

# Einführung in GeoNode 4

---

Willkommen auf der Schulungsplattform der [CSGIS gbr](#).

Diese Dokumentation führt in die Benutzung von [GeoNode 4](#) ein.

GeoNode ist ein Content Management System für räumliche Daten.

Auf der linken Seite finden Sie Übungen und Informationen für

- Benutzer die eigene Ebene publizieren möchten
- Benutzer die über Administratoren-Rechte besitzen
- Benutzer die mit der Entwicklung mit GeoNode starten möchten
- System Administratoren die auf Server Ebene das Portal warten

Wir verwenden in der Dokumentation aus Einfachheit größtenteils `User` oder `Benutzer`.  
Gemeint sind hiermit jedoch alle Geschlechter.

[Download der Dokumentation als PDF](#)

# Einführung in GeoNode 4.0 für Anwender

---

## Inhalt:

---

In dieser Schulung werden wir folgende Themen mit GeoNode behandeln:

- Publikation von Geodaten, Karten und Dokumenten
- Analysieren, Filtern und Abfragen
- Karten und Daten teilen
- Einbinden und Bereitstellen eigener WMS Dienste
- Legendenerstellung
- Kartenerstellung
- Überblick: Geostories, Dashboards, Diagramme und Widgets

## Ziel:

---

In diesem Kurs arbeiten Sie mit GeoNode mit konkreten Beispielen und Fragestellungen aus der Praxis. Wir bauen gemeinsam ein Geoportal auf. Schritt für Schritt lernen wir GeoNode kennen.

Wir beschäftigen uns mit der Publikation, dem Management und der Analyse von Geodaten in GeoNode. Den Inhalt der Schulung runden wir mit weiteren Informationen über das GeoNode Projekt ab: Organisation des Open Source Projektes, Entwicklerteam, Code Repository, etc.

Nach dem Kurs besitzen Sie einen fundierten Überblick über GeoNode, dessen wichtigste Funktionen und Möglichkeiten.

# Registrieren

Im ersten Schritt erfolgt eine Registrierung in der GeoNode Plattform. Ohne Registrierung können die Daten (Karten, Datensätze, Dokumente, etc.) visualisiert und geteilt aber nicht bearbeitet werden. Nachdem sich ein Anwender erfolgreich registriert hat, kann sich gleich anmelden und GeoNode im vollen Umfang benutzen.

Die Option zum registrieren kann nach Bedarf ausgeblendet werden sodass ein Admin die Registrierung eigenständig durchführt. Standardmäßig läuft die Registrierung automatisch. Man kann aber diese Option ändern sodass ein Admin die Registrierung bestätigen muss

## User Profil

Im Profil stehen dem User u.a. folgende Funktionen zur Verfügung:

- Andere Anwender kontaktieren und Nachrichten senden
- Emailadresse der Registrierung ändern
- Passwort ändern
- Benachrichtigungseinstellungen
- Andere Anwender einladen Ihrer GeoNode Plattform zu benutzen

*Es lohnt sich diese Optionen in Ruhe zum Lesen und hier entscheiden mit welchen Aktionen eine Email an den Anwender gesendet werden soll (standardmäßig sind alle Optionen ausgewählt).*

*Mit einer GeoNode Instance, die von vielen Anwendern aktiv benutzt wird, werden Sie dann viele Emails bekommen!)*

Außerdem können registrierte Benutzer:

- Die letzten Aktivitäten von allen Benutzern sehen und zugreifen (Ebenen, Karten, Dokumenten, Geostories und Dashboards)
- eine Liste mit allen Favoriten Inhalten sehen
- Mailbox abfragen
- Das Hilfe Dokument (mit verlinkten Inhalten) lesen

Weitere empfehlenswerte GeoNode Dokumentationen sind:

- [GeoNode Handbuch](#)
- [Mapstore Handbuch](#)

*Auf der Startseite, im Menü Über kann man visualisieren, filtern und sehen welche User im GeoNode registriert sind und welche Daten haben sie veröffentlicht*

! :exclamation: This is very important | | — — — — — |

## Übung

1. Finden Sie die Email die mit Ihrem Account verknüpft ist
2. Erstellen Sie die gewünschten Aktionen unter Benachrichtigungen



# Publikation von Ressourcen

---

Folgende Ressourcen können in GeoNode publiziert und bearbeitet werden:

- **DATENSÄTZE** (Vektor, Raster, Remote oder Zeit Serien). Folgende Dateitypen können hochgeladen werden: ESRI Shapefile, GeoTIFF, Comma Separated Value (CSV), Zip Archive, XML Metadata File, Styled Layer Descriptor (SLD)
- **DOKUMENTE** ( Bilder, Textdateien, Videos, PDF Dokumenten, Tabellen, etc ). Diese Dateitypen sind unterstützt: .txt, .log, .doc, .docx, .ods, .odt, .sld, .qml, .xls, .xlsx, .xml, .bm, .bmp, .dwg, .dxf, .fif, .gif, .jpg, .jpe, .jpeg, .png, .tif, .tiff, .pbm, .odp, .ppt, .pptx, .pdf, .tar, .tgz, .rar, .gz, .7z, .zip, .aif, .aifc, .aiff, .au, .mp3, .mpga, .wav, .afl, .avi, .avs, .fli, .mp2, .mp4, .mpg, .ogg, .webm, .3gp, .flv, .vdo, .glb, .pcd, .gltf
- **KARTEN** (Publikation von bestehenden GeoNode Datensätze). Der Anwender kann beliebigen Datensätze hinzufügen und eigenständig eine Karte mit den verschiedenen Ebenen erstellen.
- **GEOSTORIES** sind online Berichte die man mit den GeoNode hochgeladenen oder mit externen Ressourcen (videos, webseiten, etc.) erstellen kann
- **DASHBOARDS** sind Bereiche in GeoNode in dem der Benutzer zusammenfassende Geoinformationen mit Widgets wie Diagramme, Karten, Tabellen, Texte, etc ( die miteinander interaktiv verbunden sein können) präsentieren kann.

- *Alle Ressource in GeoNode sind standardmäßig öffentlich. Der Benutzer, der den GeoNode Ressource erstellt hat, kann entscheiden ob diese Daten für alle oder nur für registrierte Benutzer sichtbar sein sollen*
- *Nur die Person, die den GeoNode Ressource gehört (i.d.R. die Person, die den Ressource erzeugt hat) kann die Daten bearbeiten. Diese Person kann aber weitere Berechtigungen hinzufügen und erlauben dass andere Benutzer die Daten ansehen, herunterladen, bearbeiten oder verwalten können*
- *Der Besitzer des GeoNode Ressources kann entscheiden, ob einen anderen Benutzer die Daten gehören sollen (siehe optionale Metadaten):*

## Übung

---

1. Laden Sie in GeoNode die Shape Dateien places, natural, administrative, railways und roads hoch
2. Erstellen Sie einen Filter damit Sie nur Ihre Daten ansehen können

# Geodaten publizieren und bearbeiten

---

Nach dem die Daten hochgeladen sind stehen uns unter Datensatz ansehen folgende Funktionen zur Verfügung:

**Speichern:** sollen Änderungen an den Daten, Symbologie, Berechtigungen, etc. vorgenommen werden, können Sie diese hier dauerhaft speichern. Mit "speichern als" erzeugen Sie einen neuen Dataset.

Im Menü bearbeiten können wir:

**Informationen anzeigen:** lassen (allgemeine Informationen zu den Ressourcen). Hier haben Sie auch die Möglichkeit diesen Ressourcen als Favorit zu speichern, den Link zu teilen oder die Datei zum Downloaden.

## Daten bearbeiten

---

Geometrien und Tabellen können in GeoNode bearbeitet werden. Die Optionen für die Digitalisierungen der Geometrien sind begrenzt. Wir schauen die Editierungsoptionen am besten mit einer Übung an (detaillierte Informationen finden Sie [hier](#)). Der erste Schritt ist die Bearbeitungsmodus einzuschalten.

### Übung:

Editierungen im Dataset places

1. Starten Sie die Editierung. Wählen Sie Schwabing aus und zoomen Sie zu den Extent
2. Filtern Sie Schwabing, Schwabing-West und Schwabing-Ost, lassen Sie nur diese Orte sichtbar auf der Karte und verändern Sie die Einwohnerzahlen (Spalte population) dieser Orte.

1. Untersuchen Sie weitere Filteroptionen wie "Interessengebiet"
2. Erzeugen Sie einen Punkt und vergeben Sie die Attribute in die Tabelle
3. Verschieben Sie einen Punkt in die Karte
4. Löschen Sie einen beliebigen Punkt

Editierungen im Dataset natural

1. Fügen Sie einen neuen Polygon hinzu
2. Digitalisieren Sie einen neuen benachbarten Polygon (Snap Optionen)

Zum selektieren in die Karte nur ein Mal klicken

Selektieren von mehreren Objekten können wir über Filter machen

## Stil bearbeiten

---

Es gibt 3 verschiedene Gruppen (Regeln) von Stilen die man benutzen kann (sie können miteinander kombiniert werden)

Außerdem gibt es die Möglichkeit für die verschiedene Regel die Daten zu filtern (wenn man zum Beispiel nur bestimmten Daten darstellen möchten – type=forest -) oder nur innerhalb von einem bestimmten Maßstab die Daten darstellen zu lassen.

Innerhalb der Markierung Option haben wir 2 verschiedene Legenden zur Verfügung:

- Einfacher Stil
- Klassifizierungsstil (mit verschiedenen Methoden)

## Übung

Symbologie von places

1. Erstellen Sie diese zwei verschiedene Legende für den Ressource *places* und speichern Sie die Ergebnisse (mit der Änderung .sld):
2. Lassen Sie die *places* mit einer einfachen Symbol darstellen und laden Sie die gespeicherten Stile (sld Dateien) hoch (Bearbeiten → Stil hochladen)

## Übung

Symbologie von railways

1. Erstellen Sie die railways wie in der unteren Abbildung dar. Benutzen Sie den einfacher Stil und erstellen sie die entsprechenden Filter ein (*type=subway und type =tram*)
2. Speichern Sie den Stil als sld und Laden Sie diese Datei als Dokument in GeoNode hoch
3. Benachrichtigen Sie Ihren Kollegen dass es eine neue Legende gibt und schauen Sie Ihren Mailbox die gesendeten/empfangenen Nachrichten an.

## Übung

Symbologie von roads

1. Lassen Sie roads klassifiziert darstellen nur mit folgenden Kategorien der Spalte "Type": primary, residential, secondary, tertiary. Roads soll nur erscheinen innerhalb diesen Maßstäbe 1:144448 und 1:9028
2. Wie kann man erreichen dass nur den type residential innerhalb von diesen Maßstäbe erscheint aber dass die andere Typen immer sichtbar bleiben?

*Die symbolregel ermöglicht die Erstellung von Legenden mit Symbolen wie Grafikdateien(svg, png, etc.). Man braucht die url wo die Bilder gehostet sind*

## Metadaten bearbeiten

Die Metadaten in GeoNode haben 2 wesentlichen Zwecken:

- sie geben mehr Auskunft über die Daten
- sie vereinfachen die Suche von Ressourcen durch andere Benutzer und damit können die Daten leichter abgerufen werden.

Ausführliche Informationen über die Bearbeitung von Metadaten finden Sie [hier](#)

## Übung

## Metadaten von administrative

1. Thumbnail aktualisieren
2. Kategorie population speichern unter die Grundlegende Metadaten
3. Unter regions soll Germany gespeichert werden
4. Speichern Sie die Metadaten und suchen sie auf der Startseite in GeoNode nach Daten mit zum Beispiel die Kategorie population oder nach der Region Germany
5. Probieren Sie die Suche nach Daten mit dem Filter Option "Ausmaß"

*Im Schritt 4 (Datensatz Attribute) lässt sich unter Anzeigetyp speichern, wie die gespeicherten Daten in einer Spalte angezeigt werden sollen. Mit Klick auf einen Objekt in die Karte erscheinen die Informationen entsprechend formatiert.*

## Ressource teilen

Beim Erstellen oder Hochladen eines neuen Datensatzes müssen Sie festlegen, wer diesen Datensatz anzeigen, herunterladen, bearbeiten und verwalten kann. Standardmäßig können nur Eigentümer Datensätze bearbeiten und verwalten, jeder kann sie ansehen.

Weitere führenden Informationen finden Sie unter [Dataset permissions](#)

Sie können die folgenden Berechtigungen festlegen:

- Anzeigen (ermöglicht das Anzeigen des Datensatzes).
- Herunterladen (ermöglicht das Anzeigen und Herunterladen des Datensatzes).
- Bearbeiten (ermöglicht das Ändern der Metadaten, der Attributen und Geometrien und des Stils).
- Verwalten (ermöglicht das Bearbeiten, Löschen, Ändern der Freigabeoptionen und Publizieren eines Datensatzes). Datensätze die nicht publiziert sind können nur von Admin gesehen und bearbeitet werden.

## Übung

Berechtigungen von places

1. Erstellen Sie dass die places nicht heruntergeladen werden können. Melden Sie sich ab und prüfen Sie dass keine Option zum herunterladen angeboten ist.
2. Erlauben Sie dass die places von anderen Benutzer verwaltet werden können

## Filtern

Sie können mit dieser Option die Anzeige der Daten beeinflussen mit folgenden Möglichkeiten:

- Attribut
- Interessengebiet

Beide Optionen können auch gleichzeitig benutzt werden:

## Herunterladen



Mit dieser Funktion können Sie die datasets herunterladen in mehreren Formaten (GeoJSON, GML, Shapefile, CSV, GeoPackage oder KML).

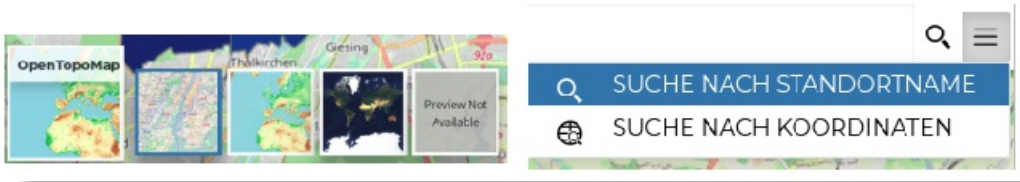
Sie können entscheiden ob Sie die Daten mit der ursprüngliche Projektion (prj Datei die beim Import der Daten benutzt wurde) herunterladen worden oder in WGS84 (EPSG 4326).

Außerdem können Sie definieren ob sie nur die gefilterte Daten oder die Daten des aktuellen Kartenfenster herunterladen möchten.

