

SplitSharp

Specification

Welcome to the Split# Specification
Here you will find information about the syntax of Split#

AT THE MOMENT WE ARE MASSIVELY WORKING ON THE SPECIFICATION TO FINISH IT AS FAST AS POSSIBLE. SO.. HELP US OR DIE

Split# is a programming language,
made for everyday work.

You can create amazing apps using Split#.

Split# allows you to create Console and Desktop Applications.
You can also create Libraries to extend the power of Split#.

Split# is for beginners, because it provides a very easy to learn syntax.
But it’s also for professionals because it allows you to do amazing things.

Split# supports Multithreading and direct memory access using pointers.

Building your own individual Application is done in just a minute.

There are two powerful IDEs for Split# that allows you to manage your Split# projects, debug your applications and optimize your code.

More about the IDEs in the downloads section.

- Specification
  - Split# Operators
    - Basic Operators
    - Assignment Operators
  - Predefined Variable types
  - Predefined Structures
  - Using the internal variables and constants
  - Defining constants
  - Importing a lib
  - Redirecting an imported lib (the whole lib) to another rootnamespace
  - Declaring a variable
  - Defining Meta Tags
  - Defining Methods
  - Defining Functions
  - Instantiating a class
  - Adding Event handlers

- Downloads
    - Split# DevEnv
    - ConnexIDE

- Documentation
  - Using the internal variables and constants
  - Defining constants
  - Importing a lib
  - Redirecting an imported lib (the whole lib) to another rootnamespace
  - Declaring a variable
  - Defining Meta Tags
  - Defining Functions
  - Instancing a class
  - Adding Event handlers
  - Library Syntax
  - Library tutorial
  - Syntax
  - Window tutorial
- Special Notes
- File Extensions
- License

Split# Operators

Basic Operators

`==`	Equality operator
`!`	Negation operator
`>>`	Bitwise shift right
`<<`	Bitwise shift left
`ror`	Bitwise rotation right
`rol`	Bitwise rotation left
`&`	Logical and
`|`	Logical or
`^`	Logical xor
`%`	Modulo
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division

Assignment Operators

=	Assignment operator
!=	Negate and assign
>>=	Bitwise shift right and assign
<<=	Bitwise shift left and assign
ror=	Bitwise rotate right and assign
rol=	Bitwise rotate left and assign
&=	Bitwise and and assign
|=	Bitwise or and assign
^=	Logical xor and assign
%=	Modulo and assign
+=	Add and assign
-=	Subtract and assign
*=	Multiply and assign
/=	Divide and assign
++	Increment by 1 and assign
--	Decrement by 1 and assign
**	Square and assign
//	Root and assign

Predefined Variable types

object

Predefined Structures

string	Usage: "text"
byte	Usage: <number>
int	Usage: <number>
float	Usage: <number.number>

Using the internal variables and constants

internal:[variable_name]<(parameters)>

Defining constants

const:[type] [NAME = "value"] #Name has to be uppercase

Importing a lib

@import [library] <from ["path"]>

Declaring a variable

Syntax:

var:<type> <name> <= ["value"]>

Defining Meta Tags

Syntax:

meta:[tagname] = "value"

Defining Functions

Syntax:

function:[type] [name] { }

Defining Structures

Syntax:

structure [name] { }

Using pointers
Syntax:

```
@ptr = value;
ref var; #returns address of var
```

Defining Functions
Syntax:

```
function:[type] [name] {
    return [variable of the same type as defined in the function] }
```

Instantiating a class
Syntax:

```
var:[type] = [type].New(<args>)
```

Declaring Delegates
Syntax:

```
delegate:[type] [name](args);
```

Downloads

Split# DevEnv

Splitty is working on it
The Split# Development Environment is a bunch of software that allows you to easily create amazing applications using Split#.

- Split# IDE
    - Create code from a template (Blank, Library, Window)
    - Manage your projects
    - Syntax highlighting and code folding
- Split# Compiler
    - The powerful Split# compiler
    - Fully included in the Split# IDE
    - Targeted in Viper 4.0

Viper Runtime environment
GruntTheDivine is working on it
Viper RE is a cross platform runtime environment written in C and is what Split# is targeted on.

ConnexIDE

zDimension is working on it

Connex is an IDE supporting Split#, written in C# by zDimension

- IDE
    - There are templates for all included languages :
        - Blank
        - Hello world!
        - Library
        - Window
- Editions
    - For beginners, you can use ConnexIDE <todo: create a name>.
    - For normal programmers, use the normal ConnexIDE.
    - For professionals, you can use ConnexIDE Pro.
    - For enterprise, use ConnexIDE Ultimate.



SplitSharp

Documentation

Welcome to the Split# documentation!
Here you will find examples and tutorials on how to use Split#.

Here is also a syntax information for each element of the Split# language.

Here in the documentation are more informations about the syntax and the use of it because the specification only describes the characteristics of the Split# language for developer who want to include Split# on one of their Projects or modify the language itself.

Using the internal variables and constants

Info:

    You cannot edit these variables and constants.

Syntax:

    internal:[variable_name]<(parameters)>

Examples :

    var:int = internal:random = A random int between 0 and 32767
    var:int = internal:random(100) = A random int between 0 and 100
    var:int = internal:random(<number>) = A random int between 0 and <number>

Defining constants

Info:

    Has to be before any code..

