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

    import flash.display.Sprite;
    import flash.events.MouseEvent;


    [SWF(height="700",width="900")]

    public class VideoCube extends Sprite
    {
        public function VideoCube()
        {
            init();
        }

        private function init():void
        {
            var baseView:VideoCubeBase = new VideoCubeBase();
            addChild(baseView);
            baseView.x = 18.75;
            baseView.y = 115.00;


        }
    }
}
```

```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 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 flash.events.MouseEvent;

    import libs.InputView;

    public class InputView extends libs.InputView
    {
        public function InputView()
        {
            super();

            this.btn_submit.buttonMode = true;

            this.tf_error.text = "";

        }

    }
}
```

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

    import libs.ResultItem;

    public class ResultItem extends libs.ResultItem
    {
        private var _video:VideoVO;

        public function ResultItem()
        {
            super();
        }

        public function get video():VideoVO
        {
            return _video;
        }

        public function set video(value:VideoVO):void
        {
            _video = value;
            this.tf_description.text = _video.description;
            this.tf_link.htmlText = "<a href='"+ _video.videoURL + "'>"+ _video.videoURL +" </a>";
        }
    }
}
```

```as
package com.vaden.ui
{
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.EventDispatcher;
    import flash.events.MouseEvent;
    import flash.geom.Rectangle;

    /**
     * The SliderManager object uses Aggregation to work with two MovieClip graphics that can be
     * used as parts of a Slider system.  When the user interacts with the SliderManager by changing the handle position,
     * the slider object will dispatch a flash.events.Event event of a type Event.CHANGE.
     *
     * @example The following example shows how to pass in a reference to the handle and track objects.
     *
     * <listing version="3.0">
     *
     * var asset:MovieClipAsset = new MovieClipAsset();
     * this.addChild( asset );
     *
     * var slider:SliderManager = new SliderManager();
     * slider.addEventListener(Event.CHANGE, slider_changeHandler);
     * slider.setUpAssets(asset.track, asset.handle);
     *
     * function slider_changeHandler(event:Event):void
     * {
     *      trace(slider.percent);
     * }
     *
     * </listing>
     */
    public class SliderManager extends EventDispatcher
    {
        protected var _track:Sprite;
        protected var _handle:Sprite;

        private var _percent:Number = 0;
        private var _potential:Number;

        protected var _vertical:Boolean = false;

