```
1   /*
2    * Author : JP. Froelicher
3    * Description : Search games
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class SearchGames
12      {
13          [DataContract]
14          public class SearchLinks
15          {
16              [DataMember]
17              public string self { get; set; }
18
19              [DataMember]
20              public string next { get; set; }
21          }
22
23          [DataMember]
24          public SearchLinks _links { get; set; }
25
26          [DataMember]
27          public Game[] games { get; set; }
28
29      }
30  }
31
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Search streams
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class SearchStreams
12      {
13          [DataContract]
14          public class SearchLinks
15          {
16              [DataMember]
17              public string self { get; set; }
18
19              [DataMember]
20              public string next { get; set; }
21          }
22
23          [DataMember]
24          public SearchLinks _links { get; set; }
25
26          [DataMember]
27          public Stream[] streams { get; set; }
28
29      }
30  }
31
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Controller of sites
4    * Date : 30/05/2015
5    */
6   using System.Collections.Generic;
7   using WebMediaManager.Models;
8   using WebMediaManager.Views;
9
10  namespace WebMediaManager.Controllers
11  {
12      class SitesController
13      {
14          private PersonalInterface _pview;
15          private Model _model;
16          private VidForm _vview;
17
18          public VidForm Vview
19          {
20              get { return _vview; }
21              set { _vview = value; }
22          }
23
24          internal Model Model
25          {
26              get { return _model; }
27              set { _model = value; }
28          }
29
30          public PersonalInterface PView
31          {
32              get { return _pview; }
33              set { _pview = value; }
34          }
35
36          /// <summary>
37          /// Constructor with the personal view
38          /// </summary>
39          /// <param name="personalView">personal view</param>
40          /// <param name="model">model</param>
41          public SitesController(PersonalInterface personalView, Model model)
42          {
43              this.Model = model;
44              this.PView = personalView;
45          }
46
47          /// <summary>
48          /// Constructor with the video view
49          /// </summary>
50          /// <param name="view">video view</param>
51          /// <param name="model">model</param>
52          public SitesController(VidForm view, Model model)
53          {
```

```
54              this.Model = model;
55              this.Vview = view;
56          }
57
58          /// <summary>
59          /// Get the last videos
60          /// </summary>
61          /// <returns>list of lastest videos</returns>
62          public List<StreamingSite.SVideo> GetLastVideos()
63          {
64              return this.Model.GetLastVideos();
65          }
66
67          /// <summary>
68          /// Get the online streams
69          /// </summary>
70          /// <returns>list of online streams</returns>
71          public List<StreamingSite.SVideo> GetOnlineStreams()
72          {
73              return this.Model.GetOnlineStreams();
74          }
75
76          /// <summary>
77          /// Get the name of all sites
78          /// </summary>
79          /// <returns>array of names</returns>
80          public string[] GetNameSites()
81          {
82              return this.Model.GetNameSites();
83          }
84
85          /// <summary>
86          /// Get the username from a website
87          /// </summary>
88          /// <param name="nameSite">namesite username</param>
89          /// <returns>username</returns>
90          public string GetUserName(string nameSite)
91          {
92              return this.Model.GetUserName(nameSite);
93          }
94
95          /// <summary>
96          /// Get the accesstoken from a website
97          /// </summary>
98          /// <param name="nameSite">name site</param>
99          /// <returns>access token</returns>
100         public string GetAccessToken(string nameSite)
101         {
102             return this.Model.GetAccessToken(nameSite);
103         }
104
105         /// <summary>
106         /// Follow a channel
```

```
107         /// </summary>
108         /// <param name="channelName">channel name</param>
109         /// <param name="siteName">site of the channel</param>
110         public void Follow(string channelName, string siteName)
111         {
112             for (int i = 0; i < this.Model.ListSite.Count; i++)
113             {
114                 if (siteName == this.Model.ListSite[i].Name)
115                     this.Model.ListSite[i].FollowChannel(channelName);
116             }
117         }
118 
119         /// <summary>
120         /// Unfollow a channel
121         /// </summary>
122         /// <param name="channelName">channel name</param>
123         /// <param name="siteName">site of the channel</param>
124         public void UnFollow(string channelName, string siteName)
125         {
126             for (int i = 0; i < this.Model.ListSite.Count; i++)
127             {
128                 if (siteName == this.Model.ListSite[i].Name)
129                     this.Model.ListSite[i]. UnFollowChannel(channelName);
130             }
131         }
132 
133         /// <summary>
134         /// Search videos
135         /// </summary>
136         /// <param name="request">the request</param>
137         /// <param name="limit">nb of result</param>
138         /// <returns>List of videos</returns>
139         public List<StreamingSite.SVideo> SearchVideos(string request, int limit)
140         {
141             return this.Model.SearchVideos(request, limit);
142         }
143 
144         /// <summary>
145         /// Get videos from site
146         /// </summary>
147         /// <param name="videos">List videos</param>
148         /// <returns>List of list video</returns>
149         public List<List<StreamingSite.SVideo>> GetVideosFromSite(List<StreamingSite.
        SVideo> videos)
150         {
151             List<List<StreamingSite.SVideo>> result = null;
152 
153             if (videos.Count > 0)
154             {
155                 result = new List<List<StreamingSite.SVideo>>();
156                 result.Add(new List<StreamingSite.SVideo>());
157                 result[0].Add(videos[0]);
158 
```

```
159                 for (int i = 0; i < videos.Count; i++)
160                 {
161                     for (int j = 0; j < result.Count; j++)
162                     {
163                         if (videos[i].siteName == result[j][i].siteName)
164                         {
165                             result[j].Add(videos[i]);
166                         }
167                         else
168                         {
169                             result[j] = new List<StreamingSite.SVideo>();
170                             result[j].Add(videos[i]);
171                         }
172                     }
173                 }
174 
175                 result[0].RemoveAt(0);
176                 return result;
177             }
178 
179             return result;
180         }
181 
182         /// <summary>
183         /// Get channel followed from site
184         /// </summary>
185         /// <param name="channels">channel</param>
186         /// <returns>List of list channels</returns>
187         public List<List<StreamingSite.SChannel>> GetChannelsFollowedFromSite(List<
          StreamingSite.SChannel> channels)
188         {
189             List<List<StreamingSite.SChannel>> result = null;
190 
191             if (channels.Count > 0)
192             {
193                 result = new List<List<StreamingSite.SChannel>>();
194                 result.Add(new List<StreamingSite.SChannel>());
195                 result[0].Add(channels[0]);
196 
197                 for (int i = 0; i < channels.Count; i++)
198                 {
199                     for (int j = 0; j < result.Count; j++)
200                     {
201                         if (channels[i].siteName == result[j][i].siteName)
202                         {
203                             result[j].Add(channels[i]);
204                         }
205                         else
206                         {
207                             result[j] = new List<StreamingSite.SChannel>();
208                             result[j].Add(channels[i]);
209                         }
210                     }
```

```
211                 }
212
213                 result[0].RemoveAt(0);
214                 return result;
215             }
216
217             return result;
218         }
219
220         /// <summary>
221         /// Get the channel followed
222         /// </summary>
223         /// <returns></returns>
224         public List<StreamingSite.SChannel> GetChannelFollowed()
225         {
226             return this.Model.GetChannelsFollowed();
227         }
228
229         /// <summary>
230         /// Get online stream from site
231         /// </summary>
232         /// <param name="siteName">name site</param>
233         /// <returns></returns>
234         public List<StreamingSite.SVideo> GetOnlineStreamFromSite(string siteName)
235         {
236             List<StreamingSite.SVideo> allOnlineStreams = this.GetOnlineStreams();
237             List<StreamingSite.SVideo> result = new List<StreamingSite.SVideo>();
238
239             if (allOnlineStreams.Count > 0)
240             {
241                 for (int i = 0; i < allOnlineStreams.Count; i++)
242                 {
243                     if(allOnlineStreams[i].siteName == siteName)
244                     {
245                         result.Add(allOnlineStreams[i]);
246                     }
247                 }
248
249                 return result;
250             }
251
252             return null;
253         }
254
255         /// <summary>
256         /// Get last videos from site
257         /// </summary>
258         /// <param name="siteName">site name</param>
259         /// <returns>list of last videos</returns>
260         public List<StreamingSite.SVideo> GetLastVideosFromSite(string siteName)
261         {
262             List<StreamingSite.SVideo> allLastVideos = this.GetLastVideos();
263             List<StreamingSite.SVideo> result = new List<StreamingSite.SVideo>();
```

```
264
265             if(allLastVideos.Count > 0)
266             {
267                 for (int i = 0; i < allLastVideos.Count; i++)
268                 {
269                     if(allLastVideos[i].siteName == siteName)
270                     {
271                         result.Add(allLastVideos[i]);
272                     }
273                 }
274
275                 return result;
276             }
277
278             return null;
279         }
280
281         /// <summary>
282         /// Get link of connexion page
283         /// </summary>
284         /// <param name="siteName">site name</param>
285         /// <returns>link</returns>
286         public string GetLinkConnexionPage(string siteName)
287         {
288             for (int i = 0; i < this.Model.ListSite.Count; i++)
289             {
290                 if(this.Model.ListSite[i].Name == siteName)
291                 {
292                     return this.Model.ListSite[i].Auth.CreateUrlConnexion();
293                 }
294             }
295
296             return null;
297         }
298
299         /// <summary>
300         /// If the site is connected
301         /// </summary>
302         /// <param name="siteName">site name</param>
303         /// <returns>true or false</returns>
304         public bool SiteIsConnected(string siteName)
305         {
306             for (int i = 0; i < this.Model.ListSite.Count; i++)
307             {
308                 if(this.Model.ListSite[i].Name == siteName)
309                 {
310                     return this.Model.ListSite[i].Auth.IsConnected;
311                 }
312             }
313
314             return false;
315         }
316
```

```
317             /// <summary>
318             /// Get access token in string url
319             /// </summary>
320             /// <param name="url">url</param>
321             /// <returns>access token</returns>
322             public string GetAccessTokenInUrl(string url)
323             {
324                 return this.Model.GetAccessTokenInUrl(url);
325             }
326 
327             /// <summary>
328             /// Connect from site
329             /// </summary>
330             /// <param name="accessToken">access token</param>
331             /// <param name="nameSite">name site</param>
332             public void Connect(string accessToken, string nameSite)
333             {
334                 for (int i = 0; i < this.Model.ListSite.Count; i++)
335                 {
336                     if (nameSite == this.Model.ListSite[i].Name)
337                     {
338                         this.Model.ListSite[i].Connect(accessToken);
339                     }
340                 }
341             }
342 
343             /// <summary>
344             /// Disconnect from site
345             /// </summary>
346             /// <param name="nameSite">site name</param>
347             public void Disconnect(string nameSite)
348             {
349                 for (int i = 0; i < this.Model.ListSite.Count; i++)
350                 {
351                     if (nameSite == this.Model.ListSite[i].Name)
352                     {
353                         this.Model.ListSite[i].Disconnect();
354                     }
355                 }
356             }
357 
358             /// <summary>
359             /// Count the connected site
360             /// </summary>
361             /// <returns></returns>
362             public int CountConnectedSite()
363             {
364                 int result = 0;
365                 for (int i = 0; i < this.Model.ListSite.Count; i++)
366                 {
367                     if (this.Model.ListSite[i].Auth.IsConnected)
368                         result++;
369                 }
```

```
370
371             return result;
372         }
373
374         /// <summary>
375         /// Check the last videos
376         /// </summary>
377         /// <returns>new videos</returns>
378         public List<List<StreamingSite.SVideo>> CheckNotificationsLastVideos()
379         {
380             return this.Model.CheckNotificationsLastVideos();
381         }
382
383         /// <summary>
384         /// Check the online streams
385         /// </summary>
386         /// <returns>new online streams</returns>
387         public List<List<StreamingSite.SVideo>> CheckNotificationsOnlineStreams()
388         {
389             return this.Model.CheckNotificationsOnlineStreams();
390         }
391     }
392 }
393
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Stream
4    * Date : 16/04/2015
5    */
6   using System;
7   using System.Runtime.Serialization;
8
9   namespace WebMediaManager.Structures.STwitch
10  {
11      [DataContract]
12      public class Stream
13      {
14          [DataMember]
15          public long _id { get; set; }
16
17          [DataMember]
18          public string game { get; set; }
19
20          [DataMember]
21          public int viewers { get; set; }
22
23          [DataMember]
24          public float average_fps { get; set; }
25
26          [DataMember]
27          public int video_height { get; set; }
28
29          [DataMember]
30          public DateTime created_at { get; set; }
31
32          [DataMember]
33          public StreamLinks _links { get; set; }
34
35          [DataContract]
36          public class StreamLinks
37          {
38              [DataMember]
39              public string self { get; set; }
40          }
41
42          [DataMember]
43          public Image preview { get; set; }
44
45          [DataMember]
46          public Channel channel { get; set; }
47
48      }
49  }
50
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : StreamingSite class
4    * Date : 29/05/2015
5    */
6   using System;
7   using System.Collections.Generic;
8
9   namespace WebMediaManager.Models
10  {
11      public class StreamingSite
12      {
13
14          #region STRUCTURES
15          public struct SVideo
16          {
17              public string id;
18              public string videoName;
19              public string channelName;
20              public int nbViews;
21              public string description;
22              public string preview;
23              public string playerLink;
24              public DateTime createdAt;
25              public string link;
26              public bool live;
27              public string siteName;
28              public string url_irc;
29              public bool channelIsFollowed;
30          }
31
32          public struct SChannel
33          {
34              public string id;
35              public string channelName;
36              public string description;
37              public string logoLink;
38              public string headerLink;
39              public int nbTotalViews;
40              public int nbFollowers;
41              public DateTime createdAt;
42              public string siteName;
43          }
44          #endregion
45
46          #region PROPERTIES
47
48          private List<SVideo> _listLastVideos;
49          private List<SVideo> _listOnlineStreams;
50          private string _userName;
51          private Authentification _auth;
52          private string _name;
53          private List<SChannel> _listChannelsFollowed;
```

```
54
55          public List<SChannel> ListChannelsFollowed
56          {
57              get { return _listChannelsFollowed; }
58              set { _listChannelsFollowed = value; }
59          }
60
61          public string Name
62          {
63              get { return _name; }
64              set { _name = value; }
65          }
66
67          internal Authentification Auth
68          {
69              get { return _auth; }
70              set { _auth = value; }
71          }
72
73          public string UserName
74          {
75              get { return _userName; }
76              set { _userName = value; }
77          }
78
79          internal List<SVideo> ListLastVideos
80          {
81              get { return _listLastVideos; }
82              set { _listLastVideos = value; }
83          }
84
85          public List<SVideo> ListOnlineStreams
86          {
87              get { return _listOnlineStreams; }
88              set { _listOnlineStreams = value; }
89          }
90
91          #endregion
92
93          /// <summary>
94          /// Constructor
95          /// </summary>
96          public StreamingSite()
97          {
98              this.ListLastVideos = new List<SVideo>();
99          }
100
101         /// <summary>
102         /// Get video by id
103         /// </summary>
104         /// <param name="id">id video</param>
105         /// <returns></returns>
106         public virtual SVideo GetVideoById(string id)
```

```
107             {
108                 throw new NotImplementedException();
109             }
110 
111             /// <summary>
112             /// Get video ID by a video link
113             /// </summary>
114             /// <param name="link"></param>
115             /// <returns></returns>
116             public virtual string GetIdVideoByLink(string link)
117             {
118                 throw new NotImplementedException();
119             }
120 
121             /// <summary>
122             ///
123             /// </summary>
124             /// <returns></returns>
125             public virtual void UpdateLastVideo()
126             {
127                 throw new NotImplementedException();
128             }
129 
130             /// <summary>
131             /// Upload the online streams
132             /// </summary>
133             public virtual void UpdateOnlineStream()
134             {
135                 throw new NotImplementedException();
136             }
137 
138             /// <summary>
139             /// Get the last video released
140             /// </summary>
141             /// <returns>list of last video</returns>
142             public virtual List<SVideo> GetLastVideos()
143             {
144                 throw new NotImplementedException();
145             }
146 
147             /// <summary>
148             /// Get the online streams
149             /// </summary>
150             /// <returns></returns>
151             public virtual List<SVideo> GetOnlineStreams()
152             {
153                 throw new NotImplementedException();
154             }
155 
156             /// <summary>
157             /// Update the channels followed
158             /// </summary>
159             public virtual void UpdateChannelsFollowed()
```

```
160         {
161             throw new NotImplementedException();
162         }
163 
164         /// <summary>
165         /// Get list of channel followed
166         /// </summary>
167         /// <returns></returns>
168         public virtual List<SChannel> GetChannelFollowed()
169         {
170             throw new NotImplementedException();
171         }
172 
173         /// <summary>
174         /// Get video by channel
175         /// </summary>
176         /// <param name="channelName">name of channel</param>
177         /// <returns></returns>
178         public virtual List<SVideo> GetVideosByChannel(string channelName)
179         {
180             throw new NotImplementedException();
181         }
182 
183         /// <summary>
184         /// Get stream by channel
185         /// </summary>
186         /// <param name="channelName">name of channel</param>
187         /// <returns></returns>
188         public virtual List<SVideo> GetStreamsByChannel(string channelName)
189         {
190             throw new NotImplementedException();
191         }
192 
193         /// <summary>
194         /// Search videos
195         /// </summary>
196         /// <param name="request"></param>
197         /// <returns>list of videos</returns>
198         public virtual List<SVideo> SearchVideos(string request, int limit)
199         {
200             throw new NotImplementedException();
201         }
202 
203         /// <summary>
204         /// Search channels
205         /// </summary>
206         /// <param name="request"></param>
207         /// <returns></returns>
208         public virtual List<SChannel> SearchChannels(string request)
209         {
210             throw new NotImplementedException();
211         }
212 
```

```
213             /// <summary>
214             /// Follow channel
215             /// </summary>
216             /// <param name="channelName"></param>
217             public virtual void FollowChannel(string channelName)
218             {
219                 throw new NotImplementedException();
220             }
221 
222             /// <summary>
223             /// Unfollow channel
224             /// </summary>
225             /// <param name="channelName"></param>
226             public virtual void UnFollowChannel(string channelName)
227             {
228                 throw new NotImplementedException();
229             }
230 
231             /// <summary>
232             /// Connect to account
233             /// </summary>
234             /// <returns></returns>
235             public virtual void Connect(string access_token)
236             {
237                 throw new NotImplementedException();
238             }
239 
240             /// <summary>
241             /// Disconnect the account
242             /// </summary>
243             /// <returns></returns>
244             public virtual void Disconnect()
245             {
246                 throw new NotImplementedException();
247             }
248 
249             /// <summary>
250             /// Get the popular videos
251             /// </summary>
252             /// <returns></returns>
253             public virtual List<SVideo> GetPopularVideos()
254             {
255                 throw new NotImplementedException();
256             }
257 
258             /// <summary>
259             /// Get the playlists
260             /// </summary>
261             /// <returns></returns>
262             public virtual List<SVideo> GetPlaylists()
263             {
264                 throw new NotImplementedException();
265             }
```

```
266
267             /// <summary>
268             /// Check if the channel is followed
269             /// </summary>
270             /// <param name="channelName">channel name</param>
271             /// <returns>bool</returns>
272             public virtual bool CheckChannelIsFollowed(string channelName)
273             {
274                 for (int i = 0; i < this.ListChannelsFollowed.Count; i++)
275                 {
276                     if (ListChannelsFollowed[i].channelName == channelName)
277                         return true;
278                 }
279                 return false;
280             }
281         }
282     }
283
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Streams
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Streams
12      {
13          [DataMember]
14          public StreamsLinks _links { get; set; }
15
16          [DataContract]
17          public class StreamsLinks
18          {
19              [DataMember]
20              public string self { get; set; }
21
22              [DataMember]
23              public string channel { get; set; }
24          }
25
26          [DataMember]
27          public Stream stream { get; set; }
28
29          [DataMember]
30          public Stream[] streams { get; set; }
31      }
32  }
33
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Subscription
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Subscription
12      {
13          [DataMember]
14          public string _id { get; set; }
15
16          [DataMember]
17          public Users user { get; set; }
18
19          [DataMember]
20          public string created_at { get; set; }
21
22          [DataContract]
23          public class SubscriptionLinks
24          {
25              [DataMember]
26              public string self { get; set; }
27          }
28
29          [DataMember]
30          public SubscriptionLinks _links { get; set; }
31      }
32  }
33
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Subscriptions
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Subscriptions
12      {
13          [DataMember]
14          public int _total { get; set; }
15
16          [DataMember]
17          public Subscription[] subscriptions { get; set; }
18
19          [DataContract]
20          public class SubscriptionLinks
21          {
22              [DataMember]
23              public string self { get; set; }
24          }
25
26          [DataMember]
27          public SubscriptionLinks _links { get; set; }
28      }
29  }
30
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Summary
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Summary
12      {
13          [DataMember]
14          public int viewers { get; set; }
15
16          [DataContract]
17          public class SummaryLinks
18          {
19              [DataMember]
20              public string self { get; set; }
21          }
22
23          [DataMember]
24          public SummaryLinks _links { get; set; }
25
26          [DataMember]
27          public int channels { get; set; }
28      }
29  }
30
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Team
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Team
12      {
13          [DataMember]
14          public int _id { get; set; }
15
16          [DataMember]
17          public string name { get; set; }
18
19          [DataMember]
20          public string info { get; set; }
21
22          [DataMember]
23          public string display_name { get; set; }
24
25          [DataMember]
26          public string created_at { get; set; }
27
28          [DataMember]
29          public string updated_at { get; set; }
30
31          [DataMember]
32          public string logo { get; set; }
33
34          [DataMember]
35          public string banner { get; set; }
36
37          [DataMember]
38          public string background { get; set; }
39
40          [DataContract]
41          public class TeamsLinks
42          {
43              [DataMember]
44              public string self { get; set; }
45          }
46
47          [DataMember]
48          public TeamsLinks _links { get; set; }
49      }
50  }
51
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Teams
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Teams
12      {
13          [DataContract]
14          public class TeamLinks
15          {
16              [DataMember]
17              public string next { get; set; }
18
19              [DataMember]
20              public string self { get; set; }
21          }
22
23          [DataMember]
24          public TeamLinks _links { get; set; }
25
26          [DataMember]
27          public Team[] teams { get; set; }
28
29      }
30  }
31
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Twitch class
4    * Date : 30/05/2015
5    */
6   using System;
7   using System.Collections.Generic;
8   using System.Linq;
9   using System.Text;
10  using WebMediaManager.Structures.STwitch;
11
12  namespace WebMediaManager.Models.Sites
13  {
14      public class Twitch : StreamingSite
15      {
16          #region CONSTS
17          private const string GET_METHOD = "GET";
18          private const string URL_API = "https://api.twitch.tv/kraken/";
19          private const string URL_SITE = "http://www.twitch.tv/";
20          private const string URL_AUTH = "https://api.twitch.tv/kraken/oauth2/authorize";
21          private const string ACCEPT_HTTP_HEADER = "application/vnd.twitchtv.v3+json";
22          private const string CLIENT_ID = "9jfbie2pedk3xzoj3s53268v7fb4zds";
23          #endregion
24
25
26          /// <summary>
27          /// Constructor
28          /// </summary>
29          public Twitch() : base()
30          {
31              this.ListOnlineStreams = new List<SVideo>();
32              this.ListLastVideos = new List<SVideo>();
33              this.ListChannelsFollowed = new List<SChannel>();
34              this.Name = "Twitch";
35              string[] scopes = new string[4] {"user_read", "user_follows_edit",
                "chat_login", "channel_read"};
36              this.Auth = new Authentification(scopes, URL_AUTH, CLIENT_ID);
37
38          }
39
40          /// <summary>
41          /// Create a video stream
42          /// </summary>
43          /// <param name="stream">stream twitch</param>
44          /// <returns>SVideo</returns>
45          private SVideo CreateVideoStream
46              (Stream stream)
47          {
48              SVideo video = new SVideo();
49              video.videoName = stream.channel.status;
50              video.channelName = stream.channel.name;
