Open Source XML & JSON Visualisation Software

1.0

Generated by Doxygen 1.8.13

Tue Apr 18 2017 16:52:59

Contents

Chapter 1

Namespace Index

1.1 Packages

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

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 Processed Documents
- 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

  Perform validation on document

4.3 OSXJV.Server Namespace Reference

Classes

- class OSXJVServer

  HTTPServer that process the incoming requests.

4.4 WebServer Namespace Reference

Classes

- class Program

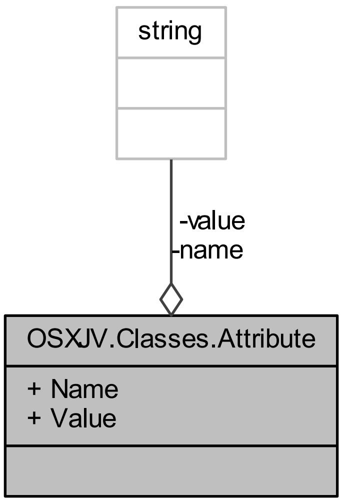
  The Initialiser

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().

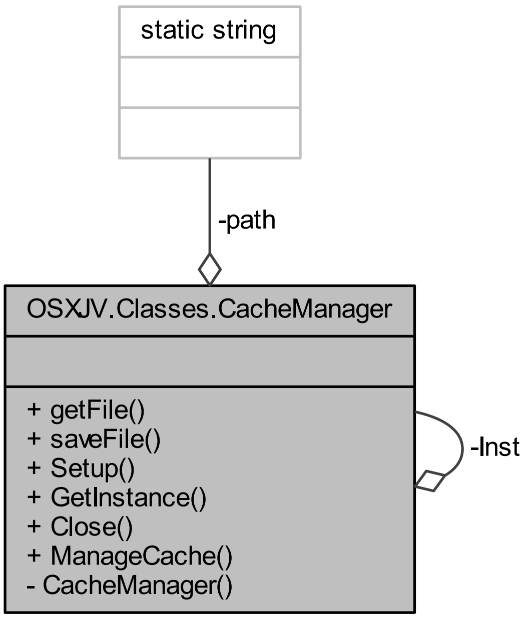
The documentation for this class was generated from the following file:

- WebServiceCSharp/OSXJVClasses/Attribute.cs

5.2 OSXJV.Classes.CacheManager Class Reference

Manages Saving an Retrieving Processed Documents

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)

  Initialises CacheManager with input path
- static CacheManager GetInstance ()

  Get the single instance of the class
- static void Close ()

  Destroys the created instance
- static void ManageCache ()

  Manages files in the cache deleting old ones than 6 hours when called

Private Member Functions

- CacheManager (string cachePath)

  Constructor setting store location;

Static Private Attributes

- static CacheManager Inst

  Instance of CacheManager
- static string path = null

  Cache folder location

5.2.1 Detailed Description

Manages Saving an Retrieving Processed Documents

Definition at line 9 of file CacheManager.cs.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 CacheManager()

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

Constructor setting store location;

Parameters

path	

Definition at line 25 of file CacheManager.cs.

```
00026        {
00027            path = cachePath;
00028        }
```

5.2.3 Member Function Documentation

5.2.3.1 Close()

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

Destroys the created instance

Exceptions

Exception	If Instance isn't previously created

Definition at line 114 of file CacheManager.cs.

```
        {
            if (Inst == null)
                throw new Exception("CacheManager Already Closed");
            else
            {
                path = null; //Clear static path
                Inst = null; //clear static instance
            }
        }
```

5.2.3.2 getFile()

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

Retrieve the file from caching

Parameters

ID	Unique ID of the file

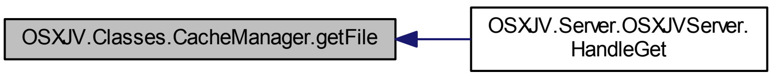
Returns

Definition at line 63 of file CacheManager.cs.

Referenced by OSXJV.Server.OSXJVServer.HandleGet().

```
        {
            if (string.IsNullOrEmpty(ID))
                throw new ArgumentException("ID cannot be null or empty");

            string filePath = path + "/" + ID.Replace("/","") + ".json";
            string output = "";

            using (StreamReader sr = new StreamReader(filePath))
            {
                output = sr.ReadToEnd();
            }

            if (!string.IsNullOrEmpty(output))
                return output;
            else
                throw new Exception("Error Reading From File");
        }
```

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

Exceptions

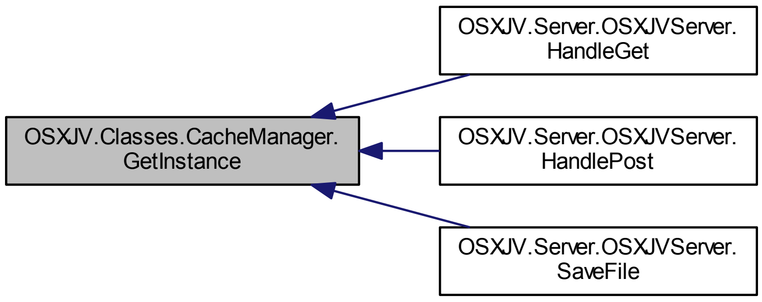
Exception	If the CacheManger has not been setup

Definition at line 50 of file CacheManager.cs.

Referenced by OSXJV.Server.OSXJVServer.HandleGet(), OSXJV.Server.OSXJVServer.HandlePost(), and OSX↩
JV.Server.OSXJVServer.SaveFile().

```
        {
            if (Inst != null)
                return Inst;
            else
                throw new Exception("CacheManger has not been setup");
        }
```

Here is the caller graph for this function:


OSXJV.Classes.CacheManager.
GetInstance
OSXJV.Server.OSXJVServer.
HandleGet
OSXJV.Server.OSXJVServer.
HandlePost
OSXJV.Server.OSXJVServer.
SaveFile

5.2.3.4 ManageCache()

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

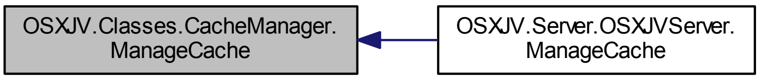
Manages files in the cache deleting old ones than 6 hours when called

Definition at line 128 of file CacheManager.cs.

Referenced by OSXJV.Server.OSXJVServer.ManageCache().

```
        {
            if (path != null)
            {
                string[] files = Directory.GetFiles(path);
                foreach (string file in files)
                {
                    if (File.GetLastAccessTime(file) < DateTime.Now.AddHours(-6.0))
                        File.Delete(file);
                }
            }
            else
                throw new Exception("CacheManger not setup");
        }
```

Here is the caller graph for this function:



5.2.3.5 saveFile()

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

Save the file to the local system for caching

Parameters

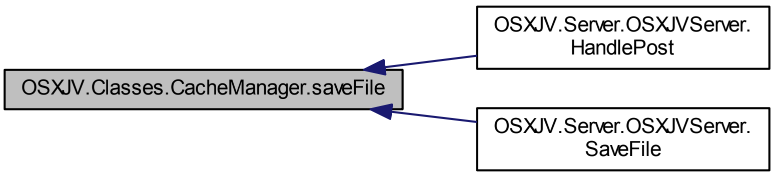
ID	Unique ID of the file
nodes	The document to be saved

Definition at line 87 of file CacheManager.cs.

Referenced by OSXJV.Server.OSXJVServer.HandlePost(), and OSXJV.Server.OSXJVServer.SaveFile().

```
        {
            if (string.IsNullOrEmpty(ID))
                throw new ArgumentException("ID cannot be null or empty");

            if (string.IsNullOrEmpty(nodes))
                throw new ArgumentException("Document cannot be null or empty");

            string filePath = path + "/" + ID + ".json";
            try
            {
                using (StreamWriter sw = new StreamWriter(filePath))
                {
                    sw.WriteLine(nodes);
                }
            }
            catch
            {
                throw new Exception("Failed to save file");
            }

            return true;
        }
```

Here is the caller graph for this function:


OSXJV.Classes.CacheManager.saveFile
OSXJV.Server.OSXJVServer.HandlePost
OSXJV.Server.OSXJVServer.SaveFile

5.2.3.6 Setup()

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

Initialises CacheManager with input path

Parameters

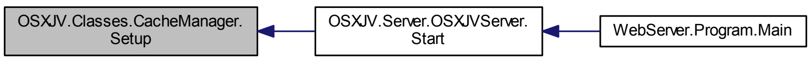
path	

Definition at line 34 of file CacheManager.cs.

Referenced by OSXJV.Server.OSXJVServer.Start().

```
        {
            if (string.IsNullOrEmpty(path))
                throw new ArgumentException("Path cannot be empty");

            if (!Directory.Exists(string.Format(@"{0}",path)))
                throw new Exception("Path is not a valid cache directory");

            return (Inst = new CacheManager(path)) != null ? true : false;
        }
```

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]`

Instance of CacheManager

Definition at line 14 of file CacheManager.cs.

5.2.4.2 path

`string OSXJV.Classes.CacheManager.path = null [static], [private]`

Cache folder location

Definition at line 19 of file CacheManager.cs.

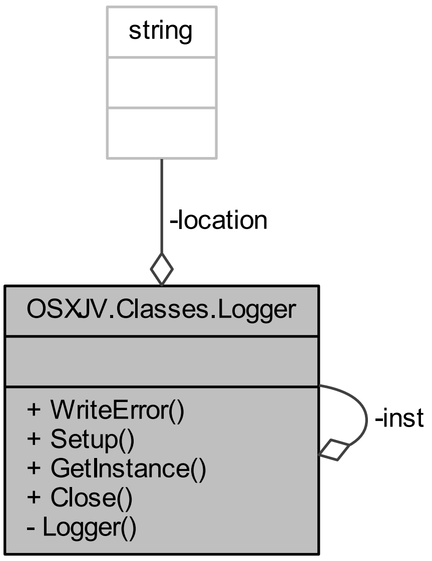
The documentation for this class was generated from the following file:

- WebServiceCSharp/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.

```
        {
            this.location = location;
        }
```

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.

```
        {
            if (inst == null)
                throw new Exception("Logger Already Closed");
            else
                inst = null;
        }
```

5.3.3.2 GetInstance()

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

Gets the single instance of Logger

Returns

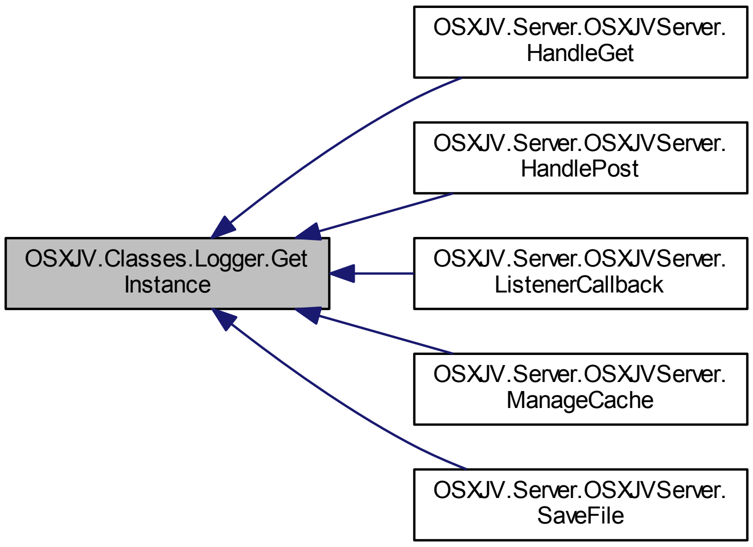
Instance of Logger

Definition at line 41 of file Logger.cs.

Referenced by OSXJV.Server.OSXJVServer.HandleGet(), OSXJV.Server.OSXJVServer.HandlePost(), OSXJV.↩ Server.OSXJVServer.ListenerCallback(), OSXJV.Server.OSXJVServer.ManageCache(), and OSXJV.Server.OS↩ XJVServer.SaveFile().

```
        {
            if (inst != null)
                return inst;
            else
                throw new Exception("Logger has not been setup");
        }
```

Here is the caller graph for this function:


OSXJV.Classes.Logger.Get Instance
OSXJV.Server.OSXJVServer. HandleGet
OSXJV.Server.OSXJVServer. HandlePost
OSXJV.Server.OSXJVServer. ListenerCallback
OSXJV.Server.OSXJVServer. ManageCache
OSXJV.Server.OSXJVServer. SaveFile

5.3.3.3 Setup()

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

Parameters

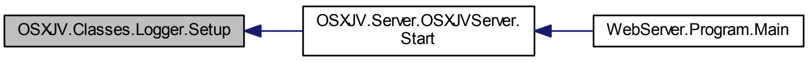
location	

Definition at line 26 of file Logger.cs.

Referenced by OSXJV.Server.OSXJVServer.Start().

```
        {
            if (string.IsNullOrEmpty(location))
                throw new ArgumentException("Location cannot be empty");

            if (!Directory.Exists(string.Format(@"{0}", location)))
                throw new Exception("Location is not a valid logger directory");

            return (inst = new Logger(location)) != null ? true:false;
        }
```

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

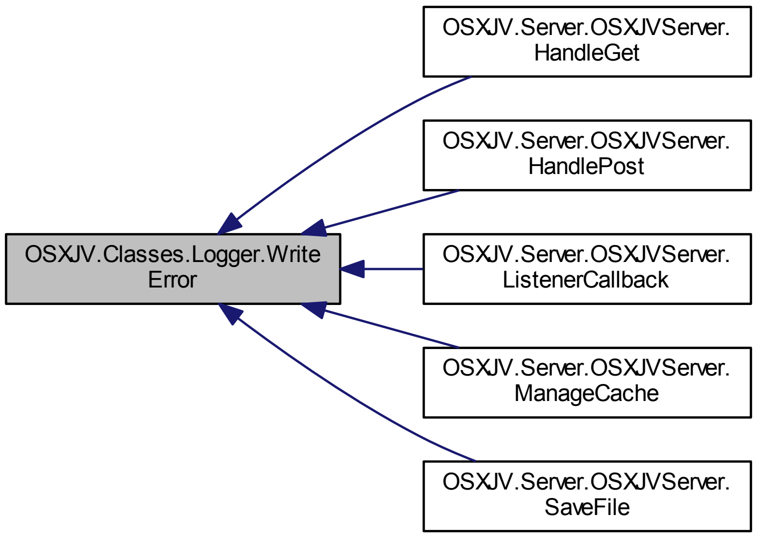
error	The error message

Definition at line 53 of file Logger.cs.

Referenced by OSXJV.Server.OSXJVServer.HandleGet(), OSXJV.Server.OSXJVServer.HandlePost(), OSXJV.Server.OSXJVServer.ListenerCallback(), OSXJV.Server.OSXJVServer.ManageCache(), and OSXJV.Server.OSXJVServer.SaveFile().

```
        {
            try
            {
                if (!string.IsNullOrEmpty(error))
                {
                    string file = string.Format(@"{0}/Error-{1}.txt", location, DateTime.Now.
   ToString("dd-MM-yy hh-mm-ss"));
                    StreamWriter sw = new StreamWriter(file);
                    sw.WriteLine(error);
                    sw.WriteLine();
                    sw.Close();
                }
            }
            catch (IOException e)
            {
                throw e;
            }
        }
```

Here is the caller graph for this function:


OSXJV.Classes.Logger.Write Error
OSXJV.Server.OSXJVServer. HandleGet
OSXJV.Server.OSXJVServer. HandlePost
OSXJV.Server.OSXJVServer. ListenerCallback
OSXJV.Server.OSXJVServer. ManageCache
OSXJV.Server.OSXJVServer. SaveFile

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.

