Leveraging the Web of Things for Rapid Prototyping of UbiComp Applications

Benedikt Ostermaier
ETH Zurich
ostermaier@inf.ethz.ch

Fabian Schlup
ETH Zurich
fschlup@alumni.ethz.ch

Matthias Kovatsch
ETH Zurich
kovatsch@inf.ethz.ch

ABSTRACT
An increasing number of real-world entities is currently being connected to the Internet and the World Wide Web. We argue that this development is the precursor of a *Web of Things (WoT)*, which in turn provides a promising way to prototype UbiComp applications, by significantly lowering the technical barriers for making things "smart". In this paper, we outline how sensors, actuators and Web services can easily be combined in the WoT in order to enable rapid prototyping of UbiComp applications.

Author Keywords Web of Things, Rapid Prototyping, Physical Mash-Ups

ACM Classification Keywords H.3.5 [Information Storage and Retrieval]: Online Information Services – Web-based services, D.2.m [Software Engineering]: Miscellaneous – Rapid prototyping

General Terms Design, Experimentation

INTRODUCTION
An increasing number of real-world entities is currently being connected to the Internet and the World Wide Web, providing real-time access to a potentially huge audience. One example of this is Bicing (http://www.bicing.com), a public bicycle-sharing system in Barcelona, Spain, where users can see the number of bicycles available at each rental station in real-time on the Web. We believe that this development is a precursor of a *Web of Things*, which gives real-world objects and places a Web presence that not only contains an HTML representation but also provides Web APIs for accessing and manipulating their states in real-time. We argue that the Web of Things fosters the creation of UbiComp applications by significantly lowering the technical barriers to make things "smart": Context required by an application could easily be gathered from the Web, the large set of Web-based services could be utilized, and Web concepts and standards are widely known and supported.

THE WEB OF THINGS
In the emerging *Web of Things (WoT)*, the Web is extended from its original document-centric design to an application layer for the real world. In this yet-to-be-standardized concept, real-world objects like consumer devices are integrated into the WWW by representing them as Web resources, which can be accessed over HTTP using lightweight APIs based on the REST principle [1,4]. For example, assume the following URL represents the status of the kitchen light: *http://example.org/kitchen/light/power*. To determine whether the kitchen light is currently on, one issues a HTTP GET request to this address and would in turn receive a textual representation[1] reading "true" or "false" respectively. To switch off the light, one would send a HTTP POST request with the payload "false" to the resource.

Although the concept of providing real-world entities with a Web presence is well known and has been investigated in the CoolTown [2] project, for example, it has recently gained a revival: Since then, the Web has extended by reaching new devices and user groups and also evolved, providing new services and concepts. Structured data is nowadays more common, various services on the Web provide powerful APIs and combining different information sources on the WWW within an interactive Web page, creating a so-called *mash-up*, is a well-known concept.

In this paper, we demonstrate how one can combine sensors and actuators on the Web of Things, creating *physical mash-ups*, for rapidly prototyping UbiComp applications. We illustrate this using a simple example: Assume you have a Web-enabled light switch and a Web-enabled lamp. Let us further assume that the light switch calls a user-specified URL every time the switch is flipped, using a HTTP POST request with the current state of the switch in the request body[2]. The callback target of the switch can found at *http://example.org/lightswitch/callback* and the lamp's current power state can be read and set at *http://example.org/lamp/power*. In order to link the light switch and the lamp, one only needs to set the callback target of the switch to the power state of the lamp. This is performed by issuing a HTTP POST request to *http://example.org/lightswitch/callback* with the message body "*http://example.org/lamp/power"*. Now, each time the light switch is flipped, it issues a HTTP POST request to *http://example.org/lamp/power* with the current state of the switch in the message body, causing the lamp to update its power state. Note the flexibility of this approach: It is merely

[1] *However, this concept does not depend on a single serialization format, as there may actually exist multiple representations for a given resource, such as HTML and JSON, for example. The returned representation is determined by HTTP's content negotiation.*

[2] *URL callbacks are also sometimes denoted as WebHooks.*

UbiComp'10, September 26–29, 2010, Copenhagen, Denmark.
ACM 978-1-4503- 0283-8/10/09.

configuration (not programming), and light switches could control other devices using the same principle, as long as on both sides the data is compatible. Likewise, other sensors or arbitrary services running in the *cloud* could control the state of the lamp. To toggle the lamp using multiple light switches, one would use an intermediary Web service, if the Web API of the lamp does not support this feature. Finally, the interface can easily be utilized by an HTML page rendered within a browser, by a command-line tool such as cURL (*http://curl.haxx.se/*) or by a programming environment. In order to be able to prototype more complex application scenarios, we use a framework called WebPlug [3].

