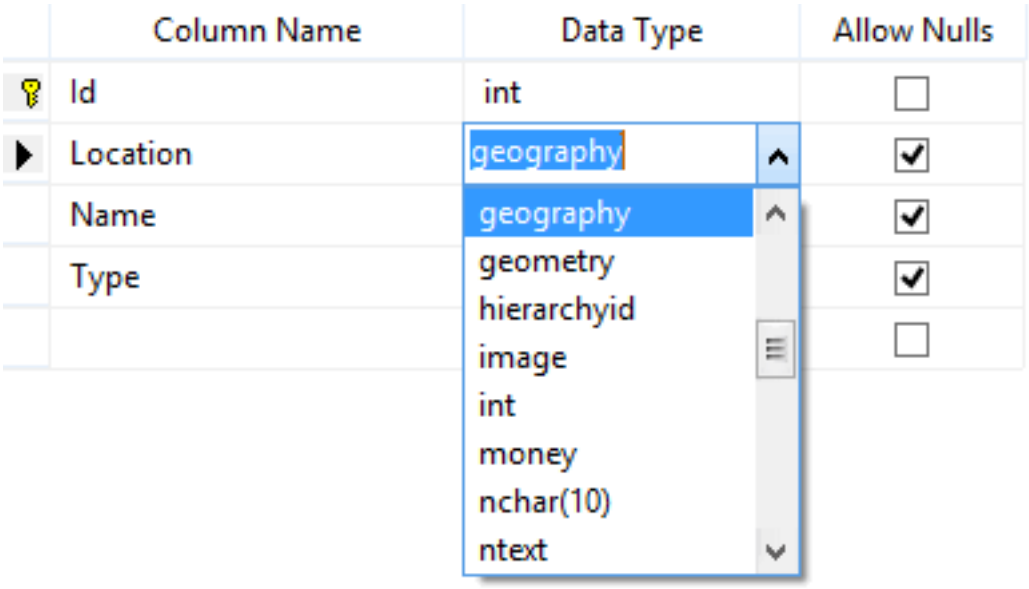
عنوان: پردازش داده‌های جغرافیایی به کمک SQL Server و Entity framework
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۳۰ ۱۰:۴۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, SQL Server, GIS

پشتیبانی SQL Server از Spatial data

از SQL Server 2008 به بعد، نوع داده جدیدی به نام geography به نوع‌های قابل تعریف ستون‌ها اضافه شده‌است. در این نوع ستون‌ها می‌توان طول و عرض جغرافیایی یک نقطه را ذخیره کرد و سپس به کمک توابع توکاری از آن‌ها کوئری گرفت.



در اینجا نمونه‌ای از نحوه‌ی تعریف و همچنین مقدار دهی این نوع ستون‌ها را مشاهده می‌کنید:

```
CREATE TABLE [Geo](
[id] [int] IDENTITY(1,1) NOT NULL,
[Location] [geography] NULL
)

 insert into Geo( Location , long, lat ) values
( geography::STGeomFromText ('POINT(-121.527200 45.712113)', 4326))
```

متد geography::STGeoFromText یک SQL CLR function است. این متد در مثال فوق، مختصات یک نقطه را دریافت کرده‌است. همچنین نیاز دارد بداند که این نقطه توسط چه نوع سیستم مختصاتی ارائه می‌شود. عدد 4326 در اینجا یک SRID یا Spatial Reference System Identifier استاندارد است. برای نمونه اطلاعات ارائه شده توسط Google و یا Bing توسط این استاندارد ارائه می‌شوند.
در اینجا متدهای توکار دیگری مانند geography::STDistance برای یافتن فاصله مستقیم بین نقاط نیز ارائه شده‌اند. خروجی آن بر حسب متر است.

پشتیبانی از Spatial Data در Entity framework

پشتیبانی از نوع مخصوص geography، در EF 5 توسط نوع داده‌ای DbGeography ارائه شد. این نوع داده‌ای immutable است. به این معنا که پس از نمونه سازی، دیگر مقدار آن قابل تغییر نیست.

در اینجا برای نمونه مدلی را مشاهده می‌کنید که از نوع داده‌ای DbGeography استفاده می‌کند:

```
using System.Data.Entity.Spatial;

namespace EFGeoTests.Models
{
    public class GeoLocation
    {
        public int Id { get; set; }
        public DbGeography Location { get; set; }
        public string Name { get; set; }
        public string Type { get; set; }

        public override string ToString()
        {
            return string.Format("Name:{0}, Location:{1}", Name, Location);
        }
    }
}
```

به همراه یک Context، تا کلاس GeoLocation در معرض دید EF قرار گیرد:

```
using System;
using System.Data.Entity;
using EFGeoTests.Models;

namespace EFGeoTests.Config
{
    public class MyContext : DbContext
    {
        public DbSet<GeoLocation> GeoLocations { get; set; }

        public MyContext()
            : base("Connection1")
        {
            this.Database.Log = sql => Console.Write(sql);
        }
    }
}
```

برای مقدار دهی خاصیت Location از نوع DbGeography می‌توان از متد ذیل استفاده کرد که بسیار شبیه به متد geography::STGeoFromText عمل می‌کند:

```
private static DbGeography createPoint(double longitude, double latitude,  int coordinateSystemId =
4326)
  {
       var text = string.Format(CultureInfo.InvariantCulture.NumberFormat,"POINT({0} {1})", longitude,
latitude);
       return DbGeography.PointFromText(text, coordinateSystemId);
  }
```

