使用 ES6 JavaScript 和模块创建 AngularJS 应用程序

韩博孙

2021 年 3 月 9 日 麻省理工学院

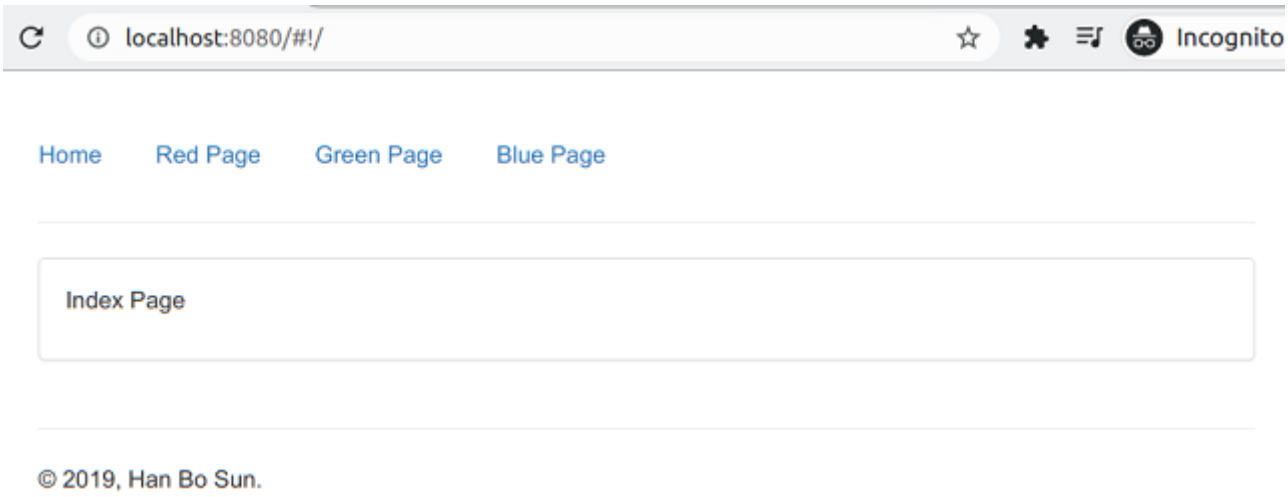
如何使用 ES6 JavaScript 和模块创建 AngularJS 应用程序

本教程将向读者介绍使用 ES6 JavaScript 和 Modules 创建 AngularJS 应用程序的方法。

下载源代码 - 490 KB

介绍

去年年底，我在思考未来。未来就在这里。但有时很难迈出第一步并拥抱未来。学习一项新技能，并将其用于新项目需要花费大量时间和精力。这就是我不愿意接*node.js 的原因.* 它支持的项目开发类型通常需要很长时间才能掌握。最有可能的是，我必须同时学习五种不同的东西，结果不会很好。学习会干扰发展。但我想继续前进并使用现代 JavaScript 语言进行开发。所以我想知道，是否可以使用基于 ES6 语法的 JavaScript 和 AngularJS 来开发应用程序，而没有 node.js 的麻烦？事实证明，这是可能的。本教程将讨论如何。



这就是我想要做的。我想创建一个单页的 AngularJS 应用程序，我想使用 ES6 基础 JavaScript 语法而不是 node.js。如前所述，这是可行的，而且很容易实现。我确实花了几个小时来研究。无论如何，本教程将引导您完成实现，并向您介绍如何使用 ES6 脚本语法编写 AngularJS。

示例应用程序架构

本教程的示例应用程序使用 Spring Boot 创建基于 Web 的应用程序。它所做的只是为用户提供静态内容，如 HTML 页面、样式表和 JavaScript 文件。它是一个单页应用程序，有四个可导航的子页面。第一个子页面是索引，它只是一个用于展示的静态页面。第二页

