Open Source XML & JSON Visualisation Software

1

Generated by Doxygen 1.8.13

Mon Apr 17 2017 18:33:50

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 Filesexi
- class Logger

  A simple class that writes errors to a single file.
- class Node

  Contain Processed Document Information
- class Output

  Creates the Output for the web page to display.
- class ProcessDocument

  Class the Processes the document
- class Request

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

  The Object containing data to send to the client
- class Validation

  Preform validation

4.3 OSXJV.Server Namespace Reference

Classes

- class OSXJVServer

  HTTPServer that process the incoming requests.

4.4 WebServer Namespace Reference

Classes

- class Program

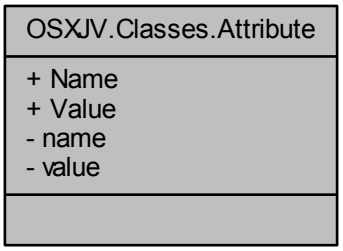
  The Initialiser

Chapter 5

Class Documentation

5.1 OSXJV.Classes.Attribute Class Reference

Collaboration diagram for OSXJV.Classes.Attribute:



Properties

- string Name `[get, set]`
- string Value `[get, set]`

Private Attributes

- string name
- string value

5.1.1 Detailed Description

Definition at line 6 of file Attribute.cs.

5.1.2 Member Data Documentation

5.1.2.1 name

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

Definition at line 8 of file Attribute.cs.

5.1.2.2 value

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

Definition at line 9 of file Attribute.cs.

5.1.3 Property Documentation

5.1.3.1 Name

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

Definition at line 15 of file Attribute.cs.

Referenced by OSXJV.Classes.Output.CreateNodeView(), OSXJV.Classes.ProcessDocument.ProcessElement(), and OSXJV.Classes.ProcessDocument.ProcessRoot().

5.1.3.2 Value

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

Definition at line 30 of file Attribute.cs.

Referenced by OSXJV.Classes.Output.CreateNodeView(), OSXJV.Classes.ProcessDocument.ProcessElement(), and OSXJV.Classes.ProcessDocument.ProcessRoot().

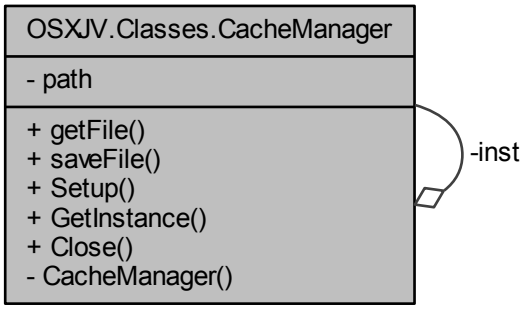
The documentation for this class was generated from the following file:

- OSXJVClasses/Attribute.cs

5.2 OSXJV.Classes.CacheManager Class Reference

Manages Saving an Retrieving Filesexi

Collaboration diagram for OSXJV.Classes.CacheManager:



Public Member Functions

- string getFile (string ID)

  Retrieve the file from caching
- bool saveFile (string ID, string nodes)

  Save the file to the local system for caching

Static Public Member Functions

- static bool Setup (string path)
- static CacheManager GetInstance ()

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

Private Member Functions

- CacheManager (string path)

Private Attributes

- string path

Static Private Attributes

- static CacheManager inst

5.2.1 Detailed Description

Manages Saving an Retrieving Filesexi

Definition at line 9 of file CacheManager.cs.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 CacheManager()

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

Definition at line 14 of file CacheManager.cs.

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

5.2.3 Member Function Documentation

5.2.3.1 Close()

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

Definition at line 99 of file CacheManager.cs.

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

5.2.3.2 getFile()

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

Retrieve the file from caching

Parameters

ID	Unique ID of the file

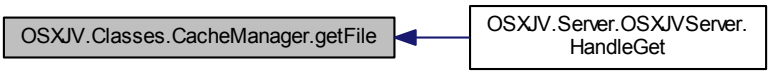
Returns

Definition at line 51 of file CacheManager.cs.

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

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

Here is the caller graph for this function:



5.2.3.3 GetInstance()

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

Get the single instance of the class

Returns

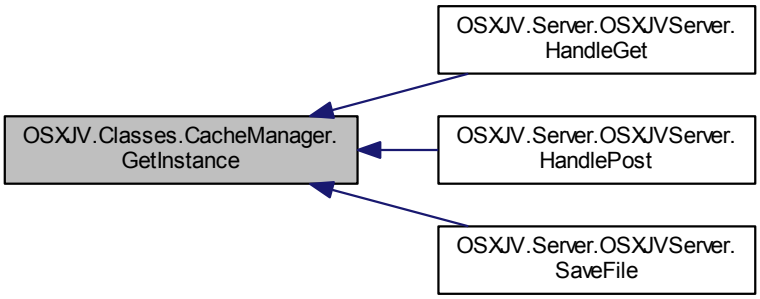
An instance of CacheManager

Definition at line 38 of file CacheManager.cs.

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

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

Here is the caller graph for this function:



5.2.3.4 saveFile()

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

Save the file to the local system for caching

Parameters

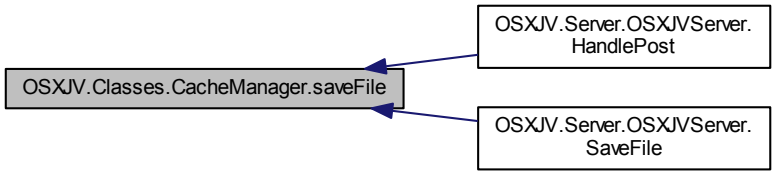
ID	Unique ID of the file
nodes	The document to be saved

Definition at line 75 of file CacheManager.cs.

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

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

Here is the caller graph for this function:



5.2.3.5 Setup()

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

Parameters

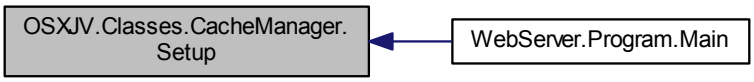
path	

Definition at line 23 of file CacheManager.cs.

Referenced by WebServer.Program.Main().

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

Here is the caller graph for this function:



5.2.4 Member Data Documentation

5.2.4.1 inst

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

Definition at line 11 of file CacheManager.cs.

5.2.4.2 path

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

Definition at line 12 of file CacheManager.cs.

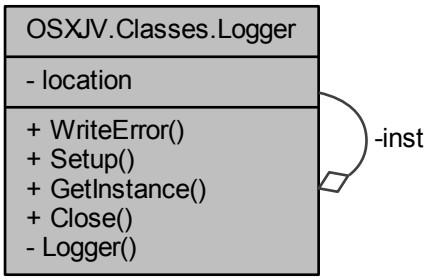
The documentation for this class was generated from the following file:

- OSXJVClasses/CacheManager.cs

5.3 OSXJV.Classes.Logger Class Reference

A simple class that writes errors to a single file.

Collaboration diagram for OSXJV.Classes.Logger:



Public Member Functions

- void WriteError (string error)

  Writes an error the location provided

Static Public Member Functions

- static bool Setup (string location)
- static Logger GetInstance ()

  Gets the single instance of Logger
- static void Close ()

Private Member Functions

- Logger (string location)

Private Attributes

- string location

Static Private Attributes

- static Logger inst

  Singleton instance of Logger

5.3.1 Detailed Description

A simple class that writes errors to a single file.

Definition at line 9 of file Logger.cs.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Logger()

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

Definition at line 17 of file Logger.cs.

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

5.3.3 Member Function Documentation

5.3.3.1 Close()

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

Definition at line 72 of file Logger.cs.

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

5.3.3.2 GetInstance()

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

Gets the single instance of Logger

Returns

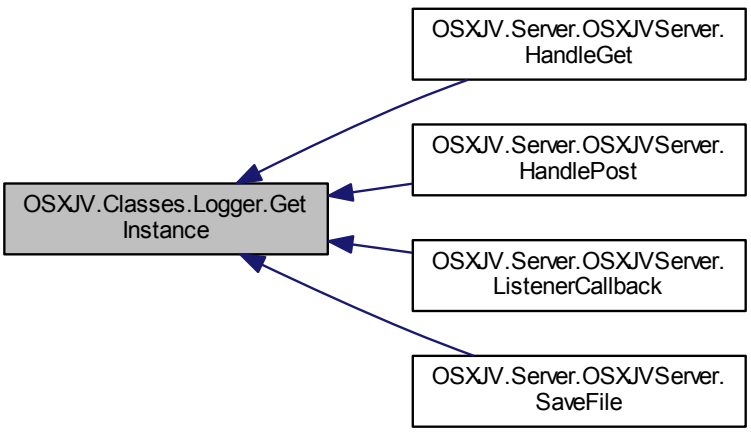
Instance of Logger

Definition at line 41 of file Logger.cs.

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

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

Here is the caller graph for this function:



5.3.3.3 Setup()

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

Parameters

location	

Definition at line 26 of file Logger.cs.

Referenced by WebServer.Program.Main().

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

Here is the caller graph for this function:

5.3.3.4 WriteError()

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

Writes an error the location provided

Parameters

error	The error message

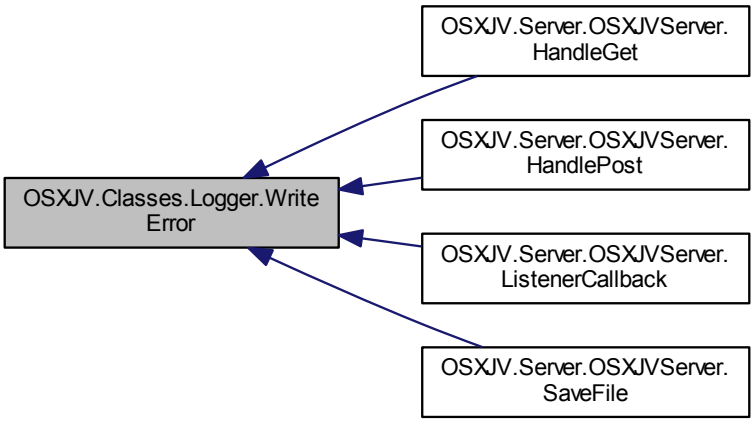
Definition at line 53 of file Logger.cs.

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

```
00054         {
```

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

Here is the caller graph for this function:



5.3.4 Member Data Documentation

5.3.4.1 inst

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

Singleton instance of Logger

Definition at line 14 of file Logger.cs.

5.3.4.2 location

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

Definition at line 15 of file Logger.cs.

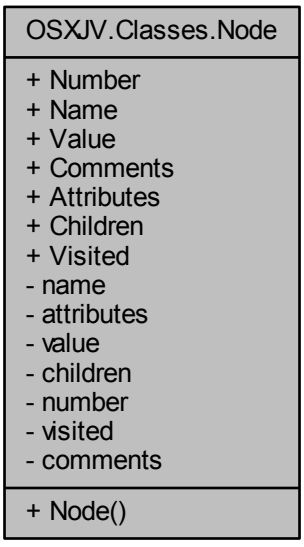
The documentation for this class was generated from the following file:

- OSXJVClasses/Logger.cs

5.4 OSXJV.Classes.Node Class Reference

Contain Processed Document Information

Collaboration diagram for OSXJV.Classes.Node:



Public Member Functions

- Node ()

  Constructor

Properties

- int Number [get, set]

  The Number of the Node
- string Name [get, set]

  The Name of Node
- string Value [get, set]

  The Value of the Node
- List< string > Comments [get, set]

  Comments That the Node Has.
- List< Attribute > Attributes [get, set]

  Attributes the Node has.
- List< Node > Children [get, set]

  Children Nodes the Node is linked to.
- bool Visited [get, set]

  If the node has been visited previous by the ProcessDocument, prevent multiple same Nodes.

Private Attributes

- string name
- List< Attribute > attributes
- string value
- List< Node > children
- int number
- bool visited
- List< string > comments

5.4.1 Detailed Description

Contain Processed Document Information

Definition at line 10 of file Node.cs.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Node()

```
OSXJV.Classes.Node.Node ( )
```

Constructor

Definition at line 23 of file Node.cs.

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

5.4.3 Member Data Documentation

5.4.3.1 attributes

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

Definition at line 13 of file Node.cs.

5.4.3.2 children

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

Definition at line 15 of file Node.cs.

5.4.3.3 comments

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

Definition at line 18 of file Node.cs.

5.4.3.4 name

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

Definition at line 12 of file Node.cs.

5.4.3.5 number

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

Definition at line 16 of file Node.cs.

5.4.3.6 value

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

Definition at line 14 of file Node.cs.

5.4.3.7 visited

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

Definition at line 17 of file Node.cs.

5.4.4 Property Documentation

5.4.4.1 Attributes

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

Attributes the Node has.

Definition at line 102 of file Node.cs.

Referenced by OSXJV.Classes.Output.CreateNodeView(), OSXJV.Classes.ProcessDocument.ProcessElement(), and OSXJV.Classes.ProcessDocument.ProcessRoot().

5.4.4.2 Children

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

Children Nodes the Node is linked to.

Definition at line 119 of file Node.cs.

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

5.4.4.3 Comments

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

Comments That the Node Has.

Definition at line 85 of file Node.cs.

Referenced by OSXJV.Classes.Output.CreateNodeChildViewsParallel(), OSXJV.Classes.Output.CreateNodeView(), and OSXJV.Classes.ProcessDocument.ProcessComment().

5.4.4.4 Name

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

The Name of Node

Definition at line 52 of file Node.cs.

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

5.4.4.5 Number

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

The Number of the Node

Definition at line 36 of file Node.cs.

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

5.4.4.6 Value

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

The Value of the Node

Definition at line 69 of file Node.cs.

Referenced by OSXJV.Classes.Output.CreateNodeView(), and OSXJV.Classes.ProcessDocument.ProcessText().

5.4.4.7 Visited

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

If the node has been visited previous by the ProcessDocument, prevent multiple same Nodes.

Definition at line 136 of file Node.cs.

Referenced by OSXJV.Classes.ProcessDocument.ProcessElement(), and OSXJV.Classes.ProcessDocument.↩ ProcessRoot().

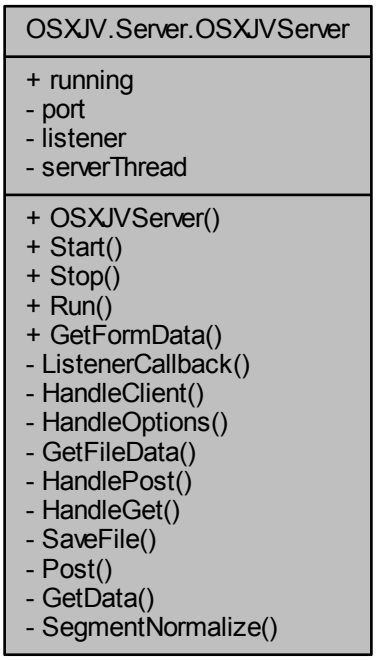
The documentation for this class was generated from the following file:

- OSXJVClasses/Node.cs

5.5 OSXJV.Server.OSXJVServer Class Reference

HTTPServer that process the incoming requests.

