OpenGIS Project

Ulm City Portal

Dilip Biswas
M.Sc. Student Geomatics
Student Nr. 35293, 3rd Semester
Karlsruhe University of Applied Science (HSKA)

………

Ulm City Background

The City of Ulm lies on the northeastern border of the German state of Baden-Württemberg, bordering the Danube River to the west. With a population of approximately 120,000, Ulm serves as the administrative seat of the Alb-Donau district. Founded in the year 850, this large city boasts a plethora of cultural attractions and a high level of economic activity. Tourists from all over the world are drawn to Ulm for several attractions, including the following major cultural sites:

- Ulmer Münster, the church with the tallest steeple in the world,
- The Fisherman's Quarter with its medieval architecture and cobblestone streets,
- The Rathaus with its 16th century murals and 15th century clock,
- The Krone Inn, a complex of medieval buildings where German kings and emperors were lodged,
- The 'Auf dem Kreuz' historic district, a residential zone filled with architecture dating from the 17th century,
- The formerly Benedictine Wiblington Abbey in the south of the city, with its late baroque and early classic architecture and rococo library,
- The Albert Einstein memorial, located at the place of his birth in Ulm,
- The Memorial for Hans and Sophie Scholl in Münsterplatz, two prominent members of the Weiße Rose, a Nazi resistance movement,
- The University of Ulm Botanical Garden,
- The City Public Library, which houses over 480,000 print media,
- Several theatres for the performing arts, including the City Theatre, where visitors can see ballet, dramatic productions, opera and even the City's professional philharmonic orchestra.

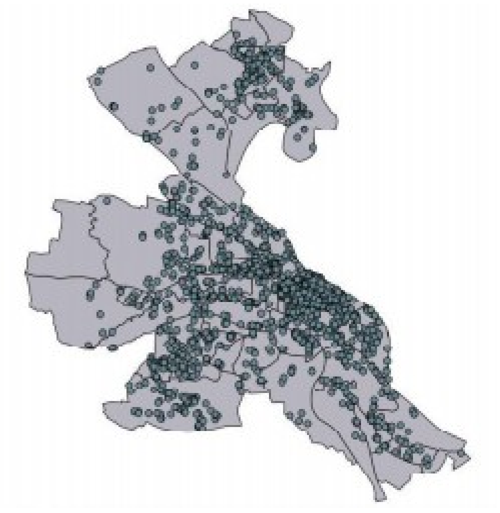
The City is also an educational center, with two major institutions:

- The University of Ulm, a scientifically oriented institution founded in 1967,
- The City University of Applied Sciences, founded in 1960, which draws much of its student body from abroad.

Goal

The goal of this project is to create a navigable, interactive GIS portal for the City of Ulm. Visitors and residents alike will be able to access the portal via the internet and interactively view information about several types of attractions, including historic and cultural sites, dining and shopping. The portal, its attendant functionality and underlying infrastructure will be constructed entirely from open source, non-proprietary GIS software that is freely available for anyone to use, GeoExt, Ext and Openlayers Javascript libraries on the client side and PHP files on the server side.

Data Sources
Point data with coordinate attributes and unique identifiers for public buildings within Ulm will be provided by the City. A table containing relevant information of interest will also be provided by the City, and will be joined to the point data within Quantum GIS. Additional point data for restaurants, shopping and attractions within the City will be extracted from freely available data hosted by Open Street Map. All basemap tiles will be integrated using the Google Maps API for Javascript.



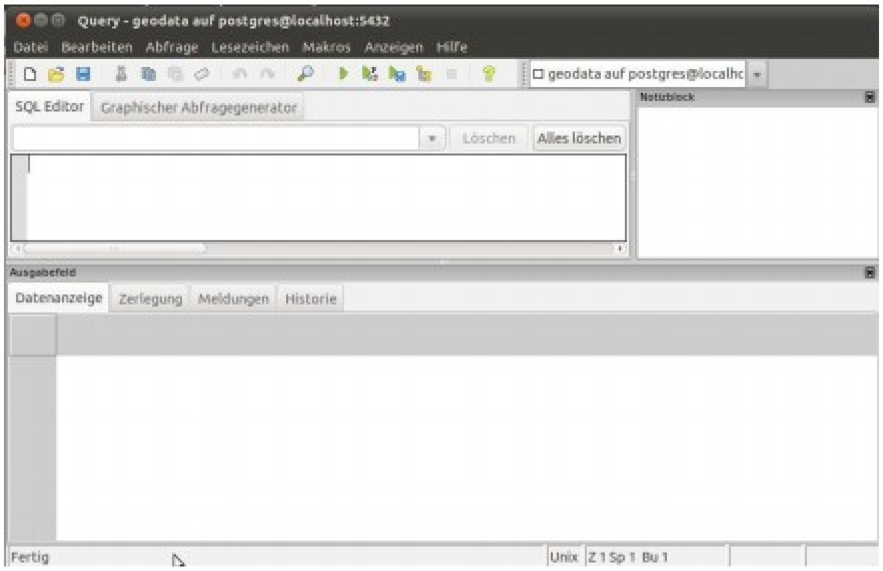
Point data from Open Street Map for the City of Ulm (left); Google basemap imagery (right).