WEBPLUG: A FRAMEWORK FOR THE WEB OF THINGS

WebPlug is a prototypical framework for the Web of Things, which does not rely on a centralized infrastructure but rather introduces several building blocks. These components could be distributed among connected devices but could also be located in the *cloud*. We will briefly sketch some of its key features, for more information we refer to [3].

Each resource managed by WebPlug can be *versioned*, i.e., the framework can keep a history of past versions of that resource. Moreover, each version of a resource is assigned an URL. This is especially helpful for sensor readings, where one is often interested in past sensor data. For example, if *http://example.org/temperature* represents a temperature sensor then *http://example.org/temperature@history/i:1* addresses the first recorded temperature reading while the penultimate reading of a temperature sensor could be found at *http://example.org/temperature@history/i:last-1*.

Furthermore, WebPlug supports a simple publish/subscribe scheme for managed resources: One can *observe* changes of a given resource by registering URL callbacks, which are called by WebPlug with the updated value whenever the corresponding resource changes. For example, to update a display whenever *http://example.org/temperature* changes, one may POST *http://example.org/display/content*, the resource representing the display's contents, to *http://example.org/temperature@observers*. Each time the value of *http://example.org/temperature changes,* WebPlug will issue an HTTP POST request to all registered observers with the updated value in the message body, thus updating the contents of the display, in this example.

An important feature of WebPlug are expressions, which can be applied to managed resources. For example, the *http://example.org/fridge/temperature>10* will return true if *http://example.org/fridge/temperature* is currently above 10°C, false otherwise. It is also possible to subscribe to the result of expressions: In order to sound an alarm when the temperature of the fridge exceeds 10°C, one could register *http://example.org/beeper/power* by POSTing it to *http://example.org/fridge/temperature>10@observers.*

Additionally, there is support for complex evaluations based on multiple input resources. WebPlug also support the integration of existing Web resources by polling them at regular intervals, and turning them into managed resources. Finally, WebPlug supports a variety of representations for managed data. For example, numeric data can be depicted graphically and historic data can be represented as iCalendar data, which can be viewed in many popular calendar applications.

PUTTING THINGS TOGETHER

In order to test and demonstrate the rapid prototyping of UbiComp applications using the Web-of-Things approach, we utilized several wireless, Web-enabled devices: A light switch and a switchable power plug, which both offer a simple REST-based API. Additionally, we used a standard webcam and a mobile phone with a REST interface where one can access the built-in sensors like the accelerometer or the GPS and control the actuators like the vibrating alert, for example. Finally, software components like a virtual display and REST-based Web services providing helper functions were utilized.

While we have tested our system only in setups of limited size, our initial experiences with our approach are promising. Creating and changing a setup requires little effort – for example, creating a motion detector based on a standard webcam and using its results to automatically switch a device connected to the Web-enabled power plug. For this scenario, we turned the webcam into a resource managed by WebPlug by utilizing a component of the framework to poll and store the images at regular intervals. Based on this managed resource, another component of WebPlug was used to automatically invoke an external Web service with the last two images taken by the camera, using the subscription and versioning mechanisms outlined above. The output of this service, a similarity metric for two given images, is again a resource managed by WebPlug. Finally, we subscribed the state of the power plug to a threshold expression for this similarity level, thus switching power whenever there is motion before the webcam.

REFERENCES

1. R.T. Fielding: Architectural Styles and the Design of Network-based Software Architectures. *Doctoral dissertation*, University of California, Irvine, 2000.
2. T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic: People, Places, Things: Web Presence for the Real World. *Mobile Networks and Applications*, vol. 7, no. 5, pp. 365–376, 2002.
3. B. Ostermaier, F. Schlup, K. Römer: WebPlug: A Framework for the Web of Things. *Proc. of the First IEEE International Workshop on the Web of Things (WOT2010)*, Mannheim, Germany, March 2010.
4. D. Guinard, V. Trifa, E. Wilde: Architecting a Mashable Open World Wide Web of Things. *Technical Report No. 663*, Institute for Pervasive Computing, ETH Zurich, February 2010.