Collaboration diagram for OSXJV.Server.OSXJVServer:



Public Member Functions

- OSXJVServer ()

  The Server Handler
- bool Start ()

  Starts server in new thread
- bool Stop ()

  Stop the listener and about all current requests
- void Run ()

  Function that constantly listens for connections
- Request GetFormData (Stream input)

  Extract the files from the request

Static Public Attributes

- static bool running = false

  True if the server is able to accept requests.

Private Member Functions

- void ListenerCallback (IAsyncResult result)

  Handles Requests 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.

Private Attributes

- int port = 8082
- HttpListener listener
- Thread serverThread = null

5.5.1 Detailed Description

HTTPServer that process the incoming requests.

Definition at line 16 of file OSXJVServer.cs.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 OSXJVServer()

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

The Server Handler

Definition at line 34 of file OSXJVServer.cs.

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

5.5.3 Member Function Documentation

5.5.3.1 GetData()

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

Get the data from the client.

Parameters

req	The request from the client

Returns

A Request Object

Definition at line 413 of file OSXJVServer.cs.

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

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

5.5.3.2 GetFileData()

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

Get Data if the data is retrieved

Parameters

input	Client Stream Input
type	The MIME type

Returns

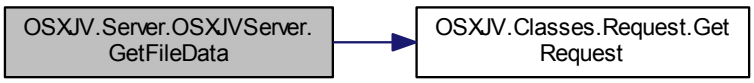
A Response object to send to the user

Definition at line 209 of file OSXJVServer.cs.

References OSXJV.Classes.Request.GetRequest().

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

Here is the call graph for this function:



5.5.3.3 GetFormData()

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

Extract the files from the request

Parameters

input	Requests input stream

Returns

New Request Object

Exceptions

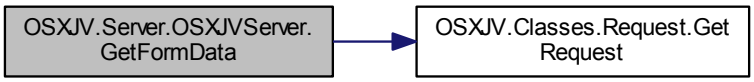
System.InvalidOperationException	Thrown when no files are included with the request

Definition at line 185 of file OSXJVServer.cs.

References OSXJV.Classes.Request.GetRequest().

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

Here is the call graph for this function:



5.5.3.4 HandleClient()

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

Handles the client

Parameters

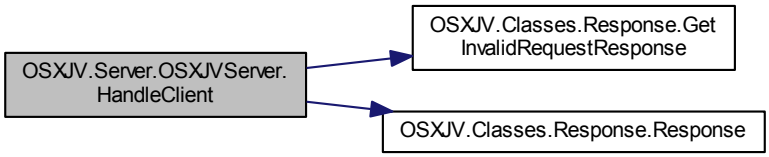
c	The Request

Definition at line 146 of file OSXJVServer.cs.

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

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

Here is the call graph for this function:



5.5.3.5 HandleGet()

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

Handles a GET request.

Parameters

req	The request to be processed.

Returns

A Response object to send to the user

Definition at line 312 of file OSXJVServer.cs.

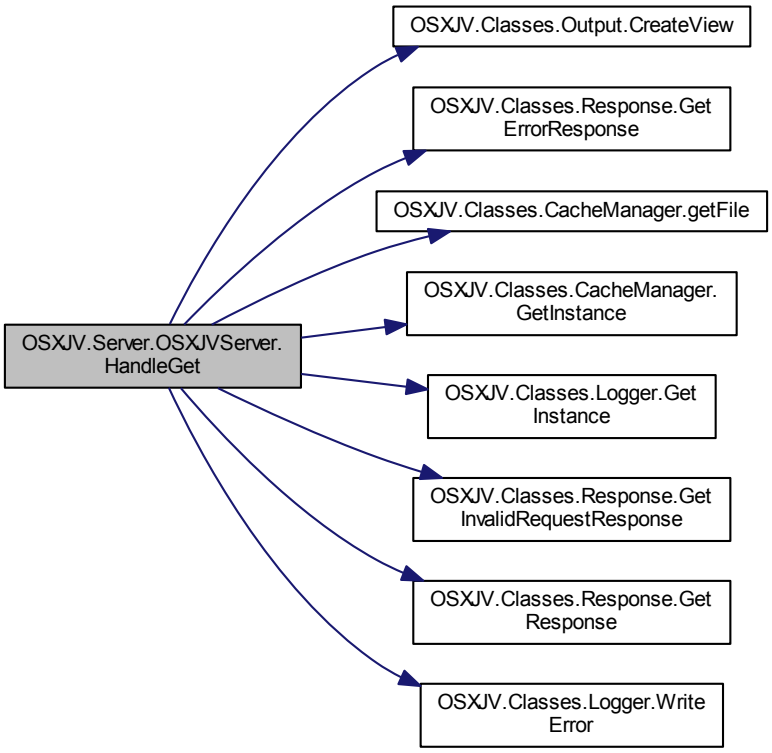
References OSXJV.Classes.Output.CreateView(), OSXJV.Classes.Response.GetErrorResponse(), OSXJV.Classes.CacheManager.getFile(), OSXJV.Classes.CacheManager.GetInstance(), OSXJV.Classes.Logger.GetInstance(), OSXJV.Classes.Response.GetInvalidRequestResponse(), OSXJV.Classes.Response.GetResponse(), and OSXJV.Classes.Logger.WriteError().

```
00313         {
00314             if (SegmentNormalize(req.Url.Segments[1]).Equals("Process"))
00315             {
00316                 if (req.Url.Segments.Length == 4)
00317                 {
00318 
00319                     Node cached;
00320                     try
00321                     {
00322                         cached = JsonConvert.DeserializeObject<Node>(
      CacheManager.GetInstance().getFile(req.Url.Segments[2]));
00323                     }
00324                     catch (Exception e)
00325                     {
00326                         Logger.GetInstance().WriteError(e.Message);
00327                         JObject eRes = new JObject();
00328                         eRes.Add("Error", "Error Creating Response");
00329                         return Response.GetErrorResponse(eRes.ToString());
00330                     }
00331                     Output o = new Output(cached);
00332                     JObject response = new JObject();
00333                     response.Add("view", o.CreateView(int.Parse(req.Url.Segments[3])));
00334                     byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
00335                     return Response.GetResponse(200, "application/json", bytes);

00336                 }
00337                 else if (req.Url.Segments.Length == 5)
00338                 {
00339 
00340                     Node cached;
00341                     try
00342                     {
00343                         cached = JsonConvert.DeserializeObject<Node>(
      CacheManager.GetInstance().getFile(req.Url.Segments[2]));
00344                     }
00345                     catch (Exception e)
00346                     {
00347                         Logger.GetInstance().WriteError(e.Message);
00348                         JObject eRes = new JObject();
00349                         eRes.Add("Error", "Error Creating Response");
00350                         return Response.GetErrorResponse(eRes.ToString());
00351                     }
00352                     Output o = new Output(cached);
00353                     JObject response = new JObject();
00354                     response.Add("view", o.CreateView(int.Parse(
      SegmentNormalize(req.Url.Segments[3])), 4, int.Parse(
      SegmentNormalize(req.Url.Segments[4]))));
00355                     byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
00356                     return Response.GetResponse(200, "application/json", bytes);

00357                 }
00358                 else
00359                     return Response.GetInvalidRequestResponse();
00360             }
00361             //If it got here its an invalid response.
00362             return Response.GetInvalidRequestResponse();
00363         }
```

Here is the call graph for this function:



5.5.3.6 HandleOptions()

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

Sends to the Client What the Server Supports

Parameters

response	The Request Response Object

Definition at line 170 of file OSXJVServer.cs.

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

5.5.3.7 HandlePost()

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

Handles a POST request.

Parameters

req	The request to be processed.

Returns

A Response object to send to the user

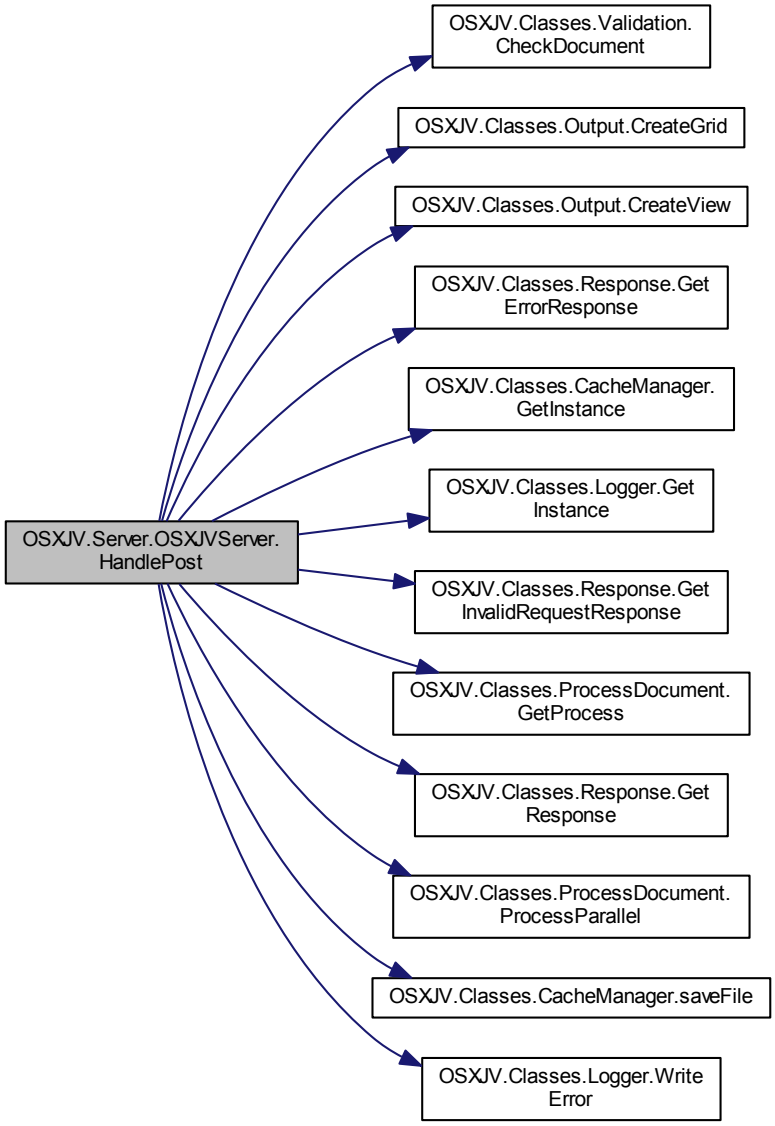
Definition at line 233 of file OSXJVServer.cs.

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

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

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

Here is the call graph for this function:



5.5.3.8 ListenerCallback()

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

Handles Requests Asyncronously

Parameters

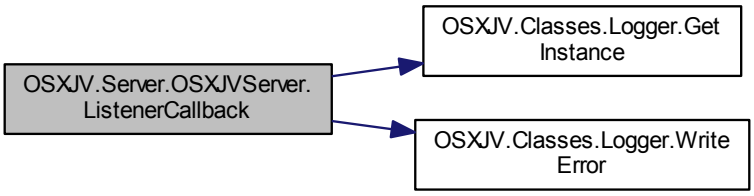
result	The Request Object Coming In.

Definition at line 124 of file OSXJVServer.cs.

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

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

Here is the call graph for this function:



5.5.3.9 Post()

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

Send data to the client.

Parameters

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 394 of file OSXJVServer.cs.

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

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

5.5.3.10 Run()

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

Function that constantly listens for connections

Definition at line 76 of file OSXJVServer.cs.

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

5.5.3.11 SaveFile()

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

Save data recievied from client.

Parameters

id	Unique ID
nodes	The Processed Data

Exceptions

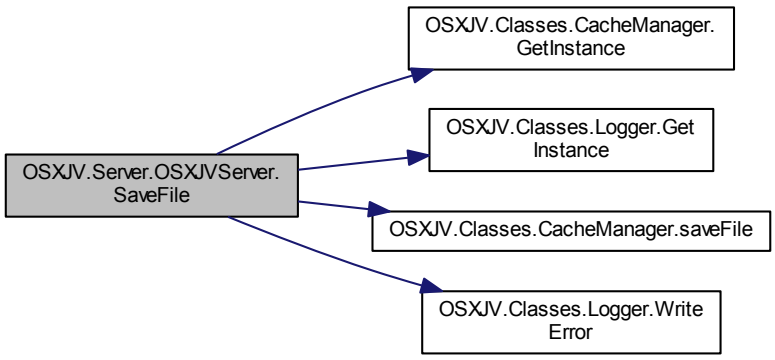
ArgumentException	Thrown when nodes is null or id is null or empty

Definition at line 371 of file OSXJVServer.cs.

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

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

Here is the call graph for this function:



5.5.3.12 SegmentNormalize()

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

Removes '/' from the string.

Parameters

input	A string from the URL

Returns

Normalised String

Definition at line 437 of file OSXJVServer.cs.

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

5.5.3.13 Start()

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

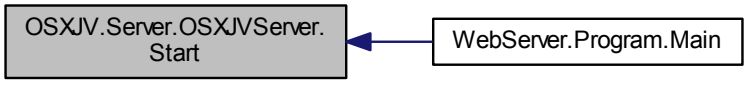
Starts server in new thread

Definition at line 43 of file OSXJVServer.cs.

Referenced by WebServer.Program.Main().

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

Here is the caller graph for this function:



5.5.3.14 Stop()

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

Stop the listener and about all current requests

Definition at line 58 of file OSXJVServer.cs.

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

5.5.4 Member Data Documentation

5.5.4.1 listener

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

Definition at line 24 of file OSXJVServer.cs.

5.5.4.2 port

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

Definition at line 18 of file OSXJVServer.cs.

5.5.4.3 running

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

True if the server is able to accept requests.

Definition at line 23 of file OSXJVServer.cs.

5.5.4.4 serverThread

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

Definition at line 29 of file OSXJVServer.cs.

