RDF Editing on the Web

Claus Stadler
Department of Computer Science, University of Leipzig
cstadler@informatik.uni-leipzig.de

Natanael Arndt
Department of Computer Science, University of Leipzig
arndt@informatik.uni-leipzig.de

Michael Martin
Department of Computer Science, University of Leipzig
martin@informatik.uni-leipzig.de

Jens Lehmann
Department of Computer Science, University of Leipzig
lehmann@informatik.uni-leipzig.de

ABSTRACT
While several tools for simplifying the task of visualizing (SPARQL accessible) RDF data on the Web are available today, there is a lack of corresponding tools for exploiting standard HTML forms directly for RDF editing. The few related existing systems roughly fall in the categories of (a) applications that are not aimed at being reused as components, (b) form generators, which automatically create forms from a given schema – possibly derived from instance data – or (c) form template processors which create forms from a manually created specification. Furthermore, these systems usually come with their own widget library, which can only be extended by wrapping existing widgets. In this paper, we present the AngularJS-based *Rdf Edit eXtension* (REX) system, which facilitates the enhancement of standard HTML forms as well as many existing AngularJS widgets with RDF editing support by means of a set of HTML attributes. We demonstrate our system though the realization of several usage scenarios.

1. INTRODUCTION
RDF is designed to provide a uniform data model for representing knowledge across domain boundaries. However, sustaining this flexibility in the context of RDF editing systems has turned out to be a difficult task. While there are Web *applications* for editing RDF data, such as WebProtege[1], there is a lack of *components* for the light-weight integration of such capabilities into Web applications. In this paper, we present a novel approach to RDF authoring, which, to the best of our knowledge, delivers unprecedented flexibility and ease in converting HTML form data to RDF and demonstrate its applicability in various use cases. Our solution is based on using the popular AngularJS framework[2] to define a custom set of HTML attributes which add an RDF context to form controls in the DOM-tree. While this approach is similar to RDFa[3], there are two fundamental differences: First, RDFa is intended for *static* annotation of HTML documents, whereas our annotations are processed by AngularJS and our framework at runtime based on the applications' state. Second, RDFa is aligned with the HTML standard, whereas our approach exploits concepts of AngularJS and builds upon its abstractions, most prominently the one introduced by *ngModel*[4] Also, a key distinguishing feature that contrasts current related work is, that our system does not introduce a widget framework of its own. Instead, by building on the AngularJS framework and its abstractions, many already available widgets can be re-used directly *without the need of wrapping*.

Our contributions in this paper are twofold: First, we present the *Rdf Edit eXtension system* (REX), which makes HTML forms RDF-aware using a set of custom HTML attributes. Second, we devise several usage patterns of our system for solving common problems related to RDF editing.

Our RDF editing system is implemented as a user interface module as part of our *Ja*vascript *S*uite for *S*PARQL *A*ccess (Jassa) library [5]. The module is freely available on Github[5] and is registered for use as a dependency with the popular *bower* tool[6] under the name `jassa-ui-angular-edit`. Further, demos are available online[7].

The structure of this paper is as follows: In Section 2, we introduce our REX approach and discuss its use in several scenarios. A brief outline of use cases is given in Section 3. In Section 4, we review existing approaches to RDF authoring and presentation on the Web and discuss their strengths and weaknesses. Finally, in Section 5 we conclude this paper.

[1]`http://protegewiki.stanford.edu/wiki/WebProtege`

[2]`https://angularjs.org/`
[3]`http://www.w3.org/TR/rdfa-syntax/`
[4]`https://docs.angularjs.org/api/ng/directive/ngModel`
[5]`https://github.com/GeoKnow/Jassa-UI-Angular/tree/master/jassa-ui-angular-edit`
[6]`http://bower.io/`
[7]`http://js.geoknow.eu/demos/rex/`

2. RDF EDIT EXTENSIONS

In this section we, explain our REX annotation system for extending HTML forms with support for RDF editing. We introduce our approach using a simple example, then explain what types of data REX manages and finally we present several patterns for solving common problems. Note, that our annotations can also be used for retrieving data intended for display. However, REX is focused on the support of bidirectional mapping of values between an RDF graph and the model of form controls. Thus, support for arbitrary SPARQL queries and post-processing of result sets, such as chosing the best label in regard to a preferences on vocabularies and languages is not strictly part of the core.