تهیه منبع داده‌ی جغرافیایی

برای تدارک یک مثال واقعی جغرافیایی، نیاز به اطلاعاتی دقیق داریم. این نوع اطلاعات عموما توسط یک سری فایل مخصوص به نام Shapefiles که حاوی اطلاعات برداری جغرافیایی هستند ارائه می‌شوند. برای نمونه اطلاعات جغرافیایی به روز ایران را از آدرس ذیل می‌توانید دریافت کنید:

http://download.geofabrik.de/asia/iran.html

http://download.geofabrik.de/asia/iran-latest.shp.zip

پس از دریافت این فایل، به تعدادی فایل با پسوندهای shp، shx و dbf خواهیم رسید.
فایل‌های shp بیانگر فرمت اشکال ذخیره شده هستند. فایل‌های shx یک سری ایندکس بوده و فایل‌های dbf از نوع بانک اطلاعاتی dBase IV می‌باشند.

همچنین اگر فایل‌های prj را باز کنید، یک چنین اطلاعاتی در آن موجودند:

```
GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137,298.257223563]],PRIMEM["Greenwich"
,0],UNIT["Degree",0.0174532925199433295]]
```

نکته‌ی مهمی که در اینجا باید مدنظر داشت، استاندارد GCS_WGS_1984 آن است. این استاندارد معادل است با استاندارد EPSG 4326. عدد 4326 آن جهت ثبت این اطلاعات در یک بانک اطلاعاتی SQL Server حائز اهمیت است (پارامتر coordinateSystemId در متد createPoint) و ممکن است از هر فایلی به فایل دیگر متفاوت باشد.

خواندن فایل‌های shp در دات نت

پس از دریافت فایل‌های shp و بانک‌های اطلاعاتی مرتبط با اطلاعات جغرافیایی ایران، اکنون نوبت به پردازش این فایل‌های مخصوص با فرمت بانک اطلاعاتی فاکس پرو مانند، رسیده‌است. برای این منظور می‌توان از پروژه‌ی سورس باز ذیل استفاده کرد:

C# Esri Shapefile Reader

این پروژه در خواندن فایل‌های shp بدون نقص عمل می‌کند اما توانایی خواندن نام‌های فارسی وارد شده در این نوع بانک‌های اطلاعاتی را ندارد. برای رفع این مشکل، سورس آن را از Codeplex دریافت کنید. سپس فایل Shapefile.cs را گشوده و ابتدای خاصیت Current آن‌را به نحو ذیل تغییر دهید:

```
/// <summary>
        /// Gets the current shape in the collection
        /// </summary>
        public Shape Current
        {
            get
            {
                if (_disposed) throw new ObjectDisposedException("Shapefile");
                if (!_opened) throw new InvalidOperationException("Shapefile not open.");

                // get the metadata
                StringDictionary metadata = null;
                if (!RawMetadataOnly)
                {
                    metadata = new StringDictionary();
                    for (int i = 0; i < _dbReader.FieldCount; i++)
                    {
                        string value = _dbReader.GetValue(i).ToString();
                        if (_dbReader.GetDataTypeName(i) == "DBTYPE_WVARCHAR")
                        {
                            // برای نمایش متون فارسی نیاز است
                            value = Encoding.UTF8.GetString(Encoding.GetEncoding(720).GetBytes(value));
                        }
                        metadata.Add(_dbReader.GetName(i),
                            value);
                    }
                }
```

در اینجا فقط سطر استفاده از Encoding خاصی با شماره 720 و تبدیل آن به UTF8 اضافه شده‌است. پس از آن بدون مشکل می‌توان برچسب‌های فارسی را از فایل‌های dBase IV این نوع بانک‌های اطلاعاتی استخراج کرد (اصلاح شده‌ی آن در فایل پیوست مطلب موجود است).

```
using System.Collections.Generic;
using System.Linq;
using Catfood.Shapefile;

namespace EFGeoTests
{
    public class MapPoint
    {
        public Dictionary<string, string> Metadata { set; get; }
        public double X { set; get; }
        public double Y { set; get; }
    }
```

```
    public static class ShapeReader
    {
        public static IList<MapPoint> ReadShapeFile(string path)
        {
            var results = new List<MapPoint>();

            using (var shapefile = new Shapefile(path))
            {
                foreach (var shape in shapefile)
                {
                    if (shape.Type != ShapeType.Point)
                        continue;

                    var shapePoint = shape as ShapePoint;
                    if (shapePoint == null)
                        continue;


                     var metadataNames = shape.GetMetadataNames();
                    if(!metadataNames.Any())
                        continue;

                    var metadata = new Dictionary<string, string>();
                    foreach (var metadataName in metadataNames)
                    {
                        metadata.Add(metadataName,shape.GetMetadata(metadataName));
                    }

                    results.Add(new MapPoint
                    {
                        Metadata = metadata,
                        X = shapePoint.Point.X,
                        Y = shapePoint.Point.Y
                    });
                }
            }

            return results;
        }
    }
}
```

در کدهای فوق به کمک کتابخانه‌ی C# Esri Shapefile Reader، اطلاعات نقاط بانک اطلاعاتی shape files را خوانده و به صورت لیست‌هایی از MapPoint بازگشت می‌دهیم. نکته‌ی مهم آن، Metadata است که از هر فایلی به فایل دیگر می‌توان متفاوت باشد. به همین جهت این اطلاعات را به شکل ویژگی‌های key/value در این نوع بانک‌های اطلاعاتی ذخیره می‌کنند.

