Semantic Annotation of Web APIs with SWEET

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. Recently technology developments in the area of services on the Web are marked by the proliferation of Web applications and APIs. The development and evolution of applications based on Web APIs is, however, hampered by the lack of automation that can be achieved with current technologies. In this paper we present SWEET– Semantic Web sErvices Editing Tool– a lightweight Web application for creating semantic descriptions of Web APIs. SWEET directly supports the creation of mashups by enabling the semantic annotation of Web APIs, thus contributing to the automation of the discovery, composition and invocation service tasks. Furthermore, it enables the development of composite SWS based applications on top of Linked Data.

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, technology developments in the area of services on the Web are marked by the proliferation of Web applications and APIs. The development and evolution of applications based on Web APIs is, however, hampered by the lack of automation that can be achieved with current technologies. Research on semantic Web services is therefore trying to adapt the principles and technologies that were devised for traditional Web services, to deal with this new kind of services.

Web APIs 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. Many Web 2.0 applications like Facebook, Google, Flickr and Twitter offer easy-to-use Web APIs, which not only provide simple access to different resources but also enable combining heterogeneous data coming from diverse sources, in order to create mashups. However, despite their success, Web APIs 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 Web APIs, even though there are some initial approaches in the area [8], [9]. As a consequence, in order to use Web APIs, developers are obliged to manually locate, retrieve, read and interpret heterogeneous documentations commonly given in HTML, and subsequently develop custom tailored software that is able to invoke and manipulate these Web APIs.

In order to support the creation of mashups and the more automated handling of Web APIs in general, we have developed SWEET[1] – Semantic Web sErvices Editing Tool. SWEET is a lightweight Web application, which requires no installation and can be invoked directly in the Web browser. It is realized by using JavaScript and ExtGWT[2] and supports the development of mashups based on linked open data and services, by enabling the creation of semantic descriptions of Web APIs. In particular, SWEET can be used to annotate Web APIs, which results in semantic descriptions that are amenable to automated reasoning and can therefore support a higher level of automation during the discovery, composition and invocation of Web APIs. As a consequence SWEET contributes to better supporting the creation of mashups and applications based on linked open data and Web APIs.

The remainder of this paper is structured as follows: Section 2, provides an overview of the existing formalisms for the semantic description of Web APIs, while Section 3 exemplifies how semantically annotated Web APIs can contribute to the scalability and flexibility of the creation of mashups. Section 4 introduces SWEET, including its components, functionalities and user support. An example for creating semantic descriptions of Web APIs is given in Section 5. Section 6 presents an overview of comparable tools, related formalisms and approaches. Finally, Section 7 presents future work that will be carried out and concludes the paper.

2 Semantic Annotation of Web APIs

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 [1], when conforming to the REST architectural style [2]. The majority of the Web API 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. Since currently most Web applications and Web APIs rely only on HTML documentation, with no fixed structure or content, we use the hRESTS [3] microformat [4], which enables the creation of machine-processable descriptions on top of existing HTML descriptions.

hRESTS 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 hRESTS in particular, uses `class` and `rel`

[1] `http://sweetdemo.kmi.open.ac.uk/war/MicroWSMOeditor.html`

[2] ExtGWT is a Java library for building rich internet applications with Google Web Toolkit (GWT). `http://extjs.com/products/gxt/`

XHTML attributes to mark key service properties (see Listing 1.1), 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. Therefore, hRESTS marks the key properties of the Web API and provides a machine-readable description based on the available HTML documentation. The result can be used as the basis for adding complementary information and annotations.

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 Web API descriptions.

In summary, we use MicroWSMO and hRESTS, to support the automation of Web API-related tasks, such as discovery, composition and invocation. 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. It only relies on the use of incremental extensions, which results in a low overhead of its uptake and practical use. In addition, the annotation process does not require extensive user training or ontology knowledge, and is very intuitive. In particular, SWEET supports users in creating semantic annotations of Web APIs by abstracting away from the technical details and assisting in the location of suitable domain ontologies from the Web.

