The ServFace Builder - A WYSIWYG Approach for Building Service-Based Applications

Tobias Nestler[1], Marius Feldmann[2], Gerald Hübsch[2], André Preußner[1], and Uwe Jugel[1]

[1] SAP Research Center Dresden
Chemnitzer Str. 48, 01187 Dresden, Germany
{tobias.nestler,andre.preussner,uwe.jugel}@sap.com
[2] Technische Universität Dresden, Department of Computer Science, Institute for Systems Architecture, Computer Networks Group
{marius.feldmann,gerald.huebsch}@tu-dresden.de

Abstract. In this paper we present the ServFace Builder, an authoring tool that enables people without programming skills to design and create service-based interactive applications in a graphical manner. The tool exploits the concept of service annotations for developing multi-page interactive applications targeting various platforms and devices.

Keywords: Service Composition at the Presentation Layer, Model-driven Development, Service Frontends.

1 Background

Developing service-based interactive applications is time consuming and nontrivial. Especially, the development of user interfaces (UIs) for web services is still done manually by software developers for every new service. This is a very expensive task, involving many error-prone activities. Parameters have to be bound to input and output fields, operations to one or several application pages, such as forms or wizards, and operation invocations to UI-events like button-clicks.

This demo presents an authoring tool called *ServFace Builder* (Fig. 1). The tool leverages web service annotations [1] enabling a rapid development of simple service-based ineractive applications in a graphical manner. The tool applies the approach of service composition at the presentation layer, in which applications are build by composing web services based on their frontends, rather than application logic or data [2]. During the design process, each web service operation is visualized by a generated UI (called service frontend), and can be composed with other web service operations in a graphical manner. Thus, the user, in his role as a service composer and application designer, creates an application in WYSIWYG (What you see is what you get) style without writing any code.

The ServFace Builder aims to support domain experts. Domain experts are non-programmers that are familiar with a specific application domain. They need to understand the meaning of the provided service functionality without having a deep understanding of technical concepts like web services or service

B. Benatallah et al. (Eds.): ICWE 2010, LNCS 6189, pp. 498–501, 2010.

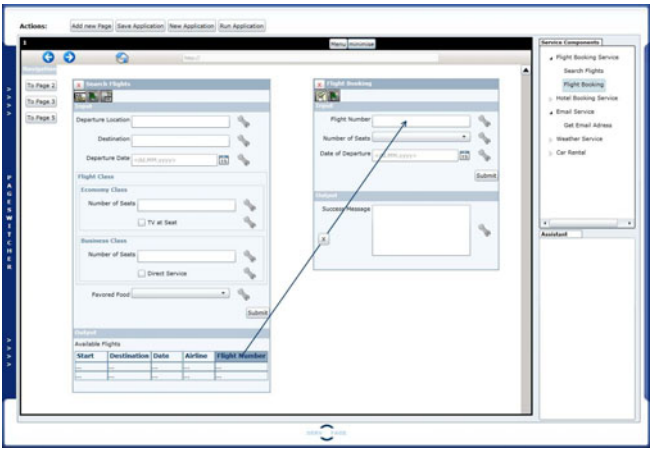

Fig. 1. The ServFace Builder

composition. Our demo shows, how we break down these aspects to the modeling of data-flow and page-flow to realize multi-page applications using our authoring tool. Finally, we show how the tool generates the executable application based on the designed models.

Technical details as well as the existing research challenges of the proposed approach are provided in [3] and [4].

2 The ServFace Builder - Components and Models

Components within the ServFace Builder are called *service frontends*, which are visualized via form-based UIs. They represent a single operation of a WSDL-described web service. The frontends consist of a nested container structure, which includes UI-elements like text fields or combo boxes that are bound to the corresponding service operation parameters. The ServFace Builder utilizes the advantages of the ServFace service annotations to visualize the service frontends already during design-time to give the user an impression of the resulting UI.

The annotations are created by the service developer to provide additional information about the web service, and present the knowledge transfered from the service developer to the service composer to facilitate the understanding and simplify the composition of web services. These annotations are reusable information fragments, which are usually not available for plain WSDL-based approaches, for which they might be non-formally defined in the service documentation. The ServFace annotations are stored in an annotation model that is based on a well-specified meta-model, which references to concrete elements within a WSDL-document. Service annotations provide extensive additional information covering visual, behavioral, and relational aspects. Visual aspects are, e.g., human readable *labels* for technical field names, *grouping* of parameters,

definition of *enumerations* for predefined values. Aspects of the behavior of UI-elements are, e.g., client-side *validation* rules, *suggestion* of input values. And finally, Relation-annotations are used to express, e. g., *semantic data type relations*). An in-depth discussion of the annotation model, including examples and a description of the referencing mechanism that links annotations to service descriptions, can be found in [1].

