BENTLEY EMPOWERED CONFERENCE
Walt Disney World Swan and Dolphin
May 23-27, 2004
Orlando, Florida USA

Developing AddIn Applications with the AddIn ProjectWizard

Hands-on class

Presenter: Mark Anderson, Technical Manager Bentley Developer Network

Bentley Systems, Incorporated
685 Stockton Drive
Exton, PA 19341
www.bentley.com

sponsored by
Bentley Institute

LESSON NAME: WHAT IS AN ADDIN?

LESSON OBJECTIVE: UNDERSTAND AN ADDIN FROM THE USER PERSPECTIVE

An Addin application is a .NET assembly that is loaded into MicroStation's runtime space. This application will load and behave similar to traditional MDL based applications. It will have an MDL descriptor to manage resources against. It can have a command table that is integrated into the MicroStation command table. It uses the COM api for the majority of interaction with MicroStation based information. It can use Microsoft Winforms for its user interface. It can be developed using any of the .NET supported programming languages.

> ***EXERCISE: LOAD AND RUN AN EXAMPLE***

With MicroStation started we will be loading a simple demo application

1. From the MicroStation keyin browser issue the command:
   mdl load demoapp
   This will load the application similar to the traditional MDL applications.
2. Use the buttons on the form to run the various commands in the application.
3. Now use the keyin commands in the keyin browser
   demoapp place
   demoapp locate
   demoapp scan

Note Take note of the user interface components as they are not MDL based.

LESSON NAME: HOW TO DEVELOP AN ADDIN

LESSON OBJECTIVE: UNDERSTAND THE ADDIN CONSTRUCTION PROCESS

To build AddIn applications you can start from a blank project and add each of the class files in the traditional approach or you can use the provided AddIn wizard with Microsoft's ViasualStudio2005. This wizard will guide you through the features available and build the skeleton for the application. The basics are that you need a class that sub-classes the MicroStationAddIn class. When the class does this it needs to add some additional information that tells it what file is used for the command definitions and the 15 character mdldescriptor name. Along with the AddIn class the typical addin will have a class to handle the keyins. This class will consist of a set of public static methods that are referenced in the command table. The command table file is an XML file that has a specific XML structure in it.

> *EXERCISE: CREATE AN ADDIN APPLICATION*

You will need to create an addin using the Wizard provided. This exercise will work through the core features of the addin. The structure that is being used is to have a single main class that sub classes from the addin base class. The next main feature is to define command entry points in a simple class. The main class will provide some common functionality that is to be used by other parts of the application. The entry points are typically in a Keyin class. The keyin class is linked to the MicroStation command table through the use of an xml file that defines the structure which is read in and compiled into the command table.

1. From the File pulldown menu select new project.
2. In the C++ Projects find the MicroStation AddIn Wizard
3. Set the name of the Project to AddInClassEx
4. Select OK to start the wizard
5. Select the language VB.NET or C# (for this class)
6. Select the Features Keyin support only (for now)
7. Select Command Type Placement (for now)
8. Add the references to …
9. edit the stdpostbuild.bat file to set the path to MicroStation
10. add a post build event to run the stdpostbuild.bat file
11. build the Addin in the IDE

LESSON NAME: DISECTING THE ADDIN

LESSON OBJECTIVE: UNDERSTAND THE OPTIONS FOR APPLICATION DEVELOPMENT WITH ADDINS

In the process of defining an AddIn application there are guides to allow the application to work in standard MicroStation fashion.

Placement Command – The implementation of a command that will add elements to the design file. To do this the application will need to implement the IPrimitiveCommandEvents. This will provide the framework to interact with MicroStation.

Locate Command – The implementation of a command that will allow the user to select elements in the design file. To do this the application will need to implement the ILocateCommandEvents. This interface has callbacks that MicroStation will call in the selection process.

Using Winforms – By using the Bentley.MicroStation.Winform.Adapter class forms can be integrated with MicroStation's windowing system. This also allows applications to leverage such common things as the tools settings dialog.

> ***EXERCISE: REVIEW SOURCE FOR DEMO APPLICATION***

1. Open the DemoApp source code file. This is the main application source file.
2. Look at the ComApp and MyAddin properties. These are provided to get a reference to the base addin class to allow various modules in the application to communicate and use common parts. The GetApp property provides a common point to get the host MicroStation instance.
3. Open the DemoAppForm source code file.
4. Review the class declaration.
   the Constructor(s)
   The
5. Open the DemoAppPlacementCmd source code file.
6. The StartPlaceCommand is the static entry point that is called from the keyin. The rest of the methods are implementation of the IPrimitiveCommandEvents interface. This class will create an instance of the form and then attach it to the toolsettings dialog. This allows AddIns to be better integrated into MicroStation's user interface.
7. Open the DemoAppLocateCmd source code file.

8. The StartLocateCommand is the static entry point that is called from the keyin. The rest of the methods are the implementation of the ILocateCommandEvents interface.
9. Add the references necessary for the application:
   From the MicroStation Dir – ustation.dll
   From the Assemblies Dir – Bentley.Interop.MicroStationDGN.dll
   - Bentley.MicroStation.dll
   - Bentley.MicroStation.General.dll
   - Bentley.Windowing.dll

   From the ECFramework Dir – Bentley.General.1.0.dll
   From the >NET tab of the Add References dialog
   System
   System.Data
   System.Drawing
   System.Windows.Forms

LESSON NAME: <INSTALLING THE ADDIN WIZARD>

LESSON OBJECTIVE:

When you go to your own computer you will need to install a few files to make the wizard work. The files have been broken into two parts. The first part is the control files and the second is the template files. You can edit the template files to have your own specific method headers.

On-Line Help Topic: <name of specific topic the user can search for in the appropriate help file>

> ***EXERCISE: <EXERCISE TITLE HERE>***

<Describe exercise objective and initial conditions>

1. Copy the MicroStationAddInControl.zip file to the VisualStudio 8\VC\vcprojects directory.
2. copy the MicroStationAddInWizard2005-Rev1.zip file to the VisualStudio 8\VC\VCWizards directory
3. unzip the MicroStationAddInControl.zip in the VC\vcrojects directory to place three files. This will give you the link into the VS New project dialog.
4. unzip the template file to expand out the MicroStationAddIn directory. This will provide the wizard mechanics and the templates for the source code.
5. Start VisualStudio look in the C++ projects for the MicroStation AddIn Wizard

LESSON NAME: <WRITING OUR OWN ADDIN>

LESSON OBJECTIVE:

This is "free" time to experiment writing some modules of your own to practice using the wizard. You will want to focus on implementing commands and adding user interface to your application.

> ***EXERCISE: FREE FORM TIME***

Create a "free form" project. There is no set name for the project. You will need to create and build the skeleton project.

1. Create a new project with the wizard.
2. Add a Primitive and Locate Command
3. Add the necessary references.
4. update the StdPostBuild.bat file to validate the addin
5. set the commands.xml file to be embedded in the dll
6. build the application
7. add a Windows form to the project and reference it into the PrimitiveCommand
8. Add User Interface Item to the form. Put a simple text field to set the text to place.
9. Add a second Windows form to the project and reference it into the LocateCommand.
10. Add User Interface Items to the forms. Put a simple text field to retrive the text string
11. Make the application locate an element and inquire the element information.

LESSON NAME: <FAQ>

LESSON OBJECTIVE:

The list of Can and Can't do's:
Do make use of the COM object model.
Do use Winforms for User Interface
Don't use the Bentley.Internal assemblies
Do build the app in the IDE
Do not copy the assembly references local.