OGC API – Styles

Winter School 2022

Ms. Prajwalita J. Chavan
IIT, Bombay

Overview

- About OGC API – Styles
- Introduction
- Consumers
- Base resources
- Requirements classes ‘core’
- Declaration of conformance classes
- Fetch styles
- Fetch style metadata
- Requirements Class "Manage styles“
- Delete a style
- Requirements Class "Validation of styles"
- Requirements Class "Mapbox Style”
- Media Types

OGC API - Styles

- **Publication Date:** Not published yet
- OGC API Styles defines API building blocks for Web APIs to enable map servers and clients as well as **visual style editors to manage and fetch styles**
- Styles consist of **symbolizing instructions** that are applied by a rendering engine on features and/or coverages.
- The API complements the current and emerging OGC API specifications **for features, maps and tiles** and builds on the conceptual model for the encoding of **styles and their metadata**

Introduction

Resource	Path	HTTP method	Document reference
Base resource	{baseResource}	GET	Base resource
Conformance declaration	/conformance	GET	Declaration of conformance classes
Styles	{baseResource}/styles	GET	Fetch styles
		POST	Create a new style
			Validate a style
Style	{baseResource}/styles/{styleId}	GET	Fetch style
		PUT	Replace a style
			Validate a style
		DELETE	Delete a style
Style metadata	{baseResource}/styles/{styleId}/metadata	GET	Fetch style metadata
		PUT	Replace the metadata of a style
		PATCH	Update the metadata of a style

Consumers

A Styles API supports several types of consumers:

- **Visual style editors** that create, update and delete styles for datasets
  - OGC API – **Coverages**
  - OGC API - **Tiles**
  - OGC API – **Maps**
  - OGC API - **Features**
- **Map clients** that fetch styles and render spatial data

Consumers: Map clients

- **Wants to visualize data** for features or tiled feature data for the collection
- **Select one of the styles** from the list
- **Provide a capability** so that users can **change the style**
- Might also fetch a hillshade style to **apply to an elevation coverage**

Consumers: visual style editor creating **a new style**

- A user **creates the style** in the visual style editor
- **Selects the native stylesheet language** for the style
- **Identifies the collection** in the dataset
- **Visual style editor executes a request** to the landing page
- **Conformance declaration** of the data access

Consumers: A visual style editor **updating an existing style**

- The user will **start from an existing style**
- **open/load the style** from the style repository
- determine **changes to queryables**
- **existing style is replaced,** the style definition will always be updated with a PUT request

Consumers: Web API implementing OGC API - Maps

- **A Web API** that implements the conformance class "Map tile" of the OGC API Maps
- **The URI template** for the map tiles is retrieved
- If **a client requests** a map tile for the collection API

Base resources

Typical base resources are:

- The API landing page at path /.
    - If the **API provides distributions of a dataset**, then the styles will be associated with the dataset.

    - If the **API does not provide access to data**, it is a general purpose Styles API and the styles will typically be applicable to a range of data resources available elsewhere.

- A data collection at path **/collections/{collectionId}.**

Requirements Class "Core"

Requirement 1	/req/core/base-resource-link
A	The content of any base resource (at path {baseResource}) with which styles are associated in the API SHALL include a link to a Styles resource at path {baseResource}/styles (link relation type 'http://www.opengis.net/def/rel/ogc/1.0/styles').

- **POST** {baseResource}/styles
- **PUT** {baseResource}/styles/{styleId}

Landing page in JSON


```
{
  "links": [
    {
      "href": "https://example.org/api/v1",
      "rel": "self",
      "type": "application/json",
      "title": "this document"
    },
    {
      "href": "https://example.org/api/v1/api",
      "rel": "service-desc",
      "type": "application/vnd.oai.openapi+json;version=3.0",
      "title": "the API definition in OpenAPI JSON"
    },
    {
      "href": "https://example.org/api/v1/api.html",
      "rel": "service-doc",
      "type": "text/html",
      "title": "the API documentation in HTML"
    },
    {
      "href": "https://example.org/api/v1/conformance",
      "rel": "http://www.opengis.net/def/rel/ogc/1.0/conformance",
      "type": "application/json",
      "title": "list of conformance classes implemented by this API"
    },
    {
      "href": "https://example.org/api/v1/styles",
      "rel": "http://www.opengis.net/def/rel/ogc/1.0/styles",
      "type": "application/json",
      "title": "the styles shared via this API"
    }
  ]
}
```

Landing Page Response Document


```
{
  "title": "Buildings in Bonn",
  "description": "Access to data about buildings in the city of Bonn via a Web API
that conforms to the OGC API Features specification.",
  "links": [
    { "href": "http://data.example.org/",
      "rel": "self", "type": "application/json", "title": "this document" },
    { "href": "http://data.example.org/api",
      "rel": "service-desc", "type":
"application/vnd.oai.openapi+json;version=3.0", "title": "the API definition" },
    { "href": "http://data.example.org/api.html",
      "rel": "service-doc", "type": "text/html", "title": "the API documentation"
},
    { "href": "http://data.example.org/conformance",
      "rel": "conformance", "type": "application/json", "title": "OGC API
conformance classes implemented by this server" },
    { "href": "http://data.example.org/collections",
      "rel": "data", "type": "application/json", "title": "Information about the
feature collections" }
  ]
}
```

Declaration of conformance classes


```
{
  "conformsTo": [
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core",
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/req/json",
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/req/oas30",
    "http://www.opengis.net/spec/ogcapi-styles-1/1.0/conf/core",
    "http://www.opengis.net/spec/ogcapi-styles-1/1.0/conf/manage-styles",
    "http://www.opengis.net/spec/ogcapi-styles-1/1.0/conf/style-validation",
    "http://www.opengis.net/spec/ogcapi-styles-1/1.0/conf/resources",
    "http://www.opengis.net/spec/ogcapi-styles-1/1.0/conf/manage-resources",
    "http://www.opengis.net/spec/ogcapi-styles-1/1.0/conf/mapbox-styles",
    "http://www.opengis.net/spec/ogcapi-styles-1/1.0/conf/sld-10",
    "http://www.opengis.net/spec/ogcapi-styles-1/1.0/conf/sld-11"
  ]
}
```

Fetch styles

Requirement 2	/req/core/styles-op
A	The server SHALL support the HTTP GET operation at the path {baseResource}/styles.

