

The TELAR Mobile Mashup Platform for Nokia Internet Tablets

Andreas Brodt
Nokia Multimedia
Yrttipellontie 6
90230 Oulu, Finland
andreas.brodt@nokia.com

Daniela Nicklas
Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart, Germany
dnicklas@acm.org

ABSTRACT

With the Web 2.0 trend and its participation of end-users more and more data and information services are online accessible, such as web sites, Wikis, or web services. The integration of this plethora of information is taken over by the community: so-called Mashups—web applications that combine data from more than one source into an integrated service—spring up like mushrooms, because they can be easily realized using script languages and web development platforms. Another trend is that mobile devices that get more and more powerful have ubiquitous access to the Web. Local sensors (such as GPS) can easily be connected to these devices. Thus, mobile applications can adapt to the current situation of the user, which can change frequently because of his or her mobility.

In this demonstration, we present the TELAR Mashup platform, a client-server solution that facilitates the creation of adaptive Mashups for mobile devices such as the Nokia Internet Tablets. On the server side, wrappers allow the integration of data from web-based services. On the client side, a simple implementation of the DCCI specification is used to integrate context information of local sensors into the mobile Web browser, which adapts the Mashup to the user's current location. We show an adaptive, mobile Mashup on the Nokia N810 Internet Tablet.

1. INTRODUCTION

The proliferation of public web services and other online data sources enables new services and applications that just combine existing information in a new manner. So-called Mashups integrate data and services from multiple sources to provide innovative services (like combining crime statistics with geographical information to visualize the risk distribution within a certain neighborhood [1]).

Mashups are mostly realized by web pages that leverage script languages such as JavaScript, which enables better user interactivity by locally executed functions and the dynamic loading of data from web services. These techniques

are often associated with the Web 2.0 trend. Typically, Mashups are created dynamically from existing data sources that have no knowledge about their participation. Thus, they can change their interfaces and data formats at any time, which adds a new flavor to the general data integration problem.

Another trend are mobile systems that have become more and more powerful in the last years. Soon, users expect their handheld devices to run the same applications as their desktop systems do. One example is the Nokia N810 Internet Tablet that possesses 128 MB of RAM, 256 MB of flash memory and an OMAP2420 microprocessor at 400 MHz. This is sufficient to run maemo (a Linux-based operating system) and thus numerous Linux applications. In particular, a Mozilla-based web browser is available for the Nokia Internet Tablets, which aims at providing the same features which desktop browsers typically provide today.

Combining these two trends—Mashups and adaptive applications on mobile devices—offers great additional value to the user. By integrating multiple data sources into one experience, new services can be created that are tailored to the user's personal needs. And by using local sensor data on a mobile device, this experience can be adapted to the user's current situation. In this demonstration, we present the TELAR Mashup platform, which facilitates the creation of location-aware Mashups for mobile devices such as the Nokia Internet Tablets.

A screenshot of an adaptive mobile Mashup that can be realized with the TELAR Mashup platform is depicted in Figure 1. The basis for the integration is a map that is gathered from an online map service (here: Google Maps) and centered to the user's current location, obtained from a GPS sensor. Three different data providers are integrated that provide so-called points of interest (POIs): WLAN access points from Fon [2], photos from Panoramio [3], and geo-referenced Wikipedia articles offered by GeoNames [4]. The POIs are displayed on the map using different symbols. As the user moves, the map stays centered to her position and a trace of her movement is displayed (this behavior can be switched off). The user can also add or remove POI data providers during run-time.

To realize mobile Mashups like the one described in this scenario, we developed a client-server solution, the TELAR Mashup platform. On the server side, wrappers allow the integration of data from web-based services. On the client side, the *Delivery Context Client Interfaces* (DCCI) specification [5] is used to integrate context information of local sensors into the mobile web browser, which adapts the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

EDBT'08, March 25–30, 2008, Nantes, France.

Copyright 2008 ACM 978-1-59593-926-5/08/0003 ...\$5.00.