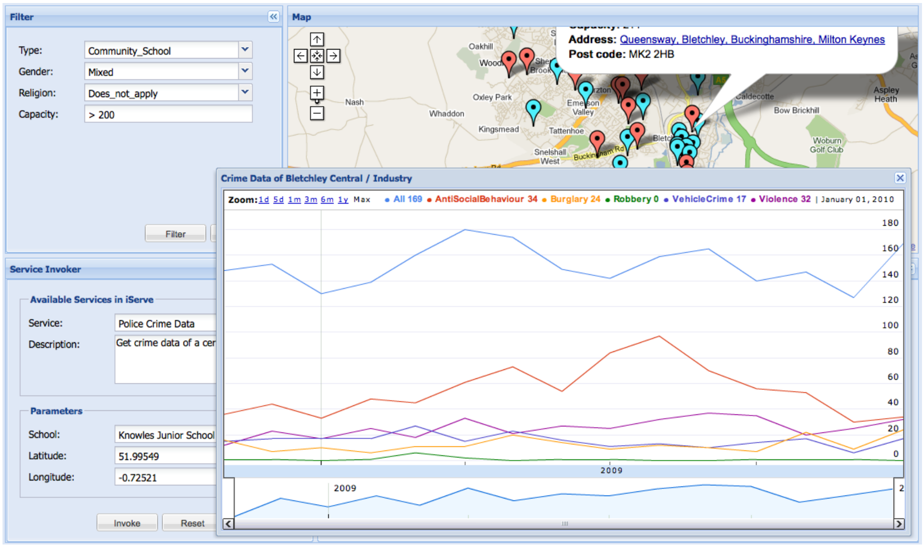
3 Supporting the Creation of Mashups

In this section we describe a data-oriented composition, i.e. a mashup, that we implemented in order to exemplify the importance of the semantic annotation of Web APIs for the creation of compositions, based on linked open data and Web APIs.

The availability of linked open data, such as the one published by the open government data initiative[3], provides the foundation for the development of diverse mashups, where distinct sources can be combined to provide added-value solutions. Moreover, these data compositions can be enhanced by integrating the existing Web APIs as additional data sources. As a result, Web applications can be created to display or synthesize linked open data and data retrieved through Web APIs.

[3] `http://data.gov.uk/`

Undoubtedly such applications can be very useful and there are already a number of existing implementations, the most common ones probably being the visualization of data by using the Google Maps API. However, currently, a developer needs to manually search for suitable Web APIs, implement the data composition and realize its invocation. The result is a mashup based on open data and Web APIs. However, applications created in this way have very restricted potential when it comes to reusability, maintenance and extensibility.


Filter

Type: Community_School

Gender: Mixed

Religion: Does_not_apply

Capacity: > 200

Service Invoker

Available Services in iServe

Service: Police Crime Data

Description: Get crime data of a cer

Parameters

School: Knowles Junior School

Latitude: 51.99549

Longitude: -0.72521

Map

Address: Queensway, Bletchley, Buckinghamshire, Milton Keynes

Post code: MK2 2HB

Crime Data of Bletchley Central / Industry

● All 169 ● AntiSocialBehaviour 34 ● Burglary 24 ● Robbery 0 ● VehicleCrime 17 ● Violence 32 | January 01, 2010

Fig. 1. Combining Local Schools Data with a Web API for Crime Statistics

In order to enable the reusability of Web APIs within mashups and to add a higher level of automation to the composition process, we propose the use of semantic descriptions of Web APIs. Figure 1 visualizes a Web application implemented by combining open government data about schools in Milton Keynes with APIs for retrieving location-based crime statistic and visualizing it together in Google Maps. However, instead of hard-coding the implementation, we use the latitude and longitude as search parameters for finding relevant Web APIs. In this way, based on the semantic descriptions, we are able to discover and provide a set of services, which take as input latitude and longitude and return location-based information such as crime statistic or local businesses.