It is important to understand, that AngularJS makes it possible to annotate HTML form controls with `ng-model="foo"`, which declares a JavaScript variable `foo` as the model of the control. When the user enters input, the model's value is changed accordingly, and when the model changes, AngularJS automatically updates the form control to reflect the new state. Our approach is based on the introduction of a set of annotations which give model values an RDF context. In the subsequent sections we explain the REX attributes.

2.1 Core HTML Attributes

Consider the example in Listing 1. First of all, the `rex-context` is used to activate the REX system on a DOM element and its descendants. The presence of the attribute creates the *rexContext* object in the corresponding scope, whose purpose is to keep track of (a) RDF values referenced in form controls, (b) initial data for prefilling forms, (c) modifications to the initial data for computing diffs, and (d) the final RDF data.

```
<div rex-context
  rex-prefix="dbr: http://dbpedia.org/resource/"
  rex-subject="dbr:Chemnitz">
    <input type="text"
      ng-model="cityName"
      rex-predicate="rdfs:label"
      rex-object rex-type="literal"
      rex-value="cityName">
</div>
```

Listing 1: A minimal REX example

Within a `rex-context` section, the basic annotations are: `rex-subject`, `rex-predicate`, and `rex-object`. While rex-subject and rex-predicate accept strings with markup[8] for IRIs as values, rex-object takes an *index*. The index refers to the *i*th RDF term for a given subject and predicate. Internally, the Talis RDF JSON[9] model is used for representing RDF data. Hence, RDF objects comprise four *components* of type string, that can be accessed using the annotations `rex-type`, `rex-value`, `rex-lang`, and `rex-datatype`. In addition, `rex-deleted` can be used to flag an object at a certain index – and thus the corresponding triple – as deleted. We refer to the combination of a subject IRI, predicate IRI, object index and component name as a *coordinate*. A coordinate uniquely references a primitive value in an RDF graph.

[8]`https://docs.angularjs.org/api/ng/service/$interpolate`

[9]`http://www.w3.org/TR/rdf-json/`

Note, that the Talis RDF JSON is used as an intermediate representation. The *final* RDF graph, that corresponds to the form state, can be serialized in any RDF syntax, including JSON-LD[10].

For convenience, prefixes can be used to abbreviate IRIs. By default, all prefixes of the RDFa initial context[11] are readily available. Custom namespaces can be enabled by specifying `rex-prefix`, where the argument must follow the same syntax as in RDFa[12]. Also, REX features the short hands `rex-iri="m"` and `rex-literal="m"`, which are expanded to `rex-type="'uri'"`, `rex-value="m"` and `rex-type="'literal'"` and `rex-value="m"`, respectively.

Whenever multiple attributes occur on the same element, the order of processing depends on their priority value. For details please refer to the documentation on Github.

2.2 Accessing the Data and Modifications

The rexContext object is a container for all relevant information about the state of the form and performed modifications. It provides the following attributes:

- *.base* The base RDF graph, in Talis RDF JSON, which holds information about the resources referenced by the form controls. This information is automatically updated accordingly when the set of subject resources and or lookup functions change.
- *.override* A *(partial)* Talis RDF JSON object which holds the state of the data entered into the form and which can be seen to "override" any base data.
- *.graph* The RDF graph that holds the *effective* set of triples. This is thus obtained by applying the overrides to the base data for all data referenced by coordinates of the form.
- *.diff.added* and *.diff.removed* RDF graphs[13] containing the triples that were added/removed in regard to the base data and referenced coordinates.

