# WebSoDa: A Tailored Data Binding Framework for Web Programmers Leveraging the WebSocket Protocol and HTML5 Microdata

Matthias Heinrich[1] and Martin Gaedke[2]

[1] SAP AG, SAP Research Dresden, Germany
matthias.heinrich@sap.com
[2] Department of Computer Science, Chemnitz University of Technology, Germany
martin.gaedke@informatik.tu-chemnitz.de

**Abstract.** The data binding pattern is an established technique to couple user interface (UI) elements and data objects. Various markup languages (e.g. Microsoft XAML, Adobe MXML) integrate advanced data binding concepts in order to ease application development. However, the HTML standard does not embrace means for data binding although being the Web markup language supported by millions of Web programmers. Therefore, we propose a standard-compliant WebSocket-based Data Binding (WebSoDa) framework. The WebSoDa framework synchronizes data objects and UI elements by orchestrating a Microdata-based data binding language as well as a client-side and a server-side messaging component. Thus, developers may speed up the tedious task of implementing binding associations in Web applications.

## 1 Introduction

In the last decade, the trend to move desktop applications to the Web continued due to the lean upgrade process, the ease of consumption and the broad device support [4]. However, bringing the desktop experience to Web applications is a demanding task. Especially challenging is the application development requiring server-side updates (e.g. stock ticker or twitter feed). The integration of server-side updates encompasses various tasks such as implementing messaging hubs, specifying a messaging format and realizing an update UI mechanism. Realizing these tasks in a distributed environment translates to a repetitive and time-consuming development activity. Therefore, we propose the WebSoDa framework which empowers developers to specify bindings in a minimal markup language. Thus, the complexity exposed by several low-level programming tasks is wrapped in a concise and declarative vocabulary. Consequently, programmers may rapidly realize two-way data binding associations.

In this paper, we proceed with a brief discussion of the state of the art in section 2. Sections 3 and 4 describe the distinct framework components and outline their interaction. In section 5, we summarize the benefits of WebSoDa.

S. Auer, O. Diaz, and G.A. Papadopoulos (Eds.): ICWE 2011, LNCS 6757, pp. 387–390, 2011.

## 2 State of the Art Data Binding Frameworks

Currently, various data binding frameworks exist offering distinct capabilities. Their main objective is to keep the UI and the model in synch while featuring a lightweight binding language. The following overview discusses these frameworks in the light of the Web development domain, which requires standard-compliance and support for distributed systems including a lean bi-directional communication protocol.

**Desktop Binding Frameworks:** They are widely adopted, because they have proven to speed up the reoccurring development task of coupling UI elements and data objects. Two prominent examples are the Microsoft Windows Presentation Foundation [6] and the Java JFace toolkit [5]. However, these frameworks are not applicable to distributed systems, such as the Web.

**Rich Internet Application (RIA) Binding Frameworks:** After succeeding in the desktop market, the binding concept was adopted in the Web domain. RIA frameworks such as Microsoft Silverlight or Adobe Flex introduced binding features [1]. Both frameworks offer markup languages (XAML, MXML) to define bindings. Nevertheless, choosing a RIA technology significantly narrows the addressable market since applications require a dedicated browser plug-in.

**JavaScript-based Binding Frameworks:** Another approach to setup data bindings is facilitated through various popular JavaScript frameworks (e.g. Prototype, YUI) [7]. They address the HTTP limitation of allowing solely client-side communication requests by leveraging expensive polling techniques (AJAX). Additionally, their binding syntax is based on pure JavaScript code rather than on declarative markup. Thus, the approach implies a steep learning curve since Web programmers have to get familiar with HTML, JavaScript and a server-side programming language.

## 3 The WebSoDa Architecture

In order to simplify the cumbersome task of implementing data bindings, we have developed the WebSoDa framework. The standard-compliant framework consists of three essential building blocks: a language to define bindings in HTML5 as well as two messaging components executing synchronization calls.

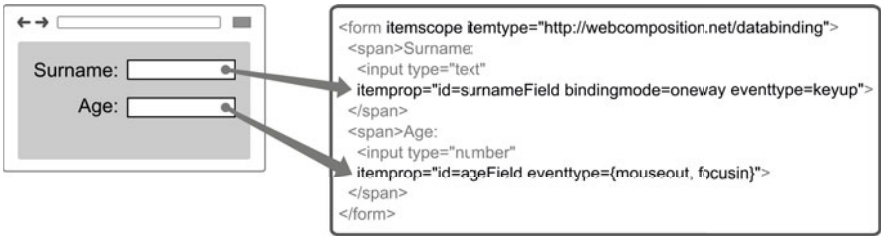


**Fig. 1.** A rendered HTML5 form and the associated client-side binding markup

The data binding language builds upon the Microdata specification [3] which is the default HTML5 extension mechanism. It offers three crucial attributes (itemscope, itemtype, itemprop) to describe annotation elements called items. While the *itemscope* attribute defines the reach of an item, the *itemtype* distinguishes the item by attaching a type URL. The item itself may contain several properties which represent key-value pairs. To create a new item property the *itemprop* keyword is used.