将显示`string`来自控制器的值，以及`click`由控制器对象的方法处理事件的按钮。其他两页`static`也仅用于显示目的。

整个想法是展示如何使用 ES6 脚本创建 JavaScript 对象。并且这个控制器（通俗地说）连接到视图（页面模板），并且可以在视图和控制器之间交换数据。此外，该应用程序将展示如何设置页面模板的导航。最后，将这两者结合到一个应用程序中。这些是 AngularJS 应用程序最重要的功能。如果这些可以用 ES6 脚本完成，那么可以使用相同的语法编写完整的应用程序。困难的部分是弄清楚如何。原来，这些都不难。所需要的只是一点点研究。

进入页面

我将从入口页面开始。我必须指出一个重要方面。为了在我的应用程序中使用 ES6 脚本语法，我需要将 JavaScript 文件添加为模块，而不是普通的脚本文件。让我告诉你区别。这是我添加到 HTML 页面的普通脚本文件：

HTML 复制代码

```
...
<script type="text/javascript" src="/assets/jquery/js/jquery.min.js"></script>
...
```

为了将 JavaScript 文件添加为模块，我必须这样做：

HTML 复制代码

```
...
<script type="module" src="/assets/app/js/TestController.js"></script>
...
```

这是必要的，因为根据文档，如果将文件添加为模块，Chrome 和 FireFox 可以执行 ES6 语法兼容的脚本。你去吧。这是使用基于 ES6 的语法的第一步。

让我向您展示这个条目页面的其余部分。我们称这些 AngularJS 应用程序为单页面应用程序的原因是因为只有一个页面，并且该页面的内容根据放置在该页面中的模板而变化。这个入口页面是在*src/main/resources/static/*子目录下的*index.html*文件。有一个非常基本的导航菜单：

HTML 复制代码

```
<div class="row">
   <div class="col-0xs-12">
      <ul class="nav nav-pills">
         <li role="presentation"><a href="./#!/">Home</a></li>
         <li role="presentation"><a href="./#!/red">Red Page</a></li>
         <li role="presentation"><a href="./#!/green">Green Page</a></li>
         <li role="presentation"><a href="./#!/blue">Blue Page</a></li>
      </ul>
   </div>
</div>
```

这是一个非常简单的导航菜单，只有 4 个链接。如果你是第一次使用 AngularJS，导航的工作方式是使用这样的链接：“*./#!/<angularjs-links>*”，即“*./#!/blue*”这些是 URL 重写规则AngularJS 路由器可以识别并正确呈现页面。

由于我们使用的是 AngularJS 路由器而不是花哨的 ui-router（与 AngularJS 一起工作的第三方组件），我们需要指定一个可以呈现子页面的区域。这是如何：

HTML 复制代码

```
<div class="row">
   <div class="col-xs-12">
      <div class="panel panel-default">
         <div class="panel-body">
            <ng-view></ng-view>
         </div>
      </div>
   </div>
</div>
```

页面将呈现为名为 的标记的子元素`<ng-view/>`。

每个 AngularJS 应用程序都有一个入口点，ngModule它被指定为ngApp. 这是在 HTML 页面顶部定义的，如下所示：

HTML 复制代码

```
<div class="container top-margin" ng-app="startup">
...
</div>
```

最后，我将 JavaScript 文件和模块添加到此网页。像这样：

HTML 复制代码

```
...
<script type="text/javascript" src="/assets/jquery/js/jquery.min.js"></script>
<script type="text/javascript" src="/assets/bootstrap/js/bootstrap.min.js"></script>
<script type="text/javascript" src="/assets/angularjs/1.7.5/angular.min.js"></script>
<script type="text/javascript" src="/assets/angularjs/1.7.5/angular-resource.min.js"></script>
<script type="text/javascript" src="/assets/angularjs/1.7.5/angular-route.min.js"></script>
<script type="module" src="/assets/app/js/app.js"></script>
...
```

前几行是为 AngularJS 源添加的 JavaScript 文件。最后一行是我的AngularJS应用入口的模块。我不必列出所有其他模块，因为该文件和后续模块文件都是相互关联的。它们相互导入并创建参考链接，因此无需在此处添加每个 JS 文件。

