Generation of Web Frontends from API Documentation with Direwolf Interaction Flow Designer

István Koren and Ralf Klamma

Advanced Community Information Systems (ACIS) Group,
RWTH Aachen University, Ahornstr. 55, 52056 Aachen, Germany
{koren,klamma}@dbis.rwth-aachen.de
http://dbis.rwth-aachen.de

Abstract. Services and their interfaces are a cornerstone of Web applications. API description formats help developers in accessing and combining service functionalities. The OpenAPI specification has gained considerable popularity over the last years. Existing tools around OpenAPI support the generation of HTML interfaces to mockup requests. While these interfaces are suited well for developers, it remains hard for non-developers to assess service functionalities. To this end, we present the Direwolf Interaction Flow Designer in this demo. It parses OpenAPI documents to generate Web frontends with the help of the Interaction Flow Modeling Language (IFML). A screencast of our tool is available online[1].

Keywords: OpenAPI; IFML; Web Components; Interaction Design

1 Introduction

Componentization in software engineering increases modularization and reusability. Application program interfaces (APIs) are a prerequisite for ensuring compatibility of the software modules. To enable automated requests and to increase usability for developers, extensive documentation is required. In the realm of service-oriented architectures, the Web services description language (WSDL) has been used as contract for service access. For the popular world of lightweight REST-based interfaces on the Web, a number of API documentation specifications have been proposed. The OpenAPI specification [5], formerly known as Swagger, can be used to create machine-readable descriptions of service interfaces. A tool ecosystem around the specification allows code and mockup generation, amongst others. Swagger UI is an example for generating a user interface out of OpenAPI; data has to be entered in pure JSON. Thus, it is mainly suitable for developers. For application designers or even end users, it is difficult to assess API functionalities out of its documentation.

[1] https://youtu.be/KFOPmPShak4

In this demo we present a collaborative Web tool that generates Web frontends based on OpenAPI. It first generates an Interaction Flow Modeling Language (IFML) model out of the documentation. IFML is a visual modeling language that aims to model user interactions of user interfaces. The model is then transformed into an HTML5 & JavaScript application, which is based on Web Components. In Section 2, we show the process of getting to Web frontends and highlight implementation details. Section 3 concludes the article and points to future work.

2 Transformation Approach

Related work on generating Web frontends from API documentation are mainly to be found in the SOA realm, like the *ServFace Builder* [4] and the adaptive user interface generation framework [3]. Rodríguez et al. parse HTML API documentation into a custom metamodel [6]. The Cameleon reference framework defines a multi-step procedural model from task models and an abstract user interface specification to a concrete user interface [2]. Other approaches like the commercial WebRatio[2] are able to generate models from API documentation, and yet others generate user interfaces out of models.

Our main goal is to create running Web application prototypes out of state-of-the-art Web documentation. As a starting point, we take an OpenAPI file. The format is human-readable and can be represented in JSON or YAML formats. It consists of various parts; in the header, versioning and server-specific information can be found. The central part maps service paths to operations and defines valid input parameters and outputs. For this reason, it uses JSON Schema types either in place or referenced in the components part of document.

As an intermediary step, we use IFML [1]. In the following, we explain its main modeling concepts. *View Containers* group multiple different user interface elements together. *View Components* are nodes that represent a certain goal of the user interface. *Events* are either system- or user-generated. *Actions* represent functionalities happening on user interaction. These model elements are connected via *navigation* and *data flows*. Navigation flows happen when the content of the user interface change. Data flows model internal data flows between model elements. The appendix of the standards presents various mappings to real user interface code, e.g. XAML or HTML. We extend the mapping and generate HTML5 & JavaScript frontends.

Figure 1 shows the transformation process of generating Web frontends from left to right. We highlighted equivalent parts within documentation, model and frontend in the same background color. The purple/dark background highlights the server endpoint information in the IFML model. This information is modeled as IFML action. As the server information cannot be directly represented in the IFML model, we save it as metadata of the model. In the resulting HTML5 markup and JavaScript in the background, the server metadata is

[2] https://www.webratio.com

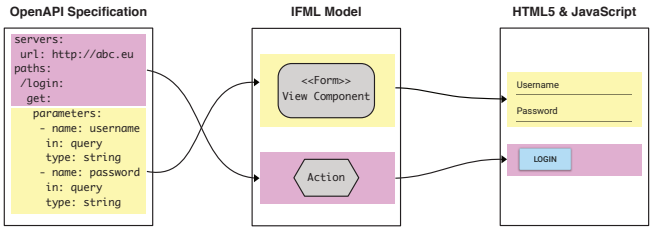

Fig. 1. Transformation of OpenAPI via IFML to HTML5 & JavaScript

equally present. We provide predefined elements that e.g. display lists, or dynamically generate forms out of the JSON Schema description. In the final user interface, upon clicking the form submit button, the element retrieves the server endpoint information from the metadata, gets the input from the form and then performs a request to the server. Similarly, the list element retrieves an array from the REST endpoint and then displays the items in a list or table.