In order to support the creation of semantic descriptions of Web APIs and contribute to the flexible and easier creation of mashups, we use SWEET.

4 SWEET

SWEET (`http://sweet.kmi.open.ac.uk/`) is a Web application developed using JavaScript and ExtGWT, which is started in a Web browser by calling the host URL. It implementation is mainly based on using scripting programming languages and is part of a fully-fledged framework supporting the lifecycle of services, particularly targeted at enabling the creation of semantic descriptions of Web APIs. 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. In comparison to previously published versions of SWEET [16] this version of the tool has some additional features, is more stable, uses a new API of Watson for searching ontologies and has proven to support the creation of composite Web applications based on joining linked open data and semantically annotated Web APIs.

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 the service as a whole (domain ontology recommendation) and for its individual properties. This component is still under development and will be available soon. The data preprocessing component implements functionalities for data preparation for the visualization component, caching mechanisms and simple rule-based analysis.

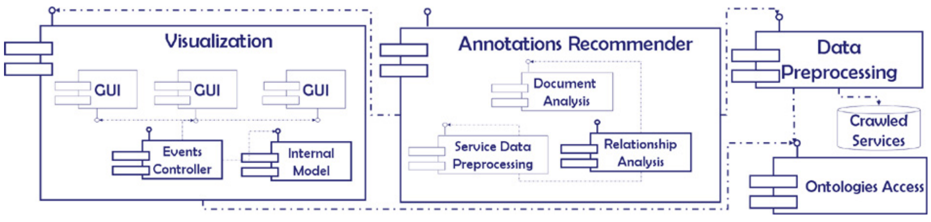
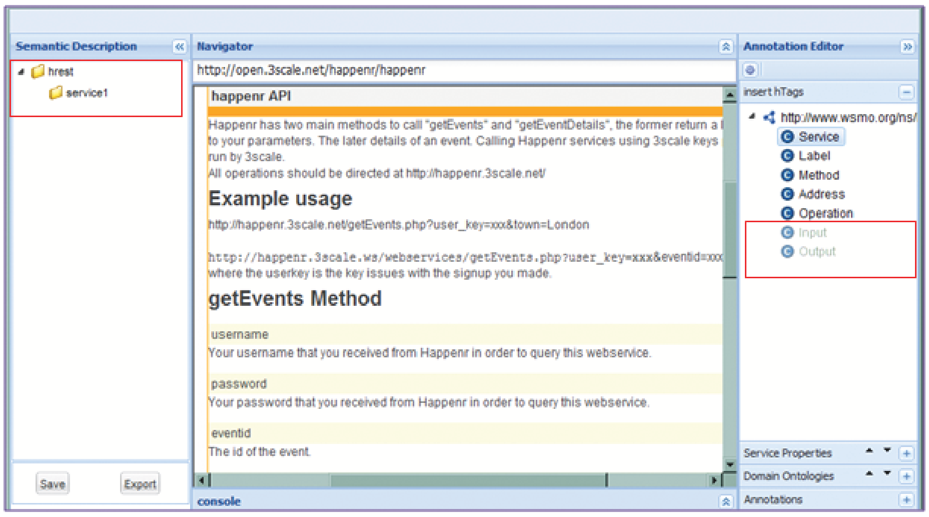


Fig. 2. SWEET Architecture

The GUI of the visualization component is shown in Figure 3 and it has three main panels. The the HTML description of the Web API 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. The result is a local representation of the HTML, which is used as a basis for the annotation process. In this way, the HTML DOM can freely be manipulated and different HTML tags can be added 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 Web API, 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. In particular, Watson assist users in locating appropriate annotations and enables the accessing of existing ontological data on the Web, in order to increase the level of reusability of services through ontology reuse. The search results include matching concepts, properties and instances from different ontologies. Based on this details the user can decide to associate a service property with particular semantic information by inserting a MicroWSMO model reference tag.



http://open.3scale.net/happenr/happenr

happenr API

Happenr has two main methods to call "getEvents" and "getEventDetails", the former return a
to your parameters. The later details of an event. Calling Happenr services using 3scale keys
run by 3scale.
All operations should be directed at http://happenr.3scale.net/

Example usage

http://happenr.3scale.net/getEvents.php?user_key=xxx&town=London

http://happenr.3scale.ws/webservices/getEvents.php?user_key=xxx&eventid=xxx
where the userkey is the key issues with the signup you made.

getEvents Method

username
Your username that you received from Happenr in order to query this webservice.

password
Your password that you received from Happenr in order to query this webservice.

eventid
The id of the event.

