ARC: appmosphere RDF Classes for PHP Developers

Benjamin Nowack

appmosphere web applications, Kruppstr. 100, 45145 Essen, Germany
bnowack@appmosphere.com

Abstract. ARC is an open source collection of lightweight PHP scripts optimized for RDF development on hosted Web servers. It currently consists of a non-validating RDF/XML parser, an N-Triples Serializer, and a "Simple Model" class providing common methods for working with resource descriptions. The three main classes are stand-alone, single-file scripts, thus facilitating the bundling with existing PHP-based applications. By partly using arrays instead of objects, ARC offers speed improvements compared to toolkits that follow approaches completely based on PHP objects.

1 Motivation

Despite the availability of the feature-rich and mature RAP - RDF API for PHP [1], there was ongoing demand for RDF libraries (especially for an RDF/XML parser) which could be bundled with existing PHP frameworks more easily. RAP is deployed as an integrated API, so that version changes usually require to completely replace a prior installation. ARC[1] uses more lightweight, stand-alone classes in order to allow independent use and to facilitate upgrading tasks.

2 RDF Classes

ARC currently consists of three basic class files for working with Semantic Web data: An RDF/XML parser, an N-Triples serializer, and a "Simple Model" class which provides common methods for working with resource descriptions.

2.1 ARC RDF/XML Parser

The *RDF/XML Parser* class can be used to parse a passed string, to load and parse data from the local machine, or to load and parse data from the Web. The built-in Web reader does not follow HTTP redirects yet, but it accepts several parameters to e.g. work with proxies, send custom HTTP headers, or limit the number of lines to

[1] ARC stands for *appmosphere RDF classes*. The classes and documentation are available online at http://www.appmosphere.com/en-arc

parse. Additionally, the class can be configured to retrieve content-negotiated RDF/XML (via an HTTP A*ccept* header) , or to keep the raw RDF/XML data and/or response headers of the remote server in memory. The code snippet below shows a usage example.

```
/* class inclusion */
include_once("ARC_rdfxml_parser.php");

/* class configuration (optional) */
$args=array(
  "encoding"=>"auto",/* auto-detect UTF-8 etc */
  "proxy_host"=>"192.168.27.1",
  "proxy_port"=>8080,
  "user_agent"=>"myParser v1.0"
);

/* instantiation */
$parser=new ARC_rdfxml_parser($args);

/* parsing a file from the Web */
$url="http://www.example.com/data.rdf";
$result=$parser->parse_web_file($url);
if(is_array($result)){
  echo count($result)." triples found";
}
else{
  echo "couldn't parse ".$url.": ".$result;
}
```

The parser does not check the parsed code for complete RDF-validity, only XML errors and some very basic RDF/XML syntax [2] errors are detected. The result of the parsing process is either an error message or an array of triples.

2.2 ARC N-Triples Serializer

The serializer class generates an N-Triples [3] string from an array of triples. The structure of the passed triple array has to conform to the following ARC triple array structure:

Each entry is an associative array with keys *s*, *p*, and *o*:
- *s* (a subject node array)
- *p* (a predicate URI string)
- *o* (an object node array)

A subject node array may have the following keys:

- *type* ("uri" or "bnode")
- *uri* (if type is "uri")
- *bnode_id* (if type is "bnode")

An object node array may contain additional key-value pairs:

- *type* ("uri", or "bnode", or "literal")
- *uri* (if type is "uri")
- *bnode_id* (if type is "bnode")
- *val* (if type is "literal")
- *dt* (datatype URI if available and type is "literal")
- *lang* (language code if available and type is "literal")

The serializer provides only a small number of configuration options. It is possible to set the linebreak and the space characters as illustrated below.

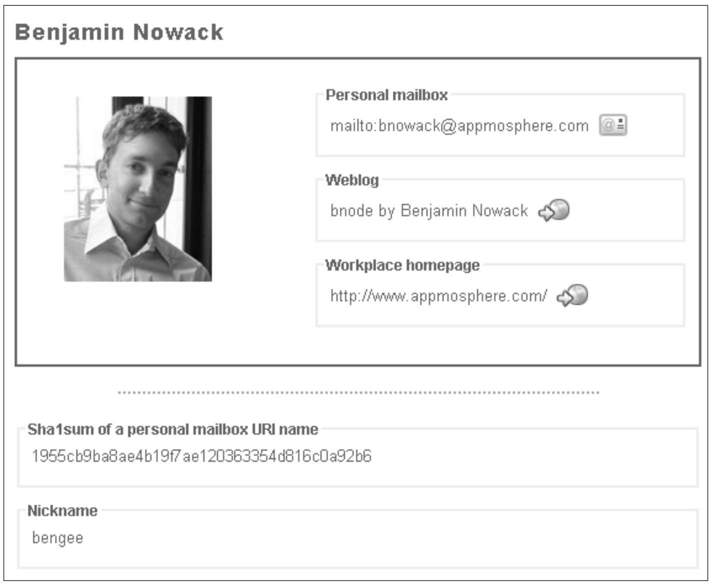
```
/* class inclusion */
include_once("ARC_ntriples_serializer.php");

/* class configuration (optional) */
$args=array("linebreak"=>"\n", "spacer"=>"\t");

/* instantiation */
$ser=new ARC_ntriples_serializer($args);

/* displaying the N-Triples code */
echo $ser->get_ntriples($triples);
```

2.3 ARC Simple Model

ARC's *Simple Model* class indexes a triple array and offers methods to e.g. return a list of resources, find resources based on a provided identifier, parse RDF lists, or extract property values. A detailed usage example is given in section 3.

