Localització

Angular-translate

Angular-translate esta provist de nombroses eines i extensions molt utils, i li ofereix una enorme flexibilitat a la hora d'aplicar customitzacions. Aquestes son algunes de les interesants funcionalitats de angular-translate:

Funcionalitats

- Components (filters/directives) per traduir el seu contingut
- Carga asincrónica de les dades i18n
- Suport de Pluralització, utilitzant **MessageFormat.js**
- Expandible a través de interfaços fácils d'utilitzar

Instal.lació

Asegúris de que estigui inclós en el document HTML.

```
<script src="path/to/angular-translate.js"></script>
```

Injecti el módul com a dependencia en la seva aplicació:

```
var app = angular.module('myApp', ['pascalprecht.translate']);
```

Ensénya-li a la aplicació a traduir:

```
app.config(['$translateProvider', function ($translateProvider) {
  $translateProvider.translations('en', {
    'TITLE': 'Hello',
    'FOO': 'This is a paragraph'
  });

  $translateProvider.translations('de', {
    'TITLE': 'Hola',
    'FOO': 'Esto es un párrafo'
  });

  $translateProvider.preferredLanguage('en');
}]);
```

Tradueixi la aplicació:

```
<h1>{{ 'TITLE' | translate }}</h1>
<p>{{ 'FOO' | translate }}</p>
```

Aquest es simplemente un exemple rápid de cóm treballa angular-translate. En poques paraules, tot consisteix en "*proveir dades i18n, comunicárseles a angular-translate, i fer-li saber cóm utilizar els seus components per interpretar aquestes traduccions (contra certs valors).*". Fácil ¿no?

Suport multilingüe

En Substitució de variables vam aprendre com passar-lis valors dinàmics als diferents components, per fer ús d'ells en les traduccions. Atès que hem cobert tota la funcionalitat bàsica que proveeixen aquests serveis (usant el servei d'interpolació per defecte), ara estem preparats per al següent nivell: Suport Multillenguatge

Per descomptat, està molt bé saber com utilitzar els components que proveeix angular-translate, però les coses es posen realment interessants quan es tracta d'ensenyar-li a la aplicació més d'un llenguatge (la qual cosa és la raó de ser d'aquest mòdul, en realitat). Així que comencem aprenent com afegir diverses taules de traducció a la vegada!

Enseñar-li diversos llenguatges a l'aplicació

El començament varem aprendre com agregar-li una taula de traducció a l'applicació, usant el mètode translations() de $translateProvider. El mateix mètode pot ser usat per afegir diverses taules de traducció alhora. Les taules donades només han d'especificar una clau de llenguatge, perquè angular-translate sigui capaç de reconèixer quina taula correspon a quin llenguatge.

Afegir una taula de traducció amb el seu corresponent llenguatge és molt simple. En lloc de passar-li només la taula com a argument a $translations(), li podem passar la clau de llenguatge com a primer argument, i la taula com a segon. Així que afegir una taula amb clau es faria així:

```
// registra una clave de traducción para 'en' (inglés)
$translateProvider.translations('en', {
  SALUDO: 'Hello world!'
});
```

Ara, per afegir una segona taula per a un altre llenguatge, diguem, castellà, simplement feu el mateix amb una clau de llenguatge diferent:

```
// registra una clave de traducción para 'es' (español)
$translateProvider.translations('es', {
  SALUDO: '¡Hola, mundo!'
```

```
});
```

¿No és realment simple? La nostre aplicació ara sap dos idiomes diferents. Es poden afegir tants idiomes com sigui necessari, no hi ha límit. En tot cas, ara tenim 2 llenguatges disponibles; ¿Quin va a utilitzar la nostra aplicació? angular-translate no prefereix cap llenguatge fins que nosaltres li ho indiquem.

Decidir quin llenguatge emprar

Com ara nosaltres hem registrat més d'una taula de traducció, `angular-translate` te que saber quina ha d'utilitzar. Aquí es a on entra en joc un nou metode de `$translateProvider`: `preferredLanguage()` que li diu a angular-translate quin dels llenguatges registrats tindria que ser utilitzat per defecte. Esta esperant un argument amb el valor d'una clau de llenguatge, la clau apunta a la taula de traducció determinada. Axí que, per a dir-li a l'aplicació que utilitzi alemán en lloc del anglés, amplii el codi d'aquesta manera:

```
// le dice a angular-translate que use español
$translateProvider.preferredLanguage('es');
```

Nota: També es pot usar $ translateProvider.use () per això, atès que també configura el llenguatge. No obstant això, sembla ser que aquesta és una mala pràctica, quan s'estan fent servir carregadors asincrònics en combinació amb un emmagatzematge (storage). En alguns casos pot passar que angular-translate hagi de fer 2 trucades asíncrones. Entrarem en detall sobre això més endavant. Per evitar això, hem introduït preferredLanguage (). Amb $translateProvider (), s'hauria d'utilitzar sempre preferredLanguage() en lloc de use().