# Karten erstellen

In einer Karte können mehreren datasets hinzugefügt werden. Karten sind eigene GeoNode Ressourcen die man auch publizieren oder mit anderen Benutzer bearbeiten/teilen kann.

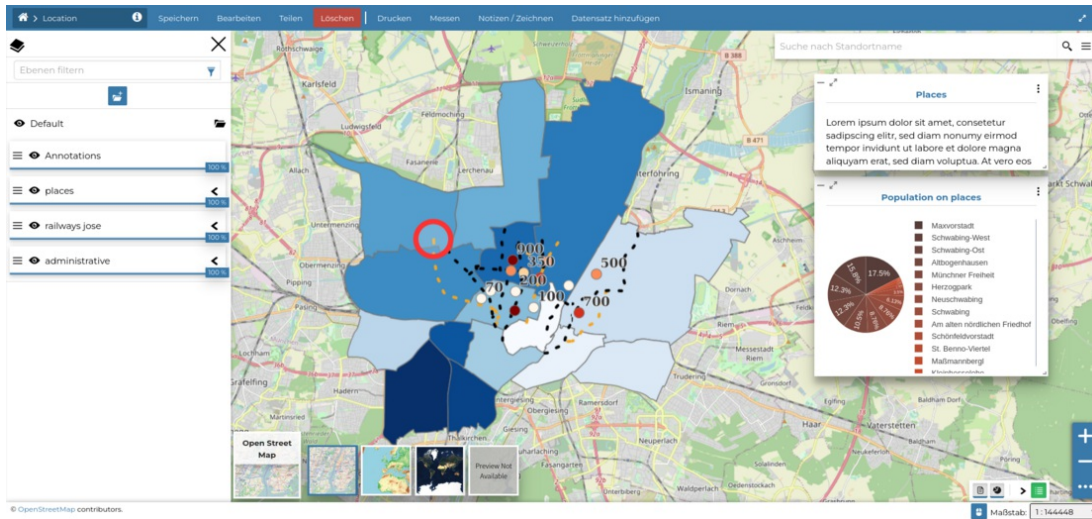
In die Karte stehen auch mehreren Hintergrundkarten oder eine Adresssuche zur Verfügung.



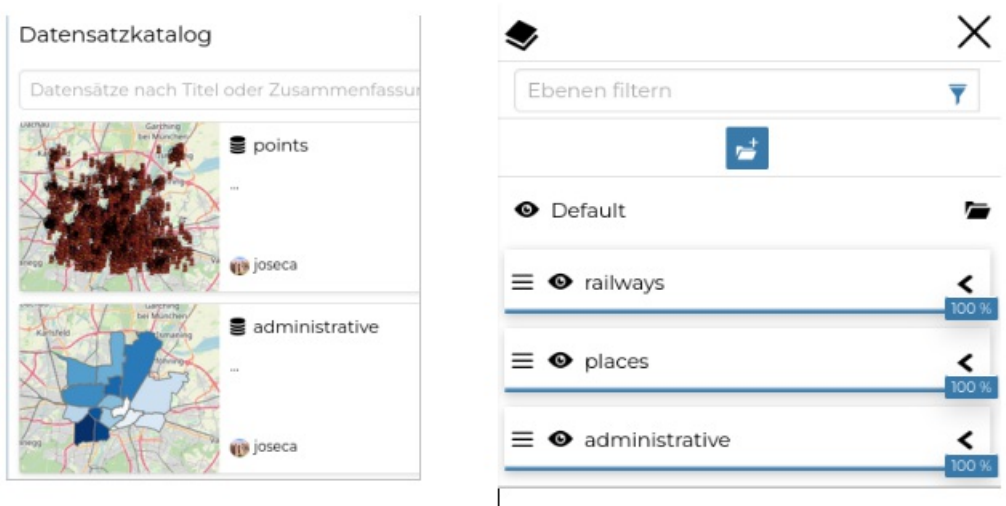
Zusätzlich können mehreren **Widgets** (z.B. Diagramme, Text, Tabellen) hinzugefügt werden um die Karte mit anderen Informationen zu begleiten.

## Übung

Erstellung folgende Karte

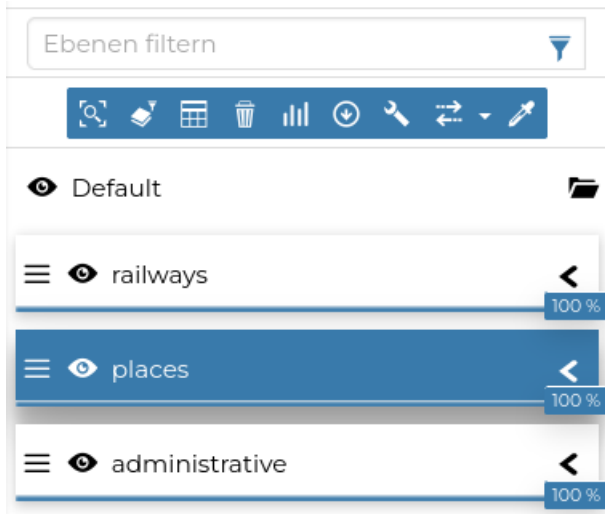


1. Ressource hinzufügen → Karte erstellen → Speichern
2. Datensatz hinzufügen → Im Datensatzkatalog die Ressource administrative, railways und places auswählen.



3. Mit klick auf einer Ebenen erscheint eine Reihe von interessanten Funktionalitäten. Wir können hier zum Beispiel:

- die Ebene filtern und nur einen Teil der Daten in die Karte anzeigen lassen
- die Attributtabelle öffnen und die Daten hier bearbeiten
- widgets für die ausgewählte Ebene erstellen
- die Symbologie der Ebene hier verändern

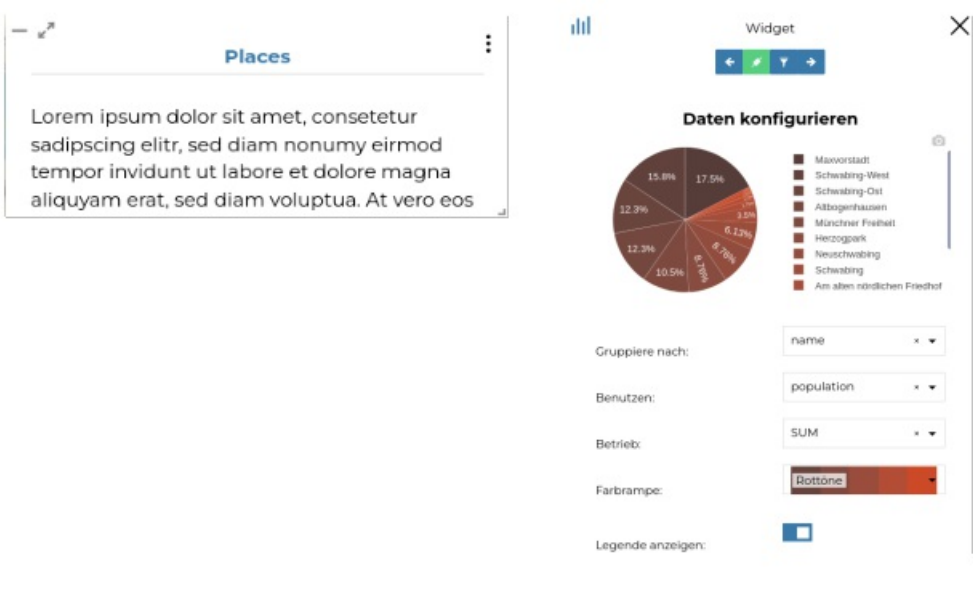


1. Verändern Sie die Symbologie von *places*: öffnen Sie die Datei *places.sld*, kopieren Sie den Inhalt und fügen Sie ihn in den Code-Editor ein.

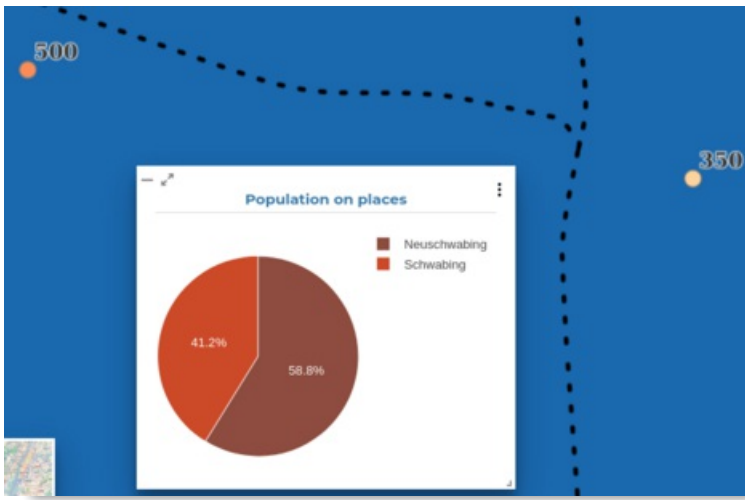
In der Karte können wir die Symbole einer Ebene ändern ohne die Symbologie der ursprünglichen Ressource zu verändern

An den Stilen vorgenommene Änderungen gelten nur für den aktuellen Layer in Ihrer Karte und nicht für den ursprünglichen Datensatz.

1. Prüfen Sie dass die Symbologie der GeoNode Ressource *places* sich nicht geändert hat.
2. Begleiten Sie die Karte mit folgenden Widgets des Layers *places*:



1. Beachten Sie dass die Diagramme zeigen immer die Daten die man in den aktuellen Ausschnitt der Karte sieht



1. Fügen Sie die Tabelle der Ebene Administrative in die Karte ein
2. Schalten Sie alle Widgets ein und aus und lassen Sie nur den Text und das Diagramm sichtbar
3. Lassen Sie die Ebene administrativ transparent darstellen

administrative

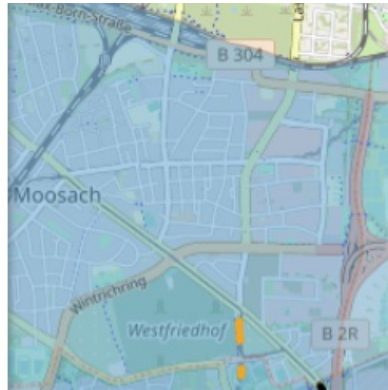
> Allgemeine Einstellungen

▼ Sichtbarekeitseinstellungen

Deckkraft %

50

Sichtbarkeitsgrenzen

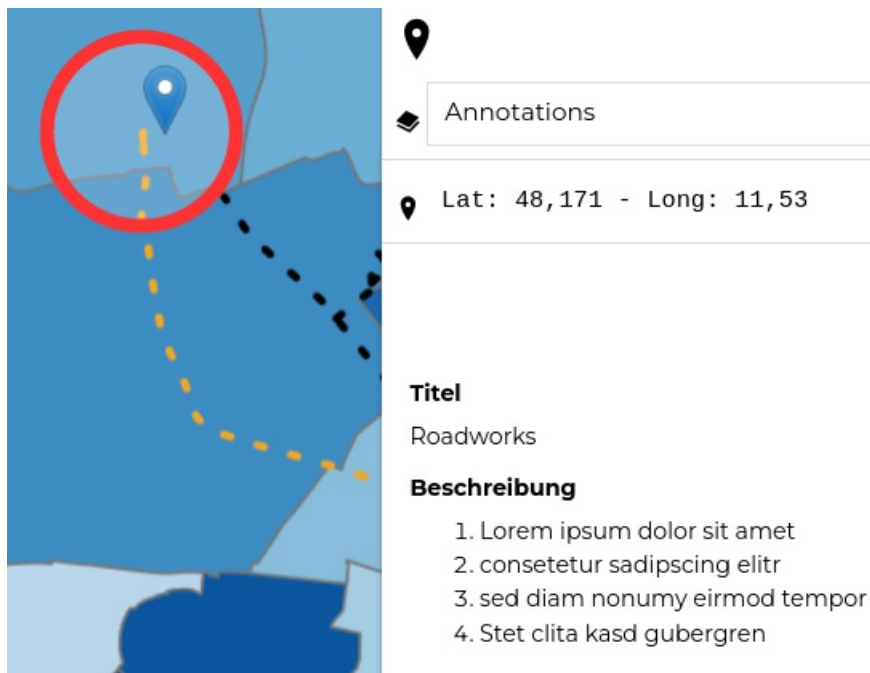


1. Mit dem Tool Notizen/Zeichen fügen Sie eine Anmerkung (Annotation) hinzu. Danach lassen Sie sich mit einfachen Klick auf die Karte die Informationen anzeigen lassen

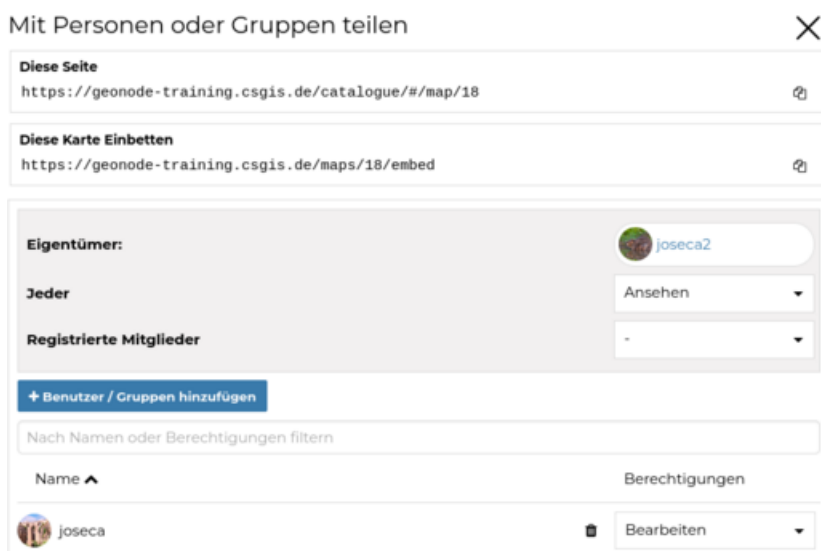
GeoNode gibt die Informationen von alle Ebenen aus, wo man geklickt und den Hinweis von den Ebenen wo dort

Für die folgenden Ebenen gibt es keine Objekte: **railways, places**


keine Daten gibt:



1. Wie alle GeoNode Ressource, Sie Können Berechtigungen an die Karte vergeben um zum Beispiel erlauben dass jemand anders diese Karte auch mitgestalten kann



1. Speichern Sie Ihre Karte und lassen Sie eine PDF in A4 Format ausdrucken lassen

<p><b>Title</b></p> <input type="text" value="Location"/>  <p><b>Beschreibung</b></p> <input type="text" value="Gib eine Beschreibung ein..."/>  <p><b>Format</b></p> <input type="text" value="PDF"/>  <p>Maßstab 1:273.438 <input checked="" type="checkbox"/> in Druck einschließen</p> <p><input type="checkbox"/> Raster mit Etiketten hinzufügen</p> <p><b>Layout</b></p> <div style="border: 1px solid #ccc; padding: 5px;"> <p><b>Blattgröße:</b></p> <input type="text" value="A4"/>    <p><input type="checkbox"/> Legende einschließen</p> <p><input type="checkbox"/> Legende auf einer separaten Seite</p> <p><input checked="" type="radio"/> Querformat    <input type="radio"/> Hochformat</p> </div> <p><b>Legenden Optionen</b></p>	<p><b>Auflösung:</b></p> <input type="text" value="96 dpi"/>    <div style="background-color: #007bff; color: white; text-align: center; padding: 5px; float: right;">Drucken</div>
---	--

# Geostories

---

## testlink

Eine Geostory ist ein GeoNode Ressource mit der man ein Artikel mit interaktive Inhalten (Videos, Text, Bilder, Webseiten, dynamische Karten, andere GeoNode Ressourcen, etc.) veröffentlichen kann.