افزودن اطلاعات جغرافیایی به بانک اطلاعاتی SQL Server به کمک Entity framework

فایل places.shp را در مجموعه فایل‌هایی که در ابتدای بحث عنوان شدند، می‌توانید مشاهده کنید. قصد داریم اطلاعات نقاط آن‌را به مدل GeoLocation انتساب داده و سپس ذخیره کنیم:

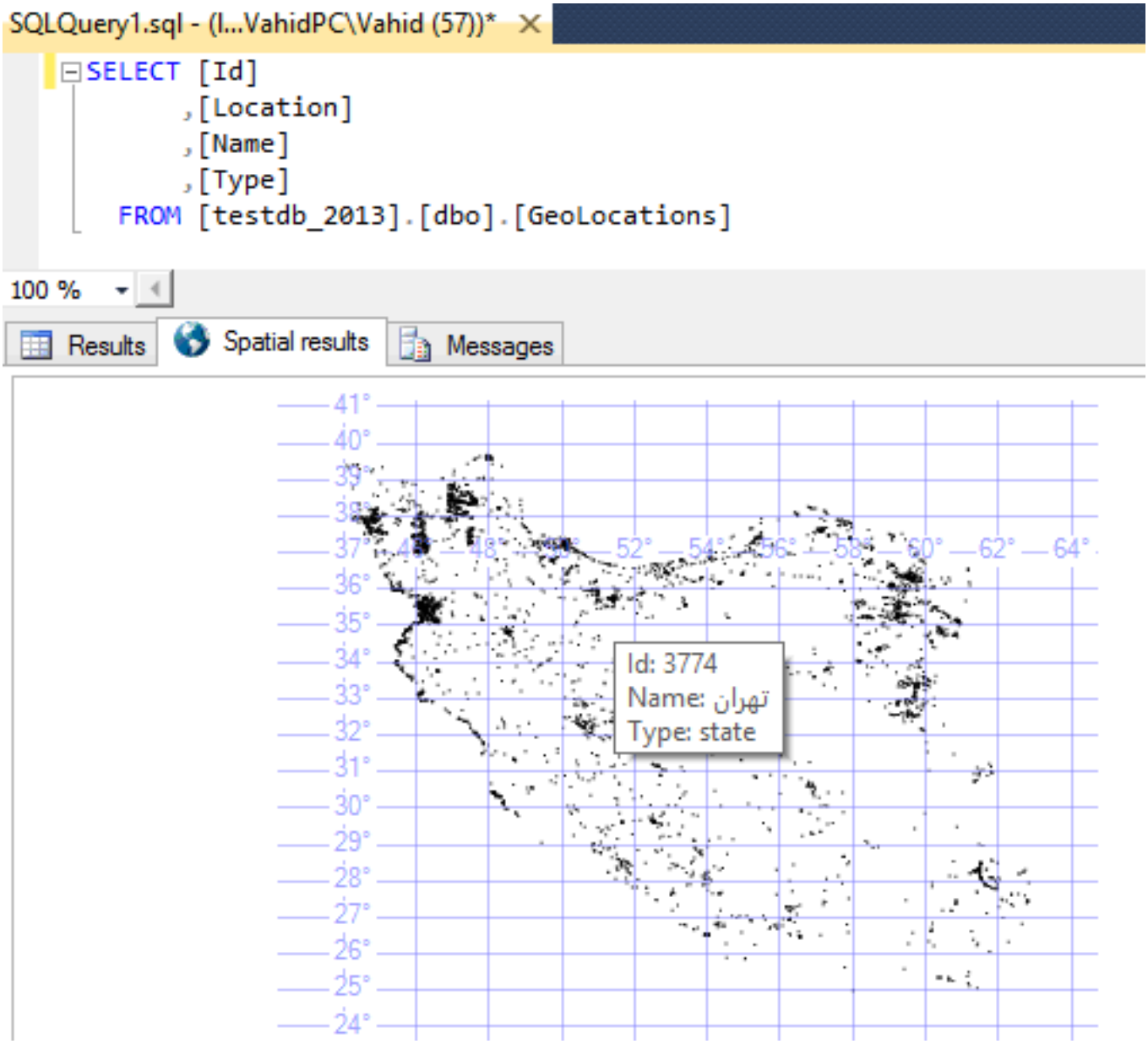
```
var points = ShapeReader.ReadShapeFile("IranShapeFiles\\places.shp");
            using (var context = new MyContext())
            {
                context.Configuration.AutoDetectChangesEnabled = false;
                context.Configuration.ProxyCreationEnabled = false;
                context.Configuration.ValidateOnSaveEnabled = false;

                if (context.GeoLocations.Any())
                    return;

                foreach (var point in points)
                {
                    context.GeoLocations.Add(new GeoLocation
                    {
                        Name = point.Metadata["name"],
                        Type = point.Metadata["type"],
                        Location = createPoint(point.X, point.Y)
                    });
                }

                context.SaveChanges();
            }
```

