AngularJS Internationalization

Topics : AngularJS
Written on January 09, 2024

In AngularJS, internationalization (i18n) refers to the process of adapting an application to different languages and regions, making it accessible to a global audience. AngularJS provides tools and best practices for implementing internationalization in your applications. Here are the key aspects of internationalization in AngularJS:

AngularJS i18n Module:

AngularJS provides an `ngLocale` module that includes tools for localization and internationalization. This module is part of the `angular-i18n.js` file, which you can include in your project to enable i18n support.

Here is an example of including the `angular-i18n.js` file for English (`en-US`) localization:

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular-locale_en-us.js"></script>
```

Replace `en-us` with the appropriate locale code for your target language.

Using AngularJS Filters for Localization:

AngularJS filters can be used to format data according to the user's locale. For example, the `date` filter can be used to format dates:

```
<p>{{ currentDate | date:'medium' }}</p>
```

Using `$locale` Service:

The `$locale` service provides information about the current locale and can be used to customize the behavior of your application based on the user's language.

```
angular.module('myApp').controller('MyController', function($scope, $locale) {
$scope.currentLocale = $locale.id;
});
```

Externalizing Texts and Translations:

To support multiple languages, it's a good practice to externalize texts and translations. This can be done by creating separate JSON files for each language, where each file contains translations for the application's texts.

Example of a JSON file for English (`en-US`):

```
// en-us.json
{
"greeting": "Hello, World!",
"welcome": "Welcome to our application."
}
```

Using AngularJS `$translate` Service:

While AngularJS itself doesn't provide a built-in translation service, you can use third-party libraries like `angular-translate` or `angular-gettext` for a more comprehensive i18n solution.

Here's a basic example using `angular-translate`:

1. Include the library in your project:

   ```
   <script
   src="https://cdnjs.cloudflare.com/ajax/libs/angular-translate/2.19.0/ang
   ular-translate.min.js"></script>
   ```

2. Configure the translation module in your AngularJS app:

   ```
   angular.module('myApp', ['pascalprecht.translate'])
   .config(function ($translateProvider) {
   $translateProvider.translations('en-US', {
   'greeting': 'Hello, World!',
   'welcome': 'Welcome to our application.'
   });
   $translateProvider.preferredLanguage('en-US');
   });
   ```

3. Use the `$translate` service in your controller or views:

   ```
   <p>{{ 'greeting' | translate }}</p>
   ```

Pluralization and Gender:

For more advanced scenarios like pluralization and gender-specific translations, libraries such as `angular-gettext` provide additional features and tools for handling these cases.

Caveats and Best Practices:

- Always externalize and store your texts and translations in a separate file or a translation service.
- Choose a comprehensive i18n library based on your project requirements.
- Ensure proper handling of dynamic content, such as dates and numbers, in different locales.
- Test your application with different locales to ensure the accuracy of translations and proper

formatting.