Datenintegration in ArcGIS Online mittels Koop Service Provider

Christian Kocourek[1], Manfred Spiess[1], Roland Grillmayer[1] und Manfred Mittelböck[2]

[1]Studiengang Informatik, FH Wiener Neustadt · 85790@fhwn.ac.at
[2]AIT – Austrian Institute of Technology / iSpace

Short paper

Zusammenfassung

Das Projekt Koop-Server stellt eine Technologie zu Verfügung welche es ermöglicht Datenquellen von Drittanbietern in Form eines ArcGIS Feature Services sowie weiteren Datenformaten, wie beispielsweise GeoJSON, darzustellen. Im Rahmen des vorliegenden Dokuments wird die Architektur von Koop skizziert und dessen Funktionalität erläutert. Des Weiteren wird der Vorgang zur Implementierung eines eigenen Datenproviders beschrieben und dessen Architektur dokumentiert. Abschließend wird die Integration des entstandenen Providers in eine existierende Instanz von Koop dargestellt. Der implementierte Provider bezieht sich auf das Schema einer vorgegebenen Datenbank. Im Falle einer eigenen Implementierung können jedoch Datensätze welche in einem beliebigen Datenbankschema vorliegen mithilfe von Koop exponiert werden. Die Anzahl und Art der Ausgabeformate sind ähnlich wie die Eingabeformate beliebig erweiterbar. Durch entsprechende Erweiterung des Providers ist die Ausgabe in mehreren gewünschten Formaten möglich.

Zusammenfassend lässt sich feststellen, dass die Verwendung von Koop einer lizenzfreie und einfache Datenintegration in ArcGIS Online ermöglicht. Somit stellt das Koop Projekt eine freie Alternative zur lizenzkostenpflichtigen ArcGIS Server Technologie dar. Das System lässt sich durch eigene Provider effektiv erweitern. Einziger Schwachpunkt des Koop Projektes zeigt sich in der spärlich verfügbaren Dokumentation der Technologie. Die Einarbeitung in dieses muss weitgehend anhand der Interpretation vorhandener Implementierungen und dem Source Code erfolgen.

1 ArcGIS Feature Service & GeoJSON Spezifikation

Der Koop-Server ist ein Webservice und bietet die Funktionalität Daten aus integrierten Providern nach Bedarf entweder als ArcGIS Feature Service oder als GeoJSON auszugeben. Ein GeoJSON Objekt besteht dabei aus einer Ansammlung von Objekten einer FeatureCollection. Als Geometrietypen stehen die in der Simpel Feature Spezifikation (OpenGIS Simple Feature Specification 1999, 2-26) definierten Geometrietypen zur Verfügung. Innerhalb eines Objekts sind geometrische und räumliche Eigenschaften unter dem Bezeichner geometry zu finden. Zusätzlich muss der Bezeichner Eigenschaften enthalten, welche zusätzliche Informationen über die Beschaffenheit des Objekts beinhalten (vgl. GEOJSON SPEZIFIKATION). Ein ArcGIS Feature Service besteht aus einer ArcGIS Feature Layer, welche beliebig viele ArcGIS Features enthalten kann. Eine Feature Layer enthält

AGIT – Journal für Angewandte Geoinformatik, 1-2015. © Herbert Wichmann Verlag, VDE VERLAG GMBH, Berlin/Offenbach. ISBN 978-3-87907-557-7, ISSN 2364-9283, doi:10.14627/537557046.

grundlegende Informationen über sich selbst, wie den Namen, die Typbezeichnung der Layer und die darzustellenden Felder. Des Weiteren beinhaltet eine Feature Layer zusätzliche Informationen zur grafischen Darstellung. Darin enthalten sind beispielsweise der Umfang der Karte, in welchem Maßstab die Karte minimal und maximal dargestellt werden kann sowie ein optionaler Copyright Text (vgl. GEOSERVICES REST SPECIFICATION VERSION 1.0).