Fig. 3. Inserting hRESTS tags with SWEET.

SWEET effectively supports users in creating semantic descriptions of Web APIs by marking service properties, by searching for suitable ontologies, and by attaching semantic information. When the user completes the semantic annotation of the HTML description, the Web API annotation can be locally saved as annotated HTML or as extracted RDF. In addition, both the annotated HTML and the RDF can be directly published to the iServe[4] [17] repository for semantic Web service descriptions. iServe can then be used to discover services, which can effectively be integrate in data mashups.

5 Annotation of Web APIs with SWEET

This section exemplifies how SWEET supports each of the tasks along the process of creating semantic descriptions of Web APIs. SWEET takes as input the

[4] `http://iserve.kmi.open.ac.uk/`

HTML Website describing the Web API and returns a semantically annotated version of the HTML or an RDF MicroWSMO description. In order to do this the user needs to complete the following four mains steps:

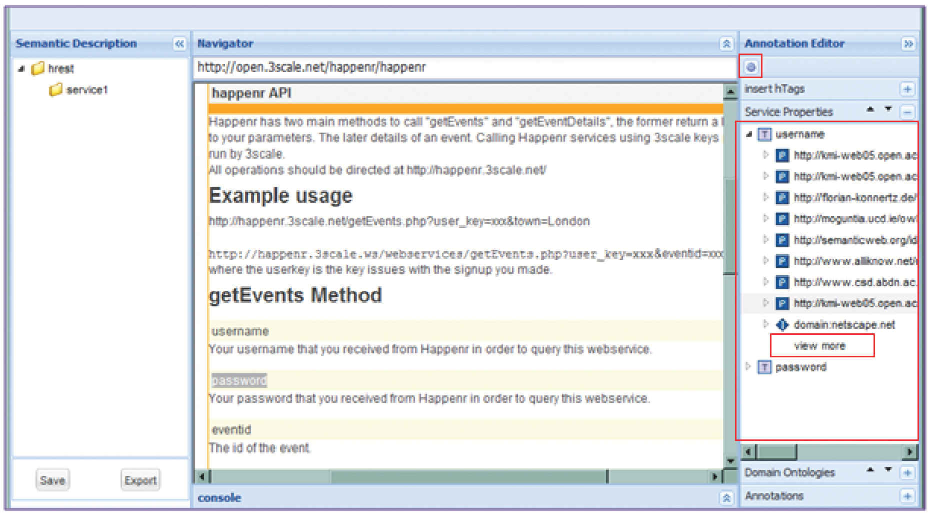
1. Identifying service properties by inserting hRESTS tags in the HTML service description.
2. Searching for domain ontologies suitable for annotating the service properties.
3. Annotating service properties with semantic information.
4. Saving or exporting the annotated Web API.

The first step can easily be completed by simply selecting the part of the HTML, which describes a particular service property, and double-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 through the process of structuring the Web API description and we limit the potential mistakes that a user could make, thus avoiding the generation of incorrect service description structures. 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 1.1.

