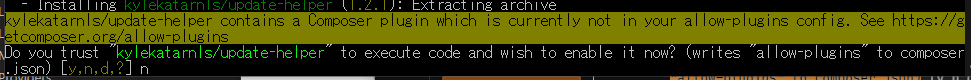
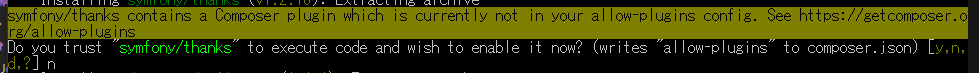
创建项目

Laravel 最新版本是 5.6，但是 5.5 是长期支持版本所以这里依旧选择使用 5.5





# blog 是项目的名称，可自行修改

composer create-project --prefer-dist laravel/laravel blog "5.5.\*"

运行项目

进入刚创建的项目，执行 php artisan serve ,访问 127.0.0.1:8000

Vue

扩展包

package.json 配置

初次接触 Vue 的同学，通常对于要安装什么扩展包感到迷茫。以下是 Laravel package.json 中默认的扩展包：

|  |
| --- |
| "devDependencies": {  "axios": "^0.17",  "bootstrap-sass": "^3.3.7",  "cross-env": "^5.1",  "jquery": "^3.2",  "laravel-mix": "^1.0",  "lodash": "^4.17.4",  "vue": "^2.5.7"  } |

bootstrap-sass 提供 sass 编写的 bootstrap 支持，因为我们要用 iview 所以这个扩展可以删除。

直接与 Vue 相关的有 vue 和 axios（网络请求工具），如果开发中、大型项目，vue-router（路由管理工具） 和 vuex（状态管理工具） 也必不可少。

还有其他常用的扩展，例如支持 cookie 操作的 js-cookie，支持多国语言开发的 vue-i18n,支持 Sass 语法的 node-sass 等。

此项目前端使用 iview 框架，加上这些常用扩展后的 package.json ：

|  |
| --- |
| "devDependencies": {  "axios": "^0.17",  "cross-env": "^5.1",  "jquery": "^3.2",  "laravel-mix": "^2.0",  "lodash": "^4.17.4",  "node-sass": "^4.7.2",  "vue": "^2.5.7"  },  "dependencies": {  "css-loader": "^0.28.9",  "iview": "^2.9.2",  "js-cookie": "^2.2.0",  "less": "^3.0.0",  "less-loader": "^4.0.5",  "particles.js": "^2.0.0",  "vue-i18n": "^7.4.2",  "vue-router": "^3.0.1",  "vuex": "^3.0.1"  } |

其中还加入了 css加载器、less解析工具、particles 前端动画等，不需要的可以自行删除。

安装扩展包

将以上配置直接粘贴到你的 package.json 文件中，执行 npm install

如果想单个安装扩展，可以执行命令如下：

cnpm install vue-router --save

前端结构搭建

vue前端的文件结构在 /recources/assets/js 目录，vue能够渲染主要是两个操作

vue 挂载到页面的节点中

vue-router 处理路由，渲染对应的组件

1. 建立 html 文件

在 resource/views 新建 index.blade.php,代码如下

|  |
| --- |
| <!DOCTYPE html>  <html style="height: 100%">  <head>  <meta charset="utf-8">  <meta name="viewport" content="width=device-width,initial-scale=1,user-scalable=0">  <meta name="csrf-token" content="{{ csrf\_token() }}">  <title>laravel-vue-iview项目</title>  </head>  <body style="height: 100%">  <div id="app">  </div>  <script src="{{ mix('js/app.js') }}"></script>  </body>  </html> |

2. 修改 /recources/assets/js.app.js

|  |
| --- |
| require('./bootstrap');  // 导入扩展包  window.Vue = require('vue');  import App from './app.vue'  import VueRouter from 'vue-router';  import iView from 'iview';  import 'iview/dist/styles/iview.css';  // 导入vue  Vue.use(iView);  Vue.use(VueRouter);  // 路由配置  const RouterConfig = {  routes: [  // ExampleComponent laravel默认的示例组件  { path: '/', component: require('./components/ExampleComponent.vue') },  ]  };  const router = new VueRouter(RouterConfig);  const app = new Vue({  el: '#app',  router: router,  render: h => h(App)  }); |