Eine Geostory kann mit folgenden Elementen aufgebaut werden:

- Titelausschnitt
- Bannerbereich
- Absatzabschnitt
- Immersive Section
- Geokarussell
- Medienabschnitt
- Webseitenabschnitt

Ein Beispiel mit allen Elementen einer Geostory finden Sie [hier](#)

Im [Handbuch von Mapstore](#) sind alle Funktionen detailliert beschrieben.

Mit der nächsten Übung bauen wir selber eine Geostory auf.

## Übung

### Erstellung einer Geostory

1. Ressource hinzufügen → Geostory erstellen → Speichern
2. Im ersten Schritt wird uns angeboten dass wir mit einem Titelausschnitt anfangen. Hier finden wir 2 widgets:

Mit einfachen Klicks bearbeiten und formatieren Sie den Titel.

Die Größe des Titelausschnittes können Sie anpassen oder einen Hintergrund hinzufügen mit Bildern, Videos oder Karten.

Der erste Teil unserer Geostory wäre fertig. Wir haben jetzt den Titelausschnitt mit 2 Elementen: Titel und Hintergrund (eine GeoNode Karte).

3. Im zweiten Schritt erweitern wir unsere Geostory mit einem Absatzabschnitt.

Hier können wir hinzufügen:

- Text
- Medien (Bilder, Videos, Karten)
- eine Webseite

4. Lassen wir unser Geostory mit einer immersive section erweitern. Sie besteht aus zwei Elementen: dem Hintergrund und dem Inhalt.

Der Unterschied mit der vorherigen Option (Absatzabschnitt) ist dass die Inhalte bleiben zusammen im gleichen Abschnitt

5. Im nächsten Schritt möchten wir uns beschäftigen mit der Geocarousel Option:

- Der Hintergrund dient für den gesamten Abschnitt.
- Der Text und das untere kleine Bild werden verknüpft mit einer bestimmten Ort in die Karte.
- Sie können damit beliebigen Anmerkungen an bestimmten Orten hinzufügen.



# Dashboards

---

Sie eignen sich gut für die Präsentation von Ergebnissen oder Zusammenfassungen von Daten (siehe [zum Beispiel](#)).

## Übung

---

### Erstellung eines Dashboards

1. Um ein Dashboard zu machen müssen Sie wie gewohnt den entsprechenden GeoNode Ressource erstellen:  
Ressource hinzufügen → Dashboard erstellen → Speichern

In einem Dashboard können Sie folgende widgets integrieren:

2. Fügen Sie zuerst die Karte die sie im Punkt 1.4 erzeugt haben
3. Danach bringen Sie in den dashboard folgende Diagramm mit den Einwohnerzahlen.

In Dashboard können die Inhalte dynamisch verknüpft mit anderen Inhalte sein (zum Beispiel wenn man in die Karte zoom würde das Diagramm nur die Daten anzeigen die man gerade in die Karte sieht)

4. Fügen Sie die Tabelle der places in den Dashboard hinzu. Diese Daten sollen auch mit der Karte und Diagramm verknüpft werden. Mit "Verbindungen ausblenden" sehen Sie eine rote Umrandung oben über die widgets die Ihnen informieren welche Elementen des Dashboards miteinander verknüpft sind.
5. Als letztes können Sie zum Beispiel einen Zähler hinzufügen mit der gesamten Einwohnerzahl.

# Geonode mit Admin Rechten

---

## Inhalt:

---

Dieser Schulungsbereich gibt eine Einführung für Administratoren.

Wir werden uns mit folgenden Themen befassen:

- Welche Berechtigung besitzt die Admin Rolle
- Wie werden Gruppen Rechte vergeben
- Die Benutzung des Django Admins um beispielsweise User, oder Datasets zu verwalten
- Was sind Ankündigungen

# Die GeoNode Rollen

---

Nach der Installation von GeoNode kennt das System folgende Rollen:

- Anonym
  - Alle nicht angemeldeten Besucher
- Registriert
  - Nutzer mit einem Benutzer Account
- Mitarbeiter
  - Registrierte Nutzer mit erweiterten Rechten
- Administratoren
  - Registrierte Nutzer mit vollständigen Rechten.

Im folgenden Betrachten wir die dritte und vierte Rolle Mitarbeiter und Administrator.

## Der Administrator im Frontend

---

Frontend meint den für die Öffentlichkeit sichtbaren Bereich. Ihm gegenüber steht das Backend, welches das System im Hintergrund bezeichnet.

Administratoren Rollen dürfen per se alles sehen, bearbeiten oder löschen.

Das erste Administratoren Konto wird bei der Installation des Systems angelegt. Die Definition des Users befindet sich [hier](#).

## Zusätzliche Menüpunkte

---

Administratoren werden im Menü zusätzliche Optionen angezeigt:

Dies sind:

- Invite User
  - Neue Benutzer einladen
- Add User
  - Neue User anlegen
- Create Group
  - Gruppen erstellen um User zu gruppieren.

### Invite User

Über die Maske Invite User besteht die Möglichkeit, eine E-Mail Einladung / Aufforderung an Dritte zu senden. Die E-Mail bittet um Registrierung.

### Add User

Über die Maske "Add User" besteht die Möglichkeit neue Nutzer anzulegen.

### Create Group

Über die Maske “Create Group” können Gruppen angelegt werden..

Gruppen sind eine Möglichkeit User zusammenzufassen. Dies ist sinnvoll um Datensätze schnell mit Rechten für einen größeren Personenkreis zu geben.

Standardmässig dürfen nur Administratoren neue Gruppen anlegen sowie Gruppen Administratoren festlegen. Wie wir auch der Mitarbeiter Rolle erlauben Gruppen anzulegen, sehen wir später.

Beim anlegen einer neuen Gruppe stehen folgende Eingabefelder zur Verfügung:

- Titel der Gruppe
- Logo der Gruppe
  - Wird für die Übersicht verwendet
- Beschreibung der Gruppe
- E-Mail
  - E-Mail die verwendet wird, um ein oder alle Gruppenmitglieder, ähnlich einer Mailingliste, zu kontaktieren.
- Schlüsselwörter
  - Eine durch Leerzeichen oder Kommas getrennte Stichwortliste
- Zugriff
  - Öffentlich: Jeder registrierte Nutzer kann eine öffentliche Gruppe betrachten und dieser beitreten.
  - Öffentlich (nur auf Einladung): Jeder registrierte Benutzer kann die Gruppe betrachten. Nur eingeladene Benutzer können beitreten.
  - Nur eingeladene Benutzer können teilnehmen.
- Kategorien
  - Weitere Gruppierung der Gruppe. Funktioniert in GeoNode 4 nur fehlerbehaftet (Bug: Kann nicht mehr gelöscht werden)

Nach dem anlegen einer neuen Gruppe erscheint diese für alle User in der Gruppen Übersicht

Weiterhin können Administratoren User hinzufügen sowie Details der Gruppe ändern.

In der Pause haben wir zu diesem Zweck Ihren Accounts Administratoren Rechte gegeben.

## Übungen

---

### Neue Gruppe

---

1. Legen Sie eine neue Gruppe an und weisen Sie der neuen Gruppe einige Mitglieder zu.

## Weiterführende Links

---

- [GeoNode Docs – Gruppen anlegen](#)

# Gruppenrechte

---

Um Ihren Datensätze Gruppenberechtigungen zuzuweisen müssen Sie keine Administrator sein. Jedoch, wie im vorherigen Kapitel gezeigt um diese zu erstellen.

## Rechte zuweisen

---

Ihren Datensätzen und Karten können neben Userrechte über das Menü teilen Gruppen Rechte hinzugefügt werden.

Im bekannten Panel zur Rechtevergabe wählen Sie die jeweiligen Gruppen:

! [[Ebenen Berechtigungen einstellen](#)

Und vergeben hiernach die gewünschte Sichtbarkeit:

## Übungen

---

1. Vergeben Sie bei Ihrem Datensatz Gruppen Rechte für eine beliebige Gruppe.

## Weiterführende Links

---

1. [GeoNode Docs, Share Options](#)

# Django Admin

---

Das GeoNode System basiert auf einem Python Webframework namens [Django](#).

Neben Hilfestellungen in vielerlei Bereichen, stellt es eine automatisierte Administrations Oberfläche zur Verfügung. Die Oberfläche generiert sich hierbei aus den Datenbank Definitionen der einzelnen “Apps”.

Im Django Kontext sind Apps, in Module aufgeteilte Abschnitte wie Shop, Benutzerverwalten etc. GeoNode besitzt zahlreiche GIS spezifische “Apps”. Wie Datasets, Maps, etc.

Um auf die Administrationsoberfläche zugreifen zu können, muss ein User Mitarbeiter, oder Administrator Status besitzen.

Die Administrationsoberfläche erreichen Sie nach dem Login über das Menü, welches die eingestellte Sprache berücksichtigt.

Die hierauf folgende Übersicht listet alle “Django Apps” die GeoNode besitzt.

# Benutzer verwalten

---

Betrachten wir zunächst einen sehr einfach jedoch wesentlichen Abschnitt des Django Admins. Die Benutzerverwaltung.

In der hierauf öffnenden Übersicht zeigen sich alle im System vorhandenen Benutzer. Die Tabelle zeigt hierbei ID, Name, E-Mail Adresse, Status und Aktiv des Benutzers.

## auflisten

---

Achten Sie darauf dass alle Nutzer eine E-Mail Adresse hinterlegt haben!

Über die Checkbox und das Menü am Seitenfuß können Sie User löschen. Sowie über den Button in der rechten oberen Ecke neue User anlegen.

## löschen

---

Durch klick auf die ID gelangen Sie zur Detailseite des jeweiligen Benutzers um diesen zu editieren.

## editieren

---

Wie auch in der Übersicht können Sie den User löschen oder Änderungen speichern (Seitenfuß). Das Passwort ändern, oder die Berechtigungen ändern:

- Aktiv
  - Legt fest, ob dieser Benutzer aktiv ist. Kann deaktiviert werden, anstatt Benutzer zu löschen.
- Mitarbeiter Status
  - Legt fest, ob sich der Benutzer an der Administrationsseite anmelden kann.
- Administrator Status
  - Legt fest, dass der Benutzer alle Berechtigungen hat, ohne diese einzeln zuweisen zu müssen.

## Rechte zuweisen

---

Wie wir gesehen haben dürfen Administratoren alle Django Admin Bereiche aufrufen und ausführen.

Benutzer mit Mitarbeiter Status können sich zwar in die Django Oberfläche einloggen, besitzen aber so gut wie keine Möglichkeiten administrative Aufgaben zu übernehmen.

## Gruppe anlegen

Um Mitarbeiter mit weiteren Rechten auszustatten, ist es nötig

- eine Gruppe für die Mitarbeiter zu erstellen
- der Gruppe die jeweiligen Rechte zu geben
- die Mitarbeiter der Gruppe hinzuzufügen

## Gruppe / Rechte erstellen

Als *Administrator* legen wir im Django Admin eine neue Gruppe mit frei wählbarem Namen an. Wir verwenden hier "Editoren".

In der Übersicht klicken wir auf Gruppen:

Und drücken in der hierauf folgenden Ansicht den Button um eine neue Gruppe zu generieren.

.

In der folgenden Übersicht vergeben wir den Namen "Editoren", weisen aus der Liste die benötigten Rechte zu und speichern die neue Gruppe abschließend ab.

## Mitarbeiter hinzufügen

In der Übersicht gehen wir hierauf auf Benutzer,

Und weisen dem jeweiligen Mitarbeiter unsere neue Editoren Rolle zu.

Loggt sich der Mitarbeiter nun mit seinem Account ein besitzt er über erweiterte Berechtigungen um Datensätze zu verwalten oder im Frontend neue User-Gruppen anzulegen.

Wir würden nun erwarten, dass der User im Frontend ebenfalls das Menü zum anlegen neuer Gruppen erhält.

Dem ist leider nicht so, hierbei handelt es sich um einen Bug über GeoNode4. Die neue Gruppe kann jedoch trotzdem bei direkt aufruf des links <https://geonode-training.csgis.de/groups/create/> oder über die Django Administrationsoberfläche erzeugt werden.



# Dataset verwalten

---

Im letzten Kapitel haben wir uns Benutzer und Gruppen näher angesehen. Gehen wir eine Stufe weiter und werfen einen Blick auf die Dataset Verwaltung.

Datasets hießen vor Version 3.3 Layer

Die Tabelle gibt uns wieder einen kurzen Überblick über nützliche Infos. Zum Beispiel wann der Datensatz erzeugt wurde, den Titel, oder die ID.

## Stapelverarbeitung

---

Die Option “dataset hinzufügen” sollte nicht verwendet werden. Neue Ebenen sollen über das Frontend hinzugefügt werden. Interessant ist jedoch die Möglichkeit mehrere Datasets auszuwählen und in Stapelverarbeitung Rechte zuzuweisen. Hierzu kann das untere Menü der Fußleiste verwendet werden.

## Dertailansicht

---

In der Detailansicht eines Datensatzes sehen wir dass zahlreiche Felder wie die Kurzbeschreibung einfach editiert werden können.

Dies entspricht der Möglichkeit aus dem Fortend die Metadaten des Datensatzes anzupassen.

Andere Felder ergeben sich rein aus der Tatsache dass der Django Admin “einen Spiegel” des Datenbanksatzes darstellt. Und hierdurch zahlreiche Felder, von den Administratoren (ohne tiefes Hintergrundwissen) nicht editiert werden sollten. Als Beispiel das CSW Feld:

# GeoNode Theming

---

GeoNode bietet standardmäßig einige Theming-Optionen, die direkt über die Administrationsoberfläche verwaltet werden können. In den meisten Fällen können Sie mit diesen Optionen das Aussehen von GeoNode ganz einfach ändern, ohne eine einzige Zeile HTML oder CSS zu verändern.

