"Lying with tilt to left",
      "description":"Lying flat with some lateral tilt, usually angled towards the␣
↪left side.   Commonly required in the last trimester of pregnancy to relieve␣
↪aortocaval compression."
   },
   "at1052":{
      "text":"Confounding factors",
```

```
      "description":"Comment on and record other incidental factors that may be␣
↪contributing to the blood pressure measurement.  For example, level of anxiety or
↪'white coat syndrome'; pain or fever; changes in atmospheric pressure etc."
   },

   "at1025":{
      "text":"Blood Pressure",
      "description":"Included archetype, just for testing"
   }

},

"es-ar":{
   "at0000":{
      "text":"Presión Arterial",
      "description":"La medición local de la tensión arterial que deriva de la medida␣
↪de la presión arterial en la circulación sistémica. Comúnmente el uso de 'presión␣
↪arterial' se refiere a la medida de la presión de la arteria braquial por encima␣
↪del pliegue del codo."
   },
   "at0001":{
      "text":"historia",
      "description":"historia Nodo estructural"
   },
   "at0003":{
      "text":"blood pressure",
      "description":"internal"
   },
   "at0004":{
      "text":"Sistólica",
      "description":"Presión arterial sistólica pico - medido en sístole o la fase de␣
↪contracción del ciclo cardíaco."
   },
   "at0005":{
      "text":"Diástole",
      "description":"Presión arterial sistémica mínima - medido durante la diástole o␣
↪fase de relajación del ciclo cardíaco."
   },
   "at0006":{
      "text":"cualquier evento",
      "description":"Evento por defecto."
   },
   "at0007":{
      "text":"state structure",
      "description":"internal"
   },
   "at0008":{
      "text":"Posición",
      "description":"La posición del individuo en el momento del registro."
   },
   "at0011":{
      "text":"estructura de lista",
      "description":"estructura tipo lista"
   },
   "at0013":{
      "text":"Tamaño del manguito",
      "description":"El tamaño del manguito usado para la toma de la presión arterial.
↪"
```

```
   },
   "at0033":{
      "text":"Comment",
      "description":"Comment on blood pressure measurement."
   },
   "at1006":{
      "text":"Presión Arterial Media",
      "description":"La presión arterial promedio que ocurre durante el ciclo entero␣
↪de la contracción y relajación del corazon."
   },
   "at1007":{
      "text":"Presión de Pulso",
      "description":"La diferencia entre la presión sistólica y la presión diastólica.
↪"
   },
   "at0033":{
      "text":"Comentario",
      "description":"Comentario sobre la medición de la presión sanguínea"
   },
   "at1000":{
      "text":"De pie",
      "description":"De pie al momento de la medición de la tensión arterial."
   },
   "at1001":{
      "text":"Sentado",
      "description":"Sentado (en la cama o en una silla) durante el registro de la␣
↪presión arterial."
   },
   "at1002":{
      "text":"Reclinado",
      "description":"Reclinado (semisentado) durante el registro de la presión␣
↪arterial."
   },
   "at1003":{
      "text":"Acostado",
      "description":"Acostado horizontal durante la medición de la presión arterial"
   },
   "at1014":{
      "text":"Acostado e inclinado levemente sobre su costado izquierdo",
      "description":"Acostado horizontal e inclinado levemente sobre su costado␣
↪izquierdo. Comúnmente se requiere durante el último trimestre del embarazo para␣
↪aliviar la compresión aortocava."
   },
   "at1052":{
      "text":"Factores confluentes",
      "description":"Comentario y registro sobre otros factores que pueden incidir␣
↪sobre la medición de la presión arterial. Por ejemplo: nivel de ansiedad o \
↪"síndrome del guardapolvo blanco\"; dolor o fiebre; cambios en la presión␣
↪atmosférica etc."
   },

   "at1025":{
      "text":"Presión Arterial",
      "description":"Included archetype, just for testing"
   }
 }
}
```

In this example, you are handling the English and the Spanish language. For each item, a textual label and a short description is provided for both languages.

Finally, you can optionally provide a *layout__blood_pressure.json* schema (for example if you want to display the item of the systolic pressure before the item of the diastolic pressure)

```
{
"sections":[
   "data"
],
"items":{
   "data[at0001]/events[at0006]/data[at0003]/items[at0004]":{
      "priority":1
   },
   "data[at0001]/events[at0006]/data[at0003]/items[at0005]":{
      "priority":2
   },
   "data[at0001]/events[at0006]/data[at0003]/items[at1006]":{
      "priority":3
   },
   "data[at0001]/events[at0006]/data[at0003]/items[at1007]":{
      "priority":4
   },
   "data[at0001]/events[at0006]/data[at0003]/items[at0033]":{
      "priority":5
   },
   "data[at0001]/events[at0006]/state[at0007]/items[at0008]":{
      "priority":1,
      "widget":"it.crs4.ehrlib.widgets.DvCodedTextAsListWidget"
   },
   "data[at0001]/events[at0006]/state[at0007]/items[at1052]":{
      "priority":2
   }
}
}
```

In this example, you use the *priority* attribute for specifying the display order of each item (items with lower priority are displayed before). If you want, you can also display an item with a custom widget, by specifying the java class representing that datatype in the *widget* attribute. Note that you don't have to specify all items of the datatype, because all items of the datatypes will be rendered anyway. A layout specifies only the displaying order, not a sub set of items to be displayed. Note that you can display a sub set of an archetype item by providing a list of items to be excluded by using a TemplateProvider, as explained later in this guide. Finally, if you prefer, you can use *aliases* for referring to each item in a more human-readable way, as follows:

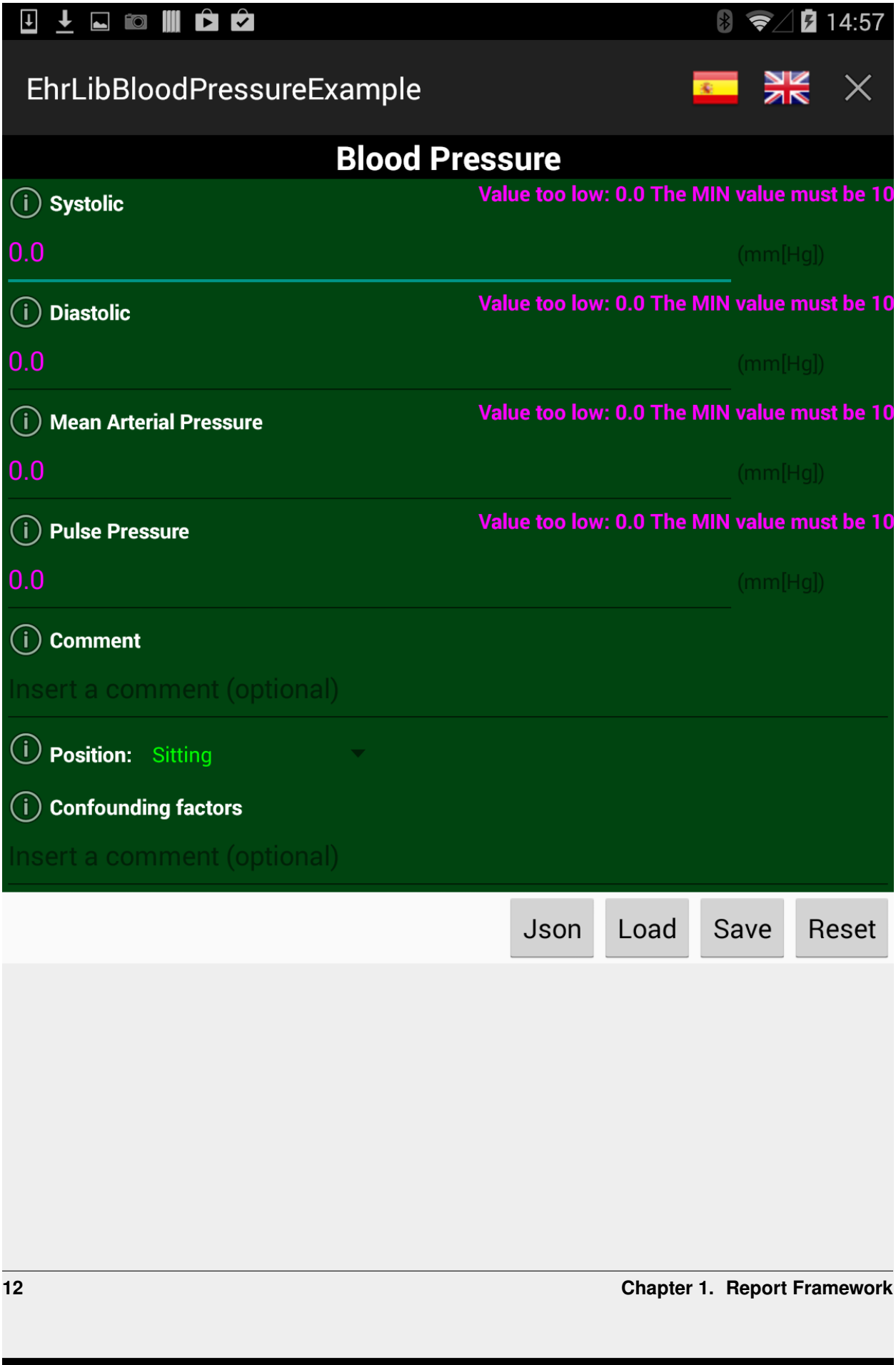
```
{
"sections":[
   "data"
],

