On Hyper-local Web Pages

D. Namiot[1(✉)] and M. Sneps-Sneppe[2]

[1] Lomonosov Moscow State University, Moscow, Russia
dnamiot@gmail.com
[2] Ventspils University College, Ventspils, Latvia
manfreds.sneps@gmail.com

Abstract. In this paper, we discuss one approach for development and deployment of web sites (web pages) devoted to the description of objects (events) with a precisely delineated geographic scope. This article describes the usage of context-aware programming models for web development. In our paper, we propose mechanisms to create mobile web applications which content links to some predefined geographic area. The accuracy of such a binding allows us to distinguish individual areas within the same indoor space. Target areas for such development are applications for Smart Cities and retail.

Keywords: Browsers · Computer networks · Context awareness · HTML5 · Indoor communication

1 Introduction

Our paper deals with mobile web presentations of location-based services. How can we present some local (attached to a certain geographical location) information to mobile users? We are talking about programming (creating) mobile web sites, which content pages correspond to the current location of the mobile user. The traditional scheme is very straightforward. We have to determine the user's location and then create a dynamic web page, the issuance of which is clearly defined by specific geographical coordinates. For example, geo-location is a part of HTML5 standard [1].

As soon as web application obtained (as per user permission, of course) geo-coordinates, it can build a dynamic web page, which content depends on the current location (content is associated with obtained location). Technically, we can render our dynamic page on the client side (right in the browser), when application requests data from server via some asynchronous calls (AJAX) [2], or right on our server (in some CGI-script). In both cases obtained location info is used as a parameter either to AJAX script or to CGI script. For some of the applications (classes of applications), we may use several location-related datasets (e.g., so-called geo-fence [3]), but the common principles are similar. It is so called Location Based Services (LBS) [4].

V. Vishnevsky and D. Kozyrev (Eds.): DCCN 2015, CCIS 601, pp. 11–18, 2016.
DOI: 10.1007/978-3-319-30843-2_2

There are different methods for obtaining location information for mobile users [5]. Not all of them use GPS (GLONAS) positioning actually. Alternative approaches use Wi-Fi, Cell ID, collaborative location, etc. [6]. The above-mentioned geo-location in HTML5 has been a wrapper (interface) for location service. For the most of LBS, their top-level architecture is standard. LBS use obtained location info as a key for any database (data store) with location-dependent data. Location info is actually no more than a key for linking physical space (location) and virtual (e.g., coupon for the store). Only a small number of services actually use the coordinates. The typical example is indoor location based services. The paradigm *Location first* requires a digital map for an indoor space. This map should be created prior to the deployment, and it should be supported in an actual state during service's life time. On the other hand, there is a direction, called context-aware computing. In context-aware computing (ubiquitous computing) services can use other information (not related to geographic coordinates) as the "characteristics" of a user's location. Simplistically, the context is any additional information on the geographical location [7,8]. In this case, additional information (context), with the presence of certain metrics can serve as a unique (up to a certain approximation, of course) feature of a user's location. Or, in other words, we can substitute geo-location with context identification. Why might it be necessary? The typical example is indoor LBS [9]. Traditional geo-positioning can be difficult and positioning accuracy may be insufficient to distinguish the position of the mobile subscriber within the same premises. And yet, it is the distinction between positions within the same space (buildings) may be important for all kinds of services (for example, the buyer is located on the first or second floor of the hall).

Actually, it is a starting point for new approaches in LBS architecture, when the stage with obtaining (detecting) location info could be completely eliminated. Indeed, if location info is no more than a key for some database, then why do not replace geo-keys (e.g., latitude and longitude) with context-related IDs? It is sufficient to identify context and use this identification to search data.

The rest of the paper is organized as follows. In Sect. 2, we describe context identification. In Sect. 3, we describe how this identification could be used in web programming. In Sect. 4, we discuss the generic approaches for incorporating sensing information into web pages.

2 Network Proximity

One of the widely used methods for the identification of context is the use of wireless network interfaces of mobile devices (Wi-Fi, Bluetooth). The reasons for this are straightforward. On the first hand, these interfaces are supported in all modern smart-phones. Secondly, for obvious reasons, monitoring of network interfaces is directly supported and executed by the mobile operating systems. Therefore, a survey of network interfaces on the application level can be simplified and not cause additional power consumption, as compared with, for example, a specially organized monitoring for the accelerometer. Information received through the

network interface is used to estimate the proximity of the mobile user to the elements of the network infrastructure (network proximity [10]). Note, that other mobile devices can act as these elements too (e.g., Wi-Fi access point, opened right onto mobile phone [11]). The classical form for collecting data about Wi-Fi devices are so-called Wi-Fi fingerprints sets [12]. Wi-Fi fingerprints are digital objects that describe availability (visibility) for network nodes. Their primary usage is navigation related tasks. The alternative approach lets users directly associate some data chunks with existing (or artificially created) network nodes. In other words, it is a set of user generated links between network nodes and some content that could be used by those in proximity to networks nodes. This approach is presented in SpotEx project and associated tools [13,14]. SpotEx lets users create a set of rules (logical productions) for linking network elements and available content. A special mobile application (context-aware browser) is based on the external set of rules (productions, if-then operators). The conditional part of the each rule includes predicates with the following objects:

