# AppMan entries

If you want to add your plugin, theme or extension to AppMan, you need to do a pull request to the `appman.xml` file adding a new entry inside the FD5 comments. Requirements for the offered plugins, themes or extensions are:

\* Code needs to be open source and reviewable in a public repository

\* The item needs to be packaged into a `.fdz` file and provide a MD5 checksum for verification

\* The `.fdz` needs to extract the files to the automated plugin directory: `$(BaseDir)/Plugins`

# Coding style

##### Use spaces instead of tabs with a width of 4

```c#

public bool IsMember()

{

if (this.value > 2)

{

return false;

}

else return true;

}

```

##### Naming should be in English and clear

```c#

private int memberProperty = 0;

```

##### Use camelCase for private members and uppercase for public properties, methods and types:

```c#

private int memberProperty = 0;

public string MemberName = "MemberName";

public bool IsMember()

{

return true;

}

```

##### Use types without explicit path:

```c#

private void OnFormActivate(object sender, /\*System.\*/EventArgs e)

{

// Do something...

}

```

##### Do not use extensive extra empty lines and keep the code clean, not like:

```c#

// Comment...

private int MemberMethod(int value)

{

// Comment for something...

if (value > 2))

{

//this.oldCode = 0;

// Random notes here

return -1;

}

// Something here too...

else return value;

//Little bit here too...

}

```

##### Use brackets and parenthesis for easier readability:

```c#

if ((val1 > val2) && (val1 > val3))

{

if (val2 > val3)

{

doThing();

}

}

```

##### Do not nest expressions without brackets:

```c#

if (val1 > val2 && val1 > val3)

if (val2 > val3)

doThing();

```

##### Use can use one liners to shorten the code:

```c#

if (val1 > val2 && val1 > val3)

{

if (val2 > val3) doThing();

}

```

##### Use explicit types:

```c#

int myValue = 0;

Point[] myPoints = new Point[]

{

new Point(1, 1),

new Point(2, 2)

}

```

##### Code example:

```c#

using System;

namespace MyNameSpace

{

class MyClass

{

// Comment here...

private int memberProperty = 0;

private int memberProperty2 = 1;

private int memberProperty3 = 2;

// Comment here...

public string MemberName = "MemberName";

// Comment here...

public static bool IsMemberProperty = false;

// Comment here...

public const int CONSTANT = 1;

/// <summary>

/// Comment here...

/// </summary>

public bool IsMember()

{

return true;

}

/// <summary>

/// Comment here...

/// </summary>

public void MemberMethod(int value)

{

int temp = CONSTANT;

if (value > 2)

{

this.memberProperty2 = temp;

this.memberProperty3 = temp;

}

else this.memberProperty3 = value;

}

/// <summary>

/// Comment here...

/// </summary>

private int MemberMethodEx(int value)

{

int temp = CONSTANT;

this.memberProperty3 = temp;

switch (value)

{

case 1: return 1;

case 2:

{

return -1;

}

default: return value;

}

}

/// <summary>

/// Comment here...

/// </summary>

private void OnFormActivate(object sender, EventArgs e)

{

this.MemberMethod(null, null);

}

}

}

```

##### More generally, be consistent and keep the code clean!