51              video.description = this.CreateChannelDescription(stream.channel.name);
52              video.createdAt = stream.created_at;
```

```
53              video.id = stream._id.ToString();
54              video.nbViews = stream.viewers;
55              video.preview = stream.preview.medium;
56              video.playerLink = URL_SITE + stream.channel.name + "/popout";
57              video.link = URL_SITE + stream.channel.name;
58              video.live = true;
59              video.url_irc = "irc.twitch.tv";
60              video.siteName = this.Name;
61              video.channelIsFollowed = this.CheckChannelIsFollowed(stream.channel.name);
62              return video;
63          }
64
65          /// <summary>
66          /// Create a video
67          /// </summary>
68          /// <param name="video">video twitch</param>
69          /// <returns>SVideo</returns>
70          private SVideo CreateVideo(Video video)
71          {
72              SVideo sVideo = new SVideo();
73              sVideo.videoName = video.title;
74              sVideo.channelName = video.channel.name;
75              sVideo.description = video.description;
76              sVideo.createdAt = video.recroded_at;
77              sVideo.id = video._id;
78              sVideo.nbViews = video.view;
79              sVideo.preview = video.preview;
80              sVideo.playerLink = URL_SITE + video.channel.name + "/popout?videoId=" +
                video._id;
81              sVideo.link = URL_SITE + video.channel.name + "/" + video._id;
82              sVideo.live = false;
83              sVideo.url_irc = "irc.twitch.tv";
84              sVideo.siteName = this.Name;
85              sVideo.channelIsFollowed = this.CheckChannelIsFollowed(video.channel.name);
86              return sVideo;
87          }
88
89          /// <summary>
90          /// Create a channel
91          /// </summary>
92          /// <param name="channelFollowed">channel twitch</param>
93          /// <returns>channel</returns>
94          private SChannel CreateChannel(WebMediaManager.Structures.STwitch.Channel
            channelFollowed)
95          {
96              SChannel channel = new SChannel();
97              channel.channelName = channelFollowed.name;
98              channel.createdAt = channelFollowed.created_at;
99              channel.description = this.CreateChannelDescription(channelFollowed.name);
100             channel.headerLink = channelFollowed.profile_banner;
101             channel.id = channelFollowed.ToString();
102             channel.logoLink = channelFollowed.logo;
103             channel.nbFollowers = channelFollowed.followers;
```

```
104                 channel.nbTotalViews = channelFollowed.views;
105                 channel.siteName = this.Name;
106 
107                 return channel;
108             }
109 
110             /// <summary>
111             /// Create a channel description
112             /// </summary>
113             /// <param name="channelName">channel name</param>
114             /// <returns>description</returns>
115             private string CreateChannelDescription(string channelName)
116             {
117                 Panel[] panels = Curl.Deserialize<Panel[]>(Curl.SendRequest(
                    "https://api.twitch.tv/api/channels/"+channelName+"/panels", GET_METHOD,
                    ACCEPT_HTTP_HEADER));
118                 StringBuilder result = new StringBuilder();
119 
120                 for (int i = 0; i < panels.Count(); i++)
121                 {
122                     result.Append("<h1>"+panels[i].data.title+"</h1>");
123                     result.Append("<a href='" + panels[i].data.link + "'><img src='" +
                        panels[i].data.image + "'/></a>");
124                     result.Append("<p>"+panels[i].data.description+"</p>");
125                 }
126 
127                 return result.ToString();
128             }
129 
130             /// <summary>
131             /// Get video by id
132             /// </summary>
133             /// <param name="id">id video</param>
134             /// <returns>video</returns>
135             public override SVideo GetVideoById(string id)
136             {
137                 Video video = Curl.Deserialize<Video>(Curl.SendRequest(URL_API+"videos/"+id,
                     "GET", ACCEPT_HTTP_HEADER));
138 
139                 return this.CreateVideo(video);
140             }
141 
142             /// <summary>
143             /// Get video ID by a video link
144             /// </summary>
145             /// <param name="link"></param>
146             /// <returns></returns>
147             public override string GetIdVideoByLink(string link)
148             {
149                 string result = null;
150                 if (link != "")
151                 {
152                     string checkSite = link.Substring(0, URL_SITE.Length);
```

```
153                 string[] split = null;
154                 if (String.Compare(checkSite, URL_SITE) == 0)
155                 {
156                     split = link.Split('/');
157                     if (split.Count() == 6)
158                         result = split[4] + split[5];
159                     else
160                         result = split[4];
161                 }
162             }else
163             {
164                 result = null;
165             }
166             return result;
167         }
168 
169         /// <summary>
170         /// Update the list of online streams
171         /// </summary>
172         public override void UpdateLastVideo()
173         {
174             this.ListLastVideos = this.GetLastVideos();
175         }
176 
177         /// <summary>
178         /// Update the online streams
179         /// </summary>
180         public override void UpdateOnlineStream()
181         {
182             this.ListOnlineStreams = this.GetOnlineStreams();
183         }
184 
185         /// <summary>
186         /// Update the channels followed
187         /// </summary>
188         public override void UpdateChannelsFollowed()
189         {
190             this.ListChannelsFollowed = this.GetChannelFollowed();
191         }
192 
193         /// <summary>
194         /// Get the last video
195         /// </summary>
196         /// <returns></returns>
197         public override List<SVideo> GetLastVideos()
198         {
199             List<SVideo> result = new List<SVideo>();
200             Videos videos = Curl.Deserialize<Videos>(Curl.SendRequest(URL_API +
                "videos/followed", "GET", this.Auth.Access_token, ACCEPT_HTTP_HEADER));
201             for (int i = 0; i < videos.videos.Count(); i++)
202             {
203                 result.Add(this.CreateVideo(videos.videos[i]));
204             }
```

```
205             return result;
206         }
207 
208         /// <summary>
209         /// Get the online stream
210         /// </summary>
211         /// <returns></returns>
212         public override List<StreamingSite.SVideo> GetOnlineStreams()
213         {
214             Streams streamsOnlineFollowed = Curl.Deserialize<Streams>(Curl.SendRequest(
                URL_API + "streams/followed", GET_METHOD, this.Auth.Access_token,
                ACCEPT_HTTP_HEADER));
215 
216             List<SVideo> listVideos = new List<SVideo>();
217 
218             for (int i = 0; i < streamsOnlineFollowed.streams.Count(); i++)
219             {
220                 SVideo video = this.CreateVideoStream(streamsOnlineFollowed.streams[i]);
221                 listVideos.Add(video);
222             }
223 
224             return listVideos;
225         }
226 
227         /// <summary>
228         /// Get the followed channel
229         /// </summary>
230         /// <returns></returns>
231         public override List<SChannel> GetChannelFollowed()
232         {
233             Follows channelFollowed = Curl.Deserialize<Follows>(Curl.SendRequest(URL_API
                 + "users/" + this.UserName + "/follows/channels", GET_METHOD, this.Auth.
                Access_token, ACCEPT_HTTP_HEADER));
234 
235             List<SChannel> listChannels = new List<SChannel>();
236 
237             for (int i = 0; i < channelFollowed.follows.Count(); i++)
238             {
239                 SChannel channel = this.CreateChannel(channelFollowed.follows[i].channel
                    );
240                 listChannels.Add(channel);
241             }
242 
243             return listChannels;
244         }
245 
246         /// <summary>
247         /// Get the result of search video
248         /// </summary>
249         /// <param name="request">request</param>
250         /// <returns>video list</returns>
251         public override List<SVideo> SearchVideos(string request, int limit)
252         {
```

```
253             SearchStreams searchStreams = Curl.Deserialize<SearchStreams>(Curl.
                SendRequest(URL_API + "search/streams?q=" + request + "&limit="+limit.
                ToString(), GET_METHOD, ACCEPT_HTTP_HEADER));
254
255             List<SVideo> listVideos = new List<SVideo>();
256
257             for (int i = 0; i < searchStreams.streams.Count(); i++)
258             {
259                 SVideo video = this.CreateVideoStream(searchStreams.streams[i]);
260                 listVideos.Add(video);
261             }
262
263             return listVideos;
264         }
265
266         /// <summary>
267         /// Get the result of search channel
268         /// </summary>
269         /// <param name="request">request</param>
270         /// <returns>channel list</returns>
271         public override List<SChannel> SearchChannels(string request)
272         {
273             SearchChannels searchChannels = Curl.Deserialize<SearchChannels>(Curl.
                SendRequest(URL_API + "search/channels?q=" + request, GET_METHOD,
                ACCEPT_HTTP_HEADER));
274
275             List<SChannel> listChannels = new List<SChannel>();
276
277             for (int i = 0; i < searchChannels.channels.Count(); i++)
278             {
279                 SChannel channel = this.CreateChannel(searchChannels.channels[i]);
280                 listChannels.Add(channel);
281             }
282             return listChannels;
283         }
284
285         /// <summary>
286         /// To follow a channel
287         /// </summary>
288         /// <param name="channelName">channel name</param>
289         public override void FollowChannel(string channelName)
290         {
291             Curl.Deserialize<Follow>(Curl.SendRequest(URL_API + "users/" + this.UserName
                 + "/follows/channels/" + channelName, "PUT", this.Auth.Access_token,
                ACCEPT_HTTP_HEADER));
292             this.UpdateLastVideo();
293             this.UpdateChannelsFollowed();
294         }
295
296         /// <summary>
297         /// To unfollow a channel
298         /// </summary>
299         /// <param name="channelName">channel name</param>
```

```
300         public override void UnFollowChannel(string channelName)
301         {
302             Curl.Deserialize<Follow>(Curl.SendRequest(URL_API + "users/" + this.UserName
                + "/follows/channels/" + channelName, "DELETE", this.Auth.Access_token,
                ACCEPT_HTTP_HEADER));
303             this.UpdateLastVideo();
304             this.UpdateChannelsFollowed();
305         }
306 
307         /// <summary>
308         /// Get the popular streams
309         /// </summary>
310         /// <returns></returns>
311         public override List<SVideo> GetPopularVideos()
312         {
313             Streams streams = Curl.Deserialize<Streams>(Curl.SendRequest(URL_API +
                "streams/", GET_METHOD, ACCEPT_HTTP_HEADER));
314 
315             List<SVideo> listVideos = new List<SVideo>();
316 
317             for (int i = 0; i < streams.streams.Count(); i++)
318             {
319                 SVideo video = this.CreateVideoStream(streams.streams[i]);
320                 listVideos.Add(video);
321             }
322 
323             return listVideos;
324         }
325 
326         /// <summary>
327         /// Connnect to twitch
328         /// </summary>
329         /// <param name="access_token">access token</param>
330         public override void Connect(string access_token)
331         {
332             this.Auth.Access_token = access_token;
333             this.Auth.IsConnected = true;
334             this.UpdateOnlineStream();
335             this.UpdateLastVideo();
336             Users user = Curl.Deserialize<Users>(Curl.SendRequest(
                "https://api.twitch.tv/kraken/user", "GET", this.Auth.Access_token,
                ACCEPT_HTTP_HEADER));
337             this.UserName = user.name;
338         }
339 
340         /// <summary>
341         /// Disconnect
342         /// </summary>
343         public override void Disconnect()
344         {
345 
                //Curl.Deserialize<AuthResponse>(Curl.SendRequest("https://api.twitch.tv/krak
                en/oauth2/authorization/"+this.Auth.Client_id, "DELETE",
```

```
                this.Auth.Access_token, ACCEPT_HTTP_HEADER));
346             this.Auth.IsConnected = false;
347             this.Auth.Access_token = "";
348
349         }
350     }
351 }
352
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Users
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Users
12      {
13          [DataMember]
14          public string display_name { get; set; }
15
16          [DataMember]
17          public string _id { get; set; }
18
19          [DataMember]
20          public string name { get; set; }
21
22          [DataMember]
23          public string type { get; set; }
24
25          [DataMember]
26          public string bio { get; set; }
27
28          [DataMember]
29          public string created_at { get; set; }
30
31          [DataMember]
32          public string updated_at { get; set; }
33
34          [DataMember]
35          public string logo { get; set; }
36
37          [DataMember]
38          public string email { get; set; }
39
40          [DataMember]
41          public bool partnered { get; set; }
42
43          [DataMember]
44          public UsersLinks _links { get; set; }
45
46          [DataContract]
47          public class UsersLinks
48          {
49              [DataMember]
50              public string self { get; set; }
51          }
52      }
53  }
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Video
4    * Date : 16/04/2015
5    */
6   using System;
7   using System.Runtime.Serialization;
8
9   namespace WebMediaManager.Structures.STwitch
10  {
11      [DataContract]
12      class Video
13      {
14          [DataMember]
15          public string title { get; set; }
16
17          [DataMember]
18          public string description { get; set; }
19
20          [DataMember]
21          public long broadcast_id { get; set; }
22
23          [DataMember]
24          public string status { get; set; }
25
26          [DataMember]
27          public string tag_list { get; set; }
28
29          [DataMember]
30          public DateTime recroded_at { get; set; }
31
32          [DataMember]
33          public string _id { get; set; }
34
35          [DataMember]
36          public string game { get; set; }
37
38          [DataMember]
39          public int length { get; set; }
40
41          [DataMember]
42          public string preview { get; set; }
43
44          [DataMember]
45          public string url { get; set; }
46
47          [DataMember]
48          public int view { get; set; }
49
50          [DataMember]
51          public string broadcast_type { get; set; }
52
53          [DataContract]
```

```
54          public class VideoLinks
55          {
56              [DataMember]
57              public string self { get; set; }
58
59              [DataMember]
60              public string channel { get; set; }
61          }
62
63          [DataMember]
64          public VideoLinks _links { get; set; }
65
66          [DataMember]
67          public Channel channel { get; set; }
68
69      }
70  }
71
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Videos
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Videos
12      {
13          [DataMember]
14          public int _total { get; set; }
15
16          [DataContract]
17          public class VideosLinks
18          {
19              [DataMember]
20              public string self { get; set; }
21
22              [DataMember]
23              public string next { get; set; }
24          }
25
26          [DataMember]
27          public VideosLinks _links { get; set; }
28
29          [DataMember]
30          public Video[] videos { get; set; }
31      }
32  }
33
```

```
1   using ChatSharp;
2   using System;
3   using System.Collections.Generic;
4   using System.ComponentModel;
5   using System.Data;
6   using System.Drawing;
7   using System.Linq;
8   using System.Text;
9   using System.Threading;
10  using System.Threading.Tasks;
11  using System.Windows.Forms;
12  using WebMediaManager.Controllers;
13  using WebMediaManager.Models;
14
15  namespace WebMediaManager.Views
16  {
17      public partial class VidForm : Form
18      {
19          private SitesController _siteController;
20          private StreamingSite.SVideo _video;
21          private IrcClient _client;
22          private IrcUser _user;
23          private IrcChat _chat;
24          private ContainersController _containerController;
25
26          internal ContainersController ContainerController
27          {
28              get { return _containerController; }
29              set { _containerController = value; }
30          }
31
32          internal IrcChat Chat
33          {
34              get { return _chat; }
35              set { _chat = value; }
36          }
37
38          public IrcUser User
39          {
40              get { return _user; }
41              set { _user = value; }
42          }
43
44          public IrcClient Client
45          {
46              get { return _client; }
47              set { _client = value; }
48          }
49
50          public StreamingSite.SVideo Video
51          {
52              get { return _video; }
53              set { _video = value; }
```

```
54          }
55
56          internal SitesController SiteController
57          {
58              get { return _siteController; }
59              set { _siteController = value; }
60          }
61
62          public VidForm(StreamingSite.SVideo video, Model model)
63          {
64              InitializeComponent();
65              this.SiteController = new SitesController(this, model);
66              this.ContainerController = new ContainersController(this, model);
67              this.Video = video;
68              this.wbbPlayer.Url = new Uri(video.playerLink);
69              this.wbbDescription.DocumentText = video.description;
70              this.lblTitle.Text = video.videoName;
71              this.lblViews.Text = video.nbViews.ToString();
72              if (this.Video.live)
73              {
74                  this.Chat = new IrcChat(this.tbxChat, this.SiteController.GetUserName(
                    this.Video.siteName), this.SiteController.GetUserName(this.Video.
                    siteName), this.SiteController.GetAccessToken(this.Video.siteName), this
                    .Video);
75                  this.Chat.ConnectIrc();
76              }
77
78              this.SetBtnSubscribe();
79              this.SetContainersList();
80          }
81
82          public void SetBtnSubscribe()
83          {
84              if (this.Video.channelIsFollowed)
85              {
86                  this.btnSubscribes.Text = "Désabonner";
87                  this.btnSubscribes.BackColor = Color.Red;
88                  this.btnSubscribes.Click += (s, e) => this.btnUnFollow(s, e);
89              }
90              else
91              {
92                  this.btnSubscribes.Text = "S'abonner";
93                  this.btnSubscribes.BackColor = Color.DarkGreen;
94                  this.btnSubscribes.Click += (s, e) => this.btnFollow(s, e);
95              }
96
97          }
98
99          private void btnSendMsg_Click(object sender, EventArgs e)
100         {
101             if (tbxSendMsg.Text != "")
102             {
103                 this.Chat.SendMessage(tbxSendMsg.Text);
```

```
104             }
105         }
106 
107         private void VidForm_FormClosed(object sender, FormClosedEventArgs e)
108         {
109             //this.Chat.Quit();
110         }
111 
112         private void VidForm_FormClosing(object sender, FormClosingEventArgs e)
113         {
114             if(this.Video.live)
115                 this.Chat.Quit();
116 
117             this.Chat = null;
118         }
119 
120         private void btnFollow(object sender, EventArgs e)
121         {
122             this.Invoke((MethodInvoker)delegate
123             {
124                 Button newbtn = (Button)sender;
125                 newbtn.Text = "Désabonner";
126                 newbtn.BackColor = Color.Red;
127                 this.SiteController.Follow(this.Video.channelName, this.Video.siteName);
128             });
129         }
130 
131         private void btnUnFollow(object sender, EventArgs e)
132         {
133             this.Invoke((MethodInvoker)delegate
134             {
135                 Button newbtn = (Button)sender;
136                 newbtn.Text = "S'abonner";
137                 newbtn.BackColor = Color.DarkGreen;
138                 this.SiteController.UnFollow(this.Video.channelName, this.Video.siteName
                    );
139             });
140         }
141 
142         private void SetContainersList()
143         {
144            if (this.Video.live)
145            {
146                this.cbxCategory.Visible = false;
147                this.cbxPlaylist.Visible = false;
148                this.btnAddCategory.Visible = false;
149                this.btnAddPlaylist.Visible = false;
150            }
151            else
152            {
153                List<string> category = this.ContainerController.GetNamesCategory();
154                List<string> playlist = this.ContainerController.GetNamesPlaylist();
155 
```

```
156                 for (int i = 0; i < category.Count; i++)
157                 {
158                     cbxCategory.Items.Add(category[i]);
159                 }
160 
161                 for (int i = 0; i < playlist.Count; i++)
162                 {
163                     cbxPlaylist.Items.Add(playlist[i]);
164                 }
165             }
166         }
167 
168         private void btnSubscribes_Click(object sender, EventArgs e)
169         {
170 
171         }
172 
173         private void btnAddCategory_Click(object sender, EventArgs e)
174         {
175             this.ContainerController.AddVideo(this.Video, this.cbxCategory.SelectedItem.
                ToString());
176         }
177 
178         private void btnAddPlaylist_Click(object sender, EventArgs e)
179         {
180             this.ContainerController.AddVideo(this.Video, this.cbxPlaylist.SelectedItem.
                ToString());
181         }
182     }
183 }
184 
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Viewutils
4    * Date : 16/04/2015
5    */
6   using System;
7   using System.Drawing;
8   using System.Windows.Forms;
9   using WebMediaManager.Models;
10
11  namespace WebMediaManager.Views
12  {
13      static class ViewUtils
14      {
15          /// <summary>
16          /// Create preview videos
17          /// </summary>
18          /// <param name="mainPanel">panel</param>
19          /// <param name="video">video</param>
20          /// <param name="id_video_line">video in line</param>
21          /// <param name="counter_for_line">nb of line</param>
22          /// <param name="model">model</param>
23          /// <returns>Panel preview</returns>
24          public static Panel CreatePreview(Panel mainPanel, StreamingSite.SVideo video,
            int id_video_line, int counter_for_line, Model model)
25          {
26              Panel gPanel = new Panel();
27              gPanel.Size = new Size(200, 200);
28              gPanel.Location = new Point((205 * id_video_line) + 10, gPanel.Size.Height*
                counter_for_line);
29
30              gPanel.BackColor = Color.White;
31              gPanel.Click += (sender, e) => OnClickVideo(sender, e, video, model);
32
33              //Create the picture box
34              PictureBox imgPreview = new PictureBox();
35              imgPreview.Size = new Size(200, 122);
36              imgPreview.SizeMode = PictureBoxSizeMode.StretchImage;
37              imgPreview.Load(video.preview);
38              imgPreview.Location = new Point(0, 0);
39              imgPreview.Click += (sender, e) =>  OnClickVideo(sender, e, video, model);
40
41              //Create the label title
42              Label title = new Label();
43              title.Location = new Point(0, 125);
44              title.Text = video.videoName;
45              title.Size = new Size(200, 20);
46              title.Font = new Font("Segoe UI", 9f);
47              title.AutoEllipsis = true;
48
49              //Create label channel
50              Label channel = new Label();
51              channel.Location = new Point(0, 145);
```

```
52              channel.Text = video.channelName;
53              channel.Size = new Size(200, 20);
54
55              //Create label views
56              Label views = new Label();
57              views.Location = new Point(0, 165);
58              views.Text = video.nbViews.ToString();
59              views.Size = new Size(200, 20);
60
61              //Add component to panel
62              gPanel.Controls.Add(imgPreview);
63              gPanel.Controls.Add(title);
64              gPanel.Controls.Add(channel);
65              gPanel.Controls.Add(views);
66
67              mainPanel.Controls.Add(gPanel);
68
69              return mainPanel;
70          }
71
72          /// <summary>
73          /// On click preview video
74          /// </summary>
75          /// <param name="sender"></param>
76          /// <param name="e"></param>
77          /// <param name="video"></param>
78          /// <param name="model"></param>
79          private static void OnClickVideo(object sender, EventArgs e, StreamingSite.
            SVideo video, Model model)
80          {
81              CreateFormVideo(video, model);
82          }
83
84          /// <summary>
85          /// Create form video
86          /// </summary>
87          /// <param name="video"></param>
88          /// <param name="model"></param>
89          public static void CreateFormVideo(StreamingSite.SVideo video, Model model)
90          {
91              VidForm videoForm = new VidForm(video, model);
92              videoForm.Show();
93          }
94
95      }
96  }
97
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Authentification class
4    * Date : 20/05/2015
5    */
6   using System.Text;
7   using WebMediaManager.Structures;
8
9   namespace WebMediaManager.Models
10  {
11      class Authentification
12      {
13          private const string REDIRECT_URI =
            "https://froelicher.github.io/WebMediaManager/WebSite/";
14
15          private string[] _scopes;
16          private string _url_auth;
17          private string _client_id;
18          private string _client_secret;
19          private string _access_token;
20          private bool _isConnected;
21
22          public bool IsConnected
23          {
24              get { return _isConnected; }
25              set { _isConnected = value; }
26          }
27
28          public string Access_token
29          {
30              get { return _access_token; }
31              set { _access_token = value; }
32          }
33
34          public string Client_secret
35          {
36              get { return _client_secret; }
37              set { _client_secret = value; }
38          }
39
40          public string Client_id
41          {
42              get { return _client_id; }
43              set { _client_id = value; }
44          }
45
46          public string Url_auth
47          {
48              get { return _url_auth; }
49              set { _url_auth = value; }
50          }
51
52          public string[] Scopes
```

```
53          {
54              get { return _scopes; }
55              set { _scopes = value; }
56          }
57
58          public Authentification()
59          {
60              this.IsConnected = false;
61          }
62
63          /// <summary>
64          /// Constructor
65          /// </summary>
66          /// <param name="scopes"></param>
67          /// <param name="url_auth"></param>
68          public Authentification(string[] scopes, string url_auth, string client_id) :
            this(scopes, url_auth, client_id, null, null)
69          {
70              //no code ...
71          }
72
73          /// <summary>
74          /// Designated constructor
75          /// </summary>
76          /// <param name="scopes"></param>
77          /// <param name="url_auth"></param>
78          /// <param name="client_id"></param>
79          /// <param name="client_secret"></param>
80          /// <param name="state"></param>
81          public Authentification(string[] scopes, string url_auth, string client_id,
            string client_secret, string state)
82          {
83              this.Scopes = scopes;
84              this.Url_auth = url_auth;
85              this.Client_id = client_id;
86              this.Client_secret = client_secret;
87          }
88
89          /// <summary>
90          /// Create the url to log
91          /// </summary>
92          /// <returns></returns>
93          public string CreateUrlConnexion()
94          {
95              string resultUrl = "";
96              resultUrl = this.Url_auth+"?response_type=token&client_id="+this.Client_id+
                "&redirect_uri="+REDIRECT_URI+"&scope="+this.ConvertStringArrayToString(this
                .Scopes);
97              return resultUrl;
98          }
99
100         /// <summary>
101         /// Get the code
```

```
102             /// </summary>
103             /// <returns></returns>
104             public string GetAccessTokenByCode(string url, string code, string accept_header)
105             {
106                 AuthResponse authResponse = Curl.Deserialize<AuthResponse>(Curl.SendRequest(
                    url, "POST", accept_header));
107                 return authResponse.access_token;
108             }
109 
110             /// <summary>
111             /// String array to string
112             /// </summary>
113             /// <param name="array"></param>
114             /// <returns></returns>
115             private string ConvertStringArrayToString(string[] array)
116             {
117                 StringBuilder builder = new StringBuilder();
118                 foreach(string value in array)
119                 {
120                     builder.Append(value);
121                     builder.Append('+');
122                 }
123 
124                 builder.Remove(builder.Length - 1, 1);
125 
126                 return builder.ToString();
127             }
128         }
129     }
130 
```