除了这个基本 HTML 文件之外，还有四个 HTML 文件，每个文件代表一个子页面。其中之一也称为*index.html*。它是默认显示的页面，其他页面以颜色命名，指定为“红色”的页面上有一些交互元素，以演示如何为页面创建控制器。另外两页是静态的。它们用于演示路由功能。

在下一节中，我将向您展示我的 JavaScript 模块代码是如何工作的。这包括应用程序入口、路由配置和测试控制器。

应用模块

每个 AngularJS 应用程序都必须定义一个模块，并指定为ngApp. 我已经展示了这是如何完成的。让我再给你看：

HTML 复制代码

```
<div class="container top-margin" ng-app="startup">
...
</div>
```

这个“ startup”ngModule在名为*app.js*的文件中定义。其内容如下所示：

JavaScript 复制代码

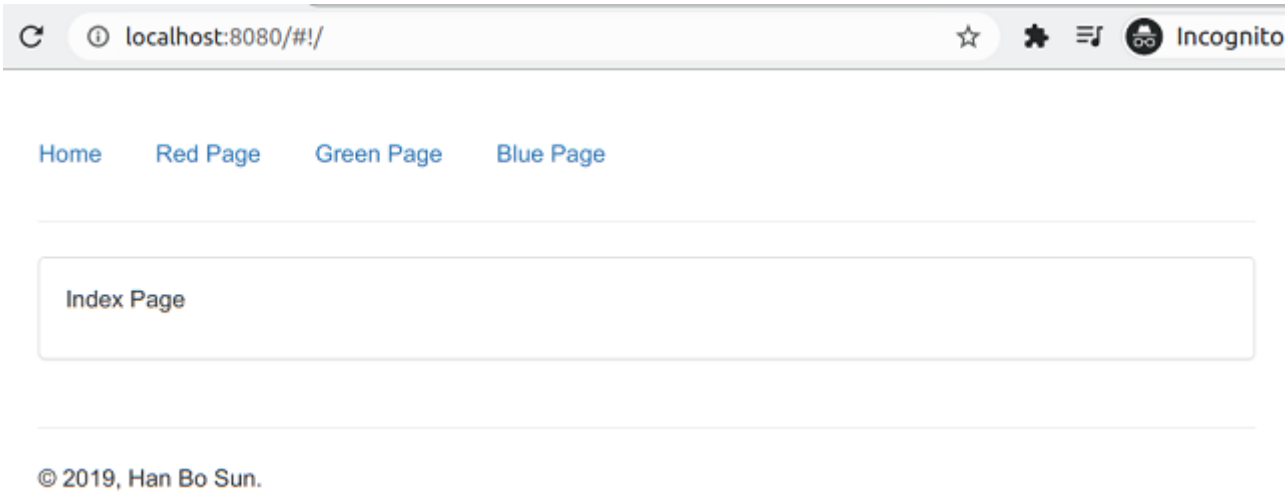
```
import { appRouting } from '/assets/app/js/app-routing.js';
import { TestController } from '/assets/app/js/TestController.js';

let app = angular.module('startup', ["ngRoute"]);
app.config(appRouting);
app.controller("TestController", TestController);
```

如您所见，这看起来与我以前编写的 JavaScript 代码有些不同。这个文件代表 ES6 意义上的一个模块。我使用新的 ES6 语法将对象（类对象或函数）从另一个模块导入到当前模块中。前两行是从其他模块中引入单个对象。第一行导入一个appRouting从模块“ */assets/app/js/app-routing.js* ”调用的函数。第二行导入一个类的对象TestController。最后三行将这些对象注入到 AngularJS 的启动调用过程中。第一行创建了一个名为“ app”的对象。它是一个 AngularJS 模块。该模块被命名为“ startup”。这是在ngAppHTML 页面上引用的名称。这 “app” 调用两个方法。第一个调用*.config()*。这是我传入函数的地方appRouting （从*app-routing.js*导入）。这基本上让 AngularJS 调用我的函数来设置子路径路由。最后一行正在让对象“ app”调用函数.controller()以注册名称为“ TestController”的控制器。这就像一个 IOC 容器，将对象添加为控制器，以便视图（HTML 页面）和 AngularJS 内部可以正确引用此控制器。有这个模块没有什么特别的，只有应用程序启动的配置。

