Mashups – Integration von Qlik in Apex Anwendungen

Moritz Schenkel
Syntegris information solutions GmbH
Neu-Isenburg

Schlüsselworte

Oracle Apex, Qlik-Sense, Qlik, JavaScript

Einleitung

Qlik Sense ist ein interaktives Tool zum Analysieren von Daten, weil es selbst Fachanwendern explorative Analysen ihrer Daten ermöglicht. In Unternehmen wird es gerade durch sein ease-of-use geschätzt. Im Hinblick auf die einfache und schnelle Datenerfassung offenbart Qlik allerdings Schächen. Apex wiederum kann genau in diesem Bereich glänzen. In diesem Vortrag soll deshalb gezeigt werden, wie Apex mittels der Qlik Mashup JS API in Apex integriert werden kann. Dabei wird die Einbindung der ursprünglichen Apex Eingabemaske bis hin zur vollständigen Integration von Qlik Elementen in eine Apex Page gezeigt.

Qlik-Sense

Mit der Veröffentlichung von Qlik-Sense im Jahr 2014 ergänzt QlikTech sein Produktportfolio im Bereich der BI-Analyse Tools. Qlik-Sense basiert auf derselben Engine wie QlikView, stellt jedoch eine neue, modernere, auf HTML5, CSS und JavaScript basierende Oberfläche zur Verfügung. Die Engine arbeitet mit einer In-Memory Technologie, bei der alle Daten und Kalkulationen im Hauptspeicher abgelegt werden. Es ermöglicht Daten aus verschiedenen Quellen zu laden und in einem assoziativen Modell zu verknüpfen.

Abb. 1: Assoziatives Datenmodell (Quelle: https://www.prisma-informatik.de/erp-blog/2016/06/unternehmensdaten-richtig-nutzen-mit-dem-assoziativen-modell-von-qlik/)

In Abbildung 1 wird eine Stärke bei der Analyse von Daten mit Qlik sichtbar: Die Art wie Daten verknüpft werden, entsprechen im SQL Dialekt einem „Full Outer Join“ sodass wirklich alles mit allem verknüpft ist. Wird ein Attribut gewählt und anschießend gefiltert, zeigt Qlik durch eine graue Färbung an, dass zu diesem Attribut keine zugehörigen Werte gefunden wurden. Dies ermöglicht

gerade in Dashboards einen guten Überblick über die Datenstruktur und bietet einen direkten Mehrwert für den Endanwender.

Qlik-Sense kann als On-Premise oder Cloud Lösung lizensiert werden. Test sind aber auch mit einer kostenlosen Desktop Variante (mit annähernd vollem Funktionsumfang) möglich.

Integration von Qlik in Apex

Da die Oberfläche von Qlik-Sense in HTML/CSS/JavaScript geschrieben ist, scheint dessen Integration in Apex Anwendungen im ersten Moment einfach. Mittels des Single Configurators lassen sich einfache Grafiken auch sehr schnell und einfach in iFrames einbinden. Das ermöglicht jedoch keinerlei direkte Interaktion der Qlik Engine mit den Eingabemasken in Apex. Um eine Interaktion zu erreichen, müssen Mashups erstellt und implementiert werden.

Voraussetzungen

QlikSense Hub: Um ein Qlik Dashboard im Ganzen oder auch Teile eines solchen zu implementieren, muss dieses im Hub erstellt worden sein.

QlikSense Dev-Hub: Ist die Anwendung im Hub angelegt, kann im Dev-Hub das zugehörige Mashup erstellt werden. Der Dev-Hub generiert eine Beispiel-Webseite, welche aus einem Template und möglichen Implementierungen (Applikationsbestandteilen) besteht. Im vorliegenden Anwendungsfall ist es nicht notwendig viel Arbeit in das Design des Mashups zu legen, da wir die einzelnen Komponenten in unsere Apex Anwendung einbetten wollen. Das eigentliche Layout wird daher in Apex bestimmt. Es ist jedoch sinnvoll zumindest die gewünschten Teile der Applikationen in das Template zu ziehen, um somit Zugriff auf die korrekten Objekt-Ids zu haben. Abbildung 2 zeigt einen Screenshot des Dev-Hubs mit automatisch generiertem Javascript Code.

