GR4PHP: A Programming API for Consuming E-Commerce Data from the Semantic Web

First Workshop on Programming the Semantic Web, ISWC 2012
November 11-15, 2012, Boston, USA

Alex Stolz, Mouzhi Ge and Martin Hepp
{firstname.lastname}@unibw.de

E-Business and Web Science Research Group
Universität der Bundeswehr Munich, Germany
Structured E-Commerce Data (GoodRelations)

Simple, focused crawl:
• 2625 shops with GoodRelations data
• 1382 product offers per shop on average
• > 3.6 Mio. offers in total

Source: Focused crawl, May 2012

Web developers would love to take advantage of this rich data corpus!
Web development based on Semantic Web data lags behind opportunities

1. Domain knowledge (e.g. vocabulary patterns) and SPARQL skills required to formulate useful queries on Semantic Web data
2. Semantic Web programming frameworks mainly help experts only

But: Web technology standards are widely used and understood even by ordinary Web developers
Web Technology Trends

Web APIs (REST)

GET /shop/item123

Programming Libraries

```
#include "shop.php"
$item123 = $shop->get("item123");
```
Similar Approaches

• NLP and Question Answering for SPARQL
  – Suffering inaccuracy

• Query Builders and Query Assistance
  – Integration problem for Web development

• Programming Frameworks
  – Little support for high-level data (conceptual level)
    Semantic Web applications
Our Approach

• Tap into existing Web technology standards
  – PHP programming library to alleviate consumption of e-commerce data on the Semantic Web

• Take advantage of Semantic Web benefits
  – E.g. decentralized architecture, flexibility of integrating diverse data sources, etc.
Benefits of betting on Web standards

• Eliminate / reduce learning efforts
  – Documentation, community support and reference implementations are already in place

• Hide complexity of Semantic Web
  – No RDF, SPARQL, GoodRelations skills needed
Implementation

• Powerful PHP library
  – Developer-friendly, straightforward programming API with uniform method signatures
  – Function calls translated to proper SPARQL queries
  – Queries cover most recurring data patterns

• Precise query mechanism under the hood

Project page:
http://code.google.com/p/gr4php/
# PHP Functions

<table>
<thead>
<tr>
<th>Function name</th>
<th>GoodRelations concept</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>getCompany</td>
<td>BusinessEntity</td>
<td>Information about legal entities</td>
</tr>
<tr>
<td>getOffers</td>
<td>Offering</td>
<td>List product or services offerings and relevant details</td>
</tr>
<tr>
<td>getProductModel</td>
<td>ProductOrServiceModel</td>
<td>Product model details, extensible to support custom product features</td>
</tr>
<tr>
<td>getStore</td>
<td>Location</td>
<td>Information about point of sales with option for filtering out stores closed at query time</td>
</tr>
<tr>
<td>getLocation</td>
<td>Location</td>
<td>List of nearby locations based on geo position, location name or GLN</td>
</tr>
<tr>
<td>getOpeningHours</td>
<td>OpeningHoursSpecification</td>
<td>List of opening hours for a given point of sale</td>
</tr>
</tbody>
</table>
Conceptual Architecture

Web server

Web application

PHP array

API call

GR4PHP library

RDF store with SPARQL endpoint

SPARQL query

JSON response
function get*(array $inputArray,  // constraints array
    [array $wantedElements=FALSE],  // selection array
    [string $mode=Configuration::MODE_LAX],  // search mode
    [int $limit=Configuration::LIMIT],  // result limit
    [array $searchProperties=FALSE])  // custom properties
GR4PHP API – Example

• Query n=10 offering descriptions with textual properties (gr:name, rdfs:label, etc.) that contain the search term ”Camcorder” and cost not more than 200 dollars.
require_once("gr4php.php"); // include library
$connection = new GR4PHP(Configuration::ENDPOINT_URI_BURNER);

$result_array = $connection->getOffers(
    array("title" => "Camcorder",
          "maxPrice" => 200,
          "currency" => "USD"), // input
    array("uri", "title", "price", "currency"), // output
    Configuration::MODE_LAX, // rigidity of search
    10 // maximum result limit
);
## GR4PHP API – Example (3)

<table>
<thead>
<tr>
<th>uri</th>
<th>title</th>
<th>price</th>
<th>currency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>...</td>
<td>Flip Video UltraHD Camcorder – Black/Chrome</td>
<td>199.99</td>
</tr>
<tr>
<td>2.</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
GR4PHP – Example (4)

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX gr: <http://purl.org/goodrelations/v1#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

WHERE {
  {?uri rdfs:label ?title. FILTER(contains(?title, 'Camcorder'))} UNION
  {?uri gr:name ?title. FILTER(contains(?title, 'Camcorder'))} UNION
  {?uri gr:description ?title. FILTER(contains(?title, 'Camcorder'))} UNION
  {?uri rdfs:comment ?title. FILTER(contains(?title, 'Camcorder'))} UNION
  {?uri dc:title ?title. FILTER(contains(?title, 'Camcorder'))}

  ?pricespec gr:hasCurrencyValue ?price. FILTER(?price <= 200)
  ?pricespec gr:hasCurrency ?currency. FILTER(?currency = 'USD')
  ?uri a gr:Offering.
} LIMIT 10
Features and Restrictions

• SPARQL 1.1
• Validation of arguments passed through library functions
• Extensibility mechanism to meet for sophisticated use cases including external vocabularies
Results

**Table 3.** Times spent (mm:ss) for the completion of the assignments

<table>
<thead>
<tr>
<th>Task</th>
<th>Group A</th>
<th>Group B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Task 1</td>
<td>3:15</td>
<td>3:30</td>
</tr>
<tr>
<td>Task 2</td>
<td>1:30</td>
<td>2:15</td>
</tr>
</tbody>
</table>

- Developer performance increased significantly by use of the library (group A) with respect to manually crafting SPARQL queries (group B).
- Independent-samples t-tests compared times of groups A and B \( \rightarrow p<0.05 \) for task 1 and task 2.
Applications

1. Implementation of and integration of semantic data into Web pages based on e-commerce data
2. Mobile application development taking into consideration contextual information (e.g. geo data)
3. Novel product search engines
Adoption by Mobile Web App Project

“Ravensburg App“

http://www.lieber-ravensburg.de/developer/
Acknowledgments

The work on GoodRelations and this paper have been supported by the German Federal Ministry of Research (BMBF) by a grant under the KMU Innovativ program as part of the Intelligent Match project (FKZ 01IS10022B).
Thank you!

Martin Hepp
martin.hepp@unibw.de

Project page:
http://code.google.com/p/gr4php/