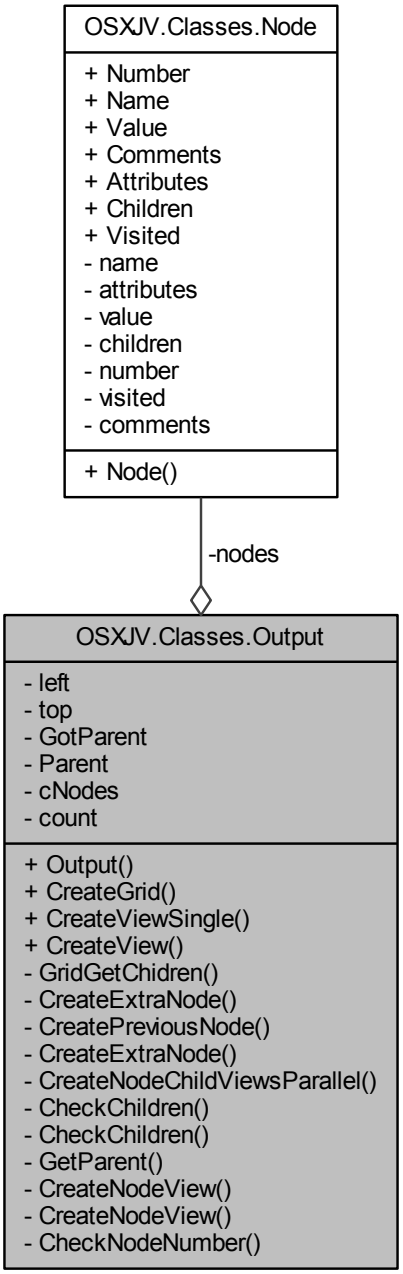
The documentation for this class was generated from the following file:

- OSXJVServer.cs

5.6 OSXJV.Classes.Output Class Reference

Creates the Output for the web page to display.

Collaboration diagram for OSXJV.Classes.Output:



Public Member Functions

- Output (Node nodes)

  Creation of a Output object.
- JObject CreateGrid ()

  Creates the grid data.
- string CreateViewSingle (int node, int nodeStart=0)

CreateView using a Single Thread

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

  Creates the view of nodes using multiple threads.

Private Member Functions

- JObject GridGetChidren (Node n)

  Recursive function to get all the nodes data for the grid .
- string CreateExtraNode (string type, int id)

  Builds a get more button to display
- string CreatePreviousNode (string type, int leftVal, int topVal, int id)

  Create a previous node button
- string CreateExtraNode (string type, int leftVal, int topVal, int id)

  Create a extra node button
- void CreateNodeChildViewsParallel (List< Node > job, int start, bool showHigher, int next, int previous)

  Generate Output HTML when using multi-threads
- string CheckChildren (Node n, int number)

  Check child nodes if the are to be part of the output.
- string CheckChildren (Node n, int number, int pCount, int nodeStart, ref bool found)
- void GetParent (Node node, int number)

  Finds the parent node.
- string CreateNodeView (Node n, string type, int leftVal, int topVal)

  Generates HTML for the Specific Node (Multi-Threaded Version)
- string CreateNodeView (Node n, string type)

  Generates HTML for specific Node (Single Threaded Version)
- bool CheckNodeNumber (Node n, int number)

  Checks if Node number and inputted number match.

Private Attributes

- int left = 100
- int top = 130
- Node nodes
- bool GotParent = false
- int Parent = 0

  Parent of node when building output (Used when getting Node other than root).
- List< Tuple< int, string > > cNodes = new List<Tuple<int, string>>()

  Used in Threading, list of calculated HTML strings.

Static Private Attributes

- static int count = 0

  Used in Single Threaded operation to keep track of div id.

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 48 of file Output.cs.

```
00049         {
00050             if (nodes == null)
00051                 throw new ArgumentException();
00052             this.nodes = nodes;
00053         }
```

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 405 of file Output.cs.

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

```
00406         {
00407             string output = "";
00408             if (CheckNodeNumber(n, number))
00409             {
00410 
00411                 List<Thread> threadList = new List<Thread>();
00412 
00413                 int count = 0;
00414                 output += CreateNodeView(n, "node");
00415                 foreach (Node n2 in n.Children)
00416                 {
00417                     count++;
00418                     output += CreateNodeView(n2, "node-child");
00419 
00420                 }
00421             }
00422             else if (n.Children.Count > 0)
00423             {
00424                 foreach (Node n2 in n.Children)
00425                 {
00426                     if (GotParent)
00427                     {
00428                         if (n2.Number == Parent)
00429                         {
00430                             output += CreateNodeView(n2, "node-parent");
00431                         }
00432                     }
00433                     output += CheckChildren(n2, number);
00434                 }
00435             }
00436 
00437             return output;
00438         }
```

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 449 of file Output.cs.

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

```
00450         {
00451             string output = "";
00452             if (CheckNodeNumber(n, number))
```

```
00453             {
00454                 found = true;
00455                 List<Thread> threadList = new List<Thread>();
00456 
00457                 int count = 0;
00458                 output += CreateNodeView(n, "node");
00459                 count++;
00460                 //output += CreateNodeView(n2, "node-child");
00461                 int childCount = 0;
00462 
00463                 if (n.Children.Count < 200)
00464                     childCount = n.Children.Count;
00465                 else
00466                 {
00467                     childCount = 200;
00468                 }
00469                 if (childCount < pCount * 2)
00470                 {
00471                     foreach(Node n2 in n.Children)
00472                     {
00473                         output += CreateNodeView(n2, "node-child");
00474                     }
00475                 }
00476                 else
00477                 {
00478                     int spread = (int)Math.Ceiling((double)childCount / (double)pCount);
00479 
00480                     if (childCount > 0)
00481                     {
00482                         for (int i = 0; i < pCount; i++)
00483                         {
00484                             int neg = 0;
00485                             if ((spread * (i + 1)) > childCount)
00486                             {
00487                                 neg = childCount - (spread * (i + 1));
00488                             }
00489                             int start = (spread * i);
00490                             int rangeStart = (spread * i) + nodeStart;
00491                             bool showHigher = nodeStart != 0 ? true : false;
00492 
00493                             List<Node> NodesToProcess = n.Children.GetRange(rangeStart, spread + neg);
00494 
00495                             if (NodesToProcess.Count > 0)
00496                             {
00497                                 Thread threadJob = new Thread(() =>
      CreateNodeChildViewsParallel(NodesToProcess, start, showHigher, childCount +
      nodeStart, nodeStart - childCount));
00498                                 threadJob.Name = i.ToString();
00499                                 threadJob.Start();
00500                                 threadList.Add(threadJob);
00501                             }
00502                         }
00503                         foreach (Thread t in threadList)
00504                         {
00505                             t.Join();
00506                         }
00507                         cNodes.Sort((x, y) => x.Item1.CompareTo(y.Item1));
00508 
00509                         foreach (Tuple<int, string> tup in cNodes)
00510                         {
00511                             output += tup.Item2;
00512                         }
00513                     }
00514                 }
00515             }
00516             else if (n.Children.Count > 0)
00517             {
00518                 foreach (Node n2 in n.Children)
00519                 {
00520                     if (GotParent)
00521                     {
00522                         if (n2.Number == Parent)
00523                         {
00524                             output += CreateNodeView(n2, "node-parent");
00525                         }
00526                     }
00527                     output += CheckChildren(n2, number,pCount,nodeStart,ref found);
00528                 }
00529             }
00530 
00531             return output;
00532         }
```

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 670 of file Output.cs.

References OSXJV.Classes.Node.Number.

```
00671         {
00672             return n.Number.Equals(number);
00673         }
```

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 170 of file Output.cs.

```
00171         {
00172             string node = "";
00173 
00174             if (type == "node")
00175             {
```

```
00176                 if (GotParent)
00177                 {
00178                     left = left + 400;
00179                 }
00180             }
00181             if (type == "node-child")
00182             {
00183                 left = left + 400;
00184             }
00185 
00186             node += "<div class='node-child type ui-draggable ui-selectee' style='left:" +
      left + "px; top:" + top + "px;margin-bottom:50px;'>";
00187             node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
    )'>Show Lower</button></span></div>";
00188             node += "</div></div>";
00189             return node;
00190         }
```

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 229 of file Output.cs.

```
00230         {
00231             string node = "";
00232 
00233             if (type == "node")
00234             {
00235                 if (GotParent)
00236                 {
00237                     leftVal = leftVal + 400;
00238                 }
00239             }
00240             if (type == "node-child")
00241             {
00242                 leftVal = leftVal + 400;
00243             }
00244             node += "<div class='node-child type ui-draggable ui-selectee' style='left:" + leftVal + "px;
       top:" + topVal + "px;margin-bottom:50px;'>";
00245             node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
    )'>Show Lower</button></span></div>";
00246             node += "</div></div>";
00247             return node;
00248         }
```

5.6.3.6 CreateGrid()

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

Creates the grid data.

Returns

A JSON object

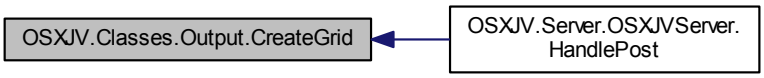
Definition at line 59 of file Output.cs.

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

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

```
00060         {
00061             JObject obj = new JObject();
00062             obj.Add("text", nodes.Name);
00063             obj.Add("id", nodes.Number);
00064             obj.Add("state", new JObject(new JProperty("selected", true)));
00065 
00066             if(nodes.Children.Count > 0)
00067             {
00068                 JArray array = new JArray();
00069                 foreach (Node n2 in nodes.Children)
00070                 {
00071                     array.Add(GridGetChidren(n2));
00072                 }
00073                 obj.Add("children", array);
00074             }
00075             return obj;
00076         }
```

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 258 of file Output.cs.

References OSXJV.Classes.Node.Comments.

```
00259         {
00260             int threadID = int.Parse(Thread.CurrentThread.Name);
00261             string type = "node-child";
00262             string output = "";
00263 
00264             if(start == 0 && showHigher)
00265             {
00266                 output += CreatePreviousNode(type, left,
      top, previous);
00267             }
00268             bool hadCommentsPrev = false;
00269             int numCommentsPrevious = 0;
00270 
00271             foreach(Node n in job)
00272             {
00273                 int extra = showHigher ? 130 * (start +1) : 130 * start;
00274 
00275                 if (hadCommentsPrev)
00276                     extra += (numCommentsPrevious * 25);
00277 
00278                 if (n.Comments.Count > 0)
00279                 {
00280                     hadCommentsPrev = true;
00281                     numCommentsPrevious = n.Comments.Count;
00282                 }
00283                 else
00284                     hadCommentsPrev = false;
00285 
00286                 output += CreateNodeView(n, type,left,top + extra);
00287                 start++;
00288                 if (start == 200)
00289                 {
00290                     output += CreateExtraNode(type, left, top + extra + 130,next);
00291                     break;
00292                 }
00293 
00294             }
00295 
00296             cNodes.Add(new Tuple<int, string>(threadID, output));
00297         }
```

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 566 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.

```
00567         {
00568             string node = "";
00569 
00570             if(type == "node")
00571             {
00572                 if(GotParent)
00573                 {
00574                     leftVal = leftVal + 400;
00575                 }
00576             }
00577             if(type == "node-child")
00578             {
00579                 leftVal = leftVal + 400;
00580             }
00581             node += "<div id='" + n.Number + "'class='" + type + " type ui-draggable ui-selectee'
      style='left:" + leftVal + "px; top:" + topVal + "px;'>";
00582             node += "<div class='head'><span><button class='nameBtn' onclick='GetNode("+n.Number+")'>" + n.
     Name + "</button></span></div>";
00583             if (!string.IsNullOrEmpty(n.Value))
00584             {
00585                 node += string.Format("<div class='blockR'><p>Value</p></div><div
      class=comment><span>{0}</span></div>", n.Value);
00586             }
00587             if(n.Comments.Count >0)
00588             {
00589                 node += "<div><p class='text-center'>Comments</p></div>";
00590 
00591                 foreach(string com in n.Comments)
00592                 {
00593                     node += "<div class='comment'>" + com + "</div>";
00594                 }
00595             }
00596             if (n.Attributes.Count > 0)
00597             {
00598                 node += "<div class='attribute'><div class='aHeader'><p><button><i class='fa
      fa-plus'></i></button>Attributes</p></div><div class='options'>";
00599                 foreach (Attribute a in n.Attributes)
00600                 {
00601                     node += string.Format("<div class='blockR'><p>{0}</p></div><div
      class='comment'><p>{1}</p></div>", a.Name, a.Value);
00602                 }
00603                 node += "</div>";
00604             }
00605             node += "</div></div>";
00606             return node;
00607         }
```

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 615 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.

```
00616         {
00617             string node = "";
00618             int leftVal = left;
00619             if (type == "node")
00620             {
00621                 if (GotParent)
00622                 {
00623                     left = left + 400;
00624                     leftVal = left;
00625                 }
00626             }
00627             if (type == "node-child")
00628             {
00629                 leftVal = leftVal + 400;
00630             }
00631             node += "<div id='" + n.Number + "'class='" + type + " type ui-draggable ui-selectee'
       style='left:" + leftVal + "px; top:" + top + "px;'>";
00632             node += "<div class='head'><span><button class='nameBtn' onclick='GetNode(" + n.Number + ")'>"
      + n.Name + "</button></span></div>";
00633             if (!string.IsNullOrEmpty(n.Value))
00634             {
00635                 node += string.Format("<div class='blockR'><p>Value</p></div><div
       class=comment><span>{0}</span></div>", n.Value);
00636             }
00637             if (n.Comments.Count > 0)
00638             {
00639                 node += "<div><p class='text-center'>Comments</p></div>";
00640 
00641                 foreach (string com in n.Comments)
00642                 {
00643                     node += "<div class='comment'>" + com + "</div>";
00644                 }
00645             }
00646             if (n.Attributes.Count > 0)
00647             {
00648                 node += "<div class='attribute'><div class='aHeader'><p><button><i class='fa
       fa-plus'></i></button>Attributes</p></div><div class='options'>";
00649                 foreach (Attribute a in n.Attributes)
00650                 {
00651                     node += string.Format("<div class='blockR'><p>{0}</p></div><div
       class='comment'><p>{1}</p></div>", a.Name, a.Value);
00652                 }
00653                 node += "</div>";
00654             }
00655             node += "</div></div>";
00656 
00657             if (type == "node-child")
00658             {
00659                 top = top + 130;
00660             }
00661             return node;
00662         }
```

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 200 of file Output.cs.

```
00201         {
00202             string node = "";
00203 
00204             if (type == "node")
00205             {
00206                 if (GotParent)
00207                 {
00208                     leftVal = leftVal + 400;
00209                 }
00210             }
00211             if (type == "node-child")
00212             {
00213                 leftVal = leftVal + 400;
00214             }
00215             node += "<div class='node-child type ui-draggable ui-selectee' style='left:" + leftVal + "px;
       top:" + topVal + "px;'>";
00216             node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
      )'>Show Higher</button></span></div>";
00217             node += "</div></div>";
00218             return node;
00219         }
```

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 306 of file Output.cs.

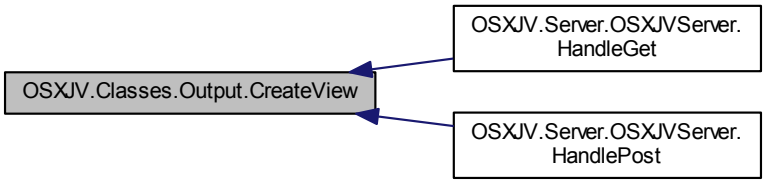
References OSXJV.Classes.Node.Children, and OSXJV.Classes.Node.Number.

