

# Silverlight memory board ( Part 2 )

---

In the first part this tutorial we created a new Silverlight project and designed the user interface. In this part, we will add some code to the project to make the memory board completely functional.

## Step 1: add a note to the memory board

- Open the project from part 1 in Visual Studio 2008 and open the file note.xaml.cs
- The constructor must accept four arguments: a unique id for the note, an x and y position and a text.
- Create a property for the id and fill in the property with the argument from the constructor.
- Fill in the Text property of the textbox with the text argument
- Also set the left and top value for the note. You can use dependency properties for this.
- This should be your code:

```
public string Id { get; set; }

public note(string id, double x, double y, string text)
{
    // Required to initialize variables
    InitializeComponent();

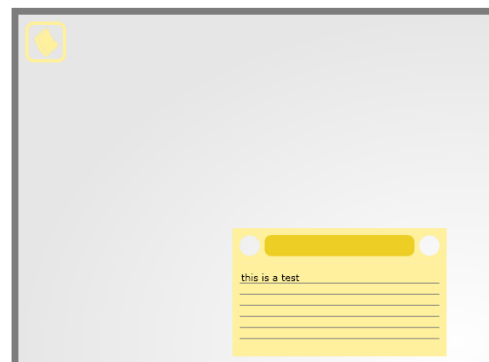
    // give a value to the properties
    this.Id = id;
    txtContent.Text = text;
    // set the top en left position on the parent canvas for the note
    this.SetValue(Canvas.LeftProperty, x);
    this.SetValue(Canvas.TopProperty, y);
}
```

- Open the file page.xaml.cs and add an eventhandler for a click on the new button.
- In the eventhandler, you create a new instance for a note and fill in the arguments for his constructor.
- Add the new instance to the main canvas.

```
public Page()
{
    InitializeComponent();
    // new eventhandler for a click on the button
    btnNew.Click += new RoutedEventHandler(btnNew_Click);
}

void btnNew_Click(object sender, RoutedEventArgs e)
{
    // create a new instance and add it to the mainCanvas
    // the current date and time makes the id unique
    Note newNote = new Note(DateTime.Now.ToString("yyyyMMddHHmmss"), 80, 8, "");
    mainCanvas.Children.Add(newNote);
}
```

If you test your program and click on the new button, you should see a new note in the memory board at the right position, with the correct text.



## Step 2: implement drag and drop for the notes

In this step, we will implement drag and drop functionality for the notes, so we can drag them around in the memory board and put them wherever we want it. After this step, it will be possible to drag the notes out of the memory board, but we will use this in the next steps.

- Open note.xaml.cs again
- Add eventhandlers for a MouseEnter, MouseLeave and MouseLeftButtonDown on the pin ( rectangle ) and eventhandlers for a MouseLeftButtonUp and MouseMove on the control.

```
// add eventhandlers
pin.MouseEnter += new MouseEventHandler (pin_MouseEnter);
pin.MouseLeave += new MouseEventHandler (pin_MouseLeave);
pin.MouseLeftButtonDown += new MouseButtonEventHandler (pin_MouseLeftButtonDown);
this.MouseLeftButtonUp += new MouseButtonEventHandler (Note_MouseLeftButtonUp);
this.MouseMove += new MouseEventHandler (Note_MouseMove);
```

If the mouse enters the pin, I will set the opacity to 1, and if the mouse leaves the pin, the opacity will fall back to 0.5 ( note: you have to set the standard opacity of the pin to 0.5 in the xaml file first)

```
void pin_MouseLeave(object sender, MouseEventArgs e)
{
    pin.Opacity = 0.5;
}

void pin_MouseEnter(object sender, MouseEventArgs e)
{
    pin.Opacity = 1;
}
```

If the user clicks on the pin, we will set a global boolean variable startDrag to true. When the user moves the mouse, the note must follow the mouse and when the user releases the leftmouse button, we set startDrag to false again.

```
void Note_MouseMove(object sender, MouseEventArgs e)
{
    // only execute this code when the user has started dragging
    if (startDrag)
    {
        // position the center of the pin to the cursor of the mouse
        this.SetValue(Canvas.LeftProperty, e.GetPosition(null).X - pin.ActualWidth / 2);
        this.SetValue(Canvas.TopProperty, e.GetPosition(null).Y - pin.ActualHeight / 2);
    }
}

void Note_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)
{
    // stop forcing the capture of the moue to this element
    pin.ReleaseMouseCapture();
    startDrag = false;
}

void pin_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
{
    // force the capture of the moue to this element
    pin.CaptureMouse();
    startDrag = true;
}
```

If you test your movie, you should be able to add multiple notes to the memory board and drag them around over the entire board.



### Step 3: use isolated storage to read and write files.

The isolated storage is a kind of a filesystem in the silverlight plugin. I will use this in my tutorial to write a file for each note and to read all the files when you open the memory board again. You can compare the isolated storage with “advanced” cookies. More information can be found on [http://msdn2.microsoft.com/en-us/library/bdts8hk0\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/bdts8hk0(VS.80).aspx).

