The SmartComposition Approach for Creating Environment-Aware Multi-Screen Mashups

Michael Krug, Fabian Wiedemann, Markus Ast, and Martin Gaedke

Technische Universität Chemnitz, Germany
{firstname.lastname}@informatik.tu-chemnitz.de

Abstract. UI mashups enable non-experts to create rich web applications. In this paper, we propose SmartComposition to enable local developers to create environment-aware multi-screen mashups. We facilitate WebComponent technologies to build SmartComponents – the building blocks for mashups. For achieving environment-awareness, our approach integrates functionalities of the Web of Things into mashups, such as controlling actors and accessing sensors. SmartComposition provides mashup composition by external communication configuration through markup. We additionally propose a messaging service utilizing WebSockets for enabling communication across multiple screens.

Keywords: Multi-screen mashup, Web Components, HTML5, Web of Things

1 Introduction

The amount of tools for creating user interface mashups (UI mashups) increased within the last years. UI mashups enable non-experts in creating rich web applications [1]. In UI mashup several components that offer a limited functionality are combined and aggregated to solve complex tasks. While other approaches for creating UI mashups focus on automatic or semi-automatic mashup creation and deployment to desktop as well as mobile screens, our approach eases the creation of UI mashups that run distributed across several screens, so called multi-screen mashups. The area of Web of Things (WoT) connects physical things to the Internet. That is, WoT offers the possibility of controlling devices in the physical world or even accessing remote sensor data [2]. In this domain two main concepts exist: actors and sensors. While actors are defined by doing something physical, such as producing something, sensors are defined by observing the physical world e.g., measures like temperature.

The purpose of SmartComposition is to enable local developers to create multi-screen mashups with environmental awareness. Our approach is based on basic web technologies, such as HTML5 and CSS. Thus, a local developer who is familiar with these technologies does not require advanced knowledge of JavaScript. For achieving a high level of reuse our approach requires loosely coupling and a suitable communication infrastructure to minimize the overhead when integrating them. For enhancing existing web applications to multi-screen

mashups, SmartComposition needs to be easily integrable. While common mashup platforms require deploying and hosting their components in a separate runtime environments, we want to eliminate this requirement and enable usage in any standard HTML5 website or application.

With the emerging field and availability of WoT entities, there is an increase in potential applications of end-user-based composition of functionality. Implementing WoT entities as components, integrating them into mashups and applying inter-component communication to them, yields a huge amount of new possible use cases and eases end-users to compose WoT entities on their own. That is, we want to show how WoT entities can composed using our SmartComposition approach.

The rest of this paper is organized as follows: In Section 2, we present the SmartComposition approach and provide the requested feature checklist in Section 3. Finally, we give an overview of how our live demonstration could look like in Section 4.

2 The SmartComposition Approach

SmartComposition describes how to define independent, encapsulated, configurable and loosely coupled components using standard web technologies as well as their composition to create single- and multi-screen mashups. Our approach leverages a new set of standards called *WebComponents* for defining and implementing *SmartComponents* – the building blocks for SmartComposition mashups. SmartComponents can form UI, data or logic components as well as a combination of those types. They can be used in any HTML5-based web application and do not require a dedicated runtime environment or portal software to be executed. To ease the development of new SmartComponents, we are using *Polymer* as an underlying framework. Polymer offers a comprehensive implementation of the WebComponents standards. It provides a declarative syntax to define new components and supports e.g., event & data binding and advanced template functionalities. Furthermore, it also enables the usage of those technologies in older browsers.

In contrast to our contribution [3] to last year's "Rapid Mashup Challenge", we shift from including communication aspects inside the components to an external composition solution while still focusing on loosely coupling. This enables a better control of the information flow inside the mashup. To achieve this, we design SmartComponents as DOM-Elements that provide their in- and output interfaces through attributes accordingly. We enable mashup composition by linking those attributes through external configuration. We connect the attribute interfaces of SmartComponents using a stand-alone *Attribute-Link* component, which is configured by a *source* and *target* selector and an optional *transformation* function. The *Attribute-Link* component is also implemented as a WebComponent and can be deployed directly in the markup without knowledge of JavaScript. To address a component within the mashup, we use the established *CSS selector* syntax[1]. This syntax is applied to both the source as well as the target selector. Since CSS

[1] Selectors Level 3 `https://www.w3.org/TR/selectors/`

selectors can return multiple elements, we also support multiple components as target and source. The attribute is addressed by its name separated by an `@` sign. To define complex inter-component communication setups, the *Attribute-Link* can be included multiple times in an applications. We use native *DOM mutation observers*[2] to watch for attribute changes without affecting the responsiveness of the application. Additionally, we support data transformation by offering to specify a transformation function within the *Attribute-Link* configuration. If such a function was specified, it will be applied and the resulting value is then propagated to the defined attribute of the according target component.