Project Development Stages & Methodology
There are several distinct steps that need to be completed in a specific sequence in order to successfully build an interactive geo-information portal based entirely on open source software. The project will consist of the following stages:

Importing Data
All data will be stored in a Postgres geodatabase on the Linux platform, created and accessed from a Windows machine via Virtual Box in pgAdminIII. This geodatabase will host the point data acquired from both Open Street Map and the City of Ulm. It remains to be seen if these two datasets will be merged into one master dataset; this decision will be made after both have been sufficiently examined.

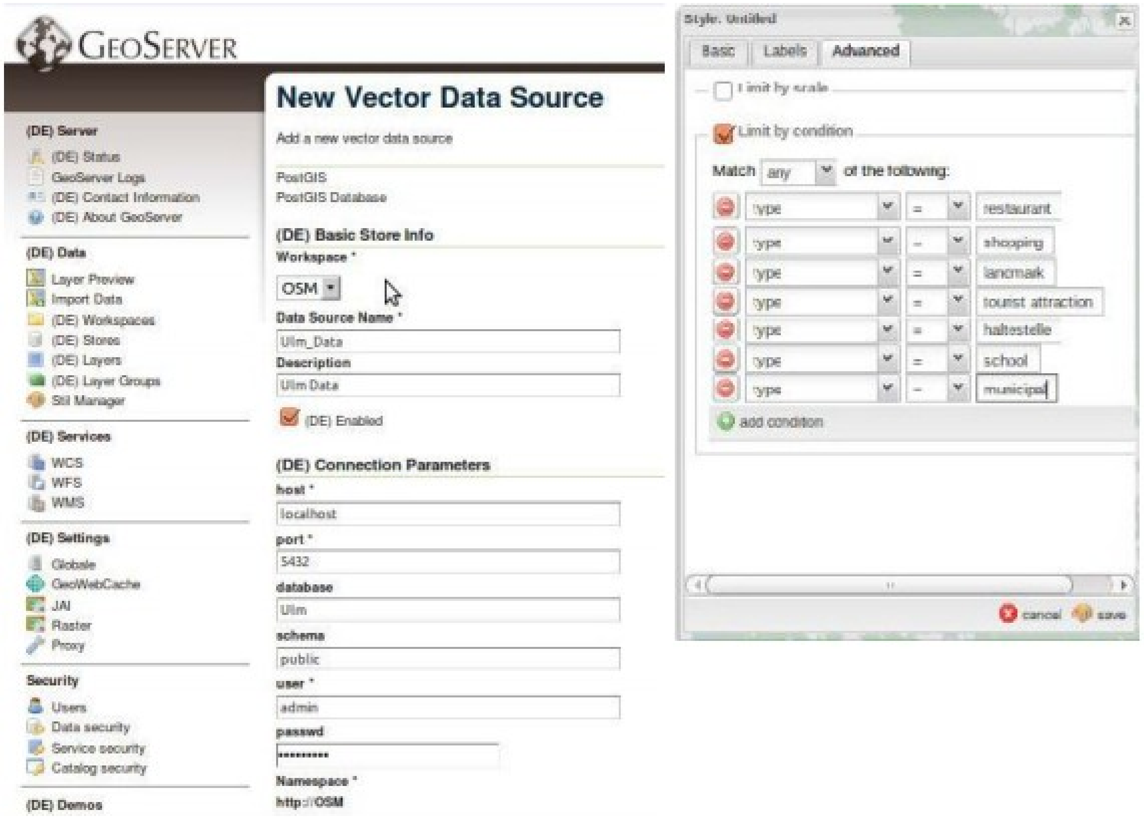
Classification of Point Data
All point data will be re-classified using the sql editor within pgAdminIII into the following classes:

