

Preferences in MITK

File Edit Show View Undo Ctrl+Z Redo Ctrl+Y Preferences... Ctrl+P

Standard View

Transversal

Coronal

Filter:

hotkeys

DataManager

Hotkeys

Hotkeys

Delete selected nodes Del

Global Reinit Ctrl+, R

Load Ctrl+, L

Make all nodes invisible V

Reinit selected nodes R

Save selected nodes Ctrl+, S

Show Node Information Ctrl+, I

Toggle visibility of selected nodes D

Import ... Export ... Apply Close

Data Storage Selection

Default DataStorage

Selected Nodes (Use Right Mouse Button for a Context Menu)

Name Data Type Visibility

1 widget1Plane Geometry2D... true

2 widget2Plane Geometry2D... true

3 widget3Plane Geometry2D... true

4 patient_003_ct Image true

Load Global Reinit

Properties

Filter:

Name

LookupTable n/a TransferFunction RTTI typeinfo: Reference Count Modified Time: 8 Debug: Off Observers: none

TransferFunction binary false

bounding box false

color

in plane resample extent by geometry false

layer 0

levelwindow L:665.5 W:3377

name patient_003_ct

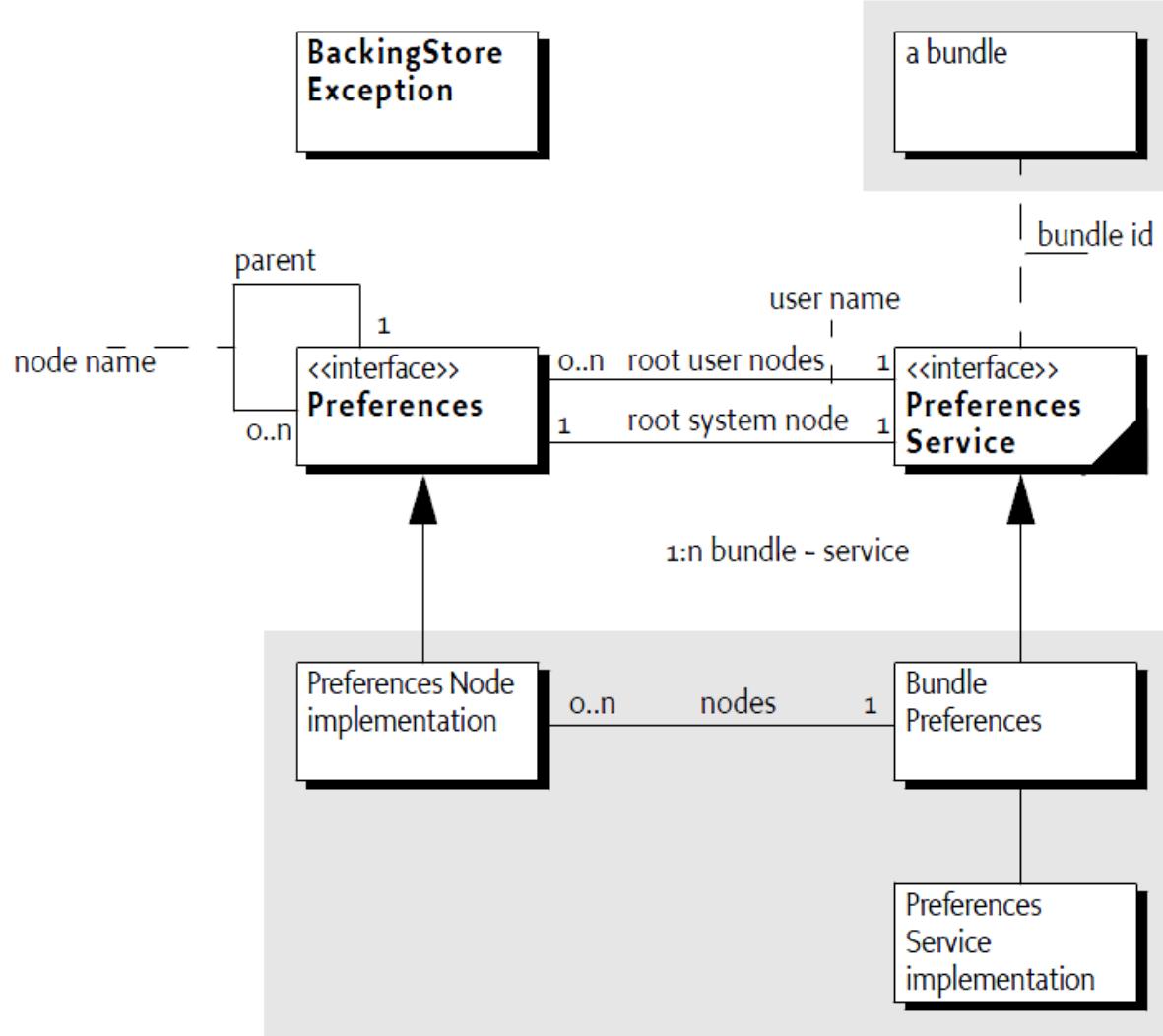
opacity 1

Position: <256; 257.63; 22.12> mm; Time: 0 ms; Pixelvalue: 25 135.55 MB (4.08%)

The screenshot shows the MITK application interface. On the left, there is a 3D volume rendering window with two slices (Transversal and Coronal) visible. Below it is a log window showing system messages. In the center, the 'Preferences' dialog is open, specifically the 'Hotkeys' page under the 'DataManager' category. It lists several hotkey assignments. At the bottom of the preferences dialog are buttons for Import, Export, Apply, and Close. To the right of the preferences dialog is the 'Datamanager' panel, which includes a 'Data Storage Selection' section with a dropdown for 'Default DataStorage' and a list of 'Selected Nodes'. Below this is a table showing node details like name, data type, and visibility. Further down are 'Load' and 'Global Reinit' buttons. The bottom half of the Datamanager panel is a 'Properties' section with a table showing various properties for a selected node, such as TransferFunction, bounding box, and opacity. A status bar at the bottom provides system information like position, time, pixel value, memory usage, and a battery icon.

