عنوان: **افزودن یک صفحه‌ی جدید و دریافت و نمایش اطلاعات از سرور به کمک Ember.js**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۵ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, ASP.NET Web API, jQuery, SPA, EmberJS

در قسمت قبل با مقدمات برپایی یک برنامه‌ی تک صفحه‌ای وب مبتنی بر Ember.js آشنا شدیم. مثال انتهای بحث آن نیز یک لیست ساده را نمایش می‌دهد. در ادامه همین برنامه را جهت نمایش لیستی از اشیاء JSON دریافتی از سرور تغییر خواهیم داد. همچنین یک صفحه‌ی about را نیز به آن اضافه خواهیم کرد.

پیشنیازهای سمت سرور

- ابتدا یک پروژه‌ی خالی ASP.NET را ایجاد کنید. نوع آن مهم نیست که Web Forms باشد یا MVC.
- سپس قصد داریم مدل کاربران سیستم را توسط یک ASP.NET Web API Controller در اختیار Ember.js قرار دهیم. مباحث پایه‌ای Web API نیز در وب فرم‌ها و MVC یکی است.

مدل سمت سرور برنامه چنین شکلی را دارد:

```
namespace EmberJS02.Models
{
    public class User
    {
        public int Id { set; get; }
        public string UserName { set; get; }
        public string Email { set; get; }
    }
}
```

کنترلر Web API ایی که این اطلاعات را در ختیار کلاینت‌ها قرار می‌دهد، به نحو ذیل تعریف می‌شود:

```
using System.Collections.Generic;
using System.Web.Http;
using EmberJS02.Models;

namespace EmberJS02.Controllers
{
    public class UsersController : ApiController
    {
        // GET api/<controller>
        public IEnumerable<User> Get()
        {
            return UsersDataSource.UsersList;
        }
    }
}
```

در اینجا UsersDataSource.UsersList صرفا یک لیست جنریک ساده از کلاس User است و کدهای کامل آن‌را می‌توانید از فایل پیوست انتهای بحث دریافت کنید.

همچنین فرض بر این است که مسیریابی سمت سرور ذیل را نیز به فایل global.asax.cs، جهت فعال سازی دسترسی به متدهای کنترلر UsersController تعریف کرده‌اید:

```
using System;
using System.Web.Http;
using System.Web.Routing;

namespace EmberJS02
{
    public class Global : System.Web.HttpApplication
    {
        protected void Application_Start(object sender, EventArgs e)
        {
            RouteTable.Routes.MapHttpRoute(
               name: "DefaultApi",
```

```
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
                );
        }
    }
}
```

پیشنیازهای سمت کاربر

پیشنیازهای سمت کاربر این قسمت با قسمت « تهیه‌ی اولین برنامه‌ی Ember.js » دقیقا یکی است.
ابتدا فایل‌های مورد نیاز Ember.js به برنامه اضافه شده‌اند:

```
PM> Install-Package EmberJS
```

سپس یک فایل app.js با محتوای ذیل به پوشه‌ی Scripts اضافه شده‌است:

```
App = Ember.Application.create();
App.IndexRoute = Ember.Route.extend({
    setupController:function(controller) {
        controller.set('content', ['red', 'yellow', 'blue']);
    }
});
```

و در آخر یک فایل index.html با محتوای ذیل کار برپایی اولیه‌ی یک برنامه‌ی مبتنی بر Ember.js را انجام می‌دهد:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="Scripts/jquery-2.1.1.js" type="text/javascript"></script>
    <script src="Scripts/handlebars.js" type="text/javascript"></script>
    <script src="Scripts/ember.js" type="text/javascript"></script>
    <script src="Scripts/app.js" type="text/javascript"></script>
</head>
<body>
    <script type="text/x-handlebars" data-template-name="application">
        <h1>Header</h1>
        {{outlet}}
    </script>
    <script type="text/x-handlebars" data-template-name="index">
        Hello,
        <strong>Welcome to Ember.js</strong>!
        <ul>
            {{#each item in content}}
            <li>
                {{item}}
            </li>
            {{/each}}
        </ul>
    </script>
</body>
</html>
```

تا اینجا را در قسمت قبل مطالعه کرده بودید.
در ادامه قصد داریم به هدر صفحه، دو لینک Home و About را اضافه کنیم؛ به نحوی که لینک Home به مسیریابی index و لینک About به مسیریابی about که صفحه‌ی جدید «درباره‌ی برنامه» را نمایش می‌دهد، اشاره کنند.

تعریف صفحه‌ی جدید About

برنامه‌های Ember.js، برنامه‌های تک صفحه‌ای وب هستند و صفحات جدید در آن‌ها به صورت یک template جدید تعریف می‌شوند که نهایتا متناظر با یک مسیریابی مشخص خواهند بود.
به همین جهت ابتدا در فایل app.js مسیریابی about را اضافه خواهیم کرد:

