Inhaltsverzeichnis

Masterportal Logo

Einführung ins Masterportal

Herzlich Willkommen beim **Mastering the Masterportal** Workshop. 👋🏻

Dieser Workshop wurde für die Verwendung auf der OSGeo-Live 15.0 DVD entwickelt und soll Ihnen einen umfassenden Überblick über das Open Source Masterportal als Web-GIS-Lösung geben.

Allgemeines

Das Masterportal Projekt ermöglicht die Erstellung modularer und somit individueller Geoportale. Zum Aufbau einer Nutzer- und Entwicklungsgemeinschaft wurde die Implementierungspartnerschaft ins Leben gerufen, die mittlerweile aus über 35 PartnerInnen auf kommunaler, förderaler und Bundesebene besteht. Die wichtigsten Vernetzungs- und Entscheidungstreffen sind:

- Strategisches Kommittee (steuert und kontroliert die strategische Richtung des Masterportals)
- Technisches Kommittee (unterstützt das Strategische Kommittee in technischen Fragen)
- Produktpflege (Technische Weiterentwicklung, Release Management, etc.)
- Maintainergroup (unterstützt die Produktpflege bei der technischen Weiterentwicklung, Bearbeitung von PullRequests usw.)
- Produktmanagement (koordiniert organisatorische Angelegenheiten, Öffentlichkeitsarbeit, Veranstaltungsplanung usw.)

Neben den regelmäßig stattfindenden Gremiensitzungen werden verschiedene Workshops organisiert, zum Beispiel zur Ersteinrichtung der Software oder zu speziellen technischen Themen wie der Integration von sicheren Geodatendiensten.

Neben den Partnern aus der öffentlichen Verwaltung gibt es verschiedene Unternehmen, die Support und Wartung anbieten und zur Weiterentwicklung von Masterportal beitragen.

Zentrale Links:

- Website
- Twitter
- Code

Autoren

- Hannes Blitza (blitza@terrestris.de)

Zuarbeit

- LGV HH, u.a. Dirk Rohrmoser (dirk.rohrmoser@gv.hamburg.de)
- Marc Jansen (COG Example) (jansen@terrestris.de)

FOSSGIS 2023Last modified: 2023-03-02 15:51:53

Vorarbeitung und generelle Informationen

Bevor wir mit dem Workshop starten können, führen Sie bitte die folgenden Schritte aus:

- Rechner mit OSGeoLive-Medium hochfahren

Es wird angenommen, dass die OSGeoLive bereits installiert ist. Falls nicht, kann der Live-Modus gestartet werden:

- *Lubuntu ohne Installation ausprobieren* auswählen
- Benutzer: user; Passwort: user (wird vermutlich nicht benötigt)

> **Warning**
>
> ⚠ Sollte die Tastaturbelegung noch auf **US** gestellt sein (siehe unten rechts in der Taskleiste), öffnen Sie ein Terminal (`Strg+T`) und führen Sie folgenen Befehl aus: `setxkbmap -layout de`

Die Startansicht der OSGeo Live 15.0 auf Ihrem Rechner.

FOSSGIS 2023Last modified: 2023-03-02 15:51:53

Basiswissen

Dieser Teil dient eher als Nachschlagewerk, es kann direkt losgelegt werden mit dem Entwicklungssetup

- npm

Linksammlung

- https://www.masterportal.org/
- https://bitbucket.org/geowerkstatt-hamburg/masterportal
- https://geoportal-hamburg.de/geo-online/
- https://geoportal.de
- https://maps.stuttgart.de/
- https://geoportal.freiburg.de/freigis/
- https://geoportal.muenchen.de/portal/master/
- https://geoportal.brandenburg.de/de/cms/portal/start

Weitere Geoportale: https://www.masterportal.org/referenzen.html

FOSSGIS 2023Last modified: 2023-03-02 15:51:53

npm

npm ist der Paketmanager für Node.js (eine JavaScript-Laufzeitumgebung) und die weltweit größte Software-Registry (mehr als 600k Pakete) mit ca. 3 Milliarden Downloads pro Woche.

npm Logo

npm kann genutzt werden, um ...

- Pakete an Anwendungen anpassen oder diese so einbinden, wie sie sind.
- Eigenständige Tools herunterladen.
- Pakete ohne Herunterladen mit npx ausführen.
- Code mit jedem npm-User teilen, überall.
- Code für bestimmte Entwickler beschränken.
- Virtuelle Teams (orgs) bilden.
- Mehrere Versionen von Code und Code-Abhängigkeiten verwalten.
- Anwendungen einfach aktualisieren, wenn der zugrunde liegende Code aktualisiert wird.
- Andere Entwickler finden, die an ähnlichen Problemen arbeiten.

package.json

Der Befehl `npm init` in Ihrem Projektordner öffnet einen interaktiven Dialog zur Erstellung eines npm-Projekts. Das Ergebnis ist die `package.json` mit allen wichtigen Einstellungen, Skripten und Abhängigkeiten Ihres Projekts.

```
{
  "name": "name_of_your_package",
  "version": "1.0.0",
  "description": "This is just a test",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "http://github.com/yourname/name_of_your_package.git"
  },
  "author": "your_name",
  "license": "ISC"
}
```

Weitere Infos hier: npm docs package.json.

Installieren von packages mittels npm

Der einfachste Weg, neue Pakete zu intallieren, ist die Nutzung der CLI:

```
npm install packagename
```

Das installierte Paket ist anschließend im Subfolder `node_modules` zu finden.

Node version manager NVM

- bash script um mehrere node Versionen zu verwalten
- Siehe hier

```
wget -qO- https://raw.githubusercontent.com/creationix/nvm/v0.34.0/install.sh | bash
nvm i v8
```

