SmartComposition: Extending Web Applications to Multi-Screen Mashups

Michael Krug, Fabian Wiedemann, and Martin Gaedke

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

Abstract. The overall objective of UI mashups is to enable non-experts to create rich web applications. While current approaches focus on creating UI mashups running on a single screen, we propose SmartComposition to enable local developers to create multi-screen mashups. For extending existing web applications to multi-screen mashups, SmartComposition facilitates the Web Component technologies to enable the hassle-free integration of new components. We support multiple types of SmartComponents, not limiting them to user interface components. For advanced developers we offer a template to define new kinds of SmartComponents. SmartComposition provides an event-based communication infrastructure which enables inter-component communication as well as message exchange across multiple screens utilizing a WebSocket-based synchronization service.

Keywords: Multi-screen mashup, Web Components, HTML5

1 Introduction

Within the last years, the amount of tools for creating user interface mashups (UI mashups) significantly increased. The overall objective of UI mashups is to enable non-experts to create rich web applications [2]. For solving complex tasks an UI mashup consists of several components that offer a limited functionality and are combined and aggregated. 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 purpose of SmartComposition is to enable local developers to create multi-screen mashups. We assume that a local developer is familiar with basic web technologies, such as HTML5 and CSS [1]. Thus, our approach should be based on these technologies and does not require advanced knowledge of JavaScript. Furthermore, we want to achieve a high level of reuse of the developed components. This 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.

Common mashup platforms require deploying and hosting their components in a separate runtime environments, such as Apache Rave or Apache Shindig. We want to eliminate this requirement and enable usage in any standard HTML5 website or application.

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.

2 Approach

The SmartComposition approach is based on the idea of creating mashups by composing loosely coupled components using standard web technologies. In [3] we proposed a component-based architecture for multi-screen web applications. We advance the presented ideas by using the *Web Components* technologies for defining and implementing SmartComponents. Our implementation uses only client-side JavaScript. The major benefit of exploiting the proposed technologies for creating modern widgets is that no runtime environment or portal software is needed to host such composed mashup applications. That means, SmartComponents can be used in any HTML5 based web application. They can work as UI as well as data or logic components. A combination of all three types is also possible. In contrast to existing UI mashup approaches, where components are mostly called *widgets*, we always use the term *component*. This is justified by not limiting our component types to user-interface elements.

SmartComponents exploit a set of new W3C technologies called *Web Components*, consisting of *Templates*, *Shadow DOM*, *Custom Elements* and *HTML Imports*. Our framework provides a template and helper methods for defining new SmartComponents that eases the usage of those new technologies. The first technology called *Templates* defines chunks of markup that is parsed by the parser, but is inactive and not rendered. Within the `<template>` tags normal HTML markup is used to describe the structure of the components static content. When creating the SmartComponent, the template's content is copied to an adjunct DOM tree called *Shadow DOM*. By exploiting the *Shadow DOM*, which forms its own scope, we ensure encapsulation and minimize the risk of conflicting styles, names or IDs of elements. SmartComponents are new types of DOM elements that can be defined by authors. The registration of new elements is done using the *Custom Elements* standard. New SmartComponents can be easily integrated in a website by using *HTML Imports*. The import statement uses the `<link>` tag to load external definition files. The new custom element tag can be instantly used in the HTML markup after importing the component's resource file. A SmartComponent is described in an HTML file containing different sections for describing the layout, the styling as well as the functionality. Since we are using *Polymer* as an underlying framework, the definition follows a declarative style and supports e.g. event and data binding and advanced template functionality.

Furthermore, the SmartComposition framework provides an event-driven communication channel using a topic-based publish/subscribe mechanism. This enables loosely coupling. An overview of the communication infrastructure can be seen in Figure 1. SmartComponents can use predefined methods for subscribing to topics and publishing information. By providing a WebSocket-based synchronization service, we enable developers to easily create multi-screen-capable mashup applications. SmartComposition offers a stand-alone solution with no dependencies and side-effects on other components. The client-side part is implemented as a JavaScript object that can be included in any HTML page that hosts SmartComponents. It works like a hook and captures all events sent by the SmartComponents. The captured events are sent to a configured endpoint. A Node.js WebSocket server works as a synchronization endpoint and distributes all received events to other connected screens, where they are again published and can be consumed by the local SmartComponents. By using the WebSocket standard, we enable low-latency, bi-directional communication in the browser.

