A simple plugin that consumes events

Creating a plugin that consumes events

To run this example you must first create a C# project that refrecnes FTC.SDKInterface.dll and Feat.Core.dll. The resulting dll from your project (along with any non .NET dependecies) must be placed in the Plugins directory in the FTC install location.

The Plugin class.

The first step is to create a class that implements the IEventListener interface. This is the type that will be instantiated by Scene Composer.

```
public class SimpleEventsConsumer : IEventListener
{
}
```

Implementing IPlugin

Next we must implement the IPluginService interface members.

```
public string Name
{
  get { return "SC event consumer plugin"; }
}

public string Description
{
   get { return "A plugin that consumes SC events and shows a message box on each event"; }
}

public string Version
{
  get { return "1.1.0"; }
}
public void Initialize(IUnityContainer unityContainer)
{
   //resolve the IDialogSvc interface. We will use it to show simple message boxes.
  dlgSvc = unityContainer.Resolve<IDialogSvc>();
}
```

Specifying what events should be handled by the plugin.

We implement the EventsToSubscribe property of the IEventListener interface. This property specifies a set of events that Scene Composer should pass to the plugin.

```
public EventId EventsToSubscribe
{
  get { return EventId.AppShutdown | EventId.AssetGeneration | EventId.ImportFinished | EventId
}
```

Implementing the event handlers of IEventListener

Next we implement event handler methods.

```
public void OnAssetGeneration(string assetFile, bool isTargetAsset, OperationStatus status)
{
  ShowMessage(String.Format("Asset generation for '{0}'  {1}.", assetFile, status));
}

public void OnImportFinished()
{
  ShowMessage("Import finished.");
}

public void OnShutdown()
{
  ShowMessage("App shutting down.");
}

public void OnSolutionChanged(FTC.SDKInterface.Solution.ISolution solution)
{
  ShowMessage("Solution changed!");
}
```

The ShowMessage method uses IDialogSvc to show a messagebox.

```
private void ShowMessage (string message)
{
  Application.Current.Dispatcher.BeginInvoke(new Action( () =>
  {
    dlgSvc.ShowMessageBox(message, "EventsPlugin");
  }), null );
}
```

The full code for the example:

SimpleEventsConsumer.cs

```
using Feat.CAL.Services;
using FTC.SDKInterface.Plugins;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;

namespace SCEventsPlugin
{
  public class SimpleEventsConsumer : IEventListener
  {
     private IDialogSvc dlgSvc;

     private void ShowMessage(string message)
     {
        Application.Current.Dispatcher.BeginInvoke(new Action(() =>
        {
           dlgSvc.ShowMessageBox(message, "EventsPlugin");
        }), null);
     }

     public EventId EventsToSubscribe
     {
        get { return EventId.AppShutdown | EventId.AssetGeneration | EventId.ImportFinished | E
     }

     public void OnAssetGeneration(string assetFile, bool isTargetAsset, OperationStatus status
     {
        ShowMessage(String.Format("Asset generation for '{0}'  {1}.", assetFile, status));
     }

     public void OnImportFinished()
     {
        ShowMessage("Import finished.");
     }

     public void OnShutdown()
     {
        ShowMessage("App shutting down.");
     }

     public void OnSolutionChanged(FTC.SDKInterface.Solution.ISolution solution)
     {
        ShowMessage("Solution changed!");
     }

     public string Description
     {
        get { return "A plugin that consumes SC events and shows a message box on each event";
     }

     public IEnumerable<int> FeatureIds
     {
        get { return new int[0]; }
     }
```

```
      public void Initialize(FTC.SDKInterface.Services.IUnityContainer unityContainer)
      {
         //resolve the IDialogSvc interface. We will use it to show a message boxes.
         dlgSvc = unityContainer.Resolve<IDialogSvc>();
      }

      public string Name
      {
         get { return "SC event consumer plugin"; }
      }

      public string Version
      {
         get { return "1.1.0"; }
      }
   }
}
```

Revision #1
Created 2023-02-15 07:58:44 UTC by Tsuyoshi.Kato
Updated 2023-02-15 07:59:21 UTC by Tsuyoshi.Kato