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

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

```
00373                     if (nodes.Number == Parent)
00374                     {
00375                         output += CreateNodeView(nodes, "node-parent");
00376                     }
00377                 }
00378                 bool found =false;
00379                 foreach (Node n2 in nodes.Children)
00380                 {
00381                     if (GotParent)
00382                     {
00383                         if (n2.Number == Parent)
00384                         {
00385                             output += CreateNodeView(n2, "node-parent");
00386                         }
00387                     }
00388                     temp += CheckChildren(n2, node,pCount,nodeStart,ref found);
00389                     if (found)
00390                         break;
00391                 }
00392                 if (!string.IsNullOrEmpty(temp))
00393                     output += temp;
00394             }
00395             output += "</div></div>";
00396             return output;
00397         }
```

Here is the caller graph for this function:



5.6.3.12 CreateViewSingle()

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

CreateView using a Single Thread

Parameters

node	Index of node to start from
nodeStart	Where to start the child nodes from

Returns

String of calculated HTML

Definition at line 107 of file Output.cs.

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

```
00108         {
00109             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'>";
00110
00111             if (nodes.Number.Equals(node))
00112             {
00113                 int count = 0;
00114                 output += CreateNodeView(nodes, "node");
00115
00116
00117                 foreach (Node n in nodes.Children)
00118                 {
00119                     if(nodeStart > 0)
00120                     {
00121                         if (count != nodeStart)
00122                             continue;
00123                     }
00124                     count++;
00125                     output += CreateNodeView(n, "node-child"); //Child(Nodes) Thread
00126
00127                     if ((count-nodeStart) == 200)
00128                     {
00129                         output += CreateExtraNode("node-child",count);
00130                         break;
00131                     }
00132                 }
00133             }
00134
00135             else
00136             {
00137                 GetParent(nodes, node);
00138                 string temp = "";
00139                 if (GotParent)
00140                 {
00141                     if (nodes.Number == Parent)
00142                     {
00143                         output += CreateNodeView(nodes, "node-parent");
00144                     }
00145                 }
00146                 foreach (Node n2 in nodes.Children)
00147                 {
00148                     if (GotParent)
00149                     {
00150                         if (n2.Number == Parent)
00151                         {
00152                             output += CreateNodeView(n2, "node-parent");
00153                         }
00154                     }
00155                     temp += CheckChildren(n2, node);
00156                 }
00157                 if (!string.IsNullOrEmpty(temp))
00158                     output += temp;
00159             }
00160             output += "</div></div>"; //Close out divs
00161             return output;
00162         }
```

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 539 of file Output.cs.

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

```
00540         {
00541             if(!CheckNodeNumber(node,number))
00542             {
00543                 foreach(Node n in node.Children)
00544                 {
00545                     if(CheckNodeNumber(n, number))
00546                     {
00547                         Parent = node.Number;
00548                         GotParent = true;
00549                     }
00550                     else
00551                     {
00552                         GetParent(n, number);
00553                     }
00554                 }
00555             }
00556         }
```

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 83 of file Output.cs.

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

```
00084         {
00085             JObject child = new JObject();
00086             child.Add("id", n.Number);
00087             child.Add("text", n.Name);
00088 
00089             if (n.Children.Count > 0)
00090             {
00091                 JArray array = new JArray();
00092                 foreach(Node n2 in n.Children)
00093                 {
00094                     array.Add(GridGetChidren(n2));
00095                 }
00096                 child.Add("children", array);
00097             }
00098             return child;
00099         }
```

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 count

```
int OSXJV.Classes.Output.count = 0  [static], [private]
```

Used in Single Threaded operation to keep track of div id.

Definition at line 42 of file Output.cs.

5.6.4.3 GotParent

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

Definition at line 26 of file Output.cs.

5.6.4.4 left

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

Definition at line 16 of file Output.cs.

5.6.4.5 nodes

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

Definition at line 21 of file Output.cs.

5.6.4.6 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.

5.6.4.7 top

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

Definition at line 16 of file Output.cs.

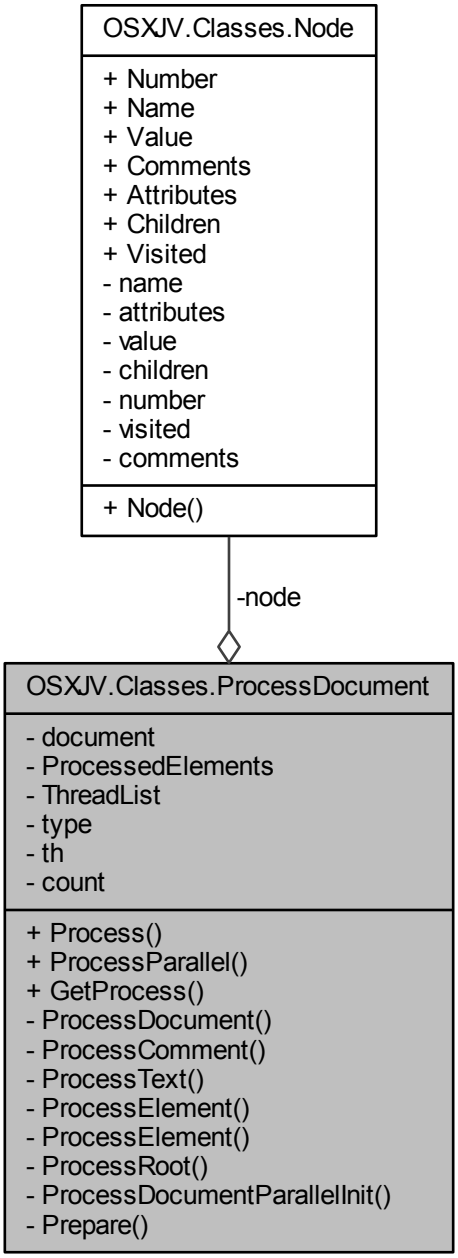
The documentation for this class was generated from the following file:

- OSXJVClasses/Output.cs

5.7 OSXJV.Classes.ProcessDocument Class Reference

Class the Processes the document

Collaboration diagram for OSXJV.Classes.ProcessDocument:



Public Member Functions

- Node Process ()

  Single Threaded Process.
- Node ProcessParallel (int pCount=4)

  Parse Document Using Multiple Threads

Static Public Member Functions

- static ProcessDocument GetProcess (string data, string type)

  Gets an instance of the ProcessDocument and prepare object.

Private Member Functions

- ProcessDocument (XDocument doc, string type)

  Constructor
- void ProcessComment (XComment e, Node node)

  Extract Comment
- void ProcessText (XText e, Node n)

  Get text from the data
- Node ProcessElement (XElement e, Node node)

  Single Threaded Process Element Version
- Node ProcessElement (XElement e, Node node, ref int nodeNumber)

  Multi-Threaded Version to process element
- Node ProcessRoot (XElement e, Node node)

  Processes first element in the document.
- void ProcessDocumentParallelInit (XDocument doc, int start)

  Method that each thread uses to process the document

Static Private Member Functions

- static XDocument Prepare (string data, string type)

  Prepares the object with setting the XDocument object to process

Private Attributes

- XDocument document

  Object the contains the parsed data ready to be processed.
- Node node = new Node()

  The Initial Node.
- List< Tuple< Node, int > > ProcessedElements = new List<Tuple<Node, int>>()

  Used with threading to keep list of processed Nodes.
- List< Thread > ThreadList = new List<Thread>()

  Used with threading to keep list of running threads.
- string type

  Document Type.
- Thread th
- int count

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

5.7.1 Detailed Description

Class the Processes the document

Definition at line 15 of file ProcessDocument.cs.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 ProcessDocument()

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

Constructor

Parameters

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

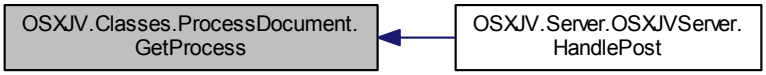
Returns

Definition at line 77 of file ProcessDocument.cs.

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

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

Here is the caller graph for this function:



5.7.3.2 Prepare()

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

Prepares the object with setting the XDocument object to process

Parameters

data	String of data
type	Data type

Returns

A XDocument object

Definition at line 111 of file ProcessDocument.cs.

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

5.7.3.3 Process()

```
Node OSXJV.Classes.ProcessDocument.Process ( )
```

Single Threaded Process.

Returns

Object of Nodes

Definition at line 127 of file ProcessDocument.cs.

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

5.7.3.4 ProcessComment()

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

Extract Comment

Parameters

e	Comment object to be parsed
node	Node to input data

Definition at line 64 of file ProcessDocument.cs.

References OSXJV.Classes.Node.Comments.

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

5.7.3.5 ProcessDocumentParallelInit()

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

Method that each thread uses to process the document

Parameters

doc	A subset of the full document
start	Start index number

Definition at line 338 of file ProcessDocument.cs.

References OSXJV.Classes.Node.Children.

```
00339         {
00340             int nodeNum = start;
00341 
00342             Node node = new Node();
00343             if (doc.Root.Nodes() != null)
00344             {
00345                 List<XNode> list = doc.Root.Nodes().ToList();
00346                 foreach (XNode n in doc.Root.Nodes())
00347                 {
00348                     switch (n.NodeType)
00349                     {
00350                         case System.Xml.XmlNodeType.Element:
00351                             nodeNum++;
00352                             Node n2 = new Node();
00353                             node.Children.Add(ProcessElement(XElement.Parse(n.ToString()), n2
    , ref nodeNum));
00354                             break;
00355                         case System.Xml.XmlNodeType.Comment:
00356                             ProcessComment(n as XComment, node);
00357                             break;
00358                         case System.Xml.XmlNodeType.Text:
00359                             ProcessText(n as XText, node);
00360                             break;
00361                         case System.Xml.XmlNodeType.Notation:
00362                             break;
00363                         case System.Xml.XmlNodeType.EndElement:
00364                             break;
00365                         default:
00366                             break;
00367                     }
00368                 }
00369             }
00370             document = null;
00371             ProcessedElements.Add(new Tuple<Node, int>(node, start));
00372         }
```

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.

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

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

5.7.3.7 ProcessElement() [2/2]

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

Multi-Threaded Version to process element

Parameters

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.

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

```
00258                           att.Value = ax.Value;
00259                           node.Attributes.Add(att);
00260                       }
00261                   }
00262
00263                   if (e.Nodes() != null)
00264                   {
00265                       List<XNode> list = e.Nodes().ToList();
00266
00267                       foreach (XNode n in e.Nodes())
00268                       {
00269                           switch (n.NodeType)
00270                           {
00271                               case System.Xml.XmlNodeType.EndElement:
00272                                   break;
00273                               case System.Xml.XmlNodeType.Element:
00274                                   nodeNumber++;
00275                                   Node n2 = new Node();
00276                                   node.Children.Add(ProcessElement(XElement.Parse(n
      .ToString()), n2, ref nodeNumber));
00277                                   break;
00278                               case System.Xml.XmlNodeType.Comment:
00279                                   ProcessComment(n as XComment, node);
00280                                   break;
00281                               case System.Xml.XmlNodeType.Text:
00282                                   ProcessText(n as XText, node);
00283                                   break;
00284                               case System.Xml.XmlNodeType.Notation:
00285                                   break;
00286
00287                               default:
00288                                   break;
00289                           }
00290                       }
00291                   }
00292                   node.Visited = true;
00293               }
00294               return node;
00295           }
```

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 379 of file ProcessDocument.cs.

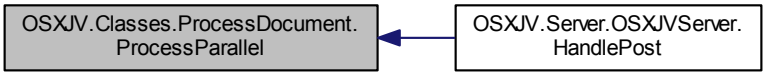
References OSXJV.Classes.Node.Children.

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

```
00380           {
00381               node = ProcessRoot(document.Root, node);
00382
00383               int nodeCount = document.Root.Nodes().Count();
```

```
00384
00385             if(nodeCount <= pCount)
00386             {
00387                 return Process();
00388             }
00389             else if (nodeCount > pCount)
00390             {
00391
00392                 List<XNode> List = document.Root.Nodes().ToList();
00393                 int spread = 0;
00394
00395                 spread = (int)Math.Ceiling((double)nodeCount / (double)pCount);
00396
00397                 int totalNodes = 1;
00398
00399                 for (int i = 0; i < pCount; i++)
00400                 {
00401                     int neg = 0;
00402                     int start = totalNodes;
00403                     if ((spread * (i+1)) > nodeCount)
00404                     {
00405                         neg = nodeCount - (spread * (i + 1));
00406                     }
00407
00408                     List<XNode> list = List.GetRange((spread * i), spread + neg);
00409                     XElement root = new XElement("Root", list);
00410                     XDocument doc = new XDocument(root);
00411
00412                     (th = new Thread(() => ProcessDocumentParallelInit(doc,
     start))).Start();
00413
00414                     ThreadList.Add(th); //Add to Threads list to keep recored of threads
      running
00415                     totalNodes += root.Descendants().Count(); //Increment start position.
00416                 }
00417                 document = null;
00418                 foreach (Thread t in ThreadList)
00419                 {
00420                     t.Join(); //Wait for threads to join
00421                 }
00422
00423                 ProcessedElements.Sort((x, y) => x.Item2.CompareTo(y.Item2)); //Sort List
      by start index so they are in order.
00424
00425                 foreach(Tuple<Node,int> tup in ProcessedElements)
00426                 {
00427                     foreach(Node n in tup.Item1.Children)
00428                     {
00429                         node.Children.Add(n);
00430                     }
00431                 }
00432             }
00433             return node;
00434         }
```

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 303 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.

```
00304         {
00305             node.Number = 1;
00306 
00307             if (!node.Visited)
00308             {
00309 
00310                 node.Name = e.Name.LocalName;
00311                 foreach (XAttribute ax in e.Attributes())
00312                 {
00313                     if (ax.Name == "id")
00314                     {
00315                         node.Name = node.Name + " #" + ax.Value;
00316                     }
00317 
00318                     if (type == "HTML")
00319                     {
00320                         if (ax.IsNamespaceDeclaration)
00321                             continue;
00322                     }
00323                     Attribute att = new Attribute();
00324                     att.Name = ax.Name.LocalName;
00325                     att.Value = ax.Value;
00326                     node.Attributes.Add(att);
00327                 }
00328             }
00329             node.Visited = true;
00330             return node;
00331         }
```

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.

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

5.7.4 Member Data Documentation

5.7.4.1 count

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

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

Definition at line 46 of file ProcessDocument.cs.

5.7.4.2 document

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

Object the contains the parsed data ready to be processed.

Definition at line 20 of file ProcessDocument.cs.

5.7.4.3 node

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

The Initial Node.

Definition at line 25 of file ProcessDocument.cs.

5.7.4.4 ProcessedElements

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

Used with threading to keep list of processed Nodes.

Definition at line 30 of file ProcessDocument.cs.

5.7.4.5 th

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

Definition at line 41 of file ProcessDocument.cs.

5.7.4.6 ThreadList

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

Used with threading to keep list of running threads.

Definition at line 35 of file ProcessDocument.cs.