FOSSGIS 2023Last modified: 2023-03-02 15:51:53

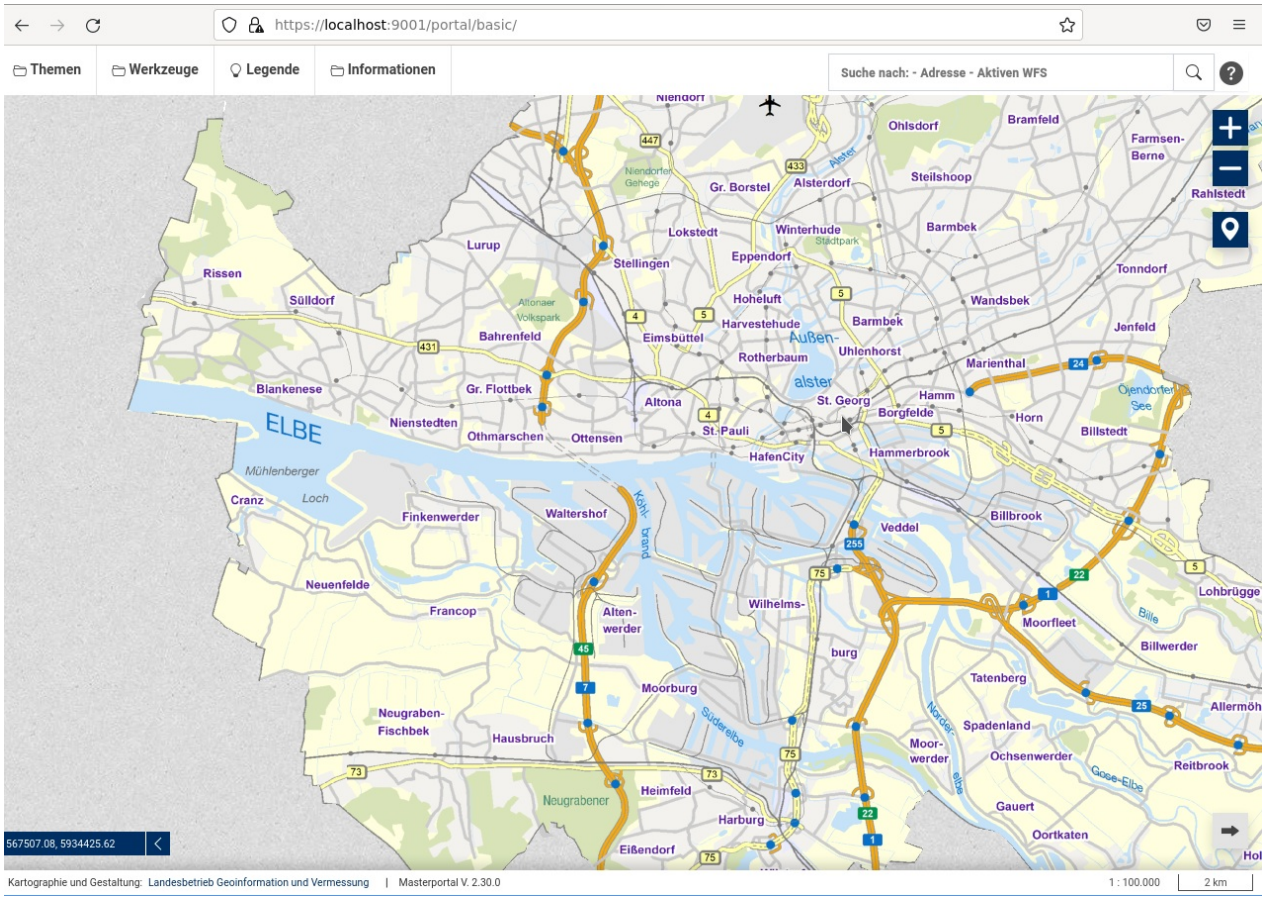
Repository und Entwicklungssetup

1. Öffnen Sie das Terminal und führen Sie den Befehl `pwd` aus.
2. Sie sollten sich im Pfad `/home/user` befinden.
3. Führen Sie den Befehl `git clone https://hblitza@bitbucket.org/geowerkstatt-hamburg/masterportal.git` aus, um das Masterportal Repository auf Ihre Festplatte zu kopieren. Navigieren Sie anschließend in das neue Verzeichnis per Befehl: `cd masterportal`.

Wie in vielen modernen Javascript Projekte, wird auch für das Masterportal ein Node.js Framework zur Entwicklung genutzt. Mithilfe des Paketsmanager npm werden sämtliche Bibliotheken und Abhängigkeiten gemanaged und installiert, wie beispielsweise webpack, der als *module bundler* fungiert.

Eine ausführliche Beschreibung dieser Entwicklungstools- und Frameworks würde den Rahmen dieses Workshops sprengen, die benötigsten Infos werden im Rahmen dieses Workshops gegeben. Eine kurzen Überblick über npm ist hier zu finden.

1. Führen Sie `node -v`, um die installierte Version von `node` auszugeben. Falls `node` nicht installiert ist, oder die Version < `16.13.2` oder > `16.18.1` ist, folgende Schritte ausführen:
   - `wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash`
   - `source ~/.bashrc` Refresh der `.bashrc`. Notwendig, um neues command `nvm` auszuführen
   - `nvm install v16.18.1`
2. Es soll auf der Version `2.31.0` gearbeitet werden, hierzu sind folgend Befehle auszuführen:
   - `git fetch origin`
   - `git checkout v2.31.0`
3. Installieren Sie alle benötigten Abhängigkeiten des Masterportals-Projekts: `npm i`.
4. Starten Sie anschließend den Entwicklungsserver: `npm run start`.
5. Nun wird der Masterportal-Quellcode kompiliert und `webpack` erstellt den *dev build*, der anschließend - sobald die Nachricht `Compiled successfully` im Terminal erscheint, im Browser unter der Adresse `localhost:9001/portal/basic` aufgerufen werden kann.
6. Möglicherweise tauchen viele Logs mit der Nachricht `ENOSPC: System limit for number of file watchers reached` auf. In diesem Fall `Strg+C` drücken um den Dev-Server zu stoppen. Dann `echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf && sudo sysctl -p` ausführen und anschließend den Dev-Server wieer starten mit `npm run start`.



Startansicht des Portals basic.

> **i**
>
> Weiterführende Infos zum Dev-Setup unter:
> https://bitbucket.org/geowerkstatt-hamburg/masterportal/src/latest/doc/setup.md

FOSSGIS 2023Last modified: 2023-03-02 15:51:53

Architektur

Paradigmen

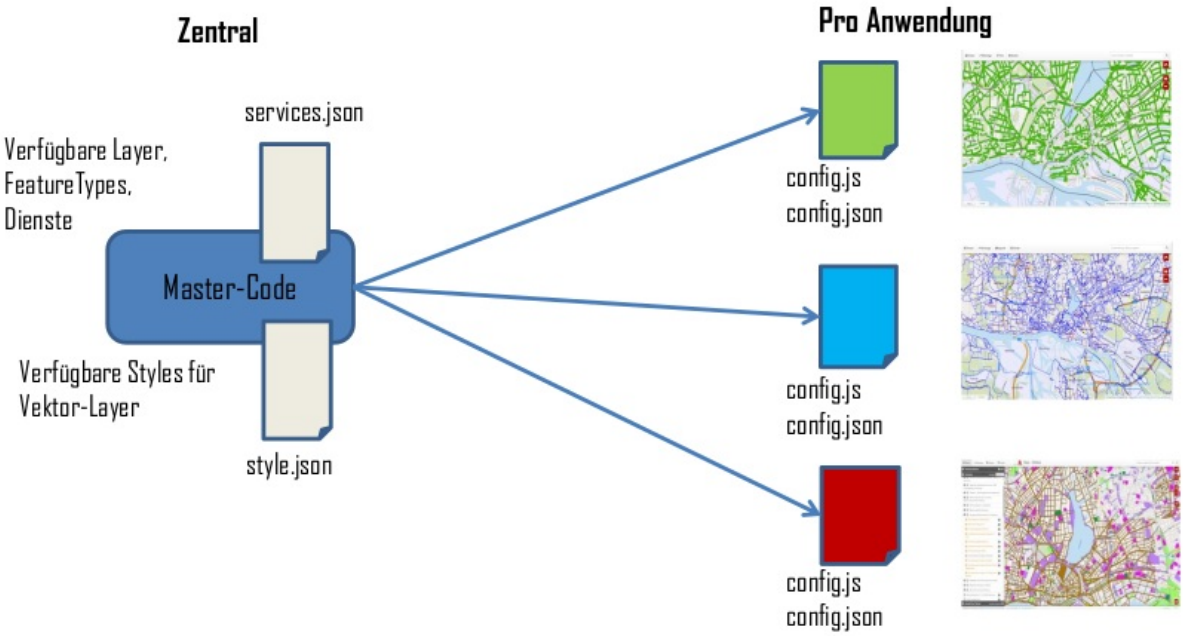
- OpenSource (MIT)
- Standardbasiert (OGC Standards, Rest APIs)
- Modular und konfigurabel
- Responsiv
- Rein clientseitig
- Gut dokumentiert für NutzerInnen und Devs: `coding conventions` , `linting` , etc.
- Nutzung weit verbreiteter und *well maintained* Bibliotheken (z.B. OpenLayers, Vue.js, Vuex)