- Historic Sites and areas
- Cultural Attractions (ie. Theatres, cinema, nightclubs, etc.)
- Restaurants
- Shopping and Tram stops


Query - geodata auf postgres@localhost:5432
Datei Bearbeiten Abfrage Lesezeichen Makros Anzeigen Hilfe
geodata auf postgres@localhc
SQL Editor
Graphischer Abfragegenerator
Löschen
Alles löschen
Notizblock
Ausgabefeld
Datenanzeige
Zerlegung
Meldungen
Historie
Fertig
Unix
Z 1 Sp 1 Bu 1

SQL Editor with pgADMINIII

Create webGIS Services for Display



New Vector Data Source

Add a new vector data source

PostGIS
PostGIS Database

(DE) Basic Store Info

Workspace * OSM

Data Source Name * Ulm_Data

Description Ulm Data

[✓ (DE) Enabled]

(DE) Connection Parameters

host * localhost

port * 5432

database Ulm

schema public

user * admin

passwd **********

Namespace * http://OSM

Limit by scale

[✓ Limit by condition]

Match any of the following:

type	=	restaurant
type	=	shopping
type	=	landmark
type	=	tourist attraction
type	=	haltestelle
type	=	school
type	=	municipal

Creating a Data Store in Geoserver (Left); Geoserver Layer Styler (Right)

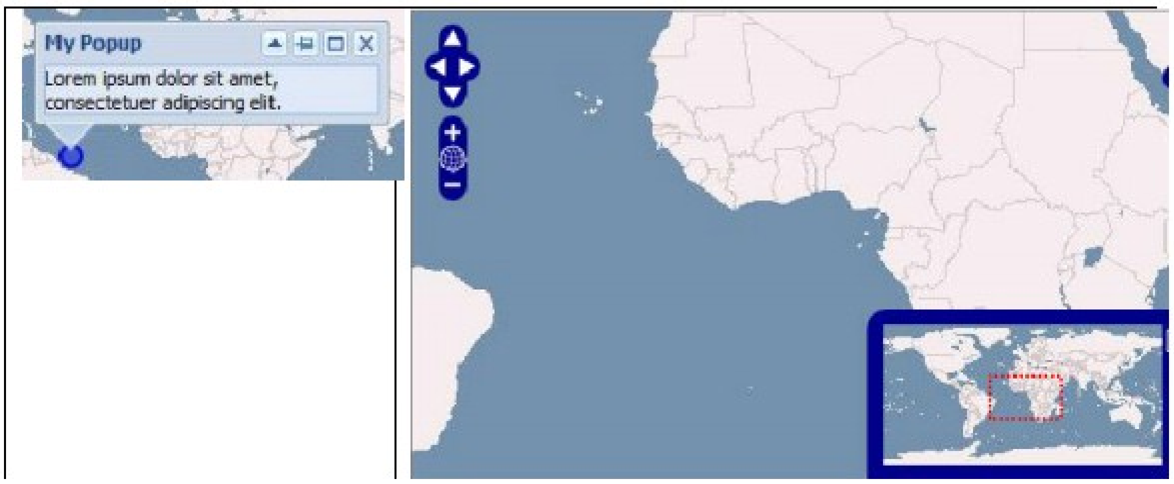
After classifying the data appropriately in pgAdminIII, a webgis service will be created, published and hosted using Geoserver. A Workspace and Data Store within Geoserver will house the point data. The data will then be prepared for display by class using the Geoserver Layer Styler.

Create the Interface
Open Layers will provide the Javascript programming libraries and virtual environment that will be used to create and display the interface. Google Maps tiles for the area will be integrated to serve as the basemap, with the classified points of interest visible on top.