```

```
42          /**
43           * Creates a SliderManager
44           * @param vertical Sets the SliderManager to vertical or horizontal.
45           * @default false
46           *
47           */
48          public function SliderManager(vertical:Boolean = false)
49          {
50              _vertical = vertical;
51          }
52          /**
53           * Sets up the handle and the track MovieClips and associates them with
    the SliderManager.
54           * @param track The draggable area of the SliderManager UI
55           * @param handle The draggable element or "handle" of the SliderManager
56           *
57           */
58          public function setUpAssets( track:Sprite, handle:Sprite):void
59          {
60              _track = track;
61              _handle = handle;
62
63              _handle.buttonMode = true;
64              _handle.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
65          }
66          private function onMouseDown(e:MouseEvent):void
67          {
68              _handle.stage.addEventListener(MouseEvent.MOUSE_UP, onMouseUp);
69
70              if(_vertical)
71              {
72                  _handle.startDrag(false, new Rectangle(0,0,0, (_track.height -
    _handle.height)));
73              }else{
74                  _handle.startDrag(false, new Rectangle(0,0,(_track.width -
    _handle.width), 0));
75              }
76              _handle.addEventListener(Event.ENTER_FRAME, calcPercent);
77          }
78          private function onMouseUp(e:MouseEvent):void
79          {
80              _handle.stopDrag();
81              _handle.stage.removeEventListener(MouseEvent.MOUSE_UP, onMouseUp);
82              _handle.removeEventListener(Event.ENTER_FRAME, calcPercent);
83          }
```

```
84          private function calcPercent(e:Event):void
85          {
86              var prc:Number;
87
88              if(_vertical)
89              {
90                  prc = _handle.y/(_track.height -_handle.height);
91              }else{
92                  prc = _handle.x / (_track.width - _handle.width);
93              }
94              if (_percent != prc)
95              {
96                  _percent = prc;
97                  var evt:Event = new Event(Event.CHANGE);
98                  this.dispatchEvent(evt);
99              }
100         }
101         /**
102          * Moves the handle to the appropriate y position based on the percent
    property
103          *
104          */
105         protected function updateHandle():void
106         {
107             if (_handle && _track)
108             {
109                 if (_vertical)
110                 {
111                     _handle.y = (_track.height - _handle.height) * _percent;
112                 }else
113                 {
114                     _handle.x = (_track.width - _handle.width) * _percent;
115                 }
116             }
117         }
118         /**
119          *
120          * The percent of the handle's position in comparison with the track's
    height
121          *
122          */
123         public function get percent():Number
124         {
125             return _percent;
126         }
```

```
127         /**
128          *
129          * @private
130          *
131          */
132         public function set percent( value:Number):void
133         {
134             if (value >1)
135             {
136                 value = 1;
137             }else if (value < 0)
138             {
139                 value = 0;
140             }
141             _percent = value;
142             updateHandle();
143         }
144         /**
145          *
146          * Determines if the SliderManager is displaying as a horizontal or
    vertical "slider"
147          * @default false
148          */
149         public function get vertical():Boolean
150         {
151             return _vertical;
152         }
153         /**
154          *
155          * @private
156          *
157          */
158         public function set vertical(value:Boolean):void
159         {
160             _vertical = value;
161         }
162     }
163 }
```

```as
package com.vaden.ui
{
    import com.vaden.FeedLoader;
    import com.vaden.events.FeedEvent;
    import com.vaden.ui.InputView;
    import com.vaden.ui.ResultItem;
    import com.vaden.ui.SliderManager;
    import com.vaden.vo.VideoVO;

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

    import libs.NewSearchBtn;
    import libs.ResultView;
    import libs.VideoCubeBase;

    public class VideoCubeBase extends libs.VideoCubeBase
    {
        private var _inputView:InputView;
        private var _feedLoader:FeedLoader;
        private var _resultView:ResultView;
        private var _holder:MovieClip;
        private var _item:ResultItem;
        private var _btn:NewSearchBtn;
        private var _sm:SliderManager;

        public function VideoCubeBase()
        {
            super();
            initButtons();
            createInputView();
        }

        private function onLoad(e:FeedEvent):void
        {
            _inputView.tfInput.text = "";

            this.removeChild(_inputView);

            _resultView = new ResultView();
            _resultView.x = 265.35;
```

```
46              _resultView.y = 78.75;
47              this.addChild(_resultView);
48
49              _resultView.mask = _resultView.mc_mask;
50
51              _holder = new MovieClip();
52              _resultView.addChild(_holder);
53
54              _sm = new SliderManager(true);
55              _sm.setUpAssets(_resultView.mc_slider.mc_track,
  _resultView.mc_slider.mc_handle);
56              _sm.addEventListener(Event.CHANGE, onChange);
57
58              _resultView.mc_slider.mc_handle.addEventListener
  (MouseEvent.MOUSE_DOWN, removeMove);
59              _resultView.mc_slider.mc_handle.addEventListener
  (MouseEvent.MOUSE_UP, addMove);
60
61              _btn = new NewSearchBtn();
62              this.addChild(_btn);
63              _btn.gotoAndStop(1);
64              _btn.x = 609.95;
65              _btn.y = 312.95;
66
67              _btn.addEventListener(MouseEvent.CLICK, onNewSearch);
68              _btn.addEventListener(MouseEvent.MOUSE_OVER, hoverNewSearch);
69              _btn.addEventListener(MouseEvent.MOUSE_OUT, hoverNewSearchOut);
70
71              var yPos:Number = 10;
72              for each (var video:VideoVO in _feedLoader.videoArray)
73              {
74                  _item = new ResultItem();
75                  _item.y = yPos;
76                  _item.x = 10;
77                  _item.video = video;
78                  _item.buttonMode = true;
79
80                  var loader:Loader = new Loader();
81                  loader.load(new URLRequest(_item.video.thumbURL));
82                  loader.contentLoaderInfo.addEventListener(Event.COMPLETE,
  imageLoaded);
83                  loader.y = yPos + 3;
84                  loader.x = 10;
85
86                  yPos += _item.height + 10;
```

```
87
88                  _holder.addChild(_item);
89              }
90          }
91
92          private function removeMove(e:MouseEvent):void
93          {
94              this.removeEventListener(MouseEvent.MOUSE_DOWN, onMove);
95          }
96
97          private function addMove(e:MouseEvent):void
98          {
99              this.addEventListener(MouseEvent.MOUSE_DOWN, onMove);
100         }
101
102         private function imageLoaded(e:Event):void
103         {
104             var images:Loader = Loader(e.target.loader);
105             _holder.addChild(images);
106         }
107
108         private function initButtons():void
109         {
110             this.mc_Close.buttonMode = true;
111             this.mc_Min.buttonMode = true;
112             this.mc_Close.addEventListener(MouseEvent.CLICK, onClose);
113             this.mc_Close.addEventListener(MouseEvent.MOUSE_OVER, hoverClose);
114             this.mc_Close.addEventListener(MouseEvent.MOUSE_OUT, hoverCloseOut);
115             this.mc_Min.addEventListener(MouseEvent.MOUSE_OUT, hoverMinOut);
116             this.mc_Min.addEventListener(MouseEvent.MOUSE_OVER, hoverMin);
117             this.mc_Min.addEventListener(MouseEvent.CLICK, onMin);
118             this.addEventListener(MouseEvent.MOUSE_DOWN, onMove);
119             this.mc_Close.stop();
120             this.mc_Min.stop();
121
122         }
123
124         private function createInputView():void
125         {
126             _inputView = new InputView();
127             addChild(_inputView);
128             _inputView.x = 373.45;
129             _inputView.y = 161.50;
130             _inputView.btn_submit.stop();
131             _inputView.btn_submit.addEventListener(MouseEvent.MOUSE_OVER,
```

```
hoverBtnSearch);
```

```
172         private function onMin(e:MouseEvent):void
173         {
174             this.stage.nativeWindow.minimize();
175         }
176 
177         private function onMove(e:MouseEvent):void
178         {
179             this.stage.nativeWindow.startMove();
180         }
181 
182         private function hoverClose(e:MouseEvent):void
183         {
184                 this.mc_Close.gotoAndStop(2);
185         }
186 
187         private function hoverMin(e:MouseEvent):void
188         {
189                 this.mc_Min.gotoAndStop(2);
190         }
191 
192         private function hoverMinOut(e:MouseEvent):void
193         {
194             this.mc_Min.gotoAndStop(1);
195         }
196 
197         private function hoverCloseOut(e:MouseEvent):void
198         {
199             this.mc_Close.gotoAndStop(1);
200         }
201 
202         private function onNewSearch(e:MouseEvent):void
203         {
204             this.removeChild(_resultView);
205             this.removeChild(_btn);
206             createInputView();
207         }
208 
209         private function hoverNewSearch(e:MouseEvent):void
210         {
211             _btn.gotoAndStop(2);
212         }
213 
214         private function hoverNewSearchOut(e:MouseEvent):void
215         {
216             _btn.gotoAndStop(1);
```

```
217         }
218 
219         private function onChange(e:Event):void
220         {
221             var sm:SliderManager = SliderManager(e.currentTarget);
222             _holder.y = -(_holder.height - _resultView.mask.height + 10) *
    sm.percent;
223         }
224     }
225 }
```

```as
package com.vaden.vo
{
    public class VideoVO
    {
        public var description:String;
        public var videoURL:String;
        public var thumbURL:String;

        public function VideoVO()
        {
        }
    }
}
```