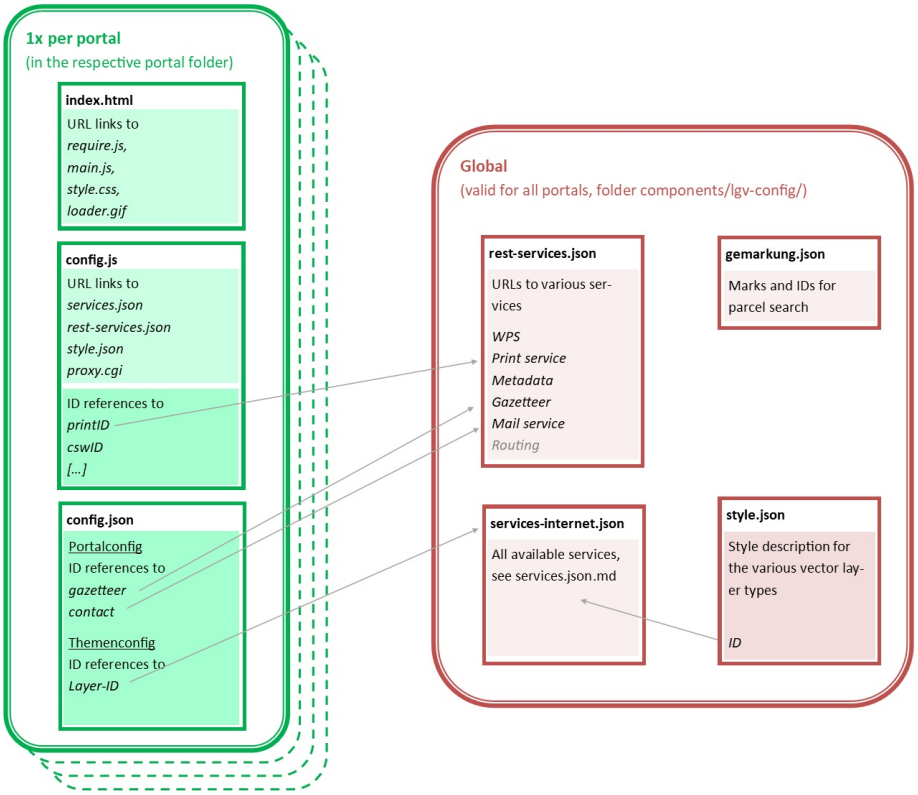
--> Schlanker Core. Zentrale Funktionen in der `MasterportalAPI` . Erweiterung durch Addons.

FOSSGIS 2023Last modified: 2023-03-02 15:51:53

Konfiguration

Der Applikationskontext teilt sich in mehrere Dateien auf, die nach Belieben angepasst werden können. Teilweise können diese automatisch erstellt werden, darauf wird im nächsten Kapitel eingegangen.





FOSSGIS 2023Last modified: 2023-03-02 15:51:53

Globale Konfiguration

services.json

Dies ist die zentrale Konfiguration für sämtliche Layer (WMS, WFS, WMTS, SensorThings-API, GeoJSON and weitere) die in den Portal dargestellt werden sollen. Sie wird in den jeweiligen Portalkonfigurationen (`config.js`) referenziert. Es kann auch auf einen API-Endpunkt verwiesen werden, der die `services.json` Datei generiert - etwa über einen Dienstemanager.

Jeder Layertyp benötigt unterschiedliche Parameter, wobei einige stets obligatorisch und andere optional sind.

WMS

Ein Beispiel für einen **WMS-Layer**:

```
{
    "id" : "8",
    "name" : "Aerial View DOP 10",
    "url" : "https://geodienste.hamburg.de/HH_WMS_DOP10",
    "typ" : "WMS",
    "layers" : "1",
    "format" : "image/jpeg",
    "version" : "1.3.0",
    "singleTile" : false,
    "transparent" : true,
    "tilesize" : "512",
    "gutter" : "0",
    "minScale" : "0",
    "maxScale" : "1000000",
    "gfiAttributes" : "ignore",
    "layerAttribution" : "nicht vorhanden",
    "legend" : false,
    "layerSequence": 1,
    "datasets" : [
        {
        "md_id" : "25DB0242-D6A3-48E2-BAE4-359FB28491BA",
        "rs_id" : "HMDK/25DB0242-D6A3-48E2-BAE4-359FB28491BA",
        "md_name" : "Digitale Orthophotos 10cm - FHHNET",
        "bbox" : "461468.97,5916367.23,587010.91,5980347.76",
        "kategorie_opendata" : [
            "Sonstiges"
        ],
        "kategorie_inspire" : [
            "nicht INSPIRE-identifiziert"
        ],
        "kategorie_organisation" : "Landesbetrieb Geoinformation und Vermessung"
        }
    ]
}
```

Hier sind zahlreiche Parameter angegeben, die den Layer optimal definieren. Beispielsweise bewirkt der `gfiAttributes: false` Parametereinstellung, dass keine Sachdatenabfrage (GetFeatureInfo) ausgeführt wird.
`tilesize" : "512"` sorgt dafür, dass die Kacheln des WMS in einer Größe von 512x512 Pixeln abgefragt werden (default 256px).

> **Hinweis**
>
> Die Parameter bedingen sich teilweise gegenseitig bzw. sind voneinander abhängig. Bei ungewünschten Verhalten der Layer sollte die vollständige Dokumentation ausführlich studiert werden.
> Ein Beispiel: `singleTile: true` hat zur Folge, dass der Ausschnitt als einzelne Kachel vom WMS abgefragt wird. `tileSize` hat dann natürlich keinen Effekt mehr.

WMTS

Ein Beispiel, das nur wenige Konfigurationsparameter bedarf ist ein `WMTS`, wenn der Parameter `optionsFromCapabilities` gesetzt ist:

```
{
    "id": "2020",
    "name": "EOC Basemap",
    "capabilitiesUrl": "https://tiles.geoservice.dlr.de/service/wmts?SERVICE=WMTS&REQUEST=GetCapabilities",
    "typ": "WMTS",
    "layers": "eoc:basemap",
    "optionsFromCapabilities": true
}
```

Hier muss bedacht werden, dass die Kacheln in der Projektion des ersten TileGrid angefragt werden, dass im Capabilities auftaucht (zumeist EPSG:3857). OpenLayers übernimmt im Client die Reprojektion der Kacheln in die aktuelle Kartenprojektion.