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Runtime.Serialization;
5   using System.Text;
6   using System.Threading.Tasks;
7
8   namespace WebMediaManager.Structures
9   {
10      [DataContract]
11      class AuthResponse
12      {
13          [DataMember]
14          public string access_token { get; set; }
15
16          [DataMember]
17          public string token_type { get; set; }
18
19          [DataMember]
20          public string scope { get; set; }
21
22          [DataMember]
23          public string user { get; set; }
24      }
25  }
26
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Channel
4    * Date : 16/04/2015
5    */
6   using System;
7   using System.Collections.Generic;
8   using System.Runtime.Serialization;
9
10  namespace WebMediaManager.Structures.STwitch
11  {
12      [DataContract]
13      public class Channel
14      {
15          [DataMember]
16          public string mature { get; set; }
17
18          [DataMember]
19          public string status { get; set; }
20
21          [DataMember]
22          public string broadcaster_language { get; set; }
23
24          [DataMember]
25          public string display_name { get; set; }
26
27          [DataMember]
28          public string game { get; set; }
29
30          [DataMember]
31          public string delay { get; set; }
32
33          [DataMember]
34          public string language { get; set; }
35
36          [DataMember]
37          public int _id { get; set; }
38
39          [DataMember]
40          public string name { get; set; }
41
42          [DataMember]
43          public DateTime created_at { get; set; }
44
45          [DataMember]
46          public DateTime updated_at { get; set; }
47
48          [DataMember]
49          public string logo { get; set; }
50
51          [DataMember]
52          public string banner { get; set; }
53
```