2 Koop-Server

Eine Instanz des Koop-Servers besteht unter anderem aus einer Sammlung von Basisklassen im zentralen Modul Koop-Server, welches die Datenkonvertierung regelt. Die gewünschten Daten werden aus den integrierten Providern geladen und an die Methoden innerhalb des Moduls Koop-Server weitergeleitet. Dieses beantwortet die Anfrage und stellt die Daten als ArcGIS Feature Service zur Verfügung die in weiterer Folge problemlos in ArcGIS Online eingebunden werden können. In Abhängigkeit der Anfrage können die Daten auch als reines GeoJSON ausgegeben werden. Auf welche Daten im speziellen der Zugriff erfolgt, wird über den HTTP-Request festgelegt. Sämtliche enthaltene Module sind mittels Node.js implementiert, die Verarbeitung der HTTP-Requests wird von Express.js (vgl. TEIXEIRA 2012, 217 ff.) übernommen. Das Zusammenspiel der in Koop enthaltenen Komponenten ist in Abbildung 1 grafisch dargestellt.

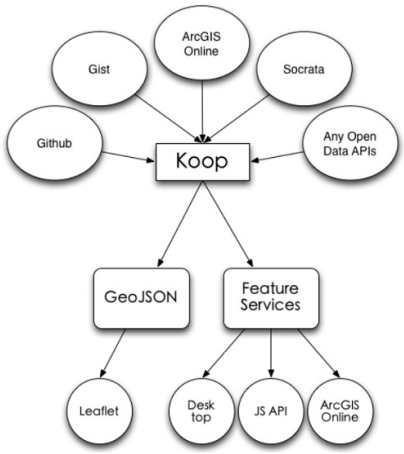

Abb. 1:
Zusammenspiel der Komponenten in Koop (HELM 2014)

2.1 Funktionalität

Bei einer Anfrage an den Koop-Server überprüft dieser das Format der angefragten Daten auf korrektes GeoJSON. Danach werden zur weiteren Verarbeitung die Datentypen innerhalb einer Feature Collection (vgl. GEOJSON SPEZIFIKATION) mit dem Schlüsselwort EsriFieldType versehen. Die so entstandenen Daten werden in ein vorgefertigtes ArcGIS Feature Service Template eingefügt und in eine interne Sqlite Datenbank geschrieben. Danach kann der jeweilige Provider das erstellte Feature Service aus selbiger Datenbank auslesen

und die erhaltene Anfrage damit beantworten. Des Weiteren benötigt der Koop-Server Zugriff auf eine externe Datenbank, welche zur Verarbeitung von Geodaten geeignet ist. Diese wird als Cache während des Konvertierungsvorgangs verwendet (HELM 2014). Der Zugriff auf den Koop-Server erfolgt immer über einen integrierten Provider.

2.1.1 Provider

Alle existierenden Provider sind in einem eigenen Github Repository verfügbar. Sie können wahlweise direkt aus diesen Repositories geklont oder mittels Node Package Manager (vgl. TEIXEIRA 2012, 9 ff.) in eine Instanz von Koop integriert werden. Jeder Provider besteht aus folgenden drei Komponenten:

- *Routen:* Enthält die vordefinierten URL-Parameter, welche angegeben werden müssen um auf die Daten des Providers zuzugreifen.
- *Controller:* Enthält die Logikkomponente des Providers. Der Controller entscheidet auf Basis der Anfrage, welche Funktionen ausgeführt werden.
- *Model:* Innerhalb dieser Komponente erfolgt der Abruf der angefragten Daten aus den jeweiligen Repositories oder der Koop-Server internen Datenbank sowie deren Rückgabe an den Controller.

Ein Provider kann über seine entsprechende URL mittels HTTP-Request angesprochen werden. Dabei folgt dem Namen des Providers der Name der dateninhabenden Organisation, beispielsweise *CityOfPhiladelphia*, das jeweilige Repository sowie der Ordner in welchem die gewünschten Daten liegen. Wird ein Provider angesprochen, kann die URL wie folgt aussehen:

http://koop.dc.esri.com/github/CityOfPhiladelphia/phl-open-geodata/
school_facilities::philadelphiaschool_facilities201302
(HELM, „Koop GitHub Provider 2014“)

In diesem Beispiel ist der Koop-Server unter der URL *koop.dc.esri.com* zu erreichen. Es wird der darin integrierte Provider *github* angesprochen und Daten der Organisation *CityOfPhiladelphia* angefordert. Diese befinden sich in dem Repository *phl-open-geodata*, in einem Ordner namens *school_facilities*. Der eigentliche Datensatz ist darin als *philadelphiaschool_facilities201302* hinterlegt.

Derzeitig werden bei der Installation von Koop folgende vier Provider automatisch in das Verzeichnis kopiert (HELM, „Esri Koop“ 2014):

- GitHub,
- ArcGIS Online,
- GitHub Gist,
- Socrata.