Requirement 3	/req/core/styles-success
A	A successful execution of the operation SHALL be reported as a response with an HTTP status code 200.

Fetch styles

Requirement 4	/req/core/style-op
A	The server SHALL support the HTTP GET operation at the path `{baseResource}/styles/{styleId}` for each style referenced from the Styles resource at `{baseResource}/styles`.

Requirement 5	/req/core/style-success
A	A successful execution of the operation SHALL be reported as a response with an HTTP status code `200`.
B	The content of that response SHALL conform to the media type stated in the `Content-Type` header.
C	The language used in linguistic text in the response SHALL be consistent with the language stated in the `Content-Language` header.

Fetch style metadata

Requirement 6	/req/core/style-md-op
A	The server SHALL support the HTTP GET operation at the path `{baseResource}/styles/{styleId}/metadata` for each style metadata referenced from the Styles resource at `{baseResource}/styles`.

Requirement 7	/req/core/style-md-success
A	A successful execution of the operation SHALL be reported as a response with an HTTP status code `200`.

Requirements Class "Manage styles"

Requirement 8	/req/manage-styles/resources-endpoint
A	For styles, the resources endpoints to create a new style SHALL be URIs specified by the URI template `{baseResource}/styles`.
B	When a new style is created, a minimal style metadata resource SHALL be created at `{baseResource}/styles/{styleId}/metadata`.

Requirement 9	/req/manage-styles/default-style-update
Condition	Server implements OGC API - Features - Part 4: Create, Replace, Update and Delete, Requirements Class "Update"
Condition	Server advertises support for media type `application/merge-patch+json` in the API definition for PATCH requests at `{baseResource}/styles`
A	The server SHALL process PATCH requests with a content type `application/merge-patch+json` to such a resource endpoint as specified by RFC 7396 (JSON Merge Patch).

Delete a style

Requirement 11	/req/manage-styles/styles-delete
A	For requests to the style metadata (template `{baseResource}/styles/{styleId}/metadata`), the DELETE operation SHALL not be supported.
B	DELETE requests to a style (template `{baseResource}/styles/{styleId}`) SHALL delete the style metadata of that style, too.

Requirements Class "Validation of styles"

Requirement 14	/req/style-validation/input
A	The server SHALL support the `Prefer` header with the `handling=strict`.

B	The server SHALL support a parameter with the name "dry-run" in POST requests to the path `{baseResource}/styles` and in PUT requests to the path `{baseResource}/styles/{styleId}` with the following schema:
	`name: validate`
	`in: query`
	`required: false`
	`style: form`
	`explode: false`
	`schema:`
	`  type: boolean`
	`  default: false`

Requirements Class "Mapbox Style"

Requirement 22	/req/mapbox-style/media-type
A	Every POST or PUT operation of the server that accepts a stylesheet document as content SHALL support the media type `application/vnd.mapbox.style+json`.

The list of operations in a server implementing all conformance classes of this draft specification is:

- **POST** {baseResource}/styles
- **PUT** {baseResource}/styles/{styleId}

Media Types

- **application/json** is the JSON media type used for all content except the stylesheets and the symbol resources.

- **text/html** is the HTML media type for all "web pages" provided by the API.

Hands-on: Create Project

QGIS

QGIS is an open GIS desktop application that allows you to display, interrogate, visualise and create geospatial information. It is also capable of interacting with geo-centric APIs (for example, a WMTS).

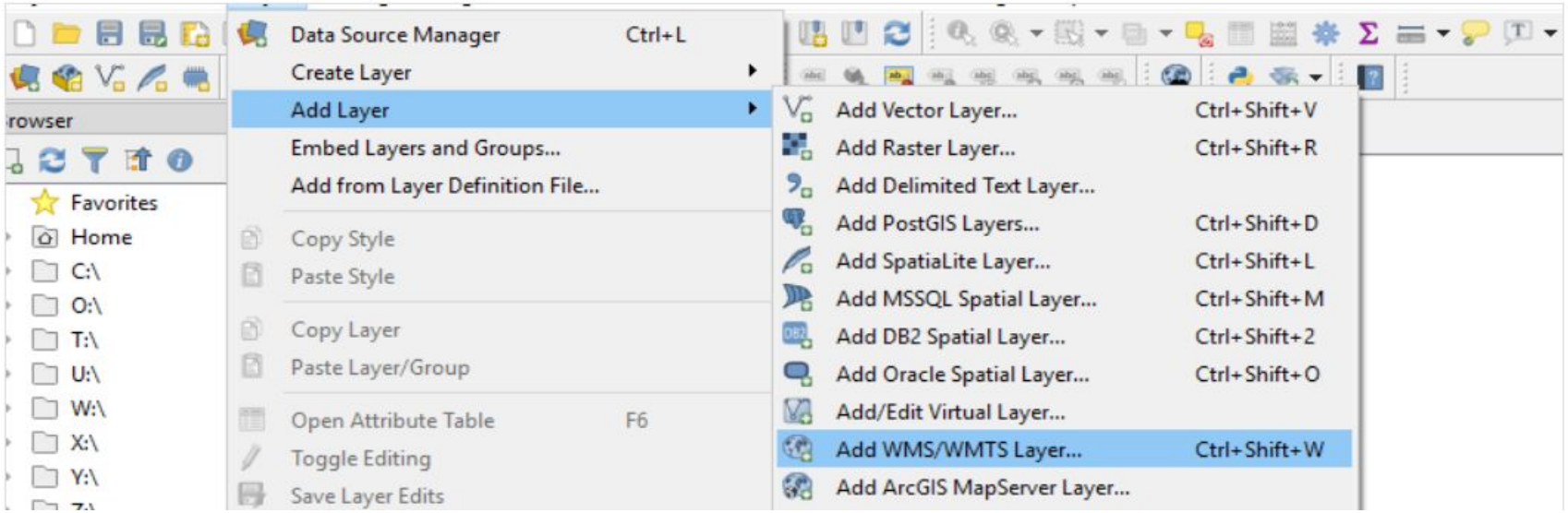
The instructions that follow demonstrate how to integrate the OS Maps API in order to produce a background map in QGIS.