Im Django Admin öffnen wir die Theme Administration über den Link Themes:

Die hierauf folgende Ansicht listet alle vorhandenen Themes. Achten Sie auf den Status “is enabled”.

Über die bekannte Schaltfläche im Rechten oberen Eck “geonode theme customisation hinzufügen” legen wir ein neues Thema an.

Die Oberfläche gibt uns folgende Möglichkeiten:

- Name
  - Der Name des Themas (Erscheint nirgendwo)
- Description
  - Eine Beschreibung (Erscheint nirgendwo)
- Is enabled
  - Setzt das Thema aktiv
- Logo
  - Einbinden eines eigenen Logos
- Custom CSS rules
  - Erlaubt das hinzufügen eigener CSS Regeln zum Styling
- Jumbotron Background
  - Erlaubt das hinzufügen einer Grafik im großen Banner Bereich
- Hide Text in jumbotron
  - Angehakt wird der Text im Banner der Startseite unterdrückt
- Welcome theme
  - Slideshow oder Jumbotron background
- Jumbotron Slideshow
  - Falls Slideshow gewählt ist, welche Slides sollendn erscheinen.
- Jumbotron Title
  - Der Titel im Banner der Startseite
- Jumbotron content
  - Der im Banner angezeigte Text

Tipp verwenden Sie den [Theme generator](#) um die Custom CSS rules zu erzeugen und hierdurch die komplette Farbgebung zu beeinflussen.

zeigt sich auf der Startseite

## Weiterführende Links

---

- [GeoNode Docs Theming](#)



# Eigene Menüpunkte erstellen

---

GeoNode bietet einige integrierte Funktionen, mit denen Sie das Menü in der oberen Leiste schnell und einfach anpassen können (siehe das Beispiel unten).

## Neues Menü anlegen

---

Die relevanten Punkte finden wir im Django Admin unter “Menu Items”, “Menu placeholders”, “Menus”.

Wir öffnen Menus und klicken in der darauf folgenden Maske auf “menu hinzufügen”. In der darauffolgenden Maske vergeben wir einen Titel für das Menü (wird nur intern verwendet). Die Position (Feld Placeholder), sowie die Reihenfolge falls mehrere Menus unter dem Placeholder erscheinen sollen.

Und speichern das Menü ab.

Placeholder bezeichnet einen Ort im Template andem das Menü ausgegeben wird. Standardmässig besitzt GeoNode 4 vordefinierte Positionen. Über das Template Tag `render_nav_menu 'CUSTOM_MENU'` könnten wir einen neuen Platzhalter an anderer Stelle definieren.

## Neuen Menüpunkt anlegen

---

Nachdem wir ein neues Menü definiert haben, legen wir in diesem Schritt zwei Neue Menüpunkte an:

- über diese Plattform
- Alle Vectordaten

In der Django Übersicht wählen wir nun den Menüpunkt “Menu Items”

## Erster Menüpunkt

Und legen uns in der darauffolgenden Maske über den bekannten rechten Buton “menu item hinzufügen” ein neues Menu Item an:

Die sich hierauf öffnende Maske bietet die Formularfelder:

- Title
  - Der Titel des Menüpunktes
- Menu
  - Das Menu indem der Butteon erscheinen sollen (wir haben vorhin eine Menü mit dem Namen “Unsere Forschungsdaten erstellt”)
- Order
  - Die Reihenfolge falls mehrere Menüpunkte erscheinen
- Blank Target
  - Gesetzt öffnet der Link in einem neuen Browser-Tab
- Url
  - die URL auf die der Link verweisen soll

Für unseren ersten Menüpunkt füllen wir die Felder wie folgt

Feld	Wert
Title	about
Menu	Unsere Forschungsdaten
Order	1
Blank Target	nicht gesetzt
Url	<a href="https://geonode-training.csgis.de/about/">https://geonode-training.csgis.de/about/</a>

Hiernach klicken wir auf “Sicher und neu hinzufügen”.

## Zweiter Menüpunkt

Für unseren nächsten Datensatz öffnen wir ein neues Browserfenster. Navigieren zu den Datensätzen und setzen den Haken bei Vektor und kopieren uns die URL aus der Adresszeile:

Zurück in der Maske des neuen Menüpunkt füllen wir die Werte wie folgt:

Feld	Wert
Title	Unsere Vektoredaten
Menu	Unsere Forschungsdaten
Order	2
Blank Target	nicht gesetzt
Url	<a href="https://geonode-training.csgis.de/catalogue/#!/search/?f=dataset&amp;f=store-vector">https://geonode-training.csgis.de/catalogue/#!/search/?f=dataset&amp;f=store-vector</a>

Wir speichern und schließen den Menüpunkt über speichern und betrachten das Ergebnis im Frontend.

## Darstellung des Menüs

Unser neues Menü zeigt sich der oberen Navigationsleiste der Seite.

Ein Klick auf About bringt die User zur About Seite:

Wie Sie diese Seite überschreiben sehen Sie im nächsten Abschnitt “Einführung in die Entwicklung”.

Ein Klick auf auf “Unsere Vektordaten” zeigt eine vorab gefilterte Liste an Vektor Datensätzen.

Mit dieser Technik ließen sich natürlich auch Links speziell einer Gruppe oder in einer Region denken.

## Weiterführende Links

- [GeoNode Docs: Menus, Items und Placeholder](#)



# Metadaten Modell

---

Besondere Bedeutung kommt den Metadaten jedes Datensatzes hinzu.

dabei Standard-Metadatenformaten wie den Metadatenstandards [ISO 19115](#) zu.

Sobald der Upload abgeschlossen ist, kann der Nutzer die Metadaten der Ressource füllen. Dieser werden hiernach über die CSW (OGC Catalogue Service) Endpunkte und APIs verfügbar gemacht.

Benutzer können auch ein XML-Metadaten-Dokument (ISO-, FGDC- und Dublin Core-Format) hochladen, um wichtige GeoNode-Metadatenelemente automatisch zu füllen.

## Endpunkte für den Metadatenkatalog

---

- Die Rest Schnittstelle
  - <https://geonode-training.csgis.de/api/v2/datasets?format=json>
- Der CSW Endpunkt
  - <https://geonode-training.csgis.de/catalogue/csw?service=CSW&version=2.0.2&request=GetCapabilities>

Dies erlaubt anderen Applikationen auf die GeoNode Daten in maschinenlesbarer Form zuzugreifen. Als Metadaten Server kommt [pycsw](#) zum einsatz. Das manchmal in Dokumentation genannte [geonetwork](#) kann aktuell nicht als Backend genutzt werden.

Wie bereits bekannt, können User den Metadatensatz Ihrer Daten setzen. Dies erfolgt im Frontend auf Ebene Datensatz über “Metadaten bearbeiten”.

Administratoren können in beschränktem Umfang Einfluss auf die angegebenen Daten nehmen. Zwei einfache Beispiele:

## Lizenzen anpassen

---

In diesem Beispiel legen wir eine neue auswählbare Lizenz an. Im Django Admin navigieren wir über den Menüpunkt

zur Liste der Lizenzen wo wir auf “lizenz hinzufügen” klicken

Natürlich hätten wir hierüber auch die Möglichkeit bestehende zu verändern oder zu löschen.

In der folgenden Maske legen wir unsere Test Lizenz an und speichern diese ab:

Hiernach steht den Usern die neue Lizenz im Frontend zur Auswahl bereit:

## Metadata Topic Categories

---

Als zweite Änderung wollen wir eine bestehende “Metadata Topic Kategorie” entfernen sowie eine neue hinzufügen.

Im Django Admin wählen wir den Link “Metadata Topic Category”:

In der hierauf folgenden Maske scrollen wir etwas nach unten und klicken auf “society”.

Wir entfernen den Haken bei “is choice” und speichern den Datensatz ab:

Zurück in der Liste aller Topic Kategorien wählen wir über den Rechten oberen Button “Hinzufügen topic category” und füllen das Formular mit Beispiel Daten.

Beim betrachten der Metadatenmaske verschwindet “Society” aus der Auswahl. Weiterhin zeigt sich unsere neue Metadaten Kategorie “Soil data”.



# GeoNode Harvesting

Harvesting bezeichnet das maschinell unterstützte sammeln und speichern von Datensätzen anderer.

GeoNode ist in der Lage, Ressourcen-Metadaten von mehreren entfernten Diensten zu sammeln.

In diesem Bereich werfen wir den Blick auf die Konfiguriert um von entfernten Dienste, eine Liste relevanter Ressourcen abzurufen und hierdurch eine Kopie als lokale Ressourcen zu speichern. Neben einem einmaligen “harvesten” besteht weiterhin die Möglichkeit die Quelle in definierten Zeitabschnitten erneut zu besuchen.

Standardmässig unterstützt GeoNode das Harvesting von:

- Andere entfernte GeoNode-Instanzen;
- OGC WMS-Servern;
- ArcGIS REST-Dienste.

Weiterhin besteht die Möglichkeit eigene Harvester zu programmieren.

## Einmaliges Harvestern

Dieser Workflow wird meist manuell vom Benutzer ausgeführt.

### Manuellen Harvester anlegen

Klicken Sie im Django Admin auf den Link zur Harvester Übersicht.

Über den Button im rechten oberen Eck legen Sie einen neuen Harvester an.

Das sich hierauf öffnenden Menü zeigt folgende Konfigurationsmöglichkeiten:

- Name
  - Der Name des Harvester zur internen Bezeichnung
- remote\_url
  - Basis-URL des zu erfassenden Remote-Dienstes, z. B. <https://master.demo.geonode.org>
- scheduling\_enabled
  - Gibt an, ob das Harvesting periodisch vom Harvesting-Scheduler durchgeführt werden soll oder nicht.
- harvesting\_session\_update\_frequency
  - Wie oft (in Minuten) sollen neue Harvesting-Sitzungen automatisch geplant werden?
- refresh\_harvestable\_resources\_update\_frequency
  - Wie oft (in Minuten) sollen neue Aktualisierungssitzungen automatisch eingeplant werden?
- Check availability frequency
  - Wie oft (in Minuten) soll die Verfügbarkeit des externen Diensts überprüft werden?
- default\_owner
  - Welcher GeoNode-Benutzer soll zum Eigentümer der Ressourcen ernannt werden?
- harvest\_new\_resources\_by\_default
  - Sollen neue Remote-Ressourcen automatisch geharvestet werden?
- delete\_orphan\_resources\_automatically
  - Verwaiste Ressourcen sind Ressourcen, die zuvor durch eine Harvesting-Operation erstellt wurden, die GeoNode aber nicht mehr auf dem zu harvestenden Remote-Dienst finden kann. Sollen diese Ressourcen automatisch aus GeoNode gelöscht werden? Dies gilt auch, wenn eine Harvesterkonfiguration gelöscht wird. In diesem Fall werden alle Ressourcen, die von diesem Harvester stammen, als verwaist betrachtet.

- harvester\_type
  - Typ des Harvester-Workers, der für das Harvesting verwendet wird.
- Harvester type specific configuration
  - Spezifische Konfiguration für den gewählten Harvester Typ. Konfigurationsobjekt wird als JSON erwartet.

Für unseren Test verwenden wir folgende Konfiguration:

Konfiguration	Wert
Name	Manueller Harvester
Remote URL	https://master.demo.geonode.org
Scheduling enabled	Nein
Harvesting session update frequency	bleibt ohne Auswirkungen da nur einmalig geharvested wird
Refresh harvestable resources update frequency	bleibt ohne Auswirkungen da nur einmalig geharvested wird
Check availability frequency	1
Check availability frequency	admin
Harvester type	geonode.harvesting.harvesters.geonodeharvester.GeonodeUnifiedHarvesterWorker
Harvester type specific configuration	{“resource_title_filter”: “tl_2018_02_anrc0”}

Nach dem speichern des Objekts wird der neue Harvester gelistet.

Wir sehen der Status zeicht “ready” und unsere Konfiguration um nur Ebenen mit dem Titel “tl\_2018\_02\_anrc0” zu harvesten wurde gesetzt.

Wir setzen den Haken vor dem Havester und wählen im unteren Menü die Option “Update harvestable resources for selected harvesters”

Zurück im Django Hauptmenü rufen wir die Liste der “Harvestable resources” auf.

Und sehen die geharvestete Resource. Um diese lokal zu speichern setzen wir am linken Rand der Tabelle den Haken, und wählen im unteren Auswahlfeld “Harvest selected resources”.

Nach einiger Zeit und neu laden der Maske sollte uns ein grünes Symbol über das erfolgreiche speichern des Datensatzes angezeigt werden.

Unser neuer Datensatz ist hiernach im GeoNode Portal sichtbar:

# Periodisches harvesten

---

Das periodisches abholen und speichern entfernter Datensätze funktioniert fast identisch zum manuellen Workflow. Um die Funktion zu illustrieren, werden wir den vorab angelegten Harvester leicht abändern.

## löschen der bestehenden Ebene

Im Django Admin navigieren wir zur [Verwaltung von Datensätzen](#) (diesmal im Django Admin). Und löschen die vorab hinzugefügte Ebene "geonode:tl\_2018\_02\_anrc0".

Die hierauf folgende Rückfrage zum löschen, bestätigen wir mit "Ja, ich bin sicher".

In der [Übersicht der Harvestable resources](#), löschen wir den vorab gefundenen Datensatz:

Zurück in der [Übersicht aller Harvester](#) editieren wir den vorab angelegten Harvester "Manueller Harvester".

In der Maske ändern wir folgende Angaben:

Konfiguration	Wert
Name	Automatischer Harvester
Scheduling enable	Checkbox angehakt
Harvesting session update frequency	2
Refresh harvestable resources update frequency	2
Harvest new resources by default	Checkbox angehakt
Delete orphan resources automatically	Checkbox angehakt

Und speichern die Konfiguration ab.

Nach einiger Zeit sollte die geharvestete Resource in der [Übersicht der zu harvesten Ressourcen](#) auftauchen.

Sowie die Ebene im Portal sichtbar werden.

## Weiterführende Links

---

- [GeoNode Docs: Harvesting](#)

# Ankündigungen (announcements)

---

Als Administrator müssen Sie von Zeit zu Zeit Ankündigungen über Ihr Portal veröffentlichen.

