Supporting the Creation of Semantic RESTful Service Descriptions

Maria Maleshkova, Carlos Pedrinaci, John Domingue

Knowledge Media Institute (KMi)
The Open University, Milton Keynes, United Kingdom
{m.maleshkova, c.pedrinaci, j.b.domingue}@open.ac.uk

Abstract. Research on semantic Web services (SWS) has been devoted to reduce the extensive manual effort required for manipulating Web services by enhancing them with semantic information. Recently, the world around services on the Web, thus far limited to "classical" Web services based on SOAP and WSDL, has significantly evolved with the proliferation of Web applications and APIs, often referred to as RESTful Web services. However, despite their success, RESTful services are currently facing similar limitations to those identified for traditional Web service technologies and present even further difficulties, such as the lack of machine-processable service descriptions. In order to address these challenges and to enable the wider adoption of RESTful service technologies, we advocate an integrated lightweight approach for formally describing semantic RESTful services. The approach is based on the use of the hRESTS (HTML for RESTful Services) and MicroWSMO microformats, which enable the creation of machine-readable service descriptions and the addition of semantic annotations, correspondingly. Finally, we present SWEET–Semantic Web sErvices Editing Tool–which effectively supports users in creating semantic descriptions of RESTful services based on the aforementioned technologies.

1 Introduction

Since the advent of Web service technologies, research on semantic Web services (SWS) has been devoted to reduce the extensive manual effort required for manipulating Web services. The main idea behind this research is that tasks such as the discovery, negotiation, composition and invocation of Web services can have a higher level of automation, when services are enhanced with semantic descriptions of their properties. Recently, the world around services on the Web, thus far limited to "classical" Web services based on SOAP and WSDL, has significantly evolved with the proliferation of Web applications and APIs, also referred to as RESTful Web services [1]. RESTful services conform to the REST architectural style [2] and are characterized by their relative simplicity and their natural suitability for the Web, which is indeed closely related to the growing popularity and use of Web 2.0 technologies.

Currently, more and more Web applications and APIs expose functionalities in the form of RESTful services. In addition, popular Web 2.0 applications

like Facebook, Google, Flickr and Twitter offer easy-to-use, resource-oriented APIs, which not only provide simple access to different resources but also enable combining heterogeneous data coming from diverse services, in order to create data-oriented service compositions called mashups. Despite their success, RESTful services are currently facing the same limitations that were identified for traditional Web service technologies and present even further difficulties. In particular, as opposed to WSDL services, there is no widely accepted structured language for describing RESTful ones, even though there are some initial approaches in this area [8], [9]. As a consequence, in order to use RESTful services, developers are obliged to manually locate, retrieve, read and interpret heterogeneous documentations of RESTful services in HTML, and subsequently develop custom tailored software that is able to invoke and manipulate them.

Therefore, the lack of machine-readable descriptions and the fact that the majority of existing RESTful service descriptions have no semantic annotations are the two main challenges, which need to be addressed in order to provide a certain level of automation of common service tasks. In this paper we present an integrated lightweight approach for formally describing semantic RESTful services, by using the hRESTS (HTML for RESTful Services) microformat [3] for the creation of machine-readable service descriptions. Microformats [4] offer means for annotating human-oriented Web pages in order to make key information machine-readable, while hRESTS, in particular, enables the creation of machine-processable Web API descriptions based on available HTML documentation.

hRESTS is complemented by the MicroWSMO [5] microformat, which supports the semantic annotation of service properties in a SAWSDL-like [6] manner. MicroWSMO introduces additional HTML classes, in order to enable the linking of ontological elements and the provisioning of machinery for transforming data exchanged between two services used in a service composition. Moreover, concrete semantics can be added by applying WSMO-Lite [7] service semantics, which enable the integration of RESTful services with WSDL-based ones. As a result, when semantic descriptions of WSDL-based services and of RESTful services are both based on WSMO-Lite, the unified query over both "classical" Web services and RESTful services is provided.

Our main contribution is practically enabling developers to create semantic RESTful services by using SWEET. SWEET (Semantic Web sErvices Editing Tool) is the first tool, which provides functionalities for both the creation of machine-readable RESTful service descriptions and the addition of semantic annotations. SWEET uses the hRESTS and MicroWSMO microformats, however, it hides formalism complexities from the user and assists him/her in making semantic annotations. The result is a structured and semantically annotated HTML description of the RESTful service, which can be saved and republished on the Web. In addition, the resulting HTML can also be transformed into an RDF MicroWSMO description, which can be used for manipulation or storage. We provide a detailed explanation of how to use the tool and an example of service annotation in Sections 3 and 4.

