使用自定义OpenLayers库构建

在地方 在整本书中，我们都将OpenLayers提供的开箱即用的功能归功于它的能力，以帮助构建功能强大的Web映射应用程序。我们也有意识地意识到以下事实：装备精良的功能伴随着更大的下载大小。

但是，本食谱将向您展示如何构建OpenLayers的自定义精简版本，其中将仅包含库中满足特定应用程序要求的代码。

为了证明这一点，我们将创建一个自定义生成，它是由合适的运行从本书早期食谱：**创建一个简单的全屏幕地图**在第1章 (/book/web_development/9781785287756/1)，**Web制图基础知识。**

该食谱的源代码可以在中找到 `ch07/ch07-custom-openlayers-build` 。

做好准备

OpenLayers构建工具具有 需要在计算机上安装两个依赖项：Node.js（https://nodejs.org (https://nodejs.org)）和Java 1.7 SDK（http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html (http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html)）。Java的最新版本也应该可以。

我们将从命令行执行以下步骤。因此，这是通过Mac或Linux上的终端或通过Windows上的命令提示符来实现的。

怎么做...

1. 在命令行中，导航到Web应用程序代码的位置。对我们来说，这是里面 `ch07/ch07-custom-openlayers-build` ：

复制

```
cd ch07/ch07-custom-openlayers-build
```

2. 使用 `npm` 软件包管理器安装OpenLayers，它将包含所有必要的源文件以构建自定义版本，该自定义版本会 `node_modules/openlayers` 在安装时创建目录：

复制

```
npm install openlayers
```

3 创建JSON格式的构建配置文件（我们将称为 `build-config.json` ），该文件将概述我们自己的OpenLayers库构建的定制：

复制

```
{
  "exports": [
    "ol.Map",
    "ol.View",
    "ol.layer.Tile",
    "ol.source.OSM",
    "ol.control.defaults",
    "ol.Collection#extend",
    "ol.control.FullScreen"
  ],
  "compile": {
    "externs": [
      "externs/oli.js",
      "externs/olx.js"
    ],
    "define": [
      "goog.DEBUG=false",
      "ol.ENABLE_DOM=false",
      "ol.ENABLE_WEBGL=false",
```

4 最后，运行OpenLayers构建文件并传入我们的自定义构建配置文件和输出文件作为参数 `node` ：

复制

```
node node_modules/openlayers/tasks/build.js build -config.json custom-ol3-build.min.js
```

怎么运行的...

`npm` 默认情况下，程序包管理器与Node.js一起安装（此食谱的“**准备就绪**”部分中列出的依赖项之一）。当我们运行时 `npm install openlayers` ，这将从 `npm` 注册表（https://www.npmjs.com/package/openlayers (https://www.npmjs.com/package/openlayers)）下载最新版本的OpenLayers，并将其放置在 `node_modules/openlayers` 本地目录中。 `node_modules` 如果目录不存在，则会创建该目录。下载的OpenLayers包含运行自定义版本所需的所有源文件。

在解释刚刚创建的构建配置文件的内容之前，让我们再来看一下应用程序代码：

复制

```
var map = new ol.Map({
  view: new ol.View({
    center: [-15000, 6700000], zoom: 5
  }),
  layers: [
    new ol.layer.Tile({source: new ol.source.OSM()})
  ],
  controls: ol.control.defaults().extend([
    new ol.control.FullScreen()
  ]), target: 'js-map'
});
```

考虑到这一点，让我们将配置JSON分解为单独的部分进行解释，如下所示：

复制

```
"exports": [
  "ol.Map",
  "ol.View",
  "ol.layer.Tile",
  "ol.source.OSM",
  "ol.control.defaults",
  "ol.Collection#extend",
  "ol.control.FullScreen"
],
```

首先 JSON对象的属性，即 `exports` ，需要一个字符串数组。这些字符串指定您的应用程序代码使用的名称（符号）。这些字符串还可以使用 `*` 和定义模式 `#` 。例如， `"exports": ["*"]` 将包括所有内容。但是，您不想这样做！

通过观察JavaScript应用程序代码中使用的内容，前面的大多数导出操作都是不言自明的。但是，值得选择此列表中的一个特定条目，它是 `"ol.Collection#extend"` 。

