Strokes for Visual Studio

Solution documentation

by Jonas Swiatek

This document serves to describe the entire flow the the extension, with the purpose to introduce new developers to the project.

Solution architecture

The project is divided into a set of sub projects (at the time of writing 4), and two Achievement Extension Projects (Achievements and Challenges).

The purpose of this division is to enable third party developers to provide additional achievements as seperate VSIX installers, without being required to run a complete copy of the source code during their development.

The main Strokes project consists of these 6 projects:

Project	**Description**
Strokes.Core	Contains the classes common to all achievements, such as the Abstract Achievement-class. Third party achievement developers will only be required to reference this project in order to author their own achievements that will work with the strokes GUI.
Strokes.GUI	WPF Project containing all GUI code and templates.
Strokes.VSX	The primary Visual Studio Extension-project. This contains the code that gets launched by Visual Studio on load and on successful builds execute the DetectionDispatcher.
Strokes.Console	Test solution to do spot-testing rather than start an entire Visual Studio experimental instance just to test a minor thing.
Strokes.BasicAchievements.VSX	Contains a set of practical achievements.
Strokes.Challanges.VSX	Challenge-type achievements (empty at the moment)

Inside Strokes.VSX

The Strokes.VSX is a Visual Studio Extension project (template type from the VS.NET 2010 SDK). The main class that is at work here is *StrokesVsxPackage.VSX.VSIXPackage.cs*.

This class is loaded by Visual Studio, as soon as it encounters this extension. Please observe the attributes on this class, and their individual meaning:

```
[PackageRegistration(UseManagedResourcesOnly = true)]
[InstalledProductRegistration("#110", "#112", "1.0", IconResourceID = 400)]
[ProvideMenuResource("Menus.ctmenu", 1)]

[ProvideAutoLoad("{adfc4e64-0397-11d1-9f4e-00a0c911004f}")]

[ProvideToolWindow(typeof(AchievementStatisticsToolWindow), Style = VsDockStyle.MDI)]
[ProvideService(typeof(IAchevementLibraryService))]

[Guid(GuidList.guidCSharpAchiever_Achiever_VSIXPkgString)]
public sealed class StrokesVsxPackage : Package, IVsUpdateSolutionEvents2, IAchevementLibraryService
```

The first three are specifications of how resources are loaded (we only use Managed) and instructions to Visual Studio on which menu items this extension provides (Tools -> C# Achievements in this case).

The *ProvideAutoLoad*-Attribute informs Visual Studio about when the extension should be initialized. The GUID passed here means Visual Studio will initialize the extension as soon as Visual Studio is loaded, and before a Solution is present.

ProvideToolWindow-attribute informs Visual Studio that this extenson is hosting a Tools Window, of the type AchievementStatisticsToolWindow, with MDI-docking. MDI docking means the window will be docked in the Multiple Document Interface (full screen, like the start page).

ProvideService instructs Visual Studio, that this extension posts a “service” of the type IAchievementLibraryService. This is relevant for Achievement-projects that hosts their achievements inside Strokes.

Please be aware that technically this extension has no idea about achievements at all, they’re even with the base extension, provided by Strokes.BasicAchievements.VSX, which produces a seperate VSIX file containing only the achievements. These VSX projects use this service, to register them selves with Strokes.

The last attribute (Guid), is the Guid of the extension, and is a requirement for the VS.NET Extensions system.

Registration of Achievements

Achievement Registration is done “by Assembly”. If an Extension has a set of Achievement

it wants Strokes to notice, it needs to pass a reference to the assembly containing these to the primary strokes extension, via the interface IAchievementLibraryService, what defines one method: RegisterAchievementAssembly(Assembly assembly). Observe this method in *StrokesVsxPackage.VSX.VSIXPackage.cs.*

To see an example of how this interface is used to register an assembly into Strokes, see the project Strokes.BasicAchievements.VXS (file: Strokes.BasicAchievements.VSXPackage.cs, method: Initialize()):

```
protected override void Initialize()
{
        Trace.WriteLine (string.Format(CultureInfo.CurrentCulture, "Entering Initialize() of: {0}",
this.ToString()));
        base.Initialize();

        var als = GetService<IAchevementLibraryService>();
        if (als != null)
        {
                als.RegisterAchievementAssembly(Assembly.GetExecutingAssembly());
        }
}
```

The Achievement-containing extension will use the Visual Studio Service-infrastructure, to obtain a reference to the IAchievementLibraryService implementation ins Strokes.VSX.

Achievement Detection after successful build

Observe the *StrokesVsxPackage.VSX.VSIXPackage.cs*-file in Strokes.VSX, the method:

```
    int IVsUpdateSolutionEvents.UpdateSolution_Done(int fSucceeded, int fModified, int fCancelCommand)
      {
          if (fSucceeded != 0)
          {
              var activeDocument = dte.ActiveDocument;
              if(activeDocument != null)
              {
                  var documentFile = activeDocument.FullName;
                  if (documentFile.EndsWith(".cs"))
                  {
                      DetectionDispatcher.Dispatch(new BuildInformation()
                      {
                          ActiveFile = documentFile
                      });
                  }
              }
          }
          return VSConstants.S_OK;
      }
```

The IVsUpdateSolutionEvents.UpdateSolution_Done method is called whenever a solution has successfully been compiled. The method obtains a reference to the current document, and passes it's full name on to the DetectionDispatcher.Dispatch method (see: DetectionDispatcher.cs of Strokes.VSX).

Note: Currently this method only passes the currently open document. This is to be changed to pass a reference to all files involved in the compile.

Detection Dispatcher

The DetectionDispatcher class will do the following things:

1. Create a DetectionSession (used to cache objects between the achievement implementations in a well abstracted manner)
2. Obtain a collection of all implementations of Strokes.Core.Achievement (via the AchievementTracker-class of Strokes.Core. See method: FindAllAchievementTypes).
3. Obtain a list of the names of all achievements the user has already completed.
4. Iterate over all achievement-class implementations found with (2):
    a. Create an instance of the class with Activator (reflections)
    b. Obtain the descriptor (contains the achievements name)
        i. If the name of the achievement is already in the completedAchievements-collection, will move to the next element.
    c. Call the abstract DetectAchievement method on the Achievement-class implementation
        i. If this method returns true, the achievement is completed, and is added to a collection of completed achievements
    d. Move to next achievement
5. Call MainAchievementGui.DisplayAchievements with the collection of achievements completed during the compilation.
    a. The Strokes.GUI project will handle showing the user the "Achievement Unlocked"-box

AchievementTracker

This class resides in Strokes.Core, and is responsible for maintaining a list of all known achievements, all completed achievevements, and persist the list of completed achievements. See: AchievementTracker.cs in Strokes.Core

Authoring Achievements

To be written...