

## **The Motif Multi-Document Interface**

The Motif Multi-Document Interface (MDI) is a collection of C++ classes that emulates the behavior of the Multi-Document Interface in Microsoft Windows. The MDI framework allows a user to view multiple documents (windows) constrained to a single parent window.

**CLASS STRUCTURE:**

****

Figure 1. Inheritance Graph for MDI classes

The *XsMDICanvas* is a self-contained component used to display and manage any number of child document windows. All documents windows are derived from the abstract base-class *XsMDIWindow*. To get the Motif-like functionality, document windows should be derived from the *XsMotifWindow* class.

**EXAMPLE:**

The process of building and displaying a Multi-Document Interface using MDI consists of the following steps:

1. Creating the application document(s)
2. Creating the MDI canvas
3. Adding the document(s) to the canvas

#include "XsMDICanvas.h"  
#include "XsMotifWindow.h"  
  
// Application document (derived from XsMotifWindow)  
  
class MyDocument : public XsMotifWindow {  
 public:  
 MyDocument (const char \*name);  
 virtual ~MyDocument ( );  
 protected:  
 virtual void \_buildClientArea (Widget parent);  
};  
   
void createCanvas (Widget parent) {  
  
// Create documents  
  
 MyDocument \*doc1 = new MyDocument ("doc1");  
 MyDocument \*doc2 = new MyDocument ("doc2");  
   
// Create the canvas  
  
 XsMDICanvas \*canvas = new XsMDICanvas ("canvas", parent);  
   
// Add documents to canvas  
  
 canvas->add (doc1);  
 canvas->add (doc2);  
  
// Show the canvas  
  
 canvas->show ( );  
}

In this example, the application document *MyDocument* is derived from *XsMotifWindow*. This provides a Motif-like window suitable for use with the *XsMDICanvas*.

Next, two *MyDocument* objects are created along with the *XsMDICanvas*. The two documents are then added to the canvas using the *add* member-function of the canvas. Lastly, the canvas is shown (managed) using the *show* member-function.

Creating the document *MyDocument* does not automatically create any widgets. Rather, it only initializes internal variables. The widgets are not created until the document is added to the canvas. The *XsMDICanvas* is responsible for calling *XsMotifWindow::\_buildClientArea()* at an appropriate time. In this member-function, the application can create the actual contents of the document.

The member-function *\_buildClientArea* is passed a widget to be used as the parent of the document contents. This parent widget is an unmanaged *XmForm* widget. The application is free to create whatever contents it needs as a child of the *XmForm* parent.

**CLASS REFERENCES:**

Of the classes in the MDI package, only the following should be of interest to MDI library users:

* [XsMDICanvas](http://docs.google.com/canvas.html)
* [XsMotifWindow](http://docs.google.com/mwindow.html)

**EXPLORING RESOURCES:**

The MDI classes support a number of different X-resources (please refer to the class manual pages for complete details). In order to get a feel for the customization capabilities of the MDI library, try running the test program (*MDItest*) with the following command-line options:

MDItest -xrm "\*showBorder:false" MDItest -xrm "\*showTitle:false" -xrm "\*showResize:false" MDItest -xrm "\*showMenu:false" -xrm "\*showMaximize:false" MDItest -xrm "\*borderSize:4" -xrm "\*buttonSize:14" MDItest -xrm "\*lowerOnIconify:true" -xrm "\*title:Hello World"

**ADDITIONAL IINFORMATION:**

The test program *MDItest.C* gives a complete example of an MDI application. It should serve as a good reference/example of the MDI library.