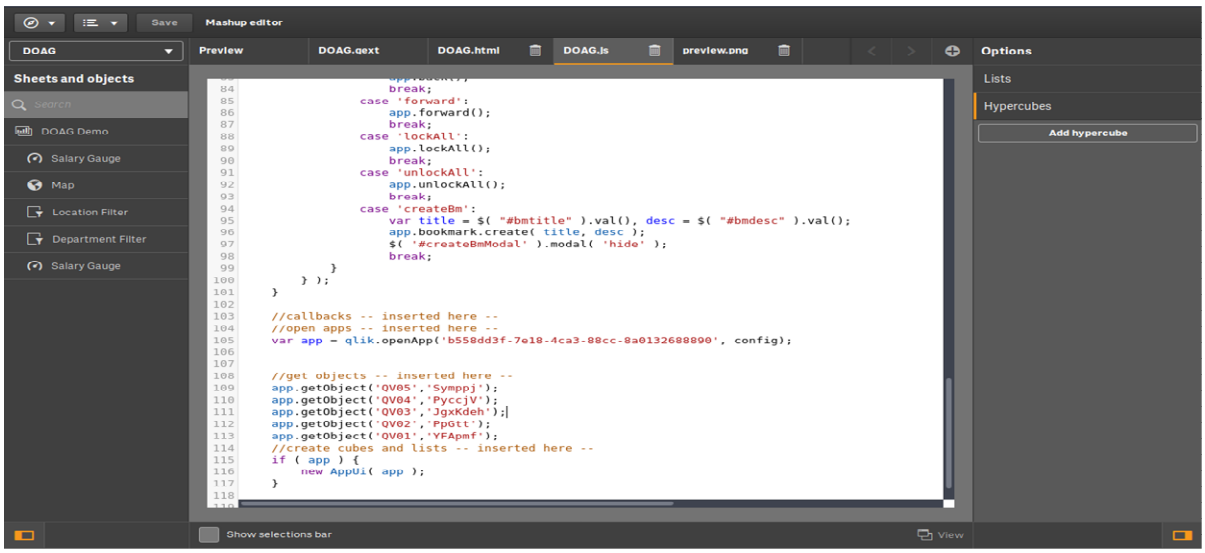

Abb. 2: Dev-Hub mit geöffnetem Javascript File

Qlik Management Console (QMC): Um die Nutzung von Qlik Bibliotheken direkt vom Qlik-Server zu ermöglichen müssen CORS Header Einstellungen am virtual Proxy vorgenommen werden. Dazu muss in der QMC der Eintrag Virtual Proxy geöffnet und der jeweilige Proxy zum Bearbeiten ausgewählt werden. Im darauffolgenden Dialog ist unter dem Property „Advanced“ folgender Eintrag unter „Additional response headers“ vorzunehmen:

Access-Control-Allow-Origin: [URL des Apex Servers]
Access-Control-Allow-Credentials: true

Außerdem ist die URL des Apex Servers auch in der darunter stehenden Host White List einzutragen.

Implementierung

In diesem Beispiel gehen wir von einer Standard Apex Seite mit Interactive Grid aus, wie sie auch der Wizard erstellt. Als Datenbasis wurde das Oracle Test-Schema HR gewählt, wie es auch auf github frei zur Verfügung steht (https://github.com/oracle/db-sample-schemas). Wer selbst nicht basteln möchte, kann die Sample-Application auch im weiteren Anhang auf der DOAG-Seite herunterladen.

Nach dem Anlegen müssen der Seite natürlich die nötigen JS/CSS Files zur Verfügung gestellt werden. Die jeweiligen URLs kann man den Sample Files im Dev-Hub entnehmen. Zu Demozwecken wurden URLs direkt auf der Page hinterlegt.

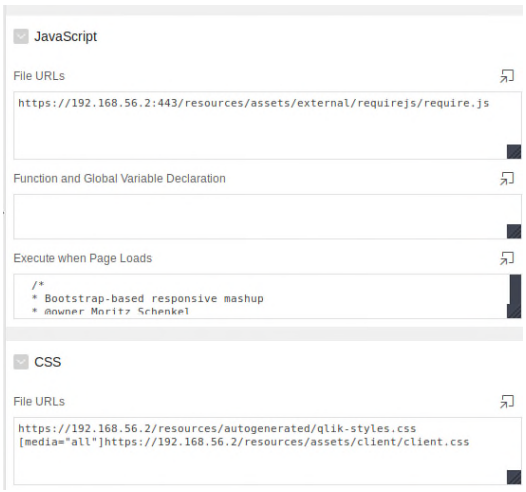

Abb.3: Screenshot der JS/CSS File-URLs

Wichtig ist hierbei auf jeden Fall die require.js Version von Qlik einzubinden, welche sich von der bereits in Apex enthaltenen Version unterscheidet und weitere notwendige Funktionen enthält.