K9915
Blaustein
K9911
Eselsberg
28
19
Ulm
Blaubeurer Straße
10
L2021
ST2023
Harthausen
Neu-Ulm
311
Europastraße
10
Breite
10
Ermingen
ST2029
K9905
Map data ©2011 GeoBasis-DE/BKG (©2009), Google - Terms of Use
Google basemap imagery integrated into the Open Layers environment

Develop Functionality
The GeoExt libraries for JavaScript will be used to create functionality that is not already included by default in the Open Layers environment. The following functionality will be included in the interface:

- Clickable points with a pop up information window displaying name, address and other pertinent information.
- Collapsible legend displaying point classes. Users will be able to click on a class to display a scrollable pick list of each point in that class; once a point in the pick list is clicked the map window will zoom to that point.
- The Google Maps direction service will be integrated, allowing users to drive, bike or walk to and from points of interest of their choice.
- Overview map so users can see where a zoomed-to point lies in the context of the City.
- Standard map navigation features, including pan, zoom, data selection, etc.



Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

Popup information window (Left), Overview Map (Right)

Optional Functionality
The interface would also ideally include the following functionality, which will be implemented based on time constraints:

- Once a user clicks on a point, displaying its information window, functionality embedded in the window would allow the user to enter a radius value to then display all other points of interest within that distance.
- Allow users to add or remove points of interest and attributes for them, or to edit attribute information of existing points.

Technical Details
The point of interest (‘hska_poi2’) and building (hausnummern2’) tables were stored in a server-side PostGIS geodatabase titled ‘Ulm’. They were then joined using the following sql query, creating a new View (‘culture’) in the database:

```
SELECT hausnummern2.hausnummer, hausnummern2.bezeichnun, poi_hska2.gid,
poi_hska2.stname, poi_hska2.name, poi_hska2.haus_number, poi_hska2.description,
poi_hska2.url, poi_hska2.thumb, poi_hska2.type, poi_hska2.the_geom, poi_hska2.pic,
poi_hska2.source FROM poi_hska2 LEFT JOIN hausnummern2 ON hausnummern2.hausnummer
= poi_hska2.haus_number AND hausnummern2.bezeichnun = poi_hska2.stname;
```

Once the culture view was created, it was necessary to create a methodology for visualizing the points on the map and allowing users to see and query feature attributes within the graphical user interface. For pure visualization within the map, we decided to use the GeoJSON format. This required the creation of a PHP file to be stored on the server for the client-side Javascript to interact with. The following lines of code within the Javascript file were used to create a feature store for the points and call the PHP code:

```
store = new GeoExt.data.FeatureStore({
        layer: vecLayer,
        fields: [{name: 'name', type: "string"},
        {name: 'type', type: "string"},
        {name: 'description', type: "string"},
        {name: 'haus_number', type: "int"},
        {name: 'stname', type: "string"},
        {name: 'url', type: "string"},],
        // define server where data is loading from
        proxy: new GeoExt.data.ProtocolProxy({
        protocol: new OpenLayers.Protocol.HTTP({
        url: "php/DB2GeoJSON.php",
        format: new OpenLayers.Format.GeoJSON()
        })
        }),
        autoLoad: true
        });//close store
```

The PHP file, titled DB2geoJSON, was created to interact with the ‘culture’ view stored within the PostGIS database. An sql query within the DB2geoJSON pulls the necessary field from the database (the_geom); it encodes this string as GeoJSON format, which is readable by Javascript; the points are then visualized in the map using Javascript code. The sql query (displayed below) within the

DB2geoJSON file also specifies other fields to be populated, including attraction name and type:

```
$sql_cnt = "SELECT gid, name, stname, type, haus_number, url, description, pic,
ST_AsGeoJson(the_geom) FROM culture WHERE name ILIKE '". $name ."%' AND type ILIKE '".
$type ."%' ORDER BY name";
```

Once the points were visualized within the map, we created rules for their display with icons by using the Openlayers.rule class, as shown in the following Javascript code:

```
var rule2 = new OpenLayers.Rule({
        title:"Cultural Site",
filter: new OpenLayers.Filter.Comparison({
        type: OpenLayers.Filter.Comparison.EQUAL_TO,
        property: "type",
        value: 'Cultural Site'
}),
symbolizer:{
        externalGraphic:'/image/famfamfam/icons/star2.png',
        graphicWidth:'20',
        graphicHeight:'20',
        fillOpacity: 1
}
});
```

The DB2geoJSON file is also being used to populate the following components:

