Things to know about the QmitkFunctionality class

Important methods to overwrite (I)

- virtual void **CreateQtPartControl** (QWidget* parent)
  - Create your GUI here, *parent is a QScrollArea*
- virtual void **ClosePart**()
  - Called when the part gets closed finally: remove your event listeners here (not in the destrcutor)
- virtual void **OnSelectionChanged**(std::vector<mitk::DataTreeNode*> nodes)
  - Called when a DataTreeNode selection was thrown by the SelectionService
- virtual void **OnPreferencesChanged**(const berry::IBerryPreferences*)
  - Called when the preferences object of this view changed

Important methods to overwrite (II)

- virtual void **Activated**()
  - Add your interactors (make changes in the StdMultiWidget, ...)
- virtual void **Deactivated**()
  - Remove your interactors, etc. ...
- virtual bool **IsExclusiveFunctionality**() const
  - Should return true, if Activated() and Deactivated() have to be called on your functionality, default: true
- virtual void **NodeAdded**(const mitk::DataTreeNode* node)
  - Called when a new node was added to the DataStorage
  - virtual void **NodeChanged**(const mitk::DataTreeNode* node)
  - virtual void **NodeRemoved**(const mitk::DataTreeNode* node)
- virtual void **DataStorageChanged**()
  - Called when NodeAdded, NodeChanged *or* NodeRemoved occured

Activated() and Deactivated()

- Activated() on functionality **X** gets called if:
  - IsExclusiveFunctionality() in **X** returns true
  - If two or more exclusive functionalities are visible at the same time and **X** receives focus
  - **Or** if **X** is or becomes the only visible exclusive functionality

- Deactivated() on functionality **X** gets called if:
  - IsExclusiveFunctionality() in **X** returns true
  - If another exclusive functionality gets the focus ...
  - **Or** if **X** gets hidden or closed

Important methods to use

- void **FireNodesSelected**(std::vector<mitk::DataTreeNode::Pointer> nodes)
  - Informs other parts of the workbench that the nodes are selected via the blueberry selection service
  - void FireNodeSelected(mitk::DataTreeNode::Pointer node)
- std::vector<mitk::DataTreeNode*> **GetCurrentSelection**()
  - Returns the selection of the currently active part
- std::vector<mitk::DataTreeNode*> **GetDataManagerSelection**()
  - Returns the current DataManager selection
- berry::IPreferences::Pointer **GetPreferences**() const
  - Returns the preferences object for this functionality (Path: „/<view-id>“, e.g. „/org.mitk.views.datamanager“)
- QmitkStdMultiWidget* **GetActiveStdMultiWidget**()
  - Returns the currently opened StdMultiWidget *or* creates a new one and returns it
- Other methods: IsActive(), IsVisible(), WaitCursorOn(), etc. (see docs)

Event emulation after CreateQtPartControl()

- **OnSelectionChanged**() and **OnPreferencesChanged()** are automatically called once directly after **CreateQtPartControl()**