Listing 1.1. 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>
```

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



happenr API

Happenr has two main methods to call "getEvents" and "getEventDetails", the former return a l
to your parameters. The later details of an event. Calling Happenr services using 3scale keys
run by 3scale.
All operations should be directed at http://happenr.3scale.net/

Example usage

http://happenr.3scale.net/getEvents.php?user_key=xxx&town=London

```
http://happenr.3scale.ws/webservices/getEvents.php?user_key=xxx&eventid=xxx
```

where the userkey is the key issues with the signup you made.

getEvents Method

username
Your username that you received from Happenr in order to query this webservice.

password
Your password that you received from Happenr in order to query this webservice.

eventid
The id of the event.

Fig. 4. SWEET: Searching for Suitable Ontologies

is session based and the user preserves his/her ontology search while annotating different service descriptions.

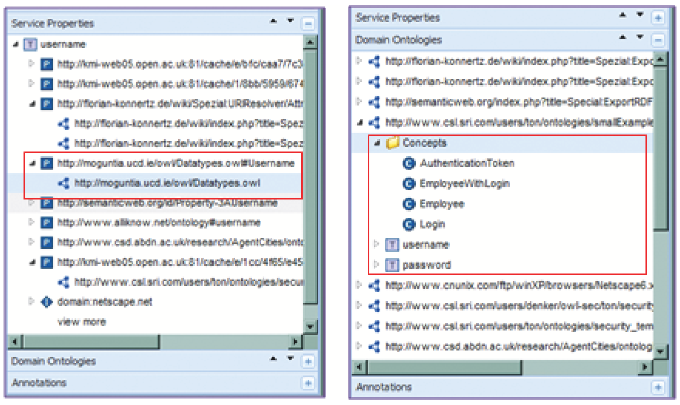

Service Properties
username
http://kmi-web05.open.ac.uk:81/cache/e/bfc/caa7/7c3
http://kmi-web05.open.ac.uk:81/cache/1/8bb/5959/674
http://florian-konnertz.de/wiki/Spezial:URIResolver/Attr
http://florian-konnertz.de/wiki/index.php?title=Spez
http://florian-konnertz.de/wiki/index.php?title=Spez
http://moguntia.ucd.ie/owl/Datatypes.owl#Username
http://moguntia.ucd.ie/owl/Datatypes.owl
http://semanticweb.org/id/Property-3AUsername
http://www.allknow.net/ontology#username
http://www.csd.abdn.ac.uk/research/AgentCities/ont
http://kmi-web05.open.ac.uk:81/cache/e/1cc/4f65/e45
http://www.csl.sri.com/users/ton/ontologies/secu
domain:netscape.net
view more
Domain Ontologies
Annotations
Service Properties
Domain Ontologies
http://florian-konnertz.de/wiki/index.php?title=Spezial:Expo
http://florian-konnertz.de/wiki/index.php?title=Spezial:Expo
http://semanticweb.org/index.php?title=Special:ExportRDF
http://www.csl.sri.com/users/ton/ontologies/smallExample
Concepts
AuthenticationToken
EmployeeWithLogin
Employee
Login
username
password
http://www.cnunix.com/ftp/winXP/browsers/Netscape6.x
http://www.csl.sri.com/users/denker/owl-sec/ton/securit
http://www.csl.sri.com/users/ton/ontologies/security_tem
http://www.csd.abdn.ac.uk/research/AgentCities/ontolog
Annotations

Fig. 5. SWEET: Exploring Domain Ontologies

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 Web API. 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.

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 entity. 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. The result is a semantically annotated HTML description, with inserted `model` and `href` tags marking the association of the particular HTML elements with the semantic entities. 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 1.2. 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 1.2 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. The annotated HTML of the Web API can be directly transformed into RDF (Listing 1.3), which serves as the basis for discovery and manipulation directly over the semantic properties.

Listing 1.3. Example RDF Service Description

