The *webinos* Architecture: A Developer's Point of View

Paolo Vergori[1], Christos Ntanos[2], Marco Gavelli[1], and Dimitris Askounis[2]

[1] Istituto Superiore Mario Boella, MultiLayer Wireless solutions (MAIN)
Turin, Italy
{vergori,gavelli}@ismb.it
[2] National Technical University of Athens,
Electrical and Computer Engineering Department
Athens, Greece
{cntanos,askous}@epu.ntua.gr

Abstract. This work describes the architecture proposed by the *webinos* EU project, which aims at developing software components for the future Internet, in the form of Web Runtime Extensions. It discusses the *webinos* architecture from a developer's point of view, presenting an overview of its main advantages, such as context-awareness capabilities and distributed APIs in an intrinsic secure environment. It also shows how these features can in practice prove beneficial to the development of ubiquitous and secure web applications based on standard technologies like HTML, CSS and JavaScript.

Keywords: webinos, mobile applications, middleware, developers, framework, architecture, context, context awareness, distributed apis.

1 Introduction

The world of web development is rapidly changing. Requirements for developers are more strict and involve scenarios that were not taken into account in the past. These changes are mainly due to the steep increase of the range of available devices, their features, their capabilities and the need for persistent network connectivity. Furthermore, in this new world of distributed information, validating the same user across diverse devices and Operating Systems can be challenging with respect to security and privacy.

These factors are increasingly forcing developers into creating cross-platform applications, with cross-service capabilities that are able to share information among devices. Such features, though, introduce new security requirements that need to be addressed. In the end, the aim is to provide appropriate tools for the development of web applications that augment the user experience, while at the same time, relying on sand-boxed environments, like browsers, and on Operating Systems' integrated security.

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 391–399, 2013.

In this paper[1] we aim to point out how the *webinos* consortium addresses those needs and simplifies development tasks, by introducing an innovative, open-source middleware, to the development process. The *webinos* framework enhances application security, regardless of the Operating System on which the framework is running, without requiring any additional effort from the developer. In the following sections, we present an overview of other related solutions and the current state-of-the-art for middleware web based applications and we follow it with a brief breakdown of the *webinos* architecture. Next, we break down the main strengths of the *webinos* approach, and point out some of the key findings from the research on context-awareness, as it is implemented in *webinos*.

2 Background

As a cutting-edge middleware framework, *webinos* is closely related to the research on the development of middleware for distributed applications, a topic that is well document in literature. In general, middleware components aim to be generic across applications and industries, run on multiple platforms, be distributed, and support standard interfaces and protocols.[1, p. 5]

It is widely suggested that *'conventional binary programs will be limited to system software, whereas the vast majority of end user software will be developed using web technologies'*[13]. Due to the already well established ubiquity of web browsers, the *webinos* framework was steered early towards serving its functionality through web content. Best practices from W3C [15] refer to a Web Application as a *'Web page (XHTML or a variant thereof + CSS) or collection of Web pages delivered over HTTP, which use server-side or client-side processing (e.g. JavaScript) to provide an 'application-like' experience within a Web browser. Web applications are distinct from simple Web content, in that they include locally executable elements of interactivity and persistent state.'*

Despite the fact that numerous technologies that can be delivered through HTML, such as JavaScript, CSS and DOM models, have enabled a more dynamic environment for Web applications, web pages, in general, are still providing an insufficient experience to nomad users. This is due to the fact that ubiquity of information and functionality has not been taken seriously into consideration yet, nor is data exchange aware of the context in which it is performed.

In order to enhance the already established specifications with an outlook towards the future of computing, the *webinos* consortium quickly underlined the importance of identifying user requirements that would cover present, but more importantly, future needs. The result was a series of original use-cases and scenarios, where the framework would play a key role in enabling developers overcome hurdles associated with the diversity of the modern and future device ecosystems.

[1] The work reported in this paper was granted by the *webinos* project co-funded by European Union Seventh Framework Programme.

