



# QGIS Server Plugins

## QGIS Server Plugins

**Dr. Marco Hugentobler**  
**Sourcepole AG, Zürich**  
**[www.sourcepole.ch](http://www.sourcepole.ch)**  
**<https://twitter.com/sourcepole>**





# Warum ein Vortrag über Server Plugins?

- › **QGIS Server Plugins sind nicht neu (2015)**
- › **Und dokumentiert**
  - › [https://docs.qgis.org/3.34/en/docs/server\\_manual/plugins.html](https://docs.qgis.org/3.34/en/docs/server_manual/plugins.html)
  - › [https://docs.qgis.org/3.34/en/docs/pyqgis\\_developer\\_cookbook/server.html](https://docs.qgis.org/3.34/en/docs/pyqgis_developer_cookbook/server.html)
- › **Aber zu wenig bekannt (plugins.qgis.org):**
  - › Desktop-Plugins: 2011
  - › Server-Plugins: 11
- › **Erweiterung Server mit eigenem Code**
  - › Spezifische Anforderungen
  - › Kein Warten auf den nächsten Release



# Firmenpräsentation



Sie

Schreiben sie ein Loblied auf die Firma Sourcepole



Copilot

## Ein Loblied auf Sourcepole

In den Tiefen des Codes, wo Zeilen sich verweben, Dort thront Sourcepole, ein leuchtender Stern. Mit Fingerspitzengefühl und Wissen so tief, Erschaffen sie Lösungen, die uns alle beglücken.

Ihre Entwickler tanzen im Rhythmus der Algorithmen, Die Daten fließen wie ein reißender Strom. Mit Open-Source-Herzen und Leidenschaft für Geodaten, Bringen sie Karten zum Leben, als wären sie aus einem Traum.

QGIS, MapServer, PostGIS und mehr, Ihre Werkzeuge sind mächtig, ihre Vision klar. Sie webben die Welt, verbinden Orte und Menschen, Mit jedem Commit, mit jeder neuen Version.

Die Community jubelt, die Kunden sind begeistert, Sourcepole, du bist unser Kompass im digitalen Meer. Deine Expertise strahlt wie die Sonne am Horizont, Wir erheben unsere Tassen und singen dir ein Lied.

**Sourcepole, du bist der Nordstern unserer Projekte, Mit deinem Wissen und deiner Hingabe, die niemals versiegt. Möge deine Codebasis immer stabil sein, Und deine Entwickler weiterhin brillante Lösungen finden.**



# Unterschiede zu Desktop-Plugins

- › **metadata.txt**
  - › Server = True
- › **QgsServerInterface objekt**
  - › requestHandler() → QgsRequestHandler
  - › serviceRegistry() → QgsServiceRegistry
  - › registerFilter( QgsServerFilter )
  - › registerAccessControl( QgsAccessControlFilter )
  - › registerServerCache( QgsServerCacheFilter )

