Development Challenges in Web Apps for Public Displays

Constantin Taivan[1], José Miguel Andrade[1], Rui José[1], Bruno Silva[1], Hélder Pinto[1], and António Nestor Ribeiro[2]

[1] Centro Algoritmi, Universidade do Minho, Portugal
{constantin,jma,rui,bruno.silva,helder}@dsi.uminho.pt
[2] Department of Informatics, University of Minho & HASLab/INESC TEC, Braga Portugal
anr@di.uminho.pt

Abstract. Digital public displays can have a key role in urban ubiquitous computing infrastructures, but they have not yet managed to fill this role. A key step in that direction would be the emergence of an application model for open display networks that would enable anyone to create applications for display infrastructures. In this work, we study the development of web-based applications for public displays. We report on our experience of application development for real world public deployment and also on an experiment with external web developers to assess their ability to create such applications using our own development tools. The results show that the web-based app model can effectively be used in the context of public displays and that web developers are able to leverage upon their expertise to create this type of applications.

Keywords: public displays, ubiquitous computing, applications.

1 Introduction

Pervasive computing technology is increasingly present in urban spaces aiming to provide new types of support for our everyday tasks [1]. Public displays, in particular, have the properties to become an important infrastructural element for Intelligent Urban Environments. Digital public displays have always been part of the ubiquitous computing vision. They were called the "boards" in the original Weiser's work [2], where they were essentially seen as interactive, yard-size displays that would complement pads and tabs to enhance spaces by enabling a broad range of casual information exchanges. However, public displays have not yet managed to fill this role. Existing display systems are designed around a content management model in which content is first orchestrated at a central location and then distributed to the displays. They are essentially seen as mere end-points for content distribution or devices that are isolated from the people and the data around them.

The recent trend towards open networks of public displays [3] challenges many of those assumptions and creates the opportunity for any third parties to create and publish content in the form of applications, promoting openness as a source of value for all the parties involved. Together with mobile phones, wireless access points and associated sensors, public displays can become an appropriate execution environment

G. Urzaiz et al. (Eds.): UCAmI 2013, LNCS 8276, pp. 135–142, 2013.

for new and innovative applications. The global nature of pervasive display applications and their strong portability requirements match very well with the core values of web technologies, such as openness, wide-spread availability and high portability [5]. Web technologies not only have the advantage of being easily available across multiple platforms and operating systems, they can also make applications much easier to deploy in large scale and they benefit from a large developers base.

In this paper, we present our experience with a web-based model for display apps. We have created and publicly deployed multiple applications and we have conducted an experiment with external web developers to assess their ability to create such applications using our own development tools. The results show that the web-based app model can effectively be used in the context of public displays and that web developers are able to leverage upon their expertise to create this type of applications.

2 Related Work

There are some obvious similarities between the concept of display apps and other popular app models such as the mobile applications market, e.g., Apple Store or Google' Play [6]. Clinch et al. [16] present a set of design considerations for app stores for public display applications. Conceiving such application stores faces specific challenges when compared with the mobile application market, such as dealing with multiple stakeholders, new business models and scheduling requirements. While in mobile devices market there are well established platforms for running the applications (Android, iOS), in public displays there is no uniform application model for crafting display applications.

Multiple public display infrastructures have experimented with web applications as a way to share content from various sources and prompt user interaction. A remarkable example is UBI hotspots – the real world displays deployment from city of Oulu, Finland [4][7]. Oulu's multi-application public displays based its design on the Web paradigm and enable content contribution from multiple third parties through the publication of services. A service that is associated with a public display and resides anywhere on the Internet may present content using a simple URL. A UBI hotspot can employ interactive services by interacting with users' mobile phones or by distributing its interface to nearby mobile devices. e-Campus public display infrastructure from Lancaster University [8] is another relevant example for using web applications as means to personalize user experience in front of large displays [9]. Based on a mobile Android application users can locate the nearby displays and configure what content to see as part of the associated display web applications. Instant Places from University of Minho [10] is yet another display infrastructure based on web technologies that aggregates place-based screen media and explores new concepts for user-generated content. Many diverse display applications have been proposed for many different domains, e.g. maps, bus schedules, galleries to interactive content generation [11][12]. The research on display prototypes was mostly conducted as part of single display and single application installations (e.g. [13][14]) and lately as key enabler for large scale of open display networks [3]. The commercial sector is mainly

concentrating on broadcasting static content though there is a growing market on personalized and interactive content enabled by third party applications [15].