```
1  <rdf:RDF><wsl:Service>
2  <rdfs:isDefinedBy rdf:resource="http://open.3scale.net/happenr/happenr#code"/>
3  <sawsdl:modelReference rdf:resource="http://example.com/events/getEvents"/>
4  <rdfs:label>Happenr</rdfs:label>
5  <hr:hasAddress rdf:datatype="http://www.wsmo.org/ns/hrests#URITemplate">
6  http://happenr.3scale.ws/webservices/getEvents.php?user_key=xxx</hr:hasAddress>
7  <wsl:hasOperation><wsl:Operation><rdfs:label>getEvents </rdfs:label>
8  <wsl:hasInputMessage><wsl:Message>
9  <sawsdl:modelReference rdf:resource="http://example.com/data/onto.owl#Username"/>
10 <sawsdl:modelReference rdf:resource="http://example.com/data/onto.owl#Password"/>
11 <sawsdl:loweringSchemaMapping rdf:resource="http://example.com/data/event.xsparql" />
12 </wsl:Message></wsl:hasInputMessage>
13 </wsl:Operation></wsl:hasOperation></wsl:Service></rdf:RDF>
```

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 fonts and colors.

In summary, SWEET effectively supports users in creating semantic descriptions of Web APIs 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 entities and for saving and exporting the resulting semantic 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.

6 Related Work

Current research in the area of Web APIs and, in particularly on 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 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 semantic descriptions, for example in the form of mashups [12], there are no existing tools or approaches supporting the creation of semantic descriptions of Web APIs, which therefore hinders the applicability.

hRESTS is not the only alternative that can be used for the creation of machine-readable descriptions of Web APIs. WADL (Web Application Description Language) [9] and even WSDL 2.0 [8] can be used as description formats. 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. Another description approach is offered by RDFa [13]. RDFa can be effectively used for embedding RDF data in HTML. 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. There are also a number of tools and approaches developed for enhancing Web content with semantics in general. Tabulator [19] supports the modification and addition of information directly, however, its use can be a bit challenging for people unfamiliar with the RDF data model. Loomp [20] aims at providing a user interface for both creating textual content as well as annotating this content using semantic representations. Finally, in [18] the authors introduce a JavaScript API that allows the independent creation of editing widgets for embedded RDFa.

7 Conclusion and Future Work

Currently, Web APIs are becoming increasingly 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. Web APIs play a particularly important task in the context of mashups, where existing linked open data can be combined with related services in order to develop applications, which facilitate a synergy effect over the combined information.

The challenges faced by Web APIs 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, Web APIs can even be discovered and included in compositions automatically, following the principles of the SWS.

In this paper, we have presented SWEET, which facilitates an integrated lightweight approach for creating semantic descriptions of Web APIs. These semantic descriptions serve as the basis for creating more flexible mashups composed of linked open data and Web APIs, as demonstrated by our example application. SWEET uses 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. As a result, SWEET effectively supports users in creating semantic descriptions of Web APIs, 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 descriptions of Web APIs. Future work will also include the development of supplementary functionalities of SWEET, which will provide additional user support. 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 descriptions of Web APIs.

8 Acknowledgments

SWEET is based upon work partially supported by the EU funding under the project SOA4All (FP7 - 215219).

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. F. Clarke, I. Ekeland: 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.
16. M. Maleshkova, C. Pedrinaci, J. Domingue: Supporting the Creation of Semantic RESTful Service Descriptions. Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2) at 8th International Semantic Web Conference, 2009.
17. C. Pedrinaci, J. Domingue, R. Krummenacher: Services and the Web of Data: An Unexploited Symbiosis. Workshop on Linked Data Meets Artificial Intelligence at AAAI Spring Symposium, 2010.
18. S. Dietzold, S. Hellmann, M. Peklo. Using JavaScript RDFa Widgets for Model/View Separation inside Read/Write Websites. In Proceedings of the 4th Workshop on Scripting for the Semantic Web, 2008.
19. T. Berners-Lee, J. Hollenbach, K. Lu, J. Presbrey, E. Pru d'ommeaux, and m.c. schraefel. Tabulator Redux: Writing Into the Semantic Web. Technical Report ECSIAMeprint14773, Electronics and Computer Science, University of Southampton.
20. M. Luczak-Roesch, R. Heese. Linked Data Autoring for non-Experts. In Workshop on Linked Data on the Web, 2009, Madrid, 2009.