Handbuch für Admins „Interaktiver Haushaltsrechner“

18. Mai 2015

Inhaltsverzeichnis

1 Einleitung

In den folgenden Seiten wird erklärt, welche Vorraussetzungen es gibt, wie man eine Drupal Instanz auf einem Server aufbaut und zu unserem Drupal anpasst, welche Module mit welchen Einstellungen wie verwendet werden und wie unser Code funktioniert und genutzt wird.

2 Vorraussetzungen

In diesem Projekt kamen Drupal 7.36 und Virtuoso Universal Server 7 zum Einsatz. Diese sind für die gängigen Betriebssysteme frei herunterladbar. Zudem wird ein http-Server mit PHP-5.x Unterstützung benötigt.

2.1 Vorrausetzungen Drupal

Drupal ist zwar ein webserverunabhänigiges Framework. Allerdings wird ein Apache 1.3 oder 2.x empfohlen, da Drupal mit diesem Server entwickelt wurde und somit mehr Erfahrungsberichte von Seiten der Community existieren, wodurch man bei Problemen auf verschiedene Foren zurückgreifen kann. Für die Installation sollte mindestens 60 MB Speicher auf dem Server frei sein und das Modul mod_rewrite muss innerhalb von Apache aktiviert sein. Da Drupal in PHP geschrieben wurde, muss auf den Server PHP (für Drupal 7.36 mind. PHP 5.2) vorhanden sein. Als Datenbankanbindungen kommen MySQL 5 sowie PostgreSQL 7.4 in Frage.

3 Daten

3.1 Virtuoso

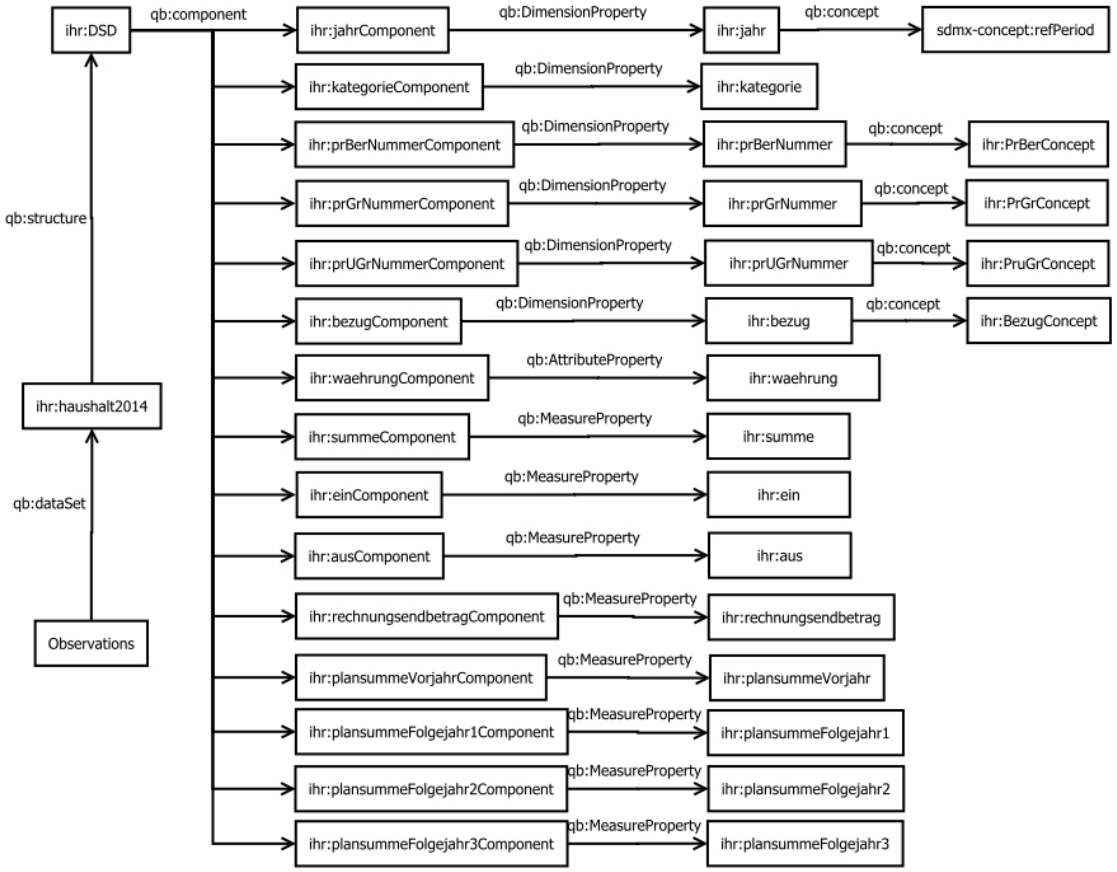
Die Datentripel des RDF Data Cubes müssen in einem Triplestore abgelegt werden und ein Endpunkt zur Abfrage bereit gestellt werden. Zu diesem Zweck kam die Software Virtuoso Universal Server 7 (`http://www.openlinksw.com/`) zum Einsatz. Die Installation einer Drupalinstanz wird in deren Dokumentation beschrieben.