接下来，我将向您展示该appRouting函数是如何定义的。为了为应用程序设置子页面导航，我需要从ngRoute模块中注入一个 AngularJS 组件，$routeProvider, 到我的appRouting()函数中。当我定义模块时，模块被注入到*app.js* 中app。这里是：

JavaScript 复制代码

```
...
let app = angular.module('startup', ["ngRoute"]);
...
```



我的“ appRouting”的整个源代码是这样的：

JavaScript 复制代码

```
export function appRouting($routeProvider) {
  $routeProvider.when("/", {
    templateUrl : "/assets/app/pages/index.html"
  });

  $routeProvider.when("/red", {
    templateUrl : "/assets/app/pages/page1.html",
    controller: "TestController",
    controllerAs: "vm"
  });

  $routeProvider.when("/green", {
    templateUrl : "/assets/app/pages/page2.html"
  });

  $routeProvider.when("/blue", {
    templateUrl : "/assets/app/pages/page3.html"
  });
}
```

除了依赖注入的工作原理之外，这个函数非常简单。我所做的就是创建一个名为appRouting. 它需要一个名为 的参数 $routeProvider。这个参数看起来有点搞笑。我是故意声明的，这就是依赖注入发生的地方。如果我按原样命名参数，则依赖项注入将正确发生。如果我使用不同的名称，null第一次使用该参数时会发生引用异常。

我调用方法$routeProvider.when()来设置子页面导航。此方法将一个对象作为参数。该对象的属性决定了子页面的行为方式。有四个调用，用于设置四个不同的子页面。他们三个长得很像。所有对象只包含一个名为“ templateUrl”的属性。我在这里做的是指定 AngularJS 将这三个页面显示为静态页面。还有另一个调用，参数包含三个属性，“ templateUrl”、“ controller”和“ controllerAs”。属性“ templateUrl”指定模板页面文件位置。属性“ controller”指定此页面的 AngularJS 控制器的名称。该物业“controllerAs" 指定控制器在模板页面上可以引用的别名。最新版本的AngularJS首选使用别名来访问控制器类的模型属性。如果你不想使用这个，那么另一种选择是使用“ $scope”来访问控制器的模型属性。这不再是首选。

该export关键字用于导出的功能，以便其他JavaScript模块可以进口。既然讨论了子页面导航配置，接下来，我们将讨论 TestController.

对于这个应用程序，我只需要创建一个非常简单的应用程序，其中一个页面与一个控制器相关联。控制器应该有一些属性是模型属性（可以在页面上使用），可以在页面上操作。然后将有用户可以按下并查看页面上的反应的按钮。这些应该足以证明使用 ES6 语法设计控制器的想法。

对于这个控制器，我将设计一个简单的加法计算器。将有两个输入字段，用户可以输入整数值。然后还有另一个输入字段将显示两个输入值的总和。如果用户输入了一些无效值，则显示总和的输入字段将重置为`null`，然后将显示警告。要进行输入验证并计算总和，用户将单击名为"**计算**"的按钮。

这是控制器的完整源代码：

JavaScript 缩小▲ 复制代码

```
export class TestController {

  constructor() {
    this._value1 = "";
    this._value2 = "";
    this._sumValue = null;
  }

  get value1 () {
     return this._value1;
  }

  set value1 (val) {
     this._value1 = val;
  }

  get value2 () {
     return this._value2;
  }

  set value2 (val) {
     this._value2 = val;
  }

  get sumValue () {
     return this._sumValue;
  }

  set sumValue (val) {
     this._sumValue = val;
  }

  clickIt() {
     let intVal1 = parseInt(this.value1);
     let intVal2 = parseInt(this.value2);

     if (isNaN(intVal1) || isNaN(intVal2)) {
        alert("The integer values entered is/are invalid. Please correct.");
        this.sumValue = null;
        return;
     }

     this.sumValue = intVal1 + intVal2;
  }
}
```

