

Open Source XML & JSON Visualisation Software

1

Generated by Doxygen 1.8.13

Mon Apr 17 2017 19:42:52



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Packages . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Namespace Documentation</b>	<b>7</b>
4.1	OSXJV Namespace Reference . . . . .	7
4.2	OSXJV.Classes Namespace Reference . . . . .	7
4.3	OSXJV.Server Namespace Reference . . . . .	7
4.4	WebServer Namespace Reference . . . . .	8
<b>5</b>	<b>Class Documentation</b>	<b>9</b>
5.1	OSXJV.Classes.Attribute Class Reference . . . . .	9
5.1.1	Detailed Description . . . . .	9
5.1.2	Member Data Documentation . . . . .	10
5.1.2.1	name . . . . .	10
5.1.2.2	value . . . . .	10
5.1.3	Property Documentation . . . . .	10
5.1.3.1	Name . . . . .	10
5.1.3.2	Value . . . . .	10
5.2	OSXJV.Classes.CacheManager Class Reference . . . . .	11

5.2.1	Detailed Description	12
5.2.2	Constructor & Destructor Documentation	12
5.2.2.1	CacheManager()	12
5.2.3	Member Function Documentation	12
5.2.3.1	Close()	12
5.2.3.2	getFile()	12
5.2.3.3	GetInstance()	13
5.2.3.4	saveFile()	14
5.2.3.5	Setup()	15
5.2.4	Member Data Documentation	16
5.2.4.1	inst	16
5.2.4.2	path	16
5.3	OSXJV.Classes.Logger Class Reference	16
5.3.1	Detailed Description	17
5.3.2	Constructor & Destructor Documentation	17
5.3.2.1	Logger()	17
5.3.3	Member Function Documentation	18
5.3.3.1	Close()	18
5.3.3.2	GetInstance()	18
5.3.3.3	Setup()	19
5.3.3.4	WriteError()	19
5.3.4	Member Data Documentation	20
5.3.4.1	inst	20
5.3.4.2	location	21
5.4	OSXJV.Classes.Node Class Reference	21
5.4.1	Detailed Description	22
5.4.2	Constructor & Destructor Documentation	22
5.4.2.1	Node()	22
5.4.3	Member Data Documentation	23
5.4.3.1	attributes	23

5.4.3.2	children	23
5.4.3.3	comments	23
5.4.3.4	name	23
5.4.3.5	number	23
5.4.3.6	value	24
5.4.3.7	visited	24
5.4.4	Property Documentation	24
5.4.4.1	Attributes	24
5.4.4.2	Children	24
5.4.4.3	Comments	25
5.4.4.4	Name	25
5.4.4.5	Number	25
5.4.4.6	Value	25
5.4.4.7	Visited	26
5.5	OSXJV.Server.OSXJVServer Class Reference	26
5.5.1	Detailed Description	28
5.5.2	Constructor & Destructor Documentation	28
5.5.2.1	OSXJVServer()	28
5.5.3	Member Function Documentation	28
5.5.3.1	GetData()	28
5.5.3.2	GetFileData()	29
5.5.3.3	GetFormData()	30
5.5.3.4	HandleClient()	31
5.5.3.5	HandleGet()	31
5.5.3.6	HandleOptions()	33
5.5.3.7	HandlePost()	34
5.5.3.8	ListenerCallback()	36
5.5.3.9	Post()	37
5.5.3.10	Run()	38
5.5.3.11	SaveFile()	39

5.5.3.12	SegmentNormalize()	40
5.5.3.13	Start()	40
5.5.3.14	Stop()	41
5.5.4	Member Data Documentation	41
5.5.4.1	listener	41
5.5.4.2	port	41
5.5.4.3	running	41
5.5.4.4	serverThread	42
5.6	OSXJV.Classes.Output Class Reference	42
5.6.1	Detailed Description	44
5.6.2	Constructor & Destructor Documentation	44
5.6.2.1	Output()	44
5.6.3	Member Function Documentation	45
5.6.3.1	CheckChildren() [1/2]	45
5.6.3.2	CheckChildren() [2/2]	46
5.6.3.3	CheckNodeNumber()	47
5.6.3.4	CreateExtraNode() [1/2]	48
5.6.3.5	CreateExtraNode() [2/2]	49
5.6.3.6	CreateGrid()	49
5.6.3.7	CreateNodeChildViewsParallel()	50
5.6.3.8	CreateNodeView() [1/2]	51
5.6.3.9	CreateNodeView() [2/2]	52
5.6.3.10	CreatePreviousNode()	53
5.6.3.11	CreateView()	54
5.6.3.12	CreateViewSingle()	56
5.6.3.13	GetParent()	57
5.6.3.14	GridGetChidren()	58
5.6.4	Member Data Documentation	58
5.6.4.1	cNodes	59
5.6.4.2	GotParent	59

5.6.4.3	left	59
5.6.4.4	nodes	59
5.6.4.5	Parent	59
5.6.4.6	top	60
5.7	OSXJV.Classes.ProcessDocument Class Reference	60
5.7.1	Detailed Description	62
5.7.2	Constructor & Destructor Documentation	63
5.7.2.1	ProcessDocument()	63
5.7.3	Member Function Documentation	63
5.7.3.1	GetProcess()	63
5.7.3.2	Prepare()	64
5.7.3.3	Process()	65
5.7.3.4	ProcessComment()	65
5.7.3.5	ProcessDocumentParallelInit()	66
5.7.3.6	ProcessElement() [1/2]	67
5.7.3.7	ProcessElement() [2/2]	68
5.7.3.8	ProcessParallel()	69
5.7.3.9	ProcessRoot()	70
5.7.3.10	ProcessText()	71
5.7.4	Member Data Documentation	72
5.7.4.1	count	72
5.7.4.2	document	72
5.7.4.3	node	72
5.7.4.4	ProcessedElements	72
5.7.4.5	th	73
5.7.4.6	ThreadList	73
5.7.4.7	type	73
5.8	WebServer.Program Class Reference	73
5.8.1	Detailed Description	74
5.8.2	Member Function Documentation	74

5.8.2.1	Main()	74
5.9	OSXJV.Classes.Request Class Reference	75
5.9.1	Detailed Description	77
5.9.2	Constructor & Destructor Documentation	77
5.9.2.1	Request()	77
5.9.3	Member Function Documentation	77
5.9.3.1	GetRequest()	77
5.9.4	Member Data Documentation	78
5.9.4.1	data	78
5.9.4.2	filename	79
5.9.4.3	type	79
5.9.5	Property Documentation	79
5.9.5.1	Data	79
5.9.5.2	Filename	79
5.9.5.3	Type	80
5.10	OSXJV.Classes.Response Class Reference	80
5.10.1	Detailed Description	81
5.10.2	Constructor & Destructor Documentation	81
5.10.2.1	Response()	81
5.10.3	Member Function Documentation	82
5.10.3.1	GetErrorResponse()	82
5.10.3.2	GetInvalidRequestResponse()	83
5.10.3.3	GetResponse()	83
5.10.3.4	GetResponseJSON()	84
5.10.3.5	GetResponseXML()	85
5.10.4	Member Data Documentation	85
5.10.4.1	data	86
5.10.4.2	mime	86
5.10.4.3	status	86
5.11	OSXJV.Classes.Validation Class Reference	86
5.11.1	Detailed Description	87
5.11.2	Constructor & Destructor Documentation	87
5.11.2.1	Validation()	87
5.11.3	Member Function Documentation	87
5.11.3.1	CheckDocument()	87
5.11.3.2	GetInstance()	89
5.11.4	Member Data Documentation	89
5.11.4.1	inst	89



<b>6 File Documentation</b>	<b>91</b>
6.1 OSXJVMClasses/Attribute.cs File Reference	91
6.2 Attribute.cs	91
6.3 OSXJVMClasses/CacheManager.cs File Reference	92
6.4 CacheManager.cs	92
6.5 OSXJVMClasses/Logger.cs File Reference	93
6.6 Logger.cs	93
6.7 OSXJVMClasses/Node.cs File Reference	94
6.8 Node.cs	94
6.9 OSXJVMClasses/Output.cs File Reference	96
6.10 Output.cs	96
6.11 OSXJVMClasses/ProcessDocument.cs File Reference	103
6.12 ProcessDocument.cs	103
6.13 OSXJVMClasses/Request.cs File Reference	107
6.14 Request.cs	107
6.15 OSXJVMClasses/Response.cs File Reference	108
6.16 Response.cs	109
6.17 OSXJVMClasses/Validation.cs File Reference	110
6.18 Validation.cs	110
6.19 OSXJVServer.cs File Reference	111
6.20 OSXJVServer.cs	111
6.21 Program.cs File Reference	115
6.22 Program.cs	116
6.23 Properties/AssemblyInfo.cs File Reference	117
6.24 AssemblyInfo.cs	117
<b>Index</b>	<b>119</b>



# Chapter 1

## Namespace Index

### 1.1 Packages

Here are the packages with brief descriptions (if available):

<a href="#">OSXJV</a>	7
<a href="#">OSXJV.Classes</a>	7
<a href="#">OSXJV.Server</a>	7
<a href="#">WebServer</a>	8



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">OSXJV.Classes.Attribute</a>	
9	
<a href="#">OSXJV.Classes.CacheManager</a>	
Manages Saving an Retrieving Filesexi . . . . .	11
<a href="#">OSXJV.Classes.Logger</a>	
A simple class that writes errors to a single file. . . . .	16
<a href="#">OSXJV.Classes.Node</a>	
Contain Processed Document Information . . . . .	21
<a href="#">OSXJV.Server.OSXJVServer</a>	
HTTPServer that process the incoming requests. . . . .	26
<a href="#">OSXJV.Classes.Output</a>	
Creates the <a href="#">Output</a> for the web page to display. . . . .	42
<a href="#">OSXJV.Classes.ProcessDocument</a>	
Class the Processes the document . . . . .	60
<a href="#">WebServer.Program</a>	
The Initialiser . . . . .	73
<a href="#">OSXJV.Classes.Request</a>	
A object containing the document to process, filename and type. . . . .	75
<a href="#">OSXJV.Classes.Response</a>	
The Object containing data to send to the client . . . . .	80
<a href="#">OSXJV.Classes.Validation</a>	
Preform validation . . . . .	86



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">OSXJVServer.cs</a>	111
<a href="#">Program.cs</a>	115
<a href="#">OSXJVClasses/Attribute.cs</a>	91
<a href="#">OSXJVClasses/CacheManager.cs</a>	92
<a href="#">OSXJVClasses/Logger.cs</a>	93
<a href="#">OSXJVClasses/Node.cs</a>	94
<a href="#">OSXJVClasses/Output.cs</a>	96
<a href="#">OSXJVClasses/ProcessDocument.cs</a>	103
<a href="#">OSXJVClasses/Request.cs</a>	107
<a href="#">OSXJVClasses/Response.cs</a>	108
<a href="#">OSXJVClasses/Validation.cs</a>	110
<a href="#">Properties/AssemblyInfo.cs</a>	117





## Chapter 4

# Namespace Documentation

### 4.1 OSXJV Namespace Reference

#### Namespaces

- namespace [Classes](#)
- namespace [Server](#)

### 4.2 OSXJV.Classes Namespace Reference

#### Classes

- class [Attribute](#)
- class [CacheManager](#)  
*Manages Saving an Retrieving Filesexi*
- class [Logger](#)  
*A simple class that writes errors to a single file.*
- class [Node](#)  
*Contain Processed Document Information*
- class [Output](#)  
*Creates the [Output](#) for the web page to display.*
- class [ProcessDocument](#)  
*Class the Processes the document*
- class [Request](#)  
*A object containing the document to process, filename and type.*
- class [Response](#)  
*The Object containing data to send to the client*
- class [Validation](#)  
*Preform validation*

### 4.3 OSXJV.Server Namespace Reference

#### Classes

- class [OSXJVServer](#)  
*HTTPServer that process the incoming requests.*

## 4.4 WebServer Namespace Reference

### Classes

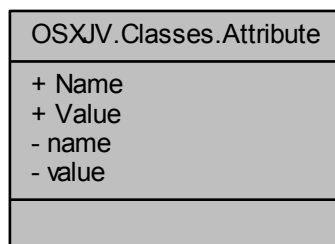
- class [Program](#)  
*The Initialiser*

## Chapter 5

# Class Documentation

### 5.1 OSXJV.Classes.Attribute Class Reference

Collaboration diagram for OSXJV.Classes.Attribute:



#### Properties

- string [Name](#) [get, set]
- string [Value](#) [get, set]

#### Private Attributes

- string [name](#)
- string [value](#)

#### 5.1.1 Detailed Description

Definition at line 6 of file [Attribute.cs](#).

## 5.1.2 Member Data Documentation

### 5.1.2.1 name

```
string OSXJV.Classes.Attribute.name [private]
```

Definition at line 8 of file [Attribute.cs](#).

### 5.1.2.2 value

```
string OSXJV.Classes.Attribute.value [private]
```

Definition at line 9 of file [Attribute.cs](#).

## 5.1.3 Property Documentation

### 5.1.3.1 Name

```
string OSXJV.Classes.Attribute.Name [get], [set]
```

Definition at line 15 of file [Attribute.cs](#).

Referenced by [OSXJV.Classes.Output.CreateNodeView\(\)](#), [OSXJV.Classes.ProcessDocument.ProcessElement\(\)](#), and [OSXJV.Classes.ProcessDocument.ProcessRoot\(\)](#).

### 5.1.3.2 Value

```
string OSXJV.Classes.Attribute.Value [get], [set]
```

Definition at line 30 of file [Attribute.cs](#).

Referenced by [OSXJV.Classes.Output.CreateNodeView\(\)](#), [OSXJV.Classes.ProcessDocument.ProcessElement\(\)](#), and [OSXJV.Classes.ProcessDocument.ProcessRoot\(\)](#).

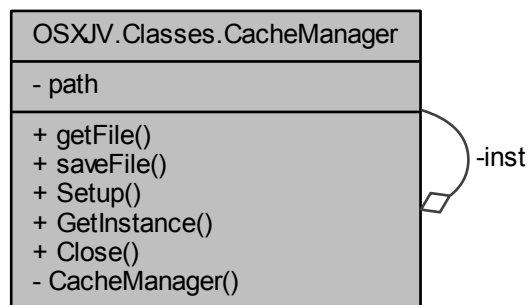
The documentation for this class was generated from the following file:

- [OSXJVCClasses/Attribute.cs](#)

## 5.2 OSXJV.Classes.CacheManager Class Reference

Manages Saving an Retrieving Filesexi

Collaboration diagram for OSXJV.Classes.CacheManager:



### Public Member Functions

- string [getFile](#) (string ID)  
*Retrieve the file from caching*
- bool [saveFile](#) (string ID, string nodes)  
*Save the file to the local system for caching*

### Static Public Member Functions

- static bool [Setup](#) (string [path](#))
- static [CacheManager GetInstance](#) ()  
*Get the single instance of the class*
- static void [Close](#) ()

### Private Member Functions

- [CacheManager](#) (string [path](#))

### Private Attributes

- string [path](#)

### Static Private Attributes

- static [CacheManager inst](#)

### 5.2.1 Detailed Description

Manages Saving an Retrieving Filesexi

Definition at line 9 of file [CacheManager.cs](#).

### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 CacheManager()

```
OSXJV.Classes.CacheManager.CacheManager (
    string path ) [private]
```

Definition at line 14 of file [CacheManager.cs](#).

```
00015         {
00016             this.path = path;
00017         }
```

### 5.2.3 Member Function Documentation

#### 5.2.3.1 Close()

```
static void OSXJV.Classes.CacheManager.Close ( ) [static]
```

Definition at line 99 of file [CacheManager.cs](#).

```
00100         {
00101             if (inst == null)
00102                 throw new Exception("CacheManager Already Closed");
00103             else
00104                 inst = null;
00105         }
```

#### 5.2.3.2 getFile()

```
string OSXJV.Classes.CacheManager.getFile (
    string ID )
```

Retrieve the file from caching

## Parameters

<i>ID</i>	Unique ID of the file
-----------	-----------------------

## Returns

Definition at line 51 of file [CacheManager.cs](#).

Referenced by [OSXJV.Server.OSXJVServer.HandleGet\(\)](#).

```

00052     {
00053         if (string.IsNullOrEmpty(ID))
00054             throw new ArgumentException("ID cannot be null or empty");
00055
00056         string filePath = path + "/" + ID.Replace("/", "") + ".json";
00057         string output = "";
00058
00059         using (StreamReader sr = new StreamReader(filePath))
00060         {
00061             output = sr.ReadToEnd();
00062         }
00063
00064         if (!string.IsNullOrEmpty(output))
00065             return output;
00066         else
00067             throw new Exception("Error Reading From File");
00068     }

```

Here is the caller graph for this function:



## 5.2.3.3 GetInstance()

```
static CacheManager OSXJV.Classes.CacheManager.GetInstance ( ) [static]
```

Get the single instance of the class

**Returns**

An instance of [CacheManager](#)

Definition at line 38 of file [CacheManager.cs](#).

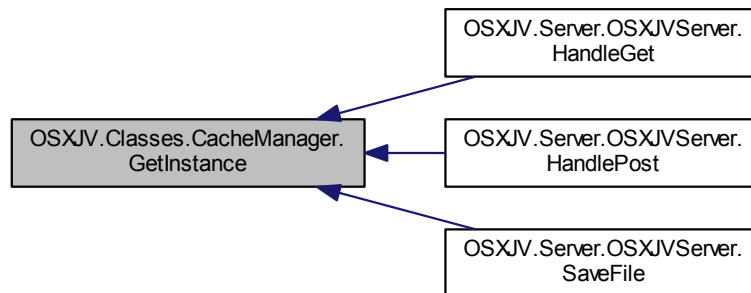
Referenced by [OSXJV.Server.OSXJVServer.HandleGet\(\)](#), [OSXJV.Server.OSXJVServer.HandlePost\(\)](#), and [OSXJV.Server.OSXJVServer.SaveFile\(\)](#).

```

00039         {
00040             if (inst != null)
00041                 return inst;
00042             else
00043                 throw new Exception("CacheManger has not been setup");
00044         }

```

Here is the caller graph for this function:

**5.2.3.4 saveFile()**

```

bool OSXJV.Classes.CacheManager.saveFile (
    string ID,
    string nodes )

```

Save the file to the local system for caching

**Parameters**

<i>ID</i>	Unique ID of the file
<i>nodes</i>	The document to be saved

Definition at line 75 of file [CacheManager.cs](#).

Referenced by [OSXJV.Server.OSXJVServer.HandlePost\(\)](#), and [OSXJV.Server.OSXJVServer.SaveFile\(\)](#).

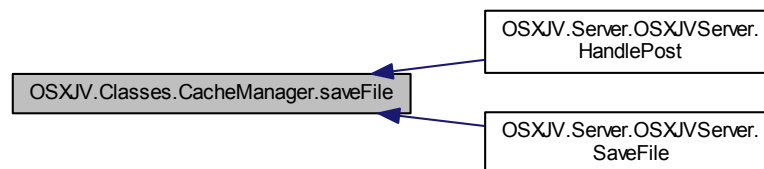


```

00076     {
00077         if (string.IsNullOrEmpty(ID))
00078             throw new ArgumentException("ID cannot be null or empty");
00079
00080         if (string.IsNullOrEmpty(nodes))
00081             throw new ArgumentException("Document cannot be null or empty");
00082
00083         string filePath = path + "/" + ID + ".json";
00084         try
00085         {
00086             using (StreamWriter sw = new StreamWriter(filePath))
00087             {
00088                 sw.WriteLine(nodes);
00089             }
00090         }
00091         catch
00092         {
00093             throw new Exception("Failed to save file");
00094         }
00095
00096         return true;
00097     }

```

Here is the caller graph for this function:



### 5.2.3.5 Setup()

```

static bool OSXJV.Classes.CacheManager.Setup (
    string path ) [static]

```

#### Parameters

<i>path</i>	
-------------	--

Definition at line 23 of file [CacheManager.cs](#).

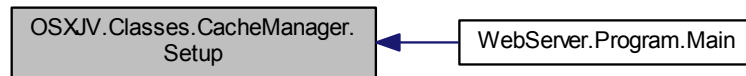
Referenced by [WebServer.Program.Main\(\)](#).

```

00024     {
00025         if (string.IsNullOrEmpty(path))
00026             throw new ArgumentException("Path cannot be empty");
00027
00028         if (!Directory.Exists(string.Format(@"{0}", path)))
00029             throw new Exception("Path is not a valid cache directory");
00030
00031         return (inst = new CacheManager(path)) != null ? true : false;
00032     }

```

Here is the caller graph for this function:



## 5.2.4 Member Data Documentation

### 5.2.4.1 inst

`CacheManager` OSXJV.Classes.CacheManager.inst [static], [private]

Definition at line 11 of file [CacheManager.cs](#).

### 5.2.4.2 path

`string` OSXJV.Classes.CacheManager.path [private]

Definition at line 12 of file [CacheManager.cs](#).

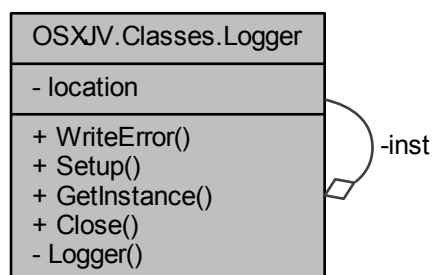
The documentation for this class was generated from the following file:

- OSXJVClasses/[CacheManager.cs](#)

## 5.3 OSXJV.Classes.Logger Class Reference

A simple class that writes errors to a single file.

Collaboration diagram for OSXJV.Classes.Logger:



## Public Member Functions

- void [WriteError](#) (string error)  
*Writes an error the location provided*

## Static Public Member Functions

- static bool [Setup](#) (string [location](#))
- static [Logger](#) [GetInstance](#) ()  
*Gets the single instance of [Logger](#)*
- static void [Close](#) ()

## Private Member Functions

- [Logger](#) (string [location](#))

## Private Attributes

- string [location](#)

## Static Private Attributes

- static [Logger](#) [inst](#)  
*Singleton instance of [Logger](#)*

### 5.3.1 Detailed Description

A simple class that writes errors to a single file.

Definition at line 9 of file [Logger.cs](#).

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 [Logger\(\)](#)

```
OSXJV.Classes.Logger.Logger (
    string location ) [private]
```

Definition at line 17 of file [Logger.cs](#).

```
00018         {
00019             this.location = location;
00020         }
```

### 5.3.3 Member Function Documentation

#### 5.3.3.1 Close()

static void OSXJV.Classes.Logger.Close ( ) [static]

Definition at line 72 of file [Logger.cs](#).

```
00073     {
00074         if (inst == null)
00075             throw new Exception("Logger Already Closed");
00076         else
00077             inst = null;
00078     }
```

#### 5.3.3.2 GetInstance()

static [Logger](#) OSXJV.Classes.Logger.GetInstance ( ) [static]

Gets the single instance of [Logger](#)

##### Returns

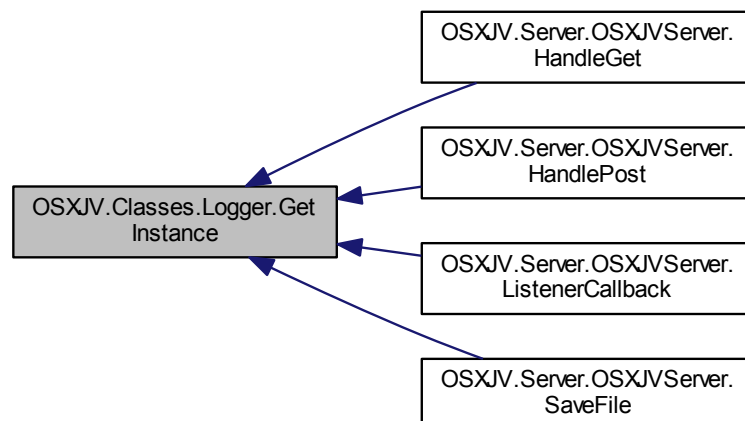
Instance of [Logger](#)

Definition at line 41 of file [Logger.cs](#).

Referenced by [OSXJV.Server.OSXJVServer.HandleGet\(\)](#), [OSXJV.Server.OSXJVServer.HandlePost\(\)](#), [OSXJV.Server.OSXJVServer.ListenerCallback\(\)](#), and [OSXJV.Server.OSXJVServer.SaveFile\(\)](#).

```
00042     {
00043         if (inst != null)
00044             return inst;
00045         else
00046             throw new Exception("Logger has not been setup");
00047     }
```

Here is the caller graph for this function:



## 5.3.3.3 Setup()

```
static bool OSXJV.Classes.Logger.Setup (
    string location ) [static]
```

## Parameters

<i>location</i>
-----------------

Definition at line 26 of file [Logger.cs](#).

Referenced by [WebServer.Program.Main\(\)](#).

```
00027     {
00028         if (string.IsNullOrEmpty(location))
00029             throw new ArgumentException("Location cannot be empty");
00030
00031         if (!Directory.Exists(string.Format(@"{0}", location)))
00032             throw new Exception("Location is not a valid logger directory");
00033
00034         return (inst = new Logger(location)) != null ? true:false;
00035     }
```

Here is the caller graph for this function:



## 5.3.3.4 WriteError()

```
void OSXJV.Classes.Logger.WriteError (
    string error )
```

Writes an error the location provided

## Parameters

<i>error</i>	The error message
--------------	-------------------

Definition at line 53 of file [Logger.cs](#).