Within the ServFace Builder, each web service is represented by a service model that is automatically inferred from the WSDL and additionally contains the corresponding annotation model. The service model bundles the necessary information about a service including their operations and parameters and holds a reference (Uniform Resource Identifier) to the WSDL file. The information stored in the service model serves as the foundation for the visualization of the frontends. Details about the service model and its elements can be found in [3].

The service model is part of the *Composite Application Model* (CAM). The CAM defines the overall application structure that is visualized within the ServFace Builder. Besides generally describing the integrated services, it describes inter-service connections as data flow, and the navigation flow of an application as page transitions coupled with operation invocations. The CAM is used as the serialization format for storing and loading the modeled interactive applications and serves as the input for the generation of executable applications. It is not bound to a specific platform, and thus can be reused for developing service-based interactive applications for various target technologies.

In the CAM, a composite application is represented by a set of pages. Every page is a container for service frontends and represents a dialog visible on the screen. The pages can be connected to each other to define a navigation flow. Adding a service operation to a page triggers the inference of the corresponding UI-elements necessary for using this operation. The CAM instance is continuously synchronized with user actions, when adding pages, integrating frontends, or modeling data flow. This instance can be serialized at any point in time as a set of Ecore-compliant XMI-files. These files are used as input for a model-to-code transformation process; the last step during the application development process. The CAM as well as the code generation process are described in [3].

3 Service Composition at the Presentation Layer

The ServFace Builder implements an approach coined *service composition at the presentation layer* [4] in order to combine services in a WYSIWYG manner. The user of the tool directly interacts with single UI-elements, entire service frontends and pages to model the desired application. No other abstraction layer is required to define data or control flows. In this demo, we present the graphical development process including the following steps to be accomplished by the user:

1. Select a target platform: As the initial step, the user has to select the target platform from a set of predefined platform templates. The size of the composition area is adjusted depending on the selection to reflect screen size limitations, and an initial CAM for the new application will be created.

2. Integrate a service frontend: After the platform selection, an initial blank page is created and the actual authoring process begins. The Service Component Browser offers all services provided via a connected service repository. To integrate a specific service operation into the application, the user has to drag it from the Service Component Browser to the Composition Area. The tool automatically creates the corresponding service frontend, which can be positioned on the page. An optional design step is to modify the configuration of the frontend, e.g., by hiding single UI elements.

3. Define a data-flow: To define a data-flow between two service frontends, the user has to connect UI-elements of both frontends to indicate the relationship. A connection is created by selecting the target UI-element that is to be filled with data and the source UI-element from which the data is to be taken. The data-flow is visualized in form of an arrow.

4. Define a page-flow: Frontends can be placed on one page or distributed over several pages to create a multi-page application. All pages are shown in a graph-like overview and can be connected to define the page flow. The definition of the page flow is done in a graphical way as well. The user has to connect two pages in order to create a page transition.

5. Deploy the application: After finishing the design process, the modeled application can be deployed according to the initial platform selected. Depending on the type of target platform, the application is either automatically deployed on a predefined deployment target (e.g., as a Spring-based web application) or is offered for download (e.g., as a Google Android application).

Acknowledgment

This work is supported by the EU Research Project (FP7) ServFace[1].

References

1. Janeiro, J., Preussner, A., Springer, T., Schill, A., Wauer, M.: Improving the Development of Service Based Applications Through Service Annotations. In: Proceedings of the WWW/Internet Conference (2009)
2. Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., Saint-Paul, R.: Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. IEEE Internet Computing 11(3), 59–66 (2007)
3. Feldmann, M., Nestler, T., Jugel, U., Muthmann, K., Hübsch, G., Schill, A.: Overview of an End User enabled Model-driven Development Approach for Interactive Applications based on Annotated Services. In: Proceedings of the 4th Workshop on Emerging Web Services Technology. ACM, New York (2009)
4. Nestler, T., Dannecker, L., Pursche, A.: User-centric composition of service frontends at the presentation layer. In: Proceedings of the 1st Workshop on User-generated Services, at ICSOC (2009)

[1] `http://www.servface.eu`