Determinar el llenguatge preferit automáticament

A partir de la versió `2.0` també hi ha un métode `determinePreferredLanguage()` en `$translateProvider`. Aquest metode tracta de determinar per sí mateix cuál sería el llenguatge preferit; busca valors de propietats de l'objecte `window.navigator`, en el ordre següent:

- `navigator.languages[0]`
- `navigator.language`
- `navigator.browserLanguage`
- `navigator.systemLanguage`
- `navigator.userLanguage`

Llavors, en lloc de invocar `$translateProvider.preferredLanguage(langKey)`, es tindria que fer algo com això:

```
// que intenti trobar el llenguatge preferit per si mateix
$translateProvider.determinePreferredLanguage();
```

¡Ens hem de fer responsables al emprar aquest metode! S'ha de tenir en compte que cada navegador pot retornar propietats diferents per aquest metode.

Un altre configuración molt útil, disponible desde la versió 2.7 es: `$translateProvider.uniformLanguageTag()`. Amb aixo es pot decidir en quins "tags" (etiquetes) de llenguatge es tindrien que transformar, aquélles que han sigut resoltes.

```
1.      $translateProvider
2.        .uniformLanguageTag('bcp47') // habilitar BCP-47, lo cual te que fer-se abans de
    determinePreferredLanguage!
3.        .determinePreferredLanguage();
```

Si aixo no s'ajusta a les nostres necessitats, també es pot pasar una funció personalitzada que determini la clau de llenguatge:

```
1.      $translateProvider.determinePreferredLanguage(function () {
2.        var preferredLangKey = '';
3.        // aquí aniria el codi personalitzat
4.        return preferredLangKey;
5.      });
```

Cambiar el llenguatge en temps d'execució

Para cambiar el llenguatge en temps d'execució, el servei `$translate` te el métode `use()`, el cual o be retorna el llenguatge actual, o, si se li pasa un argument, le diu a angular-translate quin llenguatge emprar. `translate.use()` també invoca internament els carregadors asincrónics, si es trata d'utilitzar un llenguatge cuya taula de traducció encara no hagi sigut carregada. Podeu trobar mes información sobre aixó en Carrega asincrónica.

Un bon us de `$translate.use()` sería en un controlador que controli els canvis de llenguatge. Nosaltres simplement tindríem que implementar una funció en el "scope", i després utilizar-la per a indicar-li a angular-translate que cambii el llenguatge.

```
1.      angular.module('myApp').controller('Ctrl', ['$translate', '$scope', function ($translate,
    $scope) {
2.
3.        $scope.changeLanguage = function (langKey) {
4.          $translate.use(langKey);
5.        };
6.
7.      }]);
```

Per anar tenint una idea de cóm aixó funcionaría en l'aplicació d'exemple, actualitzem l'aplicació de manera acorde. Primer, agreegarem una altre taula de traducció per a l'alemán, i afegirem 2 noves claus de traducció per a els botons que afegirem més endevant:

```
var translationsEN = {
  HEADLINE: 'What an awesome module!',
  PARAGRAPH: 'Srsly!',
  PASSED_AS_TEXT: 'Hey there! I\'m passed as text value!',
  PASSED_AS_ATTRIBUTE: 'I\'m passed as attribute value, cool ha?',
  PASSED_AS_INTERPOLATION: 'Beginners! I\'m interpolated!',
  VARIABLE_REPLACEMENT: 'Hi {{name}}',
  BUTTON_LANG_ES: 'Spanish',
  BUTTON_LANG_EN: 'English'
};

var translationsES= {
  HEADLINE: '¡Qué módulo asombroso!',
  PARAGRAPH: '¡En serio!',
  PASSED_AS_TEXT: 'Hola, me están pasando como un texto',
  PASSED_AS_ATTRIBUTE: 'Me están pasando como atributo, ¡qué bueno!',
  PASSED_AS_INTERPOLATION: '¡Aprendices! A mí me están interpolando ...'
  VARIABLE_REPLACEMENT: 'Hola {{nombre}}',
  BUTTON_LANG_ES: 'Español',
  BUTTON_LANG_EN: 'Inglés'
};
```

