Cross-Platform Mobile Geolocation Applications Based on PhoneGap

Niyigena Jean Pierre, Fan Xiumei, Gakwaya Daniel, and Gombaniro Jean Claude

Abstract—Building mobile applications have been a big challenge. This issue is mainly caused by different programming languages that one has to master when developing mobile applications. To address the latter mentioned bottleneck, PhoneGap, a cross-platform framework that only requires knowing JavaScript, HTML/HTML5 and CSS/CSS3, among many others, was invented. This paper describes the use of PhoneGap geolocation API to determine the location for most of today’s on market mobile smart phone platforms. The ease of using only a few lines of JavaScript code to access and manage the mobile device native geolocation API, in most mobile smart phone platforms, makes obvious that mobile geolocation with PhoneGap geolocation API is among the best for mobile cross-platform geolocation applications.

Index Terms—Cross-platform mobile geolocation, geolocation application, PhoneGap geolocation API, smart phones geolocation.

I. INTRODUCTION

Nowadays, smart phones are getting much more popularity notably Android and iPhone, consequently the developers are turning their attention on mobile based application researches. However, Android and iPhone applications development are based on different programming languages, the former is based on Java while the latter is based on Objective C [1]. Table I shows the usual requirements for development environments for various mobile platforms.

There are many cross-platform frameworks now, such as Titanium, Rhodes, DragonRad, MoSync and so on. PhoneGap is considered the best one. In 2009, PhoneGap won the People’s Choice award at the Web 2.0 Expo LaunchPad competition. Of course, being a project for geeks, the conference attendees voted for the winner by Short Message Service (SMS) from their mobile phones [2].

Today, developers use PhoneGap to build mobile application for several reasons like:

1) An application was built using web technology, they want to be able to deploy the application to one or more mobile application stores (such as Apple App Store, Android Market, or Blackberry App World) [2].

2) They want to build a mobile applications using usual web development skills, but also need the functions of device side (like the calendar or the camera), which mobile browsers do not support.

3) Or they want to make a rapid prototype of mobile application and do not have time to study Objective-C language or Java language.

We cannot finish this section without mentioning that applications that are developed using PhoneGap are hybrid applications [2]. This means that these applications are not purely HTML/JavaScript based, nor are they native. Parts of the application, mainly the UI, the application logic, and communication with a server, is based on HTML/JavaScript. The other part of the application that communicates and controls the device (phone or tablet) is based on the native language for that platform.

II. THE WORKING THEORY OF PHONEGAP

The working theory of PhoneGap is not complex. The application’s user interface consists of essentially a unique screen that contains a unique web view that consumes all of the available space on the device’s screen [2].

When the application launches, it loads the web application’s startup page (typically index.html but easily changed by the developer to something else) into the web view and then passes control to the web view to allow the

TABLE I. REQUIREMENTS FOR DEVELOPMENT ENVIRONMENTS (FOR VARIOUS MOBILE PLATFORMS)

<table>
<thead>
<tr>
<th>Mobile OS</th>
<th>Operating System</th>
<th>Software/IDEs</th>
<th>Programming Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>Mac only</td>
<td>Xcode</td>
<td>Objective C</td>
</tr>
<tr>
<td>Android</td>
<td>Windows/ Mac/ Linux</td>
<td>Eclipse/ ADT</td>
<td>Java</td>
</tr>
<tr>
<td>BlackBerry</td>
<td>Windows mainly</td>
<td>Eclipse/ JDE, Java</td>
<td>Java</td>
</tr>
<tr>
<td>Symbian</td>
<td>Windows/ Mac/ Linux</td>
<td>Carbide.c++</td>
<td>C++</td>
</tr>
<tr>
<td>WebOS</td>
<td>Windows/ Mac/ Linux</td>
<td>Eclipse/ WebOS plugin</td>
<td>HTML/ JavaScript/ C++</td>
</tr>
<tr>
<td>Windows 7 Phone</td>
<td>Windows mainly</td>
<td>Visual Studio 2010</td>
<td>C#, .NET, Silverlight or WPF</td>
</tr>
</tbody>
</table>
user to interact with the web application. As the user interacts with the application’s content (the web application), links or JavaScript code within the application can load other content from within the resource files packaged with this application or can reach out to the network and pull content down from a web or application server [2].

