Geospatial Web Development 2022 | Übung 3
Nikolaos Kolaxidis | 21.12.2022

Diese Übung soll den Workflow einer interaktiven Kartendarstellung von Datenbeschaffung über Prozessierung, Bereitstellung und Implementierung in einer Webseite zeigen.

Der ursprüngliche Datensatz, der frei gewählt wurde, ist recht komplex und weist neben 12 Tieraren auch für jedes Tier drei Schlachtungsarten sowie neben der Anzahl auch die Menge in Tonnen auf. Um den Datensatz zu vereinfachen und für eine Ein-Layer-Kartenanwendung nutzbarer zu machen, wurde nur eine Tierart (Schweine) ausgewählt und für diese die Schlachtungsarten sowie die Anzahl der getöteten Tiere zusammengefasst. Um die Daten dann auf die Populationsdaten der Bundesländer sowie die geographischen Flächen der Bundesländer zu beziehen, wurde jeweils ein Join mit einem gesonderten GeoJSON-File der Bundesländer durchgeführt. Dieser Datensatz wurde dann noch einmal überarbeitet und schließlich mit einem Style versehen, der die Verhältnisse, wie auf der Startseite beschrieben, verdeutlichen soll. Style (als SLD-Datei) und Datensatz (als Tabelle in einer Datenbank) wurden dann auf einen Geoserver hochgeladen bzw verbunden, miteinander verknüpft und als WMS zur Verfügung gestellt.

Auf Grundlage vorheriger Webseiten gestaltete sich die Implementierung in eine Webseite als recht einfach, da OpenLayers im selben Schema wie OSM-Layer oder andere Image-/TileLayer einen eigenen WMS-Layer/eigene WMS-Quelle zur Verfügung stellt. Dieser ist wie die meisten anderen Layer folgendermaßen grundlegend aufgebaut:

```
[...]
new ol.layer.Tile({
    source: new ol.source.TileWMS({
        url: //Link zum Geoserver,
        params: //Parametereinstellungen
        }),
    })
[...]
```

Hier ist anzumerken, dass Tiling (TileWMS) möglich aber nicht notwendig ist und entsprechend der Daten genutzt werden kann. In diesem Beispielt wurde es aktiviert, da es im WWW immer besser ist geringere Datenmengen zu laden. Trotzdem sollte angemerkt werden, dass es für diesen Datensatz und diese Karte eigentlich nicht nötig ist. Die Alternative ist ein ImageWMS, welches ähnlich funktioniert, aber leicht andere Parameter akzeptiert.

Als Parameter werden für den WMS-Layer neben des Layers und dessen Style auch weitere Infos definiert wie die Version des WMS, das Formats in welchem gerendert werden soll sowie dass es sich um eine OpenLayers-Anwendung handelt:

```
params: {
    "FORMAT": 'image/png',
    "VERSION": "1.3.0",
    tiled: true,
```

```
    "STYLES": 'Schweineschlacht Style',
    "LAYERS": 'Schweineschlacht',
    "exceptions": 'application/openlayers',
    tilesOrigin: 5.866250351000076 + "," + 47.27012360400005
        }
    })
```

Wie andere Layer auch kann dieser Layer dann der Karte, die vorher per `const map = new ol.Map({target: 'map',view: mapView});` definiert wurde, hinzugefügt werden.

Das Anzeigen von Featureattributen bei Klick bedarf eines anderen Vorgehens, hier wird eine Funktion namens `getFeatureInfoUrl` benötigt. Diese soll ausgeführt werden, wenn in die Karte geklickt wird, daher wird sie in das Event `map.on('singleclick')` integriert und in einer Funktion implementiert, die beim Klick ausgeführt wird. Dabei ist es natürlich wichtig zu wissen, wo geklickt wird und welches Feature dann aufgerufen werden soll. Dies ist möglich durch Feststellen der Koordinate in der Karte (mit Projektion und Auflösung) und dadurch des Auswählens des Features. Durch eine anschließende If-Schleife wird der HTML-Text, der durch die Funktion `getFeatureInfoUrl` zurückgegeben wird, in das HTML-Dokument integriert.

```
map.on('singleclick', function(evt) {
    document.getElementById('feature_info').innerHTML = "Wird geladen, bitte
warten...";
    const feature = wmsLayer.getSource().getFeatureInfoUrl(
        evt['coordinate'],
        map.getView().getResolution(),
        map.getView().getProjection(),
        {'INFO_FORMAT': 'text/html'});
    if (feature) {
        fetch(feature)
            .then((response) => response.text())
            .then((html) => {
            document.getElementById('feature_info').innerHTML = html;
            });
        }
    });
```

Schließlich wird eine Legende eingebunden, die eine Bilddatei vom Geoserver zieht, welche durch den Style vordefiniert wird. Interessant hierbei ist das Zusammenspiel von WMS-Layer, dessen Quelle, dessen Verkünpfung mit dem Style auf dem Geoserver und schließlich der Legende davon. Die Legende wird im HTML-Dokument an Stelle `id="legend"` eingebunden und kann mit CSS positioniert und transformiert werden.

```
const insertLegend = function (resolution) {
    const graphicUrl = wmsLayer.getSource().getLegendUrl(resolution);
    const img = document.getElementById('legend');
    img.src = graphicUrl;
  };
const resolution = map.getView().getResolution();
insertLegend(resolution);
```