3 Problembeschreibung und Zielsetzung

Um Daten aus anderen als den bereits bei der Installation zur Verfügung stehenden Datenprovidern integrieren zu können, muss eigenständig ein Datenprovider implementiert werden. Im vorliegenden Fall soll dieser die Funktionalität bieten, Daten aus einer PostgreSQL

oder einer Sqlite Datenbank mit einem definierten Datenbankschema zu extrahieren und den Zugriff auf diese über Koop zu ermöglichen.
Nach der Konvertierung der angeforderten Daten in ein ArcGIS Feature Service sollen diese innerhalb der ArcGIS Online Plattform eingebunden werden können.
Ziel des Projekts ist es, einen Prototypen zur Datenverarbeitung mittels Node.js zu implementieren und diesen in einer Instanz des Koop-Servers zu integrieren. Der Prototyp und Koop sollen dabei unter Windows Server 2008 lauffähig sein. Es soll die Möglichkeit bestehen den zu implementierenden Provider systemunabhängig einsetzen zu können, sofern eine Instanz von Koop auf dem jeweiligen System existiert.

Zu den nicht angestrebten Funktionalitäten zählt das Bearbeiten der Daten innerhalb der Datenbank, sowie die Aktualisierung der Daten innerhalb der ArcGIS Online Plattform sollten sich diese verändern.

4 Methodik

4.1 Installation Koop

Eine lauffähige Koop Installation benötigt Python ab der Version 2.7. Dabei ist sicherzustellen, dass die dazugehörige Umgebungsvariable korrekt gesetzt wurde. Ein weiteres Software Requirements ist Node.js ab der Version 0.10.0 (HELM, „Esri Koop“ 2014). Des Weiteren muss eine Version der Visual C++ Redistributable installiert sein. Nach der Installation von Node.js kann Koop mittels Node Packet Manager auf das System geklont werden. Die Installation erfolgt über den Befehl npm install.

4.2 Architektur und Implementierung

Der zu entwerfende Provider besteht aus denselben drei Komponenten wie die bereits existierenden Datenprovider für Koop. In weiterer Folge wird die Funktionalität der einzelnen Komponenten erklärt sowie deren Verantwortung dargestellt. Abbildung 2 verdeutlicht das Zusammenspiel der einzelnen Komponenten innerhalb des Providers. Um einen HTTP-Request an den zu implementierenden Provider weiterleiten zu können, benötigt Koop eine Auflistung der entsprechenden Routen, über welche dieser zu erreichen ist.

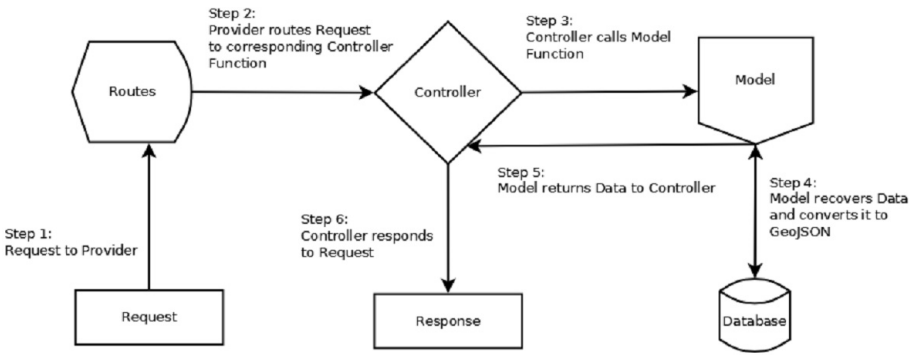

Abb. 2: Abhängigkeiten der Komponenten

4.2.1 Routes

Wie bereits in Kapitel 2.1.1 angesprochen, bildet die Definition der Routen den Einstiegspunkt in das System. Es soll möglich sein, durch die Übergabe von URL Parametern die entsprechenden Datensätze aus der jeweiligen Datenbank auszulesen. Mittels eigens definierten Routen ergibt sich des Weiteren die Möglichkeit gezielt entweder die PostgreSQL oder die Sqlite Datenbank anzusprechen. Wird diese Entscheidung nicht mittels der Eingabe der URL getroffen, greift der Provider standardmäßig auf die Daten beider Datenbanken zu. Außerdem kann mithilfe der Routen entschieden werden in welches Ausgabeformat die Daten konvertiert werden sollen. In Abhängigkeit der aufgerufenen Route leitet der Provider den Request an die zuständige Funktion innerhalb des Controllers weiter.

