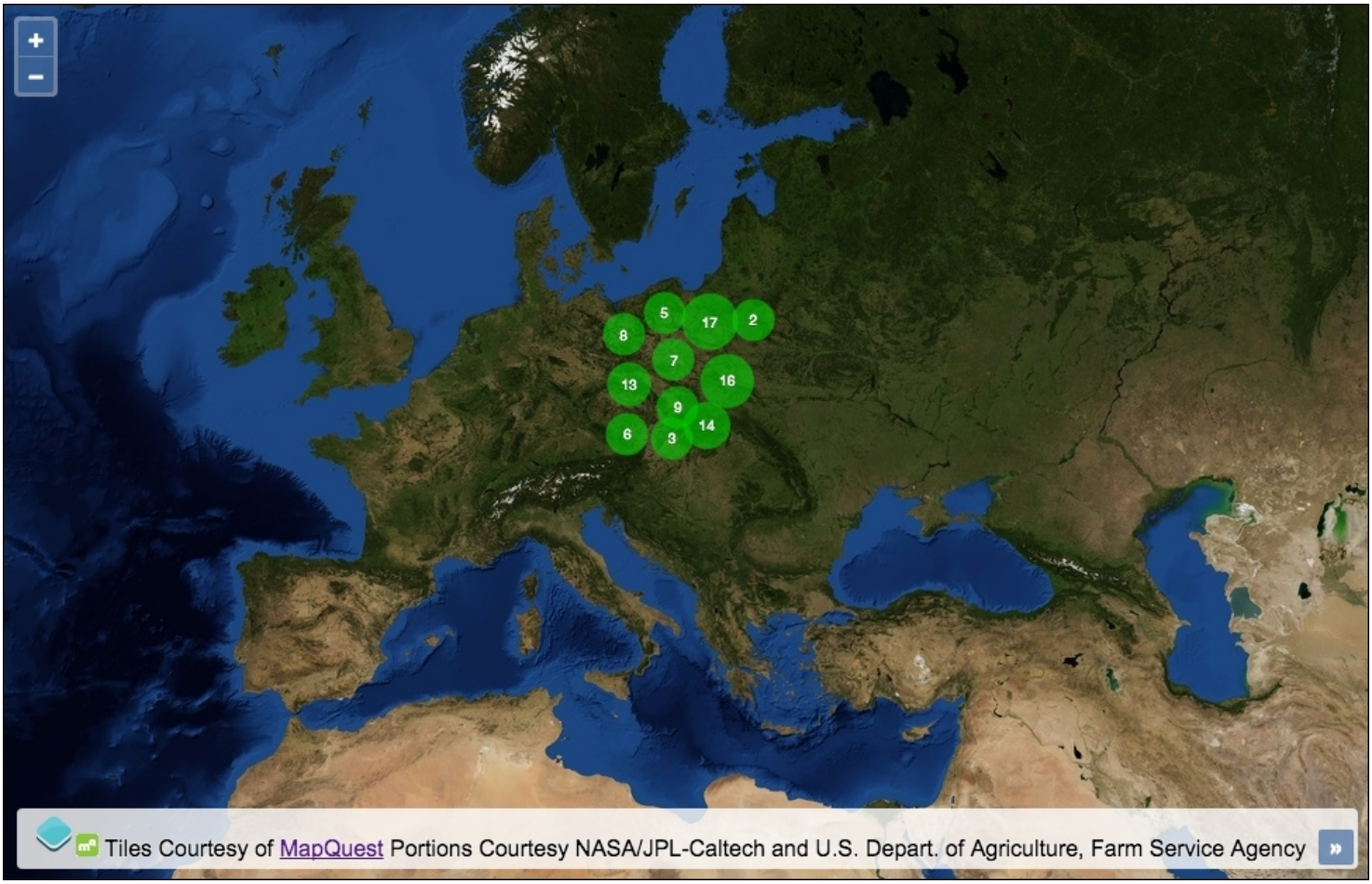
使用集群策略

想象一个场景，我们想要展示全球每个城市的所有加油站。当用户在地图上导航并设置缩放级别以查看整个世界时，会发生什么？展示了一个压倒性的优势点，它们在同一地方彼此重叠，为用户提供的视觉价值很小。

解决此问题的方法是在每个缩放级别上对功能进行聚类。OpenLayers使此操作非常容易实现。

此食谱（中的源代码 `ch03/ch03-clustering` ）显示了在向量层上应用聚类的容易程度，该向量层负责对要素进行分组，以避免出现与前面讨论的情况类似的情况，可以在以下屏幕截图中看到：



怎么做...