While the research community is already working on applications for public displays, this study is the first to specifically consider the extension of web development practices and expertise to support that type of development.

3 Web Applications for Public Displays

For the purpose of this work, a display app is a web application whose primary goal is to render content on a public display. A display application encapsulates both the content and the means to render that content on the screens. They are based on web technologies and standards, e.g., HTML, JavaScript and CSS. They are deployed on public servers from where they can be used in any public display. Display apps run in any standard web browser or other types of specially tailored web stacks and their model is optimized for the distinctive execution context and user experience of public displays.

This approach is similar to the well-known iPhone, Android or SmartTV web app models that specify how an app should be designed and implemented for a specific platform. In our case, the key property is a rich client model in which the core of the application is running on the display node. Each application will have its own JavaScript code to handle on the display side issues such as obtaining and managing the content items that the application will need, cache and pre-fetch policies, visual adaptation and network disconnections. To support interaction, applications may include a mobile contact point that allows mobile applications to expose their interactive features to users.

While we are not claiming this to be the only possible model for web apps, this is a model that has evolved over the years with our ongoing research in this topic, and we have now developed and deployed multiple display apps based on these principles [10], like the ones represented in Fig. 1. The Posters app shows posters shared by local community. The Football pins app presents content that reflects the football preferences of the users around the display.

(a) Posters (b) Footbal Pins apps

Fig. 1. Instant Places web applications

3.1 Development Process and Tools

To facilitate development of display web apps we created a developers web site with key information on how to develop these apps and also with the following set of development tools: Application Generator, Instant Places library and Media Simulator.

The Application Generator provides developers with the possibility to generate a ready-made application structure. This considerably reduces the initial development effort and it promotes the use of patterns and components that are known to work better with this type of application. This was achieved by the generation of a Hello World display app, which constituted the skeleton for the creation of other apps.

The Instant Places library provides an abstraction layer for the Instant Places service that enables applications to integrate dynamic data into their content, more specifically place-based information about their surrounding settings, i.e. sensing and interaction information associated with displays (see [10] for a detailed description).

The Media Simulator allows display apps to be tested in their target execution environment, i.e., display nodes' web browsers. Instead of deploying applications to the real display infrastructure, developers have the ability to use this tool to check in advance if a display app is ready to be shown on a public display. Based on a set of guiding reference tests, e.g., resizing the window of the application, unplug the network cable, a developer could observe the behavior of the app.

In addition to these tools, we also provided developers with a few additional guidelines on how to handle key issues such as network disconnection and visual adaptation. Building a fault-tolerant app is essential to public display environments, because we do not have an end-user that is ready to solve the problem. We included a set of code blocks for the cases when no data was fetched or it took too much time to show up, e.g., splash screens routines for masking application startup delays or show something to its audience while external data is being fetched. For example, the Hello World application generated by the Application Generator already included a splash screen hiding the error of no connectivity. To handle the diverse resolutions and orientations that public displays can have, there is a need to employ at least some basic techniques for making the application content look good and – especially – readable. Our initial Hello World app already included a technique based on CSS media queries. Is allows developers to add expressions to media type to check for certain conditions and apply different style sheets. For example, one can have one style sheet for large displays and a different style sheet specifically for mobile devices. The technique is really helpful because it allows adjusting to different resolutions and devices without changing the content. The condition that is often verified to trigger the changes is the viewport width. When the viewport is too narrow, applications can adjust the font and some box sizes.

4 Experience with Third-Party Developers

In order to consolidate our view on creating display web apps, we conducted a short term development experiment in which we investigated the learnability and usefulness of our development tools by other programmers. The assessment of our

development tools was achieved by adopting the same evaluation method as [17] and [18], that is, *informal and controlled laboratory evaluation*. We invited five participants to create a given display web app by using our guidelines and tools and interviewed them about their experiences. All of them had basic web development skills, e.g., JavaScript, HTML and CSS, and had never built a display web app.