```
54          [DataMember]
55          public string video_banner { get; set; }
56
57          [DataMember]
58          public string background { get; set; }
59
60          [DataMember]
61          public string profile_banner { get; set; }
62
63          [DataMember]
64          public string profile_banner_background_color { get; set; }
65
66          [DataMember]
67          public string partner { get; set; }
68
69          [DataMember]
70          public string url { get; set; }
71
72          [DataMember]
73          public int views { get; set; }
74
75          [DataMember]
76          public int followers { get; set; }
77
78          [DataMember]
79          public ChannelLinks _links { get; set; }
80
81          [DataContract]
82          public class ChannelLinks
83          {
84              [DataMember]
85              public string follows { get; set; }
86
87              [DataMember]
88              public string commercial { get; set; }
89
90              [DataMember]
91              public string stream_key { get; set; }
92
93              [DataMember]
94              public string chat { get; set; }
95
96              [DataMember]
97              public string features { get; set; }
98
99              [DataMember]
100             public string subscriptions { get; set; }
101
102             [DataMember]
103             public string editors { get; set; }
104
105             [DataMember]
106             public string teams { get; set; }
```

```
107
108             [DataMember]
109             public string videos { get; set; }
110         }
111     }
112 }
113
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Container class
4    * Date : 29/05/2015
5    */
6   using System;
7   using System.Collections.Generic;
8   using System.IO;
9   using System.Linq;
10
11  namespace WebMediaManager.Models
12  {
13      public class Container
14      {
15          #region CONST
16
17          #endregion
18
19          #region PROPERTIES
20          private string _name;
21          private List<StreamingSite.SVideo> _listVideos;
22          private string _filePath;
23
24          public string FilePath
25          {
26              get { return _filePath; }
27              set { _filePath = value; }
28          }
29
30          internal List<StreamingSite.SVideo> ListVideos
31          {
32              get { return _listVideos; }
33              set { _listVideos = value; }
34          }
35
36          public string Name
37          {
38              get { return _name; }
39              set { _name = value; }
40          }
41
42          #endregion
43
44          /// <summary>
45          /// Constructor
46          /// </summary>
47          /// <param name="name">constructor name</param>
48          public Container(string name)
49          {
50              this.Name = name;
51          }
52
53          /// <summary>
```

```
54          /// Set the path of category
55          /// </summary>
56          public void SetPathCategory()
57          {
58              this.FilePath = Path.Combine(Environment.GetFolderPath(Environment.
                SpecialFolder.ApplicationData), "WebMediaManager/Category.ini");
59          }
60
61          /// <summary>
62          /// Set the path of playlist
63          /// </summary>
64          public void SetPathPlaylist()
65          {
66              this.FilePath = Path.Combine(Environment.GetFolderPath(Environment.
                SpecialFolder.ApplicationData), "WebMediaManager/Playlist.ini");
67          }
68
69          /// <summary>
70          /// Fill the list videos
71          /// </summary>
72          /// <param name="videos">List videos</param>
73          public void FillListVideos(List<StreamingSite.SVideo> videos)
74          {
75              this.ListVideos = videos;
76          }
77
78          /// <summary>
79          /// Add container in file
80          /// </summary>
81          public bool AddContainer()
82          {
83              if (File.Exists(this.FilePath))
84              {
85                  if (!CheckExistContainer())
86                  {
87                      using (System.IO.StreamWriter file = new System.IO.StreamWriter(this
                        .FilePath, true))
88                      {
89                          file.WriteLine("[" + this.Name + "]");
90                          file.WriteLine("link=");
91                      }
92                      return true;
93                  }
94              }
95              return false;
96          }
97
98          /// <summary>
99          /// Check if the container exist
100         /// </summary>
101         /// <returns></returns>
102         private bool CheckExistContainer()
103         {
```

```
104             if (File.Exists(this.FilePath))
105             {
106                 string[] allVideos = File.ReadAllLines(this.FilePath);
107 
108                 for (int i = 0; i < allVideos.Length; i++)
109                 {
110                     if (allVideos[i] == "[" + this.Name + "]")
111                     {
112                         return true;
113                     }
114                 }
115             }
116 
117             return false;
118         }
119 
120         /// <summary>
121         /// Get video of container
122         /// </summary>
123         /// <returns>List of videos</returns>
124         public string[] GetVideos()
125         {
126             string[] videos = null;
127 
128             if (File.Exists(this.FilePath))
129             {
130                 string[] allVideos = File.ReadAllLines(this.FilePath);
131                 string stringVideos = "";
132 
133                 for (int i = 0; i < allVideos.Length; i++)
134                 {
135                     if (allVideos[i] == "[" + this.Name + "]")
136                     {
137                         stringVideos = allVideos[i + 1];
138                         break;
139                     }
140                 }
141 
142                 stringVideos = stringVideos.Substring("link=".Length);
143 
144                 videos = stringVideos.Split(';');
145                 if(videos[videos.Count()-1] == "")
146                     videos = videos.Take(videos.Count() - 1).ToArray();
147 
148             }
149             return videos;
150         }
151 
152         /// <summary>
153         /// Add a video in list and in ini file
154         /// </summary>
155         /// <param name="video">video</param>
156         public void AddVideo(StreamingSite.SVideo video)
```

```
157             {
158                 this.ListVideos.Add(video);
159 
160                 if(File.Exists(this.FilePath))
161                 {
162                     string[] lines = File.ReadAllLines(this.FilePath);
163 
164                     for (int i = 0; i < lines.Length; i++)
165                     {
166                         if(lines[i] == "["+ this.Name +"]")
167                         {
168                             lines[i + 1] = lines[i + 1].Replace(lines[i + 1], lines[i + 1] +
                                ";" + video.link);
169                         }
170                     }
171                     File.WriteAllLines(this.FilePath, lines);
172                 }
173             }
174 
175             /// <summary>
176             /// Delete video in list and in ini file
177             /// </summary>
178             /// <param name="video">video</param>
179             public void DeleteVideo(StreamingSite.SVideo video)
180             {
181                 this.ListVideos.Remove(video);
182 
183                 if(File.Exists(this.FilePath))
184                 {
185                     string[] lines = File.ReadAllLines(this.FilePath);
186 
187                     for (int i = 0; i < lines.Length; i++)
188                     {
189                         if(lines[i] == "["+ this.Name +"]")
190                         {
191                             lines[i + 1] = lines[i + 1].Replace(video.link+";", "");
192                         }
193                     }
194                     File.WriteAllLines(this.FilePath, lines);
195                 }
196             }
197         }
198     }
199 
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Controller of containers
4    * Date : 29/05/2015
5    */
6   using System;
7   using System.Collections.Generic;
8   using WebMediaManager.Models;
9   using WebMediaManager.Views;
10
11  namespace WebMediaManager.Controllers
12  {
13      class ContainersController
14      {
15          private PersonalInterface _view;
16          private VidForm _viewVideo;
17
18          public VidForm ViewVideo
19          {
20              get { return _viewVideo; }
21              set { _viewVideo = value; }
22          }
23
24          private Model _model;
25
26          internal Model Model
27          {
28              get { return _model; }
29              set { _model = value; }
30          }
31
32          public PersonalInterface View
33          {
34              get { return _view; }
35              set { _view = value; }
36          }
37
38          /// <summary>
39          /// Constructor with the video view
40          /// </summary>
41          /// <param name="view">video view</param>
42          /// <param name="model">model</param>
43          public ContainersController(VidForm view, Model model)
44          {
45              this.ViewVideo = view;
46              this.Model = model;
47          }
48
49          /// <summary>
50          /// Constructor with the personal view
51          /// </summary>
52          /// <param name="view">personal view</param>
53          /// <param name="model">model</param>
```

```
54          public ContainersController(PersonalInterface view, Model model)
55          {
56              this.View = view;
57              this.Model = model;
58          }
59
60          /// <summary>
61          /// Get names of categories
62          /// </summary>
63          /// <returns>list of names</returns>
64          public List<string> GetNamesCategory()
65          {
66              List<string> result = new List<string>();
67              Playlist test = new Playlist("test");
68
69              for (int i = 0; i < this.Model.ListContainer.Count; i++)
70              {
71                  if (!Object.ReferenceEquals(test.GetType(), this.Model.ListContainer[i].
                    GetType()))
72                  {
73                      result.Add(this.Model.ListContainer[i].Name);
74                  }
75              }
76              return result;
77          }
78
79          /// <summary>
80          /// Get names of playlists
81          /// </summary>
82          /// <returns>List of names</returns>
83          public List<string> GetNamesPlaylist()
84          {
85              List<string> result = new List<string>();
86              Playlist test = new Playlist("test"); ;
87
88              for (int i = 0; i < this.Model.ListContainer.Count; i++)
89              {
90                  if (Object.ReferenceEquals(test.GetType(), this.Model.ListContainer[i].
                    GetType()))
91                  {
92                      result.Add(this.Model.ListContainer[i].Name);
93                  }
94              }
95              return result;
96          }
97
98          /// <summary>
99          /// Create the containers file
100         /// </summary>
101         public void CreateFileContainers()
102         {
103             this.Model.CreateFileContainers();
104         }
```

```
105
106         /// <summary>
107         /// Open the containers files
108         /// </summary>
109         public void OpenFileContainers()
110         {
111             this.Model.OpenFileCategories();
112             this.Model.OpenFilePlaylists();
113         }
114
115         /// <summary>
116         /// Get videos of a container
117         /// </summary>
118         /// <param name="name">name of container</param>
119         /// <returns>List of videos</returns>
120         public List<StreamingSite.SVideo> GetVideosOfContainer(string name)
121         {
122             List<StreamingSite.SVideo> result = null;
123
124             for (int i = 0; i < this.Model.ListContainer.Count; i++)
125             {
126                 if (this.Model.ListContainer[i].Name == name)
127                 {
128                     result = this.Model.ListContainer[i].ListVideos;
129                 }
130             }
131
132             return result;
133         }
134
135         /// <summary>
136         /// Add video in container
137         /// </summary>
138         /// <param name="video">video</param>
139         /// <param name="containerName">name of container</param>
140         public void AddVideo(StreamingSite.SVideo video, string containerName)
141         {
142             for (int i = 0; i < this.Model.ListContainer.Count; i++)
143             {
144                 if (containerName == this.Model.ListContainer[i].Name)
145                 {
146                     this.Model.ListContainer[i].AddVideo(video);
147                 }
148             }
149         }
150
151         /// <summary>
152         /// Add a new container
153         /// </summary>
154         /// <param name="name">name of container</param>
155         /// <param name="playlist">if the container is a playlist</param>
156         public void AddContainer(string name, bool playlist)
157         {
```

```
158                 this.Model.AddContainer(name, playlist);
159             }
160         }
161     }
162
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : cURL library class
4    * Date : 16/04/2015
5    */
6
7   using Newtonsoft.Json;
8   using System;
9   using System.IO;
10  using System.Net;
11
12  namespace WebMediaManager.Models
13  {
14      public static class Curl
15      {
16          /// <summary>
17          /// Send request and get response html
18          /// </summary>
19          /// <param name="urlRequest">request url</param>
20          /// <param name="p_method">method to use</param>
21          /// <param name="p_access_token">user access token</param>
22          /// <param name="acceptHeader">the accept header html</param>
23          /// <returns>HttpWebResponse</returns>
24          public static Stream SendRequest(string urlRequest, string p_method, string
            p_access_token, string acceptHeader)
25          {
26              //Create a new http request
27              var httpWebRequest = (HttpWebRequest)WebRequest.Create(urlRequest);
28
29              //Init the http request
30              httpWebRequest.ContentType = "application/json";
31              httpWebRequest.Accept = acceptHeader;
32              httpWebRequest.Method = p_method;
33              httpWebRequest.Headers.Add("Authorization: OAuth " + p_access_token);
34               if (p_method == "PUT")
35                  httpWebRequest.ContentLength = 0;
36
37               //Create a http response
38               Stream httpResponse = httpWebRequest.GetResponse().GetResponseStream();
39
40               return httpResponse;
41          }
42
43          /// <summary>
44          /// Send request and get response html
45          /// </summary>
46          /// <param name="urlRequest">request url</param>
47          /// <param name="p_method">method to use</param>
48          /// <param name="acceptHeader">the accept header html</param>
49          /// <returns>HttpWebResponse</returns>
50          public static Stream SendRequest(string urlRequest, string p_method, string
            acceptHeader)
51          {
```