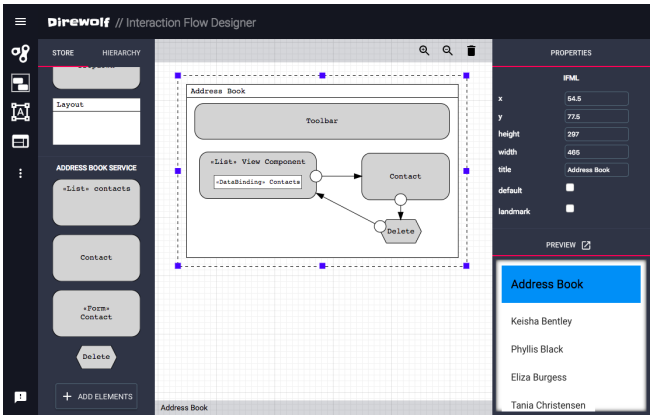

Fig. 2. Interaction Flow Designer and HTML Preview

2.1 Implementation

Figure 2 shows a screenshot of our tool with a basic address book service. It was built with Web Components and the Polymer 2.0 library based on object-oriented JavaScript. On the left, the *Model Element Palette* shows the available model items. At its bottom, an *Add Elements* button opens a dialog, where the URL of an OpenAPI documentation can be added. When confirming the dialog, the model elements are generated and added to the palette.

The *Interaction Flow Modeler* is the central part of the UI. It shows the model that can be freely dragged around and zoomed in and out. Model elements and

edges can be selected to change the size and group. Upon selecting an element, its properties are shown on the right of the screen, in the *Properties Browser*. Below the Properties Browser, the HTML5 Preview pane shows a preview of the generated model element. The whole tool is collaborative, so that users can work on the model in near real-time at remote places. The synchronization is achieved via the Yjs library[3]. For instance, the property browser is directly working on the underlying data model, so that other parts of the UI are notified in order to update the representation.

3 Conclusion and Future Work

In this demo paper, we presented Direwolf Interaction Flow Designer that generates Web frontends out of Web service descriptions. The goal is to create high-fidelity prototypes to test service functionalities.

As models are abstractions of real world phenomena, our approach includes some limitations. For instance, certain details like password input fields cannot be represented in the OpenAPI documentation. Another drawback is that we are currently not handling the security parameters present in the OpenAPI specification. We are currently embedding our tool in a complete application design life cycle. For this, we see the Internet of Things domain as a valuable target, as devices often have a lot of functionalities that are hard to grasp for end users.

Acknowledgements. The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 Programme through the project "WEKIT" (grant no. 687669).

References

1. Brambilla, M., Fraternali, P.: Interaction Flow Modeling Language: Model-Driven UI Engineering of Web and Mobile Apps with IFML. The MK/OMG press, Morgan Kaufmann (2014)
2. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for multi-target user interfaces. Interacting with computers 15(3), 289–308 (2003)
3. He, J., Yen, I.L.: Adaptive User Interface Generation for Web Services. In: IEEE International Conference on e-Business Engineering (ICEBE'07). pp. 536–539 (2007)
4. Nestler, T., Feldmann, M., Hübsch, G., Preußner, A., Jugel, U.: The ServFace Builder - A WYSIWYG Approach for Building Service-Based Applications. In: Benatallah, Boualem et al.and Casati, F., Kappel, G., Rossi, G. (eds.) Web Engineering, Lecture Notes in Computer Science, vol. 6189, pp. 498–501. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
5. OpenAPI Initiative: The OpenAPI Specification (2018), https://www.openapis.org/
6. Rodríguez, R., Espinosa, R., Bianchini, D., Garrigós, I., Mazón, J.N., Zubcoff, J.J.: Extracting Models from Web API Documentation. In: Grossniklaus, M., Wimmer, M. (eds.) Current Trends in Web Engineering, Lecture Notes in Computer Science, vol. 7703, pp. 134–145. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

[3] http://y-js.org