利用巨大的集群 OpenLayers使用以下说明的功能：

1 创建一个具有OpenLayers依赖项的HTML文件以及一个 `div` 用于保存地图的元素。

2 初始化 `map` ，如下所示：

复制

```
var map = new ol.Map({
  view: new ol.View({
    zoom: 4,
    center: [2152466, 5850795]
  }),
  target: 'js-map',
  layers: [
    new ol.layer.Tile({
      source: new ol.source.MapQuest({layer: 'sat'})
    })
  ]
});
```

3 随机生成一堆限制在特定范围内的点：

复制

```
var getRandomInt = function(min, max) {
  return Math.floor(Math.random() * (max - min + 1)) + min;
};

var features = [];
var numberOfFeatures = 0;

while(numberOfFeatures < 100) {
  var point = new ol.geom.Point([
    getRandomInt(1545862, 2568284),
    getRandomInt(6102732, 7154505)
  ]);

  features.push(new ol.Feature(point));
  numberOfFeatures++;
}
```

4 设置用于渲染群集的样式功能：

复制

```
var getStyle = function(feature) {
  var length = feature.get('features').length;
  return [
    new ol.style.Style({
      image: new ol.style.Circle({
        radius: Math.min(
          Math.max(length * 1.2, 15), 20
        ),
        fill: new ol.style.Fill({
          color: [0, 204, 0, 0.6]
        })
      }),
      text: new ol.style.Text({
        text: length.toString(),
        fill: new ol.style.Fill({
          color: 'white'
        }),
        stroke: new ol.style.Stroke({
          color: [0, 51, 0, 1],
```

5 创建 向量图层，添加聚类源，然后将该图层添加到地图：

复制

```
var vectorLayer = new ol.layer.Vector({
  source: new ol.source.Cluster({
    distance: 25,
    source: new ol.source.Vector({
      features: features
    })
  }),
  style: getStyle
});
map.addLayer(vectorLayer);
```

怎么运行的...

我们在整个欧洲的任意范围内随机放置了100个几何点。我们先前在将**功能导出为GeoJSON**配方中使用了此技术，在此我们详细解释了它的工作原理。如果您需要提醒，请继续阅读该食谱，我们将继续前进，并将注意力集中在如何实现聚类上。

`getStyle` 当集群需要渲染时，OpenLayers会调用我们的方法。它附加到 `style` 矢量层的属性。我们想要添加一个文本标签以显示集群下方的点数，并为集群提供一些非默认样式。我们在“**向几何点**配方**添加文本标签”中**看到了非常相似的样式以及附带的说明，因此，我们只看一下用于此食谱的样式的某些部分：

复制

```
image: new ol.style.Circle({
  radius: Math.min(
    Math.max(length * 1.2, 15), 20
  ),
```

圆圈将被绘制 表示点簇的地图。簇包含的点越多，圆的半径将越大。属于群集源的要素将应用自定义属性，即 `features` 。它包含该群集内所有功能的数组。我们使用 `ol.Feature` `get` 方法检索此值 `feature.get('features').length` ，并将其存储在变量中 `length` 。

我们将特征的长度乘以 `1.2` ，但是我们选择了乘法结果或 `15` 使用JavaScript `Math.max` 方法的最大数目。我们不希望任何半径小于的簇 `15` 。

相反，我们不希望任何簇的半径大于 `20` ，因此 `Math.min` 可用于包装最大半径的结果并在需要时限制半径。

复制

```
text: new ol.style.Text({
  text: length.toString(),
```

群集文本仅从群集数组中的要素数量中得出。该数字必须转换为OpenLayers的字符串类型。该 `toString` 方法是用于强制类型的本机JavaScript方法。

`ol.style.Style` 对象的其余部分仅在适用的情况下简单分配颜色，宽度和字体。

复制

```
var vectorLayer = new ol.layer.Vector({
  source: new ol.source.Cluster({
    distance: 25,
    source: new ol.source.Vector({
      features: features
    })
```

我们像往常一样创建一个矢量层，但是 `source` 是 `ol.source.Cluster` （扩展 `ol.source.Vector` 类的）实例。

`distance` 群集源的属性强制群集之间的最小距离（以像素为单位）。默认值为20。

`source` 群集源的属性是的实例 `ol.source.Vector` ，我们传入了之前的100个生成的特征。

每次缩放级别更改时，群集策略都会计算所有要素之间的距离，并将所有 符合同一群集某些参数的功能。

也可以看看

- 在**创建要素编程**食谱
- 在**从WFS服务器添加功能**食谱

上一节 (/book/web_development/9781785287756/3/ch03lvl1sec35/adding-features-from-a-wfs

下一节 (/book/web_development/9781785287756/3/ch03lvl1sec37/reading-features-directly-u