5.7.4.7 type

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

Document Type.

Definition at line 40 of file ProcessDocument.cs.

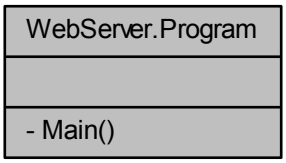
The documentation for this class was generated from the following file:

- OSXJVClasses/ProcessDocument.cs

5.8 WebServer.Program Class Reference

The Initialiser

Collaboration diagram for WebServer.Program:



Static Private Member Functions

- static void Main (string[ ] args)

  The Main function that starts the HttpServer

5.8.1 Detailed Description

The Initialiser

Definition at line 12 of file Program.cs.

5.8.2 Member Function Documentation

5.8.2.1 Main()

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

The Main function that starts the HttpServer

Parameters

args	Pass Cache Folder and Logger (Optional)

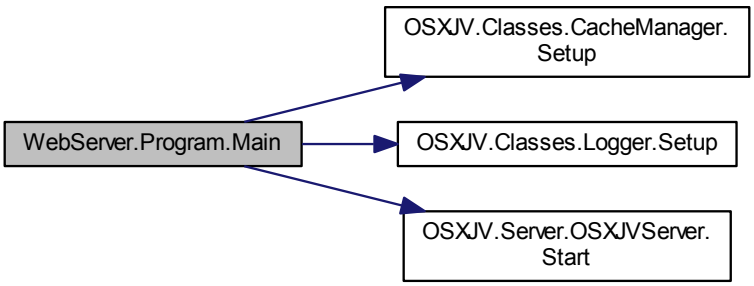
Definition at line 18 of file Program.cs.

References OSXJV.Classes.CacheManager.Setup(), OSXJV.Classes.Logger.Setup(), and OSXJV.Server.OSXJ↩VServer.Start().

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

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

Here is the call graph for this function:



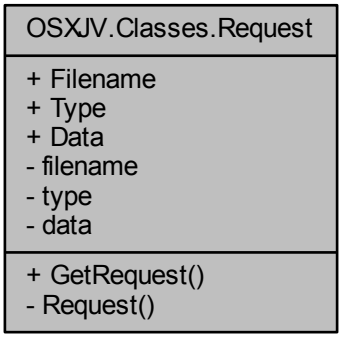
The documentation for this class was generated from the following file:

- Program.cs

5.9 OSXJV.Classes.Request Class Reference

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

Collaboration diagram for OSXJV.Classes.Request:



Static Public Member Functions

- static Request GetRequest (string filename, string type, string data)

  Creates an instance of Request.

Properties

- string Filename `[get, set]`

  To retrieve the filename of the document
- string Type `[get, set]`

  To retrieve type of document
- string Data `[get, set]`

  To retrieve the document data

Private Member Functions

- Request (string filename, string type, string data)

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

Private Attributes

- string filename

  Document Filename.
- string type

  Type of document.
- string data

  Contents of documents.

5.9.1 Detailed Description

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

Definition at line 8 of file Request.cs.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Request()

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

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

Parameters

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.

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

5.9.3 Member Function Documentation

5.9.3.1 GetRequest()

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

Creates an instance of Request.

Parameters

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

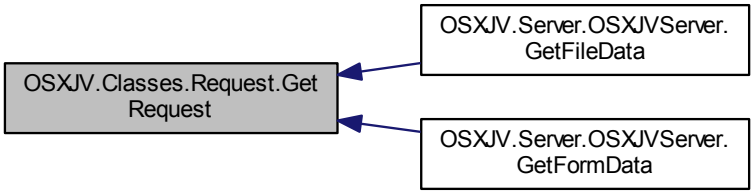
Object of Request

Definition at line 45 of file Request.cs.

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

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

Here is the caller graph for this function:



5.9.4 Member Data Documentation

5.9.4.1 data

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

Contents of documents.

Definition at line 23 of file Request.cs.

5.9.4.2 filename

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

Document Filename.

Definition at line 13 of file Request.cs.

5.9.4.3 type

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

Type of document.

Definition at line 18 of file Request.cs.

5.9.5 Property Documentation

5.9.5.1 Data

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

To retrieve the document data

Definition at line 101 of file Request.cs.

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

5.9.5.2 Filename

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

To retrieve the filename of the document

Definition at line 69 of file Request.cs.

5.9.5.3 Type

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

To retrieve type of document

Definition at line 85 of file Request.cs.

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

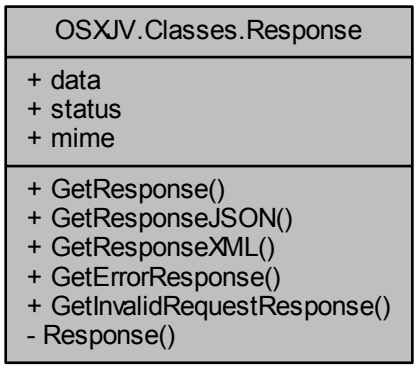
The documentation for this class was generated from the following file:

- OSXJVClasses/Request.cs

5.10 OSXJV.Classes.Response Class Reference

The Object containing data to send to the client

Collaboration diagram for OSXJV.Classes.Response:



Static Public Member Functions

- static Response GetResponse (int status, string type, byte[ ] data)

  A custom response object
- static Response GetResponseJSON (int status, byte[ ] data)

  Return an application/json response
- static Response GetResponseXML (int status, byte[ ] data)

  Return an text/xml response
- static Response GetErrorResponse (string message)

  Return a error response object
- static Response GetInvalidRequestResponse ()

  Returns an invalid response object

Public Attributes

- byte [ ] data = null

  Data
- int status

  Status Code
- string mime

  Data type e.g. "application/json"

Private Member Functions

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

  Constructor

5.10.1 Detailed Description

The Object containing data to send to the client

Definition at line 9 of file Response.cs.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 Response()

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

Constructor

Parameters

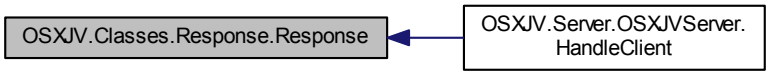
status	Status Code
mime	MIME type
buffer	Data

Definition at line 34 of file Response.cs.

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

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

Here is the caller graph for this function:



5.10.3 Member Function Documentation

5.10.3.1 GetErrorResponse()

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

Return a error response object

Returns

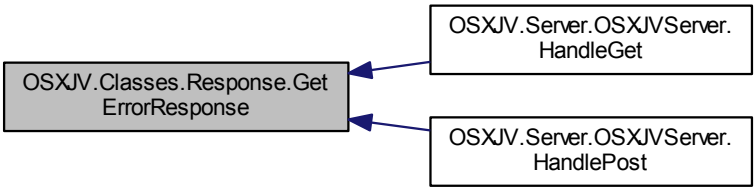
New response object

Definition at line 118 of file Response.cs.

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

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

Here is the caller graph for this function:



5.10.3.2 GetInvalidRequestResponse()

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

Returns an invalid response object

Returns

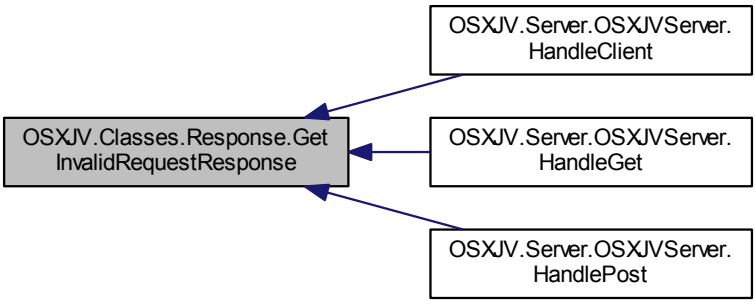
New response object

Definition at line 128 of file Response.cs.

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

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

Here is the caller graph for this function:



5.10.3.3 GetResponse()

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

A custom response object

Parameters

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

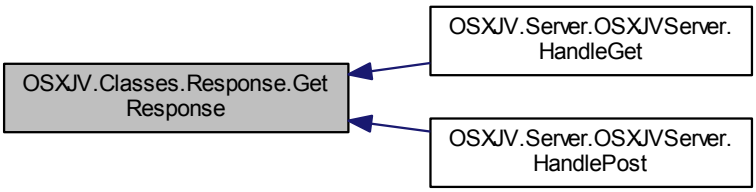
Returns

Definition at line 48 of file Response.cs.

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

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

Here is the caller graph for this function:



5.10.3.4 GetResponseJSON()

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

Return an application/json response

Parameters

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.

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

5.10.3.5 GetResponseXML()

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

Return an text/xml response

Parameters

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.

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

5.10.4 Member Data Documentation

5.10.4.1 data

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

Data

Definition at line 14 of file Response.cs.

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

5.10.4.2 mime

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

Data type e.g. "application/json"

Definition at line 24 of file Response.cs.

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

5.10.4.3 status

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

Status Code

Definition at line 19 of file Response.cs.

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

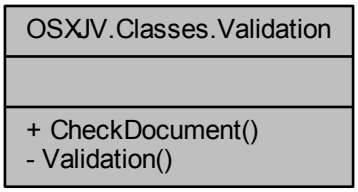
The documentation for this class was generated from the following file:

- OSXJVClasses/Response.cs

5.11 OSXJV.Classes.Validation Class Reference

Preform validation

Collaboration diagram for OSXJV.Classes.Validation:



Static Public Member Functions

- static bool CheckDocument (string data, string type)

  Checks the document and if it is valid

Private Member Functions

- Validation ()

  Constructor

5.11.1 Detailed Description

Preform validation

Definition at line 12 of file Validation.cs.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 Validation()

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

Constructor

Definition at line 17 of file Validation.cs.

```
00018         {
00019
00020         }
```

5.11.3 Member Function Documentation

5.11.3.1 CheckDocument()

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

Checks the document and if it is valid

Parameters

data	Document contents
type	Type of document

Returns

True if valid, else false

Exceptions

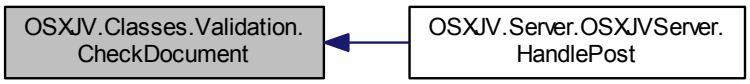
ArgumentException	Invalid data type or data and type cannot be null
XmlException	Invalid XML or HTML
JsonReaderException	Invalid JSON

Definition at line 31 of file Validation.cs.

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

```
00032         {
00033             if(string.IsNullOrEmpty(data) || string.IsNullOrEmpty(type))
00034             {
00035                 throw new ArgumentException("Data or Type cannot be Null");
00036             }
00037 
00038             if (type.Equals("XML") || type.Equals("HTML"))
00039             {
00040                 XmlReaderSettings settings = new XmlReaderSettings();
00041                 settings.DtdProcessing = DtdProcessing.Parse;
00042                 settings.MaxCharactersFromEntities = 2048;
00043                 using (XmlReader xr = XmlReader.Create(new StringReader(data),settings))
00044                 {
00045                     try
00046                     {
00047                         while (xr.Read()) { }
00048                         return true;
00049                     }
00050                     catch (XmlException ex)
00051                     {
00052                         throw ex;
00053                     }
00054                 }
00055             }
00056             else if(type.Equals("JSON"))
00057             {
00058                 try
00059                 {
00060                     JToken.Parse(data);
00061                     return true;
00062                 }
00063                 catch (JsonReaderException ex)
00064                 {
00065                     throw new JsonReaderException(ex.Message);
00066                 }
00067             }
00068 
00069             throw new ArgumentException("Invalid data or type");
00070         }
```

Here is the caller graph for this function:



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

- OSXJVClasses/Validation.cs

Chapter 6

File Documentation

6.1 OSXJVClasses/Attribute.cs File Reference

Classes

- class OSXJV.Classes.Attribute

Namespaces

- namespace OSXJV.Classes

6.2 Attribute.cs

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

6.3 OSXJVClasses/CacheManager.cs File Reference

Classes

- class OSXJV.Classes.CacheManager

  Manages Saving an Retrieving Filesexi

Namespaces

- namespace OSXJV.Classes

6.4 CacheManager.cs

```
00001 using System;
00002 using System.IO;
00003
00004 namespace OSXJV.Classes
00005 {
00009     public class CacheManager
00010     {
00011         private static CacheManager inst;
00012         private string path;
00013
00014         private CacheManager(string path)
00015         {
00016             this.path = path;
00017         }
00018
00023         public static bool Setup(string path)
00024         {
00025             if (string.IsNullOrEmpty(path))
00026                 throw new ArgumentException("Path cannot be empty");
00027
00028             if (!Directory.Exists(string.Format(@"{0}",path)))
00029                 throw new Exception("Path is not a valid cache directory");
00030
00031             return (inst = new CacheManager(path)) != null ? true : false;
00032         }
00033
00038         public static CacheManager GetInstance()
00039         {
00040             if (inst != null)
00041                 return inst;
00042             else
00043                 throw new Exception("CacheManger has not been setup");
00044         }
00045
00051         public string getFile(string ID)
00052         {
00053             if (string.IsNullOrEmpty(ID))
00054                 throw new ArgumentException("ID cannot be null or empty");
00055
00056             string filePath = path + "/" + ID.Replace("/","") + ".json";
00057             string output = "";
00058
00059             using (StreamReader sr = new StreamReader(filePath))
00060             {
00061                 output = sr.ReadToEnd();
00062             }
00063
00064             if (!string.IsNullOrEmpty(output))
00065                 return output;
00066             else
00067                 throw new Exception("Error Reading From File");
00068         }
00069
00075         public bool saveFile(string ID, string nodes)
00076         {
00077             if (string.IsNullOrEmpty(ID))
00078                 throw new ArgumentException("ID cannot be null or empty");
00079
00080             if (string.IsNullOrEmpty(nodes))
00081                 throw new ArgumentException("Document cannot be null or empty");
```

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

6.5 OSXJVClasses/Logger.cs File Reference

Classes

- class OSXJV.Classes.Logger

  A simple class that writes errors to a single file.

Namespaces

- namespace OSXJV.Classes

6.6 Logger.cs

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

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

6.7 OSXJVClasses/Node.cs File Reference

Classes

- class OSXJV.Classes.Node

  Contain Processed Document Information

Namespaces

- namespace OSXJV.Classes

6.8 Node.cs

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

```
00030         }
00031
00035         public int Number
00036         {
00037             get
00038             {
00039                 return number;
00040             }
00041
00042             set
00043             {
00044                 number = value;
00045             }
00046         }
00047
00051         public string Name
00052         {
00053             get
00054             {
00055                 return name;
00056             }
00057
00058             set
00059             {
00060                 name = value;
00061             }
00062         }
00063
00067         [JsonProperty(NullValueHandling =NullValueHandling.Ignore)]
00068         public string Value
00069         {
00070             get
00071             {
00072                 return value;
00073             }
00074
00075             set
00076             {
00077                 this.value = value;
00078             }
00079         }
00080
00084         public List<string> Comments
00085         {
00086             get
00087             {
00088                 return comments;
00089             }
00090
00091             set
00092             {
00093                 comments = value;
00094             }
00095         }
00096
00100         [JsonProperty()]
00101         public List<Attribute> Attributes
00102         {
00103             get
00104             {
00105                 return attributes;
00106             }
00107
00108             set
00109             {
00110                 attributes = value;
00111             }
00112         }
00113
00117         [JsonProperty()]
00118         public List<Node> Children
00119         {
00120             get
00121             {
00122                 return children;
00123             }
00124
00125             set
00126             {
00127                 children = value;
00128             }
00129         }
00130
00134         [Newtonsoft.Json.JsonIgnore]
00135         public bool Visited
00136         {
00137             get
```

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

6.9 OSXJVClasses/Output.cs File Reference

Classes

- class OSXJV.Classes.Output

  Creates the Output for the web page to display.

