```as
package
{
    import com.vaden.ui.PodcastViewer;

    import flash.display.Sprite;
    import flash.events.Event;

    [SWF(width="900",height="520",frameRate="60", backgroundColor="0xffffff")]

    public class Main extends Sprite
    {

        public function Main()
        {
            super();

            var viewer:PodcastViewer = new PodcastViewer();
            this.addChild(viewer);
        }
    }
}
```

```as
package com.vaden.events
{
    import flash.events.Event;

    public class FeedEvent extends Event
    {
        public static const LOAD_SUCCESS:String = "load_success";
        public static const LOAD_FAILED:String = "load_failed";

        public var episodeArray:Array;

        public function FeedEvent(type:String, bubbles:Boolean=false,
  cancelable:Boolean=false)
        {
            super(type, bubbles, cancelable);
        }

        public override function clone():Event
        {
            return new FeedEvent(type, bubbles,cancelable);
        }
    }
}
```

```as
package com.vaden.ui
{
    import com.vaden.vo.EpisodeVO;

    import libs.DetailView;

    public class DetailView extends libs.DetailView
    {
        private var _selectedEpisode:EpisodeVO;

        public function DetailView()
        {
            super();
        }

        public function get selectedEpisode():EpisodeVO
        {
            return _selectedEpisode;
        }

        public function set selectedEpisode(value:EpisodeVO):void
        {
            _selectedEpisode = value;
            this.tf_detailText.text = selectedEpisode.description;
        }
    }
}
```

```as
package com.vaden.ui
{
    import com.vaden.vo.EpisodeVO;

    import libs.ListItem;

    public class ListItem extends libs.ListItem
    {
        private var _episode:EpisodeVO;

        public function ListItem()
        {
            super();
        }

        public function get episode():EpisodeVO
        {
            return _episode;
        }

        public function set episode(value:EpisodeVO):void
        {
            _episode = value;
            this.tf_date.text = _episode.datePosted;
            this.tf_title.text = _episode.title;
        }

    }
}
```

```as
package com.vaden.ui
{
    import libs.ListView;

    public class ListView extends libs.ListView
    {

        public function ListView()
        {
            super();


        }


    }
}
```

```as
package com.vaden.ui
{
    import com.vaden.FeedLoader;
    import com.vaden.events.FeedEvent;
    import com.vaden.loaders.ImageLoaderBasic;
    import com.vaden.managers.SliderManager;
    import com.vaden.vo.EpisodeVO;

    import flash.display.Loader;
    import flash.display.MovieClip;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.MouseEvent;
    import flash.net.URLLoader;
    import flash.net.URLRequest;

    public class PodcastViewer extends Sprite
    {
        private var _feedLoader:FeedLoader;
        private var _videoPlayer:VideoPlayer;
        private var _sm:SliderManager;
        private var _episodeVO:EpisodeVO;
        private var _listView:ListView;
        private var _detailView:DetailView;
        private var _holder:MovieClip;
        private var _item:ListItem;

        public function PodcastViewer()
        {
            super();
            loadFeed();
        }

        private function loadFeed():void
        {
            //load the feed to the podcast into the FeedLoader class...
            _feedLoader = new FeedLoader("http://wddbs.com/DFP/proxy.php?url=http://feeds.feedburner.com/TEDTalks_video");

            //listen for the FeedEvent to be successful and then run the onLoad function
            _feedLoader.addEventListener(FeedEvent.LOAD_SUCCESS,onLoad);
        }

        private function onLoad(e:FeedEvent):void
```

```
44          {
45              //instantitate the VideoPlayer
46              _videoPlayer = new VideoPlayer();
47              this.addChild(_videoPlayer);
48
49              //loads the first episode from the episodeArray into the videoPlayer
50              _videoPlayer.episode = _feedLoader.episodeArray[0].videoURL;
51              _videoPlayer.title = _feedLoader.episodeArray[0].title;
52              _videoPlayer.loadVideo();
53
54              createList();
55              createDescription();
56          }
57
58          private function createList():void
59          {
60              //instantitate the listView and add to the display
61              _listView = new ListView();
62              _listView.x = 617.85;
63              _listView.y = 40;
64              this.addChild(_listView);
65
66              //enable the mask for the list view
67              _listView.mask = _listView.mc_mask;
68
69              //create a holder for each individual listView item
70              _holder = new MovieClip();
71              _listView.addChild(_holder);
72
73
74              //set up the slider for the listView items and track the change event
75              _sm = new SliderManager(true);
76              _sm.setUpAssets(_listView.mc_slider.track,
  _listView.mc_slider.handle);
77              _sm.addEventListener(Event.CHANGE, onComplete);
78
79              //loop through the items in the episode array and add to the list
  view
80              var yPos:Number = 6;
81              for each (var episode:EpisodeVO in _feedLoader.episodeArray)
82              {
83                  _item = new ListItem();
84                  _item.y = yPos;
85                  _item.x = 15;
86                  _item.episode = episode;
```

```
 87                 _item.buttonMode = true;
 88                 _item.addEventListener(MouseEvent.CLICK, clickEpisode);
 89 
 90                 //create a loader for each of the images
 91                 var loader:Loader = new Loader();
 92                 loader.load(new URLRequest(_item.episode.thumbURL));
 93                 loader.contentLoaderInfo.addEventListener(Event.COMPLETE,
    imageLoaded);
 94                 loader.y = yPos + 3;
 95                 loader.x = 18;
 96 
 97                 yPos += _item.height + 10;
 98 
 99                 _holder.addChild(_item);
100             }
101         }
102 
103         private function imageLoaded(e:Event):void