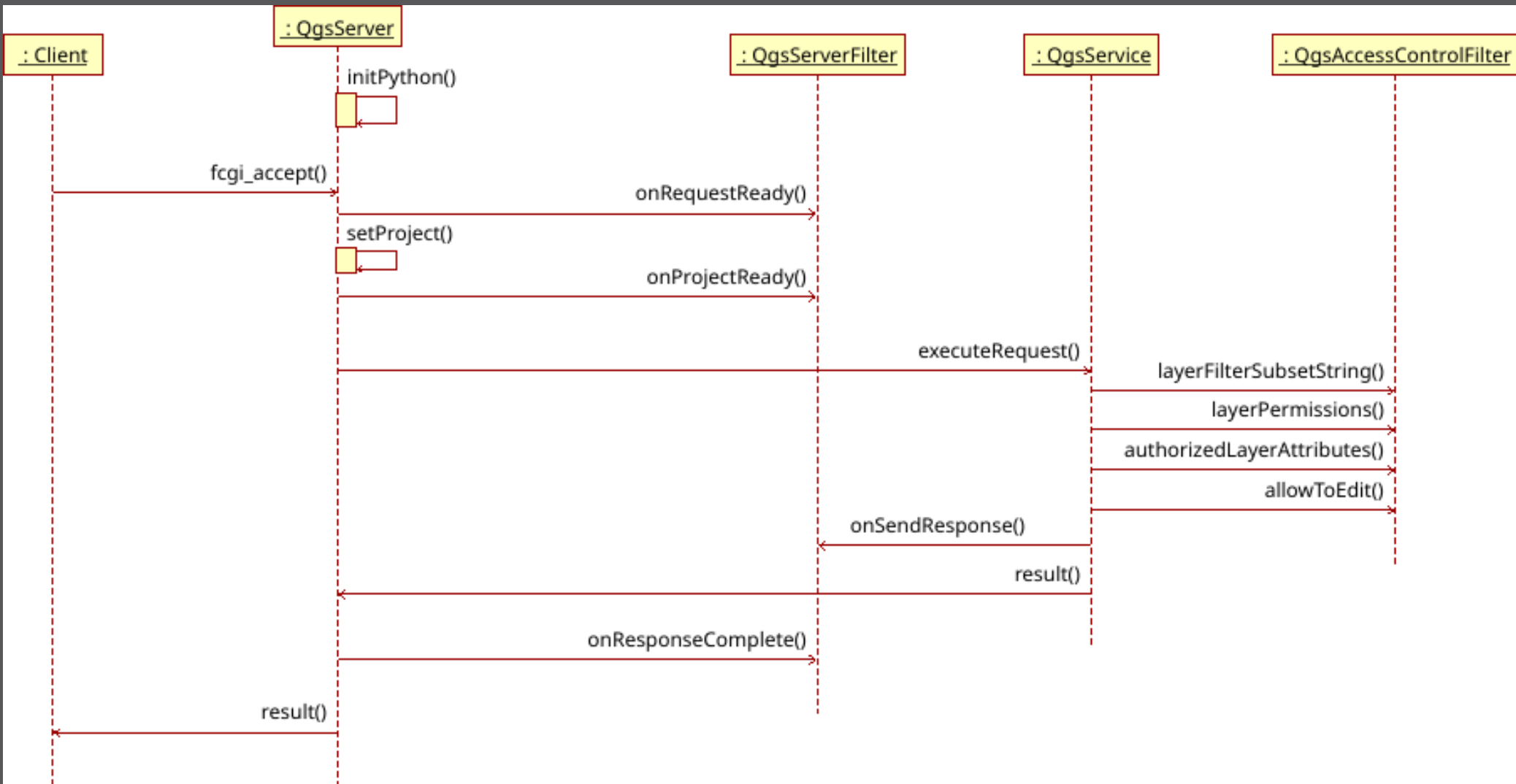


# Typen von Server-Plugins

- › **Filter Plugins**
  - › IO-Filter
  - › Access Control Filter
  - › Cache Filter
- › **Custom Service / API**
- › **Kann auch in einem Plugin gemischt sein**



# Einsprungpunkte für Python im Server





# Filter I/O Plugin

- › Häufigster Fall, Plugin modifiziert Input und/oder Output
- › Plugin leitet eigene Klasse von `QgsServerFilter` ab
- › Neue Filterklasse in Pluginkonstruktor in Filterregistry eintragen
- › Implementation von `onRequestReady()`, `onSendResponse()`, `onResponseComplete()`
- › Ab 3.38 kann auch `onProjectReady()` benutzt werden. Projekt ist dann bereits gesetzt und kann wie gewohnt über `QgsProject.instance()` verwendet werden



# Beispiele I/O-Filter Plugin

- › **Modifizierte GetFeatureInfo-Antwort**
- › **Zusätzliche Info in GetProjectSettings reinschreiben**
- › **GeoTIFF output für GetMap**
- › **Buffer um Ausschnitt in GetMap erzeugen und nachher wieder abschneiden (Labels / Punktsymbole)**
- › **Watermark auf Karten zeichnen**





# Access Control Filter Plugin

- › Unterklasse von `QgsAccessControlFilter` machen und im Plugin registrieren
- › Methoden:
  - › `layerPermissions( const QgsMapLayer *layer )` → read/update/insert/delete
  - › `authorizedLayerAttributes( const QgsVectorLayer *layer, const QStringList &attributes )` → `QStringList`
  - › `allowToEdit( const QgsVectorLayer *layer, const QgsFeature &feature )` → `bool`
  - › `layerFilterSubsetString( const QgsVectorLayer *layer )` → `QString`



# Cache Filter Plugin

- Cache-Operationen können im Plugin überschrieben werden
- Aufgerufen in:
  - WMTS GetTile
  - WMS GetLegendGraphic
  - WMS/WMTS/WFS/WCS GetCapabilities



# Neuer Service/API als Plugin

- › Technisch sind alle Services/APIs Plugins (WMS/WFS/WFS3/WCS)
- › Ableiten einer neuen Unterklasse von `QgsService` und Methode `executeRequest()` implementieren
- › In Pluginkonstruktor neue Serviceklasse in die `ServiceRegistry` eintragen
- › Und analog für API



# Beispiele Service-Plugins

- › WPS Plugin
- › Cache managen, z.B. Projektcache löschen
- › Infos, die in Capabilities nicht enthalten sind (z.B. Histogram Rasterlayer)



# Debuggen von Serverplugins

- **First-Aid Plugin kann leider nicht verwendet werden**
- **Basic: Mit QgsMessageLog ins Logfile schreiben**
- **Intermediate: Mit pdb, export QUERY\_STRING und Server von der Kommandozeile aufrufen**
  - **Nachteil: meistens nicht als Webserver-User**
  - **Nachteil: Quellcode muss modifiziert werden**
- **Advanced wäre Remote Debugging. Plugin müsste sich mit Server aus IDE verbinden**



**Vielen Dank für die  
Aufmerksamkeit**



**Marco Hugentobler**  
**marco.hugentobler@sourcepole.ch**