تعریف متد createPoint را که بر اساس X و Y نقاط، معادل قابل پذیرش آن‌را جهت SQL Server تهیه می‌کند، در ابتدای بحث مشاهده کردید.
در فایل‌های مرتبط با places.shp، متادیتا name، معادل نام شهرهای ایران است و type آن بیانگر شهر، روستا و امثال آن می‌باشد.
پس از اینکه اطلاعات مکان‌های ایران، در SQL Server ذخیره شدند، نمایش بصری آن‌ها را در management studio نیز می‌توان مشاهده کرد:



کوئری گرفتن از اطلاعات جغرافیایی

فرض کنید می‌خواهیم مکان‌هایی را با فاصله کمتر از 5 کیلومتر از تهران پیدا کنیم:

```
var tehran = createPoint(51.4179604, 35.6884243);

            using (var context = new MyContext())
            {
                // find any locations within 5 kilometers ordered by distance
```

```
                var locations = context.GeoLocations
                    .Where(loc => loc.Location.Distance(tehran) < 5000)
                    .OrderBy(loc => loc.Location.Distance(tehran))
                    .ToList();

                foreach (var location in locations)
                {
                    Console.WriteLine(location.Name);
                }
            }
```

همانطور که پیشتر نیز عنوان شد، متد Distance بر اساس متر کار می‌کند. به همین جهت برای تعریف 5 کیلومتر به نحو فوق عمل شده‌است. همچنین نحوه‌ی مرتب سازی اطلاعات نیز بر اساس فاصله از یک مکان مشخص صورت گرفته‌است.

و یا اگر بخواهیم دقیقا بر اساس مختصات یک نقطه، مکانی را بیابیم، می‌توان از متد SpatialEquals استفاده کرد:

```
var tehran = createPoint(51.4179604, 35.6884243);
            using (var context = new MyContext())
            {
                // find any locations within 5 kilometers ordered by distance
                var tehranLocation = context.GeoLocations.FirstOrDefault(loc =>
loc.Location.SpatialEquals(tehran));
                if (tehranLocation != null)
                {
                    Console.WriteLine(tehranLocation.Type);
                }
            }
```

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید:

EFGeoTests.zip

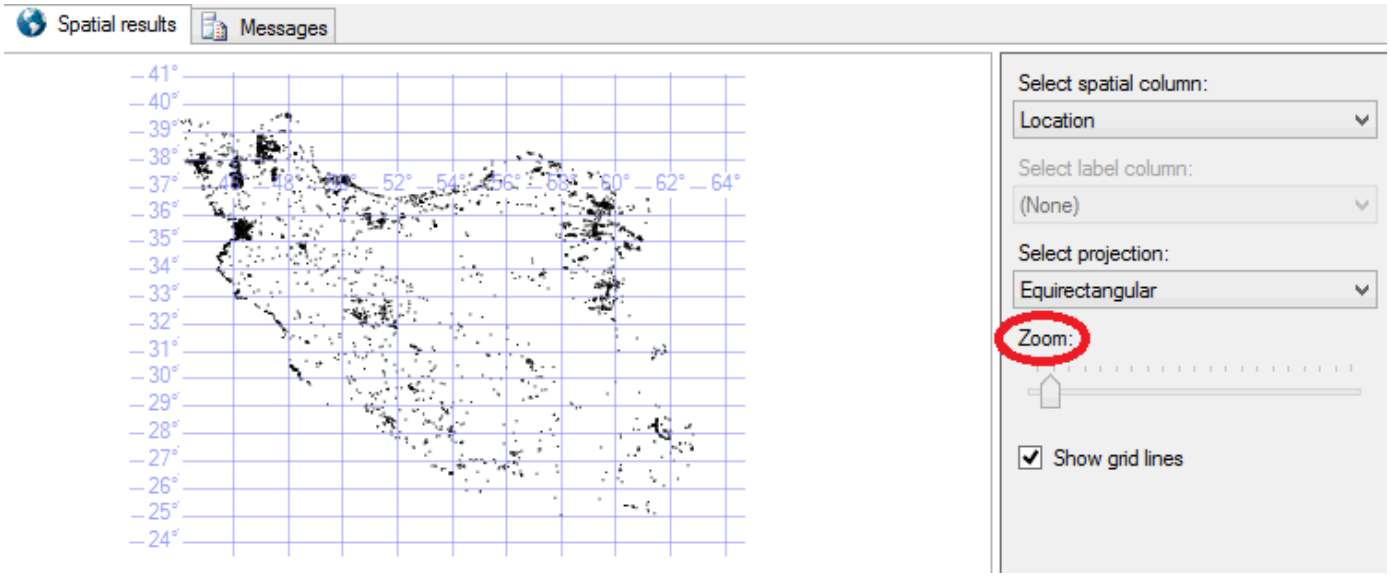
نظرات خوانندگان

نویسنده: محمد باقر سیف اللهی
تاریخ: ۱۳۹۳/۰۳/۳۱ ۱۰:۴۱

سلام
مشکلی که قبل‌تر در حین کار با فایل‌های shp برخوردم، تغییر مقیاس نقشه و نگاشت مختصات خاص و توزیع شده ای بود که باید روی نقشه نمایش داده می‌شد . (مثلا نمایش مراکز استان‌ها روی نقشه و تغییر scale نقشه و به تبع اون، تغییر مکان مختصات)
برای این کار چه راهکاری هست؟
* دسترسی به فایل های dbf و prj و sbn و sbx و shx وجود دارد.
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۳۱ ۱۱:۱۰