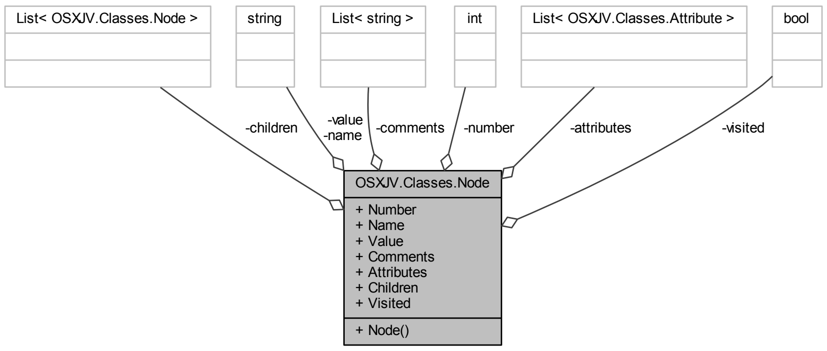
The documentation for this class was generated from the following file:

- WebServiceCSharp/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.

```
        {
            Attributes = new List<Attribute>();
            Children = new List<Node>();
            Comments = new List<string>();
            number = 0;
            visited = false;
        }
```

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.CreateNode↩ View(), 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(), OSXJ↩V.Classes.Output.CreateGrid(), OSXJV.Classes.Output.CreateNodeView(), OSXJV.Classes.Output.CreateView(), OSXJV.Classes.Output.CreateViewSingle(), OSXJV.Classes.Output.GetParent(), OSXJV.Classes.Output.Grid↩GetChidren(), 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().

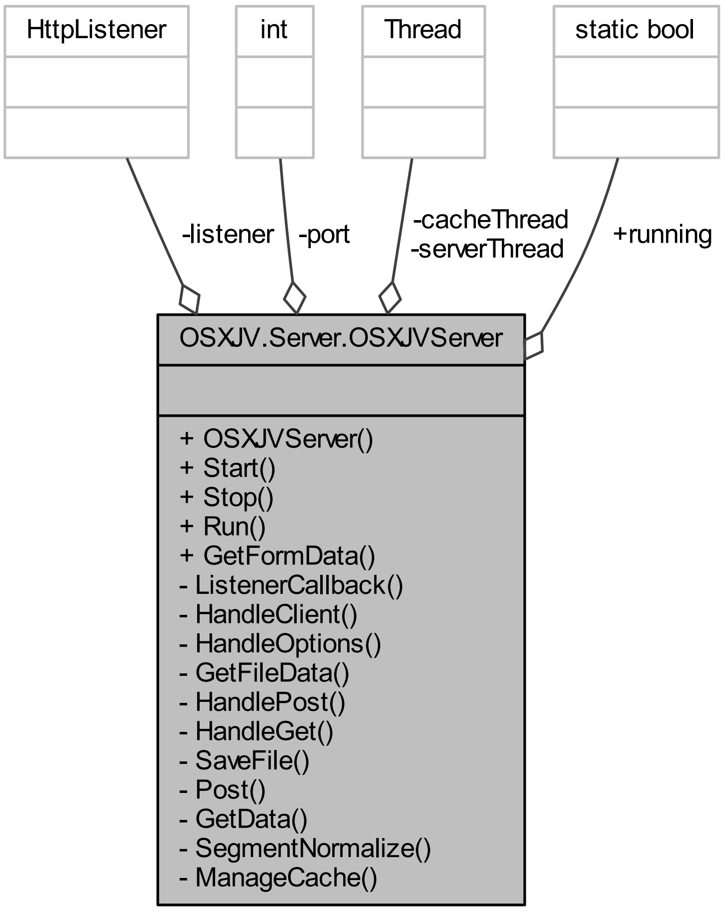
The documentation for this class was generated from the following file:

- WebServiceCSharp/OSXJVClasses/Node.cs

5.5 OSXJV.Server.OSXJVServer Class Reference

HTTPServer that process the incoming requests.

Collaboration diagram for OSXJV.Server.OSXJVServer:



Public Member Functions

- OSXJVServer ()

    The Server Handler
- bool Start (string cachePath, string loggerPath)

    Starts server in new thread

Parameters

cachePath	*Cache Folder Location*
loggerPath	*Logger Folder Location*

- 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 Asyncronously
- 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 recievied 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.
- void ManageCache ()

  Manages cache

Private Attributes

- int port = 8082
- HttpListener listener

  HttpListener
- Thread serverThread = null

  Server Thread
- Thread cacheThread = null

  Cache Controller

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 43 of file OSXJVServer.cs.

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

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

req	The request from the client

Returns

A Request Object

Definition at line 414 of file OSXJVServer.cs.

```
        {
            Request r = null;

            if (req.ContentType.Contains("application/x-www-form-urlencoded"))
            {
                r = GetFormData(req.InputStream);
            }
            else if (req.ContentType.Contains("application/json") || req.ContentType.Contains("
      application/oclet-stream"))
            {
                r = GetFileData(req.InputStream, "application/json");
            }
            else if (req.ContentType.Contains("application/xml") || req.ContentType.Contains("text/xml"))
            {
                r = GetFileData(req.InputStream, "text/xml");
            }
            return r;
        }
```

5.5.3.2 GetFileData()

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

Get Data if the data is retrieved

Parameters

input	Client Stream Input
type	The MIME type

Returns

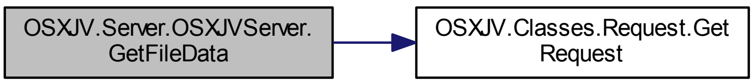
A Response object to send to the user

Definition at line 210 of file OSXJVServer.cs.

References OSXJV.Classes.Request.GetRequest().

```
        {
            string request = "";
            using (StreamReader ms = new StreamReader(input))
            {
                request = ms.ReadToEnd();
            }
            string filename = "temp";

            if (type == "text/xml")
                filename += ".xml";
            else if(type == "application/json")
                filename += ".json";
            else
                filename += ".html";

            return Request.GetRequest(filename,type, request);
        }
```

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

input	Requests input stream

Returns

New Request Object

Exceptions

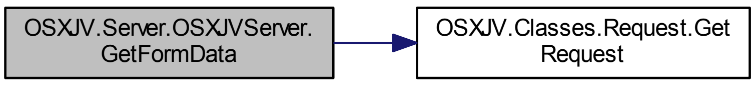
System.InvalidOperationException	Thrown when no files are included with the request

Definition at line 186 of file OSXJVServer.cs.

References OSXJV.Classes.Request.GetRequest().

```
        {
            string request = "";
            MultipartFormDataParser parser = new MultipartFormDataParser(input);
            if (parser.Files.Count > 0)
            {
                using (StreamReader ms = new StreamReader(parser.Files[0].Data))
                {
                    request = ms.ReadToEnd();
                }
            }
            else
            {
                throw new InvalidOperationException();
            }
            return Request.GetRequest(parser.Files[0].FileName, parser.Files[0].
     ContentType, request);
        }
```

Here is the call graph for this function:


OSXJV.Server.OSXJVServer.GetFormData
OSXJV.Classes.Request.GetRequest

5.5.3.4 HandleClient()

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

Handles the client

Parameters

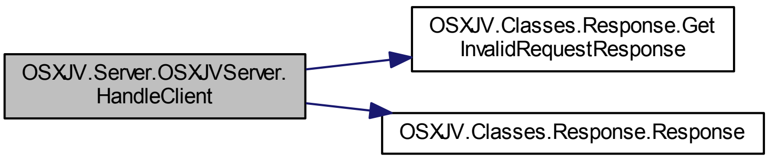
c	The Request

Definition at line 147 of file OSXJVServer.cs.

References OSXJV.Classes.Response.GetInvalidRequestResponse(), and OSXJV.Classes.Response.Response().

```
        {
            switch(c.Request.HttpMethod)
            {
                case "POST":
                    Post(HandlePost(c.Request),c.Response);
                    break;
                case "GET":
                    Post(HandleGet(c.Request), c.Response);
                    break;
                case "OPTIONS":
                    HandleOptions(c.Response);
                    c.Response.Close();
                    break;
                default:
                    Post(Response.GetInvalidRequestResponse(), c.
     Response);
                    break;
            }
        }
```

Here is the call graph for this function:


OSXJV.Server.OSXJVServer.
HandleClient
OSXJV.Classes.Response.Get
InvalidRequestResponse
OSXJV.Classes.Response.Response

5.5.3.5 HandleGet()

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

Handles a GET request.

Parameters

req	The request to be processed.

Returns

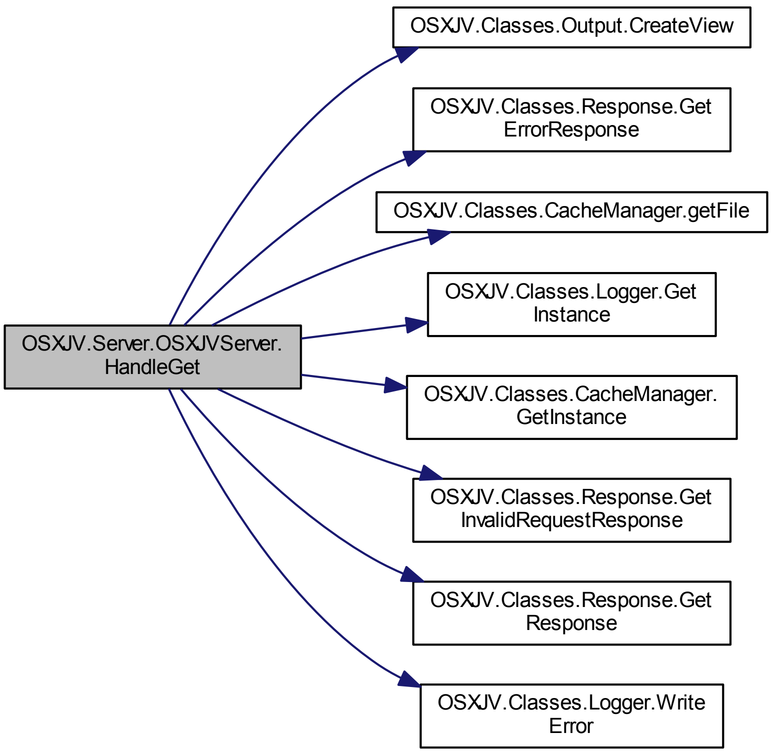
A Response object to send to the user

Definition at line 313 of file OSXJVServer.cs.

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

```
        {
            if (SegmentNormalize(req.Url.Segments[1]).Equals("Process"))
            {
                if (req.Url.Segments.Length == 4)
                {

                    Node cached;
                    try
                    {
                        cached = JsonConvert.DeserializeObject<Node>(
    CacheManager.GetInstance().getFile(req.Url.Segments[2]));
                    }
                    catch (Exception e)
                    {
                        Logger.GetInstance().WriteError(e.Message);
                        JObject eRes = new JObject();
                        eRes.Add("Error", "Error Creating Response");
                        return Response.GetErrorResponse(eRes.ToString());
                    }
                    Output o = new Output(cached);
                    JObject response = new JObject();
                    response.Add("view", o.CreateView(int.Parse(req.Url.Segments[3])));
                    byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
                    return Response.GetResponse(200, "application/json", bytes);

                }
                else if (req.Url.Segments.Length == 5)
                {

                    Node cached;
                    try
                    {
                        cached = JsonConvert.DeserializeObject<Node>(
    CacheManager.GetInstance().getFile(req.Url.Segments[2]));
                    }
                    catch (Exception e)
                    {
                        Logger.GetInstance().WriteError(e.Message);
                        JObject eRes = new JObject();
                        eRes.Add("Error", "Error Creating Response");
                        return Response.GetErrorResponse(eRes.ToString());
                    }
                    Output o = new Output(cached);
                    JObject response = new JObject();
                    response.Add("view", o.CreateView(int.Parse(
    SegmentNormalize(req.Url.Segments[3])), 4, int.Parse(
    SegmentNormalize(req.Url.Segments[4]))));
                    byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
                    return Response.GetResponse(200, "application/json", bytes);

                }
                else
                    return Response.GetInvalidRequestResponse();
            }
            //If it got here its an invalid response.
            return Response.GetInvalidRequestResponse();
        }
```

Here is the call graph for this function:


OSXJV.Classes.Output.CreateView
OSXJV.Classes.Response.Get ErrorResponse
OSXJV.Classes.CacheManager.getFile
OSXJV.Classes.Logger.Get Instance
OSXJV.Server.OSXJVServer. HandleGet
OSXJV.Classes.CacheManager. GetInstance
OSXJV.Classes.Response.Get InvalidRequestResponse
OSXJV.Classes.Response.Get Response
OSXJV.Classes.Logger.Write Error

5.5.3.6 HandleOptions()

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

Sends to the Client What the Server Supports

Parameters

response	The Request Response Object

Definition at line 171 of file OSXJVServer.cs.

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

5.5.3.7 HandlePost()

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

Handles a POST request.

Parameters

req	The request to be processed.

Returns

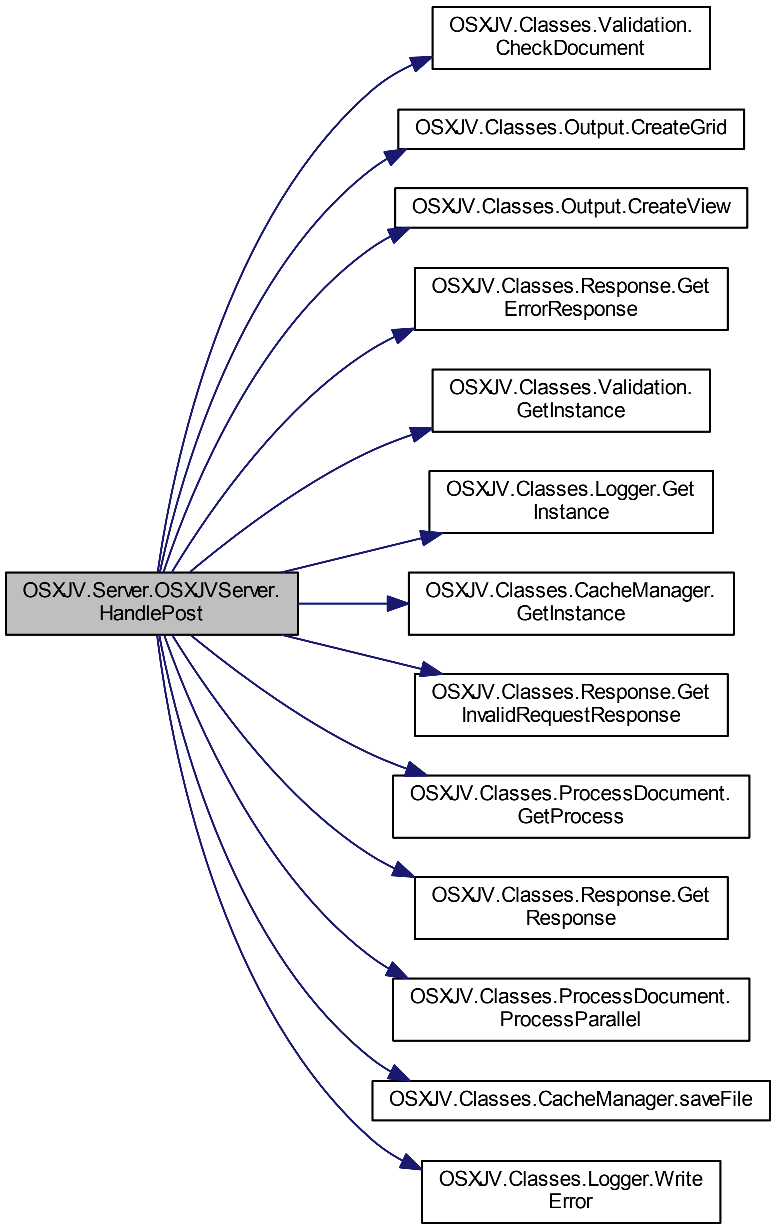
A Response object to send to the user

Definition at line 234 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.Validation.GetInstance(), OSXJV.Classes.Logger.GetInstance(), OSXJV.Classes.CacheManager.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().

```
        {

            JObject eRes = new JObject();

            if (SegmentNormalize(req.RawUrl).Equals("Process"))
            {
                if (req.HasEntityBody)
                {


                    Request r = null;
                    try
                    {
                        r = GetData(req);
                        if (r == null)
                            return Response.GetInvalidRequestResponse();
                    }
                    catch
                    {
                       return Response.GetInvalidRequestResponse();
                    }



                    try
                    {
                        Validation.GetInstance().
      CheckDocument(r.Data, r.Type);
                    }
                    catch (Exception e)
                    {
                        eRes.Add("Error", e.Message);
                        return Response.GetErrorResponse(eRes.ToString());
                    }

                    string id = Guid.NewGuid().ToString();
                    ProcessDocument pro = ProcessDocument.
      GetProcess(r.Data, r.Type);
                    Node n = pro.ProcessParallel();
                    Output o = new Output(n); //new output object
                    try
                    {
                        CacheManager.GetInstance().
      saveFile(id, JsonConvert.SerializeObject(n));
                        JObject response = new JObject();

                        n = null; //remove node as its completed;

                        response.Add("filename", id);
                        response.Add("grid", o.CreateGrid());
                        response.Add("view", o.CreateView());



                        byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
                        return Response.GetResponse(200, "application/json", bytes);
                    }
                    catch (Exception e)
                    {
                        Logger.GetInstance().WriteError(e.Message);
                        eRes.Add("Error", "Error Creating Response");
                        return Response.GetErrorResponse(eRes.ToString());
                    }

                }
                eRes.Add("Error", "No File Recieved By Server");
                return Response.GetErrorResponse(eRes.ToString());
            }
            else if (req.RawUrl.Equals("/Output"))
            {
                return Response.GetInvalidRequestResponse();
            }
            else
                return Response.GetInvalidRequestResponse();
        }
```

Here is the call graph for this function:


OSXJV.Server.OSXJVServer.HandlePost
OSXJV.Classes.Validation.CheckDocument
OSXJV.Classes.Output.CreateGrid
OSXJV.Classes.Output.CreateView
OSXJV.Classes.Response.GetErrorResponse
OSXJV.Classes.Validation.GetInstance
OSXJV.Classes.Logger.GetInstance
OSXJV.Classes.CacheManager.GetInstance
OSXJV.Classes.Response.GetInvalidRequestResponse
OSXJV.Classes.ProcessDocument.GetProcess
OSXJV.Classes.Response.GetResponse
OSXJV.Classes.ProcessDocument.ProcessParallel
OSXJV.Classes.CacheManager.saveFile
OSXJV.Classes.Logger.WriteError

5.5.3.8 ListenerCallback()

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

Handles Requests Asyncronously

Parameters

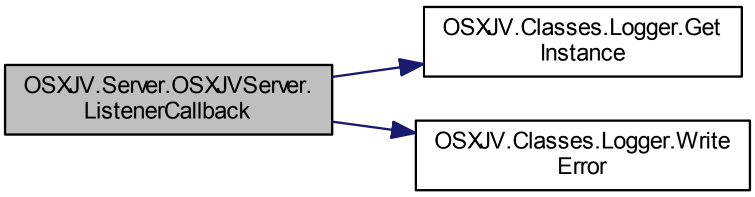
result	The Request Object Coming In.

