Today’s Lecture

- Hall of Shame / Fame
  - How much information to show?

- C# Tips / Tricks
  - Quasi-global classes
  - User control

- Software Design
  - MVC – Model View Controller
  - MV
  - Observer / Pub-Sub

Reminders
- Blog Post (Week)
- Homework 5
Hall of Shame

WARNING

Cookie:rambo amadeus@microsoft.com is a Cookie!! Are you sure you want to delete it?

Yes  No
Hall of Shame

File Upload

The following file is currently on the host but was not listed in your file list. Do you want to remove this file from the host?

apply.gif 859  Nov 08, 1996 16:20

Yes  Yes to All  No  No to All  Cancel
Hall of Fame

Add to CVS Version Control

There are 26 resources that are not under CVS version control. Do you want to add them?

Yes  No  Cancel

Checked resources will be added to CVS version control.

/\apis/src/edu/mit/csail/lapis/views/SimultaneousEditingView.java
/\apis/src/edu/mit/csail/lapis/views/SidebarPaneView.java
/\apis/src/edu/mit/csail/lapis/views/SidebarPaneRegistry.java
/\apis/src/edu/mit/csail/lapis/views/SampleView.java
/\apis/src/edu/mit/csail/lapis/views/PatternSearchView.java
/\apis/src/edu/mit/csail/lapis/views/LapisSidebar.java
/\apis/src/edu/mit/csail/lapis/preferences/LapisPreferencePage.java

Select All  Deselect All
Discussion

- Good / bad?

![Add to CVS Version Control]

![File Upload]
Q: How can I get the equivalent of a C# global?

A: Properties and static are your friend
Let’s call it database

class Database
{
    static ListCards m_Cards = new ListCards();

    public ListCards TheCards
    {
        get { return m_Cards; }
    }
}
In your XAML C# or other code

Access it via the static object

public void myFunction ()
{
    Database.TheCards.ToString();

    lbxCardList.ItemsSource = Database.TheCards;
}

12/13/2009
Other Examples

☐ Error logging
  ■ Pipe errors to the console or to a file or both
  ■ Dynamically change it based on appropriate behavior or XML file switch

☐ Translation
  ■ Hit an object for translating an English string to any other language
User Control

- Allows you to make your own “control”
  - Repetition of controls

- Pros
  - Split your XAML file
  - Re-use outside of your app

- Cons
  - Locally constrained -> can’t see outside of the user control
    - Binding to an external control gets trickier
    - C# code can’t easily see member variables in outer window
Add User Control

Will create two files
XAML file
C# (.cs) file
<UserControl x:Class="IndustryTracker.ItemChooser"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
    <Grid Loaded="itemChooser_Loaded">
        <TabControl Grid.Row="0">
            <TabItem Header="Items">
                <Grid>
                    <TreeView x:Name="treeViewItems" />
                </Grid>
            </TabItem>
            <TabItem Header="Search"/>
            <TabItem Header="Quick"/>
        </TabControl>
    </Grid>
</UserControl>
Larger Context

- Several components
  - Include the XML namespace
    - Similar to how we made resources
  - Access it via that name
    - Assume loc namespace
    - loc:ItemChooser
<UserControl x:Class="IndustryTracker.ItemChooser"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <Grid Loaded="itemChooser_Loaded">
        <TabControl Grid.Row="0">
            <TabItem Header="Items">
                <Grid>
                    <TreeView x:Name="treeViewItems" />
                </Grid>
            </TabItem>
            <TabItem Header="Search" />
            <TabItem Header="Quick" />
            <TabItem />
        </TabControl>
    </Grid>
</UserControl>
Model View Controller

- Separate concerns
  - Front end from back end
  - Input from output
- Multiple views on same data
- Re-use views / controllers
- Example: Textbox
  - Model: Mutable string
  - View: Rectangle with text inside
  - Controller: Keyboard
Model

☐ Responsible for data
  ■ Maintains application state (data fields)
  ■ Implements state changing behavior
  ■ Notifies dependent views / controllers when changes occur

☐ Sound familiar?

☐ Design issues
  ■ Granularity of changes?
    ☐ String has changed versus where it changed
  ■ Granularity of observables?
    ☐ Everything versus visible?
View

- Responsible for output
  - Occupies screen extent
  - Draws on the screen
  - Listens for changes from model
  - Queries the model to draw it

- Model versus view
  - View has only one model
  - Model may have many views
Controller

- Responsible for input
  - Listen for keyboard, mouse, touch events
  - Instructs the model or view to change
    - Text into a string

- Model versus controller
  - Controller has only one model and one view
Problem: Controller Needs Output

☐ Menu
  ■ Clearly controller-related
  ■ Right click on the menu item

☐ Also needs to be drawn
  ■ Is it its own MVC?
Problem: Selection in a Listbox

☐ Must be displayed by view
   ■ What is highlighted?

☐ Must be updated / used by controller
   ■ Arrow keys -> up / down

☐ Selection in model
   ■ Blend visual + model
   ■ Independent / dependent selections
Problem: Direct Manipulation

- User points at object and manipulates it directly
  - Scatter View

- View must provide affordances for controller
  - Scrollbar thumb

- View needs feedback about state
  - Button pressed?
Enter MV

- Instead of MVC, MV
  - Model View
  - How often are controllers re-usable
  - Re-usable view
    - Widget
    - Manages input and output
View Hierarchy

- Views are arranged hierarchically
- Containers
  - Panels, Frames, etc.
- Components
  - Button, text box, etc.
- Nearly every GUI has a view hierarchy
View Hierarchy: Output

- Drawing
  - When to draw

- Clipping
  - Edge of window / control

- Z-Order
  - Who is on top

- Coordinate System
  - Where are you x, y
View Hierarchy: Input

- Event dispatch / propagation
  - Start at lowest, work the way up
  - Seem familiar?

- Keyboard focus
  - Whom will the keyboard go to?
Who does the sizing?
- Dynamic vs. fixed
- Fonts / language
- In theory
  - Reduce level of work on layout
Observer Pattern

- Decouple model from views
  - Publish / subscribe
  - Properties
Publish / Subscribe

- Views register with the model
- Model is modified of changes
- Model notifies of an update to the views
  - Delegate in C#
  - List of functions
- Views get the new data
Questions?

- Weekly Blog
- Homework 5 (Posted Wed)