A. PhoneGap Architecture

The PhoneGap’s architecture is composed mainly of 3 layers: Web Application, PhoneGap, and OS and native API’s.

![PhoneGap Architecture Diagram]

Fig. 1. PhoneGap architecture.

In Fig. 1 the high-level layer represents the source code of the application. JavaScript and the native API make the composition of the intermediate layer. Principally, this layer is responsible for the interfacing linking the layer of PhoneGap and the web application. In addition, it also makes sure of doing interfacing between the JavaScript API’s used by the application and the native API’s used by the mobile. This layer functionality is to make sure that the relationship between JavaScript API’s and the native API’s of each mobile operating system is maintained [3]. JavaScript API’s are provided to developers by PhoneGap to allow the access to advanced device features, such as the Menus, GPS, File, Contacts, Connection, Compass, Camera, Calendar, Bluetooth, Barcode, Accelerometer, NFC, etc.

![Complete Diagram of PhoneGap Architecture]

Fig. 2. Complete diagram of PhoneGap architecture and interfacing among components [1].

In Fig. 2 is presented a more detailed architecture diagram brought forth by IBM. It shows all the components of a web application, the rendering engine of HTML, PhoneGap API’s and operating system layer. Furthermore, a number of different interfaces are presented in detail, such as the interfacing linking native API’s layers and PhoneGap API’s.

III. PHONEGAP GEOLOCATION API

The PhoneGap geolocation API gives location information for the host device, such as longitude and latitude. Frequent origins of location information cover Global Positioning System (GPS) and location deduced from network signals such as IP address, RFID, WiFi and Bluetooth MAC addresses, and GSM/CDMA cell IDs [3]. There is no promise that the API provides the device’s effective location.

In Fig. 3 is shown the output of different PhoneGap geolocation API properties in an iPhone simulator version 6.1.

document.addEventListener("deviceready", onDeviceReady, false);
function onDeviceReady() {
    navigator.geolocation.getCurrentPosition(onSuccess,
onError, {maximumAge: 30000, timeout: 5000, enableHighAccuracy: false});
}
function onSuccess (position) {
    var element = document.getElementById("geolocation");
element.innerHTML =
"Latitude: \"+ position.coords.latitude + \"<br />\nLongitude: \"+ position.coords.longitude + \"<br />\nAltitude: \"+ position.coords.altitude + \"<br />\nAccuracy: \"+ position.coords.accuracy + \"<br />\nAltitude Accuracy: \"+ position.coords.altitudeAccuracy + \"<br />\nHeading: \"+ position.coords.heading + \"<br />\nSpeed: \"+ position.coords.speed + \"<br />\nTimestamp: \"+ position.timestamp + \"<br />\n";
}

The PhoneGap geolocation API derives from the W3C Geolocation API Specification. Part of mobiles (Windows Phone 7, BlackBerry, Android, Bada, webOS and Tizen, to be specific) already have an implementation of this specification. For those mobiles, the built-in support is used instead of using in place of it the PhoneGap’s implementation. For mobiles that don’t have geolocation support, the PhoneGap implementation coheres to the W3C specification.

![PhoneGap Geolocation API Properties]

Fig. 3. PhoneGap geolocation API properties in an iPhone simulator version 6.1.
A. **PhoneGap Geolocation API Methods**

The PhoneGap geolocation API has three methods successively `geolocation.getCurrentPosition` method, `geolocation.watchPosition` method and `geolocation.clearWatch` method [4].

1) **Geolocation.getCurrentPosition**

`Geolocation.getCurrentPosition` is an asynchronous function which returns the current position of the device to the callback of `geolocationSuccess` with a parameter object of `Position`. The callback of `geolocationSuccess` is called with an object of `PositionError` in case there happens to be an error.