For the purposes of this guide the version of QGIS used is 3.4.

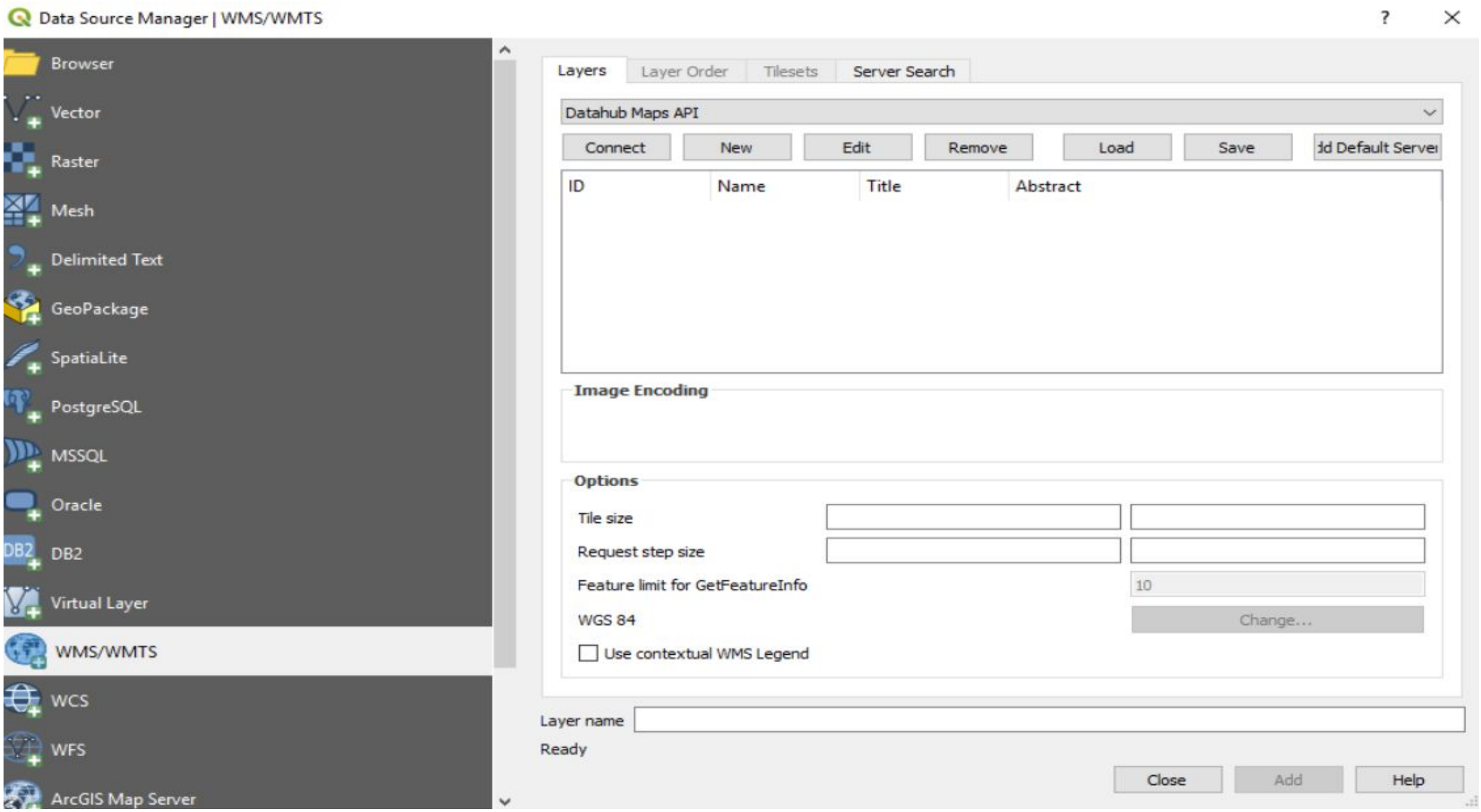
Integrating OS Maps API in QGIS