"aliases" :
  {
  "data[at0001]/events[at0006]/data[at0003]/items[at0004]": "Systolic",
  "data[at0001]/events[at0006]/data[at0003]/items[at0005]": "Diastolic",
  "data[at0001]/events[at0006]/data[at0003]/items[at1006]": "Arterial Pressure",
  "data[at0001]/events[at0006]/data[at0003]/items[at1007]": "Pulse Pressure"
  },
"items":{
   "Systolic":{
```

```
        "priority":1
    },
    "Diastolic":{
        "priority":2
    },
    "Arterial Pressure":{
        "priority":3
    },
    "Pulse Pressure":{
        "priority":4
    },
    "data[at0001]/events[at0006]/data[at0003]/items[at0033]":{
        "priority":5
    },
    "data[at0001]/events[at0006]/state[at0007]/items[at0008]":{
        "priority":1,
        "widget":"it.crs4.ehrlib.widgets.DvCodedTextAsListWidget"
    },
    "data[at0001]/events[at0006]/state[at0007]/items[at1052]":{
        "priority":2
    }
  }
}
```

Now that you have defined all json schema, you can instance a *WidgetProvider*, the library class that uses the json schemas for building the corresponding archetype. A simple way for getting an Android View containing the Blood Pressure archetype could be the following:

```
Context xtx = getActivity();
WidgetProvider  widgetProvider = new WidgetProvider(ctx,
                                    WidgetProvider.parseFileToString(ctx,"datatypes__
↪blood_pressure.json"),       // datatypes schema
                                    WidgetProvider.parseFileToString(ctx,"ontology__
↪blood_pressure.json"),        // ontology schema
                                    WidgetProvider.parseFileToString(ctx,"adl_
↪structure__blood_pressure.json"),    // adl structure schema
                                    WidgetProvider.parseFileToString(ctx,"layout__
↪blood_pressure.json"),          // layout schema
                                    "en");
↪                               // default ontology language

// build the Archetype, according to the json schemas
FormContainer formContainer = widgetProvider.buildFormView(0);

// Retrieve the ViewGroup of the form, so it can be added to the Activity context
ViewGroup rootView = formContainer.getLayout();
```

The code above assumes that you saved all your json files into the **assets** folder of your example. The final visual result inside an Android Activity could be similar to the following (note that you can found the complete source code of this example in the *examples* folder of the repository):


14:57
EhrLibBloodPressureExample
Blood Pressure
Systolic
Value too low: 0.0 The MIN value must be 10
0.0
(mm[Hg])
Diastolic
Value too low: 0.0 The MIN value must be 10
0.0
(mm[Hg])
Mean Arterial Pressure
Value too low: 0.0 The MIN value must be 10
0.0
(mm[Hg])
Pulse Pressure
Value too low: 0.0 The MIN value must be 10
0.0
(mm[Hg])
Comment
Insert a comment (optional)
Position: Sitting
Confounding factors
Insert a comment (optional)
Json
Load
Save
Reset

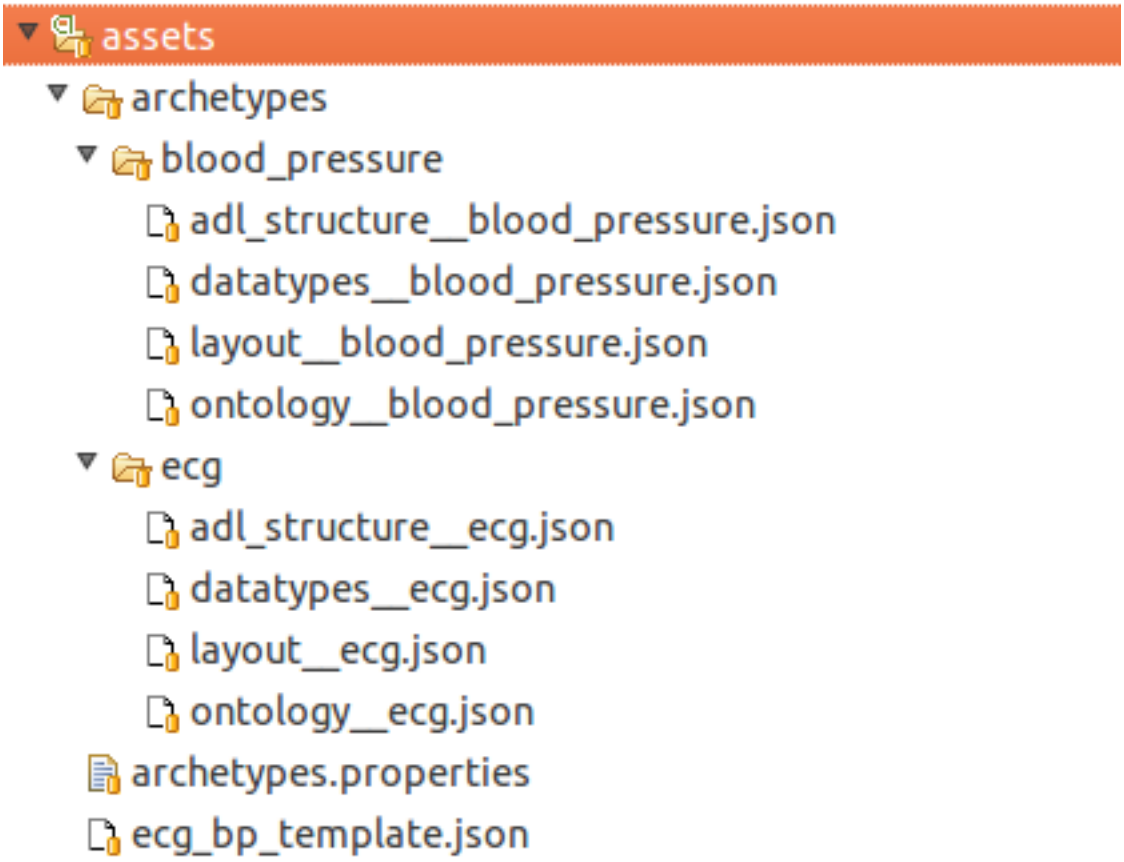
Advanced Example: how to display more than one archetype in an Activity: introduction to the TemplateProvider and the ArchetypeSchemaProvider

If you need to build and display more than one archetype in your activity, the most convenient way is to use the *TemplateProvider* and the *ArchetypeSchemaProvider* classes. This section will explain, by an easy example, how to include in the same activity two archetypes by using a Template.

Let assume you want to build a Template including the following two Open EHR archetypes:

- *openEHR-EHR-OBSERVATION.blood_pressure.v1* (the blood pressure archetype already used in the previous example)
- *openEHR-EHR-OBSERVATION.ecg.v1* (the ECG Archetype)

First of all, you have to create, inside the *assets* folder, a folder that will contain all the json schemas for both the archetypes. For instance, you can create a folder called *archetypes* and, inside of it, two other folders, called *blood_pressure* and *ecg*, containing all the json schema files related, respectively, to the blood pressure and to the ECG archetype (see the pitcure below)



Note that, in general, if you put the json schema files of a generic archetype into a folder called XXX, the name of each json file r

- adl_structure__XXX.json (mandatory, the json file containing the adl structure of the archetype)
- datatypes__XXX.json (mandatory, the json file containing the datatypes of the archetype)
- layout__XXX.json (optional, the json file containing the layout of the archetype)
- ontology__XXX.json (mandatory, the json file containing the ontology of the archetype)

These naming rules are needed if you intend to build the archetypes by using the *ArchetypeSchemaProvider* , an utility class of the Most Report Library that automatically retrieves the content of all the json schemas for each archetype.

Now you have to create a properties file (called *archetypes.properties* in this example) containing a mapping between the name of each archetype (in this example, *openEHR-EHR-OBSERVATION.blood_pressure.v1* and *openEHR-EHR-OBSERVATION.ecg.v1*) and the corresponding name of the folder containing its json schema files. The content of the *archetypes.properties* of this example is the followiing:

