Mobile Enterprise Application Development - a Cross-Platform Framework

FLORIAN WOLF, KARSTEN HUFFSTADT
Applied Research Center for Mobile Solutions
University of Applied Sciences Wuerzburg-Schweinfurt

In today's business mobility of information systems is an important topic. Almost no company can get by these days without mobile software solutions. At the same time we are going through a convergence of desktop and mobile, web and native application development. Because of this clear necessarity of mobility and mobile development it is not surprising that there is a large number of different recommendations for software developer using different technologies. This paper is going to point out the basic approach of a cross-platform application framework using the example of PhoneGap and - more importantly - highlights and answers the question if the often promised universal applicability of these frameworks really apply and a one-off source code is enough to deploy an enterprise application.

Categories and Subject Descriptors: Mobile and Ubiquitous Computing

Additional Key Words and Phrases: Mobile Business, Mobile Applications, Apps, Web Applications, Cross-Platform Application Framework, PhoneGap

1 INTRODUCTION

It was already acknowledged in the middle of the last decade that mobile ICT-based capabilities have the potential to increase productivity, efficiency and other important business performance metrics [Gebauer and Shaw 2004; Basole 2005; Nah et al. 2005; Atkins et al. 2006]. Today mobility of information systems is an important business topic. Almost no company can get by these days without mobile software solutions supporting their business.

At the same time we are going through a convergence of desktop and mobile, web and native application development [Mikkonen et al. 2009]. More and more applications run in web browsers and do not necessarily require any explicit installation or native source code. Web applications therefore enable companies to implement software written as a one-off source code in form of the three components HTML5, JavaScript and CSS rendered in a standard web browser. Using that approach, developers do no longer need to support various operating systems writing their applications in native code for iOS, Android, Windows or others. Furthermore web applications are running without the need to be downloaded and installed separately. In addition to that, they can be deployed instantly worldwide. All these factors support companies on their way to easily develop mobile enterprise applications.

What remains is the question of how a complex application for enterprises using only JavaScript, HTML5 and CSS can be supported. With the need of large database handling or access to the API-functionality of the underlying operating systems such as geolocation or accelerometer web techniques reach their limits. Because of that, the so called cross-platform application frameworks were developed which contain native code pieces to interact with the operating systems and run the web application in a webview [Charland and Leroux 2011].

Author's address: Florian Wolf, Karsten Huffstadt, University of Applied Sciences Wuerzburg-Schweinfurt, Germany, http://mobilab.fhws.de.

This paper points out the basic approach of a cross-platform application framework using the example of PhoneGap and - more importantly - highlights and answers the question if the often promised universal applicability of these frameworks really apply and a one-off source code is enough to deploy an enterprise application.

2 CROSS-PLATFORM PARADIGM

Even some years ago Mikkonen et al. [2009] recognized that web applications are the future of mobile application development. Cross-platform techniques like HTML5, JavaScript, CSS and Ajax are the preferred development environment. Willing to reach the goal of making these techniques as powerful and efficient as native software, several development tools and cross-platform frameworks are in development right now and were already developed in the past. We identified a clear distinction of frameworks in two categories.

Browser-centric cross-platform frameworks or so-called user interface frameworks use HTML5-techniques to deploy mobile applications primarily on web browsers. These UI frameworks such as jQuery Mobile or Sencha Touch 2 are HTML5-based user interface systems including a various set of built-in components to develop rich internet applications. With a library of components, widgets, layouts and stylesheets for a high GUI capability already included web applications can be rapidly developed [Yan and Chen 2011; Smutny 2012].

Native-centric cross-platform frameworks use HTML5-techniques - or in some cases a proprietary SDK - to develop mobile applications running on native platforms. Once an application is build up with the framework, it can be deployed on several mobile operating systems without additional expenditure for development. These frameworks such as PhoneGap or Appcelerator can be used in most cases as a hybrid variant of cross-platform frameworks. In this case they are combining a native container with direct access to necessary API-functions supporting a web-based frontend for building web-based user interfaces. These Hybrid-centric cross-platform frameworks provide natively coded containers with for example a simple JavaScript library acting as a bridge between the front end web application and the native API-functionality such as using the camera, the accelerometer or the GPS unit.