این‌ها بیشتر مسایل نمایشی است و توانمندی ابزار نمایش دهنده‌ی اطلاعات نقشه. نیازی نیست در اصل دیتابیس و اطلاعات، تغییری حاصل شود؛ چون اندازه‌ی نمایشی حتی اگر 10 برابر هم شود، در فاصله‌ی بین تهران و شیراز نهایتا تغییری حاصل نخواهد شد و طول و عرض جغرافیایی مکان‌ها ثابت خواهند ماند. برای نمونه اگر مثال پیوست شده را اجرا کنید، خود management studio امکان تغییر اندازه‌ی نمایشی را دارد:



در برنامه‌های دات نت هم برای مثال از SharpMap می‌شود برای نمایش این نوع اطلاعات به همراه تغییر اندازه و ابعاد خودکار نقشه استفاده کرد. برای برنامه‌های وب هم jVectorMap چنین قابلیت‌هایی را دارد.

نویسنده: بهراد ایزدی
تاریخ: ۱۳۹۳/۰۴/۰۲ ۱۵:۰

سلام
ممنون برای این آموزش خوب
بعضی مناطق فایل shp رو ندارند و فقط دو نوع OSM براشون موجود هست راهی برای خواندن اونها وجود داره ؟

ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۰۲ ۱۵:۱۲

- برنامه‌ی سورس باز ArcGIS Editor for OpenStreetMap یک چنین قابلیتی را دارد.
- همچنین یک سری 4 قسمتی را در اینجا می‌توانید در مورد تبدیل open street maps به داده‌های SQL Server مطالعه کنید.

نویسنده: شاهین کیاست
تاریخ: ۱۳۹۳/۰۷/۱۳ ۱۷:۶

آیا ممکن هست به جای نوع داده ی DbGeography از نوع داده‌ی SQLGeometry استفاده کرد؟
-ویرایش
تنها کافی است نوع Property مورد نظر را DbGeometry تعیین کرد.

عنوان: **آشنایی با Leaflet**
نویسنده: شاهین کیاست
تاریخ: ۱۳۹۳/۱۱/۱۳ ۱۹:۲۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, GIS, leafletjs, Map

مقدمه

سیستم‌های جغرافیایی و GIS اهمیت زیادی در زندگی روزمره‌ی ما دارند. GIS به نرم افزار یا سخت افزاری اطلاق می‌شود که کاربر را قادر می‌سازد تا به ذخیره، بازیابی و تجزیه و تحلیل داده‌های جغرافیایی (Spatial) بپردازد. یکی از پایه‌های نرم افزارهای GIS، نقشه و نمایش اطلاعات بر روی نقشه می‌باشد. به طور حتم در وب سایت‌ها مشاهده کرده‌اید که آدرس یک شرکت بر روی نقشه نمایان می‌شود یا به عنوان مثالی دیگر سرویس دهنده‌های اینترنت از نقشه برای نمایش میزان و کیفیت آنتن دهی در محله‌های مختلف یک شهر استفاده می‌کنند.

برای نمایش نقشه در نرم افزارهای تحت وب کتابخانه‌های JavaScript ایی زیادی وجود دارند. این مطلب به معرفی کتاب خانه‌ی کدباز و رایگان leaflet می‌پردازد. leaflet یک کتابخانه‌ی مدرن JavaScript برای کار با نقشه می‌باشد. از خصوصیات بارز این کتابخانه پشتیبانی بسیار خوب آن از موبایل و دستگاههای لمسی است. Leaflet تنها 33 کیلوبایت حجم دارد و ویژگی‌های آن اغلب نیازهای‌های توسعه دهندگان را برای پیاده سازی نرم افزارهای مبتنی بر نقشه پوشش می‌دهد. از مزایای این کتابخانه می‌توان به مشارکت جامعه‌ی بزرگ توسعه دهندگان، سورس خوانا و تمیز، مستندات خوب و تعداد زیادی پلاگین برای آن اشاره کرد.

آماده سازی صفحه

برای استفاده از Leaflet ابتدا باید فایل Style و JavaScript کتابخانه را ارجاع داد:

```
<script src="http://cdn.leafletjs.com/leaflet-0.7.3/leaflet.js"></script>
 <link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.7.3/leaflet.css" />
```

سپس یک div با یک Id مشخص را به صفحه اضافه می‌کنیم. div مورد نظر باید از ارتفاع مشخصی برخوردار باشد که به سادگی با style زیر میسر می‌گردد:

```
#map { height: 600px; }
```

پس از انجام مقدمات اکنون می‌توان یک نقشه را با تنظیمات دلخواهی در div تعریف شده نمایش داد.

