عنوان: قابلیت چند زبانه و Localization در AngularJs- بخش چهارم و نهایی: Best Practiceهای angular-translate
نویسنده: مهرداد کاهه
تاریخ: ۱۳۹۴/۰۷/۱۱ ۱۱:۲۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Localization, AngularJS

در بخش پیشین چند مورد از قابلیت‌های angular-translate را بررسی نمودیم. در این بخش به بررسی باقی موارد می‌پردازیم.

ex7_load_static_files

در این مثال خواهیم دید که چگونه یک فایل translate table در موقع فراخوانی به صورت On Demand بارگذاری خواهد شد. در قدم اول اسکریپت‌های زیر به صفحه افزوده می‌شوند.

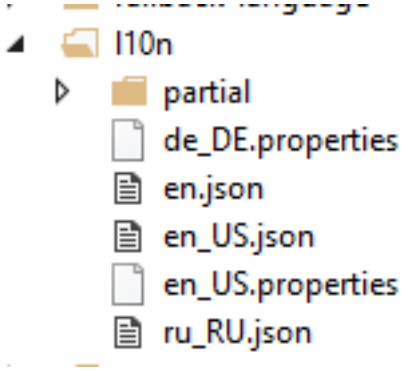
```
<script src="Scripts/angular.js"></script>
    <script src="Scripts/angular-cookies.js"></script>
    <script src="Scripts/angular-translate.js"></script>
    <script src="Scripts/angular-translate-storage-cookie.js"></script>

    <!-- for override loader methods in angular translate -->
    <script src="/src/service/loader-static-files.js"></script>
```

در ادامه درباره‌ی اسکریپت پنجم بیشتر توضیح خواهیم داد. بگذارید از آخر به اول شروع کنیم و ببینیم که نحوه‌ی فراخوانی و استفاده از امکان on demand بارگذاری شدن فایل‌های زبان به چه صورتی می‌باشد. در زیر، تکه کد اضافه شده به ex7 را مشاهده می‌کنید:

```
// Register a loader for the static files
            // So, the module will search missing translation tables under the specified urls.
            // Those urls are [prefix][langKey][suffix].
            $translateProvider.useStaticFilesLoader({
                prefix: '/l10n/',
                suffix: '.json'
            });
```

همانطور که در توضیحات آمده است، ماژول با دریافت prefix و suffix که در حقیقت همان فولدر و پسوند فایل‌های translate table هستند، هر زبانی را که مورد نیاز است و تا کنون بارگذاری نشده، بارگذاری می‌نماید. تصویر زیر محتویات فولدر l10n را نمایش می‌دهد.



حال ببینیم که این فرآیند در loader-static-files چگونه پیاده سازی شده است. در این فایل یک متد load نوشته شده است که فایل‌های static را طبق یک الگوی مشتمل بر prefix و suffix از سرور می‌خواند. لزومی ندارد که شما فایل‌ها را حتما به صورت JSON و با این پسوند ذخیره کنید. اما چیزی که قطعی است این است که فایل‌ها حتما باید به صورت key value ذخیره شده باشند.

تکه کد زیر اطلاعات فایل loader-static-files را نمایش می‌دهد.

```
angular.module('pascalprecht.translate')
.factory('$translateStaticFilesLoader', $translateStaticFilesLoader);
function $translateStaticFilesLoader($q, $http) {

  'use strict';

  return function (options) {

    if (!options || (!angular.isArray(options.files) && (!angular.isString(options.prefix) ||
!angular.isString(options.suffix)))) {
      throw new Error('Couldn\'t load static files, no files and prefix or suffix specified!');
    }

    if (!options.files) {
      options.files = [{
        prefix: options.prefix,
        suffix: options.suffix
      }];
    }

    var load = function (file) {
      if (!file || (!angular.isString(file.prefix) || !angular.isString(file.suffix))) {
        throw new Error('Couldn\'t load static file, no prefix or suffix specified!');
      }

      var deferred = $q.defer();

      $http(angular.extend({
        url: [
          file.prefix,
          options.key,
          file.suffix
        ].join(''),
        method: 'GET',
        params: ''
      }, options.$http)).success(function (data) {
        deferred.resolve(data);
      }).error(function () {
        deferred.reject(options.key);
      });

      return deferred.promise;
    };

    var deferred = $q.defer(),
        promises = [],
        length = options.files.length;

    for (var i = 0; i < length; i++) {
      promises.push(load({
        prefix: options.files[i].prefix,
        key: options.key,
        suffix: options.files[i].suffix
      }));
    }

    $q.all(promises).then(function (data) {
      var length = data.length,
          mergedData = {};

      for (var i = 0; i < length; i++) {
        for (var key in data[i]) {
          mergedData[key] = data[i][key];
        }
      }

      deferred.resolve(mergedData);
    }, function (data) {
      deferred.reject(data);
    });

    return deferred.promise;
  };
}

$translateStaticFilesLoader.displayName = '$translateStaticFilesLoader';
```

همانطور که ملاحظه می‌کنید، کد فوق یک سرویس با نام $translateStaticFilesLoader را تعریف نموده است. در صورتیکه ما

در کنترلر فایل ex7، اصلا نامی از آن نبردیم و تنها از $translateProvider.useStaticFilesLoader استفاده نمودیم! جواب در نحوه‌ی نگارش کد angular-translate نهفته است. در خط 866 فایل angular-translate تکه کد زیر مربوط به تعریف translateStaticFileLoader می‌باشد. همانطور که ملاحظه می‌کنید سرویس translateStaticFilesLoader درون فضای نام سرویس translateTable قرار گرفته است. بنابراین ما تنها با تعریف سرویس translateStaticFilesLoader، در حقیقت آن را override نموده‌ایم. در کد نمونه‌ای که در بخش‌های قبلی قرار داده‌ام یک فایل translate.js نیز قرار دارد که در فولدر src/services قرار گرفته است. این فایل نیز برخی از امکانات و سرویس‌های built-in درون angular-translate را سفارشی نموده است.

```
/**
   * @ngdoc function
   * @name pascalprecht.translate.$translateProvider#useStaticFilesLoader
   * @methodOf pascalprecht.translate.$translateProvider
   *
   * @description
   * Tells angular-translate to use `$translateStaticFilesLoader` extension service as loader.
   *
   * @param {Object=} options Optional configuration object
   */
  this.useStaticFilesLoader = function (options) {
    return this.useLoader('$translateStaticFilesLoader', options);
  };
```

در این 4 مجموعه سعی کردم تمامی آنچه را که برای ایجاد قابلیت چند زبانه و localization نیاز است و حیاتی بود، تشریح کنم. بنابراین تا کنون دانش خوبی درباره‌ی این کتابخانه کسب نموده‌اید. باقی تمرین‌ها را می‌توانید بر حسب نیاز با استفاده از مستندات موجود در angular-translate مطالعه و استفاده نمایید.