Referenced by [OSXJV.Server.OSXJVServer.HandleGet\(\)](#), [OSXJV.Server.OSXJVServer.HandlePost\(\)](#), [OSXJV.Server.OSXJVServer.ListenerCallback\(\)](#), and [OSXJV.Server.OSXJVServer.SaveFile\(\)](#).

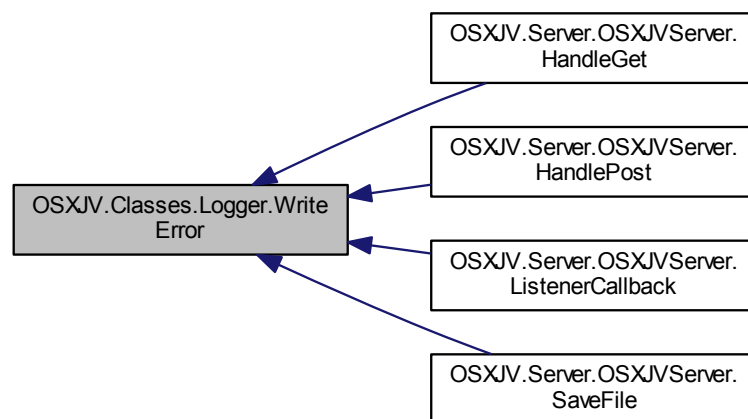
```
00054     {
```

```

00055         try
00056         {
00057             if (!string.IsNullOrEmpty(error))
00058             {
00059                 string file = string.Format(@"{0}/Error-{1}.txt", location, DateTime.Now.
ToString("dd-MM-yy hh-MM-ss"));
00060                 StreamWriter sw = new StreamWriter(file);
00061                 sw.WriteLine(error);
00062                 sw.WriteLine();
00063                 sw.Close();
00064             }
00065         }
00066         catch (IOException e)
00067         {
00068             throw e;
00069         }
00070     }

```

Here is the caller graph for this function:



### 5.3.4 Member Data Documentation

#### 5.3.4.1 inst

`Logger` OSXJV.Classes.Logger.inst [static], [private]

Singleton instance of `Logger`

Definition at line 14 of file `Logger.cs`.

#### 5.3.4.2 location

```
string OSXJV.Classes.Logger.location [private]
```

Definition at line 15 of file [Logger.cs](#).

The documentation for this class was generated from the following file:

- OSXJVClasses/[Logger.cs](#)

## 5.4 OSXJV.Classes.Node Class Reference

Contain Processed Document Information

Collaboration diagram for OSXJV.Classes.Node:



### Public Member Functions

- [Node](#) ()

*Constructor*

## Properties

- int [Number](#) [get, set]  
The Number of the [Node](#)
- string [Name](#) [get, set]  
The Name of [Node](#)
- string [Value](#) [get, set]  
The Value of the [Node](#)
- List< string > [Comments](#) [get, set]  
Comments That the [Node](#) Has.
- List< [Attribute](#) > [Attributes](#) [get, set]  
Attributes the [Node](#) has.
- List< [Node](#) > [Children](#) [get, set]  
Children Nodes the [Node](#) is linked to.
- bool [Visited](#) [get, set]  
If the node has been visited previous by the [ProcessDocument](#), prevent multiple same Nodes.

## Private Attributes

- string [name](#)
- List< [Attribute](#) > [attributes](#)
- string [value](#)
- List< [Node](#) > [children](#)
- int [number](#)
- bool [visited](#)
- List< string > [comments](#)

### 5.4.1 Detailed Description

Contain Processed Document Information

Definition at line 10 of file [Node.cs](#).

### 5.4.2 Constructor & Destructor Documentation

#### 5.4.2.1 Node()

OSXJV.Classes.Node.Node ( )

Constructor

Definition at line 23 of file [Node.cs](#).

```

00024     {
00025         Attributes = new List<Attribute>();
00026         Children = new List<Node>();
00027         Comments = new List<string>();
00028         number = 0;
00029         visited = false;
00030     }

```



### 5.4.3 Member Data Documentation

#### 5.4.3.1 attributes

`List<Attribute> OSXJV.Classes.Node.attributes [private]`

Definition at line 13 of file [Node.cs](#).

#### 5.4.3.2 children

`List<Node> OSXJV.Classes.Node.children [private]`

Definition at line 15 of file [Node.cs](#).

#### 5.4.3.3 comments

`List<string> OSXJV.Classes.Node.comments [private]`

Definition at line 18 of file [Node.cs](#).

#### 5.4.3.4 name

`string OSXJV.Classes.Node.name [private]`

Definition at line 12 of file [Node.cs](#).

#### 5.4.3.5 number

`int OSXJV.Classes.Node.number [private]`

Definition at line 16 of file [Node.cs](#).

#### 5.4.3.6 value

```
string OSXJV.Classes.Node.value [private]
```

Definition at line 14 of file [Node.cs](#).

#### 5.4.3.7 visited

```
bool OSXJV.Classes.Node.visited [private]
```

Definition at line 17 of file [Node.cs](#).

### 5.4.4 Property Documentation

#### 5.4.4.1 Attributes

```
List<Attribute> OSXJV.Classes.Node.Attributes [get], [set]
```

Attributes the [Node](#) has.

Definition at line 102 of file [Node.cs](#).

Referenced by [OSXJV.Classes.Output.CreateNodeView\(\)](#), [OSXJV.Classes.ProcessDocument.ProcessElement\(\)](#), and [OSXJV.Classes.ProcessDocument.ProcessRoot\(\)](#).

#### 5.4.4.2 Children

```
List<Node> OSXJV.Classes.Node.Children [get], [set]
```

Children Nodes the [Node](#) is linked to.

Definition at line 119 of file [Node.cs](#).

Referenced by [OSXJV.Classes.Output.CheckChildren\(\)](#), [OSXJV.Classes.Output.CreateGrid\(\)](#), [OSXJV.Classes.Output.CreateView\(\)](#), [OSXJV.Classes.Output.CreateViewSingle\(\)](#), [OSXJV.Classes.Output.GetParent\(\)](#), [OSXJV.Classes.Output.GridGetChidren\(\)](#), [OSXJV.Classes.ProcessDocument.ProcessDocumentParallelInit\(\)](#), [OSXJV.Classes.ProcessDocument.ProcessElement\(\)](#), and [OSXJV.Classes.ProcessDocument.ProcessParallel\(\)](#).

#### 5.4.4.3 Comments

```
List<string> OSXJV.Classes.Node.Comments [get], [set]
```

Comments That the [Node](#) Has.

Definition at line 85 of file [Node.cs](#).

Referenced by [OSXJV.Classes.Output.CreateNodeChildViewsParallel\(\)](#), [OSXJV.Classes.Output.CreateNodeView\(\)](#), and [OSXJV.Classes.ProcessDocument.ProcessComment\(\)](#).

#### 5.4.4.4 Name

```
string OSXJV.Classes.Node.Name [get], [set]
```

The Name of [Node](#)

Definition at line 52 of file [Node.cs](#).

Referenced by [OSXJV.Classes.Output.CreateGrid\(\)](#), [OSXJV.Classes.Output.CreateNodeView\(\)](#), [OSXJV.Classes.Output.GridGetChidren\(\)](#), [OSXJV.Classes.ProcessDocument.ProcessElement\(\)](#), and [OSXJV.Classes.ProcessDocument.ProcessRoot\(\)](#).

#### 5.4.4.5 Number

```
int OSXJV.Classes.Node.Number [get], [set]
```

The Number of the [Node](#)

Definition at line 36 of file [Node.cs](#).

Referenced by [OSXJV.Classes.Output.CheckChildren\(\)](#), [OSXJV.Classes.Output.CheckNodeNumber\(\)](#), [OSXJV.Classes.Output.CreateGrid\(\)](#), [OSXJV.Classes.Output.CreateNodeView\(\)](#), [OSXJV.Classes.Output.CreateView\(\)](#), [OSXJV.Classes.Output.CreateViewSingle\(\)](#), [OSXJV.Classes.Output.GetParent\(\)](#), [OSXJV.Classes.Output.GridGetChidren\(\)](#), [OSXJV.Classes.ProcessDocument.ProcessElement\(\)](#), and [OSXJV.Classes.ProcessDocument.ProcessRoot\(\)](#).

#### 5.4.4.6 Value

```
string OSXJV.Classes.Node.Value [get], [set]
```

The Value of the [Node](#)

Definition at line 69 of file [Node.cs](#).

Referenced by [OSXJV.Classes.Output.CreateNodeView\(\)](#), and [OSXJV.Classes.ProcessDocument.ProcessText\(\)](#).

#### 5.4.4.7 Visited

```
bool OSXJV.Classes.Node.Visited [get], [set]
```

If the node has been visited previous by the [ProcessDocument](#), prevent multiple same Nodes.

Definition at line 136 of file [Node.cs](#).

Referenced by [OSXJV.Classes.ProcessDocument.ProcessElement\(\)](#), and [OSXJV.Classes.ProcessDocument.ProcessRoot\(\)](#).

The documentation for this class was generated from the following file:

- [OSXJVClasses/Node.cs](#)

## 5.5 OSXJV.Server.OSXJVServer Class Reference

HTTPServer that process the incoming requests.

Collaboration diagram for OSXJV.Server.OSXJVServer:

OSXJV.Server.OSXJVServer
+ running - port - listener - serverThread
+ OSXJVServer() + Start() + Stop() + Run() + GetFormData() - ListenerCallback() - HandleClient() - HandleOptions() - GetFileData() - HandlePost() - HandleGet() - SaveFile() - Post() - GetData() - SegmentNormalize()

## Public Member Functions

- [OSXJVServer](#) ()  
*The [Server](#) Handler*
- bool [Start](#) ()  
*Starts server in new thread*
- bool [Stop](#) ()  
*Stop the listener and about all current requests*
- void [Run](#) ()  
*Function that constantly listens for connections*
- [Request GetFormData](#) (Stream input)  
*Extract the files from the request*

## Static Public Attributes

- static bool [running](#) = false  
*True if the server is able to accept requests.*

## Private Member Functions

- void [ListenerCallback](#) (IAsyncResult result)  
*Handles Requests Asynchronously*
- void [HandleClient](#) (HttpListenerContext c)  
*Handles the client*
- void [HandleOptions](#) (HttpListenerResponse response)  
*Sends to the Client What the [Server](#) Supports*
- [Request GetFileData](#) (Stream input, string type)  
*Get Data if the data is retrieved*
- [Response HandlePost](#) (HttpListenerRequest req)  
*Handles a POST request.*
- [Response HandleGet](#) (HttpListenerRequest req)  
*Handles a GET request.*
- void [SaveFile](#) (string id, [Node](#) nodes)  
*Save data recieved from client.*
- void [Post](#) ([Response](#) res, HttpListenerResponse stream)  
*Send data to the client.*
- [Request GetData](#) (HttpListenerRequest req)  
*Get the data from the client.*
- string [SegmentNormalize](#) (string input)  
*Removes '/' from the string.*

## Private Attributes

- int [port](#) = 8082
- HttpListener [listener](#)
- Thread [serverThread](#) = null

### 5.5.1 Detailed Description

HTTPServer that process the incoming requests.

Definition at line 16 of file [OSXJVServer.cs](#).

### 5.5.2 Constructor & Destructor Documentation

#### 5.5.2.1 OSXJVServer()

```
OSXJV.Server.OSXJVServer.OSXJVServer ( )
```

The [Server](#) Handler

Definition at line 34 of file [OSXJVServer.cs](#).

```
00035     {
00036         listener = new HttpListener();
00037         listener.Prefixes.Add("http://localhost:" + port + "/"); //change if need be
00038     }
```

### 5.5.3 Member Function Documentation

#### 5.5.3.1 GetData()

```
Request OSXJV.Server.OSXJVServer.GetData (
    HttpListenerRequest req ) [private]
```

Get the data from the client.

##### Parameters

<i>req</i>	The request from the client
------------	-----------------------------

##### Returns

A Request Object

Definition at line 413 of file [OSXJVServer.cs](#).

```
00414     {
00415         Request r = null;
00416
00417         if (req.ContentType.Contains("application/x-www-form-urlencoded"))
00418         {
```

```

00419         r = GetFormData(req.InputStream);
00420     }
00421     else if (req.ContentType.Contains("application/json") || req.ContentType.Contains("
application/oclet-stream"))
00422     {
00423         r = GetFileData(req.InputStream, "application/json");
00424     }
00425     else if (req.ContentType.Contains("application/xml") || req.ContentType.Contains("text/xml"))
00426     {
00427         r = GetFileData(req.InputStream, "text/xml");
00428     }
00429     return r;
00430 }

```

### 5.5.3.2 GetFileData()

**Request** OSXJV.Server.OSXJVServer.GetFileData (   
     Stream *input*,   
     string *type* ) [private]

Get Data if the data is retrieved

#### Parameters

<i>input</i>	Client Stream Input
<i>type</i>	The MIME type

#### Returns

A Response object to send to the user

Definition at line 209 of file [OSXJVServer.cs](#).

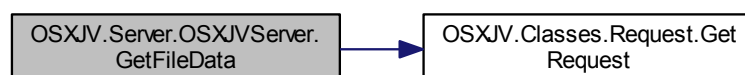
References [OSXJV.Classes.Request.GetRequest\(\)](#).

```

00210     {
00211         string request = "";
00212         using (StreamReader ms = new StreamReader(input))
00213         {
00214             request = ms.ReadToEnd();
00215         }
00216         string filename = "temp";
00217
00218         if (type == "text/xml")
00219             filename += ".xml";
00220         else if (type == "application/json")
00221             filename += ".json";
00222         else
00223             filename += ".html";
00224
00225         return Request.GetRequest(filename, type, request);
00226     }

```

Here is the call graph for this function:



### 5.5.3.3 GetFormData()

`Request` OSXJV.Server.OSXJVServer.GetFormData (   
 Stream *input* )

Extract the files from the request

#### Parameters

<i>input</i>	Requests input stream
--------------	-----------------------

#### Returns

New Request Object

#### Exceptions

<i>System.InvalidOperationException</i>	Thrown when no files are included with the request
---	--

Definition at line 185 of file [OSXJVServer.cs](#).

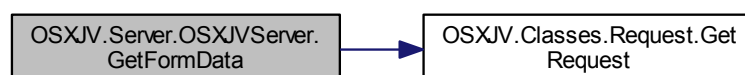
References [OSXJV.Classes.Request.GetRequest\(\)](#).

```

00186     {
00187         string request = "";
00188         MultipartFormDataParser parser = new MultipartFormDataParser(input);
00189         if (parser.Files.Count > 0)
00190         {
00191             using (StreamReader ms = new StreamReader(parser.Files[0].Data))
00192             {
00193                 request = ms.ReadToEnd();
00194             }
00195         }
00196         else
00197         {
00198             throw new InvalidOperationException();
00199         }
00200         return Request.GetRequest(parser.Files[0].FileName, parser.Files[0].
00201             ContentType, request);
00201     }

```

Here is the call graph for this function:





## 5.5.3.4 HandleClient()

```
void OSXJV.Server.OSXJVServer.HandleClient (
    HttpListenerContext c ) [private]
```

Handles the client

## Parameters

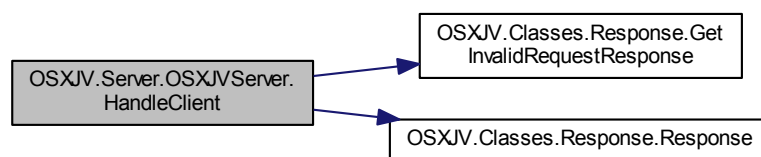
c	The Request
---	-------------

Definition at line 146 of file [OSXJVServer.cs](#).

References [OSXJV.Classes.Response.GetInvalidRequestResponse\(\)](#), and [OSXJV.Classes.Response.Response\(\)](#).

```
00147     {
00148         switch (c.Request.HttpMethod)
00149         {
00150             case "POST":
00151                 Post (HandlePost (c.Request), c.Response);
00152                 break;
00153             case "GET":
00154                 Post (HandleGet (c.Request), c.Response);
00155                 break;
00156             case "OPTIONS":
00157                 HandleOptions (c.Response);
00158                 c.Response.Close();
00159                 break;
00160             default:
00161                 Post (Response.GetInvalidRequestResponse(), c.
00162                     Response);
00163                 break;
00164         }
00165     }
```

Here is the call graph for this function:



## 5.5.3.5 HandleGet()

```
Response OSXJV.Server.OSXJVServer.HandleGet (
    HttpListenerRequest req ) [private]
```

Handles a GET request.

## Parameters

<i>req</i>	The request to be processed.
------------	------------------------------

## Returns

A Response object to send to the user

Definition at line 312 of file [OSXJVServer.cs](#).

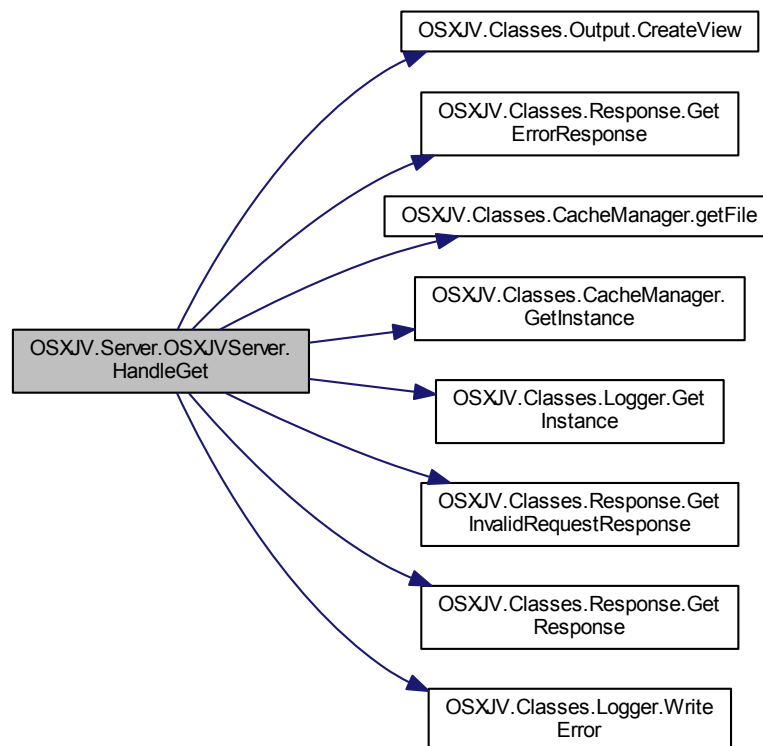
References [OSXJV.Classes.Output.CreateView\(\)](#), [OSXJV.Classes.Response.GetErrorResponse\(\)](#), [OSXJV.Classes.CacheManager.getFile\(\)](#), [OSXJV.Classes.CacheManager.GetInstance\(\)](#), [OSXJV.Classes.Logger.GetInstance\(\)](#), [OSXJV.Classes.Response.GetInvalidRequestResponse\(\)](#), [OSXJV.Classes.Response.GetResponse\(\)](#), and [OSXJV.Classes.Logger.WriteError\(\)](#).

```

00313     {
00314         if (SegmentNormalize(req.Url.Segments[1]).Equals("Process"))
00315         {
00316             if (req.Url.Segments.Length == 4)
00317             {
00318
00319                 Node cached;
00320                 try
00321                 {
00322                     cached = JsonConvert.DeserializeObject<Node>(
CacheManager.GetInstance().getFile(req.Url.Segments[2]));
00323                 }
00324                 catch (Exception e)
00325                 {
00326                     Logger.GetInstance().WriteError(e.Message);
00327                     JObject eRes = new JObject();
00328                     eRes.Add("Error", "Error Creating Response");
00329                     return Response.GetErrorResponse(eRes.ToString());
00330                 }
00331                 Output o = new Output(cached);
00332                 JObject response = new JObject();
00333                 response.Add("view", o.CreateView(int.Parse(req.Url.Segments[3])));
00334                 byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
00335                 return Response.GetResponse(200, "application/json", bytes);
00336             }
00337             else if (req.Url.Segments.Length == 5)
00338             {
00339
00340                 Node cached;
00341                 try
00342                 {
00343                     cached = JsonConvert.DeserializeObject<Node>(
CacheManager.GetInstance().getFile(req.Url.Segments[2]));
00344                 }
00345                 catch (Exception e)
00346                 {
00347                     Logger.GetInstance().WriteError(e.Message);
00348                     JObject eRes = new JObject();
00349                     eRes.Add("Error", "Error Creating Response");
00350                     return Response.GetErrorResponse(eRes.ToString());
00351                 }
00352                 Output o = new Output(cached);
00353                 JObject response = new JObject();
00354                 response.Add("view", o.CreateView(int.Parse(
SegmentNormalize(req.Url.Segments[3]), 4, int.Parse(
SegmentNormalize(req.Url.Segments[4]))));
00355                 byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
00356                 return Response.GetResponse(200, "application/json", bytes);
00357             }
00358             else
00359                 return Response.GetInvalidRequestResponse();
00360         }
00361         //If it got here its an invalid response.
00362         return Response.GetInvalidRequestResponse();
00363     }

```

Here is the call graph for this function:



#### 5.5.3.6 HandleOptions()

```
void OSXJV.Server.OSXJVServer.HandleOptions (
    HttpListenerResponse response ) [private]
```

Sends to the Client What the [Server](#) Supports

##### Parameters

<i>response</i>	The Request Response Object
-----------------	-----------------------------

Definition at line 170 of file [OSXJVServer.cs](#).

```
00171     {
00172         response.AddHeader("Access-Control-Allow-Headers", "Content-Type, Accept, X-Requested-With");
00173         response.AddHeader("Access-Control-Allow-Methods", "POST");
00174         response.AddHeader("Access-Control-Allow-Methods", "GET");
00175         response.AddHeader("Access-Control-Max-Age", "1728000");
00176         response.AppendHeader("Access-Control-Allow-Origin", "*");
00177     }
```

### 5.5.3.7 HandlePost()

```
Response OSXJV.Server.OSXJVServer.HandlePost (
    HttpListenerRequest req ) [private]
```

Handles a POST request.

#### Parameters

<i>req</i>	The request to be processed.
------------	------------------------------

#### Returns

A Response object to send to the user

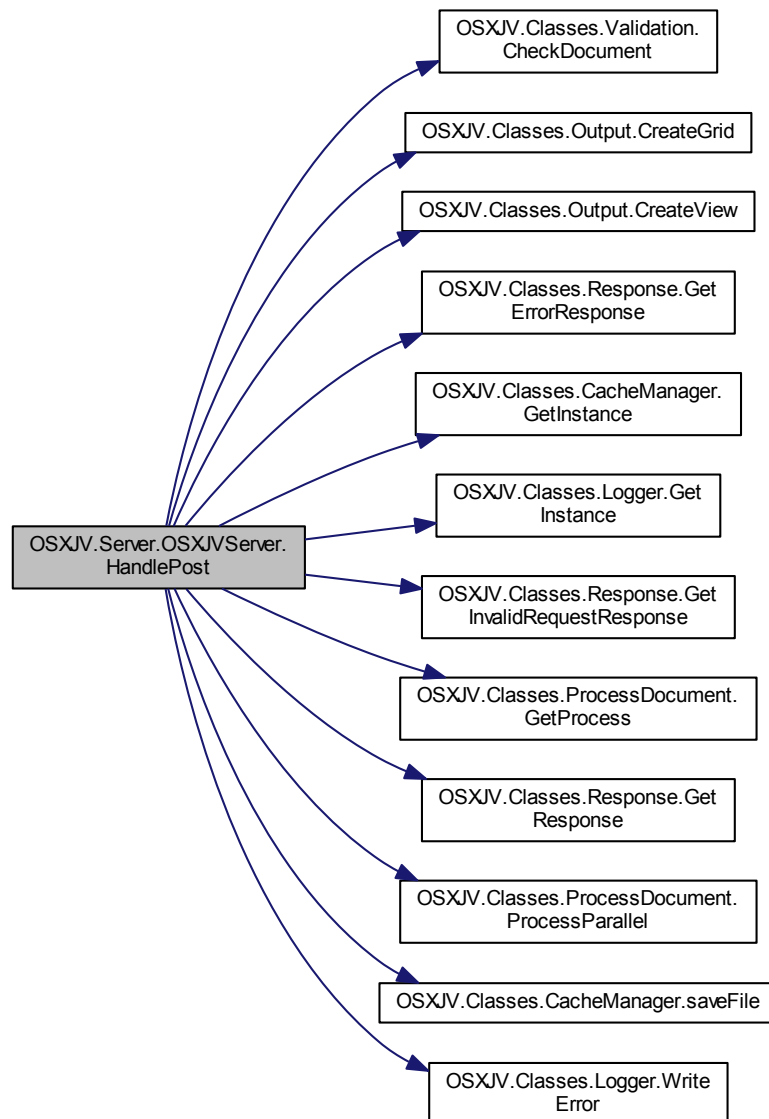
Definition at line 233 of file [OSXJVServer.cs](#).