```
openEHR-EHR-OBSERVATION.blood_pressure.v1=blood_pressure
openEHR-EHR-OBSERVATION.ecg.v1=ecg
```

Finally, you have to define the Json file containing the structure of the visual template (in this example, *ecg_bp_template.json*) that includes an ordered list of the archetypes that will be built and rendered in the activity:

```
{
"id":"my_template_id",
"name":"ECG & BLOOD PRESSURE",
"definition":[

   {
      "archetype_class":"openEHR-EHR-OBSERVATION.blood_pressure.v1",
      "exclude":["at0004", "at0005"]
   }
   ,

   {
      "archetype_class":"openEHR-EHR-OBSERVATION.ecg.v1",
      "exclude":[]
   }
   ,
 ]
}
```

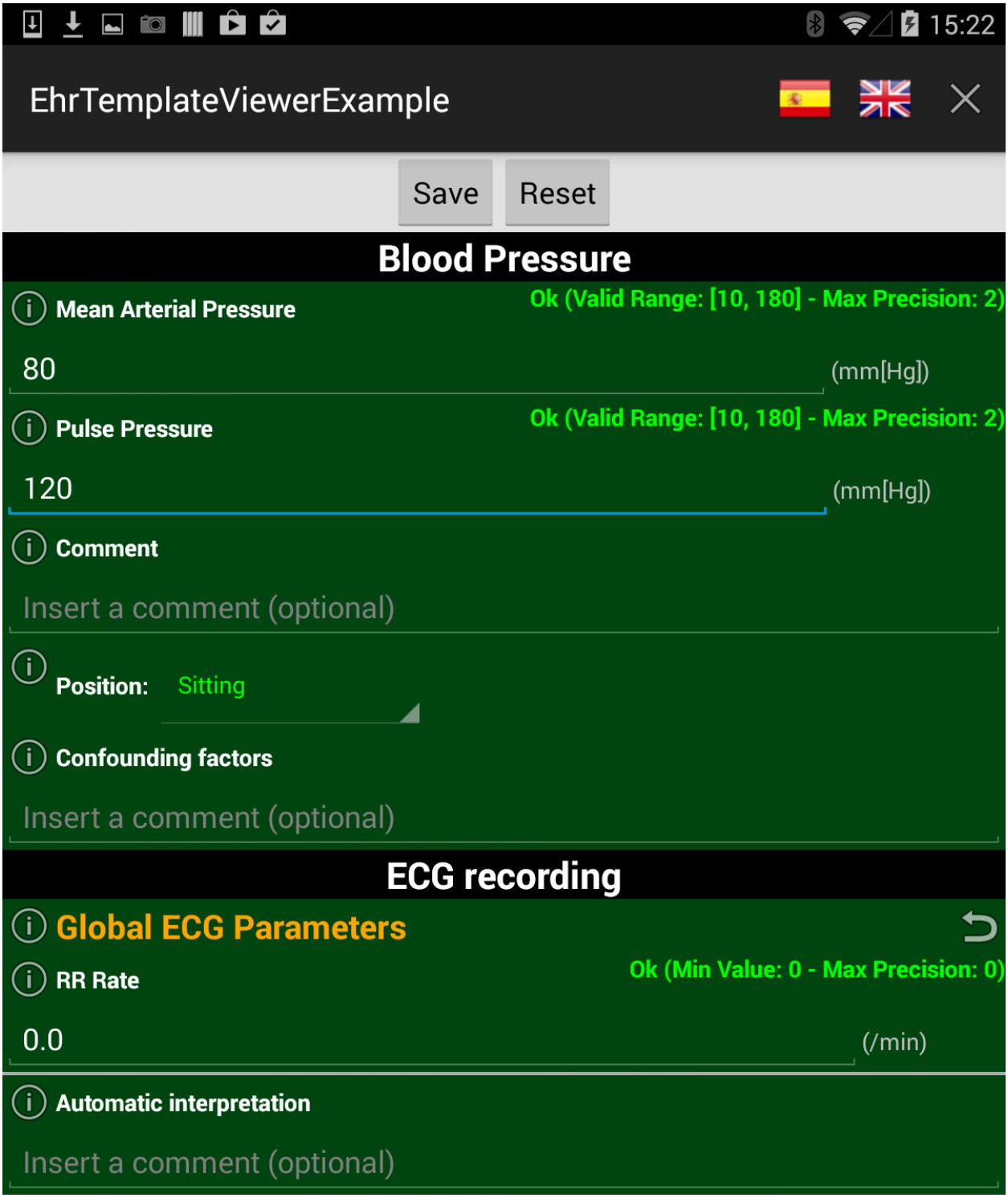
Into the section *definition* there is the ordered list of all the archetypes to be rendered, Blood Pressure and ECG. Note that the Blood Pressure definition also contains two items (the systolic pressure "at0004" , and the diastolic pressure "at0005") that will not be displayed into the activity, because specified into the *"exclude"* json array.

At this point, you are ready to create the visual template for the Android Activity:

```
Context ctx = getApplicationContext();

// build the ArchetypeSchemaProvider passing to the constructor the archetypes.
↪properties file name and the root folder name containing all the json schema␣
↪folders.
ArchetypeSchemaProvider asp = new ArchetypeSchemaProvider(ctx, "archetypes.properties
↪", "archetypes");

// Build the template, passing to the constructor the json schema describing the␣
↪template, the jsut created ArchetypeSchemaProvider and the defualt ontoly language
TemplateProvider tp = new TemplateProvider(ctx,WidgetProvider.parseFileToString(ctx,
↪"ecg_bp_template.json"), asp, "en");