104         {
105             var images:Loader = Loader(e.target.loader);
106             _holder.addChild(images);
107         }
108 
109         private function createDescription():void
110         {
111             //instantitates the DetailView
112             _detailView = new DetailView();
113             _detailView.x = 0;
114             _detailView.y = 415;
115             this.addChild(_detailView);
116 
117             //adds the first episode description to the DetailView
118             _detailView.selectedEpisode = _feedLoader.episodeArray[0];
119         }
120 
121         private function onComplete(e:Event):void
122         {
123             var sm:SliderManager = SliderManager(e.currentTarget);
124             _holder.y = -(_holder.height - _listView.mask.height + 10) *
    sm.percent;
125         }
126 
127         private function clickEpisode(e:MouseEvent):void
128         {
129             _videoPlayer.episodeVO = e.currentTarget.episode;
```

```
130             _detailView.selectedEpisode = e.currentTarget.episode;
131         }
132     }
133 }
```

```as
package com.vaden.ui
{
    import com.vaden.managers.RollManager;
    import com.vaden.managers.SliderManager;
    import com.vaden.vo.EpisodeVO;
    import flash.events.Event;
    import flash.events.MouseEvent;
    import flash.media.Sound;
    import flash.media.SoundChannel;
    import flash.media.SoundTransform;
    import flash.net.NetConnection;
    import flash.net.NetStream;
    import flash.net.URLRequest;
    import flash.net.navigateToURL;

    import libs.VideoPlayer;

    public class VideoPlayer extends libs.VideoPlayer
    {
        private var _episode:String;
        private var _netStream:NetStream;
        private var _netConnection:NetConnection;
        private var _client:Object = {};
        private var _slider:SliderManager;
        private var _scrubber:SliderManager;
        private var _soundTransform:SoundTransform;
        private var _duration:Number;
        private var _scrubbing:Boolean = false;
        private var _episodeVO:EpisodeVO;
        private var _title:String ="";

        public function VideoPlayer()
        {
            super();
            init();
            loadVideo();
        }
        private function init():void
        {
            mc_playBtn.gotoAndStop(10);
            mc_playBtn.buttonMode = true;
            mc_playBtn.mouseChildren = false;
            mc_rewind.buttonMode = true;
            mc_mute.stop();
            mc_mute.buttonMode = true;
```

```
46
47              mc_playBtn.addEventListener(MouseEvent.CLICK, onPlay);
48              mc_rewind.addEventListener(MouseEvent.CLICK, onRewind);
49              video.mask = video_mask;
50
51              _soundTransform = new SoundTransform(.5);
52
53              this.mc_mute.addEventListener(MouseEvent.CLICK, onMute);
54
55              _slider = new SliderManager();
56              _slider.setUpAssets(this.volume_slider.track,
    this.volume_slider.handle);
57              _slider.addEventListener(Event.CHANGE, onChange);
58
59
60              _scrubber = new SliderManager();
61              _scrubber.setUpAssets(this.scrubber.track, this.scrubber.handle);
62              _scrubber.addEventListener(Event.CHANGE, scrubberChange);
63
64              scrubber.addEventListener(MouseEvent.MOUSE_DOWN, startScrub);
65              scrubber.addEventListener(MouseEvent.MOUSE_UP, stopScrub);
66
67              this.addEventListener(Event.ENTER_FRAME, update);
68
69              this.tf_link.buttonMode = true;
70              this.tf_link.addEventListener(MouseEvent.CLICK, onClick);
71          }
72
73          private function onClick(e:MouseEvent):void
74          {
75              navigateToURL( new URLRequest("http://feeds.feedburner.com/
    TEDTalks_video"));
76          }
77
78          private function onChange(e:Event):void
79          {
80              _soundTransform.volume = _slider.percent;
81              _netStream.soundTransform = _soundTransform;
82          }
83
84          private function scrubberChange(e:Event):void
85          {
86                  var percent:Number = _scrubber.percent;
87                  _netStream.seek(_duration * percent);
88          }
```

```
89
90          private function startScrub(e:MouseEvent):void
91          {
92              _scrubbing = true;
93          }
94
95          private function stopScrub(e:MouseEvent):void
96          {
97              _scrubbing = false;
98          }
99
100         private function onRewind(e:MouseEvent):void
101         {
102             _netStream.seek(0);
103             _netStream.pause();
104             mc_playBtn.gotoAndStop(1);
105         }
106
107         private function onPlay(e:MouseEvent):void
108         {
109             _netStream.togglePause();
110             if (mc_playBtn.currentFrame == 1)
111             {
112                 mc_playBtn.gotoAndStop(10);
113             }
114             else
115             {
116                 mc_playBtn.gotoAndStop(1);
117             }
118         }
119
120         private function onMute(e:MouseEvent):void
121         {
122             if(mc_mute.currentFrame == 1)
123             {
124                 mc_mute.gotoAndStop(2);
125                 _soundTransform.volume = 0;
126                 _netStream.soundTransform = _soundTransform;
127                 _slider.percent = 0;
128             }
129             else
130             {
131                 mc_mute.gotoAndStop(1);
132                 _soundTransform.volume = .5;
133                 _netStream.soundTransform = _soundTransform;
```

```
134                 _slider.percent = .5;
135             }
136         }
137 
138         public function loadVideo():void
139         {
140             //set up a net connection and instantiate the stream and specify the net connection to use
141             _netConnection = new NetConnection();
142             _netConnection.connect(null);
143             _netStream = new NetStream(_netConnection);
144 
145             //attach the net stream to the video container object
146             video.attachNetStream(_netStream);
147 
148             //using getter.setter to control the episode being played...