2) **Geolocation.watchPosition**

The asynchronous function `geolocation.watchPosition` returns the current position of the device whenever a change in position is detected. When the device has recovered the new location, the callback function `geolocationSuccess` is called with the parameter of `Position` object [5]. In case there is an error, there is a call of `geolocationError` with an object of `PositionError`.

3) **Geolocation.clearWatch**

The function of `geolocation.clearWatch` brings to an end the watching changes to the location of the device by getting rid of the `geolocation.watchPosition` referred to by `watchID`.

B. **Arguments**

Each of mentioned methods in the previous section takes one, two or three arguments among `geolocationSuccess`, `geolocationError` and `geolocationOptions` [4].

1) **GeolocationSuccess**

This is the callback of the user that is invoked in case there is availability of a geolocation position (when using the function of `geolocation.getCurrentPosition`), or when there is a change of position (when using the function of `geolocation.watchPosition`).

2) **GeolocationError**

This is the callback of the user that is invoked in case there is an error for the functions of `geolocation`.

3) **GeolocationOptions**

Optional parameters to customize the retrieval of the `geolocationPosition`. [maximumAge: 3000, timeout: 5000, enableHighAccuracy: true ];

The `enableHighAccuracy` option tells that the implementation needs to receive the best results. By default, the device will try to use the approach based on the network to retrieve a location. Setting this property to true indicates the scope to use a more accurate method such as satellite positioning. The `timeout` option is the time that is allowed to pass from the call to `geolocation.getCurrentPosition` or `geolocation.watchPosition` until the corresponding callback of `geolocationSuccess` is called. If the callback of `geolocationSuccess` is not called during this time interval, the callback of `geolocationError` will be called with an error code of `PositionError.TIMEOUT` [4]. When used conjointly with `geolocation.watchPosition`, the callback of `geolocationError` can be invoked on an interval every timeout milliseconds. The `maximumAge` option accepts the cached location with age does not exceed the specified time in milliseconds.

C. **Returned Objects (Read-Only)**

1) **Position interface**

The `Position` interface is the container for the geolocation information returned by the PhoneGap geolocation API. Today’s API version allows one attribute of type `Coordinates` and a timestamp. Future versions of the API may allow additional attributes that provide other information about this position (e.g. street addresses).

```
[NoInterfaceObject]
interface Position {
    readonly attribute Coordinates coords;
    readonly attribute DOMTimeStamp timestamp;
};
```

The `coords` attribute contains a set of geographic coordinates together with their associated accuracy, as well as a set of other optional attributes such as altitude and speed. The `timestamp` attribute represents the time when the Position object was acquired and is represented as a `DOMTimeStamp` [DOMTIMESTAMP].

2) **PositionError interface**

```
[NoInterfaceObject]
interface PositionError {
    const unsigned short PERMISSION_DENIED = 1;
    const unsigned short POSITION_UNAVAILABLE = 2;
    const unsigned short TIMEOUT = 3;
    readonly attribute unsigned short code;
    readonly attribute DOMString message;
};
```

The code attribute must return the `PERMISSION_DENIED (numeric value 1)` code for the location acquisition process failed because the document does not have permission to use the Geolocation API, must return the `POSITION_UNAVAILABLE (numeric value 2)` for the position of the device could not be determined. For instance, one or more of the location providers used in the location acquisition process reported an internal error that caused the process to fail entirely and must return `TIMEOUT (numeric value 3)` for the length of time specified by the timeout property has elapsed before the implementation could successfully acquire a new Position object [5]. The `message` attribute must return an error message describing the details of the error encountered. This attribute is primarily intended for debugging and developers should not use it directly in their application user interface.

