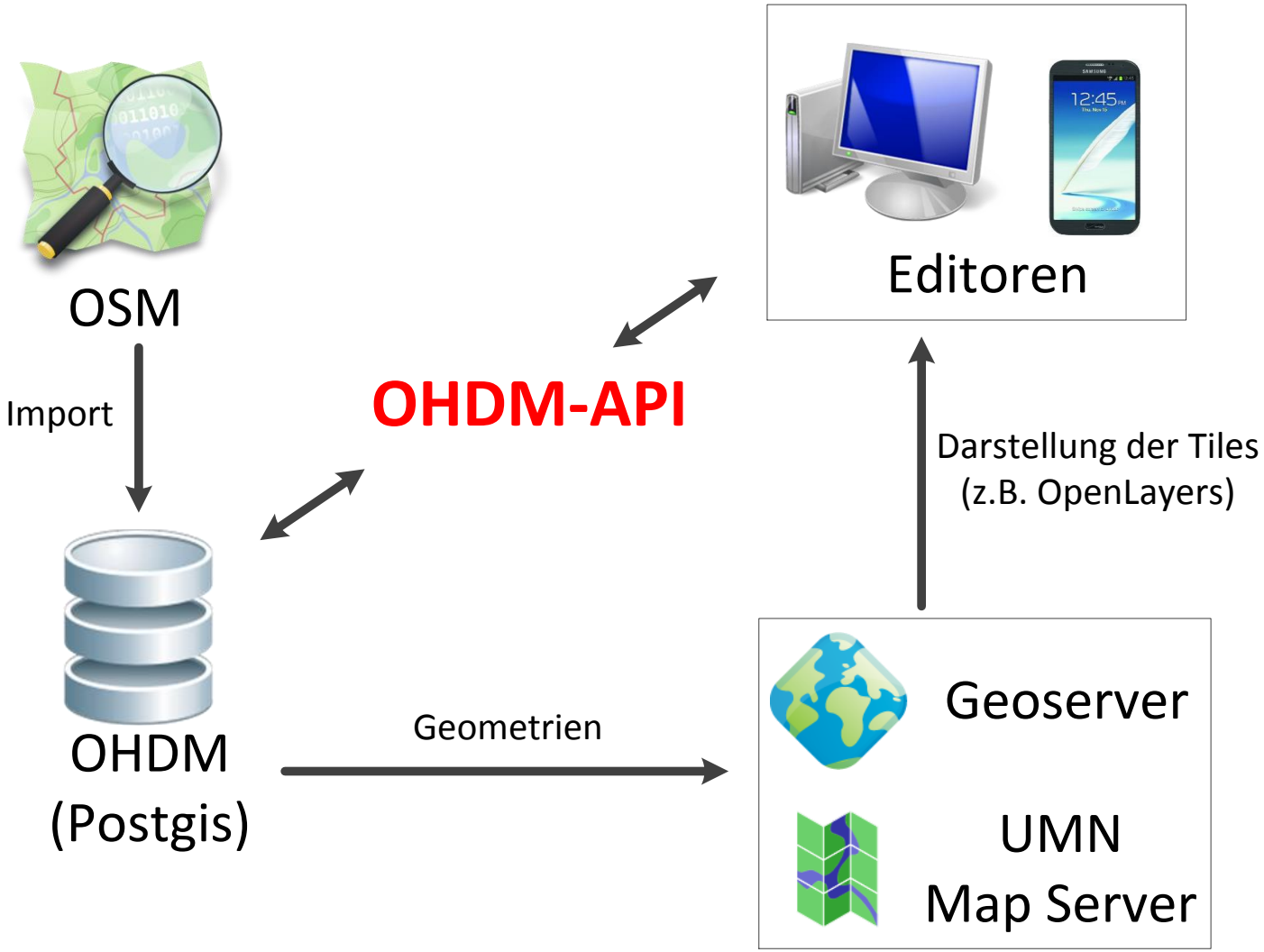
OHDM API

David Hadizadeh

04.12.2014

Einordnung der API in OHDM



Aufbau der API

- RESTful Webservice
- Implementiert in Java mit Jersey
- Zuständig für CRUD-Operationen der OHDM-DB
- Weitere Funktionen noch offen
  - Lieferung von Tiles via WMS-Server?

Was ist REST?