The remainder of this paper is structured as follows: Section 2, provides an overview of the formal description of RESTful services, while Section 3 introduces SWEET, including its components, functionalities and user support. An example of creating semantic RESTful descriptions is given in Section 4. Section 5 presents an overview of related formalisms and approaches. Finally, Section 6 presents future work that will be carried out and concludes the paper.

2 Semantic Annotation of RESTful Services

Currently, when a user is searching for RESTful services for retrieving resource information or for creating a custom mashup he/she is restricted to two main alternatives. First, the user can rely on a set of Web applications and APIs that is known from previous experience or he/she can browse for common Web 2.0 applications, guessing that they might expose some functionalities in the form of RESTful services. Second, the user can search in existing RESTful service repositories[1] that maintain API descriptions manually collected over time. Although useful, searching these repositories is unfortunately poorly supported as, most often, repositories only support browsing through a predefined and rather high-level classification, and keyword-based search. The semantics of the services are not explicitly represented in a way that a machine could benefit from, in order to provide more advanced functionalities such as search based on input and output types, ordering based on certain non-functional properties, etc. As the number of services and service repositories grow, the limitations of current technologies will hamper to an important extent the adoption of RESTful services, much like it previously happened for Web services.

Both the discovery and matching tasks can be significantly improved and automated by enriching RESTful service descriptions with semantic annotations. Currently, there is no way of distinguishing between common HTML of Websites and HTML describing RESTful services. This can be changed by extending HTML RESTful descriptions with hRESTS microformat tags, marking service properties. These tags can be automatically recognized by both crawlers and search engines, facilitating the targeted search and collection of RESTful services. In addition, similarly to SWS the semantic annotation of RESTful service properties can reduce the manual effort required for service discovery and improve the returned results.

Given that the vast majority of available RESTful services are merely described in plain HTML within normal Web pages, we advocate using microformats for structuring service descriptions and for attaching semantic annotations to them. In this paper we present a 3-steps approach visualized in Figure 1 for providing semantic annotations for RESTful services, focussing in particular on SWEET, a tool created explicitly for supporting users in this process. First, the unstructured text description of the RESTful service, in the form of HTML

[1] See for instance `www.programmableweb.com`, which contains up to 2000 API descriptions

content, is extended with hRESTS tags marking all service properties. The resulting service structure is enriched with semantic annotations by adding links pointing to semantic content. Finally, the annotated HTML can be saved and republished, or it can be used to extract RDF MicroWSMO descriptions. The user is supported in completing each of these steps by SWEET, which guides him/her in process of creating semantic RESTful service descriptions and hides formalism complexities behind an easy-to use interface.

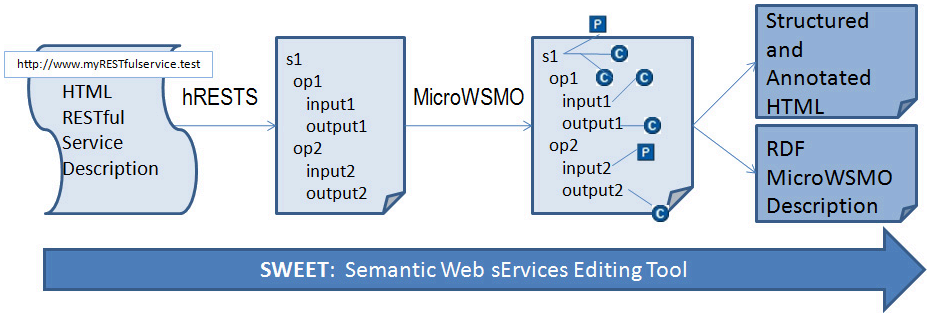

Fig. 1. Semantic Annotation of RESTful Services

2.1 RESTful Service Descriptions

The majority of the RESTful service descriptions are usually given in the form of unstructured text in a Web page, which contains a list of the available operations, their URIs and parameters, expected output, error messages and an example. The description includes all details necessary for a developer to execute the service or use it in a mashup. Based on analyzing a collection of RESTful services, we have identified three common types of RESTful service descriptions. The first type is a single Website containing only one or a number of operations[2], with their corresponding parameter and URIs. However, in contrast to such simpler RESTful services, a lot of Web 2.0 applications contain a plenitude of operations. This second type of descriptions, includes one main page for the service and a number of linked pages, each of which describes one operation[3].