Definition at line 125 of file OSXJVServer.cs.

References OSXJV.Classes.Logger.GetInstance(), and OSXJV.Classes.Logger.WriteError().

```
        {
            HttpListener listener = (HttpListener)result.AsyncState;
            HttpListenerContext context = listener.EndGetContext(result);
            try
            {
                HandleClient(context);
            }
            catch (Exception e)
            {
                Logger.GetInstance().WriteError(e.Message);
                context.Response.StatusCode = 500;
                context.Response.Close();
            }

        }
```

Here is the call graph for this function:



5.5.3.9 ManageCache()

```
void OSXJV.Server.OSXJVServer.ManageCache ( )  [private]
```

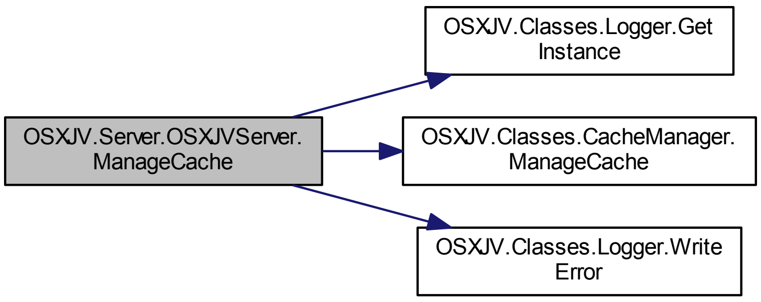
Manages cache

Definition at line 446 of file OSXJVServer.cs.

References OSXJV.Classes.Logger.GetInstance(), OSXJV.Classes.CacheManager.ManageCache(), and OSXJ↩
V.Classes.Logger.WriteError().

```
        {
            while (true)
            {
                Thread.Sleep(3600000);
                try
                {
                    CacheManager.ManageCache();
                }
                catch (Exception e)
                {
                    try
                    {
                        Logger.GetInstance().WriteError(e.Message);
                    }
                    catch
                    {
                        Console.WriteLine("Logger and Cache Manager not setup");
                    }
                }

            }
        }
```

Here is the call graph for this function:


OSXJV.Classes.Logger.Get Instance
OSXJV.Server.OSXJVServer. ManageCache
OSXJV.Classes.CacheManager. ManageCache
OSXJV.Classes.Logger.Write Error

5.5.3.10 Post()

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

Send data to the client.

Parameters

res	The Response Object
stream	The Client Output Stream

///

Exceptions

ArgumentException	Thrown when Response is null or HttpListenerResponse is null or empty

Definition at line 395 of file OSXJVServer.cs.

References OSXJV.Classes.Response.data, OSXJV.Classes.Response.mime, and OSXJV.Classes.Response.↩
status.

```
        {
            if (res == null || stream == null)
                throw new ArgumentException("Response or Client Stream cannot be NULL");

            HandleOptions(stream);
            stream.ProtocolVersion = new Version(1, 1);
            stream.StatusCode = res.status;
            stream.ContentType = res.mime;
            stream.ContentLength64 = res.data.Length;
            stream.OutputStream.Write(res.data, 0, res.data.Length);
            stream.Close();
        }
```

5.5.3.11 Run()

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

Function that constantly listens for connections

Definition at line 100 of file OSXJVServer.cs.

```
        {
            running = true;
            listener.Start();


            while(listener.IsListening)
            {

                Console.WriteLine("Waiting");

                //Wait for Listener
                IAsyncResult result = listener.BeginGetContext(new AsyncCallback(
      ListenerCallback), listener);
                result.AsyncWaitHandle.WaitOne();

                if (result.CompletedSynchronously)
                    Console.WriteLine("Completed Synchronously");
            }
        }
```

5.5.3.12 SaveFile()

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

Save data recievied from client.

Parameters

id	Unique ID
nodes	The Processed Data

Exceptions

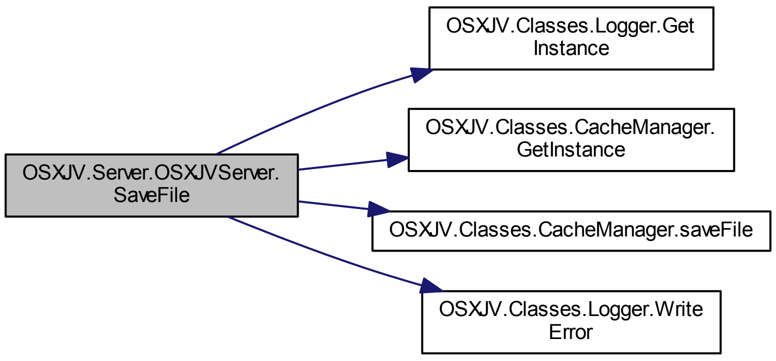
ArgumentException	Thrown when nodes is null or id is null or empty

Definition at line 372 of file OSXJVServer.cs.

References OSXJV.Classes.Logger.GetInstance(), OSXJV.Classes.CacheManager.GetInstance(), OSXJV.↩ Classes.CacheManager.saveFile(), and OSXJV.Classes.Logger.WriteError().

```
        {
            if(nodes == null || string.IsNullOrEmpty(id))
            {
                throw new ArgumentException();
            }

            try
            {
                CacheManager.GetInstance().saveFile(id, JsonConvert.
      SerializeObject(nodes));
            }
            catch (Exception e)
            {
                Logger.GetInstance().WriteError(e.Message);
            }
        }
```

Here is the call graph for this function:


OSXJV.Server.OSXJVServer.SaveFile
OSXJV.Classes.Logger.GetInstance
OSXJV.Classes.CacheManager.GetInstance
OSXJV.Classes.CacheManager.saveFile
OSXJV.Classes.Logger.WriteError

5.5.3.13 SegmentNormalize()

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

Removes '/' from the string.

Parameters

input	A string from the URL

Returns

Normalised String

Definition at line 438 of file OSXJVServer.cs.

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

5.5.3.14 Start()

```
bool OSXJV.Server.OSXJVServer.Start (
            string cachePath,
            string loggerPath )
```

Starts server in new thread

Parameters

cachePath	Cache Folder Location
loggerPath	Logger Folder Location

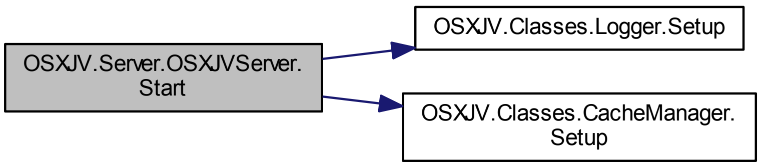
Definition at line 54 of file OSXJVServer.cs.

References OSXJV.Classes.Logger.Setup(), and OSXJV.Classes.CacheManager.Setup().

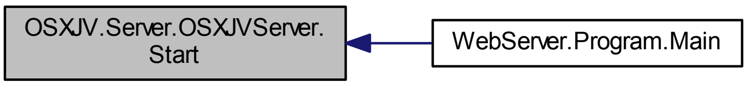
Referenced by WebServer.Program.Main().

```
        {
            bool success = false;

            success = CacheManager.Setup(cachePath);
            success = Logger.Setup(loggerPath);

            serverThread = new Thread(new ThreadStart(Run)); //Server thread
            cacheThread = new Thread(new ThreadStart(ManageCache)); //Cache manage
      thread
            try
            {
                serverThread.Start();
                cacheThread.Start();
            }
            catch(Exception e)
            {
                throw e;
            }

            success = cacheThread.IsAlive;
            success = serverThread.IsAlive;

            return success;
        }
```

Here is the call graph for this function:


OSXJV.Server.OSXJVServer.
Start
OSXJV.Classes.Logger.Setup
OSXJV.Classes.CacheManager.
Setup

Here is the caller graph for this function:



5.5.3.15 Stop()

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

Stop the listener and about all current requests

Definition at line 82 of file OSXJVServer.cs.

```
        {
            if (listener != null)
                if (listener.IsListening)
                    listener.Abort();


            if (serverThread != null)
            {
                serverThread.Join();
                serverThread = null;
            }

            return serverThread == null ?true:false;
        }
```

5.5.4 Member Data Documentation

5.5.4.1 cacheThread

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

Cache Controller

Definition at line 38 of file OSXJVServer.cs.

5.5.4.2 listener

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

HttpListener

Definition at line 28 of file OSXJVServer.cs.

5.5.4.3 port

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

Definition at line 18 of file OSXJVServer.cs.

5.5.4.4 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.

5.5.4.5 serverThread

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

Server Thread

Definition at line 33 of file OSXJVServer.cs.

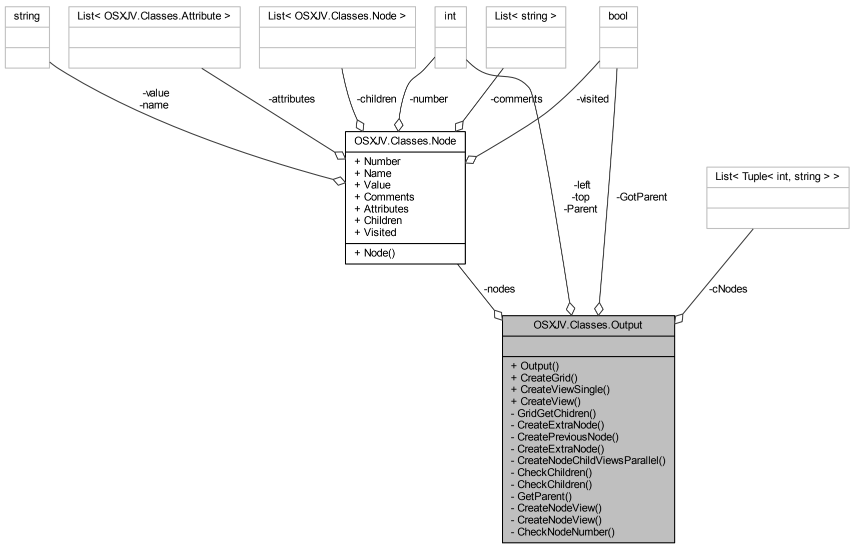
The documentation for this class was generated from the following file:

- WebServiceCSharp/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

nodes	A processed object of Nodes

Definition at line 42 of file Output.cs.

```
00043         {
```

```
            if (nodes == null)
                throw new ArgumentException();
            this.nodes = nodes;
        }
```

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

n	Node to search
number	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.

```
        {
            string output = "";
            if (CheckNodeNumber(n, number))
            {
                int count = 0;
                output += CreateNodeView(n, "node");
                foreach (Node n2 in n.Children)
                {
                    count++;
                    output += CreateNodeView(n2, "node-child");

                }
            }
            else if (n.Children.Count > 0)
            {
                foreach (Node n2 in n.Children)
                {
                    if (GotParent)
                    {
                        if (n2.Number == Parent)
                        {
                            output += CreateNodeView(n2, "node-parent");
                        }
                    }
                    output += CheckChildren(n2, number);
                }
            }

            return output;
        }
```

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

n	
number	
pCount	
nodeStart	
found	

Returns

String of calculated HTML

Definition at line 440 of file Output.cs.

References OSXJV.Classes.Node.Children, and OSXJV.Classes.Node.Number.

```
        {
            string output = "";
            if (CheckNodeNumber(n, number))
            {
                found = true;
                List<Thread> threadList = new List<Thread>();

                int count = 0;
                output += CreateNodeView(n, "node");
                count++;
                //output += CreateNodeView(n2, "node-child");
                int childCount = 0;

                if (n.Children.Count < 200)
                    childCount = n.Children.Count;
                else
                {
                    childCount = 200;
                }
                if (childCount < pCount * 2)
                {
                    foreach(Node n2 in n.Children)
                    {
                        output += CreateNodeView(n2, "node-child");
                    }
                }
                else
                {
                    int spread = (int)Math.Ceiling((double)childCount / (double)pCount);

                    if (childCount > 0)
                    {
                        for (int i = 0; i < pCount; i++)
                        {
                            int neg = 0;
                            if ((spread * (i + 1)) > childCount)
                            {
                                neg = childCount - (spread * (i + 1));
                            }
                            int start = (spread * i);
                            int rangeStart = (spread * i) + nodeStart;
                            bool showHigher = nodeStart != 0 ? true : false;

                            List<Node> NodesToProcess = n.Children.GetRange(rangeStart, spread + neg);

                            if (NodesToProcess.Count > 0)
                            {
                                Thread threadJob = new Thread(() =>
CreateNodeChildViewsParallel(NodesToProcess, start, showHigher, childCount +
nodeStart, nodeStart - childCount));
                                threadJob.Name = i.ToString();
                                threadJob.Start();
                                threadList.Add(threadJob);
                            }
                        }
                        foreach (Thread t in threadList)
                        {
                            t.Join();
                        }
                        cNodes.Sort((x, y) => x.Item1.CompareTo(y.Item1));

                        foreach (Tuple<int, string> tup in cNodes)
                        {
                            output += tup.Item2;
                        }
                    }
                }
            }
            else if (n.Children.Count > 0)
            {
                foreach (Node n2 in n.Children)
                {
                    if (GotParent)
                    {
                        if (n2.Number == Parent)
                        {
                            output += CreateNodeView(n2, "node-parent");
                        }
                    }
                    output += CheckChildren(n2, number,pCount,nodeStart,ref found);
                }
            }

            return output;
        }
```

5.6.3.3 CheckNodeNumber()

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

Checks if Node number and inputted number match.

Parameters

n	Node to search
number	Number to match with

Returns

Definition at line 661 of file Output.cs.

References OSXJV.Classes.Node.Number.