149             _netStream.play(_episode);
150 
151             _netStream.client = _client;
152             _client.onMetaData = metaData;
153         }
154 
155         private function update(e:Event):void
156         {
157             if (!_scrubbing)
158             {
159                 var percent:Number = _netStream.time / _duration;
160                 scrubber.handle.x = scrubber.track.width * percent;
161             }
162         }
163 
164         private function metaData(info:Object):void
165         {
166             _duration = info["duration"];
167         }
168 
169         public function get episode():String
170         {
171             return _episode;
172         }
173 
174         public function set episode(value:String):void
175         {
176             _episode = value;
177         }
```

```
178
179         public function get title():String
180         {
181             return _title;
182         }
183
184         public function set title(value:String):void
185         {
186             _title = value;
187             this.tf_Title.text = _title;
188         }
189
190         public function set episodeVO(value:EpisodeVO):void
191         {
192             _episodeVO = value;
193             _episode = _episodeVO.videoURL;
194             this.tf_Title.text = _episodeVO.title;
195             _netStream.play(_episode);
196         }
197     }
198 }
```

```as
package com.vaden.vo
{
    public class EpisodeVO
    {
        public var title:String;
        public var videoURL:String;
        public var description:String;
        public var uniqueID:String;
        public var datePosted:String;
        public var thumbURL:String;
        public var duration:String;

        public function EpisodeVO()
        {
        }
    }
}
```

```as
package com.vaden
{
    import com.vaden.events.FeedEvent;
    import com.vaden.vo.EpisodeVO;

    import flash.errors.IOError;
    import flash.events.Event;
    import flash.events.EventDispatcher;
    import flash.events.IOErrorEvent;
    import flash.net.URLLoader;
    import flash.net.URLRequest;

    public class FeedLoader extends EventDispatcher
    {
        public var feedURL:String;
        public var episodeArray:Array;

        public function FeedLoader(feedURL:String)
        {
            //create a URL loader and load in the data from the supplied URL
            var urlLoader:URLLoader = new URLLoader();
            urlLoader.load(new URLRequest(feedURL));

            //register the complete event
            urlLoader.addEventListener(Event.COMPLETE, onSuccess);
            urlLoader.addEventListener(IOErrorEvent.IO_ERROR, onFailure);
        }

        private function onSuccess(e:Event):void
        {
            var evt:FeedEvent = new FeedEvent(FeedEvent.LOAD_SUCCESS);

            //create the namespaces used in the RSS feed for the TED podcasts
            var itunes:Namespace = new Namespace("http://www.itunes.com/dtds/podcast-1.0.dtd");
            var media:Namespace = new Namespace("http://search.yahoo.com/mrss/");
            var feedburner:Namespace = new Namespace("http://rssnamespace.org/feedburner/ext/1.0");

            //cast the data as XML
             var xmlData:XML = XML(e.target.data);

            //place the data in the episodeArray
            episodeArray = [ ];

```

```
44              for each (var data:XML in xmlData.channel.item)
45              {
46                  var episodeVO:EpisodeVO = new EpisodeVO;
47                  episodeVO.datePosted = data.pubDate;
48                  episodeVO.description = data.itunes::summary;
49                  episodeVO.duration = data.itunes::duration;
50                  episodeVO.thumbURL = data.media::thumbnail.@url;
51                  episodeVO.title = data.itunes::subtitle;
52                  episodeVO.videoURL = data.feedburner::origEnclosureLink;
53                  episodeArray.push(episodeVO);
54              }
55
56              dispatchEvent(evt);
57          }
58
59          private function onFailure(e:IOError):void
60          {
61              var evt:FeedEvent = new FeedEvent(FeedEvent.LOAD_FAILED);
62          }
63      }
64  }
```