Die vom Dev-Hub erzeugte JavaScript Datei, welche die Applikationslogik enthält, wurde in der Demo direkt im „Execute when Page Loads“ Dialog hinterlegt. Es ist natürlich auch möglich den Aufruf in einem eigenen JavaScript File zu hinterlegen. Da viele der enthaltenen Funktionen nicht mit der geplanten Seite übereinstimmen, wurden Großteile des Codes weg gelassen. Für erste Tests ist dies aber nicht notwendig, da das entsprechende Event einfach nicht gebunden wird, wenn keine passende ID gefunden wird.

Um ein Layout für die Applikation zu schaffen, welches durch die Grafiken von Qlik gefüllt werden kann, wurde eine leere „Static Content“ Region angelegt. Damit Qlik diese füllt, ist lediglich eine ID zu setzen. Im JavaScript Code des Dev-Hubs werden diese standardmäßig mit QV01 bis QV05 benannt. Es müssen daher 5 Regions erstellt werden, die jeweils eine ID zugeordnet bekommen.

Nun ist schon der Großteil der Arbeit geschafft. Startet man die Applikation zum ersten Mal, können eine Vielzahl von Problemen auftreten. Um diese etwas geordneter abarbeiten zu können, werden mögliche Fehlerquellen in einem FAQ Block am Ende des Vortrags zusammengefasst. Im Folgenden wird noch kurz auf den Code zur Interaktion von Qlik und Apex eingegangen.

Der wichtigste Punkt damit „require“ überhaupt weitere Codeteile anfragen kann, ist die require config. In diesem Beispiel sind viele Teile hartkodiert. Anders als im Template ist dies zwingend notwendig, da der Qlik Server nicht dieselbe IP hat wie der Apex Server.

Ist require konfiguriert kann der eigentliche Code aufgerufen werden. Wirklich relevant für die Anzeige ist dabei nur ein kleiner Teil des Codes:

```
//open apps
var app = qlik.openApp('b558dd3f-7e18-4ca3-88cc-8a0132688890',
config);

//get objects
app.getObject('QV05','Symppj');
app.getObject('QV04','PyccjV');
app.getObject('QV03','JgxKdeh');
app.getObject('QV02','PpGtt');
app.getObject('QV01','YFApmf');
```

Hier wird in der ersten Zeile das app-Object über die Qlik-API abgerufen. Es stellt eine Verbindung zur Qlik Engine her. Über app.getObject kann nun, die einzelne Qlik Komponente an die ID der Region gebunden werden.

Die wenigen Zeilen genügen, um die Qlik Mashups in der Apex Anwendung verfügbar zu machen. Wird in einem der Komponenten dann eine Selektion vorgenommen, wird diese an die Engine übertragen und alle weiteren Komponenten - wenn relevant - automatisch aktualisiert.

Für die Interaktion mit dem Interactive Grid ist noch ein wenig Applikationslogik erforderlich. Diese wird leider nicht von Haus aus vorhanden und wird deshalb über eine Function integriert, welche sämtliche UI Logik abhandelt.

```
function AppUi ( app ) {
  var me = this;
  this.app = app;
  // Clear all Filters
  $( "#clear" ).on( 'click', function () {
    app.clearAll();
    var filters = apex.region("grid").call("getFilters");
    for (var i in filters) {
      var filterId = filters[i].id;
      apex.region("grid").call("deleteFilter",filterId)
    }
  });
  //use event to do a reload of the qlik source, if the interactive
  grid saves state
  $("#grid").on("interactivegridsave", function(){
    app.doReload();
  });
  // Selection
  app.getList('SelectionObject', function(reply) {
    me.selections = reply.qSelectionObject.qSelections;
    me.render();
  });
}
```

Im Beispiel haben wir drei Fälle, in denen eine Interaktion notwendig ist. Das Setzen von Filtern/Selektionen, das Löschen von Filtern/Selektionen und das Speichern von Daten im Interactive Grid.