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

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

type	Node type e.g. 'node-child'
id	The id of the node to start from

Returns

String of calculated HTML

Definition at line 164 of file Output.cs.

```
        {
            string node = "";

            if (type == "node")
            {
                if (GotParent)
                {
                    left = left + 400;
                }
            }
            if (type == "node-child")
            {
                left = left + 400;
            }

            node += "<div class='node-child type ui-draggable ui-selectee' style='left:" +
      left + "px; top:" + top + "px;margin-bottom:50px;'>";
            node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
      )'>Show Lower</button></span></div>";
            node += "</div></div>";
            return node;
        }
```

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

type	Node type e.g. 'node-child'
leftVal	Margin from the left of the display
topVal	Margin from the top of the display
id	The id of the node to start from

Returns

String of calculated HTML

Definition at line 223 of file Output.cs.

```
        {
            string node = "";

            if (type == "node")
            {
                if (GotParent)
                {
                    leftVal = leftVal + 400;
                }
            }
            if (type == "node-child")
            {
                leftVal = leftVal + 400;
            }
            node += "<div class='node-child type ui-draggable ui-selectee' style='left:" + leftVal + "px;top:" + topVal + "px;margin-bottom:50px;'>";
            node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + ")'>Show Lower</button></span></div>";
            node += "</div></div>";
            return node;
        }
```

5.6.3.6 CreateGrid()

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

Creates the grid data.

Returns

A JSON object

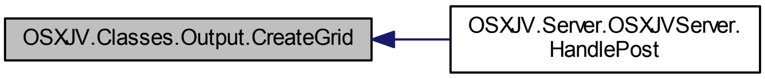
Definition at line 53 of file Output.cs.

References OSXJV.Classes.Node.Children, OSXJV.Classes.Node.Name, and OSXJV.Classes.Node.Number.

Referenced by OSXJV.Server.OSXJVServer.HandlePost().

```
        {
            JObject obj = new JObject();
            obj.Add("text", nodes.Name);
            obj.Add("id", nodes.Number);
            obj.Add("state", new JObject(new JProperty("selected", true)));

            if(nodes.Children.Count > 0)
            {
                JArray array = new JArray();
                foreach (Node n2 in nodes.Children)
                {
                    array.Add(GridGetChidren(n2));
                }
                obj.Add("children", array);
            }
            return obj;
        }
```

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

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

Definition at line 252 of file Output.cs.

References OSXJV.Classes.Node.Comments.

```
        {
            int threadID = int.Parse(Thread.CurrentThread.Name);
            string type = "node-child";
            string output = "";

            if(start == 0 && showHigher)
            {
                output += CreatePreviousNode(type, left,
      top, previous);
            }
            bool hadCommentsPrev = false;
            int numCommentsPrevious = 0;

            foreach(Node n in job)
            {
                int extra = showHigher ? 130 * (start +1) : 130 * start;

                if (hadCommentsPrev)
                    extra += (numCommentsPrevious * 25);

                if (n.Comments.Count > 0)
                {
                    hadCommentsPrev = true;
                    numCommentsPrevious = n.Comments.Count;
                }
                else
                    hadCommentsPrev = false;

                output += CreateNodeView(n, type,left,top + extra);
                start++;
                if (start == 200)
                {
                    output += CreateExtraNode(type, left, top + extra + 130,next);
                    break;
                }

            }

            cNodes.Add(new Tuple<int, string>(threadID, output));
        }
```

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

n	Node to parse
type	Type of node
leftVal	Margin left of display
topVal	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.

```
        {
            string node = "";

            if(type == "node")
            {
                if(GotParent)
                {
                    leftVal = leftVal + 400;
                }
            }
            if(type == "node-child")
            {
                leftVal = leftVal + 400;
            }
            node += "<div id='" + n.Number + "'class='" + type + " type ui-draggable ui-selectee'
       style='left:" + leftVal + "px; top:" + topVal + "px;'>";
            node += "<div class='head'><span><button class='nameBtn' onclick='GetNode("+n.Number+")'>" + n.
```

```
      Name + "</button></span></div>";
            if (!string.IsNullOrEmpty(n.Value))
            {
                node += string.Format("<div class='blockR'><p>Value</p></div><div
       class=comment><span>{0}</span></div>", n.Value);
            }
            if(n.Comments.Count >0)
            {
                node += "<div><p class='text-center'>Comments</p></div>";

                foreach(string com in n.Comments)
                {
                    node += "<div class='comment'>" + com + "</div>";
                }
            }
            if (n.Attributes.Count > 0)
            {
                node += "<div class='attribute'><div class='aHeader'><p><button><i class='fa
       fa-plus'></i></button>Attributes</p></div><div class='options'>";
                foreach (Attribute a in n.Attributes)
                {
                    node += string.Format("<div class='blockR'><p>{0}</p></div><div
       class='comment'><p>{1}</p></div>", a.Name, a.Value);
                }
                node += "</div>";
            }
            node += "</div></div>";
            return node;
        }
```

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

n	Node to parse
type	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.

```
        {
            string node = "";
            int leftVal = left;
            if (type == "node")
            {
                if (GotParent)
                {
                    left = left + 400;
                    leftVal = left;
                }
            }
            if (type == "node-child")
            {
                leftVal = leftVal + 400;
            }
            node += "<div id='" + n.Number + "'class='" + type + " type ui-draggable ui-selectee'
       style='left:" + leftVal + "px; top:" + top + "px;'>";
            node += "<div class='head'><span><button class='nameBtn' onclick='GetNode(" + n.Number + ")'>"
      + n.Name + "</button></span></div>";
            if (!string.IsNullOrEmpty(n.Value))
            {
                node += string.Format("<div class='blockR'><p>Value</p></div><div
       class=comment><span>{0}</span></div>", n.Value);
            }
            if (n.Comments.Count > 0)
            {
                node += "<div><p class='text-center'>Comments</p></div>";

                foreach (string com in n.Comments)
                {
                    node += "<div class='comment'>" + com + "</div>";
                }
            }
            if (n.Attributes.Count > 0)
            {
                node += "<div class='attribute'><div class='aHeader'><p><button><i class='fa
       fa-plus'></i></button>Attributes</p></div><div class='options'>";
                foreach (Attribute a in n.Attributes)
                {
                    node += string.Format("<div class='blockR'><p>{0}</p></div><div
       class='comment'><p>{1}</p></div>", a.Name, a.Value);
                }
                node += "</div>";
            }
            node += "</div></div>";

            if (type == "node-child")
            {
                top = top + 130;
            }
            return node;
        }
```

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

type	Node type e.g. 'node-child'
leftVal	Margin from the left of the display
topVal	Margin from the top of the display
id	The id of the node to start from

Returns

String of calculated HTML

Definition at line 194 of file Output.cs.

```
        {
            string node = "";

            if (type == "node")
            {
                if (GotParent)
                {
                    leftVal = leftVal + 400;
                }
            }
            if (type == "node-child")
            {
                leftVal = leftVal + 400;
            }
            node += "<div class='node-child type ui-draggable ui-selectee' style='left:" + leftVal + "px;
      top:" + topVal + "px;'>";
            node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
     )'>Show Higher</button></span></div>";
            node += "</div></div>";
            return node;
        }
```

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

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

Returns

String of calculated HTML

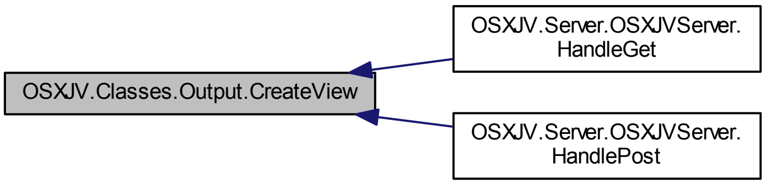
Definition at line 300 of file Output.cs.

References OSXJV.Classes.Node.Children, and OSXJV.Classes.Node.Number.

Referenced by OSXJV.Server.OSXJVServer.HandleGet(), and OSXJV.Server.OSXJVServer.HandlePost().

```
        {

            List<Thread> threadList = new List<Thread>();

            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'>";
            if (nodes.Number.Equals(node))
            {
                int childCount = 0;

                if (nodes.Children.Count < 200)
                    childCount = nodes.Children.Count;
                else
                {
                    childCount = 200;
                }

                if(childCount < pCount * 2)
                {
                    output += CreateNodeView(nodes, "node", left, top);
                    foreach(Node n2 in nodes.Children)
                    {
                        output += CreateNodeView(n2, "node-child");
                    }
                }
                else
                {
                    int spread = (int)Math.Ceiling((double)childCount / (double)pCount);

                    output += CreateNodeView(nodes, "node", left,top);  //Parent(Node) Thread

                    for (int i = 0; i < pCount; i++)
                    {
                        int neg = 0;
                        if ((spread * (i + 1)) > childCount)
                        {
                            neg = childCount - (spread * (i + 1));
                        }
                        int start = (spread * i) ;
                        int rangeStart = (spread * i) + nodeStart;
                        bool showHigher = nodeStart != 0 ? true : false;

                        List<Node> NodesToProcess = nodes.Children.GetRange(rangeStart, spread + neg);
                        Thread threadJob = new Thread(() => CreateNodeChildViewsParallel(NodesToProcess, start, showHigher, childCount + nodeStart, nodeStart - childCount));
                        threadJob.Name = i.ToString();
                        threadJob.Start();
                        threadList.Add(threadJob);
                    }
                    foreach(Thread t in threadList)
                    {
                        t.Join();
                    }

                    cNodes.Sort((x, y) => x.Item1.CompareTo(y.Item1));

                    foreach(Tuple<int,string> tup in cNodes)
                    {
                        output += tup.Item2;
                    }
                }
            }
            else
            {
                GetParent(nodes, node);
                string temp = "";
                if (GotParent)
                {
                    if (nodes.Number == Parent)
                    {
                        output += CreateNodeView(nodes, "node-parent");
                    }
                }
                bool found =false;
                foreach (Node n2 in nodes.Children)
                {
                    if (GotParent)
                    {
                        if (n2.Number == Parent)
                        {
                            output += CreateNodeView(n2, "node-parent");
                        }
                    }
                    temp += CheckChildren(n2, node,pCount,nodeStart,ref found);
                    if (found)
                        break;
                }
                if (!string.IsNullOrEmpty(temp))
                    output += temp;
            }
            output += "</div></div>";
            return output;
        }
```

Here is the caller graph for this function:


OSXJV.Classes.Output.CreateView
OSXJV.Server.OSXJVServer.HandleGet
OSXJV.Server.OSXJVServer.HandlePost

5.6.3.12 CreateViewSingle()

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

CreateView using a Single Thread

Parameters

node	Index of node to start from
nodeStart	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.

```
        {
            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'>";

            if (nodes.Number.Equals(node))
            {
                int count = 0;
                output += CreateNodeView(nodes, "node");


                foreach (Node n in nodes.Children)
                {
                    if(nodeStart > 0)
                    {
                        if (count != nodeStart)
                            continue;
                    }
                    count++;
                    output += CreateNodeView(n, "node-child"); //Child(Nodes) Thread

                    if ((count-nodeStart) == 200)
                    {
                        output += CreateExtraNode("node-child",count);
                        break;
                    }
                }
            }

            else
            {
                GetParent(nodes, node);
                string temp = "";
                if (GotParent)
                {
                    if (nodes.Number == Parent)
                    {
                        output += CreateNodeView(nodes, "node-parent");
                    }
                }
                foreach (Node n2 in nodes.Children)
                {
                    if (GotParent)
                    {
                        if (n2.Number == Parent)
                        {
                            output += CreateNodeView(n2, "node-parent");
                        }
                    }
                    temp += CheckChildren(n2, node);
                }
                if (!string.IsNullOrEmpty(temp))
                    output += temp;
            }
            output += "</div></div>"; //Close out divs
            return output;
        }
```

5.6.3.13 GetParent()

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

Finds the parent node.

Parameters

node	Node to search
number	Node number to find

Definition at line 530 of file Output.cs.

References OSXJV.Classes.Node.Children, and OSXJV.Classes.Node.Number.

```
        {
            if(!CheckNodeNumber(node,number))
            {
                foreach(Node n in node.Children)
                {
                    if(CheckNodeNumber(n, number))
                    {
                        Parent = node.Number;
                        GotParent = true;
                    }
                    else
                    {
                        GetParent(n, number);
                    }
                }
            }
        }
```

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

n	Child Node

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.

```
        {
            JObject child = new JObject();
            child.Add("id", n.Number);
            child.Add("text", n.Name);

            if (n.Children.Count > 0)
            {
                JArray array = new JArray();
                foreach(Node n2 in n.Children)
                {
                    array.Add(GridGetChidren(n2));
                }
                child.Add("children", array);
            }
            return child;
        }
```

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.

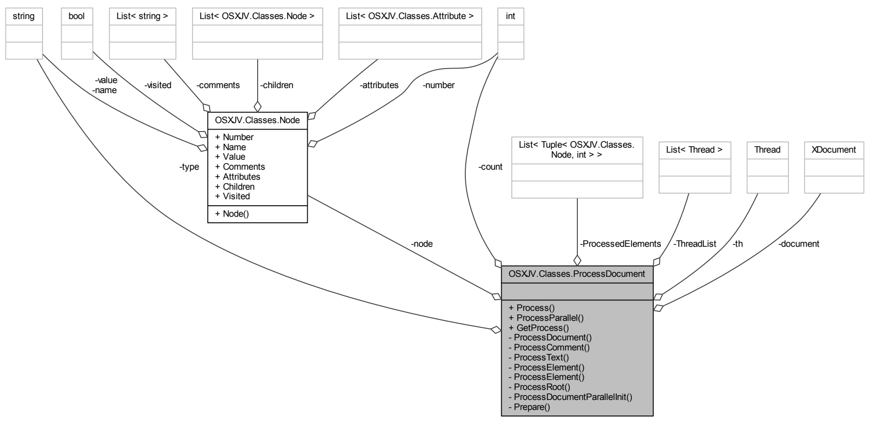
The documentation for this class was generated from the following file:

- WebServiceCSharp/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

doc	Parsed document
type	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

data	String of the document
type	Type of document

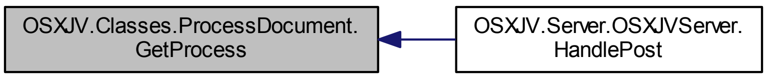
Returns

Definition at line 77 of file ProcessDocument.cs.

Referenced by OSXJV.Server.OSXJVServer.HandlePost().

```
        {
            if (string.IsNullOrEmpty(data) || string.IsNullOrEmpty(type))
            {
                throw new ArgumentException();
            }
            try
            {
                XDocument doc = null;
                doc = Prepare(data, type);
                return new ProcessDocument(doc, type);
            }
            catch (System.Xml.XmlException e)
            {
                throw e;
            }
        }
```

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

data	String of data
type	Data type

Returns

A XDocument object

Definition at line 111 of file ProcessDocument.cs.

```
        {

            if (type.Equals("JSON"))
                return new XDocument(JsonConvert.DeserializeXNode(data, "Root", false).Root.FirstNode);
            else if (type.Equals("XML") || type.Equals("HTML"))
                return XDocument.Parse(data);

            return null;
        }
```

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.

```
        {
            if (document.Nodes() != null)
            {
                foreach (XNode n in document.Nodes())
                {
                    switch (n.NodeType)
                    {
                        case System.Xml.XmlNodeType.Element:
                            count++;
                            ProcessElement(XElement.Parse(n.ToString()),
      node);
                            break;
                        case System.Xml.XmlNodeType.Comment:
                            ProcessComment(n as XComment, node);
                            break;
                        case System.Xml.XmlNodeType.Text:
                            ProcessText(n as XText, node);
                            break;
                        case System.Xml.XmlNodeType.Notation:
                            break;
                        case System.Xml.XmlNodeType.EndElement:
                            break;
                        default:
                            break;
                    }
                }
            }
            //SortArray(ref node);
            document = null;
            return node;
        }
```

5.7.3.4 ProcessComment()

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

Extract Comment

Parameters

e	Comment object to be parsed
node	Node to input data

Definition at line 64 of file ProcessDocument.cs.

References OSXJV.Classes.Node.Comments.

```
        {
            string s = "";
            s = Regex.Replace(e.Value, @"[^\w\s\.@-]", "");
            node.Comments.Add(s);
        }
```

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

doc	A subset of the full document
start	Start index number

Definition at line 336 of file ProcessDocument.cs.

