```
1 package
2 {
3     import com.vaden.Controls;
4     import com.vaden.ImageViewer;
5     import com.vaden.vo.ImageVO;
6
7     import flash.display.Sprite;
8     import flash.events.Event;
9     import flash.events.MouseEvent;
10    import flash.net.URLLoader;
11    import flash.net.URLRequest;
12
13
14    [SWF(width="760",height="500",frameRate="30", backgroundColor="0x000000")]
15
16    public class Main extends Sprite
17    {
18        private var _iv:ImageViewer;
19        private var _galleryData:Array;
20        //sets a var _currentImage which tracks which image is currently being
  viewed
21        private var _currentImage:Number = 0;
22
23        public function Main()
24        {
25            //loads the xml data from the xml file and sets function onParse to
  process the data
26            var urlLoader:URLLoader = new URLLoader();
27            urlLoader.load(new URLRequest("assets/xml/gallery.xml"));
28            urlLoader.addEventListener(Event.COMPLETE, onParse);
29        }
30
31        private function onParse(e:Event):void
32        {
33            //creates an array _galleryData to hold all of the data from the xml
  file
34            _galleryData = [ ];
35            var xmlData:XML = XML(e.target.data);
36
37            //takes the xml data and assigns a value object to each item and then
  pushes these vo's into the array _galleryData
38            for each (var galleryList:XML in xmlData.IMAGE)
39            {
40                var vo:ImageVO = new ImageVO();
41                vo.caption = galleryList.@CAPTION;
```

```
42                  vo.image = galleryList.@URL;
43                  _galleryData.push(vo);
44              }
45
46              //creates the ImageViewer class, assigns the path of the images and
  sets the image list as the image vo inside the _galleryData array
47              _iv = new ImageViewer();
48              _iv.path = "assets/images/";
49              _iv.imageList = [_galleryData[_currentImage].image];
50              _iv.display();
51              _iv.x = 30;
52              _iv.y = 20;
53              this.addChild(_iv);
54
55              //creates the controls from the Controls class and adds to stage
56              var controls:Controls = new Controls();
57              controls.x = stage.stageWidth/2 - (controls.width/2);
58              controls.y = stage.stageHeight - (controls.height);
59              this.addChild(controls);
60
61              //sets the caption of the controls class to the caption vo inside the
  _galleryData
62              controls.caption_txt.text = _galleryData[_currentImage].caption + "";
63
64              //assigns the mouse event to each button on the controls and starts
  the onNext and onBack functions
65              controls.forward_btn.addEventListener(MouseEvent.CLICK, onNext);
66              controls.back_btn.addEventListener(MouseEvent.CLICK, onBack);
67          }
68
69          //tells the imageViewer to go to the next image in the imageList array
70          private function onNext(e:Event):void
71          {
72              trace(_galleryData);
73              _iv.next();
74          }
75
76          //tells the imageViewer to go to the previous image in the imageList array
77          private function onBack(e:Event):void
78          {
79              _iv.previous();
80          }
81      }
82  }
```

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

    public class ImageEvent extends Event
    {
        public static const IMAGE_LOADED:String = "image_loaded";

        public var image:Bitmap;

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

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

```as
package com.vaden.vo
{

    public class ImageVO
    {
        public var image:String;
        public var caption:String;

        public function ImageVO()
        {
        }
    }
}
```

```as
package com.vaden
{
    import com.vaden.managers.RollManager;

    import flash.events.Event;

    import libs.Controls;

    public class Controls extends libs.Controls
    {

        public function Controls()
        {
            super();

            var rm:RollManager = new RollManager(this.back_btn);
            var rm1:RollManager = new RollManager(this.forward_btn);

            this.caption_txt.text = "";


        }
    }
}
```

```as
package com.vaden
{
    import com.vaden.events.ImageEvent;
    import flash.display.Loader;
    import flash.events.Event;
    import flash.events.EventDispatcher;
    import flash.net.URLRequest;


    public class ImageLoaderBasic extends EventDispatcher
    {
        private var ld:Loader;

        public function ImageLoaderBasic(file:String)
        {
            super();
            ld = new Loader();
            ld.load(new URLRequest(file));
            //complete event goes on the contentLoaderInfo property not the loader class
            ld.contentLoaderInfo.addEventListener(Event.COMPLETE, onLoad);

        }

        private function onLoad(e:Event):void
        {
            //make a new custom event and dispatch it here
            var evt:ImageEvent = new ImageEvent(ImageEvent.IMAGE_LOADED);
            //attach the bitmap that was loaded before dispatching the event
            evt.image = e.target.content;
            dispatchEvent(evt);

            //need to remove the event listener and urge elements from RAM
            ld.contentLoaderInfo.removeEventListener(Event.COMPLETE, onLoad);
            ld.unload();
            ld = null;
        }
    }
}
```

```as
package com.vaden
{
    import com.vaden.events.ImageEvent;

    import flash.display.Sprite;

    public class ImageViewer extends Sprite
    {
        private var _imageList:Array;
        private var _path:String;
        private var _currentImage:int;
        private var _ld:ImageLoaderBasic;

        public function ImageViewer()
        {
            super();
            init();
        }

        private function init():void
        {
            _imageList = [ ];
            _currentImage = 0;
        }

        private function loadImg():void
        {
            _ld = new ImageLoaderBasic(_path + _imageList[_currentImage]);
            _ld.addEventListener(ImageEvent.IMAGE_LOADED, onLoad);
        }

        private function onLoad(e:ImageEvent):void
        {
            //determine if an image is already there and remove it
            if (this.numChildren > 0)
            {
                this.removeChildAt(0);
            }
            this.addChild(e.image);
        }

        public function display():void
        {
            loadImg();
        }
```

```
46
47          public function next():void
48          {
49              //validate the current image property
50              _currentImage++;
51              if (_currentImage == _imageList.length)
52              {
53                  _currentImage = 0;
54              }
55              _ld = new ImageLoaderBasic(_path+_imageList[_currentImage]);
56              _ld.addEventListener(ImageEvent.IMAGE_LOADED, onLoad);
57          }
58
59          public function previous():void
60          {
61              //validate the current image property
62              _currentImage--;
63              if (_currentImage < 0)
64              {
65                  _currentImage = _imageList.length -1;
66              }
67              _ld = new ImageLoaderBasic(_path+_imageList[_currentImage]);
68              _ld.addEventListener(ImageEvent.IMAGE_LOADED, onLoad);
69          }
70
71          //set the path and set the image list
72          public function set path(value:String):void
73          {
74              _path = value;
75          }
76
77          public function set imageList(value:Array):void
78          {
79              _imageList = value;
80          }
81      }
82  }
```