References [OSXJV.Classes.Validation.CheckDocument\(\)](#), [OSXJV.Classes.Output.CreateGrid\(\)](#), [OSXJV.Classes.Output.CreateView\(\)](#), [OSXJV.Classes.Request.Data](#), [OSXJV.Classes.Response.GetErrorResponse\(\)](#), [OSXJV.Classes.CacheManager.GetInstance\(\)](#), [OSXJV.Classes.Logger.GetInstance\(\)](#), [OSXJV.Classes.Response.GetInvalidRequestResponse\(\)](#), [OSXJV.Classes.ProcessDocument.GetProcess\(\)](#), [OSXJV.Classes.Response.GetResponse\(\)](#), [OSXJV.Classes.ProcessDocument.ProcessParallel\(\)](#), [OSXJV.Classes.CacheManager.saveFile\(\)](#), [OSXJV.Classes.Request.Type](#), and [OSXJV.Classes.Logger.WriteError\(\)](#).

```
00234     {
00235
00236         JObject eRes = new JObject();
00237
00238         if (SegmentNormalize(req.RawUrl).Equals("Process"))
00239         {
00240             if (req.HasEntityBody)
00241             {
00242
00243
00244                 Request r = null;
00245                 try
00246                 {
00247                     r = GetData(req);
00248                     if (r == null)
00249                         return Response.GetInvalidRequestResponse();
00250                 }
00251                 catch
00252                 {
00253                     return Response.GetInvalidRequestResponse();
00254                 }
00255
00256
00257                 try
00258                 {
00259                     Validation.CheckDocument(r.Data, r.
00260 Type);
00261                 }
00262                 catch (Exception e)
00263                 {
00264                     eRes.Add("Error", e.Message);
00265                     return Response.GetErrorResponse(eRes.ToString());
00266                 }
00267
00268                 string id = Guid.NewGuid().ToString();
00269                 ProcessDocument pro = ProcessDocument.
00270 GetProcess(r.Data, r.Type);
00271                 Node n = pro.ProcessParallel();
00272                 Output o = new Output(n); //new output object
00273                 try
00274                 {
00275                     CacheManager.GetInstance().
00276 saveFile(id, JsonConvert.SerializeObject(n));
00277                     JObject response = new JObject();
00278
00279                     n = null; //remove node as its completed;
```

```
00278
00279         response.Add("filename", id);
00280         response.Add("grid", o.CreateGrid());
00281         response.Add("view", o.CreateView());
00282
00283
00284
00285         byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
00286         return Response.GetResponse(200, "application/json", bytes);
00287     }
00288     catch (Exception e)
00289     {
00290         Logger.GetInstance().WriteError(e.Message);
00291         eRes.Add("Error", "Error Creating Response");
00292         return Response.GetErrorResponse(eRes.ToString());
00293     }
00294
00295     }
00296     eRes.Add("Error", "No File Recieved By Server");
00297     return Response.GetErrorResponse(eRes.ToString());
00298 }
00299 else if (req.RawUrl.Equals("/Output"))
00300 {
00301     return Response.GetInvalidRequestResponse();
00302 }
00303 else
00304     return Response.GetInvalidRequestResponse();
00305 }
```

Here is the call graph for this function:



#### 5.5.3.8 ListenerCallback()

```
void OSXJV.Server.OSXJVServer.ListenerCallback (
    IAsyncResult result ) [private]
```

Handles Requests Asynchronously

##### Parameters

<i>result</i>	The Request Object Coming In.
---------------	-------------------------------

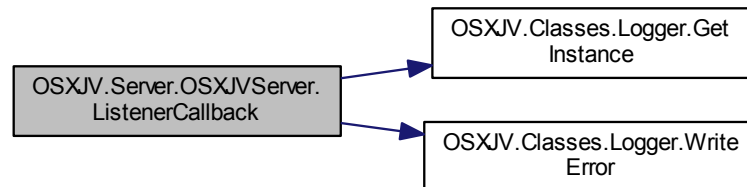
Definition at line 124 of file [OSXJVServer.cs](#).

References [OSXJV.Classes.Logger.GetInstance\(\)](#), and [OSXJV.Classes.Logger.WriteError\(\)](#).

```

00125     {
00126         HttpListener listener = (HttpListener)result.AsyncState;
00127         HttpListenerContext context = listener.EndGetContext(result);
00128         try
00129         {
00130             HandleClient(context);
00131         }
00132         catch (Exception e)
00133         {
00134             Logger.GetInstance().WriteError(e.Message);
00135             context.Response.StatusCode = 500;
00136             context.Response.Close();
00137         }
00138     }
00139 }
```

Here is the call graph for this function:



#### 5.5.3.9 Post()

```

void OSXJV.Server.OSXJVServer.Post (
    Response res,
    HttpListenerResponse stream ) [private]
```

Send data to the client.

##### Parameters

<i>res</i>	The Response Object
<i>stream</i>	The Client Output Stream

///

##### Exceptions

<i>ArgumentException</i>	Thrown when Response is null or HttpListenerResponse is null or empty
--------------------------	---

Definition at line 394 of file [OSXJVServer.cs](#).

References [OSXJV.Classes.Response.data](#), [OSXJV.Classes.Response.mime](#), and [OSXJV.Classes.Response.status](#).

```

00395     {
00396         if (res == null || stream == null)
00397             throw new ArgumentException("Response or Client Stream cannot be NULL");
00398
00399         HandleOptions(stream);
00400         stream.ProtocolVersion = new Version(1, 1);
00401         stream.StatusCode = res.status;
00402         stream.ContentType = res.mime;
00403         stream.ContentLength64 = res.data.Length;
00404         stream.OutputStream.Write(res.data, 0, res.data.Length);
00405         stream.Close();
00406     }

```

### 5.5.3.10 Run()

```
void OSXJV.Server.OSXJVServer.Run ( )
```

Function that constantly listens for connections

Definition at line 76 of file [OSXJVServer.cs](#).

```

00077     {
00078         running = true;
00079         listener.Start();
00080
00081
00082         while(listener.IsListening)
00083         {
00084
00085             Console.WriteLine("Waiting");
00086
00087             //Wait for Listener
00088             IAsyncResult result = listener.BeginGetContext(new AsyncCallback(
ListenerCallback), listener);
00089             result.AsyncWaitHandle.WaitOne();
00090
00091             if (result.CompletedSynchronously)
00092                 Console.WriteLine("Completed Synchronously");
00093
00094             /*
00095              * Old Method of Creating a Thread
00096              */
00097             Thread response = new Thread(() =>
00098             {
00099                 try
00100                 {
00101                     Console.WriteLine("Processing");
00102                     HandleClient(hlc);
00103
00104                     Console.WriteLine("Finished");
00105                 }
00106                 catch(Exception e)
00107                 {
00108                     Logger.GetInstance().WriteError(e.Message);
00109                     hlc.Response.StatusCode = 500;
00110                     hlc.Response.Close();
00111                 }
00112             });
00113             response.Start();
00114             *
00115             */
00116         }
00117     }

```



## 5.5.3.11 SaveFile()

```
void OSXJV.Server.OSXJVServer.SaveFile (
    string id,
    Node nodes ) [private]
```

Save data recieved from client.

## Parameters

<i>id</i>	Unique ID
<i>nodes</i>	The Processed Data

## Exceptions

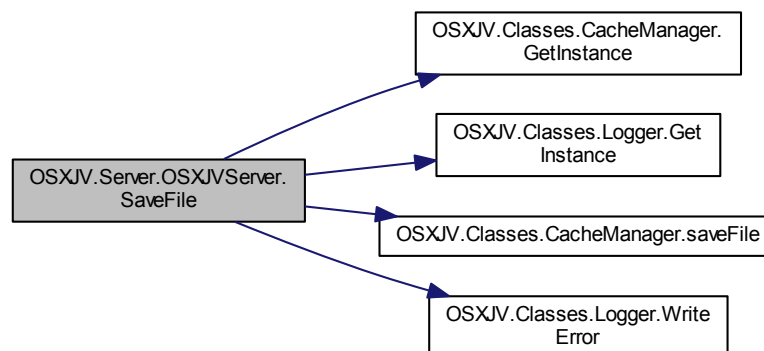
<i>ArgumentException</i>	Thrown when nodes is null or id is null or empty
--------------------------	--

Definition at line 371 of file [OSXJVServer.cs](#).

References [OSXJV.Classes.CacheManager.GetInstance\(\)](#), [OSXJV.Classes.Logger.GetInstance\(\)](#), [OSXJV.Classes.CacheManager.saveFile\(\)](#), and [OSXJV.Classes.Logger.WriteError\(\)](#).

```
00372     {
00373         if(nodes == null || string.IsNullOrEmpty(id))
00374         {
00375             throw new ArgumentException();
00376         }
00377         try
00378         {
00379             CacheManager.GetInstance().saveFile(id, JsonConvert.
00380 SerializeObject(nodes));
00381         }
00382         catch (Exception e)
00383         {
00384             Logger.GetInstance().WriteError(e.Message);
00385         }
00386     }
```

Here is the call graph for this function:



### 5.5.3.12 SegmentNormalize()

```
string OSXJV.Server.OSXJVServer.SegmentNormalize (
    string input ) [private]
```

Removes '/' from the string.

#### Parameters

<i>input</i>	A string from the URL
--------------	-----------------------

#### Returns

Normalised String

Definition at line [437](#) of file [OSXJVServer.cs](#).

```
00438     {
00439         return input.Replace("/", "");
00440     }
```

### 5.5.3.13 Start()

```
bool OSXJV.Server.OSXJVServer.Start ( )
```

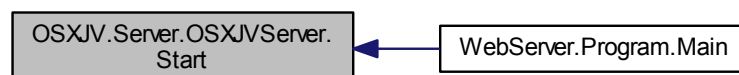
Starts server in new thread

Definition at line [43](#) of file [OSXJVServer.cs](#).

Referenced by [WebServer.Program.Main\(\)](#).

```
00044     {
00045         serverThread = new Thread(new ThreadStart(Run));
00046         try
00047         {
00048             serverThread.Start();
00049         }
00050         catch
00051         {}
00052         return serverThread.IsAlive;
00053     }
```

Here is the caller graph for this function:



#### 5.5.3.14 Stop()

```
bool OSXJV.Server.OSXJVServer.Stop ( )
```

Stop the listener and about all current requests

Definition at line 58 of file [OSXJVServer.cs](#).

```
00059     {
00060         if (listener != null)
00061             if (listener.IsListening)
00062                 listener.Abort();
00063
00064
00065         if (serverThread != null)
00066         {
00067             serverThread.Join();
00068             serverThread = null;
00069         }
00070
00071         return serverThread == null ? true : false;
00072     }
```

### 5.5.4 Member Data Documentation

#### 5.5.4.1 listener

```
HttpListener OSXJV.Server.OSXJVServer.listener [private]
```

Definition at line 24 of file [OSXJVServer.cs](#).

#### 5.5.4.2 port

```
int OSXJV.Server.OSXJVServer.port = 8082 [private]
```

Definition at line 18 of file [OSXJVServer.cs](#).

#### 5.5.4.3 running

```
bool OSXJV.Server.OSXJVServer.running = false [static]
```

True if the server is able to accept requests.

Definition at line 23 of file [OSXJVServer.cs](#).

#### 5.5.4.4 serverThread

```
Thread OSXJV.Server.OSXJVServer.serverThread = null [private]
```

Definition at line 29 of file [OSXJVServer.cs](#).

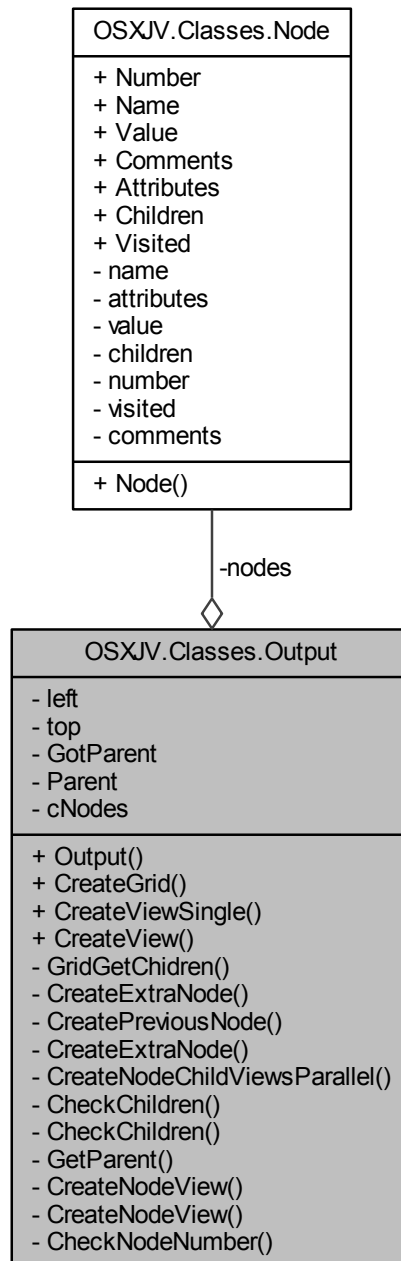
The documentation for this class was generated from the following file:

- [OSXJVServer.cs](#)

## 5.6 OSXJV.Classes.Output Class Reference

Creates the [Output](#) for the web page to display.

Collaboration diagram for OSXJV.Classes.Output:



## Public Member Functions

- [Output \(Node nodes\)](#)  
*Creation of a [Output](#) object.*
- [JObject CreateGrid \(\)](#)  
*Creates the grid data.*
- [string CreateViewSingle](#) (int node, int nodeStart=0)

*CreateView using a Single Thread*

- string [CreateView](#) (int node=1, int pCount=4, int nodeStart=0)

*Creates the view of nodes using multiple threads.*

## Private Member Functions

- JObject [GridGetChidren](#) ([Node](#) n)  
*Recursive function to get all the nodes data for the grid .*
- string [CreateExtraNode](#) (string type, int id)  
*Builds a get more button to display*
- string [CreatePreviousNode](#) (string type, int leftVal, int topVal, int id)  
*Create a previous node button*
- string [CreateExtraNode](#) (string type, int leftVal, int topVal, int id)  
*Create a extra node button*
- void [CreateNodeChildViewsParallel](#) (List< [Node](#) > job, int start, bool showHigher, int next, int previous)  
*Generate [Output](#) HTML when using multi-threads*
- string [CheckChildren](#) ([Node](#) n, int number)  
*Check child nodes if the are to be part of the output.*
- string [CheckChildren](#) ([Node](#) n, int number, int pCount, int nodeStart, ref bool found)
- void [GetParent](#) ([Node](#) node, int number)  
*Finds the parent node.*
- string [CreateNodeView](#) ([Node](#) n, string type, int leftVal, int topVal)  
*Generates HTML for the Specific [Node](#) (Multi-Threaded Version)*
- string [CreateNodeView](#) ([Node](#) n, string type)  
*Generates HTML for specific [Node](#) (Single Threaded Version)*
- bool [CheckNodeNumber](#) ([Node](#) n, int number)  
*Checks if [Node](#) number and inputted number match.*

## Private Attributes

- int [left](#) = 100
- int [top](#) = 130
- [Node](#) [nodes](#)
- bool [GotParent](#) = false
- int [Parent](#) = 0  
*Parent of node when building output (Used when getting [Node](#) other than root).*
- List< Tuple< int, string > > [cNodes](#) = new List<Tuple<int, string>>()  
*Used in Threading, list of calculated HTML strings.*

### 5.6.1 Detailed Description

Creates the [Output](#) for the web page to display.

Definition at line 11 of file [Output.cs](#).

### 5.6.2 Constructor & Destructor Documentation

#### 5.6.2.1 Output()

```
OSXJV.Classes.Output.Output (
    Node nodes )
```

Creation of a [Output](#) object.

## Parameters

<i>nodes</i>	A processed object of Nodes
--------------	-----------------------------

Definition at line 42 of file [Output.cs](#).

```

00043     {
00044         if (nodes == null)
00045             throw new ArgumentException();
00046         this.nodes = nodes;
00047     }

```

### 5.6.3 Member Function Documentation

#### 5.6.3.1 CheckChildren() [1/2]

```

string OSXJV.Classes.Output.CheckChildren (
    Node n,
    int number ) [private]

```

Check child nodes if the are to be part of the output.

## Parameters

<i>n</i>	<a href="#">Node</a> to search
<i>number</i>	Number to check

## Returns

String of calculated HTML

Definition at line 399 of file [Output.cs](#).

References [OSXJV.Classes.Node.Children](#), and [OSXJV.Classes.Node.Number](#).

```

00400     {
00401         string output = "";
00402         if (CheckNodeNumber(n, number))
00403         {
00404             int count = 0;
00405             output += CreateNodeView(n, "node");
00406             foreach (Node n2 in n.Children)
00407             {
00408                 count++;
00409                 output += CreateNodeView(n2, "node-child");
00410             }
00411         }
00412         else if (n.Children.Count > 0)
00413         {
00414             foreach (Node n2 in n.Children)
00415             {
00416                 if (GotParent)
00417                 {
00418                     if (n2.Number == Parent)
00419                     {
00420

```

```

00421             output += CreateNodeView(n2, "node-parent");
00422         }
00423     }
00424     output += CheckChildren(n2, number);
00425 }
00426 }
00427 }
00428     return output;
00429 }

```

### 5.6.3.2 CheckChildren() [2/2]

```

string OSXJV.Classes.Output.CheckChildren (
    Node n,
    int number,
    int pCount,
    int nodeStart,
    ref bool found ) [private]

```

#### Parameters

<i>n</i>	
<i>number</i>	
<i>pCount</i>	
<i>nodeStart</i>	
<i>found</i>	

#### Returns

String of calculated HTML

Definition at line 440 of file [Output.cs](#).

References [OSXJV.Classes.Node.Children](#), and [OSXJV.Classes.Node.Number](#).

```

00441     {
00442         string output = "";
00443         if (CheckNodeNumber(n, number))
00444         {
00445             found = true;
00446             List<Thread> threadList = new List<Thread>();
00447
00448             int count = 0;
00449             output += CreateNodeView(n, "node");
00450             count++;
00451             //output += CreateNodeView(n2, "node-child");
00452             int childCount = 0;
00453
00454             if (n.Children.Count < 200)
00455                 childCount = n.Children.Count;
00456             else
00457             {
00458                 childCount = 200;
00459             }
00460             if (childCount < pCount * 2)
00461             {
00462                 foreach(Node n2 in n.Children)
00463                 {
00464                     output += CreateNodeView(n2, "node-child");
00465                 }
00466             }
00467             else

```



```

00468         {
00469             int spread = (int)Math.Ceiling((double)childCount / (double)pCount);
00470
00471             if (childCount > 0)
00472             {
00473                 for (int i = 0; i < pCount; i++)
00474                 {
00475                     int neg = 0;
00476                     if ((spread * (i + 1)) > childCount)
00477                     {
00478                         neg = childCount - (spread * (i + 1));
00479                     }
00480                     int start = (spread * i);
00481                     int rangeStart = (spread * i) + nodeStart;
00482                     bool showHigher = nodeStart != 0 ? true : false;
00483
00484                     List<Node> NodesToProcess = n.Children.GetRange(rangeStart, spread + neg);
00485
00486                     if (NodesToProcess.Count > 0)
00487                     {
00488                         Thread threadJob = new Thread(() =>
00489 CreateNodeChildViewsParallel(NodesToProcess, start, showHigher, childCount +
nodeStart, nodeStart - childCount));
00489                         threadJob.Name = i.ToString();
00490                         threadJob.Start();
00491                         threadList.Add(threadJob);
00492                     }
00493                 }
00494                 foreach (Thread t in threadList)
00495                 {
00496                     t.Join();
00497                 }
00498                 cNodes.Sort((x, y) => x.Item1.CompareTo(y.Item1));
00499
00500                 foreach (Tuple<int, string> tup in cNodes)
00501                 {
00502                     output += tup.Item2;
00503                 }
00504             }
00505         }
00506     }
00507     else if (n.Children.Count > 0)
00508     {
00509         foreach (Node n2 in n.Children)
00510         {
00511             if (GotParent)
00512             {
00513                 if (n2.Number == Parent)
00514                 {
00515                     output += CreateNodeView(n2, "node-parent");
00516                 }
00517             }
00518             output += CheckChildren(n2, number, pCount, nodeStart, ref found);
00519         }
00520     }
00521     return output;
00522 }
00523

```

### 5.6.3.3 CheckNodeNumber()

```

bool OSXJV.Classes.Output.CheckNodeNumber (
    Node n,
    int number ) [private]

```

Checks if [Node](#) number and inputted number match.

#### Parameters

<i>n</i>	<a href="#">Node</a> to search
<i>number</i>	Number to match with

## Returns

Definition at line 661 of file [Output.cs](#).

References [OSXJV.Classes.Node.Number](#).

```
00662         {
00663             return n.Number.Equals(number);
00664         }
```

### 5.6.3.4 CreateExtraNode() [1/2]

```
string OSXJV.Classes.Output.CreateExtraNode (
    string type,
    int id ) [private]
```

Builds a get more button to display

## Parameters

<i>type</i>	<a href="#">Node</a> type e.g. 'node-child'
<i>id</i>	The id of the node to start from

## Returns

String of calculated HTML

Definition at line 164 of file [Output.cs](#).

```
00165         {
00166             string node = "";
00167
00168             if (type == "node")
00169             {
00170                 if (GotParent)
00171                 {
00172                     left = left + 400;
00173                 }
00174             }
00175             if (type == "node-child")
00176             {
00177                 left = left + 400;
00178             }
00179
00180             node += "<div class='node-child type ui-draggable ui-selecttee' style='left:" +
left + "px; top:" + top + "px;margin-bottom:50px;'>";
00181             node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
)'>Show Lower</button></span></div>";
00182             node += "</div></div>";
00183             return node;
00184         }
```

## 5.6.3.5 CreateExtraNode() [2/2]

```
string OSXJV.Classes.Output.CreateExtraNode (
    string type,
    int leftVal,
    int topVal,
    int id ) [private]
```

Create a extra node button

## Parameters

<i>type</i>	<a href="#">Node</a> type e.g. 'node-child'
<i>leftVal</i>	Margin from the left of the display
<i>topVal</i>	Margin from the top of the display
<i>id</i>	The id of the node to start from

## Returns

String of calculated HTML

Definition at line 223 of file [Output.cs](#).

```
00224     {
00225         string node = "";
00226
00227         if (type == "node")
00228         {
00229             if (GotParent)
00230             {
00231                 leftVal = leftVal + 400;
00232             }
00233         }
00234         if (type == "node-child")
00235         {
00236             leftVal = leftVal + 400;
00237         }
00238         node += "<div class='node-child type ui-draggable ui-selectee' style='left:" + leftVal + "px;
top:" + topVal + "px;margin-bottom:50px;'>";
00239         node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
)'>Show Lower</button></span></div>";
00240         node += "</div></div>";
00241         return node;
00242     }
```

## 5.6.3.6 CreateGrid()

```
JObject OSXJV.Classes.Output.CreateGrid ( )
```

Creates the grid data.

**Returns**

A JSON object

Definition at line 53 of file [Output.cs](#).

References [OSXJV.Classes.Node.Children](#), [OSXJV.Classes.Node.Name](#), and [OSXJV.Classes.Node.Number](#).

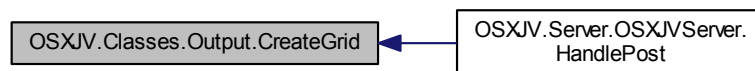
Referenced by [OSXJV.Server.OSXJVServer.HandlePost\(\)](#).

```

00054     {
00055         JObject obj = new JObject();
00056         obj.Add("text", nodes.Name);
00057         obj.Add("id", nodes.Number);
00058         obj.Add("state", new JObject(new JProperty("selected", true)));
00059
00060         if(nodes.Children.Count > 0)
00061         {
00062             JArray array = new JArray();
00063             foreach (Node n2 in nodes.Children)
00064             {
00065                 array.Add(GridGetChidren(n2));
00066             }
00067             obj.Add("children", array);
00068         }
00069         return obj;
00070     }

```

Here is the caller graph for this function:

**5.6.3.7 CreateNodeChildViewsParallel()**

```

void OSXJV.Classes.Output.CreateNodeChildViewsParallel (
    List< Node > job,
    int start,
    bool showHigher,
    int next,
    int previous ) [private]

```

Generate [Output](#) HTML when using multi-threads

**Parameters**

<i>job</i>	The Nodes to process
<i>start</i>	Start index
<i>showHigher</i>	if the are nodes higher up, show previous button
<i>next</i>	Next value for next button
<i>previous</i>	Previous value for previous button

Definition at line 252 of file [Output.cs](#).

References [OSXJV.Classes.Node.Comments](#).

```

00253     {
00254         int threadID = int.Parse(Thread.CurrentThread.Name);
00255         string type = "node-child";
00256         string output = "";
00257
00258         if(start == 0 && showHigher)
00259         {
00260             output += CreatePreviousNode(type, left,
top, previous);
00261         }
00262         bool hadCommentsPrev = false;
00263         int numCommentsPrevious = 0;
00264
00265         foreach(Node n in job)
00266         {
00267             int extra = showHigher ? 130 * (start +1) : 130 * start;
00268
00269             if (hadCommentsPrev)
00270                 extra += (numCommentsPrevious * 25);
00271
00272             if (n.Comments.Count > 0)
00273             {
00274                 hadCommentsPrev = true;
00275                 numCommentsPrevious = n.Comments.Count;
00276             }
00277             else
00278                 hadCommentsPrev = false;
00279
00280             output += CreateNodeView(n, type,left,top + extra);
00281             start++;
00282             if (start == 200)
00283             {
00284                 output += CreateExtraNode(type, left, top + extra + 130,next);
00285                 break;
00286             }
00287         }
00288
00289         cNodes.Add(new Tuple<int, string>(threadID, output));
00290     }
00291 }

```

### 5.6.3.8 CreateNodeView() [1/2]

```

string OSXJV.Classes.Output.CreateNodeView (
    Node n,
    string type,
    int leftVal,
    int topVal ) [private]

```

Generates HTML for the Specific [Node](#) (Multi-Threaded Version)

#### Parameters

<i>n</i>	<a href="#">Node</a> to parse
<i>type</i>	Type of node
<i>leftVal</i>	Margin left of display
<i>topVal</i>	Margin top of display

**Returns**

String of calculated HTML

Definition at line 557 of file [Output.cs](#).