Web Applications vs Widgets

The distinction between web applications and widgets has been thoroughly covered by a previous investigation from webinos partners [8], but it is equally relevant from the application developer's point of view. The W3C identifies[16, p. 3] a widget as a pre-packaged, purpose-built web application. Moreover, in this specification[16] it is explained that web applications and widgets *'differ from HTML5-style web applications in the sense that for web widgets, the installation formats and practices have been predefined specifically to resemble those of traditional (native) applications. For instance, each W3C web widget is packaged into a separate WGT (ZIP) file that is installed explicitly in the same fashion as binary applications. This is different from HTML5 applications that run in a web browser without explicit installation.'*

It is clear that web applications are not prepackaged for installation, they do not offer an automated installation procedure, nor a maintenance system, and therefore they are static. Widgets can be more appealing for adoption by web developers, especially in the case of distributed computing applications like the ones that the *webinos* framework serves, because they provide a dedicated distribution package that delivers contents to users, adding an intrinsic value to the contents themselves.

3 Overview and Overall Architecture

The overall architecture of the framework that the *webinos* consortium is delivering is briefly depicted in the following section.

3.1 What Is *webinos*?

Webinos is an EU funded project aiming to deliver a platform for web applications across mobile, PC, home media and in-car devices. The webinos project has over twenty partners from across Europe spanning academic institutions, industry research firms, software firms, handset manufacturers and automotive manufacturers. *webinos* is a 'Service Platform' project under the EU FP7 ICT Programme. The main features of *webinos* are:

- *webinos* bases on the achievements of the web community and extends an open source web runtime environment
- *webinos* offers a common set of APIs to allow easy access to cross-user cross-service cross-device functionality in an open and secure manner
- *webinos* aims at easy programming of applications by offering a single virtual device that can consist of all devices owned by a user
- *webinos* creates open specifications and open source reference implementations that show the feasibility of the specifications and simplify their adaptation by the industry [17].

3.2 The Concept of the Personal Zone

The Personal Zone concept stands on top of the *webinos* architecture. The *Personal Zone Area (PZA)* is delimited by the perimeter defined by the ownership of devices. Each *webinos-enabled* device that belongs to the same user is considered to be inside this perimeter.

Each *PZA* is defined by a single point of synchronization, which all devices must authenticate against. This point is called a *Personal Zone Hub (PZH)* and its aim is to provide attestation, authentication and to act as an enforcement point for incoming requests. The actual implementation relies on a cloud positioning of the personal hub, assuming it is hosted securely, even if it's on a third party server. A set of available actions is provided to the *PZH* user, through a dedicated web UI to which the user logs-in via a custom OpenID [11] authentication mechanism. The available actions through this interface include adding a device to the *Personal Zone Area*, revoking certificates for every authenticated device in that area and checking their authentication status. The certificate revocation action is irrevocable and another certificate must be issued again from the device itself, before being issued to the *PZH* again.

Figure 1 depicts the architecture of *webinos* and the components with which an application can interact.

Every *webinos-enabled* device is running an instance of a *Personal Zone Proxy (PZP)*. The connection with the *Personal Zone Hub* is done through a secure TLS channel that mutually authenticates both sides, but in case of out-of-band communication, it can peer itself with an already authenticated *PZP* that belongs to the same *PZA*, maintaining security policies, due to the duplicated nature of the *Policy Enforcement Point* and *Policy Decision Point* in both *PZH* and *PZP* [7]. Its main role is to export device APIs as services in a completely secure manner. *Webinos Service Discovery* allows these services to be discovered and consequently be invoked using Remote Process Calls (RPCs) based on the JSON-RPC 2.0 Specification [4]. The *Personal Zone Proxy* connects to the *Webinos RunTime (WRT)* with a customized websocket implementation, called *secure channel*. The *WRT* is the place where all the developed applications belong and run. It has a *Widget Manager* for the installation, deletion and updating of applications that are packaged as widgets.

Making use of this architecture and the resulting development environment, the software engineer has the opportunity to build applications with the significant advantage of having all the information and the *webinos-enabled* device features in a distributed and secure overlay network. Moreover, all personal zones have context-awareness features enabled and as it is explained in section 4, developers can create advanced ubiquitous applications that can rely on the *webinos* framework.