Listing 2 shows how a turtle representation of the current state of the RDF graph and the SPARQL Insert/Delete queries based on the diff can be rendered in real-time using AngularJS. The neccessary serialization functions are part of the REX utility belt.

```
<pre ng-init="rc=rexContext">
{{graphToTurtle(rc.graph, rexPrefixMapping)}}
{{createInsertRequest(rc.diff.added, rexPrefixMapping)}}
{{createDeleteRequest(rc.diff.removed, rexPrefixMapping)}}
</pre>
```

Listing 2: Example of serializing the current state of the RDF graph as turtle, and creating SPARQL 1.1 Update queries from the diff

[10]`http://json-ld.org/`

[11]`http://www.w3.org/2011/rdfa-context/rdfa-1.1`

[12]`http://www.w3.org/TR/rdfa-syntax/#A-prefix`

[13]`http://www.w3.org/TR/rdf-interfaces/`

2.3 Conditional RDF generation

There are cases when RDF output should be constructed conditionally based on the form and validation state. For example, consider a form that allows a user to enter a Well Known Text (WKT) string, and corresponding WGS84 lat/long triples should be generated for valid POINT geometries. One way to accomplish this is using a combination of `ng-if` and hidden fields, as shown in Listing 3.

```
<input type="hidden"
 ng-if="PointUtils.isWktPoint(wkt)"
 rex-predicate="geo:long"
 rex-literal="'' + PointUtils.wktToXy(wkt).x"
 rex-datatype="'xsd:float'">
```

Listing 3: Generating triples only for valid WKT strings

2.4 Data Lookups for Prefilling Forms

Forms can be connected to a SPARQL endpoint using `rex-sparql-service="sparqlService"` and `rex-lookup="true"`. Whenever the value of a `rex-subject` changes and `rex-lookup` is enabled, a lookup will be performed, and the data is converted to Talis RDF JSON and stored at *rexContext.json*. Listing 4 shows an example set up using a utility method from the Jassa library.

```
<script type="text/javascript">
$scope.sparqlService = new jassa.service
  .SparqlServiceHttp(endpointUrl, defaultGraphIri);
}
</script>
<form
  rex-context
  rex-sparql-service="sparqlService"
  rex-lookup="true"
  rex-subject="http://dbpdia.org/resource/Leipzig">
    <!-- ... further form controls ... -->
</form>
```

Listing 4: Registering a lookup function for prefilling out forms based on subject resources

2.5 Synchronization with Conversion

Sometimes (datatype) conversions are necessary to map values between their representation in a form control and the one in the RDF structure. For example, the model of the date picker of AngularUI Bootstrap[14] stores values of type *Date*. However, its appropriate (lexical) representation as a literal of *xsd:date* is a string conforming to ISO8601. We solve this problem by introducing a set of `sync-*` annotations for synchronizing model values, as shown in Listing 5. The source and target of a sync are declared using `sync-source`, `sync-target`. With `sync-to-target` and/or `sync-to-source` any changes in the values are propagated in the respective direction. Both attributes take a conversion function as an optional argument. If desired, syncing in one of the directions can be enabled/disabled by specifying a condition expression using `sync-to-target-cond` and/or `sync-to-source-cond`.

[14]`http://angular-ui.github.io/bootstrap/`

```
<div rex-predicate="'dbo:birthDate'"
  rex-literal="bdateStr" rex-datatype="'xsd:date'">
    <input type="text" ng-model="bdate"
      sync-source="bdate" sync-target="bdateStr"
      sync-to-target="dateToString"
      sync-to-source="stringToDate"
      datepicker-popup="dd-MMMM-yyyy"
      datepicker-options="dateOptions">
</div>
```

Listing 5: Data binding with conversions

2.6 Navigation and Filtering

Especially with nested forms it is necessary to navigate between sets of related resources. For example, consider an RDF resource that represents the DBpedia dataset, and another set of resources that represent distributions in SPARQL endpoints on the Web. Further, assume that the distributions refer to the dataset of which they are a distribution of.

```
<ul
  rex-subject="http://.../dataset/dbpedia"
  rex-nav-predicate="'o:distributionOf'"
  rex-nav-inverse="true"
  rex-nav-filter-type="iri"
  rex-nav-targets="distIris">
    <li
      ng-repeat="distIri in distIris"
      rex-subject="{{distIri}}">
        <!-- sub-form for each distribution -->
    </li>
</ul>
```