References [OSXJV.Classes.Node.Attributes](#), [OSXJV.Classes.Node.Comments](#), [OSXJV.Classes.Attribute.Name](#), [OSXJV.Classes.Node.Name](#), [OSXJV.Classes.Node.Number](#), [OSXJV.Classes.Attribute.Value](#), and [OSXJV.Classes.Node.Value](#).

```

00558     {
00559         string node = "";
00560
00561         if(type == "node")
00562         {
00563             if(GotParent)
00564             {
00565                 leftVal = leftVal + 400;
00566             }
00567         }
00568         if(type == "node-child")
00569         {
00570             leftVal = leftVal + 400;
00571         }
00572         node += "<div id='" + n.Number + "' class='" + type + " type ui-draggable ui-selectee'
style='left:" + leftVal + "px; top:" + topVal + "px;'>";
00573         node += "<div class='head'><span><button class='nameBtn' onclick='GetNode(\"+n.Number+\")'>\" + n.
Name + "</button></span></div>";
00574         if (!string.IsNullOrEmpty(n.Value))
00575         {
00576             node += string.Format("<div class='blockR'><p>Value</p></div><div
class=comment><span>{0}</span></div>", n.Value);
00577         }
00578         if(n.Comments.Count > 0)
00579         {
00580             node += "<div><p class='text-center'>Comments</p></div>";
00581
00582             foreach (string com in n.Comments)
00583             {
00584                 node += "<div class='comment'>\" + com + "</div>";
00585             }
00586         }
00587         if (n.Attributes.Count > 0)
00588         {
00589             node += "<div class='attribute'><div class='aHeader'><p><button><i class='fa
fa-plus'></i></button>Attributes</p></div><div class='options'>";
00590             foreach (Attribute a in n.Attributes)
00591             {
00592                 node += string.Format("<div class='blockR'><p>{0}</p></div><div
class='comment'><p>{1}</p></div>", a.Name, a.Value);
00593             }
00594             node += "</div>";
00595         }
00596         node += "</div></div>";
00597         return node;
00598     }

```

**5.6.3.9 CreateNodeView()** [2/2]

```

string OSXJV.Classes.Output.CreateNodeView (
    Node n,
    string type ) [private]

```

Generates HTML for specific [Node](#) (Single Threaded Version)

**Parameters**

<i>n</i>	<a href="#">Node</a> to parse
<i>type</i>	Type of node

**Returns**

String of calculated HTML

Definition at line 606 of file [Output.cs](#).

References [OSXJV.Classes.Node.Attributes](#), [OSXJV.Classes.Node.Comments](#), [OSXJV.Classes.Attribute.Name](#), [OSXJV.Classes.Node.Name](#), [OSXJV.Classes.Node.Number](#), [OSXJV.Classes.Attribute.Value](#), and [OSXJV.Classes.Node.Value](#).

```

00607     {
00608         string node = "";
00609         int leftVal = left;
00610         if (type == "node")
00611         {
00612             if (GotParent)
00613             {
00614                 left = left + 400;
00615                 leftVal = left;
00616             }
00617         }
00618         if (type == "node-child")
00619         {
00620             leftVal = leftVal + 400;
00621         }
00622         node += "<div id='" + n.Number + "' class='" + type + " type ui-draggable ui-selectee'
style='left:" + leftVal + "px; top:" + top + "px;'>";
00623         node += "<div class='head'><span><button class='nameBtn' onclick='GetNode(" + n.Number + ")'>"
+ n.Name + "</button></span></div>";
00624         if (!string.IsNullOrEmpty(n.Value))
00625         {
00626             node += string.Format("<div class='blockR'><p>Value</p></div><div
class=comment><span>{0}</span></div>", n.Value);
00627         }
00628         if (n.Comments.Count > 0)
00629         {
00630             node += "<div><p class='text-center'>Comments</p></div>";
00631         }
00632         foreach (string com in n.Comments)
00633         {
00634             node += "<div class='comment'>" + com + "</div>";
00635         }
00636         if (n.Attributes.Count > 0)
00637         {
00638             node += "<div class='attribute'><div class='aHeader'><p><button><i class='fa
fa-plus'></i></button>Attributes</p></div><div class='options'>";
00640             foreach (Attribute a in n.Attributes)
00641             {
00642                 node += string.Format("<div class='blockR'><p>{0}</p></div><div
class='comment'><p>{1}</p></div>", a.Name, a.Value);
00643             }
00644             node += "</div>";
00645         }
00646         node += "</div></div>";
00647     }
00648     if (type == "node-child")
00649     {
00650         top = top + 130;
00651     }
00652     return node;
00653 }

```

**5.6.3.10 CreatePreviousNode()**

```

string OSXJV.Classes.Output.CreatePreviousNode (
    string type,
    int leftVal,
    int topVal,
    int id ) [private]

```

Create a previous node button

**Parameters**

<i>type</i>	<a href="#">Node</a> type e.g. 'node-child'
<i>leftVal</i>	Margin from the left of the display
<i>topVal</i>	Margin from the top of the display
<i>id</i>	The id of the node to start from

**Returns**

String of calculated HTML

Definition at line 194 of file [Output.cs](#).

```

00195     {
00196         string node = "";
00197
00198         if (type == "node")
00199         {
00200             if (GotParent)
00201             {
00202                 leftVal = leftVal + 400;
00203             }
00204         }
00205         if (type == "node-child")
00206         {
00207             leftVal = leftVal + 400;
00208         }
00209         node += "<div class='node-child type ui-draggable ui-selecttee' style='left:" + leftVal + "px;
top:" + topVal + "px;'>";
00210         node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
)'>Show Higher</button></span></div>";
00211         node += "</div></div>";
00212         return node;
00213     }

```

**5.6.3.11 CreateView()**

```

string OSXJV.Classes.Output.CreateView (
    int node = 1,
    int pCount = 4,
    int nodeStart = 0 )

```

Creates the view of nodes using multiple threads.

**Parameters**

<i>node</i>	Number of node to start from. Default is 1(Root)
<i>pCount</i>	Number of Threads to use. Default is 4
<i>nodeStart</i>	Where to start the child nodes from

**Returns**

String of calculated HTML

Definition at line 300 of file [Output.cs](#).



References [OSXJV.Classes.Node.Children](#), and [OSXJV.Classes.Node.Number](#).

Referenced by [OSXJV.Server.OSXJVServer.HandleGet\(\)](#), and [OSXJV.Server.OSXJVServer.HandlePost\(\)](#).

```

00301     {
00302
00303         List<Thread> threadList = new List<Thread>();
00304
00305         string output = "<div class='text-center ui-layout-center ui-layout-pane
00306 ui-layout-pane-center'><div style ='display:inline-block' class='ui-selectable ui-droppable'>";
00307         if (nodes.Number.Equals(node))
00308         {
00309             int childCount = 0;
00310
00311             if (nodes.Children.Count < 200)
00312                 childCount = nodes.Children.Count;
00313             else
00314             {
00315                 childCount = 200;
00316             }
00317
00318             if(childCount < pCount * 2)
00319             {
00320                 output += CreateNodeView(nodes, "node",
00321 left, top);
00322                 foreach(Node n2 in nodes.Children)
00323                 {
00324                     output += CreateNodeView(n2, "node-child");
00325                 }
00326             }
00327             else
00328             {
00329                 int spread = (int)Math.Ceiling((double)childCount / (double)pCount);
00330                 output += CreateNodeView(nodes, "node",
00331 left,top); //Parent(Node) Thread
00332                 for (int i = 0; i < pCount; i++)
00333                 {
00334                     int neg = 0;
00335                     if ((spread * (i + 1)) > childCount)
00336                     {
00337                         neg = childCount - (spread * (i + 1));
00338                     }
00339                     int start = (spread * i) ;
00340                     int rangeStart = (spread * i) + nodeStart;
00341                     bool showHigher = nodeStart != 0 ? true : false;
00342
00343                     List<Node> NodesToProcess = nodes.Children.GetRange(rangeStart, spread
00344 + neg);
00345                     Thread threadJob = new Thread(() =>
00346 CreateNodeChildViewsParallel(NodesToProcess, start, showHigher, childCount +
00347 nodeStart, nodeStart - childCount));
00348                     threadJob.Name = i.ToString();
00349                     threadJob.Start();
00350                     threadList.Add(threadJob);
00351                 }
00352                 foreach(Thread t in threadList)
00353                 {
00354                     t.Join();
00355                 }
00356                 cNodes.Sort((x, y) => x.Item1.CompareTo(y.Item1));
00357                 foreach(Tuple<int,string> tup in cNodes)
00358                 {
00359                     output += tup.Item2;
00360                 }
00361             }
00362         }
00363         else
00364         {
00365             GetParent(nodes, node);
00366             string temp = "";
00367             if (GotParent)
00368             {
00369                 if (nodes.Number == Parent)
00370                 {
00371                     output += CreateNodeView(nodes, "node-parent");
00372                 }
00373             }
00374             bool found =false;
00375             foreach (Node n2 in nodes.Children)
00376             {
00377                 if (GotParent)

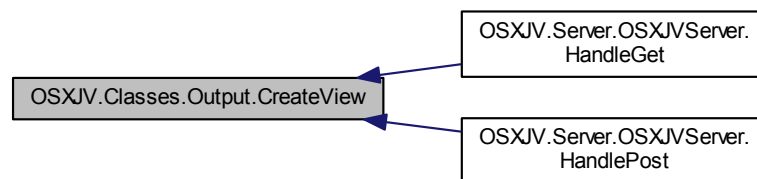
```

```

00376         {
00377             if (n2.Number == Parent)
00378             {
00379                 output += CreateNodeView(n2, "node-parent");
00380             }
00381         }
00382         temp += CheckChildren(n2, node, pCount, nodeStart, ref found);
00383         if (found)
00384             break;
00385     }
00386     if (!string.IsNullOrEmpty(temp))
00387         output += temp;
00388 }
00389 output += "</div></div>";
00390 return output;
00391 }

```

Here is the caller graph for this function:



#### 5.6.3.12 CreateViewSingle()

```

string OSXJV.Classes.Output.CreateViewSingle (
    int node,
    int nodeStart = 0 )

```

CreateView using a Single Thread

##### Parameters

<i>node</i>	Index of node to start from
<i>nodeStart</i>	Where to start the child nodes from

##### Returns

String of calculated HTML

Definition at line 101 of file [Output.cs](#).

References [OSXJV.Classes.Node.Children](#), and [OSXJV.Classes.Node.Number](#).

```

00102     {
00103         string output = "<div class='text-center ui-layout-center ui-layout-pane
                        ui-layout-pane-center'><div style ='display:inline-block' class='ui-selectable ui-droppable'>";

```

```

00104
00105         if (nodes.Number.Equals(node))
00106         {
00107             int count = 0;
00108             output += CreateNodeView(nodes, "node");
00109
00110
00111             foreach (Node n in nodes.Children)
00112             {
00113                 if(nodeStart > 0)
00114                 {
00115                     if (count != nodeStart)
00116                         continue;
00117                 }
00118                 count++;
00119                 output += CreateNodeView(n, "node-child"); //Child(Nodes) Thread
00120
00121                 if ((count-nodeStart) == 200)
00122                 {
00123                     output += CreateExtraNode("node-child",count);
00124                     break;
00125                 }
00126             }
00127         }
00128     else
00129     {
00130         GetParent(nodes, node);
00131         string temp = "";
00132         if (GotParent)
00133         {
00134             if (nodes.Number == Parent)
00135             {
00136                 output += CreateNodeView(nodes, "node-parent");
00137             }
00138         }
00139         foreach (Node n2 in nodes.Children)
00140         {
00141             if (GotParent)
00142             {
00143                 if (n2.Number == Parent)
00144                 {
00145                     output += CreateNodeView(n2, "node-parent");
00146                 }
00147             }
00148             temp += CheckChildren(n2, node);
00149         }
00150         if (!string.IsNullOrEmpty(temp))
00151             output += temp;
00152     }
00153     output += "</div></div>"; //Close out divs
00154     return output;
00155 }
00156

```

### 5.6.3.13 GetParent()

```

void OSXJV.Classes.Output.GetParent (
    Node node,
    int number ) [private]

```

Finds the parent node.

#### Parameters

<i>node</i>	Node to search
<i>number</i>	Node number to find

Definition at line 530 of file [Output.cs](#).

References [OSXJV.Classes.Node.Children](#), and [OSXJV.Classes.Node.Number](#).

```

00531     {
00532         if (!CheckNodeNumber (node, number))
00533         {
00534             foreach (Node n in node.Children)
00535             {
00536                 if (CheckNodeNumber (n, number))
00537                 {
00538                     Parent = node.Number;
00539                     GotParent = true;
00540                 }
00541                 else
00542                 {
00543                     GetParent (n, number);
00544                 }
00545             }
00546         }
00547     }

```

### 5.6.3.14 GridGetChidren()

```

JObject OSXJV.Classes.Output.GridGetChidren (
    Node n ) [private]

```

Recursive function to get all the nodes data for the grid .

#### Parameters

<i>n</i>	Child <a href="#">Node</a>
----------	----------------------------

#### Returns

JSON object

Definition at line 77 of file [Output.cs](#).

References [OSXJV.Classes.Node.Children](#), [OSXJV.Classes.Node.Name](#), and [OSXJV.Classes.Node.Number](#).

```

00078     {
00079         JObject child = new JObject();
00080         child.Add("id", n.Number);
00081         child.Add("text", n.Name);
00082
00083         if (n.Children.Count > 0)
00084         {
00085             JArray array = new JArray();
00086             foreach (Node n2 in n.Children)
00087             {
00088                 array.Add(GridGetChidren (n2));
00089             }
00090             child.Add("children", array);
00091         }
00092         return child;
00093     }

```

## 5.6.4 Member Data Documentation

#### 5.6.4.1 cNodes

```
List<Tuple<int, string> > OSXJV.Classes.Output.cNodes = new List<Tuple<int, string>>()  
[private]
```

Used in Threading, list of calculated HTML strings.

Definition at line 36 of file [Output.cs](#).

#### 5.6.4.2 GotParent

```
bool OSXJV.Classes.Output.GotParent = false [private]
```

Definition at line 26 of file [Output.cs](#).

#### 5.6.4.3 left

```
int OSXJV.Classes.Output.left = 100 [private]
```

Definition at line 16 of file [Output.cs](#).

#### 5.6.4.4 nodes

```
Node OSXJV.Classes.Output.nodes [private]
```

Definition at line 21 of file [Output.cs](#).

#### 5.6.4.5 Parent

```
int OSXJV.Classes.Output.Parent = 0 [private]
```

Parent of node when building output (Used when getting [Node](#) other than root).

Definition at line 31 of file [Output.cs](#).

#### 5.6.4.6 top

```
int OSXJV.Classes.Output.top = 130 [private]
```

Definition at line 16 of file [Output.cs](#).

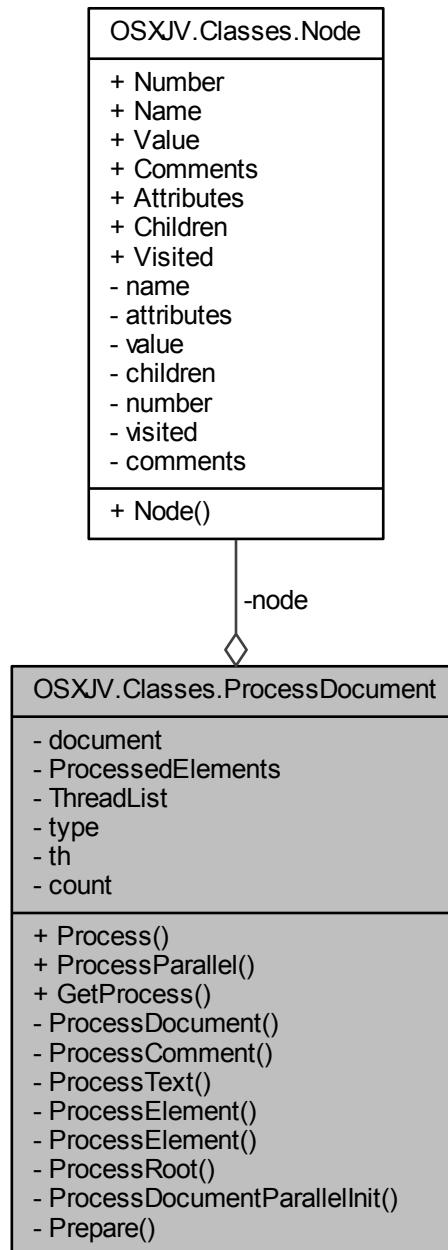
The documentation for this class was generated from the following file:

- OSXJVClasses/[Output.cs](#)

## 5.7 OSXJV.Classes.ProcessDocument Class Reference

Class the Processes the document

Collaboration diagram for OSXJV.Classes.ProcessDocument:



## Public Member Functions

- [Node Process \(\)](#)  
*Single Threaded Process.*
- [Node ProcessParallel \(int pCount=4\)](#)  
*Parse Document Using Multiple Threads*

## Static Public Member Functions

- static [ProcessDocument GetProcess](#) (string data, string [type](#))  
*Gets an instance of the [ProcessDocument](#) and prepare object.*

## Private Member Functions

- [ProcessDocument](#) (XDocument doc, string [type](#))  
*Constructor*
- void [ProcessComment](#) (XComment e, [Node node](#))  
*Extract Comment*
- void [ProcessText](#) (XText e, [Node n](#))  
*Get text from the data*
- [Node ProcessElement](#) (XElement e, [Node node](#))  
*Single Threaded Process Element Version*
- [Node ProcessElement](#) (XElement e, [Node node](#), ref int nodeNumber)  
*Multi-Threaded Version to process element*
- [Node ProcessRoot](#) (XElement e, [Node node](#))  
*Processes first element in the document.*
- void [ProcessDocumentParallelInit](#) (XDocument doc, int start)  
*Method that each thread uses to process the document*

## Static Private Member Functions

- static XDocument [Prepare](#) (string data, string [type](#))  
*Prepares the object with setting the XDocument object to process*

## Private Attributes

- XDocument [document](#)  
*Object the contains the parsed data ready to be processed.*
- [Node node](#) = new [Node](#)()  
*The Initial [Node](#).*
- List< Tuple< [Node](#), int > > [ProcessedElements](#) = new List<Tuple<[Node](#), int>>()  
*Used with threading to keep list of processed Nodes.*
- List< Thread > [ThreadList](#) = new List<Thread>()  
*Used with threading to keep list of running threads.*
- string [type](#)  
*Document Type.*
- Thread [th](#)
- int [count](#)  
*Used to by single thread operation to keep track of node id.*

### 5.7.1 Detailed Description

Class the Processes the document

Definition at line 15 of file [ProcessDocument.cs](#).



## 5.7.2 Constructor & Destructor Documentation

### 5.7.2.1 ProcessDocument()

```
OSXJV.Classes.ProcessDocument.ProcessDocument (
    XDocument doc,
    string type ) [private]
```

#### Constructor

##### Parameters

<i>doc</i>	Parsed document
<i>type</i>	Type of document

Definition at line 53 of file [ProcessDocument.cs](#).

```
00054     {
00055         document = doc;
00056         this.type = type;
00057     }
```

## 5.7.3 Member Function Documentation

### 5.7.3.1 GetProcess()

```
static ProcessDocument OSXJV.Classes.ProcessDocument.GetProcess (
    string data,
    string type ) [static]
```

Gets an instance of the [ProcessDocument](#) and prepare object.

#### Parameters

<i>data</i>	String of the document
<i>type</i>	Type of document

#### Returns

Definition at line 77 of file [ProcessDocument.cs](#).

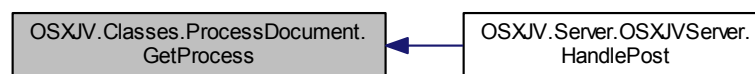
Referenced by [OSXJV.Server.OSXJVServer.HandlePost\(\)](#).

```

00078     {
00079         if (string.IsNullOrEmpty(data) || string.IsNullOrEmpty(type))
00080         {
00081             throw new ArgumentException();
00082         }
00083         try
00084         {
00085             XDocument doc = null;
00086             doc = Prepare(data, type);
00087             return new ProcessDocument(doc, type);
00088         }
00089         catch (System.Xml.XmlException e)
00090         {
00091             throw e;
00092         }
00093     }

```

Here is the caller graph for this function:



### 5.7.3.2 Prepare()

```

static XDocument OSXJV.Classes.ProcessDocument.Prepare (
    string data,
    string type ) [static], [private]

```

Prepares the object with setting the XDocument object to process

#### Parameters

<i>data</i>	String of data
<i>type</i>	Data type

#### Returns

A XDocument object

Definition at line 111 of file [ProcessDocument.cs](#).

```

00112     {
00113
00114         if (type.Equals("JSON"))
00115             return new XDocument(JsonConvert.DeserializeXmlNode(data, "Root", false).Root.FirstNode);
00116         else if (type.Equals("XML") || type.Equals("HTML"))
00117             return XDocument.Parse(data);
00118
00119         return null;
00120     }

```

### 5.7.3.3 Process()

[Node](#) OSXJV.Classes.ProcessDocument.Process ( )

Single Threaded Process.

#### Returns

Object of Nodes

Definition at line 127 of file [ProcessDocument.cs](#).

```

00128     {
00129         if (document.Nodes() != null)
00130         {
00131             foreach (XNode n in document.Nodes())
00132             {
00133                 switch (n.NodeType)
00134                 {
00135                     case System.Xml.XmlNodeType.Element:
00136                         count++;
00137                         ProcessElement(XElement.Parse(n.ToString()),
node);
00138                         break;
00139                     case System.Xml.XmlNodeType.Comment:
00140                         ProcessComment(n as XComment, node);
00141                         break;
00142                     case System.Xml.XmlNodeType.Text:
00143                         ProcessText(n as XText, node);
00144                         break;
00145                     case System.Xml.XmlNodeType.Notation:
00146                         break;
00147                     case System.Xml.XmlNodeType.EndElement:
00148                         break;
00149                     default:
00150                         break;
00151                 }
00152             }
00153         }
00154         //SortArray(ref node);
00155         document = null;
00156         return node;
00157     }

```

### 5.7.3.4 ProcessComment()

```

void OSXJV.Classes.ProcessDocument.ProcessComment (
    XComment e,
    Node node ) [private]

```

Extract Comment

#### Parameters

<i>e</i>	Comment object to be parsed
<i>node</i>	<a href="#">Node</a> to input data

Definition at line 64 of file [ProcessDocument.cs](#).

References [OSXJV.Classes.Node.Comments](#).

```

00065     {
00066         string s = "";
00067         s = Regex.Replace(e.Value, @"^\w\s\.\@-", "");
00068         node.Comments.Add(s);
00069     }

```

### 5.7.3.5 ProcessDocumentParallelInit()

```

void OSXJV.Classes.ProcessDocument.ProcessDocumentParallelInit (
    XDocument doc,
    int start ) [private]

```

Method that each thread uses to process the document

#### Parameters

<i>doc</i>	A subset of the full document
<i>start</i>	Start index number

Definition at line 336 of file [ProcessDocument.cs](#).

References [OSXJV.Classes.Node.Children](#).

```

00337     {
00338         int nodeNum = start;
00339
00340         Node node = new Node();
00341         if (doc.Root.Nodes() != null)
00342         {
00343             List<XNode> list = doc.Root.Nodes().ToList();
00344             foreach (XNode n in doc.Root.Nodes())
00345             {
00346                 switch (n.NodeType)
00347                 {
00348                     case System.Xml.XmlNodeType.Element:
00349                         nodeNum++;
00350                         Node n2 = new Node();
00351                         node.Children.Add(ProcessElement(XElement.Parse(n.ToString()), n2
, ref nodeNum));
00352                         break;
00353                     case System.Xml.XmlNodeType.Comment:
00354                         ProcessComment(n as XComment, node);
00355                         break;
00356                     case System.Xml.XmlNodeType.Text:
00357                         ProcessText(n as XText, node);
00358                         break;
00359                     case System.Xml.XmlNodeType.Notation:
00360                         break;
00361                     case System.Xml.XmlNodeType.EndElement:
00362                         break;
00363                     default:
00364                         break;
00365                 }
00366             }
00367         }
00368         document = null;
00369         ProcessedElements.Add(new Tuple<Node, int>(node, start));
00370     }

```

## 5.7.3.6 ProcessElement() [1/2]

```
Node OSXJV.Classes.ProcessDocument.ProcessElement (
    XElement e,
    Node node ) [private]
```

Single Threaded Process Element Version

## Parameters

<i>e</i>	Element to Process
<i>node</i>	The <a href="#">Node</a> to fill data with

## Returns

Definition at line 165 of file [ProcessDocument.cs](#).