3.3 Breakthroughs Introduced by *webinos*

Several similar frameworks have already been deployed in the past few years. They all have tried to address the cross-device concept on a distributed environment. Most, if not all, of these frameworks are missing one or more crucial

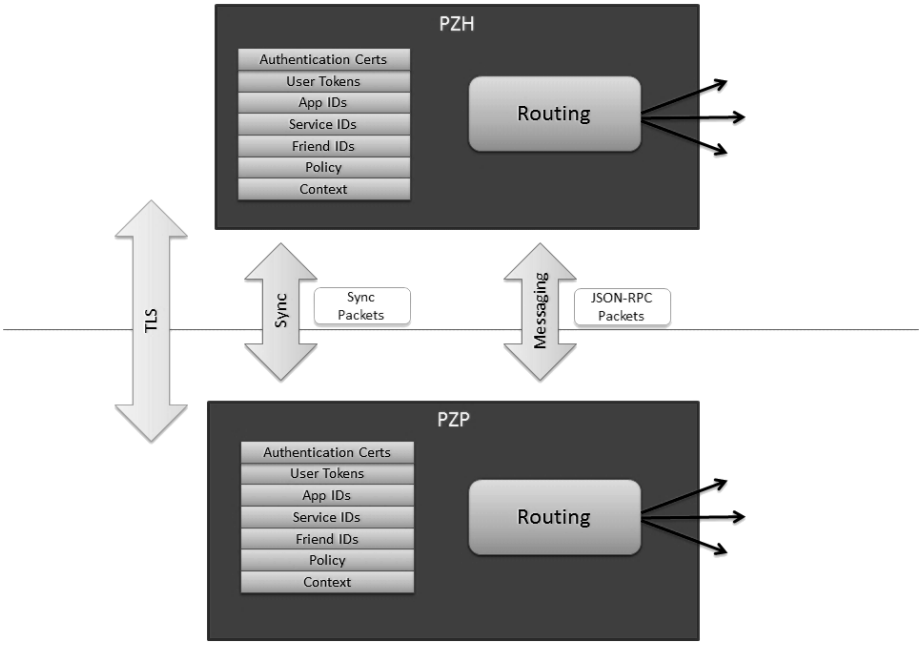

Fig. 1. The *webinos* architecture

functions that *webinos* aims to either ameliorate or introduce. PhoneGap [10] provides a precompiled framework that serves a platform dependent API subset, and runs on several Operating Systems. This approach lacks a pure multi-device, platform-independent development concept. Moreover, the compatible OSes are limited to the mobile device pool. Bondi is a web-based Operating System just for mobile devices. It creates a secure environment, in which it allows access to mobile functionality from a widget renderer. In July 2010, OMTP and the BONDI initiative were subsumed into the Wholesale Applications Community (WAC) [9]. The Wholesale Applications Community (WAC) is an organization aiming to create a unified and open platform to help mobile software developers easily write applications that can run on a variety of devices, Operating Systems and networks [14]. Both Bondi and WAC are closely related to the *webinos* Project. Nevertheless, *webinos* is more ambitious, due to its aim to overcome the mobile barrier by aiming to reach a wide variety of devices, such as set-top boxes, PCs, cars and, of course, tablets and smartphones as well.

The main goal of the *webinos* framework is to take computing to its next logical step, that of ubiquity. In order to do so, knowing the state of the device and the user at any given time, and making decisions based on that context (section 4) is crucial.

Context awareness and true cross-platform and cross-device applications cannot be achieved without the developer having access to a distributed API environment. With the use of a framework that provides such access, together with the context awareness capabilities of *webinos*, 'everyware' [3] applications are a step closer to reality.

Applications operating within a *webinos* PZA can share the state of the same application on any other device belonging to the same user, by seamlessly sharing context and application data across the personal zone. Moreover, the use of Remote Process Calls [4] from the *Webinos Run Time* to the exposed APIs of other devices within the PZA, will allow the developers to limit the functionality of their applications almost only my their own imagination.