Nachdem der Virtuoso Universal Server aufgesetzt ist, kann das Backend im Browser unter *[Serverroot]*:1524 aufgerufen werden. Die Haushaltsdaten können in einen Graphen geladen werden und über den zugehörigen SPARQL-Endpunkt unter *[Serverroot]*:1524/sparql abgefragt werden. Dazu muss man im Virtuoso Backend zum sogenannten *Conductor* wechseln. Ist man in den Conductor eingeloggt, kommt man über den Menüpunkt *Linked Data > Quad Store Upload* zur Uploadansicht. Hier muss zunächst die Datei ausgewählt werden, die die Finanzdaten enthält, und dann der *Graph Name* auf den gewünschten Wert gesetzt werden. In unserem Projekt wurde der Graph Name *[Serverroot]*/leipzighaushalt gewählt. An diese Adresse wurde die Turtle-Datei hinterlegt, so dass man die Daten in Textform einsehen kann. Hierbei ist zu beachten, dass der Name des Graphen, der abgefragt wird, genauso wie die Adresse des SPARQL-Endpunkts mit den Angaben im Software-Controller übereinstimmen müssen.

3.2 Struktur des RDF Cubes

Die vom Auftraggeber zur Verfügung gestellte Datengrundlage für unser Projekt besteht aus mehreren Excel-Dateien, die per dynamischer Listenausgabe aus SAP exportierte relevante

Daten (aus dem Jahr 2014) enthalten. Die Datenbasis wurde schrittweise in das RDF-Data-Cube-Format übertragen.



4 Drupal

4.1 Drupal aufsetzen

Drupal 7.36 diente in diesem Projekt als Basis-Framework. Zum Herunterladen und Installieren von Drupal kann die Dokumentation unter `https://www.drupal.org/` zu Hilfe gezogen werden. Da die Dokumentation dort zur grundlegenden Installation ausreichend ist, soll hier nicht weiter darauf eingegangen werden.

4.2 Module

Um den Funktionsumfang von Drupal auf unsere Bedürfnisse anzupassen, mussten einige Module zusätzlich installiert werden. Diese können über die Webseite `https://www.drupal.org/` heruntergeladen und eingefügt werden. Eine Anleitung hierzu findet sich unter `https://www.drupal.org/documentation/install/modules-themes`. Die Module sind alle unter der Gnu GPL erhältlich. Folgend eine Liste aller verwendeten Module:

1. jQuery Update

Dieses Modul stellt die aktuelle Version von jQuery zur Verfügung, die für *bootstrap* benötigt wird.

2. advanced forum
   Das Advanced Forum ergänzt das von Haus aus in Drupal integrierte Forum um einige Funktionen, die bei anderen Forensoftwares standardmäßig enthalten sind. Nach Aktivierung des Moduls können über *Struktur > Foren* nach der gewünschten Gliederung Foren und Ordner, in die Foren eingeordnet werden können, erstellt werden. Unter *Konfiguration > Inhaltserstellung > Advanced Forum* können grundlegende Einstellungen, wie das Design des Forums (im Projekt wurde das „Bootstrap"-Theme verwendet, welches wiederum ein eigenes Modul darstellt) angepasst werden.