Link zur vollständigen Dokumentation

rest-services.json

Hier werden alle Services definiert, die nicht direkt für die visuelle Darstellung von Daten benötigt werden:

- Print services (MapFish)
- Metadata sources (CSW HMDK)
- BKG geocoding service
- Gazetteer URL
- WPS
- Email Services

```
[
  {
    "id": "1",
    "name": "CSW HMDK Summary",
    "url": "http://metaver.de/csw?service=CSW&version=2.0.2&request=GetRecordById&typeNames=csw:Record&elementsetname
=summary",
    "typ": "CSW"
  },
  {
    "id": "mapfish-terrestris",
    "name": "Testserver Print",
    "url": "https://10.133.7.xx/print/",
    "typ": "Print"
  },
  {
    "id" : "11",
    "name" : "Komoot Photon Suche",
    "url" : "https://photon.komoot.io/api/?",
    "typ" : "WFS"
  },
  {
    "id" : "80002",
    "name" : "Email Service by PHP",
    "url" : "https://geoportal-hamburg.de/smtp/sendmail.php",
    "typ" : "EmailService"
  }
]
```

style.json

Vektordaten wie WFS und GeoJSON werden clientseitig gestyled (gegenüber WMS und WMTS, die serverseitig gestyled werden). Das Masterportal liest hierzu die `style.json` aus, in der x-beliebige Style im OpenLayers eigenen Stil-Format definiert werden. Bei der Konfiguration von Vektorlayern kann einem Layer ein bestimmter Stil zugewiesen werden, zudem können diese Stile für dynamisch hinzugefügte Layer (während der Laufzeit) verwendet werden oder Standard-Layer wie beispielsweise `MapMarker` .

Jeder Style beginnt mit einer `styleID` und darauffolgend mit der Definition von Stilregeln.

```
{
  "styleId": "blue-point",
  "rules": [
    {
      "style":
        {
        "circleRadius" : 6,
        "circleStrokeColor": [51, 102, 255, 1],
        "circleStrokeWidth": 2,
        "circleFillColor": [51, 102, 255, 1]
        }
    }
  ]
},
```

Der Stilregel können `conditions` hinzugefügt werden, die letztendlich attributives Styling ermöglichen:

```
{
  "styleId": "blue-point",
  "rules": [
    {
        "conditions": {
              "properties": [
                  {
                      "attrName": "housenumber",
                      "value": [0, 100]
                  }
              ]
      },
      "style":
        {
        "circleRadius" : 6,
        "circleStrokeColor": [51, 102, 255, 1],
        "circleStrokeWidth": 2,
        "circleFillColor": [51, 102, 255, 1]
        }
    }
  ]
},
```

Link zur vollständigen Beschreibung der Conditions

Vollständige Konfiguration

FOSSGIS 2023Last modified: 2023-03-02 15:51:53

Portalkonfiguration

config.js

Hier werden sämtliche Konfigurationen vorgenommen, die nicht direkt auf UI-Elementen oder Layern bezogen sind:

- Pfade zu Backends und weiteren Konfigurationsdateien
- Projektionsdefinitionen im Portal
- Liste an Addons
- Proxy Einstellungen
- Footer
- mousehover

Link zur vollständigen Dokumentation

config.json

Die `config.json` enthält die gesamte Konfiguration der Portal-Oberfläche. In ihr wird definiert, welche Elemente sich wo in der Menüleiste befinden, wo das initiale Kartenzentrum liegen soll und welche Layer geladen werden sollen. Des weiteren wird die Liste der Tools und Addons definiert und die dazugehörige Start-Konfiguration festgelegt.

Portalconfig

- Titel & Logo (portalTitle)
- Art der Themenauswahl (treeType)
- Starteinstellungen der Kartenansicht (mapView)
- Schaltflächen auf der Kartenansicht sowie mögliche Interaktionen (controls)
- Menüeinträge sowie Vorhandenheit jeweiliger Tools und deren Reihenfolge (menu)
- Typ und Eigenschaften des genutzten Suchdienstes (searchBar)
- Löschbarkeit von Themen (layersRemovable)

Beispiel aus der `basic` Portalkonfiguration:

```
"Portalconfig": {
    "treeType": "light",
    "searchBar": {
        "komoot": {
            "minChars": 3,
            "serviceId": "11",
            "limit": 20,
            "lang": "de",
            "lat": 53.6,
            "lon": 10.0,
            "bbox": "9.6,53.3,10.4,53.8"
        },
        "visibleVector": {
            "layerTypes": [
                "WFS"
            ]
        },
        "tree": {},
        "startZoomLevel": 9,
        "placeholder": "Suche nach: - Adresse - Aktiven WFS"
    },
    "mapView": {
        "backgroundImage": "./resources/img/backgroundCanvas.jpeg",
        "startCenter": [
            561210,
```

```
            5932600
        ],
        "extent": [
            510000.0,
            5850000.0,
            625000.4,
            6000000.0
        ],
        "startZoomLevel": 1
    },
    "menu": {
        "tree": {
            "name": "Themen"
        },
        "tools": {
            "name": "Werkzeuge",
            "children": {
                "gfi": {
                    "name": "Informationen abfragen",
                    "active": true
                },
                "coordToolkit": {
                    "name": "Koordinaten"
                },
                "measure": {
                    "name": "Strecke / Fläche messen"
                },
                "draw": {
                    "name": "Zeichnen / Schreiben"
                },
                "fileImport": {
                    "name": "Datei Import"
                },
                "saveSelection": {
                    "name": "Auswahl speichern"
                }
            }
        },
        "legend": {
            "name": "Legende"
        },
        "info": {
            "name": "Informationen",
            "children": {
                "staticlinks": [
                    {
                        "name": "Masterportal",
                        "url": "https://masterportal.org"
                    }
                ]
            }
        }
    },
    "controls": {
        "zoom": true,
        "orientation": {
            "zoomMode": "once"
        },
        "attributions": true,
        "mousePosition": true
    }
}
```

Themenconfig

Die Themenconfig definiert, welche Inhalte an welcher Stelle im Themenbaum vorkommen.
Die Struktur ist abhängig von der Art des Themenbaums (ob flache Hierachie oder Aufsplittung on Fach- und Hintergrundkarten).
Zudem können mehrdimensionale Daten (Time und 3D) separat definiert werden.

Ein Minimal-Beispiel (`treetype: custom`):

```
{
  "Themenconfig": {
    "Hintergrundkarten": {
      "Layer": [
        {
          "id": "452"
        },
        {
          "id": "432"
        }
      ]
    },
    "Fachdaten": {
      "Ordner": [
        {
          "Titel": "Fahrrad",
          "Layer": [
            {
              "id": "10882"
            }
          ]
        }
      ]
    }
  }
}
```