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

    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 videoArray:Array;

        public function FeedLoader(feedURL:String)
        {
            var ld:URLLoader = new URLLoader();
            ld.load(new URLRequest(feedURL));
            ld.addEventListener(Event.COMPLETE, onSuccess);
            ld.addEventListener(IOErrorEvent.IO_ERROR, onFailure);
        }

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

            var atom:Namespace = new Namespace("http://www.w3.org/2005/Atom");
            var media:Namespace = new Namespace("http://search.yahoo.com/mrss/");

            var xmlData:XML = XML(e.target.data);

            videoArray = [ ];

            for each (var data:XML in xmlData.atom::entry)
            {
                var videoVO:VideoVO = new VideoVO;
                videoVO.description = data.atom::content;
                videoVO.thumbURL = data.media::thumbnail[0].@url;
                videoVO.videoURL = data.media::player.@url;
                videoArray.push(videoVO);

                trace (videoVO.description);
```

```
46                  trace (videoVO.thumbURL);
47                  trace (videoVO.videoURL);
48              }
49
50              dispatchEvent(evt);
51
52              trace (xmlData);
53          }
54
55          private function onFailure(e:IOError):void
56          {
57              var evt:FeedEvent = new FeedEvent(FeedEvent.LOAD_FAILED);
58          }
59      }
60  }
```