Technische Dokumentation des Leipzig Data Events Widgets

Version vom 21. Februar 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.Datenbasis*, in einen eigenen lokalen ARC2 Data Store auslesen, daraus über Filterkriterien akteursspezifisch einen relevanten Teilbestand für die Präsentation auswählen und in die Webpräsenz des Akteurs als iframe einbinden. Nutzer der Webpräsenz des Akteurs können in diesen Informationen weiter selektiv suchen und sich die Ergebnisse in verschiedenen Formaten anzeigen lassen.

Nicht erfasst ist der Prozess des Fortschreibens der LD.Datenbasis durch einen Kreis dafür autorisierter Personen – hierfür ist ein zweites Projekt aufzusetzen, mit dem die bisherigen Datenerfassungsprozesse auf Protokollebene so harmonisiert werden, dass relevante und qualitätsgesicherte Daten direkt in die LD.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 Open 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

2.1 Übersicht

Über eine SPARQL-Anfrage kann der Webdienste-Anbieter regelmäßig relevante Daten aus der LD.Datenbasis auslesen und mit dem PHP Skript `Store.php` als Dump in einen eigenen lokalen ARC2 Data Store speichern. Im Rahmen der LD.Event Aktivität wird wöchentlich ein frei verfügbarer SPARQL-Endpunkt (siehe `http://leipzig-data.de/widget/sparql.php`)

mit aktuellen Event-Daten gefüllt. Damit werden die „Werkbänke“ der beteiligten Webdienste-Anbieter an den verteilten Datenbestand angebunden.

Aus diesem Datenbestand können akteursspezifische Angebote zusammengestellt werden. Dazu ist vom Webdienste-Anbieter pro Akteur ein eigenes *Thema* zu erstellen oder aus vorgegebenen Themen auszuwählen oder anzupassen, 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.

2.2 Installation

- Ausrollen der Dateien in ein eigenes Verzeichnis `widget` auf dem Webserver, das der Webserver lesen kann,
- Ausrollen von ARC2, siehe [Arc2], das als Unterverzeichnis `arc2` an `widget` zu linken ist, und Einbinden der Datei(en) im Verzeichnis `plugins` in das Verzeichnis `arc2/plugins` (etwa durch Linken).
- 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**.

[1] Dieser Präfix kann in der Datei `Store.php` geändert werden.

(a) Import mittels SPARQL-Anfrage
Dies ist die vorkonfigurierte, einfacherere Variante. In der Datei `Store.php` befindet sich die Funktion `loadDataFromEndpoint`. In dieser kann die SPARQL-Anfrage angepasst werden, um eine Teilmenge der Event-Daten aus dem Event-Endpunkt (`http://leipzig-data.de/widget/sparql.php`) abzufragen. In der Vorkonfiguration werden alle Daten übernommen. Durch einfaches Ausführen der Datei `Store.php` findet der Import automatisch statt.

(b) Import mittels Dateidump
Dies ist die komplexere, aber stabilere Lösung. An die LD.Datenbasis (`http://leipzig-data.de/Data`) kann eine eigene in irgendeiner Weise (z.B. händisch) durchgeführte SPARQL-Abfrage (z.B. siehe `Queries.txt`) gestellt werden, um Event-Daten zu extrahieren. Das Ergebnis dieser Abfrage ist in einer Datei EventsDump.ttl zu speichern und in das Widget-Verzeichnis zu kopieren. Abschließend muss der Aufruf der Funktion `loadDataFromEndpoint` durch `loadDataFromFile` ersetzt werden und `Store.php` ausgeführt werden.

Eigene Daten können zusätzlich importiert werden, indem eine der eben genannten Funktionen erweitert wird, unter der Voraussetzung, dass die Daten in einem RDF-Format vorliegen.

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 kann die SPARQL-Anfrage der Funktion `filterData` verändert werden.

Für das Konvertieren von RDF nach JSON wurden die RDF-Management-Funktionalitäten von ARC2 um das Exhibit-JSON-Plugin `plugins/ARC2_ExhibitJSONSerializerPlugin.php` erweitert, welches die Serialisierung des geparsten RDF-Graphs im Zielformat realisiert. Dabei wird eine automatische Erkennung der Exhibit-Datentypen vorgenommen, sodass z. B. ein Datum auch den Datentyp `date` zugewiesen bekommt.

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

ACHTUNG: Für das korrekte Erstellen von `data.json` muss das PHP-Skript 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 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. ACHTUNG: 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 akteursspezifische Ausschnitt aus den Grunddaten,
- ggf einem eigenen Google Maps Schlüssel `gmaps_api_key.php`,

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