- identity for Wi-Fi network (name, MAC-address)
- RSSI (signal strength),
- time of the day (optionally),

In other words, it is a set of operators like this:

IF AccessPointIsVisible ('Cafe') THEN { show content for Cafe }

Block { *show content for Cafe* } is some data (information) snippet presented in the rule. Each snippet has got a title (text) and some HTML content (it could be simply a link to any external site for example). Snippets could present coupons/discount info for malls, news data for campuses, etc. The context-aware browser (mobile application) maps current network environment against existing database, detects relevant rules (fires them) and builds a dynamic web page. This web page is presented to a mobile user in proximity. In fact, even the name of the application (context-aware browser) suggests the movement of this functionality in a mobile browser. This would eliminate the separate rule base as well as the special (separate) application. In fact, the standard mobile browser should play a role of this application. Rules for the content (data snippets) must be specified directly on the mobile web pages. And data snippets itself are HTML code chunks anyway. As applied implementations, we can mention, for example, Internet of Things applications [15,16]. The usage is very transparent. Data snippets (data, presented to mobile users) depends on visibility for some Wi-Fi access points. It lets us specify the positions for mobile users inside of some building (campus, etc.) Mobile users will see different information for different positions. And this approach does not use geo-coordinates at all. The next interesting direction is EU project FI-WARE [17]. Integration with the FI-CONTENT platform is one of the nearest goals.

3 Information Services

Technically, for the reuse of information about network proximity, we can talk about the two approaches. At the first hand, the implementation of a mobile

browser can follow the same ideology that supports geo-coding in HTML5 [18]. A function from browser's interface

navigator.geolocation.getCurrentPosition()

accepts as a parameter some user-defined callback (another function). The callback should be called as soon as geo-location is completed. Obtained data should be passed as parameters. Note, that the whole process is asynchronous. By the analogue with the above-mentioned model, a mobile browser can add a new interface function. E.g., *getNetworks()* this function will accept a user-defined callback for accumulating network information (current fingerprint). A good candidate for data model is JSON. The browser will pass fingerprint as a JSON array to a user-defined callback. Each element from this array describes one network and contains the following information:

- SSID - name for access point
- MAC - MAC-address
- RSSI - signal strength

Note, that scanning networks is an asynchronous process in mobile OS. So, callback pattern is a good fit for this. Firefox OS is closest in ideology to this approach [19]. Also, Firefox OS offers Bluetooth API [20]. It has got the similar ideology, but there is no general unifier (e.g., even fields for objects are different). It should be possible, of course, to create some unified wrapper (shell), which will give a general list of networks. The biggest problem (we are not mentioning here the own prevalence and popularity for Firefox OS) is the status for both APIs. Wi-Fi API has just been scheduled yet. At the same time, the Bluetooth API exists, but it is declared preferred (privileged). Privileged APIs can be used by the operating system only. So, it could not be used in applications. The reason for this solution is security. API combines both network scanning and network connection (data exchange). It is the wrong design by our opinion. APIs functionality should be separated. The above-mentioned SpotEx approach is not about the connectivity. Mobile OS should use two separate APIs: one for scanning (networks poll) and one for connecting. Polling for networks does not require data exchange. So, scanning API is safe, and it should not be privileged. It is simple - we should have *WiFiManager* interface (as is, and it could be privileged), and *WiFiScan* with only one function *getNetworks()*:

```
<script>
function callback_function(json_data ) { ... }
WiFiScan.getNetworks(callback_function);
</script>
```

The callback function can loop over an array of existing networks IDs and show (hide) HTML div blocks with data related (associated) to the existing (visible) networks. Actually, it is a fundamental question. Traditionally, wireless networks on mobile phones are used as networks. But they are sensors too. The fact that some network node is reachable (visible) is a separate issue. And it

could be used in mobile applications even without the ability to connect to that node. It is the main idea behind SpotEx, and it is the feature (option) we suggest to embed into mobile browsers. How can we present our rules for network proximity? As per our suggestion, each data snipped should be presented as a separate div block in HTML code. E.g., the above-mentioned example looks so:

show content for Cafe

We can use CSS styles to hide/show this block. And this CSS visibility attribute depends on the visibility of Wi-Fi (Bluetooth) nodes. Of course, CSS visibility could be changes in JavaScript. So, our rules could be implemented in JavaScript code. We can directly present the predicates in our code, or describe their parts in CSS too. HTML5 custom attributes are good candidates for new attributes [21]. It means also, that adding some set of rules to existing web page looks like as adding (including) some JavaScript code (JavaScript file).