GeoNode “announcements” ermöglichen genau das; ein Administrator hat die Möglichkeit, drei Arten von Nachrichten zu erstellen, entsprechend ihrem Schweregrad, ihre Gültigkeit in Bezug auf den Zeitraum zu bestimmen (Start- und Ablaufdatum der Ankündigung), wer sie sehen kann oder nicht (jeder oder nur die registrierten Mitglieder) und wann ein Benutzer die Nachricht ausblenden.

Es gibt drei Arten von Bekanntmachungen, die nach ihrem Schweregrad unterschieden werden: Allgemein, Warnung und Kritisch. Der Unterschied liegt in der Farbe der Ankündigungsbox.

Nur Administratoren und Mitarbeiter können Ankündigungen erstellen und verwalten.

Derzeit gibt es zwei Möglichkeiten, auf die Liste der Ankündigungen zuzugreifen und sie zu verwalten:

## Über die GeoNode-GUI, aus dem Profil-Panel

---

## Über den Django Admin

---

Die Funktionalitäten sind für beide Oberflächen fast gleich, außer dass es über das Admin-Panel möglich ist, auch die “Ablehnungen” (dismissal) zu verwalten.

“Ablehnungen” sind Aufzeichnungen von Mitgliedern, die die Ankündigung gelesen und das Nachrichtenfeld geschlossen haben. Eine Ankündigung kann eine der drei folgenden “Ablehnungsgründen” haben:

- Keine Ablehnungen erlaubt:
  - Das Mitteilungsfeld der Ankündigung kann nicht geschlossen werden.
- “Session Only Dismissal (\*)\_”,
  - die Standardeinstellung, ermöglicht es, die Nachrichtenbox der Ankündigung für die aktuelle Browser-Sitzung zu schließen. Sie wird beim nächsten Zugriff wieder angezeigt.
- Dauerhafte Ablehnungen Erlaubt:
  - die Nachrichtenbox erscheint für das aktuelle Mitglied nicht mehr, sobald sie geschlossen wurde.

# Announcement im Frontend über Profil anlegen

---

Klicken Sie im Profilbereich auf den Link Ankündigungen

Klicken Sie entweder auf “Neue Ankündigung”, um eine neue Bekanntmachung zu erstellen, oder auf den Titel einer bestehenden Ankündigung, um deren Inhalt zu verwalten.

Füllen Sie hierauf alle Felder des Formulars aus.

## Ankündigung über den Django Admin

---

Das Verwalten von Ankündigungen über das Admin-Panel ist im Grunde dasselbe; die Felder für das Formular sind identisch.

Der Zugriff auf die Ankündigungsoptionen über das Admin-Panel ermöglicht Ihnen auch die Verwaltung von "Ablehnungen". Über diese Oberfläche können Sie selektiv entscheiden, welche Mitglieder eine bestimmte Ablehnungen sehen können und welche nicht, oder Sie können sie zwingen, die Nachrichten erneut zu sehen, indem Sie die Ablehnungen entsprechend löschen.

# 1. Einführung in die Entwicklung mit GeoNode

---

## Inhalt

---

In diesem Bereich werden wir uns mit die ersten Schritten zur Entwicklung mit GeoNode ansehen. Leider ist die GeoNode Infrastruktur technsich alles andere als einfach aufgebaut.

Je nachdem an welchem Bereich Sie arbeiten wollen werden, Kenntnisse in folgenden Punkten benötigt:

Python3	Die Programmiersprache Python
CSS	Cascading Stylesheets – Auszeichnungssprache für HTML
Less	Ein “preprocessor” für CSS
HTML	Hyper Text Markup Language - Definiert die Elemente von Internetseiten
React	Ein Javascript Framework zur Erstellung von Frontends
Redux	Ein Komponente für React um den “state” zu verwalten
Mapstore2	Ein Karten Framework
RSJX	Eine Javascript Bibliothek für “reaktives Javascript”
Django	Ein Webframework basierend auf Python
Jinja2	Eine Templatesprache für Django
sql	Die von dem Datenbankserver verwendete Sprache um Abfragen zu tätigen
git	Eine Versionsverwaltung von Code
GIS	spezifisches Wissen bezüglich räumlicher Daten

# GeoNode Quelltext

---

Um im folgenden Änderungen oder Neuentwicklungen an unserer Instanz vorzunehmen werfen wir zunächst einen Blick auf das Code Repository von GeoNode. Dieses findet sich unter der URL:

<https://github.com/geonode/geonode>

Der eigentliche Programm-Code befindet sich hier im Ordner “geonode”.

Die sich hierunter befindenen Ordner, stehen für sogenannte “Django-Apps”. Wir erinnern uns, Django, das Web-Framework hinter Django ermöglicht die einzelnen Programmbereiche in Module aufzuteilen.

Dem/der aufmerksamen Zuhörer\*in wird nun auffallen, dass sich die einzelnen Ordner, im “Django Admin” des vorhergehenden Artikels wiederfinden. Grund hierfür ist, dass Django die Administrationsoberfläche automatisiert aus den einzelnen Apps erstellen kann.

Betrachten wir uns eine App wie beispielsweise die “people”-App, die die Benutzer des Portals regelt, sollen 3 Dateien mit besonderem Augenmerk beachtet werden.

- `models.py`
- `templates/*`
- `views.py`

Diese Struktur bezeichnet sich als MVT. Ein Programmiermuster dass die Grundlage aller Django-Apps darstellt.

## `models.py`

Hierbei stellt die *models.py* die jeweilige Datenbank Definition dar. Hierüber werden also alle benötigten Tabellen und Funktionen rund um die Datenbank definiert.

## `templates/*`

Im Ordner `templates`, befinden sich alle Dateien die für die Ein-/Ausgabe zuständig sind. Dies sind größtenteils HTML Dateien die über die Templatesprache “Jinja2” die Ausgabe im Browser besorgen.

## `views.py`

`views.py` stellt Funktionen, die als Bindeglied zwischen Datenbankdefinitionen (`models.py`) und Ein-/Ausgabe Ansichten (`templates/*`) stehen.

Beachtenswert ist ebenfalls noch die sich sogut wie in allen Apps befindliche Datei “`urls.py`”. Sie enthält die URL Definitionen Bereit um Browseranfragen an die richtige Stelle zu leiten.

# GeoNode Entwicklungsumgebung

---

Eine grundlegende Regel bei der Entwicklung mit GeoNode ist, den Code-Kern wenn möglich nie anzufassen. <https://github.com/GeoNode/geonode-project> gibt uns jedoch die Möglichkeit einzelne Bereiche in GeoNode zu überschreiben. Zudem kann das Projekt lokal gestartet werden um Änderungen direkt live verfolgen zu können oder automatisierte Tests ablaufen zu lassen.

Da es unter Windows zu vielerlei Problemen mit abhängigen Bibliotheken kommt, soll die lokale

Entwicklungsumgebung lediglich unter Linux oder OSX gestartet werden.

## Neues GeoNode Project erzeugen

```
git clone https://github.com/GeoNode/geonode-project.git -b master
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
mkvirtualenv --python=/usr/bin/python3 my_geonode
pip install Django==3.2.14

django-admin startproject --template=./geonode-project -e py,sh,md,rst,json,yml,ini,env,sample,pr
cd geonode_training_dev
```

Beachtenwert ist hierbei die Datei `src/requirements.txt` sie zeigt dass der GeoNode Kern Quelltext in Form eines Python Pakets als Abhängigkeit installiert wird.

```
git+https://github.com/GeoNode/geonode.git@master#egg=GeoNode
```

## Settings erzeugen

Das Herzstück der Konfiguration ist die Datei `src/settings.py` (wir werden Sie im letzten Abschnitt noch im Detail betrachten), die über Umgebungsvariablen in der Datei `.env` gespeist wird.

Wir erzeugen die Datei `.env` mit folgendem Befehl:

```
python create-envfile.py --geoserverpwd geoserver
```

## Datenbank anlegen

Anders als in der README des GeoNode-Project Repository müssen wir vorab noch zwei Postgres Datenbanken anlegen.

```
-- Geodatenbank anlegen
CREATE USER geonode_training_dev_data;
ALTER USER geonode_training_dev_data with encrypted password '< GEONODE_DATABASE_PASSWORD aus .ov
CREATE DATABASE geonode_training_dev_data;
GRANT ALL PRIVILEGES ON DATABASE geonode_training_dev_data TO geonode_training_dev_data;

-- Django Datenbank anlegen
CREATE USER geonode_training_dev;
ALTER USER geonode_training_dev with encrypted password '< GEONODE_GEODATABASE_PASSWORD aus .over
CREATE DATABASE geonode_training_dev;
GRANT ALL PRIVILEGES ON DATABASE geonode_training_dev TO geonode_training_dev;

-- in beiden Datenbanken Postgis installieren
CREATE EXTENSION IF NOT EXISTS postgis;
GRANT ALL ON geometry_columns TO PUBLIC;
GRANT ALL ON spatial_ref_sys TO PUBLIC;
```

## Abhängigkeiten installieren

Ebenfalls ist es erforderlich noch Abhängigkeiten zu installieren sowie einige Management Dateien zu generieren:



```
cd src
pip install -r requirements.txt --upgrade
pip install -e . --upgrade

# Install GDAL Utilities for Python
pip install pygdal=="`gdal-config --version`.*"

# Dev scripts
mv ../.override_dev_env.sample ../.override_dev_env
mv manage_dev.sh.sample manage_dev.sh
mv paver_dev.sh.sample paver_dev.sh

source ../.override_dev_env
```

## Starten des Servers

Mit folgenden Befehlen sollten wir abschließend den lokalen Server starten können.

```
# Using the Default Settings
sh ./paver_dev.sh reset
sh ./paver_dev.sh setup
sh ./paver_dev.sh sync
sh ./paver_dev.sh start
```

Wenn alles korrekt verlaufen ist in der Terminal Ausgabe folgende Meldung sehen:

Weiterhin das Projekt im Browser unter <http://localhost:8000> aufrufen können.

## Weiterführende Links

---

- [GeoNode Docs – GeoNode Project](#)
- [GeoNode Docs – Docker als Entwicklungsumgebung](#)

# Templates überschreiben

In diesem sehr einfachen Beispiel werden wir die HTML Ausgabe eines Django Templates überschreiben.

Unter der Seite `/about` (z.B.: <https://geonode-training.csgis.de/about/>) publiziert GeoNode eine Seite die über Sinn und Zweck der Plattform informiert.

Diese Seite generiert sich wie folgt:

## URL Defintion

Die Datei `url.py` verlinkt alle Anfragen an `about` über die durch Django bereitgestellte Funktion `TemplateView`, direkt mit der statischen HTML Datei `about.html` im Templates Ordner

```
url(r'^about/$',  
    TemplateView.as_view(template_name='about.html'),  
    name='about'),
```

[Code auf GitHub](#)

## Neue about.html anlegen

In Geonode-Projekt können wir diese Datei nun einfach überschreiben indem wir im Ordner “Templates” eine Datei mit identischem Namen anlegen.

*templates/about.html*

```
{% extends "geonode_base.html" %}  
{% load i18n %}  
{% block title %}Unsere Forschungsdaten{% endblock %}  
  
{% block body_outer %}  
    <div class="page-header">  
        <h2>Unsere Forschungsdaten</h2>  
    </div>  
    <p>  
        Diese Plattform dient der Publikation von Daten zum Thema Weltraum.</p>  
    <p>  
        Lorem ipsum dolor sit amet, consectetur adipisicing elit. Inventore similique asperiores i  
    </p>  
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Fugit sint, labore sequi autem i  
{% endblock %}
```

Da der Entwicklungsserver nach jeder Änderung automatisch neu startet sollten die Änderungen hiernach direkt im Browser sichtbar werden:

In einem Live System sind folgende Schritte nötig:

Entweder:

- neuen Build des Django Image erstellen (`docker-compose build django`; `docker-compose up -d django`)
- oder host Verzeichnis in Docker Container mounten: <https://github.com/GeoNode/geonode-project/blob/master/docker-compose.yml#L11> und den Container neu starten (`docker-compose up -d django`).

Tipp: Sie können nicht nur Templates sondern so auch die Ausgabe der E-Mails überschreiben.

## Weiterführende Links

---

- [Das Django Template System](#)
- [GeoNode Docs – Templates überschreiben](#)
- [GeoNode Docs – Homepage anpassen](#)
- [GeoNode Docs – Frontend Development](#)

# Eigene Seite anlegen

Im vorherigen Abschnitt haben wir die bestehende Seite “about.html” überschrieben. In diesem Kapitel wollen wir eine neue Seite anlegen.

Hierfür werden wir eine neue HTML Datei erstellen sowie die `urls.py` verwenden um die neue Seite zu verlinken.

## dsgvo.html anlegen

Zunächst legen wir wiederum im Templates Ordner unsere neue HTML Datei `dsgvo.html` an.

```
{% extends "geonode_base.html" %}
{% load i18n %}
{% block title %}DSGVO{% endblock %}

{% block body_outer %}
    <div class="page-header">
        <h2>{% trans "Our privacy policy" %}</h2>
    </div>
    <p>
        Lorem ipsum dolor sit amet, consectetur adipisicing elit. Inventore similique asperiores i
    </p>
    <p>
        Lorem ipsum dolor sit amet consectetur adipisicing elit. Fugit sint, labore sequi autem in
    </p>
{% endblock %}
```

## url.py bearbeiten

In einem zweiten Schritt verlinken wir eine URL Anfrage an `dsgvo` mit unserer neu angelegten Datei:

```
from geonode.urls import urlpatterns
from django.views.generic import TemplateView
from django.conf.urls import url

# You can register your own urlpatterns here
urlpatterns = [
    url(r'^dsgvo/$',
        TemplateView.as_view(template_name='dsgvo.html'),
        name='dsgvo'),
] + urlpatterns
```

Hiernach sollte die neue DSGVO Seite im Browser sichtbar werden.

Tipp: in Kombination mit einem “Custom Menu” lassen sich so einfach eigene Seiten erzeugen.

## Weiterführende links

- [Django Docs - URLs definieren](#)

# Übersetzung anlegen

In diesem Abschnitt wollen wir einen Blick auf das Internationalisierungssystem von Django werfen.

Ziel ist unsere vorab als zu übersetzend ausgezeichnete Überschrift

```
<h1>{% trans "Our privacy policy" %}</h1>
```

mit einer Übersetzung zu versehen. Django erlaubt das anlegen von Übersetzungen über sogenannte `po` und `mo` Dateien.

Die `po` Datei stellt hierbei die Quelldatei. Die `mo` Datei die kompilierte Datei mit Übersetzungen dar.

