

04/06/2014

Persistence in MITK

Alfred Michael Franz

Junior Group: Computer-assisted Interventions



**GERMAN
CANCER RESEARCH CENTER**
IN THE HELMHOLTZ ASSOCIATION

Persistence: capability of an application to permanently store

- Data
- Application Settings / States
 - of basic classes / modules
 - of the user interface
- [...]

Data Objects: MITK Data Storage

The screenshot displays the MITK (Medical Image Toolkit) software interface. At the top, three data objects are shown: **mitk::PointSet** (a collection of yellow spheres), **mitk::Surface** (a 3D white model of a human torso), and **mitk::Image** (an axial CT scan slice). Below these, the main interface features a menu bar with **Open**, **Save Project**, **Close Project**, **Undo**, and **Redo**. A red box highlights the **Open** and **Save Project** buttons, labeled **Load & Save**. The central workspace is divided into four viewports: **Axial**, **Sagittal**, **Coronal**, and a 3D perspective view. A **Data Manager** panel on the right lists the loaded objects: **mySurface**, **myCTImage**, and **myPointSet**. A red box around this panel is labeled **DataStorage**. The status bar at the bottom provides technical details: **Position: <0.49, -189.51, -238.59> mm; Index: <256, 256, 99> ; Time: 0.00 ms; Pixelvalue: -978.00** and **252.48 MB (6.23 %)**.

For custom data classes that derive from `mitk::BaseData`:

- New reader/writer implementations are needed.
- See bug squashing seminars about reader & writer for more details.
- Work in progress: a new concept for readers and writers
 - See bug 14866

Persistence for

1. UI/Qt independent modules:
→ MITK Persistence Service
2. Qt dependent modules / views:
→ Qt Settings
3. BlueBerry Plugins:
→ Persistence Features of BlueBerry

also planned for MITK:

Implementation of OSGI Configuration Service

- Store your settings, variables, etc as MITK property list
- This list is still available after restart of the application

Example:

header:

```
#include <mitkIPersistenceService.h>
//[...]
private:
    PERSISTENCE_GET_SERVICE_METHOD_MACRO
```

implementation:

```
mitk::PropertyList::Pointer propList = this->GetPersistenceService()
                                         ->GetPropertyList("org.mitk.myUniqueModule");
//set a variable:
propList->Set("deviceNumber", m_Controls->GrabbingDeviceNumber->value());
//get a variable:
int grabbingDeviceNumber = 0;
propList->Get("deviceNumber", deviceNumber);
```

- Preliminary version was added for the 2014-03 release.
- Still not fully functional, bugs might occur.
- See bug 16643 for current status.

- Qt settings can be used to permanently store variables.

Example:

header:

```
#include <QSettings>
//[...]
QSettings m_MySettings;
```

implementation:

```
//initialize the settings object (e.g. constructor):
m_MySettings("MyClass", "MyDescription")
//store settings:
m_MySettings.setValue("identifier", value);
//load settings:
int intExample = m_MySettings.value("identifier").toInt();
```

1. IMemento class:

save and restore view states

2. berryPreferencesService:

persistence of variables, etc.



A modular, cross-platform, C++ application framework

<http://blueberry.berlios.de>

- Every view that inherits from `berry::IViewPart` should overwrite these two methods:
 - `void Init(IViewSite::Pointer site, IMemento::Pointer m)`
 - `void SaveState(IMemento::Pointer m)`
- Use the `IMemento` class to save / restore your states.
- You get an instance of `IMemento` from the `Init` method of `berry::IViewPart`
- `IMemento` is reverted if the view is closed by the user

Detailed documentation:

http://mitk.org/Article_Save_and_Restore_your_View_State

- Preferences should be used whenever a user changeable value which does not have a direct representation in your view should be persisted.
- class: **berryPreferencesService**

Thank you for
your attention!

Further
information
on www.dkfz.de

dkfz.

GERMAN
CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION

50 Years – Research for
A Life Without Cancer