Overview

Preferences Class Diagram



Retrieve the Preferences Service

```
#include "cherryIPreferencesService.h"
#include "cherryPlatform.h"

...
// get the preferences service
IPreferencesService::Pointer preferencesService =
    Platform::GetServiceRegistry().GetServiceById<IPreferencesService>
( IPrefsService::ID );

...
// get a preferences NODE
IPrefs::Pointer generalPreferences
= preferencesService->GetSystemPreferences( )->Node( "/General" );
```

Getting/Putting Values

```
// put some values in a preferences NODE
generalPreferences->PutBool("a bool variable", true);
generalPreferences->Put("a string", "blaaaa");
generalPreferences->PutInt("int", 3);

// get some values from a preferences NODE
// always deliver a default value in case the key does not exist
bool boolVar = generalPreferences->GetBool("a bool variable", false);
std::string stringVar = generalPreferences->Get("a string", "");
int intVar = generalPreferences->GetInt("int", 0);

// force immediate save of preferences node
generalPreferences->Flush();
```

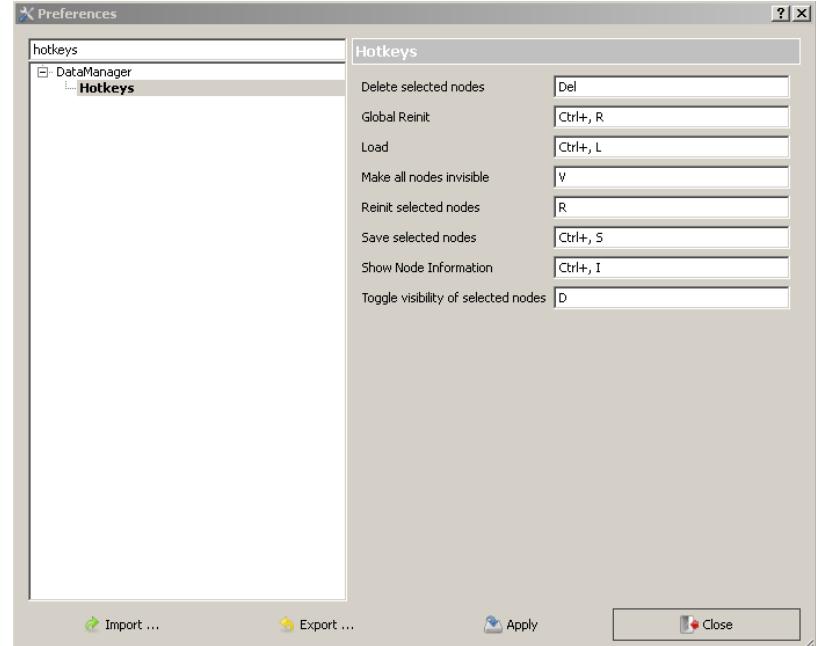
Adding a Preferences Page (I)

1. Inherit from IQtPreferencesPage

```
#include „cherryIQtPreferencePage.h“  
...  
class QmitkMyPreferencePage : public cherry::IQtPreferencePage  
{  
    Q_OBJECT  
    /// Build UI in constructor  
    QmitkGeneralPreferencePage(QWidget* parent = 0, Qt::WindowFlags f = 0);  
    /// Invoked when the OK button was clicked in the preferences dialog  
    virtual bool PerformOk();  
    /// Invoked when the Cancel button was clicked in the preferences dialog  
    virtual void PerformCancel();  
    ...  
};
```

2. Add to files.cmake

```
SET(INTERNAL_CPP_FILES  
    QmitkMyPreferencePage.cpp  
)  
SET(MOC_H_FILES  
    src/internal/QmitkMyPreferencePage.h  
)
```



Adding a Preferences Page (II)

3. Add to manifest.cpp

```
#include "src/internal/QmitkMyPreferencePage.h"

POCO_BEGIN_NAMED_MANIFEST(cherryIQtPreferencePage, cherry::IQtPreferencePage)

POCO_EXPORT_CLASS(QmitkMyPreferencePage)

POCO_END_MANIFEST
```

4. In plugin.xml:

```
<extension point="org.opencherry.ui.preferencePages">
  <page id="org.mitk.gui.qt.<mybundle>.QmitkMyPreferencePage" name="My Preferences"
        class="QmitkMyPreferencePage">
    <keywordreference id="org.mitk.gui.qt.<mybundle>.QmitkMyPreferencePageKeywords">
      </keywordreference>
  </page>
</extension>
```

Adding a Preferences Page (III)

5. (Adding keywords in plugin.xml:)

```
<extension point="org.opencherry.ui.keywords">  
  <keyword id="org.mitk.gui.qt.<mybundle>.QmitkMyPreferencePageKeywords"  
    label="my preferences keywords separated by spaces"></keyword>  
</extension>
```