```
App.Router.map(function() {
    this.resource('about');
});
```

به این ترتیب با فراخوانی آدرس /about در مرورگر توسط کاربر، منابع مرتبط با این آدرس و قالب مخصوص آن، توسط Ember.js پردازش خواهند شد.
بنابراین به صفحه‌ی index.html برنامه مراجعه کرده و صفحه‌ی about را توسط یک قالب جدید تعریف می‌کنیم:

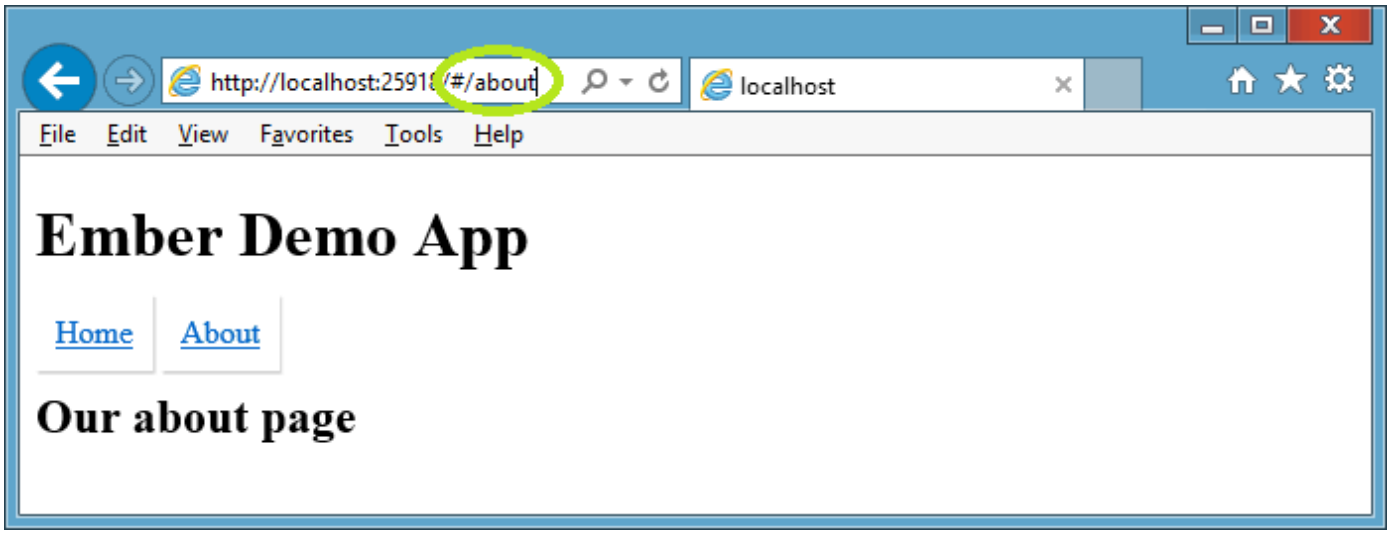
```
<script type="text/x-handlebars" data-template-name="about">
    <h2>Our about page</h2>
</script>
```

تنها نکته‌ی مهم در اینجا مقدار data-template-name است که سبب خواهد شد تا به مسیریابی about، به صورت خودکار متصل و مرتبط شود.

در این حالت اگر برنامه را در حالت معمولی اجرا کنید، خروجی خاصی را مشاهده نخواهید کرد. بنابراین نیاز است تا لینکی را جهت اشاره به این مسیر جدید به صفحه اضافه کنیم:

```
<script type="text/x-handlebars" data-template-name="application">
    <h1>Ember Demo App</h1>
    <ul class="nav">
        <li>{{#linkTo 'index'}}Home{{/linkTo}}</li>
        <li>{{#linkTo 'about'}}About{{/linkTo}}</li>
    </ul>
    {{outlet}}
</script>
```

اگر از قسمت قبل به خاطر داشته باشید، عنوان شد که قالب ویژه‌ی application به صورت خودکار با وهله سازی Ember.Application.create به صفحه اضافه می‌شود. اگر نیاز به سفارشی سازی آن وجود داشت، خصوصا جهت تعریف عناصری که باید در تمام صفحات حضور داشته باشند (مانند منوها)، می‌توان آن‌را به نحو فوق سفارشی سازی کرد.
در اینجا با استفاده از امکان یا directive ویژه‌ای به نام linkTo، لینک‌هایی به مسیریابی‌های index و about اضافه شده‌اند. به این ترتیب اگر کاربری برای مثال بر روی لینک About کلیک کند، کتابخانه‌ی Ember.js او را به صورت خودکار به مسیریابی about و سپس نمایش قالب مرتبط با آن (قالب about ایی که پیشتر تعریف کردیم) هدایت خواهد کرد؛ مانند تصویر ذیل:



همانطور که در آدرس صفحه نیز مشخص است، هرچند صفحه‌ی about نمایش داده شده‌است، اما هنوز نیز در همان صفحه‌ی اصلی برنامه قرار داریم. به علاوه در این قسمت جدید، همچنان منوی بالای صفحه نمایان است؛ از این جهت که تعاریف آن به قالب application اضافه شده‌اند.

دریافت و نمایش اطلاعات از سرور

اکنون که با نحوه‌ی تعریف یک صفحه‌ی جدید و برپایی سیم کشی‌های مرتبط با آن آشنا شدیم، می‌خواهیم صفحه‌ی دیگری را به نام Users به برنامه اضافه کنیم و در آن لیست کاربران ارائه شده توسط کنترلر Web API سمت سرور ابتدای بحث را نمایش دهیم.
بنابراین ابتدا مسیریابی جدید users را به صفحه اضافه می‌کنیم تا لیست کاربران، در آدرس users/ قابل دسترسی شود:

```
App.Router.map(function() {
    this.resource('about');
    this.resource('users');
});
```

سپس نیاز است مدلی را توسط فراخوانی Ember.Object.extend ایجاد کرده و به کمک متد reopenClass آن‌را توسعه دهیم:

```
App.UsersLink = Ember.Object.extend({});
App.UsersLink.reopenClass({
    findAll: function () {
        var users = [];
        $.getJSON('/api/users').then(function(response) {
            response.forEach(function(item) {
                users.pushObject(App.UsersLink.create(item));
            });
        });
        return users;
    }
});
```

در اینجا متد دلخواهی را به نام findAll اضافه کرده‌ایم که توسط متد getJSON جی‌کوئری، به مسیر /api/users سمت سرور متصل شده و لیست کاربران را از سرور به صورت JSON دریافت می‌کند. در اینجا خروجی دریافتی از سرور به کمک متد pushObject به آرایه کاربران اضافه خواهد شد. همچنین نحوه‌ی فراخوانی متد create مدل UsersLink را نیز در اینجا مشاهده می‌کنید (App.UsersLink.create).

پس از اینکه نحوه‌ی دریافت اطلاعات از سرور مشخص شد، باید اطلاعات این مدل را در اختیار مسیریابی Users قرار داد:

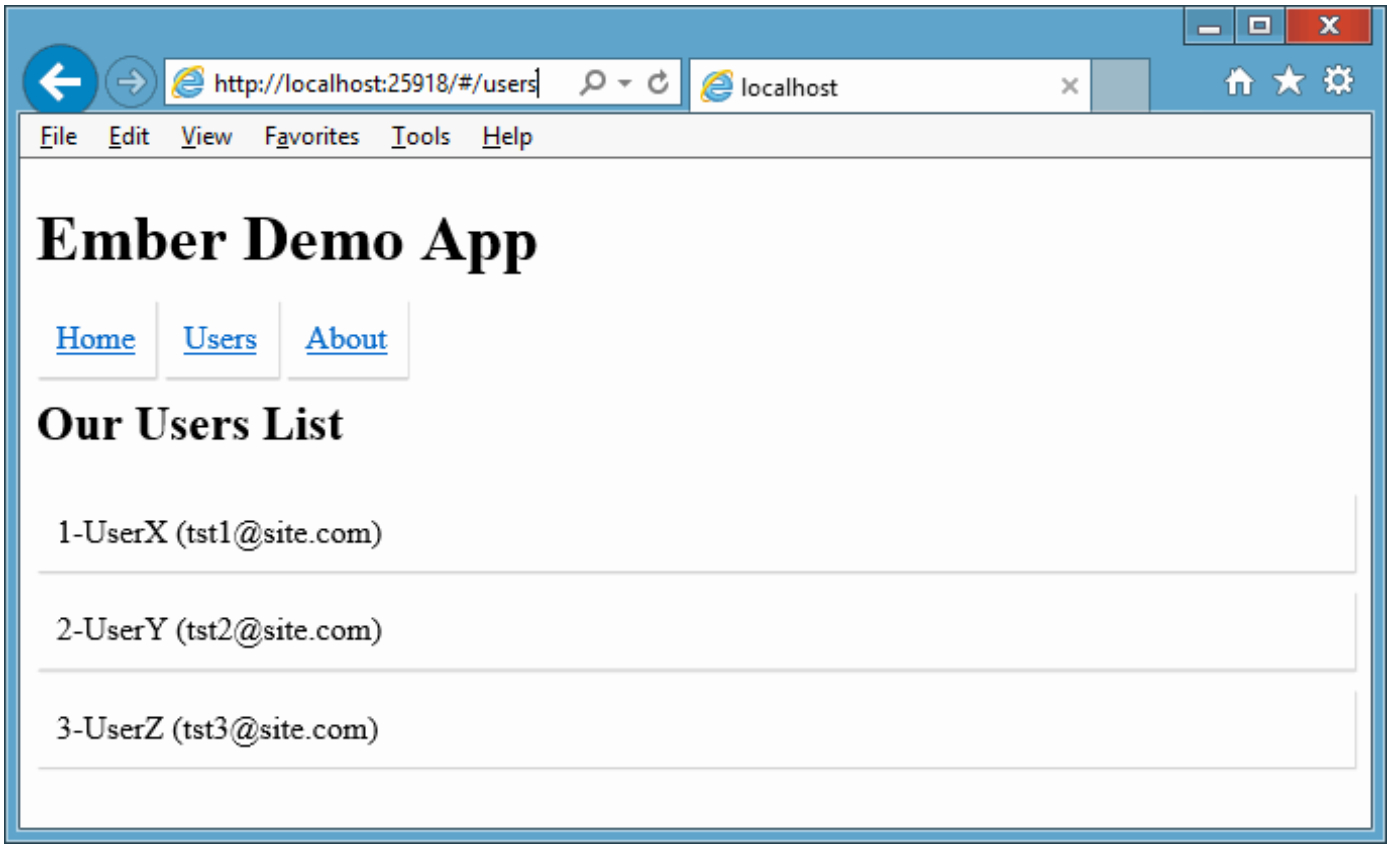
```
App.UsersRoute = Ember.Route.extend({
    model: function() {
        return App.UsersLink.findAll();
    }
});

App.UsersController = Ember.ObjectController.extend({
    customHeader : 'Our Users List'
});
```

به این ترتیب زمانیکه کاربر به مسیر users/ مراجعه می‌کند، سیستم مسیریابی می‌داند که اطلاعات مدل خود را باید از کجا تهیه نماید.
همچنین در کنترلری که تعریف شده، صرفا یک خاصیت سفارشی و دلخواه جدید، به نام customHeader برای نمایش در ابتدای صفحه تعریف و مقدار دهی گردیده‌است.
اکنون قالبی که قرار است اطلاعات مدل را نمایش دهد، چنین شکلی را خواهد داشت:

```
<script type="text/x-handlebars" data-template-name="users">
    <h2>{{customHeader}}</h2>
    <ul>
        {{#each item in model}}
        <li>
            {{item.Id}}-{{item.UserName}} ({{item.Email}})
```

```
        </li>
        {{/each}}
    </ul>
</script>
```

با تنظیم data-template-name به users سبب خواهیم شد تا این قالب اطلاعات خودش را از مسیریابی users دریافت کند. سپس یک حلقه نوشته‌ایم تا کلیه عناصر موجود در مدل را خوانده و در صفحه نمایش دهد. همچنین در عنوان قالب نیز از خاصیت سفارشی customHeader استفاده شده‌است:



کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

EmberJS02.zip