Zunächst können wir den Umfang der Sprachen in unserer `settings.py` Datei wie folgt einschränken

```
LANGUAGES = (  
    ('en-us', 'English'),  
    ('de-de', 'Deutsch'),  
)
```

## Locale Dateien anlegen

Im Hauptverzeichnis unseres GeoNode-Projekts legen wir den neuen Ordner `locale` an.

```
cd .. # wir verlassen den Ordner src  
(e) tonischoenbuchner@Tonis-MacBook-Pro ~/dev/geonode4_project/geonode_training_dev/src % mkdir locale
```

Über das durch Django bereitgestellte Kommando `makemessages` lassen wir alle zur Übersetzung angelegten Textfragmente in `po` Dateien einsammeln:

```
django-admin makemessages --no-location -l en -l de -d django -e "html"
```

`--locale` gibt hierbei die zu erzeugenden Übersetzungen an. `-e "html"` gibt die Extension der Dateien an die wir berücksichtigen wollen

```
(e) tonischoenbuchner@Tonis-MacBook-Pro % tree locale  
locale  
├── de  
│   ├── LC_MESSAGES  
│   │   └── django.po  
└── en  
    ├── LC_MESSAGES  
    │   └── django.po  
  
4 directories, 2 files
```

Hiernach sollten wir die neuen Ordner je Sprache sowie die Quelldateien im Ordner `locale` vorfinden.

## po Dateien bearbeiten

Im nächsten Schritt legen wir in `locale/de/LC_MESSAGES/django.po` eine Übersetzung für unsere Überschrift an:

```
...
msgid "Our privacy policy"
msgstr "Unsere Datenschutzgrundverordnung"
...
```

## po Dateien in mo Dateien kompilieren

Nachdem wir die Quelldateien übersetzt haben können wir Django anweisen die Dateien zu kompilieren.

```
django-admin compilemessages --locale de --locale en
processing file django.po in /Users/tonischoenbuchner/dev/github/gn-auth3/geonode_training/locale/
processing file django.po in /Users/tonischoenbuchner/dev/github/gn-auth3/ils/geonode_training/de/
```

Hiernach sollten wir in unserem locale Ordnern die neu kompilierten \*.mo Dateien sehen.

```
locale
├── de
│   ├── LC_MESSAGES
│   │   ├── django.mo
│   │   └── django.po
└── en
    ├── LC_MESSAGES
    │   ├── django.mo
    │   └── django.po
```

Um die Änderungen online zu stellen kopieren wir den locale Ordner eine Ebene tiefer ins src Verzeichnis.

Das Projektverzeichnis erhält beim erstellen von GeoNode-Project den von Ihnen angegebenen Namen. Es handelt sich hierbei um dieses Verzeichnis: [https://github.com/GeoNode/geonode-project/tree/master/src/project\\_name](https://github.com/GeoNode/geonode-project/tree/master/src/project_name)

```
cp -r locale src/geonode_training_dev
```

Wir starten den Development Server neu:

```
paver stop
paver start
```

Hiernach sollten die Übersetzungen der Sprachen im Browser angezeigt werden.

Vergessen Sie in live Umgebungen nicht den Python Server neu zu starten

## Weiterführende Links

- [Django Docs – Internationalization and localization](#)

# Export geopackage APP

In diesem Abschnitt wollen wir uns die Grundzüge einer eigenen App ansehen.

Unsere APP wird Usern die Möglichkeit geben Vektor-Datensätze auszuwählen und diese gesammelt in einem geopackage herunter zu laden.

Die Hauptarbeit wird hierbei von dem im Hintergrund agierenden Kartenserver `geoserver` erledigt. Dieser stellt über den [WPS Endpunkt](#) bereits die Möglichkeit, Ebenen als Geopackage zu exportieren. Als Input erwartet er ein XML Dokument mit den Layer Definitionen.

ACHTUNG: Der gezeigte Code ist nicht produktionsfähig!  
Er vermisst wichtige Sicherheitsabfragen und Fehlerbehandlungen.

Programm Ablauf:

```
-> User wählt Datensätze zum Export über Checkboxes
-> JS sendet Namen der Ebenen an JSON View als POST request
-> Django sendet request an GeoServer WPS Endpunkt um das geopackage erstellen zu lassen
-> JS nimmt link zu Geopackage Download entgegen und gibt diesen an User aus
```

## Eine neue Django APP erstellen

Django macht uns den Start für eigene Entwicklungen sehr leicht. Das Kommando

```
python manage.py startapp geopackage_collection
```

erzeugt uns einen neuen Ordner innerhalb unserer Verzeichnisstruktur mit den wichtigsten Dateien wie `views.py`, `models.py` etc. bereits angelegt.

```
geopackage_collection
├── __init__.py
├── __pycache__
│   ├── __init__.cpython-39.pyc
│   ├── admin.cpython-39.pyc
│   ├── apps.cpython-39.pyc
│   ├── helper.cpython-39.pyc
│   ├── models.cpython-39.pyc
│   └── views.cpython-39.pyc
├── admin.py
├── apps.py
├── helper.py
├── migrations
│   ├── __init__.py
│   └── __pycache__
│       └── __init__.cpython-39.pyc
├── models.py
├── templates
│   └── base
│       └── resourcebase_list.html
├── tests.py
└── views.py
```

5 directories, 16 files

Unsere neue App müssen wir lediglich noch in den Django settings vermerken:

```

if PROJECT_NAME not in INSTALLED_APPS:
    INSTALLED_APPS += (PROJECT_NAME,)
    INSTALLED_APPS += ('geopackage_collection',)

```

## Anlegen der views.py

Das Bindeglied zwischen Datenbank und Ausgabe stellt die Datei `views.py` dar. Wir legen diese mit folgendem Inhalt an:

```

from django.views.generic.list import ListView
from django.http import JsonResponse
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.decorators.http import require_http_methods
from geonode.base.models import ResourceBase
from .helper import get_wfs_Template, http_client
import json

class GpkgList(LoginRequiredMixin, ListView):
    """Django List View um alle Datensätze auszugeben"""
    queryset = ResourceBase.objects.filter(subtype="vector")

@login_required
@require_http_methods(['POST'])
def gpkg_json_result(request):
    """View um Datensätze entgegen zu nehmen und
    die URL zum Datensatz zurück zu geben.
    """
    request_payload = json.loads(request.body.decode("utf-8"))
    datasets = request_payload['datasets']
    wfs_template = get_wfs_Template(datasets)
    wps_return = http_client(wfs_template)
    data = {"result_link": wps_return}

    return JsonResponse(data, safe=False)

```

## Anlegen der helpers.py

Die angelegte `views.py` greift auf einige Hilfsfunktionen zurück. So zum Beispiel eine Funktion die das XML für die Anfrage an GeoServer erstellt, oder einen einfach abstrahierten HTTP Clienten. Wir legen die Funktionen in einer neuen Datei namens `helpers.py` an.

```

from django.conf import settings
import requests
import logging

# Neuen Logger erzeugen
logger = logging.getLogger("geonode")

# Zugangsdaten zum GeoServer für http_client()
geoserver_user = settings.OGC_SERVER_DEFAULT_USER
geoserver_user_password = settings.OGC_SERVER_DEFAULT_PASSWORD
geoserver_settings_url = settings.GEOSERVER_WEB_UI_LOCATION
geoserver_url = f"{geoserver_settings_url}wps"

wfs_template = """<?xml version="1.0" encoding="UTF-8"?><wps:Execute version="1.0.0" service="WPS"
    <ows:Identifier>gs:GeoPackage</ows:Identifier>
    <wps>DataInputs>

```



```

        <wps:Input>
            <ows:Identifier>contents</ows:Identifier>
            <wps>Data>
                <wps:ComplexData mimeType="text/xml; subtype=geoserver/ge
                <![CDATA[
                    <geopackage xmlns="http://www.opengis.net/gpkg
                        {partials}
                    </geopackage>
                ]]>
                </wps:ComplexData>
            </wps>Data>
        </wps:Input>
    </wps>DataInputs>
    <wps:ResponseForm>
        <wps:RawDataOutput>
            <ows:Identifier>geopackage</ows:Identifier>
        </wps:RawDataOutput>
    </wps:ResponseForm>
</wps:Execute>
"""

```

```

def get_build_partials(datasets):
    """Erzeuge für jede zu ladende Ebene einen XML Knoten."""
    partials_source = '<features identifier="{dataset}" name="places">' \
        '<featuretype>{dataset}</featuretype>' \
        '</features>'

```

```

    partials_payload = ""
    for dataset in datasets:
        partials_payload += partials_source.format(dataset = dataset)

    return partials_payload

```

```

def get_wfs_Template(datasets):
    """Gibt das fertige XML Dokument an den View zurück."""
    generated_partials = get_build_partials(datasets)
    wfs_payload = wfs_template.format(partials = generated_partials)
    return wfs_payload

```

```

def http_client(geoserver_payload):
    """Hilfefunktion um HTTP POST request abzusetzen."""
    try:
        geoserver_response = requests.post(
            geoserver_url,
            data=geoserver_payload.encode('utf-8'),
            auth=(
                geoserver_user,
                geoserver_user_password
            ))

        if (geoserver_response.status_code < 200 or geoserver_response.status_code > 201):
            logger.error(geoserver_response.content)
        return geoserver_response.text
    except Exception as e:
        raise e

```

## Template anlegen

Abschließend erstellen wir im Verzeichniss unserer App ein neues Verzeichnis namens `templates`, hierin ein Verzeichnis mit dem Namen `base` und legen die Datei `resourcebase_list.html` mit folgendem Inhalt an:

```

{% extends "geonode_base.html" %}

{% block body_outer %}
<h2>GeoPackage erzeugen</h2>
<hr>
<table class="table">
  <thead>
    <tr>
      <th scope="col">Auswahl</th>
      <th scope="col">Vector Dataset</th>
    </tr>
  </thead>
  <tbody>
    {% for object in object_list %}
    <tr>
      <th scope="row"><input type="checkbox" name="layerselect" value="{{ object.alternate }}" />
      <td>{{ object.title }}</td>
    </tr>
    {% empty %}
      No objects yet.
    {% endfor %}
  </tbody>
</table>
<div id="result"></div>
<button id="submit">Generiere Geopackage von Auswahl</button>

<script>

/* Hilffunktion um korrekten Cookie zu denden */
function getCookie(name) {
  var cookieValue = null;
  if (document.cookie && document.cookie !== '') {
    var cookies = document.cookie.split(';');
    for (var i = 0; i < cookies.length; i++) {
      var cookie = cookies[i].trim();
      if (cookie.substring(0, name.length + 1) === (name + '=')) {
        cookieValue = decodeURIComponent(cookie.substring(name.length + 1));
        break;
      }
    }
  }
  return cookieValue;
}

const result_div = document.getElementById("result");
/* Auf Button klick auswahl senden und Resultat in Empfang nehmen */
var element = document.getElementById("submit")
  .onclick = function() {
    let node_values = [];
    var checkedBoxes = document.querySelectorAll('input[name=layerselect]:checked');
    checkedBoxes.forEach((node) => {
      node_values.push(node.value)
    });

    let data = JSON.stringify({"datasets": node_values});
    let csrftoken = getCookie('csrftoken');
    let response = fetch("/gpkg/gpkg_json_result/", {
      method: 'POST',
      body: data,
      headers: { 'Accept': 'application/json, text/plain, */*',
        'Content-Type': 'application/json',
        'X-CSRFToken': csrftoken},
    })
    .then(response => response.json())

```

```

        .then(res => {
            result_div.innerHTML = `

```

## urls.py

Abschließend verknüpfen wir unsere views.py mit unseren URL-Definitionen der Datei `urls.py`.

```

from geonode.urls import urlpatterns
from django.views.generic import TemplateView
from django.conf.urls import url

from geopackage_collection.views import GpkgList, gpkg_json_result
from django.urls import path

# You can register your own urlpatterns here
urlpatterns = [
    url(r'^gpkg/$',
        GpkgList.as_view(),
        name='gpkg'),
    url(r'^gpkg/gpkg_json_result/$',
        gpkg_json_result,
        name='gpkg_json_result'),
] + urlpatterns

```

## Ergebnis

Unter unserer neuen URL `http://localhost:8000/gpkg` sollten wir alle Vektor Datensätze aufgelistet bekommen.

Treffen wir eine Auswahl und klicken “Generiere Geopackage von Auswahl” sollte der Link zum Download des Geopackage nach einiger Zeit angezeigt werden:

Öffnen wir die heruntergeladene Datei in QGIS sehen wir die beiden exportieren Ebenen:

## Fazit

Wie wir gesehen haben, ist es relativ einfach auf vorhandene Datensätze zuzugreifen oder eigenen Seiten zu erstellen die Nutzereingaben entgegen nehmen. Das Thema Entwicklung unter GeoNode bleibt dennoch komplex, verlangt es dem Entwickler doch ein breites Wissen in unterschiedlichen Bereichen ab.

Selbstverständlich ließe sich dieses Beispiel noch weiter ausarbeiten. Zum Beispiel ein eigenes Datenbank Modell erstellen um Daten zu speichern und die Django Administrationsoberfläche zu verwenden.

# Weiterführende Links

---

- [Einführung in Django Apps](#)
- [Django Datenbank Abstraktion](#)
- [GeoServer WPS Endpunkt Dokumentation](#)

# Geändertes Frontend in GeoNode Version 4

---

In den vorausgehenden Kapiteln haben wir gesehen wie wir Templates überschreiben können. Vergleichen wir jedoch die Oberfläche der GeoNode Version 3.3 mit der Oberfläche der Version 4, stechen große Unterschiede ins Auge.

Der Vergleich zeigt dass die Oberfläche für die Version 4 weitreichende Änderungen erfahren hat.

## Django Templates durch React ersetzt

Da bereits die Kartenkomponente von GeoNode3 auf dem [React](#) basierte Framework [Mapstore2](#), basiert wurde entschieden weitere Komponenten (wie die Startseite) ebenfalls hierüber abwickeln zu lassen.

React ist eine JavaScript-Programmbibliothek[3] zur Erstellung von webbasierten Benutzeroberflächen.

Das bedeutet leider dass nicht mehr alle Templates, die vormalig durch Django ausgegeben wurden, wie gezeigt überschrieben werden können. Für diese Seitenbereiche ist es unter Umständen erforderlich die Mapstore2 Komponente zu “forken” und einen eigenen Build zu erstellen.

Aufgrund der Komplexität der Komponente ist die Entwicklung hier nur *sehr* erfahrenen Javascript Entwicklern anzuraten.

## Weiterführende Links

---

- [Github GeoNode Mapstore Client Readme](#)
- [Video Tutorial Developing with Mapstore](#)

# 1. System Administration

---

In diesem Abschnitt beschäftigen wir uns mit der eigentlichen Systemarchitektur.