Die im Provider enthaltenen Routen wurden nach dem Schema */provider/format/databaseQuery* integriert. Dabei kann nach der Übergabe der Parameter für die Datenbankabfrage eine ID für die direkte Ausgabe einer einzelnen Layer des ArcGIS Feature Services folgen. Der darauffolgende Parameter enthält optional eine Callback-Funktion. Durch diese ist es beispielsweise möglich, die erhaltene Feature Layer direkt innerhalb des ausgeführten Scripts weiter zu verarbeiten. Eine demonstrative Abfrage nach diesem Schema ist im Kapitel 4.4.1 beschrieben.

4.2.2 Controller

Der Controller ist die zentrale Komponente der Applikation. Zum Systemstart wird an dieser Stelle die Verbindung zu den integrierten Datenbanksystemen aufgebaut. Dafür werden die Module node-postgres (CARLSON 2014) sowie node-sqlite*3* (MITHGOL 2015) verwendet. Erfolgt ein HTTP-Request an den Provider, wird aufgrund der aufgerufenen Route die dazugehörige Funktion aufgerufen. Der Controller speichert falls vorhanden eine übergebene Callback-Funktion. Wird zum Beispiel der Datenprovider über die ArcGIS Online-Plattform aufgerufen, beinhaltet die Callback Funktion ArcGIS Online spezifische Befehle für die Weiterverarbeitung der Daten innerhalb dieser Plattform.
Im Anschluss werden die notwendigen Parameter gemeinsam mit dem Objekt für den Zugriff auf die Datenbankan das Model übergeben. Nachdem die Befehle innerhalb der Model Komponente ausgeführt worden sind, erhält der Controller von jedem Model die ausgelesenen Daten. Abschließend wird die Funktion des Basiscontrollers, die im Modul Koop-Server enthalten ist, aufgerufen. Dieser regelt die Konvertierung der Daten in ein ArcGIS Feature Servicesowie die Beantwortung des HTTP-Requests.

4.2.3 Model

In der Einstiegsfunktion des Models werden die erhaltenen URL Parameter ausgewertet und in Form einfacher JavaScript Objekten gespeichert. Dies geschieht innerhalb einer Schleife, welche die Parameter mit ihren dazugehörigen Werten ausliest. Danach erfolgt die Anfrage an die Datenbank. Bei erfolgreicher Anfrage an die Datenbank werden die ausgelesenen Rohdaten an eine Funktion übergeben, welche eine Konvertierung in GeoJSON vornimmt. Zu diesem Zweck wird das Modul GeoJSON.js (CASEY 2015) verwendet. Um die Darstellung einer Bounding Box zu ermöglichen, werden die benötigten Koordinaten im resultierenden Objekt vermerkt. Ist dieser Vorgang abgeschlossen, erfolgt der Eintrag in die interne Datenbank des Koop-Servers. Danach werden die Daten an den Controller zurück übergeben.

Zusätzlich besteht die Möglichkeit, die Datensätze an dieser Stelle in andere Formate zu übertragen. Für Demonstrationszwecke sind die Datenformate KML (vgl. GOOGLE DEVELOPERS, „KML Reference“) und CSV (SHAFRANOVICH 2005) bereits implementiert.

4.3 Integration

Um den entstandenen Datenprovider in eine Instanz von Koop zu integrieren, müssen die Komponenten in den richtigen Ordnern abgelegt werden. Des Weiteren ist es notwendig, dass die Module Controller und Routes gleich benannt werden, um die Funktionalität von Express.js sicherzustellen. In diesem Fall ist der Controller im Ordner controller unter index.js zu finden. Des Weiteren muss der Ordner, welcher die eben benannten Unterordnern enthält, mit dem Präfix koop- benannt und in den Pfad *../koop/node-modules/* gespeichert werden. Wird Koop gestartet, erkennt das System durch selbiges Präfix die integrierten Provider.

4.3.1 Use Case Demonstrator

Als Beispiel wird eine Wetterstation gewählt, deren Messdaten in einer Postgres Datenbank als Zeitreihe abgelegt werden. Als darzustellende Phänomäne wurden Temperatur und Niederschlag gewählt.