1. Open a blank document in QGIS.

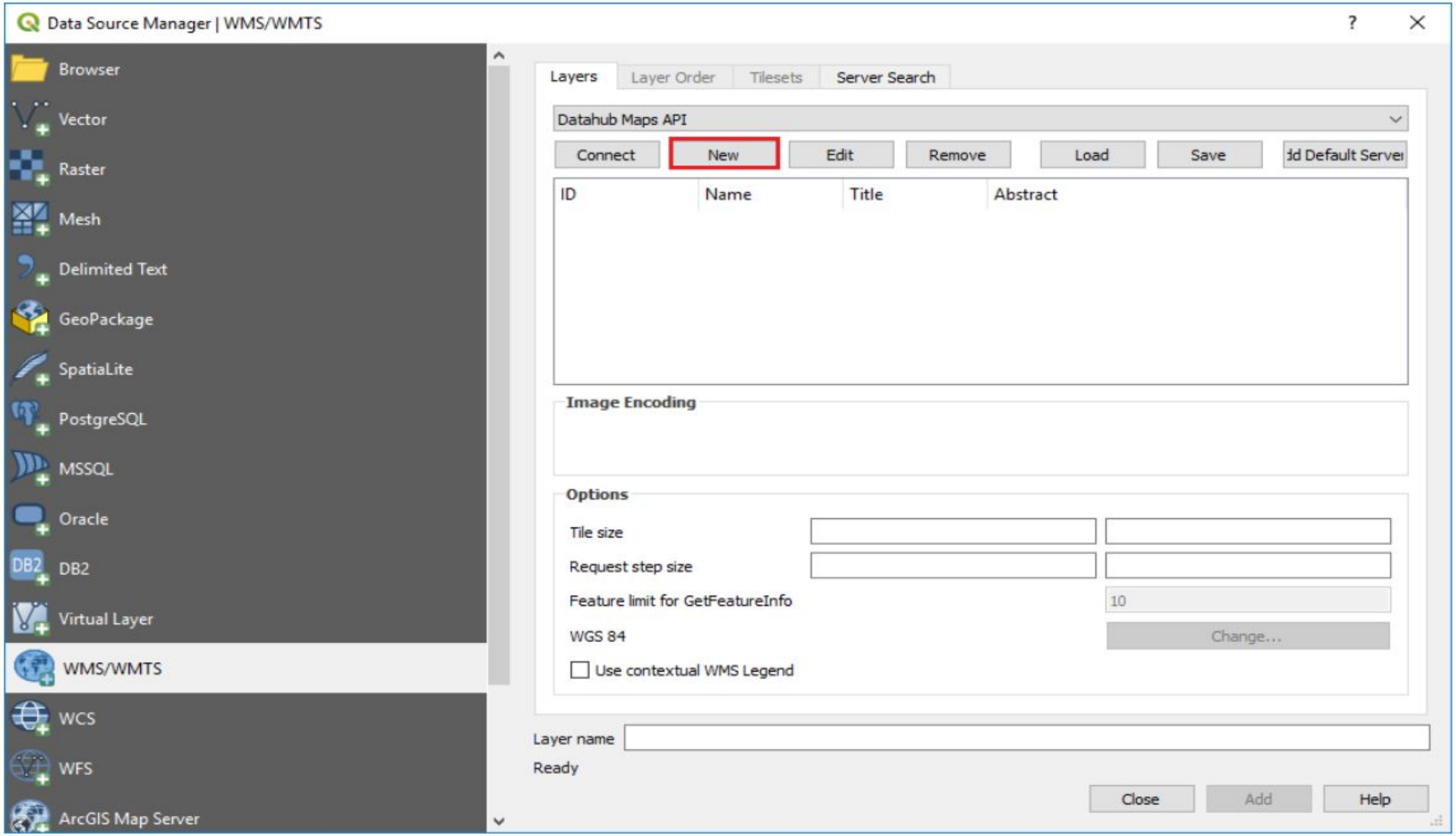
2. Navigate to Layer → Add Layer → Add WMS/WMTS Layer...