References OSXJV.Classes.Node.Children.

```
        {
            int nodeNum = start;

            Node node = new Node();
            if (doc.Root.Nodes() != null)
            {
                List<XNode> list = doc.Root.Nodes().ToList();
                foreach (XNode n in doc.Root.Nodes())
                {
                    switch (n.NodeType)
                    {
                        case System.Xml.XmlNodeType.Element:
                            nodeNum++;
                            Node n2 = new Node();
                            node.Children.Add(ProcessElement(XElement.Parse(n.ToString()), n2
, ref nodeNum));
                            break;
                        case System.Xml.XmlNodeType.Comment:
                            ProcessComment(n as XComment, node);
                            break;
                        case System.Xml.XmlNodeType.Text:
                            ProcessText(n as XText, node);
                            break;
                        case System.Xml.XmlNodeType.Notation:
                            break;
                        case System.Xml.XmlNodeType.EndElement:
                            break;
                        default:
```

```
                        break;
                    }
                }
            }
            document = null;
            ProcessedElements.Add(new Tuple<Node, int>(node, start));
        }
```

5.7.3.6 ProcessElement() [1/2]

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

Single Threaded Process Element Version

Parameters

e	Element to Process
node	The Node 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.

```
        {
            if (node.Number == 0)
            {
                node.Number = count;
            }
            if (!node.Visited)
            {

                node.Name = e.Name.LocalName;
                foreach (XAttribute ax in e.Attributes())
                {
                    if (ax.Name == "id")
                    {
                        node.Name = node.Name + " #" + ax.Value;
                    }

                    if (type == "HTML")
                    {
                        if (ax.IsNamespaceDeclaration)
                            continue;
                    }
                    Attribute att = new Attribute();
                    att.Name = ax.Name.LocalName;
                    att.Value = ax.Value;
                    node.Attributes.Add(att);
                }
            }

            if (e.Nodes() != null)
            {
                foreach (XNode n in e.Nodes())
                {
                    switch (n.NodeType)
                    {
                        case System.Xml.XmlNodeType.EndElement:
                            break;
                        case System.Xml.XmlNodeType.Element:
                            count++;
                            Node n2 = new Node();
                            node.Children.Add(ProcessElement(XElement.Parse(n.
      ToString()), n2));
                            break;
                        case System.Xml.XmlNodeType.Comment:
                            ProcessComment(n as XComment, node);
                            break;
                        case System.Xml.XmlNodeType.Text:
                            ProcessText(n as XText, node);
                            break;
                        case System.Xml.XmlNodeType.Notation:
                            break;

                        default:
                            break;
                    }
                }
            }
            node.Visited = true;
            return node;
        }
```

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

e	Element to process
node	Node to extract data from
nodeNumber	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.

```
        {
            if (!node.Visited)
            {
                if (node.Number == 0)
                {
                    node.Number = nodeNumber;
                }
                if (!node.Visited)
                {

                    node.Name = e.Name.LocalName;
                    foreach (XAttribute ax in e.Attributes())
                    {
                        if (ax.Name == "id")
                        {
                            node.Name = node.Name + " #" + ax.Value;
                        }

                        if (type == "HTML")
                        {
                            if (ax.IsNamespaceDeclaration)
                                continue;
                        }
                        Attribute att = new Attribute();
                        att.Name = ax.Name.LocalName;
                        att.Value = ax.Value;
                        node.Attributes.Add(att);
                    }
                }

                if (e.Nodes() != null)
                {
                    foreach (XNode n in e.Nodes())
                    {
                        switch (n.NodeType)
                        {
                            case System.Xml.XmlNodeType.EndElement:
                                break;
                            case System.Xml.XmlNodeType.Element:
                                nodeNumber++;
                                Node n2 = new Node();
                                node.Children.Add(ProcessElement(XElement.Parse(n
.ToString()), n2, ref nodeNumber));
                                break;
                            case System.Xml.XmlNodeType.Comment:
                                ProcessComment(n as XComment, node);
                                break;
                            case System.Xml.XmlNodeType.Text:
                                ProcessText(n as XText, node);
                                break;
                            case System.Xml.XmlNodeType.Notation:
                                break;

                            default:
                                break;
                        }
                    }
                }
                node.Visited = true;
            }
            return node;
        }
```

5.7.3.8 ProcessParallel()

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

Parse Document Using Multiple Threads

Parameters

pCount	Number of Threads to run Default = 4

Returns

A object of Node that has been processed

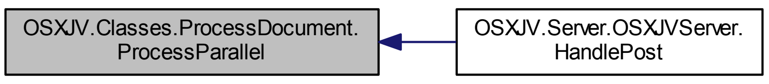
Definition at line 377 of file ProcessDocument.cs.

References OSXJV.Classes.Node.Children.

Referenced by OSXJV.Server.OSXJVServer.HandlePost().

```
        {
            node = ProcessRoot(document.Root, node);

            int nodeCount = document.Root.Nodes().Count();

            if(nodeCount <= pCount)
            {
                return Process();
            }
            else if (nodeCount > pCount)
            {

                List<XNode> List = document.Root.Nodes().ToList();
                int spread = 0;

                spread = (int)Math.Ceiling((double)nodeCount / (double)pCount);

                int totalNodes = 1;

                for (int i = 0; i < pCount; i++)
                {
                    int neg = 0;
                    int start = totalNodes;
                    if ((spread * (i+1)) > nodeCount)
                    {
                        neg = nodeCount - (spread * (i + 1));
                    }

                    List<XNode> list = List.GetRange((spread * i), spread + neg);
                    XElement root = new XElement("Root", list);
                    XDocument doc = new XDocument(root);

                    (th = new Thread(() => ProcessDocumentParallelInit(doc, start))).Start();

                    ThreadList.Add(th); //Add to Threads list to keep recored of threads running
                    totalNodes += root.Descendants().Count(); //Increment start position.
                }
                document = null;
                foreach (Thread t in ThreadList)
                {
                    t.Join(); //Wait for threads to join
                }

                ProcessedElements.Sort((x, y) => x.Item2.CompareTo(y.Item2)); //Sort List by start index so they are in order.

                foreach(Tuple<Node,int> tup in ProcessedElements)
                {
                    foreach(Node n in tup.Item1.Children)
                    {
                        node.Children.Add(n);
                    }
                }
            }
            return node;
        }
```

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

e	Element object to process
node	Node 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.

```
        {
            node.Number = 1;

            if (!node.Visited)
            {

                node.Name = e.Name.LocalName;
                foreach (XAttribute ax in e.Attributes())
                {
                    if (ax.Name == "id")
                    {
                        node.Name = node.Name + " #" + ax.Value;
                    }

                    if (type == "HTML")
                    {
                        if (ax.IsNamespaceDeclaration)
                            continue;
                    }
                    Attribute att = new Attribute();
                    att.Name = ax.Name.LocalName;
                    att.Value = ax.Value;
                    node.Attributes.Add(att);
                }
            }
            node.Visited = true;
            return node;
        }
```

5.7.3.10 ProcessText()

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

Get text from the data

Parameters

e	Text Element
n	Node to input data

Definition at line 100 of file ProcessDocument.cs.

References OSXJV.Classes.Node.Value.

```
        {
            n.Value = e.Value;
        }
```

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.

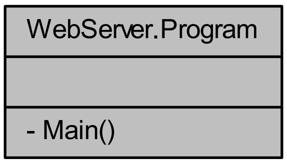
The documentation for this class was generated from the following file:

- WebServiceCSharp/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

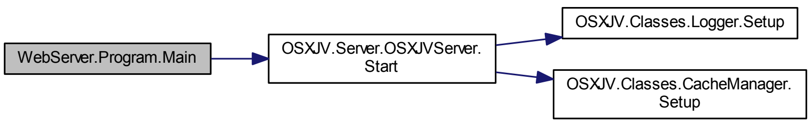
args	Pass Cache Folder and Logger (Optional)

Definition at line 18 of file Program.cs.

References OSXJV.Server.OSXJVServer.Start().

```
        {

            if (args.Length == 0)
            {
                Console.WriteLine("Using Default Cache Directory Path and Logger Directory Path");
                string dir = Directory.GetCurrentDirectory();
                Array.Resize(ref args, 2);
                args[0] = dir + "/Cache/";
                args[1] = dir + "/Logger/";
                if (!Directory.Exists(args[0]))
                    Directory.CreateDirectory(args[0]);
                if (!Directory.Exists(args[1]))
                    Directory.CreateDirectory(args[1]);
            }

            if (args[0] == args[1])
            {
                Console.WriteLine("Cache location and Log location is the same. Please enter two different
       locations");
            }
            else
            {
                try
                {
                    OSXJVServer s = new OSXJVServer();
                    s.Start(args[0], args[1]);
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    Console.WriteLine("Press any key to exit");
                    Console.Read();
                }

            }
        }
```

Here is the call graph for this function:


WebServer.Program.Main
OSXJV.Server.OSXJVServer.Start
OSXJV.Classes.Logger.Setup
OSXJV.Classes.CacheManager.Setup

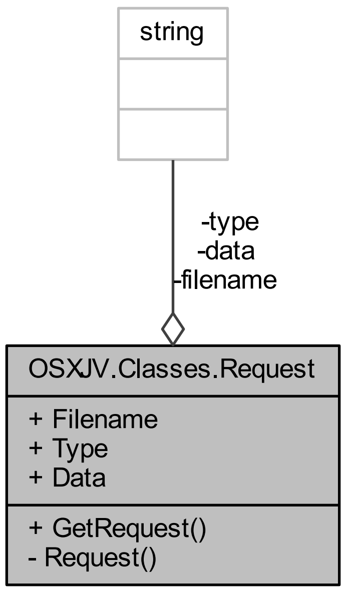
The documentation for this class was generated from the following file:

- WebServiceCSharp/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

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

Definition at line 31 of file Request.cs.

```
        {
            this.filename = filename;
            this.type = type;
            this.data = data;
        }
```

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

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

Returns

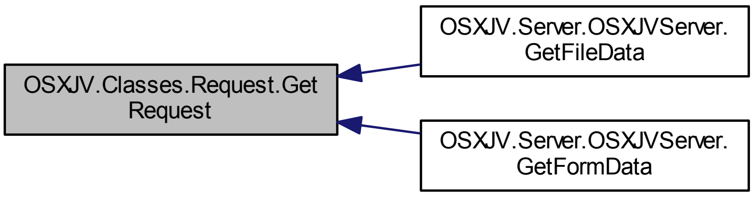
Object of Request

Definition at line 45 of file Request.cs.

Referenced by OSXJV.Server.OSXJVServer.GetFileData(), and OSXJV.Server.OSXJVServer.GetFormData().

```
        {
            string Type = "";
            if (string.IsNullOrEmpty(filename) || string.IsNullOrEmpty(
    type) || string.IsNullOrEmpty(data))
                throw new ArgumentException();
            if (type.Equals("text/xml") || type.Equals("application/xml"))
            {
                Type = "XML";
            }
            else if(type.Equals("text/html"))
            {
                Type = "HTML";
            }
            else if (type.Equals("application/json") || type.Equals("application/octet-stream"))
            {
                Type = "JSON";
            }
            return new Request(filename,Type,data);
        }
```

Here is the caller graph for this function:


OSXJV.Classes.Request.Get Request
OSXJV.Server.OSXJVServer. GetFileData
OSXJV.Server.OSXJVServer. GetFormData

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().

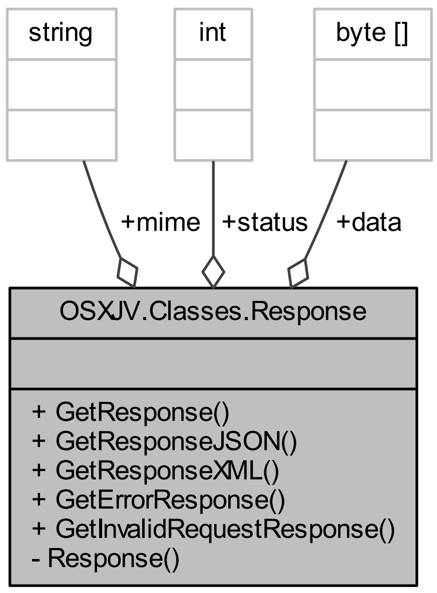
The documentation for this class was generated from the following file:

- WebServiceCSharp/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

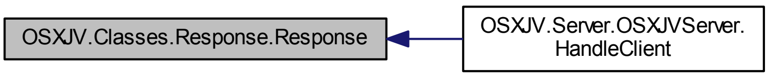
status	Status Code
mime	MIME type
buffer	Data

Definition at line 34 of file Response.cs.

Referenced by OSXJV.Server.OSXJVServer.HandleClient().

```
        {
            this.status = status;
            this.data = buffer;
            this.mime = mime;
        }
```

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

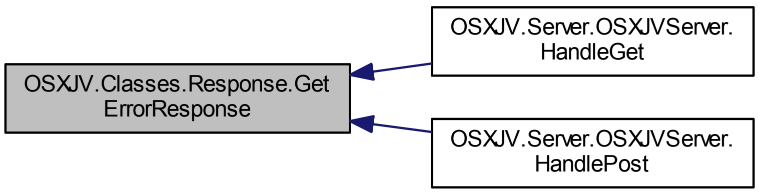
New response object

Definition at line 118 of file Response.cs.

Referenced by OSXJV.Server.OSXJVServer.HandleGet(), and OSXJV.Server.OSXJVServer.HandlePost().

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

Here is the caller graph for this function:


OSXJV.Classes.Response.Get
ErrorResponse
OSXJV.Server.OSXJVServer.
HandleGet
OSXJV.Server.OSXJVServer.
HandlePost

5.10.3.2 GetInvalidRequestResponse()

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

Returns an invalid response object

Returns

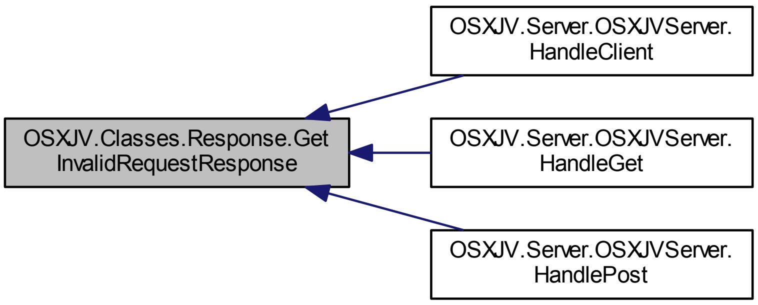
New response object

Definition at line 128 of file Response.cs.

Referenced by OSXJV.Server.OSXJVServer.HandleClient(), OSXJV.Server.OSXJVServer.HandleGet(), and OS←XJV.Server.OSXJVServer.HandlePost().

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

Here is the caller graph for this function:


OSXJV.Server.OSXJVServer.
HandleClient
OSXJV.Classes.Response.Get
InvalidRequestResponse
OSXJV.Server.OSXJVServer.
HandleGet
OSXJV.Server.OSXJVServer.
HandlePost

5.10.3.3 GetResponse()

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

A custom response object

Parameters

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

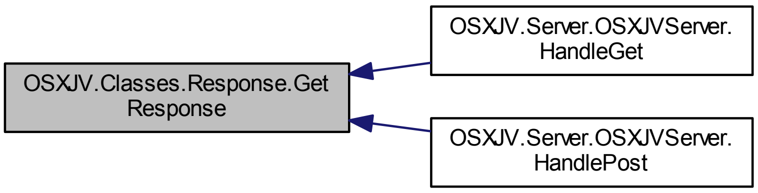
Returns

Definition at line 48 of file Response.cs.

Referenced by OSXJV.Server.OSXJVServer.HandleGet(), and OSXJV.Server.OSXJVServer.HandlePost().

```
        {
            if(string.IsNullOrEmpty(type))
                throw new ArgumentException("Type cannot be Null or empty");

            if (status.Equals(null))
                throw new ArgumentException("Status cannot be Null");
            else
               if (status == 0)
                throw new ArgumentException("Status cannot be 0");

            if (data == null)
                throw new ArgumentException("Data cannot be null");
            else
                if (data.Length == 0)
                throw new ArgumentException("No data, use invalid or error response");

            return new Response(status, type, data);
        }
```

Here is the caller graph for this function:


OSXJV.Classes.Response.Get
Response
OSXJV.Server.OSXJVServer.
HandleGet
OSXJV.Server.OSXJVServer.
HandlePost

5.10.3.4 GetResponseJSON()

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

Return an application/json response

Parameters

status	The HTTP Code to send back e.g. 200 for success
data	The data to send

Returns

New response object

Definition at line 74 of file Response.cs.

```
        {
            if (status.Equals(null))
                throw new ArgumentException("Status cannot be Null");
            else
                if (status == 0)
                    throw new ArgumentException("Status cannot be 0");

            if (data == null)
                throw new ArgumentException("Data cannot be null");
            else
                if (data.Length == 0)
                    throw new ArgumentException("No data, use invalid or error response");

            return new Response(status,"application/json", data);
        }
```

5.10.3.5 GetResponseXML()

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

Return an text/xml response

Parameters

status	The HTTP Code to send back e.g. 200 for success
data	The data to send

Returns

New response object

Definition at line 97 of file Response.cs.

```
        {
            if (status.Equals(null))
                throw new ArgumentException("Status cannot be Null");
            else
                if(status == 0)
                    throw new ArgumentException("Status cannot be 0");

            if (data == null)
                throw new ArgumentException("Data cannot be null");
            else
                if (data.Length == 0)
                    throw new ArgumentException("No data, use invalid or error response");

            return new Response(status, "text/xml", data);
        }
```

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().