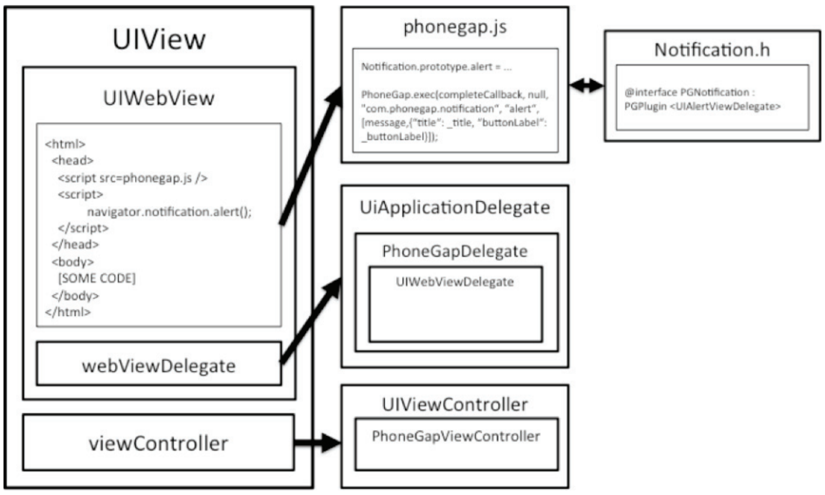
All these frameworks share the same idea: to develop a cross-platform mobile application with in most cases web techniques running on different operating systems. Corral et al. [2011] identified in this context an entirely justified evolution in development paradigms in this context. Using these frameworks to create mobile applications for web platforms leads to a web-based multiplatform development paradigm. Due to this significance we analyzed as following described the portability of cross-platform developed web applications using PhoneGap as an example.

3 THE PHONEGAP-THEORY IN DETAIL

Using web techniques is working in a sandbox, which is a jail from lower level APIs [Charland and Leroux 2011]. To access the device storages, sensors and datas this gap has to be bridged. Currently, two ways are available for developers. On the one hand most of today’s mobile browsers provide basic API functionality such as geolocation or local storage. Given that the W3C has a Device API Working Group, it can be expected that more APIs reach the browsers in the near future. On the other hand, if developers need API functions, that are not supported directly from the browser, the earlier described hybrid-centric cross-platform frameworks can be used.

The following section exemplarily explains in detail how PhoneGap is working as a hybrid-centric cross-platform framework. For the subsequent analysis of the portability and cross-platform ability we choose iOS as the lead programming platform. Therefore, the first step is to explain the context between iOS and PhoneGap. The basic indicator in iOS is the UIView class which defines a rectangular area on the device screen. This is usually an applications user interface or content that the user is intended to see by the application creator using the device native programming language Objective-C. Some applications not only use native content, but also web content to present information to the user. Therefore the programmer has to create a UIWebView object that has the ability to extend the UIView class to visualize this web content. This is the main principle of how PhoneGap works. Within a native UIWebView there is a UIWebView object, which is able to visualize the HTML pages. The logic of those pages is powered by custom JavaScript code that ensures the functionality of the PhoneGap application. Those JavaScript files are able to make use of the functions provided by the PhoneGap plugin and are therefore able to access all supported APIs.