Després, actualitzem el registre de la taula de traducció anglesa, que te la clau de traduccío corresponent, i li indiquem a angular-translate que utilitzi l'anglés com a llenguatge per defecte.

```
$translateProvider.translations('en', translationsEN);
$translateProvider.translations('es', translationsES);
$translateProvider.preferredLanguage('en');
```

¡Qué be esta quedant! Ara necesitarem controls que cambiin el llenguatge en tiempo d'execució. Actualitzem el nuestro HTML i afegirem un botó para a cada llenguatge. També configurarem una directiva `ng-click` en cada botó per que cridi una funció que canvii el llenguatge:

```
<button ng-click="changeLanguage('es')" translate="BUTTON_LANG_ES"></button>
<button ng-click="changeLanguage('en')" translate="BUTTON_LANG_EN"></button>
```

I finalment (encara que no en últim lloc), tenim que implementar l'anomenada funció en el scope del nostre controlador:

```
app.controller('Ctrl', ['$translate', '$scope', function ($translate, $scope) {

  $scope.changeLanguage = function (langKey) {
    $translate.use(langKey);
  };
```

```
6.      }]);
```

Et voilà! ¡Ara tenim una aplicación amb suport multillenguatge!

Veure Ejercici1.html i Ejercici1.js.

Afegir nou llenguatge

Ara que hem llegit que podem configurar un llenguatge preferit, i a l'hora registrar múltiples llenguatges, ens preguntarem si hi ha alguna forma de ensenyar-li a l'aplicació un llenguatge de recolzament. La resposta ja us l'avanço, es que si.

Registrar un llenguatge de recolzament

Ensenyar-li a l'aplicació un llenguatge de recolzament es fácil: simplement s'ha de invocar un metode en `$translateProvider`. Sí, tan fácil com sona. Diguem que tenim una aplicación i hem registrat una taula de traducció per a el idioma castella.

```
1.      $translateProvider
2.        .translations('es', { /* ... */ });
```

Ara, diguem que existeixen algunes Claus de traducció que están disponibles en anglés, pero no en la taula espanyola. angular-translate normalment retornaría només la clau si no pot trobar una traducció. Pero si nosaltres registrem un llenguatge de recolzament que contingui la clau de traducció en la seva taula, angular-translate retornara aquesta traducció en lloc de l'altre.

Axí que, registrarem anglés com a llenguatge de recolzament. Primer tenim que registrar el llenguatge mateix, per suposat.

```
1.      $translateProvider
2.        .translations('es', { /* ... */ })
3.        .translations('en', { /* ... */ });
```

Ara li indicarem a angular-translate que utilitzi l'anglés com a llenguatge de recolzament:

```
1.      $translateProvider
2.        .translations('es', { /* ... */ })
3.        .translations('en', { /* ... */ })
4.        .fallbackLanguage('en');
```

Y llestos. Si no existeix la determinada clau en la taula castellana, angular-translate buscará la mateixa clau en la taula anglesa. Fácil, ¿no?

Registrar un conjunt de recolzament

¿I si desitjo tenir tot un conjunt de llenguatges de recolzament? ¡angular-translate pot gestionar aixo també! Tot el que s'ha de fer, es registrar els llenguatges de recolzament com un array:

```
$translateProvider
  .translations('es', { /* ... */ })
  .translations('en', { /* ... */ })
  .translations('fr', { /* ... */ })
  .fallbackLanguage(['en', 'fr']);
```

Llavors, si angular-translate no pot trobar la corresponent clau de traducció en la taula d'anglés, que es el primer llenguatge de recolzament, pasará a la resta dels llenguatges de recolzament en ordre de prioritat. Es poden utilizar tants llenguatges de recolzament com es desitgi.

Modificar els llenguatges de recolzament en temps d'execució

Ah, sí, ¡inclos aixó es pot fer! A vegades nosaltres en trobarem en la necessitat de modificar el llenguatge de recolzament en temps d'execució, o inclós la pila completa de llenguatges de recolzament. Per exemple, imagineu que nosaltres vam configurar el alemán com llenguatge preferit, i que volem asegurarnos que l'anglés es el llenguatge de recolzament. Cap problema, aixo actualment ja ho podem fer. Pero ara imaginau que en temps d'execució passem al francés como a llenguatge de l'aplicació, i necesitarem que l'anglés (el qual era el llenguatge preferit) passi a ser el llenguatge de recolzament.