تنظیمات اولیه نقشه

با کد زیر ابتدا یک وهله از شیء map ایجاد می‌شود:

```
var map = L.map('map').setView([29.6760859,52.4950737], 13);
```

همانطور که مشاهده می‌شود شناسه‌ی div تعریف شده از طریق سازنده به map پاس داده شده است و سپس به کمک تابع setView به محل مختصات جغرافیایی مورد نظر با زوم پیشفرض 13 نمایش داده می‌شود. طراحی Leaflet به صورتی است که استفاده از متدهای زنجیروار (chainable) را میسر می‌سازد. به عنوان نمونه در کد بالا تابع setView یک شیء map را بر می‌گرداند و توسعه دهنده می‌تواند از توابع دیگر مقدار بازگشتی استفاده کند. این مورد از نظر طراحی شبیه به jQuery می‌باشد.

اگر Google Maps را مشاهده کنید، متوجه می‌شوید که یک نقشه، به صورت مستطیل مستطیل، بارگزاری می‌شود. به این مستطیل‌ها Tile گفته می‌شود. tileها همان فایل‌های png هستند و درواقع به ازای زوم‌های مختلف در محل‌های مختلف، tileهای متفاوتی با شناسه‌ی مشخصی وجود دارند. تصویر زیر نقشه‌ی Google می‌باشد؛ قبل از اینکه tileها بارگزاری شوند. اگر با دقت نگاه کنید مستطیل‌های بزرگ و کوچکی را مشاهده می‌کنید که قسمت‌های مختلف یک نقشه یا همان تایل می‌باشند.

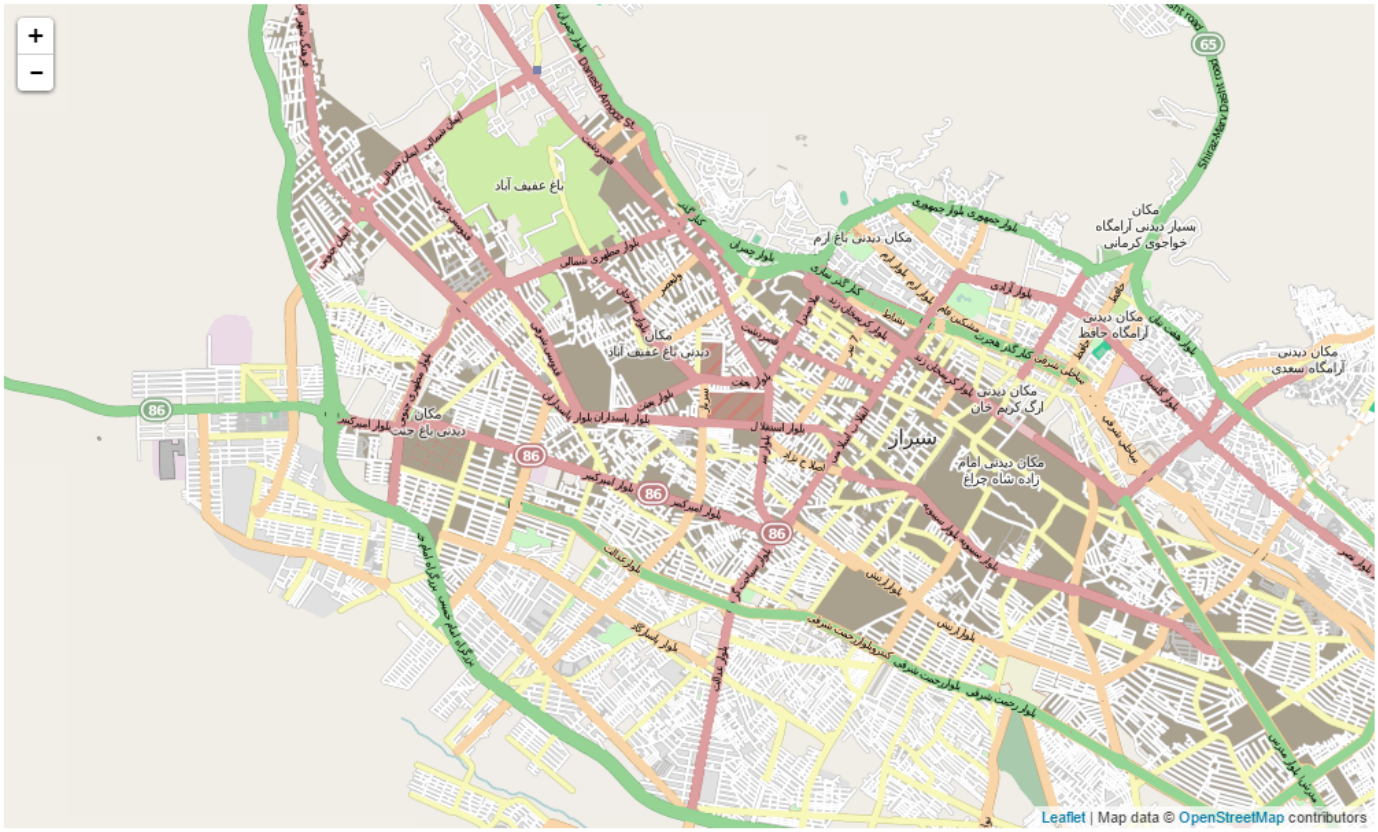
پس برای نمایش یک نقشه نیاز است tileها را از یک منبع، در اختیار نقشه قرار داد. این منبع می‌تواند یک وب سرویس باشد.

