Technische Dokumentation des Leipzig Data Events Widgets

Version vom 13. Oktober 2013

1 Allgemeines

Ziel der entwickelten Softwarelösung ist die prototypische Web-Darstellung von Event- und begleitenden Informationen aus dem Leipzig Open Data Projekt [LOD] auf Webseiten angeschlossener *Akteure* (siehe Glossar) auf der Basis des JS-Frameworks *Exhibit*.

Die Lösung bietet dem *Webdienste-Anbieter* die Möglichkeit, auf einfache Weise eine lesende Sicht auf vorhandene Daten mit Such- und Filterfunktionen zu definieren und in die Webpräsenzen der von ihm betreuten Akteure einzubauen. Der Webdienste-Anbieter kann dazu relevante Daten aus Datenbeständen, die nach einer vorgegebenen Ontologie aufgebaut sind, der *LD.Events Datenbasis*, je nach Widget-Version in einen eigenen lokalen ARC2 Data Store einlesen, daraus über Filterkriterien akteursspezifisch einen relevanten Teilbestand für die Präsentation auswählen und in die Webpräsenz des Akteurs als iframe einbinden. Alternativ kann die LD.Events Datenbasis oder Teile davon auch direkt über den Leipzig Data Sparql-Endpunkt `http://www.leipzig-data.de:8890/sparql` ausgelesen werden. Nutzer der Webpräsenz des Akteurs können in diesen Informationen weiter selektiv suchen und sich die Ergebnisse in verschiedenen Formaten optisch ansprechend anzeigen lassen.

Nicht erfasst ist der Prozess des Fortschreibens der LD.Events Datenbasis durch einen Kreis dafür autorisierter Personen – hierfür ist der bisherige Datenerfassungsprozess auf Protokollebene so zu harmonisieren, dass relevante und qualitätsgesicherte Daten direkt in die LD.Events Datenbasis übernommen werden können. In einer weiteren Ausbaustufe kann hier ein Mashup aus verschiedenen Kanälen eingebaut werden, die dem vereinbarten Protokoll folgen. Derzeit können Veranstaltungsdaten von autorisierten Personen direkt über das Ontowiki-Portal `http://leipzig-data.de/Data` des Leipzig Data Projekts eingegeben werden.

Die Softwarelösung basiert auf Javascript, die Präsentation läuft komplett mit allen Such- und Darstellungsfunktionen im Webbrowser des Nutzers ab und greift auf ein JSON-Datenobjekt zu, das mit der Javascript-Funktionalität des Widget-Framework *Exhibit* weiter aufbereitet wird.

Die Softwarelösung wurde von **Johannes Frey** im Rahmen des Leipzig Open Data Projekts [LOD] entwickelt. Das Projekt wurde durch die Stadt Leipzig im Rahmen der Open Innovation Ausschreibung 2012 unterstützt.

2 Produktübersicht

Es gibt **2 verschiedene** Widget Versionen. Welche Version momentan verwendet wird steht in der `index.html`. Die Versionen unterscheiden sich in der Art und Weise, wie Daten zur Präsentation eingespielt werden.

2.1 Allgemeines

Simple Widget. Das simple Widget baut direkt eine Verbindung mit einem SPARQL-Endpunkt auf (z. B. dem von `leipzig-data.de`) und bezieht anhand einer nutzerdefinierten Anfrage Daten aus diesem. Es ist sehr leichtgewichtig und verwendet keinen eigenen Triplestore. Es benötigt somit auch keine eigene MySQL-Datenbank und verzichtet auf weitere Frameworks.

Komplexes Widget. Das komplexere Widget benötigt einen eigenen Triplestore. In diesen können RDF-Dumps, etwa ein Dump der auf leipzig-data.de bereitgestellten Event-Daten oder eigene RDF-Daten, eingespielt werden. Zusätzlich können weitere Daten aus anderen Quellen eingespielt und aggregiert werden. Ein Filtermechanismus erlaubt es anschließend, kundenspezifisch nur bestimmte Daten für eine Präsentation auszuwählen. Als besonderes Feature stellt dieses Widget noch einen öffentlich zugänglichen und konfigurierbaren SPARQL-Endpunkt für die aggregierten Daten bereit. Das Widget benötigt im Gegensatz zur simplen Version eine MySQL Datenbank und verwendet das ARC2-Framework.