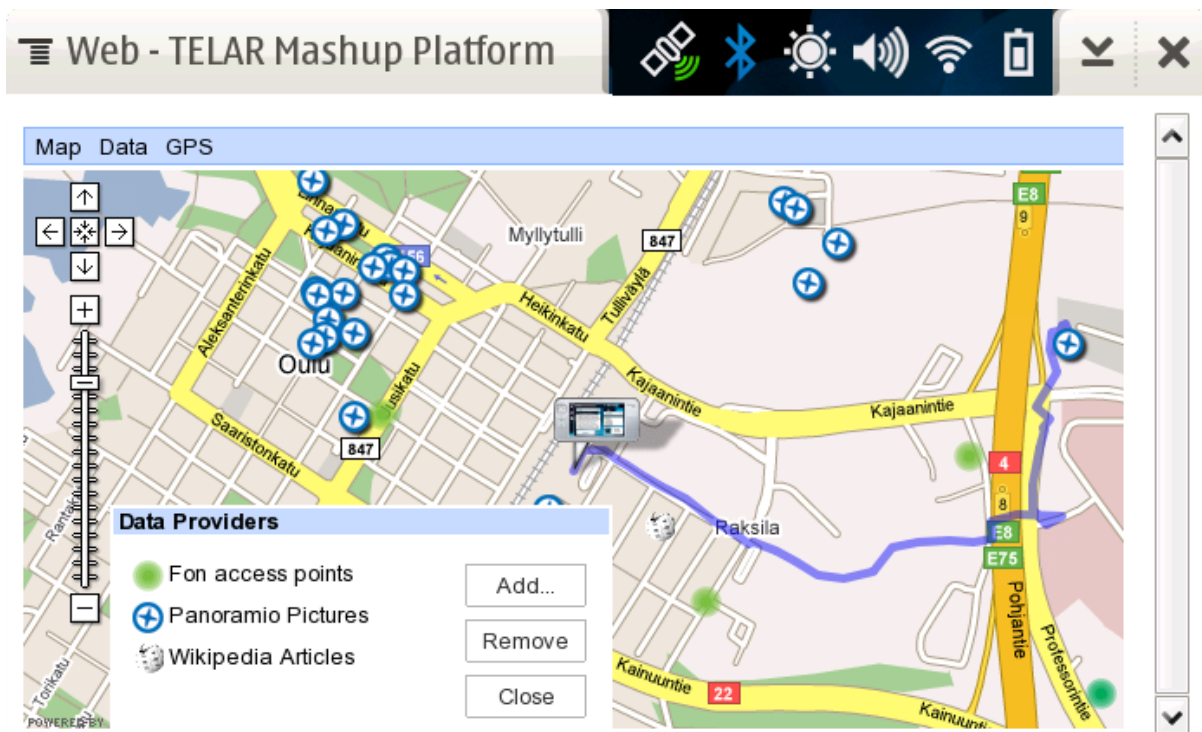


Figure 1: Screenshot of a sample Telar Mashup on a Nokia N810 Internet Tablet

Mashup to the user's current location.

In this demonstration we show:

- a three-tier system architecture for mobile Mashups that uses a general wrapper approach for data integration, and
- context integration on the device based on the DCCI framework, which enables Mashups to adapt to the user's location.

The remainder of the paper is organized as follows: in Section 2, we briefly discuss the state of the art regarding Mashups and Mashup platforms. Section 3 presents the demonstration architecture and the POI data format. In Section 4 we outline four presentable demo scenarios, and Section 5 we describe the infrastructure requirements of this demo. Finally, Section 6 concludes this demo paper.

2. STATE OF THE ART

Mashups are web application hybrids that integrate data from different sources to provide a value added service. They leverage the availability of open web services, RSS feeds, or extract information out of regular web pages using screen scraping. A popular combination is to display information from different sources on a map (Geo-Mashups), or to enrich search results from one source (e.g., a hotel finder) with information from others (e.g., recommendations and pictures).

The Mashup portal programmableWeb.com lists 2585 Mashups on December 14, 2007, with an average of 3.16 new Mashups per day. Figure 2 shows the distribution over different genres: over one third are Geo-Mashups, 16% are Multimedia Mashups (video and photo).