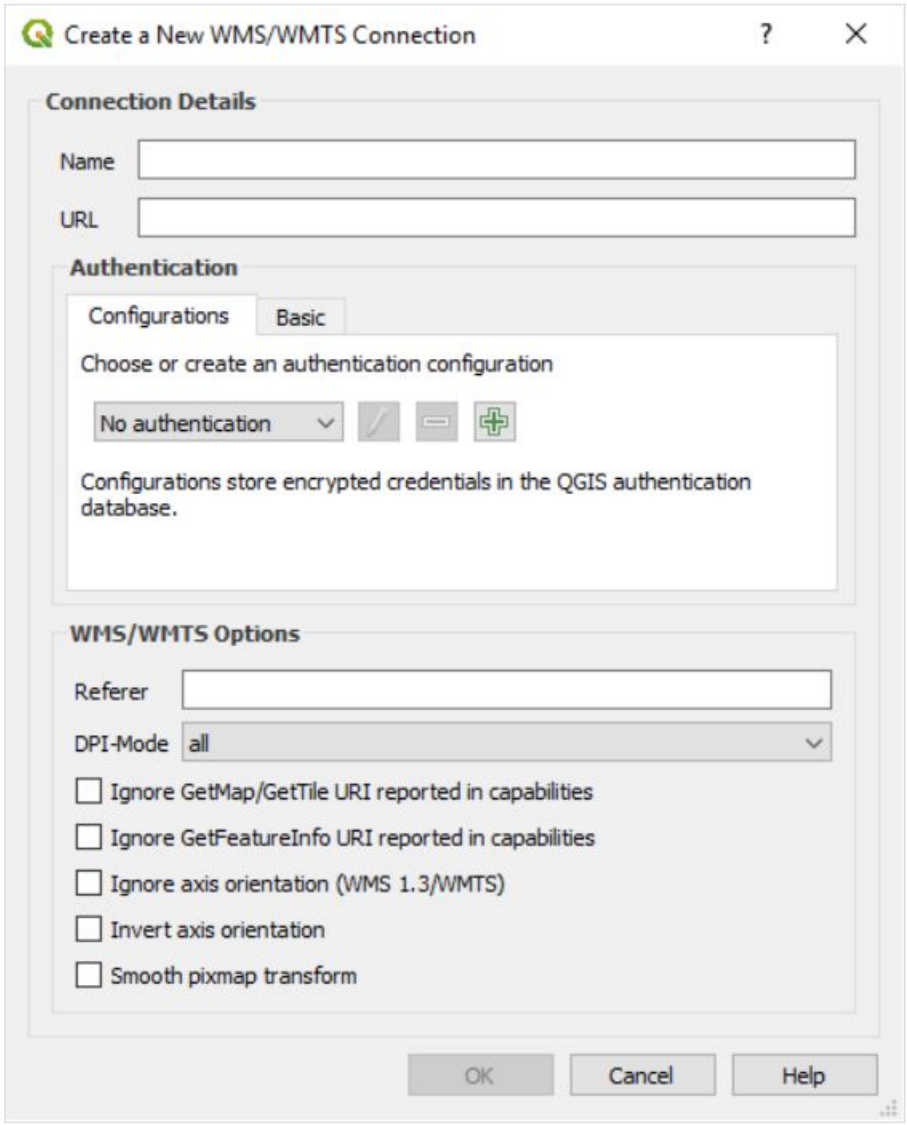
Hands-on: Create Project



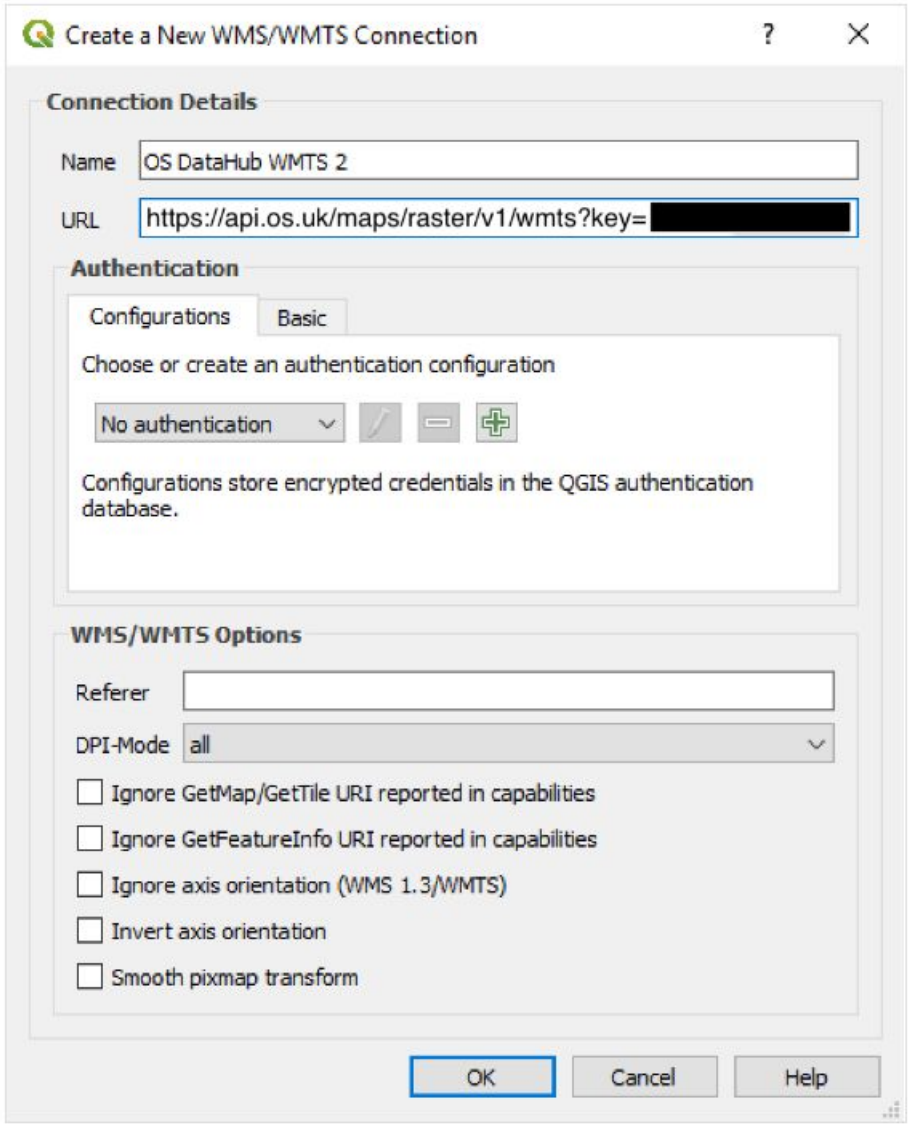
Hands-on: Create Project



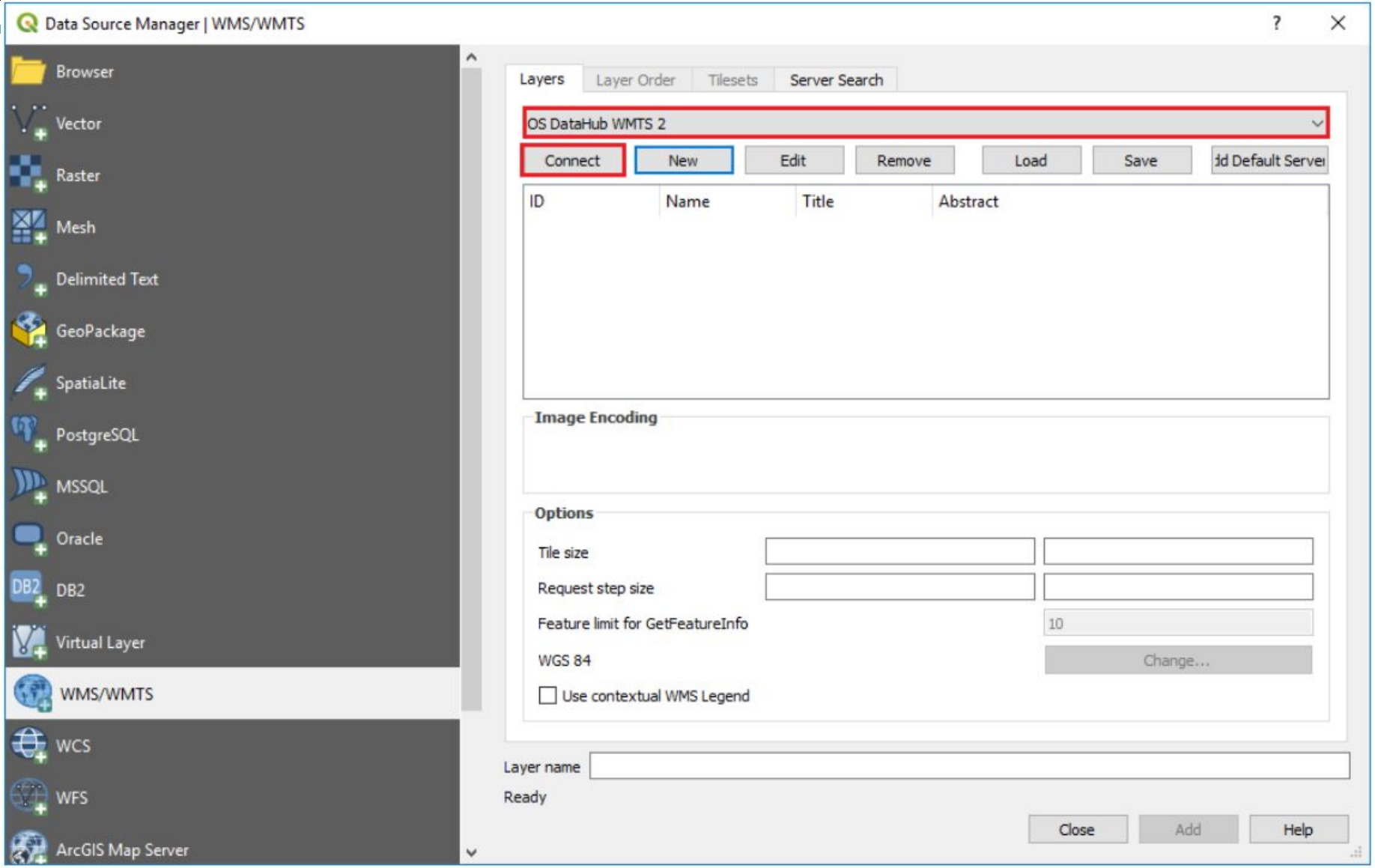