```
52              try
53              {
54                  //Create a new http request
55                  var httpWebRequest = (HttpWebRequest)WebRequest.Create(urlRequest);
56
57                  //Init the http request
58                  httpWebRequest.ContentType = "application/json";
59                  httpWebRequest.Accept = acceptHeader;
60                  httpWebRequest.Method = p_method;
61
62                  //Create a http response
63                  var httpResponse = httpWebRequest.GetResponse().GetResponseStream();
64
65                  return httpResponse;
66              }
67              catch(Exception e)
68              {
69                  System.Windows.Forms.MessageBox.Show(e.Message + " // " + e.Data);
70                  return null;
71              }
72          }
73
74          /// <summary>
75          /// Deserialize a http web response
76          /// </summary>
77          /// <typeparam name="T">Generic type</typeparam>
78          /// <param name="jsonContent">content json</param>
79          /// <returns>the object deserialized</returns>
80          public static T Deserialize<T>(Stream jsonContent)
81          {
82              var httpResponse = jsonContent;
83
84              if (httpResponse != null)
85              {
86                  //Read the response
87                  using (var streamReader = new StreamReader(httpResponse))
88                  {
89                      //Add to the generics variable the result
90                      T answer = JsonConvert.DeserializeObject<T>(streamReader.ReadToEnd
                        ());
91                      return answer;
92                  }
93              }
94
95              return default(T);
96          }
97
98          /// <summary>
99          /// Generate a stream from string
100         /// </summary>
101         /// <param name="s">string</param>
102         /// <returns>stream</returns>
103         public static Stream GenerateStreamFromString(string s)
```

```
104             {
105                 MemoryStream stream = new MemoryStream();
106                 StreamWriter writer = new StreamWriter(stream);
107                 writer.Write(s);
108                 writer.Flush();
109                 stream.Position = 0;
110                 return stream;
111             }
112         }
113     }
114
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Featured
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Featured
12      {
13          [DataMember]
14          public string image { get; set; }
15
16          [DataMember]
17          public string text { get; set; }
18
19          [DataMember]
20          public string title { get; set; }
21
22          [DataMember]
23          public bool sponsored { get; set; }
24
25          [DataMember]
26          public bool scheduled { get; set; }
27
28          [DataMember]
29          public Stream stream { get; set; }
30      }
31  }
32
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Featureds
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Featureds
12      {
13          [DataContract]
14          public class FeaturedLinks
15          {
16              [DataMember]
17              public string self { get; set; }
18
19              [DataMember]
20              public string next { get; set; }
21          }
22
23          public FeaturedLinks _links { get; set; }
24
25          [DataMember]
26          public Featured[] featureds { get; set; }
27
28      }
29  }
30
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Follow
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Follow
12      {
13          [DataMember]
14          public string created_at { get; set; }
15
16          [DataContract]
17          public class FollowsLinks
18          {
19              [DataMember]
20              public string self { get; set; }
21          }
22
23          [DataMember]
24          public FollowsLinks _links { get; set; }
25
26          [DataMember]
27          public string notifications { get; set; }
28
29          [DataMember]
30          public Channel channel { get; set; }
31
32      }
33  }
34
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Follows
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Follows
12      {
13          [DataMember]
14          public Follow[] follows { get; set; }
15      }
16  }
17
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Game
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Game
12      {
13          [DataMember]
14          public string name { get; set; }
15
16          [DataMember]
17          public int _id { get; set; }
18
19          [DataMember]
20          public int giantbomb_id { get; set; }
21
22          [DataMember]
23          public Image box { get; set; }
24
25          [DataMember]
26          public Image logo { get; set; }
27
28          [DataContract]
29          public class SearchImages
30          {
31              [DataMember]
32              public string thumb { get; set; }
33
34              [DataMember]
35              public string tiny { get; set; }
36
37              [DataMember]
38              public string small { get; set; }
39
40              [DataMember]
41              public string super { get; set; }
42
43              [DataMember]
44              public string medium { get; set; }
45
46              [DataMember]
47              public string icon { get; set; }
48
49              [DataMember]
50              public string screen { get; set; }
51          }
52
53          [DataMember]
```

```
54          public SearchImages images { get; set; }
55
56          [DataMember]
57          public int popularity { get; set; }
58      }
59  }
60
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Games
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Games
12      {
13          [DataMember]
14          public int _total { get; set; }
15
16          [DataMember]
17          public GamesLinks _links { get; set; }
18
19          [DataContract]
20          public class GamesLinks
21          {
22              [DataMember]
23              public string self { get; set; }
24
25              [DataMember]
26              public string next { get; set; }
27          }
28
29          [DataMember]
30          public Top[] top { get; set; }
31
32          [DataContract]
33          public class Top
34          {
35              [DataMember]
36              public int viewers { get; set; }
37
38              [DataMember]
39              public int channels { get; set; }
40
41              [DataMember]
42              public Game game { get; set; }
43          }
44
45      }
46  }
47
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Image
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      public class Image
12      {
13        [DataMember]
14        public string large { get; set; }
15
16        [DataMember]
17        public string medium { get; set; }
18
19        [DataMember]
20        public string small { get; set; }
21
22        [DataMember]
23        public string template { get; set; }
24
25      }
26  }
27
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Class of manage irc chat
4    * Date : 22/05/2015
5    */
6   using ChatSharp;
7   using ChatSharp.Events;
8   using System;
9   using System.Linq;
10  using System.Text;
11  using System.Windows.Forms;
12
13  namespace WebMediaManager.Models
14  {
15      class IrcChat
16      {
17          private TextBox _tbxChat;
18          private string _username;
19          private string _nickname;
20          private string _token;
21          private StreamingSite.SVideo _video;
22          private IrcClient _client;
23          private IrcUser _user;
24
25          public IrcUser User
26          {
27              get { return _user; }
28              set { _user = value; }
29          }
30
31          public IrcClient Client
32          {
33              get { return _client; }
34              set { _client = value; }
35          }
36
37          public StreamingSite.SVideo Video
38          {
39              get { return _video; }
40              set { _video = value; }
41          }
42
43          public string Token
44          {
45              get { return _token; }
46              set { _token = value; }
47          }
48
49          public string Nickname
50          {
51              get { return _nickname; }
52              set { _nickname = value; }
53          }
```

```
54
55          public string Username
56          {
57              get { return _username; }
58              set { _username = value; }
59          }
60
61          public TextBox TbxChat
62          {
63              get { return _tbxChat; }
64              set { _tbxChat = value; }
65          }
66
67          /// <summary>
68          /// Constructor
69          /// </summary>
70          /// <param name="textbox">textbox of chat</param>
71          /// <param name="username">username</param>
72          /// <param name="nickname">nickname</param>
73          /// <param name="token">token</param>
74          /// <param name="video">video</param>
75          public IrcChat(TextBox textbox, string username, string nickname, string token,
            StreamingSite.SVideo video)
76          {
77              this.TbxChat = textbox;
78              this.Username = username;
79              this.Nickname = nickname;
80              this.Token = token;
81              this.Video = video;
82          }
83
84          /// <summary>
85          /// Build message and filter
86          /// </summary>
87          /// <param name="message">msg</param>
88          public void BuildMessage(string message)
89          {
90              StringBuilder result = new StringBuilder();
91
92              if (message.Contains("PRIVMSG"))
93              {
94                  result.Append("<< ");
95                  result.Append(message.Substring(1, message.IndexOf('!') - 1));
96                  string[] split = message.Split();
97                  for (int i = 3; i < split.Count(); i++)
98                  {
99                      result.Append(split[i] + " ");
100                 }
101                 result.Append("\r\n");
102             }
103         }
104
105         /// <summary>
```

```
106         /// Event error
107         /// </summary>
108         /// <param name="sender"></param>
109         /// <param name="e"></param>
110         private void NetWorkError(object sender, SocketErrorEventArgs e)
111         {
112             try {
113                 if (this.TbxChat != null)
114                     this.TbxChat.Invoke(new MethodInvoker(delegate { this.TbxChat.
                        AppendText("Error: " + e.SocketError); }));
115             }
116             catch (ObjectDisposedException)
117             { }
118         }
119 
120         /// <summary>
121         /// Event message received
122         /// </summary>
123         /// <param name="sender"></param>
124         /// <param name="e"></param>
125         private void RawMessageReceived(object sender, RawMessageEventArgs e)
126         {
127             try{
128                 if(this.TbxChat != null)
129                     this.TbxChat.Invoke(new MethodInvoker(delegate { BuildMessage(e.
                        Message); }));
130             }catch(ObjectDisposedException)
131             { }
132         }
133 
134         /// <summary>
135         /// Event message sent
136         /// </summary>
137         /// <param name="sender"></param>
138         /// <param name="e"></param>
139         private void RawMessageSent(object sender, RawMessageEventArgs e)
140         {
141             try
142             {
143                 if (this.TbxChat != null && e.Message != "QUIT")
144                 {
145                     if (this.TbxChat != null)
146                         this.TbxChat.Invoke(new MethodInvoker(delegate { BuildMessage(e.
                            Message); }));
147                 }
148             }catch(ObjectDisposedException)
149             { }
150         }
151 
152         /// <summary>
153         /// Event channel message received
154         /// </summary>
155         /// <param name="sender"></param>
```

```
156         /// <param name="e"></param>
157         private void ChannelMessageReceived(object sender, PrivateMessageEventArgs e)
158         {
159             try
160             {
161                 if(this.TbxChat != null)
162                     this.TbxChat.Invoke(new MethodInvoker(delegate { this.TbxChat.
                        AppendText("<" + e.PrivateMessage.User.Nick + ">" + " " + e.
                        PrivateMessage.Message + "\n"); }));
163             }
164             catch (ObjectDisposedException)
165             { }
166         }
167 
168         /// <summary>
169         /// Event channel topic received
170         /// </summary>
171         /// <param name="sender"></param>
172         /// <param name="e"></param>
173         private void ChannelTopicReceived(object sender, ChannelTopicEventArgs e)
174         {
175             try
176             {
177                 if (this.TbxChat != null)
178                     this.TbxChat.Invoke(new MethodInvoker(delegate { this.TbxChat.
                        AppendText("Received topic for channel " + e.Channel.Name + ": " + e
                        .Topic + "\n"); }));
179             }
180             catch (ObjectDisposedException)
181             { }
182 
183         }
184 
185         /// <summary>
186         /// Connect to chat IRC
187         /// </summary>
188         public void ConnectIrc()
189         {
190             this.User = new IrcUser(this.Username, this.Nickname, "oauth:" + this.Token);
191             this.Client = new IrcClient(this.Video.url_irc, this.User);
192 
193             if (this.Client != null)
194             {
195 
196                 this.Client.ConnectionComplete += (s, e) => this.Client.JoinChannel("#"
                    + this.Video.channelName);
197 
198                 this.Client.NetworkError += (s, e) => this.NetWorkError(s, e);
199                 this.Client.RawMessageRecieved += (s, e) => this.RawMessageReceived(s, e
                    );
200                 this.Client.RawMessageSent += (s, e) => this.RawMessageSent(s, e);
201 
202                 this.Client.UserMessageRecieved += (s, e) =>
```

```
203                 {
204                     if (e.PrivateMessage.Message.StartsWith(".join "))
205                     {
206                         this.Client.Channels.Join(e.PrivateMessage.Message.Substring(6));
207                     }
208                     else if (e.PrivateMessage.Message.StartsWith(".list "))
209                     {
210                         var channel = this.Client.Channels[e.PrivateMessage.Message.
                            Substring(6)];
211                         var list = channel.Users.Select(u => u.Nick).Aggregate((a, b) =>
                             a + "," + b);
212                         this.Client.SendMessage(list, e.PrivateMessage.User.Nick);
213                     }
214                     else if (e.PrivateMessage.Message.StartsWith(".whois "))
215                         this.Client.WhoIs(e.PrivateMessage.Message.Substring(7), null);
216                     else if (e.PrivateMessage.Message.StartsWith(".raw "))
217                         this.Client.SendRawMessage(e.PrivateMessage.Message.Substring(5
                            ));
218                     else if (e.PrivateMessage.Message.StartsWith(".mode "))
219                     {
220                         var parts = e.PrivateMessage.Message.Split(' ');
221                         this.Client.ChangeMode(parts[1], parts[2]);
222                     }
223                     else if (e.PrivateMessage.Message.StartsWith(".topic "))
224                     {
225                         string messageArgs = e.PrivateMessage.Message.Substring(7);
226                         if (messageArgs.Contains(" "))
227                         {
228                             string channel = messageArgs.Substring(0, messageArgs.
                                IndexOf(" "));
229                             string topic = messageArgs.Substring(messageArgs.IndexOf(" "
                                ) + 1);
230                             this.Client.Channels[channel].SetTopic(topic);
231                         }
232                         else
233                         {
234                             string channel = messageArgs.Substring(messageArgs.IndexOf(
                                "#"));
235                             this.Client.GetTopic(channel);
236                         }
237                     }
238                 };
239 
240                 this.Client.ChannelMessageRecieved += (s, e) =>
241                 {
242                     this.ChannelMessageReceived(s, e);
243                 };
244                 this.Client.ChannelTopicReceived += (s, e) =>
245                 {
246                     this.ChannelTopicReceived(s, e);
247                 };
248 
249                 this.Client.ConnectAsync();
```

```
250                 }
251             }
252 
253             /// <summary>
254             /// Quit the IRC chat
255             /// </summary>
256             public void Quit()
257             {
258                 //Unsubscribes client from three chat events
259                 this.Client.NetworkError -= (s, e) => this.NetWorkError(s, e);
260                 this.Client.RawMessageRecieved -= (s, e) => this.RawMessageReceived(s, e);
261                 this.Client.RawMessageSent -= (s, e) => this.RawMessageSent(s, e);
262 
263                 this.Client.Quit();
264                 this.Client = null;
265                 this.TbxChat = null;
266             }
267 
268             /// <summary>
269             /// Send message
270             /// </summary>
271             /// <param name="msg"></param>
272             public void SendMessage(string msg)
273             {
274                 this.Client.SendRawMessage(msg);
275             }
276         }
277     }
278 
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Model class
4    * Date : 29/05/2015
5    */
6   using System;
7   using System.Collections.Generic;
8   using System.IO;
9   using System.Linq;
10  using System.Text;
11  using WebMediaManager.Models.Sites;
12
13  namespace WebMediaManager.Models
14  {
15      public class Model
16      {
17          #region CONSTANTES
18
19
20
21          #endregion
22          #region PROPERTIES
23          private List<StreamingSite> _listSite;
24          private List<Container> _listContainer;
25
26          internal List<Container> ListContainer
27          {
28              get { return _listContainer; }
29              set { _listContainer = value; }
30          }
31
32          internal List<StreamingSite> ListSite
33          {
34              get { return _listSite; }
35              set { _listSite = value; }
36          }
37
38          #endregion
39
40          public Model()
41          {
42              this.InitSite();
43              this.ListContainer = new List<Container>();
44          }
45
46
47
48
49          /// <summary>
50          /// Get lasts videos of all site
51          /// </summary>
52          /// <returns>list of last video</returns>
53          public List<StreamingSite.SVideo> GetLastVideos()
```