// Retrieve the ordered list of widget providers, the first one related to the Blood␣
↪Pressure archetype, the second one to the ECG archetype
List<WidgetProvider> wps = tp.getWidgetProviders();
```

Now you can access to any of the WidgetProvider of the template. So, as you have already seen in the previous example, you can retrieve the ViewGroup of each archetype and add it to your activity layout. The final visual result

could be similar to the following (again, you can found the complete source code of this example in the *examples* folder of the Most Report repository):


15:22
EhrTemplateViewerExample
Save
Reset
Blood Pressure
Mean Arterial Pressure
Ok (Valid Range: [10, 180] - Max Precision: 2)
80
(mm[Hg])
Pulse Pressure
Ok (Valid Range: [10, 180] - Max Precision: 2)
120
(mm[Hg])
Comment
Insert a comment (optional)
Position:
Sitting
Confounding factors
Insert a comment (optional)
ECG recording
Global ECG Parameters
RR Rate
Ok (Min Value: 0 - Max Precision: 0)
0.0
(/min)
Automatic interpretation
Insert a comment (optional)

Javadoc

it.crs4.most.ehrlib

ArchetypeFragment

public class **ArchetypeFragment** extends Fragment

This class allows you to display an Archetype in a Fragment. You just have to provide the *WidgetProvider* handling the archetype you want to include to the constructor or, if you prefer, you can use the *setWidgetProvider(WidgetProvider)* method (in this second case, remember to call this method before adding the fragment to the container).

Constructors

ArchetypeFragment

public **ArchetypeFragment**()

ArchetypeFragment

public **ArchetypeFragment**(*WidgetProvider wp*)

Create an Archetype fragment by providing the WidgetProvider

Parameters

- **wp** –

Methods

getFormContainer

public *FormContainer* **getFormContainer**()

Get the form container of this fragment

Returns the form container of this archetype fragment

getwidgetProvider

public *WidgetProvider* **getwidgetProvider**()

Returns the widget provider of this Archetype Fragment

onCreateView

public View **onCreateView**(LayoutInflater *inflater*, ViewGroup *container*, Bundle *savedInstanceState*)

setWidgetProvider

public void **setWidgetProvider**(*WidgetProvider wp*)

Set the widget provider for this fragment. This method must called BEFORE adding the fragment to itws container.

Parameters

- **wp** –

ArchetypeSchemaProvider

public class **ArchetypeSchemaProvider**

Utility class that provides a convenient way for getting the json schemas needed for loading an archetype on the EhrLibViewer.

Constructors

ArchetypeSchemaProvider

public **ArchetypeSchemaProvider**(Context *context*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *archetypesPropertyFile*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.htm *schemasRootDir*)

Provides default datatypes, layouts and ontology schema for specific archetypes. All available archetype schema must be specified in a property file provided as input argument

Parameters

- **context** – the application context
- **archetypesPropertyFile** – the path of the Archetype Property file, containing a list of key-values like =
- **schemasRootDir** – the root dir (under assets folder) containing all archetypes schemaa (note that each subfolder is a folder for a specific archetype. e.g blood_pressure)

Methods

getAdlStructureSchema

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getAdlStructureSchema**(String (http://docs.oracle.com/ja *archetype-Class*)

Get the adl schema for the specified archetype class, or null if not available

Parameters

- **archetypeClass** – the archetype class, (e.g openEHR-EHR-OBSERVATION.blood_pressure.v1)

Returns the json schema representing the internal structure of this specific archetype, or null if not available

getDatatypesSchema

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getDatatypesSchema** (String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *archetypeClass*)

Get the datatypes schema for the specified archetype class, or null if not available

Parameters

- **archetypeClass** – the archetype class, (e.g openEHR-EHR-OBSERVATION.blood_pressure.v1)

Returns the json schema containing the internal structure of datatypes used for visually representing this specific archetype, or null if not available

getLayoutSchema

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getLayoutSchema** (String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *archetypeClass*)

Get the Layout schema for the specified archetype class, or null if not available

Parameters

- **archetypeClass** – the archetype class, (e.g openEHR-EHR-OBSERVATION.blood_pressure.v1)

Returns the json schema of the default layout, or null if not available

getOntologySchema

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getOntologySchema** (String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *archetypeClass*)

Get the ontology schema for the specified archetype class, or null if not available

Parameters

- **archetypeClass** – the archetype class, (e.g openEHR-EHR-OBSERVATION.blood_pressure.v1)

Returns the json schema of the default ontology, or null if not available

FormContainer

public class **FormContainer**

A Form Container contains the list of *DatatypeWidget* widgets included in a Form along with the visual layout containing them.

Fields

index

int **index**
The index.

layout

ViewGroup **layout**
The layout.

widgets

List (http://docs.oracle.com/javase/6/docs/api/java/util/List.html)<*DatatypeWidget*<*EhrDatatype*>> **widgets**
The widgets.

Constructors

FormContainer

public **FormContainer**(ViewGroup *layout*, List (http://docs.oracle.com/javase/6/docs/api/java/util/List.html)<*DatatypeWidget*<*EhrI*
widgets, int *index*)
Creates a new Form Container

Parameters

- **layout** – the layout the layout containing all the *DatatypeWidget*
- **widgets** – the list of the *DatatypeWidget*
- **index** – the index of this form

Methods

getIndex

public int **getIndex**()
Get the index of this form container

Returns the index

getLayout

public ViewGroup **getLayout**()
Gets the layout of this form

Returns the layout

getWidgets

public List (http://docs.oracle.com/javase/6/docs/api/java/util/List.html)<*DatatypeWidget*<*EhrDatatype*>> **getWidgets** ()
Get the widgets of this form container.

Returns the widgets

resetAllWidgets

public void **resetAllWidgets** ()
Reset the content of all widgets of this form according to the current value of their underlying data types.

resetWidget

public void **resetWidget** (int *index*)
Reset the content of the selected widget according to the current value of the underlying data type.

Parameters

- **index** – the index

submitAllWidgets

public void **submitAllWidgets** ()
Submit all widgets.

Throws

- ***InvalidDatatypeException*** – if any of the widgets contains invalid data

submitWidget

public void **submitWidget** (int *index*)
Update the value of the underlying data type according to the current content of the widget.

Parameters

- **index** – the widget index

Throws

- ***InvalidDatatypeException*** – if the content cannot be converted to the datatype.

PriorityComparison

public class **PriorityComparison** implements Comparator (http://docs.oracle.com/javase/6/docs/api/java/util/Comparator.html)<*L*
Helper class for sorting widgets by priority.

Constructors

PriorityComparison

public **PriorityComparison**(JSONObject *layoutSchema*)
Instantiates a new priority comparison.

Parameters

- **layoutSchema** – the layout schema

Methods

compare

public int **compare**(*DatatypeWidget*<*EhrDatatype*> *item1*, *DatatypeWidget*<*EhrDatatype*> *item2*)

TemplateProvider

public class **TemplateProvider**
This class represents a visual Archetypes Template, according to the OpenEHR specifications. A template is an ordered list of `WidgetProvider` , each of them contains the layout of a specific archetype of the template-

Constructors

TemplateProvider

public **TemplateProvider**(Context *ctx*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *templateSchema*, *ArchetypeSchemaProvider* *archetypeSchemaProvider*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *language*)
Creates the template, building all the archetypes specified in the provided json schemas.

Parameters

- **ctx** – the application Context
- **templateSchema** – the json schema of the template
- **archetypeSchemaProvider** – the archetype schema provider
- **language** – the default ontology language

Throws

- **JSONException** – if an error occurred during the parsing of the json schemas

Methods

getId

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getId**()

getName

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getName**()

getWidgetProviders

public List (http://docs.oracle.com/javase/6/docs/api/java/util/List.html)<*WidgetProvider*> **getWidgetProviders**()
Get the list of the widget providers of this template, one for each archetype

Utils

public class **Utils**
This class contains utility methods internally used by the framework.

Methods

getLocaleStringResource

public static String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getLocaleStringResource**(Context *ctx*, String (http://docs.orac *locale*, int *resourseId*)

Gets the locale string resource.

Parameters

- **ctx** – the ctx
- **locale** – the locale
- **resourseId** – the resourse id

Returns the locale string resource

WidgetProvider

public class **WidgetProvider**
A WidgetProvider lets you build a set of visual and interactive widgets corresponding to a specific OpenEHR Archetype. The Archetype description is specified by a set of json structures (to be provided to the class constructor).

Fields

_container

protected LinearLayout **_container**
The _container.

_layout

protected LinearLayout **_layout**
The _layout.

_viewport

protected ScrollView **_viewport**
The _viewport.