Since this trend is driven by the web community, only few researchers have covered this topic yet. Floyd et al. [6] show how Mashup techniques can be used for rapid prototyping in user-centered software development processes. A study at the Human-Computer Interaction Institute of the Carnegie Mellon University showed that Mashups can be even used for user programming [7]. IBM [8] emphasizes the great benefits of so-called *Enterprise Mashups*, information heavy applications that integrate distributed business information within an enterprise in a quick and dynamic way. Erik Wilde [9] applied the Mashup idea to the management of large knowledge bases.

Most of the existing Mashups are programmed manually. However, there exist a number of Mashup platforms that facilitate the development: *Mash-o-matic* [10] can be used to generate Geo-Mashups. The focus of the *SPARCE project* is so-called superimposed information. With online tools like *Yahoo! pipes* [11] or Microsoft's *Popfly* [12], Mashups can be built out of pre-defined components and combined using interactive drag-and-drop interfaces. IBM's *QEDWiki* [13] is an AJAX interface to combine user interface components that are connected to external data providers. IBM is also working on a data Mashup service for web and enterprise information called *DAMIA* [14] (sometimes referred to as *MAFIA-Mashup Fabric In Almaden*). The *Openkapow* platform [15] realizes Mashups as a combination of so-called robots, which extract information from RSS streams, web services, or via screen scraping.

Our proposed platform differs from these approaches: The TELAR Mashup platform exploits heterogeneous third-party data providers in order to create Geo-Mashups. The data providers can be configured at run-time and are intended



Figure 2: Mashup statistics from programmableweb.com [16]

to be web-based, but do not necessarily have to be. The presentation can be modified to some extent, as the TELAR Mashup platform can be embedded into any HTML page, and dialogs and menus can be customized using CSS. In contrast to QEDWiki, we do not aim at providing a framework for arbitrary situational applications. However, the most important difference is that the TELAR Mashup platform supports integrating the user's location into a Geo-Mashup. Being intended for mobile devices, such as the Nokia Internet Tablets, the TELAR Mashup platform can create Mashups about the user's current environment, utilizing the Nokia N810's builtin GPS sensor or an external GPS device connected to a Nokia N800 via Bluetooth.

3. ARCHITECTURE

A graphical overview of the architecture of the TELAR Mashup platform is given in Figure 3. Being a typical AJAX-based Mashup, the TELAR Mashup platform consists of three tiers: a Mashup is viewed in the client tier. The web browser loads the Mashup page and starts the JavaScript code of the Mashup client AJAX application. The Mashup page is loaded from the Mashup server, which resides on the Internet and constitutes the server tier. Data offered by third-party data providers is used, which are distributed throughout the Internet. The map is loaded from Google Maps, which, together with the data providers, makes up the data provider tier.

A Mashup consists of an HTML page that includes the JavaScript files of the Mashup client. A static XML file is used to provide the Mashup with configuration parameters, most notably, which data providers to use. The Mashup client asynchronously loads the configuration file when the Mashup page is loaded. The Mashup client displays a Google map and integrates POI data from the data providers, which are queried via asynchronous HTTP requests. As the `XMLHttpRequest` JavaScript object only allows requests to the same server from which the JavaScript code was loaded, wrappers for accessing the data providers are needed. At the same time, the wrappers convert the heterogeneity of data formats used by the different data providers to a consistent uniform format understood by the Mashup client. This is required, as, typically, neither the API nor the data format of a data provider follows any standard.

The wrappers provide a consistent interface to the Mashup client for accessing POI data. As the wrappers are a number of independent scripts that reside on the Mashup server, it

is easy to add wrappers for new data providers any time. All that is needed is to implement a wrapper (usually about 30 lines of code in a scripting language) and upload it to the Mashup server. In order to use the new wrapper, its URL needs to be added to the configuration XML file. The Mashup client does not need to be modified.

As the wrappers are totally independent from each other, no data integration across the wrappers is done. This can theoretically lead to issues concerning multiple representations of identical objects. This, however, strongly depends on the selected data providers.

