

Sonettic-Cinema

Player Modules Development

© 2009 Sonettic

www.sonettic-cinema.com

Modules Development Overview

All Sonettic Cinema Players have support External Modules Integration. You have all you need to create your own unique modules. You might create modules you wish. You have no limits. Players Kits have source files figure it out. Also figure out External Module Template.

Module should be extended `SntInstanceModule` class:

Class Name: **extends `SntInstanceModule`**
Package: **`com.sonettic.externalmodules.SntInstanceModule`**

Figure out `MyModule` class:

Class Name: **`MyModule`**
Package: **`com.myproject`**

Module Connector Overview

Class Name: **`SntModulesConnector`**
Package: **`com.sonettic.modules`**

`SntModulesConnector` does connect module and set initialized data. Module might get all necessary data from `SntModulesConnector`. For more details discover this class.

Cinema Player Event Model overview and exercise

Sonettic Cinema HD Player built on Message Model. Cinema Player have one main Event Connector. This connector instance set to all modules of player. Modules uses `MediaConnector` to send and receive messages.

Event Connector is event broadcaster.

Class name: **`MediaConnector`**
Package: **`com.sonettic.connector`**

`MediaConnector` broadcast events of `MediaEvent` type:

Class name: **`MediaEvent`**
Package: **`com.sonettic.media`**

`MediaEvent` types list

Class name: **`ConnectorEventsList`**
Package: **`com.sonettic.connector`**

All about MediaEvent object and Event Type List you can get from [Player Events PDF](#).

Exercise

Create **Button** of external GUI (for internal development typed all properties, received and returned values).

```
package com.sonettic.GUI{

    import flash.display.Sprite;
    import flash.events.*;

    class MyPlayPauseButton extends Sprite{

        private var Connector:*;
        private var BValue:uint;
        private var EventName:String;

        function MyPlayPauseButton(){
            value = 0;
        }

        /**
         * Set Media Event Connector
         * @return mconnector:MediaConnector
         *
         * @see com.sinema.cconnector.MediaConnector
         * @see com.sonettic.media.MediaEvent
         * @see com.sonettic.connector.ConnectorEventsList
         */
        public function get connector():* {
            return Connector;
        }

        /**
         * @private
         */
        public function set connector(mconnector:*) :void {
            Connector=mconnector;
        }

        /**
         * Get Button Value
         * @return uint
         */
        public function get value():uint{
            return BValue;
        }

        /**
         * @private
         */
        public function set value (val:uint):void{
            BValue = val;

            if(value == 0 ){
                EventName = "uiPlayClick";
            }
            if(value == 1 ){
                EventName ="uiPauseClick";
            }
        }
    }
}
```

```
// init all event listeners
private function initEventListeners():void{
    Connector.addEventListener("mediaPlay", mediaPlayListener);
    Connector.addEventListener("mediaPause", mediaPauseListener);
    Connector.addEventListener("mediaStop", mediaStopListener);
    Connector.addEventListener("mediaComplete", mediaStopListener);

    // click on button
    this.addEventListener(MouseEvent.CLICK, mouseClickedEventListener);
}

// mouse event listener
private function mouseClickedEventListener(e:MouseEvent):void{
    // broadcast Event
    eventDispatch(EventName, this);
}

// event listeners
private function mediaPlayListener(e:MediaEvent):void{
    value=1;
}
private function mediaPauseListener(e:MediaEvent):void{
    value=0;
}
private function mediaStopListener(e:MediaEvent):void{
    value=0;
}

//-----
//      EVENT PORT
//
// Create and transfer Object to Media Connector. Media Connector broadcast it as MediaEvent
//-----
/**
 * eventDispatch() - method create and transfer Object to Media Connector.
 *                   Media Connector broadcast it as MediaEvent
 * @param type:String - Event Type
 * @param target:~~ - event owner
 * @param data:~~ @default null
 *
 * @see com.sinema.cconnector.MediaConnector
 * @see com.sonettic.media.MediaEvent
 * @see com.sonettic.connector.ConnectorEventsList
 */
protected function eventDispatch(type:String, target:*, data:*=null):void {
    var eventObject:Object=new Object();
    eventObject.type=type;
    eventObject.target=target;
    eventObject.value=data;
    if (Connector) {
        Connector.eventsPort(eventObject);
    }
}
}
```

```
var MyPPButton:MyPlayPauseButton = new MyPlayPauseButton();

// set media connector to button ( Connector is MediaConnector and initialized earlier)
MyPPButton.connector = Connector;

// add button to some container
addChild(MyPPButton);
```

