Camera Pic & GeoTagging App using PhoneGap

This article demonstrates the usage of PhoneGap for capturing photos and using gelocation API to geotag the captured photos.

Introduction

Apache Cordova PhoneGap APIs can be used to build native mobile applications using HTML, CSS and JavaScript. Cordova provides number of JavaScript libraries that can be invoked, with device-specific native code for the respective JavaScript libraries. These libraries would be compatible with a large number of popular mobile platforms viz. Windows Phone, iOS, Android, Blackberry, Palm WebOS, Bada, and Symbian.

In this article, we are going to see how to implement 'Camera photo capturing & Geo-tagging' application using PhoneGap on Windows Phone 8.

Getting Started

Getting Started with Phonegap has the detailed step by step guide on how to start setting up your first PhoneGap project.

For the benefit of beginners, we will quickly see the key steps:

- Download the latest copy of Cordova and extract its contents to a folder.
- In Visual Studio 2012 -> New Project -> JavaScript - Windows Store (Template)
- Add capabilities to **WPAppManifest.xml**:

```
<Capabilities>
    <Capability Name="ID_CAP_ISV_CAMERA" />
    <Capability Name="ID_HW_FRONTCAMERA" />
</Capabilities>
```

- Include **cordova.js** (extracted in step 1) in js folder of your project.
- Edit **index.html** and add a reference to **cordova.js**.

PhoneGap Libraries Used

Below are the PhoneGap libraries that we are going to use in this example:

- Camera API provides access to the device's default camera application.
- Geolocation API provides access to the device's GPS sensor.
- Storage API provides access to the devices storage.

Step by Step Explanation

Static Body Logic

We have the following HTML5 elements in the body section:

- Canvas - We are using canvas so that we can use the image and overlay text (geo-coordinates) on the image and later use this canvas to have a merged image (image + text).
- taggedImage - Final merged image holder.
- Capture Camera Pic button - on click of this button, default camera application of mobile will open so that the user can capture the photo. Once the photo is taken, the camera application closes and our application will be resumed.
- Save button - Save the geo-tagged image to local storage or Skydrive.

```
<body>
    <canvas id=”canvasPnl”>
    <img style="display:none;" id=" tempImg " src="" />
    <img style="display:none;width:80%;height:80%;" id="taggedImage" src="" />
    <button onclick=" captureCameraPic ();">Capture Camera Pic</button> <br>
    <button onclick="save();">Save</button> <br>
</body>
```


(C) Copyright Nokia 2014. All rights reserved.

Include the reference to cardova javascript library

```
    <script type="text/javascript" charset="utf-8" src="cordova-2.4.0.js"></script>
```

Cordova device support check

Wait for Cordova to connect with the device and once the connection is made `onDeviceReady` function will be invoked and this will be your starting point.

```
document.addEventListener("deviceready",onDeviceReady,false);
function onDeviceReady()
    {
        destinationType=navigator.camera.DestinationType;
    }
```

Take a snap from device camera

User is given a button on user interface called "Take A Snap". When user clicks the button, following function is invoked and the photo from camera is taken.

`camera.getPicture` opens the default camera application, user can take a picture and then the camera app closes and our application will be resumed. By default the `Camera.sourceType` = `Camera.PictureSourceType.CAMERA`, and we want the same (camera as source) hence not specified.

`DATA_UR`L value is 0 and it returns the image as base64 encoded string.

```
    function captureCameraPic() {
      // Take picture using device camera
      navigator.camera.getPicture(onCameraPicSuccess, onCameraPicError, { quality: 80,
        destinationType: destinationType.DATA_URL });
    }
```

Invoke geolocation process

Once the photo is taken successfully form camera, we will put it as a source to a temp img holder element. And then invoke the geo-current position function to get the location coordinates.

`getCurrentPosition` - returns the current position as a Position object.

```
    function onCameraPicSuccess (imgData)
   {
      tempImg.src = "" + imgData;

      navigator.geolocation.getCurrentPosition(onGeoSuccess,onGeoError)
    }
```

Image Tagging Process

Once the location is fetched correctly, the callback function `onGeoSuccess` will invoked. In this function, we call the `drawCanvas` function which will process the image tagging.

Once the canvas is drawn with image + geo-locations, we will put the final image in `taggedImage` element so that later it can be saved to local storage or uploaded to Skydrive.

(C) Copyright Nokia 2014. All rights reserved.

```
function onGeoSuccess(position)
{
      drawCanvas(position.coords.latitude, position.coords.longitude);

      // save canvas image as data
      var dataURL = canvas.toDataURL();

      // set canvasImg image src to dataURL
      // so it can be saved as an image
      document.getElementById(taggedImage).src = dataURL;

     taggedImage.style.display = ‘block’;
}

function drawCanvas(latitude, longitude)
{
  var canvas = document.getElementById("canvasPnl");
  var context = canvas.getContext('2d');
  var locationtxt = “Latitude: ” + latitude + “, Longitude: ” + longitude;
  context.fillText(locationtxt, 0, 0);
 }
```

Putting all pieces together

```
<!DOCTYPE html>
<html>
  <head>
    <title>Click Pic & Geotag</title>

    <script type="text/javascript" charset="utf-8" src="cordova-2.4.0.js"></script>
    <script type="text/javascript" charset="utf-8">

    var destinationType; // sets the format of returned value

    document.addEventListener("deviceready",onDeviceReady,false);

    function onDeviceReady()
    {
        destinationType=navigator.camera.DestinationType;
    }

    function captureCameraPic() {
      // Take picture using device camera
      navigator.camera.getPicture(onCameraPicSuccess, onCameraPicError, { quality: 80,
        destinationType: destinationType.DATA_URL });
    }

    function onCameraPicSuccess (imgData)
   {
      tempImg.src = "" + imgData;

      navigator.geolocation.getCurrentPosition(onGeoSuccess,onGeoError)
    }
```


(C) Copyright Nokia 2014. All rights reserved.

```
    function onGeoSuccess(position)
   {
      drawCanvas(position.coords.latitude, position.coords.longitude);

      // save canvas image as data
      var dataURL = canvas.toDataURL();

      // set canvasImg image src to dataURL
      // so it can be saved as an image
      document.getElementById(taggedImage).src = dataURL;

     taggedImage.style.display = ‘block’;
   }

   function drawCanvas(latitude, longitude)
   {
      var canvas = document.getElementById("canvasPnl");
      var context = canvas.getContext('2d');
      var locationtxt = “Latitude: ” + latitude + “, Longitude: ” + longitude;
      context.fillText(locationtxt, 0, 0);
   }


   // Called if error occurs in getPicture function.
   function onCameraPicError (errMsg)
   {
        alert('Error capturing picture: ' + errMsg);
   }

   // Called if error occurs in getCurrentPosition function.
  function onGeoError (errMsg)
  {
      alert(‘Error retrieving location information: ' + errMsg);
  }

   function save()
   {
      //Store image to local storage or upload to skydrive.
   }

  </script>
  </head>
  <body>
    <canvas id=”canvasPnl”>
    <img style="display:none;" id=" tempImg " src="" />
    <img style="display:none;width:80%;height:80%;" id="taggedImage" src="" />
    <button onclick=" captureCameraPic ();">Take A Snap</button> <br>
    <button onclick="save();">Save</button> <br>
  </body>
</html>
```

Summary

We saw how easy is to use the PhoneGap library and combining it with HTML5 technology to create multi-platform supported application.

(C) Copyright Nokia 2014. All rights reserved.

References

- Cordova (PhoneGap) and Windows Phone 8
- PhoneGap Docs

(C) Copyright Nokia 2014. All rights reserved.