Towards Augmenting Legacy Websites with Context-Awareness

Darren Carlson[1(✉)] and Lukas Ruge[2]

[1] Felicitous Computing Institute, National University of Singapore, Singapore, Singapore
carlson@comp.nus.edu.sg
[2] Ambient Computing Group, University of Luebeck, Luebeck, Germany
ruge@itm.uni-luebeck.de

Abstract. Emerging context frameworks enable Websites to interact with the Internet of Things directly from the browser; however, Websites must be specifically designed to utilize such context framework support. As such, the majority of "legacy" Websites remains context-unaware. This paper presents an open approach for dynamically injecting context-awareness capabilities into legacy Websites on-demand, without requiring browser extensions, proxies or Website reengineering. Towards this end, we developed an extensible Bookmarklet framework that serves as a conduit between the user's browser and a server-side repository of enhancement plug-ins, which can used to dynamically augment any 3rd party Website with new content, adapted behavior and context framework support.

Keywords: Ubiquitous computing · Internet of things · Web of things

The rapid evolution of the Internet of Things (IoT) has outpaced the Web's ability to take advantage of the rich contextual information, smart devices and situated services that are now available in many everyday environments. However, emerging mobile context middleware, such as Ambient Dynamix [1], now enable Websites to interact with the IoT directly from the browser. Dynamix is a community-based approach for context-aware computing in which advanced sensing and acting capabilities are deployed *on-demand* to mobile devices as plug-ins. Dynamix runs as lightweight background service on the user's mobile device, leveraging the device itself as a sensing, processing and communications platform. Applications can request context support from a local Dynamix instance using simple application programming interfaces (APIs). Dynamix automatically discovers, downloads and installs the plug-ins needed for a given sensing or acting task. When the user changes environments, new or updated plug-ins can be deployed to the device at runtime, without the need to restart the application or framework.

Dynamix supports two principal app types: Android apps and Web apps. Android apps are defined as native applications that communicate with a local Dynamix service using the Android Interface Definition Language (AIDL). Web apps are hosted by native Web browsers, such a Google Chrome and Firefox. To support communication with browser-based Web apps, Dynamix exposes two local REST-based APIs using a customized Web server that is embedded within the Dynamix Service. Web apps communicate with

© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2014
I. Stojmenovic et al. (Eds.): MOBIQUITOUS 2013, LNICST 131, pp. 694–698, 2014.
DOI: 10.1007/978-3-319-11569-6_55

Dynamix via Ajax, using two provided JavaScript support files that simplify service binding, API interactions, data serialization/deserialization, error handling and event processing (see [2]). An overview of the Dynamix architecture is shown in Fig. 1.

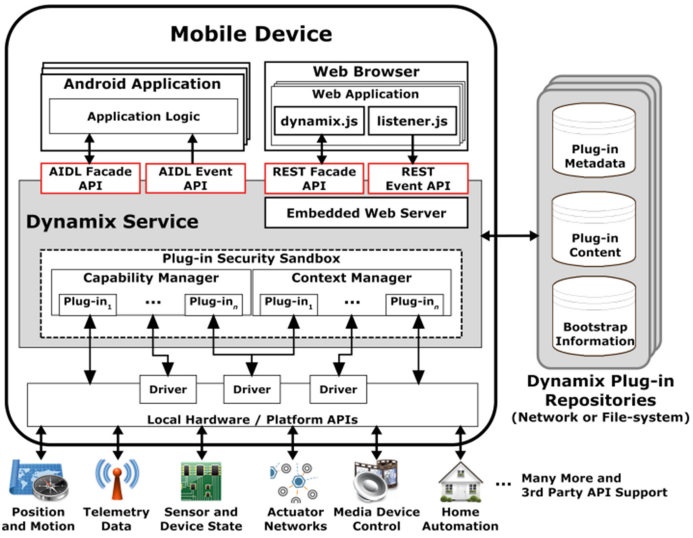

Fig. 1. Overview of the Dynamix architecture

As Websites must be specifically engineered to leverage Dynamix capabilities (e.g., they must reference and utilize the Dynamix JavaScript libraries), we investigated *augmented browsing* as a method of injecting Dynamix-based context support into 3rd party legacy Websites on-demand. Augmented browsing refers to the on-the-fly modification of web content for the purpose of customizing the browsing experience to the user. As modifications are performed outside of the origin server, Website reengineering is not required to introduce new functionality. Example augmentations include user interface (UI) changes (e.g., removing advertisements); adding supplemental information (e.g., keyword definitions); and providing new JavaScript libraries (e.g., jQuery). Common augmented browsing approaches include browser extensions, proxies, site-specific-browsers, and Bookmarklets. Of these, browser extensions are currently the most popular approach, as illustrated by the userscripts.org website, which hosts tens of thousands of augmentation scripts. Browser extensions are currently available for most desktop browsers; however, extension support is extremely limited on mobile devices at present.