References [OSXJV.Classes.Node.Attributes](#), [OSXJV.Classes.Node.Children](#), [OSXJV.Classes.Attribute.Name](#), [OSXJV.Classes.Node.Name](#), [OSXJV.Classes.Node.Number](#), [OSXJV.Classes.Attribute.Value](#), and [OSXJV.Classes.Node.Visited](#).

```
00166     {
00167         if (node.Number == 0)
00168         {
00169             node.Number = count;
00170         }
00171         if (!node.Visited)
00172         {
00173             node.Name = e.Name.LocalName;
00174             foreach (XAttribute ax in e.Attributes())
00175             {
00176                 if (ax.Name == "id")
00177                 {
00178                     node.Name = node.Name + " #" + ax.Value;
00179                 }
00180             }
00181             if (type == "HTML")
00182             {
00183                 if (ax.IsNamespaceDeclaration)
00184                     continue;
00185             }
00186             Attribute att = new Attribute();
00187             att.Name = ax.Name.LocalName;
00188             att.Value = ax.Value;
00189             node.Attributes.Add(att);
00190         }
00191     }
00192 }
00193
00194 if (e.Nodes() != null)
00195 {
00196     foreach (XNode n in e.Nodes())
00197     {
00198         switch (n.NodeType)
00199         {
00200             case System.Xml.XmlNodeType.EndElement:
00201                 break;
00202             case System.Xml.XmlNodeType.Element:
00203                 count++;
00204                 Node n2 = new Node();
00205                 node.Children.Add(ProcessElement(XElement.Parse(n.
ToString()), n2));
00206                 break;
00207             case System.Xml.XmlNodeType.Comment:
00208                 ProcessComment(n as XComment, node);
00209                 break;
00210             case System.Xml.XmlNodeType.Text:
00211                 ProcessText(n as XText, node);
00212                 break;
```

```

00213             case System.Xml.XmlNodeType.Notation:
00214                 break;
00215             default:
00216                 break;
00217         }
00218     }
00219 }
00220 }
00221 node.Visited = true;
00222 return node;
00223 }

```

### 5.7.3.7 ProcessElement() [2/2]

```

Node OSXJV.Classes.ProcessDocument.ProcessElement (
    XElement e,
    Node node,
    ref int nodeNumber ) [private]

```

Multi-Threaded Version to process element

#### Parameters

<i>e</i>	Element to process
<i>node</i>	<a href="#">Node</a> to extract data from
<i>nodeNumber</i>	The Thread internal node number

#### Returns

Definition at line 232 of file [ProcessDocument.cs](#).

References [OSXJV.Classes.Node.Attributes](#), [OSXJV.Classes.Node.Children](#), [OSXJV.Classes.Attribute.Name](#), [OSXJV.Classes.Node.Name](#), [OSXJV.Classes.Node.Number](#), [OSXJV.Classes.Attribute.Value](#), and [OSXJV.Classes.Node.Visited](#).

```

00233     {
00234         if (!node.Visited)
00235         {
00236             if (node.Number == 0)
00237             {
00238                 node.Number = nodeNumber;
00239             }
00240             if (!node.Visited)
00241             {
00242                 node.Name = e.Name.LocalName;
00243                 foreach (XAttribute ax in e.Attributes())
00244                 {
00245                     if (ax.Name == "id")
00246                     {
00247                         node.Name = node.Name + " #" + ax.Value;
00248                     }
00249                     if (type == "HTML")
00250                     {
00251                         if (ax.IsNamespaceDeclaration)
00252                             continue;
00253                     }
00254                     Attribute att = new Attribute();
00255                     att.Name = ax.Name.LocalName;
00256                 }
00257             }
00258         }
00259     }

```

```

00258             att.Value = ax.Value;
00259             node.Attributes.Add(att);
00260         }
00261     }
00262
00263     if (e.Nodes() != null)
00264     {
00265         foreach (XmlNode n in e.Nodes())
00266         {
00267             switch (n.NodeType)
00268             {
00269                 case System.Xml.XmlNodeType.EndElement:
00270                     break;
00271                 case System.Xml.XmlNodeType.Element:
00272                     nodeNumber++;
00273                     Node n2 = new Node();
00274                     node.Children.Add(ProcessElement(XElement.Parse(n
.ToString()), n2, ref nodeNumber));
00275                     break;
00276                 case System.Xml.XmlNodeType.Comment:
00277                     ProcessComment(n as XComment, node);
00278                     break;
00279                 case System.Xml.XmlNodeType.Text:
00280                     ProcessText(n as XText, node);
00281                     break;
00282                 case System.Xml.XmlNodeType.Notation:
00283                     break;
00284
00285                 default:
00286                     break;
00287             }
00288         }
00289     }
00290     node.Visited = true;
00291     return node;
00292 }
00293

```

### 5.7.3.8 ProcessParallel()

```

Node OSXJV.Classes.ProcessDocument.ProcessParallel (
    int pCount = 4 )

```

Parse Document Using Multiple Threads

#### Parameters

<i>pCount</i>	Number of Threads to run Default = 4
---------------	--------------------------------------

#### Returns

A object of [Node](#) that has been processed

Definition at line 377 of file [ProcessDocument.cs](#).

References [OSXJV.Classes.Node.Children](#).

Referenced by [OSXJV.Server.OSXJVServer.HandlePost\(\)](#).

```

00378     {
00379         node = ProcessRoot(document.Root, node);
00380
00381         int nodeCount = document.Root.Nodes().Count();
00382
00383         if (nodeCount <= pCount)

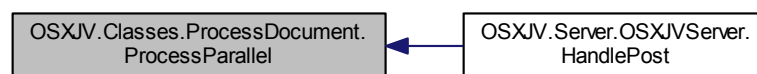
```

```

00384         {
00385             return Process();
00386         }
00387     else if (nodeCount > pCount)
00388     {
00389
00390         List<XNode> List = document.Root.Nodes().ToList();
00391         int spread = 0;
00392
00393         spread = (int)Math.Ceiling((double)nodeCount / (double)pCount);
00394
00395         int totalNodes = 1;
00396
00397         for (int i = 0; i < pCount; i++)
00398         {
00399             int neg = 0;
00400             int start = totalNodes;
00401             if ((spread * (i+1)) > nodeCount)
00402             {
00403                 neg = nodeCount - (spread * (i + 1));
00404             }
00405
00406             List<XNode> list = List.GetRange((spread * i), spread + neg);
00407             XElement root = new XElement("Root", list);
00408             XDocument doc = new XDocument(root);
00409
00410             (th = new Thread(() => ProcessDocumentParallelInit(doc,
start))).Start();
00411
00412             ThreadList.Add(th); //Add to Threads list to keep recored of threads
running
00413                 totalNodes += root.Descendants().Count(); //Increment start position.
00414             }
00415             document = null;
00416             foreach (Thread t in ThreadList)
00417             {
00418                 t.Join(); //Wait for threads to join
00419             }
00420
00421             ProcessedElements.Sort((x, y) => x.Item2.CompareTo(y.Item2)); //Sort List
by start index so they are in order.
00422
00423             foreach (Tuple<Node,int> tup in ProcessedElements)
00424             {
00425                 foreach (Node n in tup.Item1.Children)
00426                 {
00427                     node.Children.Add(n);
00428                 }
00429             }
00430         }
00431         return node;
00432     }

```

Here is the caller graph for this function:



### 5.7.3.9 ProcessRoot()

```

Node OSXJV.Classes.ProcessDocument.ProcessRoot (
    XElement e,
    Node node ) [private]

```

Processes first element in the document.



## Parameters

<i>e</i>	Element object to process
<i>node</i>	<a href="#">Node</a> to insert data to

## Returns

Definition at line 301 of file [ProcessDocument.cs](#).

References [OSXJV.Classes.Node.Attributes](#), [OSXJV.Classes.Attribute.Name](#), [OSXJV.Classes.Node.Name](#), [OSXJV.Classes.Node.Number](#), [OSXJV.Classes.Attribute.Value](#), and [OSXJV.Classes.Node.Visited](#).

```

00302     {
00303         node.Number = 1;
00304
00305         if (!node.Visited)
00306         {
00307
00308             node.Name = e.Name.LocalName;
00309             foreach (XAttribute ax in e.Attributes())
00310             {
00311                 if (ax.Name == "id")
00312                 {
00313                     node.Name = node.Name + " #" + ax.Value;
00314                 }
00315
00316                 if (type == "HTML")
00317                 {
00318                     if (ax.IsNamespaceDeclaration)
00319                         continue;
00320                 }
00321                 Attribute att = new Attribute();
00322                 att.Name = ax.Name.LocalName;
00323                 att.Value = ax.Value;
00324                 node.Attributes.Add(att);
00325             }
00326         }
00327         node.Visited = true;
00328         return node;
00329     }

```

## 5.7.3.10 ProcessText()

```

void OSXJV.Classes.ProcessDocument.ProcessText (
    XText e,
    Node n ) [private]

```

Get text from the data

## Parameters

<i>e</i>	Text Element
<i>n</i>	<a href="#">Node</a> to input data

Definition at line 100 of file [ProcessDocument.cs](#).

References [OSXJV.Classes.Node.Value](#).

```
00101         {  
00102             n.Value = e.Value;  
00103         }
```

## 5.7.4 Member Data Documentation

### 5.7.4.1 count

```
int OSXJV.Classes.ProcessDocument.count [private]
```

Used to by single thread operation to keep track of node id.

Definition at line 46 of file [ProcessDocument.cs](#).

### 5.7.4.2 document

```
XDocument OSXJV.Classes.ProcessDocument.document [private]
```

Object the contains the parsed data ready to be processed.

Definition at line 20 of file [ProcessDocument.cs](#).

### 5.7.4.3 node

```
Node OSXJV.Classes.ProcessDocument.node = new Node() [private]
```

The Initial [Node](#).

Definition at line 25 of file [ProcessDocument.cs](#).

### 5.7.4.4 ProcessedElements

```
List<Tuple<Node, int> > OSXJV.Classes.ProcessDocument.ProcessedElements = new List<Tuple<Node,  
int>>() [private]
```

Used with threading to keep list of processed Nodes.

Definition at line 30 of file [ProcessDocument.cs](#).

#### 5.7.4.5 th

```
Thread OSXJV.Classes.ProcessDocument.th [private]
```

Definition at line 41 of file [ProcessDocument.cs](#).

#### 5.7.4.6 ThreadList

```
List<Thread> OSXJV.Classes.ProcessDocument.ThreadList = new List<Thread>() [private]
```

Used with threading to keep list of running threads.

Definition at line 35 of file [ProcessDocument.cs](#).

#### 5.7.4.7 type

```
string OSXJV.Classes.ProcessDocument.type [private]
```

Document Type.

Definition at line 40 of file [ProcessDocument.cs](#).

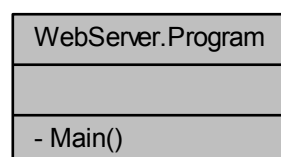
The documentation for this class was generated from the following file:

- OSXJVClasses/[ProcessDocument.cs](#)

## 5.8 WebServer.Program Class Reference

The Initialiser

Collaboration diagram for WebServer.Program:



## Static Private Member Functions

- static void [Main](#) (string[] args)

*The Main function that starts the HttpServer*

### 5.8.1 Detailed Description

The Initialiser

Definition at line 12 of file [Program.cs](#).

### 5.8.2 Member Function Documentation

#### 5.8.2.1 Main()

```
static void WebServer.Program.Main (
    string [] args ) [static], [private]
```

The Main function that starts the HttpServer

Parameters

<i>args</i>	Pass Cache Folder and Logger (Optional)
-------------	---

Definition at line 18 of file [Program.cs](#).

References [OSXJV.Classes.CacheManager.Setup\(\)](#), [OSXJV.Classes.Logger.Setup\(\)](#), and [OSXJV.Server.OSXJVServer.Start\(\)](#).

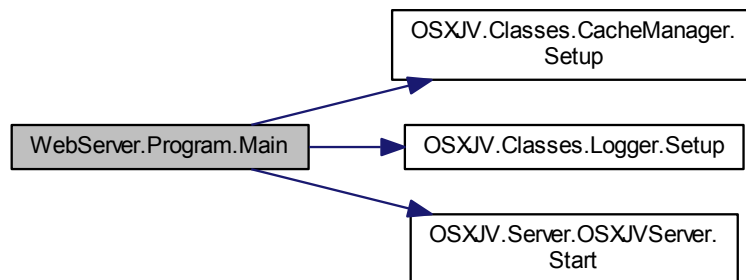
```
00019     {
00020
00021         if (args.Length == 0)
00022         {
00023             Console.WriteLine("Using Default Cache Directory Path and Logger Directory Path");
00024             string dir = Directory.GetCurrentDirectory();
00025             Array.Resize(ref args, 2);
00026             args[0] = dir + "/Cache/";
00027             args[1] = dir + "/Logger/";
00028             if (!Directory.Exists(args[0]))
00029                 Directory.CreateDirectory(args[0]);
00030             if (!Directory.Exists(args[1]))
00031                 Directory.CreateDirectory(args[1]);
00032         }
00033
00034         if (args[0] == args[1])
00035         {
00036             Console.WriteLine("Cache location and Log location is the same. Please enter two different
00037 locations");
00038         }
00039         else
00040         {
00041             bool pass = false;
00042             try
00043             {
00044                 pass = CacheManager.Setup(args[0]);
00045                 pass = Logger.Setup(args[1]);
00046             }
00047         }
00048     }
00049 }
```

```

00046         catch (Exception e)
00047         {
00048             Console.WriteLine("Error Setting Cache and Logger Directory: {0}", e.Message);
00049         }
00050         if (pass)
00051         {
00052             OSXJVServer s = new OSXJVServer();
00053             s.Start();
00054         }
00055
00056         //Check Cache every hour to remove old files
00057         while (true)
00058         {
00059             Thread.Sleep(3600000);
00060
00061             string[] files = Directory.GetFiles(args[0]);
00062
00063             foreach (string file in files)
00064             {
00065                 if (File.GetLastAccessTime(file) < DateTime.Now.AddHours(-6.0))
00066                     File.Delete(file);
00067             }
00068         }
00069     }
00070 }

```

Here is the call graph for this function:



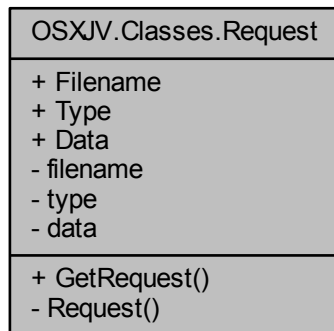
The documentation for this class was generated from the following file:

- [Program.cs](#)

## 5.9 OSXJV.Classes.Request Class Reference

A object containing the document to process, filename and type.

Collaboration diagram for OSXJV.Classes.Request:



### Static Public Member Functions

- static [Request](#) [GetRequest](#) (string [filename](#), string [type](#), string [data](#))  
*Creates an instance of [Request](#).*

### Properties

- string [Filename](#) [get, set]  
*To retrieve the filename of the document*
- string [Type](#) [get, set]  
*To retrieve type of document*
- string [Data](#) [get, set]  
*To retrieve the document data*

### Private Member Functions

- [Request](#) (string [filename](#), string [type](#), string [data](#))  
*Initialises the [Request](#) object, can only be called from [GetRequest\(...\)](#).*

### Private Attributes

- string [filename](#)  
*Document Filename.*
- string [type](#)  
*Type of document.*
- string [data](#)  
*Contents of documents.*

### 5.9.1 Detailed Description

A object containing the document to process, filename and type.

Definition at line 8 of file [Request.cs](#).

### 5.9.2 Constructor & Destructor Documentation

#### 5.9.2.1 Request()

```
OSXJV.Classes.Request.Request (
    string filename,
    string type,
    string data ) [private]
```

Initialises the [Request](#) object, can only be called from GetRequest(...).

##### Parameters

<i>filename</i>	The document filename e.g. Test
<i>type</i>	The document file type e.g. text/xml
<i>data</i>	The document data e.g. {"name":"bob","address":"123 Somewhere"}

Definition at line 31 of file [Request.cs](#).

```
00032     {
00033         this.filename = filename;
00034         this.type = type;
00035         this.data = data;
00036     }
```

### 5.9.3 Member Function Documentation

#### 5.9.3.1 GetRequest()

```
static Request OSXJV.Classes.Request.GetRequest (
    string filename,
    string type,
    string data ) [static]
```

Creates an instance of [Request](#).

##### Parameters

<i>filename</i>	The document filename e.g. Test
<i>type</i>	The document file type e.g. text/xml
<i>data</i>	The document data e.g. { "name" : "bob", "address" : "123 Somewhere" }

**Returns**

Object of [Request](#)

Definition at line 45 of file [Request.cs](#).

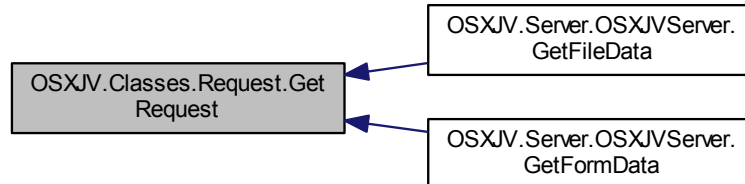
Referenced by [OSXJV.Server.OSXJVServer.GetFilesData\(\)](#), and [OSXJV.Server.OSXJVServer.GetFormData\(\)](#).

```

00046     {
00047         string Type = "";
00048         if (string.IsNullOrEmpty(filename) || string.IsNullOrEmpty(
00049             type) || string.IsNullOrEmpty(data))
00050             throw new ArgumentException();
00051         if (type.Equals("text/xml") || type.Equals("application/xml"))
00052         {
00053             Type = "XML";
00054         }
00055         else if (type.Equals("text/html"))
00056         {
00057             Type = "HTML";
00058         }
00059         else if (type.Equals("application/json") || type.Equals("application/octet-stream"))
00060         {
00061             Type = "JSON";
00062         }
00063         return new Request(filename, Type, data);
00064     }

```

Here is the caller graph for this function:



## 5.9.4 Member Data Documentation

### 5.9.4.1 data

```
string OSXJV.Classes.Request.data [private]
```

Contents of documents.

Definition at line 23 of file [Request.cs](#).



#### 5.9.4.2 filename

```
string OSXJV.Classes.Request.filename [private]
```

Document Filename.

Definition at line 13 of file [Request.cs](#).

#### 5.9.4.3 type

```
string OSXJV.Classes.Request.type [private]
```

Type of document.

Definition at line 18 of file [Request.cs](#).

### 5.9.5 Property Documentation

#### 5.9.5.1 Data

```
string OSXJV.Classes.Request.Data [get], [set]
```

To retrieve the document data

Definition at line 101 of file [Request.cs](#).

Referenced by [OSXJV.Server.OSXJVServer.HandlePost\(\)](#).

#### 5.9.5.2 Filename

```
string OSXJV.Classes.Request.Filename [get], [set]
```

To retrieve the filename of the document

Definition at line 69 of file [Request.cs](#).

### 5.9.5.3 Type

```
string OSXJV.Classes.Request.Type [get], [set]
```

To retrieve type of document

Definition at line 85 of file [Request.cs](#).

Referenced by [OSXJV.Server.OSXJVServer.HandlePost\(\)](#).

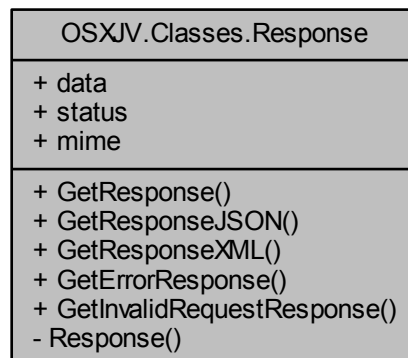
The documentation for this class was generated from the following file:

- OSXJVClasses/[Request.cs](#)

## 5.10 OSXJV.Classes.Response Class Reference

The Object containing data to send to the client

Collaboration diagram for OSXJV.Classes.Response:



### Static Public Member Functions

- static [Response GetResponse](#) (int [status](#), string type, byte[] [data](#))  
*A custom response object*
- static [Response GetResponseJSON](#) (int [status](#), byte[] [data](#))  
*Return an application/json response*
- static [Response GetResponseXML](#) (int [status](#), byte[] [data](#))  
*Return an text/xml response*
- static [Response GetErrorResponse](#) (string message)  
*Return a error response object*
- static [Response GetInvalidRequestResponse](#) ()  
*Returns an invalid response object*

## Public Attributes

- byte [] [data](#) = null  
*Data*
- int [status](#)  
*Status Code*
- string [mime](#)  
*Data type e.g. "application/json"*

## Private Member Functions

- [Response](#) (int [status](#), string [mime](#), byte[] [buffer](#))  
*Constructor*

### 5.10.1 Detailed Description

The Object containing data to send to the client

Definition at line 9 of file [Response.cs](#).

### 5.10.2 Constructor & Destructor Documentation

#### 5.10.2.1 Response()

```
OSXJV.Classes.Response.Response (
    int status,
    string mime,
    byte [] buffer ) [private]
```

#### Constructor

##### Parameters

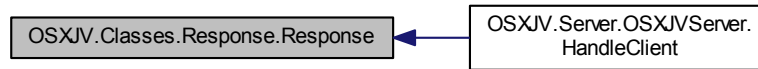
<i>status</i>	Status Code
<i>mime</i>	MIME type
<i>buffer</i>	Data

Definition at line 34 of file [Response.cs](#).

Referenced by [OSXJV.Server.OSXJVServer.HandleClient\(\)](#).

```
00035         {
00036             this.status = status;
00037             this.data = buffer;
00038             this.mime = mime;
00039         }
```

Here is the caller graph for this function:



### 5.10.3 Member Function Documentation

#### 5.10.3.1 GetErrorResponse()

```
static Response OSXJV.Classes.Response.GetErrorResponse (
    string message ) [static]
```

Return a error response object

##### Returns

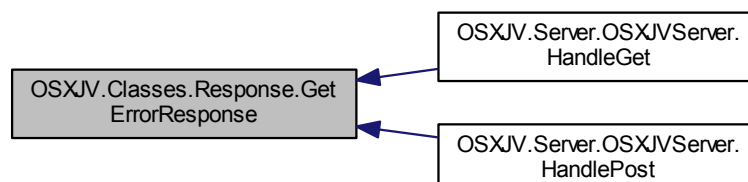
New response object

Definition at line 118 of file [Response.cs](#).

Referenced by [OSXJV.Server.OSXJVServer.HandleGet\(\)](#), and [OSXJV.Server.OSXJVServer.HandlePost\(\)](#).

```
00119     {
00120         byte[] res = Encoding.UTF8.GetBytes(message);
00121         return new Response(400, "text/html", res);
00122     }
```

Here is the caller graph for this function:



## 5.10.3.2 GetInvalidRequestResponse()

```
static Response OSXJV.Classes.Response.GetInvalidRequestResponse ( ) [static]
```

Returns an invalid response object

## Returns

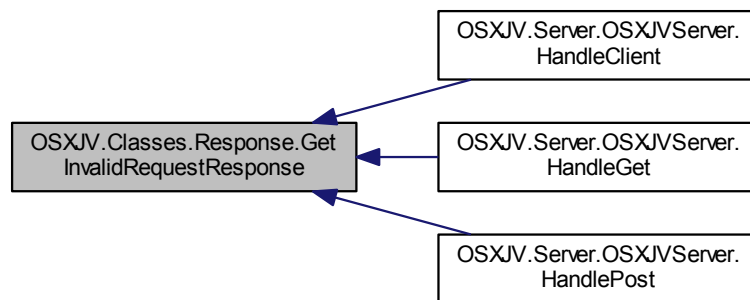
New response object

Definition at line 128 of file [Response.cs](#).

Referenced by [OSXJV.Server.OSXJVServer.HandleClient\(\)](#), [OSXJV.Server.OSXJVServer.HandleGet\(\)](#), and [OSXJV.Server.OSXJVServer.HandlePost\(\)](#).

```
00129         {
00130             return new Response(405, "text/html", new byte[0]);
00131         }
```

Here is the caller graph for this function:



## 5.10.3.3 GetResponse()

```
static Response OSXJV.Classes.Response.GetResponse (
    int status,
    string type,
    byte [] data ) [static]
```

A custom response object

## Parameters

<i>status</i>	The HTTP Code to send back e.g. 200 for success
<i>type</i>	Data type to send back e.g. application/json
<i>data</i>	The data to send

## Returns

Definition at line 48 of file [Response.cs](#).