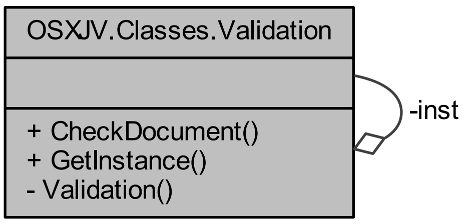
The documentation for this class was generated from the following file:

- WebServiceCSharp/OSXJVClasses/Response.cs

5.11 OSXJV.Classes.Validation Class Reference

Perform validation on document

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

Perform validation on document

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

data	Document contents
type	Type of document

Returns

True if valid, else false

Exceptions

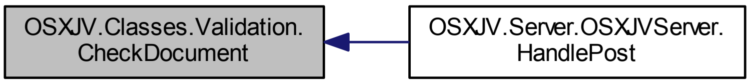
ArgumentException	Invalid data type or data and type cannot be null
XmlException	Invalid XML or HTML
JsonReaderException	Invalid JSON

Definition at line 44 of file Validation.cs.

Referenced by OSXJV.Server.OSXJVServer.HandlePost().

```
        {
            if(string.IsNullOrEmpty(data) || string.IsNullOrEmpty(type))
            {
                throw new ArgumentException("Data or Type cannot be Null");
            }

            if (type.Equals("XML") || type.Equals("HTML"))
            {
                XmlReaderSettings settings = new XmlReaderSettings();
                settings.DtdProcessing = DtdProcessing.Parse;
                settings.MaxCharactersFromEntities = 2048;
                using (XmlReader xr = XmlReader.Create(new StringReader(data),settings))
                {
                    try
                    {
                        while (xr.Read()) { }
                        return true;
                    }
                    catch (XmlException ex)
                    {
                        throw ex;
                    }
                }
            }
            else if(type.Equals("JSON"))
            {
                try
                {
                    JToken.Parse(data);
                    return true;
                }
                catch (JsonReaderException ex)
                {
                    throw new JsonReaderException(ex.Message);
                }
            }

            throw new ArgumentException("Invalid data or type");
        }
```

Here is the caller graph for this function:



5.11.3.2 GetInstance()

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

Gets the instance.

Returns

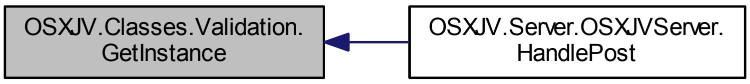
The instance.

Definition at line 28 of file Validation.cs.

Referenced by OSXJV.Server.OSXJVServer.HandlePost().

```
        {
            if (inst != null)
                return inst;
            else
                return (inst = new Validation ());
        }
```

Here is the caller graph for this function:



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:

- WebServiceCSharp/OSXJVClasses/Validation.cs

Chapter 6

File Documentation

6.1 WebServiceCSharp/OSXJVClasses/Attribute.cs File Reference

Classes

- class OSXJV.Classes.Attribute

Namespaces

- namespace OSXJV.Classes

6.2 Attribute.cs

```
namespace OSXJV.Classes
{
    public class Attribute
    {
        private string name;
        private string value;

        public string Name
        {
            get
            {
                return name;
            }

            set
            {
                name = value;
            }
        }
        public string Value
        {
            get
            {
                return value;
            }

            set
            {
                this.value = value;
            }
        }
    }
}
```

6.3 WebServiceCSharp/OSXJVClasses/CacheManager.cs File Reference

Classes

- class OSXJV.Classes.CacheManager

  Manages Saving an Retrieving Processed Documents

Namespaces

- namespace OSXJV.Classes

6.4 CacheManager.cs

```
using System;
using System.IO;

namespace OSXJV.Classes
{
    public class CacheManager
    {
        private static CacheManager Inst;

        private static string path = null;

        private CacheManager(string cachePath)
        {
            path = cachePath;
        }

        public static bool Setup(string path)
        {
            if (string.IsNullOrEmpty(path))
                throw new ArgumentException("Path cannot be empty");

            if (!Directory.Exists(string.Format(@"{0}",path)))
                throw new Exception("Path is not a valid cache directory");

            return (Inst = new CacheManager(path)) != null ? true : false;
        }

        public static CacheManager GetInstance()
        {
            if (Inst != null)
                return Inst;
            else
                throw new Exception("CacheManger has not been setup");
        }

        public string getFile(string ID)
        {
            if (string.IsNullOrEmpty(ID))
                throw new ArgumentException("ID cannot be null or empty");

            string filePath = path + "/" + ID.Replace("/","") + ".json";
            string output = "";

            using (StreamReader sr = new StreamReader(filePath))
            {
                output = sr.ReadToEnd();
            }

            if (!string.IsNullOrEmpty(output))
                return output;
            else
                throw new Exception("Error Reading From File");
        }

        public bool saveFile(string ID, string nodes)
        {
            if (string.IsNullOrEmpty(ID))
                throw new ArgumentException("ID cannot be null or empty");

            if (string.IsNullOrEmpty(nodes))
```

```
                throw new ArgumentException("Document cannot be null or empty");

            string filePath = path + "/" + ID + ".json";
            try
            {
                using (StreamWriter sw = new StreamWriter(filePath))
                {
                    sw.WriteLine(nodes);
                }
            }
            catch
            {
                throw new Exception("Failed to save file");
            }

            return true;
        }
        public static void Close()
        {
            if (Inst == null)
                throw new Exception("CacheManager Already Closed");
            else
            {
                path = null; //Clear static path
                Inst = null; //clear static instance
            }
        }

        public static void ManageCache()
        {
            if (path != null)
            {
                string[] files = Directory.GetFiles(path);
                foreach (string file in files)
                {
                    if (File.GetLastAccessTime(file) < DateTime.Now.AddHours(-6.0))
                        File.Delete(file);
                }
            }
            else
                throw new Exception("CacheManger not setup");
        }
    }
}
```

6.5 WebServiceCSharp/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

```
using System;
using System.IO;

namespace OSXJV.Classes
{
    public class Logger
    {
        private static Logger inst;
        private string location;

        private Logger(string location)
        {
            this.location = location;
        }

        public static bool Setup(string location)
        {
            if (string.IsNullOrEmpty(location))
                throw new ArgumentException("Location cannot be empty");

            if (!Directory.Exists(string.Format(@"{0}", location)))
                throw new Exception("Location is not a valid logger directory");

            return (inst = new Logger(location)) != null ? true:false;
        }

        public static Logger GetInstance()
        {
            if (inst != null)
                return inst;
            else
                throw new Exception("Logger has not been setup");
        }

        public void WriteError(string error)
        {
            try
            {
                if (!string.IsNullOrEmpty(error))
                {
                    string file = string.Format(@"{0}/Error-{1}.txt", location, DateTime.Now.ToString("
dd-MM-yy hh-MM-ss"));
                    StreamWriter sw = new StreamWriter(file);
                    sw.WriteLine(error);
                    sw.WriteLine();
                    sw.Close();
                }
            }
            catch (IOException e)
            {
                throw e;
            }
        }

        public static void Close()
        {
            if (inst == null)
                throw new Exception("Logger Already Closed");
            else
                inst = null;
        }
    }
}
```

6.7 WebServiceCSharp/OSXJVClasses/Node.cs File Reference

Classes

- class OSXJV.Classes.Node

  Contain Processed Document Information

Namespaces

- namespace OSXJV.Classes

6.8 Node.cs

```
using System.Collections.Generic;
using Newtonsoft.Json.Serialization;
using Newtonsoft.Json;

namespace OSXJV.Classes
```

```
{
    public class Node
    {
        private string name;
        private List<Attribute> attributes;
        private string value;
        private List<Node> children;
        private int number;
        private bool visited;
        private List<string> comments;

        public Node()
        {
            Attributes = new List<Attribute>();
            Children = new List<Node>();
            Comments = new List<string>();
            number = 0;
            visited = false;
        }

        public int Number
        {
            get
            {
                return number;
            }

            set
            {
                number = value;
            }
        }

        public string Name
        {
            get
            {
                return name;
            }

            set
            {
                name = value;
            }
        }

        [JsonProperty(NullValueHandling =NullValueHandling.Ignore)]
        public string Value
        {
            get
            {
                return value;
            }

            set
            {
                this.value = value;
            }
        }

        public List<string> Comments
        {
            get
            {
                return comments;
            }

            set
            {
                comments = value;
            }
        }

        [JsonProperty()]
        public List<Attribute> Attributes
        {
            get
            {
                return attributes;
            }

            set
            {
                attributes = value;
            }
        }

        [JsonProperty()]
        public List<Node> Children
        {
            get
            {
                return children;
            }

            set
            {
                children = value;
            }
        }

        [Newtonsoft.Json.JsonIgnore]
        public bool Visited
        {
            get
            {
                return visited;
            }

            set
            {
                visited = value;
            }
        }
    }
}
```