Wir werden sehen welche Komponenten agieren und wie diese zusammenhängen.

Weiterhin einen kurzen Blick auf Backups und Command Line Kommandos und weitere Möglichkeiten für System Administratoren werfen.

User die in diesem Bereich arbeiten sollten ein fundiertes Wissen in

- Linux Administration
- Docker
- shell

besitzen.

# Die Komponentenden im Zusammenspiel.

Das was wir als GeoNode beschreiben ist eigentlich ein Zusammenspiel aus unterschiedlichen Open-Source Komponenten.

Die Aufgaben sind wie folgt:

- NGINX
  - Frontend Server, nimmt Anfragen entgegen und leitet dieses weiter (Proxy)
- Lets Encrypt
  - Stellt https Zertifikate bereit
- Django
  - Das Python Framework dass sich größtenteils um die "Business Logik" des Sytems kümmert. Also zum Beispiel die Datenhaltung, die API oder die URL Definitionen regelt.
- Jenkins
  - Erlaubt die Automatisierung von diversen Abläufen. Im Kontext von GeoNode wird ist optional für die Erstellung von Backups verwendet
- GeoServer
  - Ein auf den OGC Standards basierter Kartenserver. Dieser verarbeitet/liefert letztendlich die Geo-Daten aus.
- Celery
  - Eine python Komponente um Abläufe im Hintergrund (asynchron) auszuführen. Wird zum Beispiel für das erstellen von Thumbnails verwendet um so die User Oberfläche nicht zu blockieren.
- RabbitMQ
  - Speichert die Aufgaben (Tasks) für Celery
- Postgres
  - Ein relationeller Datenbank Server der über die POSTGIS Extension leistungsstarke GIS Funktionalitäten bereitstellt.

Betrachten wir die laufenden Container einer Docker Umgebungm zeigen sich exakt diese Dienste als Container abstrahiert:

Docker ist eine Software zur Isolierung von Anwendungen mit Hilfe von Containervirtualisierung.

```
toni@localhost:~/geonode_training$ docker-compose ps
```

Name	Command	State
celery4geonode_training	/usr/src/geonode_training/ ...	Up 8000/tcp
db4geonode_training	docker-entrypoint.sh postg ...	Up (healthy) 127.0.0.1:5432->5432
django4geonode_training	/usr/src/geonode_training/ ...	Up (healthy) 8000/tcp
geoserver4geonode_training	/usr/local/tomcat/tmp/entr ...	Up (healthy) 0.0.0.0:8080->8080
gsconf4geonode_training	sleep infinity	Up (healthy)
jenkins4geonode_training	/sbin/tini -- /usr/local/b ...	Up 0.0.0.0:50000->50000
letsencrypt4geonode_training	./docker-entrypoint.sh /bi ...	Up
nginx4geonode_training	/docker-entrypoint.sh ngin ...	Up 0.0.0.0:443->443/tcp
rabbitmq4geonode_training	docker-entrypoint.sh rabbi ...	Up 25672/tcp, 4369/tcp

# Installationsarten

---

GeoNode kann auf unterschiedliche Weise in zwei verschiedenen Ausprägungen installiert werden.

## Ausprägungen

---

1. Es wird der “Geonode-Core” installiert
  - i. Wir haben den Quellcode bereits in vorausgehenden Kapiteln gesehen
2. Es wird “GeoNode Project” installiert
  - i. “GeoNode Project” ist ein “Wrapper” um den GeoNode Core der erlaubt Dateien zu überschreiben ohne die Core Daten anzufassen. Auch wenn sich hierdurch die Komplexität erhöht ist es die erste Wahl sobald es um Anpassungen oder eigene Entwicklungen geht.
3. SPC-Geonode
  - i. taucht verzeinzelt noch in der Dokumentation auf. Obsolete Geonode Konfiguration.

## Installationswege

---

Für beide Ausprägungen sind alle Installationswege möglich.

1. *Docker (Epfehlender Installationspfad)*
  - i. Die aktuell zu empfehlende Umgebung. Die einzelnen Komponenten werden automatisiert korrekt in Container abstrahiert und konfiguriert.
2. *Manuelle Installation*
  - i. Bei diesem Installationsweg installiert der System Administrator alle Komponenten einzeln “von Hand”. Dies erfolgt auf einer Ubuntu oder Debian Distribution. (Vereinzelt sind auch Installationen auf Red Hat erfolgt). Aufgrund von fehlenden Abhängigkeiten ist eine Installation auf Windows Servern nicht zu installieren.
3. *Ansible*
  - i. Ist eine Automatisierungssoftware die die manuelle Installation in sogenannten “Playbooks” automatisiert abläuft. Im GeoNode Project Verzeichnis befindet sich ein [Beispiel](#) Playbook. Verwensgundsstatus: Unklar.
4. *Vagrant*
  - i. Ist eine Automatisierung um Virtuelle Maschinen zu konfigurieren. Eine Konfigurationsdatei findet sich [hier](#). Sie wird verwendet um schnell eine virtuelle Testinstanz mit Docker zu erstellen.

## Weiterführende Links

---

•	
<a href="https://docs.geonode.org">[docs.geonode.org]</a>	Installationsanleitungen ( <a href="https://docs.geonode.org/en/master/install/advanced/index.html">https://docs.geonode.org/en/master/install/advanced/index.html</a> )



# Die Django Settings

Die vielleicht wesentlichste Datei bezüglich Einstellungen von GeoNode ist die Datei `settings.py`. Diese Einstellungsdatei ist allen Django Projekten genuin. Somit nichts GeoNode spezifisches.

Einen guten Einstiegspunkt liefert daher die [offizielle Django Dokumentation](#).

## Spezifisches für GeoNode

Im Haupt-Repository von GeoNode (core) befindet sich die Datei unter folgendem Pfad:

- <https://github.com/GeoNode/geonode/blob/master/geonode/settings.py>

In GeoNode-Projekt unter:

- [https://github.com/GeoNode/geonode-project/blob/master/src/project\\_name/settings.py](https://github.com/GeoNode/geonode-project/blob/master/src/project_name/settings.py)

Hier zeigt sich eine Besonderheit in Zeile 31ff.:

```
# Load more settings from a file called local_settings.py if it exists
try:
    from geonode_training.local_settings import *
#    from geonode.local_settings import *
except ImportError:
    from geonode.settings import *
```

Zu Beginn der `settings.py` wird versucht eine Datei Namens `local_settings.py` zu laden. Sie soll erlauben GeoNode mit eigenen Einstellungen zu erweitern bzw. bestehende zu ergänzen.

Wie schon bei der Entwicklung angesprochen soll weiterhin tunlichst vermieden werden die Quelldateien abzuändern. Ein weitaus besserer Weg ist die Anpassung über Umgebungsvariablen.

## Die .env Datei

Betrachtet man alle Einstellungen der `settings.py` fällt auf das für jede versucht wird zunächst die Einstellung über eine Umgebungsvariable zu setzen.

Beispiel

```
LANGUAGE_CODE = os.getenv('LANGUAGE_CODE', "en")
```

Dies ermöglicht uns das komplette System über die Datei `.env` zu konfigurieren.

Ihre Variablen werden in Docker Umgebungen standardmässig beim starten als Umgebungsvariablen gesetzt.

## Beachtenswerte Einstellungsmöglichkeiten

Betrachten wir anhand der `.env` Datei einige wichtige Einstellungen die Sie kennen sollten. Diese sind jene die Sie bei nicht Verwendung des Scripts zu Installation setzen müssen.

- `COMPOSE_PROJECT_NAME={{project_name}}`
  - Wird für die Namen der Docker Container verwendet
- `DJANGO_SETTINGS_MODULE={{project_name}}.settings`
  - Definiert den Namen der zu ladenden Settings Datei
- `POSTGRES_PASSWORD={pgpwd}`

- Das password für den Datenbank Super User
- GEONODE\_DATABASE={{project\_name}}
  - Der Name der Django Datenbank
- GEONODE\_DATABASE\_PASSWORD={dbpwd}
  - Das Passwort des Django Datenbank Users
- GEONODE\_GEODATABASE={{project\_name}}\_data
  - Die Postgis Datenbank für Vector Daten
- GEONODE\_GEODATABASE\_PASSWORD={geodbpwd}
  - Das Passwort für die Geodatenbank
- DATABASE\_URL=postgis://{{project\_name}}:{dbpwd}@db:5432/{{project\_name}}
  - Die Verbindungsparameter zur Datenbank
- GEODATABASE\_URL=postgis://{{project\_name}}\_data:{geodbpwd}@db:5432/{{project\_name}}\_data
  - Die Verbindungsparameter zur Geodatenbank
- SITEURL={siteurl}/
  - Die IP oder Domain der Seite inklusive http(s). Wird in vielen Templates verwendet
- ALLOWED\_HOSTS=["'django'", "\*", '{hostname}']
  - Von den hier definierten Domains darf das Portal aufgerufen werden. Der \* steht für erlaube alle
- GEONODE\_LB\_HOST\_IP={hostname}
  - Die öffentliche Domain. Wird unter Docker für das Networking benötigt.
- PUBLIC\_PORT={public\_port}
  - Der öffentliche Port
- HTTP\_HOST={http\_host}
  - Die öffentliche Domain
- HTTPS\_HOST={https\_host}
  - Die öffentliche https Domain, falls https verwendet wird
- GEOSERVER\_WEB\_UI\_LOCATION={geoserver\_ui}/geoserver/
  - Die öffentliche URL des GeoServers
- GEOSERVER\_PUBLIC\_LOCATION={geoserver\_ui}/geoserver/
  - Die öffentliche URL des GeoServers
- GEOSERVER\_ADMIN\_PASSWORD={geoserverpwd}
- ADMIN\_PASSWORD={geonodepwd}
  - Das Passwort der bei Installation erstellten Admin Rolle
- ADMIN\_EMAIL={email}
  - Die Email des Admins
- DEFAULT\_FROM\_EMAIL='{email}'
  - Absender von System Emails
- OAUTH2\_CLIENT\_ID={clientid}
  - Die ID der Geoserver oauth2 app
- OAUTH2\_CLIENT\_SECRET={clientsecret}
  - Der Key der Geoserver oauth2 app
- DEBUG={debug}
  - Debug Modus an oder aus? (True oder False). In Production immer False!
- SECRET\_KEY="{secret\_key}"
  - Ein Schlüssel den Django zur Verschlüsselung von beispielsweise Formulardaten verwendet.

Eine vollständige Übersicht aller Variablen finden Sie [hier](#).

Tipp: Lesen Sie die Datei settings.py. Hier finden sich einige hilfreiche Kommentare der Entwickler.

## Weiterführende Links

- [Geonode Docs - Übersicht der Variablen](#)

# Management Kommandos

Management Kommandos sind Hilfsfunktionen für GeoNode-Wartungsaufgaben. Sie werden normalerweise von einer SSH/Bash-Shell auf dem Server ausgeführt, auf dem GeoNode läuft.

Ein Beispielaufruf sieht wie folgt aus:

```
python manage.py migrate_baseurl --help
usage: manage.py migrate_baseurl [-h] [-f] [--source-address SOURCE_ADDRESS] [--target-address TARGET_ADDRESS]
                                [--pythonpath PYTHONPATH] [--traceback] [--no-color] [--force-color]
                                [--skip-checks]

Migrate GeoNode VM Base URL

optional arguments:
  -h, --help            show this help message and exit
  -f, --force            Forces the execution without asking for confirmation.
  --source-address SOURCE_ADDRESS
                        Source Address (the one currently on DB e.g. http://192.168.1.23)
  --target-address TARGET_ADDRESS
                        Target Address (the one to be changed e.g. http://my-public.geonode.org)
  --version              show program's version number and exit
  -v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal output, 2=verbose output, 3=very verbose output
  --settings SETTINGS   The Python path to a settings module, e.g. "myproject.settings.main". If omitted, the
                        default settings module will be used.
  --pythonpath PYTHONPATH
                        A directory to add to the Python path, e.g. "/home/djangoprojects/myproject"
  --traceback            Raise on CommandError exceptions
  --no-color             Don't colorize the command output.
  --force-color          Force colorization of the command output.
  --skip-checks          Skip system checks.
```

Ein angefügtes `--help` gibt uns also die Hilfe zum jeweiligen Kommando an.

Um sicher zu gehen dass Django mit den korrekten Settings arbeitet wird dem Befehl `DJANGO_SETTINGS_MODULE={Projekt_name}.settings` vorangestellt. `Projekt_name` bezeichnet hierbei den Namen Ihres GeoNode Projekts den sie auch in den `settings.py` verwenden. In Docker Umgebungen kann hierauf verzichtet werden.

Tipp: Sehen Sie sich auch die Ausgabe von `python manage.py --help` an. Sie enthält weitere Befehle die direkt durch Django bereit gestellt werden.

## Übersicht über GeoNode spezifische Kommandos

### migrate\_baseurl

Mit dem Verwaltungsbefehl `migrate_baseurl` können Sie alle GeoNode-Links korrigieren. Dies ist zum Beispiel nötig wenn Sie den Domännennamen oder die IP-Adresse Ihres Portals ändern wollen.

Beispiel:

```
python manage.py migrate_baseurl --source-address=127.0.0.1 --target-address=example.org
```

## Berechtigungen, Metadaten, Legenden und Download-Links aktualisieren

## sync\_geonode\_datasets

Mit diesem Befehl können Sie bereits vorhandene Berechtigungen zwischen Django und dem im Hintergrund agierenden GeoServer synchronisieren. Dies ist zum Beispiel nötig wenn Datasets trotz der richtigen Berechtigungen nicht geladen werden können.

Beispiel: Ich möchte alle Berechtigungen und Attribute der Datasets mit dem GeoServer aktualisieren/synchronisieren

```
manage.py sync_geonode_datasets --updatepermissions --updateattributes
```

Beispiel 2: Ich möchte die Thumbnails aller Datensätze neu generieren, die dem User `toni` gehören

```
python manage.py sync_geonode_datasets -u toni --updatethumbnails
```

## sync\_geonode\_maps

Dieser Befehl ist im Grunde ähnlich wie der vorherige, wirkt sich aber mit einigen Einschränkungen auf die Karten aus.

Beispiel: Ich möchte das Thumbnail der Karte neu generieren Dies ist eine Testkarte

```
manage.py sync_geonode_maps --updatethumbnails -f 'This is a test Map'
```

## set\_all\_layers\_metadata

Dieser Befehl ermöglicht das Zurücksetzen der Metadatenattribute und des Katalogschemas von Datensätzen. Der Befehl aktualisiert auch die CSW Catalogue XML und Links von GeoNode.