پس از تعریف اولیه، نیاز است یک Tile Layer ایجاد کرده و آن را به نقشه اضافه کرد:

```
var osmUrl='http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png';
var osmAttrib='Map data © <a href="http://openstreetmap.org">OpenStreetMap</a> contributors';
var osm = new L.TileLayer(osmUrl, { maxZoom: 18, attribution: osmAttrib}).addTo(map);
```

در کد بالا ابتدا آدرس tile server تعریف شده است. در این مثال از سرویس OpenStreetMap برای تهیه‌ی Tileها استفاده شده است. سپس لینک سرویس دهنده، به همراه متن attribution(نوشته‌ای که در زیر نقشه قرار می‌گیرد) به شیء TileLayer پاس داده شد و شیء ایجاد شده از طریق متد addTo به شیء map اضافه شده است.

نتیجه‌ی کار در مرورگر اینگونه خواهد بود:

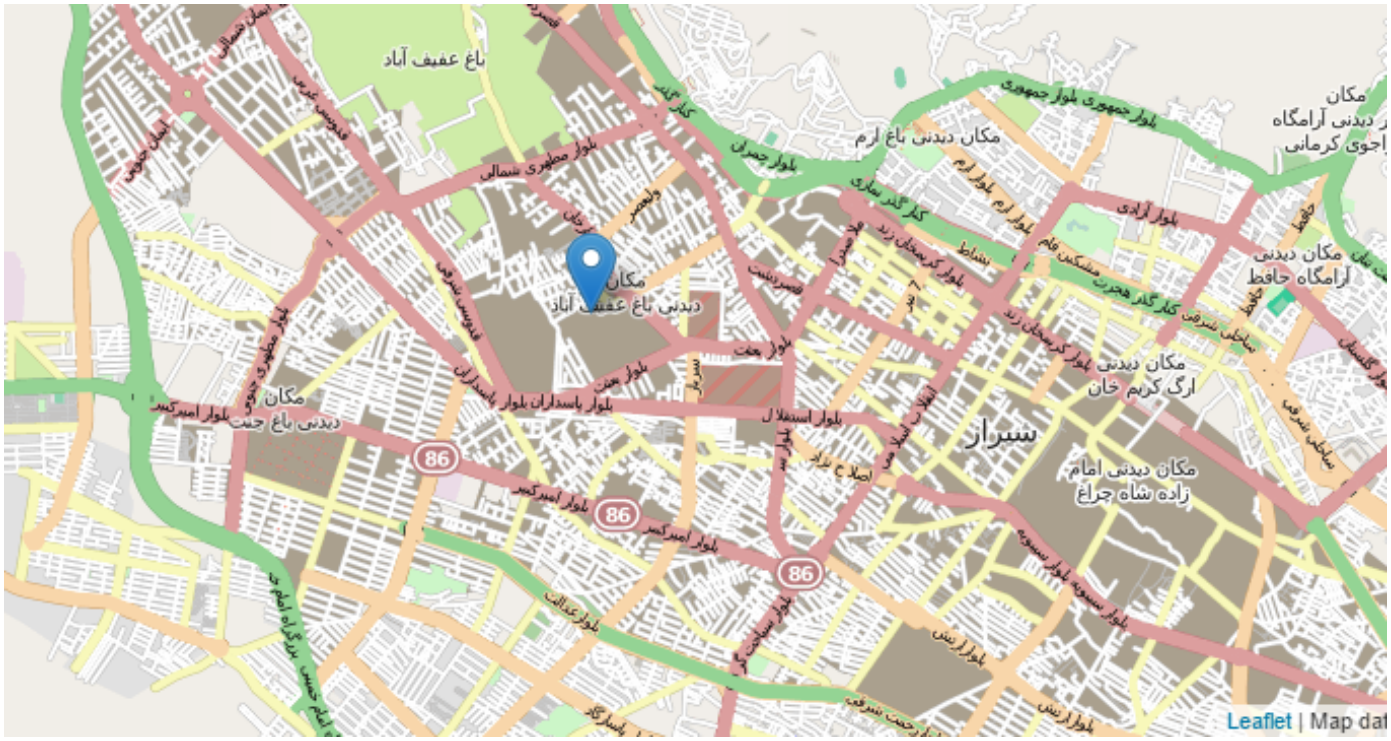


Marker، دایره و چندضلعی

در کنار نمایش Tileها می‌توان اشکال گرافیکی نیز به نقشه اضافه کرد؛ مثل مارکر(نقطه)، مستطیل، دایره و یا یک Popup. اضافه کردن یک Marker به سادگی، با کد زیر صورت می‌پذیرد:

```
var marker = L.marker([29.623116,52.497856]).addTo(map);
```

محل مورد نظر به شیء مارکر پاس داده شده و مقدار بازگشتی به map اضافه شده است.