这个类非常简单。它具有三个属性和一种`public`方法。让我告诉你它们是什么。首先，这些是属性，它们是在这个类的构造函数中声明和初始化的。

JavaScript 复制代码

```
...
  constructor() {
    this._value1 = "";
    this._value2 = "";
    this._sumValue = null;
  }
...
```

公开这些属性的一个好方法是通过 getter 和 setter（也称为访问器）。这些 getter 和 setter 是：

JavaScript 复制代码

```
...
  get value1 () {
     return this._value1;
  }

  set value1 (val) {
     this._value1 = val;
  }

  get value2 () {
     return this._value2;
  }

  set value2 (val) {
     this._value2 = val;
  }

  get sumValue () {
     return this._sumValue;
  }

  set sumValue (val) {
     this._sumValue = val;
  }
...
```

最后public这个类的方法就是处理视图上的按钮点击事件。这里是：

JavaScript 复制代码

```
...
  clickIt() {
     let intVal1 = parseInt(this.value1);
     let intVal2 = parseInt(this.value2);

     if (isNaN(intVal1) || isNaN(intVal2)) {
        alert("The integer values entered is/are invalid. Please correct.");
        this.sumValue = null;
        return;
     }

     this.sumValue = intVal1 + intVal2;
  }
...
```

仔细看看这个方法。它首先尝试将文本字段值解析为整数。然后它检查解析的值，确保它们是有效的数字。如果它们不是有效数字，则会显示一个警告弹出窗口，总和值将设置为空，并且不会执行进一步的操作。如果值有效，将计算两者的总和并将其分配给sumValue属性。成功分配值后，总和的文本字段将立即更新。

如图所示，类本身很简单。那么，这个 AngularJS 示例应用程序中的视图如何使用它呢？原来，使用也很简单。这是查看页面的源代码：

HTML 复制代码

```
<div class="row">
   <div class="col-xs-12 col-sm-offset-2 col-sm-8 col-md-offset-3 col-md-6">
     <p><input ng-model="vm.value1"> + <input ng-model="vm.value2"> =
     <input ng-model="vm.sumValue" readonly> <button class="btn btn-primary"
     ng-click="vm.clickIt()">Calculate</button></p>
   </div>
</div>
<div class="row">
   <div class="col-xs-3" style="background-color: red;">

   </div>
</div>
```

这些属性的引用方式与旧方式中的引用方式相同。我已将控制器对象指定为“ vm”。所以这些属性可以在视图中作为“ vm.<propertyName>”引用。这是属性如何绑定到 Value #1、Value #2 和 Sum 的文本字段。

HTML 复制代码

```
...
<input ng-model="vm.value1"> + <input ng-model="vm.value2"> =
<input ng-model="vm.sumValue" readonly>
...
```

这里的标记是设置值 #1 加值 2 的显示，以获得两个值的总和。还有一个按钮，可以点击它来计算两个值的总和。一旦点击，这两个值将被转换为整数，然后将这两个值之和存储在属性中sumValue。这将导致显示总和的文本字段。该按钮定义如下：

HTML 复制代码

```
...
<button class="btn btn-primary" ng-click="vm.clickIt()">Calculate</button>
...
```

如何测试示例应用程序

下载示例源代码后，请解压到安全位置。然后将所有 JavaScript 文件从*.sj重命名为*.js。然后就可以编译打包了。需要注意的是，从今年年初开始，我已经从 Java 8 切换到 Java 14。请要么使用 Java 14，要么修改 POM 文件，使项目可以用 Java 8 编译。