2.2 Individuelle Präsentation

Aus dem eingespielten Datenbestand können Webdienste-Anbieter akteursspezifische Angebote zusammenstellen. Dazu müssen diese pro Akteur ein eigenes *Thema* erstellen – oder ein Thema aus vorgegebenen Themen auswählen und anpassen –, in welchem die akteursspezifischen Vorgaben und Wünsche umgesetzt sind. Die Anpassungsmöglichkeiten beziehen sich auf

- die Auswahl des relevanten Grunddatenbestands (über einen gestaltbaren PHP-Filterprozess `getdata.php`, über den auch regelmäßig die Daten aktualisiert werden),
- Umfang und Anordnung der Such- und Filterfunktionen, die den Nutzern zur Verfügung stehen (über eine Exhibit-Template-Datei `presentation.php`, die akteursspezifisch angepasst werden kann), und
- das Layout über eine akteursspezifische CSS-Datei `styles.css`.

Im Zentrum des Konzepts steht das Javascript basierte Widget-Framework *Exhibit* in der Version 2.0. In den weiteren Ausführungen wird die Kenntnis dieses Frameworks im Umfang von [Ex] vorausgesetzt.

2.3 Installation

Die Installation unterscheidet sich je nach gewählter Version.

Simple Widget

- Ausrollen der Dateien in ein eigenes Verzeichnis `widget` auf dem Webserver, in welchem der Webserver lesen **und schreiben** kann.

- Das simple Widget ist bereits vorkonfiguriert und bezieht alle aktuellen Event-Daten vollständig aus dem SPARQL-Endpunkt von Leipzig Data. Folgen Sie einfach den Hinweisen in der `index.html`

Komplexes Widget

- Ausrollen der Dateien in ein eigenes Verzeichnis `widget` auf dem Webserver, in welchem der Webserver lesen **und schreiben** kann.
- Aufsetzen einer Datenbank und Konfiguration des ARC2 Data Store.

  Dazu ist die Datei `db_credentials.php` in eine Datei `db.php` zu kopieren und die Zugangsdaten zur lokalen Datenbank einzutragen. In dieser Datenbank werden eine Reihe neuer Tabellen mit dem Namenspräfix[1] `data_store` angelegt.
- Auswahl oder Erstellen eines Themas pro Akteur zur Datenpräsentation und Einbinden – z. B. als Webseite – in die Struktur der Website. Dazu kann aus den vorhandenen Themen (Verzeichnis `themes`) ausgewählt werden.

Die Anzeige kann durch Ändern der Präsentations- und Stildateien des Themas akteursspezifischen Bedürfnissen angepasst werden, was aber einige Vertrautheit mit dem Exhibit-Framework voraussetzt. Siehe dazu ebenfalls [Ex].

3 Grundsätzliche Struktur- und Entwurfsprinzipien

3.1 Einspielen der Event-Daten

Um die Daten in den durch die vorherigen Schritte eingerichteten ARC2-Store zu importieren gibt es **2 Möglichkeiten**.

a) **Import mittels SPARQL-Anfrage (NUR Simple Widget)**

Dies ist die einfacherere Variante. In der Datei `getdata.php` befindet sich die Funktion `getData($store)`. In dieser kann die SPARQL-Anfrage angepasst werden, um nur eine Teilmenge der Event-Daten aus unserem öffentlichen Sparql-Endpunkt[2] abzufragen. In der Vorkonfiguration werden alle Daten übernommen. Durch einfaches Ausführen der Datei `getdata.php` findet der Import automatisch statt und die für die Präsentation benötigte Datei `data.json` wird erstellt.

b) **Import mittels Dateidump(s) (NUR komplexes Widget)**

Dies ist die komplexere, aber dafür stabilere und flexiblere Lösung. An die öffentlich verfügbare Leipzig Data Events Datenbasis kann eine eigene SPARQL-Abfrage[3] gestellt werden, um Event-Daten zu extrahieren. Beispiele für solche Anfragen finden sich in der Datei `Queries.txt` der Distribution. Das Ergebnis dieser Abfrage ist in einer Datei `EventsDump.ttl` zu speichern und in das Widget-Verzeichnis zu kopieren.

