Towards Automated Invocation of Web APIs

Ning Li, Carlos Pedrinaci, Jacek Kopecky, Maria Maleshkova, dong Liu, John Domingue

Knowledge Media Institute, The Open University, Milton Keynes, United Kingdom
{n.li, c.pedrinaci, j.kopecky, m.maleshkova, d.liu, j.domingue}@open.ac.uk

Abstract. This paper, targeting the large variety of Web APIs, presents an approach towards automated invocation of Web APIs. This approach applies a data schema, to the SAWSDL lowering schema mapping, a grounding mechanism that connects the ontological representations of Web APIs with their execution messages. It is intuitive to existing standard efforts and effective in coping with the heterogeneities witnessed by a majority of Web APIs.

1. Invocation of Linked Web APIs

In recent years, the world of service on the Web has witnessed an increasing dominance of Web APIs over "classical" Web services characterized by WSDL and SOAP [1]. The Semantic Web Service (SWS) community, aiming to increase the level of automation for tasks like discovery, composition and invocation through semantic descriptions, is challenged by the large variety of Web APIs, particularly when it comes to automated invocation. Not only are Web APIs heterogeneous in the forms of elements essential to their invocation, such as address URI, but they also lack machine processable descriptions to address this heterogeneity [1]. hRESTS, a microformat called *HTML for RESTful Services*, and MicroWSMO, an extension of hRESTS that adds means for service automation, have been used, together with the lightweight ontological service model WSMO-Lite, for adding semantic annotations to existing Web APIs. However, they fall short, in their current capacities, to describe the differently documented Web APIs when it comes to enable automated invocation, particularly in the grounding process, the mechanism that connects the ontological representations of Web APIs with their on-the-wire execution messages. Ontological data needs to be grounded not only to the right format, e.g. JSON or XML, but also to the right place, e.g. underspecified path parameter or query string of URI, message header or message body. In this paper, we propose a solution towards automated invocation of Web APIs that targets the grounding process, referred to as lowering in SAWSDL.

2. Implementation

Given the fact that the data input for a Web API can be used to fill underspecified parts of the invocation address URI, of the message header, or of the message body, we propose a data schema, which indicates where an input parameter is grounded to and in what format, by working together with service annotations. The schema, shown in Fig. 1 and marked in bold when mentioned in text, indicates that input, tagged as

request data, can serve as **template parameter** to replace variables in an URI Template, or as **query parameter** being attached to the end of an address URI, or play no roles at all in forming address URI and simply appear in request **header** or **body**. There are **name** and **value** attributes, which are self-explanatory, specified to data elements **template parameter, query parameter** and **header,** and **mediaType** attribute to **body** element. When writing lowering scripts, the values of those attributes should be given in joint consideration with service annotation. XSPARQL (http://xsparql.deri.org/) is deployed as the language for writing lowering scripts due to its direct support of embedding such schema into the lowering schema mapping.

```
<requestData>
   <templateParam name="..." value="..."/>
      .......
   <queryParam name="..." value="..."/>
      .......
   <header name="..." value="..."/>
      .......
   <body mediaType="...">
      .......
   </body>
</requestData>
```

Fig. 1. Lowering data schema

```
Declare namespace soa4estate=" http://example.org/ontologies/SOA4Estate.rdf";
<requestData>
{
 for $postcode where {
     $myProperty a soa4estate:Property.
     $myProperty soa4estate:hasPostcode $postcode.}
return
<queryParam name="place_name" value="{$postcode}"/>}
</requestData>
```

Fig. 2. Nestoria lowering script

We give an example here with the Nestoria property Web API (http://www.nestoria.co.uk/help/api-search-listings), which takes a location parameter as input. Among the few options, we choose *place_name* and give *postcode* information as value for *place_name*. The lowering script is shown in Fig. 2. With the address URI described as http://api.nestoria.co.uk/api?action=search_listings and an input as :p a soa4estate: Property; hasPostcode MK7. An invocation address URI can be worked out with a little computation as http://api.nestoria.co.uk/api?action=search_listings&place_name=MK7.

3. Discussions

To cater for a comprehensive list of Web API characteristics and to avoid the injection of additional information to the lowering process, arguably for data-transformation only, alternative solutions towards automated invocation of Web APIs include extending service description model or relying on additional HTTP ontology and invocation-specific rule reasoning [2].

Acknowledgements This work was partly funded by the EU project SOA4All (FP7-215219).

References

1. Maleshkova, M., Pedrinaci, C. and Domingue, J. (2010): Investigating Web APIs on the World Wide Web, European Conference on Web Services (ECOWS), Ayia Napa, Cyprus.
2. Lambert, D. and Domingue, J. (2010) Photorealistic Semantic Web Service Groundings: Unifying RESTful and XML-RPC Groundings Using Rules, with an Application to Flickr, The 4th International Web Rule Symposium (RULEML 2010).