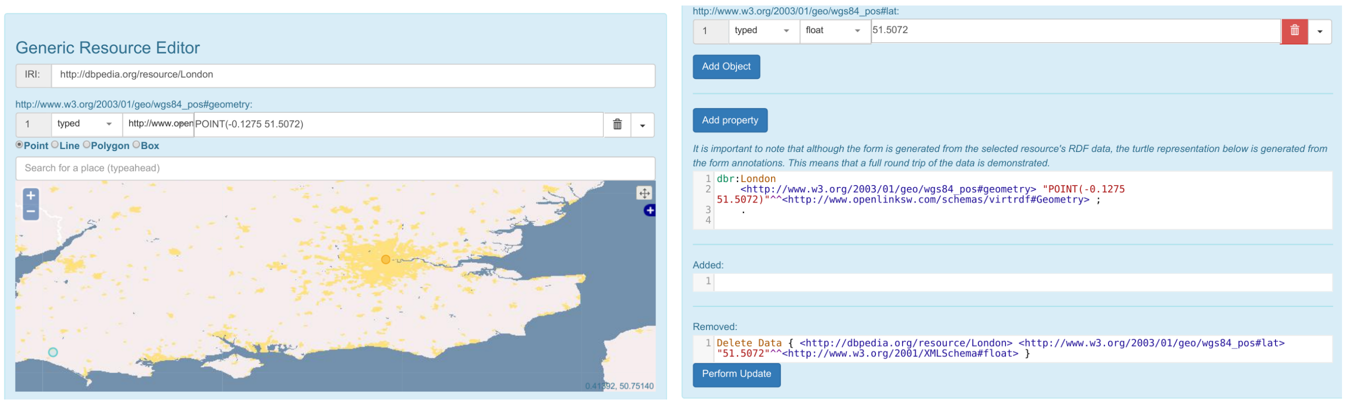
The variable *distIri* will be an array containing all distributions that are reachable from the current subject by following the predicate *o:distributionOf* in inverse direction. The array *distIri* has special behaviour attached to it: Pushing items to it will implicitly create new triples that connect the given subject with the added URI via the given property.

2.7 Renaming resources

While it may seem intriguing to make use of the `rex-subject` annotation for the purpose of renaming resources, this is not directly in the scope of the REX system. REX uses the concept of coordinates to give model values an RDF context. Under this perspective, coordinates are only *references* used to get and put model values from resp. into an RDF space, i.e. a component of a triple in an RDF graph. But coordinates are distinct from *operations* to be performed on an RDF graph.

3. USE CASES

In this section we briefly outline two use cases that we realized with REX and which are part of the online demo. The first one is the simple generic resource editor depicted in Figure 1. When the user configures a SPARQL endpoint and enters the URI of a resource to edit, a form is generated by iterating over the corresponding properties and objects of the data in *rexContext*. This data is automatically loaded


Generic Resource Editor

IRI: http://dbpedia.org/resource/London

http://www.w3.org/2003/01/geo/wgs84_pos#geometry:

1 | typed | http://www.open | POINT(-0.1275 51.5072)

Point [✓] Line Polygon Box

Search for a place (typeahead)

0.41392, 50.75140

http://www.w3.org/2003/01/geo/wgs84_pos#lat:

1 | typed | float | 51.5072

It is important to note that although the form is generated from the selected resource's RDF data, the turtle representation below is generated from the form annotations. This means that a full round trip of the data is demonstrated.

```
dbr:London
    <http://www.w3.org/2003/01/geo/wgs84_pos#geometry> "POINT(-0.1275
51.5072)"^^<http://www.openlinksw.com/schemas/virtrdf#Geometry> ;
    .
```

Added:

Removed:

```
Delete Data { <http://dbpedia.org/resource/London> <http://www.w3.org/2003/01/geo/wgs84_pos#lat>
"51.5072"^^<http://www.w3.org/2001/XMLSchema#float> }
```

Figure 1: A generic resource editor realized with REX

using REX's lookup mechanism. For each RDF term, an instance of our *rdf-term-input* widget is created. This widget is realized as an AngularJS directive whose model represents an RDF term using the attributes defined by Talis RDF JSON. We also created a *geometry-input* widget that can two-way data bind on models with WKT strings. Another use case is a simple form for registering datasets together with information about which SPARQL endpoints they are distributed in.