Namespaces

- namespace OSXJV.Classes

6.10 Output.cs

```
00001 using Newtonsoft.Json.Linq;
00002 using System;
00003 using System.Collections.Generic;
00004 using System.Threading;
00005
00006 namespace OSXJV.Classes
00007 {
00011     public class Output
00012     {
00016         private int left = 100, top = 130;
00017
00021         private Node nodes;
00022
00026         private bool GotParent = false;
00027
00031         private int Parent = 0;
00032
00036         private List<Tuple<int, string>> cNodes = new List<Tuple<int, string>>();
00037
00041         [ThreadStatic]
00042         private static int count = 0;
00043
00048         public Output(Node nodes)
00049         {
00050             if (nodes == null)
00051                 throw new ArgumentException();
00052             this.nodes = nodes;
00053         }
00054
00059         public JObject CreateGrid()
00060         {
00061             JObject obj = new JObject();
00062             obj.Add("text", nodes.Name);
00063             obj.Add("id", nodes.Number);
00064             obj.Add("state", new JObject(new JProperty("selected", true)));
00065
00066             if(nodes.Children.Count > 0)
00067             {
00068                 JArray array = new JArray();
00069                 foreach (Node n2 in nodes.Children)
00070                 {
00071                     array.Add(GridGetChidren(n2));
00072                 }
00073                 obj.Add("children", array);
00074             }
```

```
00075             return obj;
00076         }
00077
00083         private JObject GridGetChidren(Node n)
00084         {
00085             JObject child = new JObject();
00086             child.Add("id", n.Number);
00087             child.Add("text", n.Name);
00088
00089             if (n.Children.Count > 0)
00090             {
00091                 JArray array = new JArray();
00092                 foreach(Node n2 in n.Children)
00093                 {
00094                     array.Add(GridGetChidren(n2));
00095                 }
00096                 child.Add("children", array);
00097             }
00098             return child;
00099         }
00107         public string CreateViewSingle(int node, int nodeStart = 0)
00108         {
00109             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'>";
00110
00111             if (nodes.Number.Equals(node))
00112             {
00113                 int count = 0;
00114                 output += CreateNodeView(nodes, "node");
00115
00116
00117                 foreach (Node n in nodes.Children)
00118                 {
00119                     if(nodeStart > 0)
00120                     {
00121                         if (count != nodeStart)
00122                             continue;
00123                     }
00124                     count++;
00125                     output += CreateNodeView(n, "node-child"); //Child(Nodes) Thread
00126
00127                     if ((count-nodeStart) == 200)
00128                     {
00129                         output += CreateExtraNode("node-child",count);
00130                         break;
00131                     }
00132                 }
00133             }
00134
00135             else
00136             {
00137                 GetParent(nodes, node);
00138                 string temp = "";
00139                 if (GotParent)
00140                 {
00141                     if (nodes.Number == Parent)
00142                     {
00143                         output += CreateNodeView(nodes, "node-parent");
00144                     }
00145                 }
00146                 foreach (Node n2 in nodes.Children)
00147                 {
00148                     if (GotParent)
00149                     {
00150                         if (n2.Number == Parent)
00151                         {
00152                             output += CreateNodeView(n2, "node-parent");
00153                         }
00154                     }
00155                     temp += CheckChildren(n2, node);
00156                 }
00157                 if (!string.IsNullOrEmpty(temp))
00158                     output += temp;
00159             }
00160             output += "</div></div>"; //Close out divs
00161             return output;
00162         }
00163
00170         private string CreateExtraNode(string type,int id)
00171         {
00172             string node = "";
00173
00174             if (type == "node")
00175             {
00176                 if (GotParent)
00177                 {
00178                     left = left + 400;
```

```
00179                 }
00180             }
00181             if (type == "node-child")
00182             {
00183                 left = left + 400;
00184             }
00185 
00186             node += "<div class='node-child type ui-draggable ui-selectee' style='left:" + left + "px; top:
      " + top + "px;margin-bottom:50px;'>";
00187             node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
      )'>Show Lower</button></span></div>";
00188             node += "</div></div>";
00189             return node;
00190         }
00191 
00200         private string CreatePreviousNode(string type, int leftVal, int topVal, int id)
00201         {
00202             string node = "";
00203 
00204             if (type == "node")
00205             {
00206                 if (GotParent)
00207                 {
00208                     leftVal = leftVal + 400;
00209                 }
00210             }
00211             if (type == "node-child")
00212             {
00213                 leftVal = leftVal + 400;
00214             }
00215             node += "<div class='node-child type ui-draggable ui-selectee' style='left:" + leftVal + "px;
       top:" + topVal + "px;'>";
00216             node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
      )'>Show Higher</button></span></div>";
00217             node += "</div></div>";
00218             return node;
00219         }
00220 
00229         private string CreateExtraNode(string type, int leftVal, int topVal,int id)
00230         {
00231             string node = "";
00232 
00233             if (type == "node")
00234             {
00235                 if (GotParent)
00236                 {
00237                     leftVal = leftVal + 400;
00238                 }
00239             }
00240             if (type == "node-child")
00241             {
00242                 leftVal = leftVal + 400;
00243             }
00244             node += "<div class='node-child type ui-draggable ui-selectee' style='left:" + leftVal + "px;
       top:" + topVal + "px;margin-bottom:50px;'>";
00245             node += "<div class='head'><span><button class='nameBtn' onclick='GetMoreNodes(" + id + "
      )'>Show Lower</button></span></div>";
00246             node += "</div></div>";
00247             return node;
00248         }
00249 
00258         private void CreateNodeChildViewsParallel(List<Node> job,int start,
      bool showHigher,int next,int previous)
00259         {
00260             int threadID = int.Parse(Thread.CurrentThread.Name);
00261             string type = "node-child";
00262             string output = "";
00263 
00264             if(start == 0 && showHigher)
00265             {
00266                 output += CreatePreviousNode(type, left, top, previous);
00267             }
00268             bool hadCommentsPrev = false;
00269             int numCommentsPrevious = 0;
00270 
00271             foreach(Node n in job)
00272             {
00273                 int extra = showHigher ? 130 * (start +1) : 130 * start;
00274 
00275                 if (hadCommentsPrev)
00276                     extra += (numCommentsPrevious * 25);
00277 
00278                 if (n.Comments.Count > 0)
00279                 {
00280                     hadCommentsPrev = true;
00281                     numCommentsPrevious = n.Comments.Count;
00282                 }
```

```
00283                 else
00284                     hadCommentsPrev = false;
00285 
00286                 output += CreateNodeView(n, type,left,top + extra);
00287                 start++;
00288                 if (start == 200)
00289                 {
00290                     output += CreateExtraNode(type, left, top + extra + 130,next);
00291                     break;
00292                 }
00293 
00294             }
00295 
00296             cNodes.Add(new Tuple<int, string>(threadID, output));
00297         }
00298 
00306         public string CreateView(int node = 1,int pCount = 4,int nodeStart = 0) //Setting
      Defaults
00307         {
00308 
00309             List<Thread> threadList = new List<Thread>();
00310 
00311             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'>";
00312             if (nodes.Number.Equals(node))
00313             {
00314                 int childCount = 0;
00315 
00316                 if (nodes.Children.Count < 200)
00317                     childCount = nodes.Children.Count;
00318                 else
00319                 {
00320                     childCount = 200;
00321                 }
00322 
00323                 if(childCount < pCount * 2)
00324                 {
00325                     output += CreateNodeView(nodes, "node", left, top);
00326                     foreach(Node n2 in nodes.Children)
00327                     {
00328                         output += CreateNodeView(n2, "node-child");
00329                     }
00330                 }
00331                 else
00332                 {
00333                     int spread = (int)Math.Ceiling((double)childCount / (double)pCount);
00334 
00335                     output += CreateNodeView(nodes, "node",left,top);  //Parent(Node) Thread
00336 
00337                     for (int i = 0; i < pCount; i++)
00338                     {
00339                         int neg = 0;
00340                         if ((spread * (i + 1)) > childCount)
00341                         {
00342                             neg = childCount - (spread * (i + 1));
00343                         }
00344                         int start = (spread * i) ;
00345                         int rangeStart = (spread * i) + nodeStart;
00346                         bool showHigher = nodeStart != 0 ? true : false;
00347 
00348                         List<Node> NodesToProcess = nodes.Children.GetRange(rangeStart, spread +
      neg);
00349                         Thread threadJob = new Thread(() => CreateNodeChildViewsParallel(NodesToProcess,
      start, showHigher, childCount + nodeStart, nodeStart - childCount));
00350                         threadJob.Name = i.ToString();
00351                         threadJob.Start();
00352                         threadList.Add(threadJob);
00353                     }
00354                     foreach(Thread t in threadList)
00355                     {
00356                         t.Join();
00357                     }
00358 
00359                     cNodes.Sort((x, y) => x.Item1.CompareTo(y.Item1));
00360 
00361                     foreach(Tuple<int,string> tup in cNodes)
00362                     {
00363                         output += tup.Item2;
00364                     }
00365                 }
00366             }
00367             else
00368             {
00369                 GetParent(nodes, node);
00370                 string temp = "";
00371                 if (GotParent)
00372                 {
```

```
00373                     if (nodes.Number == Parent)
00374                     {
00375                         output += CreateNodeView(nodes, "node-parent");
00376                     }
00377                 }
00378                 bool found =false;
00379                 foreach (Node n2 in nodes.Children)
00380                 {
00381                     if (GotParent)
00382                     {
00383                         if (n2.Number == Parent)
00384                         {
00385                             output += CreateNodeView(n2, "node-parent");
00386                         }
00387                     }
00388                     temp += CheckChildren(n2, node,pCount,nodeStart,ref found);
00389                     if (found)
00390                         break;
00391                 }
00392                 if (!string.IsNullOrEmpty(temp))
00393                     output += temp;
00394             }
00395             output += "</div></div>";
00396             return output;
00397         }
00398
00405         private string CheckChildren(Node n, int number)
00406         {
00407             string output = "";
00408             if (CheckNodeNumber(n, number))
00409             {
00410
00411                 List<Thread> threadList = new List<Thread>();
00412
00413                 int count = 0;
00414                 output += CreateNodeView(n, "node");
00415                 foreach (Node n2 in n.Children)
00416                 {
00417                     count++;
00418                     output += CreateNodeView(n2, "node-child");
00419
00420                 }
00421             }
00422             else if (n.Children.Count > 0)
00423             {
00424                 foreach (Node n2 in n.Children)
00425                 {
00426                     if (GotParent)
00427                     {
00428                         if (n2.Number == Parent)
00429                         {
00430                             output += CreateNodeView(n2, "node-parent");
00431                         }
00432                     }
00433                     output += CheckChildren(n2, number);
00434                 }
00435             }
00436
00437             return output;
00438         }
00439
00449         private string CheckChildren(Node n, int number,int pCount, int nodeStart, ref
      bool found)
00450         {
00451             string output = "";
00452             if (CheckNodeNumber(n, number))
00453             {
00454                 found = true;
00455                 List<Thread> threadList = new List<Thread>();
00456
00457                 int count = 0;
00458                 output += CreateNodeView(n, "node");
00459                 count++;
00460                 //output += CreateNodeView(n2, "node-child");
00461                 int childCount = 0;
00462
00463                 if (n.Children.Count < 200)
00464                     childCount = n.Children.Count;
00465                 else
00466                 {
00467                     childCount = 200;
00468                 }
00469                 if (childCount < pCount * 2)
00470                 {
00471                     foreach(Node n2 in n.Children)
00472                     {
00473                         output += CreateNodeView(n2, "node-child");
```

```
00474                     }
00475                 }
00476                 else
00477                 {
00478                     int spread = (int)Math.Ceiling((double)childCount / (double)pCount);
00479 
00480                     if (childCount > 0)
00481                     {
00482                         for (int i = 0; i < pCount; i++)
00483                         {
00484                             int neg = 0;
00485                             if ((spread * (i + 1)) > childCount)
00486                             {
00487                                 neg = childCount - (spread * (i + 1));
00488                             }
00489                             int start = (spread * i);
00490                             int rangeStart = (spread * i) + nodeStart;
00491                             bool showHigher = nodeStart != 0 ? true : false;
00492 
00493                             List<Node> NodesToProcess = n.Children.GetRange(rangeStart, spread +
    neg);
00494 
00495                             if (NodesToProcess.Count > 0)
00496                             {
00497                                 Thread threadJob = new Thread(() => CreateNodeChildViewsParallel(
    NodesToProcess, start, showHigher, childCount + nodeStart, nodeStart - childCount));
00498                                 threadJob.Name = i.ToString();
00499                                 threadJob.Start();
00500                                 threadList.Add(threadJob);
00501                             }
00502                         }
00503                         foreach (Thread t in threadList)
00504                         {
00505                             t.Join();
00506                         }
00507                         cNodes.Sort((x, y) => x.Item1.CompareTo(y.Item1));
00508 
00509                         foreach (Tuple<int, string> tup in cNodes)
00510                         {
00511                             output += tup.Item2;
00512                         }
00513                     }
00514                 }
00515             }
00516             else if (n.Children.Count > 0)
00517             {
00518                 foreach (Node n2 in n.Children)
00519                 {
00520                     if (GotParent)
00521                     {
00522                         if (n2.Number == Parent)
00523                         {
00524                             output += CreateNodeView(n2, "node-parent");
00525                         }
00526                     }
00527                     output += CheckChildren(n2, number,pCount,nodeStart,ref found);
00528                 }
00529             }
00530 
00531             return output;
00532         }
00533 
00539         private void GetParent(Node node,int number)
00540         {
00541             if(!CheckNodeNumber(node,number))
00542             {
00543                 foreach(Node n in node.Children)
00544                 {
00545                     if(CheckNodeNumber(n, number))
00546                     {
00547                         Parent = node.Number;
00548                         GotParent = true;
00549                     }
00550                     else
00551                     {
00552                         GetParent(n, number);
00553                     }
00554                 }
00555             }
00556         }
00557 
00566         private string CreateNodeView(Node n, string type,int leftVal,int topVal)
00567         {
00568             string node = "";
00569 
00570             if(type == "node")
00571             {
```

```
00572                 if(GotParent)
00573                 {
00574                     leftVal = leftVal + 400;
00575                 }
00576             }
00577             if(type == "node-child")
00578             {
00579                 leftVal = leftVal + 400;
00580             }
00581             node += "<div id='" + n.Number + "'class='" + type + " type ui-draggable ui-selectee'
      style='left:" + leftVal + "px; top:" + topVal + "px;'>";
00582             node += "<div class='head'><span><button class='nameBtn' onclick='GetNode("+n.
      Number+")'>" + n.Name + "</button></span></div>";
00583             if (!string.IsNullOrEmpty(n.Value))
00584             {
00585                 node += string.Format("<div class='blockR'><p>Value</p></div><div
      class=comment><span>{0}</span></div>", n.Value);
00586             }
00587             if(n.Comments.Count >0)
00588             {
00589                 node += "<div><p class='text-center'>Comments</p></div>";
00590 
00591                 foreach(string com in n.Comments)
00592                 {
00593                     node += "<div class='comment'>" + com + "</div>";
00594                 }
00595             }
00596             if (n.Attributes.Count > 0)
00597             {
00598                 node += "<div class='attribute'><div class='aHeader'><p><button><i class='fa
      fa-plus'></i></button>Attributes</p></div><div class='options'>";
00599                 foreach (Attribute a in n.Attributes)
00600                 {
00601                     node += string.Format("<div class='blockR'><p>{0}</p></div><div
      class='comment'><p>{1}</p></div>", a.Name, a.Value);
00602                 }
00603                 node += "</div>";
00604             }
00605             node += "</div></div>";
00606             return node;
00607         }
00608 
00615         private string CreateNodeView(Node n, string type)
00616         {
00617             string node = "";
00618             int leftVal = left;
00619             if (type == "node")
00620             {
00621                 if (GotParent)
00622                 {
00623                     left = left + 400;
00624                     leftVal = left;
00625                 }
00626             }
00627             if (type == "node-child")
00628             {
00629                 leftVal = leftVal + 400;
00630             }
00631             node += "<div id='" + n.Number + "'class='" + type + " type ui-draggable ui-selectee'
      style='left:" + leftVal + "px; top:" + top + "px;'>";
00632             node += "<div class='head'><span><button class='nameBtn' onclick='GetNode(" + n.
      Number + ")'>" + n.Name + "</button></span></div>";
00633             if (!string.IsNullOrEmpty(n.Value))
00634             {
00635                 node += string.Format("<div class='blockR'><p>Value</p></div><div
      class=comment><span>{0}</span></div>", n.Value);
00636             }
00637             if (n.Comments.Count > 0)
00638             {
00639                 node += "<div><p class='text-center'>Comments</p></div>";
00640 
00641                 foreach (string com in n.Comments)
00642                 {
00643                     node += "<div class='comment'>" + com + "</div>";
00644                 }
00645             }
00646             if (n.Attributes.Count > 0)
00647             {
00648                 node += "<div class='attribute'><div class='aHeader'><p><button><i class='fa
      fa-plus'></i></button>Attributes</p></div><div class='options'>";
00649                 foreach (Attribute a in n.Attributes)
00650                 {
00651                     node += string.Format("<div class='blockR'><p>{0}</p></div><div
      class='comment'><p>{1}</p></div>", a.Name, a.Value);
00652                 }
00653                 node += "</div>";
00654             }
```

```
00655             node += "</div></div>";
00656
00657             if (type == "node-child")
00658             {
00659                 top = top + 130;
00660             }
00661             return node;
00662         }
00663
00670         private bool CheckNodeNumber(Node n, int number)
00671         {
00672             return n.Number.Equals(number);
00673         }
00674     }
00675 }
```