为了将HTML5全屏控件添加到地图，我们扩展了默认控件列表，即 `ol.Collection` 类型列表。该 `ol.Collection` 班有一个叫方法 `extend` ，其中，在默认情况下，将不包括在我们构建即使我们有一个条目 `"ol.Collection"` ，因为该条目将只导出命名空间和构造，而不是该原型方法，如 `extend` 。

为了包括此方法，我们使用了 `#` 模式仅从中导出 `extend` 方法 `ol.Collection` 。如果我们需要所有方法，则可以执行 `"ol.Collection#*"` 。

复制

```
"compile": {
  "externs": [
    "externs/oli.js",
    "externs/olx.js"
  ],
```

接下来是 `compile` 属性，该属性接受除本示例中看到的属性和值以外的许多属性和值。这些选项适用于Google Closure编译器，它是此版本（与Java一起运行的编译器）背后的主力。

首先我们包含的属性是 `externs` ，用于正在编译的代码中使用的外部名称。OpenLayers记录必须包含 `oli.js` 和 `olx.js` 。您可以 `externs` 在 `node_modules/openlayers/externs` 目录内找到所有可用属性的列表。

复制

```
"define": [
  "goog.DEBUG=false",
  "ol.ENABLE_DOM=false",
  "ol.ENABLE_WEBGL=false",
  "ol.ENABLE_PROJ4JS=false",
  "ol.ENABLE_VECTOR=false",
  "ol.ENABLE_IMAGE=false"
],
```

该 `define` 属性列出了一些将在编译时使用的常量。这是自定义构建最终输出的机会。对于我们来说，我们会排除尽可能多的代码以满足我们的需求并减少文件的大小。

我们正在为地图使用默认的画布渲染器，因此我们排除了DOM和WebGL渲染器。我们不执行任何投影转换，因此我们也排除了Proj4js集成。我们也不使用矢量层或图像层，因此也将它们排除在外。还有更多可用的功能切换，我们鼓励您搜索OpenLayers的编译调试版本，以发现可能感兴趣的其他功能。

复制

```
"compilation_level": "ADVANCED",
"output_wrapper": ";(function(){%output%})();",
"manage_closure_dependencies": true
```

最后，我们为其他三个属性分配值。首先，使用Google Closure Compiler中最先进的编译器（这样可以使我们的代码更美观，更精简）。其次，指定包装已编译代码的JavaScript。第三，确保管理闭包依赖性（OpenLayers建议这样做）。

有了所有这些配置之后，就该触发构建了，如下所示：

复制

```
node node_modules/openlayers/tasks/build.js build-config.json custom -ol3-build.min.js
```

该行以开头 `node` ，这是用于执行 `build.js` 作为第一个参数传入的文件（）的程序 `node_modules/openlayers/tasks/build.js` 。

第二个参数指向我们的自定义构建配置文件（ `build-config.json` ），在该文件中我们指定了编译器的选项。

第三个参数用于编译的JavaScript的目标。如果该文件不存在，则会为我们创建。

运行可能需要一些时间，但是一旦完成构建，我们将拥有一个新文件， `custom-ol3-build.min.js` 我们可以从HTML文件中进行引用。

如果您采取看看完整的OpenLayers库（ `node_modules/openlayers/dist/ol.js` ）的正常大小，它的大小约为半兆字节。但是，我们的自定义版本重109 KB。这是一个相当小的文件大小，值得付出额外的努力，尤其是在支持移动设备时。

还有更多...

对于大多数自定义版本，对列出的前面的配置文件属性进行修改就足够了。但是，要完全控制编译器选项的范围触手可及，请访问https://github.com/openlayers/closure-util/blob/master/compiler-options.txt (https://github.com/openlayers/closure-util/blob/master/compiler-options.txt)了解更多信息。

OpenLayers还提供了一些有关配置文件和相关任务的良好文档。如果您想多读一些书以 `config` 更好地理解不同部分，请访问https://github.com/openlayers/ol3/blob/master/tasks/readme.md (https://github.com/openlayers/ol3/blob/master/tasks/readme.md)。

也可以看看

> 将**创建一个简单的全屏幕地图**在配方第1章 (/book/web_development/9781785287756/1)，**Web制图基础知识**

上一节 (/book/web_development/9781785287756/7/ch07lvl1sec64/transitioning-between-wea

下一节 (/book/web_development/9781785287756/7/ch07lvl1sec66/drawing-in-freehand-mode)