4 Context and Distributed Information from Developers POV

The *webinos* platform aims to provide appropriate tools for software engineers, in order to enable and facilitate the development of cross-platform context-aware applications. *Context-aware computing* is a term introduced by *Bill Schilit* in 1994 [12] and was later popularized by the advancements of mobile computing, device, software and telecommunication interoperability. In computing, context can be described by its three important aspects: *where you are, whom you are with, and what resources are nearby.* Because of this, context is divided into three parts, *computing context* (e.g. available processors, devices accessible for user input and display, nearby resources) , *user context* (e.g. users location, collection of nearby people, user profiles and social situation) and *physical context* (e.g. lighting, temperature, noise and humidity level, traffic conditions).

4.1 Context-Aware Applications

A context-aware application either makes use of a single piece, a combination, or a time series of context related data, for example, the current state of the compass on a mobile phone, the current state of the ambient light sensor of a laptop computer or the geographic location and the closest Wi-Fi networks, in order to make an *automatic contextual reconfiguration* (e.g. increase the brightness of the monitor, enable Wi-Fi connectivity) or enable a *proximate selection* (e.g. highlight Points Of Interest geographically located near the user or print a document on the closest printer). In order to take advantage of such functionality, an application developer has to have access to means of *acquiring* contextual data, *storing* them, *filtering* them, *combining* them and *performing commands* based on the resulting information. For ubiquitous, distributed and context-aware computing applications, the aim is to provide appropriate *middleware* that can perform Remote Process Calls (RPCs), while at the same time introducing an *abstraction layer* that will facilitate the development process, by hiding the heterogeneity of the networking environment, supporting advanced coordination models among distributed entities and making as transparent as possible the distribution of computation [5].

4.2 Context-Awareness in *webinos*

The *webinos* project aims to provide a cross-platform level of abstraction for procedural calls, but at the same time, incorporate an additional data abstraction layer for use in third party context-aware and context-sharing applications that are *webinos-enabled*. The main data construct relating to contextual information in *webinos* is the *Context Object*. Inspired by the definition of a *meme* [2], a *Context Object* is a unit for carrying data that *uniquely defines a piece of contextual information*. For example, whereas a call to a GPS sensor will return a number of outputs (latitude, longitude, heading, speed, accuracy and altitude accuracy), one relevant Context Object that can be called *MyLocation* will contain only the most relevant data that can define the unit, by excluding some and/or adding others, in this case ending up with latitude, longitude, accuracy, altitude accuracy and time. Context data collection can be performed in three ways. First, there is an automatic mechanism that, with the permission of the user, can *intercept RPCs* made by *webinos-enabled* applications to the various *webinos* APIs. Second, Context Objects can be registered for *periodic background data collection* when the PZP is running and third, they can be d*efined and stored independently by any application*. Apart from the Context Objects already defined for the *webinos* APIs in the *webinos API Context Vocabulary*, a list of structures and rules for the automatic contextualization of intercepted RPC messages, an application developer can make use of the *webinos Application Context Vocabulary* to define custom rules and structures for storing application-specific Context Objects, or ones that are derived by any process or combination of preexisting or new contextual data. The database where these objects are stored securely is located at the user's PZH. The *Context DB* contains data from across the devices and applications in a PZA and each database is unique for that PZA. Querying the Context DB is achieved through a simple to use dedicated query builder that allows the treatment of the Context DB as an Object-Oriented Database, focusing in its main construct, the Context Object. The developer can perform queries directly to the Context API, with the prospect of acquiring any type of Context Objects, created by any API, any application and any device across the user's PZA.

4.3 Contextual Information and Privacy in *webinos*

It is very clear that users are not very good at understanding the future value of keeping personal information private [6] and are often quick to share the ownership of such information without evaluating the impact of its possible uses. With this in mind, the *webinos* platform ensures that the ownership of contextual information stays with the user, while access rights to applications to store, extract or query context data can be given by the user to the application and not the application developer. This allows the developer to build applications that can utilize Context Objects that are stored by the *webinos* platform or other applications in the same PZA. In order to further secure the privacy of personal data, all transactions with the Context DB are monitored by the *webinos* policy

manager and specific access rights to read/write, to and from the Context DB are provided *per application* and *per type* and *per source* of Context Objects.