To get a more substantiated knowledge of how the communication between web based code and native APIs in iOS works, an understanding the basic principles of delegation in iOS is needed. A delegate is based on events and handles the interaction of Objective-C objects and allows them to interact with each other without previously defining dependencies. Using a UIWebView object, the corresponding delegate is called webViewDelegate and inherits from the iOS delegate UIApplicationDelegate. That means whenever web based code is executed within a UIWebView object, the UIWebViewDelegate defines methods that a delegate of this UIWebView object can optionally implement to intervene when web content is loaded. This is where PhoneGap places its own PhoneGapDelegate in between UIApplicationDelegate and UIWebViewDelegate. By doing that, PhoneGap is able to catch specific actions made by a UIWebView object and react as intended in the PhoneGap source code.



```
<html>
  <head>
    <script src=phonegap.js />
    <script>
          navigator.notification.alert();
    </script>
  </head>
  <body>
    [SOME CODE]
  </body>
</html>
```

```
Notification.prototype.alert = ...

PhoneGap.exec(completeCallback, null,
"com.phonegap.notification", "alert",
[message,{"title": _title, "buttonLabel":
_buttonLabel}]);
```

```
@interface PGNotification :
PGPlugin <UIAlertViewDelegate>
```

Fig. 1. How PhoneGap works

Similar to that PhoneGap also establishes a PhoneGapViewController as an extension to the iOS UIViewController. This controller provides the fundamental view-management model in iOS.

Using the PhoneGapViewController as a subclass of UIViewController has the advantage of directly controlling the way a PhoneGap based application reacts on certain circumstances other than the native iOS way. Figure 1 provides a quick overview on how it all works together.

4 FRAMEWORK ANALYSIS - A BUSINESS CASE

The main objective was to analyze if the promises made by PhoneGap in matters of multi-system compatibility prove right. It is one of the frameworks big features that a programmer has to create the source code using web technologies only once and then reuse it on any supported platform without the need of further modification.

Therefore a PhoneGap-based application using iOS as lead platform was created, which means that the application was coded in Xcode and the working source was transferred to both the Android and Windows Mobile native programming environments to compile them. In theory, what should be achieved with this approach are three identically working and similar looking applications (see figure 2).

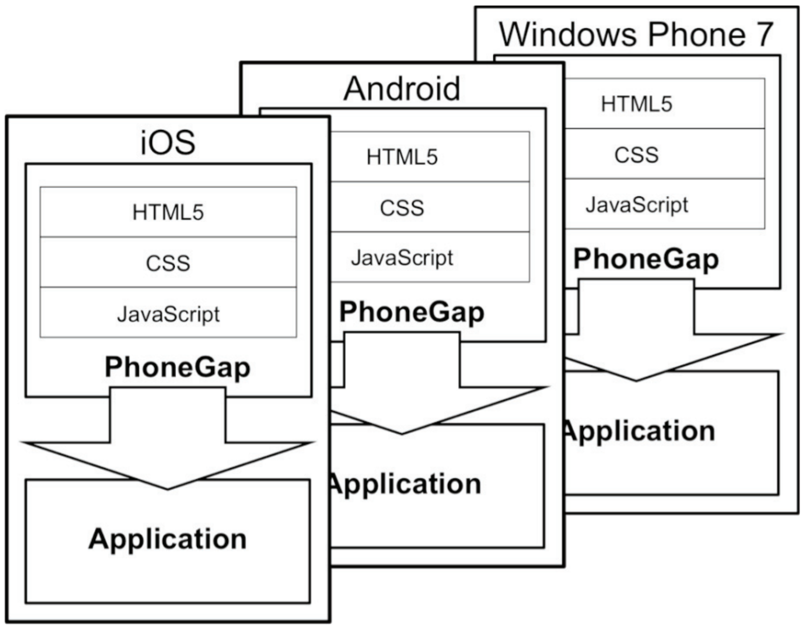


Fig. 2. Building PhoneGap Applications: lead platform iOS

The business case was represented by a logistic function of the maintenance process, especially handling releases and closures of maintenance orders. In terms of functionality, the application uses web services to communicate with an SAP ERP system. The application has to read and write from the ERPs databases to achieve this objective. All error handling and the applications logic is done in JavaScript. Because of the fact that an universal applicability should be considered the business case itself is not in the focus of the analysis and therefore not further described.

In terms of user interface the JavaScript based jQuery Mobile framework was used. Because there are no restrictions within PhoneGap on what kind of JavaScript works (everything should work), this was a possible and valid way to proceed.