Only a few projects have investigated *context-aware* Website augmentation. Most current approaches focus on page-specific context-aware adaptation, such as adding faceted browsing and novel data visualizations [3]; adding supplemental page information (e.g., providing pop-up annotations for scientific keywords) [4, 5]; customizing accessibility options to assist disabled users [6, 7]; and enabling website personalization based on sensor data provided by the browser (e.g., presenting nearby theaters showing movies mentioned in a blog post) [8]. Other approaches, such as the COIN

(COntext INjection) framework [9], automatically correlate a Web page's content with the mobile user's context – injecting context-sensitive cues that highlight relevant content (e.g., monuments passed, nearby buildings), and/or enrich page content with contextually relevant information (e.g., supplemental description).

Unlike browser extensions, *Bookmarklets* represent a viable approach for enabling augmented browsing on most current mobile browsers. Bookmarklets are JavaScript commands that are stored within the Uniform Resource Identifier (URI) of a standard bookmark. A Bookmarklet utilizes the `javascript:` scheme, which executes the scripts embedded in the URI against the Document Object Model (DOM) of the current page. As such, the script has unrestricted access to the target DOM, which it may inspect and change on-the-fly. Bookmarklets can also load additional scripts from a remote server, enabling the injection of arbitrarily complex functionality.

Using the Bookmarklet concept as a foundation, we created an open approach for dynamically injecting context-awareness capabilities into legacy Websites on-demand. Our approach, called Ambient Amp (Amp), consists of a browser-based Bookmarklet framework and a server-based repository of Amp plug-ins, which consist of JavaScripts and optional CSS styles. The Amp Bookmarklet serves as a conduit between the user's browser and the Amp repository, providing users access to page or domain specific Amp plug-ins that can be injected into the browser. The Bookmarklet framework provides plug-in discovery, installation and lifecycle services (e.g., initialization logic) along with an optional "canvas area" where each plug-in can display UI elements if needed. The Bookmarklet also injects the Dynamix JavaScript libraries into the current DOM, which enables Amp plug-ins to communicate with a local Dynamix instance. Amp augmented Websites can access the growing number of Dynamix plug-ins,[1] which provide access to rich context sensing (e.g., sleep state, heart-rate, activity recognition); home automation control (e.g., network light control); media device management (e.g., Universal Plug-and Play and AirPlay); and much more. The basic Amp workflow is shown in Fig. 2.

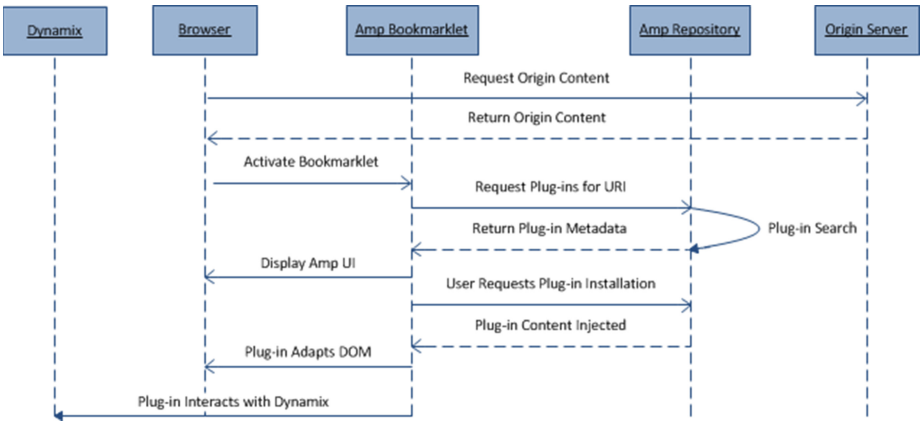

Fig. 2. The Basic Amp workflow

[1] http://ambientdynamix.org

The Amp Bookmarklet is installed into the user's browser using conventional methods (e.g., using Chrome's bookmark sync feature), where it can be activated through the bookmark menu or address bar. Once activated, the Amp Bookmarklet is displayed over the current Website as an unobtrusive menu tab, which remains in a fixed position on the right or left side of the browser window, as shown in Fig. 3. After rendering its UI, the Bookmarklet then contacts the remote Amp repository server using the address of the current page as a query term (see Fig. 2). The repository uses URI pattern matching to determine if any Amp plug-ins have been registered for the page or domain that the user is interacting with. URI-relevant plug-in metadata are returned to the Bookmarklet, where they are cached locally and shown to the user on request.