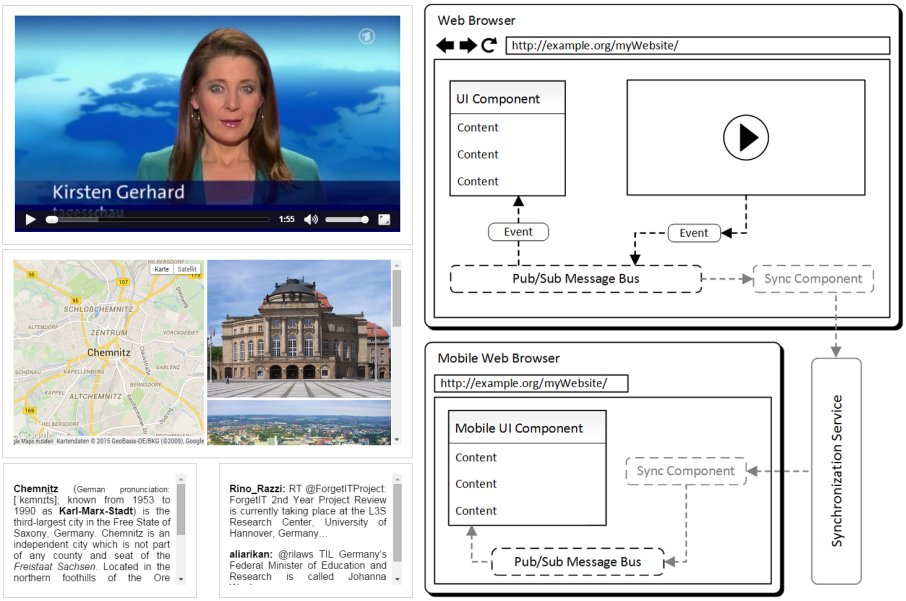

Fig. 1. Example mashup and a simplified overview of the communication infrastructure

To make multi-screen mashup applications more interactive, SmartComponents can be configured to be easily movable by drag-and-drop. Additionally, SmartComponents can also be moved to other connected screens with their state preserved. SmartComponents are stateful DOM objects and provide script interfaces. Thus, developers are able to influence the behavior of the used components on runtime with standard HTML5 DOM methods. SmartComponents can be added, removed and reconfigured at any time. By making SmartComponents available as HTML elements, users that are familiar with HTML but do not have knowledge in programming are also able to create mashups.

3 Checklist

Mashup Type	Hybrid mashups
Component Types	Data components Logic components 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 (Dynamic Modification of Running Mashup)
Interaction Technique	Editable Example
Online User Community	None

4 Planned Demonstration

The demonstration will show how we compose a mashup by using SmartComponents. We will provide a set of components that feature different functionality: e.g. a video, translation, semantic extraction, Google maps, Google images, Twitter and Wikipedia component. They can be UI components as well as data and logic components. We will show how the different components are working together and can be combined by adding them to the markup of an HTML document. Since our framework can be edited live, we demonstrate adding, removing and reconfigurating components directly in the browser by editing the markup. To proof the multi-device capabilities of our solution, we show that SmartComponents can display different kinds of information synchronized on multiple screens, and that they can even be moved between screens. For an interactive demonstration, the audience can join the session by visiting a given URL to see the synchronization live in a working example.

Bibliography

[1] Aghaee, S., Nowak, M., Pautasso, C.: Reusable Decision Space for Mashup Tool Design. In: 4th ACM SIGCHI symposium on Engineering interactive computing systems. pp. 211–220. Copenhagen, Denmark (June 2012)

[2] 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)

[3] Krug, M., Wiedemann, F., Gaedke, M.: SmartComposition: A Component-Based Approach for Creating Multi-screen Mashups. In: Web Engineering, LNCS, vol. 8541, pp. 236–253. Springer International Publishing (2014)