Listing 1.1. Resource-Oriented Description

activity	blogs	auth
– flickr . activity .userComments	– flickr . blogs . getList	– flickr . auth . checkToken
– flickr . activity .userPhotos	– flickr . blogs . postPhoto	– flickr . auth . getFrob

Finally, the last type of RESTful service descriptions are the resource-oriented ones[4]. Listing 1.1 shows parts of the `Flickr`[5] API documentation, where operations are not simply listed but they are rather grouped, based on the resources

[2] `http://open.3scale.net/happenr/happenr`, `http://delicious.com/help/api`
[3] `http://www.geonames.org/export/ws-overview.html`, `http://apidoc.digg.com/`
[4] `http://apiwiki.twitter.com/Twitter-API-Documentation`, `http://www.last.fm/api`
[5] `www.flickr.com/services/api/`

which they manipulate. In the example, there are three resources (activity, blogs and auth), each of which has a set of operations. All these types of RESTful service descriptions, can be syntactically structured by marking service properties with the hRESTS microformat.

2.2 hREST

All possible interactions with RESTful services, and services in general, are specified in the service description, which gives information about requirements and invocation methods. While Web applications and Web APIs contain HTML documentation, which is understandable for humans, it needs to be extended in order to become machine-processable as well. Some existing formalism for RESTful service descriptions include WSDL and WADL. WSDL [8] is an established standard for Web service descriptions, however, it has not found wide adoption for RESTful services and only a few such services have WSDL descriptions. Similarly, WADL [9] does not seem to be gaining acceptance among API provides and instead Web applications and APIs are usually described in textual documentation. However, in order to support the automation of RESTful services, certain key aspects of the descriptions have to be made machine-readable.

Since currently most Web applications and Web APIs rely only on HTML documentation, we use the hREST [3] microformat [4], which enables the creation of machine-processable descriptions on top of existing HTML descriptions. It contains only a few elements, is very lightweight and easy to use. Microformats, in general, facilitate the translation of the HTML tag structure into objects and properties, while hREST in particular, uses `class` and `rel` XHTML attributes to mark key service properties, leaving the visualization of the HTML description unchanged. hRESTS introduces tags for marking the service description as a whole, the used HTTP method, the operation with corresponding input and output, and the service or operation names in the form of labels. It also enables the linking of separate pages to one main page, in the case of complex RESTful service descriptions, by including `rel="section"` tags pointing to the operation Web pages and `rel="start"` tags pointing to the main service description.

2.3 MicroWSMO

hRESTS marks the key properties of the RESTful service and provides a machine-readable description based on the available HTML documentation. The result can be used as the basis for adding complimentary information and annotations, which will contribute to a higher level of automation of the discovery, composition, ranking, invocation and mediation service tasks. As a result Semantic RESTful Services (SRS) can be developed following and adapting approaches from Semantic Web Services (SWS) research.

We use MicroWSMO [5] for the semantic annotation of RESTful services, which enables the creation of SAWSDL-like [6] annotations. It has three main elements, which represent links to URIs of semantic concepts and data transformations. The `model` tag indicates that the URI is a link to an ontology entity,

while `lifting` and `lowering` point to links for lifting and lowering transformations between the level of technical descriptions (for example XML, used as a data exchange format) and the level of semantic knowledge (for example RDF, used for semantic-based manipulation such as reasoning). The MicroWSMO microformat is relatively simple but it provides all the elements necessary for attaching semantic information to RESTful service descriptions.

MicroWSMO is complemented by the WSMO-Lite service ontology [7], which specifies the content of the semantic annotations. WSMO-Lite defines four aspects of service semantics including *information model*, *functional semantics*, *behavioral semantics* and *nonfunctional descriptions*, instances of which are linked to the MicroWSMO annotations. In addition to that, it is also used in the same way for describing the content of SAWSDL annotations in WSDL. As a result, both MicroWSMO and SAWSDL can apply WSMO-Lite service semantics and RESTful services can be integrated with WSDL-based ones. Therefore, WSMO-Lite enables unified search over both WSDL-based and RESTful services and tasks such as discovery, composition and mediation can be performed based on WSMO-Lite, completely independently from the underlying Web service technology (WSDL/ SOAP or REST/HTTP).