clusterWidgetsMap

protected Map (http://docs.oracle.com/javase/6/docs/api/java/util/Map.html)<String (http://docs.oracle.com/javase/6/docs/api/java/lang/
The cluster widgets map.

defaultLayoutParams

public static final LayoutParams **defaultLayoutParams**
The Constant defaultLayoutParams.

sectionWidgetsMap

protected Map (http://docs.oracle.com/javase/6/docs/api/java/util/Map.html)<String (http://docs.oracle.com/javase/6/docs/api/java/lang/
The section widgets map.

Constructors

WidgetProvider

public **WidgetProvider**(Context *context*, *ArchetypeSchemaProvider asp*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *archetypeClassName*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *language*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *jsonExclude*)
Setup a Widget provider representing a specific archetype, according to the specified Archetype Schema Provider and archetype class name

Parameters

- **context** – get application context
- **asp** – the Archetype Schema Provider
- **archetypeClassName** – the name of the archetype class to be built (e.g: openEHR-EHR-OBSERVATION.blood_pressure.v1)

- **language** – the default ontology language
- **jsonExclude** – the json array containing a list of item ids to be excluded from the archetype

Throws

- **InvalidDatatypeException** –
- **JSONException** –

WidgetProvider

public **WidgetProvider**(Context *context*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *jsonDatatypes*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *jsonOntology*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *jsonAdlStructure*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *jsonLayoutSchema*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *language*)

Setup a Widget provider representing a specific archetype, according to the specified json datatypes schema , json archetype structure and json ontology.

Parameters

- **context** – the application context
- **jsonDatatypes** –
  - the json description of all datatypes used by this archetype, subdivided in sections
- **jsonOntology** –
  - the json ontology (it includes a textual description of each item of the archetype)
- **jsonAdlStructure** –
  - the initial json structure of the archetype (optionally including initial values)
- **jsonLayoutSchema** – (optional, it can be null) the layout schema containing informations about visual rendering (sections, custom widgets, priorities..)
- **language** –
  - the default language code (any language code included in the ontology json schema)

Throws

- **InvalidDatatypeException** –
- **JSONException** – the JSON exception

WidgetProvider

public **WidgetProvider**(Context *context*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *jsonDatatypes*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *jsonOntology*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *jsonAdlStructure*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *jsonLayoutSchema*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *language*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *jsonExclude*)

Setup a Widget provider representing a specific archetype, according to the specified json datatypes schema ,

json archetype structure and json ontology.

Parameters

- **context** – the application context
- **jsonDatatypes** –
  - the json description of all datatypes used by this archetype, subdivided in sections
- **jsonOntology** –
  - the json ontology (it includes a textual description of each item of the archetype)
- **jsonAdlStructure** –
  - the initial json structure of the archetype (optionally including initial values)
- **jsonLayoutSchema** – (optional, it can be null) the layout schema containing informations about visual rendering (sections, custom widgets, priorities..)
- **jsonExclude** – (optional, it can be null) the list of archetype items (i.e their id , like "at0004") to exclude from the viewer
- **language** –
  - the default language code (any language code included in the ontology json schema)

Throws

- **InvalidDatatypeException** –
- **JSONException** –
  - if an error occurred during the parsing of the json schemas

Methods

buildFormView

public *FormContainer* **buildFormView** (int *index*)

build a view containing all widgets according to the json archetype structure, layout and ontology, All widgets are rendered in a vertical layout, optionally ordered by section and/or item priority (if specified in the layout json schema)

Parameters

- **index** – the index of this Form Container

Returns the FormContainer containing all widgets, ordered by section and item priority in a vertical layout

getClusterWidgets

public List (http://docs.oracle.com/javase/6/docs/api/java/util/List.html)<*DatatypeWidget*<*EhrDatatype*>> **getClusterWidgets** (S
(h
ci
te
in
it
m

getContext

public Context **getContext**()
Get the application context

Returns the application context

getDatatypesSchema

public JSONObject **getDatatypesSchema**()

getOntology

public JSONObject **getOntology**()
Get the json schema containing the ontology of this archetype

Returns the ontology json schema

getOntology

public static JSONObject **getOntology**(String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *data*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *language*)
Get the ontology .

Parameters

- **data** – the ontology schema (including all available languages)
- **language** – the selected language

Returns a json object containing the ontology of the specified language

getSectionWidgets

public List (http://docs.oracle.com/javase/6/docs/api/java/util/List.html)<*DatatypeWidget*<*EhrDatatype*>> **getSectionWidgets**(S
(h
se
ti
in
it
m

getSections

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html)[] **getSections**()
Get the sections of this archetype structure.

Returns the sections

parseFileToString

public static String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **parseFileToString**(Context *context*, String (http://docs.oracle.com/ja *filename*)

Parses the file to string.

Parameters

- **context** – the context
- **filename** – the filename

Returns the string

toJson

public JSONObject **toJson**()

get a Json representation of the current state of this archetype.

Returns the JSON object

updateOntologyLanguage

public void **updateOntologyLanguage**(String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *lang*)

Update the ontology of all *DatatypeWidget* widgets.

Parameters

- **lang** – the language code (ISO 639-1:2002)

updateSectionsJsonContent

public void **updateSectionsJsonContent**(int *index*)

Update the json structure according to the current value of the datatype widgets belonging to this form.

Parameters

- **index** – the form index

Throws

- **JSONException** – the JSON exception

Returns the updated json structure

it.crs4.most.ehrlib.datatypes

DvBoolean

public class **DvBoolean** extends *EhrDatatype*
This class represents a DV_BOOLEAN item, according to the definition provided by the OpenEHR Data Type Information Model

Constructors

DvBoolean

public **DvBoolean** (String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*, JSONObject *attributes*)
Instantiates a new DV_BOOLEAN item.

Parameters

- **path** – the path
- **attributes** – the attributes

Methods

fromJSON

public void **fromJSON** (JSONObject *data*)

getValue

public boolean **getValue** ()
Gets the current value of this DV_BOOLEAN item

Returns the current text of this DV_TEXT item

setAttributes

protected void **setAttributes** (JSONObject *attributes*)

setValue

public void **setValue** (boolean *value*)
Sets the text of this DV_BOOLEAN item

Parameters

- **text** – the new text

toJSON

public JSONObject **toJSON** ()

DvCluster

public class **DvCluster** extends *EhrDatatype*
It is a particular datatype that is a container for other datatypes.

Constructors

DvCluster

public **DvCluster** (String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*, JSONObject *attributes*)
Instantiates a new DvCluster datatype

Parameters

- **path** – the path of this datatype
- **attributes** – the attributes of this datatype

Methods

fromJSON

public void **fromJSON** (JSONObject *data*)

getSectionName

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getSectionName** ()
Get the name of the datatype section containing all the datatypes of this cluster

Returns the name of the section

isCluster

public boolean **isCluster** ()

setAttributes

protected void **setAttributes** (JSONObject *attributes*)

toJSON

public JSONObject **toJSON** ()

DvCodedText

public class **DvCodedText** extends *EhrDatatype*
This class represents a DV_CODED_TEXT item, according to the definition provided by the OpenEHR Data Type Information Model

Fields

terminology

String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **terminology**
The terminology.

Constructors

DvCodedText

public **DvCodedText** (String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*, JSONObject *attributes*)
Instantiates a new DV_CODED_TEXT item.

Parameters

- **path** – the path
- **attributes** – the attributes

Methods

fromJSON

public void **fromJSON** (JSONObject *data*)

getOptions

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html)[] **getOptions** ()
Gets the options of this DV_CODED_TEXT

Returns the options

getSelectedOption

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getSelectedOption** ()
Gets the selected option.

Returns the selected option

getSelectedOptionIndex

public int **getSelectedOptionIndex** ()
Gets the selected option index.

Returns the selected option index

setAttributes

protected void **setAttributes**(JSONObject *attributes*)

setSelectedOptionIndex

public void **setSelectedOptionIndex**(int *index*)
Sets the selected option index.

Parameters

- **index** – the new selected option index