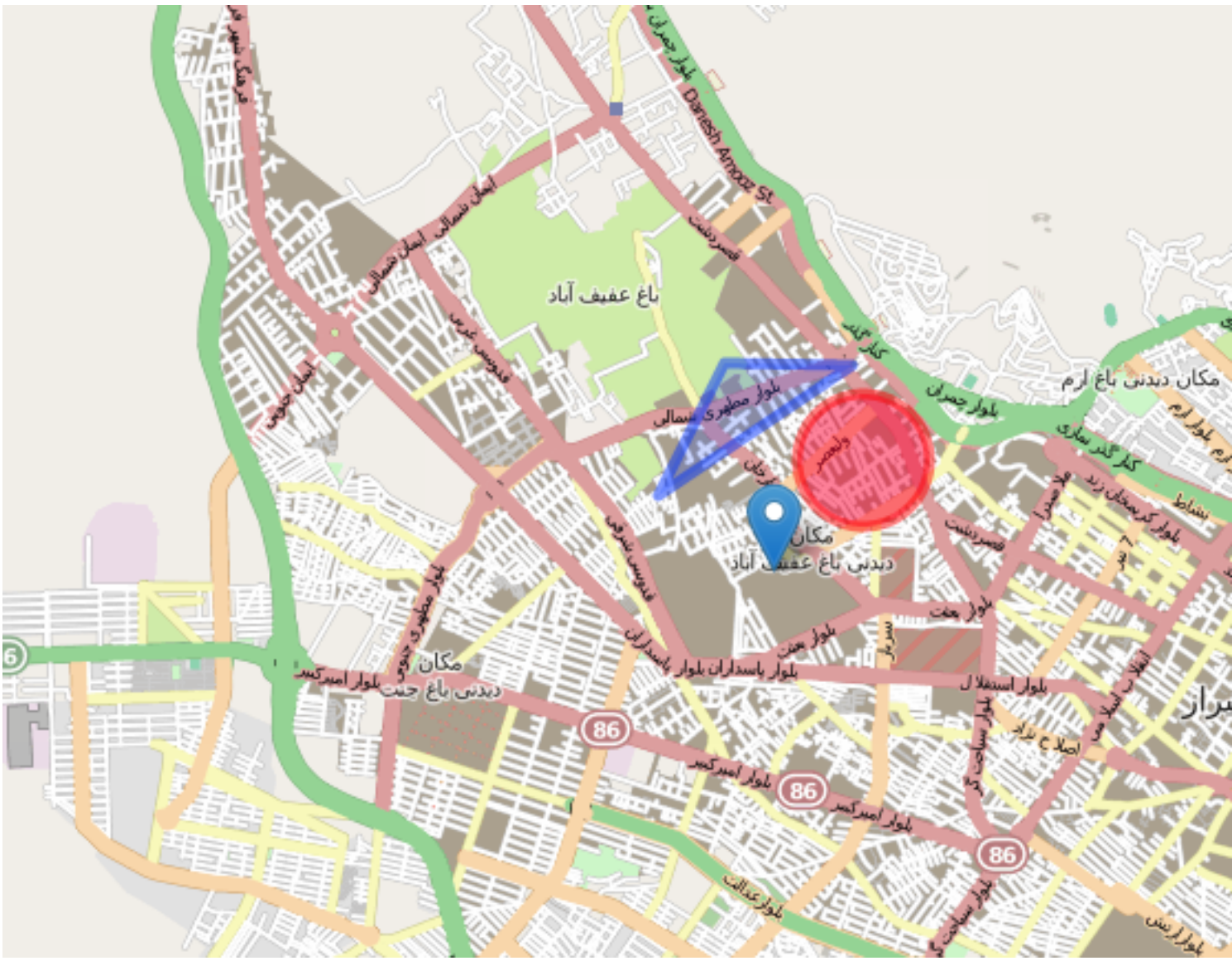
نمایش چند ضلعی و دایره هم کار ساده ای است. برای دایره باید ابتدا مختصات مرکز دایره و شعاع به متر را به L.circle پاس داد:

```
var circle = L.circle([29.6308217,52.5048021], 500, {
    color: 'red',
    fillColor: '#f03',
    fillOpacity: 0.5
}).addTo(map);
```

در کد بالا علاوه بر محل و اندازه دایره، رنگ محیط، رنگ داخل و شفافیت (opacity) نیز مشخص شده‌اند.

برای چند ضلعی‌ها می‌توان به این صورت عمل کرد:

```
var polygon = L.polygon([
[29.628453, 52.488838],
[29.637368, 52.493987],
[29.637168, 52.503987]
]).addTo(map);
```



کار کردن با Popup ها

از Popup می‌توان برای نمایش اطلاعات اضافه‌ای بر روی یک محل خاص یا یک عنوان به مانند Marker استفاده کرد. برای مثال می‌توان اطلاعاتی درباره‌ی محل یک Marker یا دایره نمایش داد. در هنگام ایجاد marker، دایره و چندضلعی مقادیر بازگشتی در متغیرهای جدایی ذخیره شدند. اکنون می‌توان به آن اشیاء یک popup اضافه کرد:

```
marker.bindPopup("باغ عفیف آباد").openPopup();
circle.bindPopup("I am a circle.");
polygon.bindPopup("I am a polygon.");
```

به علت اینکه openPopup برای Marker صدا زده شده، به صورت پیشفرض popup را نمایش می‌دهد. اما برای بقیه، نمایش با کلیک خواهد بود. البته الزاما نیازی نیست که popup روی یک شیء نمایش داده شود، می‌توان popupهای مستقلی نیز ایجاد کرد:

```
var popup = L.popup()
    .setLatLng([51.5, -0.09])
    .setContent("I am a standalone popup.")
    .openOn(map);
```