Syntax:

    const:[type] [NAME = "value"];

Examples:

    const:string HELLOWORLD = "Hello World Again!";
    const:int ANSWER_TO_EVERYTHING = 42;
    const:float MATH_PI = 3.14159265358979323;
    const:byte BYTE = 0x0F;
    const:color RED = @CLR(red);

Importing a lib

Syntax:

    @import [library] <from ["path"]>
    @import [library] <from ["path"]>

Examples:

    @import libCore #Rootnamespace imported automatically
    @import libCore from "C:\Libraries\"
    @import libCore

Declaring a variable

Syntax:

    var:<type> <name> <= ["value"]>

Examples:

```
var:string HelloWorld
var:string HelloWorld = "Hello World!"
var:int aNumber = 12
var:byte aByte = 0x0F
var:color red = @clr(red)
var:color light-green = @rgb(182,242,183)
var:color light-blue = 0x4DA6FF
var:int aNotherNumber = 0xDEADC0DE
```

Defining Meta Tags

Syntax:

```
meta:[tagname] = "value"
```

Examples:

```
meta:author = "Splitty"
meta:version = "1.0.35"
meta:copyright = "The Split# Team"
```

Defining Structures

Syntax:

```
structure [name]
{
    <variables>
}
```

Using pointers

Syntax:

```
#Write a byte to a address
@pointer = value;
#Get the value of a address
var:int AddressValue = @(address)
#Get a address
var:int aAddress = &aVariable;
```

Defining Functions

Syntax:

```
function:[type] [name]
{
	<code>
	return [variable]
}
```

Instancing a class

Syntax:

```
var:[type] = [type].New(<args>)
```

Example:

```
var:object = object.New(100)
```

Adding Event handlers

Sxntax:

```
delegate:void aHandler = function;
```

Example:

```
@import libCore use "libCore.Console"
@import libWin
var:Button btn = Button.New()
build()
{
	this->handler = printhello;
	btn->Text = "A simple button"
}
function:void printhello(var:EventArgs e)
{
	Console.PrintLn("You clicked the button!")
}
```

Split# Libraries

Here you can find informations about the optimal design of a Split# Lib.

Library Syntax

You can start a Library using the "library" keyword.
It tells the compiler that the code in this class is a library.
You cannot use access modifiers in a Library, the Compiler will ignore them.

```
library libExample
{
	base()
	{
		<meta informations>
	}
	<library methods>
}
```

Library tutorial

Ok. Let's begin with the basic Split# Library design.
In this little tutorial we will build a basic module that allows you to print a colored text.

Code:

```
/* libCore.Modules:
 *	Gain access to the internal variables that defines a Module
 * libCore.Console:
 *	Needed to use the In and Out streams
 */
@import libCore #Import needed libraries
library libExample
{
	base()
	{
		#Add the basic information to the Module
		meta:author = "Your Name";
		meta:name = "libExample";
		meta:description = "A short info about what the lib does";
		meta:version = "1.0.0";
	}
	function:void PrintLine(var:string aString, var:color aColor)
```

```
	{
		#Backup the current foreground color
		var:color backup = Console.Foreground
		#Set the new foreground color
		Console.Foreground = aColor
		#Print the text
		println(aString)

	}
}
#You just created a library!
#For a print code just copy the PrintLine method and replace
#Console.Printn(aString) with Console.Print(aString)
```

Special Notes

Note: You may use the internally implemented function kenneth() to awesomize your code. It is VERY complex. Don't even try to understand it; attempting to do so may cause death, mutilation or other bodily harm. Split# MAY NOT BE responsible for any death related to reading this code. Please speak to our lawyers for more information, but sadly they are very busy; too busy to respond most of the time, but they try to at least read your queries.

Special thanks to:
Aurora01 for a contributing great ideas.
Kenneth "The awesomizer" Head for being awesome.
zDimension/zippedfire for syntax highlighting, some very good ideas and samples.
GruntTheDivine for being awesome and writing so much compiler stuff

You have found the awesome magical sword of awesome Kenneth. Use it wisely (and awesomely). Very wisely (and awesomely).
Beware of the (not awesome) VB.NET programmers inside the (not awesome) forest, as they will try to (evilly) enforce VB.NET onto you.
If you (awesomely) wield the (awesome) magical sword of Kenneth, then you will (awesomely) defeat the (not awesome) VB.NET programmers. But if you indeed fail, there is (not awesomely) no (awesome) coming back from the grave.
Good (awesome) luck on your (awesomely awesome) adventure and you (awesomely) better be (awesomely) on your way, as the (not awesome) night will (not awesomely) soon come (not awesomely) upon us.

File Extensions

SSC: SplitSharp Code
SSF: SplitSharp File
SSRC or SSRX: SplitSharp Resource

License

Split# is licensed under the WTFPL license

DO WHAT THE FUCK YOU WANT TO PUBLIC LICENSE
Version 2, December 2004

Copyright (C) 2004 Sam Hocevar <sam@hocevar.net>

Everyone is permitted to copy and distribute verbatim or modified
copies of this license document, and changing it is allowed as long
as the name is changed.

DO WHAT THE FUCK YOU WANT TO PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Copyright © 2013 Split# Developers