toJSON

public JSONObject **toJSON**()

DvQuantity

public class **DvQuantity** extends *EhrDatatype*
This class represents a DV_QUANTITY item, according to the definition provided by the OpenEHR Data Type Information Model

Constructors

DvQuantity

public **DvQuantity**(String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*, JSONObject *attributes*)
Instantiates a new DV_QUANTITY item.

Parameters

- **path** – the path
- **attributes** – the attributes

Methods

fromJSON

public void **fromJSON**(JSONObject *content*)

getConstraintsInfo

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getConstraintsInfo**()
Gets the constraints info.

Returns the constraints info

getMagnitude

public double **getMagnitude**()
Gets the current magnitude value of this datatype.

Returns the magnitude

getMax

public int **getMax**()
Gets the maximum value admitted for this DV_QUANTITY item

Returns the maximum value admitted for this DV_QUANTITY item

getMaxPrecision

public int **getMaxPrecision**()
Gets the maximum precision (i.e the maximum number of decimal digits admitted for this DV_QUANTITY item)

Returns the maximum precision

getMin

public int **getMin**()
Gets the minimum value admitted for this DV_QUANTITY item

Returns the minimum value admitted for this DV_QUANTITY item

getUnits

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getUnits**()
Gets the unit of measure adopted by this DV_QUANTITY item

Returns the current unit of measure

getValidityMessage

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getValidityMessage**(double *value*)
Gets the validity message.

Parameters

- **value** – the value

Returns the validity message

isValid

public boolean **isValid**(double *value*)

Checks if the value provided as argument is valid for this DV_QUANTITY item or not.

Parameters

- **value** – the value to be checked

Returns `True`, if ithe calue is valid, `False` otherwise

setAttributes

protected void **setAttributes**(JSONObject *attributes*)

setMagnitude

public void **setMagnitude**(double *magnitude*)

Sets the magnitude value.

Parameters

- **magnitude** – the new magnitude value

Throws

- ***InvalidDatatypeException*** – if a not valid magnitude value is specified

setMax

public void **setMax**(int *max*)

Sets the maximum value admitted for this DV_QUANTITY item

Parameters

- **max** – the maximum value admitted for this DV_QUANTITY item

setMaxtPrecision

public void **setMaxtPrecision**(int *precision*)

Sets the maximum precision (i.e the maximum number of decimal digits admitted for this DV_QUANTITY item)

Parameters

- **precision** – the highest precision

setMin

public void **setMin**(int *min*)

Sets the minimum value admitted for this DV_QUANTITY item

Parameters

- **min** – the minimum value admitted for this DV_QUANTITY item

setUnits

public void **setUnits**(String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *units*)
Sets the unit of measure adopted by this DV_QUANTITY item

Parameters

- **units** – the new unit of measure

toJSON

public JSONObject **toJSON**()

DvText

public class **DvText** extends *EhrDatatype*
This class represents a DV_TEXT item, according to the definition provided by the OpenEHR Data Type Information Model

Constructors

DvText

public **DvText**(String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*, JSONObject *attributes*)
Instantiates a new DV_TEXT item.

Parameters

- **path** – the path
- **attributes** – the attributes

Methods

fromJSON

public void **fromJSON**(JSONObject *data*)

getText

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getText**()
Gets the current text of this DV_TEXT item

setAttributes

protected void **setAttributes**(JSONObject *attributes*)

setText

public void **setText**(String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *text*)
Sets the text of this DV_TEXT item

Parameters

- **text** – the new text

toJSON

public JSONObject **toJSON**()

EhrDatatype

public abstract class **EhrDatatype**
This is the base class for all data types included in a generic Archetype, as defined by the OpenEHR ADL structure

Fields

datatypeChangeListener

protected *EhrDatatypeChangeListener* **datatypeChangeListener**
The datatype change listener.

path

protected String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **path**
The path of the datatype

Methods

fromJSON

public abstract void **fromJSON**(JSONObject *data*)
Load the new content of this datatype from a json schema.

Parameters

- **data** – the json structure representing this datatype

Throws

- ***InvalidDatatypeException*** – the invalid datatype exception
- **JSONException** – if a malformed json structure was provided

getPath

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getPath**()
Gets the path of this datatype

Returns the path

isCluster

public boolean **isCluster**()

Returns `True` if this datatype is a container for other datatypes, `False` otherwise

isInnerArchetype

public boolean **isInnerArchetype**()

Returns `True` if this datatype is an archetype itself `False` otherwise

setAttributes

protected abstract void **setAttributes**(JSONObject *attributes*)
Sets the attributes for this datatype. Generally, different datatypes have different attributes.

Parameters

- **attributes** – the json structure containing all the attributes of this datatype.

Throws

- **JSONException** – if a malformed json structure was provided

setDatatypeChangeListener

public void **setDatatypeChangeListener**(*EhrDatatypeChangeListener* *datatypeChangeListener*)
Sets the Event listener interface for 'change' events.

Parameters

- **datatypeChangeListener** – the Listener where to notify any content modification of this datatype

setPath

protected void **setPath**(String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*)
Sets the path of this datatype.

Parameters

- **path** – the new path

toJSON

public abstract JSONObject **toJSON**()
Get the json structure representing the current state of this datatype.

Returns the JSON structure representing the current state of this datatype

EhrDatatypeChangeListener

public interface **EhrDatatypeChangeListener**<T extends EhrDatatype>
The listener interface for receiving ehrDatatypeChange events. The class that is interested in processing a ehrDatatypeChange event implements this interface, and the object created with that class is registered with a component using the component's *EhrDatatype.setDatatypeChangeListener(EhrDatatypeChangeListener)* method. When the ehrDatatypeChange event occurs, that object's appropriate method is invoked.

Parameters

- **<T>** – the generic datatype extending the `EhrDatatype`

Methods

onEhrDatatypeChanged

public void **onEhrDatatypeChanged**(T *datatype*)
Called when a datatype changed its content.

Parameters

- **datatype** – the datatype with the updated value

InnerArchetype

public class **InnerArchetype** extends *EhrDatatype*
This class represents an Archetype item included in another one.

Constructors

InnerArchetype

public **InnerArchetype**(*WidgetProvider wp*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*, JSONObject *attributes*)
Instantiates a new Archetype item.

Parameters

- **wp** – the Widget Provider of the archetype
- **path** – the absolute path of the archetype inside the json structure
- **attributes** –

Methods

fromJSON

public void **fromJSON**(JSONObject *data*)

getArchetypeClass

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getArchetypeClass**()

Returns the name of the archetype class

getWidgetProvider

public *WidgetProvider* **getWidgetProvider**()

isInnerArchetype

public boolean **isInnerArchetype**()

Returns `True` if this datatype is an archetype itself `False` otherwise

setAttributes

protected void **setAttributes**(JSONObject *attributes*)

toJSON

public JSONObject **toJSON**()

it.crs4.most.ehrlib.exceptions

InvalidDatatypeException

public class **InvalidDatatypeException** extends Exception (http://docs.oracle.com/javase/6/docs/api/java/lang/Exception.html)

This exception is called when a user attempts to set an invalid value to a *`EhrDatatype`* item.

Constructors

InvalidDatatypeException

public **InvalidDatatypeException**(String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *validityMessage*)

Instantiates a new invalid datatype exception.

Parameters

- **validityMessage** – the validity message

it.crs4.most.ehrlib.parser

AdlParser

public class **AdlParser**

This class provides methods for exploring, retrieving and updating the contents of a JSON structure representing an OpenEHR Archetype

Constructors

AdlParser

public **AdlParser**(JSONObject *jsonData*)

Instantiates a new adl parser.

Parameters

- **jsonData** – the json data

Methods

getItemsContainer

public *AdlStructure* **getItemsContainer**(String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*)

Get an ADL structure containing all the items included in an archetype path. The path must be the absolute path of any of its items. For instance, providing the path data[at0001]/events[at0006]/data[at0003]/items[at0004] , this method returns the ADL structure included in data[at0001]/events[at0006]/data[at0003]

Parameters

- **path** – the path of an item

Returns the AdlStructure

getStructureByPath

public *AdlStructure* **getStructureByPath**(String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*)

Get the structure corresponding to the specified path.

Parameters

- **path** – the path of an item

Returns the AdlStructure

replaceContent

public void **replaceContent**(String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*, int *index*, JSONObject *newContent*)

Replace the content of a json structure.

Parameters

- **path** – the path of the json substructure to be replaced
- **index** – the index of the json instance