• The attribute grid in the map's legend panel (with Type and Name fields) which is an object from the Ext.Gridpanel class. The Javascript code below illustrates this process:

```
// Attribute Panel
gridPanel = new Ext.grid.GridPanel({ // loads static data from the server via json file
        title: "Point of Interest Attributes",
        region: "east",
        store: store, // data comes from GeoJSON store declared above
        width: 298,
        height: 400,
        collapsible:true,
        columns: [{ // array for columns
        header: "Name",
        width: " 196",
        dataIndex: "name"
},{
        header: "Type",
        width: 98,
        dataIndex: "type"
}],
        sm: new GeoExt.grid.FeatureSelectionModel({ // click on record to
        show picture and pan
        autoPanMapOnSelect:'true'
}),
listener: function(){autoPanMapOnSelect:'true'}
});
```

• The popup information windows, which are objects from the GeoExt.Popup class The Javascript code below illustrates this process:

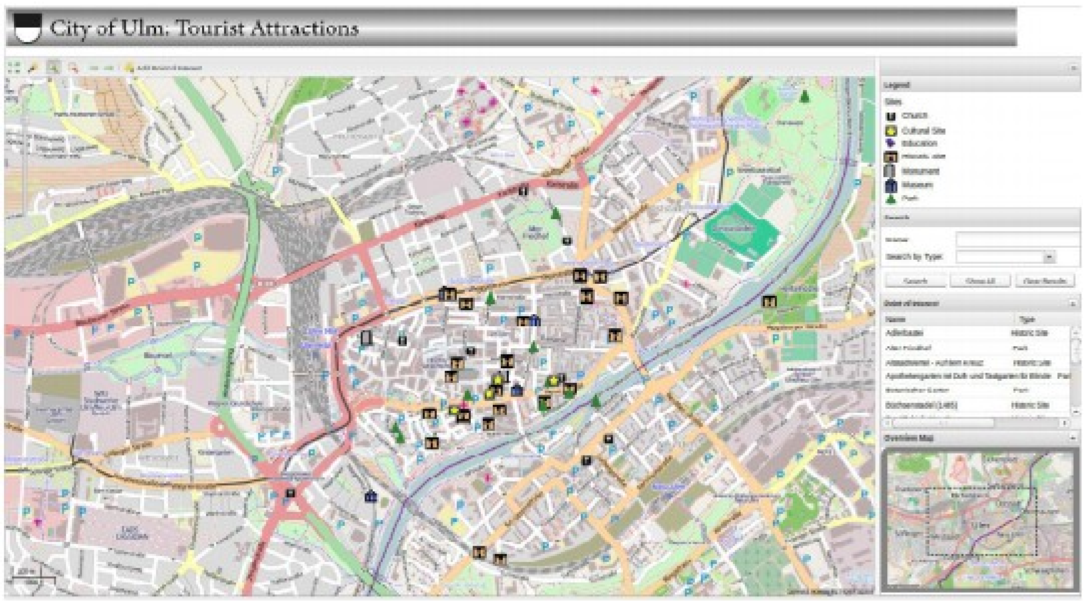
```
// Popup information windows
var popup;
function createPopup(feature){
        if (!popup){
        popup = new GeoExt.Popup({
        title: feature.attributes.name,
        location: feature,
        width:300,
        html: "<b>" + "Type: " +
        "</b>"+feature.attributes.type + "<br/><b>" +
        "Description: " + "</b>" +
        feature.attributes.description +

        "<br/><b>" + "Street: " + "</b>"+feature.attributes.stname + "<br/><b>" +
        "Street Number: " + "</b>"+feature.attributes.haus_number + "<br/><b>" +
        "</b>" +"<a href=\""+feature.attributes.url+ "\">More Information</a><br/>"+
        "<img src=\"" + feature.attributes.pic + "\"/>",
        maximizable: true,
        collapsible: true,
        draggable: true,
        listeners: {
        close: function() {
        popup = null;
        },
        }
        });
```

• The search form, which is an object of the GeoExt.FormPanel class. The Javascript code below illustrates this process:

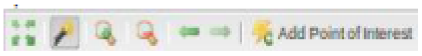
```
// search panel
var formPanel = new GeoExt.form.FormPanel({
title: "Search",
        // below protocol defines where user data is being sent
        protocol: new OpenLayers.Protocol.HTTP({
        url: "php/DB2GeoJSON.php",
        format: new OpenLayers.Format.GeoJSON(),
        params: {command: "insert"}
}),
```

Functionality:
The following interactive interface displays when the site is loaded:

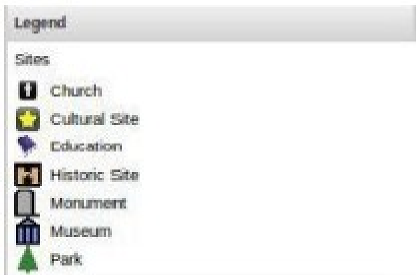


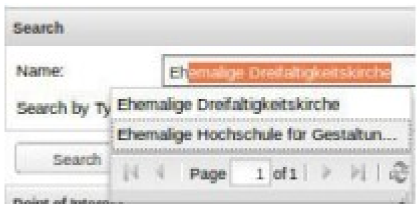
The following functionality is included in the interface:

• Navigation tools allowing the user to pan, zoom in, zoom out, zoom to previous extent, zoom to maximum extent and add a point of interest.



• A legend displaying the classified points of interest. A search form allowing users to search by points of interest either by name or type. The 'Name' field has autofill functionality, which communicates with DB2geoJSON.php to populate it with the names in the database, as shown at right. If the user decides to search by type, selecting one of the types in the combobox will populate the attribute table with only features of that type and only features of that type will be visualized in the map. This functionality is demonstrated below for Historic Sites.

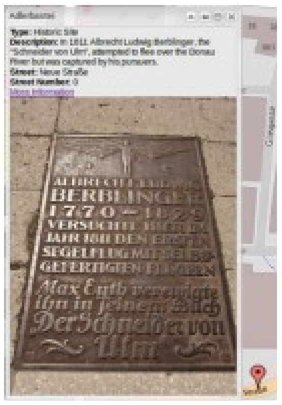



Search
Name:
Ehemalige Dreifaltigkeitskirche
Search by Ty
Ehemalige Dreifaltigkeitskirche
Ehemalige Hochschule für Gestaltun...
Search
Page 1 of 1

• The Points of Interest attribute grid below the search form also contains functionality. If a user selects a record from the table, the map will automatically pan and zoom to the point, which is represented by a red marker upon selection. A popup information window also appears containing the name of the site, a brief description, the address, a website link if available, and a picture if available.
• Users can add their own Points of Interest by clicking the 'Add Point of Interest' button located in the

toolbar to enable the functionality, followed by a single mouse click in the map at the desired location. Once the user clicks in the map, a form is displayed with attribute fields that can be filled in by the user. Once the 'Save' button is pressed, a separate server-side PHP file is called (edit.php) which contains an sql query for inserting these attributes into the database at the desired location. Since the points are being displayed as a GeoJSON vector layer, the user is required to refresh the page before the new point of interest appears on the map.

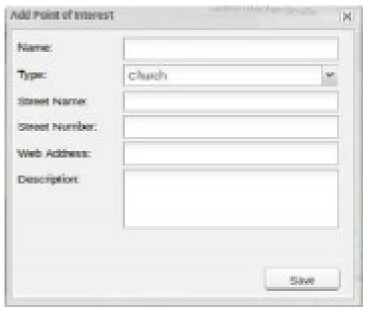
• An overview map is included in the legend panel so users are able to orient themselves in the city when they are zoomed to a detailed level in the main map.



Adlerbastei

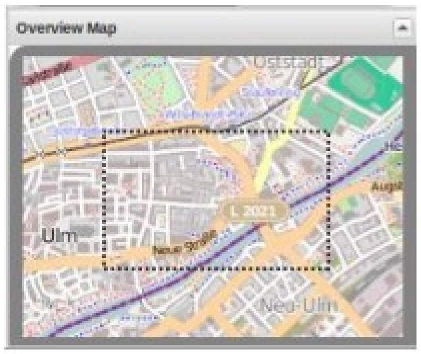
Type: Historic Site
Description: In 1811 Albrecht Ludwig Berblinger, the "Schneider von Ulm", attempted to flee over the Donau River but was captured by his pursuers.
Street: Neue Straße
Street Number: 0
More information

ALBRECHT LUDWIG
BERBLINGER
1770 – 1829
VERSUCHTE IM JAHR 1811 DEN ERSTEN SEGELFLUG MIT SELBSTGEFERTIGTEN FLÜGELN
Max Eyth verewigte ihn in seinem Buch
Der Schneider von Ulm



Add Point of Interest

Name:	
Type:	Church
Street Name:	
Street Number:	
Web Address:	
Description:	



Overview Map
Oststadt
Ulm
Neue Straße
L 2021
Neu-Ulm