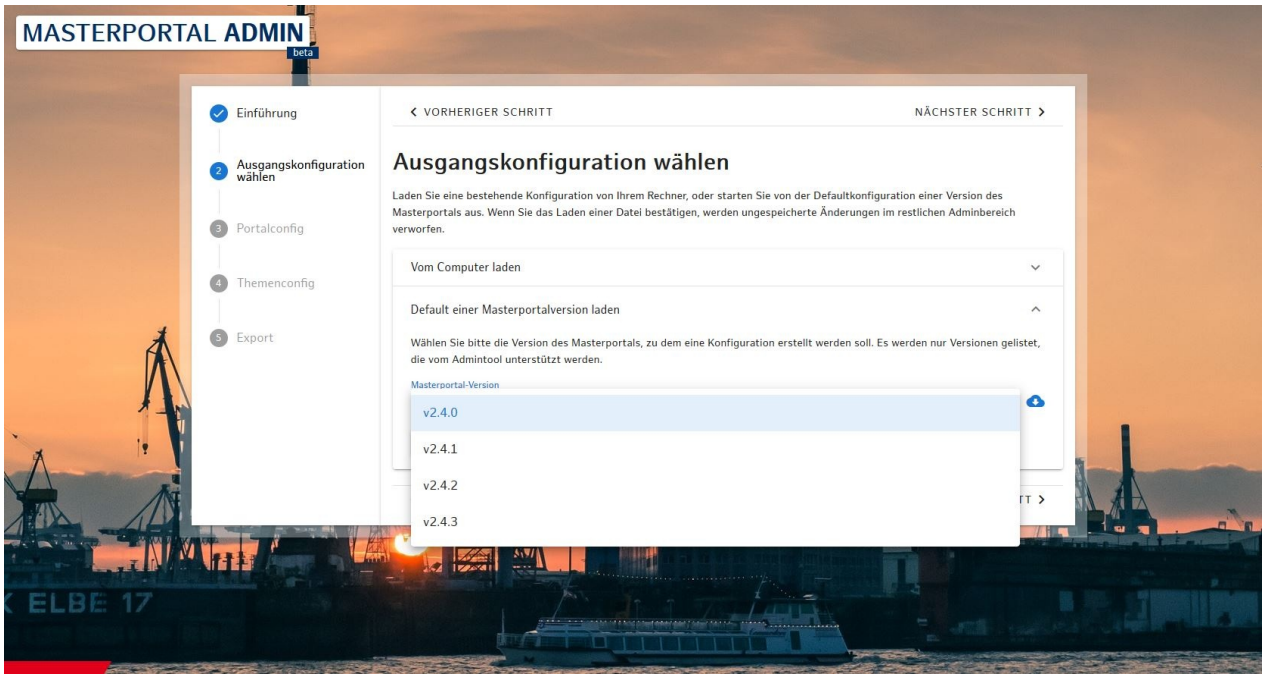
Link zur vollständigen Dokumentation

FOSSGIS 2023Last modified: 2023-03-02 15:51:53

Masterportal Admin

Mit dem „Masterportal-Admin“ steht interessierten NuterInnen eine webbasierte Anwendung mit grafischer Benutzeroberfläche zur Verfügung, die es ermöglicht, Konfigurationen für individuelle Geoportale dialoggeführt zu erstellen.

- Kurzbeschreibung
- Repository



MP Admin

FOSSGIS 2023Last modified: 2023-03-02 15:51:53

Dokumentation

Die Dokumentation für die Administration und Konfiguration des Masterportals ist in Markdown geschrieben (Ordner `doc`).

Markdown nach HTML (docs "bauen")

1. Im Masterportal repo `npm run buildMdDocs` aufrufen.
2. Die Markdowns werden dann in den Ordner `docHtml` geschrieben und können bequem in einem Webbrowser betrachtet werden.
3. `cd docHtml` und `python3 -m http.server` ausführen.
4. Nun werden die Docs unter http://localhost:8000 ausgeliefert. (z.B. http://localhost:8000/config.json.html)

Alternativen

- https://www.masterportal.org/dokumentation.html
- https://bitbucket.org/geowerkstatt-hamburg/masterportal/src/latest/
- Oder mit einem Editor (z.B. Visual Studio Code) die Markdowns rendern (Strg+Shift+v)
  FOSSGIS 2023Last modified: 2023-03-02 15:51:53

Übungsaufgaben

> ! Startpunkt ist der `basic` Branch in der Entwicklungsumgebung:
> Im Masterportal Folder (cloned Bitbucket-Repository) muss vorher ein `npm i` ausgeführt wurden sein. Dann wird das Entwicklungssetup gestartet per `npm run start`. Das Starten dauert einige Zeit, es werden einige Warnings angezeigt, die aber getrost ignoriert werden können.
> Sobald der Log `Compiled successfully` erscheint, kann losgelegt werden.
>
> Verwenden Sie `Featherpad` als Text-Editor für die folgenden Aufgaben.

Aufg. 1 Konfigurieren Sie das Messen-Tool so, dass für das Messen der Fläche stets auch Dezimeter angezeigt werden.

Hint
Schauen Sie in der `config.json.md` nach `Portalconfig.menu.tool.measure`!

Lösung
Ändern Sie die Konfiguration für das Messen-Werkzeug in der `config.json` wie folgt:

```
"measure": {
    "name": "translate#common:menu.tools.measure",
    "measurementAccuracy": "decimeter"
}
```

Info: Wenn der Name mit einem `translate#` beginnt, folgt da hinter der Key in den Übersetzungsdateien `i18n`. In mehrsprachigen Modulen sollte dies immer der Fall sein, da ansonsten der Name des Tools im Portal nicht übersetzt wird.

Aufg. 2 Definieren Sie einen neuen WMS-Layer in der `services.json` und integrieren Sie diese der Anwendung.

- Basis-Url: `https://ows.terrestris.de/osm/service`
- Er soll initial sichtbar sein
- Kachelgröße 512x512 Pixel
- Kein `GetFeatureInfo`

Hint
Schauen Sie sich eine existierende WMS-Konfiguration an. Zum Beispiel die ID 452 (Digitale Orthophotos (belaubt) Hamburg). Kopieren Sie diesen Block und Ändern Sie die ensprechenden Parameter. Ferner kann die Dokumentation der services.json hinzugezogen werden.

Lösung
Eintrag in der `services.json`:

```
{
    "id": "1000",
    "name": "OSM WMS",
    "url": "https://ows.terrestris.de/osm/service",
    "typ": "WMS",
    "layers": "OSM-WMS",
    "format": "image/png",
    "version": "1.3.0",
    "singleTile": false,
    "transparent": false,
    "tilesize": "512",
    "minScale": "0",
    "maxScale": "1000000",
    "visibility": true,
    "gfiAttributes": "ignore"
},
```

Eintrag in der `config.json`

```
"Themenconfig": {
        "Hintergrundkarten": {
            "Layer": [
        {
          "id": "1000"
        },
```

Aufg. 3 Definieren Sie einen WMTS Layer mit der Basis-URL `https://tiles.geoservice.dlr.de/service/wmts?SERVICE=WMTS&REQUEST=GetCapabilities` . Integrieren SIe diesen in das Portal.

Hint
Werfen Sie einen erneuten Blick in die services.json Dokumentation. Es werden für den WMTS Layertyp zwei Beispiele genannt.

Lösung
Definieren Sie den Layer wie folgt:

```
    {
        "id": "2020",
        "name": "EOC Basemap",
        "capabilitiesUrl": "https://tiles.geoservice.dlr.de/service/wmts?SERVICE=WMTS&REQUEST=GetCapabilities",
        "typ": "WMTS",
        "layers": "eoc:basemap",
        "optionsFromCapabilities": true
    },
```

Fügen Sie anschließend die ID in die Themenkonfiguration ein (siehe Aufg. 2).

Aufg. 4 Ändern Sie das Logo der Anwendung in folgendes Bild um: `https://www.fossgis.de/logos/FOSSGIS@2x.png`

Hint
Schauen Sie in der `config.json.md` nach `Portalconfig.portalTitle` !