The user's location is integrated into the Mashup by extending the Mozilla-based web browser for the Nokia Internet Tablets. Two Mozilla extensions are used: the DCCI module and the GPS access module. The DCCI module implements the *Delivery Context Client Interfaces* (DCCI) specification [5] and acts as the interface for providing context data to web pages. The Mashup client registers itself as an event listener to the DCCI module and is notified every time the user's location changes. The GPS access module connects to the GPS device and ships the location information to the DCCI module. The two Mozilla modules communicate directly via XPCOM, the component framework of the Mozilla browser.

The data flow inside the TELAR Mashup platform works as follows: Whenever the GPS access module obtains a new location from the GPS device, the location information is updated in the DCCI module. The Mashup client, which is registered as an event listener to the DCCI module, is notified about the change via DOM events. Subsequently, the Mashup client updates the user's location on the map and centers the map to the new location. If the area shown on the map has significantly changed, the Mashup client sends asynchronous HTTP requests to the data provider wrappers, in order to obtain POI data for the new map area. The data provider wrappers translate these requests to calls to the particular APIs of the data providers and convert the resulting data into a unique data format understood by the Mashup client. Finally, the Mashup client reads the reply sent by the wrappers and displays the data on the map.

POI Data Format

The data provider wrappers translate the various data formats used by the data providers into a uniform format understandable by the Mashup client. Our first approach was to use an extended version of GeoRSS [17] for this purpose. GeoRSS is a popular format for describing geographically annotated objects. Based on Atom or RSS, GeoRSS allows to augment objects with spatial information in a simple way. By using GeoRSS, existing tools and web pages supporting GeoRSS could be used for testing the data provider wrappers. Besides, the value of the wrappers increased as they could be potentially useful for other applications as well.

When the Mashup client and some data provider wrappers were implemented, the first tests showed that the performance of the TELAR Mashup platform was insufficient. While the Mashup client worked fine on a state-of-the-art desktop PC, it could take several minutes until a Mashup was completely constructed on a Nokia Internet Tablet. A similar experience was made on a Pentium II PC. A first analysis revealed that most of the time was spent for parsing the GeoRSS data that was retrieved from the data provider wrappers. In addition to that, it took time to add the parsed