Hands-on: Create Project



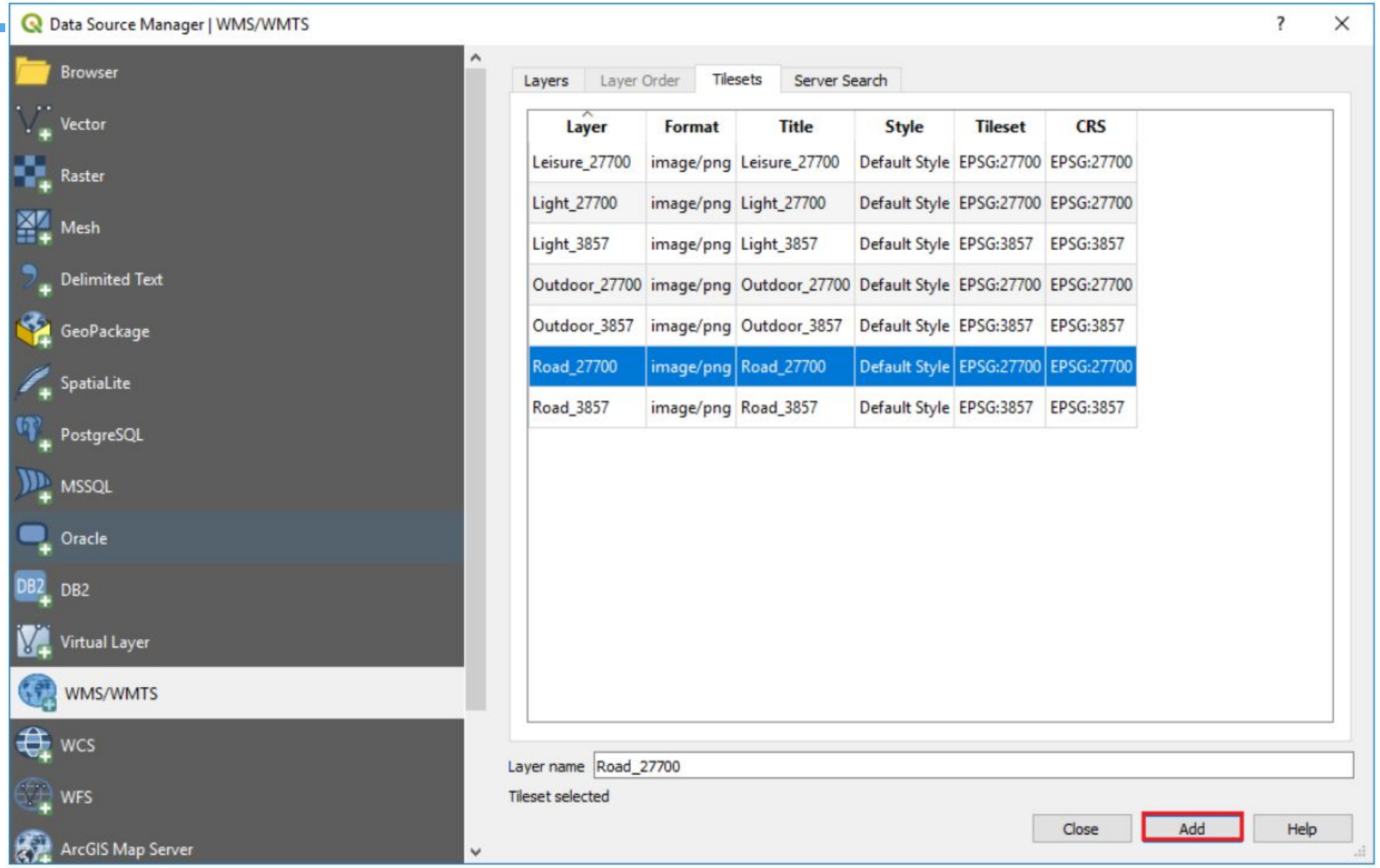
Hands-on: Create Project



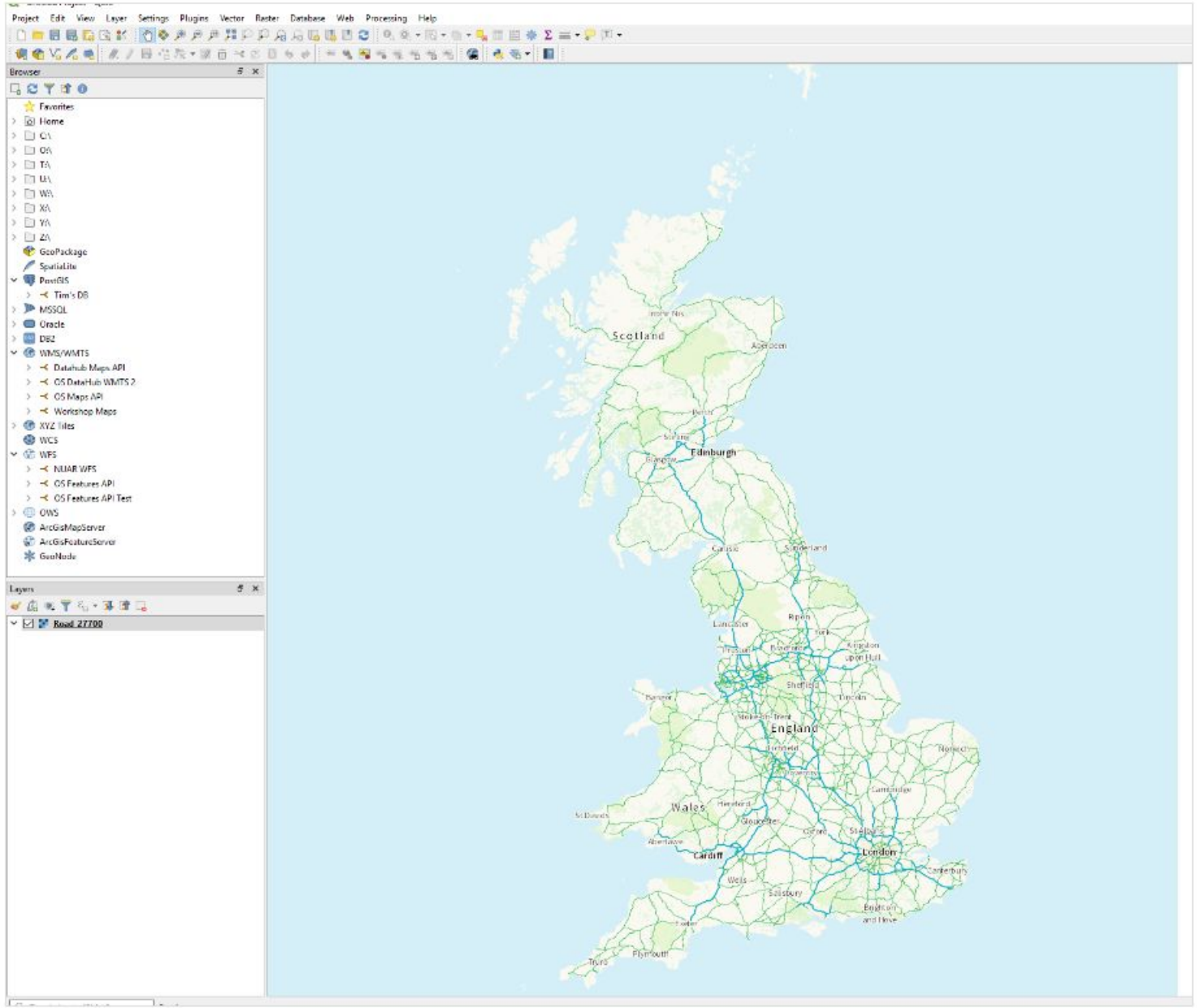
Hands-on: Create Project