Aixo es pot fer en temps d'execució anomenant el metode `fallbackLanguage()` del servei `$translate`. Es veuria aixi:

```
$scope.changeLanguage = function (langKey) {
  if (langKey === 'en') {
    $translate.fallbackLanguage('fr');
  } else if (langKey == 'de') {
    $translate.fallbackLanguage('en');
  }
  $translate.use(langKey);
};
```

Si nosaltres realitzem camviis en el conjunt de llenguatges de recolzament en temps d'execució, angular-translate es guiará per l'ordre en que les claus del llenguatge hagin sigut especificades per saber com reiterar. Per exemple, si nosaltres ho vam configurar com a `en, fr, es` com a llenguatges de recolzament, i modifiquem el llenguatge de recolzament a `fr`, en aquest cas el llenguatge de recolzament de `fr` será `es`.

Cambiar el conjunt sencer de llenguatges de recolzament en temps d'execució

Igual que en el cas anterior, tenim que fer aixo:

```
1.      $scope.changeLanguage = function (langKey) {
2.        $translate.fallbackLanguage(['es', 'en', 'fr']);
3.        $translate.use(langKey);
4.      };
```

Quan es modifca el conjunt sencer de llenguatges de recolzament, s'esta modificant el ordre en el qual angular-translate reitera sobre ella. El proces mateix de reiteració no es modifica, només el ordre sobre el que es reitera

Enmagatzematge

Aprenem molt sobre angular-translate en els darrers capítols. Sabem com podem dinamitzar les nostres traduccions. També vam aprendre a afegir més d'un idioma. Però encara hi ha una cosa que falla en usabilitat, alhora d'obrir la nostra aplicació web. Cada vegada que llancem l'aplicació, hem de fer clic al botó "alemany" una vegada i una altra i de nou (per descomptat només si sou usuari d'alemany).

Així que el problema és que la nostra aplicació no recorda el idioma que heu triat l'última vegada que l'heu obert. En aquest capítol indicarem com podeu ensenyar a la vostra aplicació a enrecordar-se dels idiomes seleccionats pels usuaris.

Permet que la teva aplicació recordi l'idioma

Per permetre que la vostra aplicació recordi els idiomes que seleccionin els usuaris, angular-translate inclou un suport per als emmagatzemaments. Qualsevol emmagatzematge que utilitzeu, angular-translate guardarà una clau d'idioma amb un identificador específic, de manera que la podeu sol·licitar la propera vegada que l'usuari llanci l'aplicació.

Angular-translate ha incorporat suport per a dos magatzems. localStoratge i cookieStorage. A on localStorage es torna a cookieStorage si no és compatible amb el navegador que l'usuari utilitza actualment.

Per utilitzar un d'aquests magatzems, heu d'instal·lar el paquet d'extensió corresponent.

Recordeu també incloure el fitxer angular-cookies.min.js en el vostre HTML i afegir 'ngCookies' com a dependència..

```
var module = angular.module('AppService', ['pascalprecht.translate','ngCookies']);
```

Utilitzant cookieStorage

Si voleu utilitzar la cookieStorage per emmagatzemar l'idioma a través de les sol·licituds creuades d'http, simplement instal·leu l'extensió d'emmagatzematge de cookies mitjantçant bower:

```
$ bower install angular-translate-storage-cookie
```

Després, assegureu-vos d'haver-lo integrat al vostre document HTML. Un cop incrustat, podeu utilitzar $translateProvider en el metode useCookieStorage() i angular-translate es fa càrrec de la resta.

```
1.    $translateProvider.useCookieStorage();
```

És tan fàcil? angular-translate ara emmagatzemarà la clau d'idioma inicial en aquest emmagatzematge i l'actualitzarà d'acord amb això una vegada que un usuari canvii l'idioma.

Usant localStorage

En cas que no vulgueu utilitzar CookieStorage per diversos motius, podeu utilitzar LocalStorage per fer el mateix. El flux és bàsicament el mateix. Instal·leu el paquet d'extensió corresponent i crideu a $translateProvider que utilitzara el localStorage mitjançant useLocalStorage(). I una altra vegada, el angular-translate s'encarrega de la resta.
Tingueu en compte que localStorage es reemplaçarà a cookieStorage si el navegador no admet localStorage. Per tant, heu de proporcionar l'extensió CookieStorage també. Podeu instal·lar el paquet d'extensió aixi:

```
$ bower install angular-translate-storage-local
```

Ara informarem a `$translateProvider` que pretenem utilizar aquest:

```
1.    $translateProvider.useLocalStorage();
```

Això és tot. Ara, la nostre aplicació utilitza un espai local per recordar l'idioma de l'usuari. Actualitzem la nostra aplicació per utilitzar localStorage també!

Veure ejercici2.html i ejercici2.js.