6.9 WebServiceCSharp/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

```
using Newtonsoft.Json.Linq;
using System;
using System.Collections.Generic;
using System.Threading;

namespace OSXJV.Classes
{
    public class Output
    {
        private int left = 100, top = 130;

        private Node nodes;

        private bool GotParent = false;

        private int Parent = 0;

        private List<Tuple<int, string>> cNodes = new List<Tuple<int, string>>();

        public Output(Node nodes)
        {
            if (nodes == null)
                throw new ArgumentException();
            this.nodes = nodes;
        }

        public JObject CreateGrid()
        {
            JObject obj = new JObject();
            obj.Add("text", nodes.Name);
            obj.Add("id", nodes.Number);
            obj.Add("state", new JObject(new JProperty("selected", true)));

            if(nodes.Children.Count > 0)
            {
                JArray array = new JArray();
                foreach (Node n2 in nodes.Children)
                {
                    array.Add(GridGetChidren(n2));
                }
                obj.Add("children", array);
            }
            return obj;
        }

        private JObject GridGetChidren(Node n)
        {
            JObject child = new JObject();
            child.Add("id", n.Number);
            child.Add("text", n.Name);

            if (n.Children.Count > 0)
            {
                JArray array = new JArray();
                foreach(Node n2 in n.Children)
                {
                    array.Add(GridGetChidren(n2));
                }
                child.Add("children", array);
            }
            return child;
        }
        public string CreateViewSingle(int node, int nodeStart = 0)
        {
            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'>";

            if (nodes.Number.Equals(node))
            {
                int count = 0;
                output += CreateNodeView(nodes, "node");


                foreach (Node n in nodes.Children)
                {
                    if(nodeStart > 0)
                    {
                        if (count != nodeStart)
                            continue;
                    }
                    count++;
                    output += CreateNodeView(n, "node-child"); //Child(Nodes) Thread

                    if ((count-nodeStart) == 200)
                    {
                        output += CreateExtraNode("node-child",count);
                        break;
                    }
                }
            }

            else
            {
                GetParent(nodes, node);
                string temp = "";
                if (GotParent)
                {
                    if (nodes.Number == Parent)
                    {
                        output += CreateNodeView(nodes, "node-parent");
                    }
                }
                foreach (Node n2 in nodes.Children)
                {
                    if (GotParent)
                    {
                        if (n2.Number == Parent)
                        {
                            output += CreateNodeView(n2, "node-parent");
                        }
                    }
                    temp += CheckChildren(n2, node);
                }
                if (!string.IsNullOrEmpty(temp))
```

```
                    output += temp;
                }
                output += "</div></div>"; //Close out divs
                return output;
            }

            private string CreateExtraNode(string type,int id)
            {
                string node = "";

                if (type == "node")
                {
                    if (GotParent)
                    {
                        left = left + 400;
                    }
                }
                if (type == "node-child")
                {
                    left = left + 400;
                }

                node += "<div class='node-child type ui-draggable ui-selectee' style='left:" + left + "px; top:
" + top + "px;margin-bottom:50px;'>";
                node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
)'>Show Lower</button></span></div>";
                node += "</div></div>";
                return node;
            }

            private string CreatePreviousNode(string type, int leftVal, int topVal, int id)
            {
                string node = "";

                if (type == "node")
                {
                    if (GotParent)
                    {
                        leftVal = leftVal + 400;
                    }
                }
                if (type == "node-child")
                {
                    leftVal = leftVal + 400;
                }
                node += "<div class='node-child type ui-draggable ui-selectee' style='left:" + leftVal + "px;
 top:" + topVal + "px;'>";
                node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
)'>Show Higher</button></span></div>";
                node += "</div></div>";
                return node;
            }

            private string CreateExtraNode(string type, int leftVal, int topVal,int id)
            {
                string node = "";

                if (type == "node")
                {
                    if (GotParent)
                    {
                        leftVal = leftVal + 400;
                    }
                }
                if (type == "node-child")
                {
                    leftVal = leftVal + 400;
                }
                node += "<div class='node-child type ui-draggable ui-selectee' style='left:" + leftVal + "px;
 top:" + topVal + "px;margin-bottom:50px;'>";
                node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
)'>Show Lower</button></span></div>";
                node += "</div></div>";
                return node;
            }

            private void CreateNodeChildViewsParallel(List<Node> job,int start,
bool showHigher,int next,int previous)
            {
                int threadID = int.Parse(Thread.CurrentThread.Name);
                string type = "node-child";
                string output = "";

                if(start == 0 && showHigher)
                {
                    output += CreatePreviousNode(type, left, top, previous);
                }
                bool hadCommentsPrev = false;
                int numCommentsPrevious = 0;

                foreach(Node n in job)
                {
                    int extra = showHigher ? 130 * (start +1) : 130 * start;

                    if (hadCommentsPrev)
                        extra += (numCommentsPrevious * 25);

                    if (n.Comments.Count > 0)
                    {
                        hadCommentsPrev = true;
                        numCommentsPrevious = n.Comments.Count;
                    }
                    else
                        hadCommentsPrev = false;

                    output += CreateNodeView(n, type,left,top + extra);
                    start++;
                    if (start == 200)
                    {
                        output += CreateExtraNode(type, left, top + extra + 130,next);
                        break;
                    }

                }

                cNodes.Add(new Tuple<int, string>(threadID, output));
            }

            public string CreateView(int node = 1,int pCount = 4,int nodeStart = 0) //Setting
     Defaults
            {

                List<Thread> threadList = new List<Thread>();

                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'>";
                if (nodes.Number.Equals(node))
                {
                    int childCount = 0;

                    if (nodes.Children.Count < 200)
                        childCount = nodes.Children.Count;
                    else
                    {
                        childCount = 200;
                    }

                    if(childCount < pCount * 2)
                    {
                        output += CreateNodeView(nodes, "node", left, top);
                        foreach(Node n2 in nodes.Children)
                        {
                            output += CreateNodeView(n2, "node-child");
                        }
                    }
                    else
                    {
                        int spread = (int)Math.Ceiling((double)childCount / (double)pCount);

                        output += CreateNodeView(nodes, "node",left,top);  //Parent(Node) Thread

                        for (int i = 0; i < pCount; i++)
                        {
                            int neg = 0;
                            if ((spread * (i + 1)) > childCount)
                            {
                                neg = childCount - (spread * (i + 1));
                            }
                            int start = (spread * i) ;
                            int rangeStart = (spread * i) + nodeStart;
                            bool showHigher = nodeStart != 0 ? true : false;

                            List<Node> NodesToProcess = nodes.Children.GetRange(rangeStart, spread +
    neg);
                            Thread threadJob = new Thread(() => CreateNodeChildViewsParallel(NodesToProcess,
    start, showHigher, childCount + nodeStart, nodeStart - childCount));
                            threadJob.Name = i.ToString();
                            threadJob.Start();
                            threadList.Add(threadJob);
                        }
                        foreach(Thread t in threadList)
                        {
                            t.Join();
                        }

                        cNodes.Sort((x, y) => x.Item1.CompareTo(y.Item1));

                        foreach(Tuple<int,string> tup in cNodes)
                        {
                            output += tup.Item2;
                        }
                    }
                }
                else
                {
                    GetParent(nodes, node);
                    string temp = "";
                    if (GotParent)
                    {
                        if (nodes.Number == Parent)
                        {
                            output += CreateNodeView(nodes, "node-parent");
                        }
                    }
                    bool found =false;
                    foreach (Node n2 in nodes.Children)
                    {
                        if (GotParent)
                        {
                            if (n2.Number == Parent)
                            {
                                output += CreateNodeView(n2, "node-parent");
                            }
                        }
                        temp += CheckChildren(n2, node,pCount,nodeStart,ref found);
                        if (found)
                            break;
                    }
                    if (!string.IsNullOrEmpty(temp))
                        output += temp;
                }
                output += "</div></div>";
                return output;
            }

            private string CheckChildren(Node n, int number)
            {
                string output = "";
                if (CheckNodeNumber(n, number))
                {
                    int count = 0;
                    output += CreateNodeView(n, "node");
                    foreach (Node n2 in n.Children)
                    {
                        count++;
                        output += CreateNodeView(n2, "node-child");

                    }
                }
                else if (n.Children.Count > 0)
                {
                    foreach (Node n2 in n.Children)
                    {
                        if (GotParent)
                        {
                            if (n2.Number == Parent)
                            {
                                output += CreateNodeView(n2, "node-parent");
                            }
                        }
                        output += CheckChildren(n2, number);
                    }
                }

                return output;
            }

            private string CheckChildren(Node n, int number,int pCount, int nodeStart, ref
    bool found)
            {
                string output = "";
                if (CheckNodeNumber(n, number))
                {
                    found = true;
                    List<Thread> threadList = new List<Thread>();

                    int count = 0;
                    output += CreateNodeView(n, "node");
                    count++;
                    //output += CreateNodeView(n2, "node-child");
                    int childCount = 0;

                    if (n.Children.Count < 200)
                        childCount = n.Children.Count;
                    else
                    {
                        childCount = 200;
                    }
                    if (childCount < pCount * 2)
                    {
                        foreach(Node n2 in n.Children)
                        {
                            output += CreateNodeView(n2, "node-child");
                        }
                    }
                    else
                    {
                        int spread = (int)Math.Ceiling((double)childCount / (double)pCount);

                        if (childCount > 0)
                        {
                            for (int i = 0; i < pCount; i++)
                            {
                                int neg = 0;
                                if ((spread * (i + 1)) > childCount)
                                {
                                    neg = childCount - (spread * (i + 1));
                                }
                                int start = (spread * i);
                                int rangeStart = (spread * i) + nodeStart;
                                bool showHigher = nodeStart != 0 ? true : false;

                                List<Node> NodesToProcess = n.Children.GetRange(rangeStart, spread + neg);

                                if (NodesToProcess.Count > 0)
                                {
                                    Thread threadJob = new Thread(() => CreateNodeChildViewsParallel(NodesToProcess, start, showHigher, childCount + nodeStart, nodeStart - childCount));
                                    threadJob.Name = i.ToString();
                                    threadJob.Start();
                                    threadList.Add(threadJob);
                                }
                            }
                            foreach (Thread t in threadList)
                            {
                                t.Join();
                            }
                            cNodes.Sort((x, y) => x.Item1.CompareTo(y.Item1));

                            foreach (Tuple<int, string> tup in cNodes)
                            {
                                output += tup.Item2;
                            }
                        }
                    }
                }
                else if (n.Children.Count > 0)
                {
                    foreach (Node n2 in n.Children)
                    {
                        if (GotParent)
                        {
                            if (n2.Number == Parent)
                            {
                                output += CreateNodeView(n2, "node-parent");
                            }
                        }
                        output += CheckChildren(n2, number,pCount,nodeStart,ref found);
                    }
                }

                return output;
            }

            private void GetParent(Node node,int number)
            {
                if(!CheckNodeNumber(node,number))
                {
                    foreach(Node n in node.Children)
                    {
                        if(CheckNodeNumber(n, number))
                        {
                            Parent = node.Number;
                            GotParent = true;
                        }
                        else
                        {
                            GetParent(n, number);
                        }
                    }
                }
            }

            private string CreateNodeView(Node n, string type,int leftVal,int topVal)
            {
                string node = "";

                if(type == "node")
                {
                    if(GotParent)
                    {
                        leftVal = leftVal + 400;
                    }
                }
                if(type == "node-child")
                {
                    leftVal = leftVal + 400;
                }
                node += "<div id='" + n.Number + "'class='" + type + " type ui-draggable ui-selectee'
     style='left:" + leftVal + "px; top:" + topVal + "px;'>";
                node += "<div class='head'><span><button class='nameBtn' onclick='GetNode("+n.
    Number+")'>" + n.Name + "</button></span></div>";
                if (!string.IsNullOrEmpty(n.Value))
                {
                    node += string.Format("<div class='blockR'><p>Value</p></div><div
     class=comment><span>{0}</span></div>", n.Value);
                }
                if(n.Comments.Count >0)
                {
                    node += "<div><p class='text-center'>Comments</p></div>";

                    foreach(string com in n.Comments)
                    {
                        node += "<div class='comment'>" + com + "</div>";
                    }
                }
                if (n.Attributes.Count > 0)
                {
                    node += "<div class='attribute'><div class='aHeader'><p><button><i class='fa
     fa-plus'></i></button>Attributes</p></div><div class='options'>";
                    foreach (Attribute a in n.Attributes)
                    {
                        node += string.Format("<div class='blockR'><p>{0}</p></div><div
     class='comment'><p>{1}</p></div>", a.Name, a.Value);
                    }
                    node += "</div>";
                }
                node += "</div></div>";
                return node;
            }

            private string CreateNodeView(Node n, string type)
            {
                string node = "";
                int leftVal = left;
                if (type == "node")
                {
                    if (GotParent)
                    {
                        left = left + 400;
                        leftVal = left;
                    }
                }
                if (type == "node-child")
                {
                    leftVal = leftVal + 400;
                }
                node += "<div id='" + n.Number + "'class='" + type + " type ui-draggable ui-selectee'
     style='left:" + leftVal + "px; top:" + top + "px;'>";
                node += "<div class='head'><span><button class='nameBtn' onclick='GetNode(" + n.
    Number + ")'>" + n.Name + "</button></span></div>";
                if (!string.IsNullOrEmpty(n.Value))
                {
                    node += string.Format("<div class='blockR'><p>Value</p></div><div
     class=comment><span>{0}</span></div>", n.Value);
                }
                if (n.Comments.Count > 0)
                {
                    node += "<div><p class='text-center'>Comments</p></div>";

                    foreach (string com in n.Comments)
                    {
                        node += "<div class='comment'>" + com + "</div>";
                    }
                }
                if (n.Attributes.Count > 0)
                {
                    node += "<div class=’attribute’><div class=’aHeader’><p><button><i class=’fa fa-plus’></i></button>Attributes</p></div><div class=’options’>";
                    foreach (Attribute a in n.Attributes)
                    {
                        node += string.Format("<div class=’blockR’><p>{0}</p></div><div class=’comment’><p>{1}</p></div>", a.Name, a.Value);
                    }
                    node += "</div>";
                }
                node += "</div></div>";

                if (type == "node-child")
                {
                    top = top + 130;
                }
                return node;
            }

            private bool CheckNodeNumber(Node n, int number)
            {
                return n.Number.Equals(number);
            }
        }
    }
```

6.11 WebServiceCSharp/OSXJVClasses/ProcessDocument.cs File Reference

Classes

- class OSXJV.Classes.ProcessDocument

  Class the Processes the document

Namespaces

- namespace OSXJV.Classes

6.12 ProcessDocument.cs

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Threading;
using System.Xml.Linq;

namespace OSXJV.Classes
{

    public class ProcessDocument
    {
        private XDocument document;

        private Node node = new Node();

        private List<Tuple<Node, int>> ProcessedElements = new List<Tuple<Node, int>>();

        private List<Thread> ThreadList = new List<Thread>();

        private string type;
        private Thread th;

        int count;

        private ProcessDocument(XDocument doc, string type)
        {
            document = doc;
            this.type = type;
        }

        private void ProcessComment(XComment e, Node node)
        {
            string s = "";
            s = Regex.Replace(e.Value, @"[^\w\s\.@-]", "");
            node.Comments.Add(s);
        }

        public static ProcessDocument GetProcess(string data, string type)
        {
            if (string.IsNullOrEmpty(data) || string.IsNullOrEmpty(type))
            {
                throw new ArgumentException();
            }
            try
            {
                XDocument doc = null;
                doc = Prepare(data, type);
                return new ProcessDocument(doc, type);
            }
            catch (System.Xml.XmlException e)
            {
                throw e;
            }
        }

        private void ProcessText(XText e, Node n)
        {
            n.Value = e.Value;
        }

        private static XDocument Prepare(string data, string type)
        {

            if (type.Equals("JSON"))
                return new XDocument(JsonConvert.DeserializeXNode(data, "Root", false).Root.FirstNode);
            else if (type.Equals("XML") || type.Equals("HTML"))
                return XDocument.Parse(data);

            return null;
        }


        public Node Process()
        {
            if (document.Nodes() != null)
            {
                foreach (XNode n in document.Nodes())
                {
                    switch (n.NodeType)
                    {
                        case System.Xml.XmlNodeType.Element:
                            count++;
                            ProcessElement(XElement.Parse(n.ToString()), node);
                            break;
                        case System.Xml.XmlNodeType.Comment:
                            ProcessComment(n as XComment, node);
                            break;
                        case System.Xml.XmlNodeType.Text:
                            ProcessText(n as XText, node);
                            break;
                        case System.Xml.XmlNodeType.Notation:
                            break;
                        case System.Xml.XmlNodeType.EndElement:
                            break;
                        default:
                            break;
                    }
                }
            }
            //SortArray(ref node);
            document = null;
            return node;
        }

        private Node ProcessElement(XElement e, Node node)
        {
            if (node.Number == 0)
            {
                node.Number = count;
            }
            if (!node.Visited)
            {

                node.Name = e.Name.LocalName;
                foreach (XAttribute ax in e.Attributes())
                {
                    if (ax.Name == "id")
                    {
                        node.Name = node.Name + " #" + ax.Value;
                    }

                    if (type == "HTML")
                    {
                        if (ax.IsNamespaceDeclaration)
                            continue;
                    }
                    Attribute att = new Attribute();
                    att.Name = ax.Name.LocalName;
                    att.Value = ax.Value;
                    node.Attributes.Add(att);
                }
            }

            if (e.Nodes() != null)
            {
                foreach (XNode n in e.Nodes())
                {
                    switch (n.NodeType)
                    {
                        case System.Xml.XmlNodeType.EndElement:
                            break;
                        case System.Xml.XmlNodeType.Element:
                            count++;
                            Node n2 = new Node();
                            node.Children.Add(ProcessElement(XElement.Parse(n.ToString()), n2));
                            break;
                        case System.Xml.XmlNodeType.Comment:
                            ProcessComment(n as XComment, node);
                            break;
                        case System.Xml.XmlNodeType.Text:
                            ProcessText(n as XText, node);
                            break;
                        case System.Xml.XmlNodeType.Notation:
                            break;

                        default:
                            break;
                    }
                }
            }
            node.Visited = true;
            return node;
        }

        private Node ProcessElement(XElement e, Node node, ref int nodeNumber)
        {
            if (!node.Visited)
            {
                if (node.Number == 0)
                {
                    node.Number = nodeNumber;
                }
                if (!node.Visited)
                {

                    node.Name = e.Name.LocalName;
                    foreach (XAttribute ax in e.Attributes())
                    {
                        if (ax.Name == "id")
                        {
                            node.Name = node.Name + " #" + ax.Value;
                        }

                        if (type == "HTML")
                        {
                            if (ax.IsNamespaceDeclaration)
                                continue;
                        }
                        Attribute att = new Attribute();
                        att.Name = ax.Name.LocalName;
                        att.Value = ax.Value;
                        node.Attributes.Add(att);
                    }
                }

                if (e.Nodes() != null)
                {
                    foreach (XNode n in e.Nodes())
                    {
                        switch (n.NodeType)
                        {
                            case System.Xml.XmlNodeType.EndElement:
                                break;
                            case System.Xml.XmlNodeType.Element:
                                nodeNumber++;
                                Node n2 = new Node();
                                node.Children.Add(ProcessElement(XElement.Parse(n.ToString()), n2, ref nodeNumber));
                                break;
                            case System.Xml.XmlNodeType.Comment:
                                ProcessComment(n as XComment, node);
                                break;
                            case System.Xml.XmlNodeType.Text:
                                ProcessText(n as XText, node);
                                break;
                            case System.Xml.XmlNodeType.Notation:
                                break;

                            default:
                                break;
                        }
                    }
                }
                node.Visited = true;
            }
            return node;
        }

        private Node ProcessRoot(XElement e, Node node)
        {
            node.Number = 1;

            if (!node.Visited)
            {

                node.Name = e.Name.LocalName;
                foreach (XAttribute ax in e.Attributes())
                {
                    if (ax.Name == "id")
                    {
                        node.Name = node.Name + " #" + ax.Value;
                    }

                    if (type == "HTML")
                    {
                        if (ax.IsNamespaceDeclaration)
                            continue;
                    }
                    Attribute att = new Attribute();
                    att.Name = ax.Name.LocalName;
                    att.Value = ax.Value;
                    node.Attributes.Add(att);
                }
            }
            node.Visited = true;
            return node;
        }

        private void ProcessDocumentParallelInit(XDocument doc,int start)
        {
            int nodeNum = start;

            Node node = new Node();
            if (doc.Root.Nodes() != null)
            {
                List<XNode> list = doc.Root.Nodes().ToList();
                foreach (XNode n in doc.Root.Nodes())
                {
                    switch (n.NodeType)
                    {
                        case System.Xml.XmlNodeType.Element:
                            nodeNum++;
                            Node n2 = new Node();
                            node.Children.Add(ProcessElement(XElement.Parse(n.ToString()), n2, ref nodeNum));
                            break;
                        case System.Xml.XmlNodeType.Comment:
                            ProcessComment(n as XComment, node);
                            break;
                        case System.Xml.XmlNodeType.Text:
                            ProcessText(n as XText, node);
                            break;
                        case System.Xml.XmlNodeType.Notation:
                            break;
                        case System.Xml.XmlNodeType.EndElement:
                            break;
                        default:
                            break;
                    }
                }
            }
            document = null;
            ProcessedElements.Add(new Tuple<Node, int>(node, start));
        }

        public Node ProcessParallel(int pCount = 4)
        {
            node = ProcessRoot(document.Root, node);

            int nodeCount = document.Root.Nodes().Count();

            if(nodeCount <= pCount)
            {
                return Process();
            }
            else if (nodeCount > pCount)
            {

                List<XNode> List = document.Root.Nodes().ToList();
                int spread = 0;

                spread = (int)Math.Ceiling((double)nodeCount / (double)pCount);

                int totalNodes = 1;

                for (int i = 0; i < pCount; i++)
                {
                    int neg = 0;
                    int start = totalNodes;
                    if ((spread * (i+1)) > nodeCount)
                    {
                        neg = nodeCount - (spread * (i + 1));
                    }

                    List<XNode> list = List.GetRange((spread * i), spread + neg);
                    XElement root = new XElement("Root", list);
                    XDocument doc = new XDocument(root);

                    (th = new Thread(() => ProcessDocumentParallelInit(doc, start))).Start();

                    ThreadList.Add(th); //Add to Threads list to keep recored of threads running
                    totalNodes += root.Descendants().Count(); //Increment start position.
                }
                document = null;
                foreach (Thread t in ThreadList)
                {
                    t.Join(); //Wait for threads to join
                }

                ProcessedElements.Sort((x, y) => x.Item2.CompareTo(y.Item2)); //Sort List by start index so
      they are in order.