3) **Coordinates interface**

```
[NoInterfaceObject]
interface Coordinates {
    readonly attribute double latitude;
    readonly attribute double longitude;
    readonly attribute double? altitude;
    readonly attribute double? altitudeAccuracy;
    readonly attribute double? heading;
    readonly attribute double? speed;
};
```

The geographic coordinate reference system used by the attributes in this interface is the World Geodetic System (2d).
[WGS84]. No other reference system is supported. The latitude and longitude attributes are geographic coordinates specified in decimal degrees. The altitude attribute denotes the height of the position, specified in meters above the WGS84 ellipsoid. If the implementation cannot provide altitude information, the value of this attribute must be null. The accuracy attribute denotes the accuracy level of the latitude and longitude coordinates [6]. It is specified in meters and must be supported by all implementations. The value of the accuracy attribute must be a non-negative real number. The altitudeAccuracy attribute is specified in meters. If the implementation cannot provide altitude information, the value of this attribute must be null. Otherwise, the value of the altitudeAccuracy attribute must be a non-negative real number. The accuracy and altitudeAccuracy values returned by an implementation should correspond to a 95% confidence level. The heading attribute denotes the direction of travel of the hosting device and is specified in degrees, where 0° ≤ heading < 360°, counting clockwise relative to the true north. If the implementation cannot provide heading information, the value of this attribute must be null. If the hosting device is stationary (i.e. the value of the speed attribute is 0), then the value of the heading attribute must be NaN [6]. The speed attribute denotes the magnitude of the horizontal component of the hosting device’s current velocity and is specified in meters per second. If the implementation cannot provide speed information, the value of this attribute must be null. Otherwise, the value of the speed attribute must be a non-negative real number.

D. Properties of Objects and Constants

1) Properties of objects

1) Coordinates properties: latitude in decimal degrees (Number), longitude in decimal degrees (Number), altitude of the position in meters above the ellipsoid (Number), accuracy level of the latitude and longitude coordinates in meters (Number), altitudeAccuracy level of the altitude coordinate in meters (Number), heading of travel, specified in degrees counting clockwise relative to the true north (Number), speed of the device, specified in meters per second (Number).

2) Position properties: It contains Position coordinates and timestamp, created by the PhoneGap geolocation API. coords is a set of geographic coordinates (Coordinates) and timestamp which is a creation timestamp for coords (Date).

3) PositionError Properties: code is one of the predefined error codes listed below and message is an error message describing the details of the error encountered.

2) PositionError constants

The PositionError object is returned to the user through the geolocationError callback function when an error occurs with PhoneGap geolocation API. PositionError.PERMISSION_DENIED is returned when the user does not allow your application to retrieve position information. This is dependent on the platform. PositionError.POSITION_UNAVAILABLE is returned when the device was unable to retrieve a position. In general this means the device has no network connectivity and/or cannot get a satellite fix, and PositionError.TIMEOUT is returned when the device was unable to retrieve a position within the time specified in the geolocationOptions’ timeout property. When using in conjunction with geolocation.watchPosition, this error could be called into the geolocationError callback every timeout milliseconds.

IV. PERMISSIONS

To use this API, different mobile devices need different permissions to be set up. Table II presents the location of the directory of the configuration file for different devices operating systems.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Permissions</th>
<th>Directory/file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>Permissions are required</td>
<td>appres/xml/config.xml</td>
</tr>
<tr>
<td>Bada</td>
<td>No permissions are required</td>
<td>appres/xml/config.xml</td>
</tr>
<tr>
<td>BlackBerry</td>
<td>Permissions are required</td>
<td></td>
</tr>
<tr>
<td>WebWorks</td>
<td>Permissions are required</td>
<td></td>
</tr>
<tr>
<td>iOS</td>
<td>Permissions are required</td>
<td>www/plugins.xml</td>
</tr>
<tr>
<td>webOS</td>
<td>No permissions are required</td>
<td>config.xml</td>
</tr>
<tr>
<td>Windows Phone</td>
<td>Permissions are required</td>
<td>Properties/WPAppMan config.xml</td>
</tr>
<tr>
<td>Tizen</td>
<td>No permissions are required</td>
<td>ifest.xml</td>
</tr>
</tbody>
</table>

V. PRIVACY CONSIDERATIONS AND SECURITY

Privacy is obvious a concern when you are sharing your physical location with a remote web server [7]. PhoneGap geolocation API is used to retrieve the location of a hosting mobile. In almost all cases, this information also reveals the location of the user mobile, thus potentially harm the privacy of the user. To ensure compliance, the PhoneGap geolocation API specification provided a mechanism to protect the privacy of the user and this mechanism ensures that no location information is revealed by this API without the express permission of the user.