Lösung
Definieren Sie in der `config.json` im Block `Portalconfig` den folgenden Block:

```
    "portalTitle": {
        "title": "FOSSGIS Testportal",
        "logo": "https://www.fossgis.de/logos/FOSSGIS@2x.png",
        "link": "https://www.fossgis.de",
        "toolTip": "FOSSGIS Logo"
    }
```

Aufg. 5 Ändern Sie den Titel der Webseite (taucht oben in der Browser Leiste auf).

Hint
Werfen Sie einen Blick in die `index.html` im Portalordner `basic` . Es gibt einen

Lösung
Passen Sie die Zeile 10 in der `index.html` an:

```
<title>FOSSGIS Portal</title> <!-- enter your own Portal Title for the website at this -->
```

Aufg. 6 Fügen Sie die Map-Control `Overview Map` dem Portal hinzu. Dies soll initial eingeblendet sein. Wählen Sie den Layer aus Aufgabe 1 für die Overview Map aus.

Hint
Schauen Sie in der `config.json.md` nach `Portalconfig.controls.overviewMap` !

Lösung
Ändern Sie die Konfiguration für die `overviewMap` Control folgendermaßen:

```
    "overviewMap": {
        "layerId": "1000",
        "isInitOpen": true
    }
```

FOSSGIS 2023Last modified: 2023-03-02 15:51:53

Erweiterte Konfiguration

1. UseCase

Ich möchte die Hauptfarben des Portals anpassen! Die Buttons sollen nicht rot, sondern blau sein - da dies besser zu unserem StyleGuide passt. 📘

Überlegung:

Es gibt im Ordner `mastercode/2_30_0/css` eine `masterportal.css`. Kann ich nicht einfach hier den Farbwert überschreiben? ❓

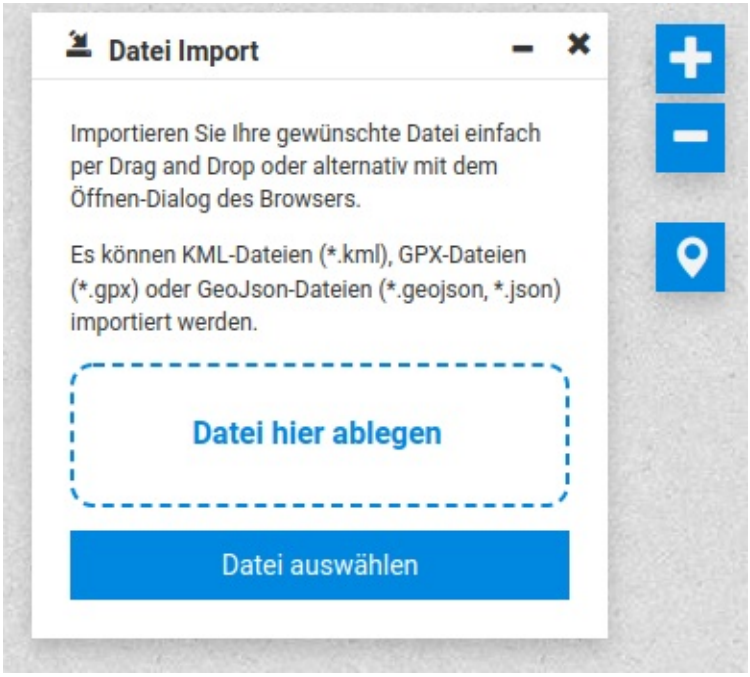
⚠️ Woher weiß man, welche CSS-Properties alle angepasst werden? Die css Datei > 2.500 Zeilen und > 300.000 Zeichen!

😕 Keine gute Idee! Bei einem Update kann sich die Datei ändern - und man muss alle Anpassungen überprüfen / erneut durchführen..

Daher:

Wir passen eine zentrale Farbvariable an im Development Modus und bauen hinterher unser Portal neu! 🆙

1. Öffnen Sie im Code-Editor die Datei `variables.scss`. Hier befinden sich sämtliche Farbwerte, aus denen die einzelnen Oberflächenelemente referenziert werden (Buttons, Hintergründe von Schaltflächen etc.).
2. Suchen sie nach dem Key `$primary` (Tipp: Zeile 94 😁).
3. Passen Sie den Wert an. Zum Beispiel auf Hellblau: `#0087e0`.
4. Webpack erkennt, dass Sie eine Änderung vorgenommen haben und kompiliert den Code erneut. Das Ergebnis ist sofort sichtbar unter `localhost:9001/portal/basic`.

   😕 Jetzt sind die Map-Controls blau, aber was ist mit den Tool-Buttons? Ich meine explizit das Tool Datei-Import.
5. Passen Sie die folgenden Variablen an:
   ```
   $secondary_focus: #0087e0;
   $accent: #0087e0;
   $accent_hover: darken($accent, 5%);
   ```
6. Betrachten Sie das Ergebnis: 

Nun ist der Footer noch blassgrau. Ich möchte diesen gerne ebenfall blau haben.

> ⚠ **Warning** Ändern Sie nicht zu viele Farbwerte. Die EntwicklerInnen des Masterportals und des Oberflächen-Frameworks (Bootstrap) haben sich bei der Wahl der Farben viele Gedanken gemacht. Es werden auch extra barrierearme color schemes verwendet, dies sollte bedacht werden. Es sollte auch immer betrachtet werden, welche Farbwerte voneinander abhängen! Goldene Regel: Nach jeder Änderung, ausführlich das Portal testen!

2. UseCase

Der MapMarker beim GFI ist schön und hat einen hohen Wiedererkennungswert. Trotzdem brauche ich für ein Spezial-Portal einen Custom-Marker Style. 🙏 Was kann ich tun?

1. Schauen Sie in der `config.js.md` nach dem Stichwort `mapMarker`. Sie finden dort folgende Erläuterungen:

Name	Required	Type	Default	Description
pointStyleId	no	String	`"defaultMapMarkerPoint"`	StyleId to refer to a `style.json` point style. If not set, the `img/mapMarker.svg` is used.
polygonStyleId	no	String	`"defaultMapMarkerPolygon"`	StyleId to refer to a `style.json` polygon style.

Example:

```
{
  "mapMarker": {
    "pointStyleId": "customMapMarkerPoint",
    "polygonStyleId": "customMapMarkerPolygon"
  }
}
```

Aha! ☝

1. Definieren Sie in der `style.json` einen Punkt Stil mit 3 Farbigen Kreisen:

```
{
  "styleId": "custom-point",
  "rules": [
    {
      "style":
      {
        "circleRadius" : 10,
        "circleFillColor": [0, 132, 255, 1],
        "circleStrokeColor": [255, 102, 0, 1],
        "circleStrokeWidth": 5
      }
    }
  ]
}
```

2. Weisen Sie in der `config.js` dem Property `pointStyleId` den neuen Stil `custom-point` zu.
3. Hier muss das Dev-Setup ggf. neugestartet werden (Strg+C, dann erneut `npm run start`), um den Effekt zu sehen.

3. UseCase

Ich hab auf der FOSSGIS 2022 in einigen Vorträgen vom `Cloud-Optimized-GeoTIFF` gehört und finde es interessant.
Nun habe ich gesehen, dass OpenLayers bereits mit dem Format umgehen kann. Wie bekomme ich das in mein Masterportal? 👀

🔥 Über ein Addon..