                foreach(Tuple<Node,int> tup in ProcessedElements)
                {
                    foreach(Node n in tup.Item1.Children)
                    {
                        node.Children.Add(n);
                    }
                }
            }
            return node;
        }
    }
}
```

6.13 WebServiceCSharp/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

```
using System;

namespace OSXJV.Classes
{
    public class Request
    {
        private string filename;

        private string type;

        private string data;

        private Request(string filename, string type, string data)
        {
            this.filename = filename;
            this.type = type;
            this.data = data;
        }

        public static Request GetRequest(string filename, string type, string data)
        {
            string Type = "";
            if (string.IsNullOrEmpty(filename) || string.IsNullOrEmpty(type) || string.IsNullOrEmpty(data))
                throw new ArgumentException();
            if (type.Equals("text/xml") || type.Equals("application/xml"))
            {
                Type = "XML";
            }
            else if(type.Equals("text/html"))
            {
                Type = "HTML";
            }
            else if (type.Equals("application/json") || type.Equals("application/octet-stream"))
            {
                Type = "JSON";
            }
            return new Request(filename,Type,data);
        }

        public string Filename
        {
            get
            {
                return filename;
            }

            set
            {
                filename = value;
            }
        }

        public string Type
        {
            get
            {
                return type;
            }

            set
            {
                type = value;
            }
        }

        public string Data
        {
            get
            {
                return data;
            }

            set
            {
                data = value;
            }
        }
    }
}
```

6.15 WebServiceCSharp/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

```
using System;
using System.Text;

namespace OSXJV.Classes
{
    public class Response
    {
        public byte[] data = null;

        public int status;

        public string mime;
        //static string format = "yyyy-MM-dd HH:mm:ss";


        private Response(int status,string mime,byte[] buffer)
        {
            this.status = status;
            this.data = buffer;
            this.mime = mime;
        }

        public static Response GetResponse(int status,string type,byte[] data)
        {
            if(string.IsNullOrEmpty(type))
                throw new ArgumentException("Type cannot be Null or empty");

            if (status.Equals(null))
                throw new ArgumentException("Status cannot be Null");
            else
               if (status == 0)
                throw new ArgumentException("Status cannot be 0");

            if (data == null)
                throw new ArgumentException("Data cannot be null");
            else
                if (data.Length == 0)
                throw new ArgumentException("No data, use invalid or error response");

            return new Response(status, type, data);
        }

        public static Response GetResponseJSON(int status,byte[] data)
        {
            if (status.Equals(null))
                throw new ArgumentException("Status cannot be Null");
            else
                if (status == 0)
                    throw new ArgumentException("Status cannot be 0");

            if (data == null)
                throw new ArgumentException("Data cannot be null");
            else
                if (data.Length == 0)
                    throw new ArgumentException("No data, use invalid or error response");

            return new Response(status,"application/json", data);
        }

        public static Response GetResponseXML(int status, byte[] data)
        {
            if (status.Equals(null))
                throw new ArgumentException("Status cannot be Null");
            else
                if(status == 0)
                    throw new ArgumentException("Status cannot be 0");

            if (data == null)
                throw new ArgumentException("Data cannot be null");
            else
                if (data.Length == 0)
                    throw new ArgumentException("No data, use invalid or error response");

            return new Response(status, "text/xml", data);
        }

        public static Response GetErrorResponse(string message)
        {
            byte[] res = Encoding.UTF8.GetBytes(message);
            return new Response(400, "text/html", res);
        }

        public static Response GetInvalidRequestResponse()
        {
            return new Response(405, "text/html", new byte[0]);
        }
    }
}
```

6.17 WebServiceCSharp/OSXJVClasses/Validation.cs File Reference

Classes

- class OSXJV.Classes.Validation

  Perform validation on document

Namespaces

- namespace OSXJV.Classes

6.18 Validation.cs

```
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.IO;
using System.Xml;

namespace OSXJV.Classes
{
    public class Validation
    {
        private static Validation inst;

        private Validation(){}

        public static Validation GetInstance()
        {
            if (inst != null)
                return inst;
            else
                return (inst = new Validation ());
        }
        public bool CheckDocument(string data, string type)
        {
            if(string.IsNullOrEmpty(data) || string.IsNullOrEmpty(type))
            {
                throw new ArgumentException("Data or Type cannot be Null");
            }

            if (type.Equals("XML") || type.Equals("HTML"))
            {
                XmlReaderSettings settings = new XmlReaderSettings();
                settings.DtdProcessing = DtdProcessing.Parse;
                settings.MaxCharactersFromEntities = 2048;
                using (XmlReader xr = XmlReader.Create(new StringReader(data),settings))
                {
                    try
                    {
                        while (xr.Read()) { }
                        return true;
                    }
                    catch (XmlException ex)
                    {
                        throw ex;
                    }
                }
            }
            else if(type.Equals("JSON"))
            {
                try
                {
                    JToken.Parse(data);
                    return true;
                }
                catch (JsonReaderException ex)
                {
                    throw new JsonReaderException(ex.Message);
                }
            }

            throw new ArgumentException("Invalid data or type");
        }
    }
}
```

6.19 WebServiceCSharp/OSXJVServer.cs File Reference

Classes

- class OSXJV.Server.OSXJVServer

  HTTPServer that process the incoming requests.

Namespaces

- namespace OSXJV.Server

6.20 OSXJVServer.cs

```
using System;
using System.Text;
using System.Net;
using System.Threading;
using System.IO;
using HttpMultipartParser;
using Newtonsoft.Json.Linq;
using Newtonsoft.Json;
using OSXJV.Classes;

namespace OSXJV.Server
{
    public class OSXJVServer
    {
        private int port = 8082;

        public static bool running = false; //sets if the server is currently running

        private HttpListener listener;

        private Thread serverThread = null;

        private Thread cacheThread = null;

        public OSXJVServer()
        {
            listener = new HttpListener();
            listener.Prefixes.Add("http://localhost:" + port + "/"); //change if need be
        }

        public bool Start(string cachePath, string loggerPath)
        {
            bool success = false;

            success = CacheManager.Setup(cachePath);
            success = Logger.Setup(loggerPath);

            serverThread = new Thread(new ThreadStart(Run)); //Server thread
            cacheThread = new Thread(new ThreadStart(ManageCache)); //Cache manage thread
            try
            {
                serverThread.Start();
                cacheThread.Start();
            }
            catch(Exception e)
            {
                throw e;
            }

            success = cacheThread.IsAlive;
            success = serverThread.IsAlive;

            return success;
        }

        public bool Stop()
        {
            if (listener != null)
                if (listener.IsListening)
                    listener.Abort();


            if (serverThread != null)
            {
                serverThread.Join();
                serverThread = null;
            }

            return serverThread == null ?true:false;
        }
        public void Run()
        {
            running = true;
            listener.Start();


            while(listener.IsListening)
            {

                Console.WriteLine("Waiting");

                //Wait for Listener
                IAsyncResult result = listener.BeginGetContext(new AsyncCallback(ListenerCallback), listener);
                result.AsyncWaitHandle.WaitOne();

                if (result.CompletedSynchronously)
                    Console.WriteLine("Completed Synchronously");
            }
        }

        //Asyncronous Handler
        private void ListenerCallback(IAsyncResult result)
        {
            HttpListener listener = (HttpListener)result.AsyncState;
            HttpListenerContext context = listener.EndGetContext(result);
            try
            {
                HandleClient(context);
            }
            catch (Exception e)
            {
                Logger.GetInstance().WriteError(e.Message);
                context.Response.StatusCode = 500;
                context.Response.Close();
            }

        }

        //Handles the client request
        private void HandleClient(HttpListenerContext c)
        {
            switch(c.Request.HttpMethod)
            {
                case "POST":
                    Post(HandlePost(c.Request),c.Response);
                    break;
                case "GET":
                    Post(HandleGet(c.Request), c.Response);
                    break;
                case "OPTIONS":
                    HandleOptions(c.Response);
                    c.Response.Close();
                    break;
                default:
                    Post(Response.GetInvalidRequestResponse(), c.
      Response);
                    break;
            }
        }

        private void HandleOptions(HttpListenerResponse response)
        {
            response.AddHeader("Access-Control-Allow-Headers", "Content-Type, Accept, X-Requested-With");
            response.AddHeader("Access-Control-Allow-Methods", "POST");
            response.AddHeader("Access-Control-Allow-Methods", "GET");
            response.AddHeader("Access-Control-Max-Age", "1728000");
            response.AppendHeader("Access-Control-Allow-Origin", "*");
        }

        public Request GetFormData(Stream input)
        {
            string request = "";
            MultipartFormDataParser parser = new MultipartFormDataParser(input);
            if (parser.Files.Count > 0)
            {
                using (StreamReader ms = new StreamReader(parser.Files[0].Data))
                {
                    request = ms.ReadToEnd();
                }
            }
            else
            {
                throw new InvalidOperationException();
            }
            return Request.GetRequest(parser.Files[0].FileName, parser.Files[0].
      ContentType, request);
        }

        private Request GetFileData(Stream input,string type)
        {
            string request = "";
            using (StreamReader ms = new StreamReader(input))
            {
                request = ms.ReadToEnd();
            }
            string filename = "temp";

            if (type == "text/xml")
                filename += ".xml";
            else if(type == "application/json")
                filename += ".json";
            else
                filename += ".html";

            return Request.GetRequest(filename,type, request);
        }

        private Response HandlePost(HttpListenerRequest req)
        {

            JObject eRes = new JObject();

            if (SegmentNormalize(req.RawUrl).Equals("Process"))
            {
                if (req.HasEntityBody)
                {


                    Request r = null;
                    try
                    {
                        r = GetData(req);
                        if (r == null)
                            return Response.GetInvalidRequestResponse();
                    }
                    catch
                    {
                       return Response.GetInvalidRequestResponse();
                    }



                    try
                    {
                        Validation.GetInstance().CheckDocument(r.Data, r.Type);
                    }
                    catch (Exception e)
                    {
                        eRes.Add("Error", e.Message);
                        return Response.GetErrorResponse(eRes.ToString());
                    }

                    string id = Guid.NewGuid().ToString();
                    ProcessDocument pro = ProcessDocument.GetProcess(r.Data, r.Type);
                    Node n = pro.ProcessParallel();
                    Output o = new Output(n); //new output object
                    try
                    {
                        CacheManager.GetInstance().saveFile(id, JsonConvert.SerializeObject(n));
                        JObject response = new JObject();

                        n = null; //remove node as its completed;

                        response.Add("filename", id);
                        response.Add("grid", o.CreateGrid());
                        response.Add("view", o.CreateView());



                        byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
                        return Response.GetResponse(200, "application/json", bytes);
                    }
                    catch (Exception e)
                    {
                        Logger.GetInstance().WriteError(e.Message);
                        eRes.Add("Error", "Error Creating Response");
                        return Response.GetErrorResponse(eRes.ToString());
                    }

                }
                eRes.Add("Error", "No File Recieved By Server");
                return Response.GetErrorResponse(eRes.ToString());
            }
            else if (req.RawUrl.Equals("/Output"))
            {
                return Response.GetInvalidRequestResponse();
            }
            else
                return Response.GetInvalidRequestResponse();
        }

        private Response HandleGet(HttpListenerRequest req)
        {
            if (SegmentNormalize(req.Url.Segments[1]).Equals("Process"))
            {
                if (req.Url.Segments.Length == 4)
                {

                    Node cached;
                    try
                    {
                        cached = JsonConvert.DeserializeObject<Node>(CacheManager.GetInstance().getFile(req.Url.Segments[2]));
                    }
                    catch (Exception e)
                    {
                        Logger.GetInstance().WriteError(e.Message);
                        JObject eRes = new JObject();
                        eRes.Add("Error", "Error Creating Response");
                        return Response.GetErrorResponse(eRes.ToString());
                    }
                    Output o = new Output(cached);
                    JObject response = new JObject();
                    response.Add("view", o.CreateView(int.Parse(req.Url.Segments[3])));
                    byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
                    return Response.GetResponse(200, "application/json", bytes);

                }
                else if (req.Url.Segments.Length == 5)
                {

                    Node cached;
                    try
                    {
                        cached = JsonConvert.DeserializeObject<Node>(CacheManager.GetInstance().getFile(req.Url.Segments[2]));
                    }
                    catch (Exception e)
                    {
                        Logger.GetInstance().WriteError(e.Message);
                        JObject eRes = new JObject();
                        eRes.Add("Error", "Error Creating Response");
                        return Response.GetErrorResponse(eRes.ToString());
                    }
                    Output o = new Output(cached);
                    JObject response = new JObject();
                    response.Add("view", o.CreateView(int.Parse(SegmentNormalize(req.Url.Segments[3])), 4, int.Parse(SegmentNormalize(req.Url.Segments[4]))));
                    byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
                    return Response.GetResponse(200, "application/json", bytes);

                }
                else
                    return Response.GetInvalidRequestResponse();
            }
            //If it got here its an invalid response.
            return Response.GetInvalidRequestResponse();
        }

        private void SaveFile(string id, Node nodes)
        {
            if(nodes == null || string.IsNullOrEmpty(id))
            {
                throw new ArgumentException();
            }

            try
            {
                CacheManager.GetInstance().saveFile(id, JsonConvert.SerializeObject(nodes));
            }
            catch (Exception e)
            {
                Logger.GetInstance().WriteError(e.Message);
            }
        }

        private void Post(Response res,HttpListenerResponse stream)
        {
            if (res == null || stream == null)
                throw new ArgumentException("Response or Client Stream cannot be NULL");

            HandleOptions(stream);
            stream.ProtocolVersion = new Version(1, 1);
            stream.StatusCode = res.status;
            stream.ContentType = res.mime;
            stream.ContentLength64 = res.data.Length;
            stream.OutputStream.Write(res.data, 0, res.data.Length);
            stream.Close();
        }

        private Request GetData(HttpListenerRequest req)
        {
            Request r = null;

            if (req.ContentType.Contains("application/x-www-form-urlencoded"))
            {
                r = GetFormData(req.InputStream);
            }
            else if (req.ContentType.Contains("application/json") || req.ContentType.Contains("application/oclet-stream"))
            {
                r = GetFileData(req.InputStream, "application/json");
            }
            else if (req.ContentType.Contains("application/xml") || req.ContentType.Contains("text/xml"))
            {
                r = GetFileData(req.InputStream, "text/xml");
            }
            return r;
        }

        private string SegmentNormalize(string input)
        {
            return input.Replace("/", "");
        }
        private void ManageCache()
        {
            while (true)
            {
                Thread.Sleep(3600000);
                try
                {
                    CacheManager.ManageCache();
                }
                catch (Exception e)
                {
                    try
                    {
                        Logger.GetInstance().WriteError(e.Message);
                    }
                    catch
                    {
                        Console.WriteLine("Logger and Cache Manager not setup");
                    }
                }

            }
        }
    }
}
```

6.21 WebServiceCSharp/Program.cs File Reference

Classes

- class WebServer.Program

  The Initialiser

Namespaces

- namespace WebServer

6.22 Program.cs

```
using System;
using System.Threading;
using System.IO;
using OSXJV.Classes;
using OSXJV.Server;

namespace WebServer
{
    class Program
    {
        static void Main(string[] args)
        {

            if (args.Length == 0)
            {
                Console.WriteLine("Using Default Cache Directory Path and Logger Directory Path");
                string dir = Directory.GetCurrentDirectory();
                Array.Resize(ref args, 2);
                args[0] = dir + "/Cache/";
                args[1] = dir + "/Logger/";
                if (!Directory.Exists(args[0]))
                    Directory.CreateDirectory(args[0]);
                if (!Directory.Exists(args[1]))
                    Directory.CreateDirectory(args[1]);
            }

            if (args[0] == args[1])
            {
                Console.WriteLine("Cache location and Log location is the same. Please enter two different
      locations");
```

```
            }
            else
            {
                try
                {
                    OSXJVServer s = new OSXJVServer();
                    s.Start(args[0], args[1]);
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    Console.WriteLine("Press any key to exit");
                    Console.Read();
                }

            }
        }
    }
}
```

6.23 WebServiceCSharp/Properties/AssemblyInfo.cs File Reference

6.24 AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("WebServiceCSharp")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("WebServiceCSharp")]
[assembly: AssemblyCopyright("Copyright ©  2016")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("a57034df-dc0f-44ce-bb8a-cddafe37db17")]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Index