```
54          {
55              List<StreamingSite.SVideo> listLastVideos = new List<StreamingSite.SVideo>();
56              for (int i = 0; i < this.ListSite.Count; i++)
57              {
58                  if (this.ListSite[i].Auth.IsConnected)
59                  {
60                      this.ListSite[i].UpdateLastVideo();
61                      for (int j = 0; j < this.ListSite[i].ListLastVideos.Count; j++)
62                      {
63                          listLastVideos.Add(this.ListSite[i].ListLastVideos[j]);
64                      }
65                  }
66              }
67
68              return listLastVideos;
69          }
70
71          /// <summary>
72          /// Get the new streams
73          /// </summary>
74          /// <returns></returns>
75          public List<StreamingSite.SVideo> GetOnlineStreams()
76          {
77              List<StreamingSite.SVideo> listLastStreams = new List<StreamingSite.SVideo
                >();
78              for (int i = 0; i < this.ListSite.Count; i++)
79              {
80                  if (this.ListSite[i].Auth.IsConnected)
81                  {
82                      this.ListSite[i].UpdateOnlineStream();
83                      if (this.ListSite[i].ListOnlineStreams != null)
84                      {
85                          for (int j = 0; j < this.ListSite[i].ListOnlineStreams.Count; j
                            ++)
86                          {
87                              if (this.ListSite[i].ListOnlineStreams[j].live)
88                                  listLastStreams.Add(this.ListSite[i].ListOnlineStreams[j
                                    ]);
89                          }
90                      }
91                  }
92              }
93
94              return listLastStreams;
95          }
96
97          /// <summary>
98          ///
99          /// </summary>
100         /// <param name="listVideos"></param>
101         /// <returns></returns>
102         public List<StreamingSite.SVideo> SortVideos(List<StreamingSite.SVideo>
            listVideos)
```

```
103         {
104             StreamingSite.SVideo[] listResult = new StreamingSite.SVideo[listVideos.
                Count];
105 
106             listVideos.Sort((x, y) => DateTime.Compare(x.createdAt, y.createdAt));
107 
108             return listVideos;
109         }
110 
111         /// <summary>
112         ///
113         /// </summary>
114         private void InitSite()
115         {
116             this.ListSite = new List<StreamingSite>();
117 
118             Youtube youtube = new Youtube();
119             Dailymotion dailymotion = new Dailymotion();
120             Vimeo vimeo = new Vimeo();
121             Twitch twitch = new Twitch();
122             this.ListSite.Add(twitch);
123         }
124 
125 
126 
127         /// <summary>
128         /// Open file categorie
129         /// </summary>
130         public void OpenFileCategories()
131         {
132             string pathFile = Path.Combine(Environment.GetFolderPath(Environment.
                SpecialFolder.ApplicationData), "WebMediaManager/Category.ini");
133             string[] videosLink = null;
134             string nameCategory = "";
135             int pFrom = 0;
136             int pTo = 0;
137 
138             if (File.Exists(pathFile))
139             {
140                 string[] allVideos = File.ReadAllLines(pathFile);
141 
142                 for (int i = 0; i < allVideos.Length; i++)
143                 {
144                     List<StreamingSite.SVideo> videos = new List<StreamingSite.SVideo>();
145                     if (allVideos[i][0] == '[')
146                     {
147                         pFrom = allVideos[i].IndexOf('[') + "[".Length;
148                         pTo = allVideos[i].LastIndexOf(']');
149 
150                         nameCategory = allVideos[i].Substring(pFrom, pTo - pFrom);
151 
152                         Container category = new Container(nameCategory);
153                         category.SetPathCategory();
```

```
154                         videosLink = category.GetVideos();
155
156                         for (int j = 0; j < this.ListSite.Count; j++)
157                         {
158                             for (int x = 0; x < videosLink.Count(); x++)
159                             {
160                                 if (this.ListSite[j].GetIdVideoByLink(videosLink[x]) !=
                                    null)
161                                     videos.Add(this.ListSite[j].GetVideoById(this.
                                        ListSite[j].GetIdVideoByLink(videosLink[x])));
162                             }
163                         }
164
165                         category.FillListVideos(videos);
166                         this.ListContainer.Add(category);
167                     }
168                 }
169
170             }
171         }
172
173         /// <summary>
174         /// Open file playlists
175         /// </summary>
176         public void OpenFilePlaylists()
177         {
178             string pathFile = Path.Combine(Environment.GetFolderPath(Environment.
                SpecialFolder.ApplicationData), "WebMediaManager/Playlist.ini");
179             string[] videosLink = null;
180             string namePlaylist = "";
181             int pFrom = 0;
182             int pTo = 0;
183
184             if (File.Exists(pathFile))
185             {
186                 string[] allVideos = File.ReadAllLines(pathFile);
187
188                 for (int i = 0; i < allVideos.Length; i++)
189                 {
190                     List<StreamingSite.SVideo> videos = new List<StreamingSite.SVideo>();
191                     if (allVideos[i][0] == '[')
192                     {
193                         pFrom = allVideos[i].IndexOf('[') + "[".Length;
194                         pTo = allVideos[i].LastIndexOf(']');
195
196                         namePlaylist = allVideos[i].Substring(pFrom, pTo - pFrom);
197
198                         Playlist playlist = new Playlist(namePlaylist);
199                         playlist.SetPathPlaylist();
200                         videosLink = playlist.GetVideos();
201
202                         for (int j = 0; j < this.ListSite.Count; j++)
203                         {
```

```
204                             for (int x = 0; x < videosLink.Count(); x++)
205                             {
206                                 if (this.ListSite[j].GetIdVideoByLink(videosLink[x]) !=
                                    null)
207                                     videos.Add(this.ListSite[j].GetVideoById(this.
                                        ListSite[j].GetIdVideoByLink(videosLink[x])));
208                             }
209                         }
210
211                         playlist.FillListVideos(videos);
212                         this.ListContainer.Add(playlist);
213                     }
214                 }
215
216             }
217
218         }
219
220         /// <summary>
221         /// Create files of containers
222         /// </summary>
223         public void CreateFileContainers()
224         {
225             string pathCategory = Path.Combine(Environment.GetFolderPath(Environment.
                SpecialFolder.ApplicationData), "WebMediaManager/Category.ini");
226             string pathPlaylist = Path.Combine(Environment.GetFolderPath(Environment.
                SpecialFolder.ApplicationData), "WebMediaManager/Playlist.ini");
227
228             if (!System.IO.Directory.Exists(Path.Combine(Environment.GetFolderPath(
                Environment.SpecialFolder.ApplicationData), "WebMediaManager")))
229             {
230                 Directory.CreateDirectory(Path.Combine(Environment.GetFolderPath(
                    Environment.SpecialFolder.ApplicationData), "WebMediaManager"));
231             }
232
233             if (!File.Exists(pathCategory))
234             {
235                 File.Create(pathCategory);
236             }
237
238             if (!File.Exists(pathPlaylist))
239             {
240                 File.Create(pathPlaylist);
241             }
242         }
243
244         /// <summary>
245         /// Add a container
246         /// </summary>
247         /// <param name="name">container name</param>
248         /// <param name="t_playlist">if is a playlist</param>
249         public void AddContainer(string name, bool t_playlist)
250         {
```

```
251                 if (t_playlist)
252                 {
253                     Playlist playlist = new Playlist(name);
254                     playlist.SetPathPlaylist();
255                     if (playlist.AddContainer())
256                         this.ListContainer.Add(playlist);
257                 }
258                 else
259                 {
260                     Container category = new Container(name);
261                     category.SetPathCategory();
262                     if (category.AddContainer())
263                         this.ListContainer.Add(category);
264                 }
265 
266             }
267 
268             /// <summary>
269             /// Check and get the new videos
270             /// </summary>
271             /// <returns></returns>
272             public List<List<StreamingSite.SVideo>> CheckNotificationsLastVideos()
273             {
274                 List<List<StreamingSite.SVideo>> listDiff = new List<List<StreamingSite.
                    SVideo>>();
275 
276                 //for all site
277                 for (int i = 0; i < this.ListSite.Count; i++)
278                 {
279                     List<StreamingSite.SVideo> diff = null;
280 
281                     if (this.ListSite[i].ListLastVideos != null)
282                     {
283                         //create a new list with the current last videos
284                         List<StreamingSite.SVideo> oldLastVideo = new List<StreamingSite.
                            SVideo>(this.ListSite[i].ListLastVideos);
285 
286                         //Update the current list of last video
287                         this.ListSite[i].UpdateLastVideo();
288 
289                         //create a new list with the current last videos
290                         diff = new List<StreamingSite.SVideo>(this.ListSite[i].
                            ListLastVideos);
291 
292                         //Compare the two list
293                         for (int x = 0; x < this.ListSite[i].ListLastVideos.Count; x++)
294                         {
295                             for (int y = 0; y < oldLastVideo.Count; y++)
296                             {
297                                 if (this.ListSite[i].ListLastVideos[x].id == oldLastVideo[y
                                    ].id)
298                                 {
299                                     //remove if the current video is on the old list video
```

```
300                                 diff.Remove(this.ListSite[i].ListLastVideos[x]);
301                             }
302                         }
303                     }
304                 }
305                 else
306                 {
307                     this.ListSite[i].UpdateLastVideo();
308                 }
309                 listDiff.Add(diff);
310             }
311 
312             return listDiff;
313         }
314 
315         /// <summary>
316         /// Check and get new online streams
317         /// </summary>
318         /// <returns></returns>
319         public List<List<StreamingSite.SVideo>> CheckNotificationsOnlineStreams()
320         {
321             List<List<StreamingSite.SVideo>> listDiff = new List<List<StreamingSite.
                SVideo>>();
322 
323             //for all site
324             for (int i = 0; i < this.ListSite.Count; i++)
325             {
326                 List<StreamingSite.SVideo> diff = null;
327 
328                 if (this.ListSite[i].ListOnlineStreams != null)
329                 {
330                     //create a new list with the current last videos
331                     List<StreamingSite.SVideo> oldLastVideo = new List<StreamingSite.
                        SVideo>(this.ListSite[i].ListOnlineStreams);
332 
333                     //Update the current list of last video
334                     this.ListSite[i].UpdateOnlineStream();
335 
336                     //create a new list with the current last videos
337                     diff = new List<StreamingSite.SVideo>(this.ListSite[i].
                        ListOnlineStreams);
338 
339                     //Compare the two list
340                     for (int x = 0; x < this.ListSite[i].ListOnlineStreams.Count; x++)
341                     {
342                         for (int y = 0; y < oldLastVideo.Count; y++)
343                         {
344                             if (this.ListSite[i].ListOnlineStreams[x].id == oldLastVideo
                                [y].id)
345                             {
346                                 //remove if the current video is on the old list video
347                                 diff.Remove(this.ListSite[i].ListOnlineStreams[x]);
348                             }
```

```
349                         }
350                     }
351                 }
352                 else
353                 {
354                     this.ListSite[i].UpdateOnlineStream();
355                 }
356                 listDiff.Add(diff);
357             }
358 
359             return listDiff;
360         }
361 
362 
363         /// <summary>
364         /// Get the name site
365         /// </summary>
366         /// <returns>array name</returns>
367         public string[] GetNameSites()
368         {
369             string[] nameSites = new string[this.ListSite.Count];
370 
371             for (int i = 0; i < ListSite.Count; i++)
372             {
373                 nameSites[i] = this.ListSite[i].Name;
374             }
375 
376             return nameSites;
377         }
378 
379         /// <summary>
380         /// Get username from site
381         /// </summary>
382         /// <param name="nameSite">name site</param>
383         /// <returns>username</returns>
384         public string GetUserName(string nameSite)
385         {
386             for (int i = 0; i < this.ListSite.Count; i++)
387             {
388                 if (this.ListSite[i].Name == nameSite)
389                 {
390                     return this.ListSite[i].UserName;
391                 }
392             }
393 
394             return null;
395         }
396 
397         /// <summary>
398         /// Get the accesstoken from a website
399         /// </summary>
400         /// <param name="nameSite">name site</param>
401         /// <returns>access token</returns>
```

```
402         public string GetAccessToken(string nameSite)
403         {
404             for (int i = 0; i < this.ListSite.Count; i++)
405             {
406                 if (this.ListSite[i].Name == nameSite)
407                 {
408                     return this.ListSite[i].Auth.Access_token;
409                 }
410             }
411 
412             return null;
413         }
414 
415         /// <summary>
416         /// Search videos
417         /// </summary>
418         /// <param name="request">request</param>
419         /// <param name="limit">limit of result</param>
420         /// <returns></returns>
421         public List<StreamingSite.SVideo> SearchVideos(string request, int limit)
422         {
423             List<StreamingSite.SVideo> result = new List<StreamingSite.SVideo>();
424             for (int i = 0; i < this.ListSite.Count; i++)
425             {
426                 List<StreamingSite.SVideo> listVideosSite = this.ListSite[i].
                    SearchVideos(request, limit);
427 
428                 for (int j = 0; j < listVideosSite.Count; j++)
429                 {
430                     result.Add(listVideosSite[j]);
431                 }
432             }
433 
434             return result;
435         }
436 
437         /// <summary>
438         /// Get channels followed
439         /// </summary>
440         /// <returns></returns>
441         public List<StreamingSite.SChannel> GetChannelsFollowed()
442         {
443             List<StreamingSite.SChannel> listChannelsSite = null;
444             for (int i = 0; i < this.ListSite.Count; i++)
445             {
446                 if(this.ListSite[i].Auth.IsConnected)
447                     listChannelsSite = this.ListSite[i].GetChannelFollowed();
448             }
449 
450             return listChannelsSite;
451         }
452 
453         /// <summary>
```

```
454             /// Get the access token in Url
455             /// </summary>
456             /// <returns></returns>
457             public string GetAccessTokenInUrl(string urlWithAccessToken)
458             {
459                 bool inToken = false;
460                 StringBuilder result = new StringBuilder();
461 
462                 for (int i = 0; i < urlWithAccessToken.Length; i++)
463                 {
464                     if (inToken)
465                     {
466                         if (urlWithAccessToken[i] != '&')
467                         {
468                             result.Append(urlWithAccessToken[i]);
469                         }
470                         else
471                         {
472                             break;
473                         }
474                     }
475 
476                     if (urlWithAccessToken[i] == '=')
477                     {
478                         inToken = true;
479                     }
480                 }
481                 if (result.ToString() != "authorize")
482                     return result.ToString();
483                 else
484                     return "";
485             }
486         }
487     }
488 
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Panel
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class Panel
12      {
13          [DataMember]
14          public long _id { get; set; }
15
16          [DataMember]
17          public int display_order { get; set; }
18
19          [DataContract]
20          public class Data
21          {
22              [DataMember]
23              public string link { get; set; }
24
25              [DataMember]
26              public string image { get; set; }
27
28              [DataMember]
29              public string title { get; set; }
30
31              [DataMember]
32              public string description { get; set; }
33          }
34
35          [DataMember]
36          public Data data { get; set; }
37
38          [DataMember]
39          public string kind { get; set; }
40
41          [DataMember]
42          public string html_description { get; set; }
43
44          [DataMember]
45          public long user_id { get; set; }
46
47          [DataMember]
48          public string channel { get; set; }
49      }
50  }
51
```

```
1   using ChatSharp;
2   using System;
3   using System.Collections.Generic;
4   using System.ComponentModel;
5   using System.Data;
6   using System.Drawing;
7   using System.IO;
8   using System.Linq;
9   using System.Net;
10  using System.Text;
11  using System.Threading;
12  using System.Threading.Tasks;
13  using System.Timers;
14  using System.Windows.Forms;
15  using WebMediaManager.Controllers;
16  using WebMediaManager.Models;
17  using WebMediaManager.Views;
18
19  namespace WebMediaManager
20  {
21      public partial class PersonalInterface : Form
22      {
23          private SitesController _sitesController;
24          private ContainersController _containersController;
25          private Model _model;
26          private System.Timers.Timer _timerCheck;
27
28          public System.Timers.Timer TimerCheck
29          {
30              get { return _timerCheck; }
31              set { _timerCheck = value; }
32          }
33
34          internal Model Model
35          {
36              get { return _model; }
37              set { _model = value; }
38          }
39
40          internal ContainersController ContainersController
41          {
42              get { return _containersController; }
43              set { _containersController = value; }
44          }
45
46          internal SitesController SitesController
47          {
48              get { return _sitesController; }
49              set { _sitesController = value; }
50          }
51
52          public PersonalInterface()
53          {
```