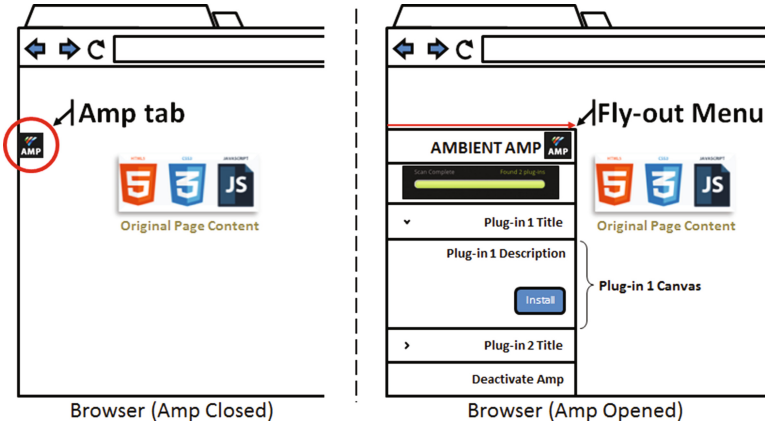

Fig. 3. The injected Amp tab (left) and Amp fly-out menu (right)

When the user taps the Amp tab, a fly-out menu opens to present the list of plug-ins that were discovered for the current page or domain. To maintain a low memory footprint, Amp plug-ins are not initially loaded; rather, only the title and description of each discovered plug-in are shown to the user, along with a button that can be used to install it. Tapping an "Install" button in the fly-out menu automatically loads the associated plug-in's JavaScript code and CSS styles into the browser. The Amp Bookmarklet monitors each plug-in's installation status and provides coordinated initialization once a plug-in's scripts and content are loaded. Initialization enables a plug-in to setup its internal state, display UI elements (if needed), bind to the local Dynamix framework, and activate its Web augmentation features. Once installed, an Amp plug-in can provide 3rd party Websites with new context-awareness features.

To our knowledge, Amp is the first Web augmentation framework to enable the injection of arbitrary context-awareness enhancements into any 3rd party Website (through Amp plug-ins); allow the installation of new (and updated) context sensing and actuation capabilities on-demand (through Dynamix); and support deployment into most current mobile browsers (through its Bookmarklet-based architecture). To validate our approach, we developed a fully operational prototype of the Amp Bookmarklet and repository, along with an example Amp plug-in called Stream to Screen (S2S),

which dynamically augments the Flickr.com Website with the ability to stream photos to networked media devices discovered in the user's physical environment (to be described in an upcoming paper). Our initial experiments indicate that Amp can be rapidly injected into many 3rd party Websites and is interoperable with most Android-based mobile browsers, including Chrome, Firefox Mobile, Dolphin HD, and others.

We are focusing on several areas of future work. First, we are investigating secure methods of opening the Amp plug-in repository to external developers. By allowing 3rd party plug-in contributions, we aim to facilitate the emergence of a vibrant Amp developer community. We are also investigating the processing and memory overhead imposed by the Amp Bookmarklet on several commodity mobile devices. Finally, we are developing an Amp plug-in that uses multiple Dynamix plug-ins simultaneously to embed data from a popular social media Website into the user's physical environment, e.g., displaying newsfeed data on nearby ambient devices; providing physical alerts for site updates (e.g., light color changes); posting life-logging data gathered from external sensors; and providing voice control for various site features.

References

1. Carlson, D., Schrader, A.: Dynamix: an open plug-and-play context framework for android. In: Proceedings of International Conference on the Internet of Things (IoT2012) (2012)
2. Carlson, D., Altakrouri, B., Schrader, A.: AmbientWeb: bridging the web's cyber-physical gap. In: Proceedings of International Conference on the Internet of Things (IoT2012) (2012)
3. Huynh, D.F., Miller, R.C., Karger, D.R.: Enabling web browsers to augment web sites' filtering and sorting functionalities. In: Proceedings of 19th Annual ACM Symposium on User Interface Software and Technology, pp. 125–134. ACM (2006)
4. Chahinian, V., et al.: Proxiris, an augmented browsing tool for literature curation. In: Proceedings of 9th International Conference on Data Integration in the Life Sciences, Springer (2013)
5. Cruz, T., Nabuco, O.: Clinical eye: a tool for augmented browsing in the health domain. In: Proceedings of 2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 432–437 (2012)
6. Bigham, J.P., Ladner, R.E.: Accessmonkey: a collaborative scripting framework for web users and developers. In: Proceedings of the 2007 International Cross-Disciplinary Conference on Web Accessibility (W4A), pp. 25–34. ACM (2007)
7. Mirri, S., Salomoni, P., Prandi, C.: Augment browsing and standard profiling for enhancing web accessibility. In: Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, pp. 1–10. ACM (2011)
8. Ankolekar, A., Vrandečić, D.: Kalpana - enabling client-side web personalization. In: Proceedings of the Nineteenth ACM Conference on Hypertext and Hypermedia, pp. 21–26. ACM (2008)
9. Casteleyn, S., Woensel, W.V., Troyer, O.D.: Assisting mobile web users: client-side injection of context-sensitive cues into websites. In: Proceedings of the 12th International Conference on Information Integration and Web-Based Applications and Services, pp. 443–450. ACM (2010)