In summary, the use of MicroWSMO and hRESTS, together with the WSMO-Lite ontology for service semantics, supports the automation of RESTful service tasks. The here presented approach is very lightweight because it relies on the use of microformats, which only enhance existing HTML descriptions with a few simple tags, without modifying the existing visualization. In addition, the annotation process does not require extensive user training or ontology knowledge, and is very intuitive. Currently, there is only little research done in the area of semantic RESTful services and there are no widely accepted alternative methods or approaches. Moreover, there are no available tools, which support users in creating semantic annotations tool support. In order to enable users to practically apply the here presented approach, we introduce SWEET.

3 SWEET

SWEET is a Web application developed using JavaScript and ExtGWT[6], which is started in a Web browser by calling the host URL. It is part of a fully-fledged framework supporting the lifecycle of services, particularly targeted at supporting the creation of semantic RESTful descriptions. SWEET takes as input an HTML Web page describing a Web API and offers functionalities, which enable users to annotate the service properties and to associate semantic information to them.

As it can be seen in Figure 2, the architecture of SWEET consists of three main components, including the visualization component, the data preprocessing component and the annotations recommender. The annotations recommender assists the user in annotating a service by suggesting suitable annotations for

[6] http://extjs.com/products/gxt/

the service as a whole (domain ontology recommendation) and for its individual properties. This component is based on a hybrid recommendation approach combining content based recommendation, implemented by computing similarity measures, between the description of the new service to be annotated and previously annotated services, and ontology-based recommendation. The data preprocessing component implements functionalities for data preparation for the visualization component, caching mechanisms and simple rule-based analysis.

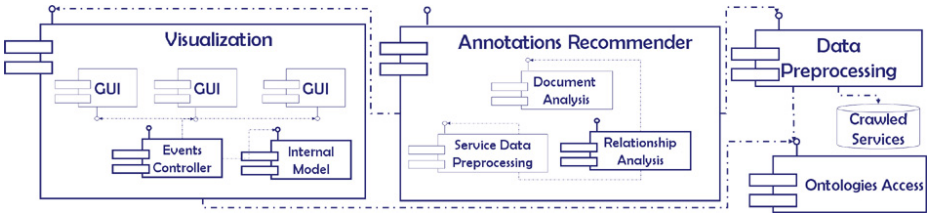

Fig. 2. SWEET Architecture

The GUI of the visualization component is shown in Figure 3 and it has three main panels. The the HTML of the RESTful services is loaded in the *Navigator* panel, which implements a reverse proxy [10] that enables the communication between the annotation functions and the HTML by rerouting all sources and connections from the original HTML through the Web application. Based on this, the HTML DOM of the RESTful service can freely be manipulated by using functionalities of the *Annotation Editor* panel. The current status of the annotation is visualized in the form of a tree structure in the *Semantic Description* panel. It is implemented using the Model-View-Control architecture pattern [10], automatically synchronizing the visualization of the service annotation with an internal model representation, every time the user manipulates it.

In addition to these three main panels, SWEET offers a number of supplementary useful functionalities. It guides the user thorough the process of marking service properties with hRESTS tags, by limiting the available tags depending on the current state of the annotation. This implements measures for reducing possible mistakes during the creation of annotations. In addition, based on the hRESTS tagged HTML, which provides the structure of the RESTful service, the user can link service properties to semantic content. This is done by selecting service properties, searching for suitable domain ontologies by accessing Watson [11] in an integrated way, and by browsing ontology information. Based on this details the user can decide to associate a service property with particular semantic information by inserting a MicroWSMO model reference tag.

While all the main components of SWEET's architecture have complete specifications, including design models and computational methodologies, the annotations recommender is not fully implemented yet. Still, SWEET effectively supports users in creating semantic RESTful descriptions by marking service properties, in searching for suitable ontologies, and in attaching semantic in-

formation. The complete implementation of the annotations recommender will add automation to these tasks, however, the here presented formal approach is fully supported by the current version of SWEET. When the user completes the semantic annotation of the HTML description, the annotated HTML can be saved and republished on the Web, representing an instance of a semantic RESTful service. Moreover, the resulting HTML can also be transformed into a RDF MicroWSMO description, which can be used for manipulation or storage.