- Programmierkonzept für Webanwendungen
- Kommunikation auf unterster Ebene eines Protokolls (meist HTTP)
- Meist genutztes Entwurfsmuster für Webservices
- Verschiedene Datenformate
- Zustandslosigkeit

REST: Vorteile

- Geringer Datentransfer
- Klare Trennung von Änderungs- und Präsentations-Operationen
- Skalierbarkeit
- HTTP-Klassen und Funktionen in vielen Programmiersprachen vorhanden

REST: Nachteile

- Protokollnähe führt zu weniger intuitiven Abfragen
- Einschränkung auf Möglichkeiten des gewählten Protokolls
- Keine standardisierte Beschreibung (Dokumentation) der angebotenen Dienste möglich

Beispiel Anfragen

- Richtige Anfragen:
  - GET http://beispiel.de/artikel=8
  - PUT http://beispiel.de/ data: { „title“:„Test“ }
  - POST http://beispiel.de/artikel=8

    data: { „title“:„Test“ }

- Nicht dem Konzept entsprechend:
  - GET http://beispiel.de/artikel=next

Datei-Upload (Blob-Dates)

- 3 Möglichkeiten:
  1. Datei mit Base64 encoden → nur ein Aufruf, Größe steigt um ca. 1/3
  2. Datei mit „multipart/form-data“ übertragen und anschließend Meta-Informationen mit der ID verknüpfen
  3. Erst Meta-Daten senden (dann Datei wie in Punkt 2)

Bereits implementiert (PUT)

- URL: /externalSource/
- Beispiel:

```
{
    "title":"Source-Titel",
    "text":"Source-Text"
}
```

Bereits implementiert (POST)

- URL: /geometricObject/<geographicObjectId>
- Beispiel:

```
{
    "valid":{
        "since":"2013-12-01",
        "until":"2014-05-14"
    },
    "multipoint":"MULTIPOINT(1 2,1 2)"
}
```

- Auch möglich mit: Multilinestring, Multipolygon, Geometrycollection

Bereits implementiert (PUT)

- URL: /geographicObject/
- Beispiel:


```
{
   "originalId":"12345",
   "attributes":{
      "key1":"value1",
      "key2":"value2",
      "key3":"value3"
   },
   "externalSourceId":400,
   "geoBlobDates":[
      {
         "valid":{
            "since":"2013-12-01",
            "until":"2014-05-14"
         }
      }
   ],
   "tagDates":[
      {
         "geoObjects":{
            "key1":"value1",
            "key2":"value2",
            "key3":"value3"
         },
         "valid":{
            "since":"2013-12-01",
            "until":"2014-05-14"
         }
      }
   ],
   "geometricObjects":[
      {
         "valid":{
            "since":"2013-12-01",
            "until":"2014-05-14"
         },
         "multipoint":"MULTIPOINT(1 2,1 2)"
      }
   ]
}
```

Client-Beispiel (Java)


```
1. URL url = new URL(address);
2. HttpURLConnection httpCon = (HttpURLConnection)
   url.openConnection();
3. httpCon.setDoOutput(true);
4. httpCon.setRequestProperty("Content-Type",
   "application/json; charset=utf-8");
5. httpCon.setRequestMethod("PUT");
6. OutputStreamWriter out = new
   OutputStreamWriter(httpCon.getOutputStream());
7. out.write(jsonString);
8. out.close();
9. httpCon.getInputStream();
```

Geplant (GET)

- /nearObjectNames/ → ID + Name + Typ
  - Parameter: beliebige Geometrie
- /geographicObject/ → ein Objekt mit allen Daten
  - Parameter: ID
- /geographicObjects/ → mehrere Daten
  - Parameter: Typ / Geometrie

Noch zu planen

- DELETE, (implizietes UPDATE bei PUT)
- Weitere gewünschte GET-Funktionalitäten?
- Vereinfachung der Anfragen an den Web Map Service
- Überprüfung der Spatial Reference Identifier (SRID) oder Unterstützung weiterer?
- Ausführliche Dokumentation (Ende des Semesters)
- Überprüfung von Duplikaten (langsam)?

Zusammenspiel mit Open Layers

- Verwendung des OHDM-Tiles-Server
- Bei Bereichsauswahl oder Erzeugen neuer Elemente:
  - Aufruf der API
    - Lieferung von Geometrien aus der Umgebung
    - Persistieren von neuen Geometrien

Erreichbarkeit

- Nur aus dem HTW-Netzwerk (VPN-Verbindung)

- Aktueller Stand zu finden auf:

  http://hadizadeh.de/ohdm/