3. advanced forum bootstrap
   Das für unser Projekt verwendete Design für das Forum – setzt Advanced Forum voraus.

4. flag / flag abuse
   Diese Pakete dienen zum Erstellen von Flags für das Melden von Kommentaren und Beiträgen. Dieses Paket ist notwendig zur Erstellung von Flags für das Melden von Beiträgen im Forum. Hierzu wurden im Projekt Flags *Comment Abuse* aus den Standardflags und die die eigens erstellte Flag *Topic Abuse* zum Melden von Kommentaren bzw. Themen im Forum verwendet. Zum Erstellen einer Flag dient der Menü-Eintrag *Struktur > Flags > Add Flags*.

5. views
   Mit Views lassen sich Ansichten gefilterter Inhalte erstellen. Zum Beipiel zum Anzeigen aller gemeldeten Kommentare oder Themen. Das Erstellen einer View sei hier am Beispiel des Views für gemeldete Kommentare erklärt.
   In der Adminansicht unter dem Menüpunkt *Struktur > Ansichten > Neue Ansicht hinzufügen* kann eine neue Ansicht erstellt werden. Dieser neuen Ansicht kann im geöffneten Dialog eine *Ansichtsname* gegeben werden und ein Pfad angegeben werden, unter dem die Ansicht später erreichbar ist. In unserem Fall wurde der Pfad *gemeldete-kommentare* gewählt. Nach Klick auf *Fortfahren und Bearbeiten* wird festgelegt, welcher Inhalt auf diesem *View* angezeigt wird. Für die Ansicht aller gemeldeten Kommentare bietet sich das Format Tabelle an. Dann kann unter *Felder* festgelegt werden, welche Eigenschaften der Kommentare in der View angezeigt werden. Im Projekt wurden *Kommentar: Titel*, *Flag: Kennzeichnungs-Link* und *Flags: Kennzeichnungs-Zeitpunkt* gewählt. Weiterhin muss unter dem Menüpunkt *Erweitert > Beziehungen* eine Beziehung zur Melde-Flag hinzugefügt werden. Mit Klick auf *hinzufügen* kann der Eintrag *Flags:Kommentar Flag* ausgewählt werden. Im folgenden Fenster kann man noch einen Bezeichner für die Flag einführen, die Kennzeichnung Abuse und den Punkt *beliebiger Benutzer* anwählen. Anschließend kann unter Filterkriterien ausgewählt werden, dass nur solche Kommentare angezeigt werden, die geflagt wurden, indem man den Eintrag *Flag Kennzeichnung* hinzufügt.

6. printfriendly

7. votingapi
   Hier können wichtige Einstellungen für das Rate Widget bestimmt werden. Es kann bestimmt werden, wann anonyme Nutzer erneut von einem Rechner wählen könnne (wenn sie es denn dürfen (dies wird für jedes Rate Widget einzeln bestimmt)). Gleiches

kann für registrierte Nutzer bestimmt werden. Bei hoher Serverlast können auch die Berechnungszeiten der Bewertungen auf bestimmte Cron-Zeiten festgelegt werden.

8. rate
   Mit rate können verschiedene Beiträge bewertet werden. Um ein neues Rate Widget zu erstellen, geht man auf *Struktur > Rate Widgets*. Dort kann man neue Rate Widgets bearbeiten und erstellen. Es kann Name, Tag, Inhaltstyp etc. festgelegt werden.

9. statistics counter
   Zusätzlich zu Installation sollte unter *Konfiguration>System> Statistiken* die Optionen *Zugriffsprotokoll aktivieren* und *Inhaltsabrufe zählen* aktiviert werden. Danach können Besucherstatistiken unter dem Menüpunkt *Berichte* eingesehen werden.