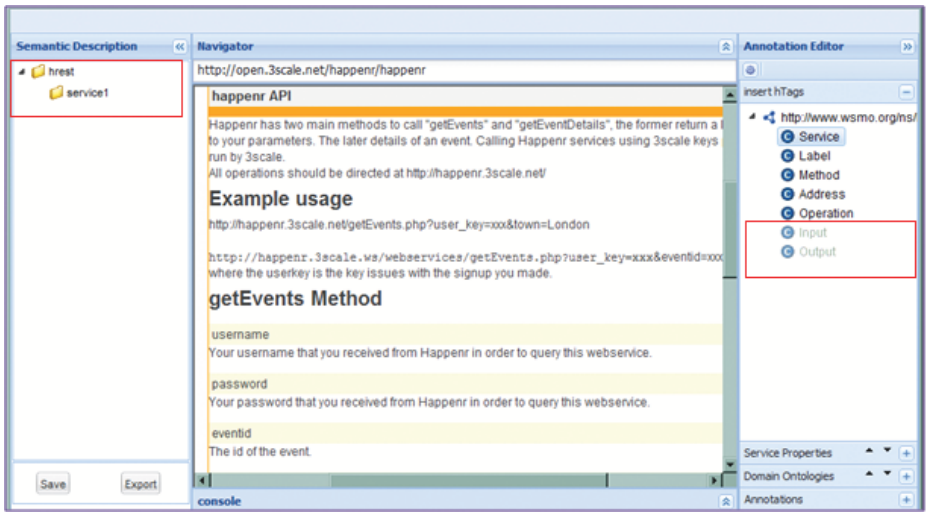

Fig. 3. SWEET: Inserting hRESTS Tags

4 Annotation of RESTful Services with SWEET

This section exemplifies how SWEET supports each of the tasks along the process of creating a semantic RESTful service description. SWEET takes as input the HTML Website description of the RESTful service and returns a semantically annotated version of the HTML or a RDF MicroWSMO description. In order to do this the user needs to complete the following four mains steps:

1. Identifying service properties by inserting hRESTS tags in the HTML service description.
2. Searching for domain ontologies suitable for annotating the service properties.
3. Annotating service properties with semantic information.
4. Saving or exporting the annotated RESTful service.

The first step can easily be completed by simply selecting the part of the HTML, which describes a particular service property, and clicking on the corresponding

tag in the *inset hTags* pane (Figure 3). In the beginning, only the *Service* node of the hRESTS tree is enabled. After the user marks the body of the service, additional tags, such as the *Operation* and *Method*, are enabled. In this way, the user is guided though the process of structuring the RESTful service description and is prevented from making annotation mistakes. The marking of HTML content with a particular hRESTS tag results in the insertion of a corresponding class HTML attribute. This formalism complexity is hidden from the user, and instead, he/she only sees the current status of the annotation reflected in the *Semantic Description* panel. In addition, each inserted tag is highlighted by a custom cascading style sheet (CSS), which visualizes the annotations the user has made. An example of an HTML with identified service properties is given in Listing 1.2.

Listing 1.2. Example hRESTS Service Description

```
<div class="service" id="s1"><h1>happenr API</h1>
<span class="label">Happenr </span>has two main methods to call "getEvents" and ...
<p>All operations should be directed at http://happenr.3scale.net/</p>
<h2>Example usage</h2>
<span class="address">http://happenr.3scale.ws/webservices/getEvents.php?user_key=xxx</span>
<p>where the userkey is the key issues with the signup you made.</p>
<div class="operation" id="op1"><h2><span class="label">getEvents </span>Method</h2>
<span class="input">
<h3>username</h3>
<p>Your username that you received from Happenr in order to query this webservice.</p>
<h3>password</h3>
<p>Your password that you received from Happenr in order to query this webservice.</p>
<h3>eventid</h3>
<p>The id of the event.</p></span></div></div>
```

