عنوان: **ساخت یک بلاگ ساده با Ember.js، قسمت اول**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۶ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, SPA, EmberJS

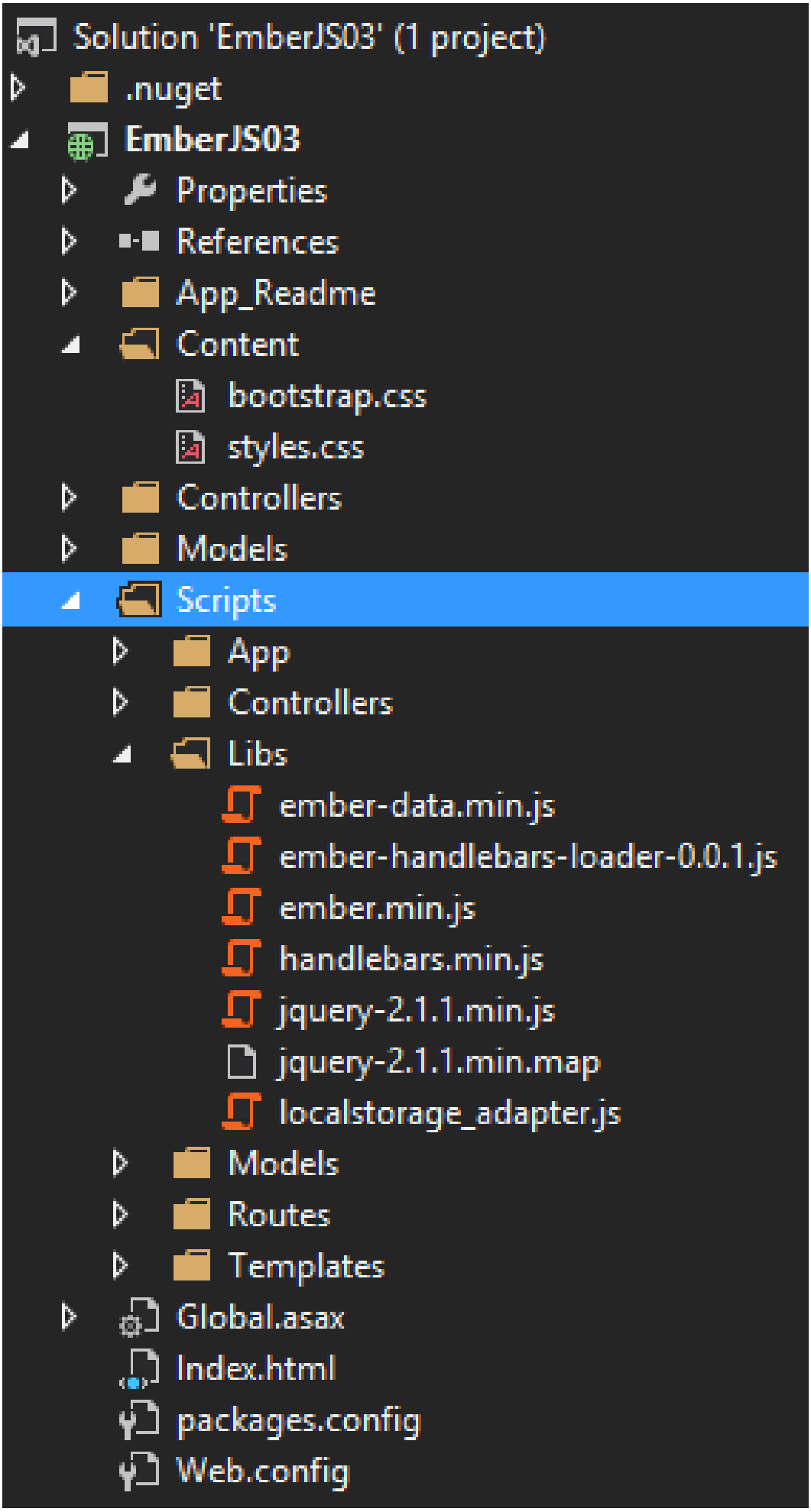
پس از آشنایی مقدماتی با اجزای مهم تشکیل دهنده‌ی Ember.js (^ و ^)، بهتر است این دانسته‌ها را جهت تکمیل یک پروژه‌ی ساده‌ی تک صفحه‌ای بلاگ، بکار بگیریم.

در این بلاگ می‌توان:

- یک مطلب جدید را ارسال کرد.
- مطالب قابل ویرایش و یا حذف هستند.
- مطالب بلاگ قسمت ارسال نظرات دارند.
- امکان گزارشگیری از آخرین نظرات ارسالی وجود دارد.
- سایت صفحات درباره و تماس با ما را نیز دارا است.

ساختار پوشه‌های برنامه

در تصویر ذیل، ساختار پوشه‌های برنامه بلاگ را ملاحظه می‌کنید. چون قسمت سمت کلاینت این برنامه کاملا جاوا اسکریپتی است، پوشه‌های Routes، Models، Libs، Controllers، App و Templates آن در پوشه‌ی Scripts تعریف شده‌اند و به این ترتیب می‌توان تفکیک بهتری را بین اجزای تشکیل دهنده‌ی یک برنامه‌ی تک صفحه‌ای وب Emeber.js پدید آورد.