FOSSGIS 2023Last modified: 2023-03-02 15:51:53

Addons

Mittels eines Addons lässt sich die Funktionalität des Masterportals beliebig erweitern, ohne dass der **Core** verändert werden muss. Es lassen sich eigenständige `Tools` und `GfiThemes` entwickeln, die zu Beginn der Laufzeit importiert werden und fortan wie eigenständige Module funktionieren.

Im Rahmen dieses Workshops soll ein kleines Addon entwickelt werden, das mit der `OpenLayers` Karte des Masterportals interagiert und dieser ein Cloud Optimized GeoTIFF hinzufügt.

Die folgenden Schritte basieren auf im wesentlichen auf den folgenden Beispielen:

- Masterportal VueAddon
- Cloud Optimized GeoTIFF (COG)

- Legen Sie im Verzeichnis `addons` die Datei `addonsConf.json` an. Diese soll den Namen des neuen Addons erhalten. Wir nennen es `COG-Importer`.

addonsConf.json

```
{
  "cogImporter": {
    "type": "tool",
    "vue": true
  }
}
```

- Erstellen Sie im selben Verzeichnis einen Ordner mit dem Namen des Addons.

- Erstellen Sie eine `index.js` in diesem Verzeichnis mit folgenden Inhalt:

index.js

```
import CogImporterComponent from "./components/CogImporter.vue";
import Store from "./store/index";
import deLocale from "./locales/de/additional.json";
import enLocale from "./locales/en/additional.json";

export default {
    component: CogImporterComponent,
    store: Store,
    locales: {
        de: deLocale,
        en: enLocale
    }
};
```

- Erstellen im selben Verzeichnis einen Unterordner `components` und legen Sie hier die Datei `COGImporter.vue` an.

COGImporter.vue

```
<script>
import ToolTemplate from "../../../src/modules/tools/Tool.vue";
import {mapGetters, mapMutations} from "vuex";
import getters from "../store/getters";
import mutations from "../store/mutations";
import GeoTIFF from 'ol/source/GeoTIFF';
import TileLayer from 'ol/layer/WebGLTile';

export default {
    name: "CogImporter",
    components: {
        ToolTemplate
    },
```

```
    data () {
        return {
            cogList: [
                2016,
                2020
            ],
            cogSelected: undefined
        }
    },
    computed: {
        ...mapGetters("Tools/CogImporter", Object.keys(getters))
    },
    created () {
        this.$on("close", this.close);
    },
    methods: {
        ...mapMutations("Tools/CogImporter", Object.keys(mutations)),
        /**
         * Closes this tool window by setting active to false
         * @returns {void}
         */
        close () {
            this.setActive(false);

            // TODO replace trigger when Menu is migrated
            const model = Radio.request("ModelList", "getModelByAttributes", {id: "cogImporter"});

            if (model) {
                model.set("isActive", false);
            }
        },
        /**
         * translates the given key, checkes if the key exists and throws a console warning if not
         * @param {String} key the key to translate
         * @param {Object} [options=null] for interpolation, formating and plurals
         * @returns {String} the translation or the key itself on error
         */
        translate (key, options = null) {
            if (key === "additional:" + this.$t(key)) {
                console.warn("the key " + JSON.stringify(key) + " is unknown to the additional translation");
            }

            return this.$t(key, options);
        },
        makeStyle (year) {
            const bandNumber = year === 2020 ? 1 : 2;
            return {
                color: [
                    'interpolate',
                    ['linear'],
                    ['band', bandNumber],
                    0, [0, 0, 0, 0],
                    10, [4, 135, 29],
                    20, [137, 222, 137],
                    30, [14, 10, 214],
                    40, [229, 109, 109],
                    50, [180, 180, 77],
                    60, [231, 231, 25],
                    255, [0, 0, 0, 0]
                ],
            }
        },
        addLayer () {
            if (!this.cogSelected) {
                console.error("please");
                return;
            }
            const source = new GeoTIFF({
                            normalize: false, // keep original indices
                            sources: [
                                {
                                    url: 'https://data.mundialis.de/geodata/lulc-germany/classification_2020/classifi
```

```
cation_map_germany_2020_v02.tif',
                                },
                                {
                                    url: "https://data.mundialis.de/geodata/lulc-germany/classification_2016/classifi
cation_map_germany_2016_v0_1.tif"
                                }
                            ]
                        });
            source.on("error", function () {
                debugger
            })
            const layer = new TileLayer({
                name: "COG",
                style: this.makeStyle(this.cogSelected),
                source: source
            });
            const map = this.$store.getters['Map/ol2DMap'];
            // map.setView(source.getView());
            map.addLayer(layer);
        },

        removeLayer () {
            const map = this.$store.getters['Map/ol2DMap'];
            const cogLayer = map.getAllLayers().find(layer => layer.get("name") === "COG");

            if (cogLayer) {
                map.removeLayer(cogLayer);
            }
            else {
                console.error("Cannot find COG layer.");
            }
        }
    }
};
</script>

<template lang="html">
    <ToolTemplate
        :title="$t(`additional:modules.tools.cogImporter.title`)"
        :icon="glyphicon"
        :active="active"
        :render-to-window="renderToWindow"
        :resizable-window="resizableWindow"
        :deactivate-gfi="deactivateGFI"
        class="cog-importer"
    >
        <template
            v-if="active"
            #toolBody
        >
        <div>
            Landcover classification map of Germany<br/>
            Provided by <a href="https://mundialis.de">mundialis</a>.<br/>
            <a href="https://data.mundialis.de/geonetwork/srv/ger/catalog.search#/metadata/9246503f-6adf-460b-a31e-73
a649182d07">Link metadata</a>
        </div>
            <form
                id="cog-form"
                @submit.prevent="addLayer"
            >
                <select aria-label="Select COG" v-model="cogSelected">
                    <option disabled selected value="">Please choose</option>
                    <option v-for="cog in cogList" :value="cog">
                        {{ cog }}
                    </option>
                </select>
                <input
                    type="submit"
                    :value="$t('additional:modules.tools.cogImporter.addLayer')"
                />
            </form>
            <button
```

```
                @click="removeLayer">
                {{ $t('additional:modules.tools.cogImporter.removeLayer') }}
            </button>
        </template>
    </ToolTemplate>
</template>

<style lang="scss">
</style>
```

Vue.js Info

Vue.js lifecycle Hooks

FOSSGIS 2023Last modified: 2023-03-02 15:51:53

Addon Store

Im Masterportal wird der `state manager` Vuex verwendet.
Bei komplexen Anwendungen ist es sehr hilfreich einen zentralen Store zu haben, über den sämtliche Module inkl. aller Tools und Addons kommunizieren. Wenn unser Addon Infos aus der `Map-Component` erhalten möchte, beispielsweise um einen Layer hinzuzufügen, funktioniert dies über Vuex. Ein einfaches Beispiel erfolgt im Laufe dieser Addon-Entwicklung.