3 Use cases

By combining the *RDF/XML Parser* (or an adjusted triple array queried from a basic RDF triple store) with the *Simple Model* class, it is easily possible to generate HTML views from RDF. Figure 1 shows a resource description summary which could be created by passing two *Simple Model* instances (one with vocabulary information, and one with data about individuals) to an HTML rendering script. The latter is not included in ARC but the PHP code snippets below Fig. 1 exemplify how data can be pre-processed by ARC in order to retrieve grouped property values and label information.



Personal mailbox
mailto:bnowack@appmosphere.com

Weblog
bnode by Benjamin Nowack

Workplace homepage
http://www.appmosphere.com/

Sha1sum of a personal mailbox URI name
1955cb9ba8ae4b19f7ae120363354d816c0a92b6

Nickname
bengee

Fig. 1. HTML view generated by combining information of two ARC Simple Models

PHP code to instantiate the two ARC Simple Model classes:

```
/* abbreviations */
$rdfs="http://www.w3.org/2000/01/rdf-schema#";
$foaf="http://xmlns.com/foaf/0.1/";

/* vocabulary model */
$v_triples=$parser->parse_web_file($foaf);
$model_args=array(
   "triples"=>$v_triples,
   "ns_abbrs"=>array($foaf=>"foaf", $rdfs=>"rdfs")
);
$v_model=new ARC_simple_model($model_args);

/* data model */
$d_url="http://www.example.com/my_foaf.rdf";
$d_triples=$parser->parse_web_file($d_url);
$model_args["triples"]=$d_triples;
$d_model=new ARC_simple_model($model_args);
```

PHP code to find a foaf:Person resource in a foaf:PersonalProfileDocument:

```
/* find PPD in data model */
$ppd_qname="foaf:PersonalProfileDocument";
if(!$ppds=$d_model->get_resources($ppd_qname)){
  return false;
}
$ppd=$ppds[0];

/* get person identifier via primaryTopic */
if(!$id=$d_model->rpv[2]($ppd, "foaf:primaryTopic")){
  return false;
}
$person=$d_model->get_resource($id);
```

PHP code to display labeled property values:

```
/* name value */
$name_val=$d_model->rpv($person, "foaf:name");
if(!$name_val){
  /* try alternative naming properties */
}

/* mbox values */
$cur_vals=array();
if($cur_props=$person["props"]["foaf:mbox"]){
  foreach($cur_props as $cur_prop){
    $cur_vals[]=$cur_prop["val"];
  }
}

/* mbox label (use vocabulary model) */
if($terms=$v_model->get_resources("foaf:mbox")){
  $term=$terms[0];/* an rdf:Property instance */
  $cur_label=$v_model->rpv($term, "rdfs:label");
}

/* display */
echo '<h2>'.$name_val.'</h2>';
echo ($cur_label) ? $cur_label : "mbox";
echo "- ".join("<br />- ", $cur_vals);
```

[2] "rpv" is an alias for the method "get_resource_prop_val" which retrieves a single property value for a given resource.

Further use cases include reading, importing, and displaying of RSS 1.0 feeds, or any other RDF/XML-compatible data. At the moment, ARC is best suited for rapid RDF development or as a small addition to existing codebases, but the classes have already been successfully tested in larger projects (e.g. the new SemanticWeb.org portal) as well.

4 Performance

RAP is currently the only other toolkit written completely in PHP (which is a prerequisite to be deployable to cheap hosted Web server environments) that conforms to the revised RDF specifications [4]. The performance tests done for this paper are limited to comparing the RDF/XML parsers. A third option (w.r.t. to parsing RDF) is Morten Frederiksen's SimpleRdfParser [5], a wrapper class for RAP that uses arrays instead of RAP's internal objects; it was tested as well. The Redland Application Framework [6] also provides bindings for PHP [7]. However, the PHP bindings can neither be installed on average hosted Web servers nor on the Windows machine that was used for the performance tests[3]. Redland was not included in the benchmarks. It would have been an interesting reference, though.

The tests were run on a desktop PC (Pentium IV 1.8 GHz, Windows 2000), using PHP 4.3.6, and the xdebug extension [8] for profiling. The long execution times in the benchmark results are caused by having to disable Zend optimizer while using the profiling extension. Switching back from xdebug to the Zend optimizer accelerates the parsers by a factor of 4.