JQuery Mobile (jQM) makes use of several HTML5 features to achieve a native like user experience based on general iOS design principles by simulating a native MenuBar and NavigationBar if needed. Another interesting aspect of using jQM is to test the overall PhoneGap experience is internal page linking. There are two ways of creating a HTML5 based application in jQM, single page templates and multi page templates. The classic way of designing a website, which also applies for most web based apps nowadays, is a multi page template approach. That means that the application consists of a certain number of HTML-pages such as index.html, about.html and so one with (if needed) each linking at each other. The second way of designing the application is by using a single page template. That means that only a single HTML-page with several div data-role=page elements in it exists which are able to simulate an arbitrary amount of different views.

The main reason not to implement the application in a single template way is that the more complex the project gets, the more complicated and unclear in terms of internal structure the single HTML-file becomes. By doing that it was also abstained from using the basic jQM way of simulation page transitions such as the native looking swipe effect on view change and it was created a workaround based on mobile.changePage() which gave back the possibility to get the same effects while using multiple HTML-pages.

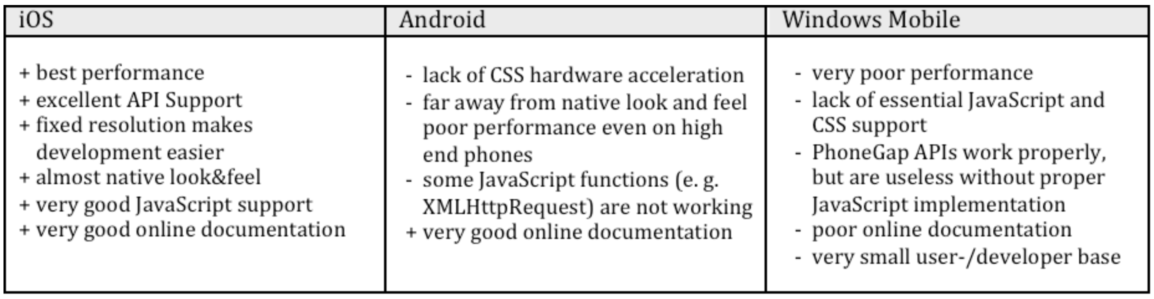
As mentioned before the whole application was created within the iOS native development environment Xcode. In general this works well, apart from limited JavaScript debugging possibilities. The best practice in checking JavaScript code in websites and web-based applications is by debugging it directly in a desktop browser. If the lead platform is iOS, than testing in Safari using the standard developer tools works just fine. Other than that almost every major browser nowadays provides developer tools and plugins such as Firebug for Firefox give the developers even more possibilities for debugging and analyzing JavaScript code.

The basic process of creating a working and well functioning iOS PhoneGap application turned out to be pretty straightforward and easy to achieve for any experienced web programmer. Phone-Gaps containers for iOS work very well with the Safari Mobile browser engine leading to a good performing though not quite native user experience.

After creating the PhoneGap application using XMLHttpRequest to handle the HTTP connection and exchange of XML data between server and client the tested and working code was transferred from Xcode to Eclipse with Android SDK. Deploying the application on an Android Samsung Galaxy S2 worked flawlessly and the application showed on the home screen of the phone as expected. Due to the lack of hardware CSS acceleration of Android the application did not perform as well as in iOS at all. Page transitions were stuttering and the overall impression in terms of performance and visual quality was not nearly as decent as in iOS. The main problem was that it was not possible to get XMLHttpRequest to work. The only solution to get the application work in Android after was by replacing all XMLHttpRequest related code with the jQuery function jQuery.ajax() and grant the application the right to communicate with our SAP web services via Manifest.xml in the Android project folder to avoid cross origin conflicts.