- **newContent** – the json structure containing the new json content

Throws

- **JSONException** – the JSON exception

AdlStructure

public class **AdlStructure**
The Class AdlStructure contains a JSON structure representing a whole or a part of an OpenEHR Archetype.

Constructors

AdlStructure

public **AdlStructure** (Object (http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html) *item*)
Instantiates a new adl structure.

Parameters

- **item** – the structure. It can be a JSONObject (a single structure or datatype) or a JSONArray (a list of structures and/or datatypes).

Methods

count

public int **count** ()
Get the amount of instances of this structure.

Returns the int

getCardinality

public *StructureCardinality* **getCardinality** ()
Gets the cardinality of this ADL structure

Returns the cardinality

getOriginalObject

public Object (http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html) **getOriginalObject** ()
Gets the original item provided for building this ADL structure.

Returns the original object

See also: `.AdlStructure(Object)`

getStructure

public JSONObject **getStructure** (int *index*)
Get the index-th occurrence of this ADL structure.

Parameters

- **index** – the index

Throws

- **JSONException** – the JSON exception

Returns the adl structure

toString

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **toString** ()

StructureCardinality

public enum **StructureCardinality**
The StructureCardinalityenum represents athe cardinality of a section of an ADL Archetype.

Enum Constants

MULTIPLE

public static final *StructureCardinality* **MULTIPLE**

UNIQUE

public static final *StructureCardinality* **UNIQUE**

UNKNOWN

public static final *StructureCardinality* **UNKNOWN**

it.crs4.most.ehrlib.widgets

DatatypeWidget

public abstract class **DatatypeWidget**<T extends EhrDatatype> implements *EhrDatatypeChangeListener*<T>
This the base class for all DatatypeWidgets. A *DatatypeWidget* is a visual and interactive widget mapped on a specific *EhrDatatype*. A user can instantiate a *DatatypeWidget* for reading, editing and saving the content of the *EhrDatatype* handled by it.

Parameters

- **<T>** – the generic *EhrDatatype*

Fields

_context

protected Context **_context**
The _context.

_name

protected String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **_name**
The _name.

_ontology

protected JSONObject **_ontology**
The _ontology.

_parent_index

protected int **_parent_index**
The _parent_index.

_priority

protected int **_priority**
The _priority.

_root_view

protected View **_root_view**
The _root_view.

_view

protected View **_view**
The _view.

_widget_provider

protected *WidgetProvider* **_widget_provider**

datatype

protected T **datatype**
The datatype.

description

protected String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **description**
The description.

displayTitle

protected String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **displayTitle**
The display title.

toolTip

protected ToolTip **toolTip**
The tool tip.

Constructors

DatatypeWidget

protected **DatatypeWidget**()

DatatypeWidget

public **DatatypeWidget**(*WidgetProvider provider*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *name*, T *datatype*, int *parent_index*)
Instantiates a new *DatatypeWidget* widget.

Parameters

- **context** – the context
- **name** – the name of this widget
- **datatype** – the *EhrDatatype* to be handled by this widget
- **ontology** – the ontology used
- **parent_index** – the parent_index

Methods

getDatatype

public T **getDatatype**()
Gets the *EhrDatatype* handled by this widget

Returns the *EhrDatatype*

getDescription

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getDescription**()
Gets the description.

Returns the description

getDisplayTitle

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getDisplayTitle**()
Gets the display title.

Returns the display title

getName

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getName**()
returns the name of this widget

Returns the name

getParentIndex

public int **getParentIndex**()
Gets the parent index.

Returns the parent index

getPriority

public int **getPriority**()
returns the visual priority of this widget (essentially this means it's physical location in the form).

Returns the priority

getToolTip

public ToolTip **getToolTip**()
Gets the tool tip.

Returns the tool tip

getView

public View **getView**()
get the Root View containing this widget's view elements.

Returns the view

getWidgetProvider

public *WidgetProvider* **getWidgetProvider**()

replaceTooltip

protected abstract void **replaceTooltip**(ToolTip *tooltip*)
Replace tooltip.

Parameters

- **tooltip** – the tooltip

reset

public abstract void **reset**()
Reset all fields of this widget according to the current underlying datatype value.

save

public abstract void **save**()
Update the value of the underlying datatype according to the current value of the fields of this widget.

Throws

- ***InvalidDatatypeException*** – if the current value of the fields cannot be converted to a datatype value

setOntology

public void **setOntology**(JSONObject *ontology*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *lang*)
Sets the ontology.

Parameters

- **ontology** – the new ontology
- **lang** – the language of the new loaded ontology

setPriority

public void **setPriority**(int *value*)
sets the visual priority of this widget essentially this means it's physical location in the form.

Parameters

- **value** – the new priority

setVisibility

public void **setVisibility**(int *value*)
set the visibility of this widget.

Parameters

- **value** – the new visibility

setupTooltip

protected void **setupTooltip**()
Setup tooltip.

updateLabelsContent

protected abstract void **updateLabelsContent**()
Update the content of the labels of the widget, according to the current ontology language.

DvBooleanWidget

public class **DvBooleanWidget** extends *DatatypeWidget*<*DvBoolean*> implements ToolTipView.OnToolTipViewClickedListener
This class represents a visual widget mapped on a `DvBoolean` datatype.

Constructors

DvBooleanWidget

public **DvBooleanWidget**(*WidgetProvider provider*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *name*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*, JSONObject *attributes*, int *parentIndex*)
Instantiates a new `DvBooleanWidget`

Parameters

- **provider** – the widget provider
- **name** – the name of this widget
- **path** – the path of the `DvBoolean` mapped on this widget
- **attributes** – the attributes of the `DvBoolean` mapped on this widget
- **parentIndex** – the parent index

Methods

onEhrDatatypeChanged

public void **onEhrDatatypeChanged**(*DvBoolean datatype*)
See also: `it.crs4.most.ehrlib.datatypes.EhrDatatypeChangeListener.onEhrDatatypeChanged(it.crs4.most.ehrlib.datatypes.EhrDatatype)`

onToolTipViewClicked

public void **onToolTipViewClicked**(ToolTipView *arg0*)

See also: `com.nhaarman.supertooltips.ToolTipView.OnToolTipViewClickedListener.onToolTipViewClicked(com.nhaarman.supertooltips.ToolTipView)`

replaceTooltip

protected void **replaceTooltip**(ToolTip *tooltip*)

See also: `it.crs4.most.ehrlib.widgets.DatatypeWidget.replaceTooltip(com.nhaarman.supertooltips.ToolTip)`

reset

public void **reset**()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.reset()*

save

public void **save**()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.save()*

updateLabelsContent

protected void **updateLabelsContent**()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.updateLabelsContent()*

DvClusterWidget

public class **DvClusterWidget** extends *DatatypeWidget*<*DvCluster*>

This class represents a visual widget mapped on a *`DvCluster`* datatype.

Fields

myToolTipView

protected ToolTipView **myToolTipView**

Constructors

DvClusterWidget

public **DvClusterWidget**(*WidgetProvider provider*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *name*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*, JSONObject *attributes*, int *parentIndex*)

Instantiate a new DvClusterWidget

Parameters

- **provider** – the widget provider
- **name** – the name of this widget
- **path** – the path of the *DvCluster* mapped on this widget
- **attributes** – the attributes of the *DvCluster* mapped on this widget
- **parentIndex** – the parent index of this widget

Methods

onEhrDatatypeChanged

public void **onEhrDatatypeChanged**(*DvCluster datatype*)

See also: `it.crs4.most.ehrlib.datatypes.EhrDatatypeChangeListener.onEhrDatatypeChanged(it.crs4.most.ehrlib.datatypes.EhrDatatype)`

replaceTooltip

protected void **replaceTooltip**(ToolTip *tooltip*)

reset

public void **reset**()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.reset()*

save

public void **save**()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.save()*

updateLabelsContent

protected void **updateLabelsContent**()

DvCodedTextAsListWidget

public class **DvCodedTextAsListWidget** extends *DatatypeWidget*<*DvCodedText*>

This class represents a visual widget mapped on a *DvCodedText* datatype. It renders all the options of the *DvCodedText* datatype in a ListView.

Constructors

DvCodedTextAsListWidget

public **DvCodedTextAsListWidget** (*WidgetProvider* *provider*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *name*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*, JSONObject *attributes*, int *parentIndex*)

Instantiates a new *DvCodedTextAsListWidget*

Parameters

- **provider** – the widget provider
- **name** – the name of this widget
- **path** – the path of the *DvCodedText* mapped on this widget
- **attributes** – the attributes of the *DvCodedText* mapped on this widget
- **parentIndex** – the parent index

Methods

onEhrDatatypeChanged

public void **onEhrDatatypeChanged** (*DvCodedText* *datatype*)