```
54              InitializeComponent();
55              this.Model = new Model();
56              this.SitesController = new SitesController(this, this.Model);
57              this.ContainersController = new ContainersController(this, this.Model);
58
59              this.ContainersController.CreateFileContainers();
60              this.ContainersController.OpenFileContainers();
61
62              DisplayButtonsSite();
63              DisplayLinkCategory();
64              DisplayLinkPlaylist();
65              CreateButtonHome();
66              DisplayHomePanel();
67
68              InitTimer();
69
70              if (!this.pnlContent.Focused)
71                  this.pnlContent.Focus();
72
73          }
74
75          private void InitTimer()
76          {
77              this.TimerCheck = new System.Timers.Timer();
78              this.TimerCheck.Enabled = true;
79              this.TimerCheck.Elapsed += new System.Timers.ElapsedEventHandler(OnTickCheck
                );
80              this.TimerCheck.Interval = 5000;
81          }
82
83          private void OnTickCheck(object sender, ElapsedEventArgs e)
84          {
85              if(this.SitesController.CountConnectedSite() > 0)
86              {
87                  this.PopNotificationVideos();
88                  this.PopNotificationStreams();
89              }
90          }
91
92          private void PopNotificationVideos()
93          {
94              List<List<StreamingSite.SVideo>> newVideos = this.SitesController.
                CheckNotificationsLastVideos();
95
96              for (int i = 0; i < newVideos.Count; i++)
97              {
98                  this.Invoke((MethodInvoker)delegate
99                  {
100                     if (newVideos[i].Count > 0)
101                     {
102                         this.Notif.TitleText = newVideos[i][0].videoName;
103                         this.Notif.ContentText = newVideos[i][0].channelName + "\n" +
                            newVideos[i][0].nbViews + "\n" + newVideos[i][0].siteName;
```

```
104                         this.Notif.ContentPadding = new Padding(1);
105                         this.Notif.TitlePadding = new Padding(1);
106
107                         WebClient wc = new WebClient();
108                         byte[] bytes = wc.DownloadData(newVideos[i][0].preview);
109                         MemoryStream ms = new MemoryStream(bytes);
110                         System.Drawing.Image img = System.Drawing.Image.FromStream(ms);
111
112                         System.Drawing.Size sizeimg = new System.Drawing.Size(150, 82);
113
114                         this.Notif.Image = img;
115                         this.Notif.ImagePadding = new Padding(5);
116                         this.Notif.ImageSize = sizeimg;
117                         this.Notif.Click += (s, e) => { ViewUtils.CreateFormVideo(
                            newVideos[i][0], this.Model); };
118                         this.Notif.Popup();
119                     }
120                 }
121                 );
122             }
123
124         }
125
126          private void PopNotificationStreams()
127          {
128              List<List<StreamingSite.SVideo>> newVideos = this.SitesController.
                 CheckNotificationsOnlineStreams();
129
130              for (int i = 0; i < newVideos.Count; i++)
131              {
132                  this.Invoke((MethodInvoker)delegate
133                  {
134                      if (newVideos[i].Count > 0)
135                      {
136                          this.Notif.TitleText = newVideos[i][0].videoName;
137                          this.Notif.ContentText = newVideos[i][0].channelName + "\n" +
                            newVideos[i][0].nbViews + "\n" + newVideos[i][0].siteName;
138                          this.Notif.ContentPadding = new Padding(1);
139                          this.Notif.TitlePadding = new Padding(1);
140
141                          WebClient wc = new WebClient();
142                          byte[] bytes = wc.DownloadData(newVideos[i][0].preview);
143                          MemoryStream ms = new MemoryStream(bytes);
144                          System.Drawing.Image img = System.Drawing.Image.FromStream(ms);
145
146                          System.Drawing.Size sizeimg = new System.Drawing.Size(150, 82);
147
148                          this.Notif.Image = img;
149                          this.Notif.ImagePadding = new Padding(5);
150                          this.Notif.ImageSize = sizeimg;
151                          this.Notif.Click += (s, e) => { ViewUtils.CreateFormVideo(
                            newVideos[i][0], this.Model); };
152                          this.Notif.Popup();
```

```
153                     }
154                 }
155                 );
156             }
157 
158         }
159 
160         public void DisplayOnlineStreams(Panel pnl)
161         {
162             List<StreamingSite.SVideo> onlineStreams = this.SitesController.
                GetOnlineStreams();
163             int j = 0;
164             int counter_for_line = 0;
165 
166             for (int i = 0; i < onlineStreams.Count; i++)
167             {
168                 if (j % 4 == 0 && j != 0)
169                 {
170                     j = 0;
171                     counter_for_line++;
172                     pnl.Size = new Size(pnl.Size.Width, pnl.Size.Height + 200);
173                 }
174                 ViewUtils.CreatePreview(pnl, onlineStreams[i], j, counter_for_line, this
                    .Model);
175                 j++;
176             }
177         }
178 
179         public void DisplayOnlineStreamsBySite(Panel pnl, string siteName)
180         {
181             List<StreamingSite.SVideo> onlineStreams = this.SitesController.
                GetOnlineStreamFromSite(siteName);
182 
183             if (onlineStreams != null)
184             {
185 
186                 int j = 0;
187                 int counter_for_line = 0;
188 
189                 for (int i = 0; i < onlineStreams.Count; i++)
190                 {
191                     if (j % 4 == 0 && j != 0)
192                     {
193                         j = 0;
194                         counter_for_line++;
195                         pnl.Size = new Size(pnl.Size.Width, pnl.Size.Height + 200);
196                     }
197                     ViewUtils.CreatePreview(pnl, onlineStreams[i], j, counter_for_line,
                        this.Model);
198                     j++;
199                 }
200             }
201         }
```

```
202
203         private void DisplayButtonsSite()
204         {
205             string[] nameSites = this.SitesController.GetNameSites();
206
207             for (int i = 0; i < nameSites.Length; i++)
208             {
209                 this.CreateButtonSite(nameSites[i], i);
210                 this.pnlLeftMid.Location = new Point(this.pnlLeftMid.Location.X, this.
                    pnlLeftMid.Location.Y + 25);
211             }
212         }
213
214         public Button CreateButtonSite(string nameSite, int index_btn)
215         {
216             Button btnSite = new Button();
217             btnSite.Size = new Size(this.pnlLeftTop.Width - 20, 25);
218             btnSite.Location = new Point(8, 30 * (index_btn));
219             btnSite.FlatStyle = FlatStyle.Flat;
220             btnSite.Text = nameSite;
221             btnSite.Click += (sender, e) => OnClickButtonSite(sender, e, nameSite);
222             this.pnlLeftTop.Controls.Add(btnSite);
223             if (index_btn != 0)
224                 this.pnlLeftTop.Size = new Size(this.pnlLeftTop.Width, this.pnlLeftTop.
                    Size.Height + 20);
225             return btnSite;
226         }
227
228         private void DisplayLinkCategory()
229         {
230             this.pnlLeftBot.Controls.Clear();
231             this.pnlLeftMid.Controls.Clear();
232             this.pnlLeftMid.Size = new Size(194, 45);
233             Label lblTitle = new Label();
234             lblTitle.Font = new Font("Arial", 12, FontStyle.Bold);
235             lblTitle.Location = new Point(0, 0);
236             lblTitle.Text = "Categories";
237
238             TextBox tbxCategory = new TextBox();
239             tbxCategory.Location = new Point(10, 20);
240             tbxCategory.Size = new Size(120, 20);
241             tbxCategory.BorderStyle = BorderStyle.FixedSingle;
242
243             Button btnAddCategory = new Button();
244             btnAddCategory.Location = new Point(15 + tbxCategory.Size.Width, 20);
245             btnAddCategory.Size = new Size(40, 22);
246             btnAddCategory.FlatStyle = FlatStyle.Flat;
247             btnAddCategory.Text = "Add";
248             btnAddCategory.Click += (sender, e) => OnClickAddContainer(sender, e,
                tbxCategory.Text, false);
249
250             this.pnlLeftMid.Location = new Point(this.pnlLeftMid.Location.X, this.
                pnlLeftTop.Size.Height + pnlLeftTop.Location.Y);
```

```
251             this.pnlLeftMid.Controls.Add(btnAddCategory);
252             this.pnlLeftMid.Controls.Add(tbxCategory);
253             this.pnlLeftMid.Controls.Add(lblTitle);
254 
255             List<string> nameCategory = this.ContainersController.GetNamesCategory();
256 
257             for (int i = 0; i < nameCategory.Count; i++)
258             {
259                 CreateLinkCategory(this.pnlLeftMid, nameCategory[i], i, this.pnlContent);
260             }
261         }
262 
263         private void DisplayLinkPlaylist()
264         {
265             Label lblTitle = new Label();
266             lblTitle.Font = new Font("Arial", 12, FontStyle.Bold);
267             lblTitle.Location = new Point(0, this.pnlLeftMid.Height + 2);
268             lblTitle.Text = "Playlist";
269 
270             TextBox tbxPlaylist = new TextBox();
271             tbxPlaylist.Location = new Point(10, lblTitle.Location.Y + lblTitle.Size.
                Height);
272             tbxPlaylist.Size = new Size(120, 20);
273             tbxPlaylist.BorderStyle = BorderStyle.FixedSingle;
274 
275             Button btnAddPlaylist = new Button();
276             btnAddPlaylist.Location = new Point(15 + tbxPlaylist.Size.Width, lblTitle.
                Location.Y + lblTitle.Size.Height);
277             btnAddPlaylist.Size = new Size(40, 22);
278             btnAddPlaylist.FlatStyle = FlatStyle.Flat;
279             btnAddPlaylist.Text = "Add";
280             btnAddPlaylist.Click += (sender, e) => OnClickAddContainer(sender, e,
                tbxPlaylist.Text, true);
281 
282             this.pnlLeftMid.Size = new Size(this.pnlLeftMid.Size.Width, this.pnlLeftMid.
                Size.Height + 45);
283             this.pnlLeftMid.Controls.Add(tbxPlaylist);
284             this.pnlLeftMid.Controls.Add(btnAddPlaylist);
285             this.pnlLeftMid.Controls.Add(lblTitle);
286 
287             List<string> namePlaylist = this.ContainersController.GetNamesPlaylist();
288 
289             for (int i = 0; i < namePlaylist.Count; i++)
290             {
291                 CreateLinkPlaylist(this.pnlLeftMid, namePlaylist[i], i, this.pnlContent);
292             }
293 
294             this.pnlLeftBot.Location = new Point(this.pnlLeftBot.Location.X, this.
                pnlLeftMid.Size.Height + this.pnlLeftMid.Location.Y);
295         }
296 
297         private void DisplayLastVideos(Panel pnl)
298         {
```

```
299             List<StreamingSite.SVideo> lastVideos = this.SitesController.GetLastVideos();
300             int j = 0;
301             int counter_for_line = 0;
302             for (int i = 0; i < lastVideos.Count; i++)
303             {
304                 if (j % 4 == 0 && j != 0)
305                 {
306                     j = 0;
307                     counter_for_line++;
308                     pnl.Size = new Size(pnl.Size.Width, pnl.Size.Height + 200);
309                 }
310                 ViewUtils.CreatePreview(pnl, lastVideos[i], j, counter_for_line, this.
                    Model);
311                 j++;
312             }
313         }
314 
315         private void DisplayLastVideosBySite(Panel pnl, string siteName)
316         {
317             List<StreamingSite.SVideo> lastVideos = this.SitesController.
                GetLastVideosFromSite(siteName);
318             int j = 0;
319             int counter_for_line = 0;
320             for (int i = 0; i < lastVideos.Count; i++)
321             {
322                 if (j % 4 == 0 && j != 0)
323                 {
324                     j = 0;
325                     counter_for_line++;
326                     pnl.Size = new Size(pnl.Size.Width, pnl.Size.Height + 200);
327                 }
328                 ViewUtils.CreatePreview(pnl, lastVideos[i], j, counter_for_line, this.
                    Model);
329                 j++;
330             }
331         }
332 
333         private void DisplayResultSearch()
334         {
335             if (this.tbxSearch.Text != "")
336             {
337                 Panel newPanel = new Panel();
338                 this.pnlContent.Controls.Clear();
339                 newPanel.Size = new Size(850, 200);
340                 this.pnlContent.Controls.Add(newPanel);
341                 List<StreamingSite.SVideo> resultSearch = this.SitesController.
                    SearchVideos(this.tbxSearch.Text, 8);
342 
343                 //LABEL TITLE
344                 Label lblRequest = new Label();
345                 lblRequest.Text = this.tbxSearch.Text;
346                 lblRequest.AutoSize = true;
347                 lblRequest.Location = new Point(10, 10);
```

```
348                 lblRequest.Font = new Font("Arial", 22, FontStyle.Bold);
349
350                 newPanel.Location = new Point(10, lblRequest.Height + lblRequest.
                    Location.Y + 50);
351
352                 this.pnlContent.Controls.Add(lblRequest);
353                 List<List<StreamingSite.SVideo>> videosSort = this.SitesController.
                    GetVideosFromSite(resultSearch);
354
355                 if (videosSort != null)
356                 {
357                     for (int i = 0; i < videosSort.Count; i++)
358                     {
359                         Label lblTitleSite = new Label();
360                         lblTitleSite.Text = videosSort[i][0].siteName;
361                         lblTitleSite.Location = new Point(10, (newPanel.Size.Height - (
                            newPanel.Size.Height - lblRequest.Size.Height))+10);
362                         lblTitleSite.Font = new Font("Arial", 18, FontStyle.Bold);
363                         this.pnlContent.Controls.Add(lblTitleSite);
364
365                         int counter_for_line = 0;
366                         int x = 0;
367                         for (int j = 0; j < videosSort[i].Count; j++)
368                         {
369                             if (j % 4 == 0 && j != 0)
370                             {
371                                 x = 0;
372                                 counter_for_line++;
373                                 newPanel.Size = new Size(newPanel.Size.Width, newPanel.
                                    Size.Height + 120);
374                             }
375
376                             ViewUtils.CreatePreview(newPanel, videosSort[i][j], x,
                                counter_for_line, this.Model);
377                             x++;
378                         }
379                     }
380                 }
381             }
382         }
383
384
385         private void DisplaySubscribes(string nameSite)
386         {
387             int j = 0;
388             this.pnlLeftMid.Location = new Point(this.pnlLeftTop.Location.X + 5, this.
                pnlLeftTop.Size.Height + this.pnlLeftTop.Location.Y);
389             this.pnlLeftBot.Visible = true;
390             this.pnlLeftBot.Controls.Clear();
391             this.pnlLeftBot.Size = new Size(185, 45);
392             Label lblTitle = new Label();
393             lblTitle.Font = new Font("Arial", 12, FontStyle.Bold);
394             lblTitle.Location = new Point(0, 0);
```