Als realitätsnahe Abfragestrategien wurden die Filterung auf die ID bestimmter Sensoren sowie die Begrenzung der Abfrage hinsichtlich eines bestimmten Zeitintervalls festgelegt.
Ziel war es, basierend auf der Abfragestruktur folgende Anfrage an das WebService zu ermöglichen:

„gttps:/myserver.org//*ispace/FeatureServer/sensor.id=1&sensor.id=2&phen.id=4&phen.id=5&phen.timestamp>=01-01-2013/0“.

Dem Namen des Providers *„ispace“* folgt die Angabe über das Ausgabeformat der Daten. *„FeatureServer“* steht dabei für die Ausgabe als ArcGIS Feature Service. I weiterer Folge werden die gewünschten Datenfilter spezifiziert. Die Sensoren, welche diese Daten erfasst haben werden mittels der, in der Datenbank vergebenen, ID erfasst und verknüpft. Dies resultiert in *„sensor.id=1&sensor.id=2“*. Wie angeführt sind für den Demonstrator die interessanten Phänomene Temperatur und Niederschlag. In diesem Fall sind Temperatur und Niederschlag in der Datenbank mit der ID 2 bzw. 3 gekennzeichnet. Daraus ergibt sich für die Abfrage an Koop *„phen.id=3&phen.id=4“*. Als abschließende Filterdefinition erfolgt die der Zeit mit *„phen.timestamp>=01-01-2013“*, damit nur jene Phänomene, die seit Beginn des Jahres 2013 aufgezeichnet wurden ausgegeben werden. Das Anfügen von */0* am Ende der URL gibt dabei an, dass aus dem entstehenden Feature Service die darin enthaltene Feature Layer mit der ID 0 ausgegeben werden soll. Ist die Ausgabe des kompletten Feature Service erwünscht, kann dieser Parameter verworfen werden. Erfolgt der Aufruf über ArcGIS Online/ArcGIS Rest API, wird die daraus resultierende Feature Layer dargestellt.

Zusammenfassung

Mit dem Koop-Server stellt Esri ein Open-Source-Code-Projekt zur Verfügung, welches es ermöglicht, unterschiedlichste Datenquellen mit raumzeitlichen Inhalten für das Produktportfolio Esri ArcGIS Desktop und Esri ArcGISOnline in Form von ArcGIS Feature Services (Vektordiensten) und GeoJSON zu öffnen. Der wesentliche Vorteil dieses Ansatzes liegt darin, dass sich die Anwender dieses Code-Projektes sicher sein können, dass sie ihre Datenquellen mit diesem Werkzeug transparent und Open Source in die firmenspezifischen Lösungen im ‚Lesemodus' integrieren können. Der hier vorgestellte Ansatz kann Organisationen, die über heterogene Datenlandschaften verfügen dabei unterstützen, diese Datenquellen auch Open Source für die ‚Esri ArcGIS Welt' zu öffnen. Wesentlicher Wermutstropfen ist dabei leider die spärliche, schwer nachvollziehbare Dokumentation dieses Code-Projektes, die eine große Hürde beim Einstieg darstellt.

Literatur

CARLSON, B., brianc/ node-postgres. https://github.com/brianc/node-postgres (2/2015).
CASEY, T., caseypt/ geojson.js. https://github.com/caseypt/GeoJSON.js (2/2015).
DEVELOPERS.GOOGLE.COM, KML Reference. https://developers.google.com/kml/documentation/kmlreference (12/ 2013).
Esri White Paper (2010), Geoservices rest specification version 1.0.
GEOJSON SPEZIFIKATION, http://geojson.org/geojson-spec.html (2/2015).
HELM, C., Esri koop. https://github.com/Esri/koop (2/2015).
HELM, C., Koop github provider. https://github.com/chelm/koop-github (2/2015).
MITHGOL, mapbox/ node-sqlite3. https://github.com/mapbox/node-sqlite3 (2/2015).
OpenGIS Simple Features Specification, For SQL Revision 1.1, 1999, 2-26. https://portal.opengeospatial.org/files/?artifact_id=829, (4/2015).
SHAFRANOVICH, Y., Rfc 4180 – common format and mime type for commaseparated values (csv) files. https://tools.ietf.org/html/rfc4180 (10/2005).
TEIXEIRA, P. (2012), Professional Node.js – Building Javascript Based Scalable Software. John Wiley & Sons, Inc., Indianapolis, Indiana.