Beispiel: Nachdem ich die Basis-URL geändert habe, möchte ich das gesamte Katalogschema neu generieren.

```
python manage.py set_all_layers_metadata -d
```

## set\_layers\_permissions

GeoNode bietet ein sehr nützliches management Kommando, mit dem ein Administrator auf einfache Weise Berechtigungen für Gruppen und Benutzer auf einem oder mehreren Datensätzen hinzufügen/entfernen kann.

Beispiel: Weisen Sie den Benutzern `user_A` und `user_B` und der Gruppe `group_C` Schreibrechte für die Datasets `layer_X` und Dataset `Y` zu.

```
manage.py set_layers-permissions -p write -u user_A user_B -g group_C -r layer_X 'Dataset Y'
```

## Daten in GeoNode laden

Es gibt Situationen, in denen es nicht möglich oder nicht sinnvoll ist, das Upload-Formular zu verwenden, um neue Datensätze über die Weboberfläche zu GeoNode hinzuzufügen. Zum Beispiel:

- Der Datensatz ist zu groß, um ihn über die Weboberfläche hochzuladen.
- Daten sollen von einem Speichermedium programmatisch importiert werden.
- Tabellen aus einer Datenbank sollen publiziert werden.

## importlayers

Lädt Dateien aus einem lokalen Verzeichnis, einschließlich Unterordnern, in eoNode. Die Datensätze werden zur Django-Datenbank, der GeoServer-Konfiguration und dem pycsw-Metadaten-Index hinzugefügt. Momentan werden nur Dateien vom Typ Esri Shapefile (.shp) und GeoTiff (.tif) unterstützt. Um den Import durchführen zu können, muss GeoNode in Betrieb sein.

Beispiel: Ordner mit shapes in GeoNode laden

```
manage.py importlayers /Users/tonischoenbuchner/Desktop/langesgrenzen_shp
```

## updatelayers

Während es wie gezeigt möglich ist, Datensätze direkt aus dem Dateisystem Ihres Servers in Ihren GeoNode zu importieren, haben Sie vielleicht einen bestehenden GeoServer, der bereits Daten/Ebenen enthält. Um diese in GeoNode zu publizieren steht das management Kommando `updatelayers` zur Verfügung

Beispiel: Importiere eine bestehende Ebene mit dem Namen `_1_SARMIENTO_ENERO_2018` von GeoServer in Geonode.

```
manage.py updatelayers -w geonode -f _1_SARMIENTO_ENERO_2018
```

## delete\_resources

Der `delete_resources` Management-Befehl erlaubt es, Ressourcen zu entfernen, die eine bestimmte Bedingung erfüllen, die in Form eines "Django Q()-Ausdrucks angegeben ist.

Beispiel: Lösche alle Ebenen die `Wasser` enthalten; dem User admin gehörten und dessen ID 1 oder 2 ist. Sowie alle hierzu gehörenden Karten.

```
manage.py delete_resources -l 'Q(pk__in: [1, 2]) | Q(title__icontains:"water")' 'Q(owner__name=admin)
```

## Weiterführende links

- [Geonode Dokumentation - Management Kommandos](#)

# Die Django shell

---

Neben den Management Kommandos des vorherigen Kapitels besitzen Administratoren mit Shell Zugang ein weiteres mächtiges Werkzeug. Die “Django Shell”.

Wir starten Sie mit

```
python ./manage.py shell
```

In dieser Python Sitzung haben wir Zugriff auf die GeoNode Datenbank und Ihre Datensätze.

## Beispiel: Ändern aller Titel bestehender Datensätze

---

```
In [1]: from geonode.base.models import ResourceBase
In [2]: all_datasets = ResourceBase.objects.all()
In [3]: for dataset in all_datasets:
...:     dataset.title += '_archive'
...:     dataset.save()
...:
```

Betrachten wir hiernach alle Titel, sehen wir dass die Datensätze bearbeitet wurden.

## Weiterführende Links

---

- [Django Dokumentation - ./manage.py](#)

# Contrib Apps

“Contrib Apps” sind Erweiterungen die sie GeoNode hinzufügen können. Diese finden Sie in folgendem Repository:

<https://github.com/GeoNode/geonode-contribs>

In Django Umgebungen sind die Contrib Apps bereits installiert. Auf zwei Erweiterungen soll hingewiesen werden.

## Ldap Erweiterung

Erlaubt die Authentifizierung von Usern über LDAP. Die Konfiguration finden sie bereits in der bekannten `settings.py`

```
# LDAP
LDAP_ENABLED=False
LDAP_SERVER_URL=ldap://<the_ldap_server>
LDAP_BIND_DN=uid=ldapinfo,cn=users,dc=ad,dc=example,dc=org
LDAP_BIND_PASSWORD=<something_secret>
LDAP_USER_SEARCH_DN=dc=ad,dc=example,dc=org
LDAP_USER_SEARCH_FILTERSTR=(amp(uid=%(user)s)(objectClass=person))
LDAP_GROUP_SEARCH_DN=cn=groups,dc=ad,dc=example,dc=org
LDAP_GROUP_SEARCH_FILTERSTR=(|(cn=abt1)(cn=abt2)(cn=abt3)(cn=abt4)(cn=abt5)(cn=abt6))
LDAP_GROUP_PROFILE_MEMBER_ATTR=uniqueMember
```

Siehe: <https://github.com/GeoNode/geonode-project/blob/master/.env.sample#L214-L223>

## keycloak-sync Erweiterung

Diese Contrib App erlaubt die Authentifizierung von Usern über einen eigenständigen [Keycloak Server](#). Keycloak als Authentifizierungsserver unterstützt wiederum eine Vielzahl an Authentifizierungsquellen.

Um das Modul zu konfigurieren müssen folgende Settings hinzugefügt werden.

```
if 'keycloaksync' not in INSTALLED_APPS:
    INSTALLED_APPS += ('keycloaksync',)

KEYCLOAK_URL=os.getenv('KEYCLOAK_URL', None)
KEYCLOAK_CLIENT=os.getenv('KEYCLOAK_CLIENT', None)
KEYCLOAK_CLIENT_ID=os.getenv('KEYCLOAK_CLIENT_ID', None)
KEYCLOAK_CLIENT_SECRET=os.getenv('KEYCLOAK_CLIENT_SECRET', None)
KEYCLOAK_REALM=os.getenv('KEYCLOAK_REALM', None)
KEYCLOAK_USER=os.getenv('KEYCLOAK_USER', None)
KEYCLOAK_PASSWORD=os.getenv('KEYCLOAK_PASSWORD', None)
KEYCLOAK_USER_REALM=os.getenv('KEYCLOAK_USER_REALM', None)
```

Beachten Sie die [Readme](#) des Repository.

## Weiterführende Links

- [GeoNode Docs – Contrib Apps](#)
- [LDAP Contrib App](#)



- [Keycloak Contrib App](#)

# Die GeoNode Datenbanken

GeoNode arbeitet mit zwei Datenbanken:

1. geonode
2. geonode\_data

Wobei “geonode” durch den jeweiligen Projektnamen ersetzt wird

- geonode\_training
- geonode\_training\_data

In der Datenbank 1. `geonode` werden alle Tabellen der einzelnen Django Apps abgelegt.

In der Datenbank 2. `geonode_data` werden alle Tabellen von Vektor Datenstätzen abgelegt. Auf diese Datenbank greift der GeoServer ebenfalls zu.

beide Datenbanken benötigen die `POSTGIS` Erweiterung!

## Datenbank Verbindung

Diese Anleitung zeigt den empfohlenen Weg sich mit den Datenbanken von GeoNode in einer Docker Umgebung zu finden.

Standardmässig ist von außen keine Verbindung zu der im Postres-Container laufenden Datenbankserver möglich. Wir ändern dies indem wir in unserer `docker-compose.yml` die Ports verlinken.

```
db:
  # use geonode official postgis 13 image
  image: geonode/postgis:13
  command: postgres -c "max_connections=${POSTGRES_MAX_CONNECTIONS}"
  container_name: db4${COMPOSE_PROJECT_NAME}
  env_file:
    - .env
  volumes:
    - dbdata:/var/lib/postgresql/data
    - dbbackups:/pg_backups
  restart: on-failure
  healthcheck:
    test: "pg_isready -d postgres -U postgres"
  # Hier die Verlinkung zwischen den Ports zwischen Host und Container
  ports:
    - "127.0.0.1:5432:5432"
```

Achtung: wir verwenden auf Host Seite nicht 5432 sondern `127.0.0.1:5432`. Bei Verwendung von 5432 würden wir den Port komplett nach außen hin öffnen was wir nicht wollen.

Um uns nun mit den Datenbanken mit einem lokalen Datenbank-Clients wie “Dbeaver” oder “pgadmin” verbinden zu können, müssen wir einen sogenannten SSH Tunnel herstellen.

```
ssh toni@geonode-training.csgis.de -L 5490:127.0.0.1:5432
```

Über die -L Angabe binden wir den Port 5490 an den Port 5432 des Servers. (In diesem Fall geonode-

training.csgis.de)

Eine letzte Überprüfung der Docker Container stellt sicher dass der Port für den Host sichtbar ist:

```
docker ps
```

Hiernach können wir uns mit unserem lokalen Datenbank-Clienten über den Tunnel verbinden:

- Wir verwenden den Tunnelport 5490
- Als Host wählen wir localhost
- Als Datenbank die gewünschte Datenbank: {projectname}\_geonode oder {projectname}\_geonode
  - Da wir hier den User postgres verwenden belasse ich die Datenbank auf postgres
- Als Benutzername und Passwort haben wir drei Möglichkeiten. Je nachdem mit wem wir und Verbinden, müssen wir ebenso die gewählte Datenbank Definition verändern.
  - Den User der {projectname}\_geonode Datenbank
  - Den User der {projectname}\_geonode
  - Den Postgres User

Hiernach können wir uns mit den Datenbanken verbinden / mit diesen arbeiten.

# Backups

GeoNode besitzt eine Mangement Kommando um alle erforderlichen Daten einer Instanz zu sichern sowie diese wieder einzuspielen.

Der Prozess lässt sich weiterhin wie in der [Dokumentation](#) über Jenkins automatisieren.

Wir raten von der Verwendung der Funktion ab, da Sie in der Vergangenheit in der User Community zu Problemen geführt hat.

## Backup Script

Eine alternative zur in GeoNode integrierten Backup-Funktionalität, stellt die Verwendung eines eigenen Backups Scripts dar. Dieses kann etnweder über einen Cronjob zu festgelegten Zeiten ausgeführt werden. Oder in Docker Umgebungen über einen eigenen Docker Service automatisiert werden.

### Einfaches Backup Script

Ein sehr einfaches Backups Script könnte wie folgt aussehen:

backup.sh

```
#!/bin/sh

NOW=$( date '+%d-%m-%y' )

# Create directories
BPTH=/backups/backup_${NOW}
mkdir -p ${BPTH}

DPTH=${BPTH}/databases
mkdir -p ${DPTH}

SPTH=${BPTH}/statics
mkdir -p ${SPTH}

GPTH=${BPTH}/geoserver-data-dir
mkdir -p ${GPTH}

# Datenbanken sichern. Beachte den erforderlichen Zugang zu
pg_dump -h db -U geonode -C -d geonode_training > ${DPTH}/${DATABASE}_daily.pgdump && echo "${DATABASE} backup successful"
pg_dump -h db -U geonode -C -d geonode_training > ${DPTH}/${DATABASE_GEO}_daily.pgdump && echo "${DATABASE_GEO} backup successful"

# Backup files
rclone copy /geonode_statics ${SPTH} --log-level ERROR && echo "geonode_statics copied successfully"
rclone copy /geoserver-data-dir ${GPTH} --log-level ERROR && echo "geoserver-data-dir copied successfully"

# create archive
tar cvfj /backups/bba-geonode.tar_${NOW}.bz2 ${BPTH} && rm -R ${BPTH}

# Delete old
find /backups -maxdepth 1 -mtime +${DAYS_TO_KEEP} -exec rm -rf {} \; && echo "Clean of /backups done"
```

Achten Sie auf die richtige Setzung der Pfade.

Die Daten die gesichert werden ist:

- Die Datenbanken

- Das Geoserver Daten Verzeichnis
- Die statischen Dateien von Django die Thumbnails und hochgeladene Dokumente besitzen

## Backup als Docker Service

---

Einen Beispiel Service dieses Scripts finden sie [hier](#).

## Weiterführende Links

---

- [GeoNode Docs - Backup und Restore](#)
- [Beispiel backup Service über Docker](#)

# Erste Hilfe bei Problemen

---

Die erste Handlung die Sie bei Problemen unternehmen, ist die Betrachtung der Log-Files. In einer Docker Umgebung setzen wir hierfür das Log-Level von GeoServer auf ein höheres als “production”.

Und betrachten hiernach die Log Ausgabe der Container

```
docker-compose logs -f
```

Um nur die Logs von einem Service zu bekommen vermerken wir diesen wie folgt:

```
docker-compose logs -f django
```

## Die häufigsten Fehler

---

### Upload schlägt fehl

---

Kontrollieren Sie das Admin Passwort des Geoservers in `.env` und setzen Sie dieses in der GeoServer Admin GUI erneut.

### Datensätze werden nicht angezeigt obwohl die Berechtigungen stimmen

---

GeoServer regelt die Regeln über “Geofence” Regeln. Um diese mit Django/GeoNode abzugleichen, verbindet sich GeoServer über oauth2 mit Django. Prüfen Sie die oauth2 Einstellungen der “geoserver app” im Django Admin (Abschnitt oauth2). Sowie die oauth2 Einstellungen auf Seiten von Geoserver.

Siehe: <https://docs.geonode.org/en/master/advanced/components/index.html?highlight=oauth2>

Weitere kurze Problembeschreibungen finden Sie in folgendem Wiki:

<https://github.com/GeoNode/geonode/wiki/Good-to-know>

## Orte um Hilfe zu finden

---

GeoNode ist ein Open-Source-Projekt deren Mitwirkende in unterschiedlichen Foren versammeln:

- Mailingliste für Benutzer: <https://lists.osgeo.org/cgi-bin/mailman/listinfo/geonode-users>
- Mailing-Liste für Entwickler: <https://lists.osgeo.org/cgi-bin/mailman/listinfo/geonode-devel>
- Gitter Chat: <https://gitter.im/GeoNode/general>

Wir empfehlen den Gitter Chat um nach Hilfe zu fragen. Erfahrungsgemäß können Fragen hier am schnellsten beantwortet werden.

Der [Issue Tracker auf Github](#) sollte nur für Bugs oder “Feature Requests” verwendet werden

## Weiterführende links

---

- [GeoNode Docs - oauth2](#)
- [Github Wiki – Good to know](#)