#### Write a file for each note

- Open note.xaml.cs and import the System.IO and System.IO.IsolatedStorage namespaces
- Create a global variable that addresses the isolated storage
- Create a function saveFile() that will create or overwrite a textfile which contains four lines: the id, x position, y position and the text.
- Create a function deleteFile() the will delete a specified file.

```
IsolatedStorageFile isoStore = IsolatedStorageFile.GetUserStoreForApplication();

void saveFile()
{
    // create or overwrite a textfile in the storage
    IsolatedStorageFileStream isoStream = new IsolatedStorageFileStream(this.Id +
        ".txt", FileMode.Create, FileAccess.Write, isoStore);

    // create a streamwriter for the file
    StreamWriter writer = new StreamWriter(isoStream);

    // write the data to the file
    writer.WriteLine(ID);
    writer.WriteLine(this.GetValue(Canvas.LeftProperty));
    writer.WriteLine(this.GetValue(Canvas.TopProperty));
    writer.WriteLine(txtContent.Text);

    // close the streams
    writer.Close();
    isoStream.Close();
}

void deleteFile()
{
    if (isoStore.FileExists(this.Id + ".txt"))
        isoStore.DeleteFile(this.Id + ".txt");
}
```

- Add an eventhandler for the TextChanged event of the textbox. Call the function saveFile in this handler.
- Go to the code for the MouseLeftButtonUp event and check if the note is released within the board or outside the board. Save the textfile when the note is released in the board, and delete it when it is released outside the board

```
void txtContent_TextChanged(object sender, TextChangedEventArgs e)
{
    saveFile();
}

void Note_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)
{
    // stop forcing the capture of the moue to this element
    pin.ReleaseMouseCapture();
    startDrag = false;

    // check if the note must be saved, or deleted
    double xPos = Convert.ToDouble(this.GetValue(Canvas.LeftProperty));
    double yPos = Convert.ToDouble(this.GetValue(Canvas.TopProperty));
    Canvas main = this.Parent as Canvas;
```

```

    if ((xPos >= main.ActualWidth || xPos < 0) || (yPos < 0 || yPos >= main.ActualHeight))
        deleteFile();
    else
        saveFile();
}

```

## Read the files from the isolated storage

Open page.xaml.cs and add a new function getFiles that will loop through all the files found in the isolated storage. For each file, we will create a new note, and add it to the main canvas.

```

IsolatedStorageFile isoStore = IsolatedStorageFile.GetUserStoreForApplication();
void getFiles()
{
    // put all the filenames in an array
    string[] files = isoStore.GetFileNames("*.txt");
    // loop through te files
    foreach (string file in files)
    {
        // create a new filestream
        IsolatedStorageFileStream isoStream = new IsolatedStorageFileStream(file,
            FileMode.Open, isoStore);
        StreamReader reader = new StreamReader(isoStream);

        // get the data from the file
        string id = reader.ReadLine().ToString();
        double x = Convert.ToDouble(reader.ReadLine());
        double y = Convert.ToDouble(reader.ReadLine());
        string content = reader.ReadToEnd().ToString();

        // create a new note and add it to the canvas
        Note newNote = new Note(id, x, y, content);
        mainCanvas.Children.Add(newNote);

        // close the streams
        reader.Close();
        isoStream.Close();
    }
}

```

Attention: don't forget to import the namespaces for the isolated storage. Finally, call the function in the constructor.

If you test your application, everything should works fine. You can add a new note, fill in a text and drag it on the board. The next time, you start the application; it should be exactly in the same place. You can also change the text, or delete the note by dragging it outside the board.

## Step 4: some extra's

### Multiline textbox

You probably noticed that you can only type one line of text in the notes. This is easily to fix. Just, set the property `AcceptReturn` for the textbox to true. You can do this in xaml or in c#.

### Add an animation

I will add a storyboard in the note.xaml file and start this storyboard with c# when a new note is created.

You can create the animation with Expression Blend or just type it in the note.xaml file. Add this xaml just below the `<UserControl>` element and above the main `<Grid>` element

```
<UserControl.Resources>
<Storyboard x:Name="fadeIn">
    <DoubleAnimationUsingKeyFrames Storyboard.TargetName="grid"
        Storyboard.TargetProperty="(UIElement.Opacity)" BeginTime="00:00:00">
        <SplineDoubleKeyFrame KeyTime="00:00:00" Value="0"/>
        <SplineDoubleKeyFrame KeyTime="00:00:01" Value="1"/>
    </DoubleAnimationUsingKeyFrames>
</Storyboard>
</UserControl.Resources>

<Grid x:Name="grid">...
```

Search for the storyboard and starts it with c#:

```
void btnNew_Click(object sender, RoutedEventArgs e)
{
    //create a new instance and add it to the mainCanvas
    Note newNote = new Note(DateTime.Now.ToString("yyyyMMddHHmmss"), 250, 250, "");
    mainCanvas.Children.Add(newNote);

    // starts the storyboard
    Storyboard fadeIn = newNote.FindName("fadeIn") as Storyboard;
    fadeIn.Begin();
}
```

Every time you create a new note, you will see this small animation