Solution 'EmberJS03' (1 project)
.nuget
EmberJS03
Properties
References
App_Readme
Content
bootstrap.css
styles.css
Controllers
Models
Scripts
App
Controllers
Libs
ember-data.min.js
ember-handlebars-loader-0.0.1.js
ember.min.js
handlebars.min.js
jquery-2.1.1.min.js
jquery-2.1.1.min.map
localstorage_adapter.js
Models
Routes
Templates
Global.asax
Index.html
packages.config
Web.config

فایل CSS بوت استرپ نیز به پوشه‌ی Content اضافه شده‌است.

دریافت پیشنیازهای سمت کاربر برنامه

در ساختار پوشه‌های فوق، از پوشه‌ی Libs برای قرار دادن کتابخانه‌های پایه برنامه مانند jQuery و Ember.js استفاده خواهیم کرد. به این ترتیب:
- نیاز به آخرین نگارش‌های Ember.js و همچنین افزونه‌ی Ember-Data آن برای کار ساده‌تر با داده‌ها و سرور وجود دارد. این فایل‌ها را از آدرس ذیل می‌توانید دریافت کنید (نسخه‌های نیوگت به دلیل قدیمی بودن و به روز نشدن مداوم آن‌ها توصیه نمی‌شوند):

http://emberjs.com/builds/#/beta

برای حالت آزمایش برنامه، استفاده از فایل‌های دیباگ آن توصیه می‌شوند (فایل‌هایی با نام اصلی و بدون پسوند prod یا min). زیرا این فایل‌ها خطاها و اطلاعات بسیار مفصلی را از اشکالات رخ داده، در کنسول وب مرورگرها، فایرباگ و یا Developer tools آن‌ها نمایش می‌دهند. نسخه‌ی min برای حالت ارائه‌ی نهایی برنامه است. نسخه‌ی prod همان نسخه‌ی دیباگ است با حذف اطلاعات دیباگ (نسخه‌ی production فشرده نشده). نسخه‌ی فشرده شده‌ی prod آن، فایل min نهایی را تشکیل می‌دهد.
- دریافت جی‌کوئری
- آخرین نگارش handlebars.js را از سایت رسمی آن دریافت کنید: http://handlebarsjs.com
- Ember Handlebars Loader: https://github.com/michaelrkn/ember-handlebars-loader
- Ember Data Local Storage Adapter: https://github.com/kurko/ember-localstorage-adapter

ترتیب تعریف و قرارگیری این فایل‌ها را پس از دریافت، در فایل index.html قرار گرفته در ریشه‌ی سایت، در کدهای ذیل مشاهده می‌کنید:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Ember Blog</title>

  <link href="Content/bootstrap.css" rel="stylesheet" />
  <link href="Content/bootstrap-theme.css" rel="stylesheet" />
  <link href="Content/styles.css" rel="stylesheet" />

  <script src="Scripts/Libs/jquery-2.1.1.min.js" type="text/javascript"></script>
  <script src="Scripts/Libs/bootstrap.min.js" type="text/javascript"></script>
  <script src="Scripts/Libs/handlebars-v2.0.0.js" type="text/javascript"></script>
  <script src="Scripts/Libs/ember.js" type="text/javascript"></script>
  <script src="Scripts/Libs/ember-handlebars-loader-0.0.1.js" type="text/javascript"></script>
  <script src="Scripts/Libs/ember-data.js" type="text/javascript"></script>
  <script src="Scripts/Libs/localstorage_adapter.js" type="text/javascript"></script>
</head>
<body>

</body>
</html>
```

اصلاح فایل ember-handlebars-loader-0.0.1.js

اگر به فایل ember-handlebars-loader-0.0.1.js مراجعه کنید، مسیر فایل‌های قالب handlebars قسمت‌های مختلف برنامه را از پوشه‌ی templates واقع در ریشه‌ی سایت می‌خواند. با توجه به تصویر ساختار پوشه‌ی پروژه‌ی جاری، پوشه‌ی template به داخل پوشه‌ی Scripts منتقل شده‌است و نیاز به یک چنین اصلاحی دارد:

```
url: "Scripts/Templates/" + name + ".hbs",
```

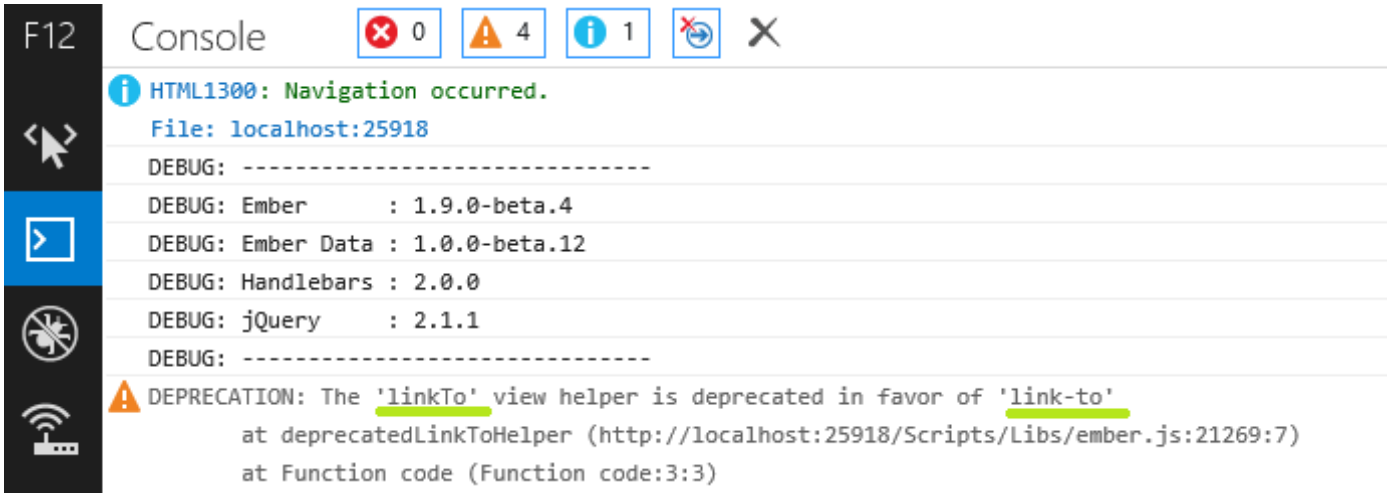
کار اسکریپت ember-handlebars-loader-0.0.1.js بارگذاری خودکار فایل‌های hbs یا handlebars از پوشه‌ی قالب‌های سفارشی

برنامه است. در این حالت اگر برنامه را اجرا کنید، خطای 404 را دریافت خواهید کرد. از این جهت که mime-type پسوند hbs در IIS تعریف نشده‌است. اضافه کردن آن نیز ساده‌است. به فایل web.config برنامه مراجعه کرده و تغییر ذیل را اعمال کنید:

```
<system.webServer>
  <staticContent>
        <mimeMap fileExtension=".hbs" mimeType="text/x-handlebars-template" />
  </staticContent>
 </system.webServer>
```

مزیت استفاده از نسخه‌ی دیباگ ember.js در حین توسعه‌ی برنامه

نسخه‌ی دیباگ ember.js علاوه بر به همراه داشتن خطاهای بسیار جامع و توضیح علل مشکلات، مواردی را که در آینده منسوخ خواهند شد نیز توضیح می‌دهد:



برای مثال در اینجا عنوان شده‌است که دیگر از linkTo استفاده نکنید و آن‌را به link-to تغییر دهید.
همچنین اگر از مرورگر کروم استفاده می‌کنید، افزونه‌ی Ember Inspector را نیز می‌توانید نصب کنید تا اطلاعات بیشتری را از جزئیات مسیریابی‌های تعریف شده و قالب‌های Ember.js بتوان مشاهده کرد. این افزونه به صورت یک برگه‌ی جدید در Developer tools آن ظاهر می‌شود.

ایجاد شیء Application

همانطور که در قسمت‌های پیشین نیز عنوان شد (^ و ^)، یک برنامه‌ی Ember.js با تعریف وهله‌ای از شیء Application آن آغاز می‌شود. برای این منظور به پوشه‌ی App مراجعه کرده و فایل جدید Scripts\App\blogger.js را اضافه کنید؛ با این محتوا:

```
Blogger = Ember.Application.create();
```

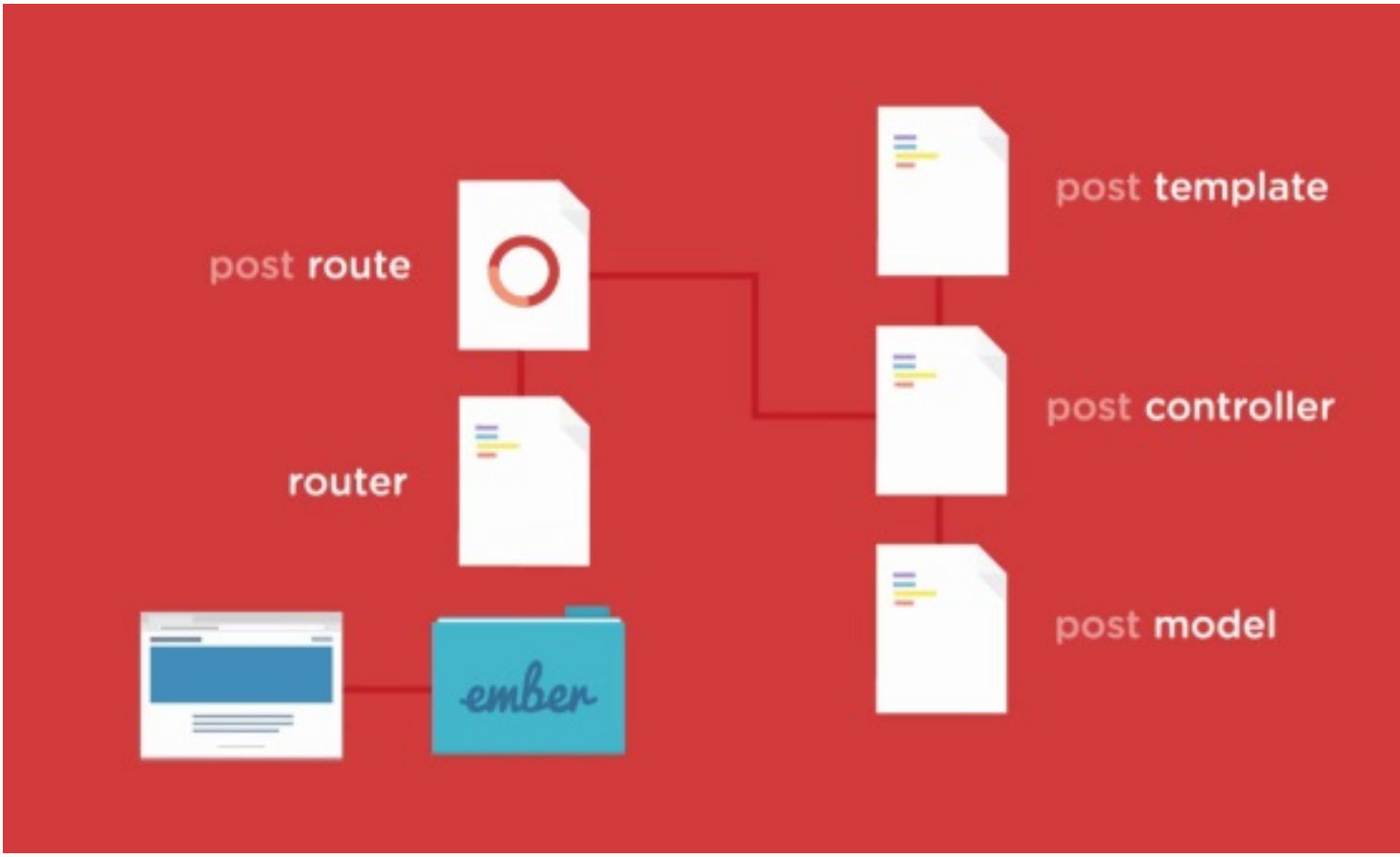
مدخل این فایل را نیز پس از تعاریف وابستگی‌های اصلی برنامه، به فایل index.html اضافه خواهیم کرد:

```
<script src="Scripts/App/blogger.js" type="text/javascript"></script>
```

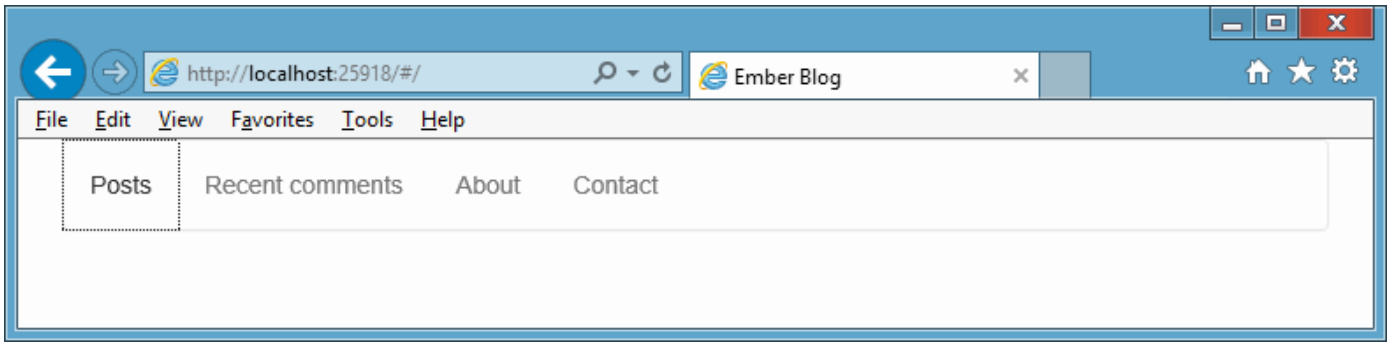
تا اینجا برپایی اولیه‌ی برنامه‌ی تک صفحه‌ای وب مبتنی بر Ember.js ما به پایان می‌رسد.

تعاریف مسیریابی و قالب‌ها

اکنون در ادامه قصد داریم لیستی از عناوین مطالب ارسالی را نمایش دهیم. در ابتدا این عناوین را از یک آرایه‌ی ثابت جاوا اسکریپتی دریافت خواهیم کرد و پس از آن از یک Web API کنترلر، جهت دریافت اطلاعات از سرور کمک خواهیم گرفت.



کار router در Ember.js، نگاشت آدرس درخواستی توسط کاربر، به یک route یا مسیریابی تعریف شده‌است. به این ترتیب مدل، کنترلر و قالب آن route به صورت خودکار بارگذاری و پردازش خواهند.
با مراجعه به ریشه‌ی سایت، فایل index.html بارگذاری می‌شود.



در اینجا تصویری از صفحه‌ی آغازین بلاگ را مشاهده می‌کنید. در آن صفحه‌ی posts همان ریشه‌ی سایت نیز می‌باشد. بنابراین نیاز است ابتدا مسیریابی آن‌را تعریف کرد. برای این منظور، فایل جدید Scripts\App\router.js را به پوشه‌ی App اضافه کنید؛ با این محتوا:

```
Blogger.Router.map(function () {
   this.resource('posts', { path: '/' });
});
```

علت تعریف قسمت path این است که ریشه‌ی سایت (/) نیز به مسیریابی posts ختم شود. در غیر اینصورت کاربر با مراجعه به ریشه‌ی سایت، یک صفحه‌ی خالی را مشاهده خواهد کرد؛ زیرا به صورت پیش فرض، آدرس قابل ترجمه‌ی یک صفحه، با آدرس و نام مسیریابی آن یکی است.

همچنین مدخل آن‌را نیز در فایل index.html تعریف نمائید:

```
<script src="Scripts/App/blogger.js" type="text/javascript"></script>
<script src="Scripts/App/router.js" type="text/javascript"></script>
```

در اینجا Blogger همان شیء Application برنامه است که پیشتر در فایل Scripts\App\blogger.js تعریف کردیم. سپس به کمک متد Blogger.Router.map، اولین مسیریابی برنامه را افزوده‌ایم.

افزودن مسیریابی و قالب posts

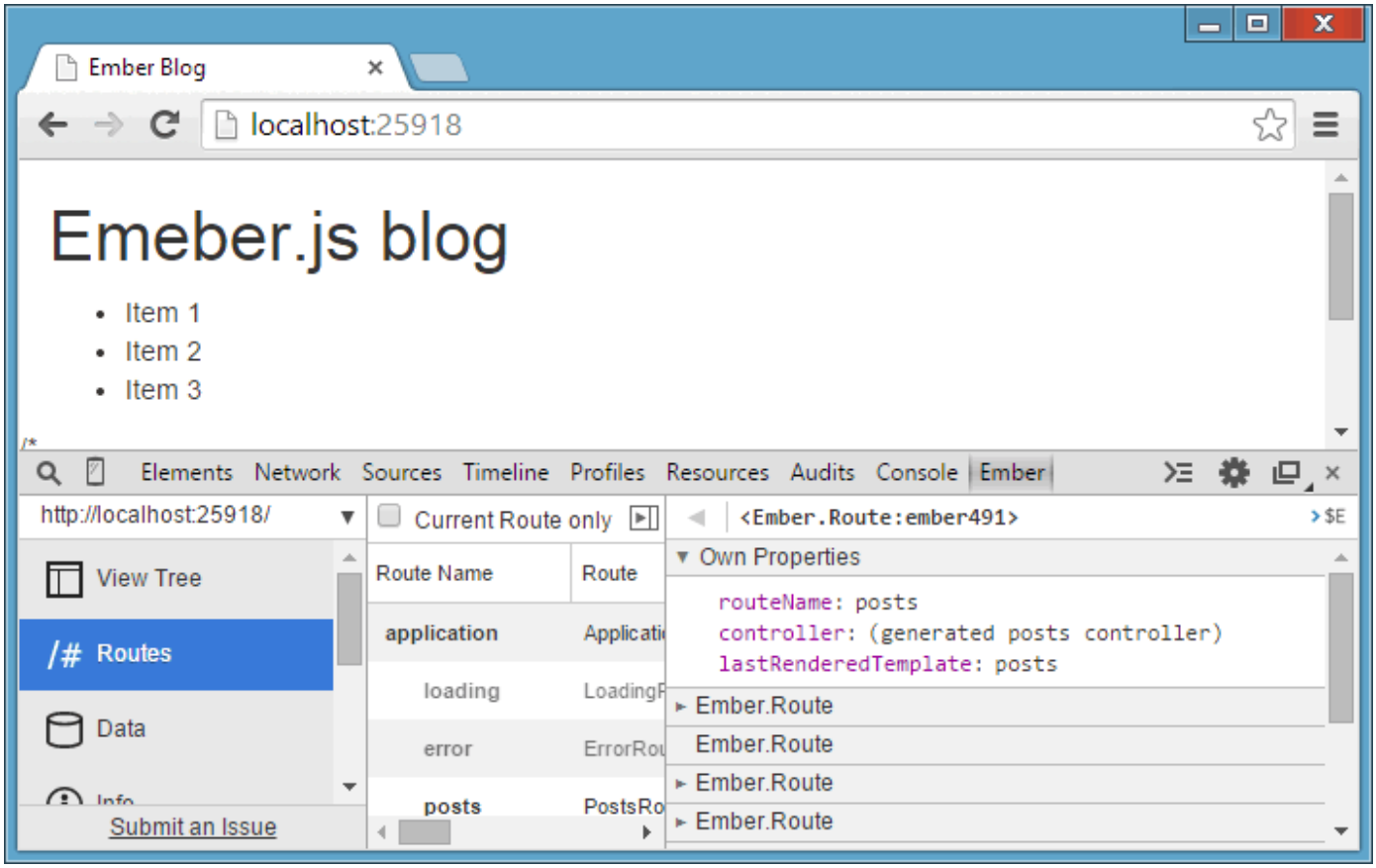
در ادامه فایل جدید Scripts\Templates\posts.hbs را اضافه کنید. به این ترتیب قالب خالی مطالب به پوشه‌ی templates اضافه می‌شود. محتوای این فایل را به نحو ذیل تنظیم کنید:

```
<div class="container">
  <h1>Emeber.js blog</h1>
  <ul>
   <li>Item 1</li>
   <li>Item 2</li>
   <li>Item 3</li>
  </ul>
</div>
```

در اینجا دیگر نیازی به ذکر تگ script از نوع text/x-handlebars نیست.
برای بارگذاری این قالب نیاز است آن‌را به template loader توضیح داده شده در ابتدای بحث، در فایل index.html اضافه کنیم:

```
<script>
 EmberHandlebarsLoader.loadTemplates([
 'posts'
 ]);
</script>
```

اکنون برنامه را اجرا کنید. به تصویر ذیل خواهید رسید که در آن افزونه‌ی Ember Inspector نیز نمایش داده شده‌است:



افزودن مسیریابی و قالب about

در ادامه قصد داریم صفحه‌ی about را اضافه کنیم. مجددا با افزودن مسیریابی آن به فایل Scripts\App\router.js شروع می‌کنیم:

```
Blogger.Router.map(function () {
  this.resource('posts', { path: '/' });
  this.resource('about');
});
```

سپس فایل قالب آن‌را در مسیر Scripts\Templates\about.hbs ایجاد خواهیم کرد؛ برای مثال با این محتوای فرضی:

```
<h1>About Ember Blog</h1>
<p>Bla bla bla!</p>
```

اکنون نام این فایل را به template loader فایل index.html اضافه می‌کنیم.

```
<script>
 EmberHandlebarsLoader.loadTemplates([
 'posts', 'about'
 ]);
</script>
```

اگر از قسمت قبل به خاطر داشته باشید، ساختار کلی قالب‌های ember.js یک چنین شکلی را دارد:

```
<script type="text/x-handlebars" data-template-name="about">

</script>
```

اسکریپت template loader، این تعاریف را به صورت خودکار اضافه می‌کند. مقدار data-template-name را نیز به نام فایل متناظر با آن تنظیم خواهد کرد و چون ember.js بر اساس ایده‌ی convention over configuration کار می‌کند، مسیریابی about با کنترلری به همین نام و قالبی هم نام پردازش خواهد شد. بنابراین نام فایل‌های قالب را باید بر اساس مسیریابی‌های متناظر با آن‌ها تعیین کرد.
برای آزمایش این مسیر و قالب جدید آن، آدرس http://localhost/#/about را بررسی کنید.

اضافه کردن منوی ثابت بالای سایت

روش اول این است که به ابتدای هر دو قالب about.hbs و posts.hbs، تعاریف منو را اضافه کنیم. مشکل این‌کار، تکرار کدها و پایین آمدن قابلیت نگهداری برنامه است. روش بهتر، افزودن کدهای مشترک بین صفحات، در قالب application برنامه است. نمونه‌ی آن‌را در مثال قسمت قبل مشاهده کرده‌اید. در اینجا چون قصد نداریم به صورت دستی قالب‌ها را به صفحه اضافه کنیم و همچنین برای ساده شدن نگهداری برنامه، قالب‌ها را در فایل‌های مجزایی قرار داده‌ایم، تنها کافی است فایل جدید Scripts\Templates\application.hbs را به پوشه‌ی قالب‌های برنامه اضافه کنیم؛ با این محتوا:

```
<div class='container'>
 <nav class='navbar navbar-default' role='navigation'>
  <ul class='nav navbar-nav'>
  <li>{{#link-to 'posts'}}Posts{{/link-to}}</li>
  <li>{{#link-to 'about'}}About{{/link-to}}</li>
  </ul>
 </nav>

  {{outlet}}
</div>
```

و سپس همانند قبل، نام فایل قالب اضافه شده را به template loader فایل index.html اضافه می‌کنیم:

```
<script>
 EmberHandlebarsLoader.loadTemplates([
 'posts', 'about', 'application'
 ]);
</script>
```

افزودن مسیریابی و قالب contact

برای افزودن صفحه‌ی تماس با مای سایت، ابتدا مسیریابی آن‌را در فایل Scripts\App\router.js تعریف می‌کنیم:

```
Blogger.Router.map(function () {
  this.resource('posts', { path: '/' });
  this.resource('about');
  this.resource('contact');
});
```

سپس قالب متناظر با آن‌را به نام Scripts\Templates\contact.hbs اضافه خواهیم کرد؛ فعلا با این محتوای اولیه:

```
<h1>Contact</h1>
<ul>
  <li>Phone: ...</li>
  <li>Email: ...</li>
</ul>
```

و بعد template loader فایل index.html را از وجود آن مطلع خواهیم کرد:

```
<script>
 EmberHandlebarsLoader.loadTemplates([
 'posts', 'about', 'application', 'contact' ]);
</script>
```

همچنین لینکی به مسیریابی آن‌را به فایل Scripts\Templates\application.hbs که منوی سایت در آن تعریف شده‌است، اضافه می‌کنیم:

```
<div class='container'>
 <nav class='navbar navbar-default' role='navigation'>
  <ul class='nav navbar-nav'>
  <li>{{#link-to 'posts'}}Posts{{/link-to}}</li>
  <li>{{#link-to 'about'}}About{{/link-to}}</li>
  <li>{{#link-to 'contact'}}Contact{{/link-to}}</li>
  </ul>
 </nav>

  {{outlet}}
</div>
```

تعریف مسیریابی تو در تو در صفحه‌ی contact

در حال حاضر صفحه‌ی Contact، ایمیل و شماره تماس را در همان بار اول نمایش می‌دهد. می‌خواهیم این دو را تبدیل به دو لینک جدید کنیم که با کلیک بر روی هر کدام، محتوای مرتبط، در قسمتی از همان صفحه بارگذاری شده و نمایش داده شود.
برای اینکار نیاز است مسیریابی را تو در تو تعریف کنیم:

```
Blogger.Router.map(function () {
  this.resource('posts', { path: '/' });
  this.resource('about');
  this.resource('contact', function () {
   this.resource('email');
   this.resource('phone');
  });
});
```

اگر مسیریابی‌های email و یا phone را به صورت مستقل مانند about و یا posts تعریف کنیم، با کلیک کاربر بر روی لینک متناظر با هر کدام، یک صفحه‌ی کاملا جدید نمایش داده می‌شود. اما در اینجا قصد داریم تنها قسمت کوچکی از همان صفحه‌ی contact را با محتوای ایمیل و یا شماره تماس جایگزین نمائیم. به همین جهت مسیریابی‌های متناظر را در اینجا به صورت تو در تو و ذیل مسیریابی contact تعریف کرده‌ایم.

پس از آن دو فایل قالب جدید Scripts\Templates\email.hbs را با محتوای:

```
<h2>Email</h2>
<p>
<span></span> Email name@site.com.
</p>
```

و فایل قالب Scripts\Templates\phone.hbs را با محتوای:

```
<h2>Phone</h2>
<p>
<span></span> Call 12345678.
</p>
```

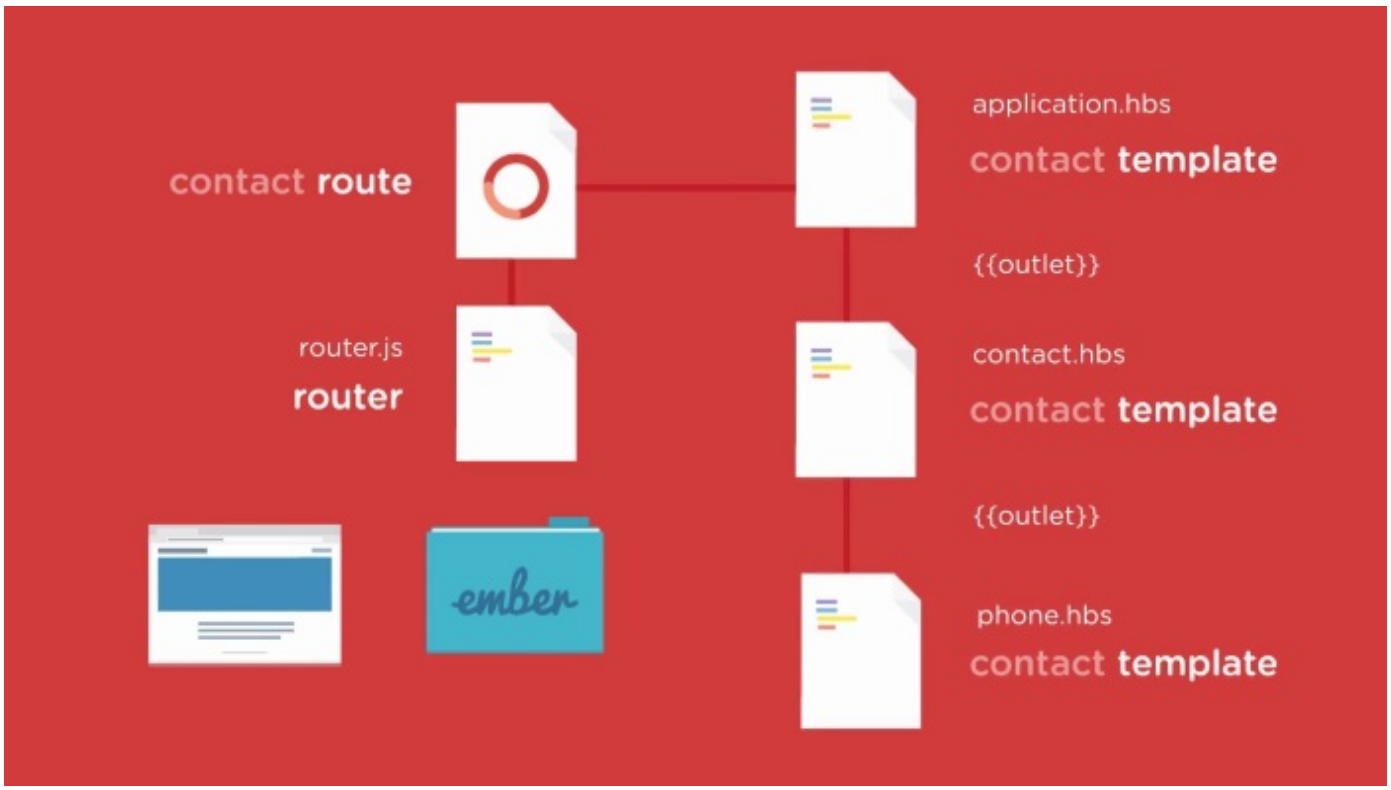
به پوشه‌ی قالب‌ها اضافه نمائید به همراه معرفی نام آن‌ها به template loader برنامه در صفحه‌ی index.html :