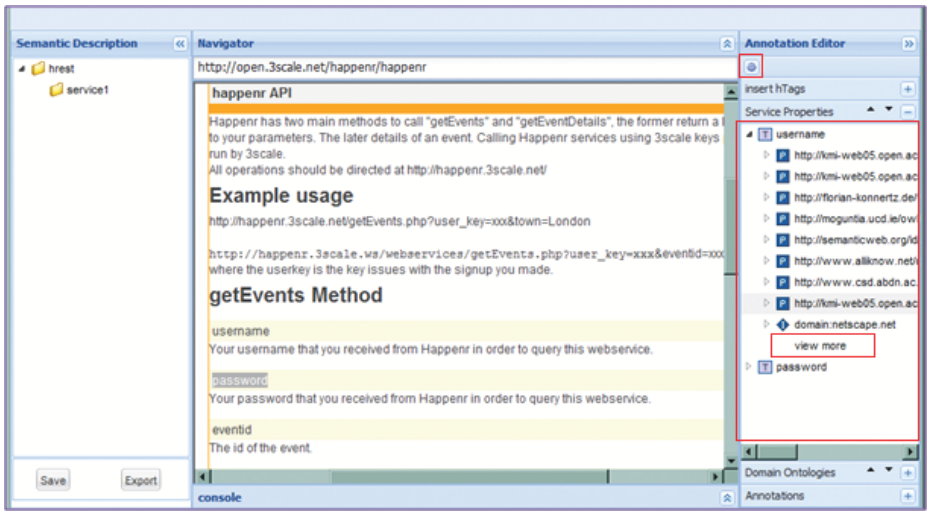

Fig. 4. SWEET: Searching for Suitable Ontologies

After the user structures the HTML description and identifies all service properties, the adding of semantic information can begin. SWEET supports users

in searching for suitable domain ontologies by providing an integrated search with Watson [11]. The search is done by selecting a service property and sending it as a search request to Watson. The result is a set of ontology entities, matching the service property search, which are displayed in the *Service Properties* panel visualized in Figure 4. If the first set of ontology results is insufficient, the user can search for more results by clicking on "view more". In addition, the search is session based and the user preserves his/her ontology search while annotating different service descriptions.

The implementation of the *Service Properties* and *Domain Ontologies* panels supports the user in choosing a suitable ontology for annotating the individual service properties or the complete RESTful service. These supporting functionalities are visualized in Figure 5. The user can view the URI of each of the matching concepts, properties or instances and the corresponding ontology. Additional information is available in the *Domain Ontologies* panel, which shows all service properties that can be annotated with one particular ontology as well as a list of all concepts. The entries of both panels can be expanded or collapsed in order to ease the navigation.

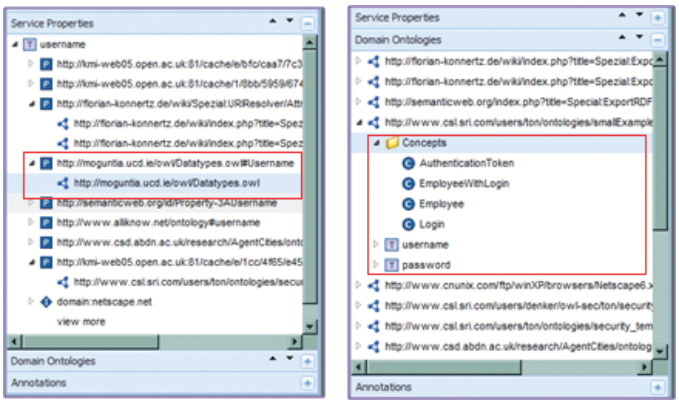

Fig. 5. SWEET: Exploring Domain Ontologies

Once the user has decided, which ontology to use for the service property annotation, he/she can do an annotation by selecting a part of the service HTML description and clicking on *Semantic Annotation* in the *Service Properties* context menu. This results in inserting a `model` attribute and a reference pointing to the URI, of the linked semantic concept. MicroWSMO also contains elements for `lifting` and `lowering`, which point to links for lifting and lowering transformations. Even though, the current version of SWEET does not support the insertion of these elements, they can still be manually added by the user, if necessary. The result is a semantically anntoated HTML description, with inserted `model` and `href` tags marking the association of the particular HTML elements with the semantic concepts. A summary of the already made annotations is given in the *Annotations* panel. These annotations can be removed by choosing

"Delete" from the context menu. In this way, the user can remove incorrect annotations and substitute them with new ones without having to reload the tool and start the annotation process from the very beginning.