See also: `it.crs4.most.ehrlib.datatypes.EhrDatatypeChangeListener.onEhrDatatypeChanged(it.crs4.most.ehrlib.datatypes.EhrDatatype)`

replaceTooltip

protected void **replaceTooltip** (ToolTip *tooltip*)

See also: `{@linkit.crs4.most.ehrlib.widgets.DatatypeWidget.replaceTooltip(com.nhaarman.supertooltips.ToolTip)}`

reset

public void **reset** ()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.reset()*

save

public void **save** ()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.save()*

updateLabelsContent

protected void **updateLabelsContent** ()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.updateLabelsContent()*

DvCodedTextWidget

public class **DvCodedTextWidget** extends *DatatypeWidget*<*DvCodedText*>

This class represents a visual widget mapped on a `DvCodedText` datatype. It renders all the options of the `DvCodedText` datatype in a Combobox.

Constructors

DvCodedTextWidget

public **DvCodedTextWidget** (*WidgetProvider provider*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *name*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*, JSONObject *attributes*, int *parentIndex*)

Instantiates a new `DvCodedTextWidget`

Parameters

- **provider** – the widget provider
- **name** – the name of this widget
- **path** – the path of the `DvCodedText` mapped on this widget
- **attributes** – the attributes of the `DvCodedText` mapped on this widget
- **parentIndex** – the parent index

Methods

onEhrDatatypeChanged

public void **onEhrDatatypeChanged** (*DvCodedText datatype*)

See also: `it.crs4.most.ehrlib.datatypes.EhrDatatypeChangeListener.onEhrDatatypeChanged(it.crs4.most.ehrlib.datatypes.EhrDatatype)`

replaceTooltip

protected void **replaceTooltip** (ToolTip *tooltip*)

reset

public void **reset** ()

See also: `it.crs4.most.ehrlib.widgets.DatatypeWidget.reset()`

save

public void **save** ()

See also: `it.crs4.most.ehrlib.widgets.DatatypeWidget.save()`

updateLabelsContent

protected void **updateLabelsContent**()

DvQuantityWidget

public class **DvQuantityWidget** extends *DatatypeWidget*<*DvQuantity*>
This class represents a visual widget mapped on a `DvQuantity` datatype.

Fields

_input

protected EditText **_input**
The _input.

_labUnity

protected TextView **_labUnity**
The _lab unity.

_title

protected TextView **_title**
The _title.

_txtvalidity

protected TextView **_txtvalidity**
The _txtvalidity.

Constructors

DvQuantityWidget

public **DvQuantityWidget**(*WidgetProvider provider*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *name*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*, JSONObject *attributes*, int *parentIndex*)
Instantiates a new `DvQuantityWidget`

Parameters

- **provider** – the widget provider
- **name** – the name of this widget
- **path** – the path of the `DvQuantity` mapped on this widget
- **attributes** – the attributes of the `DvQuantity` mapped on this widget
- **parentIndex** – the parent index

Methods

onEhrDatatypeChanged

public void **onEhrDatatypeChanged**(*DvQuantity datatype*)

See also: `it.crs4.most.ehrlib.datatypes.EhrDatatypeChangeListener.onEhrDatatypeChanged(it.crs4.most.ehrlib.datatypes.EhrDatatype)`

replaceTooltip

protected void **replaceTooltip**(ToolTip *tooltip*)

See also: `it.crs4.most.ehrlib.widgets.DatatypeWidget.replaceTooltip(com.nhaarman.supertooltips.ToolTip)`

reset

public void **reset**()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.reset()*

save

public void **save**()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.save()*

updateLabelsContent

protected void **updateLabelsContent**()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.updateLabelsContent()*

DvTextWidget

public class **DvTextWidget** extends *DatatypeWidget*<*DvText*> implements ToolTipView.OnToolTipViewClickedListener

This class represents a visual widget mapped on a *DvText* datatype.

Constructors

DvTextWidget

public **DvTextWidget**(*WidgetProvider provider*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *name*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*, JSONObject *attributes*, int *parentIndex*)

Instantiates a new *DvTextWidget*

Parameters

- **provider** – the widget provider
- **name** – the name of this widget

- **path** – the path of the *DvText* mapped on this widget
- **attributes** – the attributes of the *DvText* mapped on this widget
- **parentIndex** – the parent index

Methods

onEhrDatatypeChanged

public void **onEhrDatatypeChanged**(*DvText datatype*)

See also: `it.crs4.most.ehrlib.datatypes.EhrDatatypeChangeListener.onEhrDatatypeChanged(it.crs4.most.ehrlib.datatypes.EhrDatatype)`

onToolTipViewClicked

public void **onToolTipViewClicked**(ToolTipView *arg0*)

See also: `com.nhaarman.supertooltips.ToolTipView.OnToolTipViewClickedListener.onToolTipViewClicked(com.nhaarman.supertooltips.ToolTipView)`

replaceTooltip

protected void **replaceTooltip**(ToolTip *tooltip*)

See also: `it.crs4.most.ehrlib.widgets.DatatypeWidget.replaceTooltip(com.nhaarman.supertooltips.ToolTip)`

reset

public void **reset**()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.reset()*

save

public void **save**()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.save()*

updateLabelsContent

protected void **updateLabelsContent**()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.updateLabelsContent()*

InnerArchetypeWidget

public class **InnerArchetypeWidget** extends *DatatypeWidget*<*InnerArchetype*>

This class represents a visual widget mapped on a *InnerArchetype* datatype.

Fields

myToolTipView

protected ToolTipView **myToolTipView**

Constructors

InnerArchetypeWidget

public **InnerArchetypeWidget**(*WidgetProvider provider*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *name*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *path*, JSONObject *attributes*, int *parentIndex*)

Instantiate a new InnerArchetypeWidget

Parameters

- **provider** – the widget provider
- **name** – the name of this widget
- **path** – the path of the *InnerArchetype* mapped on this widget
- **attributes** – the attributes of the *InnerArchetype* mapped on this widget
- **parentIndex** – the parent index of this widget

Methods

onEhrDatatypeChanged

public void **onEhrDatatypeChanged**(*InnerArchetype datatype*)

See also: `it.crs4.most.ehrlib.datatypes.EhrDatatypeChangeListener.onEhrDatatypeChanged(it.crs4.most.ehrlib.datatypes.EhrDatatype)`

replaceTooltip

protected void **replaceTooltip**(ToolTip *tooltip*)

reset

public void **reset**()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.reset()*

save

public void **save**()

See also: *it.crs4.most.ehrlib.widgets.DatatypeWidget.save()*

setOntology

public void **setOntology** (JSONObject *ontology*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *language*)

Sets the ontology.

Parameters

- **ontology** – the new ontology

updateLabelsContent

protected void **updateLabelsContent** ()

it.crs4.most.ehrlib.widgets.filters

DvQuantityFilter

public class **DvQuantityFilter**

This class is internally used by the *DvQuantityWidget*.

Fields

TAG

protected static final String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **TAG**

The Constant TAG.

validityTextView

TextView **validityTextView**

The validity text view.

Constructors

DvQuantityFilter

public **DvQuantityFilter** (EditText *editText*, TextView *validityTextView*, *DvQuantity* *dvQuantity*)

Instantiates a new dv quantity filter.

Parameters

- **editText** – the edit text
- **validityTextView** – the validity text view
- **dvQuantity** – the dv quantity

Examples

All the following examples are located in the folder *client/android/examples/* of the Most-Report repository.

- **EhrLibArchetypeViewerExample** This example shows you:
- how to load an Archetype into an Activity, according to the provided json schema files
- how to read, edit and save the content of the Archetype
- how to get textual informations about any item of the archetype
- how to read the json file containing the current content of the archetype
- **EhrTemplateViewerExample** This example explains you:
- how to load a template of archetypes into an Activity, according to the provided json schema files
- how to read, edit and save the content of the Archetypes of the template
- how to get textual informations about any item of each archetype of the template
- **NestedArchetypeActivityExample** This example contains all the features of the *EhrTemplateViewerExample* example, and in addition, shows you:
- how to read the json file containing the current content of each archetype of the template
- how the library supports archetypes including other archetypes inside it

For running the Android examples, open your preferred IDE (e.g Eclipse) and do the following:

- Import the EhrLib project library (located in the folder client/android/src/EhrLib of the Most-Report Repository)
- Import the ‘SuperTooltips Lib <https://github.com/nhaarman/supertooltips/>‘_library and add it as a dependency of the EhrLib project
- Import your preferred Android project example located in the *android/examples* folder and add the *EhrLib* project as Library reference
- Build the projects
- Deploy the application on an android device or emulator

License

```
/*!
 * Project MOST - Moving Outcomes to Standard Telemedicine Practice
 * http://most.crs4.it/
 *
 * Copyright 2014, CRS4 srl. (http://www.crs4.it/)
 * Dual licensed under the MIT or GPL Version 2 licenses.
 * See license-GPLv2.txt or license-MIT.txt
 */
```

GPL2: https://www.gnu.org/licenses/gpl-2.0.txt

MIT: http://opensource.org/licenses/MIT

Authors

Code author: Stefano Leone Monni <stefano.monni@crs4.it>

Code author: Cecilia Mascia <cmascia@crs4.it>

Code author: Francesco Cabras <francesco.cabras@crs4.it>

Indices and tables

- genindex
- modindex
- search

Index

Symbols

A

B

C

D

P

R

S

T

U

V

W