10. relation, ctools
    Helper-Module, die für Funktionen der anderen, essentiellen Module benötigt werden.

4.3 IHR-Theme hinzufügen

Das Design der Website basiert auf dem freien CSS Framework „Bootstrap“, das Gestaltungsvorlagen für Typografie, Formulare, Buttons, Tabellen etc. und optionale JavaScript-Erweiterungen enthält. Für die Projektseite wurde ein Basis-Theme verwendet (`https://www.drupal.org/project/bootstrap`), das die grundlegenden Elemente des Frameworks in die Funktionalität von Drupal einbindet. Zur besseren Annäherung an die offizielle Website der Stadt wurde ein sogenanntes „Subtheme“ erstellt, welches das oben genannte Theme ergänzt.

Zur Aktivierung des Themes reicht es, das Bootstrap-Theme gemeinsam mit dem IHR-Theme in den theme-Ordner der installierten Drupal-Instanz zu laden (Standard: */themes/*) und anschließend im Design-Administrationsmenü zu aktivieren.

4.3.1 Menüs erstellen

Die Menüs der Website werden über den Punkt *Struktur > Menüs* administriert. Dabei stellt das Menü mit dem Namen „Main menu“ das Hauptmenü im Kopfbereich der Seite dar – zusätzlich dazu kann ein Menü in der Fußleiste erstellt werden, in dem Links wie das Impressum, Datenschutzerklärung und Kontaktformular abgelegt werden können. Dazu muss unter oben genanntem Menü auf „Menü hinzufügen“ geklickt werden und der Titel so gewählt werden, dass im rechten Bereich der maschinenlesbare Name *menu-footer-menu* entsteht. Danach können Links hinzugefügt werden, die dann im rechten unteren Fußbereich der Website erscheinen.

4.3.2 Pages erstellen

Im Releasepaket sind im Ordner „Drupal Pages“ drei Dateien enthalten, die nach Installation von Drupal über *Inhalt > Inhalt hinzufügen > Basic Page* eingepflegt werden können. In *EinnahmenDia.html* und *AusgabenDia.html* ist der Code für die beiden Diagramme enthalten. Wichtig ist hierbei, als Textformat *Full HTML* auszuwählen.
In der Datei *indexpage.php* ist der Inhalt der bei Projektende erstellten Startseite enthalten. Hier ist bei Hinzufügen *PHP Code* auszuwählen.

5 Code

5.1 Grundlegender Aufbau

Da die Kernfunktion des Projekts auf einer festen und einheitlichen Basis von Datensätzen basiert, haben wir uns an das MVC-Architekturmodell gehalten. Dabei ist die Aufgabe des „Models“ bei der Datenvisualisierung, die Daten aus einer RDF Datenbank zu lesen, sie in ihre Bestandteile zu zerlegen und sie dem View zugänglich zu machen. Dazu wird die BorderClound SPARQL 1.1 PHP Bibliothek genutzt. Das „View“ benutzt die Informationen, die es vom „Model“ erhält, um die Daten entsprechend zu visualisieren (z. B. in einem Kreisdiagramm). Daneben ist die zentrale Aufgabe des „Views“ die Verwaltung von Formularen und Dokumenten zur allgemeinen und übersichtlichen Informationspräsentation für die Benutzer. Der „Controller“ reagiert nach Signalübermittlung des Views und führt die enstprechenden - d. h. vom Nutzer gewünschten - Anfragen aus.

Sämtlicher im Laufe des Projekts entstandener Code (und die zur Verbindung mit dem SPARQL-Endpunkt benötigte BorderCloud Bibliothek) ist im theme-Ordner unter */data/* abgelegt.

5.2 Interne Dokumentation

Als internes Dokumentationsformat wird einerseits für in PHP programmierte Elemente das Software-Dokumentationswerkzeug PHPDoc verwendet. Diese Dokumentation kann mit entsprechenden Tools wie z. B. phpDocumentor aus dem Quelltext generiert werden. Andererseits wird für die Dokumentation des Diagramm-Quellcodes JSDoc verwendet.

