

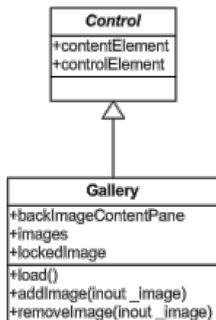
# How to do an image gallery in WRTKit using S60 5th

In this post we'll show you how to do an image gallery using the WRTKit and the new S60 5th edition WRT API.

## Extending WRTKit

In this [post](#), was showed how the developer can extend the WRTKit and to create new UI components. To do the image gallery, let's do the same thing. The image gallery will be a UI components that can be added in other applications easily.

Below you see the class diagram of the image gallery (with all methods and attributes) and how it extends the WRTKit.



Let's understand this picture:

- The **Control** class is a base class of the WRTKit. All UI components have to extend this class, directly or indirectly. There are many methods and attributes in this class, but for now we have to understand only two attributes, `contentElement` and `controlElement`.
- - The **controlElement** is the base html element of any WRT application, all UI components (Button, Label, Text and the gallery we're doing) have to be added in this attribute using the `controlElement.appendChild` method.
- - The **contentElement** is the base html element of a specific UI component. If an UI component have more than one html elements (images, links, buttons, etc), all this elements have to be added in the `contentElement`. After this, the `contentElement` has to be added in the `controlElement` attribute to be showed in the screen.
- The **Gallery** class is the image gallery we're doing. This class has three attributes (`backImageContentPane`, `images` and `lockedImage`), let's understand each one of them:
  - - **backImageContentPane** is an image html element, this is the background of the gallery, this is necessary because we can't capture mouse events that occurs on div html elements (The `contentElement` is a div html element because it will have all images of the mobile phone inside it).
  - - **images** is a array with the file path and file name of all pictures of the mobile phone. The file information about the pictures will be get using the new S60 5th edition WRT API. This'll be showed below in this post.
  - - **lockedImage** is an image html element, this is the image selected by the user. In this application the user can move any picture to any position of the Gallery component. To do this, he has to click an image (locking a image to do actions on it) and then click in another position of the gallery. The image will be moved to the new position.

Below, you can see the code of the Gallery class.

```

// Constructor.
function Gallery(id, caption, width, height) {
    if (id != UI_NO_INIT_ID) {
        this.init(id,caption, width, height);
    }
}

// Gallery inherits from Control.
Gallery.prototype = new Control(UI_NO_INIT_ID);

// Array of all images in the gallery
Gallery.prototype.images = new Array();

// Background image of the gallery
Gallery.prototype.backImageContentPane;

// The selected image by the user
Gallery.prototype.lockedImage = null;

// init method, called in the constructor
// Width and height parameters can be numeric or string.
Gallery.prototype.init = function(id,caption, width, height){
    ullogger.debug("Map.init(" + id + ", " + caption + ")");

    Control.prototype.init.call(this, id, caption);

    // Create the content element
    this.contentElement = document.createElement("div");
    this.contentElement.style.width = _width;
    this.contentElement.style.height = _height;

    // When mouse up the button over the contentElement, the image must be selected or moved to a position.
    var me = this;
    this.contentElement.onmouseup = function(_event){
        var image = _event.target;
        if(me.lockedImage == null && image != me.backImageContentPane){
            //Clicked image must be locked

```

```

        image.border = 2;
        image.style.borderColor = "#00FF00";
        me.lockedImage = image;
        var clickX = Math.abs(_event.clientX - image.x);
        var clickY = Math.abs(_event.clientY - image.y);
    }else{
        me.lockedImage.parentNode.style.left = _event.clientX - (me.lockedImage.width/2);
        me.lockedImage.parentNode.style.top = _event.clientY - (me.lockedImage.height/2);
        me.lockedImage.border = 0;
        me.lockedImage = null;
    }
};

this.backImageContentPane = document.createElement("img");
this.backImageContentPane.style.width = _width;
this.backImageContentPane.style.height = _height;
this.backImageContentPane.src = "/js/pixel.jpg";
this.backImageContentPane.border = 1;
this.contentElement.appendChild(this.backImageContentPane);

this.controlElement.appendChild(this.contentElement);
}

// Method to load all images and show the gallery
Gallery.prototype.load = function(){
    //Put all images inside the contentElement
    for(var i = 0;i < this.images.length;i++){
        var imageHTMLElement = document.createElement("img");
        imageHTMLElement.src = this.images[i];
        imageHTMLElement.width = 100;
        imageHTMLElement.height = 100;
        imageHTMLElement.border = 0;

        var divImageHTMLElement = document.createElement("div");
        divImageHTMLElement.style.position = "absolute";
        divImageHTMLElement.style.left = this.backImageContentPane.x + ((i+1) * 10);
        divImageHTMLElement.style.top = 80;
        divImageHTMLElement.appendChild(imageHTMLElement);

        this.contentElement.appendChild(divImageHTMLElement);
    }
}

// Add a new image to the Gallery.
// _image : String location of an image in the mobile phone
Gallery.prototype.addImage = function(_image){
    this.images.push(_image);
}

// Remove a image from the Gallery.
// _image : String location of an image in the mobile phone
Gallery.prototype.removeImage = function(_image){
    var indexImage = this.images.indexOf(_image);
    this.images.splice(indexImage,1);
}

// An indexOf method was created in the Array class.
Array.prototype.indexOf = function(_element){
    for(var i = 0;i < this.length;i++){
        if(this[i] == _element)
            return i;
    }
    return -1;
}

```

## Getting Media Information

How was told previously on this post, the images attribute of the Gallery class is an array with file path and name about all the pictures in the mobile phone. To get this information we can use the new S60 5th Edition WRT API.