```
395             lblTitle.Text = "Abonnements";
396             lblTitle.AutoSize = true;
397 
398             List<StreamingSite.SChannel> channelsFollowed = this.SitesController.
                GetChannelFollowed();
399 
400             this.pnlLeftBot.Controls.Add(lblTitle);
401             this.pnlLeftBot.Location = new Point(this.pnlLeftBot.Location.X + 5, this.
                pnlLeftMid.Size.Height + pnlLeftMid.Location.Y + 10);
402 
403             if (channelsFollowed != null)
404             {
405                 for (int i = 0; i < channelsFollowed.Count; i++)
406                 {
407                     if (channelsFollowed[i].siteName == nameSite)
408                     {
409                             Label lblChannel = new Label();
410                             lblChannel.Font = new Font("Arial", 9);
411                             lblChannel.Text = channelsFollowed[i].channelName;
412                             lblChannel.Location = new Point(10, lblTitle.Size.Height + (
                            lblChannel.Size.Height*j));
413                             lblChannel.MouseEnter += new EventHandler(MouseEnterLabel);
414                             lblChannel.MouseLeave += new EventHandler(MouseLeaveLabel);
415                             lblChannel.Click += (sender, e) => OnClickLabelContainer(
                            sender, e);
416 
417                             this.pnlLeftBot.Controls.Add(lblChannel);
418                             this.pnlLeftBot.Size = new Size(this.pnlLeftBot.Size.Width,
                            this.pnlLeftBot.Size.Height + lblChannel.Size.Height);
419                             j++;
420                     }
421                 }
422             }
423         }
424 
425         private void DisplayChannel(string siteName)
426         {
427             this.pnlContent.Controls.Clear();
428             Panel pnlOnlineStreams = new Panel();
429         }
430 
431         private void DisplaySitePanel(string siteName)
432         {
433             //TODO : Créer deux panels : Last videos et online stream
434             this.pnlContent.Controls.Clear();
435 
436             Panel pnlOnlineStreams = new Panel();
437             Panel pnlLastVideos = new Panel();
438 
439             Label lblOnline = new Label();
440             Label lblLast = new Label();
441 
442             //STREAMS ONLINE
```

```
443                 lblOnline.Text = "Online streams";
444                 lblOnline.AutoSize = true;
445                 lblOnline.Location = new Point(10, 10);
446                 lblOnline.Font = new Font("Arial", 18, FontStyle.Bold);
447 
448                 pnlOnlineStreams.Location = new Point(10, lblOnline.Location.Y + lblOnline.
                    Height + 10);
449                 pnlOnlineStreams.Size = new Size(850, 200);
450 
451                 this.pnlContent.Controls.Add(lblOnline);
452                 this.pnlContent.Controls.Add(pnlOnlineStreams);
453 
454                 DisplayOnlineStreamsBySite(pnlOnlineStreams, siteName);
455 
456                 //LAST VIDEO RELEASED
457                 lblLast.Text = "Last videos";
458                 lblLast.AutoSize = true;
459                 lblLast.Location = new Point(10, pnlOnlineStreams.Size.Height +
                    pnlOnlineStreams.Location.Y);
460                 lblLast.Font = new Font("Arial", 18, FontStyle.Bold);
461 
462                 pnlLastVideos.Location = new Point(10, lblLast.Location.Y + lblLast.Size.
                    Height + 10);
463                 pnlLastVideos.Size = new Size(850, 200);
464 
465                 this.pnlContent.Controls.Add(lblLast);
466                 this.pnlContent.Controls.Add(pnlLastVideos);
467 
468                 DisplayLastVideosBySite(pnlLastVideos, siteName);
469 
470             }
471 
472 
473             private void DisplayHomePanel()
474             {
475                 //TODO : Créer deux panels : Last videos et online stream
476                 this.pnlContent.Controls.Clear();
477                 Panel pnlOnlineStreams = new Panel();
478                 Panel pnlLastVideos = new Panel();
479 
480                 Label lblOnline = new Label();
481                 Label lblLast = new Label();
482 
483                 //STREAMS ONLINE
484                 lblOnline.Text = "Online streams";
485                 lblOnline.AutoSize = true;
486                 lblOnline.Location = new Point(10, 10);
487                 lblOnline.Font = new Font("Arial", 18, FontStyle.Bold);
488 
489                 pnlOnlineStreams.Location = new Point(10, lblOnline.Location.Y + lblOnline.
                    Height+10);
490                 pnlOnlineStreams.Size = new Size(850, 200);
491 
```

```
492             this.pnlContent.Controls.Add(lblOnline);
493             this.pnlContent.Controls.Add(pnlOnlineStreams);
494 
495             DisplayOnlineStreams(pnlOnlineStreams);
496 
497             //LAST VIDEO RELEASED
498             lblLast.Text = "Last videos";
499             lblLast.AutoSize = true;
500             lblLast.Location = new Point(10, pnlOnlineStreams.Size.Height +
                pnlOnlineStreams.Location.Y);
501             lblLast.Font = new Font("Arial", 18, FontStyle.Bold);
502 
503             pnlLastVideos.Location = new Point(10, lblLast.Location.Y + lblLast.Size.
                Height + 10);
504             pnlLastVideos.Size = new Size(850, 200);
505 
506             this.pnlContent.Controls.Add(lblLast);
507             this.pnlContent.Controls.Add(pnlLastVideos);
508 
509             DisplayLastVideos(pnlLastVideos);
510         }
511 
512         public void CreateButtonHome()
513         {
514             Button btnHome = new Button();
515             btnHome.Size = new Size(pnlLeftTop.Width - 20, 25);
516             btnHome.Location = new Point(8, 5);
517             btnHome.FlatStyle = FlatStyle.Flat;
518             btnHome.Text = "Personnal interface";
519             btnHome.Click += new EventHandler(OnClickButtonHome);
520             pnlLeft.Controls.Add(btnHome);
521         }
522 
523         public void CreateLinkCategory(Panel pnlContainers, string name, int index_cont,
             Panel pnlContent)
524         {
525             Label lblCategory = new Label();
526             lblCategory.Font = new Font("Arial", 9);
527             lblCategory.AutoSize = true;
528             lblCategory.Text = name;
529             lblCategory.Location = new Point(10, (15 * index_cont) + 50);
530             lblCategory.MouseEnter += new EventHandler(MouseEnterLabel);
531             lblCategory.MouseLeave += new EventHandler(MouseLeaveLabel);
532             lblCategory.Click += (sender, e) => OnClickLabelContainer(sender, e);
533             pnlContainers.Size = new Size(pnlContainers.Size.Width, pnlContainers.Size.
                Height + 16);
534             pnlContainers.Controls.Add(lblCategory);
535         }
536 
537         public void CreateLinkPlaylist(Panel pnlContainers, string name, int index_play,
             Panel pnlContent)
538         {
539             Label lblPlaylist = new Label();
```

```
540                 lblPlaylist.Font = new Font("Arial", 9);
541                 lblPlaylist.AutoSize = true;
542                 lblPlaylist.Text = name;
543                 lblPlaylist.Location = new Point(10, pnlContainers.Size.Height);
544                 lblPlaylist.MouseEnter += new EventHandler(MouseEnterLabel);
545                 lblPlaylist.MouseLeave += new EventHandler(MouseLeaveLabel);
546                 lblPlaylist.Click += (sender, e) => OnClickLabelContainer(sender, e);
547                 pnlContainers.Size = new Size(pnlContainers.Size.Width, pnlContainers.Size.
                    Height + 15);
548                 pnlContainers.Controls.Add(lblPlaylist);
549             }
550 
551             private void OnClickDisconnect(object sender, EventArgs e, string nameSite)
552             {
553                 this.wbrConnexion.Navigated -= (s, ev) => this.OnChangeUrl(s, ev, nameSite);
554                 this.SitesController.Disconnect(nameSite);
555                 this.ClearCookies();
556                 this.wbrConnexion.AllowNavigation = false;
557             }
558 
559             private void MouseEnterLabel(object sender, EventArgs e)
560             {
561                 Label lbl = sender as Label;
562                 lbl.ForeColor = Color.Red;
563             }
564 
565             private void MouseLeaveLabel(object sender, EventArgs e)
566             {
567                 Label lbl = sender as Label;
568                 lbl.ForeColor = Color.Black;
569             }
570 
571             private void OnClickLabelContainer(object sender, EventArgs e)
572             {
573                 Label lbl = sender as Label;
574                 CreateFullPanelVideos(lbl.Text);
575             }
576 
577             private void OnClickChannelFollowed(object sender, EventArgs e)
578             {
579                 Label lbl = sender as Label;
580                 this.DisplayChannel(lbl.Text);
581             }
582 
583             private void OnClickButtonHome(object sender, EventArgs e)
584             {
585                 this.pnlLeftBot.Visible = false;
586                 DisplayLinkCategory();
587                 DisplayLinkPlaylist();
588                 DisplayHomePanel();
589             }
590 
591             private void OnClickButtonSite(object sender, EventArgs e, string nameSite)
```

```
592             {
593                 this.pnlLeftBot.Visible = true;
594 
595                 if(this.SitesController.SiteIsConnected(nameSite))
596                 {
597                     this.DisplayOptionAccount(nameSite);
598                     this.DisplaySubscribes(nameSite);
599                     this.DisplaySitePanel(nameSite);
600                 }
601                 else
602                 {
603                     this.wbrConnexion.Navigated += (s, ev) => this.OnChangeUrl(s, ev,
                        nameSite);
604                     this.DisplayConnexionPage(nameSite);
605                 }
606 
607             }
608 
609             private void OnClickAddContainer(object sender, EventArgs e, string name, bool
                playlist)
610             {
611                 if (name != "")
612                 {
613                     this.ContainersController.AddContainer(name, playlist);
614                     this.DisplayLinkCategory();
615                     this.DisplayLinkPlaylist();
616                 }
617             }
618 
619             public void CreateFullPanelVideos(string name)
620             {
621                 List<StreamingSite.SVideo> listVideos = this.ContainersController.
                    GetVideosOfContainer(name);
622                 Panel newPanel = new Panel();
623                 this.pnlContent.Controls.Clear();
624 
625 
626                 if (listVideos != null)
627                 {
628                     newPanel.Size = new Size(this.pnlContent.Size.Width - 10, this.
                        pnlContent.Height - 50);
629                     newPanel.Location = new Point(10, 50);
630                     pnlContent.Controls.Add(newPanel);
631 
632                     Label lblTitle = new Label();
633                     lblTitle.Location = new Point(0, 15);
634                     lblTitle.Font = new Font("Arial", 16, FontStyle.Bold);
635                     lblTitle.Text = name;
636                     lblTitle.AutoSize = true;
637 
638                     pnlContent.Controls.Add(lblTitle);
639 
640                     int j = 0;
```

```
641                 int counter_for_line = 0;
642                 for (int i = 0; i < listVideos.Count; i++)
643                 {
644                     if (j % 4 == 0 && j != 0)
645                     {
646                         j = 0;
647                         counter_for_line++;
648                         newPanel.Size = new Size(newPanel.Size.Width, newPanel.Size.
                            Height + 120);
649                     }
650                     ViewUtils.CreatePreview(newPanel, listVideos[i], j, counter_for_line
                        , this.Model);
651                     j++;
652                 }
653             }
654         }
655 
656         private void btnSearch_Click(object sender, EventArgs e)
657         {
658             this.DisplayResultSearch();
659         }
660 
661         private void DisplayOptionAccount(string nameSite)
662         {
663             this.pnlLeftMid.Controls.Clear();
664             this.pnlLeftMid.Size = new Size(194, 45);
665             Label lblTitle = new Label();
666             lblTitle.Font = new Font("Arial", 12, FontStyle.Bold);
667             lblTitle.Location = new Point(0, 0);
668             lblTitle.Text = "Options";
669 
670             CheckBox cbxNotif = new CheckBox();
671             cbxNotif.Font = new Font("Arial", 9);
672             cbxNotif.Location = new Point(10, lblTitle.Size.Height);
673             cbxNotif.Text = "Notification";
674 
675             Label lblAccount = new Label();
676             lblAccount.Font = new Font("Arial", 9);
677             lblAccount.Location = new Point(10, cbxNotif.Location.Y + cbxNotif.Size.
                Height);
678             lblAccount.Text = "Account";
679 
680             Label lblDisconnect = new Label();
681             lblDisconnect.Font = new Font("Arial", 9);
682             lblDisconnect.Location = new Point(10, lblAccount.Location.Y + lblAccount.
                Size.Height);
683             lblDisconnect.Text = "Disconnect";
684             lblDisconnect.Click += (s, e) => this.OnClickDisconnect(s, e, nameSite);
685 
686             this.pnlLeftMid.Location = new Point(5, this.pnlLeftTop.Height);
687             this.pnlLeftMid.Size = new Size(this.pnlLeftMid.Size.Width, lblTitle.Size.
                Height + cbxNotif.Size.Height + lblAccount.Size.Height + lblDisconnect.Size.
                Height);
```

```
688                 this.pnlLeftMid.Controls.Add(lblTitle);
689                 this.pnlLeftMid.Controls.Add(cbxNotif);
690                 this.pnlLeftMid.Controls.Add(lblAccount);
691                 this.pnlLeftMid.Controls.Add(lblDisconnect);
692             }
693 
694             private void DisplayConnexionPage(string siteName)
695             {
696                 this.pnlContent.Controls.Clear();
697                 this.wbrConnexion.AllowNavigation = true;
698                 this.wbrConnexion.Dock = DockStyle.Fill;
699                 this.wbrConnexion.Navigate(this.SitesController.GetLinkConnexionPage(
                    siteName));
700                 this.pnlContent.Controls.Add(this.wbrConnexion);
701             }
702 
703             private void OnChangeUrl(object sender, WebBrowserNavigatedEventArgs e, string
                siteName)
704             {
705                 if (this.wbrConnexion.Url != null)
706                 {
707                     string accessToken = this.SitesController.GetAccessTokenInUrl(this.
                        wbrConnexion.Url.ToString());
708 
709                     if (accessToken != "")
710                     {
711                         this.SitesController.Connect(accessToken, siteName);
712                         this.DisplaySitePanel(siteName);
713                         this.DisplayOptionAccount(siteName);
714                         this.DisplaySubscribes(siteName);
715                     }
716 
717                     Console.WriteLine(this.wbrConnexion.Url);
718                 }
719             }
720 
721             private void ClearCookies()
722             {
723                 this.wbrConnexion.Navigate("javascript:void((function(){var
                    a,b,c,e,f;f=0;a=document.cookie.split('
                    ');for(e=0;e<a.length&&a[e];e++){f++;for(b='.'+location.host;b;b=b.replace(/^
                    (?:%5C.|[^%5C.]+)/,'')){for(c=location.pathname;c;c=c.replace(/.$/,'')){docum
                    ent.cookie=(a[e]+'; domain='+b+'; path='+c+'; expires='+new Date((new
                    Date()).getTime()-1e11).toGMTString());}}}})())");
724 
725             }
726         }
727     }
728 
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : Playlist class
4    * Date : 29/05/2015
5    */
6   namespace WebMediaManager.Models
7   {
8       class Playlist : Container
9       {
10          private int _indexCurrentVideo;
11
12          public int IndexCurrentVideo
13          {
14              get { return _indexCurrentVideo; }
15              set { _indexCurrentVideo = value; }
16          }
17
18          /// <summary>
19          /// Constructor
20          /// </summary>
21          /// <param name="name">playlist name</param>
22          public Playlist(string name) : base(name)
23          {
24              this.IndexCurrentVideo = 0;
25          }
26
27          /// <summary>
28          /// Next video
29          /// </summary>
30          public void Next()
31          {
32              if(this.IndexCurrentVideo != this.ListVideos.Count-1)
33                  this.IndexCurrentVideo++;
34          }
35
36          /// <summary>
37          /// Reset video
38          /// </summary>
39          public void Reset()
40          {
41              this.IndexCurrentVideo = 0;
42          }
43
44          /// <summary>
45          /// Previous video
46          /// </summary>
47          public void Previous()
48          {
49              if (this.IndexCurrentVideo != 0)
50                  this.IndexCurrentVideo--;
51          }
52      }
53  }
```

```
1   /*
2    * Author : JP. Froelicher
3    * Description : SearchChannels
4    * Date : 16/04/2015
5    */
6   using System.Runtime.Serialization;
7
8   namespace WebMediaManager.Structures.STwitch
9   {
10      [DataContract]
11      class SearchChannels
12      {
13          [DataContract]
14          public class SearchLinks
15          {
16              [DataMember]
17              public string self { get; set; }
18
19              [DataMember]
20              public string next { get; set; }
21          }
22
23          [DataMember]
24          public SearchLinks _links { get; set; }
25
26          [DataMember]
27          public Channel[] channels { get; set; }
28
29          [DataMember]
30          public int _total { get; set; }
31      }
32  }
33
```