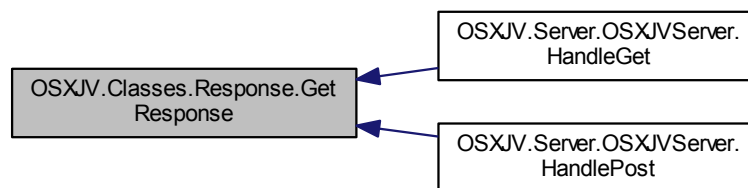
Referenced by [OSXJV.Server.OSXJVServer.HandleGet\(\)](#), and [OSXJV.Server.OSXJVServer.HandlePost\(\)](#).

```

00049     {
00050         if(string.IsNullOrEmpty(type))
00051             throw new ArgumentException("Type cannot be Null or empty");
00052
00053         if (status.Equals(null))
00054             throw new ArgumentException("Status cannot be Null");
00055         else
00056             if (status == 0)
00057                 throw new ArgumentException("Status cannot be 0");
00058
00059         if (data == null)
00060             throw new ArgumentException("Data cannot be null");
00061         else
00062             if (data.Length == 0)
00063                 throw new ArgumentException("No data, use invalid or error response");
00064
00065         return new Response(status, type, data);
00066     }

```

Here is the caller graph for this function:



#### 5.10.3.4 GetResponseJSON()

```

static Response OSXJV.Classes.Response.GetResponseJSON (
    int status,
    byte [] data ) [static]

```

Return an application/json response

## Parameters

<i>status</i>	The HTTP Code to send back e.g. 200 for success
<i>data</i>	The data to send

## Returns

New response object

Definition at line 74 of file [Response.cs](#).

```
00075     {
00076         if (status.Equals(null))
00077             throw new ArgumentException("Status cannot be Null");
00078         else
00079             if (status == 0)
00080                 throw new ArgumentException("Status cannot be 0");
00081
00082         if (data == null)
00083             throw new ArgumentException("Data cannot be null");
00084         else
00085             if (data.Length == 0)
00086                 throw new ArgumentException("No data, use invalid or error response");
00087
00088         return new Response(status, "application/json", data);
00089     }
```

### 5.10.3.5 GetResponseXML()

```
static Response OSXJV.Classes.Response.GetResponseXML (
    int status,
    byte [] data ) [static]
```

Return an text/xml response

## Parameters

<i>status</i>	The HTTP Code to send back e.g. 200 for success
<i>data</i>	The data to send

## Returns

New response object

Definition at line 97 of file [Response.cs](#).

```
00098     {
00099         if (status.Equals(null))
00100             throw new ArgumentException("Status cannot be Null");
00101         else
00102             if (status == 0)
00103                 throw new ArgumentException("Status cannot be 0");
00104
00105         if (data == null)
00106             throw new ArgumentException("Data cannot be null");
00107         else
00108             if (data.Length == 0)
00109                 throw new ArgumentException("No data, use invalid or error response");
00110
00111         return new Response(status, "text/xml", data);
00112     }
```

## 5.10.4 Member Data Documentation

#### 5.10.4.1 data

```
byte [] OSXJV.Classes.Response.data = null
```

Data

Definition at line 14 of file [Response.cs](#).

Referenced by [OSXJV.Server.OSXJVServer.Post\(\)](#).

#### 5.10.4.2 mime

```
string OSXJV.Classes.Response.mime
```

Data type e.g. "application/json"

Definition at line 24 of file [Response.cs](#).

Referenced by [OSXJV.Server.OSXJVServer.Post\(\)](#).

#### 5.10.4.3 status

```
int OSXJV.Classes.Response.status
```

Status Code

Definition at line 19 of file [Response.cs](#).

Referenced by [OSXJV.Server.OSXJVServer.Post\(\)](#).

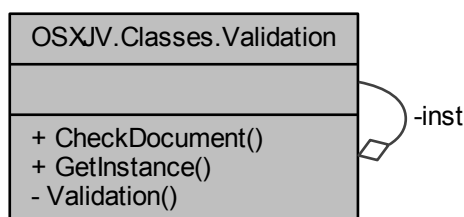
The documentation for this class was generated from the following file:

- OSXJVClasses/[Response.cs](#)

## 5.11 OSXJV.Classes.Validation Class Reference

Preform validation

Collaboration diagram for OSXJV.Classes.Validation:





## Public Member Functions

- bool [CheckDocument](#) (string data, string type)  
*Checks the document and if it is valid*

## Static Public Member Functions

- static [Validation GetInstance](#) ()  
*Gets the instance.*

## Private Member Functions

- [Validation](#) ()  
*Constructor*

## Static Private Attributes

- static [Validation inst](#)  
*The inst.*

### 5.11.1 Detailed Description

Preform validation

Definition at line 12 of file [Validation.cs](#).

### 5.11.2 Constructor & Destructor Documentation

#### 5.11.2.1 Validation()

```
OSXJV.Classes.Validation.Validation ( ) [private]
```

Constructor

Definition at line 22 of file [Validation.cs](#).

```
00022 {}
```

### 5.11.3 Member Function Documentation

#### 5.11.3.1 CheckDocument()

```
bool OSXJV.Classes.Validation.CheckDocument (
    string data,
    string type )
```

Checks the document and if it is valid

**Parameters**

<i>data</i>	Document contents
<i>type</i>	Type of document

**Returns**

True if valid, else false

**Exceptions**

<i>ArgumentException</i>	Invalid data type or data and type cannot be null
<i>XmlException</i>	Invalid XML or HTML
<i>JsonReaderException</i>	Invalid JSON

Definition at line 44 of file [Validation.cs](#).

Referenced by [OSXJV.Server.OSXJVServer.HandlePost\(\)](#).

```
00045     {
00046         if(string.IsNullOrEmpty(data) || string.IsNullOrEmpty(type))
00047         {
00048             throw new ArgumentException("Data or Type cannot be Null");
00049         }
00050
00051         if (type.Equals("XML") || type.Equals("HTML"))
00052         {
00053             XmlReaderSettings settings = new XmlReaderSettings();
00054             settings.DtdProcessing = DtdProcessing.Parse;
00055             settings.MaxCharactersFromEntities = 2048;
00056             using (XmlReader xr = XmlReader.Create(new StringReader(data), settings))
00057             {
00058                 try
00059                 {
00060                     while (xr.Read()) { }
00061                     return true;
00062                 }
00063                 catch (XmlException ex)
00064                 {
00065                     throw ex;
00066                 }
00067             }
00068         }
00069         else if(type.Equals("JSON"))
00070         {
00071             try
00072             {
00073                 JToken.Parse(data);
00074                 return true;
00075             }
00076             catch (JsonReaderException ex)
00077             {
00078                 throw new JsonReaderException(ex.Message);
00079             }
00080         }
00081         throw new ArgumentException("Invalid data or type");
00082     }
00083 }
```

Here is the caller graph for this function:



### 5.11.3.2 GetInstance()

```
static Validation OSXJV.Classes.Validation.GetInstance ( ) [static]
```

Gets the instance.

#### Returns

The instance.

Definition at line 28 of file [Validation.cs](#).

```
00029      {
00030          if (inst != null)
00031              return inst;
00032          else
00033              return (inst = new Validation ());
00034      }
```

## 5.11.4 Member Data Documentation

### 5.11.4.1 inst

```
Validation OSXJV.Classes.Validation.inst [static], [private]
```

The inst.

Definition at line 17 of file [Validation.cs](#).

The documentation for this class was generated from the following file:

- [OSXJVClasses/Validation.cs](#)



## Chapter 6

# File Documentation

### 6.1 OSXJVClasses/Attribute.cs File Reference

#### Classes

- class [OSXJV.Classes.Attribute](#)

#### Namespaces

- namespace [OSXJV.Classes](#)

### 6.2 Attribute.cs

```
00001 namespace OSXJV.Classes
00002 {
00006     public class Attribute
00007     {
00008         private string name;
00009         private string value;
00010
00014         public string Name
00015         {
00016             get
00017             {
00018                 return name;
00019             }
00020
00021             set
00022             {
00023                 name = value;
00024             }
00025         }
00029         public string Value
00030         {
00031             get
00032             {
00033                 return value;
00034             }
00035
00036             set
00037             {
00038                 this.value = value;
00039             }
00040         }
00041     }
00042 }
```

## 6.3 OSXJVClasses/CacheManager.cs File Reference

### Classes

- class [OSXJV.Classes.CacheManager](#)  
*Manages Saving an Retrieving Filesexi*

### Namespaces

- namespace [OSXJV.Classes](#)

## 6.4 CacheManager.cs

```

00001 using System;
00002 using System.IO;
00003
00004 namespace OSXJV.Classes
00005 {
00006     public class CacheManager
00007     {
00008         private static CacheManager inst;
00009         private string path;
00010
00011         private CacheManager(string path)
00012         {
00013             this.path = path;
00014         }
00015
00016         public static bool Setup(string path)
00017         {
00018             if (string.IsNullOrEmpty(path))
00019                 throw new ArgumentException("Path cannot be empty");
00020
00021             if (!Directory.Exists(string.Format(@"{0}", path)))
00022                 throw new Exception("Path is not a valid cache directory");
00023
00024             return (inst = new CacheManager(path)) != null ? true : false;
00025         }
00026
00027         public static CacheManager GetInstance()
00028         {
00029             if (inst != null)
00030                 return inst;
00031             else
00032                 throw new Exception("CacheManger has not been setup");
00033         }
00034
00035         public string getFile(string ID)
00036         {
00037             if (string.IsNullOrEmpty(ID))
00038                 throw new ArgumentException("ID cannot be null or empty");
00039
00040             string filePath = path + "/" + ID.Replace("/", "") + ".json";
00041             string output = "";
00042
00043             using (StreamReader sr = new StreamReader(filePath))
00044             {
00045                 output = sr.ReadToEnd();
00046             }
00047
00048             if (!string.IsNullOrEmpty(output))
00049                 return output;
00050             else
00051                 throw new Exception("Error Reading From File");
00052         }
00053
00054         public bool saveFile(string ID, string nodes)
00055         {
00056             if (string.IsNullOrEmpty(ID))
00057                 throw new ArgumentException("ID cannot be null or empty");
00058
00059             if (string.IsNullOrEmpty(nodes))
00060                 throw new ArgumentException("Document cannot be null or empty");
00061         }
00062     }
00063 }

```

```

00082
00083         string filePath = path + "/" + ID + ".json";
00084         try
00085         {
00086             using (StreamWriter sw = new StreamWriter(filePath))
00087             {
00088                 sw.WriteLine(nodes);
00089             }
00090         }
00091         catch
00092         {
00093             throw new Exception("Failed to save file");
00094         }
00095
00096         return true;
00097     }
00098
00099     public static void Close()
00100     {
00101         if (inst == null)
00102             throw new Exception("CacheManager Already Closed");
00103         else
00104             inst = null;
00105     }
00106 }
00107 }

```

## 6.5 OSXJVClasses/Logger.cs File Reference

### Classes

- class [OSXJV.Classes.Logger](#)  
A simple class that writes errors to a single file.

### Namespaces

- namespace [OSXJV.Classes](#)

## 6.6 Logger.cs

```

00001 using System;
00002 using System.IO;
00003
00004 namespace OSXJV.Classes
00005 {
00009     public class Logger
00010     {
00014         private static Logger inst;
00015         private string location;
00016
00017         private Logger(string location)
00018         {
00019             this.location = location;
00020         }
00021
00026         public static bool Setup(string location)
00027         {
00028             if (string.IsNullOrEmpty(location))
00029                 throw new ArgumentException("Location cannot be empty");
00030
00031             if (!Directory.Exists(string.Format(@"{0}", location)))
00032                 throw new Exception("Location is not a valid logger directory");
00033
00034             return (inst = new Logger(location)) != null ? true:false;
00035         }
00036
00041         public static Logger GetInstance()
00042         {
00043             if (inst != null)
00044                 return inst;

```

```

00045         else
00046             throw new Exception("Logger has not been setup");
00047     }
00048
00053     public void WriteError(string error)
00054     {
00055         try
00056         {
00057             if (!string.IsNullOrEmpty(error))
00058             {
00059                 string file = string.Format(@"{0}/Error-{1}.txt", location, DateTime.Now.ToString("
dd-MM-yy hh-MM-ss"));
00060                 StreamWriter sw = new StreamWriter(file);
00061                 sw.WriteLine(error);
00062                 sw.WriteLine();
00063                 sw.Close();
00064             }
00065         }
00066         catch (IOException e)
00067         {
00068             throw e;
00069         }
00070     }
00071
00072     public static void Close()
00073     {
00074         if (inst == null)
00075             throw new Exception("Logger Already Closed");
00076         else
00077             inst = null;
00078     }
00079 }
00080 }

```

## 6.7 OSXJVClasses/Node.cs File Reference

### Classes

- class [OSXJV.Classes.Node](#)  
*Contain Processed Document Information*

### Namespaces

- namespace [OSXJV.Classes](#)

## 6.8 Node.cs

```

00001 using System.Collections.Generic;
00002 using Newtonsoft.Json.Serialization;
00003 using Newtonsoft.Json;
00004
00005 namespace OSXJV.Classes
00006 {
00010     public class Node
00011     {
00012         private string name;
00013         private List<Attribute> attributes;
00014         private string value;
00015         private List<Node> children;
00016         private int number;
00017         private bool visited;
00018         private List<string> comments;
00019
00023         public Node()
00024         {
00025             Attributes = new List<Attribute>();
00026             Children = new List<Node>();
00027             Comments = new List<string>();
00028             number = 0;
00029             visited = false;

```



```
00030     }
00031
00032     public int Number
00033     {
00034         get
00035         {
00036             return number;
00037         }
00038
00039         set
00040         {
00041             number = value;
00042         }
00043     }
00044
00045     public string Name
00046     {
00047         get
00048         {
00049             return name;
00050         }
00051
00052         set
00053         {
00054             name = value;
00055         }
00056     }
00057
00058     [JsonProperty(NullValueHandling = NullValueHandling.Ignore)]
00059     public string Value
00060     {
00061         get
00062         {
00063             return value;
00064         }
00065
00066         set
00067         {
00068             this.value = value;
00069         }
00070     }
00071
00072     public List<string> Comments
00073     {
00074         get
00075         {
00076             return comments;
00077         }
00078
00079         set
00080         {
00081             comments = value;
00082         }
00083     }
00084
00085     [JsonProperty()]
00086     public List<Attribute> Attributes
00087     {
00088         get
00089         {
00090             return attributes;
00091         }
00092
00093         set
00094         {
00095             attributes = value;
00096         }
00097     }
00098
00099     [JsonProperty()]
00100     public List<Node> Children
00101     {
00102         get
00103         {
00104             return children;
00105         }
00106
00107         set
00108         {
00109             children = value;
00110         }
00111     }
00112
00113     [Newtonsoft.Json.JsonIgnore]
00114     public bool Visited
00115     {
00116         get
```

```

00138         {
00139             return visited;
00140         }
00141
00142         set
00143         {
00144             visited = value;
00145         }
00146     }
00147 }
00148 }

```

## 6.9 OSXJVClasses/Output.cs File Reference

### Classes

- class [OSXJV.Classes.Output](#)  
*Creates the [Output](#) for the web page to display.*

### Namespaces

- namespace [OSXJV.Classes](#)

## 6.10 Output.cs

```

00001 using Newtonsoft.Json.Linq;
00002 using System;
00003 using System.Collections.Generic;
00004 using System.Threading;
00005
00006 namespace OSXJV.Classes
00007 {
00011     public class Output
00012     {
00016         private int left = 100, top = 130;
00017
00021         private Node nodes;
00022
00026         private bool GotParent = false;
00027
00031         private int Parent = 0;
00032
00036         private List<Tuple<int, string>> cNodes = new List<Tuple<int, string>>();
00037
00042         public Output(Node nodes)
00043         {
00044             if (nodes == null)
00045                 throw new ArgumentException();
00046             this.nodes = nodes;
00047         }
00048
00053         public JObject CreateGrid()
00054         {
00055             JObject obj = new JObject();
00056             obj.Add("text", nodes.Name);
00057             obj.Add("id", nodes.Number);
00058             obj.Add("state", new JObject(new JProperty("selected", true)));
00059
00060             if(nodes.Children.Count > 0)
00061             {
00062                 JArray array = new JArray();
00063                 foreach (Node n2 in nodes.Children)
00064                 {
00065                     array.Add(GridGetChidren(n2));
00066                 }
00067                 obj.Add("children", array);
00068             }
00069             return obj;
00070         }
00071     }

```

```

00077     private JObject GridGetChidren(Node n)
00078     {
00079         JObject child = new JObject();
00080         child.Add("id", n.Number);
00081         child.Add("text", n.Name);
00082
00083         if (n.Children.Count > 0)
00084         {
00085             JArray array = new JArray();
00086             foreach(Node n2 in n.Children)
00087             {
00088                 array.Add(GridGetChidren(n2));
00089             }
00090             child.Add("children", array);
00091         }
00092         return child;
00093     }
00101     public string CreateViewSingle(int node, int nodeStart = 0)
00102     {
00103         string output = "<div class='text-center ui-layout-center ui-layout-pane
00104 ui-layout-pane-center'><div style ='display:inline-block' class='ui-selectable ui-droppable'>";
00105
00106         if (nodes.Number.Equals(node))
00107         {
00108             int count = 0;
00109             output += CreateNodeView(nodes, "node");
00110
00111             foreach (Node n in nodes.Children)
00112             {
00113                 if (nodeStart > 0)
00114                 {
00115                     if (count != nodeStart)
00116                         continue;
00117                 }
00118                 count++;
00119                 output += CreateNodeView(n, "node-child"); //Child(Nodes) Thread
00120
00121                 if ((count-nodeStart) == 200)
00122                 {
00123                     output += CreateExtraNode("node-child", count);
00124                     break;
00125                 }
00126             }
00127         }
00128         else
00129         {
00130             GetParent(nodes, node);
00131             string temp = "";
00132             if (GotParent)
00133             {
00134                 if (nodes.Number == Parent)
00135                 {
00136                     output += CreateNodeView(nodes, "node-parent");
00137                 }
00138             }
00139             foreach (Node n2 in nodes.Children)
00140             {
00141                 if (GotParent)
00142                 {
00143                     if (n2.Number == Parent)
00144                     {
00145                         output += CreateNodeView(n2, "node-parent");
00146                     }
00147                 }
00148                 temp += CheckChildren(n2, node);
00149             }
00150             if (!string.IsNullOrEmpty(temp))
00151                 output += temp;
00152         }
00153         output += "</div></div>"; //Close out divs
00154         return output;
00155     }
00156
00157     private string CreateExtraNode(string type, int id)
00158     {
00159         string node = "";
00160
00161         if (type == "node")
00162         {
00163             if (GotParent)
00164             {
00165                 left = left + 400;
00166             }
00167         }
00168         if (type == "node-child")

```

```

00176         {
00177             left = left + 400;
00178         }
00179
00180         node += "<div class='node-child type ui-draggable ui-selectee' style='left:" + left + "px; top:"
" + top + "px;margin-bottom:50px;'>";
00181         node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
)'>Show Lower</button></span></div>";
00182         node += "</div></div>";
00183         return node;
00184     }
00185
00194     private string CreatePreviousNode(string type, int leftVal, int topVal, int id)
00195     {
00196         string node = "";
00197
00198         if (type == "node")
00199         {
00200             if (GotParent)
00201             {
00202                 leftVal = leftVal + 400;
00203             }
00204         }
00205         if (type == "node-child")
00206         {
00207             leftVal = leftVal + 400;
00208         }
00209         node += "<div class='node-child type ui-draggable ui-selectee' style='left:" + leftVal + "px;
top:" + topVal + "px;'>";
00210         node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
)'>Show Higher</button></span></div>";
00211         node += "</div></div>";
00212         return node;
00213     }
00214
00223     private string CreateExtraNode(string type, int leftVal, int topVal,int id)
00224     {
00225         string node = "";
00226
00227         if (type == "node")
00228         {
00229             if (GotParent)
00230             {
00231                 leftVal = leftVal + 400;
00232             }
00233         }
00234         if (type == "node-child")
00235         {
00236             leftVal = leftVal + 400;
00237         }
00238         node += "<div class='node-child type ui-draggable ui-selectee' style='left:" + leftVal + "px;
top:" + topVal + "px;margin-bottom:50px;'>";
00239         node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
)'>Show Lower</button></span></div>";
00240         node += "</div></div>";
00241         return node;
00242     }
00243
00252     private void CreateNodeChildViewsParallel(List<Node> job,int start,
bool showHigher,int next,int previous)
00253     {
00254         int threadID = int.Parse(Thread.CurrentThread.Name);
00255         string type = "node-child";
00256         string output = "";
00257
00258         if(start == 0 && showHigher)
00259         {
00260             output += CreatePreviousNode(type, left, top, previous);
00261         }
00262         bool hadCommentsPrev = false;
00263         int numCommentsPrevious = 0;
00264
00265         foreach(Node n in job)
00266         {
00267             int extra = showHigher ? 130 * (start +1) : 130 * start;
00268
00269             if (hadCommentsPrev)
00270                 extra += (numCommentsPrevious * 25);
00271
00272             if (n.Comments.Count > 0)
00273             {
00274                 hadCommentsPrev = true;
00275                 numCommentsPrevious = n.Comments.Count;
00276             }
00277             else
00278                 hadCommentsPrev = false;
00279

```

```

00280         output += CreateNodeView(n, type, left, top + extra);
00281         start++;
00282         if (start == 200)
00283         {
00284             output += CreateExtraNode(type, left, top + extra + 130, next);
00285             break;
00286         }
00287     }
00288 }
00289
00290 cNodes.Add(new Tuple<int, string>(threadID, output));
00291 }
00292
00300 public string CreateView(int node = 1, int pCount = 4, int nodeStart = 0) //Setting
Defaults
00301 {
00302
00303     List<Thread> threadList = new List<Thread>();
00304
00305     string output = "<div class='text-center ui-layout-center ui-layout-pane
ui-layout-pane-center'><div style ='display:inline-block' class='ui-selectable ui-droppable'>";
00306     if (nodes.Number.Equals(node))
00307     {
00308         int childCount = 0;
00309
00310         if (nodes.Children.Count < 200)
00311             childCount = nodes.Children.Count;
00312         else
00313         {
00314             childCount = 200;
00315         }
00316
00317         if (childCount < pCount * 2)
00318         {
00319             output += CreateNodeView(nodes, "node", left, top);
00320             foreach (Node n2 in nodes.Children)
00321             {
00322                 output += CreateNodeView(n2, "node-child");
00323             }
00324         }
00325         else
00326         {
00327             int spread = (int) Math.Ceiling((double) childCount / (double) pCount);
00328
00329             output += CreateNodeView(nodes, "node", left, top); //Parent (Node) Thread
00330
00331             for (int i = 0; i < pCount; i++)
00332             {
00333                 int neg = 0;
00334                 if ((spread * (i + 1)) > childCount)
00335                 {
00336                     neg = childCount - (spread * (i + 1));
00337                 }
00338                 int start = (spread * i);
00339                 int rangeStart = (spread * i) + nodeStart;
00340                 bool showHigher = nodeStart != 0 ? true : false;
00341
00342                 List<Node> NodesToProcess = nodes.Children.GetRange(rangeStart, spread +
00343 neg);
00344                 Thread threadJob = new Thread(() => CreateNodeChildViewsParallel(NodesToProcess,
00345 start, showHigher, childCount + nodeStart, nodeStart - childCount));
00346                 threadJob.Name = i.ToString();
00347                 threadJob.Start();
00348                 threadList.Add(threadJob);
00349             }
00350             foreach (Thread t in threadList)
00351             {
00352                 t.Join();
00353             }
00354             cNodes.Sort((x, y) => x.Item1.CompareTo(y.Item1));
00355
00356             foreach (Tuple<int, string> tup in cNodes)
00357             {
00358                 output += tup.Item2;
00359             }
00360         }
00361     }
00362     else
00363     {
00364         GetParent(nodes, node);
00365         string temp = "";
00366         if (GotParent)
00367         {
00368             if (nodes.Number == Parent)
00369             {
00370                 output += CreateNodeView(nodes, "node-parent");
00371             }
00372         }
00373     }
00374 }

```

```

00370         }
00371     }
00372     bool found = false;
00373     foreach (Node n2 in nodes.Children)
00374     {
00375         if (GotParent)
00376         {
00377             if (n2.Number == Parent)
00378             {
00379                 output += CreateNodeView(n2, "node-parent");
00380             }
00381         }
00382         temp += CheckChildren(n2, node, pCount, nodeStart, ref found);
00383         if (found)
00384             break;
00385     }
00386     if (!string.IsNullOrEmpty(temp))
00387         output += temp;
00388 }
00389 output += "</div></div>";
00390 return output;
00391 }
00392
00399 private string CheckChildren(Node n, int number)
00400 {
00401     string output = "";
00402     if (CheckNodeNumber(n, number))
00403     {
00404         int count = 0;
00405         output += CreateNodeView(n, "node");
00406         foreach (Node n2 in n.Children)
00407         {
00408             count++;
00409             output += CreateNodeView(n2, "node-child");
00410         }
00411     }
00412 }
00413 else if (n.Children.Count > 0)
00414 {
00415     foreach (Node n2 in n.Children)
00416     {
00417         if (GotParent)
00418         {
00419             if (n2.Number == Parent)
00420             {
00421                 output += CreateNodeView(n2, "node-parent");
00422             }
00423         }
00424         output += CheckChildren(n2, number);
00425     }
00426 }
00427 return output;
00428 }
00429
00430 private string CheckChildren(Node n, int number, int pCount, int nodeStart, ref
00440 bool found)
00441 {
00442     string output = "";
00443     if (CheckNodeNumber(n, number))
00444     {
00445         found = true;
00446         List<Thread> threadList = new List<Thread>();
00447
00448         int count = 0;
00449         output += CreateNodeView(n, "node");
00450         count++;
00451         //output += CreateNodeView(n2, "node-child");
00452         int childCount = 0;
00453
00454         if (n.Children.Count < 200)
00455             childCount = n.Children.Count;
00456         else
00457         {
00458             childCount = 200;
00459         }
00460         if (childCount < pCount * 2)
00461         {
00462             foreach (Node n2 in n.Children)
00463             {
00464                 output += CreateNodeView(n2, "node-child");
00465             }
00466         }
00467         else
00468         {
00469             int spread = (int)Math.Ceiling((double)childCount / (double)pCount);
00470