Managing to get the PhoneGap application to work both in iOS and Android the process was repeated by transferring the code into the native Windows Mobile development environment. The process was as simple as before in Android and the application ran on the Samsung Omnia 7 test device shortly. Problems began as soon as the application was started. There was a noticeable delay before the initial login screen appeared and none of the implemented functionality such as user login or transition to another page worked. The previously described workaround to apply page transitions to multiple page templates using jQuery Mobiles mobile.changePage() was not recognized by Windows Phones IE Mobile at all, which seems to only accept page transitions via Windows.location.href(). Apart from that all of the efforts to successfully establish either a jQuery.ajax() or XMLHttpRequest connection failed. Due to the lack of documentation and the numerous possibilities to cause this problem there was no way to find the cause and fix it leading to the conclusion that, in this case, the PhoneGap approach in combination with Windows Mobile failed.

The following figure shows yet again the summarized findings comparing iOS, Android and Windows Mobile as the target platform for the deployed application. This overview does not aim to be complete and illustrates, by way of information, the most important aspects of the analysis.



iOS	Android	Windows Mobile
+ best performance + excellent API Support + fixed resolution makes development easier + almost native look&feel + very good JavaScript support + very good online documentation	- lack of CSS hardware acceleration - far away from native look and feel poor performance even on high end phones - some JavaScript functions (e. g. XMLHttpRequest) are not working + very good online documentation	- very poor performance - lack of essential JavaScript and CSS support - PhoneGap APIs work properly, but are useless without proper JavaScript implementation - poor online documentation - very small user-/developer base

Fig. 3. Findings of the analysis

5 CONCLUSION

The intention of our research was to prove the often promised universal applicability of native-centric cross-platform frameworks using HTML5-techniques to develop mobile applications running on various native platforms. In this precise case it was analyzed if the one-off source code created by using the example of PhoneGaps is enough to deploy an application for iOS, Android or Windows Mobile. In this context it becomes clear that we have to differentiate the target operating systems.

In general it can be summarized that the realization of the business case could be implemented without problems on the iOS as the leading platform. The excellent API support allowed a rapid development and implementation.

The transition to Android was problematic and the result was an application with poor performance and a user interface far away from the native look and feel.

A proper implementation on Windows Mobile was practically impossible to achieve. It becomes clear that PhoneGap does not support all JavaScript functions really for cross platforms and in case of Windows there is a high potential of failure. The poor documentation increases the problem and does not allow a simple, uncomplicated and clearly structured implementation.

Based on these findings, the use of a native-centric cross-platform framework can only be recommended partially. The question is whether it is possible to create an application mostly with HTML5-techniques. Even Charland and Leroux [2011] assume that web browser are able to support more and more API functions. In the meantime - when some specific APIs are necessary - a small native container should be developed using the native development environment.

REFERENCES

ATKINS, A. S., PENGIRAN-HAJIALI, A., AND SHAH, H. 2006. Extending e-business applications using mobile technology. Proceedings of the 3rd international conference on Mobile technology, applications systems.

BASOLE, R. C. 2005. Mobilizing the enterprise: a conceptual model of transformational value and enterprise readiness. 26th ASEM National Conference Proceedings, 364-371.

CHARLAND, A. AND LEROUX, B. 2011. Mobile application development: Web vs. native. Communications of the ACM 54, 5, 49-53.

CORRAL, L., SILLITTI, A., SUCCI, G., GARIBBO, A., AND RAMELLA, P. 2011. Evolution of mobile software development from platform-specific to web-based multiplatform paradigm. Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reections on programming and software.

GEBAUER, J. AND SHAW, M. J. 2004. Success factors and impacts of mobile business applications: result from a mobile e-procurement study. International Journal of Electronic Commerce 8, 3, 19-41.

MIKKONEN, T., TAIVALSAARI, A., AND TERHO, M. 2009. Lively for qt: A platform for mobile web applications. Proceedings of the 6th International Conference on Mobile Technology, Application Systems.

NAH, F., SIAU, K., AND SHENG, H. 2005. The value of mobile applications: a utility company study. Communications of the ACM 48, 2, 85-90.

SMUTNY, P. 2012. Mobile development tools and cross-platform solutions. Carpathian Control Conference (ICCC), 2012 13th International, 653-656.

YAN, B. AND CHEN, G. 2011. Appjoy: Personalized mobile application discovery. Proceedings of the 9th international conference on Mobile systems, applications, and services.