- Legen Sie im Pfad addons/cogImporter einen Ordner namens `store` an.
- Legen Sie eine `index.js` mit folgendem Inhalt an:

index.js

```
import getters from "./getters";
import mutations from "./mutations";
import state from "./state";

export default {
    namespaced: true,
    state: {...state},
    mutations,
    getters
};
```

- Legen Sie die folgenden Dateien an:

Zunächst den State des Addons:

state.js

```
/**
 * User type definition
 * @typedef {Object} cogImporterState
 * @property {Boolean} active if true, CogImporter will be rendered
 * @property {String} id id of the CogImporter component
 * @property {String} glyphicon icon next to title (config-param)
 * @property {Boolean} renderToWindow if true, tool is rendered in a window, else in sidebar (config-param)
 * @property {Boolean} resizableWindow if true, window is resizable (config-param)
 * @property {Boolean} isVisibleInMenu if true, tool is selectable in menu (config-param)
 * @property {Boolean} deactivateGFI flag if tool should deactivate gfi (config-param)
 * @property {string} cogList List of avaiable COG
 * @property {string} cogSelected Selected COG
 */
const state = {
    active: false,
    id: "cogImporter",
    // defaults for config.json parameters
    name: "translate#additional:modules.tools.cogImporter.title",
    glyphicon: "glyphicon-wrench",
    renderToWindow: true,
    resizableWindow: true,
    isVisibleInMenu: true,
    deactivateGFI: true,
    cogList: undefined,
    cogSelected: undefined
};

export default state;
```

Anschließend die `getters` und `mutations` :

getters.js

```
import {generateSimpleGetters} from ".../../../src/app-store/utils/generators";
```

```
import state from "./state";

const getters = {
    ...generateSimpleGetters(state)
};

export default getters;
```

mutations.js

```
import {generateSimpleMutations} from "../../../src/app-store/utils/generators";
import state from "./state";

const mutations = {
    /**
     * Creates from every state-key a setter.
     * For example, given a state object {key: value}, an object
     * {setKey:   (state, payload) => *   state[key] = payload * }
     * will be returned.
     */
    ...generateSimpleMutations(state)
};

export default mutations;
```

FOSSGIS 2023Last modified: 2023-03-02 15:51:53

Übersetzungsdateien

Analog zu den `i18n` Dateien im Core, gibt es für jedes Addon entsprechende `json` Dateien.

- Legen Sie einen Ordner namens `locales` im Pfad des Addons an.
- Legen Sie einen Ordner `de` an und fügen Sie folgende Datei ein:

additional.json

```
{
    "modules": {
        "tools": {
            "cogImporter": {
                "title": "COG Importieren",
                "addLayer": "Hinzufügen",
                "removeLayer": "Entfernen"
            }
        }
    }
}
```

- Legen Sie einen Ordner `en` an und fügen Sie folgende Datei ein:

additional.json

```
{
    "modules": {
        "tools": {
            "cogImporter": {
                "title": "Import COG",
                "addLayer": "Add",
                "removeLayer": "Remove"
            }
        }
    }
}
```

FOSSGIS 2023Last modified: 2023-03-02 15:51:53

Addon Konfiguration

Ein Addon muss an zwei Stellen im Masterportal konfiguriert werden.

1. config.js

- Fügen Sie an beliebiger Stelle das folgende Property hinzu:

```
    addons: ["cogImporter"],
```

2. config.json

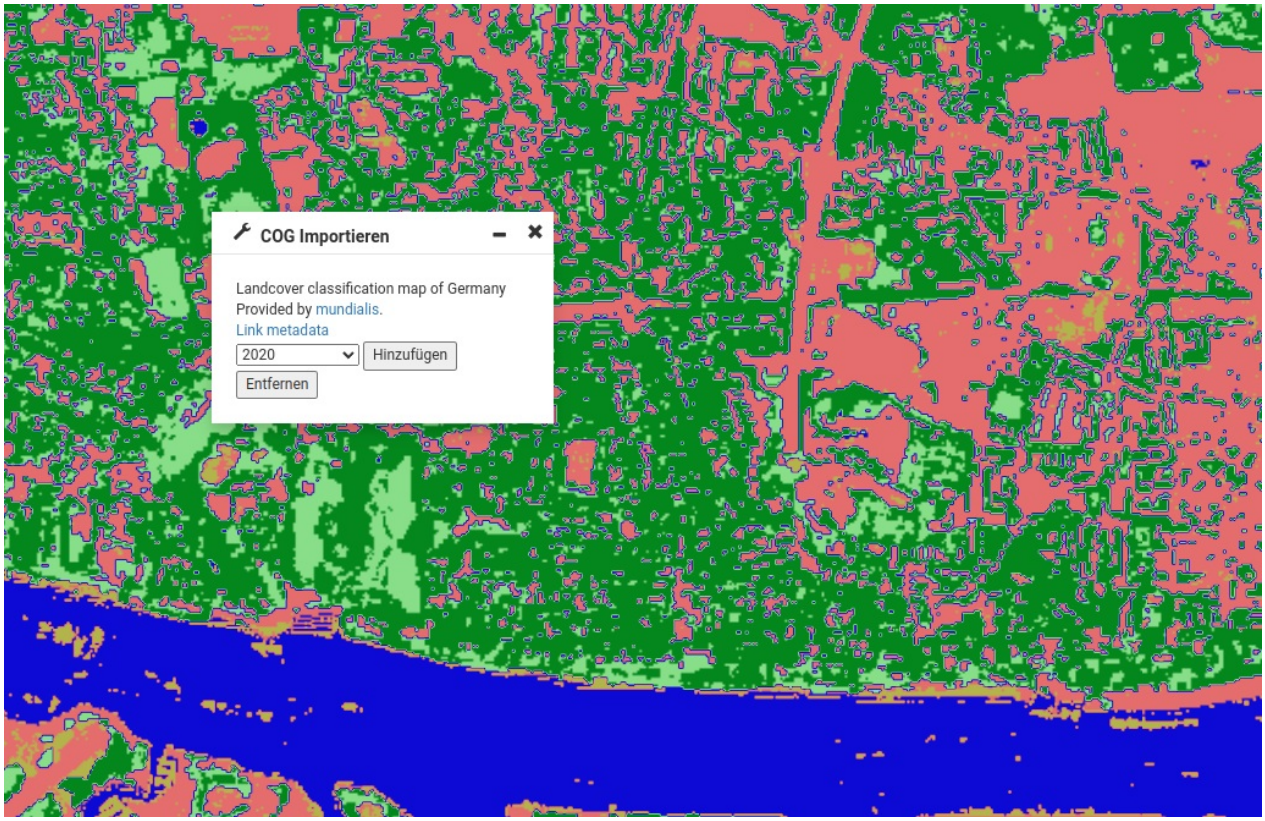
Fügen Sie im Block `tools` -> `children` den folgenden Block hinzu:

```
        "cogImporter": {
          "name": "translate#additional:modules.tools.cogImporter.title",
          "glyphicon": "glyphicon-wrench"
        }
```

Nun ist das Addon fertig eingebunden. ❗ Bitte beachten Sie, dass dies keinesfalls für eine Produktiv-Anwendung gedacht ist, es dient lediglich der Demonstration.

❓ Was fehlt: :

- Dokumentation
- Code Dokumentation
- Tests
- Konfigurierbarkeit (welche COG können eingebunden werden)
- Hinzufügen der Layer in den Tree
- ...



COG Addon

FOSSGIS 2023Last modified: 2023-03-02 15:51:53