```

```

00471         if (childCount > 0)
00472         {
00473             for (int i = 0; i < pCount; i++)
00474             {
00475                 int neg = 0;
00476                 if ((spread * (i + 1)) > childCount)
00477                 {
00478                     neg = childCount - (spread * (i + 1));
00479                 }
00480                 int start = (spread * i);
00481                 int rangeStart = (spread * i) + nodeStart;
00482                 bool showHigher = nodeStart != 0 ? true : false;
00483
00484                 List<Node> NodesToProcess = n.Children.GetRange(rangeStart, spread +
neg);
00485
00486                 if (NodesToProcess.Count > 0)
00487                 {
00488                     Thread threadJob = new Thread(() => CreateNodeChildViewsParallel(
NodesToProcess, start, showHigher, childCount + nodeStart, nodeStart - childCount));
00489                     threadJob.Name = i.ToString();
00490                     threadJob.Start();
00491                     threadList.Add(threadJob);
00492                 }
00493             }
00494             foreach (Thread t in threadList)
00495             {
00496                 t.Join();
00497             }
00498             cNodes.Sort((x, y) => x.Item1.CompareTo(y.Item1));
00499
00500             foreach (Tuple<int, string> tup in cNodes)
00501             {
00502                 output += tup.Item2;
00503             }
00504         }
00505     }
00506 }
00507 else if (n.Children.Count > 0)
00508 {
00509     foreach (Node n2 in n.Children)
00510     {
00511         if (GotParent)
00512         {
00513             if (n2.Number == Parent)
00514             {
00515                 output += CreateNodeView(n2, "node-parent");
00516             }
00517         }
00518         output += CheckChildren(n2, number, pCount, nodeStart, ref found);
00519     }
00520 }
00521 }
00522 return output;
00523 }
00524
00530 private void GetParent(Node node, int number)
00531 {
00532     if (!CheckNodeNumber(node, number))
00533     {
00534         foreach (Node n in node.Children)
00535         {
00536             if (CheckNodeNumber(n, number))
00537             {
00538                 Parent = node.Number;
00539                 GotParent = true;
00540             }
00541             else
00542             {
00543                 GetParent(n, number);
00544             }
00545         }
00546     }
00547 }
00548
00557 private string CreateNodeView(Node n, string type, int leftVal, int topVal)
00558 {
00559     string node = "";
00560
00561     if (type == "node")
00562     {
00563         if (GotParent)
00564         {
00565             leftVal = leftVal + 400;
00566         }
00567     }
00568     if (type == "node-child")

```

```

00569         {
00570             leftVal = leftVal + 400;
00571         }
00572         node += "<div id='" + n.Number + "' class='" + type + " type ui-draggable ui-selectee'
style='left:" + leftVal + "px; top:" + topVal + "px;'>";
00573         node += "<div class='head'><span><button class='nameBtn' onclick='GetNode(\"+n.
Number+')'>>\" + n.Name + "</button></span></div>";
00574         if (!string.IsNullOrEmpty(n.Value))
00575         {
00576             node += string.Format("<div class='blockR'><p>Value</p></div><div
class=comment><span>{0}</span></div>", n.Value);
00577         }
00578         if (n.Comments.Count > 0)
00579         {
00580             node += "<div><p class='text-center'>Comments</p></div>";
00581
00582             foreach (string com in n.Comments)
00583             {
00584                 node += "<div class='comment'>" + com + "</div>";
00585             }
00586         }
00587         if (n.Attributes.Count > 0)
00588         {
00589             node += "<div class='attribute'><div class='aHeader'><p><button><i class='fa
fa-plus'></i></button>Attributes</p></div><div class='options'>";
00590             foreach (Attribute a in n.Attributes)
00591             {
00592                 node += string.Format("<div class='blockR'><p>{0}</p></div><div
class='comment'><p>{1}</p></div>", a.Name, a.Value);
00593             }
00594             node += "</div>";
00595         }
00596         node += "</div></div>";
00597         return node;
00598     }
00599
00600     private string CreateNodeView(Node n, string type)
00601     {
00602         string node = "";
00603         int leftVal = left;
00604         if (type == "node")
00605         {
00606             if (GotParent)
00607             {
00608                 left = left + 400;
00609                 leftVal = left;
00610             }
00611         }
00612         if (type == "node-child")
00613         {
00614             leftVal = leftVal + 400;
00615         }
00616         node += "<div id='" + n.Number + "' class='" + type + " type ui-draggable ui-selectee'
style='left:" + leftVal + "px; top:" + top + "px;'>";
00617         node += "<div class='head'><span><button class='nameBtn' onclick='GetNode(\" + n.
Number + \")'>>\" + n.Name + "</button></span></div>";
00618         if (!string.IsNullOrEmpty(n.Value))
00619         {
00620             node += string.Format("<div class='blockR'><p>Value</p></div><div
class=comment><span>{0}</span></div>", n.Value);
00621         }
00622         if (n.Comments.Count > 0)
00623         {
00624             node += "<div><p class='text-center'>Comments</p></div>";
00625
00626             foreach (string com in n.Comments)
00627             {
00628                 node += "<div class='comment'>" + com + "</div>";
00629             }
00630         }
00631         if (n.Attributes.Count > 0)
00632         {
00633             node += "<div class='attribute'><div class='aHeader'><p><button><i class='fa
fa-plus'></i></button>Attributes</p></div><div class='options'>";
00634             foreach (Attribute a in n.Attributes)
00635             {
00636                 node += string.Format("<div class='blockR'><p>{0}</p></div><div
class='comment'><p>{1}</p></div>", a.Name, a.Value);
00637             }
00638             node += "</div>";
00639         }
00640         node += "</div></div>";
00641
00642         if (type == "node-child")
00643         {
00644             top = top + 130;
00645         }
00646     }

```



```

00652         return node;
00653     }
00654
00661     private bool CheckNodeNumber(Node n, int number)
00662     {
00663         return n.Number.Equals(number);
00664     }
00665 }
00666 }

```

## 6.11 OSXJVClasses/ProcessDocument.cs File Reference

### Classes

- class [OSXJV.Classes.ProcessDocument](#)  
*Class the Processes the document*

### Namespaces

- namespace [OSXJV.Classes](#)

## 6.12 ProcessDocument.cs

```

00001 using Newtonsoft.Json;
00002 using System;
00003 using System.Collections.Generic;
00004 using System.Linq;
00005 using System.Text.RegularExpressions;
00006 using System.Threading;
00007 using System.Xml.Linq;
00008
00009 namespace OSXJV.Classes
00010 {
00011
00015     public class ProcessDocument
00016     {
00020         private XDocument document;
00021
00025         private Node node = new Node();
00026
00030         private List<Tuple<Node, int>> ProcessedElements = new List<Tuple<Node, int>>();
00031
00035         private List<Thread> ThreadList = new List<Thread>();
00036
00040         private string type;
00041         private Thread th;
00042
00046         int count;
00047
00053         private ProcessDocument(XDocument doc, string type)
00054         {
00055             document = doc;
00056             this.type = type;
00057         }
00058
00064         private void ProcessComment(XComment e, Node node)
00065         {
00066             string s = "";
00067             s = Regex.Replace(e.Value, @"[^\w\s\.\@-]", "");
00068             node.Comments.Add(s);
00069         }
00070
00077         public static ProcessDocument GetProcess(string data, string type)
00078         {
00079             if (string.IsNullOrEmpty(data) || string.IsNullOrEmpty(type))
00080             {
00081                 throw new ArgumentException();
00082             }
00083             try
00084             {

```

```

00085         XDocument doc = null;
00086         doc = Prepare(data, type);
00087         return new ProcessDocument(doc, type);
00088     }
00089     catch (System.Xml.XmlException e)
00090     {
00091         throw e;
00092     }
00093 }
00094
00095 private void ProcessText(XText e, Node n)
00096 {
00097     n.Value = e.Value;
00098 }
00099
00100 private static XDocument Prepare(string data, string type)
00101 {
00102     if (type.Equals("JSON"))
00103         return new XDocument(JsonConvert.DeserializeXmlNode(data, "Root", false).Root.FirstNode);
00104     else if (type.Equals("XML") || type.Equals("HTML"))
00105         return XDocument.Parse(data);
00106
00107     return null;
00108 }
00109
00110 public Node Process()
00111 {
00112     if (document.Nodes() != null)
00113     {
00114         foreach (XNode n in document.Nodes())
00115         {
00116             switch (n.NodeType)
00117             {
00118                 case System.Xml.XmlNodeType.Element:
00119                     count++;
00120                     ProcessElement(XElement.Parse(n.ToString()), node);
00121                     break;
00122                 case System.Xml.XmlNodeType.Comment:
00123                     ProcessComment(n as XComment, node);
00124                     break;
00125                 case System.Xml.XmlNodeType.Text:
00126                     ProcessText(n as XText, node);
00127                     break;
00128                 case System.Xml.XmlNodeType.Notations:
00129                     break;
00130                 case System.Xml.XmlNodeType.EndElement:
00131                     break;
00132                 default:
00133                     break;
00134             }
00135         }
00136     }
00137     //SortArray(ref node);
00138     document = null;
00139     return node;
00140 }
00141
00142 private Node ProcessElement(XElement e, Node node)
00143 {
00144     if (node.Number == 0)
00145     {
00146         node.Number = count;
00147     }
00148     if (!node.Visited)
00149     {
00150         node.Name = e.Name.LocalName;
00151         foreach (XAttribute ax in e.Attributes())
00152         {
00153             if (ax.Name == "id")
00154             {
00155                 node.Name = node.Name + " #" + ax.Value;
00156             }
00157
00158             if (type == "HTML")
00159             {
00160                 if (ax.IsNamespaceDeclaration)
00161                     continue;
00162             }
00163             Attribute att = new Attribute();
00164             att.Name = ax.Name.LocalName;
00165             att.Value = ax.Value;
00166             node.Attributes.Add(att);
00167         }
00168     }
00169 }

```

```

00193
00194         if (e.Nodes() != null)
00195         {
00196             foreach (XNode n in e.Nodes())
00197             {
00198                 switch (n.NodeType)
00199                 {
00200                     case System.Xml.XmlNodeType.EndElement:
00201                         break;
00202                     case System.Xml.XmlNodeType.Element:
00203                         count++;
00204                         Node n2 = new Node();
00205                         node.Children.Add(ProcessElement(XElement.Parse(n.ToString()), n2));
00206                         break;
00207                     case System.Xml.XmlNodeType.Comment:
00208                         ProcessComment(n as XComment, node);
00209                         break;
00210                     case System.Xml.XmlNodeType.Text:
00211                         ProcessText(n as XText, node);
00212                         break;
00213                     case System.Xml.XmlNodeType.Notations:
00214                         break;
00215
00216                     default:
00217                         break;
00218                 }
00219             }
00220         }
00221         node.Visited = true;
00222         return node;
00223     }
00224
00232 private Node ProcessElement(XElement e, Node node, ref int nodeNumber)
00233 {
00234     if (!node.Visited)
00235     {
00236         if (node.Number == 0)
00237         {
00238             node.Number = nodeNumber;
00239         }
00240         if (!node.Visited)
00241         {
00242
00243             node.Name = e.Name.LocalName;
00244             foreach (XAttribute ax in e.Attributes())
00245             {
00246                 if (ax.Name == "id")
00247                 {
00248                     node.Name = node.Name + " #" + ax.Value;
00249                 }
00250
00251                 if (type == "HTML")
00252                 {
00253                     if (ax.IsNamespaceDeclaration)
00254                         continue;
00255
00256                     Attribute att = new Attribute();
00257                     att.Name = ax.Name.LocalName;
00258                     att.Value = ax.Value;
00259                     node.Attributes.Add(att);
00260                 }
00261             }
00262
00263             if (e.Nodes() != null)
00264             {
00265                 foreach (XNode n in e.Nodes())
00266                 {
00267                     switch (n.NodeType)
00268                     {
00269                         case System.Xml.XmlNodeType.EndElement:
00270                             break;
00271                         case System.Xml.XmlNodeType.Element:
00272                             nodeNumber++;
00273                             Node n2 = new Node();
00274                             node.Children.Add(ProcessElement(XElement.Parse(n.ToString()), n2,
ref nodeNumber));
00275
00276                             break;
00277                         case System.Xml.XmlNodeType.Comment:
00278                             ProcessComment(n as XComment, node);
00279                             break;
00280                         case System.Xml.XmlNodeType.Text:
00281                             ProcessText(n as XText, node);
00282                             break;
00283                         case System.Xml.XmlNodeType.Notations:
00284                             break;
00285
00286                         default:

```

```

00286             break;
00287         }
00288     }
00289 }
00290     node.Visited = true;
00291 }
00292     return node;
00293 }
00294
00301 private Node ProcessRoot(XElement e, Node node)
00302 {
00303     node.Number = 1;
00304
00305     if (!node.Visited)
00306     {
00307
00308         node.Name = e.Name.LocalName;
00309         foreach (XAttribute ax in e.Attributes())
00310         {
00311             if (ax.Name == "id")
00312             {
00313                 node.Name = node.Name + " #" + ax.Value;
00314             }
00315
00316             if (type == "HTML")
00317             {
00318                 if (ax.IsNamespaceDeclaration)
00319                     continue;
00320             }
00321             Attribute att = new Attribute();
00322             att.Name = ax.Name.LocalName;
00323             att.Value = ax.Value;
00324             node.Attributes.Add(att);
00325         }
00326     }
00327     node.Visited = true;
00328     return node;
00329 }
00330
00336 private void ProcessDocumentParallelInit(XDocument doc,int start)
00337 {
00338     int nodeNum = start;
00339
00340     Node node = new Node();
00341     if (doc.Root.Nodes() != null)
00342     {
00343         List<XNode> list = doc.Root.Nodes().ToList();
00344         foreach (XNode n in doc.Root.Nodes())
00345         {
00346             switch (n.NodeType)
00347             {
00348                 case System.Xml.XmlNodeType.Element:
00349                     nodeNum++;
00350                     Node n2 = new Node();
00351                     node.Children.Add(ProcessElement(XElement.Parse(n.ToString()), n2, ref
nodeNum));
00352                     break;
00353                 case System.Xml.XmlNodeType.Comment:
00354                     ProcessComment(n as XComment, node);
00355                     break;
00356                 case System.Xml.XmlNodeType.Text:
00357                     ProcessText(n as XText, node);
00358                     break;
00359                 case System.Xml.XmlNodeType.Notations:
00360                     break;
00361                 case System.Xml.XmlNodeType.EndElement:
00362                     break;
00363                 default:
00364                     break;
00365             }
00366         }
00367     }
00368     document = null;
00369     ProcessedElements.Add(new Tuple<Node, int>(node, start));
00370 }
00371
00377 public Node ProcessParallel(int pCount = 4)
00378 {
00379     node = ProcessRoot(document.Root, node);
00380
00381     int nodeCount = document.Root.Nodes().Count();
00382
00383     if (nodeCount <= pCount)
00384     {
00385         return Process();
00386     }
00387     else if (nodeCount > pCount)

```

```

00388         {
00389             List<XNode> List = document.Root.Nodes().ToList();
00390             int spread = 0;
00391             spread = (int)Math.Ceiling((double)nodeCount / (double)pCount);
00392             int totalNodes = 1;
00393             for (int i = 0; i < pCount; i++)
00394             {
00395                 int neg = 0;
00396                 int start = totalNodes;
00397                 if ((spread * (i+1)) > nodeCount)
00398                 {
00399                     neg = nodeCount - (spread * (i + 1));
00400                 }
00401                 List<XNode> list = List.GetRange((spread * i), spread + neg);
00402                 XElement root = new XElement("Root", list);
00403                 XDocument doc = new XDocument(root);
00404                 (th = new Thread(() => ProcessDocumentParallelInit(doc, start))).Start();
00405                 ThreadList.Add(th); //Add to Threads list to keep recored of threads running
00406                 totalNodes += root.Descendants().Count(); //Increment start position.
00407             }
00408             document = null;
00409             foreach (Thread t in ThreadList)
00410             {
00411                 t.Join(); //Wait for threads to join
00412             }
00413             ProcessedElements.Sort((x, y) => x.Item2.CompareTo(y.Item2)); //Sort List by start index so
00414             they are in order.
00415             foreach (Tuple<Node,int> tup in ProcessedElements)
00416             {
00417                 foreach (Node n in tup.Item1.Children)
00418                 {
00419                     node.Children.Add(n);
00420                 }
00421             }
00422             return node;
00423         }
00424     }
00425 }
00426 }
00427 }
00428 }
00429 }
00430 }
00431 }
00432 }
00433 }
00434 }

```

## 6.13 OSXJVClasses/Request.cs File Reference

### Classes

- class [OSXJV.Classes.Request](#)  
A object containing the document to process, filename and type.

### Namespaces

- namespace [OSXJV.Classes](#)

## 6.14 Request.cs

```

00001 using System;
00002
00003 namespace OSXJV.Classes
00004 {
00005     public class Request
00006     {
00007         private string filename;
00008     }
00009 }

```

```

00018     private string type;
00019
00023     private string data;
00024
00031     private Request(string filename, string type, string data)
00032     {
00033         this.filename = filename;
00034         this.type = type;
00035         this.data = data;
00036     }
00037
00045     public static Request GetRequest(string filename, string type, string data)
00046     {
00047         string Type = "";
00048         if (string.IsNullOrEmpty(filename) || string.IsNullOrEmpty(type) || string.IsNullOrEmpty(data))
00049             throw new ArgumentException();
00050         if (type.Equals("text/xml") || type.Equals("application/xml"))
00051         {
00052             Type = "XML";
00053         }
00054         else if (type.Equals("text/html"))
00055         {
00056             Type = "HTML";
00057         }
00058         else if (type.Equals("application/json") || type.Equals("application/octet-stream"))
00059         {
00060             Type = "JSON";
00061         }
00062         return new Request(filename, Type, data);
00063     }
00064
00068     public string Filename
00069     {
00070         get
00071         {
00072             return filename;
00073         }
00074
00075         set
00076         {
00077             filename = value;
00078         }
00079     }
00080
00084     public string Type
00085     {
00086         get
00087         {
00088             return type;
00089         }
00090
00091         set
00092         {
00093             type = value;
00094         }
00095     }
00096
00100     public string Data
00101     {
00102         get
00103         {
00104             return data;
00105         }
00106
00107         set
00108         {
00109             data = value;
00110         }
00111     }
00112 }
00113 }

```

## 6.15 OSXJVClasses/Response.cs File Reference

### Classes

- class [OSXJV.Classes.Response](#)

*The Object containing data to send to the client*

## Namespaces

- namespace [OSXJV.Classes](#)

## 6.16 Response.cs

```

00001 using System;
00002 using System.Text;
00003
00004 namespace OSXJV.Classes
00005 {
00009     public class Response
00010     {
00014         public byte[] data = null;
00015
00019         public int status;
00020
00024         public string mime;
00025         //static string format = "yyyy-MM-dd HH:mm:ss";
00026
00027
00034         private Response(int status,string mime,byte[] buffer)
00035         {
00036             this.status = status;
00037             this.data = buffer;
00038             this.mime = mime;
00039         }
00040
00048         public static Response GetResponse(int status,string type,byte[] data)
00049         {
00050             if(string.IsNullOrEmpty(type))
00051                 throw new ArgumentException("Type cannot be Null or empty");
00052
00053             if (status.Equals(null))
00054                 throw new ArgumentException("Status cannot be Null");
00055             else
00056                 if (status == 0)
00057                     throw new ArgumentException("Status cannot be 0");
00058
00059             if (data == null)
00060                 throw new ArgumentException("Data cannot be null");
00061             else
00062                 if (data.Length == 0)
00063                     throw new ArgumentException("No data, use invalid or error response");
00064
00065             return new Response(status, type, data);
00066         }
00067
00074         public static Response GetResponseJSON(int status,byte[] data)
00075         {
00076             if (status.Equals(null))
00077                 throw new ArgumentException("Status cannot be Null");
00078             else
00079                 if (status == 0)
00080                     throw new ArgumentException("Status cannot be 0");
00081
00082             if (data == null)
00083                 throw new ArgumentException("Data cannot be null");
00084             else
00085                 if (data.Length == 0)
00086                     throw new ArgumentException("No data, use invalid or error response");
00087
00088             return new Response(status,"application/json", data);
00089         }
00090
00097         public static Response GetResponseXML(int status, byte[] data)
00098         {
00099             if (status.Equals(null))
00100                 throw new ArgumentException("Status cannot be Null");
00101             else
00102                 if(status == 0)
00103                     throw new ArgumentException("Status cannot be 0");
00104
00105             if (data == null)
00106                 throw new ArgumentException("Data cannot be null");
00107             else
00108                 if (data.Length == 0)
00109                     throw new ArgumentException("No data, use invalid or error response");
00110
00111             return new Response(status, "text/xml", data);
00112         }

```

```

00113
00118     public static Response GetErrorResponse(string message)
00119     {
00120         byte[] res = Encoding.UTF8.GetBytes(message);
00121         return new Response(400, "text/html", res);
00122     }
00123
00128     public static Response GetInvalidRequestResponse()
00129     {
00130         return new Response(405, "text/html", new byte[0]);
00131     }
00132 }
00133 }

```

## 6.17 OSXJVClasses/Validation.cs File Reference

### Classes

- class [OSXJV.Classes.Validation](#)  
*Perform validation*

### Namespaces

- namespace [OSXJV.Classes](#)

## 6.18 Validation.cs

```

00001 using Newtonsoft.Json;
00002 using Newtonsoft.Json.Linq;
00003 using System;
00004 using System.IO;
00005 using System.Xml;
00006
00007 namespace OSXJV.Classes
00008 {
00012     public class Validation
00013     {
00017         private static Validation inst;
00018
00022         private Validation(){}
00023
00028         public static Validation GetInstance()
00029         {
00030             if (inst != null)
00031                 return inst;
00032             else
00033                 return (inst = new Validation ());
00034         }
00044         public bool CheckDocument(string data, string type)
00045         {
00046             if(string.IsNullOrEmpty(data) || string.IsNullOrEmpty(type))
00047             {
00048                 throw new ArgumentException("Data or Type cannot be Null");
00049             }
00050
00051             if (type.Equals("XML") || type.Equals("HTML"))
00052             {
00053                 XmlReaderSettings settings = new XmlReaderSettings();
00054                 settings.DtdProcessing = DtdProcessing.Parse;
00055                 settings.MaxCharactersFromEntities = 2048;
00056                 using (XmlReader xr = XmlReader.Create(new StringReader(data), settings))
00057                 {
00058                     try
00059                     {
00060                         while (xr.Read()) { }
00061                         return true;
00062                     }
00063                     catch (XmlException ex)
00064                     {
00065                         throw ex;

```



```

00066         }
00067     }
00068 }
00069 else if (type.Equals("JSON"))
00070 {
00071     try
00072     {
00073         JToken.Parse(data);
00074         return true;
00075     }
00076     catch (JsonReaderException ex)
00077     {
00078         throw new JsonReaderException(ex.Message);
00079     }
00080 }
00081
00082 throw new ArgumentException("Invalid data or type");
00083 }
00084 }
00085 }

```

## 6.19 OSXJVServer.cs File Reference

### Classes

- class [OSXJV.Server.OSXJVServer](#)  
*HTTPServer that process the incoming requests.*

### Namespaces

- namespace [OSXJV.Server](#)

## 6.20 OSXJVServer.cs

```

00001 using System;
00002 using System.Text;
00003 using System.Net;
00004 using System.Threading;
00005 using System.IO;
00006 using HttpMultipartParser;
00007 using Newtonsoft.Json.Linq;
00008 using Newtonsoft.Json;
00009 using OSXJV.Classes;
00010
00011 namespace OSXJV.Server
00012 {
00013     public class OSXJVServer
00014     {
00015         private int port = 8082;
00016
00017         public static bool running = false; //sets if the server is currently running
00018         private HttpListener listener;
00019
00020         private Thread serverThread = null;
00021
00022         public OSXJVServer()
00023         {
00024             listener = new HttpListener();
00025             listener.Prefixes.Add("http://localhost:" + port + "/"); //change if need be
00026         }
00027
00028         public bool Start()
00029         {
00030             serverThread = new Thread(new ThreadStart(Run));
00031             try
00032             {
00033                 serverThread.Start();
00034             }
00035             catch
00036             {}
00037         }
00038     }
00039 }

```

```

00052         return serverThread.IsAlive;
00053     }
00054
00058     public bool Stop()
00059     {
00060         if (listener != null)
00061             if (listener.IsListening)
00062                 listener.Abort();
00063
00064         if (serverThread != null)
00065         {
00066             serverThread.Join();
00067             serverThread = null;
00068         }
00069
00070         return serverThread == null ? true : false;
00071     }
00072
00076     public void Run()
00077     {
00078         running = true;
00079         listener.Start();
00080
00081         while (listener.IsListening)
00082         {
00083             Console.WriteLine("Waiting");
00084
00085             //Wait for Listener
00086             IAsyncResult result = listener.BeginGetContext(new AsyncCallback(ListenerCallback),
00087 listener);
00088             result.AsyncWaitHandle.WaitOne();
00089
00090             if (result.CompletedSynchronously)
00091                 Console.WriteLine("Completed Synchronously");
00092
00093             /*
00094              * Old Method of Creating a Thread
00095              */
00096             Thread response = new Thread(() =>
00097             {
00098                 try
00099                 {
00100                     Console.WriteLine("Processing");
00101                     HandleClient(hlc);
00102
00103                     Console.WriteLine("Finished");
00104                 }
00105                 catch (Exception e)
00106                 {
00107                     Logger.GetInstance().WriteError(e.Message);
00108                     hlc.Response.StatusCode = 500;
00109                     hlc.Response.Close();
00110                 }
00111             });
00112             response.Start();
00113             *
00114             */
00115         }
00116     }
00117
00118     //Asynchronous Handler
00119
00124     private void ListenerCallback(IAsyncResult result)
00125     {
00126         HttpListener listener = (HttpListener)result.AsyncState;
00127         HttpContext context = listener.EndGetContext(result);
00128         try
00129         {
00130             HandleClient(context);
00131         }
00132         catch (Exception e)
00133         {
00134             Logger.GetInstance().WriteError(e.Message);
00135             context.Response.StatusCode = 500;
00136             context.Response.Close();
00137         }
00138     }
00139
00140     //Handles the client request
00146     private void HandleClient(HttpContext c)
00147     {
00148         switch (c.Request.HttpMethod)
00149         {
00150             case "POST":
00151                 Post(HandlePost(c.Request), c.Response);