6.11 OSXJVClasses/ProcessDocument.cs File Reference

Classes

- class OSXJV.Classes.ProcessDocument

  Class the Processes the document

Namespaces

- namespace OSXJV.Classes

6.12 ProcessDocument.cs

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

```
00079             if (string.IsNullOrEmpty(data) || string.IsNullOrEmpty(type))
00080             {
00081                 throw new ArgumentException();
00082             }
00083             try
00084             {
00085                 XDocument doc = null;
00086                 doc = Prepare(data, type);
00087                 return new ProcessDocument(doc, type);
00088             }
00089             catch (System.Xml.XmlException e)
00090             {
00091                 throw e;
00092             }
00093         }
00094 
00100         private void ProcessText(XText e, Node n)
00101         {
00102             n.Value = e.Value;
00103         }
00104 
00111         private static XDocument Prepare(string data, string type)
00112         {
00113 
00114             if (type.Equals("JSON"))
00115                 return new XDocument(JsonConvert.DeserializeXNode(data, "Root", false).Root.FirstNode);
00116             else if (type.Equals("XML") || type.Equals("HTML"))
00117                 return XDocument.Parse(data);
00118 
00119             return null;
00120         }
00121 
00122 
00127         public Node Process()
00128         {
00129             if (document.Nodes() != null)
00130             {
00131                 foreach (XNode n in document.Nodes())
00132                 {
00133                     switch (n.NodeType)
00134                     {
00135                         case System.Xml.XmlNodeType.Element:
00136                             count++;
00137                             ProcessElement(XElement.Parse(n.ToString()), node);
00138                             break;
00139                         case System.Xml.XmlNodeType.Comment:
00140                             ProcessComment(n as XComment, node);
00141                             break;
00142                         case System.Xml.XmlNodeType.Text:
00143                             ProcessText(n as XText, node);
00144                             break;
00145                         case System.Xml.XmlNodeType.Notation:
00146                             break;
00147                         case System.Xml.XmlNodeType.EndElement:
00148                             break;
00149                         default:
00150                             break;
00151                     }
00152                 }
00153             }
00154             //SortArray(ref node);
00155             document = null;
00156             return node;
00157         }
00158 
00165         private Node ProcessElement(XElement e, Node node)
00166         {
00167             if (node.Number == 0)
00168             {
00169                 node.Number = count;
00170             }
00171             if (!node.Visited)
00172             {
00173 
00174                 node.Name = e.Name.LocalName;
00175                 foreach (XAttribute ax in e.Attributes())
00176                 {
00177                     if (ax.Name == "id")
00178                     {
00179                         node.Name = node.Name + " #" + ax.Value;
00180                     }
00181 
00182                     if (type == "HTML")
00183                     {
00184                         if (ax.IsNamespaceDeclaration)
00185                             continue;
00186                     }
```

```
00187                     Attribute att = new Attribute();
00188                     att.Name = ax.Name.LocalName;
00189                     att.Value = ax.Value;
00190                     node.Attributes.Add(att);
00191                 }
00192             }
00193 
00194             if (e.Nodes() != null)
00195             {
00196                 foreach (XNode n in e.Nodes())
00197                 {
00198                     switch (n.NodeType)
00199                     {
00200                         case System.Xml.XmlNodeType.EndElement:
00201                             break;
00202                         case System.Xml.XmlNodeType.Element:
00203                             count++;
00204                             Node n2 = new Node();
00205                             node.Children.Add(ProcessElement(XElement.Parse(n.ToString()), n2));
00206                             break;
00207                         case System.Xml.XmlNodeType.Comment:
00208                             ProcessComment(n as XComment, node);
00209                             break;
00210                         case System.Xml.XmlNodeType.Text:
00211                             ProcessText(n as XText, node);
00212                             break;
00213                         case System.Xml.XmlNodeType.Notation:
00214                             break;
00215 
00216                         default:
00217                             break;
00218                     }
00219                 }
00220             }
00221             node.Visited = true;
00222             return node;
00223         }
00224 
00232         private Node ProcessElement(XElement e, Node node, ref int nodeNumber)
00233         {
00234             if (!node.Visited)
00235             {
00236                 if (node.Number == 0)
00237                 {
00238                     node.Number = nodeNumber;
00239                 }
00240                 if (!node.Visited)
00241                 {
00242 
00243                     node.Name = e.Name.LocalName;
00244                     foreach (XAttribute ax in e.Attributes())
00245                     {
00246                         if (ax.Name == "id")
00247                         {
00248                             node.Name = node.Name + " #" + ax.Value;
00249                         }
00250 
00251                         if (type == "HTML")
00252                         {
00253                             if (ax.IsNamespaceDeclaration)
00254                                 continue;
00255                         }
00256                         Attribute att = new Attribute();
00257                         att.Name = ax.Name.LocalName;
00258                         att.Value = ax.Value;
00259                         node.Attributes.Add(att);
00260                     }
00261                 }
00262 
00263                 if (e.Nodes() != null)
00264                 {
00265                     List<XNode> list = e.Nodes().ToList();
00266 
00267                     foreach (XNode n in e.Nodes())
00268                     {
00269                         switch (n.NodeType)
00270                         {
00271                             case System.Xml.XmlNodeType.EndElement:
00272                                 break;
00273                             case System.Xml.XmlNodeType.Element:
00274                                 nodeNumber++;
00275                                 Node n2 = new Node();
00276                                 node.Children.Add(ProcessElement(XElement.Parse(n.ToString()), n2,
    ref nodeNumber));
00277                                 break;
00278                             case System.Xml.XmlNodeType.Comment:
00279                                 ProcessComment(n as XComment, node);
```

```
00280                                 break;
00281                             case System.Xml.XmlNodeType.Text:
00282                                 ProcessText(n as XText, node);
00283                                 break;
00284                             case System.Xml.XmlNodeType.Notation:
00285                                 break;
00286 
00287                             default:
00288                                 break;
00289                         }
00290                     }
00291                 }
00292                 node.Visited = true;
00293             }
00294             return node;
00295         }
00296 
00303         private Node ProcessRoot(XElement e, Node node)
00304         {
00305             node.Number = 1;
00306 
00307             if (!node.Visited)
00308             {
00309 
00310                 node.Name = e.Name.LocalName;
00311                 foreach (XAttribute ax in e.Attributes())
00312                 {
00313                     if (ax.Name == "id")
00314                     {
00315                         node.Name = node.Name + " #" + ax.Value;
00316                     }
00317 
00318                     if (type == "HTML")
00319                     {
00320                         if (ax.IsNamespaceDeclaration)
00321                             continue;
00322                     }
00323                     Attribute att = new Attribute();
00324                     att.Name = ax.Name.LocalName;
00325                     att.Value = ax.Value;
00326                     node.Attributes.Add(att);
00327                 }
00328             }
00329             node.Visited = true;
00330             return node;
00331         }
00332 
00338         private void ProcessDocumentParallelInit(XDocument doc,int start)
00339         {
00340             int nodeNum = start;
00341 
00342             Node node = new Node();
00343             if (doc.Root.Nodes() != null)
00344             {
00345                 List<XNode> list = doc.Root.Nodes().ToList();
00346                 foreach (XNode n in doc.Root.Nodes())
00347                 {
00348                     switch (n.NodeType)
00349                     {
00350                         case System.Xml.XmlNodeType.Element:
00351                             nodeNum++;
00352                             Node n2 = new Node();
00353                             node.Children.Add(ProcessElement(XElement.Parse(n.ToString()), n2, ref
    nodeNum));
00354                             break;
00355                         case System.Xml.XmlNodeType.Comment:
00356                             ProcessComment(n as XComment, node);
00357                             break;
00358                         case System.Xml.XmlNodeType.Text:
00359                             ProcessText(n as XText, node);
00360                             break;
00361                         case System.Xml.XmlNodeType.Notation:
00362                             break;
00363                         case System.Xml.XmlNodeType.EndElement:
00364                             break;
00365                         default:
00366                             break;
00367                     }
00368                 }
00369             }
00370             document = null;
00371             ProcessedElements.Add(new Tuple<Node, int>(node, start));
00372         }
00373 
00379         public Node ProcessParallel(int pCount = 4)
00380         {
00381             node = ProcessRoot(document.Root, node);
```

```
00382
00383             int nodeCount = document.Root.Nodes().Count();
00384
00385             if(nodeCount <= pCount)
00386             {
00387                 return Process();
00388             }
00389             else if (nodeCount > pCount)
00390             {
00391
00392                 List<XNode> List = document.Root.Nodes().ToList();
00393                 int spread = 0;
00394
00395                 spread = (int)Math.Ceiling((double)nodeCount / (double)pCount);
00396
00397                 int totalNodes = 1;
00398
00399                 for (int i = 0; i < pCount; i++)
00400                 {
00401                     int neg = 0;
00402                     int start = totalNodes;
00403                     if ((spread * (i+1)) > nodeCount)
00404                     {
00405                         neg = nodeCount - (spread * (i + 1));
00406                     }
00407
00408                     List<XNode> list = List.GetRange((spread * i), spread + neg);
00409                     XElement root = new XElement("Root", list);
00410                     XDocument doc = new XDocument(root);
00411
00412                     (th = new Thread(() => ProcessDocumentParallelInit(doc, start))).Start();
00413
00414                     ThreadList.Add(th); //Add to Threads list to keep recored of threads running
00415                     totalNodes += root.Descendants().Count(); //Increment start position.
00416                 }
00417                 document = null;
00418                 foreach (Thread t in ThreadList)
00419                 {
00420                     t.Join(); //Wait for threads to join
00421                 }
00422
00423                 ProcessedElements.Sort((x, y) => x.Item2.CompareTo(y.Item2)); //Sort List by start index so
      they are in order.
00424
00425                 foreach(Tuple<Node,int> tup in ProcessedElements)
00426                 {
00427                     foreach(Node n in tup.Item1.Children)
00428                     {
00429                         node.Children.Add(n);
00430                     }
00431                 }
00432             }
00433             return node;
00434         }
00435     }
00436 }
```

6.13 OSXJVClasses/Request.cs File Reference

Classes

- class OSXJV.Classes.Request

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

Namespaces

- namespace OSXJV.Classes

6.14 Request.cs

```
00001 using System;
00002
00003 namespace OSXJV.Classes
00004 {
00008     public class Request
00009     {
00013         private string filename;
00014
00018         private string type;
00019
00023         private string data;
00024
00031         private Request(string filename, string type, string data)
00032         {
00033             this.filename = filename;
00034             this.type = type;
00035             this.data = data;
00036         }
00037
00045         public static Request GetRequest(string filename, string type, string data)
00046         {
00047             string Type = "";
00048             if (string.IsNullOrEmpty(filename) || string.IsNullOrEmpty(type) || string.IsNullOrEmpty(data))
00049                 throw new ArgumentException();
00050             if (type.Equals("text/xml") || type.Equals("application/xml"))
00051             {
00052                 Type = "XML";
00053             }
00054             else if(type.Equals("text/html"))
00055             {
00056                 Type = "HTML";
00057             }
00058             else if (type.Equals("application/json") || type.Equals("application/octet-stream"))
00059             {
00060                 Type = "JSON";
00061             }
00062             return new Request(filename,Type,data);
00063         }
00064
00068         public string Filename
00069         {
00070             get
00071             {
00072                 return filename;
00073             }
00074
00075             set
00076             {
00077                 filename = value;
00078             }
00079         }
00080
00084         public string Type
00085         {
00086             get
00087             {
00088                 return type;
00089             }
00090
00091             set
00092             {
00093                 type = value;
00094             }
00095         }
00096
00100         public string Data
00101         {
00102             get
00103             {
00104                 return data;
00105             }
00106
00107             set
00108             {
00109                 data = value;
00110             }
00111         }
00112     }
00113 }
```