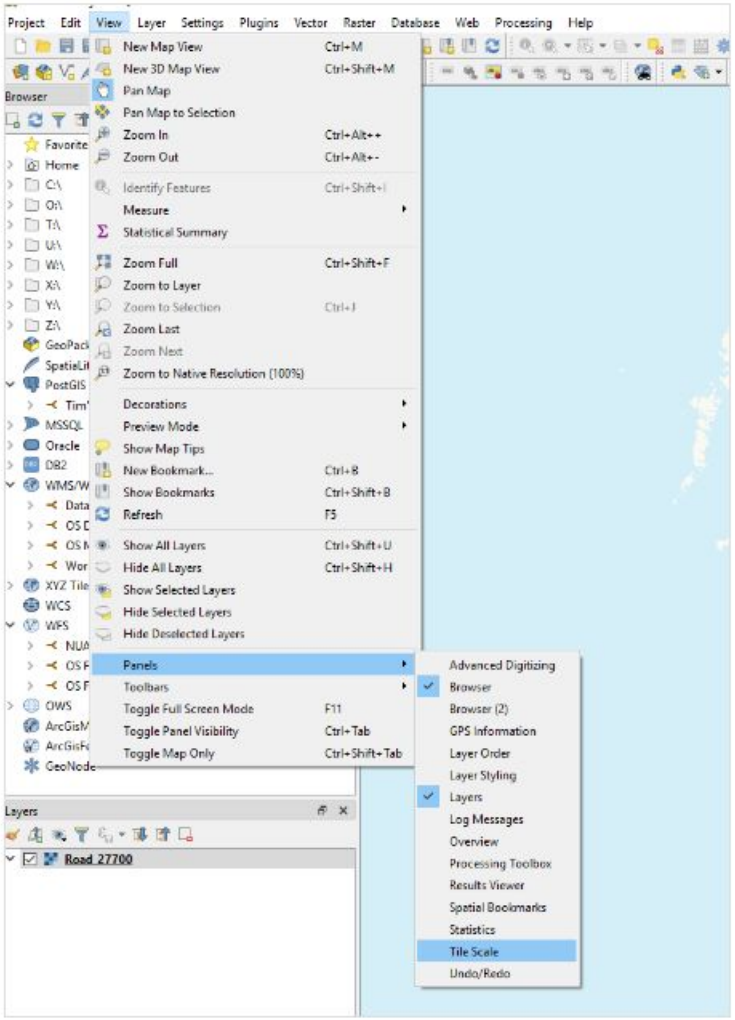
Hands-on: Create Project



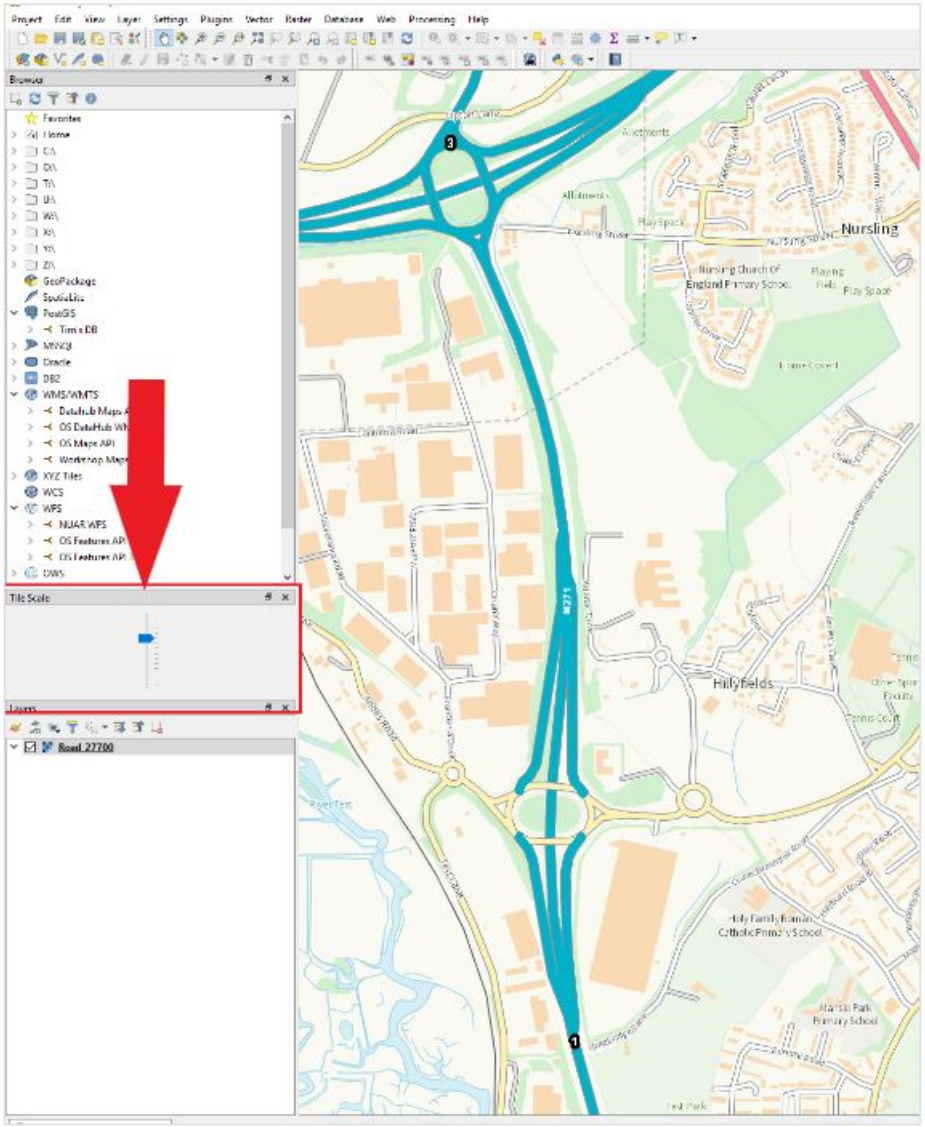
Hands-on: G



Hands-on: Create Project



Hands-on: Create Project



Hands-on: Exercise

Q 1. Create API for Features using OS Data

Q 2. Create a website and add API for maps, features

THANK YOU!

prajwalita.chavan@gmail.com
prajwalita@iitb.ac.in

#OGCAPI