3. 建立 Layout

在 /recources/assets/js 新建 app.vue, 将 iview 的 layout 代码搬过来：

|  |
| --- |
| <style scoped>  .layout-con{  height: 100%;  width: 100%;  }  .menu-item span{  display: inline-block;  overflow: hidden;  width: 69px;  text-overflow: ellipsis;  white-space: nowrap;  vertical-align: bottom;  transition: width .2s ease .2s;  }  .menu-item i{  transform: translateX(0px);  transition: font-size .2s ease, transform .2s ease;  vertical-align: middle;  font-size: 16px;  }  .collapsed-menu span{  width: 0px;  transition: width .2s ease;  }  .collapsed-menu i{  transform: translateX(5px);  transition: font-size .2s ease .2s, transform .2s ease .2s;  vertical-align: middle;  font-size: 22px;  }  </style>  <template>  <div class="layout">  <Layout :style="{minHeight: '100vh'}">  <Sider collapsible :collapsed-width="78" v-model="isCollapsed">  <Menu active-name="1-2" theme="dark" width="auto" :class="menuitemClasses">  <MenuItem name="1-1">  <Icon type="ios-navigate"></Icon>  <span>Option 1</span>  </MenuItem>  <MenuItem name="1-2">  <Icon type="search"></Icon>  <span>Option 2</span>  </MenuItem>  <MenuItem name="1-3">  <Icon type="settings"></Icon>  <span>Option 3</span>  </MenuItem>  </Menu>  </Sider>  <Layout>  <Header :style="{background: '#fff', boxShadow: '0 2px 3px 2px rgba(0,0,0,.1)'}"></Header>  <Content :style="{padding: '0 16px 16px'}">  <Breadcrumb :style="{margin: '16px 0'}">  <BreadcrumbItem>Home</BreadcrumbItem>  <BreadcrumbItem>Components</BreadcrumbItem>  <BreadcrumbItem>Layout</BreadcrumbItem>  </Breadcrumb>  <Card>  <div style="height: 600px">Content</div>  </Card>  </Content>  </Layout>  </Layout>  </div>  </template>  <script>  export default {  data () {  return {  isCollapsed: false  };  },  computed: {  menuitemClasses: function () {  return [  'menu-item',  this.isCollapsed ? 'collapsed-menu' : ''  ]  }  }  }  </script> |

4. 添加组件

路由访问的组件会渲染到 <router-view></router-view>, 所以我们修改上面的 app.vue，将 content 文字修改为 <router-view></router-view>，上面的代码是已经修改过得。

然后修改组件 /recources/assets/js/components/ExampleComponent.vue 的内容：

|  |
| --- |
| <template>  <Tabs>  <TabPane label="macOS" icon="social-apple">标签一的内容</TabPane>  <TabPane label="Windows" icon="social-windows">标签二的内容</TabPane>  <TabPane label="Linux" icon="social-tux">标签三的内容</TabPane>  </Tabs>  </template> |

5. 创建 web 路由

在 /routes/web.php 中新建一个访问我们挂载着vue页面的路由。

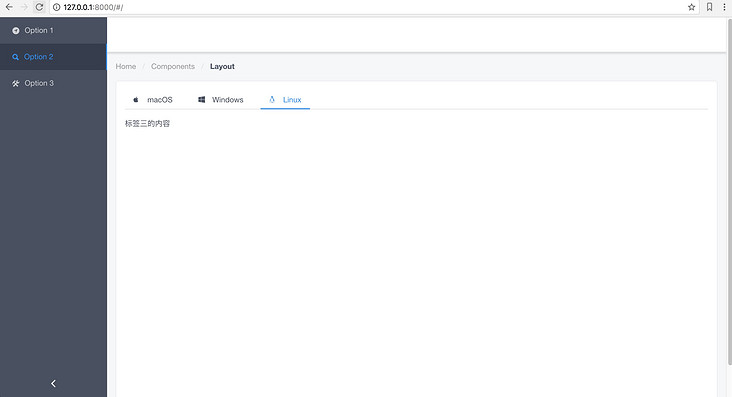
|  |
| --- |
| Route::get('/', function () {  return view('index');  }); |

6. 编译运行

常用的编译命令如下：

|  |
| --- |
| # 本地环境编译  npm run dev  # 本地环境编译 + 监控文件修改  npm run watch  # 生产环境编译  npm run prod |

这里我使用 npm run dev，编译成功后访问项目就可以看到我们创建的 vue 页面了。



至此，只是完成了基础的搭建过程，一个完整的项目还需要合理的 项目结构 和 基础的权限管理 等，这些内容也会逐步在此项目中完善。最终的目的是为了开发新项目时，可以 拿来就用,而不是每次重新再配置一遍。