Aurelia快速入门

## 准备环境

Aurelia是一个了不起的框架，它包含简单而干净的代码，而不会牺牲功能

全局安装Aurelia CLI本身：npm install -g aurelia-cli

*当安装Windows的Git，有一个选项use git bash only，run git from windows command prompt或run git and included unix tools from windows command prompt。您必须选择run git from windows command prompt或run git and included unix tools from windows command prompt才能使Aurelia CLI正常工作*

## 创建一个新的Aurelia项目

要创建项目，请从命令行运行au new。您将看到许多选项。将项目命名为“ todo”，然后根据最适合您的情况选择“ Default ESNext”或“ Default TypeScript”选项。

一旦安装了依赖项（需要几分钟），您的项目就可以开始了。只需将目录更改为项目文件夹，然后输入即可运行它au run --open。这将运行该应用程序，打开一个浏览器选项卡，并查看项目源以进行更改。如果一切设置正确，您应该会看到消息“ Hello World！“在浏览器中。

## 基本原理

src/app.js

{code}

export class App {

constructor() {

this.heading = 'Todos';

this.todos = [];

this.todoDescription = '';

}

addTodo() {

if (this.todoDescription) {

this.todos.push({

description: this.todoDescription,

done: false

});

this.todoDescription = '';

}

}

removeTodo(todo) {

let index = this.todos.indexOf(todo);

if (index !== -1) {

this.todos.splice(index, 1);

}

}

}

{code}

## 渲染应用

有src/main.js文件，该文件是程序的起点，它通常使用src/app.js由默认根UI组件

为了渲染，我们需要为app组件创建一个视图。这介绍了Aurelia的下一个约定。要为任何类创建视图，只需创建一个与ES Next模块同名的HTML文件，但将文件扩展名更改为.html。在该视图内部，可以放置一个带有数据绑定表达式的HTML 5模板，该模板声明该视图应如何呈现该类的实例

src/app.html

{code}

<template>

<h1>${heading}</h1>

</template

{code}

首先，所有视图都包装在Web Components template元素中.

其次，您注意到${heading}语法了吗？好吧，在视图内部，您可以访问与该视图关联的类实例的所有属性和方法，并且可以使用上述ES Next模板字符串语法在任何元素或属性的内容内部访问它们。上面的语法创建到heading属性的单向数据绑定。“单向”是指数据流是单向的，只有对heading属性的更改才会影响视图。从视图返回视图模型没有“反向”流

### 演示模式

*我们将View的关联类称为*View-Model，*因为它是*View的模型*或模型。大多数Aurelia开发都利用“模型-视图-视图模型”模式。但是，Aurelia足够灵活，可以根据需要使用诸如监督控制器，被动视图和模型-视图-控制器之类的模式*

### 绑定命令

*Aurelia将对所有表单控制值绑定使用双向绑定，对所有其他内容使用单向绑定。但是，您始终可以使用显式绑定命令来覆盖它。例如，而不是使用的.bind就可以使用.one-way，.two-way或.one-time。同样，您可以使用.delegate事件委派代替.trigger*

### 样式和css属性

*如果要支持这些浏览器，则在进行字符串插值时 请使用style属性的别名，css以确保您的应用程序与Internet Explorer和Edge兼容*

## 构建应用程序外壳

要将路由添加到您的应用程序，您要做的就是configureRouter在App类中添加一个方法。框架将调用此方法，并将其传递给a RouterConfiguration和a Router。您可以使用配置对象来获取带有所需路由的路由器设置。使用该map方法将路由模式映射到应处理模式的模块。最少，每条路线至少需要一个route模式和一个moduleId。

配置要点：

1. 注册两条路线。第一条路线为空，以表示route: ''。这是没有片段时匹配的默认路由。第二条路线具有模式contacts/:id。这将与文字匹配，contacts/后跟一个我们称为的参数id。当此路由匹配时，路由器将加载contact-detail模块，以便我们可以显示所选的联系人
2. 设置了config.title属性。这将设置基本的“标题”，以用于浏览器的文档标题中。我们还可以title在每个路线上设置一个
3. 其次，config.options.pushState = true; config.options.root = '/';是可选的。他们将打开基于[pushState的路由](https://aurelia.io/docs/routing/configuration" \l "options)。如果删除这两行，Aurelia路由器将使用基于默认哈希的路由，该路由甚至可以在IE9上使用。路由选项示例：
4. 基于pushState： <http://localhost:8080/contacts/1>
5. 基于哈希： <http://localhost:8080/#/contacts/1>

*我们使用虚线命名约定将单词*contact-list*分隔为我们的自定义元素名称。该类的名称应使用UpperCamelCase版本*ContactList*定义*

## 添加发布/订阅消息

## 添加加载指示器

每当我们在屏幕之间导航或发出WebAPI请求时，让我们在应用程序顶部显示加载指示器。为此，我们将使用3rd party库并创建一个自定义的Aurelia元素来包装它

1. 使用 nprogress使用以下命令安装库
2. npm install nprogress