Bei Bedarf können weitere RDF-Dateien eingespielt werden, indem entsprechende Ladeanweisungen der Form

[1] Dieser Präfix kann in der Datei `Store.php` geändert werden.
[2] `http://leipzig-data.de:8890/sparql`
[3] Über deren Sparql-Endpunkt `http://leipzig-data.de:8890/sparql`.

```
$store->query("LOAD <file:Datei.ttl>");
```

in der Funktion `loadDataFromFile` der Datei `Store.php` ergänzt werden.

Zusätzlich ist es möglich, mit der Funktion `loadDataFromEndpoint` noch Dateien über einen SPARQL-Endpunkt einzubinden. Diese muss and die Bedürfnisse angepasst und dann noch an entsprechender Stelle in der Datei `Store.php` eingebunden werden. Sollten jedoch lediglich Daten über einen SPARQL-Endpunkt geladen werden, ist das Simple Widget die bessere Wahl.

Nach dieser Konfiguration erfolgt das Laden aller Daten in den Store mittels Aufruf von `Store.php`, dann das Erzeugen der Präsentationdsatei und ggf. Filtern der Daten durch angepasste SPARQL-Anfrage durch `getdata.php`.

3.2 Anlegen der json-Datei

Die Datei `data.json` enthält alle relevanten Informationen im Exhibit-spezifischen JSON-Format und ist für das gewählte Thema mit dem dortigen PHP-Skript `getdata.php` zu erzeugen. Hierbei kann eine weitere nutzerspezifische Datenauswahl durch den Webdienste-Anbieter erfolgen, um z. B. in verschiedenen Themen unterschiedliche Daten darzustellen. Dazu muss die SPARQL-Anfrage in der Funktion `filterData` verändert werden.

Für das Konvertieren von RDF nach JSON wurden die RDF-Management-Funktionalitäten von ARC2 um ein Exhibit-JSON-Plugin[4] erweitert, welches die Serialisierung des geparsten RDF-Graphen in das Zielformat realisiert. Dabei wird eine automatische Erkennung der Exhibit-Datentypen vorgenommen, sodass z. B. ein Datum auch den Datentyp `date` zugewiesen bekommt. In der simplen Version kommt einen Anpassung dieses Plugins zum Einsatz, was funktional identisch ist, jedoch ohne ARC2 funktioniert.

Die Struktur der Exhibit-JSON-Datei und die verwendeten Datentypen sind in [Ex] genauer beschrieben.

> *Hinweis:* Für das korrekte Erstellen von `data.json` muss der das PHP-Skript ausführende Prozess über Schreibrechte in dem Verzeichnis verfügen.

3.3 Anzeige der Daten

Zur Anzeige der Daten wird das JSON-Datenobjekt aus der json-Datei geladen und mit den Layout-Informationen der Webseite über verschiedene Tag-Properties zusammenführt.

Neben der json-Datei und `exhibit-api.js` werden die Erweiterungen `time-extension.js` und `calendar-extension.js` verwendet. Das json-Datenobjekt wird über eine Link-Header Deklaration mit `rel="exhibit/data"` eingebunden.

Zur Aktivierung der Google Maps Kartenfunktion wird ein eigener Google Maps Schlüssel benötigt. Dieser ist in die Datei `gmaps_api_key.php` einzufügen.

Die Anzeige auf der Webseite wird von Exhibit entsprechend der angegebenen `ex:...` Attribute verschiedener `<div>` Tags gesteuert. So lassen sich auf einfache Weise verschiedene Strukturelemente kombinieren, über die unterschiedliche Sichten auf die Datenauswahl eingestellt

[4] Siehe die Datei `plugins/ARC2_ExhibitJSONSerializerPlugin.php` in der Distribution.

und zusammengeführt werden können. Die ausgewählten Daten werden dann im Hauptfeld angezeigt.