5 Conclusions

It is becoming increasingly apparent that in order to open the gates of the Future Internet era, applications will be required to be mobile, secure, platform independent and context-aware. Practical and unrestricted ubiquitous computing is gradually leaving the theoretical sphere and is becoming a tangible requirement from end-users. Application developers that need to address these requirements do not have the resources to provide cross-platform implementations of their software or even different builds per Operating System and/or device. Handling cross-device communications, security, privacy, hardware changes, Operating System updates and advanced features, such as context-awareness is next to impossible in most cases. The innovative, open-source and holistic solution that is introduced by *webinos* is a significant step towards real ubiquity of computing, where the software developer will not have to worry about any of these issues, but rather focus on what is really important in developing unique and useful applications: his own ingenuity.

Acknowledgments. The research included in this work has been co-founded by the European Union Seventh Framework Programme (FP7-ICT-2009-5 Objective 1.2) [17].

The researchers that with their work have made this paper possible and deserve a special mention are: Riccardo Scopigno (scopigno@ismb.it), Michele Morello (morello@ismb.it), Nadir Raimondo (raimondo@ismb.it), Enrico Baccaglini (bac-caglini@ismb.it), Andreas Botsikas (abot@epu.ntua.gr) and the rest of the *webinos* team.

References

1. Bernstein, P.A.: Middleware: a model for distributed system services. Commun. ACM 39(2), 86–98 (1996)
2. Dawkins, R.: The Selfish Gene. Oxford paperbacks. Oxford University Press (2006)
3. Greenfield, A.: Everyware: The Dawning Age of Ubiquitous Computing. Peachpit Press, Berkeley (2006)
4. JSON-RPC Working Group. Json-rpc 2.0 specification, `http://www.jsonrpc.org/specification`
5. Issarny, V., Caporuscio, M., Georgantas, N.: A perspective on the future of middleware-based software engineering. In: 2007 Future of Software Engineering, FOSE 2007, pp. 244–258. IEEE Computer Society, Washington, DC (2007)
6. Jedrzejczyk, L., Price, B.A., Bandara, A.K., Nuseibeh, B.: On the impact of real-time feedback on users' behaviour in mobile location-sharing applications. In: Proceedings of the Sixth Symposium on Usable Privacy and Security, SOUPS 2010, pp. 14:1–14:12. ACM, New York (2010)

7. Lyle, J., Monteleone, S., Faily, S., Patti, D., Ricciato, F.: Scross-platform access control for mobile web applications. In: IEEE International Symposium on Policies for Distributed Systems & Networks (2012)
8. Lyle, J., Faily, S., Fléchais, I., Paul, A., Göker, A., Myrhaug, H., Desruelle, H., Martin, A.: On the Design and Development of *webinos*: A Distributed Mobile Application Middleware. In: Göschka, K.M., Haridi, S. (eds.) DAIS 2012. LNCS, vol. 7272, pp. 140–147. Springer, Heidelberg (2012)
9. Open Mobile Terminal Platform organisation. Bondi website, `http://bondi.omtp.org/`
10. PhoneGap. Phonegap website, `http://www.phonegap.com`
11. Recordon, D., Reed, D.: Openid 2.0: a platform for user-centric identity management. In: Proceedings of the Second ACM Workshop on Digital Identity Management, DIM 2006, pp. 11–16. ACM, New York (2006)
12. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: First Workshop on Mobile Computing Systems and Applications, WMCSA 1994, pp. 85–90 (December 1994)
13. Taivalsaari, A., Mikkonen, T.: The web as an application platform: The saga continues. In: 2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), August 30-September 2, pp. 170–174 (2011)
14. Tanner, J.: Cellcos get wac on os fragmentation (March 2011)
15. The W3C. Mobile Web Application Best Practices. Technical report (2010)
16. The W3C. Widget Packaging and XML Configuration. Technical report (2011)
17. The webinos Consortium. The webinos consortium website (2012), `http://webinos.org`