Schauen wir uns zuerst das Löschen von Filtern an: Mittels jquery wird ein Event an ein Object mit der ID „clear“ gebunden. Wenn der clear-Filters Button betätigt wird, führt das dazu, dass nachfolgender Code ausgeführt wird. Die Selektionen in Qlik zu entfernen ist einfach. app.clearAll() entfernt komfortabel alle Selektionen für den jeweiligen User auf dem Modell. Für die Filter auf dem Interactive Grid muss eine Schleife über alle Filter auf dem Grid geschrieben werden, um diese einzeln zu entfernen.

Ebenfalls denkbar simpel ist das Speichern im Interactive Grid. Da die Auswirkungen, die das Update hat, auch im Qlik Modell sichtbar werden sollen, macht man sich am besten das interactivegridsave-Event zu Nutzen und lässt die Qlik Engine ihre Datenquelle mit doReload( ) neu laden. Ein direktes Update der Daten im Hauptspeicher ist leider nicht vorgesehen, weshalb das gesamte Modell neu geladen werden muss.

Zuletzt noch ein Blick auf das Setzen von Selektionen und damit Filtern auf das interactive Grid. Hier wartet der größte Aufwand auf den Programmierer, da zuerst alle Selektionen ausgelesen und dann mittels der Grid API im richtigen Format übertragen werden müssen.

Mittels app.getList und dem Parameter „SelectionObject“ bekommt man von der API ein Object das alle gesetzten Selektionen enthält. Dieses wird direkt in einer anonymen Funktion aufgerufen, um es zu verarbeiten. Eine selbst geschriebene Render Funktion für AppUI Object löscht wie die clear Aktion erst einmal alle Filter vom Grid. Danach erfolgt ein Schleifendurchlauf durch alle Felder im SelectionObject. Da auch Felder ohne Selektion hierin erscheinen, wird zuerst noch überprüft, ob es überhaupt Selektionen auf diesem Feld gibt. Ist das der Fall muss ein Filter hierfür erstellt werden. Um mehrfach-Selektionen zu erlauben (z.B. Deutschland und Frankreich) wird ein IN Filter erzeugt. Damit die Einträge korrekt konkateniert werden, wird durch die Filtereinträge eines Feldes durchgeloopt und alle Einträge werden zunächst in einem Array abgelegt. Dieser kann dann in der addFilter Funktion des InteractiveGrids gejoint werden. Hierbei ist es wichtig die Einträge mittels des Unicode Characters \u0001 zu joinen, auch wenn der normale Filtereintrag ein Komma anzeigt.

FAQ – Frequently Asked Questions

Q: Die Seite lädt nicht und zeigt JavaScript Fehler an?

A: Oftmals liegt das an falschen oder nicht vorhandenen Einträgen im Virtual Proxy in der QMC. Die CORS und WhiteList Einträge müssen korrekt getätigt sein. Dadurch, dsas der Browser das Laden der JavaScript Files verweigert, können die Objekte nicht aufgerufen werden und verursachen JavaScript Fehler.
Wenn Qlik Desktop verwendet wird, ist weiterhin zu beachten, dass Qlik auf demselben Rechner wie die Apex Umgebung installiert sein muss, da der Netzwerkzugriff in Qlik Desktop deaktiviert ist.

Q: Die Seite lädt ohne JavaScript Fehler, ich sehe jedoch keine Qlik Elemente?

A: Es ist notwendig die Höhe der Regions festzulegen, in denen die Qlik Elemente gerendert werden. Dies kann einfach über CSS Einträge wie „#QV01 {height: 520px;}“ erfolgen. Setzt man keine Höhe rendert Qlik das Element mit 0 Pixel Höhe.

Q: Ich bekomme eine Qlik Fehlermeldung, dass ich nicht angemeldet bin?

A: Zuvor an Qlik anmelden, um somit ein Session Ticket zu bekommen oder in der QMC Lizenzen der anonymen Anmeldung zuweisen.

Kontaktadresse:
Moritz Schenkel
Syntegris information solutions
Hermannstraße 54-56
D-63263 Neu-Isenburg

Telefon:	+49 (0) 6102 – 29 86 68
Fax:	+49 (0) 6102 – 55 88 06
E-Mail	moritz.schenkel@syntegris.de
Internet:	www.syntegris.de