```
<script>
 EmberHandlebarsLoader.loadTemplates([
 'posts', 'about', 'application', 'contact', 'email', 'phone' ]);
</script>
```

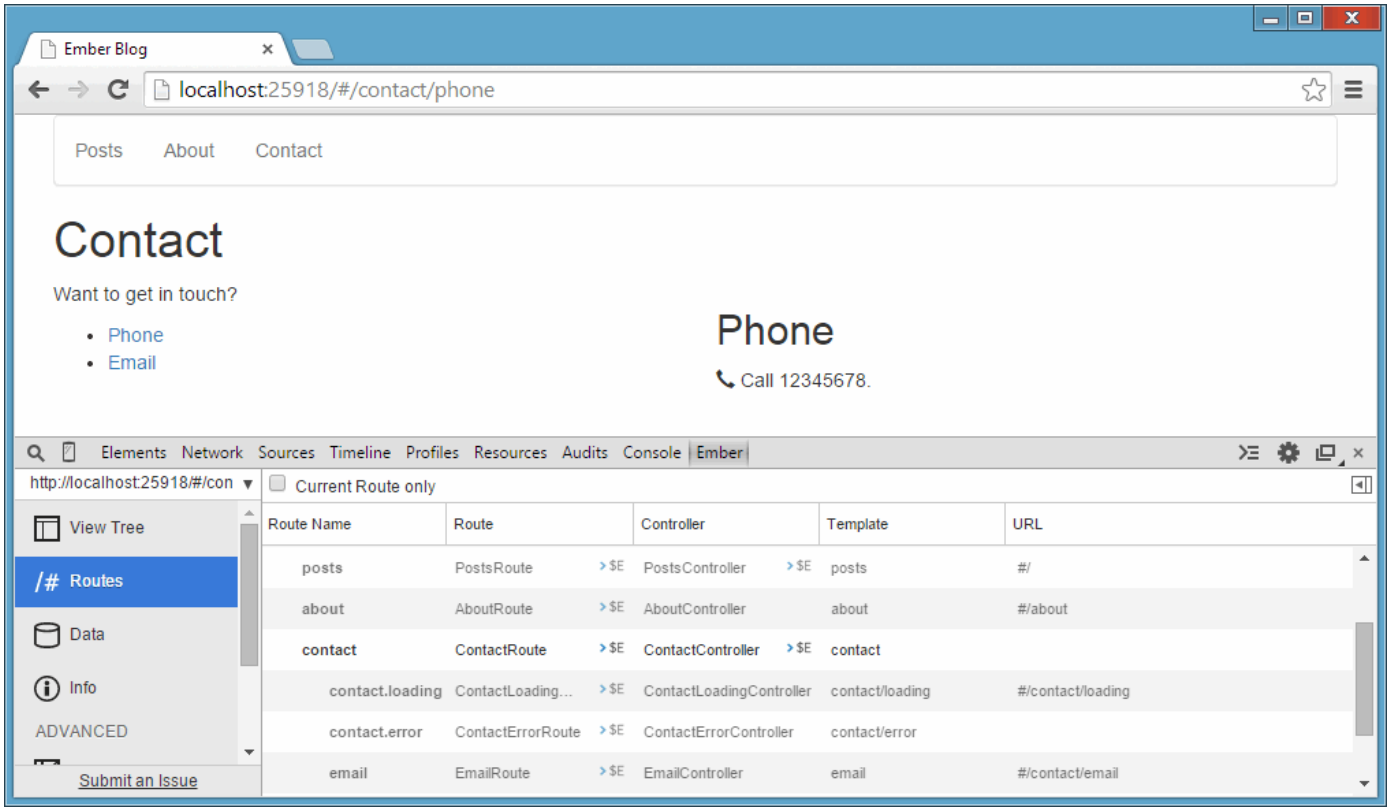
اکنون به قالب contact.hbs مجددا مراجعه کرده و تعاریف آن را به نحو ذیل تغییر دهید:

```
<h1>Contact</h1>
<div class="row">
  <div class="col-md-6">
   <p>
    Want to get in touch?
    <ul>
      <li>{{#link-to 'phone'}}Phone{{/link-to}}</li>
      <li>{{#link-to 'email'}}Email{{/link-to}}</li>
    </ul>
   </p>
  </div>
  <div class="col-md-6">
   {{outlet}}
  </div>
</div>
```

در اینجا دو لینک به مسیریابی‌های Phone و Email تعریف شده‌اند. همچنین {{outlet}} نیز قابل مشاهده است. با کلیک بر روی لینک Phone، مسیریابی آن فعال شده و سپس قالب متناظر با آن در قسمت {{outlet}} رندر می‌شود. در مورد لینک Email نیز به همین نحو رفتار خواهد شد.



در اینجا نحوه‌ی پردازش مسیریابی contact را ملاحظه می‌کنید. ابتدا درخواستی جهت مشاهده‌ی آدرس http://localhost/#/contact دریافت می‌شود. سپس router این درخواست را به مسیریابی همنامی منتقل می‌کند. این مسیریابی ابتدا قالب عمومی application را رندر کرده و سپس قالب اصلی و همنام مسیریابی جاری یا همان contact.hbs را رندر می‌کند. در این صفحه چون مسیریابی تو در تویی تعریف شده‌است، اگر درخواستی برای مشاهده‌ی http://localhost/#/contact/phone دریافت شود، محتوای آن‌را در {{outlet}} قالب contact.hbs جاری رندر می‌کند.



کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

EmberJS03_01.zip