5.3 Diagramm

Das Diagramm ist unter Nutzung der Google Charts API geschrieben worden und dient der Ausgabe von Daten. Es hat Funktionen sowohl zur graphischen Darstellung als auch zur übersichtlichen textlichen Darstellung von Informationen. Diese Informationen erhält es stets von einem PHP-Script, dessen Position in der Variable *DiaURL* beschrieben wird. Zum Erhalt der jeweiligen Daten wird Ajax verwendet. Am Anfang wird nur der äußerste Diagrammring initialisiert. Diesem werden verschiedene Eventlistener zugeordnet, unter anderem ein selectHandler (die Funktion im Quellcode heißt direkt so). Diese Funktion ist für den Aufbau weiterer, innerer Diagrammringe zuständig. Sie wird aktiviert, sobald ein User auf eines der Segmente des Diagramms klickt. Dann wird eine Ajax-Anfrage erstellt, die Daten für den zusätzlichen Ring anfordert. Die Daten kommen als String an und müssen deshalb mit Json.parse in ein Array umgewandelt werden. Anschließend werden die Daten nochmal mit der formatTable Funktion aufbereitet, um das Diagramm übersichtlicher zu gestalten. Der Quelltext des Diagramms kann größtenteils kopiert und in eine Node von Drupal eingefügt werden. Nachfolgendes muss dabei jedoch beachtet werden:

5.3.1 Globale System-Arrays

- *colorRing*: speichert die Farbwerte der für die Funktion genColorObj
- *selectedArr*: speichert den Auswahlpfad des Benutzers

- *chartArray*: speichert alle vorhandenen Diagramme; wird z.B. in selectHandler benötigt, um eine Referenz für die verschieden Diagramme zu haben
- *lengthCounts*: speichert die Anzahl der Segmente eines Diagramms (das Segment „sonstige“ wird immer mit gezählt!)
- *colorFields*: speichert das Farbobjekt von jedem Ring ab
- *selectListenerArr*: speichert die Funktion für den Eventlistener für 'select' von jedem Ring ab
- *mousemoveListenerArr*: speichert die Funktion für den Eventlistener (nicht den der Google API) für 'mousemove' von jedem Ring ab

5.3.2 Eine neue Datenquelle einfügen

Seine Daten bekommt das Diagramm grundsätzlich von einem PHP-Script, welches auf einem Server abgelegt ist. Dazu muss lediglich eine URL, die zu eben jenem Script führt, in der Variable DiaURL als String angegeben werden. Das Diagramm erwartet von eben jenem PHP-Script einen String der Form '[[TableheadNameString,TableheadNameString,TableheadNameString],[System NamenString1,Zahlenwert1,KlarnamenString1],[SystemNamenString2,Zahlenwert2,KlarnamenS tring2],....,[SystemNamenStringN,ZahlenwertN,KlarnamenStringN]]' als Input.

5.3.3 Farben des Diagramms verändern & Funktion: genColorObj

Sollen andere Farben für das Diagramm verwendet werden, müssen lediglich die Farbwerte aus dem Array *colorRing* ausgetauscht werden. Soll sich jedoch die Anzahl der verwendeten Farben verändern, so muss ein neuer Eintrag in colorRing angelegt werden und die Anzahl(11) der Einträge in der Funktion *genColorObj* verändert werden. Am schwierigsten ist es, die Farben eines Diagrammrings individuell festzulegen. Dazu muss ein Objekt angelegt werden, dessen Attribute die Farbgebung des Diagramms beschreibt. Lesen Sie dafür unter `https://developers.google.com/chart/interactive/docs/gallery/piechart` unter „Configuration Options > slices“ nach, wie ein entsprechendes Objekt auszusehen hat. Dieses muss dann statt der *genColorObj* Funktion eingesetzt werden *und* im Array *colorFields* unter dem Index des Diagramms abgespeichert werden.