使用以下命令编译和打包应用程序：

复制代码

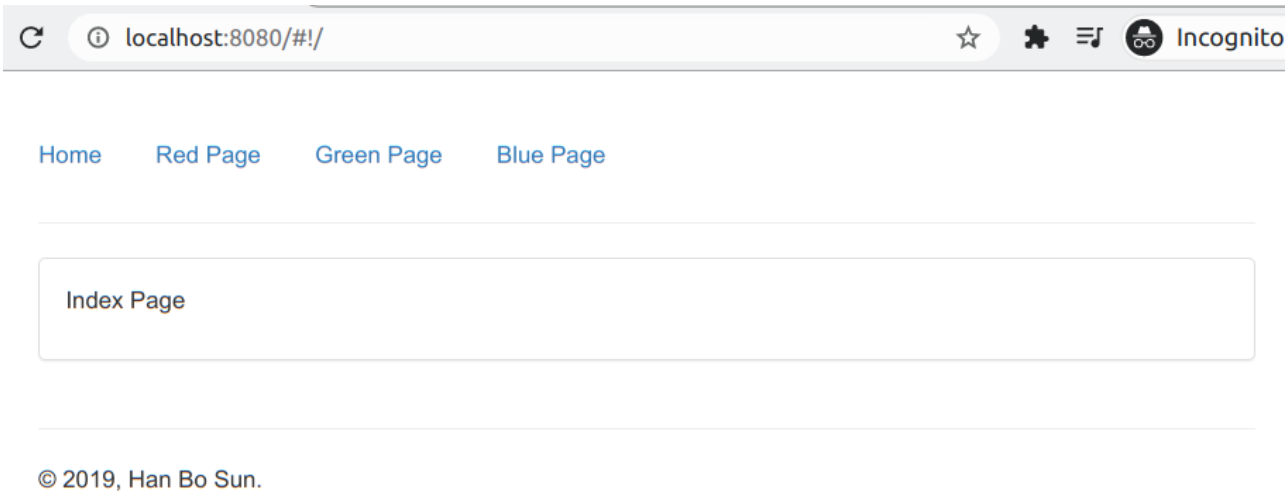
```
mvn clean install
```

编译打包成功后，使用以下命令启动应用程序：

复制代码

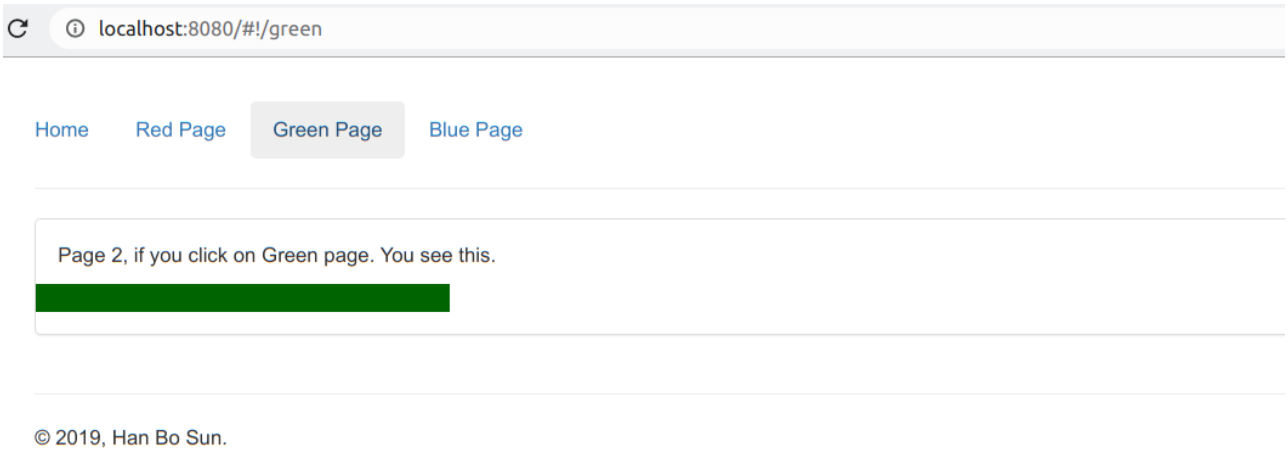
```
java -jar target/hanbo-angular-es6sample2-1.0.1.jar
```

应用程序成功启动后，您可以使用以下 URL 在浏览器中运行该应用程序。请使用 Chrome 或 Firefox 运行此应用程序。我相信这也可以在基于 WebKit 的 Internet Explorer 中运行。这从未被测试过。页面显示后，您将看到与第一个屏幕截图相同的页面显示。让我在这里再次张贴这个screehshot：