Listing 1.3. Example MicroWSMO Service Description

```
1  <div class="service" id="s1"><h1>happenr API</h1>
2  <a rel="model" href="http://example.com/events/getEvents">
3  <span class="label">Happenr </span>has two main methods to call "getEvents" and ...</a>
4  <p>All operations should be directed at http://happenr.3scale.net/</p>
5  <h2>Example usage</h2>
6  <span class="address">http://happenr.3scale.ws/webservices/getEvents.php?user_key=xxx</span>
7  <p>where the userkey is the key issues with the signup you made.</p>
8  <div class="operation" id="op1"><h2><span class="label">getEvents </span>Method</h2>
9  <span class="input">
10 <h3><a rel="model" href="http://example.com/data/onto.owl#Username">username</a>
11 (<a rel="lowering" href="http://example.com/data/event.xsparql">lowering</a>)</h3>
12 <p>Your username that you received from Happenr in order to query this webservice.</p>
13 <h3><a rel="model" href="http://example.com/data/onto.owl#Password">password<a>
14 (<a rel="lowering" href="http://example.com/data/event.xsparql">lowering</a>)</h3>
15 <p>Your password that you received from Happenr in order to query this webservice.</p>
```

Listing 1.3 shows our example service description annotated with MicroWSMO by using SWEET. Line 2 uses the `model` relation to indicate that the service searches for events, while line 10 associates the input parameter *username* with the class *Username*. The lowering schema for the recipient is also provided in line 11.

SWEET also provides options for customizing the way service descriptions are viewed. First, if the *Navigator* panel displays HTML service descriptions, which already contain MicroWSMO elements, these elements will be recognized and automatically highlighted so that the user can manipulate them and integrate them in his/her own annotation of the service. Second, the way the service properties and semantic information is highlighted can be modified by simply substituting the current CSS file with a new one, which uses different text font and colors.

In summary, SWEET effectively supports users in creating semantic RESTful service descriptions by using the hRESTS and the MicroWSMO microformats. In particular, it provides functionalities for marking service properties by inserting tags, for searching for suitable domain ontologies, for linking service properties with semantic concepts and for saving and exporting the resulting RESTful description both as annotated HTML or directly as RDF. In this way, SWEET contributes to a higher level of automation of common service tasks, such as discovery, composition and invocation.

5 Retrieval of Annotated RESTful Services

The benefits for the retrieval of services by using MicroWSMO for RESTful service annotation are threefold. First, the use of the microformats enables that HTML RESTful service descriptions can be distinguished from simple HTML websites and can be automatically collected. Second, the service search itself can be improved and automated by using the attached semantic information. Finally,

since MicroWSMO provides SAWSDL-like annotations of RESTful services and relies on the WSMO-Lite service ontology, both WSDL-based and RESTful services can be retrieved by using the same queries. As a result, all type of services, whether WSDL-based or RESTful can be retrieved in a unified way.

Listing 1.4 shows an example query for retrieving all services that use "Username". This will retrieve both WSDL services with SAWSDL annotations and RESTful services with MicroWSMO annotations. As are results both types will effectively be discovered only based on the semantic information and they can be used interchangeably in common compositions.

Listing 1.4. Example Service Query

```
1 SELECT DISTINCT ?s
2 WHERE {
3 ?s rdf:type wsl:Service .
4 ?s sawsdl:modelReference <http://example.com/data/onto.owl#Username>
5 }
```

6 Related Work

Current research in the area of semantic RESTful services is mostly focused around the definition of formalisms for creating semantic annotations. As already mentioned, MicroWSMO is one such formalism, which relies on hRESTS for describing the main aspects of a service such as its operations, inputs and outputs, and uses hooks for linking these to semantic information. SA-REST [12], on the other hand, uses the grounding principles of SAWSDL [6] and RDFa for marking service properties. Even though, there is some research done targeted at supporting the use of SRS, for example in the form of mashups [12], there are no existing tools or approaches supporting the creation of semantic RESTful service descriptions, which therefore hinders the applicability.

hRESTS is not the only alternative that can be used for the creation of machine-readable descriptions of RESTful services. WADL (Web Application Description Language) [9] and even WSDL 2.0 [8] can be used as description formats. They provide well-structured and detailed forms of descriptions. However, probably due to the user-centered context of Web 2.0 and of the resulting API descriptions, WADL and WSDL seem to add complexity and still the majority of the API descriptions are provided in unstructured text. We use hRESTS, which is relatively simple, easy to use, can be applied directly on the existing HTML descriptions, supports the extraction of RDF and can provide a basis, for the future adoption of dedicated formats such as WADL.

Another description approach if offered by RDFa [13]. RDFa can be effectively used for embedding RDF data in HTML. However, following the simplicity and lightweight principles perused with hRESTS, it needs to be investigated to what extent and in which use cases RDFa can be used for hRESTS. A parallel approach to RDFa would be the use of GRDDL [14] on top of hRESTS. GRDDL is a mechanism for extracting RDF information from Web pages and is particularly suitable for processing microformats.