5.3.4 Größenveränderung des Diagramms & Funktion: resizeHandler/repaintDiaRecursive

Der äußerste Ring des Diagramms wird mit der Größe, die vor der Funktion *drawChart* in der Variable *SIZE* steht, geladen. Eine weitere Anpassung der Größe findet erst statt, wenn die Größe des Browserfensters verändert wird. Dann wird die Funktion *resizeHaendler* aufgerufen, welche die neue Größe des Browserfensters ermittelt und die Größe des Diagramms verändert. In den Variablen *UpperWidthSize, lowerWidthSize, UpperHeightSize* und *lowerHeightSize* stehen die Maße des Browserfensters, zwischen denen das Diagramm verändert wird. Das Diagramm vergrößert und verkleinert sich dabei linear zwischen einem Maximalwert (maxSIZE) und einem Minimalwert (*minSIZE*). Anschließend wird die Funktion *repaintDiaRecursive* aufgerufen, welche sämtliche Diagramme neu zeichnet.

5.3.5 Infofeld & Funktion: mouseoverHandler

Jedes Mal, wenn ein User mit der Maus über das Diagramm fährt, wird die Funktion mouseoverHandler aufgerufen, welche das Infofeld unter dem Diagramm aktualisiert. Das Infofeld selbst wird dabei standardmäßig über die ID *infoFieldData* angesprochen.

5.3.6 Unterschied zwischen dem Ausgaben- und Einnahmen-Diagramm

Beide Diagramme sind sich sehr ähnlich, unterscheiden sich jedoch dahingehend, dass das EinnahmenDia positive Zahlenwerte erwartet, während das AusgabenDIA negative Zahlenwerte erwartet.

5.4 ModelController

Der Controller dient dazu, SPARQL-Anfragen an das Model zu übergeben, damit das Model die Daten aus dem Datacube auslesen kann. Dabei werden entweder die Ausgaben oder die Einnahmen der Stadt ausgelesen, je nachdem wie der Modus gesetzt ist. Zudem parst der Controller noch die richtigen Namen der Kostenbeschreibung hinzu.

5.4.1 EndpointHandler (= Model)

Im EndpointHandler werden die Klarnamen der Produktnummer hinzugefügt, zudem wird die URI so gegekürzt, dass sie nur die Beschreibung anzeigt. Dies geschieht in der Funktion *sparqling*. Dabei werden die Daten in JSON-Format geparst. Dieses Format wird für das Diagramm benötigt. Im Konstruktor der Klasse wird der Endpunkt zu dem Datacube gelegt, welche über die Variable endpoint festgelegt wird.

5.4.2 QueryWriter (= Controller)

In der Variable *prefix* werden alle benötigten Präfixe für die Abfrage gespeichert. Der Präfix *xsd* wird für die Typkonvertierung benötigt, während *qb* für die Zeitangaben im Cube benötigt wird.

In der Variablen *ihr_uri* wird die URI des Datacube gespeichert. Wird der Name des Datacube geändert, so muss auch diese Variablen geändert werden. Dabei bleibt *PREFIX ihr:* immer gleich, da ansonsten Fehler in der Abfrage entstehen. Das Array *attribut_array* zeigt alle Kategorien des Datacube in chronologischer Reihenfolge an. Diese werden für die Erstellung der Abfrage benutzt. Das Array *amount* wird für die Unterscheidung von Ausgaben und Eingaben benutzt. Die Variable *choose_amount* setzt den Modus, ob Einnahmen oder Ausgaben genutzt werden. Diese Variable wird über die Funktion *setAmount* entweder auf 0 (Einnahmen) oder 1 (Ausgaben) gesetzt. Die Funktion *firstRing(year)* erstellt den ersten Ring, der den Nutzer sieht. Es wird das Jahr des Haushalt als Parameter erwartet. Die Funktion summiert auch gleich die Einzelen Kategorien auf, sodass nur die Katagorien nicht doppelt vorkommen. Die Funktionen *sparqlTransformation(array category, year)* erstellt eine SPARQl-Anfrage dynamisch auf Grundlage des Pfades, welcher der Nutzer ausgewählt hat.