An Ontology-Based Method for Rendering Execution Results of Dynamically Invoked Web Services

Chao-liang Zhong¹, Akihiko Matsuo², Jun Zhang¹, Zhu-long Wang¹, Hao Yu¹

¹Information Technology Laboratory, Fujitsu R&D Center Co., Ltd.
²Software and Solution Laboratories, Fujitsu Laboratories Ltd.

Introduction

Rendering and displaying the execution results of Web services is necessary and important in some scenarios. Web services technology, however, is mainly designed for the interoperability among machines, whose inputs and outputs are defined in the form of XML schema without presentation information, hence it is unable to present the execution results of Web services to human beings directly and friendly.

For statically invoked services, since the type of the output can be determined in the stage of design, programmers are able to render the results by hand-coding. While for dynamically invoked services, it is impossible to forecast the type of the execution results in the stage of design, because the type varies every time it executes. In this case, the hand-coding method does not work. Instead, templates configured according to the XML schema of each type may be utilized to render the execution results. The templates method is flexible, since it enables changing the rendering by modifying templates, rather than modifying codes. The templates method alone, however, is still not appropriate because the XML schemas are designed by different service providers independently without common understandings, which will result in too many templates that are unique in syntax level but duplicated in semantic level. The templates are not related to each other and thus difficult to be maintained.

To solve the above problem, this paper proposed the following method.

Method

The method proposed in this paper takes advantages of an ontology that describes the concepts of data types and their relationships. The ontology is utilized to annotate the types of inputs and outputs when creating an OWL-S¹ Web service. Templates are generated and stored according to each data type in the ontology, and can be used into an application as an input, and returns the rendered results in HTML format.

The templates are generated and stored according to each data type in the ontology, and can be retrieved with the URL of corresponding data type. The HTML language is used to describe the template. In the template, the property whose type is DataProperty or ObjectProperty represents an attribute of the data type.

The templates are generated and stored according to each data type in the ontology, and can be retrieved with the URL of the data type. The HTML language is used to describe the template. In the template, the property whose type is DataProperty has to be explicitly described. If it is explicitly described, the property will appear at the specified position in the rendered results. Or else, it will be appended to the end of the rendered results.

Conclusion

In summary, with the templates, the execution results of dynamically invoked services can be rendered flexibly, and since the templates are mapped to the types in the ontology, the semantically duplicated templates are aggregated as a single template and thus can be well organized and managed. Moreover, if the template of a type is missed, system can automatically find and use the template of a similar type that has the shortest distance from the required type in ontology.

Further research will focus on improving the algorithm of rendering engine, e.g., handling the cycles in ontology, etc. The integration with XSLT is also a direction of future research.

References