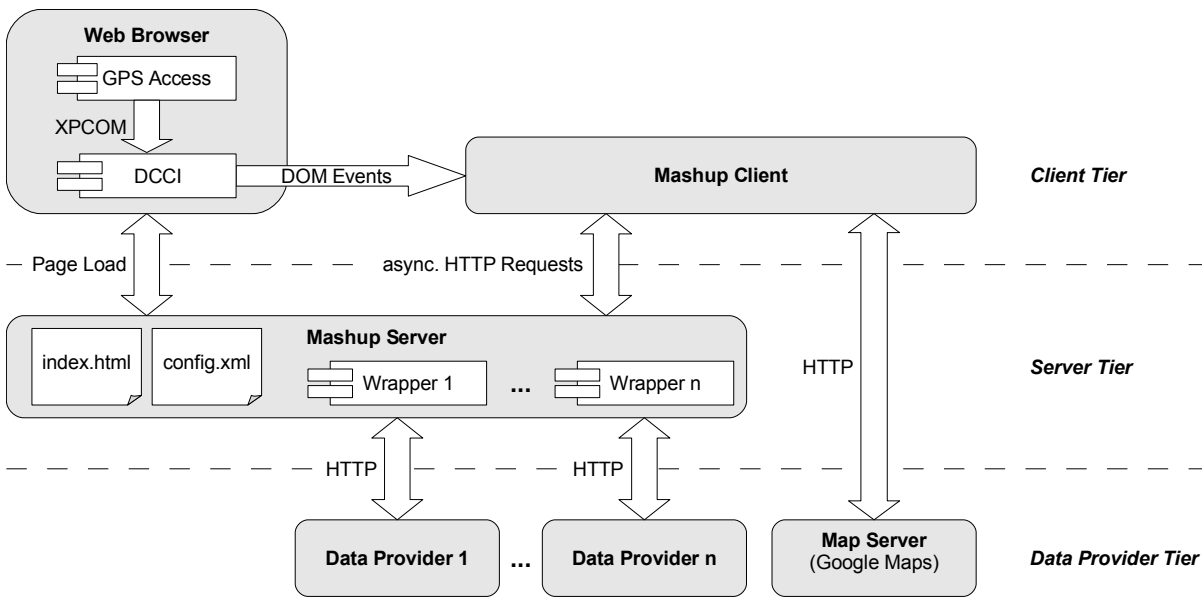


Figure 3: Architecture of the Telar Mashup platform

POIs to the map. Both steps are done by JavaScript code interpreted in the browser. In contrast to a desktop PC, the processor of the Nokia Internet Tablet was not powerful enough to do this job quickly.

In order to improve performance, the standardized GeoRSS data format was replaced by a proprietary JSON format [18]. As JSON is a subset of JavaScript, it can be parsed very efficiently using the `eval()` JavaScript function, which the web browser implements in native code.

4. DEMO SCENARIOS

In the demonstration we show four scenarios: simulated positioning, manual positioning, data provider management, and, if possible, real GPS positioning outside.

4.1 Simulated Positioning

For simulated positioning, the TELAR Mashup platform is modified, so that the Web browser periodically receives simulated GPS data. The Mashup Client responds by centering the map to the simulated GPS position, marking the position on the map and drawing the simulated trace. POI data from different data providers is loaded according to the current map area and shown. Additional information about a POI is shown when the user clicks on it. The simulated positioning scenario can be shown on a desktop or laptop browser as well as on a Nokia N810 Internet Tablet.

4.2 Manual Positioning

The scenario without automatic positioning illustrated the case when no GPS data is available. This can be due to the user being indoors, because the user does not have a GPS device, or when the Mashup is viewed from a (e.g. desktop) browser without the browser extensions of the TELAR Mashup platform. As no position information is available, the Mashup is centered to its configured default position. The user has the possibility to drag the map to different positions. As above, POI data from different data providers

is loaded and shown according to the current map area.

4.3 Data Provider Management

The data provider management scenario shows how the data providers, from which the POI data is queried, can be dynamically added and removed. The user can list the data providers that are currently in use and can remove them. The POIs are subsequently removed from the map. New data providers can be added by entering their URL.

4.4 Real GPS Positioning (optional)

Real GPS positioning is the identical scenario to simulated positioning, except that the location data is obtained from a real GPS device. Naturally, for this demonstration it must be possible to receive GPS signals. The user can walk around and will see her position being updated on the map periodically. The walked route will be drawn on the map as a track.

5. DEMO INFRASTRUCTURE REQUIREMENTS

This section outlines the infrastructure requirements we need to show the demo.

5.1 WLAN Connectivity and Internet Access

For the client devices to be able to draw a map, a connection to the Internet is required. For the Nokia N810 this can be achieved via Bluetooth and a mobile phone as well, but for performance and reliability, we strongly suggest WLAN connectivity.

5.2 Mashup Server

The TELAR Mashup platform requires a server from which the Mashup is loaded and where the data provider wrappers reside. For demonstration purposes, the data provider wrappers can be modified to return local data, so that Internet access is not mandatory for the server, as long as the clients can

access the server. Depending on the chosen implementation language of the data provider wrappers, naturally, the server must provide the respective runtime environment. The current data provider wrappers are implemented in PHP 5 and require the XSL and JSON extensions. As long as the server is accessible, it can reside anywhere in the Internet. However, for demonstration purposes we will show the Mashup server on a laptop.

5.3 GPS signal

The Real GPS positioning scenario described in Section 4.4 requires the possibility to receive a GPS signal. As we do not expect to have a GPS repeater available, this most likely has to be solved by going outside into an area where some open sky is visible (to get the satellite signal).