In general, this approach could change the paradigm of designing mobile web sites. It eliminates the demand to make separate versions for local sites or events. It is enough to have one common site with local offers (events, etc.) placed in hidden blocks. Blocks will be visible to mobile users in a proximity of some network nodes. Local blocks visibility depends on the network nodes visibility and so, it depends on the current location of mobile users. E.g., for the above-mentioned example, mobile users opened Cafe site being physically present in the proximity of Cafe, will see different (additional) data compared with any regular mobile visitor.

Of course, single data source (just one web site) support simplifies (makes it cheaper) the maintenance during life time. Web Intents [22] present the next interesting model for this approach. The Web Intents formation is a client framework (everything is executed in the browser) for the monitoring (polling) and building services interaction within the application. Interactions include data exchange and transfer of control. Web Intents form the core architecture of Android OS [23], but their future status is still unknown after some initial experiments from Google. We should note in this context a similar (by its concept) initiative from Mozilla Labs - Web Activities [24]. But the further status of this initiative is also unclear.

The next possible toolbox is seriously underrated in our opinion. It is a local web server. The first implementation, as far as we know, refers to the Nokia [25]. In our opinion, this is one of the most promising areas for communicating with phone sensors. The next possible idea resembles in some ways the old projects with WAP (Wireless Access Protocol). In this case, a mobile device used some intermediate server (WAP Gateway) for access to internet resources. This intermediate server should be able to collect sensing information (including network sensors). Internet service will get sensing info from our proxy.

4 A Generic Approach for Web Sensing

In this part, we would like to discuss the more generic approach (approaches) for embedding sensing information into web pages. As a workaround and prototype for this development, we can present a custom *WebView* for Android. On Android platform is possible to access from JavaScript to Java code for a web page, loaded into *WebView* control. Java code will provide a list of nearby network nodes (calculate the network fingerprint). The key moment here is the need for an asynchronous call from JavaScript, because scanning for wireless networks in Java is the asynchronous process. Let us describe this approach a bit more detailed. On Android side we activate JavaScript interface:

```
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
WebView webView = new WebView(this);
setContentView(webView);
WebSettings settings = webView.getSettings();
settings.setJavaScriptEnabled(true);
webView.addJavascriptInterface(new MyJavascriptInterface(), “Network”); }
```

Now we can describe our Java code for getting network fingerprint. As a parameter, we will pass a name for callback function in JavaScript.

```
@JavascriptInterface
public void getNetworks(final String callbackFunction) { }
```

We skip the code for network scanning and demonstrate the final part only. As soon as a fingerprint in obtained, we can present it as JSON array and invoke our callback:

```
webView.loadUrl(“javascript:” + callbackFunction + “(‘ “ + data + ” ’)”);
```

And on our web page, we can describe our callback function and call Java code:

```
function f_callback(json) { }
Network.getNetworks(“f_callback”);
```

This approach lets us proceed network proximity right in JavaScript (in other words, right on the web page). Actually, by the similar manner we can work with other sensors too. It is so-called Data Program Interface [26]. We would like to see something similar as a standard feature in the upcoming versions of Android.

5 Conclusion

The paper discusses the use of information about the network environment to create dynamic web pages. We propose several approaches to the implementation of a mobile browser that can handle data on a network (network proximity) to provide users with information tied to the current context. Also, we considered possible implementation details. The basic idea is to separate the functional for scanning network information and real data exchange.