4. RELATED WORK

There exist several tools for matching, transforming and templating RDF data for visual presentation: *Fresnel*[3] was one of the first approaches in this regard. *Sgvizler* [4] is a library that enables one to use SPARQL queries as attribute values in HTML elements and bind their result sets to different types of visualizations. Recent approaches in this category are the AngularJS-based *RDF Stylesheet Language Transformations* [2] and Uduvudu[15].

A different approach is taken by frameworks that feature RDF widgets: *SemWidg* [6] is a JavaScript based library that provides a framework and tooling for creating widgets, a widget library and a path query language called *SemwidgQL*. *Vie.js* is a JavaScript library for building *decoupled Content Management Systems*, and implements a bridge between Backbone.js and Semantic Web data. Vie also features a form generation component[16]. *RDFauthor*[17] is a JavaScript library that adds a feature-rich authoring component to either a specified region of a Web page or its own window-like overlay.

Semantic Web platforms, such as *OntoWiki* [1], *PoolParty*[18] and VocBench [19] offer features for many aspects of RDF data management, as for example data aquisition, analysis and publishing.

5. CONCLUSIONS & FUTURE WORK

In this demo paper, we presented the REX system for creating highly dynamic HTML forms for RDF data based on the AngularJS framework. We showed how the system is used to realize a number of use cases, such as prefilling form fields, conditionally emitting data, converting values between the fields and the RDF graph, and creating SPARQL 1.1 Update requests for persisting changes. We showed how our system enables idiomatic re-use of existing widgets, such as the date picker of AngularUI Bootstrap, and our own widgets for editing RDF terms and geometries. Further, investigating the use of REX for form generation approaches, which could arrange predefined HTML building blocks to create appropriate forms for RDF resources seems worthwhile. Finally, there are several possibilities of combining this work with other efforts, such as the RDF Changeset systems or access control approaches in general.

Acknowledgment

This work was supported by grants from the EU's 7th Framework Programme provided for the GeoKnow project (GA no. 318159).

[15]https://github.com/uduvudu/uduvudu
[16]http://viejs.org/widgets/forms/
[17]https://github.com/AKSW/RDFauthor
[18]http://www.poolparty.biz
[19]http://vocbench.uniroma2.it/

References

[1] P. Frischmuth et al. OntoWiki—An Authoring, Publication and Visualization Interface for the Data Web. *Semantic Web Journal*, 2014.

[2] S. Peroni and F. Vitali. Rslt: Rdf stylesheet language transformations. In *Proceedings of the ESWC Developers Workshop 2015*, CEUR Workshop Proceedings, 2015.

[3] E. Pietriga, C. Bizer, D. Karger, and R. Lee. Fresnel: A browser-independent presentation vocabulary for rdf. In *The semantic web-ISWC 2006*, pages 158–171. 2006.

[4] M. G. Skjæveland. Sgvizler: A javascript wrapper for easy visualization of sparql result sets. In *ESWC (Satellite Events)*, volume 7540 of *Lecture Notes in Computer Science*, pages 361–365. Springer, 2012.

[5] C. Stadler, P. Westphal, and J. Lehmann. Jassa - A javascript suite for sparql-based faceted search. In *Proc. ISWC Developers Workshop 2014*, pages 31–36, 2014.

[6] T. Stegemann and J. Ziegler. Semwidgjs: A semantic widget library for the rapid development of user interfaces for linked open data. In *44. Jahrestagung der Gesellschaft für Informatik, Informatik 2014, Big Data - Komplexität meistern, 22.-26. September 2014 in Stuttgart, Deutschland*, pages 479–490, 2014.