



OSMF Release Samples

Walkthrough 3: Handling Different Media

Overview:

In this walkthrough you will extend the media player so it can handle different types of media. This will show how with the use of a MediaFactory and the generic MediaElement so the player can handle any of the supported media types without having to know what kind of media is being loaded in. This way the player isn't tied to the VideoElement for instance that won't be able to play a SWF or a dynamic multi-bitrate stream.

Objectives:

- Gain a better understanding of specific media element types and their limitation when explicitly created
- Implement support to the media playback application to handle different types of media (SWF, Video, and Audio) through the use a MediaFactory
- Use a F4M external XML based file to define and load the appropriate content
- Implement Dynamic Streaming via the use of a F4M file

Setup

1. Open the file WT03_HandlingDifferentMedia.as in the {SAMPLES_PROJECT}/src directory.
2. Set the class file as the application file to compile. There are two different ways of doing this depending on which program you are building your application in.

Flash Builder

Right-click the WT03_HandlingDifferentMedia.as file and select Set as Default Application from the context menu that appears. This will add the project to the list of compilable applications. A blue dot on the file icon indicates that the file is the default application file.

Flash Professional

Open the OSMF_SampleTemplate.fla and save it as WT03_HandlingDifferentMedia.fla. Then change the document class for the file (in the Properties panel) to WT03_HandlingDifferentMedia.

In this start file we added 2 additional paths for media. One which is a SWF and another that is an F4M file for dynamic streaming. Let's try to play the SWF in the media player's current state.

Handling Different Types of Media

3. Change the variable that is being passed to the URLResource constructor from STREAMING_MP4_PATH to LOCAL_SWF. Save and run the application. Nothing should happen.

The VideoElement object doesn't know how to deal with a SWF.

Using the MediaFactory

4. Create a new class property named mediaFactory and typed as a DefaultMediaFactory.

```
public var player:MediaPlayer;
public var container:MediaContainer;
public var mediaFactory:DefaultMediaFactory;
```

5. Inside the initPlayer() method, remove the line that creates the netLoader object. Replace that line with a line that sets the mediaFactory property equal to a new instance of a DefaultMediaFactory object.

```
var resource:URLResource = new URLResource( LOCAL_SWF );
mediaFactory = new DefaultMediaFactory();
```

6. Next, replace the code that sets the element variable with code that calls the createMediaElement() method on the mediaFactory property - you will need to pass the resource variable to the createMediaElement() method. You will also need to change the variable type from VideoElement to MediaElement. *The MediaFactory will generate the appropriate MediaElement for the content dynamically for you - thus abstracting the content type from the player capabilities.*

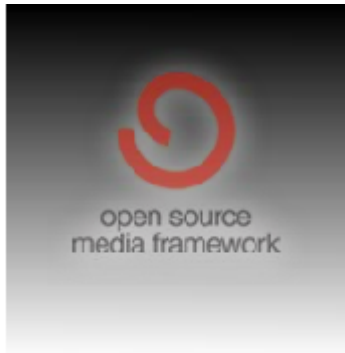
Before:

```
var element:VideoElement = new VideoElement( resource, netLoader );
```

After:

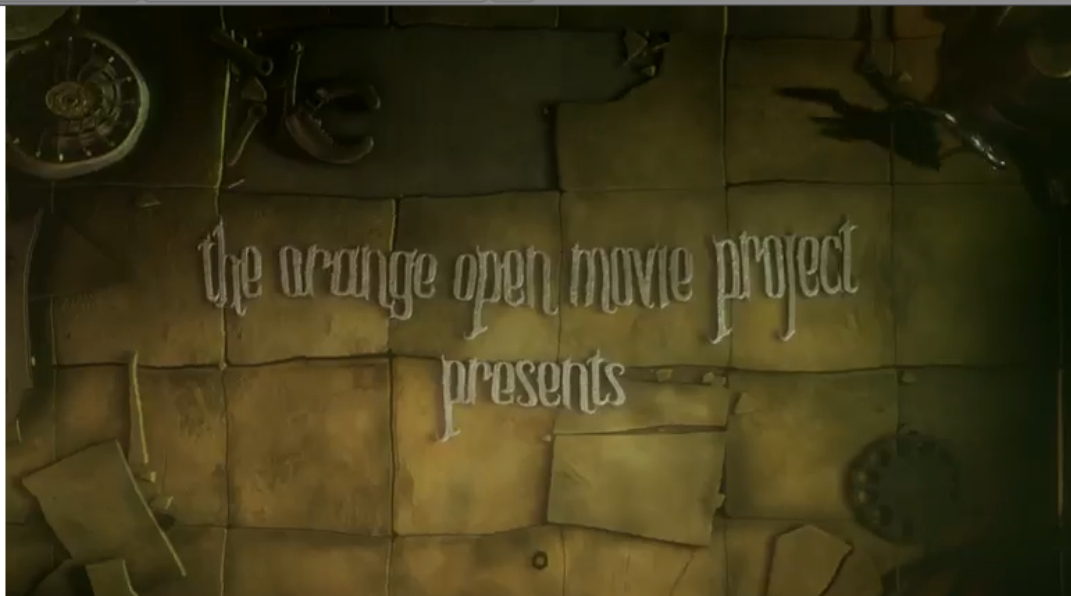
```
var element:MediaElement = mediaFactory.createMediaElement( resource );
```

7. Save the file and run the application. This time the SWF (a looping fade in of the OSMF logo with text below) should display.



OSMF reduces the complexity of player development, allowing the developer more time to focus on the overall user experience. OSMF's flexible architecture allows the developer to easily customize their player for the browser, incorporating plug-ins for advertising, reporting, and content delivery along with standard player features such as play/pause, seek, volume/mute, download progress, buffering, and bitrate switching.

8. Now, replace the LOCAL_SWF variable passed to the URLResource constructor with the DYNAMIC_STREAMING variable. This should play the dynamic stream defined by loaded F4M file.



9. The Flash Media Manifest contains information about a Flash media asset such as location of the media, multi-bitrate information, and additional metadata. The OSMF parses this XML based file to create a DynamicStreamResource.

NOTE: If you'd like to see how to manually define a dynamic streaming resource, a sample has been provided in the DynamicStreamSample.as file.