In the area of tools supporting the semantic annotation of services, ASSAM [15] enables the annotations of services with WSDL-based descriptions. It provides user interface tools as well as some automatic recommendation components, however, it can only be used on WSDL-based descriptions and does not support RESTful services.

7 Conclusion and Future Work

Currently, RESTful services are becoming more and more popular, however, their general adoption is hindered by the fact that they cannot be found, interpreted and invoked without the extensive user involvement and a multitude of manual tasks. This challenges can be addressed through the creation of machine-readable descriptions, which server as the basis for automatically finding services, through crawlers and search engines, and processing them. Moreover, extended with semantic annotations, RESTful services can even be discovered, composed and invoked automatically, following the principles of the SWS.

In this paper, we have presented an integrated lightweight approach for formally describing semantic RESTful services. It is based on two microformats: the hRESTS microformat that enables the tagging of key service properties and therefore supports the creation of machine-readable service descriptions; and the MicroWSMO microformat that enables the linking of semantic information to service properties. The approach is facilitated by SWEET, a tool which effectively supports users in creating semantic descriptions of RESTful services, by providing functionalities for both the creation of machine-readable descriptions and the addition of semantic annotations based on hRESTS and MicroWSMO.

Future work will focus on further developing SWEET's annotations recommender component, in order to reduce the number of manual tasks that the user has to complete. This will result in the faster and more correct creation of semantic RESTful service descriptions. Future work will also include the development of supplementary functionalities of SWEET, which will provide additional user support. Better visualization components, such as structure and properties highlighting are planned. In addition, some work will be devoted to the automatic recognition of service properties such as operations and input parameters, so that the user only has to verify the pre-marked service properties. The goal is that future versions of SWEET will even better support users in creating semantic RESTful service descriptions.

8 Acknowledgments

SWEET is based upon work partially supported by the EU funding under the project SOA4All (FP7 - 215219). The authors would like to thank Simone Spaccarotella for his contribution to the development of SWEET.

References

1. L. Richardson, S. Ruby: RESTful Web Services. O'Reilly Media, May 2007.
2. R. T. Fielding: Architectural styles and the design of network-based software architectures. PhD thesis, University of California, 2000.
3. J. Kopecký , K. Gomadam, T.Vitvar: hRESTS: an HTML Microformat for Describing RESTful Web Services. Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence (WI-08), 2008.
4. R. Khare, T. Celik: Microformats: a pragmatic path to the semantic web (Poster). Proceedings of the 15th international conference on World Wide Web, 2006.
5. J. Kopecký, T. Vitvar, D. Fensel, K. Gomadam: hRESTS & MicroWSMO. Technical report, available at `http://cms-wg.sti2.org/TR/d12/`, 2009.
6. J. Kopecký, T. Vitvar, C. Bournez, J. Farrel. SAWSDL: Semantic Annotations for WSDL and XML Schema. IEEE Internet Computing, 11(6):60-67, 2007.
7. T. Vitvar, J. Kopecký, J. Viskova, and D. Fensel. WSMO-Lite Annotations for Web Services. In the Semantic Web: Research and Applications, ESWC 2008.
8. Web Services Description Language (WSDL) Version 2.0. Recommendation, W3C, June 2007. Available at `http://www.w3.org/TR/wsdl20/`.
9. M. J. Hadley: Web Application Description Language (WADL). Technical report, Sun Microsystems, November 2006. Available at `https://wadl.dev.java.net`.
10. E. Gamma, R. Helm, R. Johnson, J. M. Vlissides: Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, November 1994.
11. Watson - The Semantic Web Gateway: Ontology Editor Plugins. `http://watson.kmi.open.ac.uk`. Online November 2008.
12. A. P. Sheth, K. Gomadam, J. Lathem: SA-REST: Semantically Interoperable and Easier-to-Use Services and Mashups. In IEEE Internet Computing, 11(6):9194, 2007.
13. RDFa in XHTML: Syntax and Processing. Proposed Recommendation, W3C, September 2008. Available at `http://www.w3.org/TR/rdfa-syntax/`.
14. Clarke, F., Ekeland, I.: Gleaning Resource Descriptions from Dialects of Languages. Recommendation, W3C, September 2007. `http://www.w3.org/TR/grddl/`.
15. A. Hess, E., Johnston, N., Kushmerick: ASSAM: A tool for semi-automatically annotating semantic web services. In Proceedings of International Semantic Web Conference, 2004.