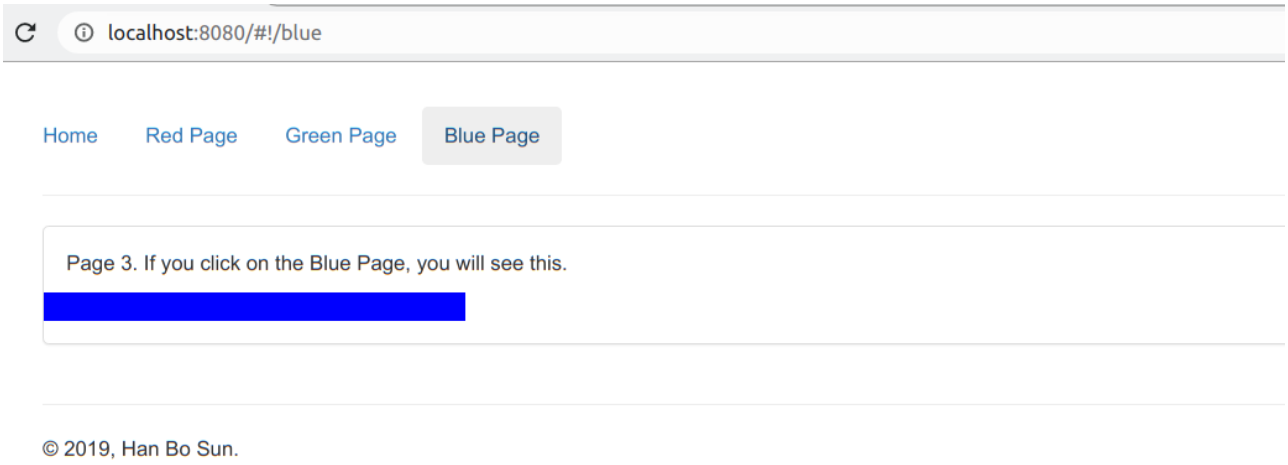


屏幕截图显示了此应用程序的索引子页面。还有三个页面，“**红页**”、“**绿页**”和“**蓝页**”。“**绿页**”和“**蓝页**”就像索引页，它们是静态的。“**红页**”**页面**是具有加法计算逻辑的页面。这是“**红页**”的截图：

这是"**绿页**"的截图：



这是"**蓝页**"的截图：



就是这个。玩得开心！

概括

另一个教程完成。我玩得很开心。这是一个简单的网络应用程序。我设计它的目的是展示 ES6 的新语法，并集成了 AngularJS。对于这个应用程序，我有两个重点，一是弄清楚如何设置导航路由，二是弄清楚如何使用新语法配置和使用 AngularJS 控制器。经过一些研究，两个问题都得到了解决，正如我在本教程中所展示的那样。

但是，可以从本教程中衍生出更多教程。最常用的是 AngularJS 中的工厂，用于创建服务对象。我可能会用这个做一个小教程。使用 AngularJS 创建指令/组件也是经常使用的，这可以是另一个教程。我想我可以使用 ES6 模块将两者结合到一个教程中，该教程将于今年晚些时候推出。不管怎样，从现在到今年年底，我会发布很多好的教程，请查看我的 CodeProject 个人资料页面以获取这些新教程。感谢您的阅读！

历史

- 2021 年 3 月9日：初稿

执照

分享

关于作者

韩博孙

组长 The Judge Group

美国

手表该会员

没有提供传记

评论和讨论

添加评论或问题

电子邮件提醒

Search Comments

第一 页上一页 下一页

弹簧靴?

BillWoodruff 4-Apr-21 11:39

刷新 1

一般 新闻 建议 问题 错误 答案 笑话 赞美 咆哮 管理员

使用Ctrl+Left/Right 切换消息，Ctrl+Up/Down 切换主题，Ctrl+Shift+Left/Right 切换页面。