A week before the experiment, we sent participants the URL of the development web site so that they could learn the basics of the process. At the beginning of the experiment we gave them a brief tutorial of about 10 minutes in which we introduced the concept of display apps and explained the APIs. They were then asked to build a new display app, i.e., a poster grid app, based on the Hello World example. To do this, we formulated three development tasks that led developers to create the given app. The first task was to put the Hello World app running and test its execution. For this, they needed to install the App Generator and output the necessary application example and Media Simulator for being able to test it. The second task was to use the Instant Places library for getting place related data, such as the *place name*, *place image* and *posters*. Finally, participants were asked to show the posters in a grid by using some CSS rules. In this step, developers needed to use splash screens and configure them to last for at least 3 seconds; support fault tolerance functionality (lack of data, lack of connectivity); prepare the app to be displayed correctly in an iPad or in another device of similar dimensions and test the application using a desktop web browser and Media Simulator tool. Throughout the experiment, participants were encouraged to raise questions and they had four hours to complete all the tasks. At the end, each of them was interviewed about their experiences with our display apps development tools. The interviews were audio recorded and the code produced by developers was kept for subsequent analysis.

4.1 Results and Discussions

The overall view of this experiment was positive, even for the less skilled developers. All the participants have achieved the key development goals without wasting too much time in writing the code. They also preferred to use our toolkit rather than starting to build the application from scratch.

Participants had some initial effort to grasp the specific concepts associated with displays apps, but after that they were quickly able to master the process. Developers could easily follow the documentation provided by our development web site. Even though this was optional, all participants used the Hello World app generated by the App Generator tool as a template to start implementing the new display app. The participants didn't even think very much about the structure of the application, so we may conclude the Application Generator was effective. When we asked developers how it would be to develop without this tool all of them responded that would it be difficult or even very difficult.

Developers had enthusiasm for this experiment despite their weaker experience with some the required web technologies. This is demonstrated by the fact that all of them succeeded in applying their web development skills to develop a display app. However, some of them experienced some difficulties in understanding and using all

specific development and testing scenarios, e.g., implementing the splash screens or providing the required code blocks for a fault tolerant display app. Only one of them could entirely test the app execution behavior.

Due to the fact that our display app was not too complex, it just required a set of API requests, the code source is quite identic among the participants and the final applications share the same structure and very similar lines of code. Having a previously scaffold app structure proved to be comfortable to the participants and reduced the amount of code they had to write. Developers ended up not writing much code and not changing the application structure at all. Instead their effort was mostly applied into answering questions like "What do I need to change from this app?" instead of "What do I need to build my app?". One student noted that the integration of our code blocks was straightforward, while making various customizations of the provided code fragments was not so easy.

Using the Instant Places API library was something that proved to be very handy. Although there were some initial problems in understanding the meaning of our API and the related code blocks, after getting the place name, they easily succeeded to get further data, such as posters.

Developers had difficulties when testing their apps because they weren't familiar with any tools to accomplish this task, e.g. Fiddler. Most tests were made using a common web browser while the Media Simulator tool was just periodically used to rule out eventual errors related to the different web engine of display players. Only one student did not test at all the new application execution, neither in desktop web browser nor in Media Simulator tool. The others tested the application but encountered various difficulties.

Participants were really motivated by the innovative field of usage of display web apps and recognized their big potential when deployed in real world settings. They associated display apps with mechanisms to publish content, such as replacing the traditional paper based posters with digital forms of content. In their final comments, they all referred particular features for display web apps, e.g., a display app should provide content that is dynamic, personalized and place-based. In Table 1 we provide a short description about the development challenges that we addressed in this work.

Table 1. Summary of the develoment challenges

Development Challenges	**Techniques employed**	**Developers' opinions**
Content management policies	Rich client model includes logic to cache and pre-fetch content.	Instant Places API library helps with high-level application concepts.
Fault tolerance support	Code blocks that can be configured for specific purposes, e.g., splash screens	Code blocks are easy to integrate but not to customize ; Fault-tolerance testing scenarios were complex
Rich visual adaptation	CSS media queries	No special issues were mentioned. Most participants did not test this feature.