A. Privacy Considerations for Implementers of PhoneGap Geolocation API

Without the express permission of the user, the device has not to send location information to the application [7]. The application must obtain permission from the user interface, unless it has been prearranged trust relationship with the user, as described below. The user interface must contain the host components of the file URI [URI]. This is done through the user interface and access to these permissions, which must be revocable, will be preserved beyond the current browsing session and the application must respect revoked permissions.

Some applications may have no such user interface and there will be trust relationships that have been organized in advance. For example, while some applications shows a user interface when performs requests of the device geolocation, a PhoneGap VOIP application can not show a user interface when requesting location information to make an emergency-calling system.

B. Privacy Considerations for Recipients of Location Information
With PhoneGap geolocation API, the location information beneficiary only requests information when necessary. He can only use location information for the tasks it is provided, must discard location information once that task is finished, unless explicitly the user allows to it. He must also take steps to protect this information from being used illegally. The application should allow the user to update and delete location information if this information is stored.

Location information cannot be retransmitted by location information beneficiary without the express permission of the user. Precautions must be considered and the use of encryption is encouraged when forwarding.

The location information beneficiary must clearly and conspicuously disclose the fact that location data is being collected, collection purpose, data retaining duration, how the data is obtained and how it is shared, how users access it, update it, and delete it, and the user’s many other choice with regard to data. There must be explanation of any exceptions to the guidelines listed above.

C. Other Considerations of the Implementation

According with the requirements discussed in the previous section, the performors of the PhoneGap geolocation API are also required to consider the following aspects that may have a negative impact on the privacy of the user: In some cases, the mobile device can inadvertently grant authorization to the application to disclose their location. In other cases, the content hosted to a particular location information beneficiary only requests information when necessary. He is now a master’s student in the Department of Computer Science and Technology at Beijing institute of technology.

VI. CONCLUSION

This paper has described mobile cross-platform geolocation based on PhoneGap geolocation API. PhoneGap is a mobile cross-platform development framework invented to solve the problem of different programming languages that developers challenge/encounter when building applications for different mobile platform. PhoneGap geolocation API is supported for multiple today’s famous mobile operating systems such as iOS, Android, Black Berry, Symbian, WebOS and Windows 7 Phone. PhoneGap geolocation API uses JavaScript, HTML and CSS to determine the location of mobile devices by returning different location properties notably Longitude, Altitude, Accuracy, Altitude Accuracy, Heading, Speed and Timestamp when queried. There is no promise that the PhoneGap geolocation API provides the mobile device’s effective location.

REFERENCES


Niyigana Jean Pierre was born in Rwanda on 7th September 1984. He holds a bachelor’s degree in computer science and technology earned in 2012 from Beijing institute of technology, China. Currently, he is a master’s student in the Department of Computer Networks and Security, Faculty of Computer Science and Technology at Beijing institute of technology.

Fan Xiuxiu is a Ph.D. She was born in China in 1967. She is now an associate professor at Beijing Institute of Technology (BIT). She is also the 2007 new century China ministry of education excellent talents inductee. In BIT, her research interests mainly includes computer networks quality of service control, wireless ad-hoc networks routing technology, delay tolerant networks, and network performance evaluation.

Gombaniro Jean Claude was born in Rwanda on 17th December 1984. He holds a bachelor’s degree in computer science and technology earned in 2012 from Beijing institute of technology, China. He is now a master’s student in the Department of Information Management, Data Mining and Science Reference Management, Faculty of Computer Science and Technology at Beijing institute of technology.

Gakwaya Daniel was born in Rwanda on 1st August 1984. He holds a bachelor’s degree in electrical engineering earned in 2012 from Beijing Institute of Technology, China. He is now a master’s student in the School of Computer Science and Technology, Department of Computer Technology at Beijing Institute of Technology. His research mainly focuses on data networks, networks on chip.