How to write plugins

Extractor plugins

All extractors are actually plugins that are bound to a syntax. Projbook engine will discover, load and callback snippets while processing snippet extraction.

In order to write a plugin you need to install Projbook.Extension from nuget after what you can implement the plugin interface:

```
/// <summary>
/// Defines interface for snippet extractor.
/// </summary>
[InheritedExport]
public interface ISnippetExtractor
{
    /// <summary>
    /// Defines the target type.
    /// </summary>
    TargetType TargetType { get; }

    /// <summary>
    /// Extracts a snippet.
    /// </summary>
    /// <param name="fileSystemInfo">The file system info.</param>
    /// <param name="pattern">The extraction pattern.</param>
    /// <returns>The extracted snippet.</returns>
    Snippet Extract(FileSystemInfoBase fileSystemInfo, string pattern);
}
```

You can reuse the default extractor implementation that will take case of the content loading and let you focus on your plugin:

```
/// <summary>
/// Extractor in charge of browsing source directories. load file content and extract requested member.
/// </summary>
[Syntax(name: "csharp")]
public class CSharpSnippetExtractor : DefaultSnippetExtractor
{
    // ...
}
```

> This plugin will be trigerred every time a code snippet is using the `csharp` syntax.

The `TargetType` will indicate Projbook what kind of validation needs to be applied on the snippet like file or folder existence and error reporting:

```
namespace Projbook.Extension.Spi
{
    /// <summary>
    /// Represents an extraction target.
    /// </summary>
    public enum TargetType
    {
        /// <summary>
        /// Free text target, used by plugins extracting from free value.
        /// </summary>
        FreeText,

        /// <summary>
        /// File target, used by plugins extracting from a file.
        /// </summary>
        File,

        /// <summary>
        /// Folder target, ised bu plugins extracting from a folder.
        /// </summary>
        Folder
    }
}
```

While implementing an extractor plugin you return an implementation of:

```
/// <summary>
/// Represents a snippet that has been extracted from source directories.
/// </summary>
public class Snippet
{
    // ...
}
```

When extracting text-based snippet like source code, you need to use the `PlainTextSnippet` implementation wraping the snippet content it will be injected in the code block:

```
/// <summary>
/// The text content.
/// </summary>
public readonly string Text;
```

When extracting tree-based snippets like file system, you need to use the `NodeSnippet` implementation wraping the tree-based structure and will be rendered using jstree:

```
/// <summary>
/// The node content.
/// </summary>
public readonly Node Node;
```

Plugins are loaded with MEF from the plugins directory:

- Projbook.1.1.0-cr2

> All plugins dependencies need to be packaged at the same place

Look at CSharp, Xml or FileSystem plugin source code for a full and detailed example.