Add new button to Player Submenu

Get Player SubMenu

```
private var ModuleName:String = "MyModuleName";
private var SubMenu:*;
private var BubbleController:*;

//-----
// Get Objects from Player Objects Repository
//-----
private function getObjectFromRepository():void{

    // Method get objectRepository
    // Obhets Repository contain alot modules and objects and this module too
    // get Object from repository by Object Name
    var MObjetRepository:* = ModuleController.objectRepository;

    // get player Submenu from ObjectsRepository
    //com.sonettic.utilities.ObjectsRepository
    SubMenu = MObjetRepository.getObject("SUBMENU");
    if(SubMenu == null){
        // wait when object will be added to ObjectRepository
        // add event objectSet_ + object name
        // remove event objectDelete_ + object name
        // all objects removed objectClear
        MObjetRepository.addEventListener("objectSet_SUBMENU",
                                         submenuAddedObjectRepositoryEventListener);
    }

    //-----
    // SubMenu added to ObjectsRepository Event Listener
    //-----
    private function submenuAddedObjectRepositoryEventListener(e:Event):void{
        // create module icon and set it to player Submenu
        createModuleIcon();
    }
}
```

To add new Button in player standard submenu we have to create icons graphics contained in Sprite. If button should have two states, we should to create two icons graphics contained in Sprite.

Icon sizes 64X64

1. – state **ON** (active)

2. – state **OFF** (passive)

Go to create button

```
//-----  
// CREATE MODULE ICON IN PLAYER SUBMENU  
//-----  
private function createModuleIcon():void{  
    // if submenu is null  
    if(!SubMenu ) return;  
    //figure out button pattern at com.sonettic.GUI.icons.IconPattern;  
  
    // UIControlComander is skin controller of player  
    var SharedIcons:* = UIControlComander.getIconsLibrary();  
    // trace Shared Library Components List  
  
    // Shared Library contain buttons and components  
    trace(SharedIcons.getIconsList);  
  
    // get Play Button from Shared Library  
    var PlayIcon:*=SharedIcons.getIcon("Play");  
  
    // we should get ButtonPatter to create new Button  
    var ModuleIconInstance:* = SharedIcons.getIcon("ButtonPattern");  
  
    // set active icon graphic  
    ModuleIconInstance.activeicon = new ModuleActiveIcon();  
  
    // set passive icon graphic  
    ModuleIconInstance.passiveicon = new ModulePassiveIcon();  
  
    // set Icon Events names  
    // click event  
    ModuleIconInstance.eventName = ModuleName+"_uiIconClick";  
  
    // RollOver event  
    ModuleIconInstance.eventRollOverName = ModuleName+"_uiIconOver";  
  
    // RollOut event  
    ModuleIconInstance.eventRollOverName = ModuleName+"_uiIconOver";  
  
    // DoubleClick event  
    ModuleIconInstance.eventDoubleClickName = ModuleName+"_uiIconDoubleClick";  
  
    // set double click enable  
    // it will be set to true if boubleClickEventName set  
    //ModuleIconInstance.doubleClickEnable = false;  
  
    // Button Hilt  
    ModuleIconInstance.hint="Module Icon";  
  
    // set hint from language object  
    //ModuleIconInstance.hintName = language object field or language XML node name  
  
    // add icon to player submenu  
    //addItem(item:*, sortindex:int=100, uniqueitem:Boolean=true)  
    SubMenu.createItem(ModuleIconInstance, 0);  
}
```

Create new Bubble Control and add it to Player Bubble Controller

Bubble Control is some components added to Bubble Container through Bubble Controller.