References

1. Holdener, A.T.: HTML5 Geolocation. O'Reilly Media Inc., Sebastopol (2011)
2. Namiot, D., Sneps-Sneppe, M.: Where are they now – safe location sharing. In: Andreev, S., Balandin, S., Koucheryavy, Y. (eds.) NEW2AN/ruSMART 2012. LNCS, vol. 7469, pp. 63–74. Springer, Heidelberg (2012)
3. Namiot, D., Sneps-Sneppe, M.: Geofence and network proximity. In: Balandin, S., Andreev, S., Koucheryavy, Y. (eds.) NEW2AN/ruSMART 2013. LNCS, vol. 8121, pp. 117–127. Springer, Heidelberg (2013)
4. Prasad, M.: Location Based Services, GIS Development, pp. 3–35 (2002)
5. Tabbane, S.: Location management methods for third generation mobile systems. IEEE Commun. Mag. **35**(8), 72–78 (1997). 10.1109/35.606034
6. Namiot, D.: Context-aware browsing - a practical approach. In: 6th International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST), pp. 18–23 (2012). http://dx.doi.org/10.1109/NGMAST.2012.13
7. Schilit, G., Theimer, B.: Disseminating active map information to mobile hosts. IEEE Netw. **8**(5), 22–32 (1994). 10.1109/65.313011
8. Namiot, D., Sneps-Sneppe, M.: Context-aware data discovery. In: 2012 16th International Conference on Intelligence in Next Generation Networks (ICIN), pp. 134–141 (2012). http://dx.doi.org/10.1109/ICIN.2012.6376016
9. Kolodziej, K., Danado, J.: In-building positioning: modeling location for indoor world. In : Proceedings of the 15th International Workshop on Database and Expert Systems Applications, pp. 830–834. IEEE (2004). http://dx.doi.org/10.1109/DEXA.2004.1333579
10. Sharma, P., Xu, Z., Banerjee, S., Lee, S.: Estimating network proximity and latency. ACM SIGCOMM Comput. Commun. Rev. **36**(3), 39–50 (2006). 10.1145/1140086.1140092
11. Namiot, D.: Network proximity on practice: context-aware applications and Wi-Fi proximity. Int. J. Open Inf. Technol. **1**(3), 1–4 (2013)
12. Cheng, Y.C., Chawathe, Y., LaMarca, A., Krumm, J.: Accuracy characterization for metropolitan-scale Wi-Fi localization. In: Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services, pp. 233–245. ACM (2005). http://dx.doi.org/10.1145/1067170.1067195
13. Namiot, D., Schneps-Schneppe, M.: About Location-aware Mobile Messages: Expert System Based on WiFi Spots. In: 5th International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST), pp. 48–53. IEEE (2011). http://dx.doi.org/10.1109/NGMAST.2011.19
14. Namiot, D., Sneps-Sneppe, M.: Wi-Fi proximity as a service. In: 1st International Conference on Smart Systems, Devices and Technologies, SMART, pp. 62–68 (2012)
15. Schneps-Schneppe, M., Namiot, D.: Open API for M2M applications: what is next?. In: 8th Advanced International Conference on Telecommunications, AICT, pp. 18–23 (2012)
16. Namiot, D., Schneps-Schneppe, M.: Smart cities software from the developer's point of view. In: 6th International Conference on Applied Information and Communication Technologies (AICT), LUA Jelgava, Latvia, pp. 230–237 (2013)
17. Castrucci, M., Cecchi, M., Priscoli, F.D., Fogliati, L., Garino, P., Suraci, V.: Key concepts for the future internet architecture. In: Future Network & Mobile Summit (FutureNetw), pp. 1–10. IEEE (2011). http://dx.doi.org/10.1145/1067170.1067195

18. Aghaee, S., Cesare, P.: Mashup development with HTML5. In: Proceedings of the 3rd and 4th International Workshop on Web APIs and Services Mashups, pp. 10. ACM (2010). http://dx.doi.org/10.1145/1944999.1945009
19. Amatya, S., Kurti, A.: Cross-platform mobile development: challenges and opportunities. In: Trajkovik, V., Anastas, M. (eds.) ICT Innovations 2013. Advances in Intelligent Systems and Computing, vol. 231, pp. 219–229. Springer, Heidelberg (2014)
20. Paul, A., Steglich, S.: Virtualizing devices. In: Crespi, N., Magedanz, T., Bertin, Emmanuel (eds.) Evolution of Telecommunication Services. LNCS, vol. 7768, pp. 182–202. Springer, Heidelberg (2013)
21. Cazenave, F., Quint, V., Roisin, C.: Timesheets. js: when SMIL meets HTML5 and CSS3. In: Proceedings of the 11th ACM Symposium on Document Engineering, pp. 43–52. ACM (2011). http://dx.doi.org/10.1145/2034691.2034700
22. Zheng, C., Shen, W., Ghenniwa, H.H.: An intents-based approach for service discovery and integration. In: IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 207–212. IEEE (2013). http://dx.doi.org/10.1109/CSCWD.2013.6580964
23. Chin, E., Felt, A.P., Greenwood, K., Wagner, D.: Analyzing inter-application communication in android. In: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, pp. 239–252. ACM (2011). http://dx.doi.org/10.1145/1999995.2000018
24. Firtman, M.: Programming the Mobile Web, pp. 619–624. O'Reilly Media Inc, Sebastopol (2013)
25. Oliveira, L., Ribeiro, A.N., Campos, J.C.: The mobile context framework: providing context to mobile applications. In: Streitz, N., Stephanidis, C. (eds.) DAPI 2013. LNCS, vol. 8028, pp. 144–153. Springer, Heidelberg (2013)
26. Namiot, D., Sneps-Sneppe, M.: On software standards for smart cities: API or DPI. In: Proceedings of the 2014 ITU Kaleidoscope Academic Conference: Living in a Converged World-Impossible Without Standards?, pp. 169–174. IEEE (2014)