When implementing WoT entities as components and exposing their inputs and outputs as attributes accordingly, the *Attribute-Link* component can also be used to connect WoT entities with other components together. This is not restricted to connecting WoT entities only among themselves, but they can also be connected with other components representing ordinary web services. Doing so allows for implementing complex workflows among sensors, actors and other web services.

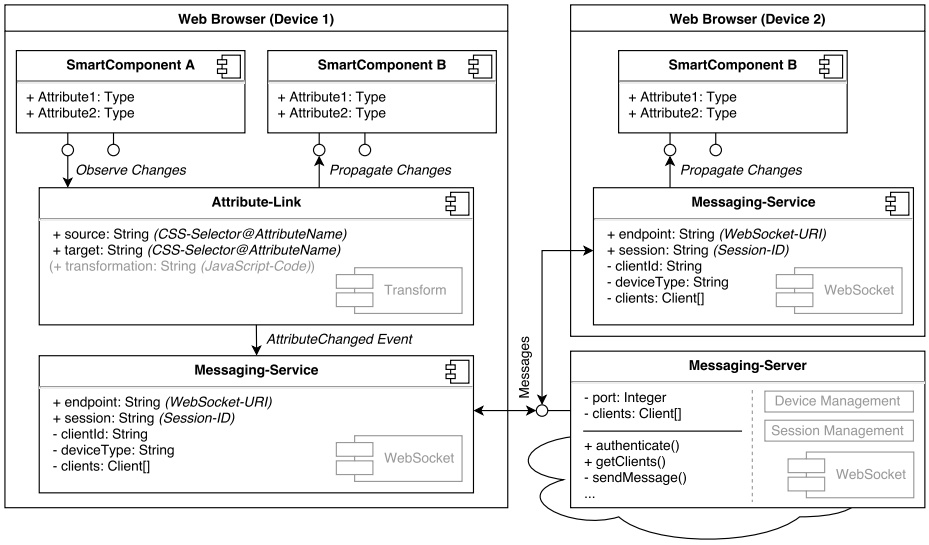

Fig. 1. Components of the SmartComposition Approach

In our approach, we also focus on the creation of distributed or multi-screen mashups. Thus, we also enable the addressing of target components on multiple connected devices (browsers running the mashup application with the same endpoint configured that share the same context). We achieve this by providing a *Messaging-Service* component and a *Messaging-Server*. The *Attribute-Link* component seamlessly integrates with the *Messaging-Service* by placing it in the DOM as a child element. Changed attributes that shall be distributed are signaled by dispatching a custom event. The *Messaging-Service* contacts the *Messaging-Server*, which then distributes the message to other connected devices

[2] DOM Standard `https://dom.spec.whatwg.org/#mutation-observers`

(cf. Figure 1). The inter-device communication is realized using the *WebSocket protocol*[3]. To propagate the received messages to the target components on the remote devices running the mashup, they need to have instantiated an accordingly configured *Messaging-Service* component.

3 Feature Checklist

Mashup Type	Hybrid mashups
Component Types	Data, Logic & UI components
Runtime Location	Both Client and Server
Integration Logic	Choreographed integration
Instantiation Lifecycle	Short-living
Targeted End-User	Local Developers
Automation Degree	Manual
Liveness Level	Level 4
Interaction Technique	Editable Example
Online User Community	None

4 Demonstration Idea

In our live demonstration, we will show the composition of different components, with the focus on showcasing the composition of WoT devices. That is, besides functionality like a video, translation, semantic extraction, Google maps, Twitter and Wikipedia, we will also showcase how a Philips Hue light bulb can be composed into mashups. For the demonstration, besides turning the light bulb on and off, we could connect it to a video stream or to keywords from semantic extraction. This video stream input is then used to change the light bulb's color accordingly. Doing the former would create an environment light that harmonizes with the video content and the latter could be used to notify if certain keywords occur. The overall showcase will be done live in a multi-device environment to demonstrate our multi-screen capabilities accordingly. Corresponding configuration of inter-component communication will be done by connecting the component's interfaces by using the proposed Attribute-Link.

Bibliography

[1] Chudnovskyy, O., Fischer, C., Gaedke, M., Pietschmann, S.: Inter-Widget Communication by Demonstration in User Interface Mashups. In: Web Engineering, LNCS, vol. 7977, pp. 502–505. Springer Berlin Heidelberg (2013)

[2] Kopetz, H.: Internet of Things. In: Real-time Systems, pp. 307–323. Springer (2011)

[3] Krug, M., Wiedemann, F., Gaedke, M.: SmartComposition: Extending Web Applications to Multi-Screen Mashups. In: Rapid Mashup Development Tools: First International Rapid Mashup Challenge, pp. 50–62. Communications in Computer and Information Science, Springer International Publishing (2016)

[3] The WebSocket Protocol `http://tools.ietf.org/html/rfc6455`