Class Name: **BubbleController**

Package: **com.sonettic.GUI.bubblecontrols**

To create own new Bubble we should use BubblePattern:

Class Name: **BubblePattern**

Package: **com.sonettic.GUI.bubblecontrols**

```
//-----  
// Get Objects from Player Objects Repository  
//-----  
private function getObjectsFromRepository():void{  
  
    // Method get objectRepository  
    // Obhets Repository contain alot modules and objects and this module too  
    // get Object from repository by Object Name  
    var MObjetRepository:* = ModuleController.objectRepository;  
  
    //get BubbleController from ObjectsRepository  
    //BubbleController com.sonettic.GUI.bubblecontrols.BubbleController  
    //ObjectsRepository com.sonettic.utilities.ObjectsRepository  
    BubbleController = MObjetRepository.getObject( "BUBBLECONTROLLER" );  
  
    // create bubble control  
    createModuleBubble();  
}  
  
//-----  
// Get Module Bubble Control  
//-----  
private function createModuleBubble():void{  
    if(BubbleController == null) return;  
    // get Bubble Pattern from Player Bubble Controller  
    var MBubble:* = BubbleController.getBubblePattern();  
    // type 1 without bubble icon  
    // type 2 with bubble icon  
    MBubble.initPattern(1);  
    // set icon if bubble is type 2  
    MBubble.setIcon(new ModuleActiveIcon());  
    // set bubble name  
    MBubble.iname = ModuleName+"Bubble";  
    // set bubble content  
    MBubble.content= new BubbleContent();  
  
    // show bubble when event occurred  
    MBubble.showEvent = ModuleName+"_uiIconClick";  
    // add bubble to BubbleController  
    BubbleController.addBubble = MBubble;  
}
```

Add Modules to Cinema Player as external

Set External Modules List to ModuleConnector:

1. through **flashvars**
2. through **external XML Properties** file
3. **internal properties**, file **MainProperties**

property – modulelist = http://mysite.com/externalmodules.xml

External Modules List

List might contain single or multi items

```
<?xml version="1.0" encoding="UTF-8"?>
<modulelist>
<module url="externalmodules/module1.swf" initvalue="SomeValue" initvalue2="SomeValue" />
<module url="externalmodules/module2.swf" initvalue="SomeValue" initvalue2="SomeValue" />
<module url="externalmodules/module3.swf" initvalue="SomeValue" initvalue2="SomeValue" />
</modulelist>
```

```
<module url="PATH TO MODULE SWF" value1="1" value2="2" />
```

all attributes will be set to module with initializations

initvalues = AttributesValuesObject

```
/**
 * @private
 */
override public function set initvalues(val:Object):void{
    ModulesInitValue = val;
    // module url
    // ModulesInitValue.value
    // values object
    // object fields are attributes from module node
    // sample
    // ModulesInitValue.initvalue;
    // ModulesInitValue.mysomevalue;
}

// trace value1
trace (ModulesInitValue.value1);
```

Add Modules to Cinema Player as internal

Create module with unique namespace and connect you module through SntModulesConnector.

Class Name: **SntModulesConnector**

Package: **com.sonettic.modules**

```
var ModuleConnector: SntModulesConnector = SntModulesConnector.getInstance();

var ValuesObject:Object=new Object();
ValuesObject.value1=1;
ValuesObject.value2=2;
var myModule:MySuperModule = new MySuperModule();

// add module and initialization values
ModuleConnector.connectModule(myModule, ValuesObject);

// just add module
ModuleConnector.connectModule(myModule);
```