3.4 Aktualisierung

Es sind die Schritte 3.1 sowie 3.2 erneut auszuführen, um neue Event-Daten einzuspielen.

> *Hinweis:* Sollten anschließend in der Präsentation keine Veränderungen zu sehen sein, so hilft meist das Löschen des Browser-Caches weiter.

4 Testen

Zu ergänzen.

5 Glossar

5.1 Begriffe

Akteur. Anbieter realweltlicher Events, die im verteilten Kalendersystem erfasst werden sollen, um die Zielgruppe dieses Anbieters auf das Event aufmerksam zu machen. Typischerweise ein Verein oder anderweitiger Betreiber eines Orts, an dem regelmäßig oder unregelmäßig Events stattfinden, der einen *Webdienste-Anbieter* mit Design und technischer Unterstützung beim Betrieb der eigenen Webpräsenz beauftragt hat (oder diese Kompetenz selbst vorhält).

ARC2. Flexibles RDF-System für PHP-basierte Semantic Web Anwendungen, siehe [Arc2]. Für diese Anwendung muss das mitgelieferte JSON-Plugin installiert werden, das die erforderlichen Daten im Exhibit-spezifischen Ausgabeformat formatiert.

Exhibit. Javascript basiertes Publikations-Framework für datenreiche interaktive Webseiten, siehe `http://www.simile-widgets.org/exhibit`. In dieser Anwendung wird Exhibit 2.0 verwendet.

Nutzer. Besucher der Website des Akteurs. Restriktionen (etwa in der Behindertengerechtheit der Anzeige) ergeben sich aus dem verwendeten Framework, das vom Browser des Nutzers unterstützt werden muss (und im Normalfall auch unterstützt wird). Der Nutzer muss Javascript aktiviert haben.

OntoWiki. Projekt der AKSW-Gruppe am Institut für Informatik der Universität Leipzig, siehe `http://ontowiki.net/Projects/OntoWiki`.

Thema. Akteursspezifische Anpassung des Frameworks, bestehend aus

- einem PHP-Filterskript `getdata.php`,
- einem Exhibit-Präsentationstemplate `presentation.php`,
- einer Stylesheet-Datei `styles.css`,
- einer mit `getdata.php` zu generierenden und regelmäßig zu aktualisierenden JSON-Datendatei `data.json` mit einem akteursspezifischen Ausschnitt aus den Grunddaten,
- ggf einem eigenen Google Maps Schlüssel `gmaps_api_key.php`,

die vom Webdienste-Anbieter akteursspezifisch aus mehreren Vorlagen auszuwählen und anzupassen oder selbst herzustellen ist.

Webdienste-Anbieter. Betreiber einer oder mehrerer Websites von *Akteuren* mit Zugriff auf das Dateisystem des Webservers, um dort weitere PHP-Funktionalität zu integrieren. Neben dem Betrieb der Websites leistet der Webdienste-Anbieter *first level support* im Bereich der Schulung der Datenverantwortlichen auf Akteursseite und bietet Beratung beim Design der Website des Akteurs an.

LD.Events Datenbasis. Datenbasis des Leipzig Open Data Projekts [LOD], auf die über eine SPARQL-Schnittstelle oder ein git-Repo zugegriffen werden kann. Perspektivisch soll diese Datenbasis zu einer verteilten gemeinsamen Datenbasis der am Projekt beteiligten Webdienste-Anbieter weiterentwickelt werden, auf die nach Open Data Prinzipien zugegriffen werden kann.

6 Quellen

[**Arc2**] Flexible RDF system for semantic web and PHP practitioners.
`https://github.com/semsol/arc2/wiki`

[**Ex**] Getting Started with Exhibit.
`http://simile-widgets.org/wiki/Getting_Started_with_Exhibit`

[**LOD**] Das Leipzig Open Data Projekt. Nov. 2012 – April 2013.
`http://leipzig-netz.de/index.php5/LD.OpenInnovation-12`