5 Conclusions

In this paper we have presented our early experiences in developing display web applications. In order to guide development and speed up the process of creating display application we implemented a collection of tools, which were evaluated based on an experiment with developers. The experiment showed that there is a short learning curve even for first-time developers and building a display web app is quite an appealing challenge for web skilled persons. The participants were able to put in practice their web programing experience though this was not straight-forward. The specific field of public displays put forward a set of specificities that requires clear development specifications and guidelines. These have a strong effect in leveraging on the expertise of web developers to create display apps and subsequently attracting an increasing number of people with appropriate web development skills. Based on these insights we are looking forward to collect long-term development feedback from more web experienced developers and to extend the specification of our model for display web applications.

Acknowledgment. The authors acknowledge the financial support of the Future and Emerging Technologies (FET) programme within the 7th Framework Programme for Research of the European Commission, under FET-Open grant number: 244011 and "Fundação para a Ciência e a Tecnologia", under the research grant SFRH/BD/75868/2011.

References

1. Greenfield, A.: Everyware: The Dawning Age of Ubiquitous Computing, p. 272. New Riders Publishing (2006)
2. Weiser, M.: The Computer for the 21st Century. Scientific Am., 94–104 (1991)
3. Davies, N., Langheinrich, M., José, R., Schmidt, A.: Open Display Networks: A Communications Medium for the 21st Century. IEEE Computer, 58–64 (May 2012)
4. Ojala, T., Kostakos, V., Kukka, H., Heikkinen, T., Linden, T., Jurmu, M., Hosio, S., Kruger, F., Zanni, D.: Multipurpose Interactive Public Displays in the Wild: Three Years Later. Computer 45(5), 42–49 (2012)
5. Pawan, V.: Web Applications Design Patterns, p. 448. Morgan Kaufmann (2009)
6. Holzer, A., Ondrus, J.: Mobile application market: A developer's perspective. Telematics and Informatics 28(1), 22–31 (2011)
7. Lindén, T., Heikkinen, T., Kostakos, V., Ferreira, D., Ojala, T.: Towards multi-application public interactive displays. In: Proceedings of the 2012 International Symposium on Pervasive Displays, PerDis 2012, pp. 1–5 (2012)
8. Friday, A., Davies, N., Efstratiou, C.: Reflections on Long-Term Experiments with Public Displays. Computer 45(5), 34–41 (2012)
9. Kubitza, T., Clinch, S., Davies, N., Langheinrich, M.: Using mobile devices to personalize pervasive displays. ACM SIGMOBILE Mobile Computing and Communications Review 16(4), 26–27 (2012)

10. José, R., Pinto, H., Silva, B., Melro, A., Rodrigues, H.: Beyond interaction: tools and practices for situated publication in display networks. In: Proceedings of the 2012 International Symposium on Pervasive Displays, PerDis 2012, pp. 1–6 (2012)
11. Memarovic, N., Elhart, I., Langheinrich, M.: FunSquare. In: Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia, MUM 2011, pp. 175–184 (2011)
12. Alt, F., Kubitza, T., Bial, D., Zaidan, F., Ortel, M., Zurmaar, B., Lewen, T., Shirazi, A.S., Schmidt, A.: Digifieds: insights into deploying digital public notice areas in the wild. In: Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia, MUM 2011, pp. 165–174 (2011)
13. Huang, E.M., Mynatt, E.D.: Semi-public displays for small, co-located groups. In: Proceedings of the Conference on Human Factors in Computing Systems, CHI 2003, vol. 5(5), p. 49 (2003)
14. Izadi, S., Brignull, H., Rodden, T., Rogers, Y., Underwood, M.: Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media. In: Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology, UIST 2003, pp. 159–168 (2003)
15. Locamoda, Fifteen Seconds or More. Engaging Audiences With Place-Based Social Media, In: White Paper (2010)
16. Clinch, S., Davies, N., Kubitza, T., Schmidt, A.: Designing application stores for public display networks. In: Proceedings of the 2012 International Symposium on Pervasive Displays, PerDis 2012, pp. 1–6 (2012)
17. Klemmer, S.R., Li, J., Lin, J., Landay, J.A.: "Papier-Mâché: Toolkit Support for Tangible Input," in. In: Proceedings of the 2004 Conference on Human Factors in Computing Systems, CHI 2004, pp. 399–406 (2004)
18. Heer, J., Card, S.K., Landay, J.A.: prefuse: a toolkit for interactive information visualization. In: Proceedings of the SIGCHI Conference on Human Factors in Computing System, CHI 2005, p. 421 (2005)