```

```

00152             break;
00153         case "GET":
00154             Post(HandleGet(c.Request), c.Response);
00155             break;
00156         case "OPTIONS":
00157             HandleOptions(c.Response);
00158             c.Response.Close();
00159             break;
00160         default:
00161             Post(Response.GetInvalidRequestResponse(), c.
Response);
00162             break;
00163     }
00164 }
00165
00170 private void HandleOptions(HttpListenerResponse response)
00171 {
00172     response.AddHeader("Access-Control-Allow-Headers", "Content-Type, Accept, X-Requested-With");
00173     response.AddHeader("Access-Control-Allow-Methods", "POST");
00174     response.AddHeader("Access-Control-Allow-Methods", "GET");
00175     response.AddHeader("Access-Control-Max-Age", "1728000");
00176     response.AppendHeader("Access-Control-Allow-Origin", "*");
00177 }
00178
00185 public Request GetFormData(Stream input)
00186 {
00187     string request = "";
00188     MultipartFormDataParser parser = new MultipartFormDataParser(input);
00189     if (parser.Files.Count > 0)
00190     {
00191         using (StreamReader ms = new StreamReader(parser.Files[0].Data))
00192         {
00193             request = ms.ReadToEnd();
00194         }
00195     }
00196     else
00197     {
00198         throw new InvalidOperationException();
00199     }
00200     return Request.GetRequest(parser.Files[0].FileName, parser.Files[0].
ContentType, request);
00201 }
00202
00209 private Request GetFileData(Stream input, string type)
00210 {
00211     string request = "";
00212     using (StreamReader ms = new StreamReader(input))
00213     {
00214         request = ms.ReadToEnd();
00215     }
00216     string filename = "temp";
00217
00218     if (type == "text/xml")
00219         filename += ".xml";
00220     else if (type == "application/json")
00221         filename += ".json";
00222     else
00223         filename += ".html";
00224
00225     return Request.GetRequest(filename, type, request);
00226 }
00227
00233 private Response HandlePost(HttpListenerRequest req)
00234 {
00235     JObject eRes = new JObject();
00236
00237     if (SegmentNormalize(req.RawUrl).Equals("Process"))
00238     {
00239         if (req.HasEntityBody)
00240         {
00241             Request r = null;
00242             try
00243             {
00244                 r = GetData(req);
00245                 if (r == null)
00246                     return Response.GetInvalidRequestResponse();
00247             }
00248             catch
00249             {
00250                 return Response.GetInvalidRequestResponse();
00251             }
00252         }
00253     }
00254 }
00255
00256
00257

```

```

00258         try
00259         {
00260             Validation.CheckDocument(r.Data, r.
Type);
00261         }
00262         catch (Exception e)
00263         {
00264             eRes.Add("Error", e.Message);
00265             return Response.GetErrorResponse(eRes.ToString());
00266         }
00267
00268         string id = Guid.NewGuid().ToString();
00269         ProcessDocument pro = ProcessDocument.
GetProcess(r.Data, r.Type);
00270         Node n = pro.ProcessParallel();
00271         Output o = new Output(n); //new output object
00272         try
00273         {
00274             CacheManager.GetInstance().
saveFile(id, JsonConvert.SerializeObject(n));
00275             JObject response = new JObject();
00276
00277             n = null; //remove node as its completed;
00278
00279             response.Add("filename", id);
00280             response.Add("grid", o.CreateGrid());
00281             response.Add("view", o.CreateView());
00282
00283
00284
00285             byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
00286             return Response.GetResponse(200, "application/json", bytes);
00287         }
00288         catch (Exception e)
00289         {
00290             Logger.GetInstance().WriteError(e.Message);
00291             eRes.Add("Error", "Error Creating Response");
00292             return Response.GetErrorResponse(eRes.ToString());
00293         }
00294
00295     }
00296     eRes.Add("Error", "No File Recieved By Server");
00297     return Response.GetErrorResponse(eRes.ToString());
00298 }
00299 else if (req.RawUrl.Equals("/Output"))
00300 {
00301     return Response.GetInvalidRequestResponse();
00302 }
00303 else
00304     return Response.GetInvalidRequestResponse();
00305 }
00306
00312 private Response HandleGet(HttpListenerRequest req)
00313 {
00314     if (SegmentNormalize(req.Url.Segments[1]).Equals("Process"))
00315     {
00316         if (req.Url.Segments.Length == 4)
00317         {
00318
00319             Node cached;
00320             try
00321             {
00322                 cached = JsonConvert.DeserializeObject<Node>(
CacheManager.GetInstance().getFile(req.Url.Segments[2]));
00323             }
00324             catch (Exception e)
00325             {
00326                 Logger.GetInstance().WriteError(e.Message);
00327                 JObject eRes = new JObject();
00328                 eRes.Add("Error", "Error Creating Response");
00329                 return Response.GetErrorResponse(eRes.ToString());
00330             }
00331             Output o = new Output(cached);
00332             JObject response = new JObject();
00333             response.Add("view", o.CreateView(int.Parse(req.Url.Segments[3])));
00334             byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
00335             return Response.GetResponse(200, "application/json", bytes);
00336
00337         }
00338         else if (req.Url.Segments.Length == 5)
00339         {
00340             Node cached;
00341             try
00342             {
00343                 cached = JsonConvert.DeserializeObject<Node>(
CacheManager.GetInstance().getFile(req.Url.Segments[2]));

```

```

00344         }
00345         catch (Exception e)
00346         {
00347             Logger.GetInstance().WriteError(e.Message);
00348             JObject eRes = new JObject();
00349             eRes.Add("Error", "Error Creating Response");
00350             return Response.GetErrorResponse(eRes.ToString());
00351         }
00352         Output o = new Output(cached);
00353         JObject response = new JObject();
00354         response.Add("view", o.CreateView(int.Parse(SegmentNormalize(req.Url.Segments
[3])), 4, int.Parse(SegmentNormalize(req.Url.Segments[4]))));
00355         byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
00356         return Response.GetResponse(200, "application/json", bytes);
00357     }
00358     else
00359     {
00360         return Response.GetInvalidRequestResponse();
00361     }
00362     //If it got here its an invalid response.
00363     return Response.GetInvalidRequestResponse();
00364 }
00371 private void SaveFile(string id, Node nodes)
00372 {
00373     if(nodes == null || string.IsNullOrEmpty(id))
00374     {
00375         throw new ArgumentException();
00376     }
00377     try
00378     {
00379         CacheManager.GetInstance().saveFile(id, JsonConvert.
SerializeObject(nodes));
00380     }
00381     catch (Exception e)
00382     {
00383     }
00384     Logger.GetInstance().WriteError(e.Message);
00385 }
00386 }
00387
00394 private void Post(Response res,HttpListenerResponse stream)
00395 {
00396     if (res == null || stream == null)
00397     {
00398         throw new ArgumentException("Response or Client Stream cannot be NULL");
00399     }
00400     HandleOptions(stream);
00401     stream.ProtocolVersion = new Version(1, 1);
00402     stream.StatusCode = res.status;
00403     stream.ContentType = res.mime;
00404     stream.ContentLength64 = res.data.Length;
00405     stream.OutputStream.Write(res.data, 0, res.data.Length);
00406     stream.Close();
00407 }
00413 private Request GetData(HttpListenerRequest req)
00414 {
00415     Request r = null;
00416     if (req.ContentType.Contains("application/x-www-form-urlencoded"))
00417     {
00418         r = GetFormData(req.InputStream);
00419     }
00420     else if (req.ContentType.Contains("application/json") || req.ContentType.Contains("
application/oclet-stream"))
00421     {
00422         r = GetFileData(req.InputStream, "application/json");
00423     }
00424     else if (req.ContentType.Contains("application/xml") || req.ContentType.Contains("text/xml"))
00425     {
00426         r = GetFileData(req.InputStream, "text/xml");
00427     }
00428     return r;
00429 }
00430 }
00431
00437 private string SegmentNormalize(string input)
00438 {
00439     return input.Replace("/", "");
00440 }
00441 }
00442 }

```

## 6.21 Program.cs File Reference

## Classes

- class [WebServer.Program](#)  
*The Initialiser*

## Namespaces

- namespace [WebServer](#)

## 6.22 Program.cs

```

00001 using System;
00002 using System.Threading;
00003 using System.IO;
00004 using OSXJV.Classes;
00005 using OSXJV.Server;
00006
00007 namespace WebServer
00008 {
00012     class Program
00013     {
00018         static void Main(string[] args)
00019         {
00020
00021             if (args.Length == 0)
00022             {
00023                 Console.WriteLine("Using Default Cache Directory Path and Logger Directory Path");
00024                 string dir = Directory.GetCurrentDirectory();
00025                 Array.Resize(ref args, 2);
00026                 args[0] = dir + "/Cache/";
00027                 args[1] = dir + "/Logger/";
00028                 if (!Directory.Exists(args[0]))
00029                     Directory.CreateDirectory(args[0]);
00030                 if (!Directory.Exists(args[1]))
00031                     Directory.CreateDirectory(args[1]);
00032             }
00033
00034             if (args[0] == args[1])
00035             {
00036                 Console.WriteLine("Cache location and Log location is the same. Please enter two different
locations");
00037             }
00038             else
00039             {
00040                 bool pass = false;
00041                 try
00042                 {
00043                     pass = CacheManager.Setup(args[0]);
00044                     pass = Logger.Setup(args[1]);
00045                 }
00046                 catch (Exception e)
00047                 {
00048                     Console.WriteLine("Error Setting Cache and Logger Directory: {0}", e.Message);
00049                 }
00050                 if (pass)
00051                 {
00052                     OSXJVServer s = new OSXJVServer();
00053                     s.Start();
00054                 }
00055
00056                 //Check Cache every hour to remove old files
00057                 while (true)
00058                 {
00059                     Thread.Sleep(3600000);
00060
00061                     string[] files = Directory.GetFiles(args[0]);
00062
00063                     foreach (string file in files)
00064                     {
00065                         if (File.GetLastAccessTime(file) < DateTime.Now.AddHours(-6.0))
00066                             File.Delete(file);
00067                     }
00068                 }
00069             }
00070         }
00071     }
00072 }

```

## 6.23 Properties/AssemblyInfo.cs File Reference

### 6.24 AssemblyInfo.cs

```
00001 using System.Reflection;
00002 using System.Runtime.CompilerServices;
00003 using System.Runtime.InteropServices;
00004
00005 // General Information about an assembly is controlled through the following
00006 // set of attributes. Change these attribute values to modify the information
00007 // associated with an assembly.
00008 [assembly: AssemblyTitle("WebServiceCSharp")]
00009 [assembly: AssemblyDescription("")]
00010 [assembly: AssemblyConfiguration("")]
00011 [assembly: AssemblyCompany("")]
00012 [assembly: AssemblyProduct("WebServiceCSharp")]
00013 [assembly: AssemblyCopyright("Copyright © 2016")]
00014 [assembly: AssemblyTrademark("")]
00015 [assembly: AssemblyCulture("")]
00016
00017 // Setting ComVisible to false makes the types in this assembly not visible
00018 // to COM components. If you need to access a type in this assembly from
00019 // COM, set the ComVisible attribute to true on that type.
00020 [assembly: ComVisible(false)]
00021
00022 // The following GUID is for the ID of the typelib if this project is exposed to COM
00023 [assembly: Guid("a57034df-dc0f-44ce-bb8a-cddafe37db17")]
00024
00025 // Version information for an assembly consists of the following four values:
00026 //
00027 //      Major Version
00028 //      Minor Version
00029 //      Build Number
00030 //      Revision
00031 //
00032 // You can specify all the values or you can default the Build and Revision Numbers
00033 // by using the '*' as shown below:
00034 // [assembly: AssemblyVersion("1.0.*")]
00035 [assembly: AssemblyVersion("1.0.0.0")]
00036 [assembly: AssemblyFileVersion("1.0.0.0")]
```





# Index

- Attributes
  - OSXJV::Classes::Node, [24](#)
- attributes
  - OSXJV::Classes::Node, [23](#)
- cNodes
  - OSXJV::Classes::Output, [58](#)
- CacheManager
  - OSXJV::Classes::CacheManager, [12](#)
- CheckChildren
  - OSXJV::Classes::Output, [45](#), [46](#)
- CheckDocument
  - OSXJV::Classes::Validation, [87](#)
- CheckNodeNumber
  - OSXJV::Classes::Output, [47](#)
- Children
  - OSXJV::Classes::Node, [24](#)
- children
  - OSXJV::Classes::Node, [23](#)
- Close
  - OSXJV::Classes::CacheManager, [12](#)
  - OSXJV::Classes::Logger, [18](#)
- Comments
  - OSXJV::Classes::Node, [24](#)
- comments
  - OSXJV::Classes::Node, [23](#)
- count
  - OSXJV::Classes::ProcessDocument, [72](#)
- CreateExtraNode
  - OSXJV::Classes::Output, [48](#)
- CreateGrid
  - OSXJV::Classes::Output, [49](#)
- CreateNodeChildViewsParallel
  - OSXJV::Classes::Output, [50](#)
- CreateNodeView
  - OSXJV::Classes::Output, [51](#), [52](#)
- CreatePreviousNode
  - OSXJV::Classes::Output, [53](#)
- CreateView
  - OSXJV::Classes::Output, [54](#)
- CreateViewSingle
  - OSXJV::Classes::Output, [56](#)
- Data
  - OSXJV::Classes::Request, [79](#)
- data
  - OSXJV::Classes::Request, [78](#)
  - OSXJV::Classes::Response, [85](#)
- document
  - OSXJV::Classes::ProcessDocument, [72](#)
- Filename
  - OSXJV::Classes::Request, [79](#)
- filename
  - OSXJV::Classes::Request, [78](#)
- GetData
  - OSXJV::Server::OSXJVServer, [28](#)
- GetErrorResponse
  - OSXJV::Classes::Response, [82](#)
- getFile
  - OSXJV::Classes::CacheManager, [12](#)
- GetFileData
  - OSXJV::Server::OSXJVServer, [29](#)
- GetFormData
  - OSXJV::Server::OSXJVServer, [30](#)
- GetInstance
  - OSXJV::Classes::CacheManager, [13](#)
  - OSXJV::Classes::Logger, [18](#)
  - OSXJV::Classes::Validation, [89](#)
- GetInvalidRequestResponse
  - OSXJV::Classes::Response, [82](#)
- GetParent
  - OSXJV::Classes::Output, [57](#)
- GetProcess
  - OSXJV::Classes::ProcessDocument, [63](#)
- GetRequest
  - OSXJV::Classes::Request, [77](#)
- GetResponse
  - OSXJV::Classes::Response, [83](#)
- GetResponseJSON
  - OSXJV::Classes::Response, [84](#)
- GetResponseXML
  - OSXJV::Classes::Response, [85](#)
- GotParent
  - OSXJV::Classes::Output, [59](#)
- GridGetChidren
  - OSXJV::Classes::Output, [58](#)
- HandleClient
  - OSXJV::Server::OSXJVServer, [30](#)
- HandleGet
  - OSXJV::Server::OSXJVServer, [31](#)
- HandleOptions
  - OSXJV::Server::OSXJVServer, [33](#)
- HandlePost
  - OSXJV::Server::OSXJVServer, [33](#)
- inst
  - OSXJV::Classes::CacheManager, [16](#)
  - OSXJV::Classes::Logger, [20](#)

- OSXJV::Classes::Validation, [89](#)
- left
  - OSXJV::Classes::Output, [59](#)
- listener
  - OSXJV::Server::OSXJVServer, [41](#)
- ListenerCallback
  - OSXJV::Server::OSXJVServer, [36](#)
- location
  - OSXJV::Classes::Logger, [20](#)
- Logger
  - OSXJV::Classes::Logger, [17](#)
- Main
  - WebServer::Program, [74](#)
- mime
  - OSXJV::Classes::Response, [86](#)
- Name
  - OSXJV::Classes::Attribute, [10](#)
  - OSXJV::Classes::Node, [25](#)
- name
  - OSXJV::Classes::Attribute, [10](#)
  - OSXJV::Classes::Node, [23](#)
- Node
  - OSXJV::Classes::Node, [22](#)
- node
  - OSXJV::Classes::ProcessDocument, [72](#)
- nodes
  - OSXJV::Classes::Output, [59](#)
- Number
  - OSXJV::Classes::Node, [25](#)
- number
  - OSXJV::Classes::Node, [23](#)
- OSXJV.Classes, [7](#)
- OSXJV.Classes.Attribute, [9](#)
- OSXJV.Classes.CacheManager, [11](#)
- OSXJV.Classes.Logger, [16](#)
- OSXJV.Classes.Node, [21](#)
- OSXJV.Classes.Output, [42](#)
- OSXJV.Classes.ProcessDocument, [60](#)
- OSXJV.Classes.Request, [75](#)
- OSXJV.Classes.Response, [80](#)
- OSXJV.Classes.Validation, [86](#)
- OSXJV.Server, [7](#)
- OSXJV.Server.OSXJVServer, [26](#)
- OSXJV::Classes::Attribute
  - Name, [10](#)
  - name, [10](#)
  - Value, [10](#)
  - value, [10](#)
- OSXJV::Classes::CacheManager
  - CacheManager, [12](#)
  - Close, [12](#)
  - getFile, [12](#)
  - GetInstance, [13](#)
  - inst, [16](#)
  - path, [16](#)
  - saveFile, [14](#)
  - Setup, [15](#)
- OSXJV::Classes::Logger
  - Close, [18](#)
  - GetInstance, [18](#)
  - inst, [20](#)
  - location, [20](#)
  - Logger, [17](#)
  - Setup, [18](#)
  - WriteError, [19](#)
- OSXJV::Classes::Node
  - Attributes, [24](#)
  - attributes, [23](#)
  - Children, [24](#)
  - children, [23](#)
  - Comments, [24](#)
  - comments, [23](#)
  - Name, [25](#)
  - name, [23](#)
  - Node, [22](#)
  - Number, [25](#)
  - number, [23](#)
  - Value, [25](#)
  - value, [23](#)
  - Visited, [25](#)
  - visited, [24](#)
- OSXJV::Classes::Output
  - cNodes, [58](#)
  - CheckChildren, [45](#), [46](#)
  - CheckNodeNumber, [47](#)
  - CreateExtraNode, [48](#)
  - CreateGrid, [49](#)
  - CreateNodeChildViewsParallel, [50](#)
  - CreateNodeView, [51](#), [52](#)
  - CreatePreviousNode, [53](#)
  - CreateView, [54](#)
  - CreateViewSingle, [56](#)
  - GetParent, [57](#)
  - GotParent, [59](#)
  - GridGetChildren, [58](#)
  - left, [59](#)
  - nodes, [59](#)
  - Output, [44](#)
  - Parent, [59](#)
  - top, [59](#)
- OSXJV::Classes::ProcessDocument
  - count, [72](#)
  - document, [72](#)
  - GetProcess, [63](#)
  - node, [72](#)
  - Prepare, [64](#)
  - Process, [64](#)
  - ProcessComment, [65](#)
  - ProcessDocument, [63](#)
  - ProcessDocumentParallelInit, [66](#)
  - ProcessElement, [66](#), [68](#)
  - ProcessParallel, [69](#)
  - ProcessRoot, [70](#)

- ProcessText, [71](#)
- ProcessedElements, [72](#)
- th, [72](#)
- ThreadList, [73](#)
- type, [73](#)
- OSXJV::Classes::Request
  - Data, [79](#)
  - data, [78](#)
  - Filename, [79](#)
  - filename, [78](#)
  - GetRequest, [77](#)
  - Request, [77](#)
  - Type, [79](#)
  - type, [79](#)
- OSXJV::Classes::Response
  - data, [85](#)
  - GetErrorResponse, [82](#)
  - GetInvalidRequestResponse, [82](#)
  - GetResponse, [83](#)
  - GetResponseJSON, [84](#)
  - GetResponseXML, [85](#)
  - mime, [86](#)
  - Response, [81](#)
  - status, [86](#)
- OSXJV::Classes::Validation
  - CheckDocument, [87](#)
  - GetInstance, [89](#)
  - inst, [89](#)
  - Validation, [87](#)
- OSXJV::Server::OSXJVServer
  - GetData, [28](#)
  - GetFileData, [29](#)
  - GetFormData, [30](#)
  - HandleClient, [30](#)
  - HandleGet, [31](#)
  - HandleOptions, [33](#)
  - HandlePost, [33](#)
  - listener, [41](#)
  - ListenerCallback, [36](#)
  - OSXJVServer, [28](#)
  - port, [41](#)
  - Post, [37](#)
  - Run, [38](#)
  - running, [41](#)
  - SaveFile, [38](#)
  - SegmentNormalize, [39](#)
  - serverThread, [41](#)
  - Start, [40](#)
  - Stop, [40](#)
- OSXJVClasses/Attribute.cs, [91](#)
- OSXJVClasses/CacheManager.cs, [92](#)
- OSXJVClasses/Logger.cs, [93](#)
- OSXJVClasses/Node.cs, [94](#)
- OSXJVClasses/Output.cs, [96](#)
- OSXJVClasses/ProcessDocument.cs, [103](#)
- OSXJVClasses/Request.cs, [107](#)
- OSXJVClasses/Response.cs, [108](#), [109](#)
- OSXJVClasses/Validation.cs, [110](#)
- OSXJVServer
  - OSXJV::Server::OSXJVServer, [28](#)
  - OSXJVServer.cs, [111](#)
  - OSXJV, [7](#)
  - Output
    - OSXJV::Classes::Output, [44](#)
  - Parent
    - OSXJV::Classes::Output, [59](#)
  - path
    - OSXJV::Classes::CacheManager, [16](#)
  - port
    - OSXJV::Server::OSXJVServer, [41](#)
  - Post
    - OSXJV::Server::OSXJVServer, [37](#)
  - Prepare
    - OSXJV::Classes::ProcessDocument, [64](#)
  - Process
    - OSXJV::Classes::ProcessDocument, [64](#)
  - ProcessComment
    - OSXJV::Classes::ProcessDocument, [65](#)
  - ProcessDocument
    - OSXJV::Classes::ProcessDocument, [63](#)
  - ProcessDocumentParallelInit
    - OSXJV::Classes::ProcessDocument, [66](#)
  - ProcessElement
    - OSXJV::Classes::ProcessDocument, [66](#), [68](#)
  - ProcessParallel
    - OSXJV::Classes::ProcessDocument, [69](#)
  - ProcessRoot
    - OSXJV::Classes::ProcessDocument, [70](#)
  - ProcessText
    - OSXJV::Classes::ProcessDocument, [71](#)
  - ProcessedElements
    - OSXJV::Classes::ProcessDocument, [72](#)
  - Program.cs, [115](#)
  - Properties/AssemblyInfo.cs, [117](#)
  - Request
    - OSXJV::Classes::Request, [77](#)
  - Response
    - OSXJV::Classes::Response, [81](#)
  - Run
    - OSXJV::Server::OSXJVServer, [38](#)
  - running
    - OSXJV::Server::OSXJVServer, [41](#)
  - SaveFile
    - OSXJV::Server::OSXJVServer, [38](#)
  - saveFile
    - OSXJV::Classes::CacheManager, [14](#)
  - SegmentNormalize
    - OSXJV::Server::OSXJVServer, [39](#)
  - serverThread
    - OSXJV::Server::OSXJVServer, [41](#)
  - Setup
    - OSXJV::Classes::CacheManager, [15](#)
    - OSXJV::Classes::Logger, [18](#)
  - Start

- OSXJV::Server::OSXJVServer, [40](#)
- status
  - OSXJV::Classes::Response, [86](#)
- Stop
  - OSXJV::Server::OSXJVServer, [40](#)
- th
  - OSXJV::Classes::ProcessDocument, [72](#)
- ThreadList
  - OSXJV::Classes::ProcessDocument, [73](#)
- top
  - OSXJV::Classes::Output, [59](#)
- Type
  - OSXJV::Classes::Request, [79](#)
- type
  - OSXJV::Classes::ProcessDocument, [73](#)
  - OSXJV::Classes::Request, [79](#)
- Validation
  - OSXJV::Classes::Validation, [87](#)
- Value
  - OSXJV::Classes::Attribute, [10](#)
  - OSXJV::Classes::Node, [25](#)
- value
  - OSXJV::Classes::Attribute, [10](#)
  - OSXJV::Classes::Node, [23](#)
- Visited
  - OSXJV::Classes::Node, [25](#)
- visited
  - OSXJV::Classes::Node, [24](#)
- WebServer, [8](#)
- WebServer.Program, [73](#)
- WebServer::Program
  - Main, [74](#)
- WriteError
  - OSXJV::Classes::Logger, [19](#)