Figure 1 depicts the application of Microdata annotations. A form-container exposes two input elements which accept textual user input. An itemscope attribute creates a new typed item valid within the opening and the closing form-tag. Furthermore, two properties are defined with keys encapsulating the binding expressions. The corresponding values are the inputs provided by the end-user.

The data binding language is a minimal set of attributes. First an identifier named *id* has to be defined. Secondly, an attribute *bindingmode* expresses whether UI changes are only propagated to the model (oneway) or if model changes are also reflected on the UI (twoway). The third attribute *eventtype* specifies a set of JavaScript events which trigger the client to propagate changes to the server. All attributes are specified within the itemprop value field. In Fig. 1 two client-side data binding associations are defined using different binding modes and distinct update events.

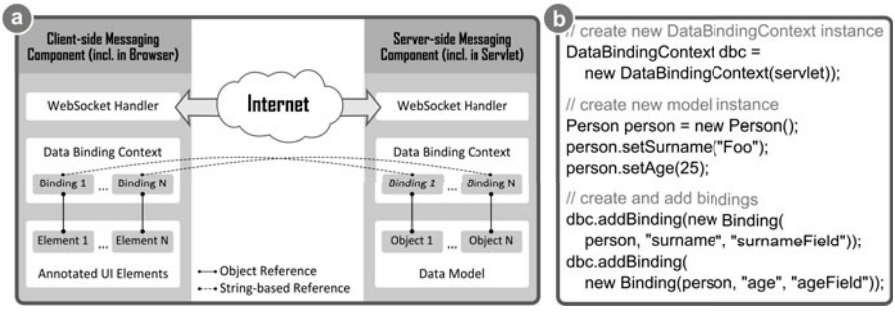


**Fig. 2.** The WebSoDa architecture and an example of a server-side binding expression

Besides defining a binding language, the WebSoDa framework features a client-side and a server-side messaging component depicted in Fig. 2a. These components communicate over the bi-directional WebSocket protocol [2] which is currently standardized by the IETF. The WebSocket protocol was chosen because of its native two-way communication support, its standard-compliance, the broad browser support and the minimal protocol overhead.

The client-side messaging component is packaged in an external JavaScript file named WebSoDa.js. According to Fig. 2a it is comprised of a WebSocket handler, a data binding context and the HTML5 description containing the annotated UI elements. Once the browser loads the HTML5 as well as the JavaScript definition, the data binding context registers all annotated UI elements and creates

a binding object for each. As soon as the user changes the input of a registered form field, the WebSocket handler assembles an update message which is sent to the server-side messaging component. This component retrieves the attached server-side binding which is identified by the provided *id* attribute. After selecting a binding, the UI change can eventually be propagated to the referenced model object. Updates can also be triggered by the server. In case of a model change, a message will be sent to the client and the associated field will be updated. Currently, the server-side component is implemented as a Java Servlet. The Java source code to setup a server-side data binding is illustrated in Fig. 2b.

## 4 Online Demo

A screencast showing the process of setting up the data binding associations, the client-side JavaScript and the server-side Java source code are available at `http://vsr.informatik.tu-chemnitz.de/demo/WebSoDa/`.

## 5 Conclusion

In this paper, we described a WebSocket-based data binding framework which fosters the development efficiency by integrating declarative binding expressions in HTML5. A client-side messaging component parses the binding expressions and automatically establishes bi-directional connections to the server-side model. Thus, Web developers may profit from the data binding concept without introducing a new technology. In contrast to the traditional form-processing approach, the proposed framework supports a two-way binding reflecting model changes instantly on the UI. Furthermore, bindings are highly configurable with respect to triggering events and binding modes. Besides offering flexibility, the lightweight framework embraces Web standards. Consequently, it only requires a state of the art Web browser supporting WebSockets and Microdata. Hence, a broad range of applications might benefit from the WebSoDa framework.

## References

1. Deitel, P.: Internet & World Wide Web: How to Program. Prentice-Hall, Englewood Cliffs (2007)
2. Hickson, I.: The WebSocket protocol draft-76 (2010), `http://tools.ietf.org/html/draft-hixie-thewebsocketprotocol-76/`
3. Hickson, I.: HTML Microdata - W3C Working Draft (2011), `http://www.w3.org/TR/microdata/`
4. Jazayeri, M.: Some trends in web application development. In: 2007 Future of Software Engineering, FOSE 2007, pp. 199–213. IEEE Computer Society, Washington, DC,USA (2007)
5. McAffer, J., Lemieux, J.-M., Aniszczyk, C.: Eclipse Rich Client Platform. Addison-Wesley Professional, London (2010)
6. Nathan, A.: Windows Presentation Foundation Unleashed. Sams (2006)
7. Orchard, L.M., Pehlivanian, A., Koon, S., Jones, H.: Professional JavaScript Frameworks: Prototype,YUI, ExtJS, Dojo and MooTools. Wrox (2009)