Parsing RDF/XML Files

ARC (v0.2.0), SimpleRdfParser, and RAP (v0.9.1) were tested with 3 different files:
1. An RSS 1.0 news feed (338 triples)
2. The W3C RDF test cases manifest file[4] (2160 triples)
3. A large FOAF file (8601 triples)

RDF files beyond this size are usually stored in a database, these tests cover common in-memory parsing tasks only. As the SimpleRdfParser can't load files directly, each test passes the RDF/XML code as a single string variable to the parser.

	RAP	**SimpleRdfParser**	**ARC**
Processing (total)	2.921 s	1.984 s	0.834 s
Triples/s (total)	~116	~170	~405
Triples/s (excl. compiling)	~149	~265	~431

Table 1. Parsing an RSS 1.0 news feed containing 338 triples.

[3] Installing the bindings was not possible with the existing PHP engine. It would have been neccessary to re-compile the PHP sources.

[4] http://www.w3.org/2000/10/rdf-tests/rdfcore/Manifest.rdf

	RAP	**SimpleRdfParser**	**ARC**
Processing (total)	16.257 s	4.372 s	2.037 s
Triples/s (total)	~133	~494	~1060
Triples/s (excl. compiling)	~134	~553	~1131

Table 2. Parsing the W3C test cases manifest file containing 2160 triples.

	RAP	**SimpleRdfParser**	**ARC**
Processing (total)	61 s	12 s	6 s
Triples/s (total)	~141	~717	~1434

Table 3. Parsing a large FOAF file containing 8601 triples, xdebug turned off[5].

The benchmark results[6] show that using the SimpleRdfParser is about two to four times faster than building a complete in-memory model via RAP. ARC is about twice as fast as the SimpleRdfParser. The difference between using PHP objects vs. arrays becomes more obvious with a growing number of triples in a model. While the SimpleRdfParser is 1.5 times faster than RAP when parsing the smallest RDF file, it is 3.7 times faster at parsing the 2160 triples, and about 5 times faster when processing the large file.

These tests don't consider that RAP is not only parsing but also building indexes and validating the parsed code. But it could still make sense to examine the xdebug results in more detail in order to see if RAP may be partly made faster. ARC's speed could for example be improved by working around PHP's automatic type conversion mechanism (e.g. by using "`===`" instead of "`==`") and by applying simple string functions such as `strpos` instead of regular expressions.

5 Conclusion and Next Steps

ARC is a lightweight alternative (or addition) to RAP's feature-rich API when only limited functionality such as parsing or displaying is required. Being open source and written entirely in PHP, the RDF classes can be bundled with PHP frameworks and used in shared Web hosting environments. The advantage of using PHP arrays instead of object structures increases with the size of parsed RDF/XML documents.

The main idea of ARC is to provide lightweight RDF classes. However, several options have already been added to ARC which might not be needed for certain use cases. It is planned to build a simple parser generation service which will allow developers to tailor the ARC scripts to their needs. It will be possible to e.g. exclude the method for using an HTTP socket instead of PHP's one-line `fopen` function, or to automatically remove comments. Together with other configuration options the parser's size could be reduced to 20KB or less.

[5] xdebug had to be disabled for this test as it needed to much main memory to profile RAP
[6] The complete benchmark details can be downloaded from http://www.appmosphere.com/2005/php_parser_benchmarks_2005_04_15.pdf

Other considerations include extending the parser with more validation features, making the Web reader follow HTTP redirects, or adding an RDF store interface to the ARC collection. A basic SPARQL [9] implementation is already available [10].

References

1. Bizer, C., Oldakowski, R.: *RAP: RDF API for PHP*. Berlin (2004). http://www.wiwiss.fu-berlin.de/suhl/radek/pub/RAP-oldakowski.pdf
2. Beckett, D.: *RDF/XML Syntax Specification (Revised)*. W3C (2004). http://www.w3.org/TR/rdf-syntax-grammar/
3. Grant, J., Beckett, D.: *RDF Test Cases*. W3C (2004). http://www.w3.org/TR/rdf-testcases/
4. *Resource Description Framework (RDF)*. W3C (2004). http://www.w3.org/RDF/
5. Frederiksen, M.: *Easy RDF-parsing with PHP*. http://www.wasab.dk/morten/blog/archives/2004/05/31/easy-rdf-parsing-with-php
6. Beckett, D.: *Redland RDF Application Framework*. http://librdf.org/
7. Beckett, D.: *Redland RDF Language Bindings - PHP Interface*. http://librdf.org/docs/php.html
8. xdebug. http://xdebug.org/
9. Prud'hommeaux, E., Seaborne, A.: *SPARQL Query Language for RDF*. W3C (2005). http://www.w3.org/TR/rdf-sparql-query/
10. Nowack, B.: *ARC SPARQL Parser*. http://www.appmosphere.com/pages/en-arc_sparql_parser