This new API provide some interfaces to access and manage device resources, like GPS, SMS, Contacts and Media. The below code show the first step to access this resources, let's see the code and understand it.

```

try {
    var so = device.getServiceObject("Service.MediaManagement", "IDataSource");
    console.info("setup: so: %s", so);
} catch(e) {
    alert('Error:' + e);
}

// Setup input params using dot syntax
var criteria = new Object();
criteria.Type = 'FileInfo';
criteria.Filter = new Object();
criteria.Filter.FileType = 'Image';
criteria.Sort = new Object();
criteria.Sort.Key = 'FileSize';

try {
    // Media Management supports asynchronous call
    so.IDataSource.GetList(criteria, callback);
} catch (e) {
    alert ("Error: " + e);
}

```

First of all, you have to get access to S60 resources. This must be done with the **device.getServiceObject** method. This method receives two parameters: A service and an interface to be used.

- The service parameter represents the service you want to use. There are a lot of services available, but in this post we'll use only media management service. The media management service give to the developer access to media informations about any media that exist in the mobile phone (video, audio or image).
- The second parameter of **device.getServiceObject** is the access interface to get the informations about the service.

An other important line in the code is where the information of the service is retrieved. The **so.IDataSource.GetList** is the method to do this. This method retrieves a list with informations about all media files that exists in the mobile phone.

To do this job, this method receives two parameters:

- The **first parameter** is a criteria object used to filter the retrieved information (for example, you maybe want to retrieve only video files). This criteria object is also used to sort the list retrieved. In our gallery application, we filtered the retrieved information (using the criteria parameter) to get only information about image files.

```
criteria.Filter = new Object();
criteria.Filter.FileType = 'Image';
```

- The **second parameter** is a callback function. Let's understand how the callback function works and how the developer can use it:
- This function is called by the device (not by the developer) when the information is retrieved. The developer have to implement this function, but never call it. It has three attributes, the last attribute is the more important because it has all the retrieved information after the criteria be applied. Below you can see the code of this callback function

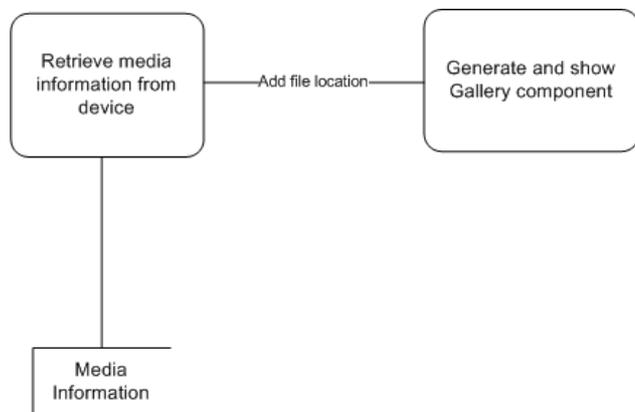
```
// This is the asynchronous callback handler
function callback(transId, eventCode, result)
{
    console.info("callback: transId: %d eventCode: %d result.ErrorCode: %d", transId, eventCode, result.ErrorCode);
    showImages(result.ReturnValue);
}
```

In the above code, the callback function calls a showImage method with the returned value as parameter, let's understand this function.

```
function showImages(iterator)
{
    try
    {
        iterator.reset();
        var item;
        var gallery = new Gallery(null,null,"100%",500);
        while (( item = iterator.getNext()) != undefined ){
            gallery.addImage(item.FileNameAndPath);
        }
        listView.addControl(gallery);
        gallery.load();
    }
    catch(e)
    {
        alert(' ' + e);
    }
    return msg;
}
```

In the **showImages** function we iterate over the list and add each file path and name to the gallery object. This object is an instance of the new UI component we have previously created in this post. After add all images in the gallery object the developer have to call the **gallery.load** method to generate and show the gallery.

Notice that you don't need to use the Gallery component with a 5th device. You can retrieve file path and name from any other location (maybe an url location) or you can retrieve the media device information using a different way. But this work is very easy when using the new S60 5th edition WRT API. The below two pictures maybe help you to understand how this works.



Getting information without S60 5th edition WRT API. The information exist, but you have to develop a way to get it. Maybe a XML file, using a internal web server to get resource information or anything else.