6. CONCLUSION

In this demonstration, we present the TELAR Mashup platform, a client-server solution that facilitates the creation of location-based Mashups for mobile devices such as the Nokia Internet Tablets. On the server side, wrappers allow the integration of data from web-based services. On the client side, a simple implementation of the DCCI specification is used to integrate context information of local sensors into the mobile web browser, which adapts the Mashup to the user's current location. For that, a simple context ontology was used that can be exchanged as soon as real semantic standards for representing context appear. This is important since context is not only location: to realize real context-aware applications that adapt to the mobile user's current situation, more context information has to be considered. With our design, this can be achieved by developing additional browser extensions that provide additional context information within the context ontology. Because of its event-based realization, DCCI-based web applications can then dynamically react on the availability of new context sources and use the information for their adaptation.

We share our experience with rich web applications making intensive use of AJAX on a mobile device with limited resources: the performance can be sufficient for small applications but is not yet really satisfying. We address the issue of heterogeneous data provider APIs and data formats in the context of Web 2.0 Mashups. Especially when building location-based systems, it is currently a challenge to integrate the numerous existing data providers into one application. The concept of offering simple wrappers at the Mashup site that the applications choose dynamically is only a first step towards sophisticated web data integration methods.

7. REFERENCES

- [1] A. Holovaty and W. Miner. Chicago Crime website. <http://chicagocrime.org>. retrieved at 2007-06-13.
- [2] Fon website. <http://www.fon.com/>. retrieved at 2007-09-21.
- [3] Panoramio website. <http://www.panoramio.com/>. retrieved at 2007-09-21.
- [4] Geonames wikipedia articles. <http://www.geonames.org/export/wikipedia-webservice.html#wikipediaBound%ingBox>. retrieved at 2007-09-21.
- [5] K. Waters, R. A. Hosn, D. Raggett, S. Sathish, M. Womer, M. Froumentin, and R. Lewis. Delivery

Context: Client Interfaces (DCCI) 1.0. Working draft, W3C, July 2007.

- [6] I. R. Floyd, M. C. Jones, D. Rathi, and M. B. Twidale. Web mash-ups and patchwork prototyping: User-driven technological innovation with Web 2.0 and Open Source software. In *HICSS*. IEEE Computer Society, 2007.
- [7] J. Wong and J. I. Hong. Making mashups with marmite: towards end-user programming for the web. In M. B. Rosson and D. J. Gilmore, editors, *CHI*. ACM, 2007.
- [8] A. Jhingran. Enterprise information mashups: Integrating information, simply. In U. Dayal, K.-Y. Whang, D. B. Lomet, G. Alonso, G. M. Lohman, M. L. Kersten, S. K. Cha, and Y.-K. Kim, editors, *VLDB*. ACM, 2006.
- [9] E. Wilde. Knowledge organization mashups. TIK Report 245, ETH Zürich (Swiss Federal Institute of Technology), March 2006. available at <http://dret.net/netdret/publications#wil06f>.
- [10] S. Murthy, D. Maier, and L. M. L. Delcambre. Mash-o-matic. In D. C. A. Bulterman and D. F. Brailsford, editors, *ACM Symposium on Document Engineering*. ACM, 2006.
- [11] Yahoo! pipes website. <http://pipes.yahoo.com>. retrieved at 2007-06-13.
- [12] Microsoft Developer Division: Popfly project website. <http://www.popfly.ms/>. retrieved at 2007-09-21.
- [13] B. Curtis, W. Vicknair, and S. N. (IBM). QEDWiki project website. <http://services.alphaworks.ibm.com/qedwiki/>. retrieved at 2007-09-21.
- [14] IBM Almaden Research Center and IBM Information Management: DAMIA project website. <http://services.alphaworks.ibm.com/damia/>. retrieved at 2007-09-21.
- [15] kapow Technologies: openkapow. <http://openkapow.com/>. retrieved at 2007-06-13.
- [16] J. Musser. Programmableweb portal. <http://www.programmableweb.com>. retrieved at 2007-12-14.
- [17] C. Reed, R. Singh, R. Lake, J. Lieberman, and M. Maron. An introduction to GeoRSS: A standards based approach for geo-enabling RSS feeds. White Paper OGC 06-050r3, Open Geospatial Consortium Inc., July 2006.
- [18] D. Crockford. The application/json media type for javascript object notation (JSON). Request for Comments 4627, The Internet Society, July 2006.