6.15 OSXJVClasses/Response.cs File Reference

Classes

- class OSXJV.Classes.Response

  The Object containing data to send to the client

Namespaces

- namespace OSXJV.Classes

6.16 Response.cs

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

```
00098         {
00099             if (status.Equals(null))
00100                 throw new ArgumentException("Status cannot be Null");
00101             else
00102                 if(status == 0)
00103                     throw new ArgumentException("Status cannot be 0");
00104 
00105             if (data == null)
00106                 throw new ArgumentException("Data cannot be null");
00107             else
00108                 if (data.Length == 0)
00109                     throw new ArgumentException("No data, use invalid or error response");
00110 
00111             return new Response(status, "text/xml", data);
00112         }
00113 
00118         public static Response GetErrorResponse(string message)
00119         {
00120             byte[] res = Encoding.UTF8.GetBytes(message);
00121             return new Response(400, "text/html", res);
00122         }
00123 
00128         public static Response GetInvalidRequestResponse()
00129         {
00130             return new Response(405, "text/html", new byte[0]);
00131         }
00132     }
00133 }
```

6.17 OSXJVClasses/Validation.cs File Reference

Classes

- class OSXJV.Classes.Validation

  Preform validation

Namespaces

- namespace OSXJV.Classes

6.18 Validation.cs

```
00001 using Newtonsoft.Json;
00002 using Newtonsoft.Json.Linq;
00003 using System;
00004 using System.IO;
00005 using System.Xml;
00006 
00007 namespace OSXJV.Classes
00008 {
00012     public class Validation
00013     {
00017         private Validation()
00018         {
00019 
00020         }
00021 
00031         public static bool CheckDocument(string data, string type)
00032         {
00033             if(string.IsNullOrEmpty(data) || string.IsNullOrEmpty(type))
00034             {
00035                 throw new ArgumentException("Data or Type cannot be Null");
00036             }
00037 
00038             if (type.Equals("XML") || type.Equals("HTML"))
00039             {
00040                 XmlReaderSettings settings = new XmlReaderSettings();
00041                 settings.DtdProcessing = DtdProcessing.Parse;
00042                 settings.MaxCharactersFromEntities = 2048;
00043                 using (XmlReader xr = XmlReader.Create(new StringReader(data),settings))
```

```
00044                 {
00045                     try
00046                     {
00047                         while (xr.Read()) { }
00048                         return true;
00049                     }
00050                     catch (XmlException ex)
00051                     {
00052                         throw ex;
00053                     }
00054                 }
00055             }
00056             else if(type.Equals("JSON"))
00057             {
00058                 try
00059                 {
00060                     JToken.Parse(data);
00061                     return true;
00062                 }
00063                 catch (JsonReaderException ex)
00064                 {
00065                     throw new JsonReaderException(ex.Message);
00066                 }
00067             }
00068 
00069             throw new ArgumentException("Invalid data or type");
00070         }
00071     }
00072 }
```

6.19 OSXJVServer.cs File Reference

Classes

- class OSXJV.Server.OSXJVServer

  HTTPServer that process the incoming requests.

Namespaces

- namespace OSXJV.Server

6.20 OSXJVServer.cs

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

```
00043         public bool Start()
00044         {
00045             serverThread = new Thread(new ThreadStart(Run));
00046             try
00047             {
00048                 serverThread.Start();
00049             }
00050             catch
00051             {}
00052             return serverThread.IsAlive;
00053         }
00054 
00058         public bool Stop()
00059         {
00060             if (listener != null)
00061                 if (listener.IsListening)
00062                     listener.Abort();
00063 
00064 
00065             if (serverThread != null)
00066             {
00067                 serverThread.Join();
00068                 serverThread = null;
00069             }
00070 
00071             return serverThread == null ?true:false;
00072         }
00076         public void Run()
00077         {
00078             running = true;
00079             listener.Start();
00080 
00081 
00082             while(listener.IsListening)
00083             {
00084 
00085                 Console.WriteLine("Waiting");
00086 
00087                 //Wait for Listener
00088                 IAsyncResult result = listener.BeginGetContext(new AsyncCallback(ListenerCallback),
      listener);
00089                 result.AsyncWaitHandle.WaitOne();
00090 
00091                 if (result.CompletedSynchronously)
00092                     Console.WriteLine("Completed Synchronously");
00093 
00094                 /*
00095                  * Old Method of Creating a Thread
00096                  *
00097                 Thread response = new Thread(() =>
00098                 {
00099                     try
00100                     {
00101                         Console.WriteLine("Processing");
00102                         HandleClient(hlc);
00103 
00104                         Console.WriteLine("Finished");
00105                     }
00106                     catch(Exception e)
00107                     {
00108                         Logger.GetInstance().WriteError(e.Message);
00109                         hlc.Response.StatusCode = 500;
00110                         hlc.Response.Close();
00111                     }
00112                 });
00113                 response.Start();
00114                 *
00115                 */
00116             }
00117         }
00118 
00119         //Asyncronous Handler
00124         private void ListenerCallback(IAsyncResult result)
00125         {
00126             HttpListener listener = (HttpListener)result.AsyncState;
00127             HttpListenerContext context = listener.EndGetContext(result);
00128             try
00129             {
00130                 HandleClient(context);
00131             }
00132             catch (Exception e)
00133             {
00134                 Logger.GetInstance().WriteError(e.Message);
00135                 context.Response.StatusCode = 500;
00136                 context.Response.Close();
00137             }
00138 
```

```
00139         }
00140
00141         //Handles the client request
00146         private void HandleClient(HttpListenerContext c)
00147         {
00148             switch(c.Request.HttpMethod)
00149             {
00150                 case "POST":
00151                     Post(HandlePost(c.Request),c.Response);
00152                     break;
00153                 case "GET":
00154                     Post(HandleGet(c.Request), c.Response);
00155                     break;
00156                 case "OPTIONS":
00157                     HandleOptions(c.Response);
00158                     c.Response.Close();
00159                     break;
00160                 default:
00161                     Post(Response.GetInvalidRequestResponse(), c.
      Response);
00162                     break;
00163             }
00164         }
00165
00170         private void HandleOptions(HttpListenerResponse response)
00171         {
00172             response.AddHeader("Access-Control-Allow-Headers", "Content-Type, Accept, X-Requested-With");
00173             response.AddHeader("Access-Control-Allow-Methods", "POST");
00174             response.AddHeader("Access-Control-Allow-Methods", "GET");
00175             response.AddHeader("Access-Control-Max-Age", "1728000");
00176             response.AppendHeader("Access-Control-Allow-Origin", "*");
00177         }
00178
00185         public Request GetFormData(Stream input)
00186         {
00187             string request = "";
00188             MultipartFormDataParser parser = new MultipartFormDataParser(input);
00189             if (parser.Files.Count > 0)
00190             {
00191                 using (StreamReader ms = new StreamReader(parser.Files[0].Data))
00192                 {
00193                     request = ms.ReadToEnd();
00194                 }
00195             }
00196             else
00197             {
00198                 throw new InvalidOperationException();
00199             }
00200             return Request.GetRequest(parser.Files[0].FileName, parser.Files[0].
      ContentType, request);
00201         }
00202
00209         private Request GetFileData(Stream input,string type)
00210         {
00211             string request = "";
00212             using (StreamReader ms = new StreamReader(input))
00213             {
00214                 request = ms.ReadToEnd();
00215             }
00216             string filename = "temp";
00217
00218             if (type == "text/xml")
00219                 filename += ".xml";
00220             else if(type == "application/json")
00221                 filename += ".json";
00222             else
00223                 filename += ".html";
00224
00225             return Request.GetRequest(filename,type, request);
00226         }
00227
00233         private Response HandlePost(HttpListenerRequest req)
00234         {
00235
00236             JObject eRes = new JObject();
00237
00238             if (SegmentNormalize(req.RawUrl).Equals("Process"))
00239             {
00240                 if (req.HasEntityBody)
00241                 {
00242
00243
00244                     Request r = null;
00245                     try
00246                     {
00247                         r = GetData(req);
00248                         if (r == null)
```

```
00249                           return Response.GetInvalidRequestResponse();
00250                   }
00251                   catch
00252                   {
00253                      return Response.GetInvalidRequestResponse();
00254                   }
00255
00256
00257
00258                   try
00259                   {
00260                       Validation.CheckDocument(r.Data, r.
      Type);
00261                   }
00262                   catch (Exception e)
00263                   {
00264                       eRes.Add("Error", e.Message);
00265                       return Response.GetErrorResponse(eRes.ToString());
00266                   }
00267
00268                   string id = Guid.NewGuid().ToString();
00269                   ProcessDocument pro = ProcessDocument.
      GetProcess(r.Data, r.Type);
00270                   Node n = pro.ProcessParallel();
00271                   Output o = new Output(n); //new output object
00272                   try
00273                   {
00274                       CacheManager.GetInstance().
      saveFile(id, JsonConvert.SerializeObject(n));
00275                       JObject response = new JObject();
00276
00277                       n = null; //remove node as its completed;
00278
00279                       response.Add("filename", id);
00280                       response.Add("grid", o.CreateGrid());
00281                       response.Add("view", o.CreateView());
00282
00283
00284
00285                       byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
00286                       return Response.GetResponse(200, "application/json", bytes);
00287                   }
00288                   catch (Exception e)
00289                   {
00290                       Logger.GetInstance().WriteError(e.Message);
00291                       eRes.Add("Error", "Error Creating Response");
00292                       return Response.GetErrorResponse(eRes.ToString());
00293                   }
00294
00295               }
00296               eRes.Add("Error", "No File Recieved By Server");
00297               return Response.GetErrorResponse(eRes.ToString());
00298           }
00299           else if (req.RawUrl.Equals("/Output"))
00300           {
00301               return Response.GetInvalidRequestResponse();
00302           }
00303           else
00304               return Response.GetInvalidRequestResponse();
00305       }
00306
00312       private Response HandleGet(HttpListenerRequest req)
00313       {
00314           if (SegmentNormalize(req.Url.Segments[1]).Equals("Process"))
00315           {
00316               if (req.Url.Segments.Length == 4)
00317               {
00318
00319                   Node cached;
00320                   try
00321                   {
00322                       cached = JsonConvert.DeserializeObject<Node>(
      CacheManager.GetInstance().getFile(req.Url.Segments[2]));
00323                   }
00324                   catch (Exception e)
00325                   {
00326                       Logger.GetInstance().WriteError(e.Message);
00327                       JObject eRes = new JObject();
00328                       eRes.Add("Error", "Error Creating Response");
00329                       return Response.GetErrorResponse(eRes.ToString());
00330                   }
00331                   Output o = new Output(cached);
00332                   JObject response = new JObject();
00333                   response.Add("view", o.CreateView(int.Parse(req.Url.Segments[3])));
00334                   byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
00335                   return Response.GetResponse(200, "application/json", bytes);
```

```
00336                 }
00337                 else if (req.Url.Segments.Length == 5)
00338                 {
00339 
00340                     Node cached;
00341                     try
00342                     {
00343                         cached = JsonConvert.DeserializeObject<Node>(
      CacheManager.GetInstance().getFile(req.Url.Segments[2]));
00344                     }
00345                     catch (Exception e)
00346                     {
00347                         Logger.GetInstance().WriteError(e.Message);
00348                         JObject eRes = new JObject();
00349                         eRes.Add("Error", "Error Creating Response");
00350                         return Response.GetErrorResponse(eRes.ToString());
00351                     }
00352                     Output o = new Output(cached);
00353                     JObject response = new JObject();
00354                     response.Add("view", o.CreateView(int.Parse(SegmentNormalize(req.Url.Segments
      [3])), 4, int.Parse(SegmentNormalize(req.Url.Segments[4]))));
00355                     byte[] bytes = Encoding.UTF8.GetBytes(response.ToString());
00356                     return Response.GetResponse(200, "application/json", bytes);

00357                 }
00358                 else
00359                     return Response.GetInvalidRequestResponse();
00360             }
00361             //If it got here its an invalid response.
00362             return Response.GetInvalidRequestResponse();
00363         }
00364 
00371         private void SaveFile(string id, Node nodes)
00372         {
00373             if(nodes == null || string.IsNullOrEmpty(id))
00374             {
00375                 throw new ArgumentException();
00376             }
00377 
00378             try
00379             {
00380                 CacheManager.GetInstance().saveFile(id, JsonConvert.
      SerializeObject(nodes));
00381             }
00382             catch (Exception e)
00383             {
00384                 Logger.GetInstance().WriteError(e.Message);
00385             }
00386         }
00387 
00394         private void Post(Response res,HttpListenerResponse stream)
00395         {
00396             if (res == null || stream == null)
00397                 throw new ArgumentException("Response or Client Stream cannot be NULL");
00398 
00399             HandleOptions(stream);
00400             stream.ProtocolVersion = new Version(1, 1);
00401             stream.StatusCode = res.status;
00402             stream.ContentType = res.mime;
00403             stream.ContentLength64 = res.data.Length;
00404             stream.OutputStream.Write(res.data, 0, res.data.Length);
00405             stream.Close();
00406         }
00407 
00413         private Request GetData(HttpListenerRequest req)
00414         {
00415             Request r = null;
00416 
00417             if (req.ContentType.Contains("application/x-www-form-urlencoded"))
00418             {
00419                 r = GetFormData(req.InputStream);
00420             }
00421             else if (req.ContentType.Contains("application/json") || req.ContentType.Contains("
      application/oclet-stream"))
00422             {
00423                 r = GetFileData(req.InputStream, "application/json");
00424             }
00425             else if (req.ContentType.Contains("application/xml") || req.ContentType.Contains("text/xml"))
00426             {
00427                 r = GetFileData(req.InputStream, "text/xml");
00428             }
00429             return r;
00430         }
00431 
00437         private string SegmentNormalize(string input)
00438         {
00439             return input.Replace("/", "");
```

```
00440         }
00441     }
00442 }
```

6.21 Program.cs File Reference

Classes

- class WebServer.Program

  The Initialiser

Namespaces

- namespace WebServer

6.22 Program.cs

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

```
00060
00061                     string[] files = Directory.GetFiles(args[0]);
00062
00063                     foreach (string file in files)
00064                     {
00065                         if (File.GetLastAccessTime(file) < DateTime.Now.AddHours(-6.0))
00066                             File.Delete(file);
00067                     }
00068                 }
00069             }
00070         }
00071     }
00072 }
```

6.23 Properties/AssemblyInfo.cs File Reference

6.24 AssemblyInfo.cs

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

Index